Proyecto Fin de Grado
Grado en Ingeniería Electrónia, Robótica y Mecatrónica

Sistema de seguimiento de objetos usando OpenCv, ArUco y Filtro de Kalman extendido

Autor: Rafael Jiménez Bravo
Tutores: José Ramiro Martínez de Dios
Alejandro Suárez Fernández-Miranda

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Proyecto Fin de Grado
Grado en Ingeniería Electrónia, Robótica y Mecatrónica

Sistema de seguimiento de objetos usando OpenCv, ArUco y Filtro de Kalman extendido

Autor:

Rafael Jiménez Bravo

Tutores:

José Ramiro Martínez de Dios

Alejandro Suárez Fernández-Miranda

Dpto Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Proyecto Fin de Carrera: Sistema de seguimiento de objetos usando OpenCv, ArUco y Filtro de Kalman extendido

Autor: Rafael Jiménez Bravo

Tutores: José Ramiro Martínez de Dios

Alejandro Suárez Fernández-Miranda

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El secretario del Tribunal

A mi familia

A mi pareja

A mis compañeros

Agradecimientos

Quisiera expresar mi mas sentido agradecimiento a todas aquellas personas que me han ayudado a alcanzar el objetivo. En primer lugar, a mi familia, que me ha ofrecido su apoyo incondicional desde el primer momento, y sin cuyo esfuerzo no podría haber llegado tan lejos. A mis amigos, por esos despropósitos capaces de relajar los momentos de mayor presión. A mi pareja, sostén fundamental, por soportarme en mis peores momentos y aguantarme en los mejores. A mi tutor, por la paciencia que ha demostrado conmigo. A mis profesores, por compartir sus conocimientos.
Por último, agradecer los buenos momentos compartidos con los distintos compañeros de clase con los que he convivido durante este largo camino, con los cuales, hasta en los momentos más difíciles sacábamos hueco para echarnos unas risas.

Rafael Jiménez Bravo

Sevilla, 2018

Resumen

En este proyecto se presenta el desarrollo y validación experimental de un sistema cuyo objetivo principal es la estimación de posición de un objeto móvil mediante visión, lo que involucra la detección y seguimiento del mismo, garantizando cierta tolerancia a cambios de luminosidad. El equipo que conforma dicho sistema consta de dos cámaras y un marcador. La implementación de este consta de un módulo de localización, un bloque de posicionamiento de cámaras y un módulo estimador (filtro de Kalman). El primero de ellos engloba la detección, el seguimiento del objeto y la redetección en caso de pérdida. El segundo de ellos permite conocer la posición de las distintas cámaras respecto a un sistema de referencia global, dado por un marcador de la librería ArUco. El último bloque, implementa el filtro de Kalman extendido de los datos obtenidos por ambas cámaras, proporcionándonos la posición del objetivo en cada instante de tiempo. El proyecto se ha desarrollado en C++ haciendo uso de las librerías OpenCV y ArUco. Los resultados obtenidos proporcionan la posición del objetivo con una elevada exactitud, con un error inferior a los 3 cm en cada eje del sistema de referencia global.

Abstract

This project presents the development and experimental validation of a system whose main objective is the estimation of the position of a mobile object through vision, which involves the detection and monitoring of it, guaranteeing a certain tolerance to luminosity changes. The equipment that allow the correct operation is constituted by two cameras and a marker. The implementation of this consists of a location module, a camera positioning block and an estimator module (Extended Kalman filter). The first one includes detection, object tracking and redetection in case of loss. The second one, allows to know the position of the different cameras with respect to a global reference system, given by a marker from the ArUco library. The last block, implements the extended Kalman filter of the data obtained by both cameras, providing the position of the objective.

The project has been developed in C ++ using OpenCV and ArUco libraries. The results obtained provide the position of the objective with a high accuracy, with an error of less than 3 cm in each axis of the global reference system.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

RGB	Red, Green, Blue (Rojo, Verde ,Azul)
CMY	Cyan, Magenta, Yellow (Cian, Magenta y Amarillo)
HSV	Hue, Saturation, Value (Matiz, Saturación, Valor)
m_{ij}	Momento de orden i,j
X_c	Componente x del centro de gravedad del objetivo
Y_c	Componente y del centro de gravedad del objetivo
$\mathcal{N}(\mu, \sigma)$	Normal de media μ y varianza σ
sin	Función seno
cos	Función coseno
$\boldsymbol{R}$	Matriz de rotación de Euler
×	Producto vectorial
$\{E\}$	Sistema de referencia de la Tierra.
${}^{E}\boldsymbol{r_T}$	Posición del objetivo respecto al sistema de referencia {E}
${}^{E}\boldsymbol{r_{CAMj}}$	Posición de la cámara respeto al sistema de referencia {E}
${}^{CAMj}\boldsymbol{r_T}$	Posición del objetivo respecto del sistema de referencia de la cámara
${}^{E}\boldsymbol{R_{CAM}}$	Rotación del sistema de referencia de la cámara respecto al principal
${}^{CAMj}\boldsymbol{r_{T,i}}$	Componente i-ésima de la posición del objetivo respecto del sistema de referencia de la cámara

1 INTRODUCCIÓN

Los científicos estudian el mundo tal como es; los ingenieros crean el mundo que nunca ha sido

- Theodore Von Karman-

La evolución de la electrónica y las comunicaciones en las últimas décadas ha permitido que las tecnologías de la información se instauren en diferentes ámbitos de la vida cotidiana. En esta línea, una parte importante de los esfuerzos de los investigadores en los últimos años se ha centrado en la creación de "entornos inteligentes" en los que los usuarios pueden interaccionar de manera natural con los diferentes sistemas y servicios computacionales que les facilitan la realización de sus tareas diarias. Las aplicaciones de los espacios inteligentes son numerosas y se encuentran en auge en la actualidad, abarcando diferentes ámbitos tales como la robótica de servicios, tareas de vigilancia automática, ayudas a personas con discapacidad o los servicios domóticos.

En este trabajo, se considera un espacio inteligente a un área física dotada de cámaras, ubicadas en posiciones fijas, que son controladas por un sistema de supervisión dotado de capacidad de análisis y toma de decisiones.

Una de las tareas esenciales en los espacios inteligentes, cuando la actuación se realiza mediante robots, es la localización de los mismos. Existen diferentes alternativas que permiten la localización de los robots móviles usando cámaras externas. Las propuestas más importantes pueden dividirse en dos grupos dependiendo del conocimiento a priori de los robots que requieren. El primer grupo incluye las técnicas que precisan de un gran conocimiento previo de los robots, y usan marcas artificiales a bordo de los mismos . Por otro lado, el segundo grupo en el que se encuadra este trabajo incluye las alternativas que emplean únicamente la apariencia natural de los robots y la geometría de las cámaras para el posicionamiento.

1.1. Motivación y objetivo

Actualmente es posible encontrar una amplia variedad de aplicaciones de vehículos multirotor, típicamente quadrotors o hexarotors. Este sistema de vehículo aéreo no tripulado (UAVS / RPAS / VANT) ofrece un gran número de posibilidades en muchos sectores, tanto comerciales como en materia de seguridad.

Si bien desde hace algunas décadas las aeronaves no tripuladas han sido motivo de interés, en particular en el ámbito militar, no ha sido hasta los últimos años que han pasado de sistemas experimentales a equipos aptos para su uso profesional. Su actual capacidad de desarrollar misiones reales se ha visto difundida no solo en los ámbitos restringidos de los investigadores, fabricantes o usuarios afines a esta tecnología, sino que también ha sido dada a conocer, por diferentes medios , a la opinión pública general, que comienza a conocer su existencia

y utilidad.

El uso de multirotors ha supuesto una gran mejora en muchos puestos de trabajo, ya que se han cambiado las formas de realizar tareas que eran complicadas o de difícil acceso. Una excelente herramienta de trabajo que supone una revolución tecnológica y que avanza a pasos agigantados para ofrecer nuevas prestaciones.

Para conseguir el correcto funcionamiento de estos en las diversas tareas a las que se deseen destinar, es necesario llevar a cabo multitud de experimentos y pruebas. Para ello es necesario contar con un sistema de seguimiento que permita conocer la posición de dicho dispositivo en cada instante de tiempo. En muchas aplicaciones es necesario conocer la posición del dron en el espacio para poder controlarlo, por ejemplo, en las maniobras de despegue y aterrizaje. El uso de sensores GPS está restringido principalmente a exteriores, en zonas de cobertura, y además ,el error de posicionamiento puede oscilar algunos metros. Por ello, resulta interesante el desarrollo de un sistema de posicionamiento para drones ad-hoc, que se pueda desplegar rápidamente para dar servicio de posicionamiento en un área determinada. Por último, el precio de estos sistemas no es asequibles para cualquier empresa o particular. Por lo tanto, la intención de este proyecto es desarrollar un sistema de bajo coste, que nos permita conocer la posición de un objeto móvil en cada instante de tiempo, de forma que asegure su correcto funcionamiento ante cambios de luminosidad en el entorno.

1.2. Estructura de este documento

El resto del documento se organiza de la siguiente forma. Constará de siete capítulos independientemente de la introducción. En el primero de ellos, denominado *Métodos para la estimación de posición basada en visión*, se describen los fundamentos teóricos necesarios para el desarrollo de este sistema. Se detallarán conceptos generales sobre el tratamiento de imágenes. También, se presentará el filtro de Kalman extendido (EKF) y su aplicación a la estimación de posición basada en visión. En este capítulo, también se describirá el algoritmo primitivo proporionado por la librería OpenCV a partir del cual se ha diseñado el descrito en este documento. Por último, se detallarán las librerías empleadas.

Tras esta descripción de conceptos básicos del capítulo 2, se expondrán los modulos de los que consta el sistema diseñado. Se especificará el funcionamiento de cada uno de los módulos y las distintas transiciones existentes entre ellos.

En el capítulo 4, se desarrolla la explicación de como se ha resuelto los problemas de detección, seguimiento y redetección del objetivo tras pérdida.

Tras detallar los distintos módulos del algoritmo de seguimiento, se definirá la solución adoptada para establecer un sistema de referencia global, y conocer así, la posición de cada cámara respecto a este.

En el sexto capítulo, se explicará cómo se obtiene la posición del objetivo respecto al sistema de referencia global, a partir de la posición de las cámaras y la localización del objetivo determinada por el algoritmo de tracking en cada una de ellas.

En el penúltimo capítulo, se muestran los resultados obtenidos en los diversos experimentos realizados, presentando las conclusiones en el capítulo 8, en el que también se recogen los posibles trabajos futuros.

1.3. Equipo

Para poder conocer la posición del objetivo, es necesario contar con un equipo específico que permita desarrollar cada uno de los bloques de los que consta el sistema. Los elementos de los que consta éste son los siguientes:

1.1.1 Cámara Logitech HD Webcam C525

Se trata de una pequeña cámara con una resolución HD 720p, con enfoque automático de gama alta. Se ha empleado este módelo por tratarse de una cámara con una buena relación calidad-precio.A pesar de que se podría haber elegido otros modelos con una mayor resolución, que permitiesen obtener una mayor precisión y reducir las pérdidas del objetivo en condiciones de baja luminosidad, se ha decidido utilizar esta cámara porque sus características permiten realizar el seguimiento del objetivo en la mayoría de los casos.

Figura 1-1. Logitech C525

1.1.2 Cámara Logitech HD Webcam C920

Se trata de un modelo más avanzado que el anterior. Está cámara puede alcanzar una resolución de hasta 1080p. Además, cuenta con enfoque automático al igual que el modelo anterior, una característica fundamental para el seguimiento de un objetivo móvil. Esta cámara tiene un precio superior, no obstante, es necesario contar con una cámara que proporcione imágenes con una mayor calidad para evitar la pérdida del objetivo en condiciones pésimas de luminosidad.

Figura 1-2. Logitech C920

1.1.3 Marcador ArUco

Dicho elemento permite conocer la orientación y posición relativa de cada una de las cámaras respecto a este. La librería ArUco permite interpretar este tag como un sistema de referencia, de forma que calibrando cada una de las cámaras es posible conocer la posición de cada de ellas.

Existen numerosos tipos de marcadores, cada uno de ellos pertenecen a un diccionario. ArUco proporciona su propio diccionario. El diseño de un diccionario es importante ya que la idea es que sus marcadores deben de ser tan diferentes como sea posible.

El diccionario aconsejado a usar por ArUco es ARUCO_MIP_36h12. Por lo tanto, el primer paso es descargar dicho diccionario e imprimir un marcador en papel.

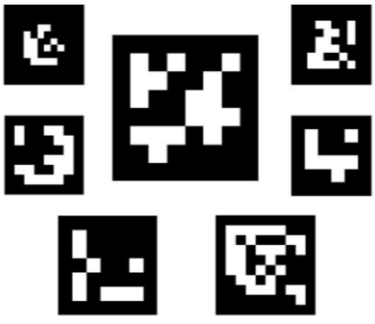


Figura 1-3. Marcador ArUco

1.1.4 Prototipo del Sistema

El sistema ideado para conocer la posición de un objetivo se basa en realizar el seguimiento de este en cada una de las cámaras. El módulo de seguimiento será el encargado de proporcionar el pixel en el que se encuentra el objetivo para cada cámara. Posteriormente, conociendo la posición y orientación de cada una de ellas gracias al marcador de ArUco, será posible estimar la posición del móvil aplicando el filtro de Kalman Extendido. Estos pasos se ejecutan siguiendo el orden que aparece en la siguiente figura.

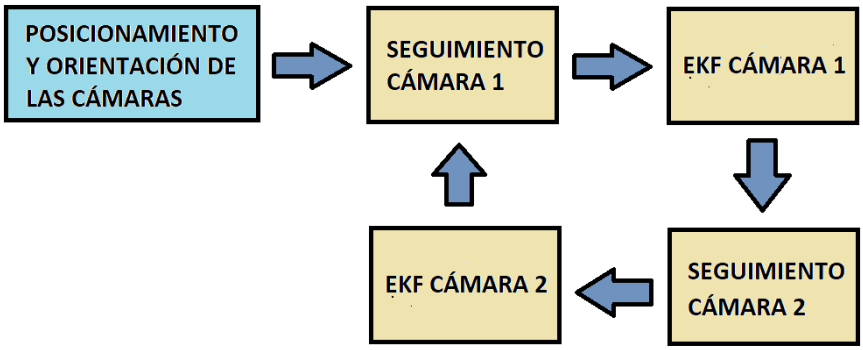


Figura 1-4. Idea fundamental implementada

Es posible observar como el módulo de posicionamiento solo se ejecuta al comienzo, por lo que las cámaras no podrán desplazarse de la posición inicial. Tras finalizar este proceso, se ejecutan de forma cíclica los bloques que implementan el seguimiento y el filtro de Kalman Extendido para cada una de las cámaras tal y como se puede apreciar en la Figura 1-4.

En la Figura 1-5 se puede observar un prototipo del sistema ideado:

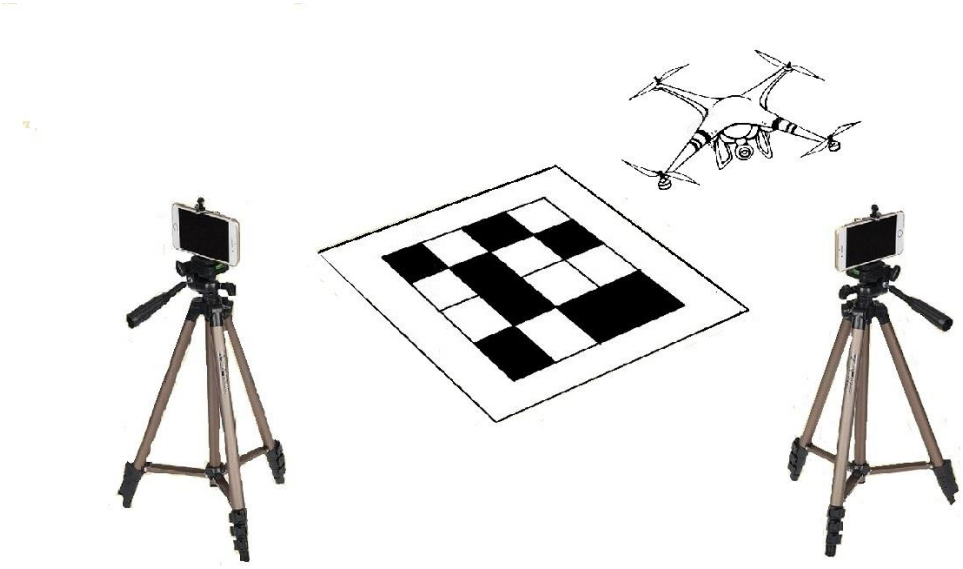


Figura 1-6. Prototipo del equipo empleado

Por último, resaltar que el marcardor de ArUco se debe de encontrar dentro del rango de visión de ambás cámaras.

2 MÉTODOS PARA LA ESTIMACIÓN DE POSICIÓN BASADA EN VISIÓN

En este capítulo se presentan los fundamentos teóricos necesarios para poder desarrollar este proyecto. Se comenzará mostrando los conceptos fundamentales vinculados al tratamiento de imágenes, se continuará con una breve explicación de la librería OpenCv y el algoritmo utilizado para conseguir el seguimiento del objetivo. Tras esto, se explicará brevemente la librería ArUco y, por último, se describirá el Filtro de Kalman Extendido.

2.1 Conceptos generales

Una imagen digital se define como una matriz de tamaño MxN en la que cada elemento, denominado píxel, discretiza una región del espacio. Para hacer referencia a cada punto de una imagen, se suele expresar mediante una componente, *i*, que haga referencia a la fila de la matriz a la que pertenece dicho pixel, y otra componente, *j*, que haga referencia a la columna a la que pertenece. Así, cada píxel de una imagen, *Img,* cualquiera se puede expresar como $Img(i, j)$ o $Img_{i,j}$.

Normalmente se trabajar con un rango de 8 bits de información por cada punto, lo que equivale a un total de 2^8 valores distintos que podrán representar el nivel de intensidad de un píxel en el intervalo [0, 255]. El valor 0 representa la ausencia de intensidad (lo que se correspondería con un píxel negro en la imagen), al incrementar este valor aparecen colores grisáceos cada vez más claros hasta llegar al valor 255, para el cual el pixel represetará la máxima intensidad posible (píxel blanco).

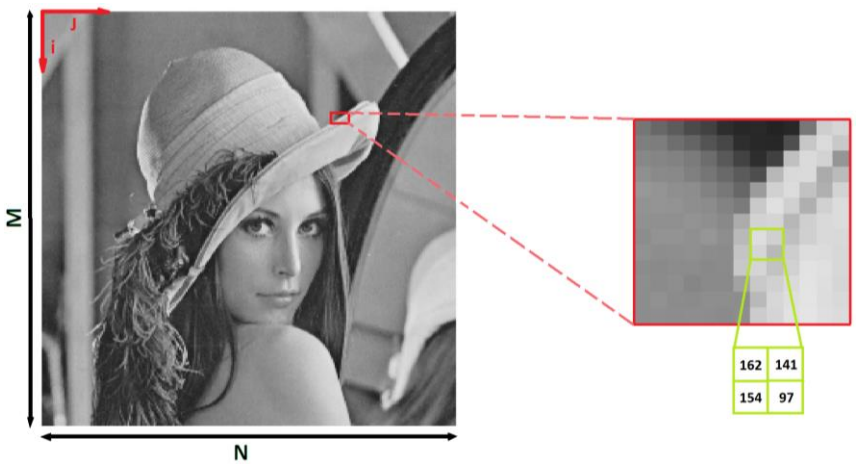


Figura 2-1. Conceptos generales imagen

2.1.1 Color

La luz blanca solar se compone de radiaciones de diversa longitud de onda, a cada una de las cuales corresponde un determinado color. Si se coloca un prisma de vidrio en la trayectoria de un rayo solar, se pueden observar en los rayos refractados siete colores que se alinean uno tras otro. Las ondas cortas (violeta) experimentan una desviación mayor que las largas (rojo). Newton probó experimentalmente esta característica alrededor del 1666.

Figura 2-2. Diversas radiaciones que componen la luz blanca

Existen dos sistemas fundamentales de mezcla de color: Modelo aditivo y modelo sustractivo.

- Modelos aditivos

  Según la teoría de Thomas Young, en la retina disponemos de tres sensores para la percepción del color, cada uno de ellos, sensible, a una de estas tres longitudes de onda: rojo, verde y azul. Por lo que consideraremos estos como colores primarios. El resto de colores se obtienen mediante la suma de estos.

Figura 2-3. Modelos aditivos

- Modelos sustractivos

  Los pigmentos muestran el color absorbiendo las longitudes de onda de modo selectivo y reflejando las que no absorben. La mezcla de color pigmento se llama sustractiva porque cuantos más colores se mezclan, más se sustraen radiaciones, mostrando menos luminosidad.

  En este modelo los colores primarios son el cian, magenta y amarillo.

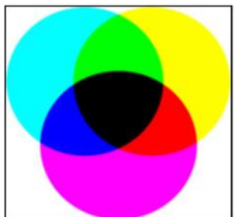


Figura 2-4. Modelos sustractivos

Para la representación del color se pueden usar varios modelos normalizados, siendo los más conocidos el *RGB*, el *CMY* y el *HSV*, que podemos ver en la siguiente figura:

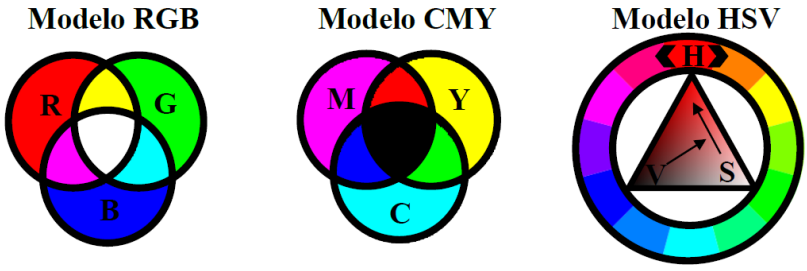


Figura 2-5. Modelos normalizados para la representación del color

El modelo RGB (del inglés, *Red, Green, Blue)* es un modelo aditivo que consigue nuevas tonalidades a partir de la mezcla de los haces primarios de luz. Las imágenes en este formato están contituidas por tres matrices, las cuales hacen referencia a un color primario, de forma que el color percibido de cada pixel se obtiene como la suma de los diversos canales de dicho pixel. Si cada pixel perteneciente a un determinado canal se representa mediante 8 bits, el color final que tome este puede tomar un color de entre 256·256·256 posibilidades. Por esta razón se suele llamar color verdadero, ya que abarca $16 \cdot 10^6$ colores, con variaciones difícil de apreciar.
El CMY (del inglés *Cyan*, *Magenta*, *Yellow*) es un modelo sustractivo, es decir, que se basa en la mezcla de pigmentos para crear nuevos colores, y se utiliza principalmente en funciones de impresión.
El modelo HSV (del inglés Hue, Saturation, Value) es una transformación no lineal del modelo RGB en coordenadas cilíndricas de manera que cada color viene definido por las siguientes dimensiones:

- Tono o matiz: es el angulo que representa el matiz, normalmente definido entre 0º y 360º.

- Saturación: representa la saturación del color, dado entre 0 y 1, 0 representa sin saturación alguna (blanco), hasta 1 que sería el matiz en toda su intensidad. Es común también darlo en percentiles 0%-100%.

- Brillo: Nivel del brillo entre 0 y 1. 0 es negro; 1, blanco. Al igual que la saturación puede darse en porcientos entre 0% y 100%.

Este módelo será el empleado en el algoritmo encargado de llevar a cabo el seguimiento del objeto, ya que este modelo permite agrupar las diferentes tonalidades de color, lo cual difiere al modelo RGB, donde los colores no están necesariamente tan agrupados.

2.1.2 Histograma

Una forma usual de representar la frecuencia con la que aparece cada tono de gris en la imagen es mediante su histograma, que no es más que un gráfico donde se representa cada nivel de intensidad frente al número de píxeles que tienen dicha intensidad. Los histogramas pueden ser muy útiles porque proporcionan información general de la imagen sin necesidad de tenerla presente, permitiendo identificar algún posible inconveniente global como problemas de brillo o de contraste, de los que hablaremos en las siguientes subsecciones. También es posible analizar su forma para extraer información de más alto nivel o para solucionar el problema de la separación de regiones. En este proyecto se trabajará con histogramas tanto en la etapa de seguimiento del objeto(CAMShift), como en la fase de redetección.

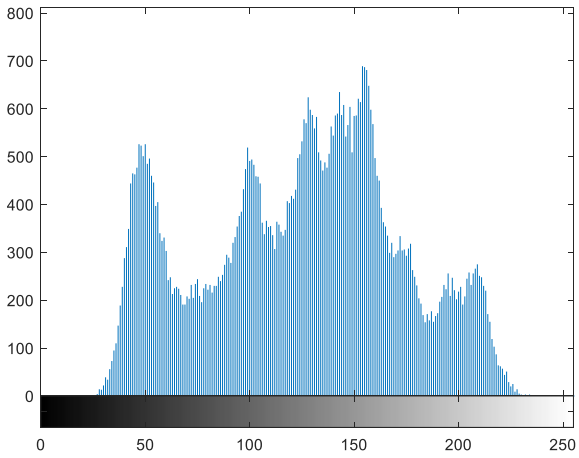


Figura 2-6. Ejemplo histograma

2.2 OpenCV

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicaciones de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

Open CV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estérea y visión robótica.

El proyecto pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado realizando su programación en código C y C++ optimizados, aprovechando además las capacidades que proveen los procesadores multinúcleo.

Figura 2-7. Logo OpenCV

OpenCV tiene una estructura modular. Los módulos principales de OpenCV son:

- core:

Este es el módulo básico de OpenCV. Incluye las estructuras de datos básicas y las funciones básicas de procesamiento de imágenes. Este módulo también es usado por otros módulos como highgui.

- highhui:

Este módulo provee interfaz de usuario, códecs de imagen y vídeo y capacidad para capturar imágenes y vídeo, además de otras capacidades como la de capturar eventos del ratón.

- imgproc:

Este módulo incluye algoritmos básicos de procesamiento de imágenes, incluyendo filtrado de imágenes y tranformación de imágenes.

- video:

Este módulo de análisis de vídeo incluye algoritmos de seguimiento de objetos, entre otros.

- objdetect:

Incluye algoritmos de detección y reconocimiento de objetos para objetos estándar.

2.3 CAMShift

CAMShift (Continously Adaptive Mean Shift) es el algoritmo utilizado para llevar a cabo el seguimiento del objetivo. Este algoritmo es una adaptación del algoritmo Mean Shift para poder tratar las distribuciones de probabilidad dinámicas (con cambios en su tamaño y su posición) que representan los objetos en movimiento. Todas las imágenes serán pasadas del modelo RGB al modelo HSV ya que se utilizará la componente de color (canal H del modelo HSV) para segmentar los objetos en el algoritmo CAMShift.

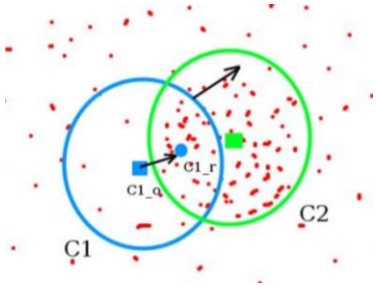


Figura 2-8. Algoritmo CAMShift

La aplicación del algoritmo CAMShift se resume de la siguiente forma:

- Paso 1: El usuario fija una ventana inicial que contenga el objeto deseado a seguir.
- Paso 2: Se calcula el histograma de la componente de hue sobre la ventana de búsqueda de la primera imagen. Se omiten los píxeles con valores bajos (S < 30) de saturación o luminancia (V < 10) ya que los valores de matiz correspondientes no son representativos. Los valores del histograma h(x) se escalarán para que se encuentren en el rango [0, 255]. De este modo, el histograma representará la distribución de probabilidad del color del objeto a seguir; es decir, la distribución de probabilidad objetivo.
- Paso 3: Para cada nueva imagen, calcular la retroproyección (back-projection) del histograma del paso 2. Esta operación consiste en generar una imagen en escala de grises donde cada píxel tendrá como intensidad el valor del histograma correspondiente al matiz de dicho pixel en la imagen procesada. Así, el valor de cada píxel de esta imagen identificará la probabilidad de que dicho píxel en la imagen procesada pertenezca al objeto.
- Paso 4: Calcular el centroide de la imagen de retroproyección mediante el algoritmo Meanshift. Este seguirá los siguientes pasos:
    - Calcula el centroide en la ventana de búsqueda actual. Este centroide viene dado por la siguiente expresión:

$$X_c = \frac{m_{10}}{m_{00}}, Y_c = \frac{m_{01}}{m_{00}} \quad (2\text{-}1)$$

      Siendo m_{00} el momento geométrico[1]de orden cero y m_{10} y m_{01} los momentos de orden uno.
    - Establecer el centroide obtenido como nuevo centro de la ventana de búsqueda
    - Repetir los dos pasos anteriores hasta la convergencia, es decir, hasta que el centro de la ventana se mueva menos de una cierta cota predeterminada o bien un número fijo máximo de iteraciones.
- Paso 5: Establecer una nueva ventana de búsqueda para las siguientes imágenes; utilizando como centro el centroide obtenido en el paso 4 y como tamaño, una función del momento de orden 0.
- Se repiten los pasos tres, cuatro y cinco para cada frame.

A continuación, se puede observar el diagrama de bloques de este algoritmo:

[1] Los momentos geométricos m_{kl} de una región(p) se definen de la siguiente forma: $m_{kl} = \sum_i \sum_j i^k \cdot j^l \cdot p(i,j)$

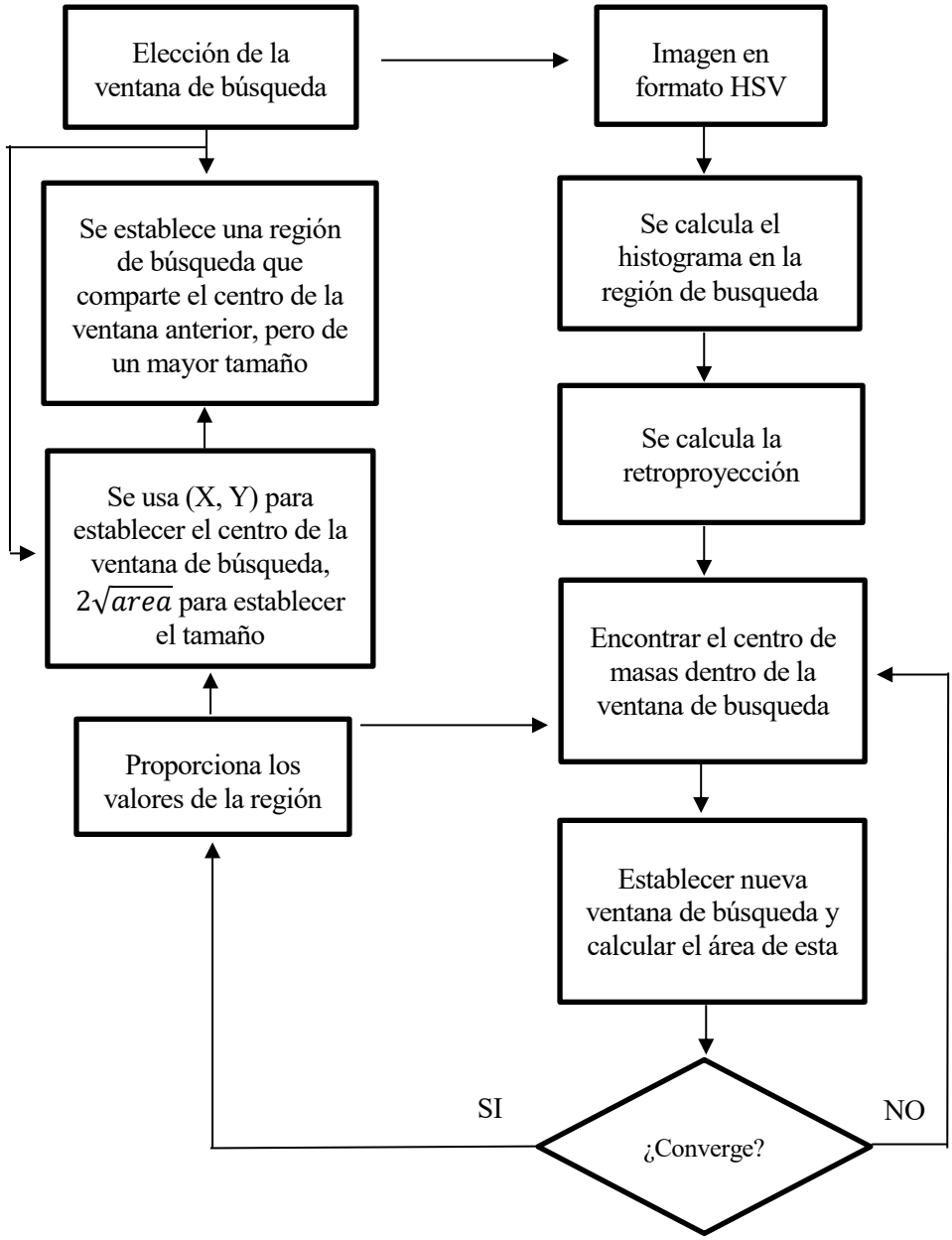


Figura 2-9. Diagrama de bloques del algoritmo CAMShift

2.4 ArUco

Aruco es una la librería de código abierto basada en OpenCV que permite detectar marcadores cuadrados de referencia en imágenes. Además, si la cámara esta calibrada es posible detectar la posición de la cámara respecto al marcador. La librería ha sido desarrollada empleando el lenguaje C++ . Existen multitud de tipos de marcadores, cada uno de ellos pertenecen a un diccionario. ArUco proporciona su propio diccionario optimizado.

El diccionario aconsejado para estimar la posición de las cámaras por ArUco es ARUCO_MIP_36h12.

2.4.1 Marcadores

Cada marcador esta delimitado por un borde exterior de color negro y una región interior a este que codifica un patrón binario. Este patrón binario es único e identifica cada marcador. Dependiendo del diccionario al que pertenezca dicho marcador, encontraremos marcadores con mas o menos bits. Cuantos más bits tenga el marcador menor será la posibilidad de confusión . Sin embargo, un mayor numero de bits significa disponer de una mayor resolución para una correcta detección.

Los marcadores se pueden usar como puntos de referencia 3D para estimación de posición de la cámara. Denotamos como s el tamaño del marcador una vez que está impreso en un trozo de papel. La siguiente imagen muestra el sistema de coordenadas empleado.

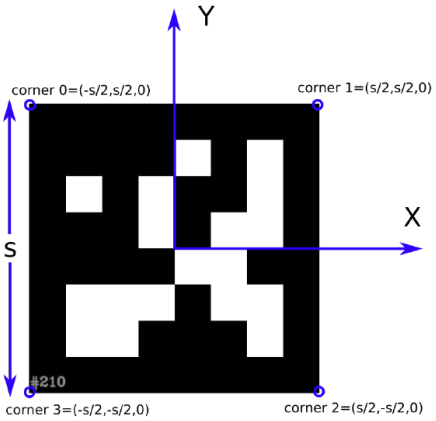


Figura 2-10. Ejemplo marcador estandar

Existe un tipo de marcadores denominados *enclosed markers*, que permiten una localización de las esquinas de los marcadores más precisa. Esto permite una estimación de la posición más precisa. No obstante, la detección de este tipo de marcadores es ligeramente más compleja que en los marcadores simples.

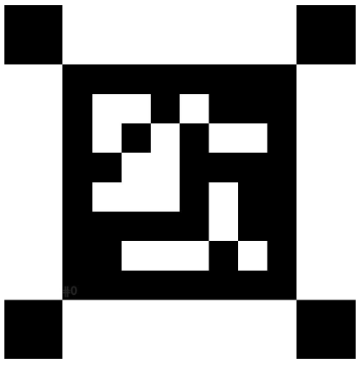


Figura 2-11. Ejemplo marcador cerrado

2.4.2 Proceso de detección

Es importante conocer el procedimiento que sigue ArUco para detectar estos marcadores. Este consta de una serie de pasos que se explican a continuación:

- En primer lugar, se aplica un umbral adaptativo hasta obtener los bordes del marcador, tal y como se puede observar en la imagen número 1 de la ilustración 14.
- Tras encontrar los bordes se buscan los contornos del marcador. Una vez encontrados, no solo se detectan los contornos del marcador, sino que se detectan gran cantidad de bordes no deseados. El resto de los pasos se llevan a cabo con el objetivo de eliminar aquellos contornos no deseados.
- El primer paso para acabar con los bordes no deseados es eliminar los bordes con un pequeño numero de puntos. El resultado se puede observar en la imagen número 2 de la ilustración 14.
- A continuación, se lleva a cabo una aproximación poligonal de los contornos encontrados y se mantiene aquellos con cuatro esquinas. Se enumeran las esquinas en sentido contrario a las agujas del reloj. El resultado se puede observar en la imagen número 3 de la ilustración 14.
- Se eliminan los rectángulos demasiados cercanos. El resultado se puede observar en la imagen número 4 de la ilustración 14.
- Identificador del marcador:
    - Se elimina la perspectiva de proyección para obtener una vista frontal del área rectangular usando una homografía.
    - Se emplea el método de Otsu para calcular el umbral óptimo que separá el marcador del resto. Los algoritmos de Otsu asumen una distribución bimodal y encuentra el umbral que maximiza la varianza extra-clase manteniendo una baja varianza dentro de la clase.
    - Identificación del código interno.
    - Para los marcadores válidos, refina las esquinas mediante la interpolación de subpíxeles.

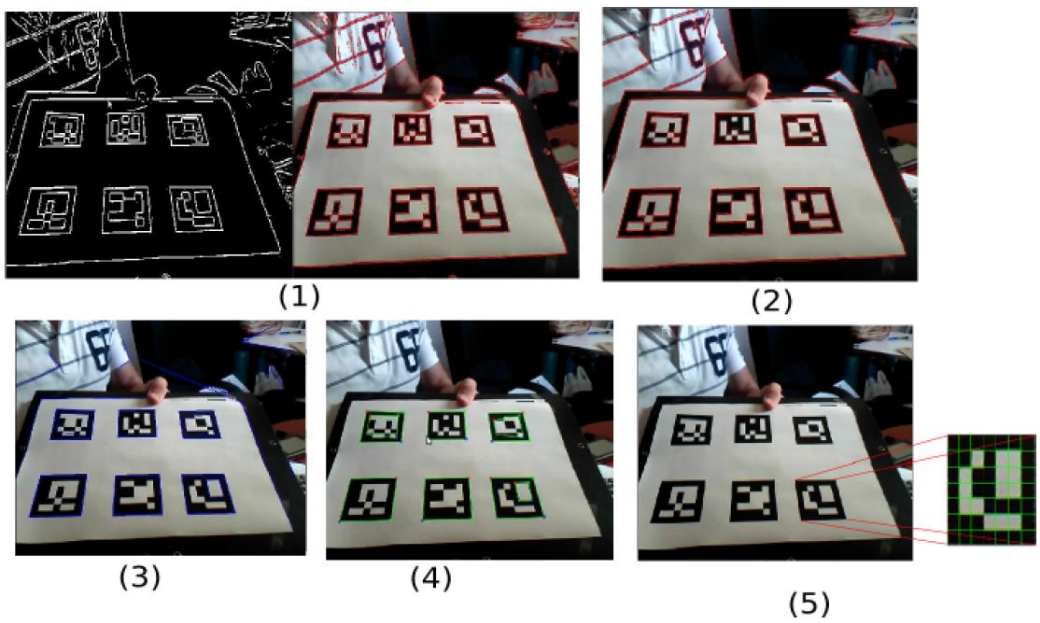


Figura 2-12. Fases ejecutadas para detectar los marcadores de ArUco

2.4.3 Aplicaciones

Las distintas tareas que se pueden llevar a cabo con esta librería son las siguientes:

- Calibración de cámaras a partir de una batería de imágenes tomadas sobre un tablero de calibración.

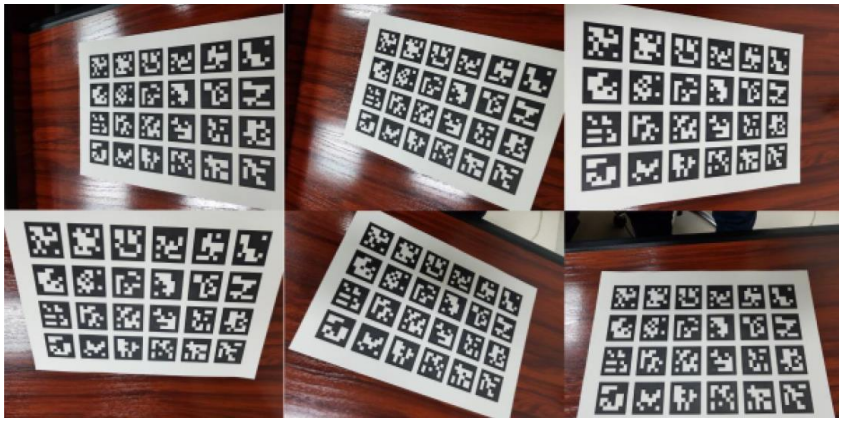


Figura 2-13. Imágenes tomadas para la calibración

- Estimación de la posición de una cámara respecto a un marcador. Para llevar a cabo esta tarea es necesario haber calibrado la cámara previamente.

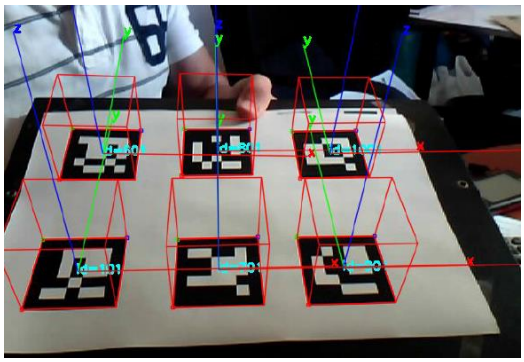


Figura 2-14. Estimación de la posición de una cámara

- Estimación de la posición en un mapa a partir de la visión obtenida por la cámara de cada marcador.

Figura 2-15. Estimación de la posición en un mapa

- Aplicaciones de realidad aumentada

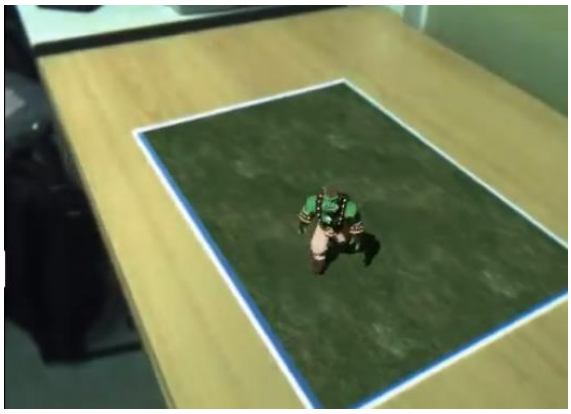


Figura 2-16. Ogre ArUco

2.5 Filtro de Kalman Extendido

2.5.1 Introducción

El filtro de Kalman, aparecido a principios de la década de los sesenta, es de una importancia comparable a los trabajos realizados por Nyquist y Bode en la década de los veinte y los de Wiener en los años treinta. El filtro de Kalman permite estimar en tiempo real el vector de estado de un sistema dinámico lineal a partir de medidas ruidosas indirectas que de él se van tomando. Estas estimaciones en tiempo real del estado del sistema son valiosas en sí mismas cuando el sistema opera en lazo abierto; pero considerando la posibilidad de operación en lazo cerrado, las mismas pueden ser utilizadas para sintetizar una acción de control adecuada que lleve el sistema al estado deseado. Esta última posibilidad es la que motiva nuestro interés en el filtro. En el contexto de la teoría de control lineal, el filtro de Kalman ocupa un lugar central como observador de estado óptimo y en este contexto está bien entendida su utilización. No obstante, cada vez con más frecuencia, se manejan modelos no-lineales en ámbitos de aplicación tan diversos como la ingeniería, biología, ecología y economía. El uso de estos modelos más complejos se ha visto estimulado por el desarrollo tecnológico de la computación, que mediante el cálculo numérico ha facilitado el análisis de los mismos y la implementación de métodos de control también no-lineales. Resulta entonces natural buscar algún método de estimación de estado en tiempo real, que pueda aplicarse a sistemas no-lineales. Es donde aparece el filtro de Kalman extendido. Destacar, que para implementarlo se ha acudido al tutorial desarrollado por Gabriel A. Terejanu.

2.5.2 Notación y modelo

Un sistema dinámico puede ser descrito por el modelo espacio estado de la siguiente manera:

$$x_k = f(x_{k-1}) + w_{k-1}, \quad w_{k-1} \sim \mathcal{N}(0, Q_t) \tag{2-2}$$

$$z_k = h(x_k) + v_k, \quad v_k \sim \mathcal{N}(0, R_t) \tag{2-3}$$

Las ecuaciones anteriores representan un sistema dinámico, donde:

- La función *f* es un operador de transición que mapea el espacio de estado dentro del mismo espacio de estado.
- La función *h* es un operador que mapea el espacio de estado dentro del espacio de observaciones.
- x_k denota al vector de estados desconocidos en un tiempo k.
- w_k es un error aleatorio de estimación del estado (incertidumbre en el modelo).
- z_k es el vector de observaciones.
- v_k es un error aleatorio de observación (incertidumbre en la medida).

2.5.3 Descripción del método

El estado inicial, x_0, es un vector aleatorio de media conocida $\mu_0 = E[x_0]$ y covarianza $P_0 = E[(x_0 - \mu_0)(x_0 - \mu_0)^T]$

El filtro de Kalman extendido (EKF) resuelve el problema de la estimación del estado x_k generado por un sistema no lineal, utilizando la expansión de la serie de Taylor que aproxima las ecuaciones no lineales de estado y de observación, sobre el valor actual estimado del estado ($\widehat{x_k}$); igualmente, proporciona una estimación de la varianza mínima del estado basado en la información estadística sobre el modelo. Se supone que el vector de ruidos es un proceso Gaussiano de media cero y matrices de varianza covarianzas dadas por:

$$\left(w_k w_k^T\right) = Q_t \ , E\left(v_k v_k^T\right) = R_t \tag{2-4}$$

$$E(w_k) = 0 \ , \ E(v_k) = 0 \tag{2-5}$$

$$E\left(w_k w_j^T\right) = 0 \ , E\left(v_k v_j^T\right) = 0, \ para\ k \ \neq j \tag{2-6}$$

$$E\left(w_k v_j^T\right) = 0, \qquad \text{para todo } k \text{ y } j \tag{2-7}$$

El proceso iterativo del filtro de Kalman se descompone en dos etapas: una primera fase de predicción del estado actual a partir del estado anterior y las ecuaciones dinámicas y una segunda fase de corrección de la predicción usando la observación del estado actual.

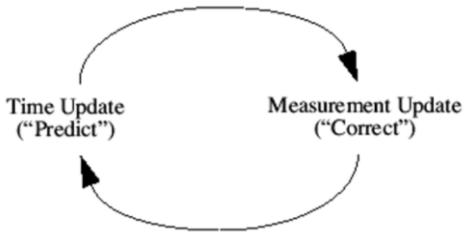


Figura 2-17. Fases de predicción y corrección

La fase de predicción lleva a cabo los siguientes pasos:

- Predice el vector de estados (x_k^f) a partir del vector de estados del instante anterior.

$$x_k^f \approx f(x_{k-1}^a) \tag{2-8}$$

- Predice la matriz de covarianza del error (P_k^f). Dicha matriz inicialmente se puede definir como la matriz identidad.

$$P_k^f = J_f(x_{k-1}^a) \cdot P_{k-1} \cdot J_f^T(x_{k-1}^a) + Q_{k-1} \tag{2-9}$$

siendo J_f el jacobiano de la función *f*, definido como:

$$J_f = \begin{pmatrix} \frac{df_1}{dx_1} & \frac{df_1}{dx_2} & \cdots & \frac{df_1}{dx_n} \\ \vdots & & \ddots & \vdots \\ \frac{df_n}{dx_1} & \frac{df_n}{dx_2} & \cdots & \frac{df_n}{dx_n} \end{pmatrix}$$

y Q_{k-1} una matriz de covarianza que modela el ruido asociado al modelo del sistema.

Una vez llevada a cabo la primera fase, se procede a corregir el vector de estados y la matriz de covarianza predichos. La etapa de corrección consiste en los pasos:

- La corrección comienza con el calculo de la ganancia, K_k .

$$K_k = P_k^f \cdot J_h^T(x_k^f) \cdot \left(J_h(x_k^f) \cdot P_k^f \cdot J_h^T(x_k^f) + R_k\right)^{-1} \quad (2\text{-}10)$$

  siendo J_h el jacobiano de la función *h*, definido como:

$$J_f = \begin{pmatrix} \frac{dh_1}{dx_1} & \frac{dh_1}{dx_2} & \cdots & \frac{dh_1}{dx_n} \\ \vdots & & \ddots & \vdots \\ \frac{dh_n}{dx_1} & \frac{dh_n}{dx_2} & \cdots & \frac{dh_n}{dx_n} \end{pmatrix}$$

  y R_k una matriz de covarianza que modela la incertidumbre asociada a las medidas.

- Se calcula el vector de estados corregido, X_k^a .

$$X_k^a = X_k^f + K_k \cdot (Z_k - h(x_K^f)) \quad (2\text{-}11)$$

- Por último, se corrige la matriz de covarianza del error, P_k , predicha en la primera fase.

$$P_k = (I - K_k \cdot J_h) \cdot P_k^f \quad (2\text{-}12)$$

3 DESCRIPCIÓN MODULAR DEL SISTEMA

En este capítulo se describirá a alto nivel como se ha resuelto el problema propuesto. Se presentará cada uno de los módulos de los que consta el sistema, se hará una pequeña descripción de cada uno y se definirán las distintas transiciones entre todos ellos. En los capítulos siguientes se describirá con mas detalle cada uno de ellos.

Tal y como se ha descrito anteriormente, el objetivo del proyecto es desarrollar un sistema que permita, tras haber seleccionado un objetivo, conocer la posición de este. Para ello se emplearán 2 cámaras que permitirán estimar la profundidad. Trabajar con dos cámaras hace que el esquema diseñado se complique, ya que hay que contemplar multitud de casos que aparecen cuando se trabajan con dos cámaras en tiempo real.

En primer lugar, es necesario conocer como se ha de lanzar el programa encargado de estimar la posición del objeto seleccionado. El comando encargado de lanzar el programa es el siguiente:

```
./EstimationPose live:1 -c <ruta archivo calibración cámara 1> live:2 -c
<ruta archivo calibración cámara 2> -s <tamaño marcador aruco en metros>
```

Tras lanzarlo aparecerán una serie de ventanas, que se irán abriendo y cerrando en función del bloque que se este ejecutando. En todo momento, el terminal mostrará toda la información relevante en el proceso. El sistema constará de 5 bloques principales: configuración, inicialización, tracking, redetección y estimación cooperativa de posición. A continuación, se profundizará en cada uno de ellos. En el siguiente diagrama se puede apreciar el orden de ejecución, así como los parámetros o ficheros necesarios en cada bloque.

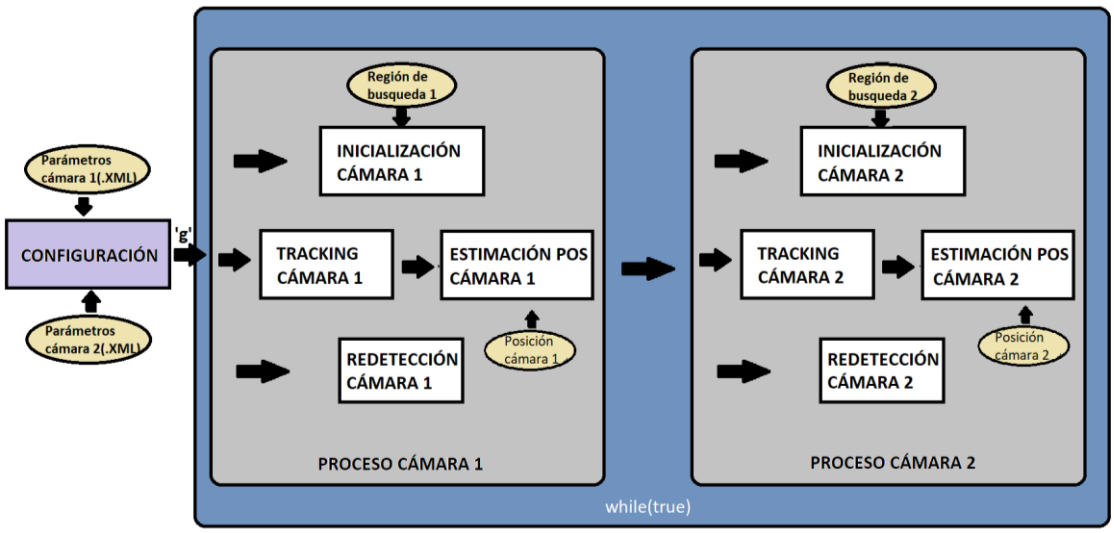


Figura 3-1. Secuencia de bloques

3.1 Configuración

Permite conocer la posición de cada una de las cámaras respecto al marcador de ArUco. Este bloque únicamente se ejecuta al principio del programa, por lo que si se modifica la posición de las cámaras habría que ejecutarlo de nuevo. El final de este modulo lo marcará la pulsación de la tecla *g,* dando lugar a la ejecución del módulo de inicialización. Para llevar a cabo este proceso de configuración, es necesario haber calibrado cada una de las cámaras con antelación. Concretamente, se deben de calibrar empleando el método facilitado por esta librería, la cual proporciona un fichero de extensión *.YML* donde se almacenan los resultados obtenidos. En la ejecución del programa se deberá de añadir como argumento la ruta a cada uno de los archivos de calibración. Durante el tiempo en el que se ejectura este bloque se mostrarán 4 ventanas. Dos de ellas mostrarán las imágenes capturadas por cada una de las cámaras en cada iteración, apreciando el sistema de coordenadas generado en el marcador de ArUco. Las dos restantes mostrarán los contornos percibidos por cada cámara tras haber ejecutado todos los pasos descritos en la sección 2.4.

Al pulsar la tecla *'g'* , se cerrán estas ventanas y aparecerán otras cuatro, dos de ellas mostrarán de nuevo las imágenes capturadas por cada una de las cámaras, mientras que las otras dos mostrarán cuanto se ha modificado la imagen actual respecto a la capturada en la iteración anterior.

3.2 Inicialización

Tras conocer la posición de cada cámara, el siguiente paso es seleccionar el objetivo en cada una de ellas. Para ello será necesario marcar una región en cada una de las ventanas denominadas *'Tracking'* correspondientes a cada cámara. Se ha de tener en cuenta que la selección de las dos regiones no es simultanea. Durante el tiempo que transcurre desde que se selecciona una región en una ventana, hasta que se selecciona la correspondiente en la otra, pueden ocurrir diversos sucesos que se han de considerar. Por ejemplo, es posible que durante este tiempo el objetivo seleccionado se pierda , por lo que habría que activar la etapa de redetección para esta ventana, mientras que la segunda se encontraría aun en dicha etapa de inicialización. Otro caso posible, es que, tras seleccionar el objetivo en una de ellas, se active el modulo de tracking, mientras que la segunda aun este esperando la selección correspondiente. Por lo tanto, es posible que cada cámara se encuentre en una determinada fase del sistema.

3.3 Tracking

El modulo encargado de realizar el seguimiento del objetivo es el módulo de tracking. Este constará de dos etapas, las cuales se explicarán con mayor profundidad en el siguiente capítulo. Como resumen, destacar que este bloque permite conocer la posición del objetivo combinando una versión avanzada del algoritmo CAMShift proporcionada por OpenCV y un algoritmo de extracción de fondo, desarrollado para eliminar aquellas regiones estáticas de la imagen para mejorar los resultados en exteriores. Tal y como ya se ha descrito en el apartado anterior, es posible que una cámara se encuentre ejecutando la primera fase del tracking, mientras que la otra ejecuta la segunda fase de este, o cualquier otro de los módulos(redetección o inicialización).

3.4 Redetección

Es posible que el objetivo seleccionado salga del rango de visión de la cámara o que las condiciones lumínicas en la zona en la que se encuentra sean tan deficientes que no permitan su apreciación, en estos casos, el módulo de tracking dará paso al modulo de redetección, encargado de encontrar de nuevo el objetivo perdido. Aunque el objetivo solo se haya perdido en una cámara, este bloque no empleará la información proporcionada por la otra. El algoritmo desarrollado se basa en momentos estadísticos y en una división en rejillas de la imagen obtenida. Además, emplea el algoritmo de extracción de fondo comentado en el apartado anterior. Tras localizar el objetivo en la cámara, de nuevo se activará el módulo de tracking. Este bloque se describirá con mas detalles en el siguiente capítulo.

3.5 Estimación cooperativa de posición

El bloque de tracking proporciona en todo momento las coordenadas del objetivo en píxeles, no obstante, el objetivo es conocer la posición de este respecto al sistema de referencia proporcionado por el marcador de ArUco. Con el fin de estimar la profundidad, se ha implementado el filtro de Kalman extendido, el cual permite fusionar la información obtenida por ambas cámaras. Para implementarlo se han de adquirir diversos conocimientos previos, como el método Pinhole de la cámara, las coordenadas homogéneas para tranformaciones proyecticas y el método conocido como *Rodrigues'rotation* para calcular rotaciones entre dos sistemas de referencia. Todo esto se recoge en el capítulo 5 de este documento. Resaltar que dicho módulo solo se ejecutará cuando en la iteración anterior el objeto se encuentre detectado en ambas cámaras, para evitar que el EKF integre medidas de una única cámara, ya que el error de estimación , perdiendose información de profundidad (caso estimación monocular).

4 SELECCIÓN Y DETECCIÓN, SEGUIMIENTO Y REDETECCIÓN DEL OBJETIVO

En este capítulo, se definirá el procedimiento seguido para conseguir realizar el segimiento de un objetivo seleccionado. Para lograr esto, se debe de identificar tres bloques fundamentales: un bloque de selección y deteción del objetivo, otro de seguimiento y, un último bloque de redetección.

Estos tres bloques se implementan en cada una de las cámaras, de forma que el módulo de redetección se activará siempre que se pierda el objetivo en alguna de las dos cámaras. Respecto al algoritmo implementado para realizar el seguimiento del objetivo , destacar que se ha desarrollado una evolución del algoritmo denominado CAMShift. La modificación realizada se basa en el almacenamiento de un histórico de histogramas, de forma que el algoritmo CAMShift se realiza sobre un histograma dinamico.

Los tres procesos comentados se implementarán mediante una máquina de estados siguiendo el siguiente esquema:

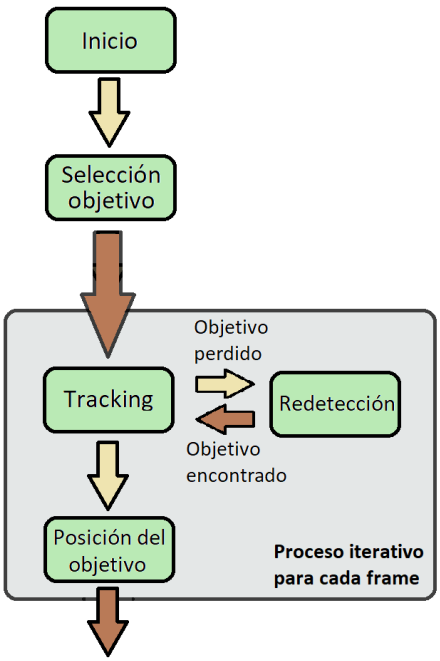


Figura 4-1. Esquema Tracking

Tras capturar una imagen y tratarla, se ejecuta una serie de instrucciones en función del estado en el que se encuentre el sistema. El tratamiento de la imagen comentado consiste en o siguiente:

En primer lugar, se cambia el formato d ela imagen capturada, transformando la imagen de formato RGB a HSV. Tras esto se construyen tres matrices correspondientes a cada uno de los canales del formato HSV(Hue, Saturation, Value). La primera se empleará para llevar a cabo el calculo del histograma, mientras que los otros dos canales servirán para definir la máscara, la que se despreciará aquellos valores de brillo y saturación para los cuales el algoritmo CAMShift no funciona correctamente.

Como se puede observar en la figura 4-1, tras seleccionar el objetivo comienza el proceso de seguimiento del móvil, proporcionando en cada iteración las coordenadas en píxeles del objetivo. Si en algún momento, hay indicios de que el objetivo ha salido del rango de visión se inicia el proceso de redetección, encargado de localizar al objetivo cuando aparezca de nuevo en el campo de visión de la cámara. Tras localizarlo de nuevo se ejecuta el bloque de tracking.

A continuación, se describirán cada uno de los bloques con un mayor nivel de detalle.

4.1 Selección y detección

Como se puede observar en la figura 4-2, el proceso comienza con el bloque de selección del objetivo. En este se selecciona la región en la que se encuentra el objeto a seguir. Para llevar a cabo esta selección se ha desarrollado la función *onMouse().* Esta se encarga de detectar el píxel en el que se pulsa cuando se comienza a seleccionar la región de interés, y detecta también el píxel final en el que se deja de pulsar, de forma que a partir de los dos puntos detectados se construye un rectángulo que contendrá el objeto del cual queremos obtener su posición.

Para que resulte más sencillo seleccionar el objetivo en movimiento, la región seleccionada se multiplica por una máscara, en la cual tendrán como valor 1 aquellos píxeles que hayan sufrido una variación respecto al instante anterior (imagen capturada en la iteración anterior). Esta máscara se calcula en la función *detectaMovimiento,* la cual recibe como argumentos la imagen actual y la anterior, y devuelve una plantilla como resultado, definida con valor 1 en aquellos píxeles en los que se haya detectado movimiento. En ella, primero se convierten ambas imágenes a formato blanco y negro, para que posteriormente se calcule la diferencia entre ambas matrices. Restando cada una de las componentes obtenemos información sobre cuanto ha variado la imagen capturada por la cámara respecto a la almacenada en el instante anterior, de esta forma, si el valor es distinto de cero se habrá producido cierta variación. Para eliminar posible ruido aparecido en la imagen se define un umbral de detección de movimiento, de forma que, si la diferencia entre los pixeles correspondientes a ambas imágenes supera dicho umbral, se considerará que se habrá producido movimiento, asignándole un 1 al píxel correspondiente en la plantilla resultante. Este proceso es conocido como **background detection**.

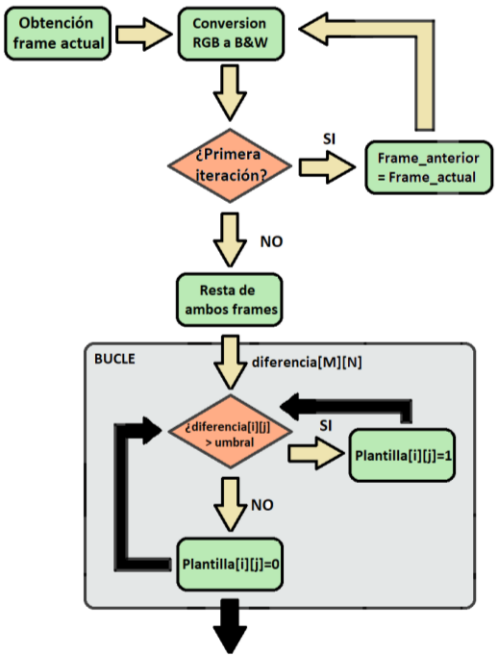


Figura 4-2. Background detection

Tras llevar a cabo la selección del objetivo comienza el proceso de seguimiento o tracking del objetivo.

Cabe destacar, que este proceso de extracción de fondo o detección de movimiento se lleva a cabo en cada iteración, independientemente del estado en el que se encuentre el sistema. Esto es debido a que se emplea en todos ellos para el cálculo de la mascara.

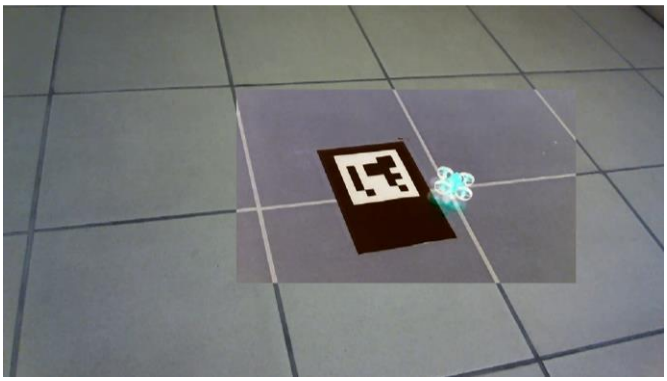


Figura 4-3. Selección objetivo

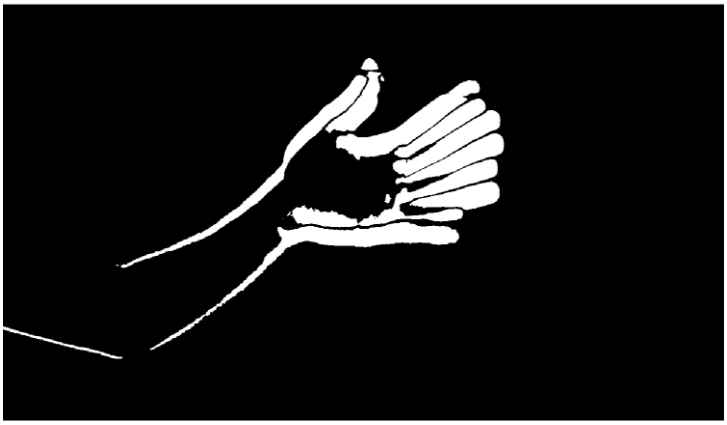


Figura 4-4. Resultado background-detection al mover un brazo

4.2 Seguimiento

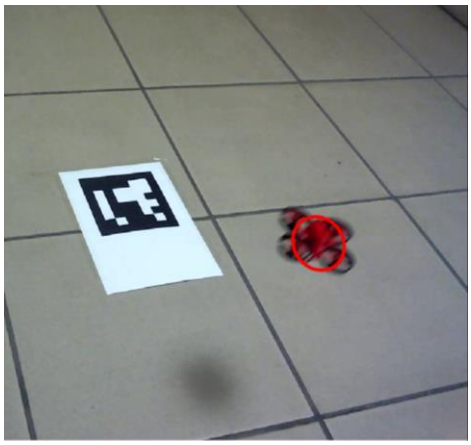


Figura 4-5. Seguimiento objetivo

Tras haber seleccionado el objetivo en la etapa anterior, se procede a llevar a cabo el tracking del objetivo. En esta etapa de seguimiento podemos distinguir dos fases: una fase de inicialización, llevada a cabo solo en la primera iteración, y otra de seguimiento dinámico basado en la acumulaión de histogramas de instantes anteriores. Si se recuerda, el algoritmo CAMShift proporcionado por la librería OpenCV, explicado en el capítulo 2, trabajaba con un histográma estático, es decir, se calculaba una sola vez tras haber seleccionado la región de interés, de forma que si se produciesen cambios en la luminosidad del ambiente, el objeto se perdería. Para obtener resultados robustos ante este tipo de cambios, se ha diseñado una mejora del algoritmo mencionado, en el que, en cada iteración se calcula el histográma correspondiente a la ventana de tracking obtenida en el instante anterior y se compara con un histórico de histogramas calculados en instantes anteriores. Dicho histórico de histogramas lo conforma un array de histogramas. En este, no se almacenarán los histogramas calculados en todas las iteraciones anteriores, sino solo aquellos cuya menor diferencia respecto a las diferentes componentes se encuentra entre dos umbrales (*umbralApilacion y umbralPerdida*). Si esta diferencia, es menor que el primer umbral preestablecido(*umbralApilacion*), no se incluirá como nueva componente, ya que no aporta grandes novedades respecto a la componente cuya diferencia es mínima. Si por el contrario supera a ambos umbrales(*umbralApilacion y umbralPerdida*), se considerará que el objetivo se ha perdido y comenzará el proceso de redetección.

3.2.1. Fase inicialización

Como se ha descrito anteriormente, esta fase se lleva a cabo únicamente en la primera iteración del algoritmo tras haber selecciónado un objetivo. Este módulo parte de la información proporcionada por la etapa de selección y detección, la cual proporciona: una región de interés, en la que se encuentra el objetivo, y una máscara indicadora de cuanto ha variado la imagen capturada respecto a la capturada en el instante anterior (background detection). A continuación, se decribirá los pasos seguidos para implementar esta fase de inicialización:

En primer lugar, se calcula el histograma de la región seleccionada haciendo uso de la función *calcHist()* proporcionada por OpenCV.

```
calcHist (region of interest, number of images, dimension, mask, Output histogram, histogram dimension, range)
```

Posteriormente, se normaliza el histograma calculado utilizando la función normalize(). Esta permite limitar el rango de valores del array calculado, de forma que se le asigna el valor α al menor de sus valores, y el valor β al mayor de ellos. El resto de los valores se obtienen siguiendo la proporcionalidad establecida.

```
normalize (input array, output array, α, β, NORM_MINMAX[2])
```

Tras normalizar, el histograma se apila en el vector descrito con anterioridad. En el código se ha denominado *histProm.*

El siguiente paso es calcular la retroproyección(back-projection) de la imagen capturada en el instante actual haciendo uso del histograma calculado. Esta operación consiste en generar una imagen en escala de grises donde cada píxel tendrá como intensidad el valor del histograma correspondiente al matiz de dicho píxel en la imagen

[2] Tipo de normalización. En este caso, sirve para asignar el valor α a la menor componente del array y β al mayor. Otros tipos vienen dados por las expresiones NORM_INF, NORM_L1 y NORM_L2. Se puede obtener información de ellas visitando la documentación proporcionada por OpenCv.

procesada. Así, el valor de cada píxel de esta imagen identificará la probabilidad de que dicho píxel en la imagen procesada pertenezca al objeto.

calcBackProject (hue channel, number of images, dimension, histogram, output matrix, range)

Tras haber calculado la retroproyección se ejecuta la función encargada de desarrollar el algoritmo CAMShift.

trackBox = CamShift (back-projection, tracking Window, stop criterion)

Por último, se modifica la posición de la región del interés, asignándole el valor proporcionado por la función anterior y se dibuja la elipse en la posición calculada.

Esta fase solo se ejecutará una vez tras haber seleccionado la ventana de búsqueda.

3.2.1. Fase seguimiento dinámico

Tras haber cumplido la etapa de inicicialización del bloque de seguimiento, se desarrolla la segunda fase de este, donde se lleva a cabo el proceso de comparación del histograma de la imagen actual con el histórico de histogramas, y la apilación de este en función de la diferencia respecto a ellos. Esta fase denominada seguimiento dinámico se desarrollará cíclicamente en cada imagen tomada por la cámara, siempre y cuando el objetivo se encuentre dentro del rango de visión.

El principio de funcionamiento de esta fase se basa en el calculo del histograma de la región en la que se encuentra el objetivo, calculada en el instante anterior. Para ello se hace uso de la máscara de movimiento generada en la primera fase, para descriminar aquellos pixeles en los que no haya movimiento. Tras calcular el histograma se lleva a cabo una comparación de este con cada componente que conforma el array en el que se encuentran almacenados los diferentes histogramas con los que se ha trabajado lo largo de la prueba. Se almacenará el valor más pequeño obtenido. En función de este, la acción ha realizar será una u otra:

- Si es inferior al *umbralApilacion* no se incluye en el histórico de histogramas.
- Si se encuentra comprendido entre los límites *umbralApilacion y umbralPerdida* se incluye en el histórico.
- Si es superior al *umbralPerdida* se activa la fase de redetección.

A continuación, se va a explicar con mayor detalle el procedimiento seguido:

En primer lugar, se calcula la mascará que permitirá descriminar aquellos píxeles en los que no se han producido movimiento y aquellos en los que el valor de los canales Value y Saturation no se encuentran dentro de los límites establecidos(background detection).

Tras esto, se calcula el histograma de la región de búsqueda actual(función *calcHist()*) y se normaliza (normalize()).

El siguiente paso consiste en la inicialización de un array de diferencias en el que se va a a almacenar las diferencias entre la componente correspondiente del histórico de histogramas y el histograma actual.

Tras haber inicializado dicho array, se recorre el histórico de histogramas, se comparan y se almacenan en el array de diferencias. Tras finalizar este proceso almacenamos la posición y el valor de la componente más pequeña. La comparación se lleva a cabo empleando la función de OpenCv *compareHist(),* que recibe como parámetros los dos histogramas a comparar y el método de computar la comparación: correlación, chi-cuadrado, intersección y distancia de Bhattacharyya. El método empleado ha sido este último.

El siguiente paso es determinar que acción se va a desarrollar, tal y como se ha explicado anteriormente. En el caso de que sea inferior al líminte *umbralApilacion* se calculará la retroproyección de dicho histograma y se ejecutará la función encargada de desarrollar el algoritmo CAMShift. Si la mínima diferencia almacenada se encuentra entre ambos umbrales, la primera acción a realizar es comprobar si el número de histogramas apilados

supera a la longitud del array. Esto es fundamental para evitar un error por desbordamiento de buffer[3].

En el caso de que el número de histogramas almacenados iguale al tamaño de este, se realiza un proceso de eliminación. El criterio tomado ha sido el de eliminación de las componentes más antiguas. Concretamente se elimina el primer 25 % de la dimensión máxima. Esto se lleva a cabo empleando la función *histProm.erase()*,la cual recibe el rango de componentes que se desean eliminar.

```
histProm.erase(Componente inicial, Componente final)
```

Tras la eliminación, desplaza el resto de las componentes a ocupar las primeras posiciones. El histograma actual se apila y se calcula la retroproyección de este. Finalmente, se ejecuta el algoritmo CAMShift y se dibuja la elipse en la región obtenida.

Por último, existe la posibilidad de que el objeto se haya perdido. Esto será cuando la diferencia supere el umbral *umbralPerdida.* En este caso, no se ejecutará ningúna función, simplemente se modificará el valor del estado para que en la siguiente iteración comience el proceso de búsqueda. Este desarrollo se puede observar en la siguiente ilustración:

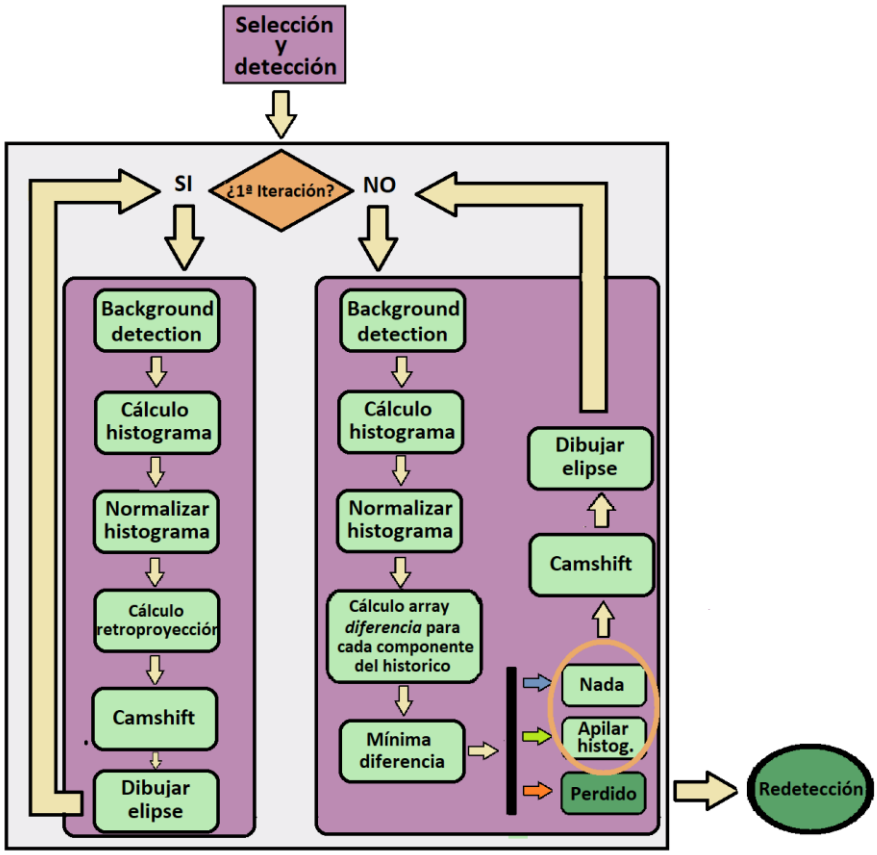


Figura 4-6. Esquema de bloques del algoritmo de tracking

[3] Un desbordamiento de búfer (del inglés buffer overflow o buffer overrun) es un error de software que se produce cuando un programa no controla adecuadamente la cantidad de datos que se copian sobre un área de memoria reservada a tal efecto (buffer): Si dicha cantidad es superior a la capacidad preasignada, los bytes sobrantes se almacenan en zonas de memoria adyacentes, sobrescribiendo su contenido original, que probablemente pertenecían a datos o código almacenados en memoria.

4.3 Redetección

En este apartado se desarrolla la explicación teórica relacionada con la detección del objetivo cuando se haya perdido, debido a una oclusión o salida del rango de visión de la cámara. Como ya se ha explicado en la sección anterior, esta fase se activa tras llevar a cabo el cálculo de la mínima diferencia entre el histograma actual y el histórico de histogramas. Es implementado mediante la función *redeteccionObjeto()*. La idea principal desarrollada en esta consiste en una división de la imagen en regiones, calculando el momento de orden cero de cada una de ellas. Se discriminan todas aquellas regiones con un momento de orden cero nulo, y el resto se compara con un momento de orden cero de referencia. Si la diferencia respecto a este se encuentra dentro de un cierto umbral se almacenará dicha región. A continuación, se repite este proceso subdividiendo la región anterior en otras más pequeñas de igual tamaño. Este proceso se repetirá 3 veces y cada una de las regiones se subdivide en otras 4 de igual tamaño.

Respecto al momento de orden cero de referencia, decir que este viene dado por el momento de orden cero calculado en la región de interés, en la última iteración en la que el objetivo se encontraba dentro del rango de visión. Esto nos permite descartar aquellos objetos que entren dentro del campo de visión con un aspecto similar al del objetivo, pero de diferente tamaño. En la siguiente imagen es posible observar como se realiza la división de la imagen en rejillas para detectar el objeto perdido.

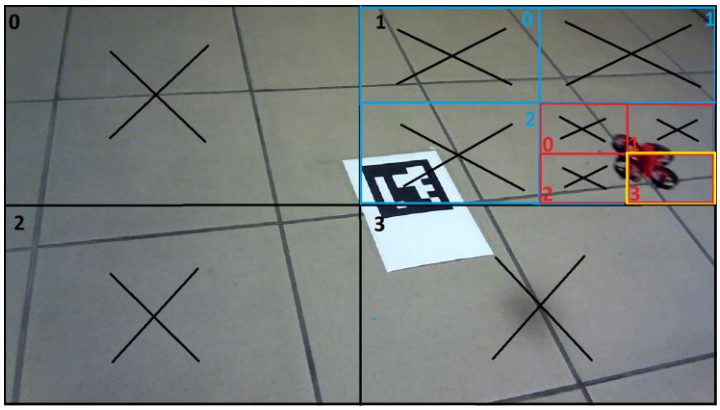


Figura 4-7. Ejemplo redetección. Cuadrante resultante: 1-3-3

Por otro lado, el trabajar con 2 cámaras implica que la pérdida del objetivo se puede producir en una o en ambas cámaras, de forma que habría que contemplar multitud de posibilidades. En primer lugar, es posible que tras seleccionar el objetivo únicamente en una de las cámaras, este se pierda, activándose el proceso de redetección unicamente en dicha cámara, mientras que la otra se encuentra esperando la selección de un objetivo. Tambien, es posible que el objetivo se pierda en una de las cámaras mientras que en la otra se encuentra en la etapa de tracking, ya sea en la fase de inicializición de tracking o en la etapa cíclica de tracking. Todos estos casos se encuentran recogidos en la función *redeteccionObjeto()*. Estas posibilidades se han de tener en cuenta a la hora de calcular la estimación de la posición aplicando el filtro de Kalman. El EKF podría seguir integrando medidas de una única cámara, pero se sabe que el error de estimación crecerá con el tiempo debido a que el sensor de imagen sólo proporciona una medida 2D y se pierde la información de profundidad (caso estimación monocular). Por lo tanto, se ha decidido que se ejecute el módulo de estimación unicamente cuando el objetivo se encuentra identificado por ambas cámaras.

5 POSICIONAMIENTO DE LAS CÁMARAS MEDIANTE ARUCO

En este apartado se va a desarrollar la explicación teórica vinculada al procedimiento empleado para conocer la posición y orientación de cada una de las cámaras. Este proceso es llevado a cabo en el bloque de configuración. Como ya se describió en el capítulo 3, únicamente se ejecuta al comienzo del programa. Permite ubicar las cámaras donde se deseen, siempre y cuando, el marcador de ArUco se ubique dentro del campo de visión de cada una de ellas. Tras haberlas situado, se ha de pulsar la tecla *'g'* para seleccionar el objetivo deseado en cada una de las cámaras e iniciar la estimación de posición del objetivo.

Es imprescindible, una correcta calibración de cada una de las cámaras para conseguir conocer con exactitud la posición y orientación de cada una de ellas.

Tanto la calibración como la estimación de la posición y orientación de cada cámara se han conseguido haciendo uso de la librería ArUco.

5.1 Calibración

4.1.1 Concepto

La calibración de una cámara es el proceso mediante el cual se obtienen los parámetros fundamentales de una cámara. Estos parámetros permiten determinar donde se proyecta un punto 3D del espacio en el sensor de la cámara. Los parámetros de esta se clasifican en intrínsecos y extrínsecos. Lo primeros, son aquellos que describen el funcionamiento de una cámara, mientras que los segundos, definen la posición y orientación del cuadro de referencia de la cámara respecto al mundo real, es decir, dan la orientación externa de la cámara. Los parámetros intrínsecos son:

- $f_x \; y \; f_y$: Distancia focal de la lente de la cámara en ambos ejes. Normalmente expresados en píxeles.
- $C_x \; y \; C_y$: Centro óptico del sensor. Expresado en píxeles.
- $k_1,\ k_2,\ p_1,\ p_2,\ k_3$: Coeficientes de distorsión.

En una cámara ideal, un punto (X, Y, Z) en el espacio se proyectaría en el píxel determinado por las siguientes ecuaciones:

$$x = -X \cdot \frac{f}{Z} + C_x \quad (5\text{-}1)$$

$$y = -Y \cdot \frac{f}{Z} + C_y \quad (5\text{-}2)$$

Sin embargo, las lentes de las cámaras distorsionan la escena, haciendo que la distancia de los diferentes puntos respecto al centro de la imagen varie. Por lo tanto, si se desea conocer con exactitud la proyección de los diferentes puntos en la imagen se debe de considerar estos coeficientes de distorsión. La distorsión se puede descomponer en dos componentes, una componente radial y otra tangencial, ambas dependientes del ángulo y de la distancia respecto al centro de la imagen. Todo esto se contempla en los coeficientes de distorsión calculados en la calibración ($k_1,\ k_2,\ p_1,\ p_2,\ k_3$).

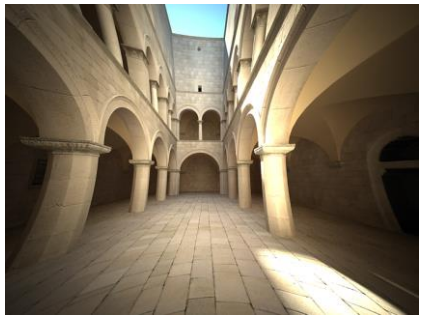


Figura 5-1. Ejemplo de distorsión

Si se desea conocer la proyección de un punto referido a un sistema de referencia arbitrario es necesario hacer uso de los parámetros extrínsecos. Los parámetros extrínsecos son básicamente las rotaciones 3D ($R_{vec} = \{R_x,\ R_y,\ R_z\}$) y las translaciones ($T_{vec} = \{T_x,\ T_y,\ T_z\}$) necesarias para traducir el sistema de referencia de la cámara al arbitrario. También se pueden tener en cuenta cambios de escala y cambios de perspectiva. Una forma de llevar a cabo este cambio de sistema de referencia es haciendo uso de las coordenadas homogéneas.

4.1.1 Procedimiento

ArUco permite llevar a cabo este proceso de calibración. En primer lugar, es necesario imprimir un tablero de calibración proporcionada por esta librería. Este tablero es el que se puede observar en la siguiente figura:

Ilustración 8. Tablero de calibración

Una vez impreso el tablero, es necesario medir con la mayor precisión posible la dimensión de los marcadores. Respecto a la unidad de medida, no es un dato relevante, no importa si la medida esta en centímetros, metros o milímetros. No obstante, el resultado final lo obtendremos en la unidad de medida en la que hayamos tomado esta.

Una vez impreso el tablero y anotada la dimensión de los marcadores, se procede a tomar fotos de este desde

diferentes localizaciones y perspectivas. Se tomará al menos 15 imágenes y se tratará de que se observen en todas ella el tablero completo. A continuación, se pueden apreciar algunos ejemplos:

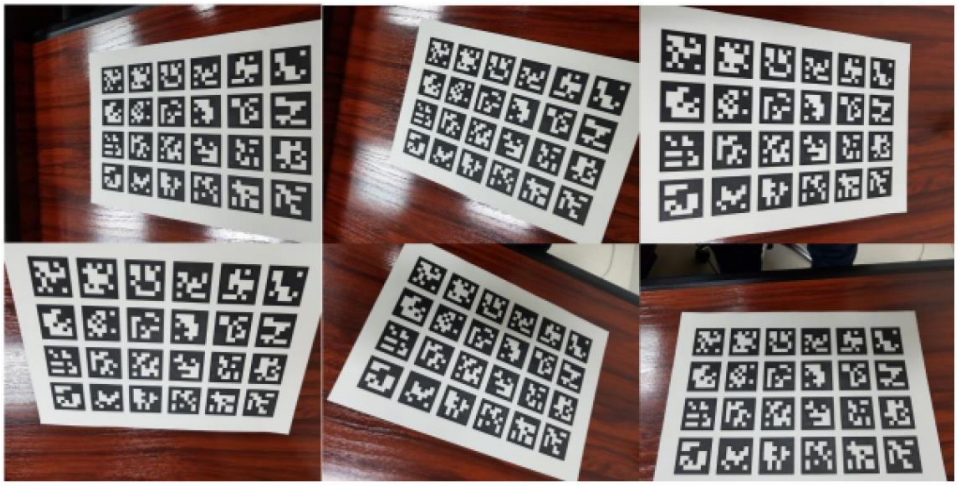


Figura 5-2. Imágenes tomadas para la calibración

Tras haber tomado las fotos del tablero, se ejecutará el programa proporcionado por la librería denominado *aruco_calibration_fromimages* que se puede encontrar en la carpeta *utils_calibration.*

Para ejecutar este programa se debe ejecutar el siguiente comando en el terminal de Ubuntu:

```
Aruco_calibration_fromimages mycalibrationfile.yml pathToDirWithImage -size 0.03
```

El parámetro *mycalibrationfile.yml* es la salida del proceso. *PathToDirWithImage* es el directorio en el que se encuentran las diferentes imágenes tomadas para la calibración. El último parámetro es el tamaño de los marcadores del tablero.

El fichero obtenido tiene el siguiente aspecto:

```
%YAML:1.0
---
image_width: 1280
image_height: 720
camera_matrix: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 9.5801400105445532e+02, 0., 6.1849613240361305e+02, 0.,
       9.5820189344904907e+02, 3.6247688954105831e+02, 0., 0., 1. ]
distortion_coefficients: !!opencv-matrix
   rows: 1
   cols: 5
   dt: d
   data: [ 2.9351483428023819e-02, -8.1573429085994631e-02,
       4.0976198756401653e-04, -2.1662558403973170e-03,
       -6.0054855468679516e-02 ]
```

Figura 5-3. Archivo .yml obtenido tras la calibración

En la ilustración anterior es posible apreciar que mediante esta función proporcionada por ArUco es posible determinar los parámetros intrínsecos de la cámara(distancia focal en los ejes X e Y, y las coordenadas del centro

óptico), al igual que los coeficientes de distorsión.

5.2 Posicionamiento

Una vez calibrada la imagen se puede proceder a conocer la posición de la cámara respecto a un tag de ArUco. La detección de las cuatro esquinas de un marcador permite aplicar estimadores de pose planar. No obstante, la estimación de la pose que usa solo 4 puntos coplanarios está sujeta a ambigüedad. Como se muestra en la siguiente figura, un marcador podría proyectarse en los mismos píxeles en dos ubicaciones de cámara diferentes. En general, la ambigüedad puede resolverse si la cámara está cerca del marcador. Sin embargo, a medida que el marcador se hace pequeño, aumentan los errores en la estimación de la posición.

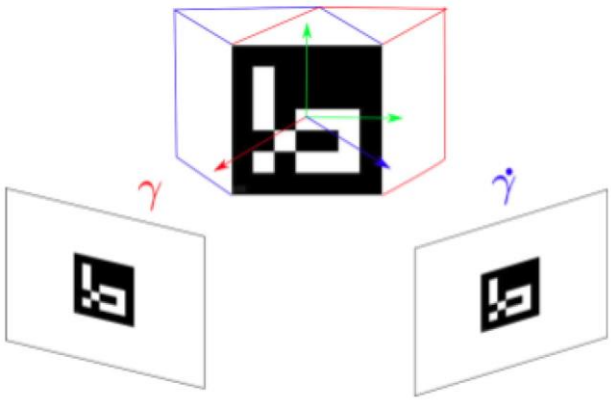


Figura 5-4. Problema de ambigüedad en la estimación de posición.

Cuando se conocen los parámetros intrínsecos de la cámara y la dimensión del marcador, es posible determinar la posición relativa del marcador respecto de la cámara. Este algoritmo permite obtener cuanto debe de girar y transladarse el sistema de referencia del marcador para hacerlo coincidente con los ejes de la cámara. Esta transformación viene dada por los vectores Tvec y Rvec. El primero de ellos es un vector de 3 componentes en las que se almacena la translación sobre cada uno de los ejes correspondientes. El segundo es otro vector de 3 componentes que define el giro al que hay que someter al sistema de referencia del marcador para hacerlo paralelo con respecto al de la cámara. Esta rotación viene definida por el método conocido como *Rodrigues' rotation,* muy empleado en la librería OpenCv.

Para conocer la posición de un punto respecto al sistema de referencia de la cámara será necesario llevar a cabo una transformación que permita expresar el punto referido al sistema de referencia del marcador en el sistema de referencia de la cámara. Para ello, se hará uso de las coordendas homogéneas.

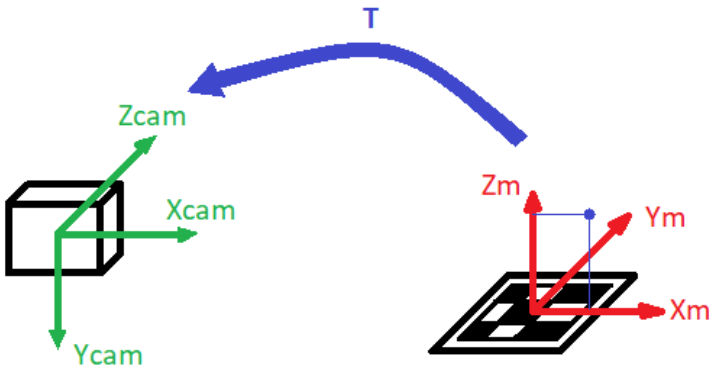


Figura 5-5. Transformada T coordenadas homogeneas

A continuación, es posible observar el sistema de referencia generado gracias a la librería ArUco y el código desarrollado:

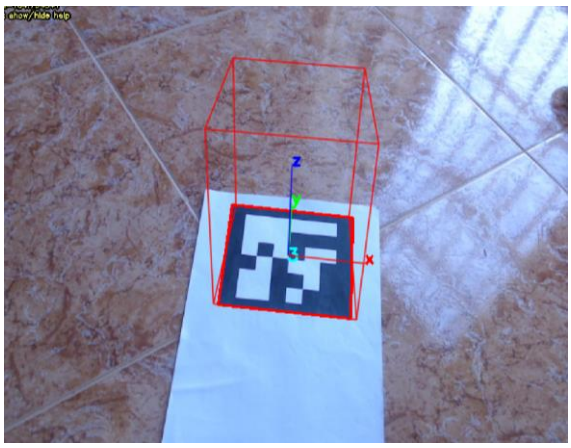


Figura 5-6. Sistema de referencia generado en el marcador

5.2.1 Rodrigues' rotation

En muchas areas de la robótica, la vision artificial es necesaria para rotar puntos u objetos unos determinados ángulos sobre sus ejes correspondientes. Una forma de hacer esto es usando la formula conocida como *Rodrigues' rotation.*

$$R = I + [n]_x \sin\alpha + [n]_x{}^2(1 - \cos\alpha) \quad (5\text{-}3)$$

Donde $[n]_x$ es una inclinación simétrica del vector normalizado del eje de rotación y α es el ´ngulo de rotación en radianes. El resultado (R) es una matriz de rotación 3x3.

La forma más básica de representar rotaciones es mediante los ángulos de Euler, donde culquier rotación en el espacio es representada mediante tres rotaciones 2D en los planos xy, yz y xz.

$$\boldsymbol{A} = \begin{pmatrix} cos\,\varphi & sin\varphi & 0 \\ -sin\,\varphi & cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \boldsymbol{B} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\mu & sin\mu \\ 0 & -sin\,\mu & cos\mu \end{pmatrix} \quad \boldsymbol{C} = \begin{pmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{pmatrix}$$

La matriz de rotación, por tanto, es el resultado de multiplicar estas tres matrices.

$$\boldsymbol{R = ABC} \quad (5\text{-}4)$$

Es evidente que el resultado de la matriz de rotación depende del orden en el que se lleven a cabo las rotaciones, debido a que el producto de matrices no es conmutativo. Los ángulos de rotación de Euler son un método simple de entender y visualizar, pero no es práctico en una gran cantidad de aplicaciones. Rodrigues' rotation supone una alternativa que soluciona estos problemas. A continuación, se desarrollará brevemente su fundamento teórico.

Se parte de un punto V, el cual se desea rotar un Angulo α, alrededor de un vector normalizado *n* . Para hacer esto se define un vector *v,* que parte desde el origen y llega al punto V, y un plano en el origen y perpendicular al eje de rotación n.

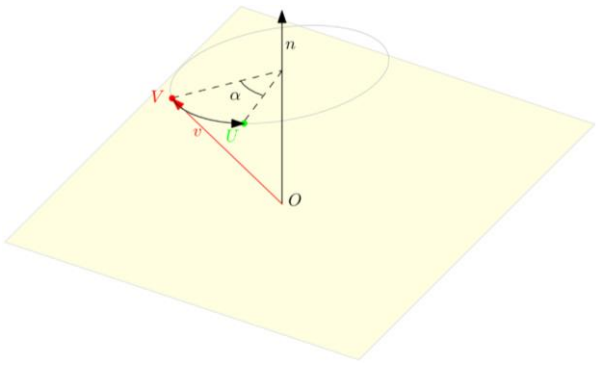


Figura 5-7. Definición punto *v*

El Segundo paso es proyectar el vector *v* en el eje de rotación y en el plano definido en el origen. El vector proyectado sobre el eje de rotación *n* se denomina v_p y el vector proyectado sobre el plano se denomina v_r .

$$v_p = (n \cdot v)n \tag{5-5}$$

$$v_r = v - v_p \tag{5-6}$$

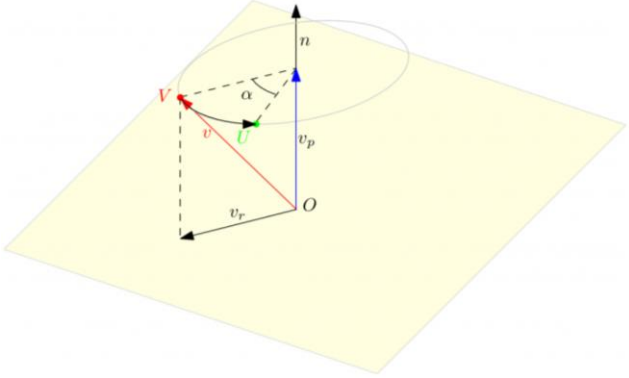


Figura 5-8. Definición de los vectores v_p y v_r

Una vez construidos los vectores anteriores, se define uno nuevo denominado *w,* perpendicular al plano conformado por los vectores *v* y v_p .

$$w = n \; x \; v \tag{5-7}$$

Se define un vector v_{rr} , generado al hacer girar un cierto ángulo α el vector v_r en el plano conformado por los vectores v_r y w.

$$v_{rr} = v_r \cos \alpha + w \sin \alpha \tag{5-8}$$

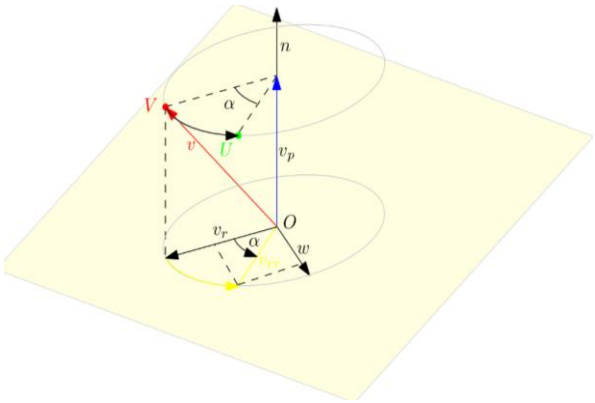


Figura 5-9. Definición vector v_{rr}

Por último, se contruye un vector, *u*, que parte desde el origen y va hacia el punto U. Este vector se genera sumando v_{rr} y v_p.

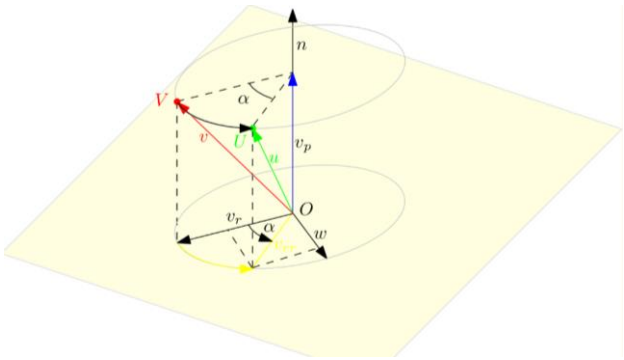


Figura 5-10. Definición vector *u*

Recopilando las ecuaciones definidas:

$$v_p = (n \cdot v)n \quad (5\text{-}5)$$

$$v_r = v - v_p \quad (5\text{-}6)$$

$$w = n \, x \, v \quad (5\text{-}7)$$

$$v_{rr} = v_r \cos\alpha + w \sin\alpha \quad (5\text{-}8)$$

$$u = v_{rr} + v_p \quad (5\text{-}9)$$

Sustituyendo las ecuaciones anteriores y simplificando se obtiene la expression:

$$u = v\cos\alpha + (n \cdot v)n(1 - \cos\alpha) + (n\ x\ v)\sin\alpha \quad (5\text{-}10)$$

Es posible simplificar la expression anterior convirtiendo los productos escalares y vectoriales en multiplicaciones. Asumiendo que todos los vectores son vectores columnas el producto escalar se puede calcular como:

$$v_p = (v \cdot n)n = (n^T v)n = nn^T v \quad (5\text{-}11)$$

Para calcular el producto vectorial entre n y v se convertirá el vector n por una matriz simétrica con diagonal nula, de forma que el producto vectorial entre ambos vectores se calculará de la siguiente forma:

$$w = n\ x\ v = [n]_x v \quad (5\text{-}12)$$

$$[n]_x = \begin{pmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{pmatrix}$$

Aplicando ambas simplificaciones y alguna más, se obtiene la formula conocida como *rodrigues' rotation*.

$$R = I + [n]_x \sin\alpha + [n]_x{}^2(1\ - \cos\alpha) \quad (5\text{-}9)$$

De esta forma, se obtiene una manera de rotar cierto ángulo un detetrminado punto alrededor de un eje.

5.2.2 Coordenadas homogeneas

El método empleado para estimar la posición nos porporciona una matriz de giro y una translación en los diferentes ejes. Estos dos parámetros permiten modificar el Sistema de referencia de un determinado punto, de forma que es posible expresar la posición del objetivo respecto al Sistema de referencia de la cámara conociendo la posición de este respecto al sistema de referencia del marcador. Para llevar a cabo esta transformación se emplean las coordenadas homogeneas.

$$x = Tu \quad (5\text{-}14)$$

Siendo:

- x: coordenadas en $\{O_{cam}, X_{cam}, Y_{cam}, Z_{cam}\}$ (en este caso, sistema de referencia de la cámara).
- u: coordenadas en $\{O_m, X_m, Y_m, Z_m\}$ (en este caso, sistema de referencia del marcador).

Mediante T se pueden representar translaciones(p), giros(R), cambios de escala(w) y cambios de perspectiva

(f). La matriz T tiene una dimensión de 4x4 y se define de la siguiente forma:

$$T = \begin{pmatrix} R_{3x3} & p_{3x1} \\ f_{1x3} & w_{1x1} \end{pmatrix}$$

En nuestro caso, las componentes referentes al cambio de perspectiva serán nulas. La componente *w* tendra como valor la unidad.

6 ESTIMACIÓN COOPERTIVA DE POSICIÓN

TRas haber realizado la calibración de la cámara y el calculo de la posición relativa de cada una de ellas respecto al sistema de referencia, el algoritmo de tracking comienza a proporcionar coordenadas de la posición del objeto seleccionado en el plano imagen. Es necesario, por tanto, llevar a cabo una transformación de un modelo 2D a uno tridimensional para conocer su posición en el espacio. Para lograrlo, se combina la información obtenida por las dos cámaras, realizando de forma cooperativa una estimación de las coordenadas en un espacio tridimensional. Para conseguirlo, se ha empleado el modelo Pinhole de la cámara, las coordenadas homogéneas para realizar transformaciones proyecticas ,y por último, el filtro de Kalman. Por lo tanto, el esquema desarrollado es el que se puede apreciar en la siguiente ilustración:

$\{\boldsymbol{E}\}$	Sistema de referencia de la Tierra.
${}^{\boldsymbol{E}}\boldsymbol{r}_{\boldsymbol{T}}$	Posición del objetivo respecto al sistema de referencia {E}
${}^{\boldsymbol{E}}\boldsymbol{r}_{\boldsymbol{CAMj}}$	Posición de la cámara respeto al sistema de referencia {E}. Dado por ArUco.
${}^{\boldsymbol{CAMj}}\boldsymbol{r}_{\boldsymbol{T}}$	Posición del objetivo respecto al sistema de referencia de dicha cámara.
${}^{\boldsymbol{E}}\boldsymbol{R}_{\boldsymbol{CAM}}$	Rotación de la cámara.
$({}^{j}\boldsymbol{C}_{\boldsymbol{x}}, {}^{j}\boldsymbol{C}_{\boldsymbol{y}})$	Proyección del objetivo en el plano de la imagen.

Tabla 6-1: Notación

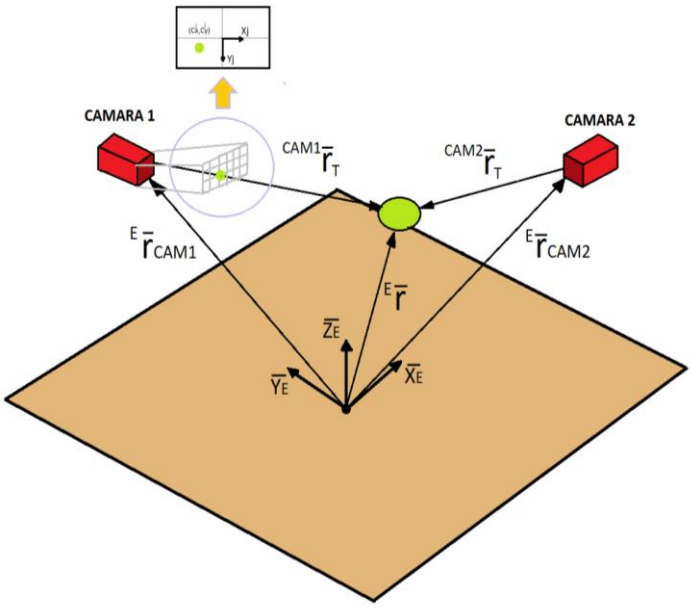


Figura 6-1. Estimación cooperativa de posición

6.1 Modelo pinhole

El modeó teórico de la cámara puede ser descrito de acuerdo a varios modelos, así como la formación de una imagen a partir de la cámara. El más utilizado es el modelo *pinhole*.

Una cámara *pinhole* es una cámara muy simple que no tiene lente y que tiene una sola y muy pequeña apertura, de ahí su nombre, *pinhole*, que es la traducción al inglés de la palabra "hoyo de aguja". Esta cámara puede ser vista de otra manera como una caja a prueba de luz con un pequeño orificio en uno de sus lados, siendo ese orificio la única entrada de luz que permite la caja. Cuando la luz de una imagen pasa a través de este agujero entonces se forma una imagen invertida en el lado opuesto de la caja.

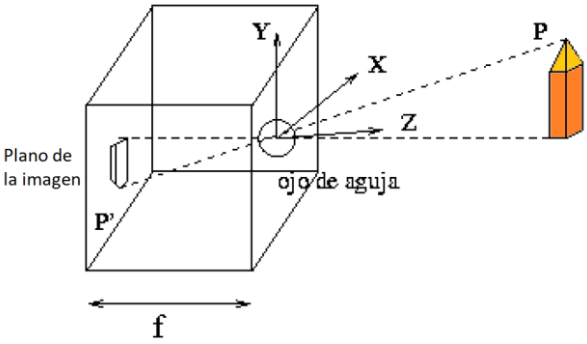


Figura 6-2. Modelo pinhole

El modelo de la cámara pinhole describe la relación matemática que existe entre las coordenadas de un punto en 3D y su proyección en el plano de imagen, donde la apertura de la cámara es descrita como un punto (infinitesimalmente pequeña) y no se utiliza ningún tipo de lente para enfocar la luz. En otras palabras, este modelo de cámara puede solamente ser utilizado como una aproximación de primer orden en el trazo de un mapa de una escena 3D a una imagen 2D.

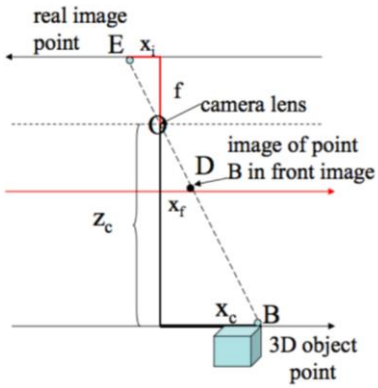
De esta manera, el modelo de la cámara pinhole es frecuentemente utilizado como una descripción razonable de cómo una cámara representa una escena en 3D, como en el caso de visión por computadora.

El modelo matemático de esta cámara se basa en la proyección cónica y cambios de sistema de referencia. Las expresiones matemáticas que definen este modelo son las siguientes:

$$ {}^{j}C_x = - {}^{CAMj}\boldsymbol{r}_{T,X} \cdot \frac{f}{{}^{CAMj}\boldsymbol{r}_{T,Z}} \quad (6\text{-}1) $$

$$ {}^{j}C_y = - {}^{CAMj}\boldsymbol{r}_{T,Y} \cdot \frac{f}{{}^{CAMj}\boldsymbol{r}_{T,Z}} \quad (6\text{-}2) $$

En la siguiente ilustración en 2D se aprecia como es posible obtener dichas ecuaciones aplicando el Teorema de Tales.



$$ \frac{x_i}{f} = \frac{x_c}{z_c} $$

Figura 6-3. Modelo pinhole 2D

La distancia focal de la cámara es conocida, sin embargo, la distancia en el eje Z no. Esta se estimará mediante el filtro de Kalman extendido.

6.2 EKF aplicado a la estimación cooperativa de posición

En el capítulo 2, se ha desarrollado el Filtro de Kalman Extendido de forma teórica para un caso genérico. En este caso, se profundizará en él, pero orientado a resolver el problema de la estimación cooperativa de posición. Partiendo de la base teórica proporcionada en el capítulo dos, el filtro de Kalman extendido consta de una fase de inicialización, que se ejecuta únicamente la primera vez, una fase de predicción y otra de corrección, siendo estas dos últimas las que se ejecutan en el resto de las iteraciones.

Cómo ya se comentó en el capítulo 2, las ecuaciones dinámicas que modelan el sistema son las siguientes:

$$X_k = f(X_{k-1}) + W_{k-1} \tag{6-3}$$

$$Z_k = h(X_k) + V_k \tag{6-4}$$

Podemos identificar los siguientes vectores en las ecuaciones anteriores:

- X_k: Vector de estado, definido por la posición y velocidad del objetivo.

$$X_k = \begin{pmatrix} X \\ Y \\ Z \\ V_X \\ V_y \\ V_z \end{pmatrix}$$

- W_{k-1}: Vector que modeliza el ruido asociado al modelo del sistema. Se trata de un vector aleatorio gaussiano blanco de media cero y matriz de covarianza Q_k.

$$W_{k-1} = \begin{pmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \sigma_{Vx} \\ \sigma_{Vy} \\ \sigma_{Vz} \end{pmatrix}$$

- Z_k : Vector de observación, definido por la posición del objeto proporcionada por el algoritmo de tracking.

$$Z_k = \begin{pmatrix} C_x \\ C_y \end{pmatrix}$$

- V_k: Vector aleatorio que modeliza la incertidumbre asociada a la medida. Se trata de un vector aleatorio gaussiano blanco de media cero y matriz de covarianza R_k.

$$V_k = \begin{pmatrix} \sigma_{Cx} \\ \sigma_{Cy} \end{pmatrix}$$

Para implementar las distintas fases del Filtro de Kalman extendido se ha desarrollado la función *updateEstimation* , que recibe los parámetros:

- Posición de la cámara respecto al tag de ArUco (dada por ArUco).
- Rotación de la cámara respecto al tag de ArUco (dada por ArUco).
- Posición del objetivo obtenida por la cámara (Se obtiene una medida en pixeles que se debe pasar a cm utilizando los datos de la cámara).
- ID de la cámara.

A continuación, se procederá a describir como se ha implementado el EKF para nuestro sistema de seguimiento. Se describirá cada una de las fases y los valores tomados para cada una de las variables existentes.

Fase inicialización

En primer lugar, es necesarior inicializar el vector de estado inicial, X_0^a. Este debe ser un vector aleatorio de media μ_0 y covarianza P_0.

En este caso, el vector de estados inicial se ha inicializado con los siguientes valores:

$$X_0^a = \begin{pmatrix} 0 \\ 0 \\ 0.5 \\ 0.1 \\ 0.1 \\ 0.1 \end{pmatrix}$$

Además, también es necesario inicializar la matriz de covarianza del error en el instante anterior (P_{k-1}) . Se le ha asignado la matriz identidad:

$$P_{k-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Fase de predicción

Los pasos que seguir dentro de dicha función serán los siguientes:

En primer lugar, se calcula el vector de estados predicho:

$$X_{forecast,k} = \begin{pmatrix} X \\ Y \\ Z \\ V_{X,k} \\ V_{y,k} \\ V_{z,k} \end{pmatrix} = \begin{pmatrix} X_{k-1} + \Delta t \cdot V_x \\ Y_{k-1} + \Delta t \cdot V_y \\ Z_{k-1} + \Delta t \cdot V_z \\ V_{x,k-1} \\ V_{y,k-1} \\ V_{z,k-1} \end{pmatrix}$$

Se va a suponer que Δt es pequeño y , que, por lo tanto , se puede utilizar un modelo del sistema lineal. La velocidad se considera constante en los distintos ejes. Para calcular el incremento de tiempo, cada vez que se acceda a la función, se almacenará el instante de tiempo en el que se ejecuta, de forma que se almacenará en la variable *t_anterior* el instante de tiempo en el que se ejecutó por última vez el EKF. De esta forma es posible conocer el incremento de tiempo entre cada iteración.

En segundo lugar, se calcula la matriz de covarianza del error producido en la predicción:

$$P_{f,k} = J_f \cdot P_{k-1} \cdot J_f^T + Q_{k-1} \quad (6\text{-}5)$$

Siendo:

- $J_f = \begin{pmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$, ya que $J_f(i,j) = \frac{df(i)}{dq_i}$

- $Q_{k-1} = \begin{pmatrix} \sigma_x & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_y & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_z & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{Vx} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{Vy} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{Vz} \end{pmatrix}, con\ \sigma_i = \sigma_{Vi} = 0.001^2$

  - P_{k-1}: matriz de covarianza del error calculada en la iteración anterior, o en el caso de ser la primera iteración, matriz identidad de dimensión 6x6.

Fase de corrección

Tras haber calculado el vector de estados predichos, es necesario someterlo a un proceso de corrección a partir del vector de observación obtenido. Esta etapa de corrección comienza calculando el vector de observación en dicha iteración. Como se ha comentado con anterioridad, este se encuentra definido por la posición del objetivo proporcionada por el algoritmo de tracking, no obstante, este proporciona una posición en píxeles, por lo que el primer paso es construir transformar dicha medida en metros.

Tras haber construido el vector de observación Z_k ,se calcula la ganancia de corrección K_k mediante la siguiente expresión:

$$K_k = P_k^f \cdot J_h^T(x_k^f) \cdot \left(J_h(x_k^f) \cdot P_k^f \cdot J_h^T(x_k^f) + R_k\right)^{-1} \quad (6\text{-}6)$$

Siendo:

- P_k^f: matriz de covarianza del error producido en la predicción. Calculado en la fase anterior.
- R_k: matriz de covarianza que modela la incertidumbre asociada a las medidas. El valor asignado es constante.

$$R_k = \begin{pmatrix} 0.01^2 & 0 \\ 0 & 0.01^2 \end{pmatrix}$$

- J_h : matriz jacobiana de la ecuación de observación. Cada componente viene definida por la siguiente expresión:

$$J_h(i,j) = \frac{dh(i)}{dq_i} \quad (6\text{-}7)$$

Es evidente que, para poder calcular la ganancia de corrección, K_k, es necesario calcular con anterioridad la

matriz J_h. El calculo de esta no es trivial, por lo que se va a desarrollar el procedimiento realizado para calcularla. En primer lugar, se debe de conocer como se define la función *h* (operador que mapea el espacio de estado dentro del espacio de observaciones). Esta viene dada por el modelo pinhole de la cámara:

$$Z_k = h(X_k) = \begin{pmatrix} C_x \\ C_y \end{pmatrix} = \begin{pmatrix} -f \cdot \frac{{}^{CAMj}X_T}{{}^{CAMj}Z_T} \\ -f \cdot \frac{{}^{CAMj}Y_T}{{}^{CAMj}Z_T} \end{pmatrix} \quad (6\text{-}8)$$

Respecto a la etapa de predicción, se conoce la posición estimada del objetivo, pero con respecto al sistema de referencia proporcionado por el marcador de ArUco, que es el que se ha considerado como sistema de referencia principal. En este caso, es necesario conocer la posición del objetivo respecto al sistema de referencia de la cámara. Para conseguir este cambio en el sistema de referencia se han empleado las coordenadas homogéneas (explicadas en el capítulo 5):

$$\begin{pmatrix} {}^{CAMj}X_T \\ {}^{CAMj}Y_T \\ {}^{CAMj}Z_T \\ 1 \end{pmatrix} = T \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, siendo\ T = \begin{pmatrix} R_{3x3} & p_{3x1} \\ f_{1x3} & w_{1x1} \end{pmatrix} \quad (6\text{-}9)$$

En nuestro caso, la matriz de transformación T tendrá el siguiente aspecto:

$$T = \begin{pmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Por lo que, la ecuación de coordenadas homogéneas queda:

$$\begin{pmatrix} {}^{CAMj}X_T \\ {}^{CAMj}Y_T \\ {}^{CAMj}Z_T \end{pmatrix} = \begin{pmatrix} R_{11} \cdot X + R_{12} \cdot Y + R_{13} \cdot Z + t_1 \\ R_{21} \cdot X + R_{22} \cdot Y + R_{23} \cdot Z + t_2 \\ R_{31} \cdot X + R_{32} \cdot Y + R_{33} \cdot Z + t_3 \end{pmatrix} \quad (6\text{-}10)$$

Sustituyendo los valores anteriores en la ecuación (6-8) :

$$Z_k = h(X_k) = \begin{pmatrix} C_x \\ C_y \end{pmatrix} = \begin{pmatrix} -f \cdot \frac{R_{11} \cdot X + R_{12} \cdot Y + R_{13} \cdot Z + t_1}{R_{31} \cdot X + R_{32} \cdot Y + R_{33} \cdot Z + t_3} \\ -f \cdot \frac{R_{21} \cdot X + R_{22} \cdot Y + R_{23} \cdot Z + t_2}{R_{31} \cdot X + R_{32} \cdot Y + R_{33} \cdot Z + t_3} \end{pmatrix} \quad (6\text{-}11)$$

Una vez expresado el operador que mapea el espacio de estado dentro del espacio de observaciones en función de las componentes del vector de estados, se procede a calcular la matriz J_h

Por lo tanto, la matriz de inercias Jh, se define como:

$$J_h = -f \begin{pmatrix} J_{h11} & J_{h12} & J_{h13} & 0 & 0 & 0 \\ J_{h21} & J_{h22} & J_{h23} & 0 & 0 & 0 \end{pmatrix} \quad (6\text{-}12)$$

Donde:

- $J_{h11} = \frac{Y(R_{11}R_{32}-R_{31}R_{12})+Z(R_{11}R_{33}-R_{31}R_{13})+R_{11}t_3-R_{31}t_1}{(R_{31}X+R_{32}Y+R_{33}Z+t_3)^2}$
- $J_{h12} = \frac{X(R_{12}R_{31}-R_{32}R_{11})+Z(R_{12}R_{33}-R_{32}R_{13})+R_{12}t_3-R_{32}t_1}{(R_{31}X+R_{32}Y+R_{33}Z+t_3)^2}$
- $J_{h13} = \frac{X(R_{13}R_{31}-R_{33}R_{11})+Y(R_{13}R_{32}-R_{33}R_{12})+R_{13}t_3-R_{33}t_1}{(R_{31}X+R_{32}Y+R_{33}Z+t_3)^2}$
- $J_{h21} = \frac{Y(R_{21}R_{32}-R_{31}R_{22})+Z(R_{21}R_{33}-R_{31}R_{23})+R_{21}t_3-R_{31}t_2}{(R_{31}X+R_{32}Y+R_{33}Z+t_3)^2}$
- $J_{h22} = \frac{X(R_{22}R_{31}-R_{32}R_{21})+Z(R_{22}R_{33}-R_{32}R_{23})+R_{22}t_3-R_{32}t_2}{(R_{31}X+R_{32}Y+R_{33}Z+t_3)^2}$
- $J_{h23} = \frac{X(R_{23}R_{31}-R_{33}R_{21})+Y(R_{23}R_{32}-R_{33}R_{22})+R_{23}t_3-R_{33}t_2}{(R_{31}X+R_{32}Y+R_{33}Z+t_3)^2}$

Tras haber calculado la matriz jacobiana del espacio de observación J_h y, posteriormente, la ganancia de corrección K_k , el siguiente paso es calcular el vector de estado corregido X_k^a empleando la siguiente ecuación:

$$X_k^a = X_k^f + K_k \cdot (Z_k - h(x_K^f)) \quad (6\text{-}13)$$

En la ecuación anterior, es posible identificar como se le aplica una ganancia K_k al error producido entre la posición del objetivo proporcionada por el algoritmo de tracking y el punto en el que debería de estar a partir del vector de estados predicho en la etapa anterior.

Por último, se actualiza la matriz de covarianza del error producido en la predicción P_k:

$$P_k = (I - K_k \cdot J_h) \cdot P_k^f \quad (6\text{-}14)$$

7 EXPERIMENTOS Y RESULTADOS

EN este capítulo se mostrarán los resultados obtenidos en los diversos experimentos realizados. Se han llevado a cabo cuatro tipos de experimentos diferentes. El primero de ellos se basa en estimar la posición de un objetivo en una posición aproximadamente fija, utilizando diferentes matrices de covarianza (R_k , Q_k). El siguiente experimento, consiste en estimar la posición del objetivo realizando este un movimiento oscilatorio en la dirección del eje X, respecto al sistema de referencia global dado por el marcador de ArUco. Similar al anterior es el tercer experimento realizado, consistente en estimar la posición del objetivo realizando este un movimiento oscilatorio, pero en la dirección del eje Y. El último experimento consiste en la estimación de la posición del objetivo cuando este realiza de forma aproximada una trayectoria circular.

7.1 Objetivo fijo

Tal y como se ha descrito en la introducción del capítulo, en primer lugar, se mostrará la información obtenida estimando la posición de un objetivo que permanece ,de forma aproximada, estático en un cierto punto. El experimento se ha llevado a cabo realizando el seguimiento de un quadrotor en el aire, por lo que la posición de este sufrirá cierta oscilación.

Las siguientes estimaciones, se han realizado utilizando un tiempo de captura de unos 60 segundos aproximadamente. De aquí se van a obtener algunas medidas estadísticas como el valor medio, varianza, y máxima desviación.

7.1.1 Experimento I

En este primer caso, en el que se estima la posición estática del objetivo se han empleado las siguientes matrices de covarianza:

$$R_k = \begin{pmatrix} 1^2 & 0 \\ 0 & 1^2 \end{pmatrix} \qquad Q_k = \begin{pmatrix} 0.1^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1^2 \end{pmatrix}$$

Los resultdos obtenidos para estos valores han sido:

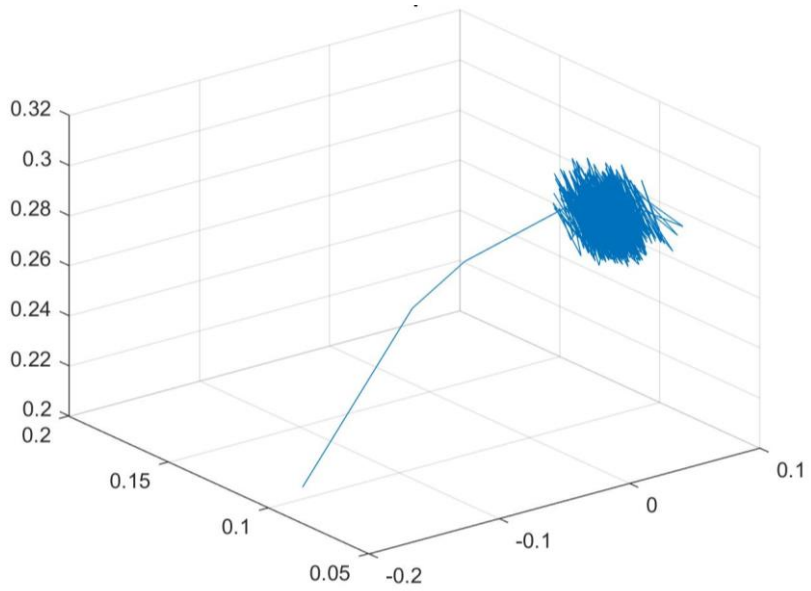


Figura 7-1. Representación 3D para el primer caso de estimación de la posición del objetivo en condiciones estáticas

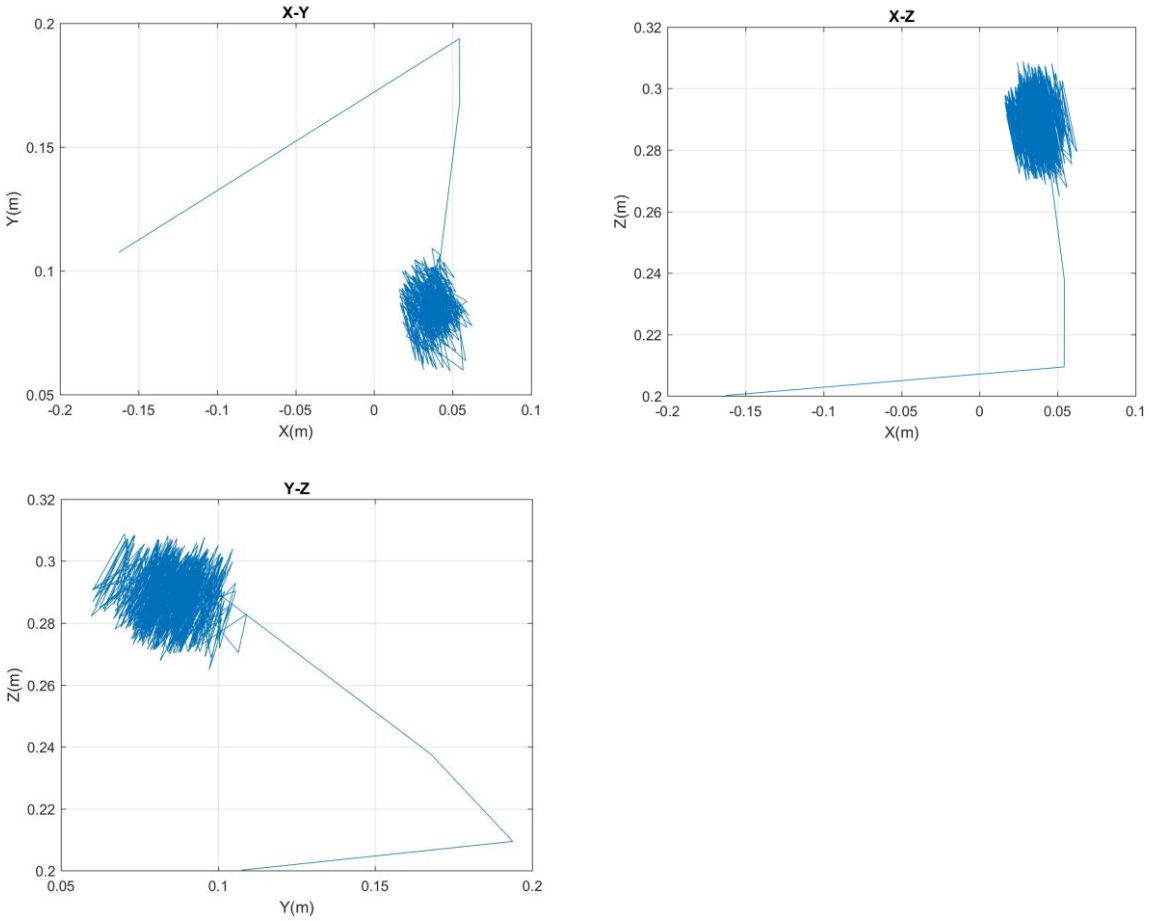


Figura 7-2. Diferentes vistas procedentes de la figura 7-1

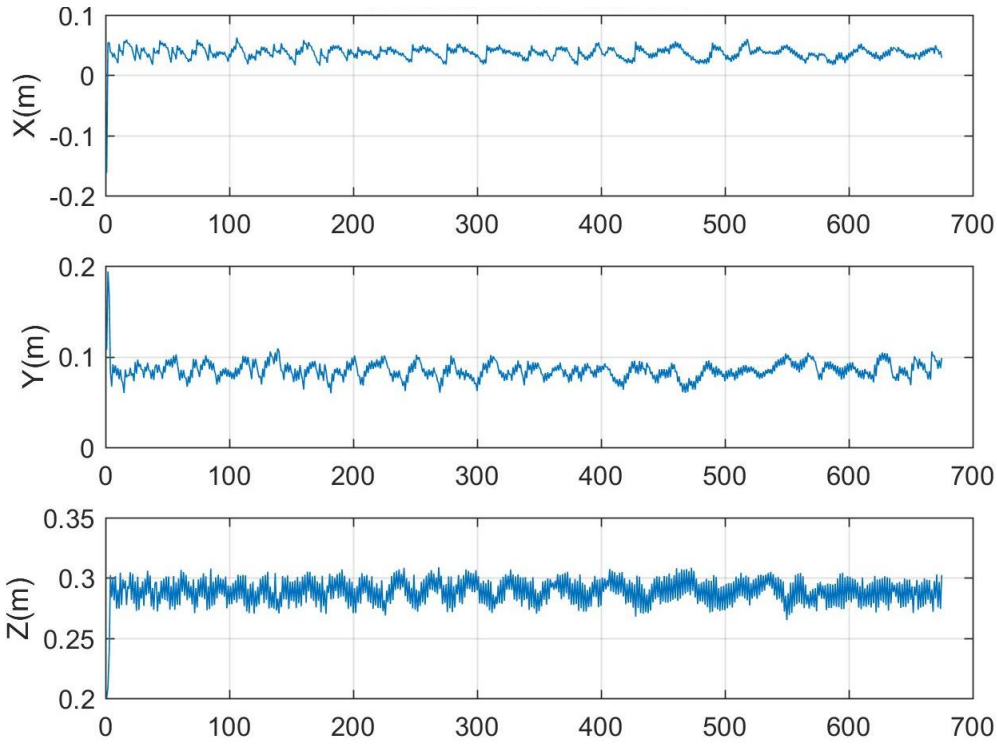


Figura 7-3. Representación de los valores estimados para cada eje pertenecientes al primer experimento

Para este experimento se han obtenido los siguientes parámetros estadísticos:

Parámetro estadístico	X	Y	Z
Media (m)	0.0367	0.085	0.2889
Varianza (m^2)	$1.37 \cdot 10^{-4}$	$1.07 \cdot 10^{-4}$	$6.108 \cdot 10^{-5}$
Máxima desviación (m)	0.0254	0.0251	0.0242

Tabla 7-1. Medidas estadísticas experimento I

Observando la tabla 7-1 , se puede comprobar como se obtiene una desviación máxima respecto de la media de unos 2 centímetros aproximadamente en cada eje. La varianza calculada para cada uno de los ejes, se encuentra comprendidoa entre 1 cm^2 (ejes X e Y) y 6 cm^2 (eje Z).

7.1.2 Experimento II

En este primer caso, en el que se estima la posición estática del objetivo se han empleado las siguientes matrices de covarianza:

$$R_k = \begin{pmatrix} 0.1^2 & 0 \\ 0 & 0.1^2 \end{pmatrix} \qquad Q_k = \begin{pmatrix} 0.1^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1^2 \end{pmatrix}$$

Los resultdos obtenidos para estos valores han sido:

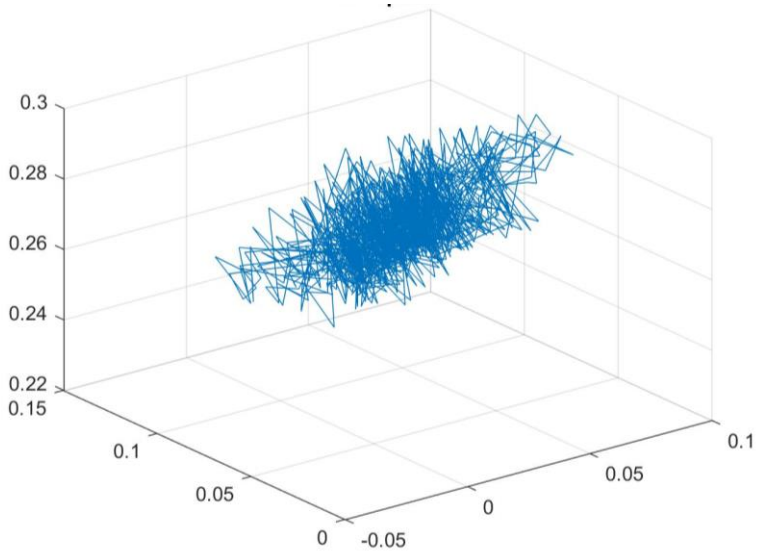


Figura 7-4. Representación 3D para el segundo caso de estimación de la posición del objetivo en condiciones estáticas

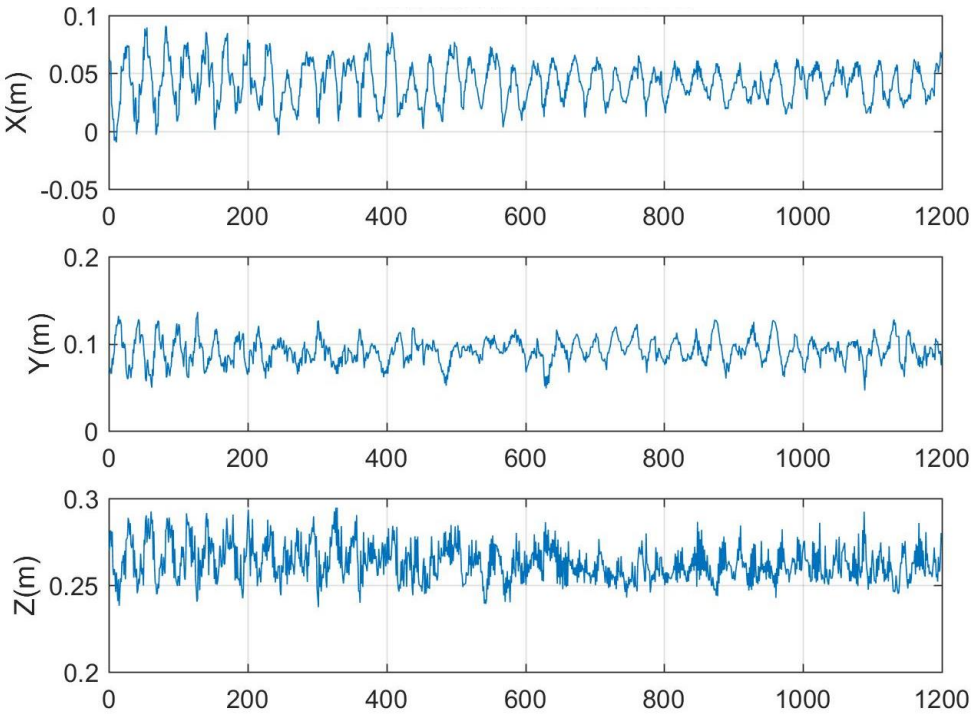


Figura 7-5. Representación de los valores estimados para cada eje pertenecientes al segundo experimento

Para este experimento se han obtenido los siguientes parámetros estadísticos:

Parámetro estadístico	**X**	**Y**	**Z**
Media (m)	0.041	0.0918	0.2628
Varianza (m^2)	$1{,}27 \cdot 10^{-4}$	$1.06 \cdot 10^{-4}$	$1.117 \cdot 10^{-5}$
Máxima desviación (m)	0.0516	0.0455	0.0319

Tabla 7-2. Medidas estadísticas experimento II

En este experimento, se ha reducido en un orden de magnitud los valores de la matriz de covarianza que modela el ruido de observación, por lo que la varianza calculada debería de disminuir respecto al experimento anterior en cada eje. Observando la tabla 7-2, se puede comprobar ha sido así.

7.1.3 Experimento III

En este primer caso, en el que se estima la posición estática del objetivo se han empleado las siguientes matrices de covarianza:

$$R_k = \begin{pmatrix} 0.1^2 & 0 \\ 0 & 0.1^2 \end{pmatrix} \quad Q_k = \begin{pmatrix} 0.001^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.001^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.001^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.001^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.001^2 \end{pmatrix}$$

Respecto a los experimentos anteriores, en este caso la matriz de covarianza del modelo estimado toma valores 2 ordenes de magnitud inferior a los experimentos anteriores. Esto conllevaría un menor error en la estimación del objetivo, no obstante, la convergencia del modelo se vuelve más compleja, lo que podría suponer que el programa nos proporcionase unos valores erróneos.

Los resultdos obtenidos para estas matrices han sido:

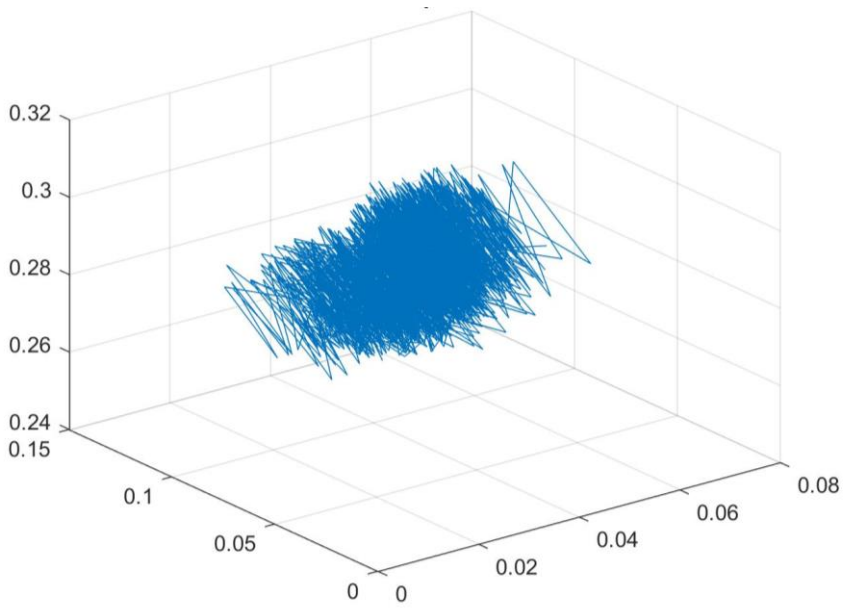


Figura 7-6. Representación 3D para el tercer caso de estimación de la posición del objetivo en condiciones estáticas

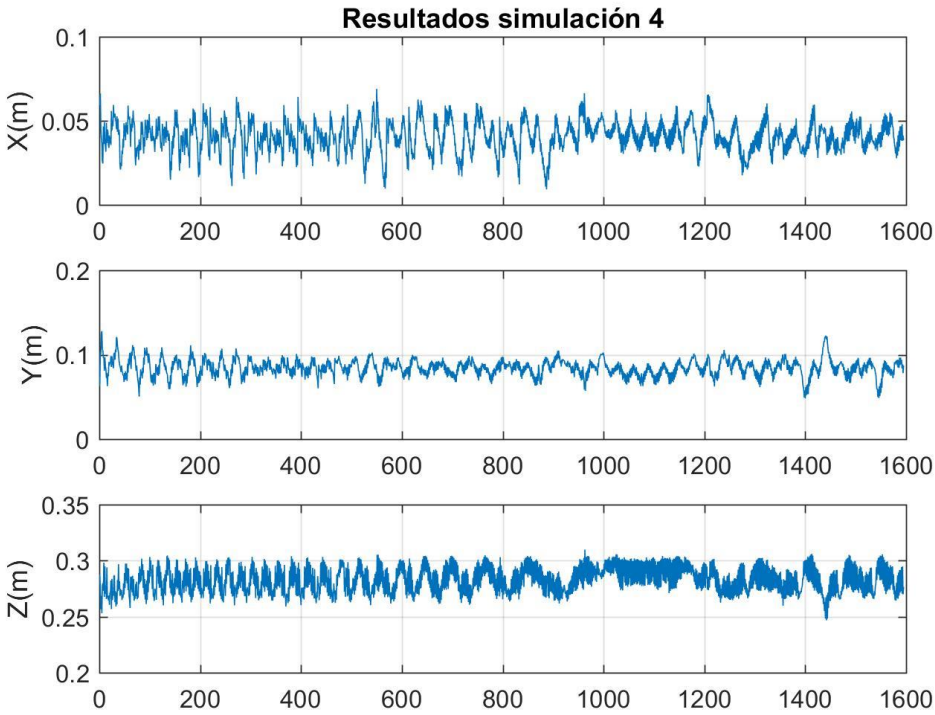


Figura 7-7. Representación de los valores estimados para cada eje pertenecientes al tercer experimento

Para este experimento se han obtenido los siguientes parámetros estadísticos:

Parámetro estadístico	X	Y	Z
Media (m)	0.0407	0.0845	0.2827
Varianza (m^2)	$8.816 \cdot 10^{-5}$	$1.058 \cdot 10^{-4}$	$1.538 \cdot 10^{-5}$
Máxima desviación (m)	0.0315	0.0435	0.0309

Tabla 7-3. Medidas estadísticas experimento III

En este experimento, se ha mantenido la matriz de covarianza de observación con respecto a la sección anterior, no obstante, se han reducido los valores de la matriz de covarianza que modela la incertidumbre del modelo en varios ordenes de magnitud. La varianza obtenida en los ejes X e Y es inferior a la obtenida en los casos anteriores, sin embargo, en el eje Z el valor a aumentado. Esto es debido a que la posición del objetivo sudre cierta oscilación, por lo que cada experimento no se realiza exactamente bajo las mismas condiones.

7.2 Oscilación en torno al eje X

En esta sección, se mostrará los resultados obtenidos para el caso en el que la posición del objetivo no es estática. Concretamente, el movimiento que realiza es un movimiento rectilineo en la dirección del eje X, respecto al sistema de referencia global (ArUco).

Los resultdos obtenidos para experimento han sido:

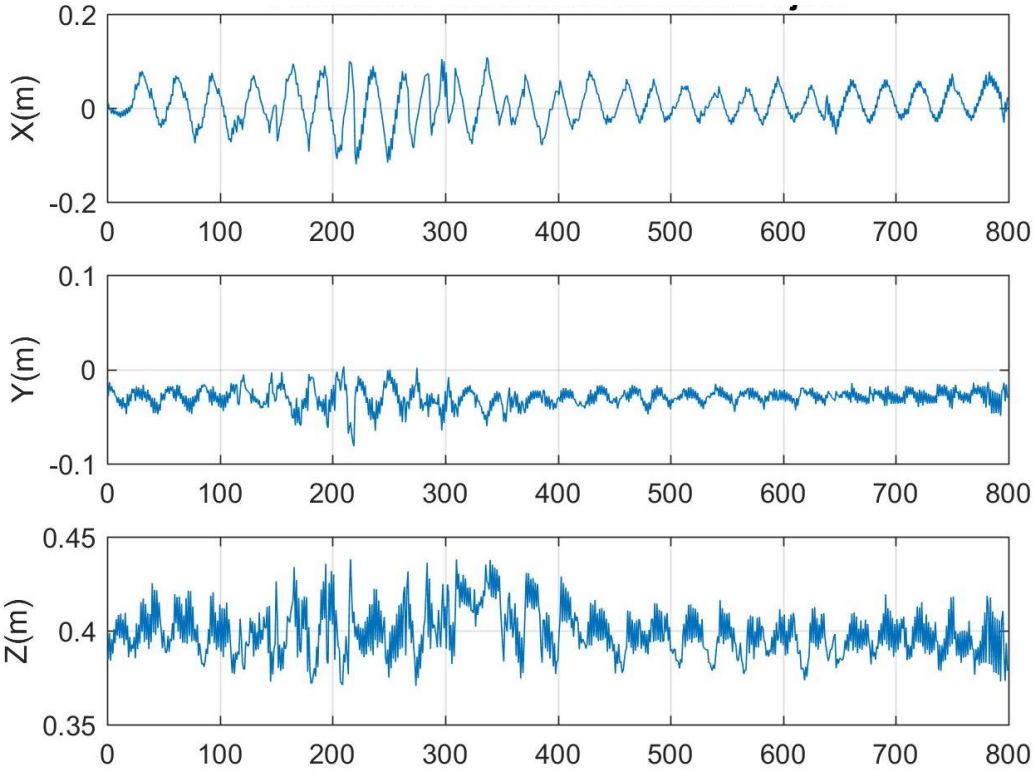


Figura 7-8. Representación de los valores estimados para
cada eje para el experimento en el que la posición del objetivo oscila en la dirección del eje X

En la Figura 7-8 se puede observar la oscilación a la que se somete el objetivo en la dirección del eje X, concretamente una oscilación que en algunos instantes alcanza unos ± 13 cm. La máxima oscilación que aparece en el eje Z alcanza unos ± 3cm.

7.3 Oscilación en torno al eje Y

En esta sección, se mostrará los resultados obtenidos para el caso en el que la posición del objetivo no es estática. Concretamente, el movimiento que realiza es un movimiento rectilineo en la dirección del eje Y, respecto al sistema de referencia global (ArUco).

Las medidas capturadas para este experimento han sido:

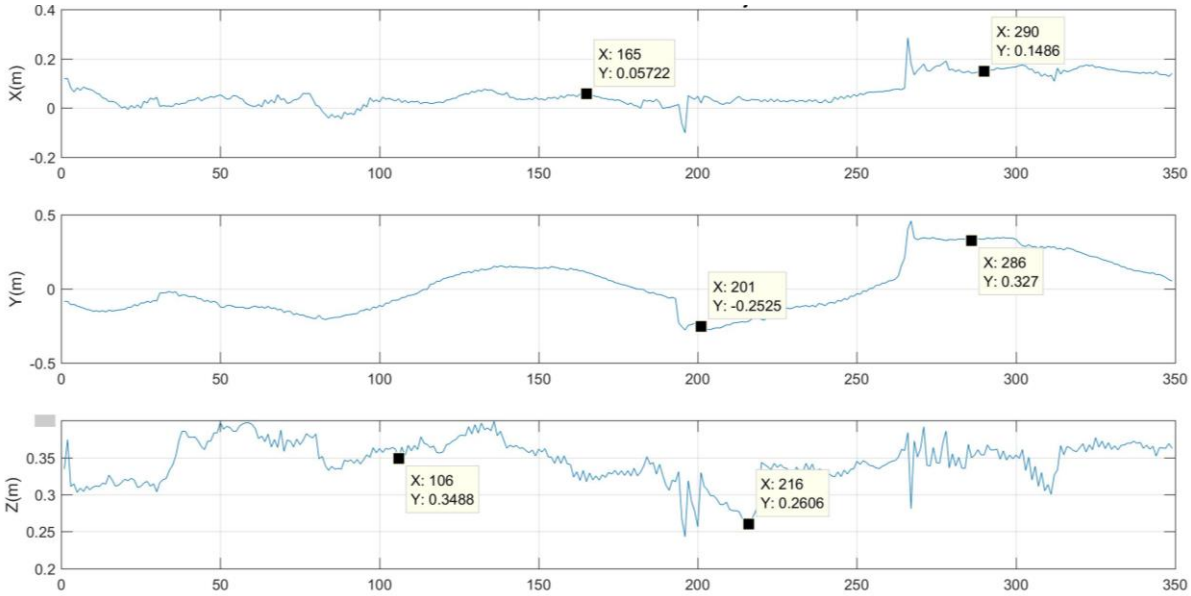


Figura 7-9. Representación de los valores estimados para
cada eje para el experimento en el que la posición del objetivo oscila en la dirección del eje Y

En la Figura 7-9 se puede observar la oscilación a la que se somete el objetivo en la dirección del eje Y, concretamente una oscilación que en algunos instantes supera unos ± 25 cm. En este experimento , se ha producido una oscilación mayor en los otros dos ejes, respecto al experimento anterior, debido a la dificultad de llevar a cabo el control del quadrotor realizando un movimiento oscilatorio.

7.4 Movimiento helicoidal

En esta sección, se mostrarán los resultados obtenidos en el último experimento llevado a cabo. En este, se ha estimado la posición del móvil cuando este realiza , de forma aproximada , una trayectoria circular. Concretamente, se ha intentado seguir una trayectoria helicoidal con el quadrotor, con el que también se modifica la componente Z de la posición del objetivo.

En las siguientes figuras se observan los resultados obtenidos:

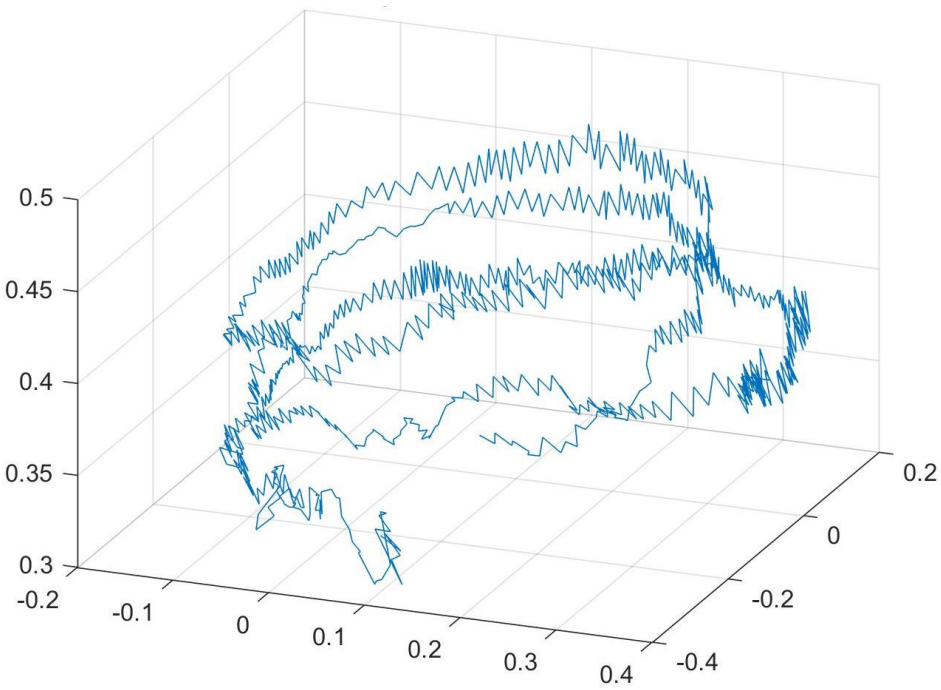


Figura 7-10. Representación 3D para el caso en el que el móvil realiza un movimiento helicoidal

En la siguiente figura, se puede observar una vista de pájaro de dicho movimiento:

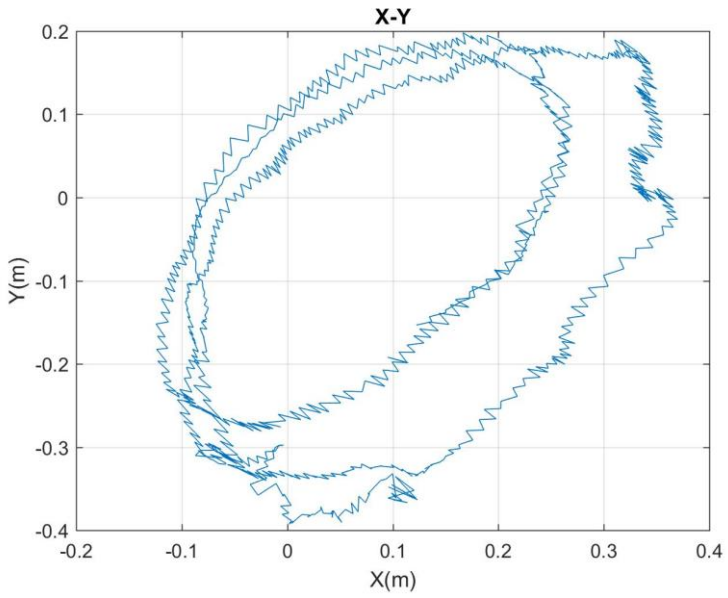


Figura 7-11. Vista de pájaro movimiento helicoidal

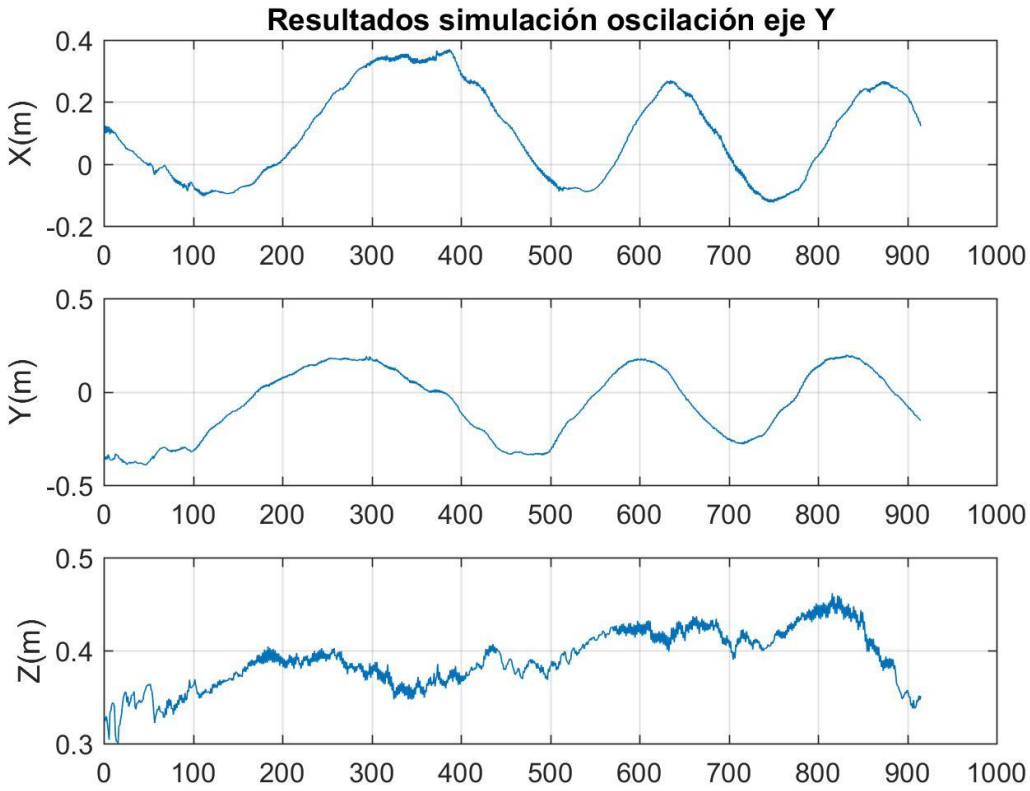


Figura 7-12. Representación de los valores estimados para
cada eje en el experimento en el que el móvil realiza un movimiento helicoidal

8 CONCLUSIONES Y TRABAJOS FUTUROS

En la última década, el número de aplicaciones vinculadas a la visión artificial ha incrementado notoriamente, apareciendo en campos donde no había aparecido con anterioridad, como en el sector del automovilismo y en el comercial. Por esta razón, es valorable tener un nivel importante de conocimientos en esta área. Tras llevar a cabo este proyecto, destacar lo sencillo que puede resultar iniciarse en este mundo de visión artificial mediante la librería OpenCV, sin embargo, esta sencillez inicial se transforma en algo complejo cuando se desea trabajar en entornos reales, donde la luminosidad varia continuamente y el tiempo de procesamiento no debe de superar ciertos valores.

Tras los experimentos llevados a cabo, se puede llegar a la conclusión de que el sistema diseñado proporciona una estimación de la posición de gran calidad, con un error absoluto máximo de 5 cm.

A pesar de estos resultados, el sistema diseñado acepta una gran variedad de mejoras que supondría una mejora de resultados y disminuiría las restricciones apreciables en el sistema. En primer lugar, en el sistema diseñado las diversas cámaras no cuentan con un sistema *Pitch & Roll*, que permitan modificar el ángulo de cada una de ellas. Esta modificación permitiría variar la orientación de las cámaras, siendo posible conocer la posición y orientación sin la necesidad de que el marcador de ArUco se encuentre dentro del rango de visión de cada cámara. Esta modificación, sería posible llevarla a cabo tras conocer la posición y orientación de cada una respecto al marcador de ArUco inicialmente. La nueva matriz de transformación de coordenadas homogéneas se modificaría en función de la matriz inicial , calculada observando el marcador(en la primera iteración sería necesario que el marcador se encontrase dentro del rango de visión de cada cámara), y en función de la rotación llevada a cabo en los distintos ejes de la cámara. De esta forma sería posible conocer la posición y orientación de cada una de ellas sin observar el marcador.

Otra posible mejora por destacar sería la implementación de este sistema en Android, de forma que se pudiese trabajar con las cámaras de cada dispositivo móvil. Esto supondría una mayor comodidad para el usuario que no tendría que portar las cámaras continuamente.

Por último, dicho sistema permite realizar el seguimiento de un solo objetivo. Sería posible mejorar el sistema de forma que se pudiese conocer la posición de varios objetos de forma simultanea. Esto se podría implementar desarrollando un entorno multihilo.

REFERENCIAS

[1] Leonhard Euler, "Problema algebraicum ob affectiones prorsus singulares memorabile", Commentatio 407 Indicis Enestoemiani, Novi Comm. Acad. Sci. Petropolitanae 15 (1770), 75–106.

[2] Gabriel A. Terejanu," Extended Kalman Filter Tutorial"

[3} Leandro M. Di Matteo, Juan Carlos Gómez, Claudio Verrastro, "Algoritmo de seguimiento de objetos basado en visión asistida por computador en tiempo real utilizando CAMShift e histogramas"

[4] Weisstein, Eric W. "Rodrigues' Rotation Formula". Available: http://electroncastle.com/wp/?p=39

[5] Alejandro Pascual,"EKF y UKF:dos extensiones del filtro de Kalman para sistemas no lineales". Available: https://iie.fing.edu.uy/ense/asign/tes/monografias/anteriores/alejandro_pascual/TES2004_pascual.pdf

[6] Rafael Muñoz Salinas, " ArUco: An efficient library for detection of planar markers and camera pose estimation". Available: https://docs.google.com/document/d/1QU9KoBtjSM2kF6ITOjQ76xqL7H0TEtXriJX5kwi9Kgc/edit#heading=h.1z8vm961dhos

[7] M. R. Arahal, *Apuntes de Sistemas de Percepción.*

[8] << OpenCV documentation >> Available: https://docs.opencv.org/2.4/index.html

[9] «MATLAB - El lenguaje del cálculo técnico,» [En línea]. Available: https://es.mathworks.com/products/matlab.html

[10] << Rodrigues' rotation >> Available: http://electroncastle.com/wp/?p=39

[11] Alejandro Suarez, Anibal Ollero, Guillermo Heredia, "Cooperative Virtual Sensor for Fault Detection and Identification in Multi-UAV Applications"

[12] Alejandro Suarez, Anibal Ollero, Guillermo Heredia, "Cooperative sensor fault recovery in multi-UAV systems"