Universidad Zaragoza
1542

Proyecto Fin de Carrera
Ingeniería Industrial

Sistema de Realidad Aumentada en Entornos Reales Adversos

Autor

Miguel Ángel Montañés Laborda

Director
Mario F. Rodríguez Martínez
Ponente
José Elías Herrero Jaraba

Escuela de Ingeniería y Arquitectura de Zaragoza

2014

Sistema de Realidad Aumentada en Entornos Reales Adversos

RESUMEN

En este trabajo proponemos una aplicación para plataformas móviles capaz de realizar la proyección de una imagen virtual sobre un entorno real de forma consistente a lo largo de un periodo de tiempo más o menos prolongado. Se realiza una detección del entorno para seleccionar la superficie de proyección de la imagen y, a lo largo de la duración de toda la observación, se mantiene la proyección de forma estable sobre ella.
En el escenario seleccionado confluyen diferentes elementos adversos que dificultan las tareas de detección y seguimiento, como son cambios de iluminación, escala, giros y movimientos de la cámara y oclusiones por lo que hemos diseñado nuestro sistema para que sea capaz de gestionar todos estos distractores propios de ciertos escenarios adversos. Adicionalmente, se propone un sistema de reinicialización para aquellos casos en los que los distractores hagan imposible el seguimiento de la superficie y una proyección consistente.

Los objetivos de este proyecto son el estudio de las técnicas existentes para realizar una aplicación de realidad aumentada como esta, la implementación de la propia aplicación integrando las técnicas que mejores resultados aportan al procesamiento del escenario y la aplicación de métodos de corrección que hagan posible que el sistema sea estable frente a los principales distractores que aparecen en los entornos reales. La aplicación final requiere el funcionamiento sobre un dispositivo móvil por lo que los requisitos de tiempo real han sido claves en la selección de los métodos utilizados. Adicionalmente, se han aplicado técnicas para mejorar la sensación visual permitiendo la detección de objetos en primeros planos y su correcta visualización por delante de la imagen virtual.
La evaluación del sistema se ha realizado sobre diferentes videos englobando todos los distractores mencionados comparando los resultados con varios métodos y comprobando de forma cualitativa las mejoras obtenidas por el sistema propuesto.
Los resultados finales concluyen que el sistema es capaz de obtener un procesado suficiente como para disfrutar de una sensación de tiempo real, mejorando al mismo tiempo la fiabilidad del sistema frente a oclusiones leves, siendo capaz, al mismo tiempo, de detectar las oclusiones más severas, permitiendo iniciar un proceso de reinicialización en estos casos.

INDICE

A mis directores y guías Mario, Elías y Carlos por su paciencia y empuje para concluir este trabajo a tiempo y con éxito. A todas aquellas personas que me han apoyado durante este largo tiempo, sobre todo a aquellos a quienes impulsaba la fe. Pero muy especialmente a quienes ya no están conmigo y que tanto me acompañaron, Nati, Casio, Carmen y Ángel. Y por último, a mi apoyo diario, Teresa.

Capítulo 1: Introducción

En este capítulo se realiza una introducción general al desarrollo de este Proyecto Fin de Carrera, se describe el contexto donde se ha realizado el proyecto, un análisis del estado del arte para conocer los trabajos previos relevantes anteriores, y los objetivos. En capítulos posteriores y anexos se amplían algunos de estos aspectos. En el Capítulo 2 se describe en profundidad el sistema propuesto, sus etapas y las soluciones adoptadas a las diferentes problemáticas. A los largo del Capítulo 3 se analizan las alternativas chequeadas en cada una de las etapas y el porqué de las elecciones adoptadas. El Capítulo 4 contiene las conclusiones y resultados; y por último el Capítulo 5 el trabajo futuro. Al final del documento se añaden 4 anexos con información ampliada de distintos aspectos y por último la bibliografía.

Introducción

En los últimos tiempos el gran crecimiento y desarrollo tecnológico de los dispositivos móviles han conseguido cambiar el concepto de herramientas informáticas y de ocio. Los últimos modelos de *Smartphones* y *Tablets*, pueden compararse en potencia y funcionalidad a algunos pequeños equipos informáticos portátiles o de sobremesa añadiéndoles la gran ventaja de la movilidad absoluta.

Paralelamente a este desarrollo e impulsado por él, las compañías de telecomunicaciones y entes públicos y privados, han aceptado el reto de proporcionar cobertura de datos móviles y redes wifi abarcando prácticamente cualquier lugar de las poblaciones.

Todo ello hace que se planteen nuevos retos tanto a nivel de aplicaciones informáticas portables como de ocio, puesto que ahora cualquier momento y lugar puede representar una situación elegida para realizar algún trabajo de forma remota o para tomarse un descanso con algún tipo de aplicación lúdica haciendo uso de nuestros dispositivos móviles.

Con el afán de proporcionar nuevas herramientas y aplicaciones, nace el proyecto TAMA (Tecnologías Audivisuales Multimodales Avanzadas) de mano del Instituto de Investigación en Ingeniería de Aragón (I3A) y que pretende desarrollar un sistema que

permita integrar contenidos audiovisuales sobre elementos reales utilizando estas nuevas potencialidades de los dispositivos móviles. A esta combinación de elementos virtuales sobre un entorno físico del mundo real es lo que se conoce como “Realidad Aumentada (RA)”. Actualmente estos sistemas de RA, y muchos de los sistemas basados en visión por computador, se encuentran con dificultades en ciertos entornos, lo que hemos dado en llamar entornos adversos. Así, en condiciones de exterior donde la iluminación cambia constantemente, los observadores cambian su posición, los objetos del entorno se interponen en la línea de visión ocluyendo la zona de interés, etc... Estos sistemas dejan de ser estables y son necesarias herramientas de corrección para mantener los procesos de proyección y seguimiento.

Este Proyecto Fin de Carrera se ha realizado dentro del proyecto TAMA y consta de unas tareas de estudio, desarrollo de métodos e integración de resultados en un sistema que permitan la proyección de elementos virtuales dentro de un entorno real adverso.

El proyecto TAMA en su fase inicial, y a través del desarrollo de este PFC, pretende la proyección consistente de contenidos multimedia virtuales sobre elementos estáticos de un escenario real. Para ello, es necesario que imágenes de este escenario real lleguen hasta el sistema, bien provenientes de un video grabado o de un stream de imágenes capturadas en el momento. En ambos casos para la obtención de las imágenes se utiliza un dispositivo móvil tipo *Smartphone* o *Tablet*.

El sistema desarrollado en este PFC realiza la proyección de los contenidos sobre una superficie plana concreta denominada superficie de proyección. Dicho sistema es robusto a entornos adversos que contemplan los principales distractores que influyen en los sistemas basados en visión: movimiento de la cámara, rotación, cambios de escala, oclusiones, etc.

En el entorno elegido como ejemplo, la superficie de proyección será una de las fachadas del cubo de la Plaza de la Seo de Zaragoza sobre la que se quiere proyectar un cuadro de Goya. El resultado que se quiere obtener es la imagen del cuadro de Goya sobre la superficie de proyección elegida como se puede ver en la siguiente figura (Figura 1):

Figura 1: A la derecha se observa la imagen del Cubo de la Seo. A la izquierda se representa la proyección de imagen "virtual" sobre la superficie de proyección elegida

El sistema debe ser robusto a cambios de perspectiva, iluminación tamaño de imagen, posición del observador, etc… (ver Figura 2) de tal forma que, una vez conseguida una primera proyección, sea consistente con el flujo continuo de imágenes posteriores permitiendo que el observador tenga la impresión de que se trata de un objeto real (ver Figura 3).

Figura 2: Inicialización del sistema en distintas condiciones

Figura 3: Proyección coherente durante la secuencia

De todos los distractores que afectan a los sistemas de visión por computador, probablemente el más difícil de tratar sea el problema de las oclusiones. Por lo tanto, además de los distractores anteriores, merece especial atención que el sistema sea capaz de actuar frente a oclusiones. En este aspecto, el comportamiento del sistema será el siguiente:

- Frente a oclusiones "leves", el sistema debe poder superarlas (ver Figura 4).

- Frente a oclusiones “graves”, el sistema contemplará un detector de la situación En este momento se quedará a la espera de que la oclusión desaparezca o sea lo suficientemente pequeña como para permitir que se reanude el seguimiento (ver Figura 5).

Figura 4: Gestión de oclusión leve

Figura 5: Gestión de oclusión grave, mecanismo de detección

Contexto

El trabajo que se va a exponer en estos documentos se ha realizado como Proyecto Final de Carrera (PFC) dentro de la Titulación de Ingeniería Industrial por la Universidad de Zaragoza. Está dirigido por el Ingeniero en Telecomunicación Mario F. Rodríguez Martínez y actúa como ponente el Dr. José Elías Herrero Jaraba ambos miembros del Laboratorio de Visión por Computador (*CvLab*).

La aplicación que proponemos se enmarca dentro de las líneas de trabajo abiertas en este Laboratorio (*CvLab*) que forma parte del departamento de Ingeniería de Electrónica Comunicaciones de la Universidad de Zaragoza y del Instituto de Investigación en Ingeniería de Aragón (*I3A*).

El Grupo (*CvLab*) está formado por personal permanente de la Universidad en calidad de Investigadores y un conjunto variable de ayudantes y estudiantes que realizan proyectos final de carrera o doctorados en el seno del grupo. El autor del proyecto pertenece como proyectando en este laboratorio.

El grupo de *CvLab* tiene varias líneas de investigación abiertas en temas de detección y seguimiento de personas, reconocimiento de actividades, monitorización del tráfico, reconocimiento de patrones, realidad aumentada o segmentación de imagen. Sobre estas áreas se han realizado aportaciones que se plasman en publicaciones relevantes.

Dentro de las diferentes líneas de investigación mencionadas se encuentra el Proyecto de Tecnologías Audiovisuales Multimodales Avanzadas o TAMA financiado por el Gobierno de Aragón, que pretende avanzar en el desarrollo de tecnologías audiovisuales desde una perspectiva global y multimodal, mediante la fusión de distintos campos de investigación como son la Informática Gráfica (IG), el Reconocimiento de Patrones (PR), Machine Learning (ML), Procesamiento de Imágenes (IP), Visión por Computador (CV), Procesamiento del habla (SP), Realidad Aumentada (AR), etc. Ello permitirá avanzar el estado del arte de las tecnologías audiovisuales y desarrollar sistemas multimodales interactivos avanzados que lleven a cabo la deseada sinergia entre las personas y las máquinas de una forma transparente y sencilla. Es en este proyecto donde se enmarca la labor desarrollada en este PFC.

Estado del arte

Para abordar un proyecto de cualquier índole, es necesario disponer de una visión de los avances realizados en el campo en que se enmarca dicho proyecto. Para este trabajo en concreto, se requiere el conocimiento de técnicas de Visión por Computador que permitan realizar las siguientes tareas: extracción de puntos característicos, emparejamiento de características, seguimiento o tracking de puntos, etc.

- **Extracción de características específicas de una imagen.** Se busca un tipo de características que permita identificar puntos de la imagen de una forma precisa y lo más unívoca posible. El proceso de extracción de características consta de dos fases:
    - La primera de ellas consiste en una extracción de las coordenadas de los puntos que mejor cumplen con las características que impone el método utilizado.
    - En la segunda fase, por cada uno de los puntos obtenidos, se extrae un descriptor del mismo. Un descriptor es una representación del punto basada en las características de la imagen en el entorno de dicho punto característico y que sirve para modelarlo e identificarlo de la forma más única posible pues será a través de este descriptor que se realizarán las tareas de emparejamiento posteriores.

  Existen diversos tipos de características (o '*features*') descritas en la bibliografía con distintas virtudes y defectos.

    a. **Características SIFT (**Scale-Invariant Feature Transform) [1] [2]. Son características muy robustas debido a que los descriptores en las coordenadas adecuadas son invariantes a escala y a rotación. Como contrapartida, su extracción de coordenadas es muy costosa computacionalmente.
    b. **Características SURF (**Speeded Up Robust Features) [3] [4]. También son invariantes a escala y rotación e inspiradas en las anteriores SIFT. La extracción de coordenadas características es más rápida que en SIFT, basadas en sumas de 2D Haar wavelet [5] [6] haciendo uso eficiente de las imágenes integrales. Son algo menos robustas que las SIFT frente a

escalado debido a que sacrifican algunas de las operaciones dedicadas a este aspecto en pro de ganar en velocidad de procesado.

c. **Características FAST (**Features from Accelerated Segment Test) [7] [8] es un método de extracción de esquinas (ver Anexo 1) que puede ser usado para detección de coordenadas característicos y tracking de objetos. La ventaja más destacada es su eficiencia computacional. Se pueden utilizar varios tipos de descriptores como los obtenidos en SIFT o SURF. También se emplean descriptores diseñados para este tipo de características como el BRIEF **(**Binary Robust Independent Elementary Features) [9]. Se trata de un descriptor que, mediante lo que denomina '*test binarios*', obtiene un vector $\{0,1\}^N$ (al que llama '*bitstring*') como descriptor eficiente de puntos característicos. Son descriptores robustos a cambios de iluminación, distorsión de la perspectiva y desenfoque, pero muy sensibles a rotación. Como medida de similaridad puede utilizarse la distancia de Hamming cuyo cómputo es más eficiente que la norma L_2 utilizada habitualmente. Su descripción detallada puede encontrarse en el Anexo 2.

d. **Características oFAST** (Oriented-FAST) [10]**.** Se trata de un algoritmo para extraer coordenadas características de tipo FAST aprovechando así su eficiencia computacional y que, además, contengan información de la orientación. El descriptor diseñado para este tipo de características es el rBRIEF **(**Rotated BRIEF) [10]. Los descriptores rBRIEF son una modificación de los BRIEF que les permiten ser invariantes a las rotaciones en el plano. Son un tipo de descriptores que permiten utilizar la información de rotación que incorporan los puntos oFAST para construir '*bitstring*' de palabras binarias invariante a rotación. Para facilitarles mayor fiabilidad frente a la perdida de varianza y reducir la correlación entre los '*test binarios*', se incorpora un método de aprendizaje para la elección de un buen subconjunto de los '*test binarios*'.

e. **Características "Good Features to Track"** [11]. En realidad se trata de un método de extracción de coordenadas mediante detección de esquinas robusto y medición de la similaridad entre fotogramas que permita un mejor seguimiento. El método utiliza dos modelos de

movimiento en la imagen. El primero de ellos propone seleccionar puntos con un alto contenido en '*textura*' y por lo tanto inmunes al ruido y al problema de la apertura (ver la descripción de estas características en el Anexo 1 para más información). Estos puntos esencialmente son puntos '*esquina*'. El segundo de ellos propone un emparejamiento de características entre fotogramas mediante una medida de "similaridad" basada en transformaciones afines que contemplen deformaciones y no solamente desplazamientos. El artículo propone descriptores propios aunque se pueden utilizar otros como los ya descritos.

- **Emparejamiento robusto de características.** Para realizar las correspondencias de características entre dos imágenes es necesario unos descriptores lo más unívocos posible. Determinados los descriptores, las características detectadas en una imagen pueden ser emparejadas con ellas mismas si aparecen en otra imagen distinta. Al proceso de reconocer una misma característica en dos imágenes diferentes es lo que se denomina como emparejamiento o *''matching''*. Es importante entonces encontrar descriptores que cambien lo menos posible al variar la perspectiva, la escala, la iluminación, etc… para una misma característica, de tal forma que si el descriptor cumple estos requisitos sería fácil detectar esta característica en dos imágenes distintas cualesquiera. Existen distintos métodos para el emparejamiento de descriptores siendo los siguientes algunos de los más habituales:
  - Norma L2
  - Distancia de Hamming.
  - Sum of Square Difference (SSD).
  - Otros.

  Para obtener un emparejamiento o *'matching'* robusto se utilizan técnicas de refinado del emparejamiento. El más conocido y quizá el más confiable es el método RANSAC [12] [13]. Este método de ajuste de características a geometrías conocidas es especialmente útil en el filtrado de puntos para el cálculo de homografías entre planos [14]. Se puede consultar una explicación más exhaustiva en el Anexo 2.
- **Seguimiento (o "tracking") de características.** Una vez detectadas una serie de características en la imagen, puede realizarse un seguimiento de las mimas a

través de las imágenes consecutivas de un video o recibidas de dispositivos de streaming como cámaras o móviles. El proceso es conceptualmente distinto al de emparejamiento de características ya que en este proceso se dispone de información de unas características ya encontradas en un fotograma previo por lo que se puede asumir algunas restricciones. Al tratarse de imágenes consecutivas, es de esperar que las condiciones de iluminación y escala sean similares, así como que el desplazamiento de las características de una a otra imagen no sean excesivo y/o siga algún patrón. Muchos de los algoritmos de *'matching'* en realidad se basan en emparejamiento de características aprovechando estas restricciones.

- **Algoritmo de tracking KLT de flujo óptico disperso** [15] [16]: Algoritmo de seguimiento basado de Lucas-Kanade calculando el flujo óptico de forma piramidal. Es un método muy rápido y que obtiene muy buenos resultados en ausencia de oclusiones.
- **Algoritmo de tracking de Kalman** [17] [18]. El filtro de Kalman es un algoritmo desarrollado por Rudolf E. Kalman en 1960 que sirve para identificar el estado oculto (no medible) de un sistema dinámico lineal, pero sirve además cuando el sistema está sometido a ruido blanco aditivo. La diferencia del filtro de Kalman sobre otros métodos radica en que Kalman es capaz de escoger la ganancia K de realimentación del error de forma óptima cuando se conocen las varianzas de los ruidos que afectan al sistema. En este proyecto se utiliza la implementación piramidal detallada en [18].
- **Algoritmo de tracking FFME (**Fast Feature-based Motion Estimation) [19]. Es un método de estimación del movimiento robusto al ruido y al problema de la apertura. Esto se logra imponiendo restricciones relacionadas con la magnitud del gradiente, el nivel de ruido adaptativo, la similitud o no con un punto esquina, y la diferenciación apreciable frente a puntos vecinos. Para la caracterización de cada punto singular se utiliza un descriptor complejo, que presenta una gran robustez a los cambios de iluminación y las pequeñas variaciones en el punto de vista de la cámara. La correspondencia entre los puntos singulares de imágenes consecutivas se realiza mediante la minimización de la distancia Euclídea como medida de similaridad con el descriptor del

punto en el fotograma anterior. El conjunto de correspondencias produce un campo vectorial disperso de movimiento que describe con precisión el movimiento de la imagen.

Objetivos

Los objetivos que se plantean en este proyecto son los siguientes:

1. Inicialización robusta. La secuencia de imágenes que se analizan, bien de un video o de un '*stream*' de imágenes, no tiene por qué comenzar con la superficie de proyección bien encuadrada. Es necesario que el sistema disponga de una forma de detección de este hecho y sea capaz de permanecer a la espera, o incluso de guiar al usuario hasta que la superficie donde se quiere proyectar aparezca en el campo de visión.
2. Tracking consistente. Una vez localizada la superficie de proyección, se debe proyectar la imagen virtual de manera consistente durante todo el tiempo que dure la secuencia de imágenes, es decir, debe observarse estable, sin vibraciones visualmente molestas y en la posición ocupada por la superficie real.
3. Robustez a cambios de iluminación, rotaciones, cambios de escala. El sistema debe superar los distractores habituales que afectan a un sistema basado en visión por computador. Durante la secuencia de imágenes pueden producirse pequeños cambios de iluminación, cambios de escala debido a que el observador puede acercarse o alejarse de la superficie de proyección o rotaciones y desplazamientos propios de un sistema móvil producidos por el propio movimiento del usuario.
4. Gestión de oclusiones. Solventar las leves y reiniciar en las graves. El principal problema aún sin resolver de un sistema de visión como el propuesto, es el manejo de las oclusiones que afectan al objeto en estudio. En este proyecto se han analizado secuencias de oclusiones leves y severas y se ha preparado el sistema para que supere las primeras y detecte las segundas. En caso de una oclusión grave, el sistema queda a la espera de que desaparezca para retomar el seguimiento y la proyección del contenido.
5. Detección de profundidad: Para que la proyección del contenido tenga consistencia visual para el observador, debe realizarse una detección de los objetos que se encuentran en primer plano por delante de la superficie de

proyección. Esta detección nos permite superponer a la imagen proyectada los objetos de primer plano, dando una sensación visual más real y con sensación de profundidad.

6. Procesamiento en tiempo real. El sistema debe funcionar a una tasa de procesamiento que ofrezca al usuario una sensación de continuidad. Esta restricción obliga a ciertas elecciones que sacrifican cierto grado de fiabilidad en favor de la velocidad de procesado.

Capítulo 2: Estudio del sistema propuesto

Como se ha descrito en la Introducción, el sistema objetivo de este trabajo realiza la proyección estable de una imagen sobre un entorno real fusionando ambos contenidos para dar la impresión al observador de que se trata de una imagen real.
Los requisitos y objetivos se han descrito en el apartado de Objetivos y aquí se van a detallar la infraestructura utilizada tanto software como hardware y el funcionamiento general de todas las partes del sistema.

2.1. Infraestructura

A pesar de que el objetivo final del sistema es su funcionamiento en plataformas móviles, para el desarrollo, pruebas y comparación de resultados de diversas alternativas se ha utilizado un ordenador tipo PC. Las características concretas de la plataforma de desarrollo son:

1. Ordenador Intel Core2 Duo E7400 de 2.8GHz y 4 GB de memoria RAM.
2. Sistema operativo Windows 7 de 64 bits.
3. IDE de desarrollo Eclipse IDE for C/C++ Developers Version: Juno Service Release 2. CDT para C/C++ 8.1.2.
4. Compilador MinGW32-g++ 4.8.1.
5. Biblioteca OpenCV de procesamiento de imagen versión 2.4.6 [20] [21].
6. Videos grabados con varios dispositivos móviles de resoluciones de 1280x720, 640x480 y 1920x1080.

2.2. Funcionamiento del Sistema

El funcionamiento del sistema desarrollado sigue un esquema de control en bucle cerrado que permite una realimentación de los resultados obtenidos. Este bucle de realimentación permite realizar correcciones sobre el funcionamiento y es el principal causante de la estabilidad del proceso frente a las oclusiones (ver Figura 6).

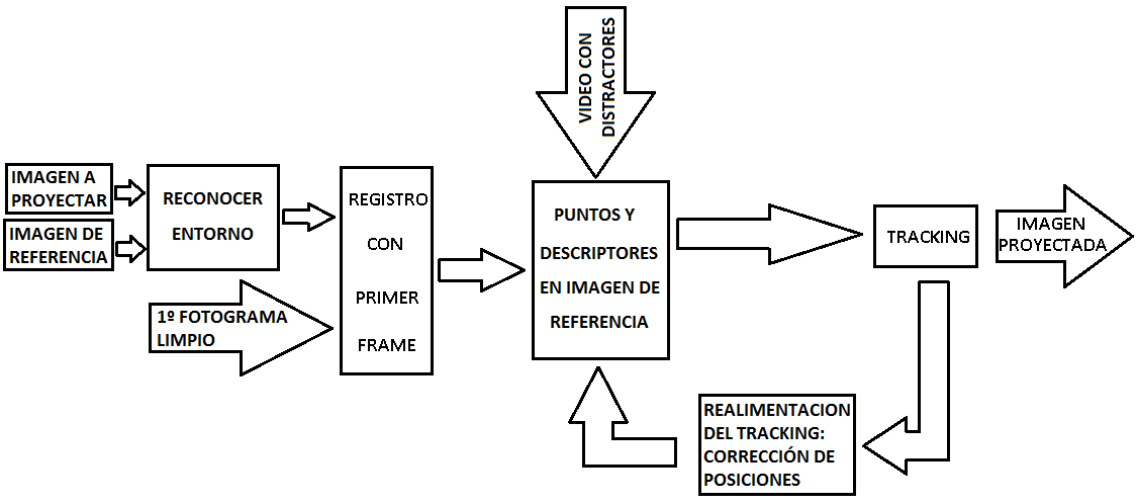


Figura 6: Esquema de control en bucle cerrado

En este esquema se observa que a partir de la '*Imagen a Proyectar*' ('I_P'), y la '*Imagen de Referencia*' ('I_R') etiquetada se consigue establecer una posición del observador en el entorno, lo que se traduce como una forma de relacionar ambas imágenes. Posteriormente, mediante un primer fotograma limpio ('I_0') de la secuencia se consigue lo propio con la posición del nuevo observador. Con ambas imágenes, la de referencia ('I_R') y la del fotograma ('I_0'), se extraen características y descriptores que se emparejan y se envían al módulo de tracking. Este proceso, tomando la secuencia de fotogramas del video y los descriptores, realiza un seguimiento de las posiciones de las características. Dado que en el video aparecen descriptores que alteran el funcionamiento normal del tracking, se hace necesario un bucle de realimentación para corregir las posibles desviaciones de las posiciones. El resultado es la imagen proyectada sobre la superficie seleccionada.

Una visión más detallada del sistema se puede observar en la Figura 7.

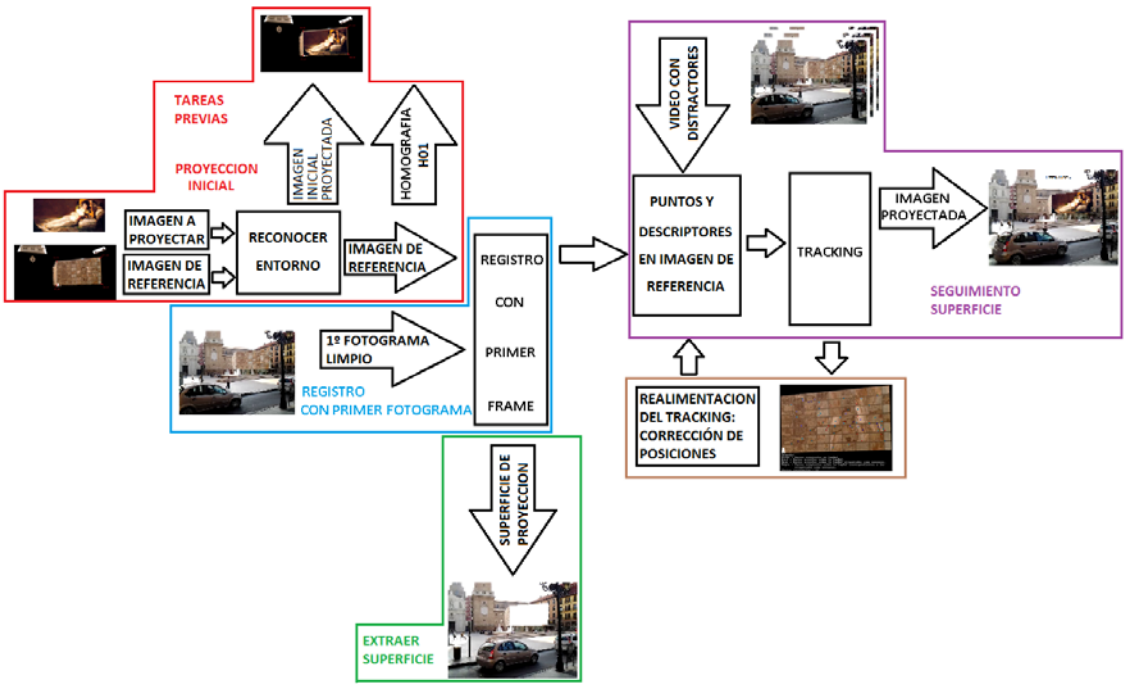


Figura 7: Esquema detallado del sistema

Tareas previas

Este bloque del sistema consta de una serie de tareas que pueden y deben realizarse con anterioridad a la puesta en marcha del sistema. Son tareas principalmente manuales que permiten calibrar y agilizar el procesamiento posterior.

Para realizar una proyección de una imagen sobre una superficie, es necesario obtener los siguientes datos previos:

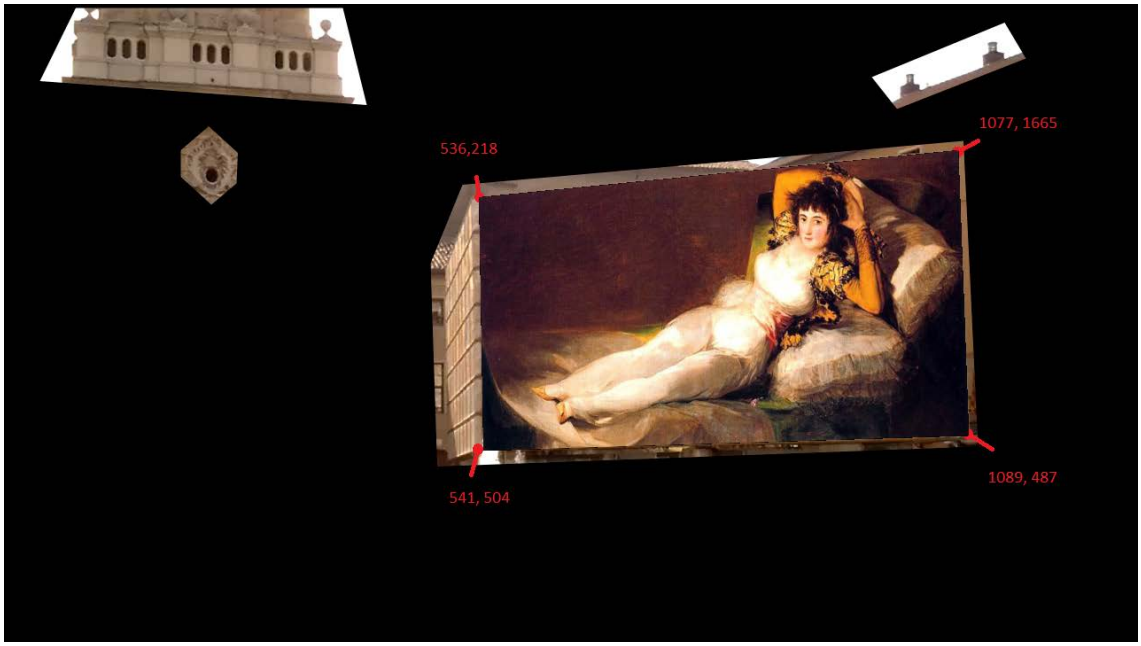
1. Obtención de la '*Imagen a Proyectar*' ('I_P'). Como imagen ejemplo de realidad aumentada se ha elegido el cuadro de la 'Maja vestida' de Goya pero puede ser cualquiera. Además de la propia imagen es necesario disponer de las posiciones de las 4 esquinas del cuadro. Es aconsejable que la imagen se encuentre bien encuadrada y sin giros ya que, de esta forma, las cuatro esquinas se corresponden con el propio tamaño de la imagen, ancho y alto.

2. Obtención de una o una serie de imágenes del escenario real sobre el que se va a proyectar. Para conseguir que estas imágenes sirvan como inicialización para el sistema, es necesario etiquetar a mano los puntos correspondientes a las 4 esquinas de la fachada de la Seo (o en su caso de la superficie sobre la que se quiere proyectar). A esta imagen inicial etiquetada la llamaremos '*Imagen de Referencia*' ('I_R'), y tiene como objetivo servir como una referencia al sistema cuando un usuario intenta reconocer el escenario por primera vez. Esta 'I_R' también puede disponer de información adicional que servirá para realizar la primera correspondencia de imágenes, ya sea en forma de recorte de zonas de interés adicionales a la fachada o ya sea con algún tipo de descriptor de características de imagen o semánticos.

3. Obtención de la correspondencia entre la 'I_P' y la 'I_R'. Se obtiene una matriz de transformación que permite proyectar la 'I_P' sobre la 'I_R' *ya* etiquetada. Al

tratarse de dos superficies planas, esta transformación se corresponde con una homografía, matriz de 3x3 (‘H_{P_R}’) que permite el cambio de coordenadas y la proyección de una imagen sobre la otra (ver explicación completa en el apartado *‘matching’* del Anexo 2). A este proceso es al que denominamos como ‘*Reconocer entorno*’ en la Figura 7.

4. Obtención de la imagen inicial proyectada. Este proceso sirve como comprobación de que las tareas previas se han realizado correctamente. Se puede observar el resultado sin más que aplicar esta transformación ‘H_{P_R}’ a la ‘I_P’.



5. Extracción de los puntos característicos de la ‘I_R’ y sus descriptores por el método SIFT, ‘*puntos_ref*’ y guardarlos puesto que serán los datos que se utilizarán en el emparejamiento y tracking posteriores.
6. Adquisición de secuencias de imágenes: Es necesaria la grabación de videos de pruebas en diferentes momentos del día y con distintas condiciones que permitan tener ejemplos de las distracciones que se pretender tratar en el procesado.

Registro con primer fotograma

Esta etapa es la primera que se realiza ya con el sistema en funcionamiento continuo. Consiste en la correspondencia entre la primera imagen limpia recibida por el sistema desde la fuente de imágenes y la ‘I_R’ entrenada en la etapa anterior. Es necesario que el primer fotograma, ‘I_0’ en que la superficie de proyección se vea completamente, éste se

encuentre limpio de oclusiones ya que, en el estado actual del sistema, no es posible detectar oclusiones en un instante tan temprano. Por lo tanto, el sistema se queda en espera de recibir un fotograma en el cual la superficie de proyección se encuentre completamente encuadrada dentro del campo de visión de la cámara:

Algoritmo:

Do

Recibir fotograma 'I_T'.

Extraer puntos característicos y descriptores por método SIFT de la imagen I_T, '*puntos_ini*'.

Emparejar '*puntos_ini*' con '*puntos_ref*' para obtener la homografía 'H_{R_T}'.

Proyectar las 4 esquinas de la 'I_P' sobre la imagen 'I_T'.

if 'I_P' * 'H_{R_T}' está completa sobre 'I_T'

then end bucle Do

while ('I_P' * 'H_{R_T}' NO está completa sobre 'I_T')

Guardar la 'I_T' como primer fotograma limpio.

En la Figura 8 se muestra que, aunque el sistema ha conseguido emparejar las características suficientes entre 'I_R' e 'I_T' como para obtener una buena 'H_{R_T}', sigue buscando una situación en la que la zona de proyección se encuentre completa dentro del campo de visión de la cámara.

Figura 8: Proceso de inicialización. Se buscan puntos SIFT para detectar la superficie de proyección. Si la zona de proyección no es visible completamente (parte izquierda de la figura) se sigue buscando el momento en que lo sea (parte derecha de la figura). A partir de ese instante el tracking KLT puede comenzar normalmente.

Este bloque está compuesto por:

Entradas:

1. 'I_R' etiquetada.
2. Primer fotograma limpio de la secuencia de imágenes 'I_0'.

Salidas:

1. Descriptores de características. Como resultado de este registro, se obtienen una serie de puntos característicos en la superficie de proyección de la 'I_R' y las coordenadas de estos mismos puntos en la superficie de proyección del primer fotograma de la secuencia, 'I_0'. Tomando estos puntos en ambas imágenes, se extraen sus descriptores SIFT y se guardan para procesos de emparejamiento y filtrados posteriores (ver Figura 9). La extracción de puntos SIFT y sus descriptores se detalla en los apartados Características SIFT**.** y Descriptores de los anexos Anexo 1 y Anexo 2, respectivamente.
2. Primer fotograma limpio, 'I_0'. Este fotograma y en concreto la superficie de proyección, se guarda como fondo para el proceso de cálculo de objetos de primer plano que se describe en Etapa de Extracción de la superficie de proyección del Capítulo 3. Llamaremos '*fondo_ini*' a la imagen de la superficie de proyección en este instante.

Proceso:

1. Extracción de puntos SIFT en la región de proyección de la 'I_R', a los que llamaremos '*puntos_ref*'. Es posible restringir la extracción a esta zona gracias a las tareas previas que nos permiten extraer cuál es la superficie de proyección mediante la correspondencia homográfica a través de 'H_{P_R}'.
2. Extracción de puntos SIFT en el 'I_0', a los que llamaremos '*puntos_ini*'. En este caso debe hacerse la extracción sobre toda la imagen puesto que aún no se ha delimitado cuál será la superficie de proyección en esta imagen.
3. Emparejamiento robusto de los '*puntos_ref*' y los '*puntos_ini*' mediante la técnica del doble emparejamiento por fuerza bruta y tomando como medida de similaridad la 'norma L_2'. Este proceso se detalla en el apartado *'matching'* del Anexo 2.
4. Filtrado de puntos espurios o 'outlier mediante el cálculo de la homografía por RANSAC dando como resultado 'H_{R_0}'. Este proceso se detalla en el apartado *'matching'* del Anexo 2.

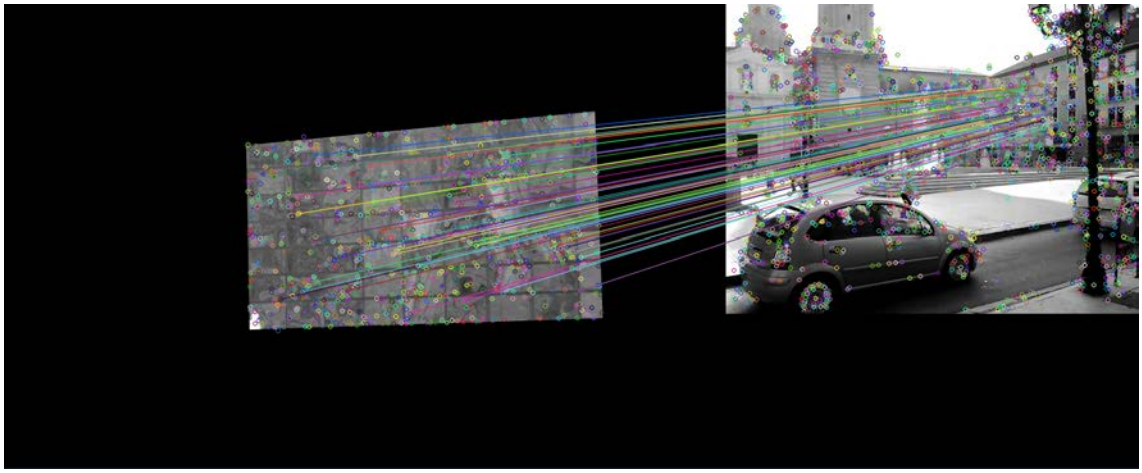


Figura 9: Extracción de puntos SIFT y *'matching'* inicial

Para una imagen 'I_0' de resolución 640x480 píxeles en la que se extraen 2432 características SIFT, el proceso completo de registro con la 'I_R' consume 'T_r' ≈ 1.17 segundos. Esta carga computacional se encuentra lejos del procesado en tiempo real, aunque en este momento no es tan crítico dado que el registro completo sólo es necesario con el primer fotograma.

Sin embargo, durante el transcurso de esos ~1.2 segundos aproximadamente, es muy posible que el observador modifique la posición del dispositivo respecto al escenario por el simple hecho de sostenerlo o andar con él en la mano. Es necesario pensar en un sistema que relacione la posición del dispositivo justo en el momento en que se toma la imagen 'I_0' con la posición de dicho dispositivo cuando concluye el registro 'T_r' = 1.2 segundos después.

Para ello, antes de iniciar el proceso de registro, se lanza en paralelo un sistema de tracking KLT con características "*Good Features To Track*" mucho más ligeras de calcular (ver Figura 10). El resultado de este tracking permitirá hacer corresponder, mediante la correspondiente acumulación de homografías, 'H_{0_Tr}', la imagen 'I_0' con la situación 'I_{0+Tr}':

$$H_{0\_T_r} = \prod_{t=0}^{t=T_r-1} H_{t\_t+1}$$

Dado que el registro inicial da una matriz de homografías 'H_{R_0}', que relaciona la 'I_R' con la 'I_0', podemos relacionar la 'I_0' con la 'I_{Tt}' en este instante componiendo las dos matrices, H_{R_Tr}':

$$H_{R\_T_r} = H_{0\_T_r} * H_{R\_0}$$

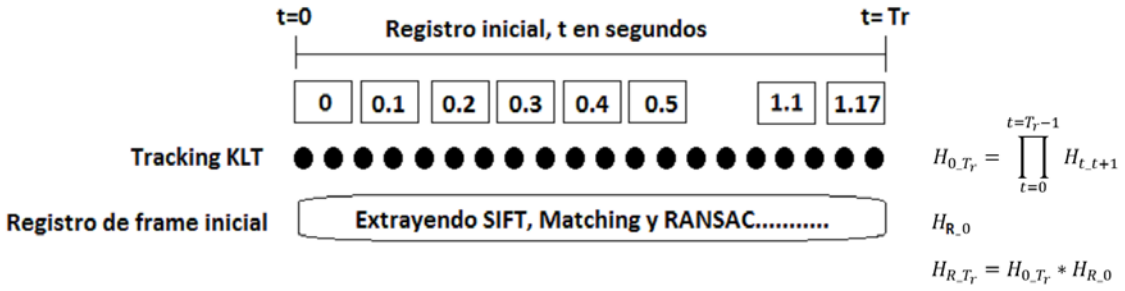


Figura 10: Tracking KLT en paralelo con el registro del primer fotograma para corregir desplazamientos durante el procesado.

Extraer superficie de proyección

Este proceso es resultado directo del emparejamiento anterior. Este resultado obtenido se puede caracterizar como una segunda matriz de homografía que permite hacer corresponder la superficie de proyección de la 'I_R' con la superficie de proyección en el primer fotograma, 'H_{R_0}'. Recordemos que una homografía solo es capaz de hacer corresponder puntos coplanares. El razonamiento es sencillo:

1. Se dispone de una '*Imagen a Proyectar*', 'I_P' relacionada directamente con la 'I_R' a través de 'H_{P_T}', resultado de las Tareas previas.
2. Se ha conseguido, mediante la etapa anterior de registro, relacionar la 'I_R' con la 'I_0' de la secuencia de imágenes por medio de 'H_{R_0}'.
3. Por lo tanto, se ha conseguido relacionar la 'I_P' y la 'I_0' sin más que componer las dos homografías como se puede ver en la Figura 11:

$$H_{P\_0} = H_{R\_0} * H_{P\_R}$$

Merece la pena comentar brevemente el proceso de proyección de la 'I_P' sobre la superficie de proyección en el fotograma correspondiente.

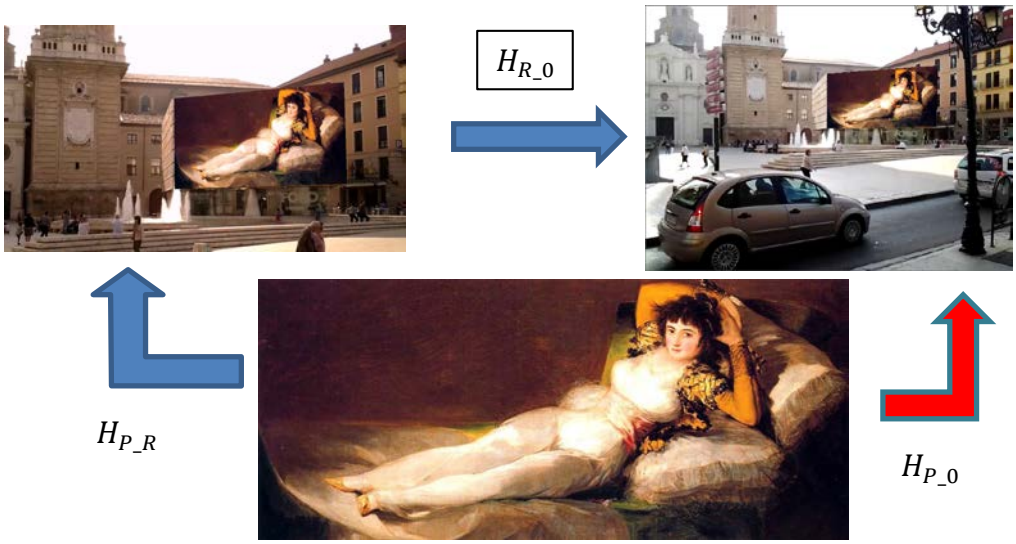


Figura 11: Doble correspondencia homográfica, relación entre la Imagen a Proyectar, I_P y el primer fotograma de la secuencia, I_0

Como se observa en la Figura 11, esta proyección no muestra ningún espacio ni hueco sin rellenar tal como cabría esperar si se proyectase la 'I_P' directamente sobre el fotograma, dado que la transformación en la perspectiva debería dejar puntos en la superficie para los que no hay una correspondencia directa. Para asegurar que no ocurre esto, el cálculo de la 'I_P' sobre el fotograma se hace en el sentido inverso, es decir:

- Se toman todos los puntos del fotograma 'I_T', '*puntos_t*'.
- Se les aplica la homografía inversa para situarlos en el sistema de referencia de la 'I_P', '*puntos_t_inv*'.
- Se toman todos los '*puntos_t_inv*' que caen sobre la 'I_P'.
- Se toma el valor del píxel (en sus tres canales) de la 'I_P' y se aplica al '*puntos_t*' correspondiente.
- En caso de que uno de los '*puntos_t_inv*' no tenga una correspondencia exacta con un píxel de la 'I_P', se interpola linealmente entre sus píxeles vecinos (siendo $w_{i,j}$ los pesos correspondientes a cada uno de los 4 puntos vecinos en función de la distancia al punto p_{inv}).

$$\forall p(x,y) \in I_T, \rightarrow p_{inv}(x,y) = H_{P\_T_{inv}} * p(x,y)$$

$$Sea\ la\ interpolación\ lineal, p^*_{inv}(x,y) = \sum_{i=1,j=1}^{i=4,j=4} w_{i,j} * p_{inv}(x_i, y_j)$$

$$p(x,y) = \begin{cases} p^*_{inv}(x,y)\ en\ I_P: & Si\ p_{inv}(x,y) \in I_P \\ p(x,y)\ en\ I_T: & Si\ p_{inv}(x,y) \notin I_P \end{cases}$$

Computacionalmente, este proceso de emparejamiento es el más costoso. Se han probado diversos métodos con distintos resultados y costes computacionales. La descripción completa de todos ellos se detalla en el Capítulo 3 y en el apartado Coste computacional de los diferentes métodos de tracking del Anexo 3.

Tracking de la superficie

Realizado el registro con el fotograma inicial de la secuencia, el sistema se centra en el seguimiento de los '*puntos_ini*' y sus descriptores, '*descriptores_ini*' obtenidos durante la etapa anterior de un fotograma en el instante t al siguiente en el instante $t + 1$.

Si el sistema es capaz de realizar el seguimiento de estos puntos de un fotograma al siguiente de una forma consistente, el problema se resuelve.

El emparejamiento utilizado en esta etapa es el KLT, Kanade-Lucas Tracker [15]. En concreto, el KL basado en flujo óptico de forma piramidal [16] explicado en el apartado Métodos de tracking del Anexo 3. Es un método que ofrece un muy buen compromiso entre velocidad de procesamiento y fiabilidad de resultados en ausencia de oclusiones.

Este proceso permite extraer una relación homográfica, 'H_{R_T}', entre la 'I_R' y el fotograma actual 'I_T' que, compuesta con la inicial 'H_{P_R}', nos relaciona la 'I_P' con la 'I_T' como se muestra en la Figura 12:

$$H_{P\_T} = H_{R\_T} * H_{P\_R}$$

El cálculo de esta nueva relación homográfica, 'H_{R_T}', también es sometido al proceso de filtrado mediante RANSAC lo cual permite filtrar puntos espurios que el tracking KLT no ha sido capaz de detectar.

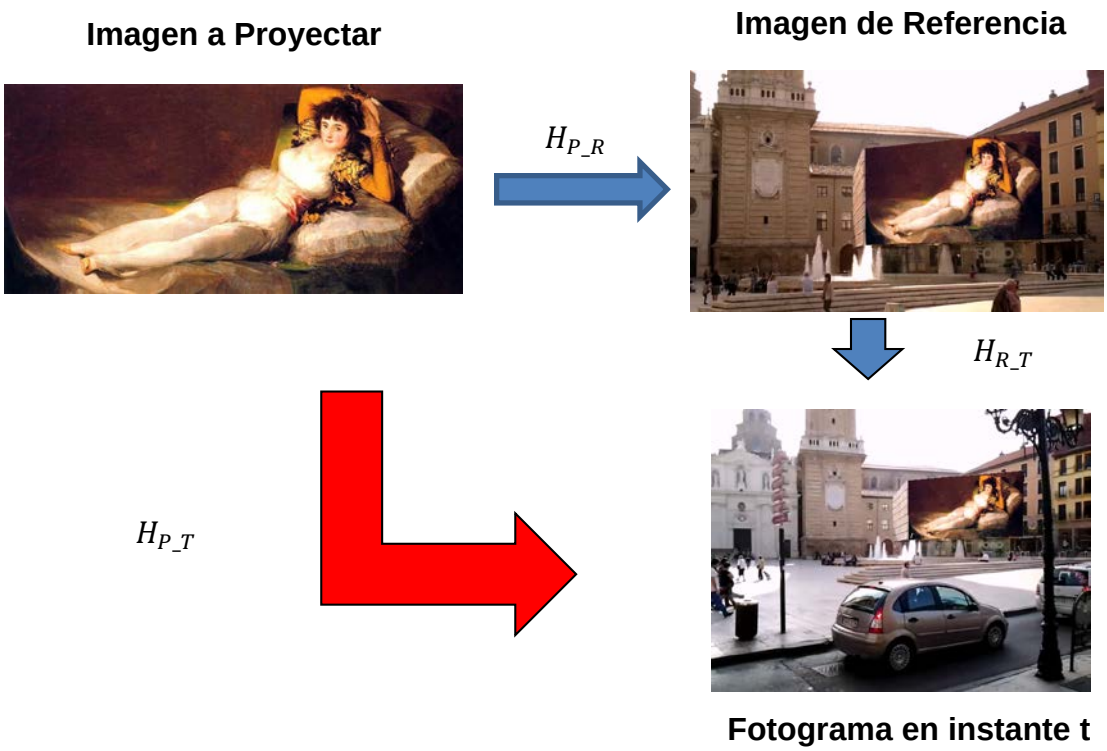


Figura 12: Proyección de la imagen sobre cualquier fotograma de la secuencia por medio del tracking de puntos

Este sistema hasta este momento en bucle abierto, cumple muy bien con muchos de los objetivos planteados en todos los casos chequeados si nos encontramos en una situación ausente de oclusiones:

1. Inicialización robusta. El registro inicial mediante emparejamiento de puntos SIFT permite una inicialización muy robusta (ver Figura 2).
2. Tracking consistente. El tracking KLT en esta implementación piramidal es resistente a cambios de escala y rotaciones no demasiado bruscas (ver Figura 3).
3. Robusto a cambios de iluminación, rotaciones, cambios de escala. Se ha comprobado que si las variaciones de iluminación, rotaciones o cambios de escala (al variar la distancia a la superficie de proyección) no son excesivos, este sistema funciona de una forma bastante correcta (se detectan ciertas vibraciones en algunos casos que dan una impresión visual molesta) (ver Figura 3 y Figura 13).

Figura 13: Sistema de tracking KLT soporta giros no muy bruscos con cierta vibración

4. Gestión de oclusiones, solventar las leves y reiniciar en las graves. En las secuencias chequeadas en que la superficie de proyección es ocluida en parte o casi totalmente por algún objeto cercano a la cámara, el sistema en bucle abierto falla (ver Figura 14). Es para resolver este problema para el que es necesario plantear un lazo de realimentación que permita detectar y corregir las desviaciones observadas en el proceso de seguimiento.

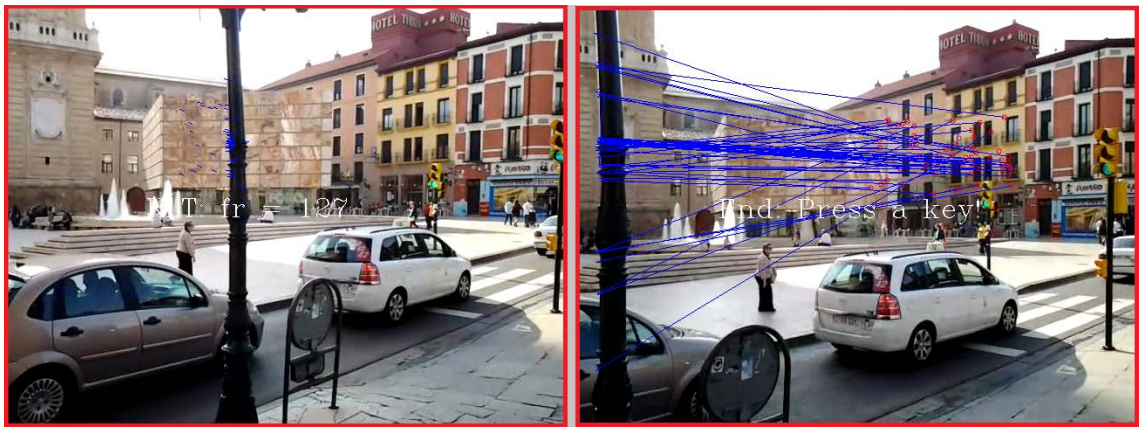


Figura 14: Sistema de tracking KLT falla en situaciones con oclusiones.

Realimentación del tracking y corrección de errores

Como se ha comentado en el apartado anterior, es necesario implementar un sistema de realimentación que permita al sistema de tracking detectar y corregir situaciones en las que, principalmente por alguna oclusión, los puntos característicos no están siendo emparejados correctamente entre dos fotogramas consecutivos:

Entradas:

1. Coordenadas de los '*puntos_next*'. Como ya se ha comentado, hemos llamado '*puntos_ini*' a las posiciones de los puntos del fotograma en el instante t. Estos puntos se toman como la entrada del proceso del tracking KLT. A las posiciones correspondientes a los '*puntos_ini*' en el instante t + 1 que el proceso de tracking devuelve como salida, las llamaremos '*puntos_next*'.
2. Coordenadas de los '*puntos_ref*'. Tal como se describe en el apartado de la descripción del proceso en la etapa de Registro con primer fotograma, los '*puntos_ref*' son las coordenadas de los puntos SIFT extraídos y emparejados en la 'I_R'.
3. Homografía que relaciona la 'I_R' con 'I_T'. Esta homografía sería la compuesta por:

$$H_{R\_T} = H_{R\_T}(t)$$

Salidas:

1. Coordenadas de '*puntos_next*' válidos. Son los '*puntos_next*' que el filtrado que se describe en el apartado siguiente de Proceso extrae como puntos correctamente emparejados. Llamaremos '*puntos_next_ok*' a estos puntos.
2. Coordenadas de '*puntos_next*' no válidos. Son aquellos '*puntos_next*' que el filtrado que se describe en el apartado siguiente de Proceso extrae como puntos incorrectamente emparejados. Llamaremos '*puntos_next_ko*' a estos puntos.
3. Homografía que relaciona la 'I_R' con la 'I_T' corregida. Tomando como emparejamientos los '*puntos_next_ok*' anteriores y los '*puntos_ref*' correspondientes a estos puntos, se recalcula la 'H_{R_T}'.

$$H_{R\_T_{OK}} = findHomography(puntos\_next\_ok, puntos\_ref)$$

4. Coordenadas de '*puntos_next*' corregidas. El resultado final del proceso devuelve un conjunto de '*puntos_next*' modificado:
    a. Los '*puntos_next*' correspondientes a los '*puntos_next_ok*' se mantienen sin ninguna variación puesto que son aquellos puntos no afectados por distractores y que el KLT ha seguido correctamente.
    b. Los '*puntos_next*' correspondientes a los '*puntos_next_ko*' son corregidos de la siguiente forma:
        - Se toman los '*puntos_ref*' que corresponden a los '*puntos_next_ko*'.

- Se calculan las coordenadas de los '*puntos_ref*' en el fotograma actual aplicándoles la homografía corregida con los '*puntos_next_ok*':

$$puntos\_next\_ko\_in\_t_1 = H_{R\_T_{OK}} * puntos\_ref$$

Es decir, el resultado del filtrado son los nuevos puntos en coordenadas del fotograma actual de los puntos que el KLT ha seguido correctamente y la proyección de los puntos en la 'I_R' sobre el fotograma actual de aquellos puntos erróneos.

Proceso:

1. Convertir las coordenadas de los '*puntos_next*' a la 'I_R'. Dado que el tracking KLT devuelve un emparejamiento entre los '*puntos_ini*' (puntos en el fotograma anterior) y los '*puntos_next*' (en el fotograma actual), se puede calcular una homografía entre 'I_T' y la 'I_R':

$$H_{R\_T} = findHomography(puntos\_next, puntos\_ref)$$

Utilizando la inversa de dicha transformación se pueden proyectar los '*puntos_next*' sobre la 'I_R'. A estos puntos los llamaremos '*puntos_next_inv*'.

2. Calcular la media y varianza de los desplazamientos en X e Y entre los '*puntos_next_inv*' y los '*puntos_ref*'. Dado que se dispone de los '*puntos_ref*' y su emparejamiento con los '*puntos_next*' (y por lo tanto con los '*puntos_next_inv*') dado por el tracking KLT, se pueden calcular las distancias en coordenadas X e Y entre ellos. Con estos datos de calculan media y varianza de los desplazamientos.
3. Filtrado de los puntos cuya confianza sea menor al 95%. Con los datos de media y varianza (y desviación típica) obtenidas, se seleccionarán como puntos válidos aquellos cuyo desplazamiento en X e Y se encuentre dentro del margen de confianza del 95%, y como no válidos aquellos que se encuentren fuera de dicho margen.
4. Recalculo de la homografía $H_{R\_T_{OK}}$. Dado que se han conseguido separar los '*puntos_next_inv*' en dos grupos, válidos y no válidos, es posible refinar el cálculo de la 'H_{R_T}' para obtener una mejor homografía $H_{R\_T_{OK}}$:

$$H_{R\_T_{OK}} = findHomography(puntos\_next\_inv\_validos, puntos\_ref)$$

5. Proyección de los '*puntos_ref*' correspondientes a los '*puntos_next_ko*' al fotograma actual con la nueva $H_{R\_T_{OK}}$:

$$puntos\_next\_ko\_in\_t_1 = H_{R\_T_{OK}} * puntos\_ref$$

Capítulo 3: Comparativa de métodos

En este capítulo se van a describir cada una de las alternativas posibles que se han probado en las diferentes etapas del sistema indicando en cada caso los pros y contras para finalmente argumentar la solución adoptada para cada proceso. Para obtener los datos indicados se utiliza uno cualquier de los videos de test. En este capítulo, no se pretende hacer notar tanto las cifras concretas como el argumentar la solución adoptada de forma cualitativa.

- Etapa de Registro con el primer fotograma.

  Tal como se indica en el proceso de Registro con primer fotograma del apartado 2.2 del Capítulo 2, en esta etapa se extraen una serie de puntos característicos de la 'I_R' y del primer fotograma de la secuencia y se emparejan. En el apartado de Estado del arte se describen diferentes tipos de características y descriptores que pueden ser utilizados en esta tarea.

  Para realizar el emparejamiento robusto de puntos, se ha utilizado el mismo método en todos los casos, es decir y de forma simplificada, un emparejamiento robusto mediante doble emparejamiento, test de simetría y filtrado por ratio y RANSAC. El procedimiento completo se detalla en el apartado de *'matching'* del Anexo 2, obteniendo finalmente la matriz de homografía 'H_{R_T}'.

  Para realizar una comparación entre los distintos métodos de extracción de características, descriptores, sus emparejamientos y el tracking posterior, se selecciona uno cualquiera de los videos, sobre el cual se aplican todos los procesos que se detallan a continuación.

  Los resultados obtenidos con los diferentes tipos de características son los siguientes:

  a. Características SIFT.

     Estos puntos son los que tienen mejor comportamiento de entre los chequeados frente a cambios de iluminación y de escala. El tiempo consumido en la obtención de puntos SIFT es de unos 50-70 ms por cada 100 características. El emparejamiento obtenido utilizando estas características y sus descriptores es excelente. Con los umbrales por defecto este método extrae 1038 puntos en la 'I_R' y 2432 en el primer

fotograma de la secuencia. El emparejamiento de estos puntos por fuerza bruta utilizando como medida de similaridad la norma L2, da como resultado 84 correspondencias bidireccionales, es decir, que, en estos casos, el emparejamiento de los puntos de la 'I_R' y los del primer fotograma coincide con el emparejamiento de los puntos a la inversa (ver Figura 15).

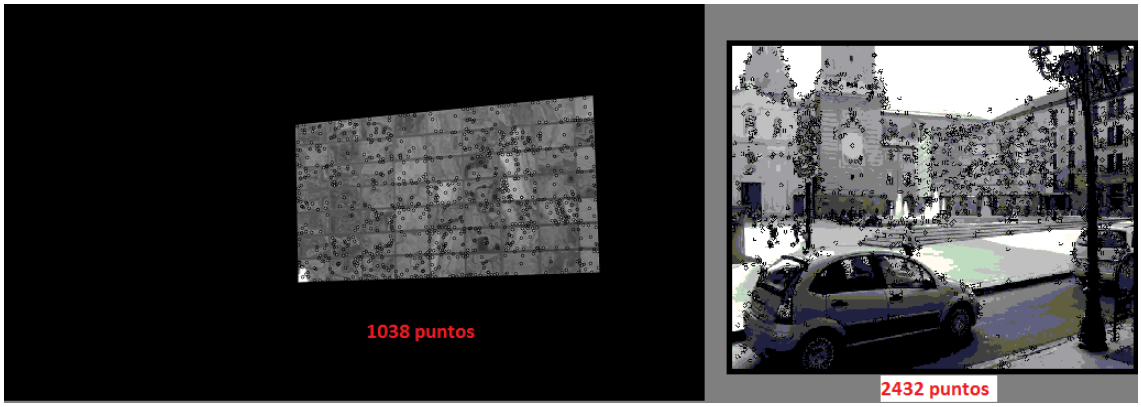


Figura 15: Puntos característicos de tipo SIFT

b. Características SURF.

Estos tipos de puntos no son tan robustos como los SIFT pero son más rápidos de extraer y procesar. El tiempo consumido en la obtención de estos puntos es de unos 40 ms por cada 100 características. El resultado del emparejamiento es bueno. El emparejamiento obtenido utilizando estas características y sus descriptores es excelente. Con los umbrales por defecto este método extrae 1005 puntos en la 'I_R' y 1802 en el primer fotograma de la secuencia. El emparejamiento de estos puntos por fuerza bruta utilizando como medida de similaridad la norma L2, da como resultado 17 correspondencias bidireccionales, es decir, que, en estos casos, el mejor emparejamiento de los puntos de la 'I_R' y los del primer fotograma coincide con el mejor emparejamiento de los puntos a la inversa (ver Figura 16).

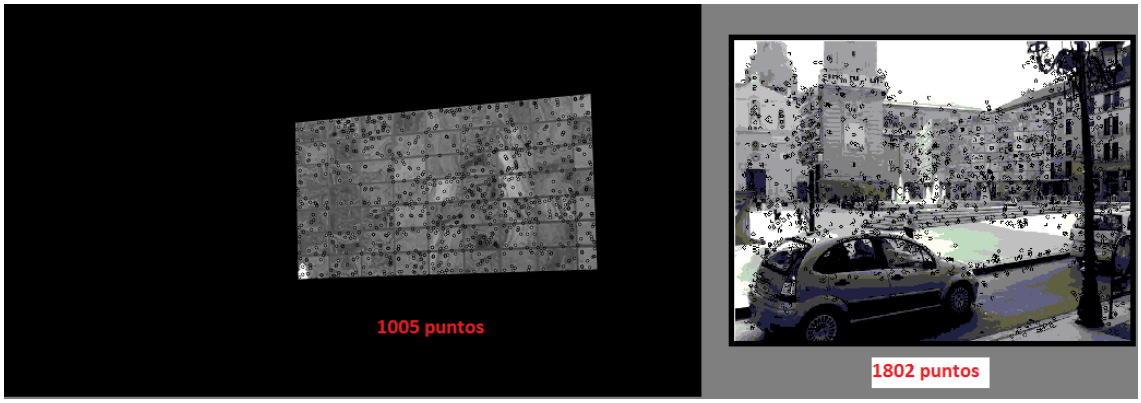


Figura 16: Puntos característicos de tipo SURF

c. Características FAST.

La extracción de estos puntos característicos es la más rápida de las tres. El tiempo que se emplea en la obtención de éstos es de unos 7 ms por cada 100 características. Este tipo de características produce un resultado muy malo cuando se intenta un emparejamiento por fuerza bruta en toda la imagen. Esto es debido a que no son robustos a cambios de escala e iluminación por lo que no son adecuados para la etapa de registro. Con los umbrales por defecto este método extrae 1184 puntos en la 'I_R' y 3999 en el primer fotograma de la secuencia (en uno de los videos de test). Utilizando descriptor de tipo SURF, el emparejamiento de estos puntos por fuerza bruta utilizando como medida de similaridad la norma L2, da como resultado solo 3 correspondencias bidireccionales (ver Figura 17).

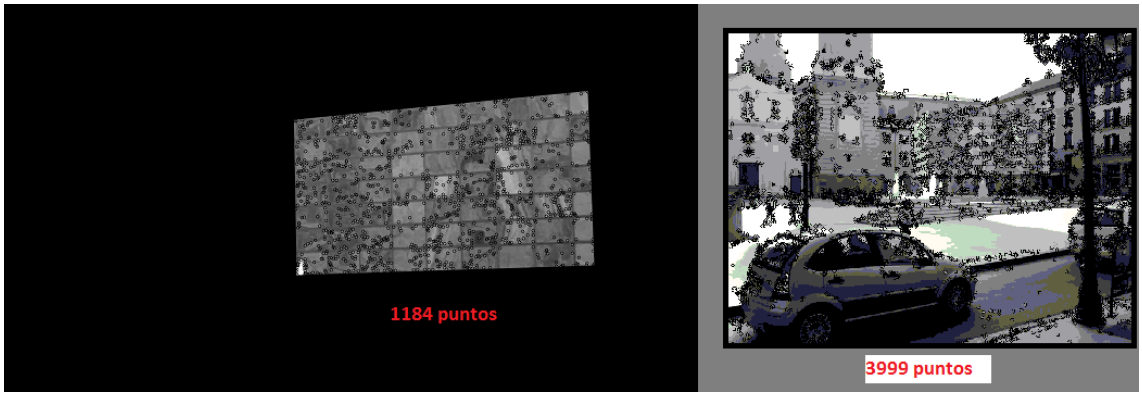


Figura 17: Puntos característicos de tipo FAST

El resultado de los emparejamientos obtenidos con los distintos tipos de puntos característicos anteriores se muestra en la Figura 18:

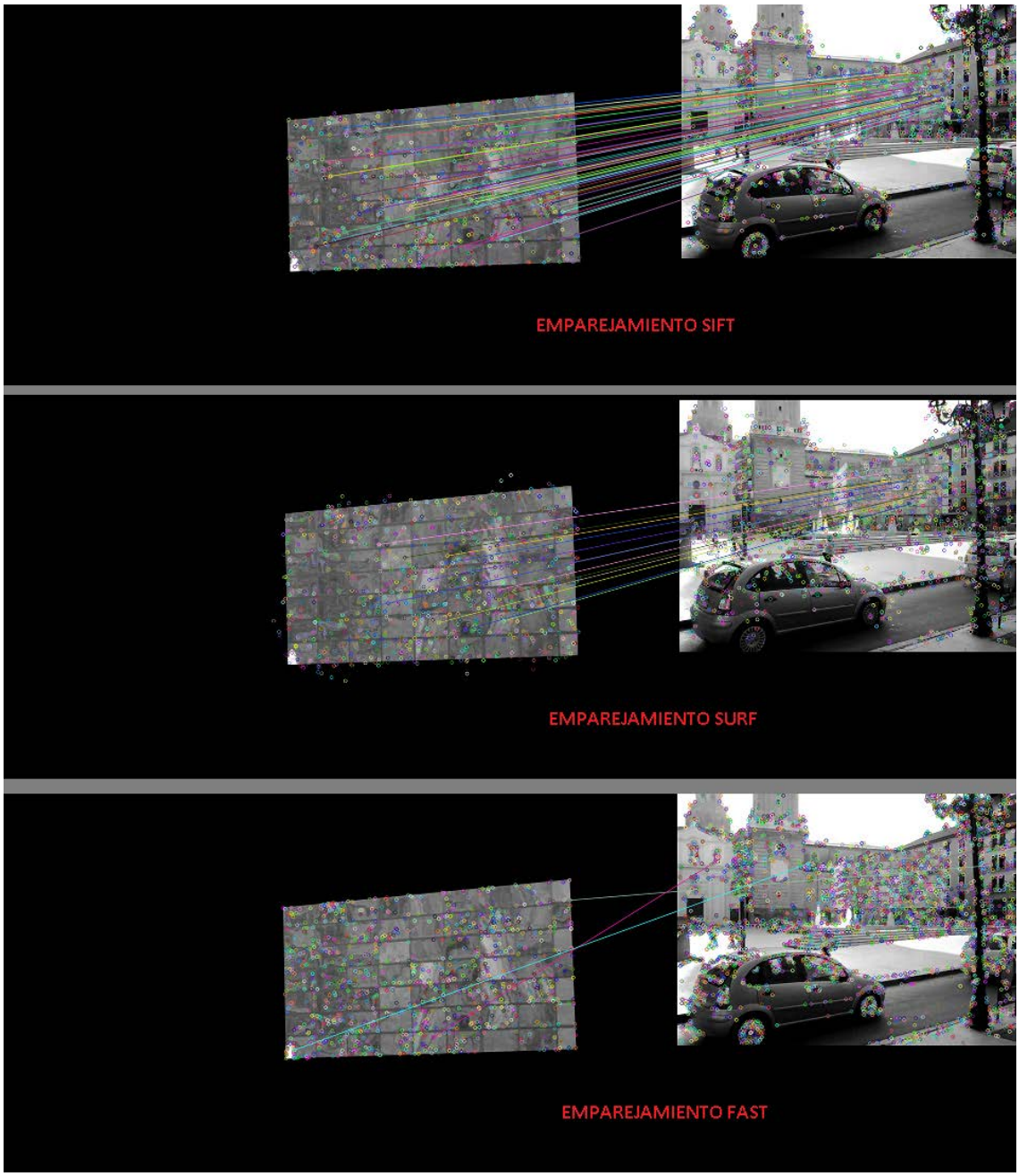


Figura 18: Resultados del emparejamiento con los distintos tipos de características.

A la vista de los resultados podemos extraer las siguientes conclusiones:

- Las características FAST no son adecuadas para la etapa de registro.
- Las características SURF, a pesar de ser más rápidas que las SIFT, tiene un grado de fiabilidad inferior a éstas, dando un volumen de correspondencias limitado cuando se utilizan sobre toda la imagen. Esto conlleva que la fiabilidad en el cálculo de la homografía 'H_{R_0}' sea mejor y en algunos casos pueda ser imposible por falta de puntos detectados.

- Las características SIFT, son las más adecuadas para esta etapa por dos razones:
    - El número de correspondencias obtenidas es elevado incluso entre imágenes de distintas características y casi un orden de magnitud mayor que las obtenidas con SURF.
    - La etapa de registro inicial no tiene una restricción de tiempo tan importante como las etapas siguientes puesto que solamente debe realizarse una vez en la etapa inicial, lo que conlleva que puede invertirse un poco más de tiempo si esto se traduce en un registro más adecuado.

- Etapa de Extracción de la superficie de proyección.

  Una vez seleccionado el método de extracción de características (por las razones argumentadas anteriormente el método seleccionado son las características tipo SIFT) y su emparejamiento, en esta etapa solo es necesario utilizar un proceso de cálculo de la matriz de homografía que relacionará la 'I_R' con el primer fotograma utilizado en el registro. El proceso se describe en el apartado Extraer superficie de proyección del Capítulo 2.

  Con el objetivo de eliminar puntos espurios que puedan afectar al cálculo de esta homografía se aplica un filtro RANSAC para la detección de 'outliers'. En el apartado de *'matching'* del Anexo 2 se detalla como RANSAC solventa la detección de 'outliers' excluyendo estos puntos del cálculo de la homografía.

  Aunque el proceso anterior obtiene la superficie de proyección, es posible que existan elementos en primer plano que oculten parcialmente esta superficie.

  Para que la proyección de la imagen aporte al usuario una sensación visual coherente, es necesario un mecanismo de detección de profundidad. Es decir, se hace necesario poder detectar si algún objeto se encuentra situado en un plano más cercano al observador que la propia superficie de proyección, puesto que, en ese caso, dicho objeto ocluye la visión de la proyección y debe proyectarse por encima de la imagen.

  Por este motivo se incluye un procesamiento adicional que permite detectar estos posibles objetos. El proceso realizado es el siguiente:

  a. Como resultado de los procesos anteriores, se puede calcular una máscara que selecciona la superficie de proyección haciendo uso de las

matrices de homografía necesarias. Se entiende por máscara a una imagen binaria del mismo tamaño que el fotograma actual donde se disponen '1s' en la zona ocupada por la superficie de proyección en este fotograma y '0s' en el resto de la imagen (ver Figura 19).

b. Se calcula una operación 'AND' entra la máscara anterior y el fotograma actual, extrayendo así la imagen de la superficie de proyección (ver Figura 19), a la que llamaremos '*fondo_next*'.

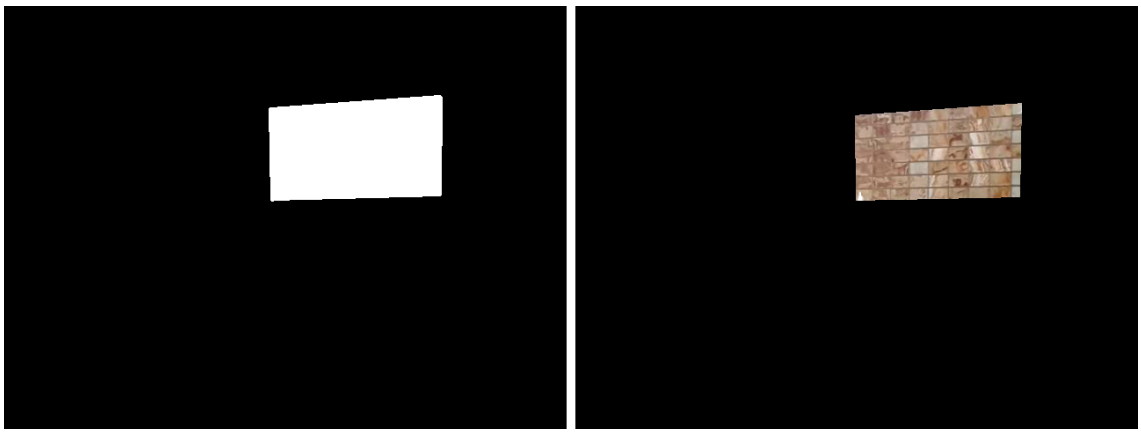


Figura 19: Mascara binaria y superficie de proyección del fotograma actual, respectivamente.

c. Esta imagen de '*fondo_next*', mediante la homografía a la 'I_P' ('H_{P_0}' o 'H_{P_T}' dependiendo del método utilizado) se convierte a este sistema de coordenadas. Con esta conversión obtenemos la imagen de la superficie de proyección en el mismo sistema de coordenadas que la 'I_P'.

d. Se realiza la misma conversión de coordenadas con la imagen '*fondo_ini*' obtenida en la etapa de Extraer superficie de proyección del Capítulo 2. Esta imagen será tomada como fondo limpio con el que comparar los fotogramas de la secuencia. Por este motivo se hace hincapié en que el primer fotograma detectado en la inicialización debe ser un fotograma sin oclusiones.

e. Se realiza una operación de resta en los tres canales de la imagen entre '*fondo_ next*' y '*fondo_ini*', obteniendo '*fondo_dif*'.

$$fondo\_dif = fondo\_ini - fondo\_next$$

f. Se calcula una operación de umbralizado en los tres canales por separado. Mediante esta operación, cualquier valor de '*fondo_dif*' que sea mayor de dicho umbral en cualquiera de los tres canales (en valor absoluto) se detecta como "diferente del fondo" y por lo tanto se clasifica

como objeto en un plano más cercano al observador que la propia superficie de proyección.

g. Para mejorar la detección de estos objetos se practican varias operaciones de erosión y dilatación a *'fondo_dif'* umbralizada que permiten eliminar pequeños elementos ruidosos. Así mismo y con el objetivo de obtener una máscara que permita detectar los objetos que ocluyen la superficie de proyección, se utiliza un algoritmo de detección de contornos exteriores [22] (ver Figura 20).

Figura 20: Contornos de los objetos en primer plano.

h. Se calcula el área total de la superficie de proyección y de los objetos detectados.

i. Si el área de los objetos detectados es superior a un porcentaje determinado del área total (se ha tomado un 75% del total) durante una

serie de fotogramas consecutivos (por defecto 5), se entiende que existe una oclusión grave y el sistema dará una advertencia y procederá a reiniciarse.

- Etapa de Tracking.

  El objetivo de esta etapa es obtener un método para seguir los puntos característicos, obtenidos mediante emparejamiento en la Etapa de Registro con el primer fotograma, de un fotograma a otro del video.

  Para este cometido se plantean dos alternativas:

  - Utilizar un método de 'tracking' de características. Esta opción consiste en utilizar alguno de los métodos de seguimiento de características entre un fotograma y el siguiente de entre los que ya existen o bien ideado para el caso concreto. En esta alternativa se incluyen los métodos de tracking KLT, filtro de Kalman, BRIEF, ORB y FFME.
  - Utilizar un método de 'emparejamiento de características' entre un fotograma y el siguiente, o entre la 'I_R' y el fotograma actual. Este sistema sería similar a realizar emparejamiento continuo de un cierto número y tipo de características entre dos imágenes. Se han explorado los métodos de FAST + SSD y un método basado en características SIFT.

  Hay que tener en cuenta que los métodos de tracking que se explicarán a continuación, realizan el seguimiento de los puntos de un fotograma en el instante t al siguiente instante $t + 1$. Por lo tanto es necesaria una modificación del sistema que permita relacionar la 'I_P' con el fotograma del instante $t + 1$.

  Este proceso adicional necesario permite extraer una relación homográfica, 'H_{T_T+1}', entre dos fotogramas consecutivos (y por acumulación entre el fotograma 0 y el t+1, 'H_{0_T+1}') que, compuesta con la anterior 'H_{P_0}', nos relaciona la 'I_P' con el fotograma actual como se muestra en la Figura 21:

$$H_{P\_T+1} = \prod_{i=0}^{t+1} H_{T\_T+1_i} * H_{R\_0} * H_{P\_R}$$

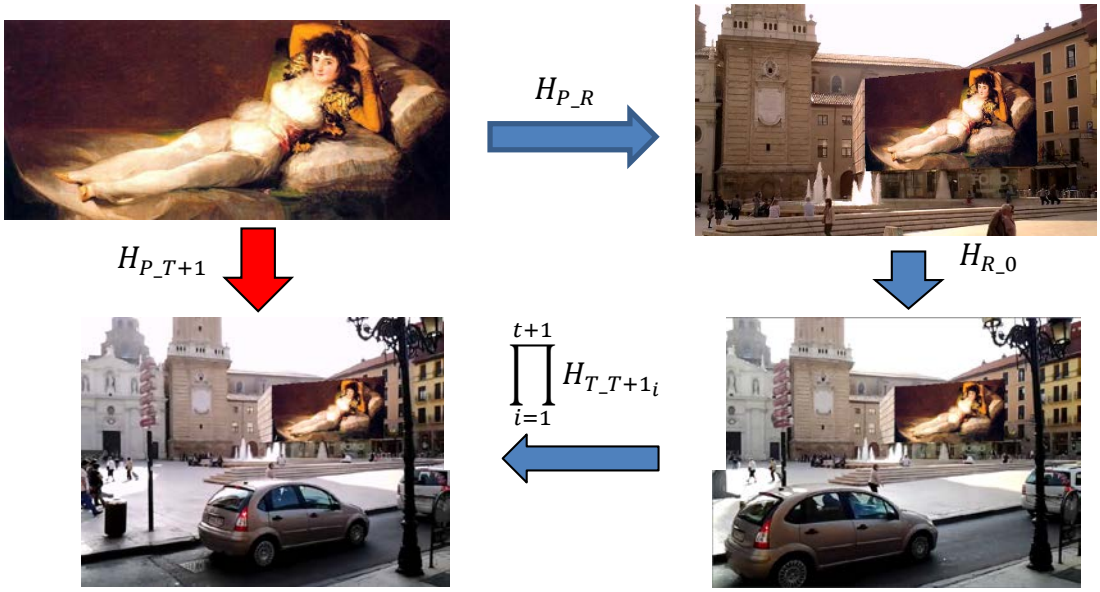


Figura 21: Proyección de la imagen sobre cualquier fotograma de la secuencia por medio del tracking de puntos

El cálculo de esta nueva relación homográfica, 'H_{T_T+1}', también es sometido al proceso de filtrado mediante RANSAC lo cual permite filtrar puntos espurios.

Métodos de tracking

1) Tracking KLT [15] [16]. Algoritmo de seguimiento Lucas-Kanade implementado en OpenCV. Este método es muy rápido y da buenos resultados en ausencia de oclusiones. El tracking lleva unos 5 ms por fotograma para imágenes de 640x480 y 68 puntos característicos.

   Dado que el sistema tiene dificultades con las oclusiones, se realiza una modificación del sistema de tracking. Llamaremos a este método de tracking de KLT modificado como '*REETIQ*'. En este ejemplo, se ha aplicado el tracking KLT como en el ejemplo anterior, con la diferencia de que, cada cierto número de fotogramas (5 en este caso), se aplica una técnica de registro completa, igual que se hace con la imagen inicial, es decir:

   - Se hace un primer registro de la 'I_R' con el primer fotograma, obteniéndose los puntos '*puntos_ref*' y '*puntos_ini*'.
   - A estos puntos '*puntos_ini*' se les aplica el algoritmo de tracking durante 5 fotogramas.

- En los fotogramas múltiplos de 10, se realiza el reemparejamiento. Para ello, se extraen características SIFT (en el ejemplo aunque también pueden ser FAST, SURF o de cualquier otro tipo) del fotograma actual y se realiza un emparejamiento robusto con los '*puntos_ref*' obtenidos en la 'I_R'.
- Se continúa el tracking con los nuevos puntos corregidos. El resultado se observa en la Figura 22.

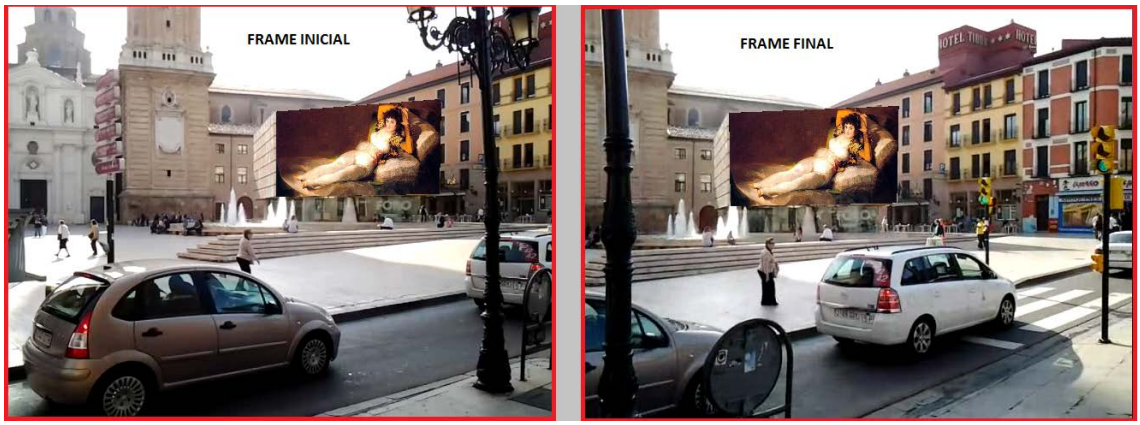


Figura 22: Algoritmo de Tracking KLT frente a oclusiones cuando se realiza un reemparejamiento cada 10 fotogramas mediante SIFT

Esta alternativa da unos resultados excelentes frente a oclusiones de tipo leve, pero tiene la enorme desventaja de que el coste computacional del reemparejamiento en el fotograma correspondiente asciende a 1.2 segundos para imágenes de 640x480 y extracción de características de tipo SIFT. De hecho si se realiza un emparejamiento completo en todos los fotogramas el resultado es casi perfecto. El problema fundamental es el coste computacional tan elevado.

Una mejora a esta última aproximación que permite reducir el tiempo de cómputo sustancialmente, consiste en restringir la búsqueda de las características SIFT a una zona en torno a la región de interés (donde se está realizando la proyección) en los fotogramas correspondientes al reemparejamiento. Para conseguir esto, se extrae una máscara de la región de proyección obtenida en el fotograma anterior y se amplía en todas las direcciones con un margen de seguridad que puede calcularse, por ejemplo, en función de alguna medida del desplazamiento de los puntos. En este

ejemplo de la Figura 23 el tiempo de reemparejamiento desciende a unos 145 ms.

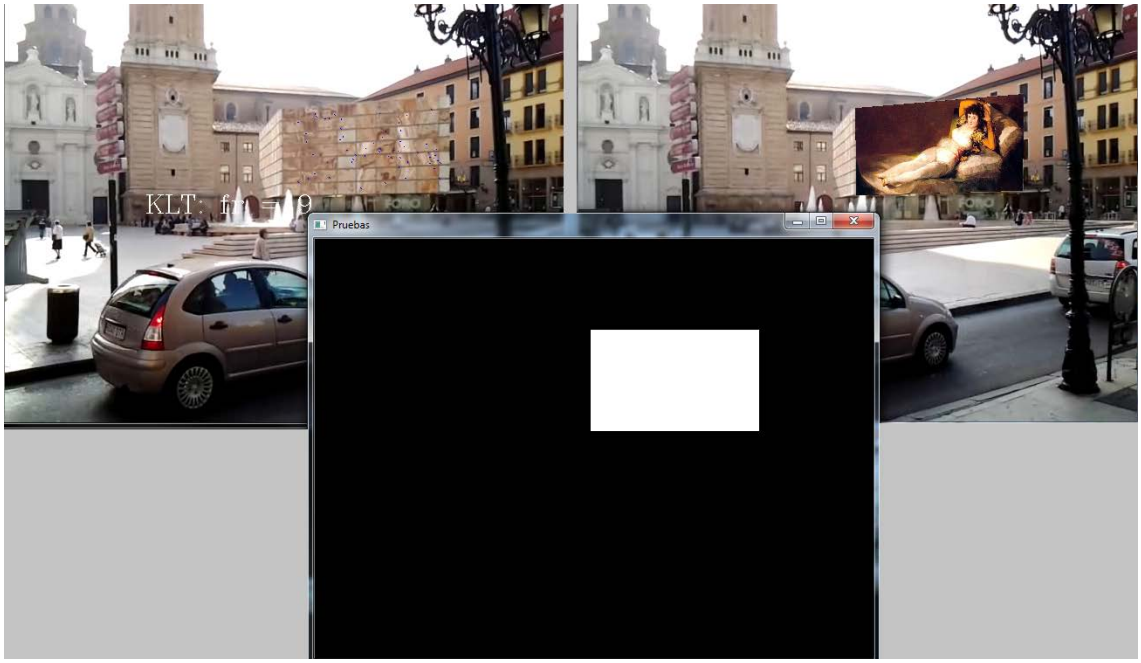


Figura 23: Algoritmo de Tracking KLT frente a oclusiones cuando se realiza un reemparejamiento cada 10 fotogramas mediante SIFT y máscara de búsqueda en la ROI del fotograma anterior

2) Filtro de Kalman [17] [18]. Se ha valorado la utilización de un filtro de Kalman como método de tracking del sistema. Estudiando el proceso (ver el apartado Algoritmo de tracking de Kalman del Anexo 3), se puede observar que el filtro de Kalman necesita en cada uno de los instante, una serie de datos de medidas para poder alimentar el sistema y calcular su estado. Dado que no disponemos de medidas, se han intentado utilizar los resultados que proporciona un tracking KLT para simular dichas medidas. El resultado no solventa el problema de las oclusiones dado que las posiciones de los puntos característicos son arrastradas por los objetos que ocluyen la superficie de proyección de igual forma que ocurre al KLT por separado. El resultado es que, este arrastre modifica poco a poco la posición, lo cual se refleja en una modificación de la predicción de la dinámica lineal, y en definitiva, supone el arrastre de la posición estimada por el filtro de Kalman. Por lo tanto, la conclusión es que no es posible su utilización en este sistema.

3) Tracking BRIEF [9]. Este algoritmo de tracking combina la extracción de características FAST con los descriptores BRIEF. Dado que los descriptores BRIEF son descriptores robustos a cambios de iluminación, distorsión de la perspectiva y desenfoque, pero muy sensibles a rotación el resultado no es demasiado bueno, como puede observarse en la Figura 24, por lo que se ha descartado su uso.

Figura 24: Tracking mediante FAST y descriptores BRIEF. Resultado bastante inestable.

4) Tracking ORB [10]. Este algoritmo de tracking combina la extracción de características FAST con información de orientación (oFAST, descritas en el Anexo 1) con los descriptores rBRIEF descritos en el Anexo 2. El funcionamiento concreto se explica en el apartado Algoritmo de tracking ORB de los Métodos de tracking en el Anexo 3. Se han realizado pruebas con este método de seguimiento pero los resultados incluso sin oclusiones no mejoran los obtenidos por el KLT por lo que se ha desechado (ver Figura 25).

Figura 25: Tracking ORB. Resultados inestables incluso sin oclusiones.

5) Tracking FFME [19]. Es un método de estimación del movimiento robusto al ruido y al problema de la apertura. El resultado de las correspondencias produce un campo vectorial disperso de movimiento que describe con precisión el movimiento de la imagen. No mejora los obtenidos por el KLT en los casos de oclusiones (ver Figura 26) y es significativamente más lento (en torno a 110 ms por fotograma) por lo que se ha desechado.

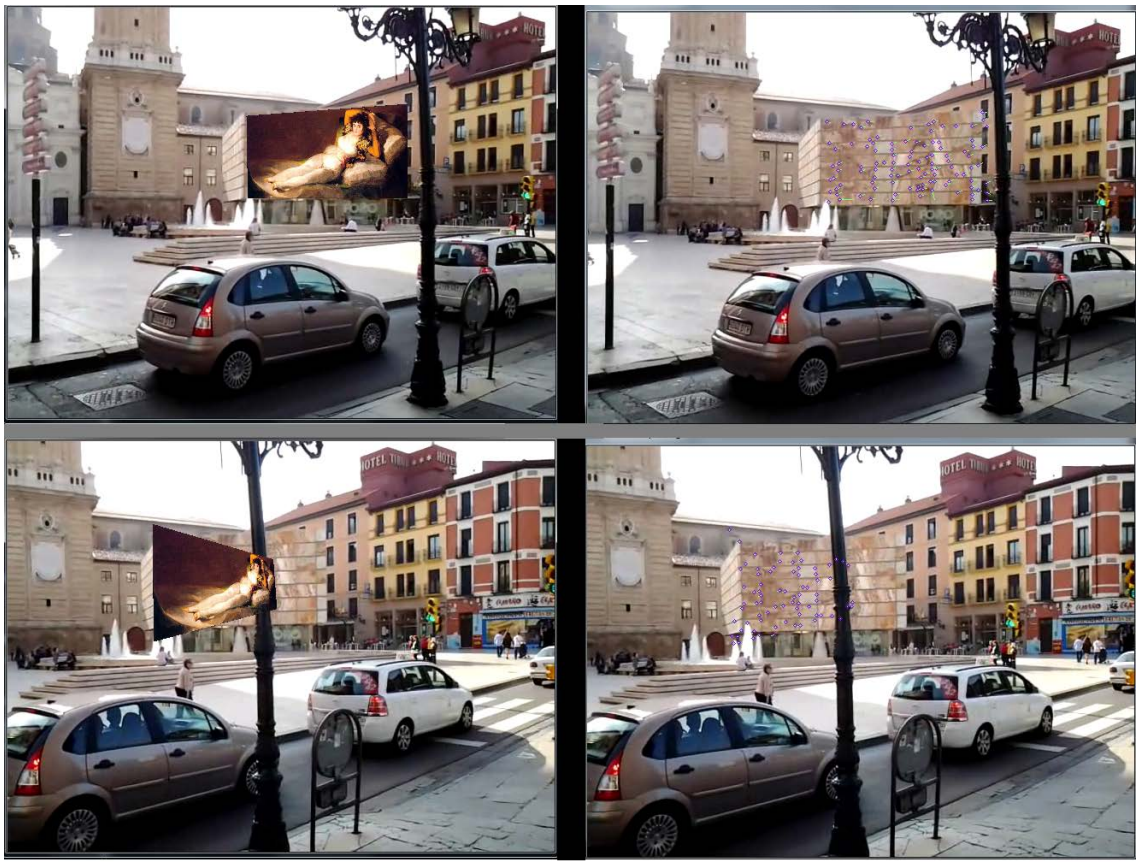


Figura 26: Tracking con método FFME. No supera oclusiones.

Métodos de emparejamiento de características

Se han probado diferentes métodos de tracking con la intención de combinar las ventajas de las diferentes características estudiadas. Los principales métodos probados son dos:

a. **Método de tracking p-SIFT.** Los puntos SIFT y sus descriptores son las características que mejor resultado han dado en el emparejamiento en las escenas que se han evaluado. Sin embargo computacionalmente son demasiado pesados para su utilización en tiempo real. Se ha probado un sistema basado en el cálculo de unos puntos que llamamos p-SIFT pues están inspirados en ellos, pero eliminando parte de los cálculos más pesados. El cálculo exacto de estos puntos y sus descriptores se detalla en el Anexo 1. A estos p-SIFT se les aplica un método de predicción de medida y posterior búsqueda por 'Hill climbing' entre fotogramas de la secuencia. El sistema calcula los p-SIFT de la nueva imagen y predice la posición de los puntos en el nuevo fotograma en base al movimiento de

los N fotogramas anteriores. En torno a los 3x3 vecinos del nuevo punto se busca del pixel de menor distancia al punto p-SIFT en el fotograma anterior. Si no coincide con el pixel predicho se vuelve a calcular la distancia de los 3x3 vecinos del nuevo pixel y así sucesivamente hasta encontrar el mínimo local o superar un máximo de iteraciones. El proceso es similar a '*escalar una montaña*' (en este caso descender pues se busca el punto mínimo) de ahí su nombre. El resultado obtenido es bueno, soportando oclusiones leves (ver Figura 27). Se ha descartado frente al seleccionado debido a que es significativamente más lento, en torno a 170 ms por fotograma.

Figura 27: Seguimiento mediante características p-SIFT. Soporta oclusiones leves.

b. **Método de tracking de puntos FAST y emparejamiento con medida de similaridad SSD**. Este método pretende combinar la ligereza computacional del cálculo de puntos FAST con un sistema más robusto de emparejamiento de puntos que los descriptores basados en FAST. Para este caso se ha pensado utilizar un patch de 5x5 píxeles en torno al punto característico como descriptor. El emparejamiento se hace por fuerza bruta y la medida de dismilaridad utilizada es la SSD, mostrándose el cálculo completo del método en el Anexo 3. El resultado no ha sido demasiado bueno (ver Figura 28) y su implementación es bastante más lenta (1.3 segundos por fotograma) que la conseguida con el sistema propuesto en el Capítulo 2, por lo que se ha descartado.

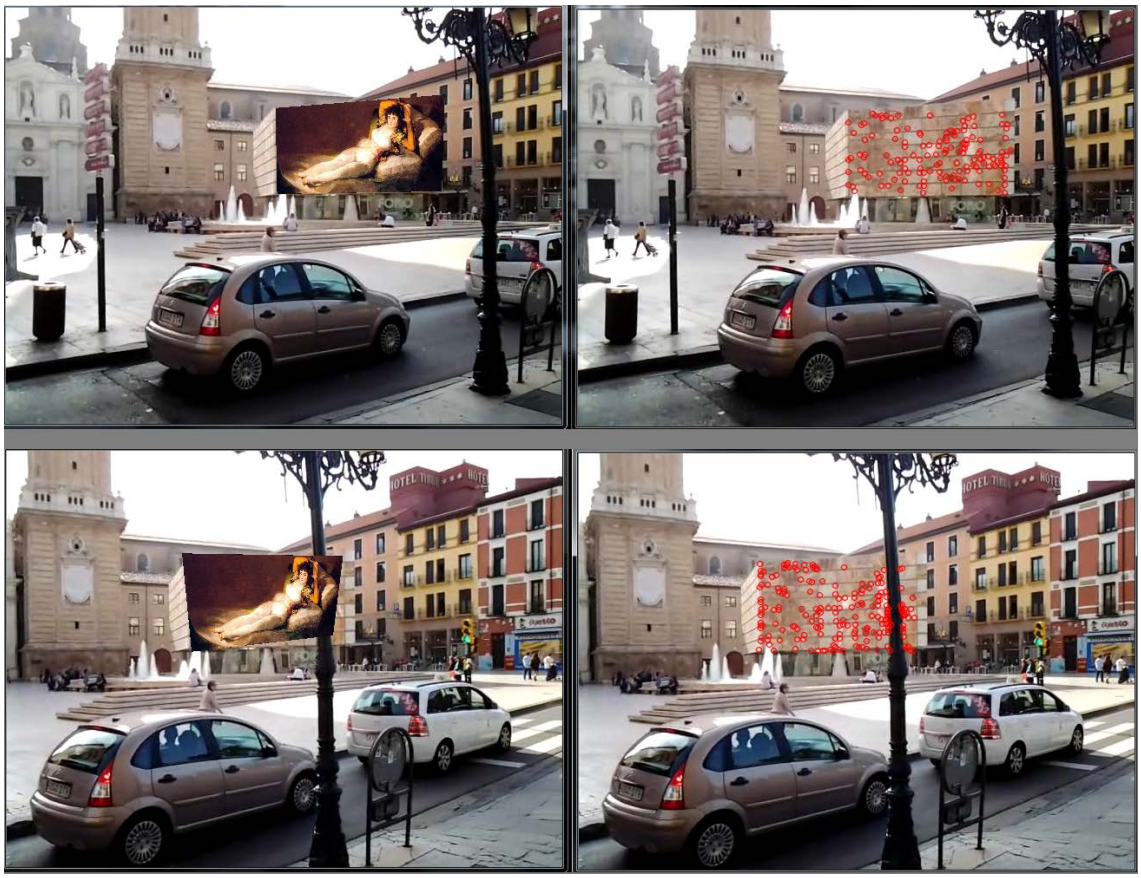


Figura 28: Seguimiento mediante FAST y medida de similaridad SSD. Resultado no soporta oclusiones.

- Etapa de Corrección de errores.

  Como se ha comentado en apartados anteriores, es necesario implementar un sistema de realimentación que permita al sistema de tracking detectar y corregir situaciones en las que existen oclusiones. Algunos de los métodos que se explican a continuación realizan un seguimiento entre fotogramas consecutivos al que denominaremos '*Reproyección Indirecta*', a diferencia del método explicado en el apartado 2.2. Funcionamiento del Sistema del Capítulo 2 al que llamaremos '*Reproyección Directa*'.

  La '*Reproyección Indirecta*' necesita de un paso intermedio para relacionar la 'I_R' con el fotograma actual: Es necesario calcular una homografía que relaciona el fotograma en instantes anteriores con el fotograma actual. Esta homografía formalmente sería la compuesta por:

$$H_{R\_T+1} = \prod_{i=0}^{t+1} H_{T\_T+1_i} * H_{R\_0}$$

donde $\prod_{i=0}^{t+1} H_{T\_T+1_i}$ son las homografías acumuladas desde el fotograma inicial hasta el actual.

Se han probado varios métodos de corrección de errores, todos ellos basados en un filtrado de las posiciones en coordenadas de la 'I_R' (ver Figura 29). El proceso sigue los mismos pasos que los explicados en el apartado Realimentación del tracking y corrección de errores del Capítulo 2. En resumen:

a. Con las posiciones de los puntos en el instante t se calculan las posiciones en t + 1 mediante KLT.
b. Con las posiciones obtenidas y las correspondientes en la 'I_R' (método de '*Reproyección Directa*') o en el fotograma anterior (método de '*Reproyección Indirecta*'), se obtiene la matriz de homografía 'H_{R_T}' o 'H_{R_T+1}' (según sea el método utilizado).
c. Se proyectan las posiciones de los puntos en el instante actual sobre la 'I_R' con dicha homografía.
d. En este sistema de referencia, se compara la posición original de los puntos '*puntos_ref*' con la obtenida de la proyección de los actuales '*puntos_next_inv*'.

Se han probado tres métodos para el filtrado de los puntos:

1. Distancia máxima absoluta. Se establece un umbral máximo. Si la distancia entre un '*puntos_ref*' y su correspondiente '*puntos_next_inv*' es mayor que este umbral, se clasifica este punto como no válido. Este método tiene el problema de que necesita establecer un umbral "ad hoc", y que a priori no puede ser intuido dado, entre otras cosas, a que depende de la velocidad de desplazamiento del observador o incluso de la velocidad de procesamiento, ya que si es necesario no procesar algún fotograma por mantener la sensación de tiempo real, el desplazamiento entre puntos es diferente. A este método de seguimiento de KLT con reproyección y filtrado por distancia máxima es al que llamamos '*REPROY*'.
2. Distancia máxima como función de la distancia media. Se establece un factor umbral máximo, '*dist_th_factor*'. Se calcula la distancia media de todos los '*puntos_ref*' y sus correspondientes '*puntos_next_inv*', '*dist_mean*'. Se clasifican como puntos no válidos a todos aquellos cuya distancia a su

correspondiente '*puntos_ref*' este por encima de la '*dist_mean*' * '*dist_th_factor*':

$$distance(x, y; t) = \begin{cases} > dist\_th\_factor * dist\_mean; & Punto\ no\ válido \\ < dist\_th\_factor * dist\_mean; & Punto\ válido \end{cases}$$

Este método, igual que el anterior, requiere un umbral "ad hoc", pero tiene la ventaja de que no es tan dependiente de factores externos como el anterior.

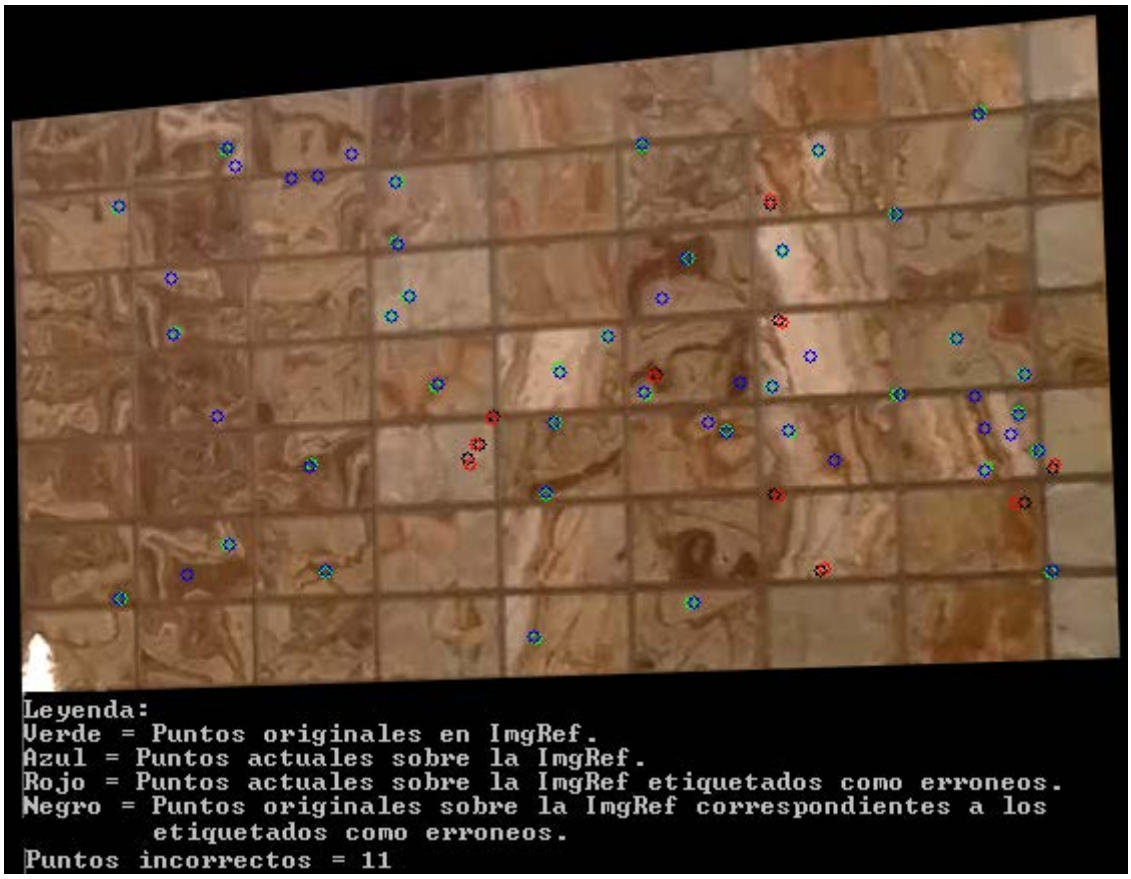


Figura 29: Algoritmo de Tracking KLT frente a oclusiones cuando se realiza una validación por reproyección de las posiciones sobre la '*Ir*' cada 10 fotogramas y un filtrado por distancia media.

Con este método, el tracking KLT es capaz de superar oclusiones leves con un tiempo de procesado aceptable (~45 ms en los fotogramas en los que se ejecuta la corrección). El inconveniente principal es que, durante la proyección del contenido, en alguno de los pasos intermedios se producen vibraciones esporádicas visualmente incómodas (ver Figura 30).

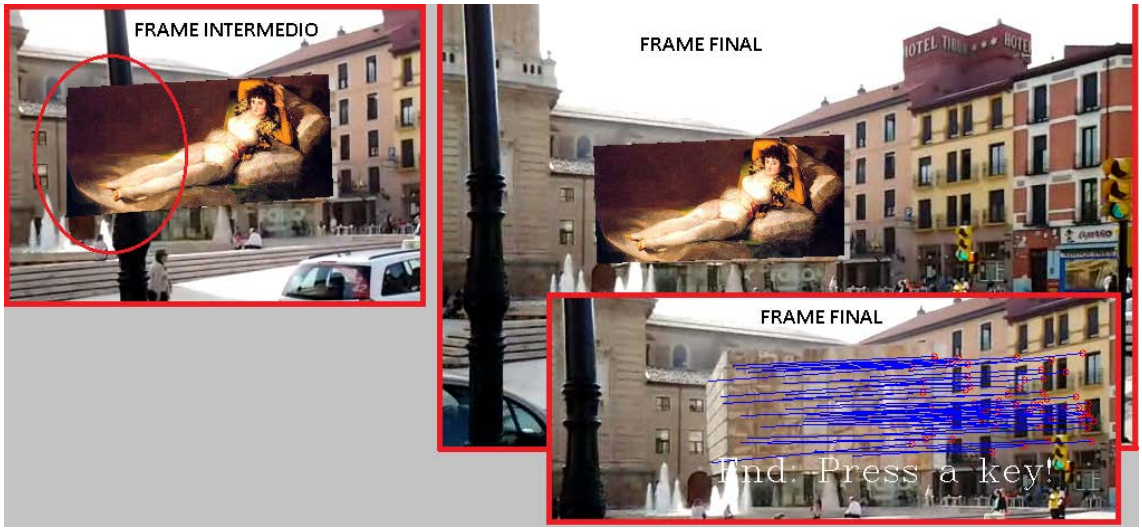


Figura 30: Vibraciones de la imagen. Tracking KLT frente a oclusiones con validación por reproyección de las posiciones sobre la '*Ir*' y filtrado por distancia media.

3. Distancia máxima como función de la varianza de las distancias. Este es el filtro explicado en apartado Realimentación del tracking y corrección de errores del Capítulo 2: Sistema objetivo del estudio y el seleccionado como el mejor de ellos. Obtiene los mejores resultados visuales y es capaz de gestionar oclusiones leves con tiempos de ejecución razonables, ≈ 45 ms por fotograma ó ≈ 60ms en el fotograma concreto en que se ejecuta el filtrado. A este método de seguimiento de KLT con reproyección y filtrado basado en la confianza de la medida del desplazamiento es al que llamamos '*REPROY+FILTRO*'.

Cualquiera de los tres métodos anteriores, permiten clasificar los puntos obtenidos en válidos y no válidos. Esta clasificación nos permite utilizar los puntos obtenido como válidos para refinar la homografía que nos ha facilitado el tracking, '$H^*_{R\_T+1}$', y corregir la posición de los puntos no válidos. La posición de estos para el siguiente fotograma, será la obtenida por la proyección de sus correspondientes 'puntos_ref' con la nueva homografía refinada. Esto nos proporciona una posición para todos los puntos, la obtenida por el KLT en el caso de los puntos válidos ('puntos_next') y la obtenida por la proyección de los correspondientes 'puntos_ref' con la nueva homografía en los puntos no válidos.

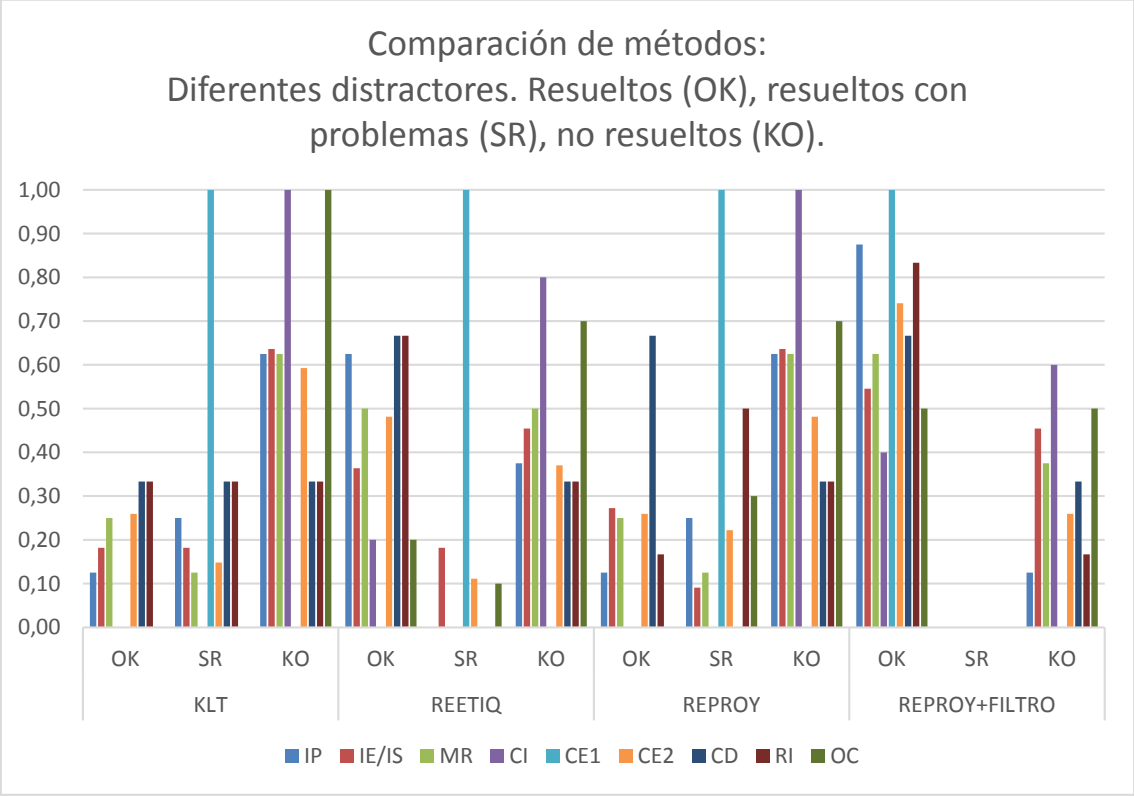
$$\text{puntos\_next}_{\text{corregidos}} = H^*_{R\_T+1} * \text{puntos\_ref}_{\text{puntos\_next\_inv}_{\text{no validos}}}$$

Capítulo 4: Conclusiones y resultados

Las pruebas realizadas han tomado como secuencias de fotogramas los videos grabados por varios dispositivos móviles con diferentes características. Estos videos han sido adquiridos pensando en proporcionar al sistema entornos reales con diferentes distractores que el sistema real encontrará en un escenario real.

La lista de videos y los distractores incluidos en cada uno de ellos se detalla en el Anexo 4. En este mismo anexo se explican los 4 métodos utilizados en la comparación (KLT, REETIQ, REPROY y REPROY+FILTRO) y las categorías de clasificación de resultados en el procesado de cada uno de estos videos. Como resumen diremos que IP, IE/IS, MR, CI, CE1, CE2, CD, RI y OC son los tipos de distractores que aparecen en los videos, mientras que OK (resultado correcto), SR (sistema se recupera de pérdidas momentáneas) y KO (resultado de pérdida total del seguimiento) son las tres categorías de resultados. Para realizar la clasificación en las diferentes categorías, un usuario debe visualizar y comprobar de forma manual los videos de los resultados.

En la Gráfica 1 se indican los resultados globales obtenidos:

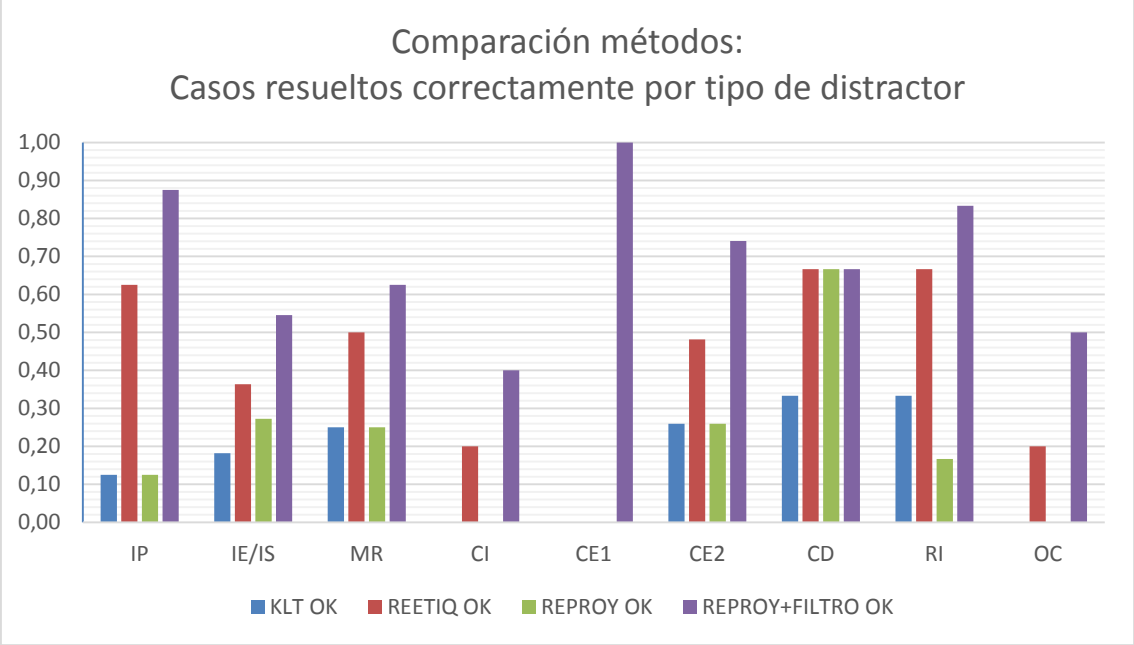


Gráfica 1: Resultados globales obtenidos por métodos y distractores.

En esta gráfica se observan en ordenadas el porcentaje en tanto por 1 de los resultados, mientras que en abscisas se han representado los resultados por métodos y tipo de resultado. En esta gráfica se extrae una visión muy general, de cuyos resultados se observa:

- El grupo de resultados OK en los métodos KLT y REPROY, son bastante bajos con la pequeña excepción del caso del distractor CD (cambio de dirección) en el método REPROY.
- El grupo de resultados OK en el método REPROY+FILTRO, son bastante mejores incluso que con el método de REETIQ, alcanzando resultados del 100% con distractores como el CE1 (Cambio de escala durante el video).
- El distractor CI (cambio de iluminación) resulta extremadamente crítico en todos los sistemas, provocando el fracaso del método hasta en el 100% de los casos en los sistemas KLT y REPROY.
- El distractor OC (oclusiones) resulta también crítico en todos los sistemas, alcanzando fracasos de hasta en el 100% de los casos en el sistema KLT.

Para obtener un mejor detalle de los resultados, se observa la Gráfica 2. En ella se representan los resultados de los casos que se han completado con éxito para comparar por métodos y tipos de distractores cuales son las mejores alternativas:



Gráfica 2: Resultados correctos obtenidos por métodos y tipo de distractor.

Las conclusiones que se observan de esta gráfica son varias:

- El método REPROY+FILTRO supera a todos los demás (incluido el método de REETIQ) frente a todos los distractores, obteniendo un 100% en algunos casos.
- Frente a distractores tan críticos como OC o CI, el comportamiento del método REPROY+FILTRO es relativamente bueno dado que ningún otro a excepción del REETIQ obtiene ningún resultado.
- Comparando por distractores, se observa que el método REPROY+FILTRO es hasta 2.5 veces mejor que el segundo método más efectivo (REETIQ) frente a distractores como el OC o hasta 7 veces mejor que el peor de los métodos en distractores como el IP. El método REPROY+FILTRO resulta 1.55 veces mejor de media que el segundo mejor método (REETIQ) y 3.73 veces mejor que el peor de los métodos.

Como conclusiones a este trabajo se pueden extraer las siguientes:

1) Se ha obtenido un sistema capaz de realizar una inicialización robusta. Este objetivo se ha satisfecho en sus dos aspectos fundamentales:
   a) Una inicialización robusta requiere que el sistema sea capaz de reconocer el entorno sabiendo situarse en él y lanzando su ejecución cuando se confirma que la visualización de la superficie objetivo de la proyección es suficiente.
   b) Inherente al anterior, el reconocer el entorno implica que el sistema es suficientemente estable frente a cambios de perspectiva, escalado, iluminación y condiciones del entorno. Nuestro sistema también cumple con este requisito.
2) Se ha conseguido un tracking consistente que realiza la proyección de una imagen sobre la superficie seleccionada de una forma continua y sin vibraciones visualmente molestas.
3) Nuestro procesado sigue siendo estable ante distractores importantes como cambios de iluminación, rotaciones, cambios de escala o desplazamientos propios de un sistema móvil producidos por el propio movimiento del usuario.
4) Se ha obtenido un sistema de realimentación que permite gestionar las oclusiones de forma que las oclusiones leves son superadas con éxito mientras que las oclusiones graves son detectadas. Es en estos casos cuando el sistema alerta al usuario y permanece a la espera de que dicha oclusión concluya con el fin de reiniciar el seguimiento.

5) Se ha implementado un procesado que permite detectar razonablemente los objetos que se encuentran en primer plano, permitiendo superponerlos a la proyección facilitando al usuario una visión más realista y con sensación de profundidad.
6) El tiempo de procesamiento del sistema está en torno a los 100 ms por fotograma, por lo que se pueden calcular hasta 10 fotogramas por segundo. Pese a que los sistemas de tiempo real en visión consideran 'tiempo real' en torno a los 25 fotogramas por segundo, la tasa obtenida es suficiente para que el observador tenga una buena sensación visual.

Como conclusión final, creemos haber conseguido los objetivos marcados inicialmente, aportando varias alternativas en cada una de las tareas intermedias, comparando con un gran número de opciones y seleccionado la más adecuadas para la resolución del problema.

Capítulo 5: Trabajo futuro

Como se ha comentado al principio de este trabajo, el objetivo final era realizar una aplicación que pueda ser ejecutada en una plataforma móvil. Para cumplir este objetivo, y conseguir una sensación de tiempo real junto con una visión coherente de la escena, todavía fatal varios trabajos que realizar.

1. Los algoritmos deben ser revisados con el fin de optimizar su ejecución y comprobar si existen formas de acelerar o paralelizar algunos procesos. Sería posible lanzar varios hilos de ejecución para realizar el filtrado de los puntos por reproyección a la 'I_R' en uno de ellos, mientras se prepara la detección de objetos de primer plano en otro, por ejemplo. Los dispositivos móviles actuales disponen de varios núcleos, con lo que se podría investigar esta posibilidad.
2. El proceso de registro de la imagen 'I_R' con la imagen inicial 'I_0' mediante emparejamiento de características SIFT es un proceso lento, como se ha explicado en el apartado Registro con primer fotograma del Capítulo 2. En este mismo apartado se explica el proceso paralelo necesario para evitar compensar los desplazamientos que sufra el dispositivo móvil mientras concluye el proceso de registro con el primer fotograma. Este proceso paralelo no se encuentra actualmente implementado pero es un método utilizado en otros trabajos por lo que pensamos que es perfectamente aplicable a nuestro trabajo [23].
3. Será necesario comprobar que el sistema funciona en otros escenarios de forma similar sin modificaciones considerables, o estudiar el tipo de dificultades que añaden escenarios diferentes. Creemos que el escenario elegido tiene una dificultad adicional debido a que los colores y texturas del Cubo de la Seo son muy similares al entorno en que se encuentra, por lo que pensamos que nuestro sistema se comportará bien en la mayor parte de los escenarios.
4. Es necesario mejorar el sistema de detección de objetos de primer plano, dado que, actualmente, al no existir un mecanismo de actualización del fondo inicial, se detecta demasiado ruido y la imagen proyectada se degrada más de lo esperado. Se trataría de algún algoritmo que permita que el ruido debido a pequeños cambios de iluminación o movimiento de la cámara se incorporen a la imagen de fondo para que no sean detectados como objetos. En el CvLab hay trabajos extensos sobre el tema por lo que es posible estudiar la mejor alternativa para ello.

5. Se debe realizar una implementación sobre diferentes plataformas móviles y estudiar su comportamiento en ellas, tanto algorítmicamente (no sabemos si los mismos algoritmos que en un ordenador están funcionando correctamente también lo harán sobre Android o iOS) como en rendimiento (no es posible extrapolar el rendimiento de un *Smartphone* o una *Tablet* en relación a un ordenador), aunque el avance tecnológico tanto en hardware como en software que los diferentes dispositivos están experimentando, hace suponer que las posibles restricciones actuales serán solventadas en un futuro cercano.

El proyecto TAMA continua por lo que este Proyecto Fin de Carrera será comprobado en el ámbito de un demostrador real.

Bibliografía

[1] D. G. Lowe, «Object Recognition from Local Scale-Invariant Features,» de *Proceedings of the International Conference on Computer Vision, ICCV '99*, Washington, DC, USA, 1999.

[2] D. G. Lowe, «Distinctive Image Features from Scale-Invariant Keypoints,» *Int. J. Comput. Vision*, vol. 60, nº 2, pp. 91--110, 2004.

[3] H. a. E. A. a. T. T. a. V. G. L. Bay, «Speeded-Up Robust Features (SURF),» *Comput. Vis. Image Underst.*, vol. 110, nº 3, pp. 346--359, June, 2008.

[4] H. Bay, T. Tuytelaars y L. V. Gool, «Surf: Speeded up robust features,» de *In ECCV*, 2006.

[5] A. Haar, Zur Theorie der orthogonalen Funktionensysteme, Georg-August-Universitat, Gottingen., 1909.

[6] P. Porwik y L. Agnieszka, «The New Graphic Description of the Haar Wavelet Transform.,» de *International Conference on Computational Science*, 2004.

[7] E. Rosten y T. Drummond, «Fusing Points and Lines for High Performance Tracking,» de *IN International Conference on Computer Vision*, 2005.

[8] E. Rosten, R. Porter y T. Drummond, «Faster and Better: A Machine Learning Approach to Corner Detection,» *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, nº 1, pp. 105-119, jan. 2010.

[9] M. Calonder, V. Lepetit, C. Strecha y P. Fua, «BRIEF: binary robust independent elementary features,» de *Proceedings of the 11th European conference on Computer vision: Part IV*, Berlin, Heidelberg, 2010.

[10] E. Rublee, V. Rabaud, K. Konolige y G. R. Bradski, «ORB: An efficient alternative to SIFT or SURF.,» de *International Conference of Computer Vision*, 2011.

[11] J. Shi y C. Tomasi, «Good Features to Track,» de *IEEE Conference on Computer Vision and Pattern Recognotion*, Seattle, 1994.

[12] K. D. Derpanis, «Overview of the RANSAC Algorithm,» York University, Toronto, Canada, 2010.

[13] M. A. Fischler y R. C. Bolles, «Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography,» *Commun. ACM*, vol. 24, nº 6, pp. 381-395, 1981.

[14] R. Zhang, «Automatic Computation of a Homography by RANSAC Algorithm,» ECE661 Computer Vision, 2008.

[15] B. D. Lucas y T. Kanade, «An Iterative Image Registration Technique with an Application to Stereo Vision.,» de *International Joint Conference on Artificial Intelligence*, 1981.

[16] J.-Y. Bouguet, *Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm*, Intel Corporation Microprocessor Research Labs, 2000.

[17] R. E. Kalman, «A New Approach to Linear Filtering and Prediction Problems,» *Journal of Basic Engineering*, vol. 82, pp. 35-45, 1960.

[18] G. Welch y G. Bishop, «An Introduction to the Kalman Filter,» University of

North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995.

[19] C. R. del Blanco, F. Jaureguizar, L. Salgado y N. García, «Motion estimation through efficient matching of a reduced number of reliable singular points,» de *SPIE Real-Time Image Processing 2008*, San Jose, CA, 2008.

[20] G. Bradski, «The OpenCV Library,» *Dr. Dobb's Journal of Software Tools*, 2000.

[21] G. Bradski y A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, 1st ed., O'Reilly Media, 2008.

[22] S. Suzuki y K. Abe, «Topological structural analysis of digitized binary images by border following.,» *Computer Vision, Graphics, and Image Processing*, vol. 30, nº 1, pp. 32-46, 1985.

[23] S. Bae, A. Agarwala y F. Durand, «Computational Rephotography,» *ACM Trans. Graph.*, vol. 29, nº 3, pp. 24:1--24:15, #jul# 2010.

[24] C. Harris y M. Stephens, «A Combined Corner and Edge Detector,» de *Proceedings of the 4th Alvey Vision Conference*, 1988.

[25] P. Viola y M. Jones, «Rapid object detection using a boosted cascade of simple features,» 2001.

[26] J. E. Bresenham, «Algorithm for Computer Control of a Digital Plotter.,» *IBM Systems Journal*, vol. 4, nº 1, pp. 25-30, 1965.

[27] M. Muja y D. G. Lowe, «Fast approximate nearest neighbors with automatic algorithm configuration,» de *In VISAPP International Conference on Computer Vision Theory and Applications*, 2009.

[28] «Sistema y Procedimiento de Observación Ampliada». 2007.

[29] A. Gionis, P. Indyk y R. Motwani, «Similarity Search in High Dimensions via Hashing,» de *Proceedings of the 25th International Conference on Very Large Data Bases*, San Francisco, CA, USA, 1999.

[30] Q. Lv, W. Josephson, Z. Wang, M. Charikar y K. Li, «Multi-probe LSH: Efficient Indexing for High-dimensional Similarity Search,» de *Proceedings of the 33rd International Conference on Very Large Data Bases*, 2007.

[31] H. Voorhees y T. Poggio, «Detecting Textons and Texture Boundaries in Natural Images.,» 1987.

[32] P. L. Rosin, «Edges: saliency measures and automatic thresholding.,» *Mach. Vis. Appl.*, vol. 9, nº 4, pp. 139-159, 1997.

[33] G. Klein y D. Murray, «Parallel Tracking and Mapping for Small {AR} Workspaces,» de *Proc. Sixth {IEEE} and {ACM} International Symposium on Mixed and Augmented Reality {(ISMAR'07)}*, Nara, Japan, 2007.

Tabla de Figuras

Tabla de Gráficas

Anexos

Anexo 1: Tipos de Características

La aplicación desarrollada en este trabajo requiere conocer la situación relativa del observador respecto de un escenario mediante las imágenes captadas por un dispositivo móvil que este observador lleve consigo.
Para realizar esta tarea, se necesita hacer corresponder ciertas superficies que aparecen en las imágenes captadas con las superficies de proyección que se han utilizado en el entrenamiento de este escenario. A este proceso de emparejamiento utilizando varias imágenes captadas desde diferentes puntos de vista y con diferentes condiciones, se le denomina Registro de Imágenes.

Los métodos más estudiados en la bibliografía de registro de imágenes se basan en la extracción y emparejamiento de características muy concretas como por ejemplo esquinas o bordes de objetos o edificios, cambios bruscos de color que indiquen posibles objetos diferentes o características similares, cambios de texturas, etc...

Se ha estudiado varias de las características más importantes de la bibliografía implementando pruebas para evaluar la aportación de cada una de ellas. En este anexo se hace una descripción de su extracción y su representación mediante descriptores, así como las ventajas e inconvenientes de cada una de ellas.

1. **Características SIFT.**
   '*Scale-Invariant Feature Transform*' [1] [2]. Son características muy robustas, invariantes a escala y a rotación. En [2] se proporciona un método robusto de *'matching'* que permite una sustancial solidez frente a distorsiones afines, cambios de punto de vista, ruido y cambios de iluminación.
   El coste de extracción de las características es minimizado mediante una aproximación en cascada donde los filtros y las operaciones más costosas sólo son aplicados a las localizaciones que superan el primer test.
   Las etapas de la extracción de características son las siguientes:
   - **Detección de extremos en el espacio de escalas.** Se busca identificar localizaciones y escalas que puedan ser reconocidas de forma repetible bajo diferentes puntos de vista del objeto. Para esta tarea, se utiliza un

kernel gaussiano dependiente de la escala con el que se obtiene la función "espacio-escala" ('*escale space*') de una imagen, L(x,y,σ). Esta función resulta de la convolución de un kernel gaussian dependiente de la escala, G(x,y,σ) sobre la imagen de entrada, I(x,y):

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

donde * es la operación de convolución en x e y, y:

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

Para la localización eficiente de '*keypoints*' estables en posición y escala, se propone buscar los extremos en la función diferencia de las gaussianas (DoG, '*Difference of Gaussians*') con las que se ha convolucionado la imagen, D(x,y,σ). Las funciones '*DoG*' pueden ser calculadas mediante la diferencia de 2 escalas cercanas entre sí pero separadas por un factor multiplicativo y constante k:

$$\begin{aligned} D(x, y, \sigma) &= \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned}$$

El uso de la función de diferencia de gaussianas con escala lleva implícita una normalización de escala con el factor σ^2 que se requiere para la invariabilidad a escala.

Una forma eficiente de construir la función D(x,y,σ), se muestra en Figura 30.

La imagen inicial es incrementalmente convolucionada con gaussianas que producen imágenes separadas por un factor constante k en el espacio de escalas. Se elige dividir cada octava del espacio de escalas (correspondiente a doblar el valor de σ) en un número entero, s, de intervalos, lo que supone que $k = 2^{\frac{1}{s}}$. Por lo tanto se deben producir s+3 imágenes por cada octava para que al final la detección de los extremos cubra la octava al completo. Experimentalmente se ha comprobado que $s = 3$ da el mejor resultado. Las imágenes de la escala adyacentes se restan para producir la '*DoG*' como se muestran en la parte derecha de la Figura 30.

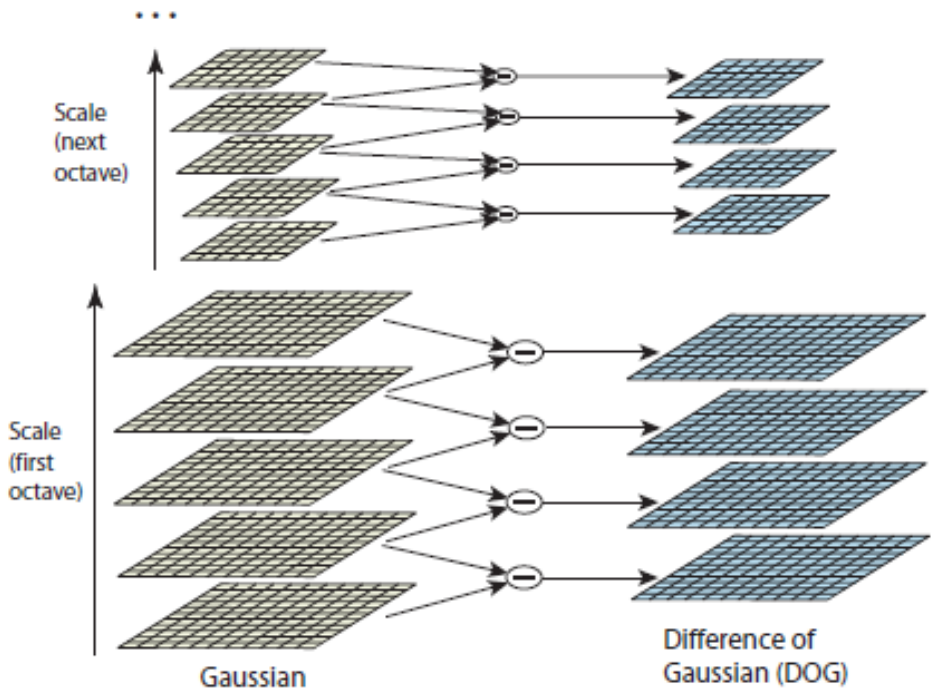


Figura 31: Para cada octava del espacio de escalas, la imagen inicial es repetidamente convolucionada con Gaussianas para producir un conjunto de imágenes en el espacio de escalas como el mostrado a la izquierda. Imagenes convolucionadas adyacentes se sustraen para producir imágenes de diferencia de gaussianas como el mostrado a la derecha. Después, para cada octava, la imagen se muestrea reduciendo en un factor 2 y el proceso se repite.

Una vez que la octava se ha procesado completamente, se submuestrea la imagen de gaussianas para que tenga dos veces el valor inicial de σ, tomando todos los píxeles pares de cada fila y columna.

- **Localización de los *'keypoints'*.** Para detectar los mínimos y máximos locales de D(x,y,σ), cada punto se compara con los 8 vecinos de la propia escala y con los 9 vecinos de las escalas superior e inferior (ver Figura 31).

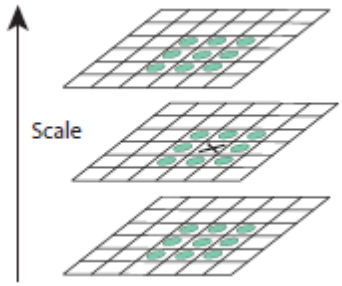


Figura 32: Búsqueda de máximo y mínimo en las imágenes diferencia de gaussianas mediante la comparación del píxel marcado con un X y sus 26 vecinos de la región 3x3 de la imagen actual y las adyacentes en escala superior e inferior marcados con círculos.

Este punto es seleccionado únicamente si es mayor o menor que todos sus vecinos. En [2] se detalla un método para ajustar la localización de los '*keypoints*' mediante el ajuste de una función cuadrática en 3D que descarta los puntos con poco contraste. Para asegurar la estabilidad no es suficiente rechazar estos puntos, también es necesario rechazar aquellos puntos con baja similitud a puntos esquina. Los puntos localizados a lo largo de bordes rectos no son fiables para estimar el movimiento dado a que son muy sensibles a pequeños ruidos. Estos puntos se caracterizan por una baja similitud a puntos esquina lo que significa que tienen una curvatura principal a lo largo del borde pero otra muy pequeña en la dirección perpendicular. Las curvaturas principales de un punto son proporcionales a la matriz Hessiana, H, calculada en la posición del punto. Para reducir el coste computacional se adopta la aproximación de Harris y Stephen [24] que calcula el ratio de las curvaturas principales sin calcular los valores propios explícitamente. La traza ('*Tr*') y el determinante ('*Det*') de H se calculan como:

$$T_r(H) = D_{xx} + D_{yy}$$

$$Det(H) = D_{xx}D_{yy} + \left(D_{xy}\right)^2$$

donde D_{xx}, D_{yy} y D_{xy} son los elementos de H que se calculan utilizando un filtro de Sobel como operador derivada:

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

El ratio entre el valor propio más grande y el más pequeño, 'R_{eig}', es referido a '*Tr*' y a '*Det*':

$$\frac{T_r(H)^2}{Det(H)} < \frac{\left(R_{eig} + 1\right)^2}{R_{eig}}$$

- **Asignación de la orientación.** Gracias a la asignación de una orientación consistente a cada '*keypoint*' basada en propiedades locales de la imagen, el descriptor de los '*keypoints*' puede ser representado de forma relativa a esta orientación y por lo tanto adquiere invarianza a la rotación de la imagen. Se selecciona la escala del '*keypoint*' para seleccionar la imagen suavizada mediante la gaussiana, L. Para cada

muestra, L(x,y), en esta escala, se precalculan los valores de magnitud del vector gradiente, *m(x,y)*, y de la dirección del vector gradiente, *θ(x,y)*, utilizando diferencias entre píxeles:

$$m(x,y) = \sqrt{\big(L(x+1,y) - L(x-1,y)\big)^2 + \big(L(x,y+1) - L(x,y-1)\big)^2}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{\big(L(x,y+1) - L(x,y-1)\big)}{\big(L(x+1,y) - L(x-1,y)\big)}\right)$$

Se forma un histograma de orientación mediante las orientaciones del gradiente de los puntos de muestra dentro de una región alrededor del '*keypoint*'. El histograma de orientación tiene 36 grupos o '*bins*' que cubren los 360 grados del rango de orientaciones. Los picos en el histograma de orientaciones corresponden a direcciones dominantes de los gradientes locales. Se detecta el pico mñas alto en el histograma y todo aquellos picos locales más altos del 80% de aquel.

- **Descriptor de los '*keypoints*'.** Este apartado se explica en detalle en el apartado Descriptores del Anexo 2.

2. **Características SURF.**

'*Speeded Up Robust Features*' [3] [4]. También muy robustas, invariantes a escala y rotación e inspiradas en las anteriores SIFT. Son más rápidas que las SIFT y basadas en sumas de 2D Haar wavelet haciendo uso eficiente de las imágenes integrales [25]. Una imagen integral, I_Σ, es una representación de una imagen de entrada *I*. Cada *píxel* de la I_Σ en una de las localizaciones, x = (x,y), representa la suma de todos los píxeles de la imagen de entrada *I* de la región rectangular formada por el punto x y el origen, $I_\Sigma(x) = \sum_{i=0}^{i\leq x}\sum_{j=0}^{j\leq y} I(i,j)$. Calculada I_Σ sólo son necesarias 4 operaciones de suma para calcular la suma de las intensidades de cualquier área rectangular.

La fase de detección de puntos se basa en la matriz Hessiana pero utilizando una aproximación muy simplificada, inspirada en '*DoG*', '*Difference of Gaussians*'. La matriz Hessiana es el resultado de la convolución de filtros derivativos de segundo orden de tipo Gaussiano que, aunque son óptimos para el análisis en el espacio en la práctica necesitan ser discretizadas y recortadas. En la Figura 32 se muestra como se realiza esta discretización sobre los filtros gaussianos de

segundo orden en la dirección de Y y en la dirección de XY. Estas discretizaciones pueden ser evaluadas con imágenes integrales muy rápidamente.

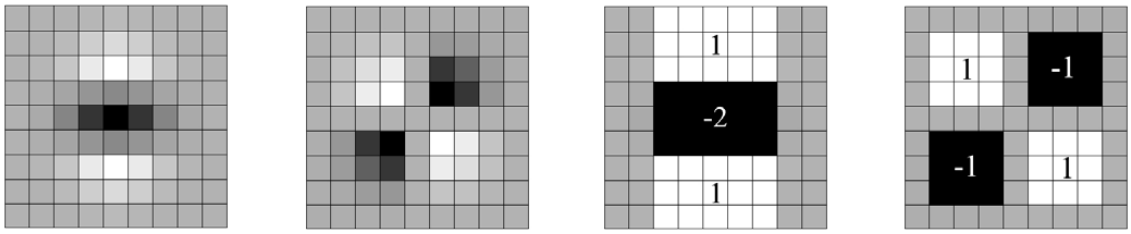


Figura 33: Filtros gaussianos de segundo orden en la dirección Y, y en la dirección XY. Discretizaciones.

Con el fin de que el método sea robusto a cambios de escala, se calculan representaciones de la imagen I a diferentes escalas mediante la implementación en imágenes piramidales. Debido al uso de los filtros y de las imágenes integrales, es posible aplicar los filtros a la imagen I sin reducir su tamaño, sin más que reescalando el tamaño de los filtros al tamaño de la escala superior iterativamente. Posteriormente se le aplica a cada punto de interés un algoritmo de eliminación de puntos no-máximo en la vecindad del punto.

3. **Características FAST.**

'*Features from Accelerated Segment Test*' [7]. Es un método de extracción de esquinas cuya ventaja más destacada es su eficiencia computacional. Para detectar si en un punto p existe una característica FAST, se examinan los 16 píxeles de un círculo alrededor del punto p. Los píxeles elegidos para el test se muestran en la Figura 33 y son los que Bresenham propone para un círculo de radio 3 [26].

Una característica FAST es detectada en p si todas las intensidades de al menos 12 píxeles contiguos son mayores (o menores) que la intensidad del propio punto p más (o menos) un cierto umbral, t.

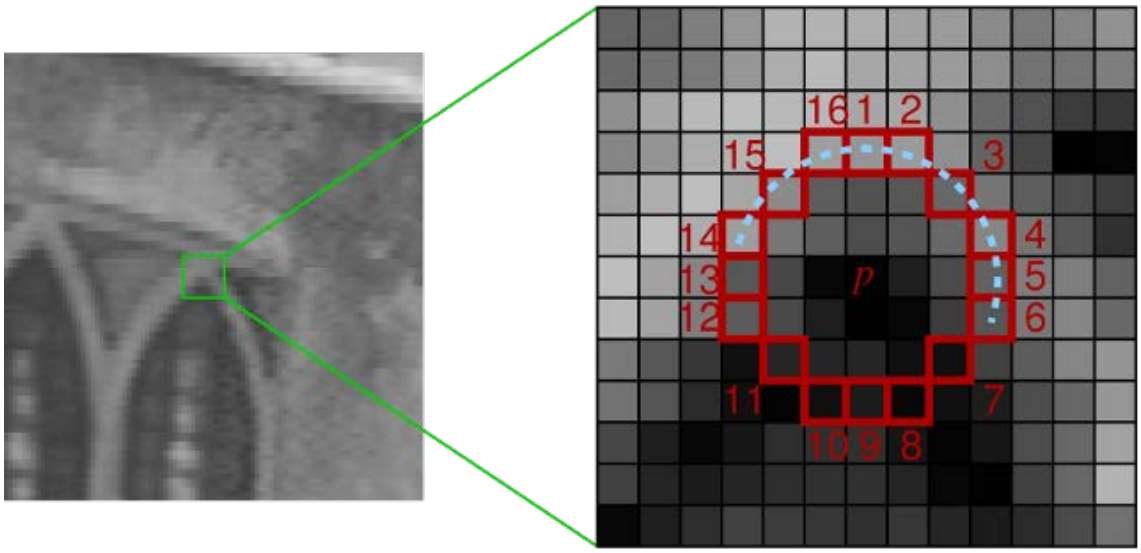


Figura 34: Píxeles del círculo de Bresenham de radio 3.

El chequeo de esta condición puede ser optimizado examinando los píxeles 1, 9, 5 y 13 para descartar candidatos más rápidamente debido a que una característica solo puede existir si tres de estos puntos de test disponen de una intensidad superior o inferior a *p*.

Este tipo de detección de esquinas tiende naturalmente al uso de las intensidades de los 16 píxeles del círculo como vector característico. Este test permite clasificar a las características como *positivas* o *negativas* según si la intensidad de los píxeles son mayores o menores que las del centro respectivamente. Esta clasificación puede ser útil, dado que no es necesario comparar características *positivas* con *negativas* ni viceversa.

4. **Características oFAST.**

‘*Oriented-FAST*’ [10]. Este tipo de características añaden el cómputo eficiente de la orientación a las características de tipo FAST aprovechando así su eficiencia computacional en la extracción. La idea básica es calcular un centroide ponderando las intensidades del ‘patch’ localizado en torno al centro de la esquina detectada. La dirección del vector desde el punto esquina al centroide se toma como la orientación del punto FAST. Para mejorar la invarianza a la rotación se calculan los momentos X e Y que deberían encontrarse dentro de una región circular de radio r, donde r es el tamaño del ‘patch’.

Algoritmo:

a. Se extraen puntos FAST en la imagen estableciendo un umbral de intensidad entre el pixel central y aquellos en un círculo en torno al centro de radio r (r = 9 da buenos resultados habitualmente).
b. Se utiliza la medida de puntos "esquina" de Harris para ordenar los keypoints FAST.
c. Para obtener un número N de keypoints se establece un umbral suficientemente bajo para la medida Harris y se eligen los N mejores keypoints.
d. Debido a que FAST no produce características multi-escala, se emplea una escala piramidal de la imagen y se extraen características FAST en cada nivel de la pirámide.
e. Se utiliza como medida de orientación la "intensidad del centroide". Este método asume que la intensidad del punto "esquina" está desplazada desde su centro y este vector podría ser utilizado para asignar una orientación. Definiendo los momentos:

$$m_{pq} = \sum_{x,y} x^p y^q \, I(x,y)$$

se puede encontrar el centroide:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right)$$

f. Por lo tanto se puede construir un vector desde el centro del punto "esquina", O, al centroide, OC. La orientación del patch se toma como:

$$\theta = atan2(m_{01}, m_{10})$$

5. **Características "Good Features to Track".**

'*Good Features to Track*' [11]. Este método utiliza dos modelos de movimiento en la imagen en lugar de sólo uno.

- Búsqueda de puntos por textura: No todas las partes de una imagen contienen información completa del movimiento. A esta dificultad para observar el movimiento global con solo información local se le denomina '*problema de la apertura'*. Para subsanar esta dificultad los investigadores proponen hacer seguimiento de puntos '*esquina*'. La detección de puntos "esquina" se consigue analizando los valores propios de la matriz de gradientes horizontales y verticales de la imagen en cada punto:

$$Z = \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix}$$

Con el fin de que los puntos seleccionados sean robustos, la matriz Z debe cumplir requisitos derivados del ruido y debe tratarse de una matriz bien condicionada. Los requisitos por ruido implican que ambos valores propios deben ser grandes y para tratarse de una matriz bien condicionada estos valores propios deben encontrarse en los mimos órdenes de magnitud. Dos valores propios pequeños significarían que el contenido en "textura" del punto es pobre. Un valor propio mucho mayor que el otro indicaría un patrón de textura unidireccional. Mientras que dos valores propios grandes y del mismo orden de magnitud representan puntos "esquina".

- Medida de "similaridad": Se propone una medida de "similaridad" basada en transformaciones afines que contemplen deformaciones y no solamente desplazamientos, dado que esto permite a menudo saber que alguna cosa ha ido mal en el tracking. Debido al potencialmente alto número de fotogramas para el que se realiza el seguimiento una medida de "similaridad" basada en un modelo traslacional puro no funcionará correctamente. Por lo tanto se propone una medida de "similaridad" como la siguiente:

$$\epsilon = \iint_W [J(Ax + d) - I(x)]^2 w(x) dx$$

donde W es una ventana en torno a la característica, w(x) es una función de pesos (por ejemplo una gausiana), I y J son las imágenes en fotogramas consecutivos y Ax+d es la transformación afín que incluye la matriz de deformación y de desplazamiento.

Algoritmo de extracción de "GoodFeaturesToTrack":

a. Cálculo de la medida de la calidad de las esquinas mediante el método de los valores propios mínimos.
b. Eliminado de aquellos puntos no máximos en un entorno de 3x3 vecinos.
c. Las esquinas cuyo valor propio más pequeño sea menor que un umbral o cuyos valores propios sean muy diferentes entre sí, son eliminados.
d. Se ordenan los puntos de mayor a menor calidad.
e. Se realiza un filtrado de los puntos. El filtrado consiste en rechazar aquellas esquinas que se encuentran a una distancia menor a un umbral de la previa con mejor calidad

Anexo 2: Descriptores y Emparejamiento o 'matching' de características.

Una vez que se dispone de una serie de las coordenadas de las características en varias imágenes, es necesario identificar cuáles de ellas se corresponden entre sí, es decir, cuáles de esas características son la misma pero vista desde diferente perspectiva. A este proceso de hacer corresponder una característica en una imagen consigo misma vista desde otra perspectiva en otra imagen se le denomina emparejamiento de características o '*matching*'.

Este proceso de '*matching*' necesita de una forma de identificar lo más unívocamente posible a cada una de esas características con independencia de cualquier factor dependiente de la iluminación, la perspectiva, el enfoque, etc… Este identificador es lo que se conoce como descriptor de la característica.

Descriptores

Existen multitud de descriptores de características en la bibliografía de los cuales, en este proyecto, se han utilizado los siguientes:

- **Descriptores SIFT.**

  Los pasos para la detección de los '*keypoints*' SIFT (detallados en el apartado Características SIFT del Anexo 1), han asignado una localización, una escala y una orientación a cada uno de los '*keypoints*'. En la Figura 34 se ilustra el cálculo del descriptor del '*keypoint*'.

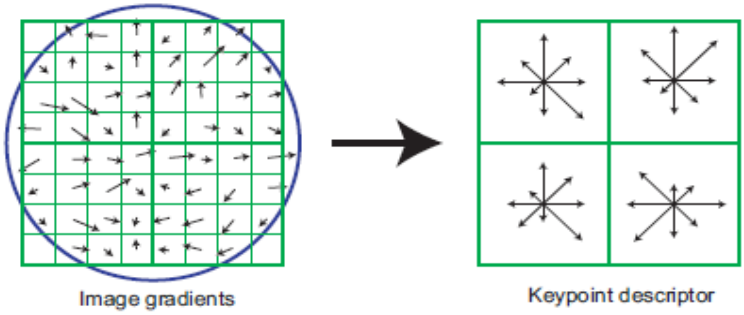


Figura 35: El descriptor de un *keypoint* se crea calculando la magnitud del vector gradiente y la orientación de cada punto de muestra en una región alrededor de la localización del *keypoint*, como se muestra a la izquierda. A la ventana se la aplica una gaussiana ponderada que está representada por el circulo sobreimpreso. Las muestras son acumuladas en histogramas de orientación resumiendo el contenido de subregiones 4x4, como se muestra en la derecha. La longitud de cada flecha se corresponde con la suma de las magnitudes del gradiente en cada dirección dentro de la región. Esta figura muestra un descriptor de 2x2 calculado en conjuntos de muestras de 8x8.

En primer lugar se muestrea la imagen de la magnitud del vector gradiente y las orientaciones alrededor del '*keypoint*' utilizando la escala del propio '*keypoint*' para seleccionar el nivel de la gaussiana que suavizará la imagen. Para conseguir invarianza a orientación las coordinadas del descriptor y las orientaciones de los gradientes se rotan en relación a la orientación del '*keypoint*'.

Se utiliza una función gaussiana ponderada con σ igual a la mitad del ancho de la ventana del descriptor para asignar un peso a la magnitud de cada punto de muestra. El descriptor que se utiliza permite pequeños desplazamientos en las posiciones del gradiente mediante la creación de de histogramas de orientación sobre regiones de 4x4. La Figura 34 muestra 8 direcciones para cada histograma de orientación, en los cuales se indica la magnitud de cada entrada del histograma mediante la longitud de la flecha.

Para evitar los efectos de borde, se aplica una interpolación trilineal para distribuir los valores de cada muestra del gradiente en varios histogramas adyacentes.

El descriptor se forma partiendo del vector que contiene los valores de todas las entradas del histograma de orientaciones. Los mejores resultados se alcanzan con vectores de 4x4 histogramas de 8 orientaciones cada uno, por lo tanto el vector de un descriptor utiliza 4x4x8 = 128 elementos por cada '*keypoint*'.

Finalmente el vector de la característica se modifica para reducir los efectos de los cambios de iluminación normalizándolo a la longitud unidad.

- **Descriptores SURF.**

  La construcción del descriptor conlleva dos pasos:

  - En primer lugar se busca información reproducible de la orientación en una región circular en torno al punto de interés con el fin de que el descriptor sea invariante a rotación. Se calcula la respuesta a las máscaras de Haar, '*Haar-wavelet*' en las direcciones x e y con las máscaras que se muestran en Figura 35, y en una vecindad

compuesta por un circulo de radio 6s donde s es la escala del punto de interés detectado.

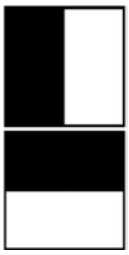


Figura 36: Máscaras Haar en las direcciones X e Y.

En este paso se hace uso de nuevo de las imágenes integrales apra acelerar el proceso.

Una vez calculada la respuesta a la máscara y ponderada por una Gaussiana centrada en el punto de interés, se representa como un vector con el valor de la componente horizontal en abscisas y el valor de la componente vertical en ordenadas. La orientación dominante se estima calculando la suma de todas las respuestas dentro de una ventana deslizante de orientaciones que cubre un ángulo de $\frac{\pi}{3}$. Las respuestas en las componentes horizontal y vertical se suman dando un nuevo vector.

- El segundo paso es construir una región cuadrada alineada con la orientación seleccionada y extraer el descriptor SURF de ella. Para ello el primer paso es construir una región cuadrada centrada en el punto de interés y orientada a lo largo de la dirección de orientación seleccionada en el paso previo.

  La región se divide en 4 x 4 subregiones cuadradas más pequeñas. Para cada subregión se calculan unas pocas características (5x5) en unos puntos de muestra espaciados regularmente. Se llaman d_x y d_y a la respuesta de las máscaras Haar en direcciones horizontal y vertical respectivamente, entendiendo por “horizontal” y “vertical” a éstas direcciones referidas a la orientación seleccionada. Las respuestas d_x y d_y son sumadas sobre cada subregión formando un primer conjunto de entradas en el vector característico. Con el fin de obtener información sobre la ‘polaridad’ de la intensidad se extraen también estas sumas en valor absoluto $|d_x|$ y $|d_y|$. De esta forma en cada

subregión se dispone de un vector del descriptor de 4 dimensiones, $v = \left(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right)$. Esto resulta en un descriptor para todas las subregiones de tamaño 4x4 con una longitud de 64. La respuesta a las máscaras hace al descriptor invariante a cambios de iluminación global. La invarianza al contraste y a escala se obtiene normalizando el vector a la unidad.

- **Descriptores BRIEF.**

  Binary Robust Independent Elementary Features [9]. Se trata de un descriptor que propone el uso de palabras binarias como descriptor eficiente de puntos característicos. Son descriptores robustos a cambios de iluminación, distorsión de la perspectiva y desenfoque, pero muy sensibles a rotación. Como medida de similaridad puede utilizarse la distancia de Hamming cuyo cómputo es más eficiente que la norma L_2 utilizada habitualmente.

  Para la obtención de los bits individuales del descriptor se comparan las intensidades de pares de puntos pero sin la necesidad de una fase de entrenamiento como ocurre en otros métodos.

  El proceso de construcción del vector de bits del descriptor es el siguiente:

  Algoritmo

  - Se calcula un suavizado o smooth del patch p de la imagen en torno al punto del que se quiere obtener el descriptor.
  - Se define como test **τ** en el patch **p** de tamaño **S x S** como:

    $$\tau(p; x, y) := \begin{cases} 1 & if\ p(x) < p(y) \\ 0 & otherwise \end{cases}$$

    donde **p(x)** es la intensidad del pixel en la versión suavizada de **p** en $\mathbf{x} = (u,v)^T$
  - Se elige un conjunto de $n_d(x,y)$ pares de localizaciones lo cual define un conjunto de test binarios. Este método toma el descriptor BRIEF como la cadena de bit n_d-dimensional:

    $$f_{n_d}(p) := \sum_{1 \le i \le n_d} 2^{i-1}\, \tau(p; x_i, y_i)$$

- Las distintas pruebas realizadas en [9] indican que el tamaño del patch de 9x9 píxeles es necesario y suficiente.
- Los mejores resultados en la elección de las n_d localizaciones de los test binarios se consiguen con una distribución como la que sigue: Distribución Gaussiana isotrópica donde:

$$\frac{s}{2} = \frac{5}{2}\sigma \Leftrightarrow \sigma^2 = \frac{1}{25}S^2$$

- **Descriptores rBRIEF.**

Rotated BRIEF [10]. Los descriptores rBRIEF son una modificación de los BRIEF que les permiten ser invariantes a las rotaciones en el plano. Son un tipo de descriptores que permiten utilizar la información de rotación que incorporan los puntos oFAST para construir un descriptor de palabras binarias invariante a rotación. En [10] se introduce el descriptor steered-BRIEF para dotar al descriptor BRIEF de invariabilidad frente a rotaciones en el plano a través de la información de la orientación de los keypoints. Así, para todo el conjunto de características de n test binarios en las localizaciones (x_i, y_i), se define la matriz 2 x n:

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{x_1}, & \dots, & \boldsymbol{x_n} \\ \boldsymbol{y_1}, & \dots, & \boldsymbol{y} \end{pmatrix}$$

utilizando la orientación del patch θ y la correspondiente matriz de rotación R_θ, se construye una versión "steered" S_θ de S:

$$S_\theta = R_\theta S$$

por lo que el operador steered BRIEF queda así:

$$g_n(p, \theta) := f_n(p) | (x_i, y_i) \in S_\theta$$

discretizando el ángulo a incrementos de 2π/30 (12 grados) y construyendo una look-at-table se pueden tener pre-calculados los patrones BRIEF.

Se demuestra en este artículo que es necesario reducir la correlación entre los test binarios y evitar la pérdida de la varianza que supone el método de steered-BRIEF por lo que se propone un método de aprendizaje para seleccionar un buen conjunto de test binarios.

Algoritmo:

- Se establece un conjunto de entrenamiento de N keypoints.
- Se enumeran todos los posibles test binarios con patch de tamaño 31 x 31. Cada test es una pareja de 5x5 sub-ventanas del patch.
- Se ejecuta cada uno de los test contra todos los patches de entrenamiento.
- Se ordenan los test por distancia a una media de 0.5, formando el vector T.
- Se realiza una búsqueda exhaustiva:
  - Se coloca el primer test en el vector R y se elimina de T.
  - Se toma el siguiente test de T y se compara con todos los test de R. Si la correlación absoluta es mayor que un umbral, se descarta, si no se añade a R.
  - Se repite el paso anterior hasta que se han añadido 256 test a R. Si no se consiguen al menos 256 test se aumenta el umbral y se repite el proceso.

Al resultado obtenido, es a lo que se denomina rBRIEF.

Matching

El proceso de '*matching*' es el encargado de hacer corresponder los descriptores de un grupo de características (al que se denomina '*query*') con los descriptores de otro grupo de entrenamiento o previo (al que se denimona '*train*') de la forma más consistente posible utilizando algún método de emparejamiento y mediante una medida concreta de similaridad.

En la bibliografía se destacan dos métodos de emparejamiento principalmente:

- Emparejamiento por fuerza bruta. Este método consiste en realizar una búsqueda barriendo todas las posibilidades existentes, es decir, en el caso de emparejamiento de características, todos los descriptores del grupo '*query*' se comparan con todos los descriptores del grupo '*train*' extrayendo la mejor correspondencia para cada uno de ellos.
- Método FLANN de emparejamiento (Fast Library for Approximate Nearest Neighbors) [27]. Permite un emparejamiento más rápido que el de fuerza bruta para un número elevado de muestras gracias a un entrenamiento previo.

Las medidas de similaridad más frecuentemente utilizadas son:

- Norma L_2. La norma L_2 se define como la raíz cuadrada de la suma de las diferencias cuadráticas entre los dos descriptores, d_{query}, d_{train}.

$$L_2 = \sqrt{\sum_{k=1}^{n} \left(d_{query_k} - d_{train_k} \right)^2}$$

- Distancia de Hamming. La distancia de Hamming se define como el número de bits que tienen que cambiarse para transformar una palabra válida en otra palabra válida. Si dos palabras de código difieren en una distancia d, se necesitan d cambios para convertir una en la otra. Si se utiliza en el ámbito de comparación de los descriptores, representa al número de bits de un descriptor '*query*' que deben cambiar para convertirse en el descriptor '*train*' con el que se está comparando.
- Sum of Square Differences, '*SSD*'. Se trata de un método para medir la similaridad entre dos elementos compuestos como pueden ser descriptores o regiones de una imagen. Si 'I_1' es un elemento compuesto por (u,v) elementos simples que se quiere comparar con un elemento compuesto 'I_2' del mismo tamaño, la SSD se define como:

$$SDD = \sum_{u,v} \left(I_1(u,v) - I_2(u,v) \right)^2$$

Con el objetivo de que el proceso de emparejamiento sea lo más robusto posible, se realizan varias operaciones:

- Doble emparejamiento.
  - Emparejamiento de los descriptores de la 'I_0', '*query*', con los descriptores de la 'I_R', '*train*'. En esta fase se utiliza la medida de similaridad Norma L_2 explicada anteriormente. Se obtienen los '*matches1*' correspondientes a los emparejamientos más similares.
  - Etapa inversa a la anterior. Emparejamiento de los descriptores de la 'I_R', '*train*' con los descriptores de la 'I_0', '*query*'. Se obtienen los '*matches2*'.

  - Test de simetría. De entre los emparejamientos '*matches1*' y '*matches2*', solo se seleccionan aquellos de los '*matches1*' que han dado resultados correctos en '*matches2*' y que simultáneamente los '*matches2*' han dado en '*matches1*' los mismos emparejamientos. Así se seleccionan los '*matches*'.

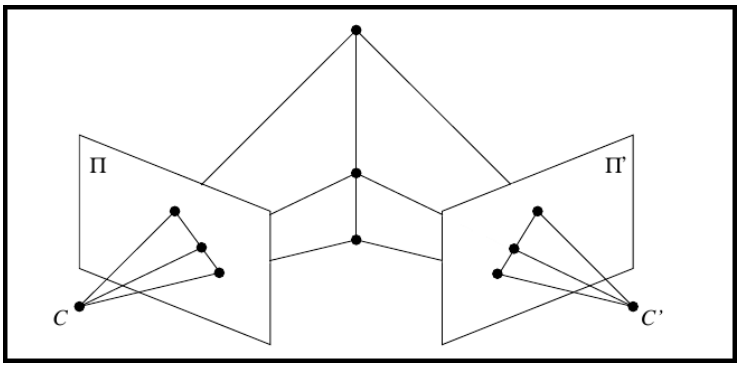
- Filtrado por '*ratio*'. Dado que en cada uno de los emparejamientos se buscan los 2 mejores '*train*' para cada '*query*'y viceversa, se puede hacer un filtrado por ratio. Este filtrado consiste en comparar entre si cada uno de los 2 descriptores '*train*' encontrados para un mismo '*query*' y descartar este emparejamiento si la distancia entre ellos es mínima. Recíprocamente se filtran para los descriptores '*query*' encontrados para un mismo '*train*'. En definitiva, los dos descriptores encontrados cuando se busca el descriptor más parecido a uno dado, deben ser diferentes y diferenciables.
- Filtrado por RANSAC. Se realiza un test de simetría, es decir, Se someten los '*matches*' a un test de RANSAC para eliminar espurios tomando como modelo geométrico una transformación homográfica. Este método de filtrado se explica a continuación.

En visión por computador existen técnicas que permiten mejorar y refinar el emparejamiento de características. El más conocido y quizá más fiable es el método RANSAC, RANdom Sample Consensus [12] [13]. Este método de ajuste de características a geometrías conocidas es especialmente útil en el filtrado de puntos para el cálculo de homografías entre planos [14].

Homografía

Una homografía es una transformación lineal no singular de los puntos pertenecientes a un plano. Intuitivamente se puede ver como la transformación proyectiva (no perspectiva) de un plano. Geométricamente y matemáticamente una homografía será una matriz de transformación afín que nos pondrá en correspondencia los puntos pertenecientes a un plano entre diferentes vistas del mismo.

Una vista es una proyección perspectiva del mundo real (en este caso, el mundo real será un plano).



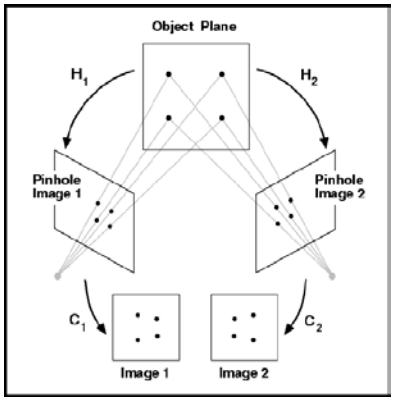
En el mundo real una homografía pondrá en correspondencia ideal todos los puntos de una vista tan sólo para aquellas vistas en las que únicamente exista movimiento rotacional de la cámara, es decir, sin cambio de perspectiva.

Sin embargo, dado que una homografía permite hacer correspondencias entre los puntos de un mismo plano aún cuando entre las diferentes vistas exista traslación, giro o simetría, el uso de homografías es utilizado en multitud de aplicaciones:

- Corrección de perspectiva.
- Creación de panoramas.
- Captura, restauración y corrección de documentos.
- Calculo de velocidades, distancias y trayectorias, etc.

El proceso de corrección de perspectiva es un problema recurrente en muchas aplicaciones de visión artificial y en concreto en tareas como ayuda al tracking, cálculo de velocidades y distancias reales, tracking multi-cámara, ampliación del campo de visión [28], etc…

Se suele utilizar principalmente cuando se quiere una vista concreta de la escena u objeto de interés mediante una transformación perspectiva. El objetivo es el de llevar la proyección de un plano desde una vista origen a otra destino. Lo primero es recuperar la geometría del plano. La forma de hacer esto es con correspondencias puntuales.



Tras ello necesitamos calcular una matriz de transformación que nos mapee los puntos desde la vista origen a la destino. Este tipo de transformación se le conoce como transformación proyectiva y se trata de una extensión de la transformación afín (la cual es un

caso particular de homografía con h32=0 y h33=1).

$$\begin{pmatrix} u_i \\ v_i \\ w_i \end{pmatrix} = \tilde{x}_i' = H\tilde{x}_i = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

Ecuación 1: Cálculo de una Homografía

Partiendo de la Ecuación 1 se obtienen dos ecuaciones por punto (en realidad son 3 ecuaciones pero la tercera es linealmente dependiente de las otras dos):

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$
$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

Tenemos por lo tanto 9 parámetros a calcular, pudiendo ser reducidos a 8 mediante una restricción de escala a la unidad a causa de la igualdad de resultados tras la normalización del punto. Para ello forzaríamos el valor del último parámetro: H33 = 1.

$$(sH)\tilde{x}_i = s\tilde{x}_i' = \tilde{x}_i'$$

Nos queda un sistema matricial con 8 grados de libertad. Por ello se hace necesario al menos 4 puntos para su resolución, cada uno de los puntos genera 2 ecuaciones, y tenemos así 8 variables efectivas. Pasando por lo tanto a forma matricial todas las ecuaciones obtenemos el sistema final a resolver.

$$\begin{array}{l} \\ \textbf{Point 1} \\ \\ \textbf{Point 2} \\ \\ \textbf{Point 3} \\ \\ \textbf{Point 4} \\ \\ \textbf{additional} \\ \textbf{points} \end{array} \overset{\textbf{2N x 8}}{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x_2' & -y_2x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y_2' & -y_2y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x_3' & -y_3x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y_3' & -y_3y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x_4' & -y_4x_4' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y_4' & -y_4y_4' \end{bmatrix}} \overset{\textbf{8 x 1}}{\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix}} = \overset{\textbf{2N x 1}}{\begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \end{bmatrix}}$$

Una vez definido el sistema de ecuaciones a resolver, tan solo nos queda resolverlo mediante alguna de las técnicas existentes para ello. No obstante, el

ruido de la imagen y los sensores hace necesario el refinamiento de las técnicas básicas para que el error numérico no se dispare.

Para la resolución del sistema es común utilizar el algoritmo DLT (‘*Direct Linear Transform*’). Para cada correspondencia puntual se generan sus dos correspondientes ecuaciones y se obtiene el sistema final de ecuaciones en forma matricial.

$$\underset{2Nx9}{\mathbf{A}}\ \underset{9x1}{\mathbf{h}} = \underset{2Nx1}{\mathbf{0}}$$

Se desarrolla de la siguiente forma:

$$\underset{9x2N}{\mathbf{A}^T}\ \underset{2Nx9}{\mathbf{A}}\ \underset{9x1}{\mathbf{h}} = \underset{9x2N}{\mathbf{A}^T}\ \underset{2Nx1}{\mathbf{0}}$$

$$\underbrace{(\mathbf{A}^T\ \mathbf{A})}_{9x9}\ \underset{9x1}{\mathbf{h}} = \underset{9x1}{\mathbf{0}}$$

Obteniéndose así los valores singulares mediante una descomposición SVD de la matriz no paramétrica. La solución al sistema será la columna de U correspondiente al menor valor propio de D.

$$\mathbf{SVD\ of\ A^TA} = \mathbf{U}\ \mathbf{D}\ \mathbf{U^T}$$

Para evitar el error numérico producido por diversos factores como “*outliers*”, el uso de algoritmos de filtrado como RANSAC se hacen casi obligatorios, mientras que los errores producidos por ruido en las mediciones obligan a utilizar procesos de normalización.

El algoritmo de normalización consiste en trasladar el centro de masas al origen y escalar la dispersión de los datos en este caso concreto suele escalarse a Sqrt(2) [14].

RANSAC está diseñado para tratar con conjuntos de datos que contienen una gran cantidad de espurios o datos erróneos. Se trata de una técnica de muestreo que genera soluciones candidatas utilizando el mínimo número de observaciones requerido para satisfacer los parámetros de un modelo determinado. RANSAC utiliza el conjunto más pequeño posible y procede a acrecentarlo con puntos “dato” consistentes.

El algoritmo básico se puede resumir como sigue:

Algoritmo RANSAC

1. Selección aleatoria de un conjunto de puntos de tamaño mínimo para determinar los parámetros del modelo. En el caso del cálculo de una homografía se seleccionan 4 correspondencias.
2. Resolver los parámetros del modelo, chequeando en el caso particular de las homografías que los puntos no se encuentran en la misma línea.
3. Determinar cuántos puntos del conjunto de todos los puntos se ajustan al modelo calculado con una tolerancia ϵ. En el cálculo de homografías este proceso consiste en:
    a. Calcular la homografía H_{curr} mediante la normalización DLT de los 4 pares de puntos.
    b. Para cada correspondencia, calcular para la actual H_{curr}, la distancia:
    $$d_i = d\left(\vec{X}_i', H_{curr}\vec{X}_i\right) + d\left(\vec{X}_i, H_{curr}^{-1}\vec{X}_i'\right)$$
    c. Calcular la desviación estándar de la distancia de inlier curr_std.
4. Si la fracción del número de puntos dentro de tolerancia o "inliers" sobre el total de puntos del conjunto excede un determinado umbral τ, se re-estiman los parámetros del modelo utilizando todos los "inliers" identificados y se termina. Para el cálculo de homografías este proceso consiste en:
    a. Contar el número de inliers m que tienen distancia menor de un umbral T_DIST, di < T_DIST.
    b. Si m > MAX_inlier o (m == MAX_inlier & curr_std < MIN_std) se actualiza la H = Hcurr con la mejor de las calculadas y se guardan los inliers, donde MAX_inlier = τ, y MIN_std es el umbral para la desviación.
    c. Se actualiza N utilizando la Ecuación 2.
    d. Se re-estima H mediante el algoritmo DLT utilizando los inliers.
5. De otro modo, repetir los pasos del 1 al 4 un máximo de N iteraciones.

El número de iteraciones, N, se elige suficientemente grande para asegurar con una probabilidad ρ (normalmente del 0.99) de que al menos uno de los conjuntos de muestras aleatorias no incluye ningún "outlier".

De esta forma si u representa la probabilidad de que cualquier punto seleccionado es un "inlier" and v = 1 - u la probabilidad de que sea "outlier", se necesitan N iteraciones del número mínimo de puntos (denotado por m), donde:

$$1 - p = (1 - u^m)^N$$

Reordenando y acomodando la fórmula:

$$N = \frac{\log(1-p)}{\log(1-(1-v)^m)}$$

Ecuación 2: Cálculo del número de iteraciones del algoritmo de RANSAC

Anexo 3: Métodos de Tracking

Se entiende por tracking en el campo de la Visión por Computador al proceso de estimar la ubicación de uno o más objetos móviles a lo largo del tiempo mediante el uso de técnicas y algoritmos de análisis de imagen.
El seguimiento de objetos puede ser un proceso lento debido a la gran cantidad de datos que contiene un video. Además, la posible necesidad de utilizar técnicas de reconocimiento de objetos para realizar el seguimiento incrementa su complejidad.

Los principales retos que hay que tener en cuenta en el diseño de un sistema de seguimiento de objetos están relacionados con la similitud de aspecto entre el objeto de interés y el resto de objetos en la escena, así como la variación de aspecto del propio objeto. Dado que el aspecto tanto del resto de objetos como el fondo puede ser similar al del objeto de interés, inferir su posición puede resultar una tarea compleja. En ese caso, las características extraídas de esas áreas no deseadas puede ser difícil de diferenciar de las que se espera que el objeto de interés genere. Este fenómeno se conoce con el nombre de '*clutter*'.

Dificultades del seguimiento

Además del reto de seguimiento que causa el '*clutter*', los cambios de aspecto del objeto en el plano de la imagen dificultan el seguimiento causado por uno o varios de los siguientes factores:

- **Cambios de posición.** El objeto móvil de interés varia su aspecto cuando se proyecta sobre el plano de la imagen, por ejemplo, al girar.
- **Iluminación ambiente.** La dirección, la intensidad y el color de la luz ambiente influyen en el aspecto del objeto de interés. Los cambios en la iluminación global son, con frecuencia, un gran reto en las escenas al aire libre.
- **Ruido.** El proceso de adquisición introduce un cierto grado de ruido en la señal de la imagen que depende de la calidad del sensor. Las observaciones del objeto de interés pueden dañarse y por tanto afectar al rendimiento del seguimiento.

- **Oclusiones.** Puede ser que un objeto de interés no se observe bien cuando sea parcial o totalmente tapado por otros objetos en la escena. Las oclusiones son generalmente debidas a:
    - Un objeto de interés que se mueve detrás de un objeto estático, como por ejemplo una columna.
    - Otros objetos que se mueven en la escena de manera que entorpecen la visión de un objeto de interés.

Métodos de tracking

Los objetos se pueden representar mediante sus formas o apariencias, aunque una de las caracterizaciones más utilizadas generalmente se basan en la extracción de un conjunto de puntos de interés.

Éste es el método principal utilizado durante este proyecto y todos los métodos utilizados para conseguir un tracking robusto se basan en el seguimiento de estos puntos de interés.

1. **Flujo óptico disperso y tracking KLT.**

   Algoritmo de seguimiento basado de Lucas-Kanade [15] que calcula del flujo óptico de forma piramidal. Es un método muy rápido y que obtiene muy buenos resultados en ausencia de oclusiones [16].

   En visión por computador se conoce por flujo óptico al movimiento aparente de los píxeles de una imagen de un fotograma a otro dentro de una secuencia de video. Con puntos en tres dimensiones, el flujo óptico es definido como una medida de cómo se movieron dichos puntos en el espacio. El cálculo del flujo óptico no es un problema sencillo y computarlo para tomas realizadas en un ambiente real puede llegar a ser muy complicado. Por eso los algoritmos existentes se basan en determinadas hipótesis (intensidad constante, rigidez de los objetos, coherencia espacial, entre otras) que generalmente no se cumplen totalmene en escenarios reales.

Tipos de flujo óptico

Independientemente de la técnica que implementen, por los resultados que producen, los métodos pueden clasificarse en dos grandes grupos, algoritmos de flujo óptico disperso y algoritmos de flujo óptico denso.

- Flujo óptico disperso

  Como su nombre lo sugiere el flujo óptico disperso no computa los vectores de velocidad para cada píxel de la imagen. El primer paso es seleccionar una serie de regiones de la imagen que sean buenas para rastrear (bordes, esquinas, manchas, etc). Luego para ese conjunto reducido de puntos se calcula el flujo óptico para toda la imagen. Uno de los primeros algoritmos que uso este enfoque fue el de Lukas-Kanade propuesto en 1981.

- Flujo óptico denso

  El flujo óptico denso a diferencia del disperso se caracteriza por computar el vector de velocidad para cada píxel de la imagen. Generalmente este es un proceso computacionalmente costoso por lo que no es factible aun hacerlo en tiempo real, aunque con el desarrollo de las tarjetas de video y los algoritmos recientes se han logrado grandes avances en este sentido. Un exponente clásico de este grupo es el algoritmo de Horn y Shunk que data de 1981.

La principal hipótesis detrás del cálculo del flujo óptico es la conservación de la intensidad o el color de un píxel en el tiempo a pesar del posible cambio de posición. Matemáticamente se expresa mediante la ecuación.

$$f(x + \Delta x, y + \Delta y, t + \Delta t) \approx f(x, y, t)$$

La ecuación no brinda en todos los casos una solución única. Esto es conocido como el problema de apertura. El problema de computar la velocidad total de la imagen se transforma entonces en encontrar otra restricción que permita despejar la incógnita restante. Existen varias técnicas para intentar dar solución a esta ecuación, técnicas diferenciales, de análisis de frecuencia o energía y las basadas en correlación. Estas últimas se basan en la búsqueda de correspondencias

utilizando ventanas o patrones alrededor de cada píxel. La ventaja que tienen con respecto a los anteriores, es que se utiliza información de los vecinos que en muchos casos hacen que la búsqueda de la correspondencia sea más efectiva. Lo que se busca con estos métodos es asociar píxeles a través de regiones similares en las imágenes que se obtienen por maximización de alguna medida de similaridad. Serán este tipo de técnicas las que utilizaremos en este proyecto.

En concreto en [16] el método compara la intensidad de dos imágenes en escala de grises *I* y *J* en un entorno de los puntos de interés. El objetivo del tracking de características es encontrar la localización $v = u + d = [u_x + d_x\ u_y + d_y]^T$ en la segunda imagen *J*, de los puntos $x = [x\ y]^T$ en la primera imagen *I*. El vector $d = [d_x\ d_y]^T$ es considerado como la velocidad de la imagen en **x**, y $u = [u_x\ u_y]^T$ es el punto en la imagen *I*.

Se define la velocidad de la imagen como el vector que minimiza la función residual definida por:

$$\epsilon(d) = \epsilon(d_x, d_y) = \sum_{u_x - w_x}^{u_x + w_x} \sum_{u_y - w_y}^{u_y + w_y} \left(I(x,y) - J(x + d_x, y + d_y)\right)^2$$

Ecuación 3: Función residual para el cálculo del flujo óptico

Con el doble objetivo de mejorar la fiabilidad frente a cambios de escala y al mismo tiempo poder manejar grandes desplazamientos de los píxeles, esta implementación al igual que otras implementaciones de métodos de extracción de características, construyen una representación piramidal de imágenes.

Se define una representación piramidal de una imagen genérica *I* de tamaño n_x x n_y como una representación recursiva de *I*, $I^L(x,y)$ donde:

$$\begin{aligned} I^L(x,y) = {} & \frac{1}{4} I^{L-1}(2x, 2y) \\ & + \frac{1}{8}\left(I^{L-1}(2x-1,2y) + I^{L-1}(2x+1,2y) + I^{L-1}(2x,2y-1)\right. \\ & \left. + I^{L-1}(2x,2y+1)\right) \\ & + \frac{1}{16}\left(I^{L-1}(2x-1,2y-1) + I^{L-1}(2x+1,2y+1)\right. \\ & \left. + I^{L-1}(2x-1,2y+1) + I^{L-1}(2x+1,2y+1)\right) \end{aligned}$$

Por lo tanto el nuevo tamaño de esta representación es:

$$n_x^L \leq \frac{n_x^{L-1}+1}{2}$$

$$n_y^L \leq \frac{n_y^{L-1}+1}{2}$$

Donde n_x , n_y son el ancho y alto de la nueva representación de la imagen respectivamente.

Por lo tanto, calcular el flujo óptico de los puntos de interés en uno cualquiera de los niveles de la estructura piramidal, significa, aplicando una generalización de la Ecuación 3, resolver la función residual:

$$\epsilon^L(d^L) = \epsilon^L\left(d_x^L, d_y^L\right) = \sum_{u_x^L-w_x}^{u_x^L+w_x}\sum_{u_y^L-w_y}^{u_y^L+w_y}\left(I^L(x,y) - J^L\left(x+g_x^L+d_x^L, y+g_y^L+d_y^L\right)\right)^2$$

Ecuación 4: Función residual para el cálculo del flujo óptico en el nivel L de la pirámide

Donde $g^L = [g_x^L \text{ y } g_y^L]^T$ es la posición de inicialización del flujo óptico para el nivel L.

A continuación el resultado de este cálculo se propaga a próximo nivel L-1 transmitiendo una nueva posición inicial g^{L-1}, se resuelve de nuevo la Ecuación 4, y así sucesivamente.

El método iterativo para el cálculo del flujo óptico por Lucas-Kanade tiene por objetivo encontrar el vector d^L que minimiza la función ϵ^L definida en la Ecuación 4.

Dado que el punto óptimo se alcanza cuando la primera derivada de ϵ con respecto al vector desplazamiento v es cero, y haciendo aproximaciones por la Taylor de primer orden se llega a la expresión del cálculo óptimo para el vector de flujo óptico:

$$\overline{\nu}_{\text{opt}} = G^{-1}\,\overline{b}.$$

Donde,

$$G \doteq \sum_{x=p_x-\omega_x}^{p_x+\omega_x}\sum_{y=p_y-\omega_y}^{p_y+\omega_y}\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \qquad \overline{b} \doteq \sum_{x=p_x-\omega_x}^{p_x+\omega_x}\sum_{y=p_y-\omega_y}^{p_y+\omega_y}\begin{bmatrix} \delta I\, I_x \\ \delta I\, I_y \end{bmatrix}$$

Aplicando un método iterativo al cálculo del vector de flujo óptico se alcanzará el desplazamiento con el margen de error necesario. El pseudo-código que describe el proceso es como sigue:

Objetivo: Sea u un punto en la imagen *I*. Encontrar su localización correspondiente v en la imagen *J*.

Se construye las representaciones piramidales de I y J: $\{I^L\}_{L=0,\ldots,Lm}$ *y* $\{J^L\}_{L=0,\ldots,Lm}$.

Inicialización de posiciones en la representación piramidal: $g^{Lm} = [g_x^{Lm}\ g_y^{Lm}]T = [0\ 0]^T$

for L=Lm **to** 0 **con step de** -1

Localización del punto u en la imagen I^L: $u^L = \left[p_x\ p_y\right]^T = \frac{u}{2^L}$

Derivada de I^L con respecto a x: $I_x(x,y) = \frac{I^L(x+1,y)-I^L(x-1,y)}{2}$

Derivada de I^L con respecto a y: $I_y(x,y) = \frac{I^L(x,y+1)-I^L(x,y-1)}{2}$

Matriz de gradiente espacial:

$$G = \sum_{p_x-w_x}^{p_x+w_x}\sum_{p_y-w_y}^{p_y+w_y}\begin{bmatrix} I_x^2(x,y) & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y^2(x,y)\end{bmatrix}$$

Inicialización de la iteración L-K: $\bar{v}^0 = [0\ 0]^T$

for k=1 **to** K **con step de** 1 (o hasta $\|\mu^k\|<$ umbral de seguridad)

Diferencia de imágenes:

$$\delta I_k(x,y) = I^L(x,y) - J^l\left(x + g_x^L + v_x^{k-1}, y + g_y^L + v_y^{k-1}\right)$$

Vector de diferencias de imágenes:

$$\bar{b}_k = \sum_{p_x-w_x}^{p_x+w_x}\sum_{p_y-w_y}^{p_y+w_y}\begin{bmatrix}\delta I_k(x,y)I_x(x,y) \\ \delta I_k(x,y)I_y(x,y)\end{bmatrix}$$

Flujo óptico L-K: $\bar{\mu}^k = G^{-1}\bar{b}_k$

Predicción para la próxima iteración: $\bar{v}^k = \bar{v}^{k-1} + \bar{\mu}^k$

end for en k

Flujo óptico final en el nivel L: $d^L = \bar{v}^k$

Predicción para el siguiente nivel L-1: $g^{L-1} = \left[g_x^{L-1}\ g_y^{L-1}\right]^T = 2(g^L + d^L)$

end for en L

Vector de flujo óptico final: $d = g^0 + d^0$

Localización del punto en J: $v = u + d$

Solución: El correspondiente punto está en la localización v de la imagen *J*.

Este algoritmo declara como características "perdidas" aquellas que se encuentran en dos casos particulares:

- El nuevo punto cae fuera de la imagen *J*.
- El "patch" de la imagen alrededor del punto en seguimiento varia demasiado entre las imágenes *I* y *J*.

2. Algoritmo de tracking de Kalman.

El filtro de Kalman [17] es un algoritmo desarrollado por Rudolf E. Kalman en 1960 que sirve para poder identificar el estado oculto (no medible) de un sistema dinámico lineal, pero sirve además cuando el sistema está sometido a ruido blanco aditivo. La diferencia del filtro de Kalman sobre otros métodos radica en que Kalman es capaz de escoger la ganancia K de realimentación del error de forma óptima cuando se conocen las varianzas de los ruidos que afectan al sistema. En este proyecto se utiliza la implementación detallada en [18].

El filtro de Kalman aborda el problema general de intentar estimar el estado x de un proceso controlado y discreto en el tiempo que está gobernado por la ecuación diferencial estocástica lineal:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}$$

con una medida dada por:

$$z_k = Hx_k + v_k$$

Las variables aleatorias w_k, y v_k representan el ruido del proceso y ruido de la medida respectivamente. Se asumen que son independientes, blanco, y de distribución de probabilidad normal:

$$p(w) \sim N(0, Q)$$
$$p(v) \sim N(0, R)$$

La covarianza del ruido de proceso, Q y covarianza del ruido de medida, R, se asumen constantes.

La matriz A n x n relaciona el estado en el instante anterior $k-1$ con el instante actual k, en ausencia de ruido de proceso. La matriz B n x l relaciona las entradas de control opcionales u con el estado x. La matriz H m x n relaciona el estado con la medida z_k. Todas se asumen constantes.

Se define como estado estimado a priori en el instante k siendo conocido el proceso hasta el instante anterior al k, y el estado estimado a posteriori en el instante k conocida la medida z_k respectivamente a:

$$\hat{x}_k^- \in R^n, \hat{x}_k \in R^n$$

El objetivo del filtro de Kalman es encontrar una ecuación que calcule un estado estimado a posteriori como combinación lineal del estado a priori y la diferencia ponderada entre la medida y la predicción de la medida:

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-)$$

3. Algoritmo de tracking ORB.

Este método, descrito en [10], se basa en la combinación de la extracción de características FAST con información de orientación (oFAST) y los descriptores rBRIEF descritos en el Anexo 1. Está diseñado para combinar la ligereza computacional de las características FAST con un método que aporta robustez frente a giros y cambios de escala. Este método añade información de la orientación en los descriptores y utiliza fuerza bruta en el proceso de *'matching'*. Dado que rBRIEF es un patrón binario, se elige Locality Sensitive Hashing [29] como método de búsqueda del vecino más cercano. En LSH, los puntos son almacenados en varias tablas '*hash*' en diferentes bloques. Dado un descriptor, se calcula el bloque al que pertenece y se empareja por fuerza bruta con los elementos del bloque.

Para características binarias, la función '*hash*' es simplemente un subconjunto de los bits de la firma: el bloque en la tabla '*hash*' contiene descriptores con subconjuntos de datos comunes de la firma. La distancia utilizada en el emparejamiento es la distancia de Hamming. Se utiliza 'multi-probe LSH' [30] que mejora el tradicional LSH mediante búsqueda en los bloques vecinos cuando la búsqueda normal falla.

4. Algoritmo de tracking FFME.

Fast Feature-based Motion Estimation [19] es un método de estimación del movimiento robusto al ruido y al problema de la apertura. La selección de los puntos singulares se lleva a cabo mediante la imposición de tres restricciones:

- La primera restricción selecciona los puntos cuya magnitud del vector gradiente ('*gradient magnitude*') sea mayor que un cierto umbral de ruido

adaptativo (‘Th_{gm}’), obteniéndose así ‘SP_{gm}’. La magnitud del vector gradiente,$\|\nabla I(x,y)\| = \sqrt{I_x^2 + I_y^2}$, se calcula utilizando una ‘*look-up-table*’ que se explicará más adelante. La imagen de la magnitud del vector gradiente se umbraliza por medio de ‘Th_{gm}’, el cual es calculado como una función de la distribución del ruido en la imagen. Asumiendo que se trata de un ruido Gaussiano, la correspondiente imagen de la magnitud del vector gradiente tiene una distribución de Rayleigh [31]:

$$R(\|\nabla I(x,y)\|) = \frac{\|\nabla I(x,y)\|}{\sigma^2} e^{-\frac{\|\nabla I(x,y)\|^2}{2\sigma^2}}$$

Así, el ‘Th_{gm}’ se calcula como un parámetro de la función de Rayleig, σ:

$$Th_{gm} = \sigma\sqrt{-2\ln(P_f)}$$

donde *Pf* es la proporción aceptable de confianza para los puntos singulares debida a los picos de ruido. La aproximación de Rosin [32] basada en el algoritmo ‘*Least Median Squares*’ (‘*LMedS*’) calcula una buena aproximación de σ:

$$\hat{\sigma} = 0.968 \min_j(median_{\forall i}(H(i) - j)^2)$$

donde H(i) es el valor del histograma de la magnitud del vector gradiente i, y 0.968 es el factor de corrección relativo a la estimación LMedS de una distribución de Rayleigh no contaminada con σ.

- La segunda restricción rechaza de entre los ‘SP_{gm}’ aquellos puntos con baja similitud a puntos esquina, resultando los ‘SP_{cor}’. Los puntos ‘SP_{gm}’ localizados a lo largo de bordes rectos no son fiables para estimar el movimiento dado a que son muy sensibles a pequeños ruidos. Estos puntos se caracterizan por una baja similitud a puntos esquina lo que significa que tienen una curvatura principal a lo largo del borde pero otra muy pequeña en la dirección perpendicular. Las curvaturas principales de un punto son proporcionales a la matriz Hessiana, H, calculada en la posición del punto. Para reducir el coste computacional se adopta la aproximación de Harris y Stephen [24] que calcula el ratio de las curvaturas principales sin calcular los valores propios explícitamente. La traza (‘*Tr*’) y el determinante (‘*Det*’) de H se calculan como:

$$T_r(H) = D_{xx} + D_{yy}$$

$$Det(H) = D_{xx}D_{yy} + \left(D_{xy}\right)^2$$

donde D_{xx}, D_{yy} y D_{xy} son los elementos de H que se calculan utilizando un filtro de Sobel como operador derivada:

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

El ratio entre el valor propio más grande y el más pequeño, 'R_{eig}', es referido a '*Tr*' y a '*Det*':

$$\frac{T_r(H)^2}{Det(H)} < \frac{\left(R_{eig} + 1\right)^2}{R_{eig}} = Th_{cor}$$

Como resultado de aplicar la restricción de similitud a puntos esquina, se obtienen los 'SP_{cor}'.

- La selección final, 'SP_{fin}', se obtiene aplicando un algoritmo de supresión de no-máximos en el espacio de los puntos esquina que elimina puntos que no son significativos en relación a sus vecinos.

Para realizar el cálculo del descriptor se utiliza la aproximación de Lowe que consiste en permitir pequeños desplazamientos en la localización de los valores de los gradientes durante el cálculo de la medida de similaridad. Para realizar el cálculo del descriptor de Lowe se calculan la magnitud del vector gradiente ('*gradient magnitude*') y la dirección del vector gradiente ('*gradient phase*') en la vecindad de los puntos singulares definiendo una ventana cuadrada en su entorno llamada ventana de descripción.

La magnitud del vector gradiente se suaviza con una función Gaussiana. La ventana de descripción se divide en N_h x N_h regiones cuadradas compuestas por N_p x N_p píxeles cada una. Después, se calcula un histograma de orientaciones en cada región. Un histograma de orientaciones es definido como un histograma de direcciones del vector gradiente compuesto por N_d bloques o '*bins*' donde la contribución de cada '*bin*' es el valor de la magnitud del vector gradiente. Para evitar que el descriptor varíe abruptamente se aplica una interpolación trilineal.

El descriptor, 'DV_{hist}', se forma concatenando el '*bin*' de dirección ('phase') de todos los histogramas de orientaciones por filas. La longitud del descriptor es de

$$L_{DV} = N_h \times N_h \times N_d.$$

$$DV_{hist} = \left[Hist_{(1,1)}(1), Hist_{(1,1)}(2), \ldots Hist_{(r,s)}(i), \ldots, Hist_{(N_h,N_h)}(N_d)\right];$$

$$0 < i < N_d, 0 < j < N_h$$

donde $Hist_{(r,s)}(i)$ es el i-ésimo '*bin*' del histograma de orientaciones en las coordenadas (r,s).

Puntos singulares en fotogramas consecutivos, SP_{fin}^{n} y SP_{fin}^{n+1}, son emparejados mediante la distancia Euclidea calculada en el dominio de los descriptores. Considerando pequeños desplazamientos el emparejamiento puede ser restingido a puntos singulares cuya distancia entre ellos sea menor a un umbral, 'Th_{dist}'.

$$Match\left(\overrightarrow{p_i} \in SP_{fin}^{n}\right) = \min_{\overrightarrow{p_j} \in SP_{p_j}^{n+1}} \left\| DV_{fin}(\overrightarrow{p_i}) - DV_{fin}(\overrightarrow{p_j}) \right\|$$

donde || || es la distancia Euclidea y $SP_{p_j}^{n+1} = \left\{ p_j \in SP_{fin}^{n+1} \middle| \left\| \overrightarrow{p_i} - \overrightarrow{p_j} \right\| < Th_{dist} \right\}$.

Se comparan las distancias en el dominio de los descriptores entre el primer y el segundo mejor emparejamientos de un punto para determinar la fiabilidad de cada correspondencia. Una correspondencia correcta debe satisfacer la condición de $d_{FM} > 0.51\ dS_M$, donde d_{FM} y d_{FM} son la distancia Euclidea entre los correspondientes vectores descriptores del primer y el segundo mejor emparejamientos respectivamente. Las correspondencias que satisfacen las condiciones previas representan un campo vectorial disperso de movimiento, '*SVMF*'.

La estrategia propone el uso de tres '*look-up-tables*' diferentes para reducir el coste computacional. La primera de ellas se utiliza para calcular la magnitud del vector gradiente. Los índices de acceso son los valores horizontal y vertical del gradiente. La segunda '*look-up-table*'hace lo propio con la dirección del vector gradiente. La última de las '*look-up-table*' se utiliza para el cálculo de la interpolación lineal.

5. **Algoritmos propios basados en emparejamiento.**

Se han probado diferentes métodos de tracking basado en emparejamientos con la intención de combinar las ventajas de las diferentes características y descriptores estudiados. Los principales métodos combinados son dos:

a. **Método de tracking p-SIFT.**

Los puntos SIFT y sus descriptores son las características que mejor resultado han dado en el emparejamiento en las escenas que se han evaluado. Sin embargo computacionalmente son demasiado pesados para su utilización en tiempo real. Se ha probado un sistema basado en el cálculo de

unos puntos que llamamos p-SIFT pues están inspirados en ellos, pero eliminando parte de los cálculos más pesados.

El proceso completo para el cálculo de los puntos y sus descriptores es el siguiente:

- Se calcula la imagen media de 'I_0'. Se calcula una imagen en 'escala de grises' en la que cada píxel es la media de los 3 canales en ese píxel. Se calcula el valor medio de todos los píxeles de esta imagen, 'I_{0_mean}'.
- Se calcula una nueva imagen 'I_0^*' como la resta de 'I_0' y el valor medio: $I_0^* = I_0 - I_{0\_mean}$
- Se extraen las imágenes de gradientes en los ejes X e Y mediante filtros sobre 'I_0', obteniendo 'I_X' e 'I_Y'.
- Se calcula la imagen de magnitud de los gradientes como $I_{mag} = \sqrt{{I_X}^2 + {I_Y}^2}$
- Se calcula la imagen del ángulo de los gradientes como $I_{theta} = \tan^{-1}(I_Y, I_X)$
- Se calcula las imágenes de seno y coseno del ángulo de los gradientes como $I_{sin} = \sin(I_{theta})$ y $I_{cos} = \cos(I_{theta})$.
- Se divide el rango de ángulos máximo en '*num_angles*' partes con '*num_angles*' = 8 y se calculan las 8 imágenes de orientación como:

$$Sea\ i\ \in [0,7], angles(i) = \frac{360 * i}{8}, alpha = 9 \rightarrow$$

$$I_{orientacion_i} = \left[\left(I_{cos} * \cos\left(angles(i)\right) + I_{sin} * \sin\left(angles(i)\right) \right)^{alpha} \right] * I_{mag}$$

- Se realiza una convolución con un kernel como el de la Figura 36:

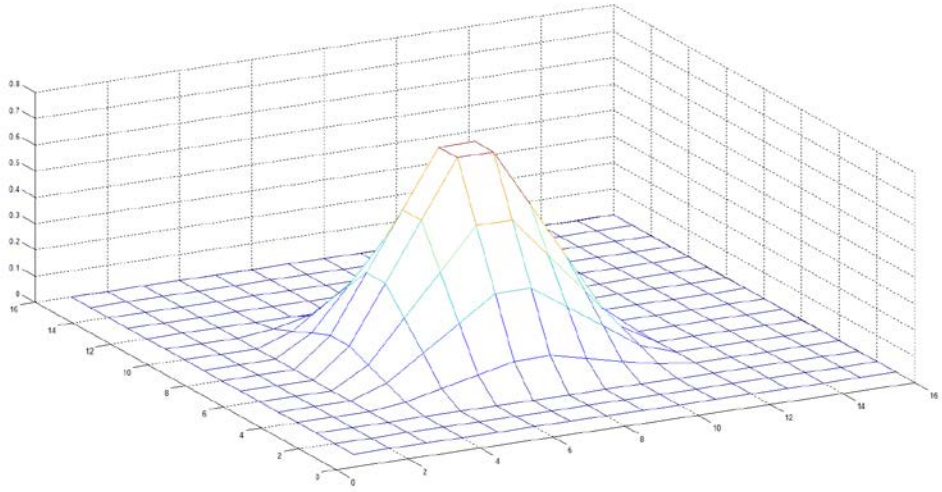


Figura 37: Kernel de convolución para p-SIFT.

- Para cada punto característico, se va a componer su descriptor. El descriptor es una palabra de 128 elementos y para componerlo se van a tomar los valores de las 8 imágenes de ángulos en 16 posiciones concretas. Estas posiciones se disponen en un patch de 16 x 16 en torno al punto característico en cuestión. La posición concreta de los 16 píxeles se muestra en la Figura 37.

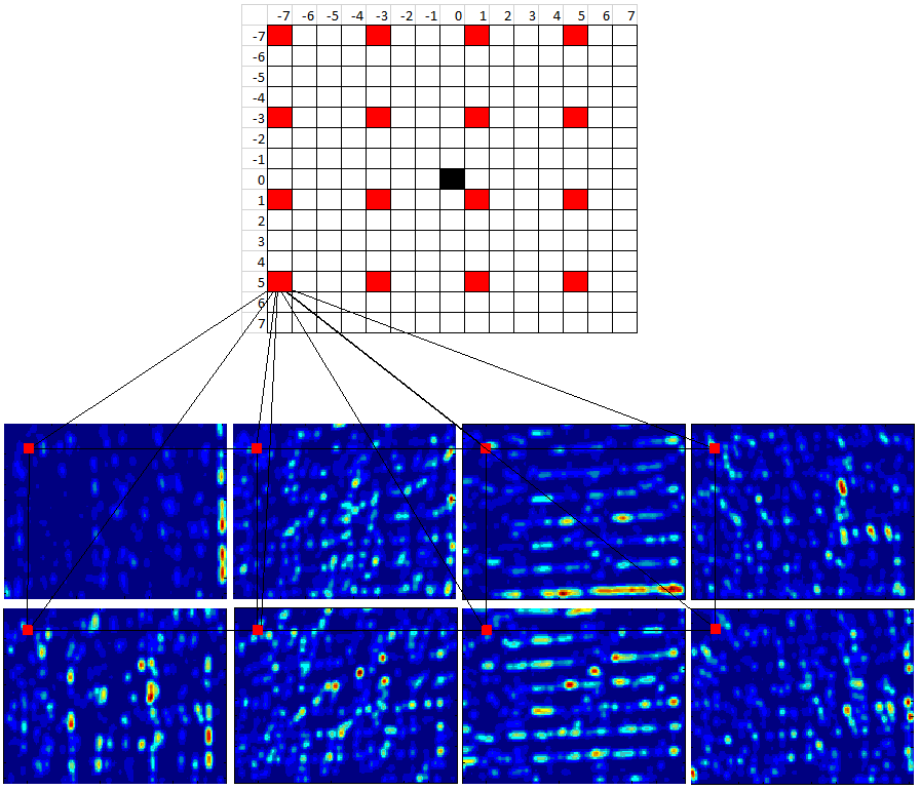


Figura 38: Posiciones de los 16 puntos (en rojo) en torno al punto de interés (en negro) de donde se extraen los valores de las 8 imágenes de ángulos.

- Por último, se normalizan los valores de los descriptores.

  Durante el proceso de emparejamiento se comparan los descriptores de los puntos '*train*' con los puntos '*query*' del fotograma actual. Los puntos '*train*' en este caso serán los puntos que se hayan extraído en la 'I_R' al inicializar o algunos concretos que se hayan entrenado en las tareas previas. Los puntos '*query*' serán los que buscamos iterativamente en cada fotograma mediante el siguiente proceso:
- Se toma la posición del punto en el instante anterior y se predice su nueva posición sumándole la media de los desplazamientos de todos los puntos de los N fotogramas anteriores (en nuestro caso N = 4).
- Se toma un patch de 5x5 en torno al punto corregido y se calculan los p-SIFT en las 25 posiciones.
- De cada una de estas 25 posiciones se toma el descriptor $desc_{query_{i,j}}, i,j \in [1,5]$ y se extrae una matriz de notas de la comparación, B_{5x5}, con el descriptor del punto '*train*', $desc_{train}$:

  $\forall i,j \in [1,5]$,

  $$B(i,j) = \sum_{k=1}^{128} \left(\sqrt{desc_{train}(k)} * \sqrt{desc_{query_{i,j}}(k)} \right)$$
- De los 25 resultados, se toma el más alto. Si se corresponde con la posición central del patch de 5x5, significa que la predicción es correcta y se ha encontrado el mejor punto 'query' correspondiente al 'train'. Si no se corresponde con el punto central del patch, se repite el proceso tomando como nueva posición central del patch de 5x5 el punto que mejor nota a dado.
- Este proceso de búsqueda del máximo local recibe el nombre de "*hill climbing*", y se establece un máximo de 10 iteraciones para asegurar la condición de finalización.
- Una vez que se dispone de la correspondencia, se analiza la nota obtenida. Se toma como válida la correspondencia si la nota es superior a 0.95. Si la correspondencia es válida, se actualiza la posición del punto y su desplazamiento se contará para el cálculo de la media del fotograma siguiente. Si no es válida la posición que se predice para este

punto se calcula sumando la media de los desplazamientos a la posición anterior de este punto.

b. **Método de tracking de puntos FAST y emparejamiento con medida de similaridad SSD**.

Este método pretende combinar la ligereza computacional del cálculo de puntos FAST con un sistema más robusto de emparejamiento de puntos que los descriptores basados en FAST. El proceso de extracción de las coordenadas características son las propias del método FAST y explicadas en el Anexo 1.

La diferencia con otros métodos como el ORB o el FFME es el método de construcción de los descriptores y su emparejamiento que, aunque se hace por fuerza bruta igualmente, la medida de similaridad es diferente.

Con la intención de mejorar la robustez de las características FAST frente a cambios de iluminación o escala se construyen unos descriptores basados en la '*Zero-Mean Normalized Sum of Square Difference*', '*ZM-SSD*', utilizados en proyectos como el PTAM (http://www.robots.ox.ac.uk/~gk/PTAM/) [33]. Se basan en la medida de similaridad '*SSD*'. Esta medida es sensible a los cambios de iluminación existentes entre las dos imágenes, por lo que se plantea dos mejoras a la medida '*SSD*':

- Restar la media de las dos imágenes para eliminar el brillo.
- Normalizar por la varianza para eliminar el contraste.

$$\mathrm{ZM-SSD} = \sum_{u,v}\left(\frac{(I_1(u,v)-\bar{I_1})}{\sqrt{\sum_{u,v}(I_1(u,v)-\bar{I_1})^2}} - \frac{(I_2(u,v)-\bar{I_2})}{\sqrt{\sum_{u,v}(I_2(u,v)-\bar{I_2})^2}}\right)^2$$

Ecuación 5: Zero-Mean Normalized Sum of Square Difference.

Los cálculos para la construcción de descriptores '*ZM-SSD*' son los siguientes (ver Figura 38):

- Para construir el descriptor '*ZM-SSD*' de un pixel determinado se toma un patch de nP x nP pixels a su alrededor, en este caso de 3 x 3 pixels.

- A este patch se le aplica la fórmula de la Ecuación 5 en cada uno de los elementos de la matriz 3x3 donde como $\overline{I_1} o \overline{I_2}$ se toma la media de la imagen completa, no solo del patch.
- Se guardan los datos del descriptor para ser comparados con los de otro descriptor cuando sea necesario.

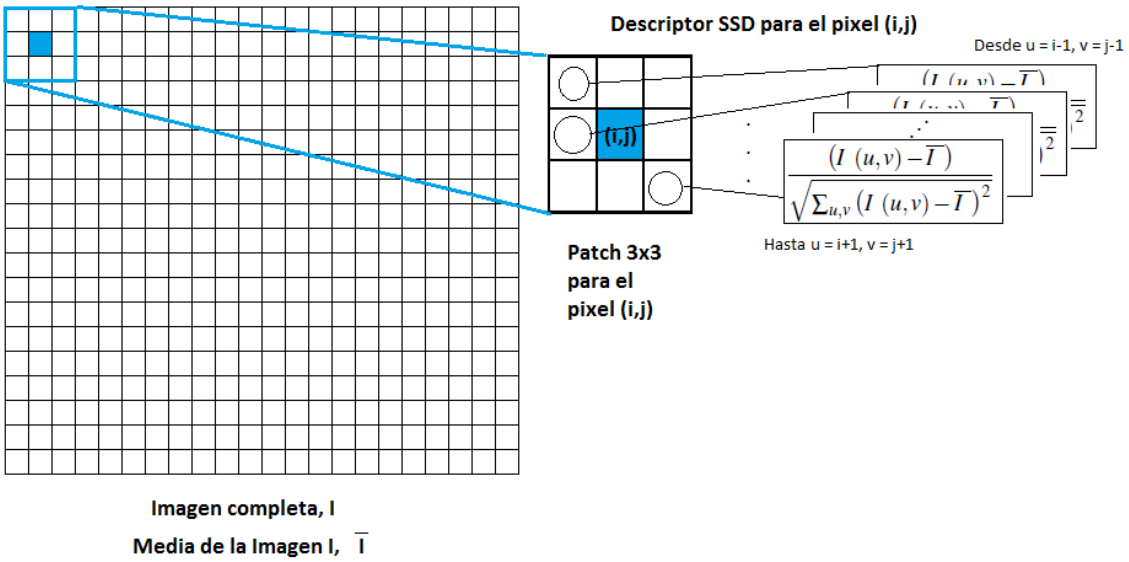


Figura 39: Construcción del descriptor SSD de un pixel.

En este caso, el emparejamiento se realiza de forma diferente a los métodos SIFT y SURF. Cuando se necesita hacer el emparejamiento entre un punto '*query*' y un punto '*train*' se efectúan las siguientes operaciones:

- Se calcula el descriptor del punto actual, '*query*' (o se recupera si ya estaba calculado anteriormente).
- Se hace la suma cuadrática acumulada de la diferencia ('*SSD*') de cada uno de los datos del patch:

$$d_{query-train} = \sum_{u,v} \left(p(u,v)_{query} - p(u,v)_{train}\right)^2$$

donde p(u,v) es el valor de cada uno de los elementos del patch de 3x3 píxeles en torno al punto característico '*query*' o '*train*'.

- El emparejamiento por fuerza bruta, implica realizar esta operación para cada uno de los puntos característicos contra todos los demás. El

emparejamiento del punto '*query*' se realiza con aquel '*train*' con el cual la distancia sea menor.

A pesar de que la caracterización '*ZM-SSD*' se supone que es robusta a cambios de iluminación y contraste, en un emparejamiento completo de todas las características de la 'I_R' con una imagen cualquiera de una secuencia se observa un resultado muy pobre cuando se toman como 'I_R' e 'I_0' del emparejamiento las imágenes de situaciones con distinta escala o iluminación (ver Figura 39). En situaciones de similares condiciones del entorno el resultado es correcto (ver Figura 40).

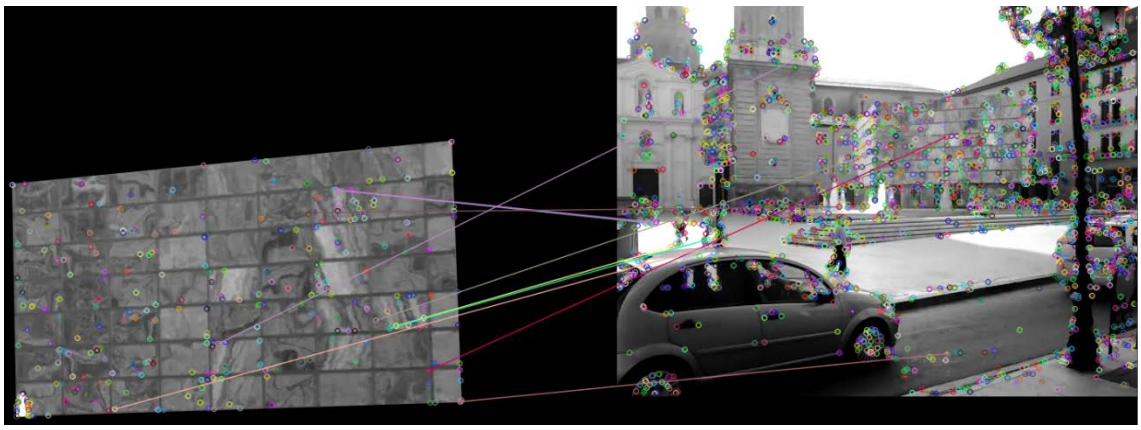


Figura 40: Emparejamiento de I_R e I_0 en situaciones con diferente iluminación y escala. Método FAST + ZM-SSD.

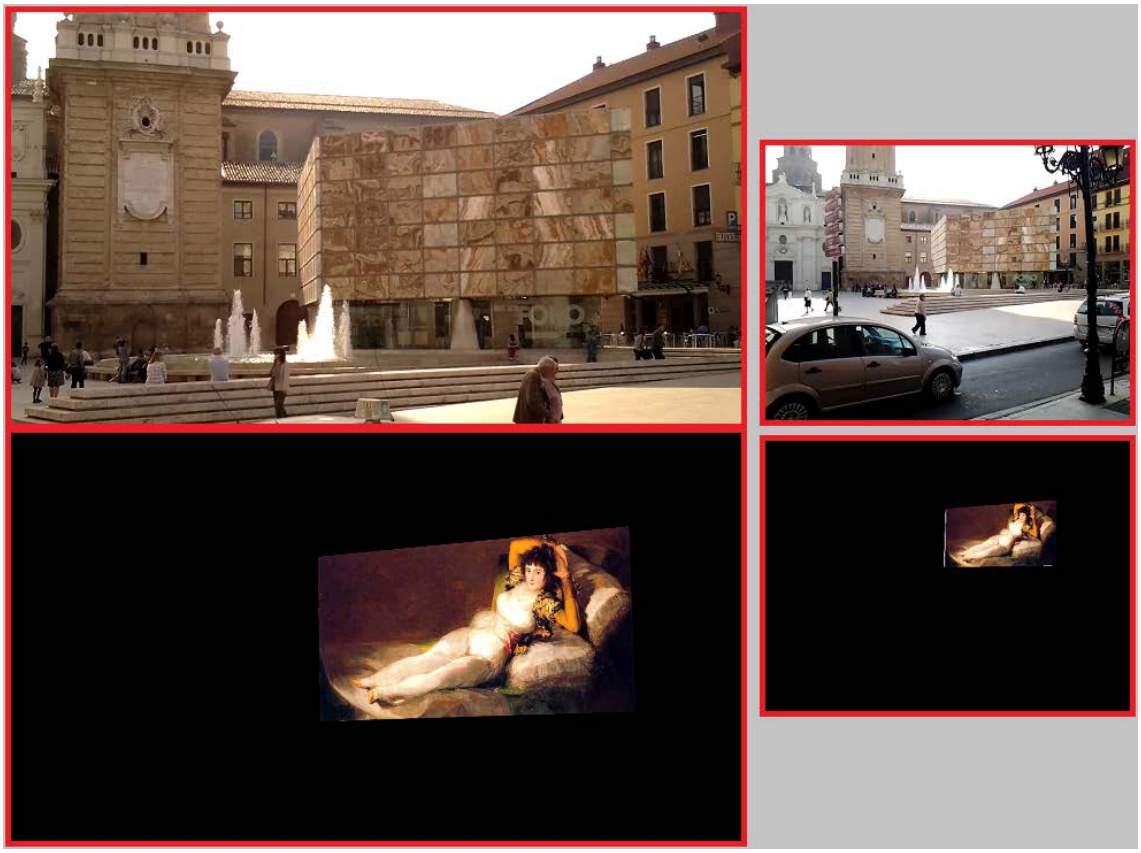


Figura 41: Emparejamiento de I_R e I_0 en situaciones con similares condiciones de iluminación y escala. Método FAST + ZM-SSD.

En vista de los resultados pobres de un emparejamiento inicial basado en SSD, y con la misma idea de aprovechar el bajo coste computacional de las características FAST intentando mejorar la robustez de las mismas frente a cambios de iluminación o escala, se ha desarrollado esta metodología. La aplicación del método tiene varias fases:

- Se realiza un primer emparejamiento de la 'I_R' y el 'I_0' de la secuencia mediante características SIFT y emparejamiento robusto de sus descriptores.
- Con este primer registro de imágenes, ya se puede determinar, a través de la homografía correspondiente, una región de búsqueda en 'I_0' ('ROI_{I0}'). Esta región se corresponderá con la zona ocupada por la imagen que se quiere proyectar sobre la fachada ('I_P') una vez aplicada la homografía 'H_{R_0}' y proyectada sobre 'I_0'.
- Sobre la 'ROI_{I0}' se buscan en la 'I_0' los puntos característicos FAST ('*puntos_ini*') y sus descriptores basados en SSD y se emparejan con los FAST que se detectan sobre la 'I_R' ('*puntos_ref*').

- Estos puntos emparejados correspondientes a la 'I_R' son los puntos que inicialmente van a ser seguidos en cada uno de los fotogramas de la secuencia y posteriormente actualizados con los de los siguientes fotogramas.
- Para el emparejamiento en cada fotograma, se extraen FAST y descriptores SSD ('*puntos_next*') y se utiliza fuerza bruta para emparejar los FAST detectados con los del fotograma anterior (inicialmente los '*puntos_ref*' y posteriormente los '*puntos_prev*').
- Antes de cada nuevo fotograma los '*puntos_next*' pasan a ser los '*puntos_prev*' y la 'ROI_{I0}' se actualiza con la nueva homografía calculada.

  Los resultados obtenidos no se comportan demasiado bien frente a oclusiones, tal como se muestra en la Figura 41.Además, el resultado es más lento del que se hubiese esperado. Esto parece ser debido al emparejamiento completo de todos los descriptores. Se descarta su utilización por requisitos de tiempo de procesado.

Figura 42: Emparejamiento Inicial mediante SIFT. Tracking entre frames mediante extracción de puntos FAST + ZM-SSD.

Anexo 4: Base de datos, Métodos y Resultados

Base de Datos: Características y tipos de distractores.

Para este proyecto se ha dispuesto de una base de datos de 37 videos del entorno del Cubo de la Seo grabados en diferentes condiciones. En la lista siguiente se muestran dichas condiciones de grabación las cuales influyen en el funcionamiento del sistema:

1. Tres resoluciones diferentes:
    a. R1[ancho, alto] de 1280 x 720. De esta resolución se dispone de 7 videos.
    b. R2[ancho, alto] de 640 x 480. De esta resolución se dispone de 29 videos.
    c. R3[ancho, alto] de 720 x 480. De esta resolución se dispone de 1 video.
2. Ocho elementos distractores distintos:
    a. Inicialización Parcial (*IP*). Videos en los que la superficie de proyección no aparece inicialmente o aparece solo en parte. Se dispone de 8 ejemplos.
    b. Iluminación elevada o escasa (*IE* o *IS*). En estos videos la imagen se encuentra con mucha o muy poca iluminación. Se dispone de 11 ejemplos.
    c. Movimiento rápido (*MR*). Se producen movimientos rápidos o bruscos de la grabación. Se dispone de 8 ejemplos.
    d. Cambios de Iluminación (*CI*). Durante el video la iluminación cambia significativamente. Se dispone de 5 ejemplos.
    e. Cambio de escala (*CE*). En este caso se distinguen dos distractores diferentes.
        i. *CE1*. Durante la grabación del video el tamaño de la superficie de proyección cambia. Esto es debido a que el observador se acerca o se aleja. Se dispone de 1 ejemplos.
        ii. *CE2*. El tamaño de las imágenes del video es diferente al tamaño de la '*Ir*'. Se dispone de 27 ejemplos.

f. Cambio de dirección (*CD*). Durante la grabación el observador se mueve en una dirección y posteriormente en otra diferente. Se dispone de 3 ejemplos.

g. Rotación de la Imagen (*RI*). Durante la grabación se produce un movimiento de la cámara en torno a la línea de visión, lo que se traduce en una rotación de la superficie de proyección. Se dispone de 6 ejemplos.

h. Oclusiones (*OC*). Uno o varios elementos, objetos o personas pasan por delante de la línea de visión del observador ocluyendo la superficie de proyección en diferentes grados. Se dispone de 10 ejemplos.

En la siguiente tabla (Tabla 1) se reflejan cuántos y cuáles de los videos contienen los distractores indicados y en qué grado. Se marca con una 'X' si el video contiene el distractor indicado y con 'XX' si este distractor es especialmente marcado.

Nombre del video						CE				
	Resolución	IP	IE/IS	MR	CI	CE1	CE2	CD	RI	OC
20131026_114604	1280x720		X					X		
20131026_114633	1280x720									
20131026_114647	1280x720				X					
20131026_114701	1280x720	X		X						
20131026_114708	1280x720	X								
20131026_114721	1280x720	X						X		
20131026_114730	1280x720									
20131026_114828	640x480						X			
20131026_114841	640x480						X			
20131026_114849	640x480			X			X			
20131026_114858	640x480	X					X		X	
20131026_115006	640x480						X			X
20131026_115020	640x480			X			X			
20131026_115053	640x480									
20131026_115106	640x480									
CAM00010	640x480		X				X			
CAM00011	640x480	X	X				X			
CAM00012	640x480		X				X			
CAM00013	640x480		X				X			
CAM000105	640x480		X			X	X			
CAM000106	640x480						X		X	
CAM000107	640x480			X			X		X	
CAM000108	640x480	X		X			X			
CAM000109	640x480				X		X			
CAM000110	640x480		X	X	X		X			X
CAM000111	640x480		X	X	X		X		X	X

CAM000112	640x480						X			X
CAM000113	640x480						X		X	X
CAM000114	640x480						X			X
CAM000115	640x480						X			XX
CAM000116	640x480	X	X	X	X		X	X		XX
CAM000117	640x480		X				X			
CAM000118	640x480		X				X			
CAM000119	640x480						X			X
CAM000120	640x480						X		X	
CAM000121	640x480						X			XX
VID-20131027-WA0000	720x480	X								
Nombre del video	**Resolución**	**IP**	**IE/IS**	**MR**	**CI**	**CE1**	**CE2**	**CD**	**RI**	**OC**
						CE				

Lista 1: Base de datos de videos.

Métodos evaluados.

Los métodos elegidos para evaluar nuestro sistema son cuatro:

1. Método de tracking por flujo óptico de Lucas-Kanade, KLT. Este método en bucle abierto, utiliza únicamente como método de seguimiento el algoritmo de Lucas-Kanade. Al no disponer de una realimentación de las posiciones, el sistema es bastante inestable con resultados pobres en presencia de distractores.
2. Método de tracking KLT asistido esporádicamente con un reetiquetado completo, REETIQ. Se trata de nuevo de un sistema en bucle abierto, sin realimentación. Es una modificación del anterior en el que, cada cierto número de fotogramas (10 por defecto), el tracking KLT se sustituye por un reetiquetado completo de la imagen. Es decir, cada cierto número de fotogramas, se realiza una búsqueda de puntos SIFT en la imagen y se reemparejan con los puntos SIFT de la '*Ir*' obtenidos en la secuencia de inicialización. Dado que el método de emparejamiento de SIFT es el que mejores resultados ha demostrado, este sistema es muy fiable y capaz de recuperarse de muchos distractores. Idealmente, reetiquetando en todos los fotogramas sería el método más estable de todos ellos. El coste computacional de este emparejamiento hace inviable su uso en sistemas de tiempo real. Es por este motivo por el que se realiza cada cierto número de fotogramas, aumentando considerablemente el tiempo de cómputo en estos instantes concretos. Los resultados son muy buenos, pero no

es aplicable a tiempo real al menos sin algún método que permita reducir el tiempo de proceso en los fotogramas de reetiquetado.

3. Método de tracking KLT con corrección de errores mediante reproyección sobre la '*Ir*', REPROY. Este método incorpora un lazo de realimentación de las coordenadas. Cada cierto número de fotogramas (10 por defecto) las posiciones de los puntos obtenidas por el tracking KLT se reproyectan sobre la '*Ir*'. En este sistema de coordenadas, se calcula la distancia euclídea media de todos los puntos reproyectados con sus homólogos extraídos inicialmente y se clasifican como no válidos aquellos que superan un umbral.
4. Método de tracking KLT con corrección de errores mediante reproyección sobre la '*Ir*' y filtrado de puntos en base a la varianza de los desplazamientos, REPROY+FILTRO. Este método también incorpora un lazo de realimentación de las coordenadas. Cada cierto número de fotogramas (10 por defecto) las posiciones de los puntos obtenidas por el tracking KLT se reproyectan sobre la '*Ir*'. En este sistema de coordenadas, se calcula el desplazamiento medio de todos los puntos reproyectados con sus homólogos extraídos inicialmente en ambos ejes X e Y. Se calcula media y varianza de ambos datos, se toma un porcentaje de confianza del 95% y se clasifican como no válidos aquellos cuyo desplazamiento en alguno de los dos ejes no se encuentra en este margen de confianza. Es el método que se ha explicado en el Capítulo 2.

Tipos de resultados.

Para evaluar el sistema se han clasificado los resultados obtenidos en 3 categorías:

1. Resultado correcto (*OK*). Se califica como resultado OK, aquellos casos en los que el sistema ha sido capaz de realizar correctamente la proyección de la imagen durante toda la duración del video y sin vibraciones visualmente molestas.
2. Resultado sistema se recupera (*SR*). Se califica como resultado SR, aquellos casos en los que el sistema ha sufrido dificultades como pueden ser pérdida momentánea del seguimiento, o vibraciones molestas de la proyección, pero ha sido capaz de recuperar el seguimiento y la proyección.
3. Resultado sistema sufre una pérdida completa (*KO*). Se califica como resultado KO, aquellos casos en los que el sistema ha sufrido dificultades como pueden ser pérdida momentánea del seguimiento, o vibraciones molestas de la

proyección, tan severas que ha sido incapaz de recuperar el seguimiento y la proyección produciéndose una pérdida completa.

En la Tabla 2 se muestran el tipo de resultado obtenido en cada uno de los videos en función del método utilizado:

	Método			
Nombre del video	KLT	REETIQ	REPROY	REPROY+FILTRO
20131026_114604	SR	OK	OK	OK
20131026_114633	OK	OK	OK	OK
20131026_114647	KO	OK	KO	OK
20131026_114701	KO	OK	KO	OK
20131026_114708	SR	OK	SR	OK
20131026_114721	OK	OK	OK	OK
20131026_114730	OK	OK	OK	OK
20131026_114828	OK	OK	OK	OK
20131026_114841	OK	OK	OK	OK
20131026_114849	SR	OK	OK	OK
20131026_114858	SR	OK	SR	OK
20131026_115006	KO	OK	SR	OK
20131026_115020	OK	OK	OK	OK
20131026_115053	OK	OK	OK	OK
20131026_115106	OK	OK	OK	OK
CAM00010	KO	KO	KO	KO
CAM00011	KO	OK	KO	OK
CAM00012	OK	OK	OK	OK
CAM00013	OK	OK	OK	OK
CAM00105	SR	SR	SR	OK
CAM00106	SR	OK	OK	OK
CAM00107	OK	OK	SR	OK
CAM00108	KO	KO	KO	OK
CAM00109	KO	KO	KO	OK
CAM00110	KO	KO	KO	KO
CAM00111	KO	KO	KO	KO
CAM00112	KO	KO	KO	KO
CAM00113	KO	KO	SR	OK
CAM00114	KO	SR	KO	OK
CAM00115	KO	KO	KO	KO
CAM00116	KO	KO	KO	KO
CAM00117	KO	SR	KO	OK
CAM00118	KO	KO	KO	KO
CAM00119	KO	OK	SR	OK
CAM00120	OK	OK	KO	OK
CAM00121	KO	KO	KO	OK
VID-20131027-WA0000	KO	KO	KO	OK

Tabla 2: Tipo de resultado obtenido por cada video en función del método utilizado en el sistema.