UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Máster en Ingeniería de Telecomunicación

TRABAJO FIN DE MÁSTER

DESARROLLO DE UN TOOLBOX PARA POSICIONAMIENTO 2D DE CÁMARAS.

Sandra Gaytán Grande.
Tutor: Juan Carlos San Miguel Avedillo.
Ponente: Jose María Martínez Sánchez.

Septiembre 2017

DESARROLLO DE UN TOOLBOX PARA POSICIONAMIENTO 2D DE CÁMARAS.

Sandra Gaytán Grande
Tutor: Juan Carlos San Miguel Avedillo
Ponente: Jose María Martínez Sánchez

Video Processing and Understanding Lab
Departamento de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Septiembre 2017

Trabajo parcialmente financiado por el Ministerio de Economía y Competitividad del Gobierno de España bajo el proyecto TEC2014-53176-R (HAVideo) (2015-2017)

Resumen

En los últimos años, los sistemas de video-vigilancia, y entre ellos los sistemas de posicionamiento de cámaras, están siendo ampliamente desarrollados debido a la capacidad de porporcionar información de gran utilidad. En estos entornos, determinar un adecuado posicionamiento de las cámaras previo al desarrollo del sistema será clave para minimizar el número de cámaras, obtener mejores resultados y reducir el coste de posteriores modificaciones. El objetivo de este trabajo radica en el desarrollo de un toolbox de Matlab que permita calcular el posicionamiento de las cámaras en función de las restricciónes de formulación del problema: formulación continua o discreta.

Para la utilización de este toolbox se ha desarrollado una interfaz que permite al usuario seleccionar el mapa de suelo a analizar, las restricciones del coverage y el tipo de optimización a utilizar. Adicionalmente, se ha incluido una funcionalidad en la interfaz gráfica que permite al usuario incluir objetos sobre el mapa de suelo simplemente pinchando sobre el mapa de suelo abierto en la interfaz, así como fijar su altura en proporción a la altura del techo.

Como resultado, el usuario podrá comprobar visualmente la localización de las cámaras y el campo de visión correspondiente.

Palabras clave

Video-vigilancia, Posicionamiento 2D de cámaras, Candidatas, *Coverage*, Interfaz, Usuario, *Matlab*, Campo de Visión, Optimización.

Abstract

In recent years, video-surveillance systems, and more specifically camera positioning systems, are now being deployed widely thanks to their ability. In this context, optimal camera configuration will reduce the total number of cameras used achieving the same or greater level of utility, as well as better results and lower cost asociated with future modifcations. The objetive of this project is the development of a Matlab *toolbox which allows the calculation of the camera's position according to the constraints associated to the problem's formultion: continuous or discrete domain.*

The toolbox makes use of an interface that allows the user to select the floor map that is to be analyzed, the coverage's restrictions and the type of optmization to be used. Additionally, a graphical interface functionality has been included which allows the user to insert objects on the floor map by directly clicking on the interface, as well as fixing their hight relative to the ceiling's.

The final result will allow the user to visually examine the location of the cameras and the field of view.

Keywords

Video-surveillance, 2D camera placement, coverage, interface, user, MATLAB, field of vision (FOV), optimization.

Agradecimientos

En primer lugar, agradecer a mi tutor Juan Carlos por sus consejos y su guía hasta el resultado final. A Chema y al laboratorio VPU por la oportunidad de realizar este trabajo, .

Agradecer también a mis amigos de la universidad, a aquellos que después de tantos años siempre siguen dispuestos a ayudar en todo, y si hay cervezas de por medio mejor: a Pali E.T.E.L.S, por su ayuda y sus ideas; a Marta, Edu y María. Los de siempre y para siempre.

Y por supuesto a mi familia, mis padres, el enano...por el apoyo, los ánimos, las risas, por estar siempre ahí, por todo.

A Mario, por ser tú.

Sandra.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

Este proyecto se enmarca en el ámbito del diseño de sistemas de video-seguridad. En los últimos años, los sistemas de video-seguridad han adquirido una gran popularidad en diversos espacios tales como lugares públicos, transportes e infraestructuras críticas, en dónde las últimas investigaciones en el campo persiguen dos restricciones base:

- Maximizar la utilidad y eficiencia de los sistemas de video-vigilancia.
- Minimizar el coste.

Para todos estos sistemas de video-vigilancia resulta de vital importancia que la configuración óptima de las cámaras (p.e., localización óptima, orientación, etc.) se determine antes de seleccionar el tipo de cámara, puesto que el coste asociado a su modificación sería excesivo. Una configuración óptima de las cámaras puede ayudar a minimizar el número de cámaras necesarias logrando el mismo o mayor nivel de utilidad. Además, la configuración adecuada de una red de cámaras conlleva amplios beneficios en posibles tareas posteriores de análisis computacional. Por ejemplo, considere un posicionamiento óptimo de cámaras en un espacio de forma que el 100 % de espacio queda cubierto, posteriormente el desarrollo de un algoritmo multi-cámara basado en el *tracking,* entendido como seguimiento, de personas quedará considerablemente simplificado.

En los últimos años, las redes de cámaras están siendo ampliamente desarrolladas en el campo de la video-vigilancia inteligente (IVS - *Intelligent Video Surveillance*) debido a la capacidad de propocionar abundante información de video así como cubrir amplios espacios físicos de interés. En estos entornos, se hace imprescindible disponer

de algoritmos inteligentes de posicionamiento de cámaras. Se pueden observar una serie de ejemplos de posicionamiento de cámaras en la Figura 1.1. La complejidad de estos algoritmos reside principalmente en la capacidad de adaptación a los distintos requerimientos de cada usuario, así como al desarrollo de un sistema de optimización *offline* efectivo en función del objetivo del usuario, ya sea detección, localización, tracking o reconocimiento de objetivos de interés. Actualmente existen múltiples herramientas comerciales para este fin [1, 2, 3, 4] cuya mayor desventaja reside en la complejidad de uso, la baja adaptación a los diferentes requisitos de cada usuario, el alto grado de especialización de cada uno de ellos y el coste. En este trabajo se busca hacer frente a estas adversidades y desarrollar una herramienta genérica de posicionamiento que permita al usuario fijar sus propios requerimientos de manera sencilla y calcular un posicionamiento óptimo de las cámaras.

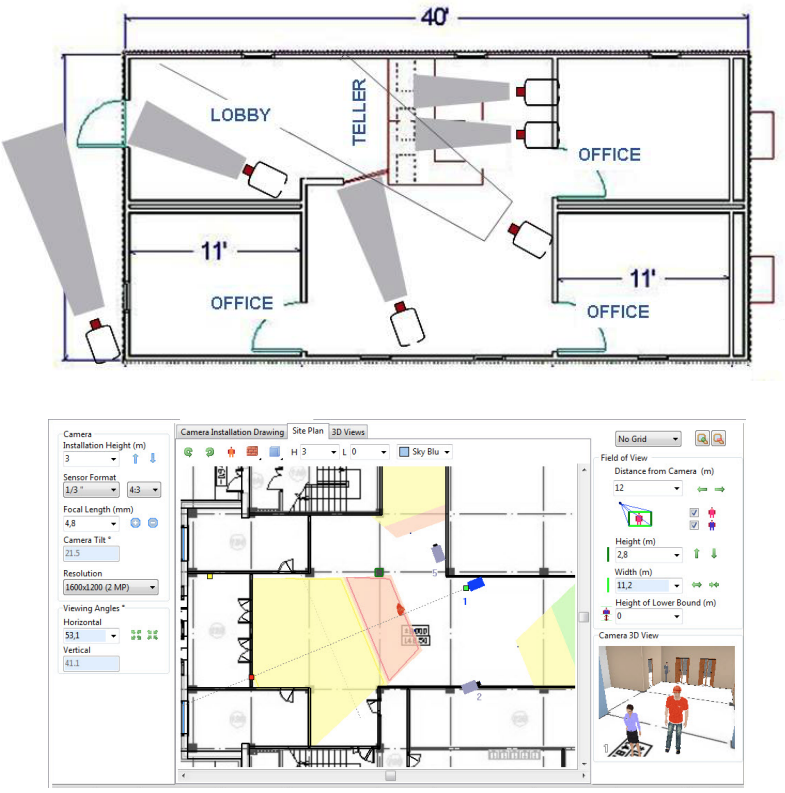


Figura 1.1: Ejemplos de posibles posicionamientos de cámaras

1.2. Objetivo

El objetivo de este proyecto se basa en desarrollar un toolbox en Matlab para posicionamiento de cámaras no direccionales en diferentes entornos que trate de hacer frente a las limitaciones actuales, en lo que se refiere al coste del sistema, la heterogeneidad de funcionamiento, la posibilidad de personalización por parte del usuario y la idoneidad de los resultados obtenidos.

Este objetivo puede ser desglosado en varios puntos a implementar en este trabajo:

1. Estudio y análisis del estado del arte de los desarrollos y técnicas implementados para posicionamiento de cámaras.
2. Búsqueda y estudio de las herramientas actuales existentes en el mercado en relación al cálculo del posicionamiento de cámaras.
3. Implementación y desarrollo de un sistema propio de posicionamiento de cámaras agrupado en un toolbox en Matlab que incluye diferentes posibilidades de optimización y modos de operación, entre las que incluimos el posicionamiento de las cámaras para calcular el campo de visión máximo acumulado y el cálculo de la localización de las cámaras necesarias para cubrir una región determinada del espacio. Paralelamente se desarrolla una interfaz gráfica que posibilita al usuario elegir el mapa de suelo a analizar y fijar sus restricciones de forma sencilla visualmente.
4. Análisis y comparativa de resultados destacando las ventajas y desventajas de cada método implementado.

1.3. Organización de la memoria

La memoria consta de los siguientes capítulos:

- Capítulo 1. Introducción
- Capítulo 2. Estado del arte
- Capítulo 3. Diseño y desarrollo
- Capítulo 4. Evaluación
- Capítulo 5. Conclusiones y trabajo futuro
- Bibliografía.
- Anexos.

Capítulo 2

Estado del arte

2.1. Introducción

El posicionamiento de cámaras constituye un subcampo dentro del gran terreno de la planificación de posición de sensores, dónde la cámara será considerada un sensor sofisticado. Esta sofisticación queda descrita en dos grandes áreas; por un lado, la formación de la imagen constituye un proceso complejo, en donde un amplio rango de factores tales como la longitud focal y la resolución deben ser considerados. En segundo lugar, la cámara proporciona una gran cantidad de información que debe ser procesada para explotar todo el potencial de este particular sensor.

El primer trabajo relacionado fue publicado en 1970 por Chvtal y O'Rourke en el campo de la geometría computacional [5]. Este estudio, cuyo objetivo consistía en el posicionamiento de guardias en una galería de arte, marca el inicio del estudio teórico del posicionamiento de cámaras que busca maximizar la utilidad y la eficiencia así como reducir los costes. Una aproximación convencional al posicionamiento de cámaras propone cuatro fases de actuación: parámetros de entrada, preparación, resolución y parámetros de salida. Tras los parámetros de entrada habrá que determinar si el problema se basa en un entorno dinámico, en donde habrá que realizar una discretización paramétrica y posterior propuesta de candidatas, o en un entorno continuo, en donde se progresará directamente a la formulación del objetivo y aplicación de técnicas de optimización adecuadas.

Con el fin de poder entender todas las necesidades de estudio del posicionamiento de cámaras, se hace necesario dedicar un apartado a los diferentes modelos de cámara y su visibilidad. Entre los modelos de cámaras más utilizados en el posicionamiento de cámaras, en función de la direccionalidad del campo de visión podemos diferenciar entre:

1. Cámaras direccionales. Estas cámaras se caracterizan por tener un ángulo de visión variable y capacidad de rotación. En función de sus grados de libertad (p.e., rotación de los ejes, movimiento, posición..) podemos diferenciar entre: cámaras de perspectiva y cámaras PTZ.

2. Cámaras no direccionales o *DOMO*: Cámaras motorizadas de alta velocidad, con función de inclinación/zoom, incorporadas en una caja protectiva que permiten una cobertura continua a 360° de la escena, sea horizontal ó vertical. En este trabajo se van a utilizar este tipo cámaras, así como su característica de cobertura continua a 360º, como base del cálculo del posicionamiento de cámaras (véase Figura 2.1) [6].

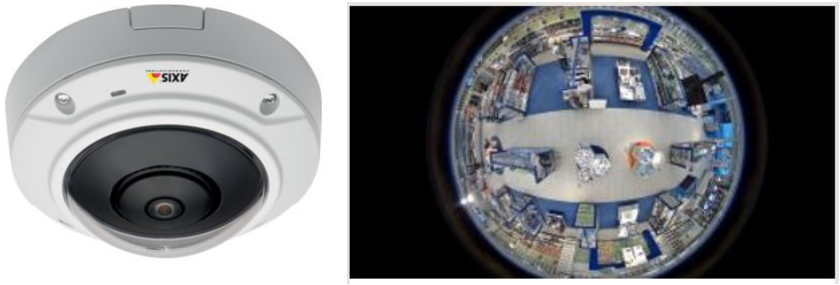


Figura 2.1: Ejemplo de cámara base de este trabajo caracterizada por su capacidad no direccional con cobertura continua a 360º de la escena, y su correspondiente imagen captada en 360º (Imágenes extraídas de [6]).

2.2. Formulación y variables de diseño

El objetivo del problema de optimización offline del posicionamiento de cámaras se centra en dos enfoques opuestos:

1. Maximización de la utilidad, asociada al grado de cobertura del objetivo.

2. Minimización del coste del sistema, relacionado con el número de cámaras necesarias

Las variables de diseño principales que se deben fijar para lograr cubrir los objetivos del sistema son: número de cámaras, cobertura, región de cobertura crítica, ángulo de orientación, visibilidad del objetivo, error de localización y nivel de solapamiento de zona de visión que permita el rastreo. A continuación se procede a explicar más detalladamente cada una de estas variables.

2.2.1. Número de cámaras

El tratamiento del problema varía enormemente en función de si el número de cámaras es conocido o bien es una variable a determinar en el proceso de optimización.

En caso de que el número de cámaras sea desconocido, nos encontramos con una formulación discreta en dónde será necesario determinar un conjunto de candidatas previamente a seleccionar el subconjunto que optimiza el problema.

Por otro lado, en el caso opuesto, nos encontramos con un número fijo de cámaras inicial, de forma que se pasará directamente a la res0lución del problema. Se puede apreciar visualmente la diferencia entre problemas discretos y continuos en la Figura 2.2 .

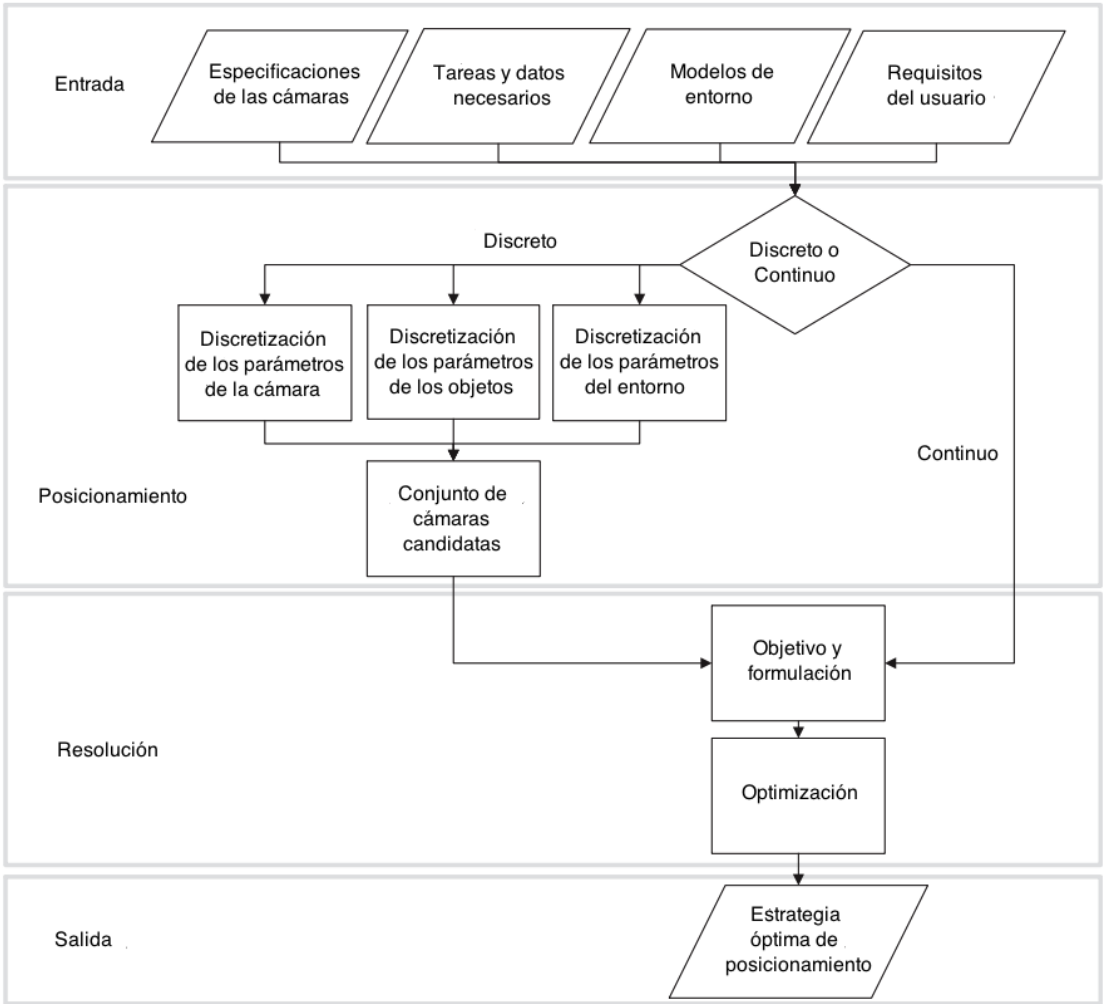


Figura 2.2: Representación gráfica de las diferencias entre una formulación discreta y continua del problema de posicionamiento de cámaras.

2.2.2. Cobertura

La tarea más importante en todo estudio de vigilancia de superficie es conseguir máxima cobertura. La cobertura de una cámara es la zona del círculo de luz proyectado por un objetivo en el que las condiciones de definición e iluminación son aceptables (véase Figura 2.3) [7]. Por lo tanto y como se puede ver en la figura, no todo lo que proyecta el objetivo es aprovechable si no solo una zona circular concéntrica al círculo de luz proyectado.

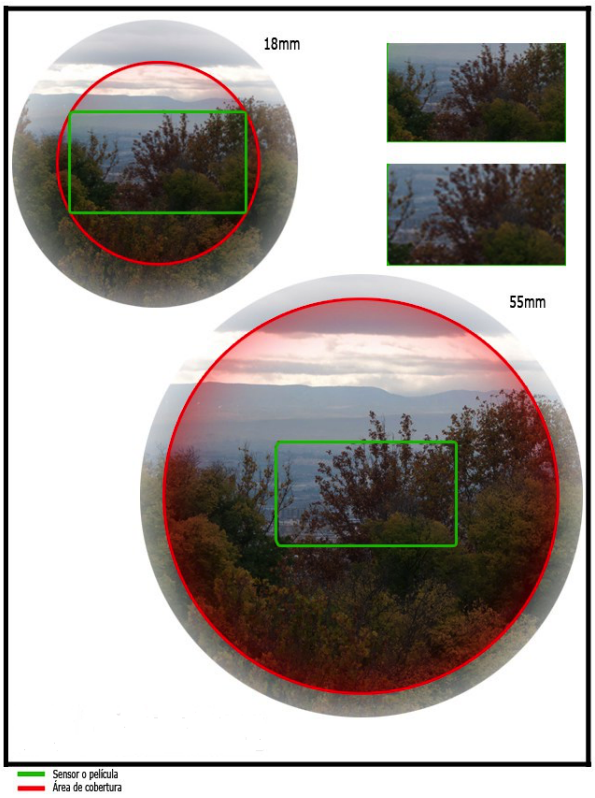


Figura 2.3: Representación gráfica del área de cobertura útil de una cámara. (Imagen extraida de [7]).

2.2.3. Región de cobertura crítica

Adicionalmente al área de cobertura, existen ciertas regiones que pueden requerir una monitorización extra. En la problemática de buscar el sistema con coste mínimo, si resulta necesaria esta cobertura crítica, será necesario agregarla a la ecuación de minimización básica del coste [5]:

$$min \quad \sum_{i=1}^{N_{cand}} e_i x_i - \omega \sum_{j=1}^{N_{region}} (1 - s_j) t_j \tag{2.1}$$

$$st.\ Ax \geq b\ and\ x_i\, \epsilon \{0,1\}$$

En donde e_i hace relación al coste asocidado a la cámara candidata i, x_i es una variable binaria que indica si la cámara i esta entre las N_{cand} candidatas, s_j será 1 si la región es crítica (y 0 en caso contrario); y t_j será 1 si la región no es crítica pero está cubierta (y 0 en caso contrario) [5].

Para medir lo óptima que es la cobertura de una región crítica, usaremos la siguiente función gaussiana como métrica de calidad [5]:

$$Q = \exp\left\{-\beta \left(\frac{\text{máx}\,(F - A, A - F)}{F}\right)^{\alpha}\right\}, \tag{2.2}$$

dónde α y β son variables de control. Como se puede ver, la función (2.2) se comporta como una función normal, siendo máxima cuando la región de cobertura F de las cámaras coincide con las regiones críticas A.

2.2.4. Ángulo de orientación

En las cámaras direccionales, se trataría de considerar el ángulo de orientación óptimo para maximizar el área de cobertura de las cámaras en relación al objeto de interés. Para ello se intentará maximizar una función de calidad dependiente de la función de ganancia que considera la orientación de cada una de las posibles trayectorias de cada cámara [8].

Se expone una descripción gráfica del ángulo de orientación en la Figura 2.4 [5].

2.2.5. Visibilidad del objetivo

La visibilidad (V) de un objeto queda definida por el tamaño de la proyección del objeto en el plano imagen de la cámara. Ésta queda parametrizada por la localización del objetivo (p_j), el vector del objetivo sobre el plano z (v_j) y el ángulo de oclusión en el peor caso (β_j). Si la longitud proyectada de la línea central es mayor que un umbral predefinido (T), entonces el objetivo será visible para la cámara [5] (véase Ecuación 2.3).

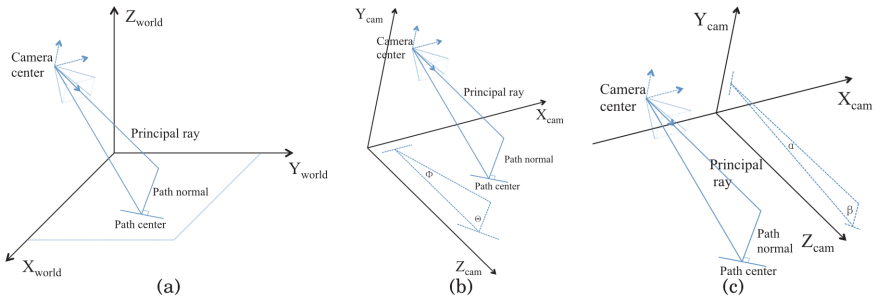


Figura 2.4: Orientación relativa entre la cámara y el centro del camino. (a) Muestra la orientación relativa entre un camino la cámara originante en el plano de coordenadas. (b) Muestra la proyección de la relación entre el camino y la cámara en el plano x,z. (c) Muestra la proyección de la relación entre el camino y la cámara en el plano y,z. (Imagen extraida de [5]).

$$\eta_{i,j} = \Big\{ 1 \; if \; V(p_j, v_j, \beta_j | l, c_{i,E}) > T \quad 0 \; otherwise \tag{2.3}$$

Y la función de optimización queda definida en la Ecuación 2.4:

$$min \quad \sum_{i=1}^{N_{cand}} e_i x_i \tag{2.4}$$

$$st. \; \textstyle\sum_{i=1}^{N_{cand}} x_i \eta_{i,j} \geq 1 \; and \; x_i \, \epsilon \{0,1\}$$

2.2.6. Error de localización

Se entiende por error de localización aquel error asociado a un desalineamiento entre el centro de proyección de la cámara y el centro del actuador, una imprecisión mecánica del sistema actuador de vigilancia o un efecto de cuantificación de los pixeles debido a una precisión finita de las cámaras [9]. El objetivo es desarrollar una estrategia optima de posicionamiento de cámaras que minimice el error de localización del objetivo. Este error quedará definido en términos de una función cuadrática media de error que toma en consideración el número de cámaras y la orientación de las mismas [8].

2.2.7. Nivel de solapamiento de zona de visión que permita el tracking

El último problema común reside en la optimización de los parámetros de la cámara de forma que exista un solapamiento suficiente entre cámaras vecinas para

asegurar un correcto *tracking* del objetivo (véase Figura 2.5) [8]. Teniendo esto en mente, será necesario revaluar la función de optimización incorporando nuevos valores de medida de resolución (M_r) así como de distancias margen (M_d).

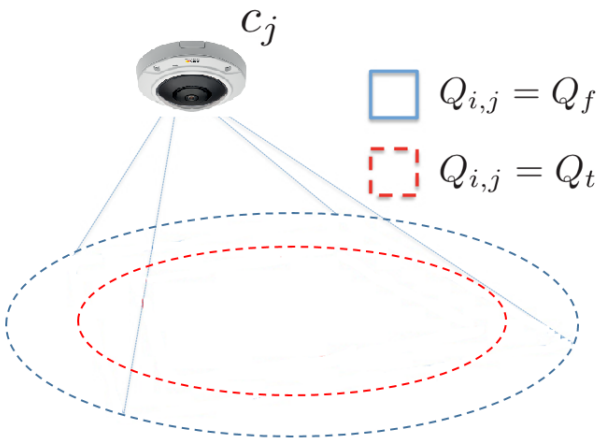


Figura 2.5: Margen de seguridad para la cámara j siendo i el índice del grid discretizado. El área fuera de la región de la línea azul continua son áreas invisibles en donde la métrica umbral Q de la cámara no sobrepasa el umbral de fallo ($Q_{i,j} < Q_f$). Por otro lado, el área comprendida dentro de la línea roja discontinua es la región visible donde el *tracking* del objetivo se cumple con una única cámara ($Q_{i,j} > Q_t$). El área entre las dos líneas es el margen de seguridad de solapamiento ($Q_t > Q_{i,j} > Q_f$) donde el objetivo sigue siendo visible pero debe ser solapado por una cámara diferente para asegurar un *tracking* multi-cámara persistente. (Imagen extraida de [5]).

2.3. Herramientas empresariales existentes en el mercado

Como primer contacto con las herramientas de posicionamiento de cámaras existentes en el mercado se ha diseñado una tabla (véase la Tabla 2.2) comparativa de los parámetros estimados por cada una de estas herramientas, donde las "**X**" señalan aquellos parámetros estimados por cada una de las herramientas a analizar. Se detalla el particular funcionamiento de cada una en las siguientes subsecciones.

	Parámetros estimados					Modelado		Comentarios
	Area de visión (FoV) y profundidad de campo	Personas / objetos	Almacenamiento y ancho de banda	Parámetros eléctricos e ilumination	Distancia focal	2D	3D	
AXIS Design Tool			X			X		Herramienta sencilla para proyectos de posicionamiento básicos
VideoCAD	X	X		X		X	X	Herramienta compleja con gran funcionalidad y muy visual
IP Video System Design (JVSG)	X		X		X	X	X	Herramienta avanzada para proyectos complejos de posicionamiento.
IPVM Google Maps	X	X					X	Herramienta user friendly, visual gracias a su integración con googlemaps y con buena funcionalidad

Tabla 2.2: Tabla comparativa de las diferentes herramientas de posicionamiento de cámaras existentes

2.3.1. AXIS Design Tool

Herramienta diseñada para estimar el ancho de banda y almacenamiento necesario en proyectos de vigilancia con cámaras. La herramienta permite seleccionar un escenario y unas opciones de vista, grabación y compresión del entorno en función de las cuales obtiene los parámetros indicados (véase Figura 2.6) [1].

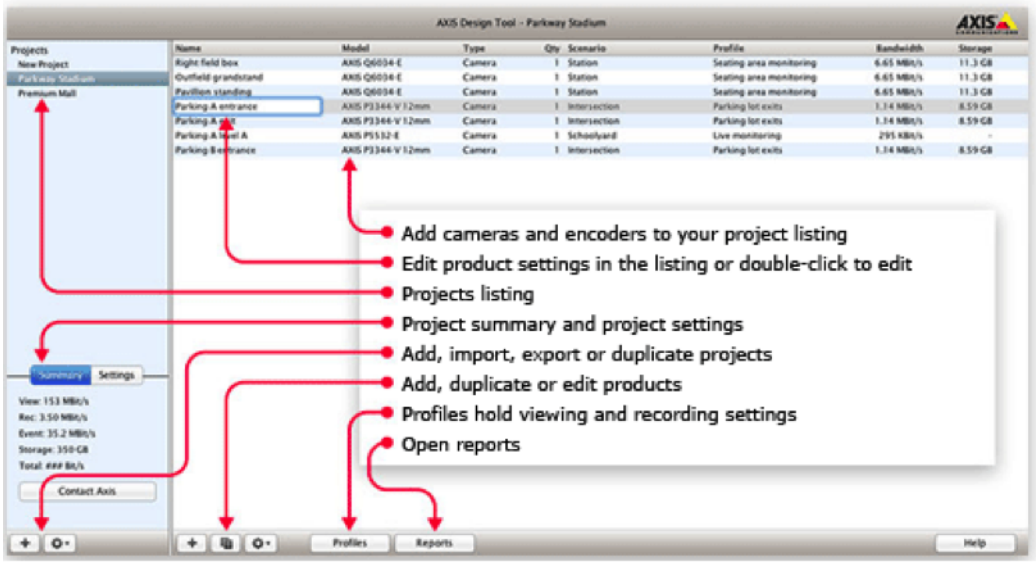


Figura 2.6: Axis Desig Tool

2.3.2. VideoCAD

Herramienta multifuncional para diseños profesionales de vigilancia por video. A través de esta herramienta podemos modelar parámetros tanto de la imagen de video como del equipamiento del mismo. Es capaz de calcular: los parámetros geométricos del área de visión de cada cámara, el área activa de visión del sensor de la cámara, proyecciones horizontales de personas/objetos, profundidad de campo, longitud y parámetros eléctricos de los cables de potencia de las cámaras e iluminación. Además, permite dos modelos de trabajo en función de la complejidad del entorno: 2D y 3D (véase Figura 2.7 extraída de [2]).

2.3.3. IP Video System Design (JVSG) integrada con VIVOTEK

Permite calcular una longitud focal precisa así como los ángulos de visión de un conjunto de cámaras en segundos, comprobando el campo de visión de cada cámara, permitiendo descubrir zonas ciegas. Usa modelos tanto 2D como 3D en función de la complejidad. Además, la herramienta permite obtener estimaciones precisas de ancho de banda, almacenamiento, etc.. Permitiendo el diseño de un sistema de vigilancia completo (véase Figura 2.8 extraída de [4]).

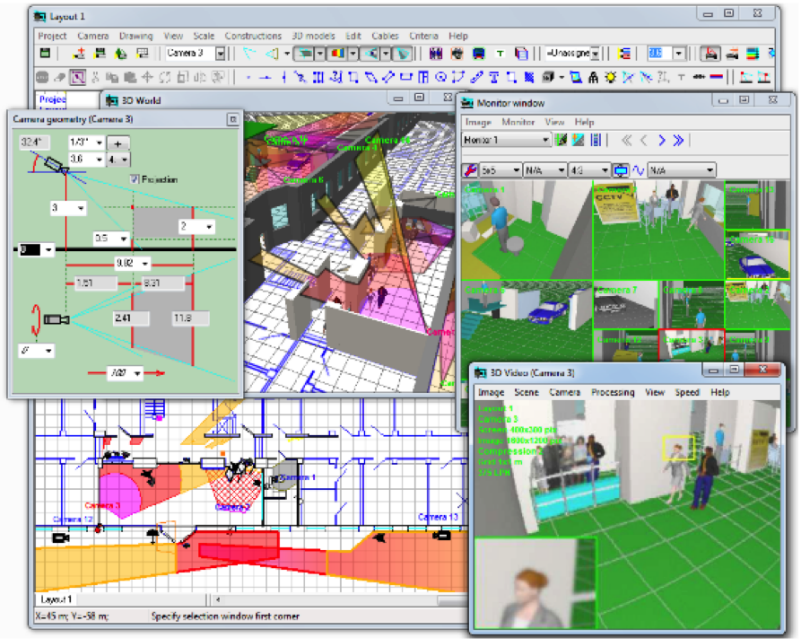


Figura 2.7: Herramienta VideoCAD

2.3.4. IPVM Google Maps Camara Calculator

Herramienta que permite diseñar sistemas de vigilancia directamente introduciendo la dirección deseada y empezando a colocar cámaras (sobre google maps). Al ir añadiendo cámaras, es capaz de mostrar el campo de visión (FoV) de las mismas sobre la superficie real, calcular la densidad de pixel (*PPM (pixel per meter / PPF (pixel per foot)*), así como la distancia y ajustes de la cámara (véase Figura 2.9 extraída de [3]).

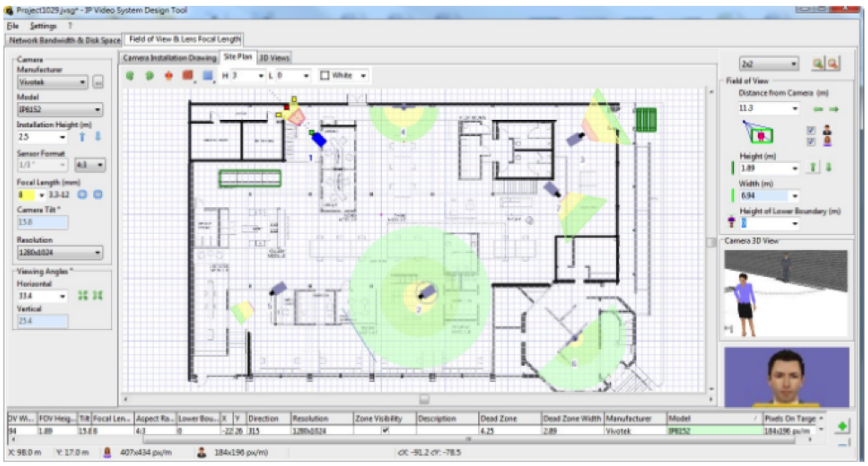


Figura 2.8: IP Video System Design (JVSG) integrada con VIVOTEK

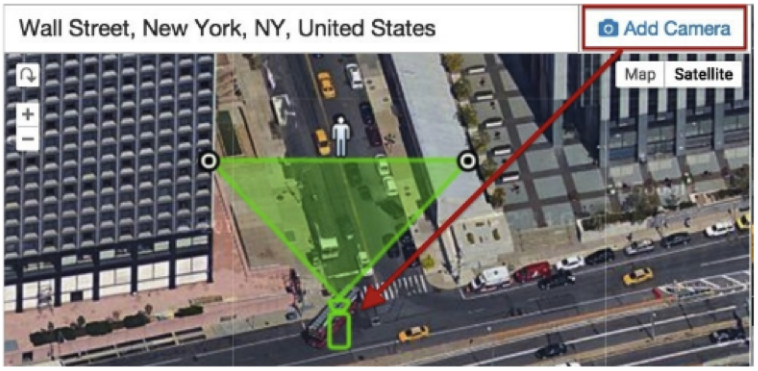


Figura 2.9: IPVM Google Maps Camara Calculator

Capítulo 3

Diseño y desarrollo

3.1. Introducción

Una vez se ha analizado el estado del arte, se busca diseñar un sistema de videovigilancia centrado en el posicionamiento de cámaras no direccionales. Adicionalmente a la idea inicial de este trabajo, basada en desarrollar un sistema de posicionamiento 2D de cámaras, se ha decidido incluir el factor altura en la implementación de la herramienta, por lo que a partir de este punto se pasará a analizar el posicionamiento 3D de cámaras, en el plano XYZ.

Toda la herramienta se basa en dos algoritmos clave: cálculo del campo de visión y optimización de búsqueda de candidatas, entendiendo por candidatas las posibles posiciones de la cámara.

Para el desarrollo y programación de este trabajo se ha hecho uso de los siguientes toolbox y funciones asociadas de *Matlab*:

- Función viewshed, del toolbox "*MappingToolbox*" [10], utilizada en el cálculo del campo de visión.

- Funciones fmincon y patternsearch, del toolbox "*OptimizationToolbox*" [11] de *Matlab*, utilizadas como funciones de optimización para el análisis de las posiciones de las cámaras.

El flujo de trabajo del sistema se basa en cinco fases: introducción de los parámetros de entrada, preprocesado de la imagen, división en bloques para procesamiento, optimización y decisión final. Este proceso queda reflejado en la Figura 3.1.

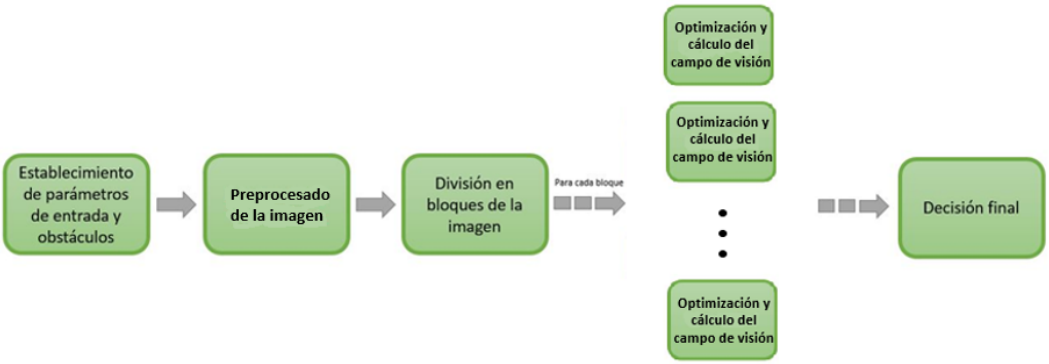


Figura 3.1: Flujo de trabajo del programa implementado.

3.2. Campo de visión

El campo de visión (FoV) es el volumen dentro del cual los objetos pueden ser proyectados en el plano de imagen de una cámara.

Tal y como se ha comentado, se ha decidido utilizar la función *viewshed*, del "*MappingToolbox*" [10], para el cálculo del campo de visión en el desarrollo de este trabajo.

Originalmente esta función estaba diseñada para calcular el campo de visión a partir de un mapa de coordenadas espaciales, un mapa de alturas y la latitud y longitud de la cámara sobre dicho mapa. Puesto que en este trabajo se van a utilizar mapas de suelo y no de coordenadas espaciales, se ha diseñado una adaptación de la función *viewshed* (*voxelviehshed*) de forma que se puedan definir las coordenadas del mapa XYZ y calcular el campo de visión dada la posición de una cámara en píxeles (*camxyz*):

$$vis = voxelviewshed(X, Y, Z, camxyz)$$

Para evaluar el mapa de suelo se obtendrán los puntos de cuadrícula utilizando la función *meshgrid* de Matlab, que dará como resultado los puntos X,Y a utilizar como valores de entrada de la función *viewshed* adaptada. Adicionalmente, como mapa de alturas, utilizaremos las alturas de los objetos introducidos por el usuario, en caso de existir.

Por último, para la representación del cálculo de visión se ha utilizado un modelo de representación 3D que permite visualizar la posición de cada cámara y el campo de visión asociado a la misma. A continuación se expone una imagen ejemplo del campo de visión calculado por la función *voxelviewshed* (véase Figura 3.2).

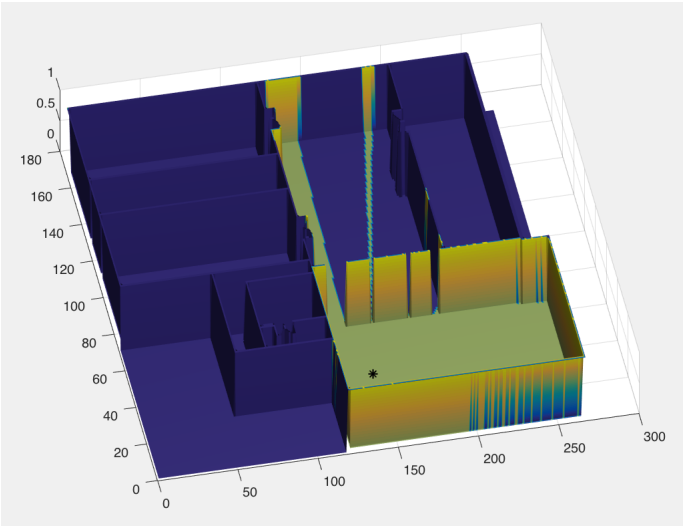


Figura 3.2: Ejemplo de campo de visión calculado a traves de la funcion *voxelviewshed* implementada.

3.3. Modos de operación

En función del escenario y las restricciones fijadas por el usuario, se han programado dos modos de operación. Estos dos modos de operación se asocian directamente a las ya explicadas formulación continua y discreta del problema (véase Figura 2.2 del Capítulo 2).

- Cálculo del campo de visión máximo acumulado, dado un número inicial de cámaras (Formulación continua).
- Cálculo de las posiciones de las cámaras necesarias para cubrir una zona crítica del mapa (Formulación discreta).

3.3.1. Cálculo del campo de visión máximo acumulado

Se entiende por campo de visión acumulado al conjunto de volumen espacial total cubierto por varias cámaras a la vez. El flujo de funcionamiento de este modo se resume en los siguientes pasos:

1. Se determinan los parámetros de entrada: el usuario selecciona el mapa de suelo a analizar, los obstáculos existentes en el mismo y el número de cámaras del que se dispone.
2. Procesamiento de la imagen: debido al elevado coste computacional que suponen ciertos mapas de suelo debido a su tamaño, se realiza un redimensionamiento

de la imagen por diferente factor, buscando una imagen de, como máximo, 100 pixeles en su largo/ancho. Además, para evitar pérdida de información se pasa la imagen a binario y se realiza una apertura posterior al redimensionado en caso de ser necesario para minimizar la pérdida de detalle.

3. Para la primera cámara, se divide la imagen procesada en un grid de 10x10 bloques (véase Figura 3.3) para tener la máxima granularidad posible y maximizar la probabilidad de encontrar la posición óptima de la cámara. Para cada bloque, se aplicará una técnica de optimización para encontrar aquella localización dentro del bloque cuyo campo de visión sea máximo en el conjunto de toda la imagen. Según se va iterando entre bloques, la localización de la posición cuyo campo de visión es máximo se irá actualizando siempre que el campo de visión de la nueva localización sea mayor que el anterior. Como resultado de este proceso, tendremos la localización óptima de la primera cámara.

4. Para el resto de cámaras, se partirá de un campo de visión inicial y se recorrerá de nuevo el grid por bloques buscando, mediante optimización, aquella posición cuyo campo de visión junto con el campo de visión inicial/acumulado (en función de la iteracción en la que se encuentre el algoritmo) sea máximo. Como resultado de este proceso, tendremos el resto de localizaciones de las cámaras, que en conjunto con la primera cámara resultan en un campo de visión acumulado máximo sobre el total de la imagen.

3.3.2. Cálculo de las posiciones de las cámaras necesarias para cubrir una zona crítica del mapa

El flujo de funcionamiento de este modo, descrito gráficamente en la Figura 3.4, se resume en los siguientes pasos:

1. Se determinan los parámetros de entrada: el usuario selecciona el mapa de suelo a analizar, los obstáculos existentes en el mismo y la zona específica a cubrir del mapa.

2. Procesamiento de la imagen: debido al elevado coste computacional que suponen ciertos mapas de suelo debido a su tamaño, se realiza un redimensionamiento de la imagen por un factor variable, buscando una imagen de, como máximo, 100 pixeles en su largo/ancho. Además, para evitar pérdida de información se pasa la imagen a binario y se realiza una apertura posterior al redimensionado en caso de ser necesario.

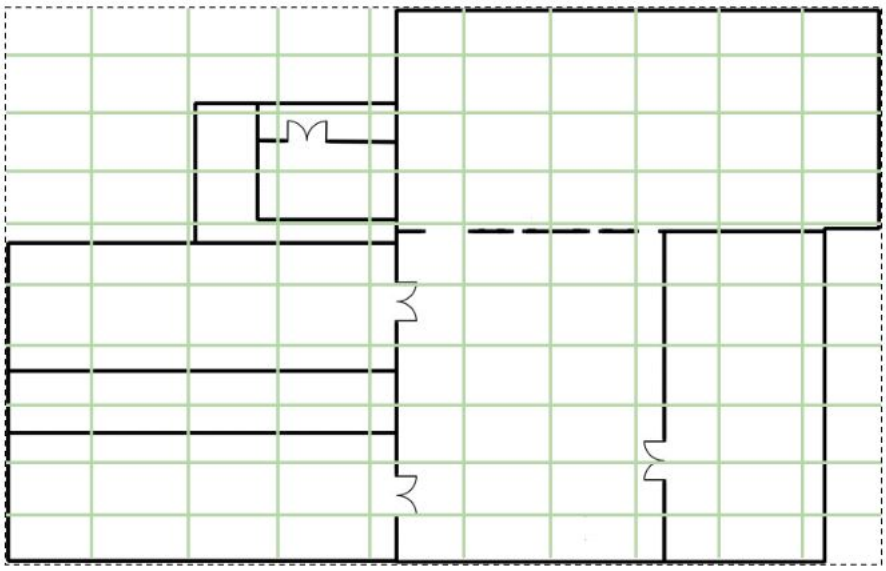


Figura 3.3: Grid 10x10 utilizado para la búsqueda de localizaciones candidatas por bloques.

3. Se genera una máscara con las dimensiones de la zona a visualizar seleccionada por el usuario.

4. Para la primera cámara, se divide la imagen procesada en un grid de 10x10 bloques (véase Figura 3.3) con el fin de tener la máxima granularidad posible y maximizar la probabilidad de encontrar la posición óptima de la cámara. Para cada bloque, se aplicará una técnica de optimización para encontrar aquella localización dentro del bloque cuyo campo de visión cubra el máximo de la máscara. Según se va iterando entre bloques, la localización de la posición cuyo campo de visión sea máximo sobre la máscara se irá actualizando. Como resultado de este proceso, tendremos la localización óptima de la primera cámara.

5. Para el resto de cámaras, en caso de que la máscara no quede totalmente cubierta, se generará una nueva máscara excluyendo la zona ya cubierta y se recorrerá de nuevo el grid por bloques buscando, mediante optimización, aquella posición cuyo campo de visión cubra la nueva máscara. Como resultado de este proceso, tendremos el resto de localizaciones de las cámaras, que en conjunto con la primera cámara resultan en una cobertura total de la zona objetivo.

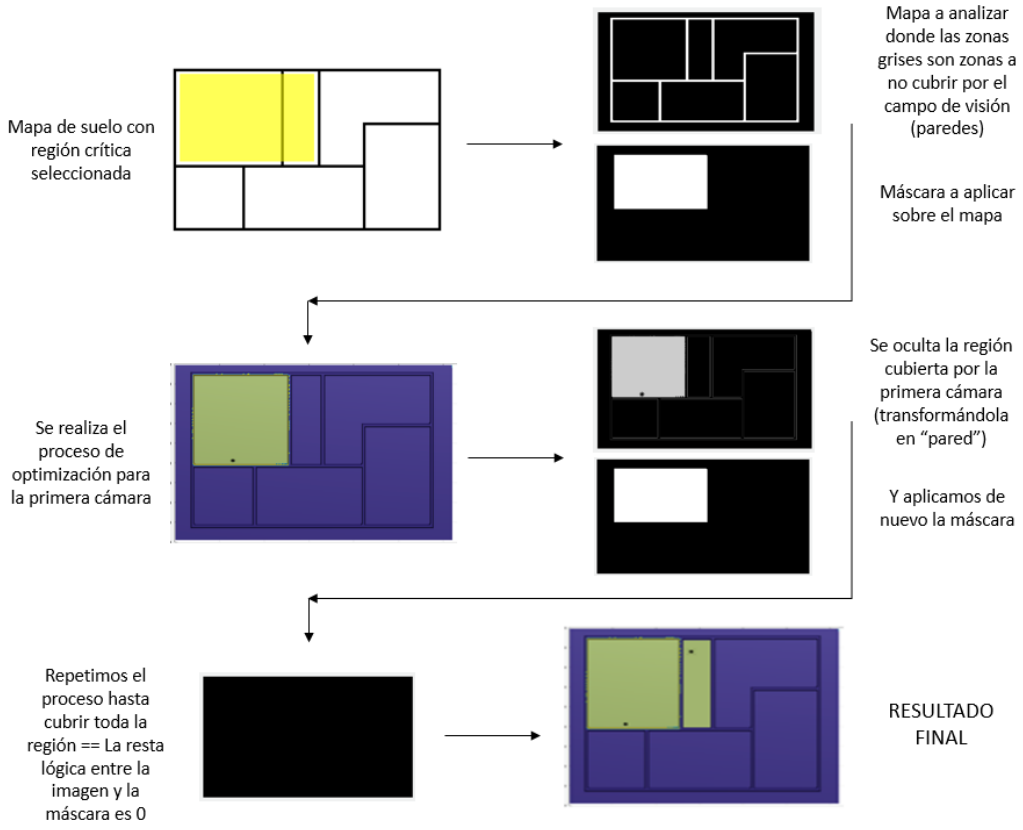


Figura 3.4: Proceso de cálculo del número de cámaras para cubrir una región crítica

3.4. Técnicas de optimización

La necesidad de utilizar una técnica de optimización en este trabajo radica en la inviabilidad de evaluar todas las posibles localizaciones de una cámara sobre el mapa. Incluso considerando un mapa de suelo reducido, el tiempo de coste computacional resultaría muy elevado.

Para solventar este problema, se han implementado múltiples técnicas de optimización para la evaluación de localizaciones candidatas, considerando como candidatos óptimos los puntos XY sobre el mapa de suelo que optimizan las restricciones del *coverage* fijadas.

Técnicas de optimización evaluadas e implementadas:

- Optimización propia basada en el ratio de obstáculos
- Técnica de optimización perteneciente al toolbox de optimización de Matlab: *fmincon*[12]
- Técnica de optimización perteneciente al toolbox de optimización de Matlab: *fminunc* [13]
- Técnica de optimización perteneciente al toolbox de optimización de Matlab: *patternsearch* [14]

3.4.1. Optimización propia

Como punto de partida se ha desarrollado una técnica de optimización propia basada en el ratio de obstáculos, considerando obstáculos tanto las paredes del mapa como los propios obstáculos introducidos por el usuario.

Mediante esta técnica, para cada bloque del grid [Figura 3.3] se calcula el ratio de obstáculos que presenta y en función de este valor se evalúan más o menos posibles candidatas dentro de dicha cuadrícula. De esta forma, para una cuadrícula con muchos obstáculos se evaluará un número mayor de posibles candidatas (posiciones a evaluar calculadas mediante una función de aleatoriedad) con el fin de cubrir la máxima zona evitando el obstáculo; en el caso de no haber obstáculos parece lógico evaluar menos puntos puesto que la problemática de cubrir el máximo campo de visión se reduce.

3.4.2. Optimización Fmincon

Función de optimización incluida en el toolbox de optimización de Matlab. Se trata de una función no lineal con restricciones, en lo relativo a un número finito de

soluciones, fijando el punto de partida y punto de finalización de la búsqueda. Esta función busca el mínimo de un problema especificado por:

$$\min_x \ f(x), \tag{3.1}$$

siendo $f(x)$ una función que devuelve un escalar, y x el vector o matriz a minimizar [12].

La nomenclatura de la función del toolbox utilizada es:

$$x = fmincon(fun,\ x0), \tag{3.2}$$

siendo x el mínimo de la función objetivo *fun*. *x0* es el vector real que especifica el punto inicial de búsqueda.

En este caso, siendo la variable la candidata a localización optima de cada cámara, dicha función buscará:

- El inverso del campo de visión máximo acumulado, en el caso de que la restricción del *coverage* sea el cálculo del campo de visión máximo.
- El mínimo de la diferencia entre el campo de visión acumulado y una máscara de la región a cubrir o visualizar, en el caso de que la restricción del *coverage* sea cubrir una región determinada del espacio.

Como se ha comentado, adicionalmente habrá que fijar los límites de búsqueda, es decir, los límites superior e inferior del bloque de la imagen.

3.4.3. Optimización Fminunc

Función de optimización incluida en el toolbox de optimización de Matlab. En contraposición a la anterior función de optimización (fmincon), se trata de una función no lineal sin restricciones, en lo relativo a la consideración de un número infinito de soluciones, que busca el mínimo global de un problema especificado por una función objetivo f [13].

$$\min_x \ f(x), \tag{3.3}$$

siendo $f(x)$ una función que devuelve un escalar, y x el vector o matriz a minimizar.

La nomenclatura de la función del toolbox de Matlab utilizada es:

$$x = fminunc\,(fun,\ x0)\,, \tag{3.4}$$

siendo x el mínimo de la funcion objetivo *fun*. *x0* es el vector real que especifica el punto inicial de búsqueda.

En este caso, siendo la variable la candidata a localización óptima de cada cámara, dicha función buscará:

- El inverso del campo de visión máximo acumulado, en el caso de que la restricción del *coverage* sea el cálculo del campo de visión máximo.
- El mínimo de la diferencia entre el campo de visión acumulado y una máscara de la región a cubrir o visualizar, en el caso de que la restricción del *coverage* sea cubrir una región determinada del espacio.

Puesto que el número de puntos candidatos a evaluar es muy elevado, en los resultados se debería reflejar una mejora con el uso de esta función de optimización en contraposición a fmincon.

3.4.4. Optimización Patternsearch

Función de optimización incluida en el toolbox de optimización de Matlab. Se trata de una función de optimización que busca el mínimo local de un problema especificado por una función objetivo y un punto inicial de búsqueda:

$$\min_{x}\ f\,(x)\,, \tag{3.5}$$

siendo $f(x)$ una función que devuelve un escalar, y x el vector o matriz a minimizar [14].

La nomenclatura de la función del toolbox de Matlab utilizada es:

$$x = patternsearch\,(fun,\ x0)\,, \tag{3.6}$$

siendo x el mínimo de la función objetivo *fun*. *x0* es el vector real que especifica el punto inicial de búsqueda.

En este caso, siendo la variable de la función objetivo la candidata a localización optima de cada cámara, dicha función buscará:

- El inverso del campo de visión máximo acumulado, en el caso de que la restricción del *coverage* sea el cálculo del campo de visión máximo.
- El mínimo de la diferencia entre el campo de visión acumulado y una máscara de la región a cubrir o visualizar, en el caso de que la restricción del *coverage* sea cubrir una región determinada del espacio.

Esta función es especialmente útil cuando el índice de variación entre iteraciones de la función de optimización es bajo, de forma que evita la convergencia hacia el punto inicial de búsqueda. En la sección de resultados se corroborará con datos cómo esta función de optimización será la que más se aproxima al objetivo óptimo en todas las variantes del *coverage*, puesto que, debido al entorno de aplicación de este trabajo, el índice de variación es bajo. Esto cambiaría en caso de incluir entre las funciones de *coverage* el tracking de un objetivo móvil, en donde la variabilidad del entorno aumentaría considerablemente.

3.5. Interfaz gráfica

Como intermediario entre el usuario y el programa implementado, se ha desarrollado una interfaz gráfica que permite al usuario:

- La introducción de objetos dentro del mapa seleccionado, así como la fijación de la altura de los mismos.
- La selección de los parámetros de entrada:
  - ✧ Modo de operación (véase Sección 3.3) en función de la finalidad de uso para cada entorno.
  - ✧ Técnica de optimización (véase Sección 3.4).

3.5.1. Introducción de obstáculos

Tras la selección del mapa, el usuario será capaz de añadir obstáculos en caso de creerlo necesario para simular su entorno real, simplemente clickando en la región del mapa que aparece en pantalla (véase Figura 3.5).

El usuario final podrá introducir tantos objetos como considere necesario (véase Figura 3.6).

3.5.2. Desarrollo y opciones de ejecución

Por último, el usuario será capaz de determinar el modo de operación y el tipo de técnica a utilizar (véase Figura 3.7).

3.6. Conclusiones

Concluyendo, se ha desarrollado un toolbox de posicionamiento 2D de cámaras que resuelve la problemática de determinar la posición óptima de las cámaras en función de las variables del entorno y el objetivo final. La gran ventaja de esta herramienta reside en la posibilidad de personalización, de manera que el usuario es capaz de determinar y/o modificar las variables indicadas en la Tabla 3.1.

Variables	Incluidas en la herramienta desarrollada
Mapa de suelo a analizar	**SI**
Posición de los posibles obstáculos	**SI**
Altura de los obstáculos	**SI**
Número de cámaras	**SI**
Ángulo de visión de la cámara	**FIJO (360º)**
Rotación de la cámara	**NO**
Modo de operación	**SI**
Técnica de optimización a usar	**SI**
Región crítica de cobertura	**SI**

Tabla 3.1: Variables que se pueden modificar en el toolbox desarrollado.

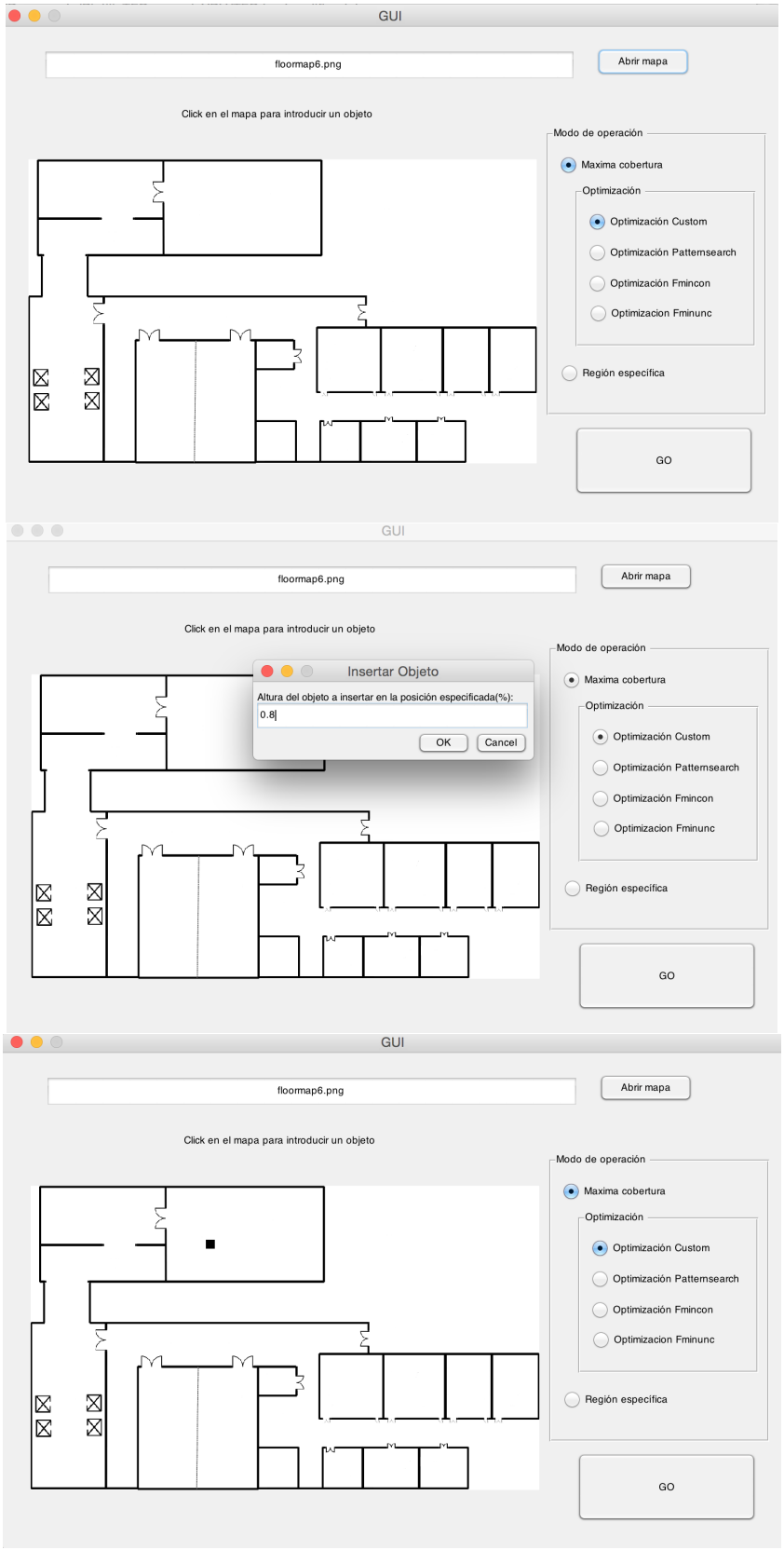


Figura 3.5: Ejemplo de introducción de objetos y transiciones entre los diferentes pasos que seguirá el usuario.

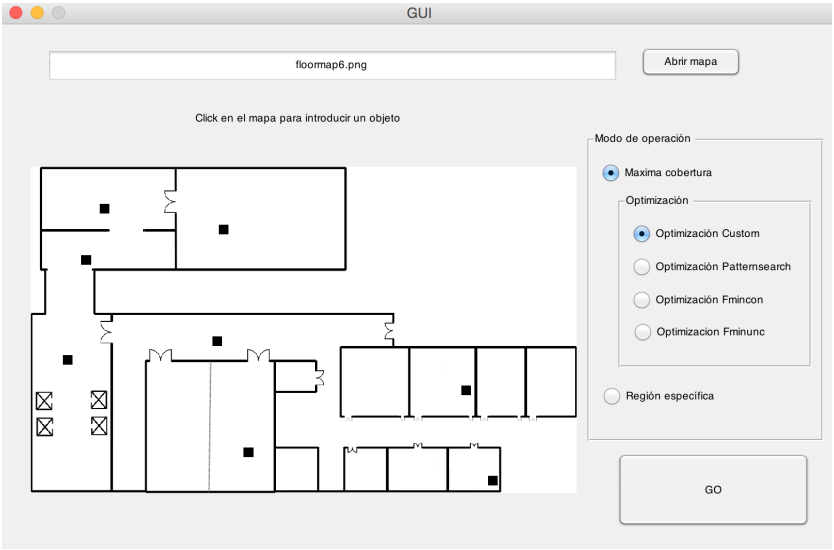


Figura 3.6: Ejemplo de inserción de múltiples objetos en el mapa.

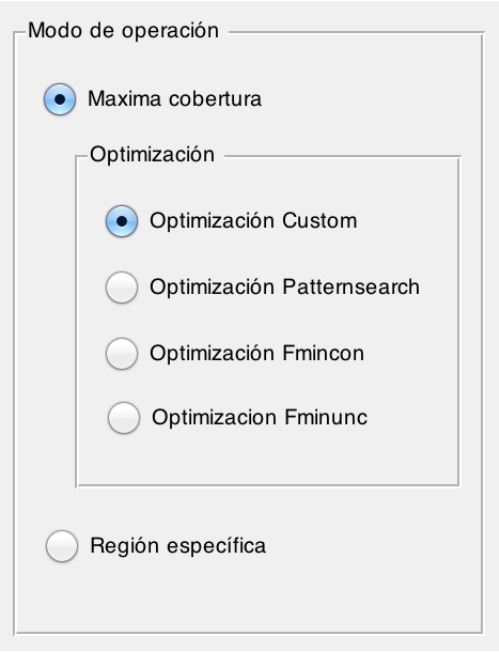


Figura 3.7: Cuadro de elección de modo de operación y técnica de optimización a utilizar.

Capítulo 4

Evaluación

4.1. Introducción

Una vez terminado el desarrollo de la herramienta se ha procedido a realizar una variedad de pruebas correspondientes a los distintos elementos del dataset, así como a las diferentes posibilidades que ofrece la herramienta. Con el fin de evaluar los resultados obtenidos se ha medido el tiempo de ejecución y el campo de visión, este último de manera numérica y gráfica.

Para la obtención de las medidas cuantitativas de tiempo de ejecución se ha utilizado un equipo con un procesador Intel® Core i5, 2.6GH de CPU y 8GB de RAM. El sistema operativo es OS X Yosemite versión 10.10.5.

4.2. Marco de evaluación

4.2.1. *Dataset*

Actualmente, no existe en el ámbito del posicionamiento de cámaras ningún *dataset* oficial que contenga diferentes mapas de suelo con los que probar la funcionalidad del trabajo. Debido a esta casuística, se han obtenido una serie de mapas de suelo de internet, y se han adaptado a las características requeridas por el trabajo, eliminando texto y figuras que alterasen los resultados finales.

A continuación se exponen los mapas de suelo utilizados en esta evaluación (véase Figura 4.1).

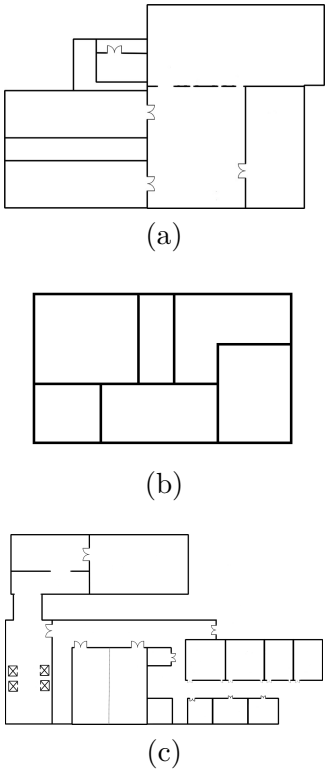


Figura 4.1: Imágenes del dataset utilizado: (a) Mapa 1 (b) Mapa 2 (c) Mapa 3.

4.2.2. Métricas

Para la evaluación de los algoritmos se utilizarán diferentes métricas en función del modo de operación seleccionado por el usuario:

1. Para el modo de cálculo del campo de visión máximo acumulado se ha desarrollado una función en *Matlab* que evalúa el porcentaje de mapa cubierto por todas las cámara.

2. Para el modo de visualización de una zona específica del mapa, la evaluación de los diferentes algoritmos se medirá por el número de cámaras, así como su posición, necesarias para cubrir la región específica seleccionada.

En ambos casos, los resultados estarán apoyados:

- Por una representación gráfica del campo visual cubierto así como de la posición exacta sobre el mapa de las cámaras establecidas.

- Por el coste de procesamiento del algoritmo (medido en unidades de tiempo de ejecución).

4.2.3. Algoritmos comparados

Como ya se ha explicado en el apartado de diseño, los algoritmos comparados se basan en el tipo de técnica de optimización: Optimización propia, Optimización *paternsearch*[14], Optimización *fmincon*[12], Optimización *fminunc*[13].

La idoneidad de los resultados de estas técnicas dependerá también de las características del entorno (principalmente el grado de variabilidad del mismo) y del modo de operación seleccionado.

4.3. Pruebas y resultados

Se ha decidido dividir el análisis de los resultados en función del modo de operación seleccionado por el usuario.

En la última subsección se extraerán las conclusiones y comparativas de todos los modos y técnicas de optimización. Para estas conclusiones se han tenido en cuenta todos los resultados agrupados en los Anexos A y B con el fin de ser lo más fieles posibles a la realidad de la herramienta desarrollada.

4.3.1. Resultados asociados al cálculo del campo de visión máximo acumulado

El objetivo de esta sección es exponer, de manera tanto numérica como gráfica, una comparativa de los resultados obtenidos en función de la técnica de optimización utilizada para el modo de operación por el cual el usuario determinará un número inicial de cámaras como parámetro inicial con el fin de obtener el posicionamiento 2D de las mismas que optimice el cálculo del campo de visión máximo acumulado.

Con el fin de facilitar el análisis al lector, se procede a dividir los resultados obtenidos en función de la técnica de optimización seleccionada.

A modo de resumen, en esta sección de la memoria se exponen los datos correspondientes a 2 mapas del dataset, y un número de cámaras inicialmente establecido de 2, 3 y 4 cámaras. En ambos casos, para una análisis completo, tanto en variedad de mapas como número de cámaras diríjase al Anexo A.

Como aclaración referente al tiempo de procesamiento asociado al número de cámaras utilizadas, se ha concluido que aumenta exponencialmente al número de cámaras, por lo que para cierto algoritmos, un número mayor a 7 cámaras resulta en un tiempo de procesamiento muy elevados.

Los resultados obtenidos para las diferentes técnicas de optimización quedan reflejados en las figuras (4.2) y (4.3).

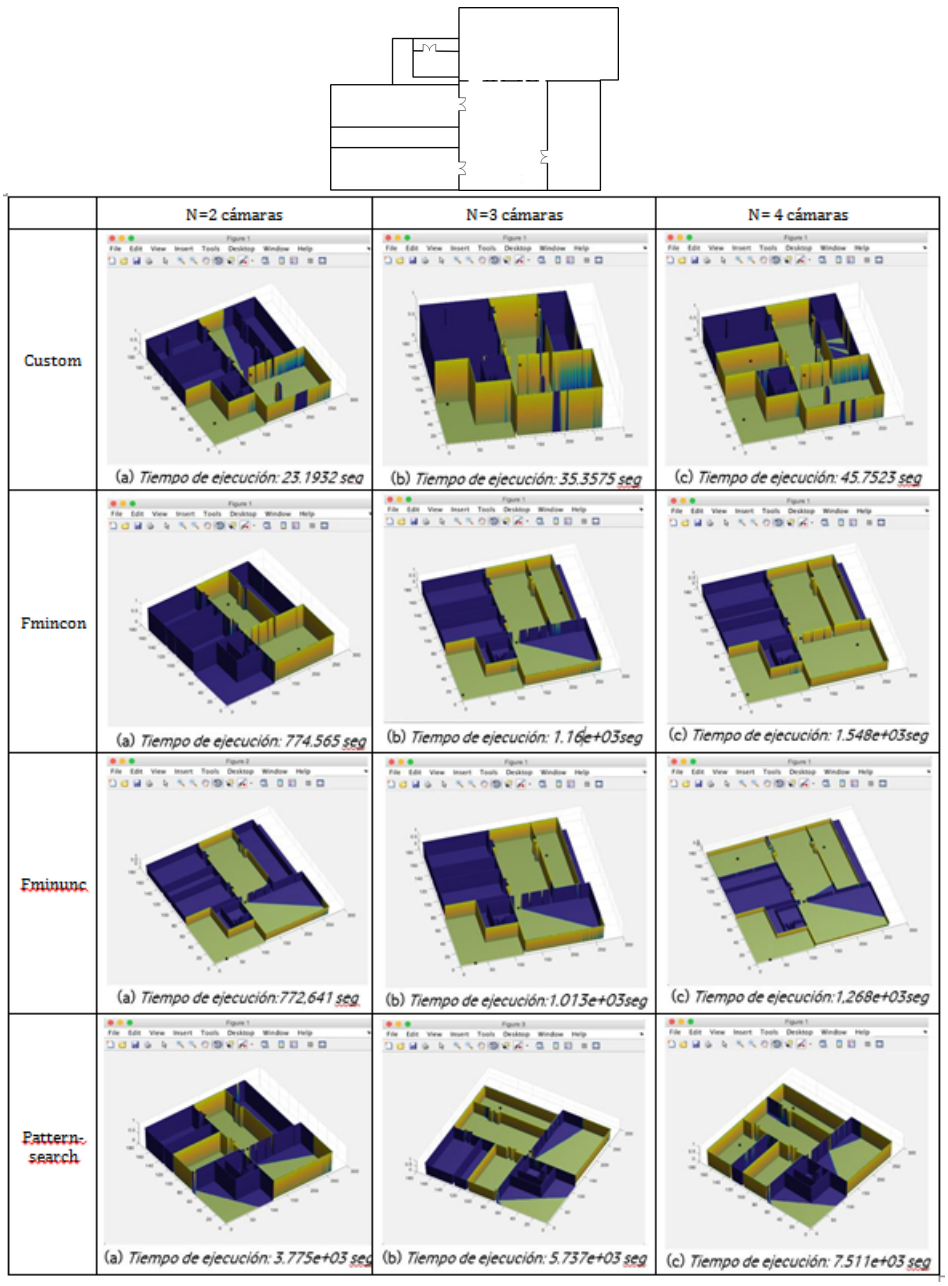
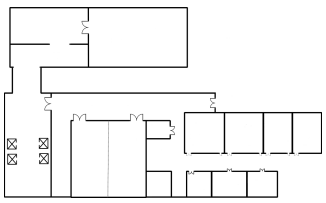


Figura 4.2: Partiendo del mapa de suelo indicado al inicio de la figura, se muestran los resultados obtenidos asociados a cada método de optimización (a) Representación gráfica y tiempo de procesamiento para 2 cámaras. (b) Representación gráfica y tiempo de procesamiento para 3 cámaras. (c) Representación gráfica y tiempo de procesamiento para 4 cámaras. Donde '*' es la posición de la cámara; la zona amarilla es el campo de visión (conjunto) de las cámaras; y la zona azul es el campo oculto no visible por las cámaras.



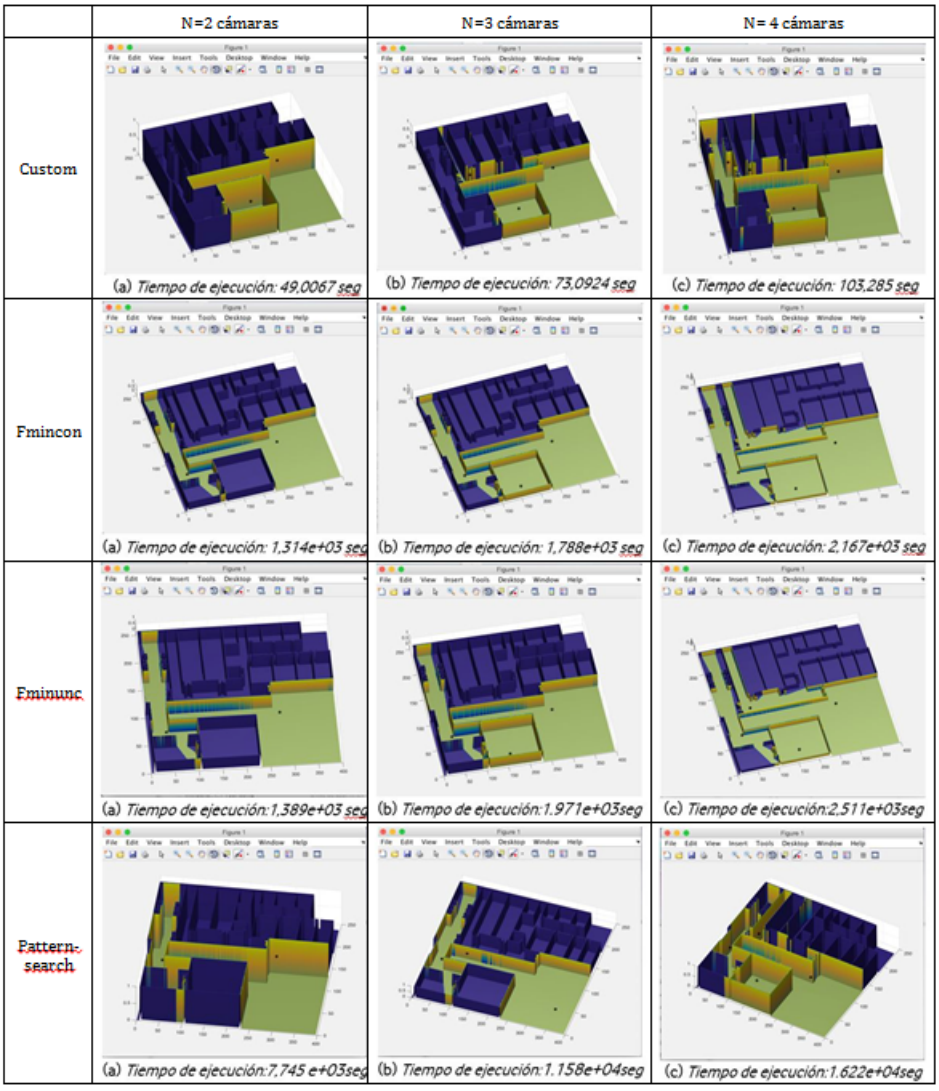


Figura 4.3: Partiendo del mapa de suelo indicado al inicio de la figura, se muestran los resultados obtenidos asociados a cada método de optimización (a) Representación gráfica y tiempo de procesamiento para 2 cámaras. (b) Representación gráfica y tiempo de procesamiento para 3 cámaras. (c) Representación gráfica y tiempo de procesamiento para 4 cámaras. Donde '*' es la posición de la cámara; la zona amarilla es el campo de visión (conjunto) de las cámaras; y la zona azul es el campo oculto no visible por las cámaras.

Con los resultados obtenidos se pueden sacar las siguientes conclusiones:

1. Con respecto a la optimización propia, se observa como en todos los mapas obtienen unos resultados aceptables con un tiempo de ejecución óptimo, suponiendo una velocidad de procesado ~30 menor que el 2º algoritmo más rápido. Por tanto, este método resulta útil en escenarios donde la restricción es el elevado tiempo de procesado.

2. Los algoritmos de optimización *fmincon* y *fminunc* devuelven resultados prácticamente iguales. Esta casuística es debida a que se trata de dos optimizaciones válidas para escenarios con alta variabilidad, sin embargo, en nuestro caso, los pasos de búsqueda son muy pequeños y los escenarios constantes, hecho que provoca que las funciones converjan a sus valores iniciales, es decir, a los puntos iniciales de búsqueda cada sección del grid, y a una altura '0'. Como restricción adicional, se debe considerar el elevado tiempo de procesado asociado a estas optimizaciones, al igual que se observa con *patternsearch*.

3. Para solventar el problema de convergencia presente en las funciones de optimización *fmincon* y *fminunc*, se ha hecho uso de la función *patternsearch*, caracterizada por no converger ante escenarios con baja variabilidad a costa de un aumento del tiempo de ejecución. Como se observa en los resultados mostrados en la sección anterior, los resultados de la optimización *patternsearch* resultan óptimos para aquellos escenarios en los que se requiere un alta precisión y exactitud en el posicionamiento de las cámaras, y el factor de tiempo de procesado sea secundario.

A continuación se expone una tabla ilustrativa (véase Cuadro 4.1) que incluye los tiempos de procesado y los porcentajes de área cubierta por el campo de visión total acumulado para cada técnica de optimización:

4.3.2. Resultados asociados a la visualización de una zona crítica del mapa

El objetivo de esta sección es exponer, de manera tanto numérica como gráfica, una comparativa de los resultados obtenidos para el modo de operación por el cual el usuario seleccionará una región específica del mapa a cubrir por el campo de visión de las cámaras, obteniendo el posicionamiento 3D del número mínimo (y óptimo) de cámaras necesarias para visualizar la región seleccionada.

Teniendo en cuenta las conclusiones obtenidas en la sección anterior, se ha considerado óptimo el utilizar, este caso, solo la técnica de optimización *patternsearch*, la

Tabla 4.1: Tabla comparativa de las diferentes técnicas de optimización en relación al porcentaje de cobertura del campo de visión y el tiempo de ejecución.

	2 cámaras		3 cámaras		4 cámaras	
	% de cobertura	Tiempo de procesado	% de cobertura	Tiempo de procesado	% de cobertura	Tiempo de procesado
Optimización propia	46,28 %	**49,007 seg**	54,26 %	**56,093 seg**	62,03 %	**103,28 seg**
Optimización fmincon	49,39 %	1314,3 seg	55,13 %	1788,7 seg	63,41 %	2167 seg
Optimización fminunc	49,39 %	1389,4 seg	55,13 %	1970,9 seg	63,41 %	2511 seg
Optimización patternsearch	**49,61 %**	7745 seg	**56,23 %**	11588 seg	**63,87 %**	16222 seg

cual se ha demostrado ser la más precisa y por tanto la más útil en el caso concreto de buscar el número mínimo de cámaras para conseguir la cobertura de una región específica; descartando por tanto las funciones que convergen y/o menos exactas.

De manera espejo a la sección anterior, los datos se muestran únicamente para 2 mapas del *dataset*. Para un análisis completo de los resultado diríjase al Anexo B.

Los resultados obtenidos para esta técnica de optimización quedan reflejados en la Figura (4.4).

4.4. Conclusión

A través de las pruebas realizadas, se puede concluir que la herramienta más precisa será aquella que use la función patternsearch de optimización, siempre y cuando el tiempo de procesamiento no sea una restricción. En este último caso habrá que valorar el nivel de restricción que supone el tiempo para cada entorno, y por tanto, cual sería la función de optimización óptima para cada caso.

Se ha buscado apoyar las conclusiones obtenidas a través de las pruebas realizadas con una tabla numérica ilustrativa (véase Cuadro 4.1), en dónde se muestra el porcentaje de mapa cubierto por el campo de visión para cada técnica de optimización, en función del número de cámaras inicialmente fijado; así como el tiempo de ejecución ya indicado en cada prueba de las secciones anteriores. Como es obvio, esta comparativa tiene sentido para el cálculo del campo de visión máximo acumu-

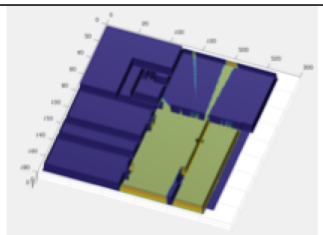
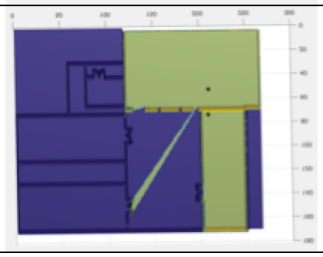
Mapa y región crítica	Resultado	Nº de cámaras necesarias	Tiempo de procesado
		2	1,498e+03 seg
		2	47,221 seg

Figura 4.4: Ejemplo de zonas seleccionadas a cubrir por el usuario y el correspondiente posicionamiento 3D de las cámaras necesarias para conseguir el resultado óptimo con la técnica de optimización *patternsearch*. Dónde '*' es la posición de la cámara; la zona amarilla es el campo de visión (conjunto) de las cámaras; y la zona azul es el campo oculto no visible por las cámaras.

lado y se pretende obtener una comparación numérica de las diferentes técnicas de optimización.

Observando la Tabla 4.1, y aclarando que los incrementos y diferenciales entre porcentajes, en general, se mantienen para los diferentes mapas, se puede concluir que la herramienta más precisa será aquella que use la función patternsearch de optimización, siempre y cuando el tiempo de procesamiento no sea una restricción. En este último caso habrá que valorar el nivel de restricción que supone el tiempo para cada entorno, y por tanto, cual sería la función de optimización óptima para cada caso.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

El objetivo principal de este proyecto ha sido el desarrollo de una herramienta de posicionamiento 3D de cámaras, basada en funciones de *Matlab*, que permite el cálculo de las posiciones óptimas de las cámaras sobre un mapa de suelo, tanto si el objetivo es cubrir el máximo espacio posible con un número limitado de cámaras, como si el fin resulta en cubrir una región específica del mapa.

Partiendo de estas premisas, el resultado final se basa en el desarrollo de un toolbox compuesto por una variedad de funciones que permiten al usuario calcular de manera precisa y visual el posicionamiento deseado de las cámaras.

Como limitaciones de la herramienta nos encontramos con un ángulo de visión fijo de la cámara (360^{o}) y por tanto una orientación fija de las mismas, lo que imposibilita la evaluación de una rotación o movimiento de las cámaras sobre su eje a la hora de determinar las posiciones óptimas.

5.2. Trabajo futuro

A la vista de los resultados que se han obtenido en este trabajo se propone trabajar en:

- Adaptar al herramienta para posibilitar la utilización de cámaras direccionales, introduciendo la posibilidad de variación del ángulo de visión de la cámara como parámetro de entrada a fijar por el usuario.
- Incluir un nuevo modo de operación basado en el seguimiento (tracking) de un objetivo móvil por el espacio reevaluando la función de optimización e incorporando nuevos valores de medida de resolución que permitan implementar un

nivel de solapamiento entre campos de visión de cámaras vecinas suficiente para obtener un tracking útil de un objeto en movimiento.

- Incluir una rotación de la cámara sobre su eje con el fin de fijar la orientación óptima de la misma y no solo la posición.

Bibliografía

[1] A. Communications, "Axis design tool user's guide." 2, 12

[2] CCTVCAD, "Cctvcad lab toolkit - tools for practical measuring cctv camera parameters." 2, 13

[3] IPVM, "Camera design tool with google maps integration." 2, 14

[4] Vivotek, "Ip video system design tool - intuitively design surveillance systems with 2d/3d views." 2, 13

[5] V. Chvatal, "A combinatorial theorem in plane geometry," *Journal of Combinatorial Theory, Series B*, vol. 18, no. 1, pp. 39–41, 1975. 5, 8, 9, 10, 11

[6] A. Communications, "CÃ¡mara de red axis m3007-pv." 6

[7] J. Marinetto, "Cobertura de un objetivo." 8

[8] C. F. Junbin Liu, Sridhar Sridharan, "Recent advances in camera planning for large area surveillance: A comprehensive review," *ACM Computing Surveys (CSUR)*, vol. 49, no. 6, 2016. 9, 10, 11

[9] A. C. Hamid Aghajan, *Multi-Camera Networks: Principles and Application*. Elsevier, 2009. 10

[10] MATLAB, "Mapping toolbox." 17, 18

[11] MATLAB, "Optimization toolbox." 17

[12] MathWorks, "Fmincon optimization function." 23, 24, 33

[13] MathWorks, "Fminunc optimization function." 23, 24, 33

[14] MathWorks, "Patternsearch optimization function." 23, 25, 33

Apéndice A

Resultados de las pruebas para el cálculo del campo de visión máximo acumulado

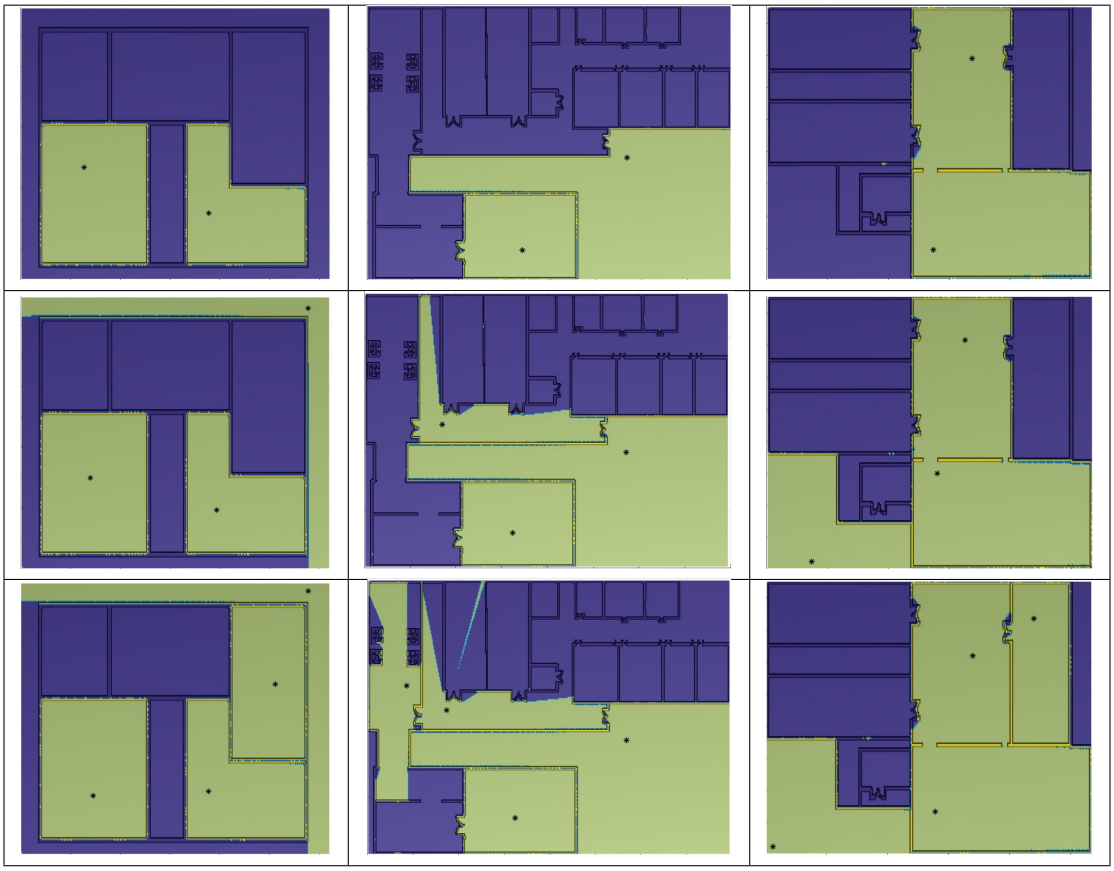


Figura A.1: Resultados de las pruebas asociados a la optimización propia dónde cada fila contiene las imágenes relativas a 2, 3 y 4 cámaras respectivamente.

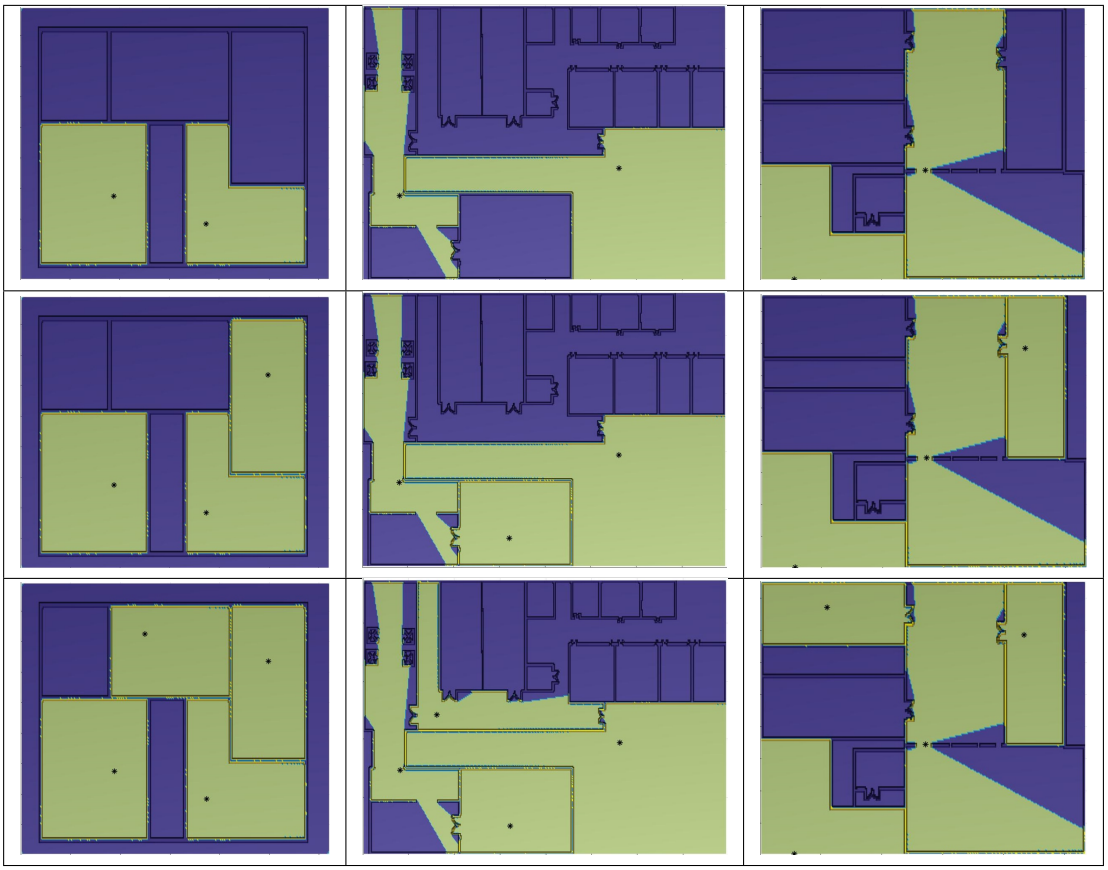


Figura A.2: Resultados de las pruebas asociados a la optimización fmincon dónde cada fila contiene las imágenes relativas a 2, 3 y 4 cámras respectivamente.

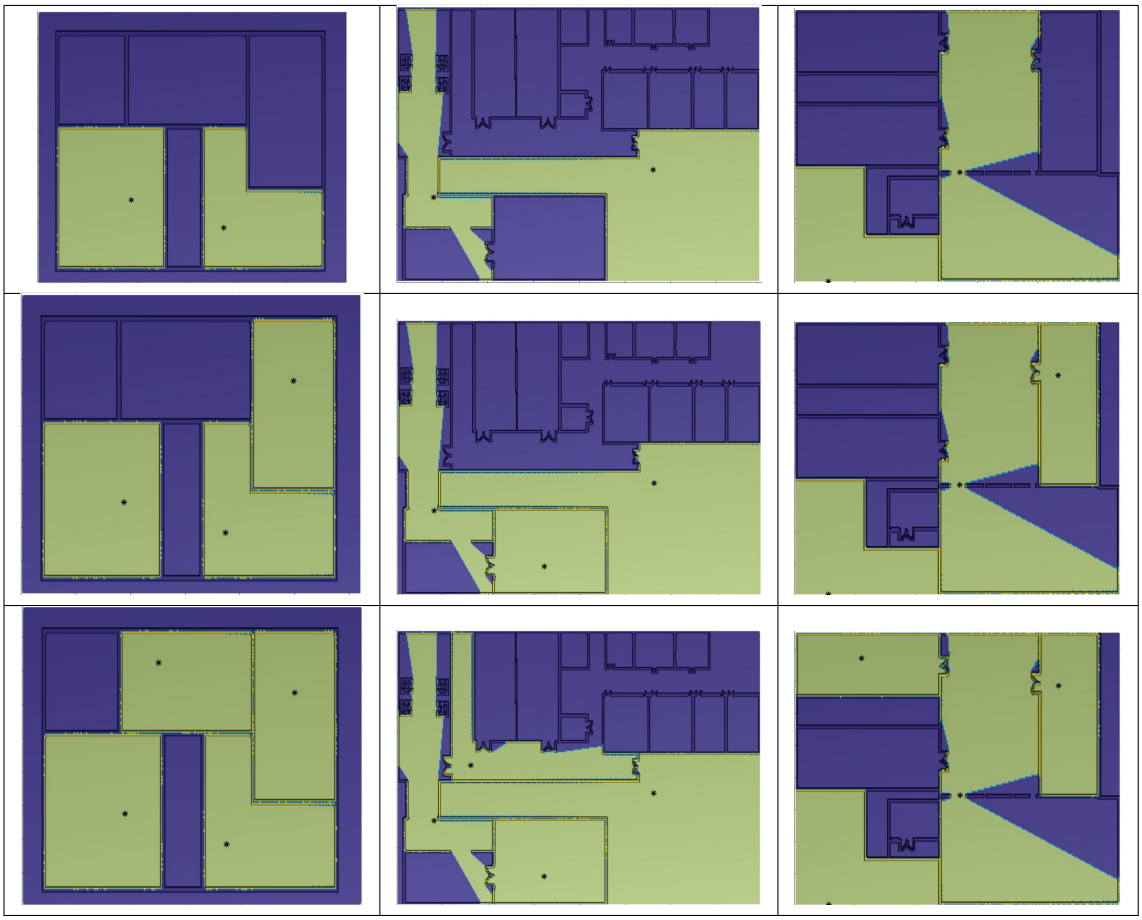


Figura A.3: Resultados de las pruebas asociados a la optimización fminunc dónde cada fila contiene las imágenes relativas a 2, 3 y 4 cámras respectivamente.

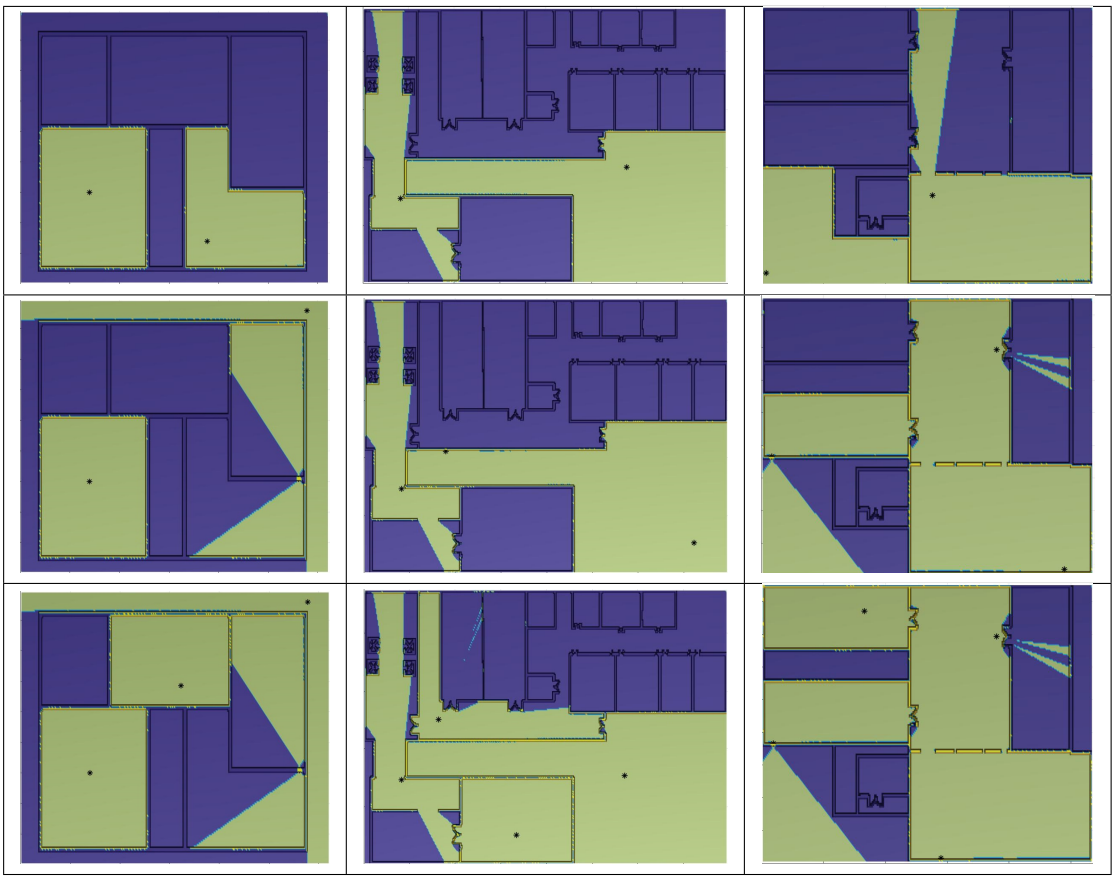


Figura A.4: Resultados de las pruebas asociados a la optimización patternsearch dónde cada fila contiene las imágenes relativas a 2, 3 y 4 cámras respectivamente.

Apéndice B

Resultados de las pruebas para el cálculo de la cobertura de una región crítica

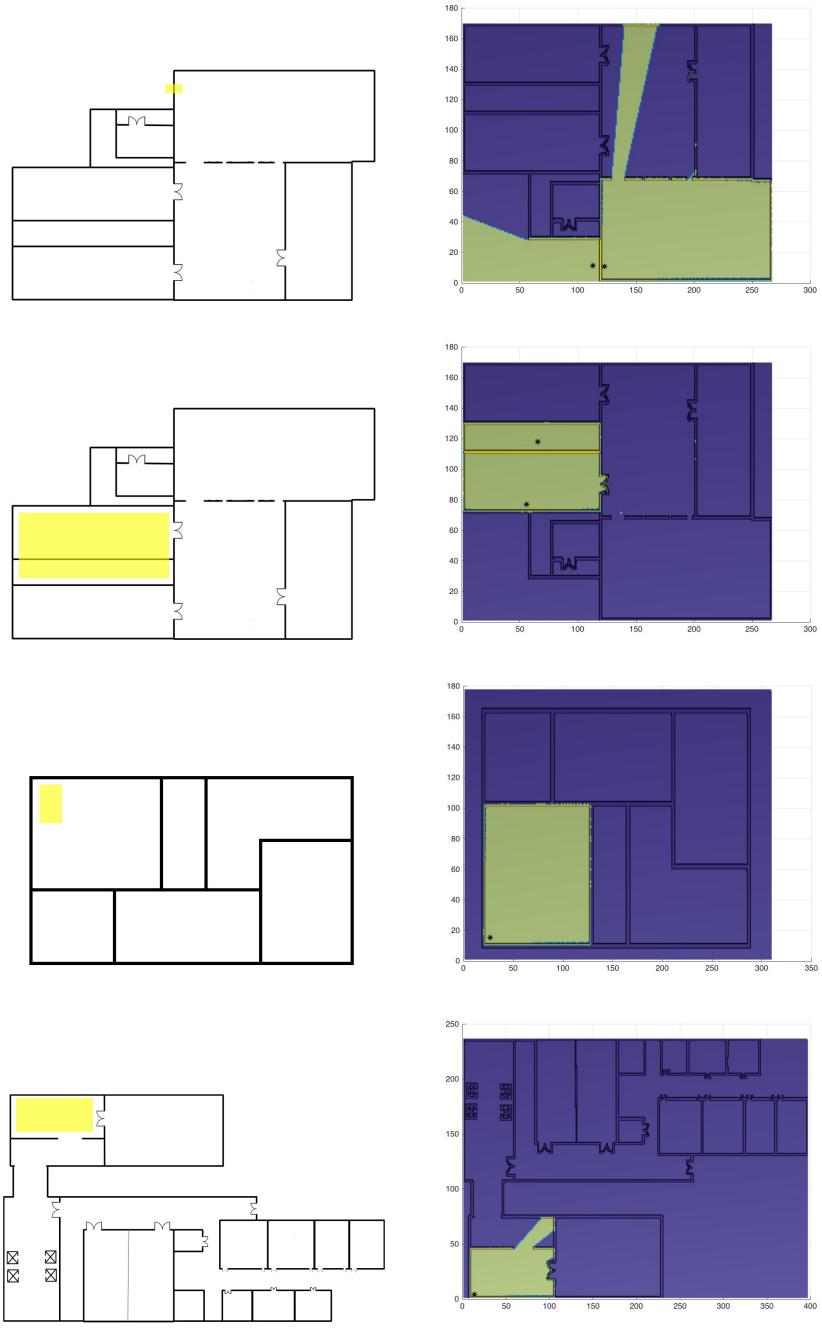


Figura B.1: Resultados de las pruebas asociados al cálculo del número de cámaras y su localización para cubrir una región crítica del mapa.