**UNIVERSIDAD AUTÓNOMA DE MADRID**
ESCUELA POLITÉCNICA SUPERIOR



# DETECCIÓN DE ROBO/ABANDONO DE OBJETOS EN INTERIORES UTILIZANDO CÁMARAS DE PROFUNDIDAD

**-PROYECTO FIN DE CARRERA-**

**Fabricio Alexander Córdova Lucero**
**Diciembre 2012**

# DETECCIÓN DE ROBO/ABANDONO DE OBJETOS EN INTERIORES UTILIZANDO CÁMARAS DE PROFUNDIDAD

**Autor: Fabricio Alexander Córdova Lucero**
**Tutor: Juan Carlos San Miguel Avedillo**
**Ponente: José María Martínez Sanchez**
email: Fabricio.Cordova@estudiante.uam.es {Juancarlos.Sanmiguel, JoseM.Martinez}@uam.es


**Video Processing and Understanding Lab**
**Departamento de Tecnología Electrónica y de las Comunicaciones**
**Escuela Politécnica Superior**
**Universidad Autónoma de Madrid**
**Diciembre 2012**

Trabajo parcialmente financiado por el gobierno español bajo el proyecto TEC2011-25995 (EventVideo)

## Abstract

*The detection of abandoned and stolen objects is a very important application of video surveillance systems. Among the many stages involved, the segmentation of these objects from the background becomes a very fundamental and critical stage to perform the detection. In the same way, another significant step is the objects discrimination stage, which decides the final event (stolen or abandoned). In this work, we provide new procedures for these two stages by making use of a new technology of image capturing: The Kinect Sensor. This device provides us with a new type of information, which indicates the depth of each pixel from a given color image. The combination of depth and color information gives us synchronized color-depth video streams (RGBD) whose data is more accurate and reliable. More specifically, we propose to take advantage of the features of this new information (depth map) such as immunity to changes in illumination or shadows discrimination to increase the robustness of the current systems. The improvements are applied to the already existing prototype in the VPU-Lab group. The improved system is then subjected to several tests, whose results are then compared to those obtained with the previous system.*



## Resumen

La detección de objetos abandonados y robados es una aplicación muy importante de los sistemas de video vigilancia. De entre las múltiples etapas que intervienen, la segmentación de estos objetos respecto al fondo se convierte en una etapa fundamental y crítica para realizar la detección. Asímismo, otra de las etapas significativas, es la etapa de discriminación de objetos, que decide el evento final (robo o abandono). En este proyecto se aportan nuevos procedimientos para dichas etapas, haciendo uso de una nueva tecnología de captura de imágenes proporcionada por el sensor Kinect. Este dispositivo nos ofrece un nuevo tipo de información, que indica la profundidad de cada uno de los píxeles de una imagen de color cualquiera. La combinación de la información de color y profundidad nos proporciona un *stream* de video color-profundidad (RGBD) sincronizado, donde los datos son más precisos y fiables. Más específicamente, se propone aprovechar las características de esta nueva información (mapa de profundidad), tales como la inmunidad a los cambios en la iluminación o la discriminación de sombras para aumentar la robustez de los sistemas actuales en la etapa de segmentación y a su vez, de la detección de objetos robados y/o abandonados. Las mejoras se aplican al prototipo ya existente en el grupo VPU-Lab. El sistema mejorado se somete entonces a varias pruebas, cuyos resultados se comparan con los obtenidos con el sistema de partida, demostrándose las mejoras alcanzadas.

## Palabras clave

Análisis de video, video-vigilancia, robo-abandono, *background* (BG), *foreground* (FG), imagen de profundidad, Kinect Depth Sensor, luz infraroja estructurada, segmentación, sensor de profundidad

# Agradecimientos

*If it wasn't hard, everyone would do it. It's the hard that makes it great.*

Tom Hanks

Este proyecto no habría sido posible sin la imprescindible orientación de mi Tutor, Juan Carlos. Su optimismo, sus ideas y sus estimulantes palabras de apoyo han sido un incentivo inapreciable para afrontar con entusiasmo cada nuevo reto.

Por su puesto, gracias también a mis profesores Jesús Bescós y José María Martínez por haber confiado en mí para la realización de este proyecto y habérmelo asignado.

Agradezco también a mi familia, en especial mi hermano y mi madre. Él siempre aportándome ideas y siempre presente ahí para lo que fuese. Ella, el pilar de la familia, apoyándome en todo, a través del gran esfuerzo de su trabajo y sus alentadoras palabras de ánimo.

Y como no, a mis compañeros, Fernando y Miriam, por haber hecho que no todo sea trabajo duro durante la realización de este proyecto. Por esas actividades de recreación a las asistimos juntos.

Por último, este trabajo se lo quiero dedicar a la memoria de mi padre, que, por desgracia, ya no nos acompaña. A Él más que a nadie le habría gustado ser testigo y partícipe de este logro.

Fabricio Alexander Córdova Lucero
Noviembre 2012

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción

## 1.1. Motivación

La creciente demanda en seguridad [1] exige la creación de sistemas de video vigilancia cada vez más precisos, capaces de detectar, analizar y clasificar determinados comportamientos con márgenes de errores muy reducidos utilizando información visual. Entornos cerrados como estaciones de metro, aeropuertos, museos, etc. [2] necesitan un permanente control sobre los distintos eventos que acontecen en sus alrededores.

Concretamente, el tipo de eventos analizados este proyecto son los de robo y abandono de objetos. Los sistemas destinados a la supervisón de este tipo de eventos pasan básicamente por las siguientes etapas: segmentación de primer plano o sustracción de fondo, detección de regiones estacionarias, clasificación de esas regiones (personas u objetos) y, finalmente, la discriminación entre objetos robados o abandonados para las zonas clasificadas como tal.

En sistemas a gran escala es más eficiente supervisar estos eventos de forma automática que delegar esta tarea a un operador humano (un observador de múltiples cámaras). Esto exige que tales sistemas estén dotados de la suficiente capacidad y autonomía de modo que su precisión y fiabilidad sean aceptables. Estas exigencias hacen necesario la exploración de nuevas técnicas o alternativas para mejorar los sistemas ya existentes. En esta búsqueda se explotan las posibilidades de un nuevo tipo de información facilitada por el sensor Kinect, un dispositivo de reciente presencia en el mercado. Esta información revela la profundidad de cada uno de los elementos presentes en la escena (figura 1.1).

Combinar la información de color con la de profundidad para el análisis de imagen no es nuevo, sin embargo, los medios existentes para conseguirlo hasta hace poco eran bastante


Figura 1.1: Imagen de color (izquierda) e imagen de profundidad (derecha)

costosos. El lanzamiento del sensor Kinect en Noviembre del 2010 supondría una revolución en el uso de este tipo de dispositivos capaces de capturar la profundidad. Su bajo coste y su aceptable precisión en los datos entregados hicieron que pronto éste fuese utilizado en multitud de aplicaciones. En principio, su uso estaba pensado para la industria del entretenimiento (videojuegos), pero no se tardó en ver el potencial del dispositivo, extendiéndose su uso a campos como la robótica [3] o la medicina [4] (figura 1.2).

Figura 1.2: $\pi$ robot (izquierda) y manipulación de imágenes médicas sin contacto (derecha).

Son varias las áreas en las que se ha probado este dispositivo; sin embargo, se ha descuidado su utilidad para fines de seguridad (video-vigilancia) como es el caso que se aborda en este proyecto. Concretamente, el problema a resolver, para un escenario dado (espacios interiores, en nuestro caso) consiste en detectar objetos que son abandonados o robados como se puede apreciar en la figura 1.3.

Figura 1.3: Objeto abandonado (izquierda) y objeto robado (derecha)

En la situación de la figura 1.3, la detección se lleva a cabo por medio de una técnica muy empleada y conocida como sustracción de fondo. En esencia, este método consiste en elegir un modelo de fondo del escenario a supervisar y posteriormente diferenciar los *frames*[1] subsiguientes con dicho modelo. El resultado de esta diferenciación, tras un umbralizado adecuado, es una imagen binaria, cuyas zonas "blancas" (*blobs*) indican los objetos que han cambiado con respecto al fondo de la escena. Ahora bien, la realidad es que no todo funciona perfectamente. Factores intrínsecos como el ruido interno de los dispositivos de captura, o factores extrínsecos, como el cambio los cambios de iluminación, fondo inestable (i.e. hojas de árboles en movimiento), sombras o la aparición de objetos de un color similar al del fondo (camuflaje), hacen que la máscara binaria obtenida del proceso de diferenciación no sea lo suficientemente fiable. Las etapas posteriores en el proceso de detección de robo y abandono de objetos dependen directamente de esta fase, por lo que su precisión conviene que sea lo más alta posible. Es aquí donde toma forma este proyecto, en intentar mejorar en la mayor medida de lo posible la fiabilidad de esta etapa. Esta vez, se dispone de un nuevo tipo de información, que a diferencia del color, es inmune a muchos de los problemas antes mencionados (aunque presenta otros). A modo de ejemplo, en la imagen de la derecha de la figura 1.3, si el objeto "robado" fuese de color blanco, su desaparición sería indetectable; no obstante, si el análisis se basa en el nuevo tipo de información, la profundidad, dicho cambio sería reflejado en un proceso de diferenciación, similar al utilizado con *frames* de color. Esta es sólo una manera de cómo la información de profundidad puede ayudar a solventar algunos de los problemas presentes en este tipo de análisis. A lo largo de esta memoria se detallarán los distintos modos en que se puede aprovechar la información de profundidad para aumentar la robustez de los sistemas que lo utilizan.

[1] A lo largo de este proyecto, se utilizan los términos *frame* e "imagen" de manera indistinta

## 1.2. Objetivos

El objetivo principal de este proyecto es mejorar la capacidad y eficacia de un sistema de detección de objetos robados o abandonados en entornos interiores. Las restricciones del entorno se deben a las limitaciones del dispositivo de caputura de profundidad empleado (el sensor Kinect). Concretamente, el estudio se realizará sobre el prototipo ya existente en el grupo VPU-Lab (Video Processing and Understanding Lab) de la Universidad Autónoma de Madrid (UAM). Los esfuerzos de este trabajo se centran sobre todo en la etapa de segmentación, dada su determinante influencia sobre las etapas posteriores de todo el proceso. Además, es donde mejor se puede aprovechar las características del Sensor Kinect. Dada la versatilidad del dispositivo, algunas etapas distintas a la segmentación, también son modificadas a fin de adaptar alguna de dichas características. Por ejemplo, en la última etapa, a la hora de determinar si un objeto ha aparecido o desaparecido de la escena, para algunos casos, simplemente se tiene en cuenta la distancia de la región marcada como frente y la clasificación es inmediata.

Para conseguir el objetivo propuesto, basaremos nuestro esquema trabajo en la siguiente secuencia de tareas:

1. Estudio del dispositivo de captura de información de profundidad (Sensor Kinect)

   Antes de empezar a utilizar la herramienta clave durante la realización de este proyecto, el sensor Kinect, conviene familiarizarse con sus distintos componentes. Primordialmente, se estudia con detalle el funcionamiento y manejo de los elementos de captura de imagen, tanto de color como de profundidad, así como el del conjunto. Se determina el entorno de desarrollo más adecuado para controlar el sensor, de acuerdo a las prestaciones que ofrecen y a como se adapten a nuestras necesidades. Asimismo, se busca la integración de dicho entorno con librerías de análisis de imagen (OpenCV).

2. Análisis del estado del arte relacionado.

   Se describen las etapas del proceso de detección de robo/abandono para sistemas basados en información de color. Asímismo, se estudian las distintas técnicas para la obtención de la información de profundidad, centrando la atención en aquella empleada por el sensor Kinect. De éste se describen sus principales características así como sus limitaciones, comparándolas con otros dispositivos destinados al mismo fin.

3. Estudio del prototipo VPU-lab para detección de robo/abandono.

   Se analizan en detalle cada uno de los módulos que comprende el sistema del VPU-lab para detección de robo/abandono. Se identifican sus limitaciones originadas por

el uso de imágenes de color. La finalidad es localizar los módulos donde puede ser beneficioso el uso de profundidad.

4. Propuesta e implementación de mejoras del prototipo VPU-Lab con imágenes de profundidad.

   Se realiza una segmentación de primer plano, haciendo uso únicamente de imágenes de profundidad. Se evalúa su efectividad comparándola con la obtenida por segmentación con imágenes de color. Asimismo, se evalúa la eficacia obtenida de la combinación de los resultados de segmentación obtenidos por separado. Las imágenes de profundidad son también utilizadas en la última etapa (etapa de discriminación) para determinar si un objeto ha sido robado o abandonado, de acuerdo a ciertas condiciones de la escena.

5. Diseño de un set de pruebas y puesta en marcha de éste.

   Para poner a prueba la validez del sistema mejorado, se realizan pruebas con imágenes obtenidas por el propio sensor (tanto las de color como las de profundidad). Las pruebas se realizan en tiempo real y sobre diversos escenarios, siempre interiores, variando las condiciones del entorno y las interacciones de las personas presentes. Así, es posible obtener una clasificación del set de pruebas en función de la complejidad de la escena.

6. Evaluación de resultados.

   Los resultados obtenidos con el sistema mejorado son comparados con los obtenidos por detector basado únicamente en información de color. Los resultados se evalúan de tres formas distintas. Primero, se tiene solamente en cuenta los resultados de la segmentación basada en profundidad; en segundo lugar, los resultados obtenidos por segmentación de color (las generadas por el prototipo del VPU-Lab) y, por último, considerando la combinación de los resultados de ambas segmentaciones. El objetivo es comprobar cuanto mejora (o empeora) la detección en las tres situaciones mencionadas.

## 1.3. Organización de la memoria

La estructura de esta memoria es la siguiente:

- Capítulo 1. En este capítulo se presenta la motivación, los objetivos y la estructura de este documento.
- Capítulo 2. Este capítulo da una visión detallada de la literatura relacionada con el trabajo presentado en este documento.

- Capítulo 3. En este capítulo se analiza el sistema de video vigilancia existente en el grupo VPU-Lab para la detección de objetos robados o abandonados.
- Capítulo 4. Aquí se describen las mejoras propuestas para incluir la información de profundidad y como adaptarlos a los distintos módulos del sistema de detección del VPU-Lab.
- Capítulo 5. Aquí se presentan los resultados experimentales del sistema mejorado.
- Capítulo 6. Este capítulo resume los principales logros del trabajo, se analizan los resultados obtenidos y se proporciona sugerencias para el trabajo futuro.
- Apéndice
    - A Entornos de desarrollo: OpenNI y MSKD
    - B Integración de OpenCV con SDKs

# Capítulo 2

# Estado del arte

## 2.1. Introducción

En este capítulo se ofrece una descripción breve del trabajo previo en el campo de la detección de objetos robados/abandonados, así como una visión más amplia en la adquisición y las principales aplicaciones de la información de profundidad. En las secciones siguientes se describen, en primer lugar, los sistemas actuales de detección (sección 2.2), posteriormente, las diferentes tecnologías que existen para la obtención de la profundidad (sección 2.3), seguidamente, se detallan las características de una de estas tecnologías, concretamente el sensor Kinect, dispositivo utilizado para la realización de este proyecto (sección 2.4) y, en la última sección, se citan las aplicaciones más relevantes de la información de profundidad (sección2.5).

## 2.2. Sistemas de robo-abandono

Los sistemas actuales de detección de eventos del robo o abandono de objetos se apoyan fundamentalmente en el análisis de imágenes de color procedentes de una secuencia de video. El tratamiento y procesamiento de dichas imágenes se realiza a través de una serie de etapas claramente identificadas como se ilustra en la figura 2.1.

En la etapa de **detección del primer plano**[1], lo que se intenta es aislar partes de la escena de una secuencia de imágenes que no aparecían en otra que se toma como referencia. Esta imagen es conocida como el modelo de fondo y es su diversidad en la forma en la que puede ser adquirida lo que hace variar la eficiencia de esta etapa. Dadas las limitaciones del ruido y las condiciones de la escena, es bastante complicado conseguir un modelo estable durante varios *frames* consecutivos. Inconvenientes como el ruido interno de la cámara, los

[1]Los términos "primer plano", "frente" y *foreground* se utilizan de manera indistinta para referirse a la máscara binaria con los nuevos elementos aparecidos en la escena

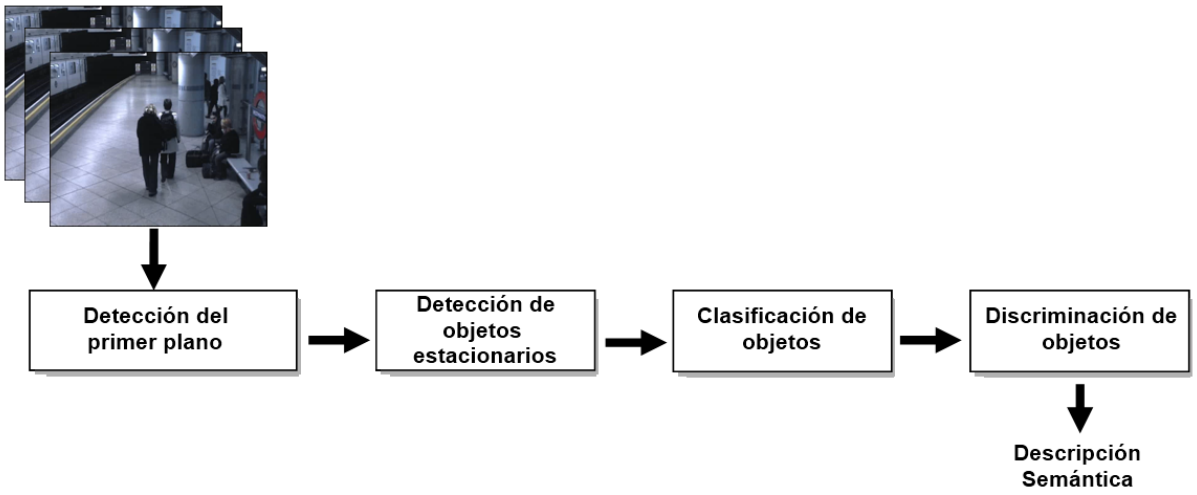



Figura 2.1: Etapas en detección de robo-abandono de objetos

cambios de iluminación, las oclusiones, el camuflaje, las sombras y reflejos o movimientos periódicos de ciertas áreas de la escena (p. ej. una pantalla encendida) hacen que no sea suficiente con elegir un sólo *frame* como modelo de fondo, sino más bien un conjunto de ellas. La figura 2.2 ilustra dos escenarios donde se pueden apreciar todos estos problemas.

Las operaciones aplicadas sobre el conjunto de imágenes para modelar del fondo pue-

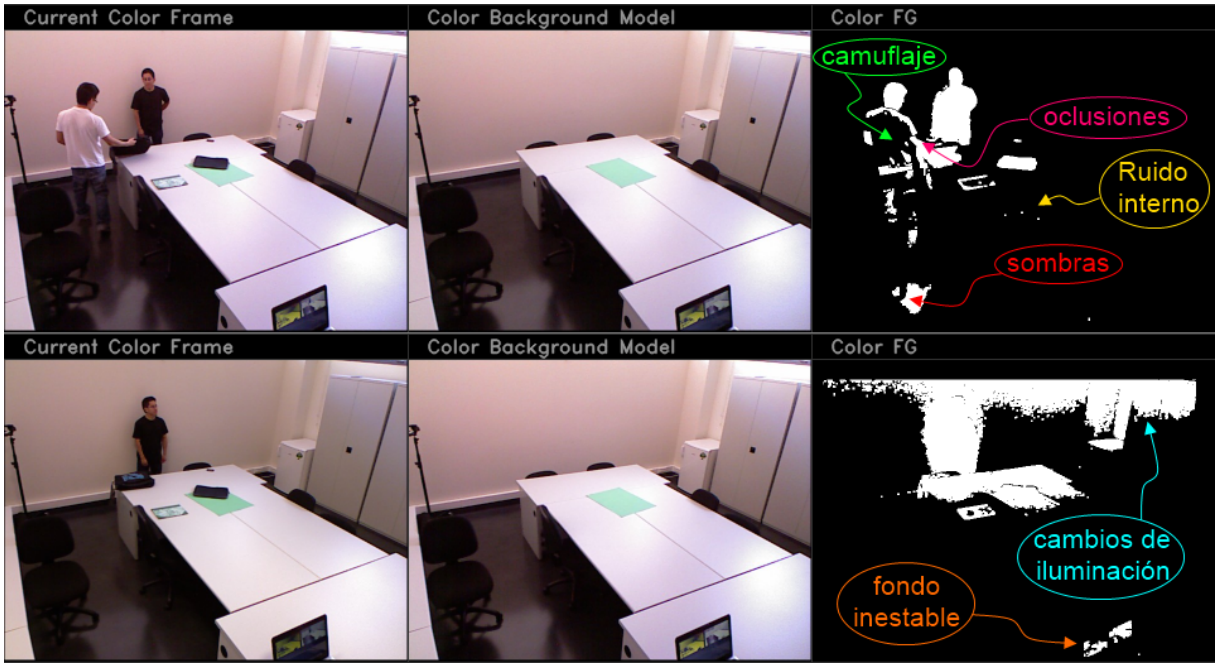



Figura 2.2: Inconvenientes en imágenes de color

den ir desde lo más sencillo, como puede ser una media aritmética, hasta aquellas en la que cada pixel se modela de acuerdo a su comportamiento en el tiempo. En esta línea se encuentran los modelos basados en Guassianas [5], que, básicamente, consiste en caracterizar cada pixel del modelo con valores que siguen una distribución gaussiana. También está el modelo basado en mezcla de Gaussianas [6] y el segmentador Gamma [7]. En cualquier caso, cualquiera que sea el método elegido para modelar el fondo, éste debe ser lo suficientemente robusto como para absorber todas variaciones indeseadas de la escena. Asimismo, debe ser capaz de incorporar (o quitar) de la escena elementos que pasan a formar parte del mismo.

El resultado de la etapa anterior es una máscara binaria que marca las zonas (*blobs*) del *frame* actual donde han aparecido (o desaparecido) ciertos componentes. Interesa no perder el rastro de aquellos *blobs* inmóviles, pues son los únicos susceptibles de pertenecer a un objeto robado o abandonado. De esta tarea se encarga la etapa de **detección de objetos estacionarios**. De entre las múltiples técnicas para llevar a cabo esta tarea, se tienen las aproximaciones clásicas (*tracking*), consistentes en realizar un seguimiento de todos los *blobs* y observar la velocidad de su movimiento[8][9]. En esta misma línea están las técnicas basadas en la acumulación de *foregrounds*, donde se almacenan una serie de máscaras con el frente de la escena y posteriormente se realiza algún tipo de operación sobre ellas (p. ej. una operación AND binaria) [10][11]. Se tienen también los métodos basados en el modelo de fondo empleado, donde se aprovechan las características de ciertos parámetros del modelo de fondo utilizado. Por ejemplo, en un modelo basado en mezcla de gaussianas (MoG), observar el peso de cada gaussiana y su media, puede ser un indicador fiable para determinar la estacionariedad de una región [12][13]. Los métodos basados en máscaras de *foreground* muestreados son también de uso muy común. Aquí, las secuencias de video se analizan a diferentes velocidades, de modo que una región estacionaria permanece inalterada, independientemente de cuál sea la tasa de muestreo [14]. Otra de las técnicas se basan en el uso de varios modelos de fondo, donde la estacionariedad de una región se determina aprovechando las características que presentan (en esa región) los modelos de fondo, que son muestreados a diferentes tasas binarias cada uno [15].

La siguiente etapa es la de **clasificación de objetos**. En ella el objetivo es identificar de entre los *blobs* estáticos obtenidos en la etapa anterior aquellos que pertenecen a objetos y los que pertenecen a personas. La eficacia de esta clasificación depende de las técnicas utilizadas, que, a su vez, dependen de las características de la escena bajo estudio. Se puede extraer información de la propia geometría del *blob* como esquinas, líneas y círculos. Características como el área, la densidad (relación entre el área del *bounding box* y el área de la región estática) o la razón de aspecto de los *bounding boxes* contribuyen también a esta clasificación. Existen técnicas más complejas que localizan partes del cuerpo como

la piel o la cara [16] o aquellas que utilizan datos previamente entrenados de rasgos del cuerpo como siluetas, articulaciones o modelos volumétricos [17] para la detecciones de personas. Otra de las técnicas más ampliamente utilizada dada su simplicidad y bajo coste computacional es la descrita en [18], cuyo funcionamiento, en esencia, se basa en rodear una región estática con elipses de forma iterativa. La relación de aspecto y las iteraciones necesarias para encontrar las elipses es lo que determina si una silueta pertenece a un objeto o a una persona.

Finalmente, unas vez que se han detectado todos los objetos estáticos de la escena, en la última fase, **discriminación de objetos**, se decide si los mismos han sido robados o abandonados. Las técnicas aquí empleadas se pueden categorizar en tres grupos, en función del tipo de información que usan: aquellas que hacen uso de los contornos del objeto, las que hacen uso de la información de color y aquellas en las que se combinan ambos tipos de información. En las aproximaciones basadas en contornos se calcula la energía en los bordes de la región estática. Esta energía se supone que es alta cuando el objeto ha sido introducido en la escena y baja cuando ha sido retirado [19]. Los métodos que utilizan el color analizan esta información (el color) en las regiones estáticas, tanto en el *frame* actual como en el modelo de fondo. Si un objeto es removido de la escena, se espera que la porción descubierta tenga propiedades de color similares a la de sus bordes y viceversa, si el objeto es abandonado, dichas propiedades serán diferentes [2]. Una mejor aproximación se consigue combinando ambos tipos de información a la vez (tercer grupo). En [20] se crean modelos probabilísticos de cada método para cada clase (robado o abandonado), que luego serán comparados con los obtenidos de cada *frame* en una región concreta. Otras técnicas basadas en contornos activos y contraste del contorno se proponen en [21] y [22], respectivamente .

## 2.3. Información de profundidad

Existe una amplia variedad de técnicas para capturar la profundidad de una determinada escena. El esquema de la figura 2.3 resume las técnicas más conocidas y empleadas. De las tres ramas del esquema (microondas, ondas de luz y ondas ultrasónicas), el estudio de aquellos métodos basados en ondas de luz son las de mayor interés, pues es donde se enmarca el principio de funcionamiento del dispositivo utilizado para este proyecto, el sensor Kinect. Como se puede ver también en el esquema, son cuatro las técnicas que basan su actividad en las ondas de luz. Concretamente, estas son: triangulación (visión estéreo), ToF *(Time of flight)*, luz estructurada y escaneo laser.

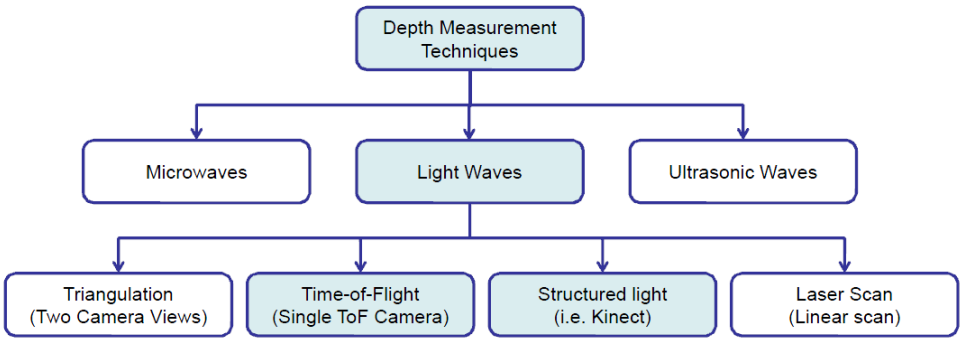



Figura 2.3: Métodos de captura de profundidad

De acuerdo a estudios previos [23, 24, 25, 26], los cuatro métodos pueden ser clasificados como activos (*Time of flight*, luz estructurada y escaneo laser) o pasivos (Triangulación). Los primeros, proyectan intencionadamente un haz de rayos luminosos sobre la escena para hacer que las características de la misma sean más identificables; mientras que los métodos pasivos simplemente intentan buscar correspondencias entre un par de imágenes, de las cuales no se conoce nada *a priori*. En las subsecciones siguientes se describe en detalle la manera de operar de cada técnica.

### 2.3.1. Triangulación (visión estéreo)

En este método la profundidad se obtiene considerando dos puntos de vista (de las cámaras) desde dos posiciones conocidas [27, 28]. Del par de imágenes obtenidas, se intenta buscar correspondencias entre píxeles. Esto se puede hacer, o bien buscando características específicas de la escena como esquinas; o, lo que es más común, eligiendo una ventana espacial arbitraria en una de las imágenes y buscando su correspondencia a través de la línea epipolar [29] en la segunda imagen. Más específicamente, este procedimiento minimiza una función de emparejamiento, que puede ser descrita mediante la siguiente expresión:

$$||I_1(V_s(x_1)) - I_2(V_s(x_2))||^2$$

Donde $I_1$ es la intensidad en la imagen 1, $I_2$, la intensidad en la imagen 2, y $V_s$ es un vector de píxeles de una vecindad próxima a $x_1$ (ó $x_2$). Asumiendo $x_1$ fijo, el valor de $x_2$ que haga mínima la expresión anterior, será el valor que mejor se empareja con $x_1$. Así, una vez identificados los píxeles $x_1$ y $x_2$ correspondientes a un mismo punto 3D, y conociendo la ubicación de las cámaras, es inmediato estimar la profundidad para ese pixel a través de la siguiente fórmula $z = b\frac{f}{d}$, donde $z$ es la profundidad, $b$ es la separación entre las cámaras, $f$ es la distancia focal de las cámaras y $d$ es la dispardidad. El esquema de la figura 2.4 muestra la procedencia de cada uno de los parámetros de dicha fórmula.

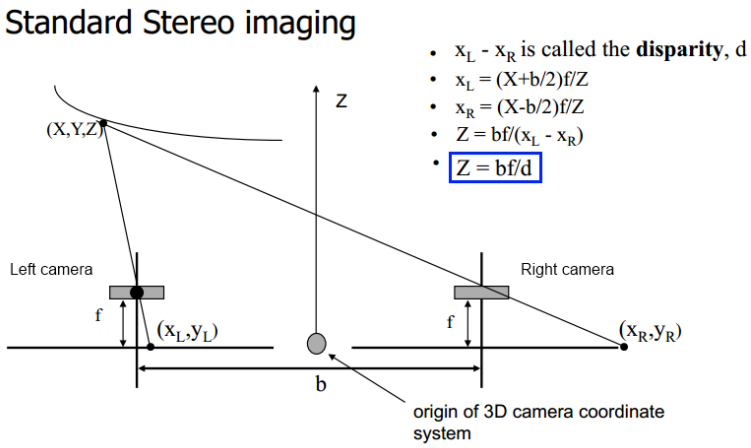



Figura 2.4: Obtención estandar de la profundidad estéreo

Existe un compromiso a la hora de elegir el tamaño de la ventana. Por ejemplo, si se elige muy pequeña (caso extremo, un pixel), hay mayor probabilidad de que hayan píxeles de valores similares en la otra imagen. Si el tamaño es mayor, normalmente se obtiene más desambiguación en la información. Por otro lado, no siempre esto es así, ya que regiones de color constante no aportan nueva información. Esto provocará que hayan distorsiones y discontinuidades en la imagen de profundidad.

### 2.3.2. Escaneo laser (*Laser scanning*)

Esta técnica consiste en proyectar un haz de luz (láser) sobre los objetos, cuya profundidad se desea estimar [30][31]. La figura 2.5 ilustra el principio de funcionamiento de este método.

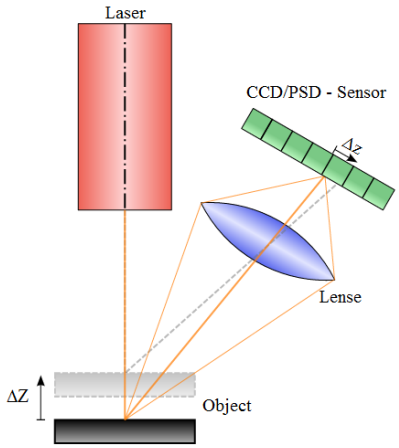
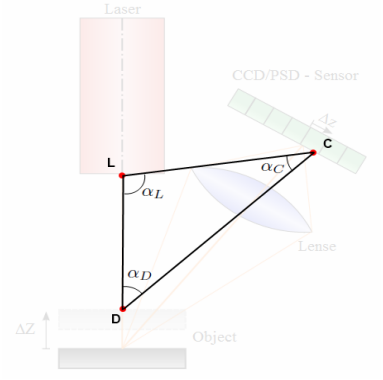



Figura 2.5: Principio de funcionamiento del escaneo laser

El rayo laser (móvil) se proyecta sobre una superficie y, dependiendo de la distancia a la misma, el punto de luz aparece en diferentes lugares en el campo de visión de la cámara, la cual es rotada intermitentemente de modo que la lente siempre enfoque al punto proyectado por el laser. El proyector ($L$), el punto proyectado ($D$) y la cámara ($C$) forman los vértices de un triángulo, del cual se conoce uno de los lados ($\overline{LC}$). El ángulo del vértice $L$ también es conocido. El ángulo del vértice $C$ se puede determinar buscando el punto del laser en el campo de visión de la cámara. Con estos tres datos se puede obtener de manera inmediata la profundidad de la superficie bajo análisis. Para ello, basta con aplicar el *Teorema de los senos* ilustrado en la figura 2.6.




$$\frac{\overline{LD}}{\sin\alpha_C} = \frac{\overline{DC}}{\sin\alpha_L} = \frac{\overline{CL}}{\sin\alpha_D}$$

Figura 2.6: Teorema de los senos

En este caso $\overline{LD}$ es valor que se corresponde con la profundidad de la superficie. Utilizando el primer y último término de la igualdad se puede despejar fácilmente dicho valor.

### 2.3.3. ToF (*time of flight*)

La capacidad para medir distancias se basa en medir el tiempo de viaje de la luz. En particular, esta tecnología mide el tiempo transcurrido desde que el haz de luz sale del proyector hasta que se refleja en el sensor (figura 2.7).

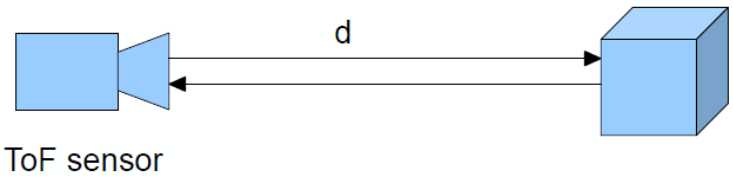



Figura 2.7: Principio de funcionamiento de ToF

Si se considera $t$ como el tiempo que tarda el haz de luz en alcanzar al objeto, $d$ la distancia al objeto y $c$ la velocidad de la luz ($3{,}10^8[m/s]$), entonces es inmediato obtener el

valor de profundidad ($d$ ) como:

$$d = \frac{c.t}{2}$$

El dato físico real medido en el sensor es el desplazamiento de fase $\triangle\varphi$ entre la señal emitida y la recibida. Este retardo de fase permite calcular el tiempo de viaje $t$ de la onda de luz. La señal producida por el proyector es una onda sinusoidal modulada en amplitud (AM) para permitir su transmisión por el aire. Dicha señal $s(t)$ se expresa como:

$$s(t) = A\cos(2\pi f_c t)\cos(2\pi f_s t)$$

Donde $f_c$ es la frecuencia de modulación y $f_s$ es la frecuencia creada por el sensor. El receptor por su lado, puede recibir la señal de acuerdo a la siguiente expresión:

$$r(t) = R.s(t-\tau) + B + w(t)$$

Donde $R$ es la amplitud de la señal recibida, $\tau$ es el retardo de la transmisión, $B$ es la iluminación ambiente y $w(t)$ representa el error en la transmisión.

Los dispositivos ToF poseen limitaciones que tienen que ver con el propio principio de funcionamiento, las condiciones del entorno y las propiedades de los objetos a detectar [32]. En la práctica, la tecnología ToF calcula el desfase entre onda emitida y onda recibida en función de la carga almacenada en cuatro capacitores (Q1 a Q4) durante un tiempo denominado "tiempo de integración" [33]. La carga solo se realiza en los flancos de subida de la onda reflejado o recibida y cuando las señales C1 a C4 (desfasadas 90º una de la otra) están activas, como se observa en la figura 2.8.

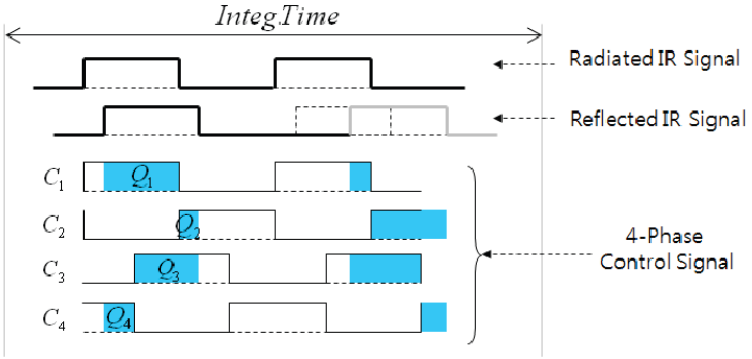



Figura 2.8: Cantidad de carga eléctrica (Q1 a Q4) para la señal de control (C1 a C4, respectivamente)

A partir de las cargas $Q_{1-4}$ se obtiene la profundidad como: $Depth(t_d) = \arctan(\frac{nQ_3 - nQ_4}{n1Q - nQ_2})$, donde Q1 a Q4 representan la cantidad de carga eléctrica para las señales de control C1 a

C4, respectivamente y $n$ es el número de ciclos utilizados para realizar el cálculo. La duración del tiempo de integración (dependiente de $n$) determina la resolución y precisión del mapa de profundidad. En la imagen a) de la figura 2.9 se puede ver que el mapa de profundidad de la derecha tiene una mejor relación señal ruido que el mapa de la izquierda, ya que su tiempo de integración es mayor.

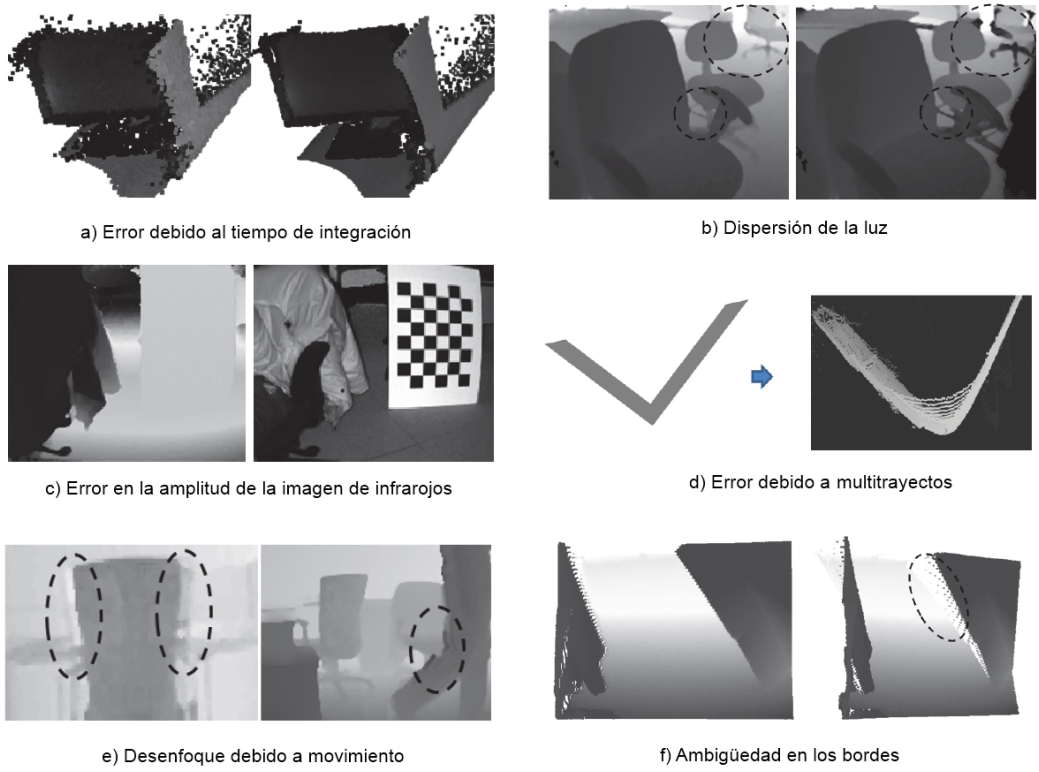



Figura 2.9: Errores comunes en cámaras ToF

En esta misma figura (figura 2.9) se pueden ver reflejadas una gran variedad de errores de distinta naturaleza. Los objetos muy cercanos al sensor IR que pueden provocar una saturación del mismo, lo genera distorsiones en otras partes como se puede ver en la imagen b). Este fenómeno de dispersión de la luz hacia otras partes de la escena se conoce como *scattering* [34]. Objetos a la misma distancia del sensor muestran distinta amplitud en la imagen de infrarrojos (imagen c)). Esto también es un factor de ruido a la hora de estimar la profundidad [35]. Los objetos cóncavos pueden dar lugar también a estimaciones erróneas de profundidad, concretamente, a errores de multitrayecto. En la imagen d) se puede apreciar este efecto en la concavidad de la esquina. Los objetos en movimiento generan un error conocido como *motion blur* o desenfoque por movimiento [36]. Su origen también reside en el modo de operar de los sensores ToF. Concretamente, si durante el tiempo de

integración, el objeto sobre el que está rebotando la onda no permanece estático, entonces la estimación de profundidad se hace sobre distancias distintas, lo que produce el efecto observado en la imagen e). Los bordes de los objetos también son fuente de distorsiones. En ellos, el valor de los píxeles se corresponde en algunos casos con la profundidad del objeto y otras con la del fondo, como se puede apreciar en la imagen f).

### 2.3.4. Luz estructurada

Es el principio que usa el sensor Kinect (sección 2.4) para su funcionamiento. Este método consiste proyectar un patrón de luz sobre la escena y observar la deformación del patrón en la superficie de los objetos [37][38][39]. En la figura 2.10 se muestra a la izquierda un patrón de líneas y a la derecha un patrón de puntos utilizado por el sensor Kinect, cuyo funcionamiento se detalla en la sección 2.4.

El patrón de luz es proyectado, bien por un proyector LCD (Luz no coherente) o bien por un barrido laser. Una cámara desplazada ligeramente respecto del proyector, captura la deformación de las líneas (o puntos) y calcula la distancia de cada punto utilizando una técnica similar al de la triangulación [40].

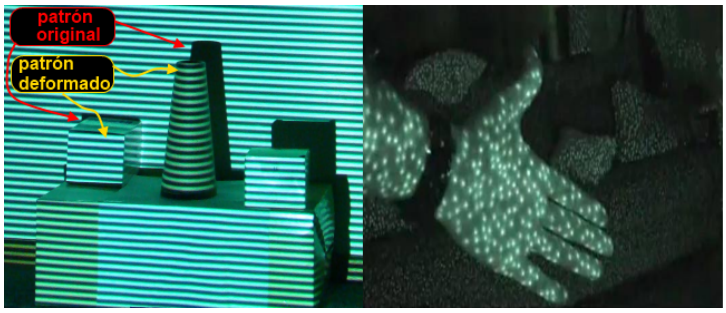



Figura 2.10: Patrones de luz estructurada con líneas y puntos (sensor Kinect)

En la figura 2.11 se puede observar cómo se emplea la técnica de triangulación para el caso de patrones de líneas. En una superficie plana, es de esperar que la línea capturada por la cámara sea recta. Una pequeña deformación en la misma puede ser directamente convertida a una coordenada 3D [41][42]. Para ello se tiene que identificar cada línea, lo que se logra mediante el rastreo de cada línea (método de reconocimiento de patrones) o simplemente contándolas. Otro método muy común consiste en proyectar patrones alternativos formando una secuencia de *código Gray*[2] que identifica el número de cada línea proyectada sobre el objeto.

[2]The Gray Code by R. W. Doran: `http://www.cs.auckland.ac.nz/CDMTCS//researchreports/304bob.pdf`

Otro indicio de profundidad se puede obtener analizando el grosor del las líneas (o puntos), que varía en función de la inclinación de la superficie del objeto. Asímismo, la transformada *Wavelet* también está siendo discutida para el mismo propósito [43]. En la práctica se combinan varios métodos para obtener información lo menos ambigua posible .

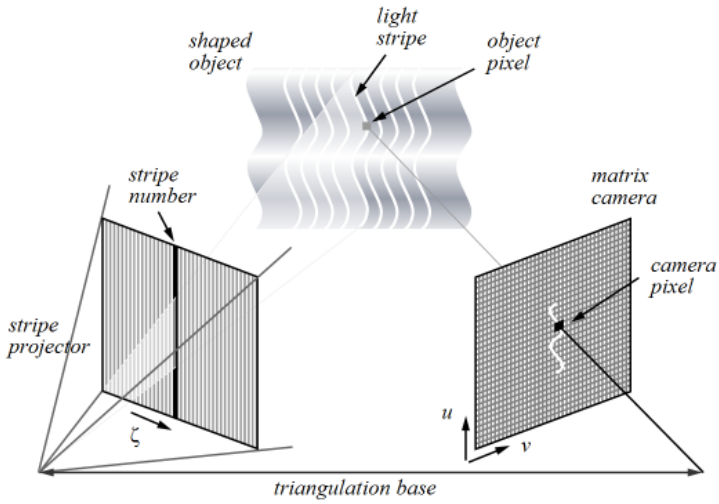



Figura 2.11: Principio de triangulación aplicada a patrones de líneas

## 2.4. El sensor Kinect

El sensor Kinect es un dispositivo lanzado en Noviembre del 2010 por *Microsoft*, orientada principalmente a la industria de los videojuegos, concretamente, como periférico de la video-consola *Xbox 360* de *Microsoft*. Su principal innovación es que permite a los usuarios controlar e interactuar con la consola sin necesidad de tocar ningún controlador de juego físicamente, a través de una interfaz de usuario natural basado en gestos y comandos de voz. La apariencia de este dispositivo es la que se muestra en la figura 2.12.

### 2.4.1. Características del sensor Kinect

Como se puede observar en la figura 2.12, son cuatro los elementos principales que componen el dispositivo [44][45]: una cámara RGB en resoluciones de 640x480 (VGA) y 1280x1024 píxeles, sensor de profundidad (*IR projector* + *IR camera*) en resoluciones de 640x480 (VGA), 320x240 (QVGA) y 80x60 píxeles; un motor para controlar la inclinación del dispositivo y un *array* de cuatro micrófonos distribuidos a lo largo del sensor. A continuación se describen los que resultan de interés para este proyecto.

- Cámara RGB

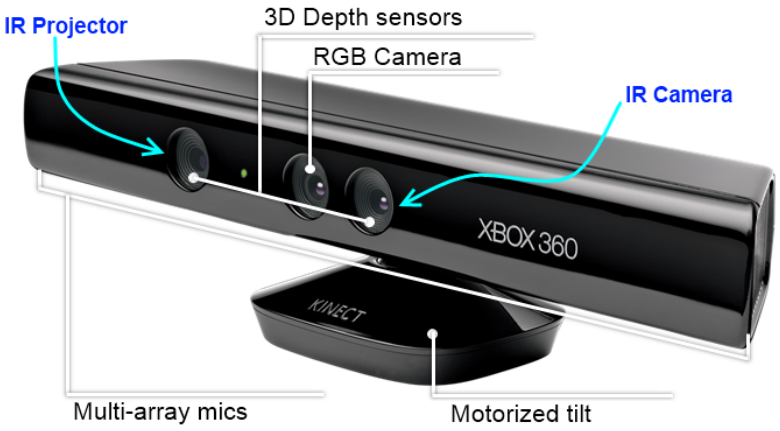



Figura 2.12: El sensor Kinect de Microsoft

Dependiendo del SDK[3] utilizado, esta cámara es capaz de operar con dos formatos de imagen: formato RGB y formato YUV. En ambos formatos, hasta treinta imágenes pueden ser generadas por segundo (30 fps). Las imágenes en formato YUV sólo están disponibles en una resolución de 640x480 píxeles y a solo 15 fps .

- Sensor de profundidad

  Este sensor utiliza luz estructurada infrarroja (sección 2.3.4) para su funcionamiento. La fuente de luz infrarroja (laser más rejilla de difracción), proyecta un patrón de puntos sobre la escena que es leído por un sensor de infrarrojos monocromático CMOS. El sensor detecta los segmentos de puntos reflejados y estima la profundidad a partir de la intensidad y la distorsión de los mismos. Microsoft, la empresa propietaria del sensor, no ha hecho ninguna publicación en relación al algoritmo empleado para estimar la profundidad. Sin embargo, varios investigadores han intentado deducirlo por medio de un proceso de ingeniería inversa. El más acertado, de acuerdo con la salida del dispositivo, es la explicación que proporciona ROS[4][46]. La figura 2.13 muestra un patrón de puntos claros y oscuros proyectados por el sensor Kinect. Según ROS, el algoritmo comienza calculando la profundidad de un plano de referencia a partir de los nueve puntos que aparecen muy marcados y guarda el patrón para ese plano. Posteriormente, la profundidad para cada pixel se calcula eligiendo una ventana de correlación pequeña (9x9 ó 9x7) y se compara el patrón local en ese píxel con el patrón memorizado en ese píxel y los 64 píxeles vecinos en una ventana horizontal.

[3]SDK son las iniciales de Software Development Kit. Es un conjunto de herramientas de desarrollo de *software* que permite la creación de aplicaciones para un cierto paquete de *software*, plataforma *software*, plataforma *hardware* o cualquier sistema informático en general.

[4]ROS son las siglas de *Robot Operating System*. Es un *framework* para desarrollo de *software* para robots a través clúster informático heterogéneo con las funcionalidades de un sistema operativo común. (www.ros.org)

La mejor coincidencia da un *offset* (disparidad) respecto a la profundidad del plano de referencia, en términos de píxeles. Dadas la profundidad del plano memorizado y la disparidad, una profundidad estimada para cada píxel puede ser calculada por triangulación, de acuerdo a la expresión (utilizada por un sistema estéreo normal, sección 2.3.1) $z = b\frac{f}{d}$, donde $z$ es la profundidad (en metros), $b$ es la separación entre el proyector y la cámara de infrarojos (en metros), $f$ es la distancia focal de la cámara (en píxeles) y $d$ es la dispardidad (en píxeles).

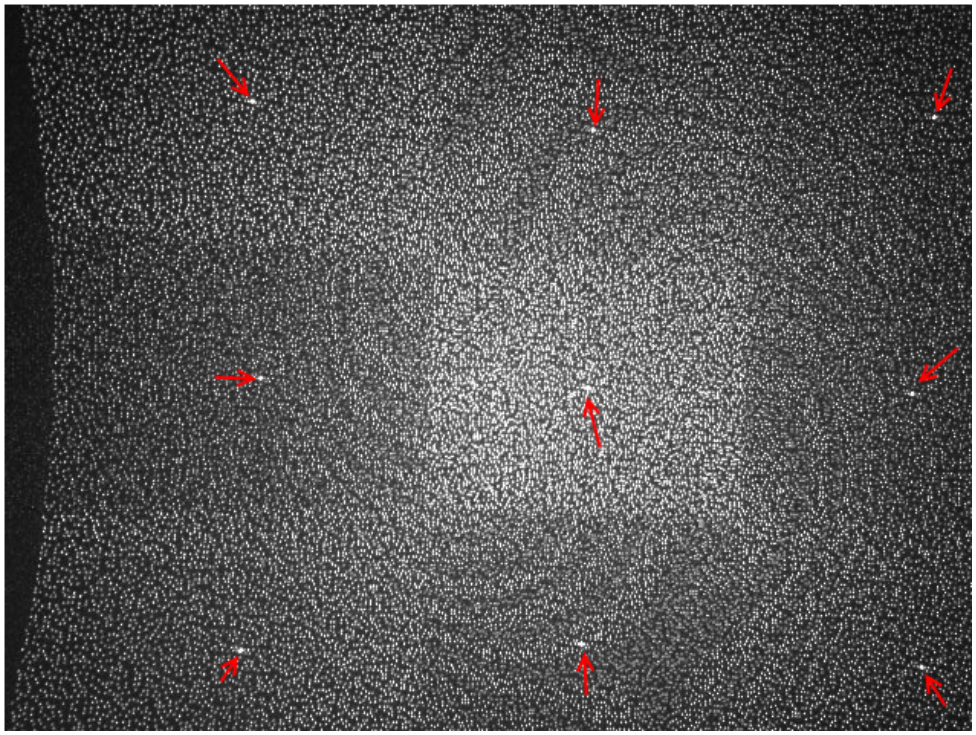


Figura 2.13: Patrón de puntos emitidos por el sensor Kinect

Por otro lado, la resolución en la dimensión $z$ es de aproximadamente un centímetro, mientras que la resolución espacial (ejes $x$ e $y$) es del orden de milímetros. La información de profundidad se devuelve en un mapa de píxeles con una frecuencia máxima de 30 imágenes por segundo. Cada pixel está representado por dos bytes (16 bits), cuyo valor representa teóricamente la distancia del objeto al sensor. Para un funcionamiento correcto, deben satisfacerse las siguientes condiciones teóricas:

- Máxima (mínima) distancia del objeto al sensor: 3.5 (1.2) metros.
- Los objetos deben estar dentro del ángulo de visión: $\pm 43^{\text{º}}$ verticalmente y de $\pm 57^{\text{º}}$ horizontalmente.
- Relacionadas con el tipo de objeto (translucido, especular, cóncavo, etc.).

- Relacionadas con la iluminación (luz solar fuerte).

Si el valor de un pixel del mapa es cero, significa que el sensor no puedo estimar la profundidad para esa región por no cumplirse alguna de estas condiciones.

### 2.4.2. Limitaciones

El sensor Kinect tiene varias limitaciones que hacen que la profundidad de ciertas regiones de la escena no se pueda estimar o si se estima, la fiabilidad de los datos no es aceptable. Estas limitaciones vienen condicionadas tanto por factores internos, debidos a la arquitectura del dispositivo; como externos, debidos a la naturaleza de la escena. En el primer grupo (factores internos) se tienen las siguientes limitaciones.

- Los puntos de luz no cubren de forma continua la superficie de los objetos (imagen de la derecha de la figura 2.10), lo que conlleva a que algunos pixeles de la imagen de profundidad tienen que ser interpolados. Esto implica que el valor de profundidad de un pixel determinado tiene asociado un margen de error. Este margen es mayor cuanto más alejado está el objeto, puesto que, para una misma superficie, los puntos de luz están más separados. Según [47][48], a mayores distancias, los valores de profundidad devueltos para objetos cercanos entre sí tienden a ser muy similares. Sin embargo, si el objeto está a demasiada distancia del sensor, no se calcula ninguna distancia para ese punto (figura 2.14). Esto ocurre así, porque la potencia de luz del haz de infrarrojos se atenúa en el trayecto recorrido, haciendo que sea imperceptible para el sensor de infrarrojos.

Figura 2.14: Imagen de profundidad para distancias muy grandes

En la figura 2.15 se puede observar como varía el nivel de incertidumbre en función de la distancia a la que se encuentre el objeto [48]. Así, por ejemplo, para una distancia

de 2 metros, el margen de error es de $\pm 10mm$. Este margen puede incrementar si se tienen en cuenta las limitaciones debidas a las propiedades de la superficie de los objetos, como las que se describen más adelante.

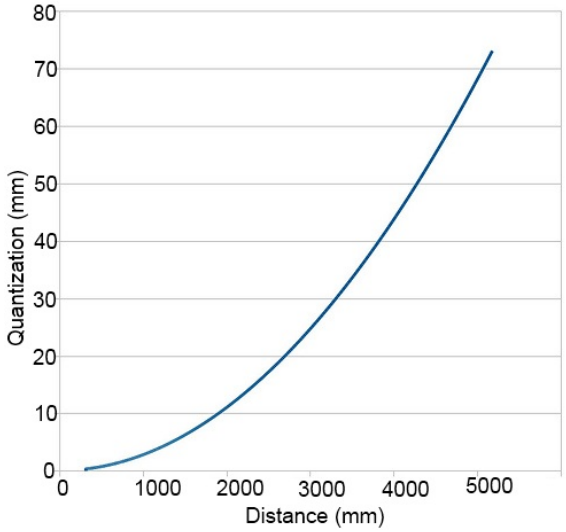



Figura 2.15: Cuantización en función de la distancia

- Del problema descrito en el punto anterior se deriva también una imprecisión en los bordes de los objetos. Dado que para la estimación de la profundidad se considera más de un pixel y al estar éstos dispersos, unas veces se tomará la profundidad del objeto más cercano y otras las del más lejano. Este problema es similar al mostrado en la imagen f) de la figura 2.9.

- Otro de los grandes problemas es el causado por la propia naturaleza de la luz proyectada por el sensor. La luz emitida por el proyector de infrarrojos, al impactar sobre un objeto, genera una sombra de éste en otro a mayor distancia, como se puede apreciar en la figura 2.16. El resultado es que no se puede determinar la profundidad en las zonas afectadas por dichas sombras. Esto se manifiesta como píxeles de valor cero ("zonas negras") en la imagen de profundidad.

Como ocurre con todo sistema óptico, la otra gran limitación viene determinada por las características de los objetos (factores externos) [49]. En general, atendiendo a la forma y propiedades de las superficies, se puede hacer la siguiente clasificación de objetos:

- Objetos translúcidos: los puntos de luz que impactan sobre éstos sufren una dispersión, haciendo indistinguible la deformación del patrón para el sensor.

- Objetos especulares: los puntos de luz tienden a impactar sobre el objeto reflejado.

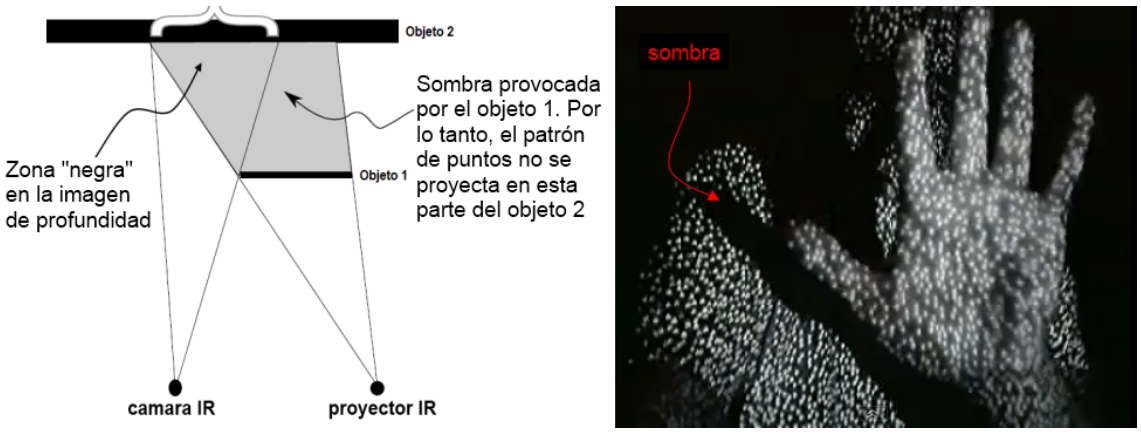



Figura 2.16: Efecto de las sombras en el sensor Kinect

Igual que antes, el patrón deformado no se puede distinguir. Un claro ejemplo son los suelos reflectantes.

- Objetos cóncavos o cavidades reflectantes: éstos pueden producir reflexiones dobles e inter-reflexiones. Es decir, un punto de luz que caiga sobre una superficie cóncava, puede rebotar en otra zona del mismo objeto (cóncavo), lo que produce un solapamiento de los puntos de luz, haciéndolos nuevamente irreconocibles parte el sensor.

Para mitigar el efecto negativo producido por esta clase objetos se emplean técnicas tan rudimentarias como, por ejemplo, cubrir con una laca opaca los objetos translucidos (o transparentes). Evidentemente, este método es solo válido para propósitos de medición. Existen otras técnicas alternativas que han sido propuestas para tratar objetos transparentes y especulares [50]. Recientemente, se han hecho grandes esfuerzos por tratar con escenas ópticamente complejas a través del rediseño de los patrones de luz [51]. Estos métodos han dado resultados verdaderamente prometedores en entornos tradicionalmente complejos, como cavidades metálicas altamente especulares y velas de cera translúcida [51].

Las zonas cuya profundidad no pudo ser resuelta, aparecerán como zonas negras (pixeles de valor cero) en la imagen de profundidad. Aparte, como ya se mencionó anteriormente, también hay que tener en cuenta la atenuación que sufre la luz, por lo que para objetos muy lejanos, tampoco se podrá determinar su profundidad. Asimismo, la inclinación de la superficie de los objetos respecto al proyector del haz de luz limita la detección de la profundidad. Si el rayo de luz es casi paralelo a la superficie no podrá incidir sobre la misma, haciendo imposible la estimación de la profundidad. En la imagen 2.17 se puede ver este efecto en la mesita de la esquina inferior derecha y en la superficie del suelo.

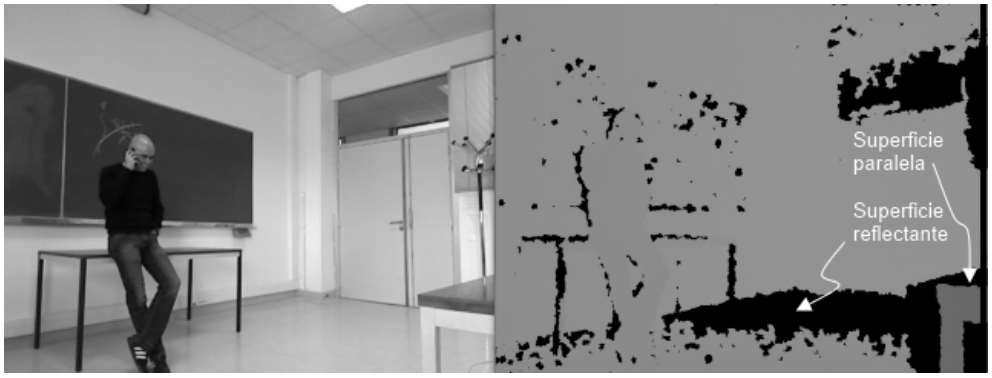



Figura 2.17: Efecto de superficie reflectante y paralela a los rayos de luz

### 2.4.3. Software para controlar el sensor Kinect

Existen una gran variedad de librerías disponibles, cada uno con sus *drivers* propios para controlar el sensor Kinect. Esto significa que no es posible combinar dos librerías diferentes (de propietarios diferentes) para crear una aplicación concreta.

Obviando las peculiaridades de cada entorno, el objetivo primordial de los mismos es obtener las imágenes de color y profundidad sincronizadas en el tiempo y alineadas, de modo que, para cada pixel, se tengan sus tres coordenadas $(x, y, z)$ en un espacio tridimensional de referencia dado. La sincronización en el tiempo no es posible en el sensor [52] según *Suat Gedikli*[5], con lo cual hay que tener cuidado en la elección de la aplicación a la que va destinado. En todos los entornos que se describen a continuación, la obtención de la imagen de color no tiene ninguna complejidad y sigue un procedimiento similar en todos ellos, por lo que centraremos nuestra atención en la obtención de la imagen de profundidad, que sí varía para cada entorno.

- MSDK (*Microsoft Development Kit*)

  Este es el kit de desarrollo oficial lanzado por la corporación de *Microsoft*. La documentación de Microsoft [45] afirma que el rango de distancias válido para el sensor de profundidad es de 4 a 11.5 pies (1.2 a 3.5 metros). Los valores de profundidad para cada pixel son devueltos en milímetros en dos bytes (16 bits), pero sólo los 12 bits menos significativos contienen la información "real" de profundidad, como ilustra la figura 2.18.

  Para valores de profundidad mayores de 3.5 metros, los bits restantes serán utilizados. Microsoft no dice nada acerca de la precisión de estos valores (mayores de 3.5 metros); pero, como es lógico, las degradaciones en la información de profundidad serán mayores.

[5] *Suat Gedikli* es ingeniero investigador en *Willow Garage* http://www.willowgarage.com/pages/people/suatgedikli

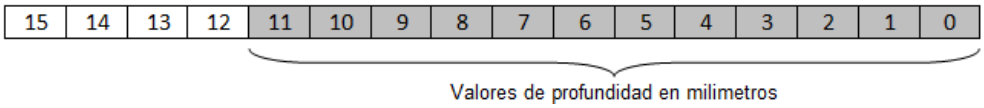



Figura 2.18: Formato de datos de profundidad en MSDK

Con respecto al alineamiento de las imágenes de color y profundidad, Microsoft proporciona una función (bastante inestable) para hacerlo. Esta función recibe como parámetros el valor y las coordenadas $x$ e $y$ (en píxeles) de un pixel de profundidad. Con esos datos, devuelve las coordenadas del pixel correspondiente en la imagen de color.

MSDK permite capturar los datos procedentes de la Kinect de dos maneras distintas:

- Modelo por sondeo (*polling model*): la aplicación abre un *stream* de datos (color o profundidad) y posteriormente llama a una función, que no retorna hasta que haya nuevos datos disponibles. Como parámetro de dicha función, se puede especificar un tiempo de espera (*timeout*). Un valor del *timeout* igual a 0 indica retornar inmediatamente, mientras que un valor de INFINITE indica no retornar hasta que hayan datos disponibles. En el caso de retornar inmediatamente, los datos que se cogen son los últimos almacenados en el *buffer* del *stream*.
- Modelo de eventos (*event model*): este modelo está basado en eventos. Cuando se crea un *stream* de datos, se le asocia a éste un objeto de clase *Evento*, que será "disparado" cuando dicho *stream* tenga nuevos datos.

Dado que la sincronización entre la cámara de color y el sensor de infrarrojos no es posible, cuando se abren dos *stream* de datos (color y profundidad) a la vez, se puede elegir una de las dos modalidades siguientes para leer los datos de los *buffers*. La primera consiste en proceder a la lectura cuando cualquiera de los dos *streams* tenga datos disponibles. El desfase entre las imágenes de color y profundidad, es el que se muestra en la gráfica izquierda de la figura 2.19. La segunda modalidad consiste en leer los datos cuando ambos *streams* tienen datos nuevos. El desfase en este caso se ilustra en la gráfica derecha del la figura 2.19.

- OpenNI SDK

  Este SDK pertenece a PrimeSense [53], compañía líder en "interfaces naturales" y encargada de la fabricación del sensor de profundidad para el dispositivo Kinect. OpenNI es la organización que se encarga del desarrollo del *software* para todos los dispositivos fabricados por PrimeSense. Al igual que MSDK, el valor de profundidad se devuelve en dos *bytes* y en milímetros.

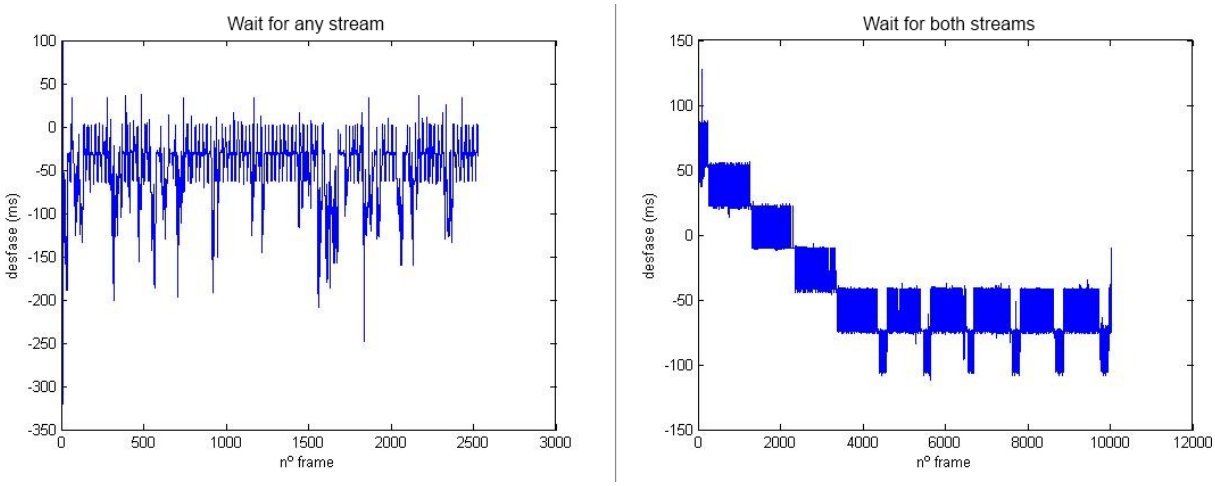



Figura 2.19: Desfase entre imágenes de color y profundidad en MSDK

Esta vez, la lectura de datos simultánea (imágenes de color y profundidad) se puede hacer de cuatro maneras distintas y con las siguientes funciones:

- ***WaitAndUpdateAll***: la aplicación abre los dos *streams* de datos (color y profundidad) y posteriormente llama a esta función que no retorna hasta que los dos *streams* tengan datos nuevos.
- ***WaitOneUpdateAll***: la aplicación abre los dos *streams* de datos y posteriormente llama a esta función, pasándole como parámetro uno de los dos *streams* («generadores»). La función no retorna hasta que el *stream* pasado a esta función tenga nuevos datos disponibles.
- ***WaitAnyUpdateAll***: la aplicación abre los dos *streams* de datos y posteriormente llama a esta función que no retorna hasta que cualquiera los dos *streams* tengan datos nuevos.
- ***WaitNoneUpdateAll***: La aplicación abre los dos *streams* de datos y posteriormente llama a esta función que retorna inmediatamente. Los datos que se leen son los últimos almacenados.

El desfase producido entre las imágenes de color y profundidad es el que se muestra en las gráficas de la figura 2.20. El desfase se calcula teniendo en cuenta el *timestamp* del último frame generado por cada *stream*.

En cuanto al alineamiento entre las imágenes de color y profundidad, OpenNI proporciona una función que permite mover el campo de visión del sensor de profundidad a del de color.

- OpenKinect

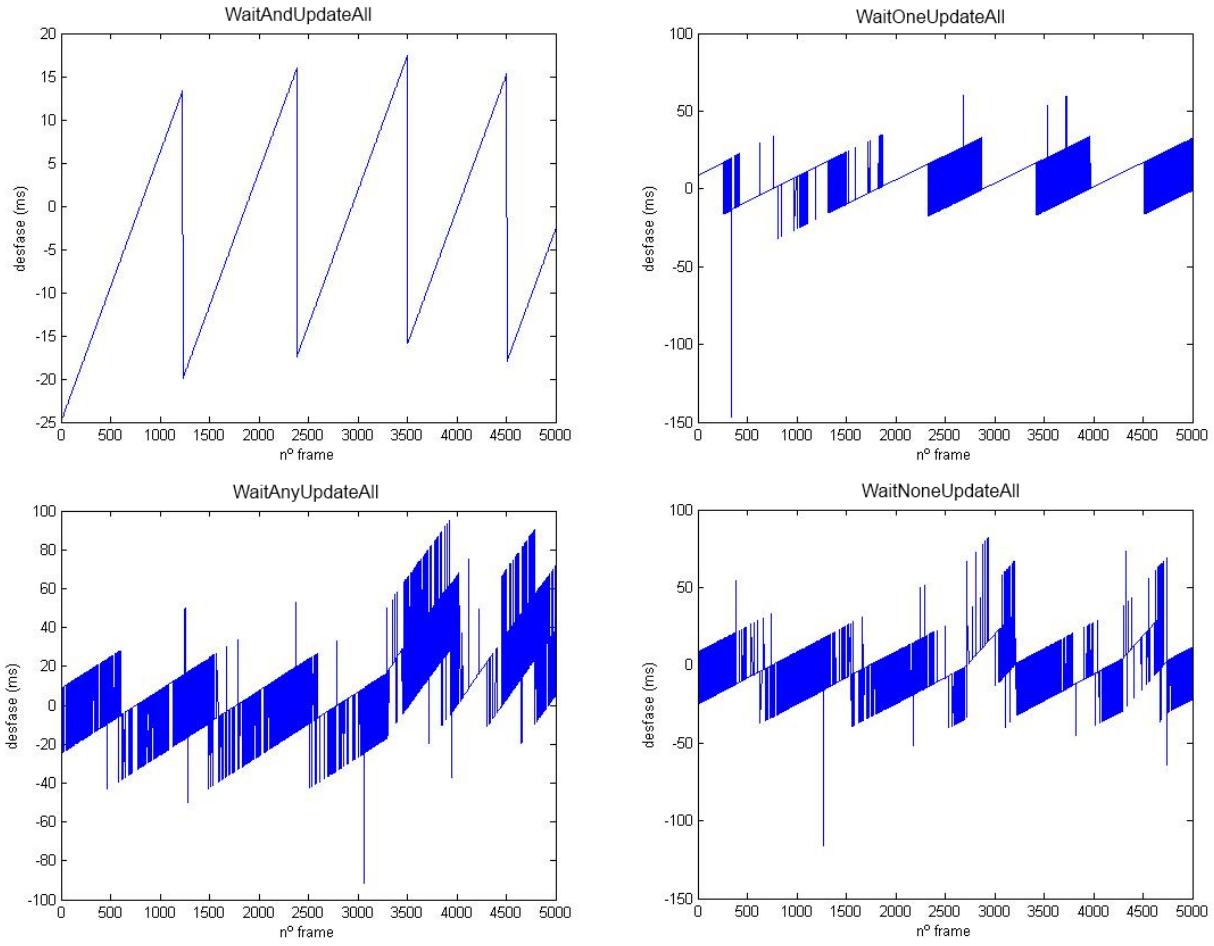



Figura 2.20: Desfase entre imágenes de color y profundidad en OpenNI

Antes de la aparición de los entornos de desarrollo oficiales (MSDK y OpenNI), desarrolladores de todo el mundo hicieron grandes esfuerzos de ingeniería inversa sobre la Kinect para poder interpretar y manipular los datos entregados por ésta. Una de las comunidades de desarrolladores que se ha consolidado fuertemente es precisamente OpenKinect [54], cuya labor está centrada en el desarrollo de *libfreenect*, el paquete *software* para controlar el sensor Kinect.

El formato de los datos de profundidad, en este caso, es el que se muestra en la figura 2.21. Los valores de profundidad son devueltos en 2 bytes, pero solo los 11 bits menos significativos contienen información de profundidad.

A diferencia de MSDK u OpenNI, estos valores no representan la profundidad directamente. Para obtener la distancia real (en metros) hay que utilizar la siguiente fórmula: $\frac{1}{r(-0{:}0030711016)+3{,}3309495161}$ [55], donde $r$ es el valor bruto de profundidad

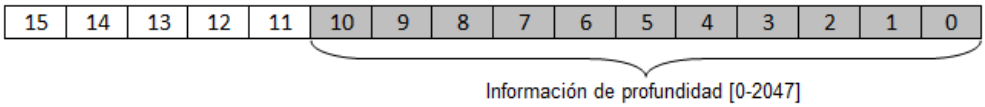



Figura 2.21: Formato de datos de profundidad en OpenKinect

devuelto en los 11 bits indicados en la figura 2.21. La representación gráfica de la función anterior se muestra en la figura 2.22.

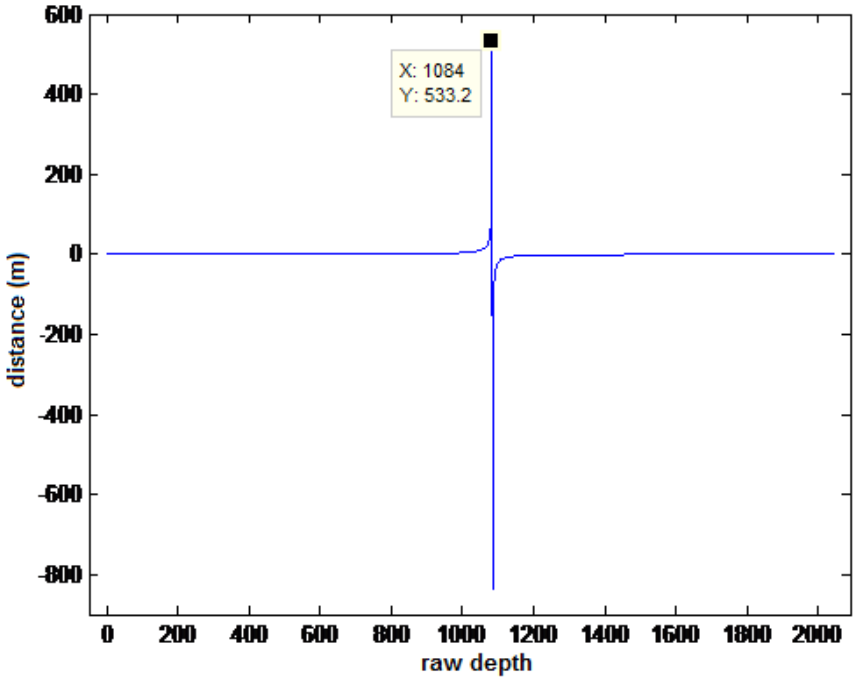



Figura 2.22: Valores brutos de profundidad frente a distancia real

Como se puede apreciar en la figura 2.22, solamente los valores de profundidad comprendidos entre 0 y 1084 son válidos, ya que son los únicos que dan valores de distancia positivos.

El alineamiento entre las imágenes de color y profundidad se puede realizar siguiendo el procedimiento indicado en [56].

### 2.4.4. Kinect vs ToF vs visión estéreo

Los dispositivos que utilizan la tecnología ToF (i.e. MESA SR4k o XZ422 (Canesta Inc.)) son la alternativa más directa al sensor Kinect. Sin embargo, su elevado coste (600€ - 5000€) dependiente de la resolución, precisión y alcance (4m - 60m), hace éstos sean utili-

zados en la mayoría de los casos dentro de laboratorios con fines de investigación. A continuación se citan algunas de las bondades y defectos de su funcionamiento y se lo compara con el sensor Kinect. En la figura 2.23 se muestran las imágenes de profundidad e intensidad obtenidas por los sensores IR de una cámara ToF (MESA SR4k) y el sensor Kinect.

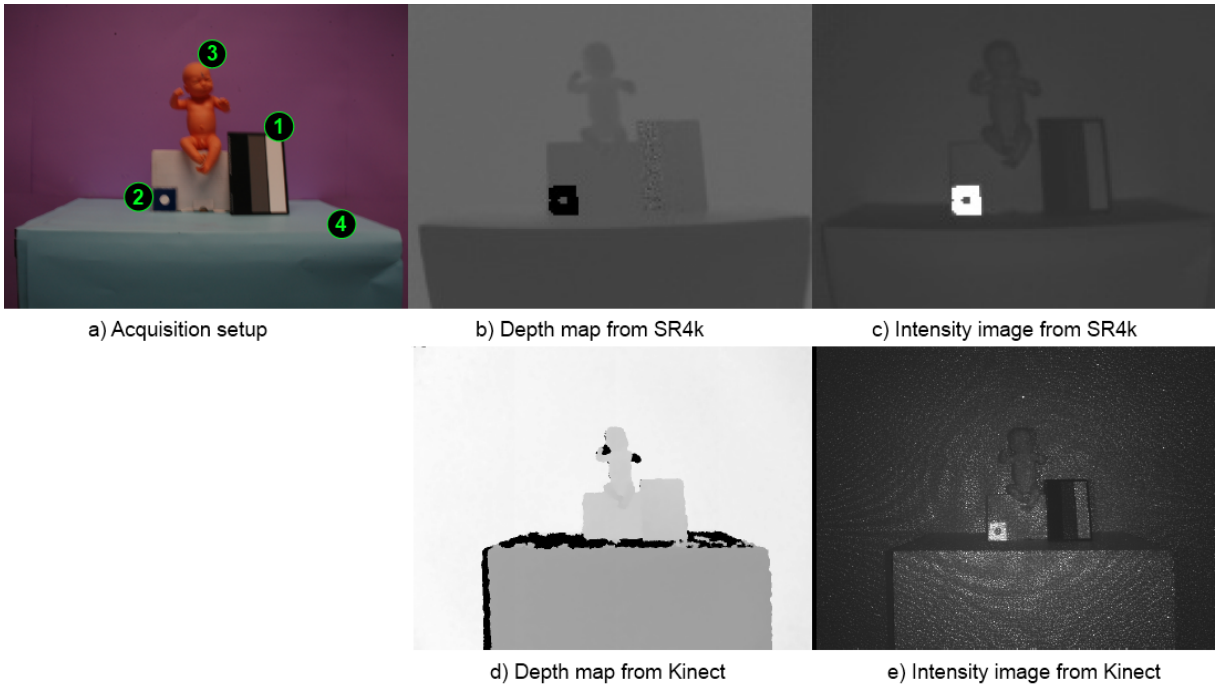



Figura 2.23: ToF (SR4k) vs Kinect

En la escena se han dejado objetos de diversas propiedades que pueden importunar su detección por parte del sensor de infrarrojos. Los objetos presentes y sus efectos son los siguientes:

- Una tabla de escala de grises con parches de diferente reflectividad (1). En el caso del SR4k es bastante visible la presencia de ruido gaussiano en el mapa de profundidad para el parche situado a la izquierda. El sensor Kinect, no tiene problemas con este objeto.
- Un marcador de la fotogrametría, caracterizado por una alta reflectancia cerca del IR (2). Las propiedades de este marcador causa una alta saturación en el caso de ToF, lo que produce una profundidad nula (color negro). En el caso del sensor Kinect, a pesar de que en la imagen de IR tiene una alta intensidad, es posible estimar la profundidad.
- Un muñeco que representa un objeto articulado en la escena (3). Mientras que el SR4k produce un mapa de profundidad regular en presencia de figuras articuladas,

unos “agujeros negros” afectan a la profundidad del sensor Kinect, donde los valores reales de profundidad no se pueden estimar (zona de los brazos).

- Una mesa grande con una superficie inclinada (4). El sensor Kinect tiene grandes problemas para detectar estas regiones como se explica en la sección 2.4.2. En el caso de ToF, no hay ningún inconveniente, puesto que la densidad de la luz IR es mayor que en el caso de la Kinect, donde la luz es muy dispersa, debida al patrón de puntos.

Como ya se mencionó en la sección 2.3, los métodos para la obtención de la profundidad se pueden clasificar en activos (*Time of flight*, luz estructurada y escaneo laser) o pasivos (Triangulación). Los primeros resaltan las características de la escena proyectando algún tipo de radiación especial (luz infrarroja) y los segundos se basan en imágenes de color obtenidas directamente en condiciones normales. En las tablas 2.1 2.2 2.3 se ofrece una comparativa entre estos dos métodos, analizando concretamente las características de los sensores ToF, Kinect y los sistemas de visión estéreo, así como sus ventajas y desventajas en el contexto de la video vigilancia [57].

| | **sensor ToF** | **sensor Kinect** | **visión estéreo** |
|---|---|---|---|
| **Resolución de profundidad** | Sub-centímetro (si las condiciones de cromaticidad se satisfacen). | Sub-centímetro (en función de la distancia de los objetos). | Sub-milímetro (si las imágenes son altamente texturizadas). |
| **Resolución espacial** | Media (alrededor de QCIF, 144x176 píxeles). | Resoluciones de 640×480 (VGA) y 320×240 (QVGA) | Alta (por encima de 4CIF, 704x576 píxeles). |
| **Portabilidad** | Las dimensiones son las mismas de una cámara normal. | Tamaño fijo, ligeramente superior a las cámaras normales. | Dos cámaras de vídeo son necesarios y también fuente de luz externa. |
| **Coste computacional** | La medición de fase e intensidad de la señal tienen lugar en una FPGA. | El requerido para procesar la imágen de intensidad (imagen de infrarojos) y obtener un mapa de profundidad. | Alta carga de trabajo (la etapa de calibración y el proceso de búsqueda de correspondencias son difíciles). |
| **precio** | Elevado para un prototipo adaptado (600€ - 5000€). | 150€ | Esto depende de la calidad del sistema de visión estéreo. |

Tabla 2.1: Comparación de características entre ToF, Kinect y visión estéreo

| | **sensor ToF** | **sensor Kinect** | **visión estéreo** |
|---|---|---|---|
| **Condiciones de iluminación** | Medición exacta de la profundidad en cualquier condición de iluminación. | Sensible a las variaciones de iluminación. No puede operar en entornos con sobreiluminados como la luz solar. | Sensible a las variaciones de iluminación y luces artificiales. No se puede operar en ambientes oscuros/claros. |
| **Presencia de sombras** | No afecta. | No afecta. | Disminución del rendimiento en la segmentación, reconocimiento, etc.. |
| **Oclusiones parciales** | Objetos parcialmente ocluidos se detectan como separados (si están a diferentes profundidades). | Objetos parcialmente ocluidos se detectan como separados (si están a diferentes profundidades). | Debido a la ambigüedad proyectiva, las oclusiones son difíciles de detectar (fusión de blobs) y manejar (estrategias de seguimiento predictivo). |
| **Apariencia de los objetos** | Se evita el camuflaje, pero la apariencia de los objetos puede afectar a la precisión de los objetos. | Las propiedades de algunos objetos pueden provocar valores de profundidad nulos. | Los efectos de camuflaje aparecen cuando el primer plano y el fondo presentan la misma apariencia. |

Tabla 2.2: Ventajas del sensor ToF en el contexto de la video vigilancia

De las desventajas mostradas en la tabla 2.3, las relacionadas con los objetos reflectantes y el campo de visión también afectan al sensor Kinect como se vio en la sección 2.4.2. Las propiedades de la superficie de los objetos afectan a la estimación de la profundidad, así como la disposición de éstos respecto a la campo de visión (i.e. superficies paralelas a los rayos de luz).

| | **Descripción** |
|---|---|
| ***Aliasing*** | Reduce la máxima profundidad sin ambigüedades alcanzable (hasta 7,5 m). |
| **Efectos de multitrayecto** | La medida de profundidad resulta muy dañada cuando la superficie del objetivo presenta esquinas. |
| **Objetos reflectantes** | Los materiales que tienen diferentes colores muestran propiedades diferentes reflexión que afectan la intensidad de luz reflejada y, por lo tanto, la resolución de profundidad. |
| **Campo de visión** | Por lo general, está limitado de modo que es necesario un posicionamiento preciso. Una arquitectura de *pan-tilt* (inclinación) podría ser útil. |

Tabla 2.3: Desventajas del sensor ToF en el contexto de la video vigilancia

## 2.5. Uso de la información de profundidad en vídeo

La información de profundidad está presente en una gran diversidad aplicacciones. A continuación se se muestra su uso para segmentación de objetos (sección 2.5.1) y su uso para el análisis de eventos en general (sección 2.5.2).

### 2.5.1. Segmentación

En el área de investigación, concretamente en procesamiento de imagen, la información de profundidad permite resolver problemas que solo con información de color son difíciles de tratar. En particular, el hecho de disponer de una tercera dimensión para cada pixel de la imagen (eje $z$) hace que problemas tan básicos y fundamentales como la segmentación, se resuelvan con mayor facilidad y precisión. A modo de ejemplo, uno de los problemas que se aborda en este proyecto es precisamente ese. En la figura 2.24 se observa una segmentación basada en la profundidad: lo que está más allá de cierta distancia se ignora.

La segmentación basada en profundidad se puede realizar siguiendo una amplia varie-

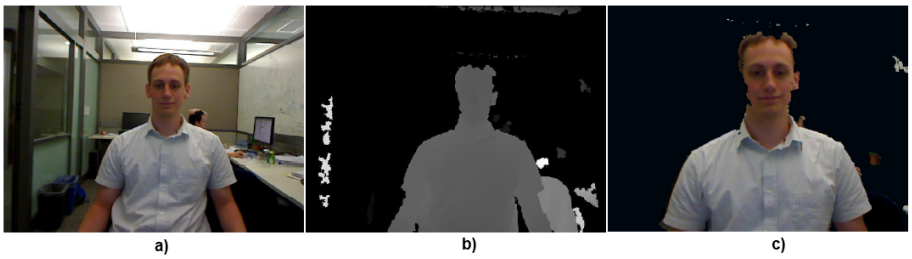



Figura 2.24: Segmención basada en la profundidad. a) Imagen en color. b) Imagen de profundidad. c) Imagen segmentada

dad de criterios. Es el caso de [58], donde, en primer lugar se corrigen los defectos de la imagen de profundidad (obtenida del sensor Kinect), utilizando la imagen de color correspondiente, y posteriormente se realiza la segmentación, con la profundidad ya corregida. En la misma línea, en [59] se utiliza la profundidad (obtenida de un sensor ToF) para mejorar la segmentación basada en color en espacios interiores. En este caso, para la obtención del modelo de fondo se modelan los píxeles de acuerdo a una distribución gaussiana de acuerdo a la ecuación 2.1.

$$P(X_t) = \frac{1}{(2\pi)^{\frac{3}{2}}\sigma_t^3} e^{-\left(\frac{(X_t-\mu_t)^2}{2\sigma_t^2}\right)} \tag{2.1}$$

Donde $\mu_t$ y $\sigma_t^2$ son la media y la varianza de los valores de los píxeles y $X_t$ es el píxel de entrada en el *frame* $t$.

El modelo de fondo final para el pixel $X_t$ se puede obtener considerando ambos modelos (color y de profundidad) como dos canales distintos, lo que generaría un modelo de fondo gaussiano bi-modal. Es decir, se tendría un modelo de fondo en el que cada pixel está modelado por 2 gaussianas (MoG). Sin embargo, se prueba que se obtienen mejores resultados considerando únicamente el *foreground* basado en profundidad, cuya obtención se resume en la ecuación 2.2.

$$|X_t - \mu_t| > k\sigma_t \tag{2.2}$$

Donde $k$ es un umbral escalado por la variación local del pixel $\sigma_t$.

El sistema se prueba en dos situaciones extremas: en la primera hay una fuerte componente de falsos positivos provocados por un proyector y en la segunda, una fuerte sobre iluminación provocada por la luz solar (figura 2.25).

En el afán por aumentar la fiabilidad de la máscara de *foreground*, en [60] se utilizan la información de color y de profundidad (obtenida estereográficamente) a la vez para obtener el *foreground* final. El método empieza por obtener un modelo de fondo (gaussiano) para cada tipo de información. Para cada tipo se consideran varios modos, pero solamente se elige uno para el modelo. En el caso de la profundidad, se elige el modo más alto en profundidad y cuyos valores sean válidos (profundidad no nula) en al menos un $T\,\%$ (T=10) de la porción de frames considerados. Asimimo, para el caso de color, se elige el modo más común.

Posteriormente, el *foreground* basado en color se obtiene según la ecuación 2.3.

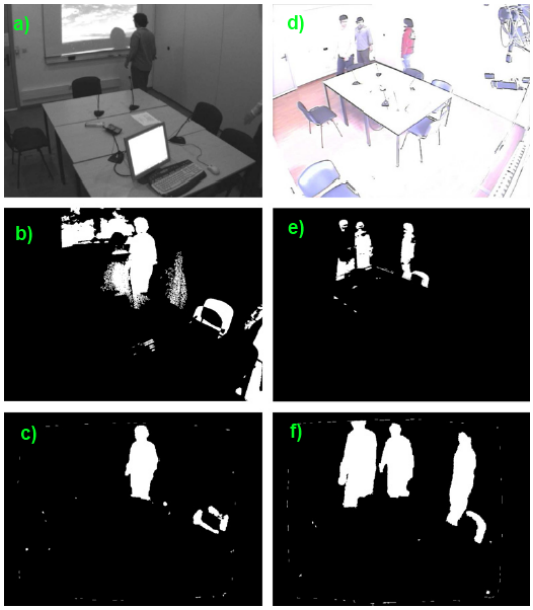



Figura 2.25: Segmentación del frente. a) Escena con un proyector. b) Detección del frente de *a)* con color. c) Detección del frente de *a)* con profundidad. d) Habitación con luz solar. e) Detección del frente de *d)* con color. f) Detección del frente de *d)* con profundidad.

$$
\begin{aligned}
Fc \quad \equiv \quad & (Y_{Valid}(Y_m) \wedge Y_{Valid}(Y_i) \wedge (\triangle color > c\sigma)) \vee \\
& (Y_{Valid}(Y_m) \wedge ((\frac{Y_i}{Y_m} < shad) \vee (\frac{Y_i}{Y_m} > reflect)) \vee \\
& (Y_{Valid}(Y_m) \wedge (Y_i > \alpha Y_{min}))
\end{aligned}
\tag{2.3}
$$

Donde $Y_i$ es la luminancia del *frame* i-ésimo, $Y_m$ es la luminancia del modelo de fondo; $Y_{Valid}(Y) \equiv Y > Y_{min}$, siendo $Y_{min}$ un umbral de luminancia mínimo; $\triangle color$ es la diferencia de matiz (H) entre el *frame* actual y el modelo de fondo, $shad$ y $reflrect$ son los umbrales para discriminar las sombras e interreflexiones, respectivamente; $\alpha$ es un umbral escalado por $Y_{min}$ para filtrar del *frame* actual según la luminancia, $\sigma$ es la varianza del del modelo, y $c$ es un parámetro ajustado dinámicamente, según la información de profundidad. Su valor es más alto, cuando en el *foreground* de profundidad los píxeles de *frame* i-ésimo aparecen como pertenecientes al *background*.

Por otro lado, la máscara de *foreground* basada en profundidad se obtiene de acuerdo a la ecuación 2.4.

$$
Fr \equiv Valid(r_i) \wedge (\nabla r_i < G)^{\cdot} \wedge \neg(Valid(r_m) \wedge (|r_i - r_m| < k\sigma) \tag{2.4}
$$

Donde, $r_i$ es la profundidad el *frame* i-ésimo, $r_m$ es la profundidad en el modelo de fondo, $k\sigma$ es el parámetro de umbralización, y $\nabla r_i$ es el gradiente local de $r_i$. Los valores de gradiente por encima de $G$ representan discontinuidades en la profundidad, por lo que este valor se fija en función de la "lisura" esperada de objetos en primer plano.

Finalmente, se obtiene el *foreground* final como un OR lógico entre ambas máscaras (ecuación 2.5).

$$F \equiv F_r \vee F_c \tag{2.5}$$

Este método presenta inconvenientes en zonas amplias sin profundidad y en objetos de color muy similar al fondo. Además, los objetos segmentados presentan un "halo" a su alrededor debido al *foreground* de profundidad. La figura 2.26 ilustra el resultado de este método.

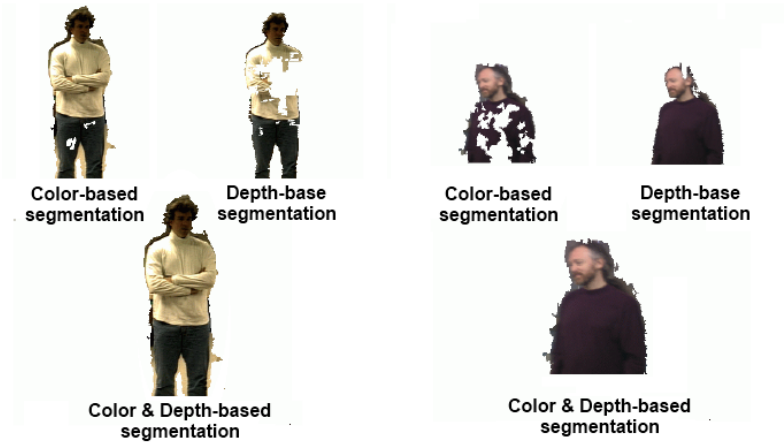



Figura 2.26: Resultados de segmentación

En otras aproximaciones para mejorar la máscara de *foreground,* como la empleada en[61], se combinan la información de color y las imagenes proporcionadas por un sensor PMD (Photonic Mixer Device), entre las que se encuentra la de profundidad. La idea es obtener un *foreground* para cada tipo de imagen y luego decidir si un pixel realmente pertenece al frente si aparecen en al menos uno de los *foregrounds* mencionados. La principal desventaja es la presencia de detectados como *foreground* erróneamente, debido a la imprecisión de las imágenes utilizadas. La figura 2.27 ilustra las imágenes implicadas.

### 2.5.2. Eventos

La industria del entretenimiento, sobre todo, está tomando gran ventaja de la información de profundidad con la creación videojuegos que pueden ser controlados sin necesidad


Figura 2.27: *RGB values*: Imagen de color. *Distance* $d$: Imagen de profundidad. *Amplitude* $A$: Intensidad de la señal utilizada para calcular $d$. *Intensity* $I$: Luminancia de la escena.

de mandos físicos, simplemente con señales gestuales de las manos o del cuerpo entero [62]. En la figura 2.28 aparece una persona realizando una serie de gestos corporales que son vectorizados y posteriormente convertidos a instrucciones. En este caso, el entorno de aplicación está controlado, de modo que se pueda obtener una imágen de profundidad lo más uniforme posible. Además, la distancia del frente con respecto al fondo es bastante alta, con lo que la segmentación es menos ambiugua. Con todo, el problema se encuentra únicamente en corregir pequeños defectos de las imágenes de profundidad (i.e. interpolación) y en la complejidad implicada en los algoritmos para el análisis de gestos.

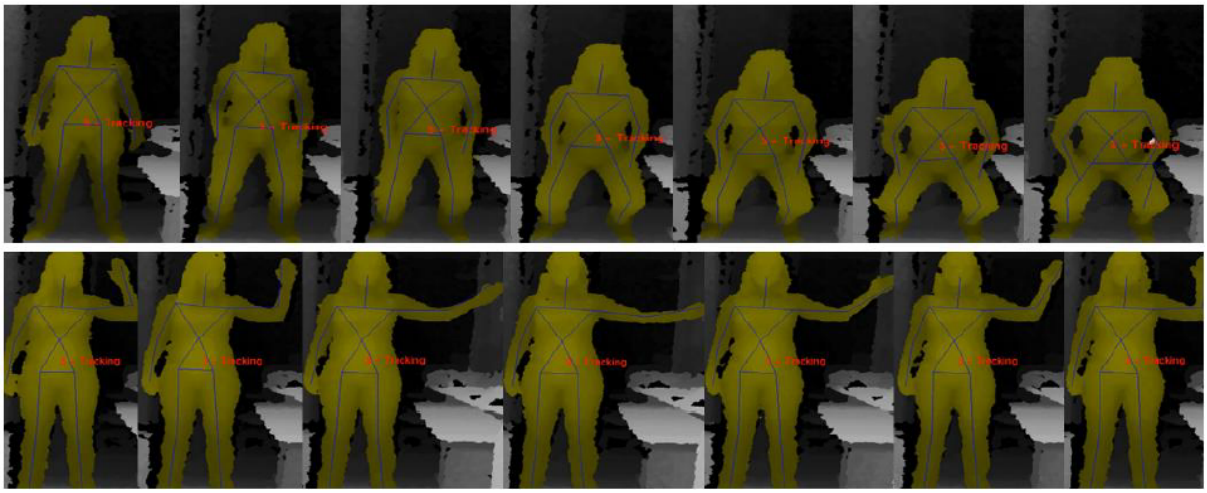


Figura 2.28: Modelos creados a partir de gestos corporales

Basado también en el análisis de gestos corporales y extensas bases de datos, en [63] se pretende identificar las actividades que una persona realiza en su vida cotidiana (Figura 2.29). En las secuencias de la base de datos, las distancias horizontales y verticales de la cámara respecto al centro de la escena es de aproximadamente 2 metros y la distancia media de una persona es de 3 metros. Esta configuración geométrica es adecuada para el hogar o la supervisión de una sala de hospital, por ejemplo. El problema principal aquí es obtener los parámetros adecuados de un determinado evento para compararlo con los

almacenados en las bases de datos.

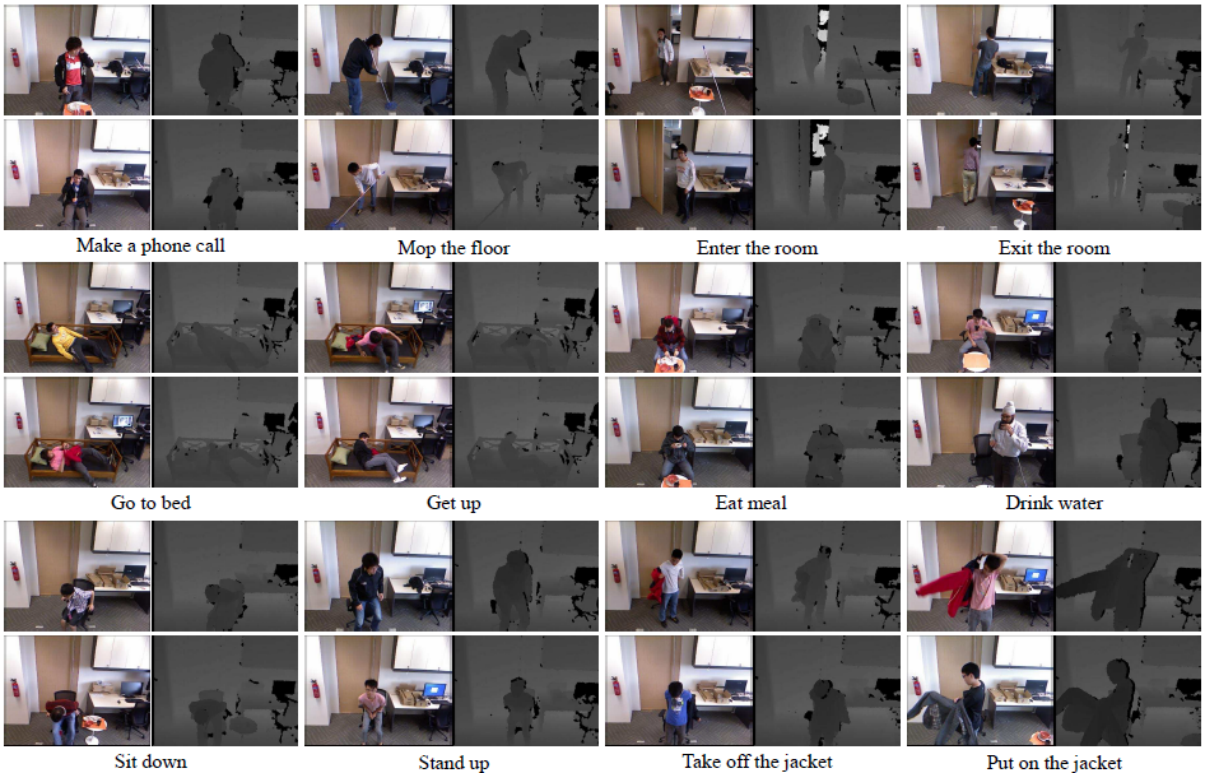



Figura 2.29: *Dataset* de actividades humanas cotidianas

Una de las tareas que supone un desafío en el análisis de eventos es el *tracking* de personas. En este campo, en [64] el tracking se realiza a nivel de *blob*, explotando la posición y velocidad del centroide. Los *blobs* se obtienen por una segmentación basada en profundidad. Las oclusiones en el sistema son poco frecuentes, dado que la cámara está en el techo. En el caso de ocurrir oclusiones (personas muy jutas), el problema se resuelve mediante una segmentación iterativa. Es decir, se va reduciendo el umbral de segmentación (distancia persona-sensor) hasta tener todos los *blobs* aislados. La figura [64] ilustra los resultados de este método.

Otro método novedoso para el *tracking* de personas basado en profundidad es el que se propone en [65], donde se detecta a la persona analizando una parte del cuerpo, concretamente la cabeza. Para ello se vale de un modelado 3D de la cabeza y el contorno de la misma, obtenida por un detector de bordes (Canny). Una vez localizada la cabeza, se extrae el resto del cuerpo (figura 2.31). El sistema no genera falsos positivos, pero sí puede presentar falsos negativos para cabezas ocluidas o cuando la mitad del cuerpo de la persona está fuera del *frame*. Además, el sistema solo funciona para espacios interiores debido a las limitaciones del dispositivo utilizado (sensor Kinect).

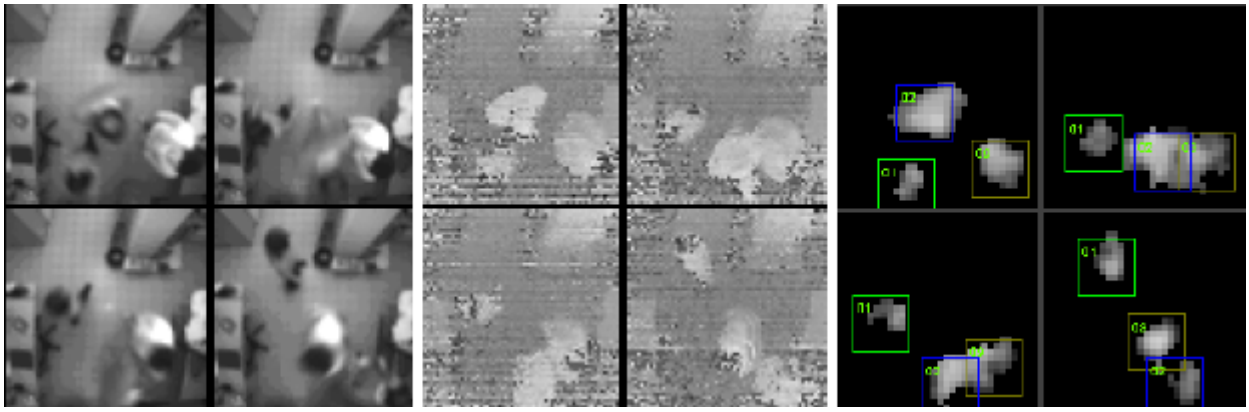


Figura 2.30: Seguimiento de personas

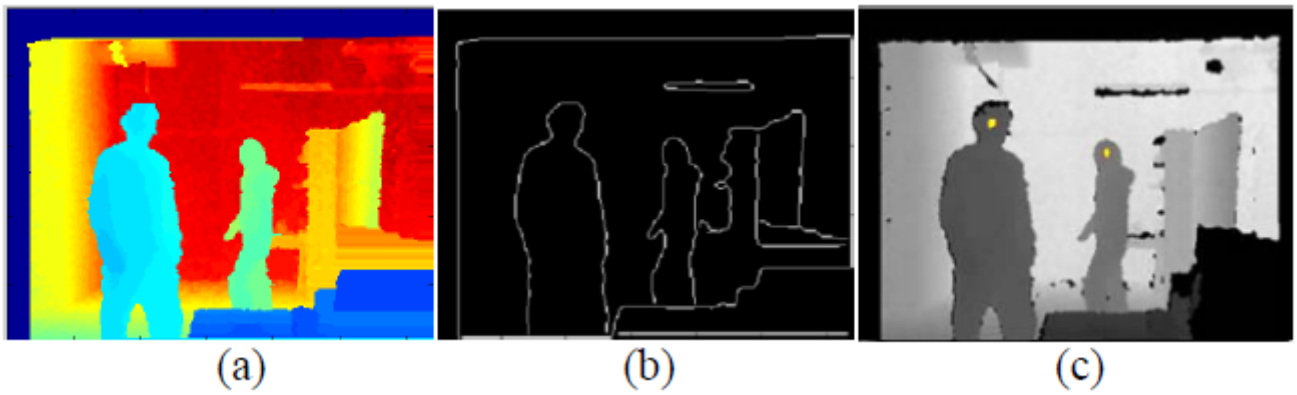



Figura 2.31: *Traking* de personas basado en la detección de cabezas. (a) Imagen de profundidad. (b) Bordes de la escena. (c) Cabezas detectadas

Por último, la aparición del sensor Kinect ha hecho que aplicaciones de este tipo se incrementen exponencialmente. En [66] se puede ver una gran multitud de aplicaciones de esta índole. Dada la precisión relativamente alta y el bajo coste del sensor Kinect, su uso se ha extendido a campos como la educación, terapias o la medicina, como ya se mencionó antes [4].

# Capítulo 3

# Prototipo del VPU-Lab

## 3.1. Introducción

El prototipo existente en el grupo VPU-Lab sigue un esquema muy similar al descrito en la sección 2.2. Una vez adquiridos los *frames* de la secuencia de video, se realiza una segmentación para extraer los píxeles cambiantes (Detección del primer plano). La etapa siguiente (Detección de *blobs* estacionarios) analiza las regiones conexas de la máscara binaria entregada por la etapa anterior. El resultado es otra máscara binaria con todos los *blobs* estáticos listos para ser clasificados como pertenecientes a objetos o personas. Esta tarea se lleva a cabo en la etapa de Clasificación de objetos. Por último, descartados los blobs pertenecientes a personas, la etapa de Discriminación de objetos determina qué *blobs* corresponden a objetos robados y cuales a objetos abandonados.

Dada la versatilidad del sistema, su uso se puede extender a la detección de otros tipos de eventos previamente definidos. Por ejemplo, la velocidad y la dirección en la que se mueve una persona. O también se pueden utilizar los datos de video generados para indexar videos almacenados para una posterior búsqueda semántica *offline*. Si además se utiliza información de contexto, como por ejemplo que en ciertas zonas jamás puede haber personas, se puede incrementar la precisión y la fiabilidad de cada etapa.

En el resto de este capítulo se presentan las diferentes etapas del sistema: detección del primer plano (sección 3.2), detección de objetos estacionarios (sección 3.3), clasificación de objetos (sección 3.4), y discriminación de objetos (sección 3.5).

## 3.2. Detección del primer plano

El método utilizado en el sistema del VPU-Lab es el que se propone en [67]. Para la obtención del modelo de fondo, se considera una pequeña porción de *frames* consecutivos y

se calcula el valor medio de cada uno de los píxeles. Posteriormente, se calcula la distancia de cada *frame* entrante respecto a dicho modelo. Para ello, para cada pixel se elige una ventana a su alrededor de dimensiones $2Wx2W$ y se calcula el cuadrado de la diferencia entre el *frame* actual y el modelo de fondo para todos los pixeles contenidos en la ventana. El resultado final es máscara binaria que se obtiene umbralizando el sumatorio de esos cuadrados como se muestra en la siguiente expresión:

$$F(I[x,y]) \Longleftrightarrow \sum_{i=-W}^{W} \sum_{j=-W}^{W} (|I[x+i,y+j] - B[x+i,y+j]|)^2 > \beta \tag{3.1}$$

Donde $W$ es una ventana cuadrada centrada en cada pixel (de coordenadas $x$ e $y$), $\beta$ es un umbral para la segmentación del frente, $I$ es el *frame* actual, $B$ es el modelo de fondo y $F$ es la máscara binaria con los píxeles del frente a uno.

Antes de pasar la máscara $F$ a la siguiente etapa es necesario realizar un post-procesado para eliminar las sombras debidas a personas u objetos y el ruido introducido por el propio dispositivo de captura. Para abordar el primer problema se realiza un procesado haciendo uso del espacio de color HSV [68]. Este modelo color permite tener bien diferenciados tres propiedades de cada pixel: matiz (*Hue*), saturación (*Saturation*) e intensidad (*Value*). Así, es de esperar que las sombras afecten en mayor medida a la intensidad y la saturación, siendo menor su efecto en el matiz o cromacidad. Dicho esto, la decisión de si una región marcada como frente es sombra o no se rige por la siguiente expresión.

$$SP(x,y) = \begin{cases} 1 & if\ \alpha \leq \frac{I_V(x,y)}{B_V(x,y)} \leq \beta \wedge D_S \leq \tau_S \wedge D_H \leq \tau_H \\ 0 & otherwise \end{cases} \tag{3.2}$$

Donde $SP(x,y)$ es la máscara de primer plano que resalta los píxeles que pertenecen a sombras en las coordenadas $(x,y)$; $I$ y $B$ son el *frame* actual y el modelo de fondo, respectivamente. Los subíndices $H$, $S$ y $V$ indican el canal en el espacio de color HSV; $DS$ y $DH$ denotan la diferencia de cromaticidad entre el *frame* actual y el modelo de fondo para ambos canales (matiz y saturación); y $\beta$, $\tau_S$ y $\tau_H$ son umbrales de decisión.

Al problema de ruido introducido por la cámara hay que añadir las "manchas" dejadas por el proceso de eliminación de sombras. Para tratar el problema se utilizan operaciones morfológicas de erosión y reconstrucción, más específicamente se utiliza un método conocido como "*Opening by Reconstruction of Erosion*" [69]. Esta técnica consigue eliminar pequeños *blobs* considerandos como ruido, mientras que mantiene la forma y tamaño de otros. Las instrucciones seguidas por dicha técnica se muestran en el algoritmo 3.1.

**Algoritmo 3.1** *Opening by Reconstruction of Erosion*

1) Imagen de entrada: $X$ (*foreground* con sombras suprimidas)
2) Obtener una imagen marcador $Y$, resultado de erosionar $X$ con el elemento estructurante $B$

- $Y = X \ominus B$ (erosión)

3) Inicializa $H$ con $Y$ ($H = Y$) y procede iterativamente así:

- $D = H_k \oplus B$ (dilatación)
- $H_{k+1} = D \cap X$
- Condición de parada: $H_{k+1} = H_k$

## 3.3. Detección de objetos estacionarios

Una vez que se han suprimido las sombras y se ha filtrado el ruido, esta etapa analiza los *blobs* de la máscara binara del frente y determina cuales pertenecen a regiones estáticas. La figura 3.1 ilustra el procedimiento llevado a cabo en esta etapa para conseguir este objetivo [70].

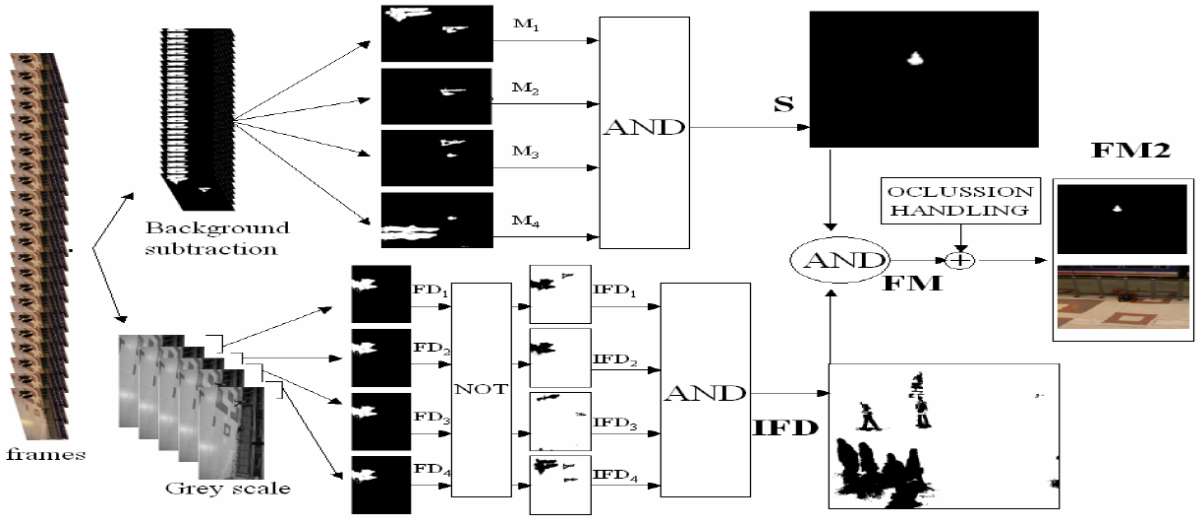



Figura 3.1: Algoritmo para la detección de regiones estacionarias

Se parte de las máscaras del *foreground* obtenidas de la etapa previa (sección 3.2). En ellas, los píxeles marcados a "1" destacan las regiones pertenecientes a objetos del frente. La decisión de si dichas regiones son estáticas o no se realiza en cuatro fases. En cada una de ellas se obtienen las máscaras $S$, $IFD$, $FM$ y $FM2$, respectivamente, como se puede

apreciar en la figura 3.1. La máscara $S$ es el resultado de un AND lógico de 6 muestras del *foreground* obtenidas en los últimos 30 segundos. Los positivos de esta máscara recalcan las regiones estacionarias; sin embargo, también pueden aparecer como positivos (falsos positivos) aquellas zonas donde haya mucha afluencia de personas, ya que puede darse el caso de que justo en el momento del muestreo haya alguien pasando por esa zona. La segunda fase se encarga de mitigar este problema haciendo uso de una técnica conocida como *frame difference,* que consiste en diferenciar el *frame* actual respecto al inmediatamente anterior. Una umbralización sobre la máscara entregada por esta técnica produce una máscara binaria, cuya negación produce la máscara $IFD$ que marca con "1" los píxeles donde no ha habido movimiento. Así, los pixeles a "1" de $S$ que coincidan con los píxeles a "0" de $IFD$ son los que corresponden a personas en movimiento. Por lo tanto, la máscara con las regiones estáticas ($FM$) se obtiene de hacer un AND lógico entre las máscaras $S$ e $IFD$. No obstante, algunas de la regiones descartadas como estáticas pueden pertenecer a objetos ocluidos por las personas en movimiento. Este problema se trata realizando un seguimiento de cada *blob* marcado como estático en $FM$. Si en el análisis actual desaparece una región marcada como estática justo antes, se comprueba que si realmente es una región estática desaparecida o se trata de una oclusión. Será una oclusión cuando se detecte una región en movimiento ($S = 1$ e $IFD = 0$). Si se detecta una oclusión, se elige el *bounding box* del objeto ocluido (almacenado en memoria) y se calcula el porcentaje de píxeles activos. Si este porcentaje es superior a un umbral determinado $\tau$, entonces se modifica $FM$ para conservar el objeto ocluido, lo que da como resultado la máscara $FM2$, que es la máscara final con todas las regiones estáticas. El umbral $\tau$ se elige en función del porcentaje de píxeles activos de la región inicial sin oclusiones.

## 3.4. Clasificación de objetos

En esta etapa, las regiones estacionarias de la máscara obtenida de la etapa anterior se asocian bien a personas o bien a objetos. Para llevar a cabo esta tarea se aprovechan dos propiedad de las regiones estáticas: su *bounding box* y su densidad [71]. El primero es un rectángulo que rodea el *blob* entero con la mínima área posible y el segundo es el porcentaje de píxeles activos del *blob* respecto al rectángulo que lo rodea, su *bounding box*.

Del *bounding box* se calcula su relación de aspecto ($ancho/altura$) y este valor se modela de acuerdo a una distribución gaussiana de media $\mu$ y varianza $\sigma$. Experimentalmente, una media de $\mu = 0{,}3$ y una varianza de $\sigma = 0{,}2$ son los que mejor se ajustan al perfil de una persona. Asimismo, el valor de densidad que define a una persona es de un $70 - 75\,\%$ . Ambos valores (relación de aspecto y densidad) se combinan por medio de una inferencia Bayesiana para realizar la clasificación final y así poder distinguir entre dos tipos: personas

y no-personas.

## 3.5. Discriminación de objetos

El proceso de detección culmina en esta etapa y en ella se determina si las regiones estáticas no pertenecientes a personas corresponden a objetos robados o abandonados. Además de la máscara con las regiones estáticas, procedente de la etapa anterior, es necesaria la máscara del frente, obtenida en la primera etapa (sección 3.2), así como los *frames* de color actuales y del modelo de fondo. El esquema de la figura 3.2 muestra como se combina esta información para llevar a cabo la discriminación entre objetos robados y abandonados [22].

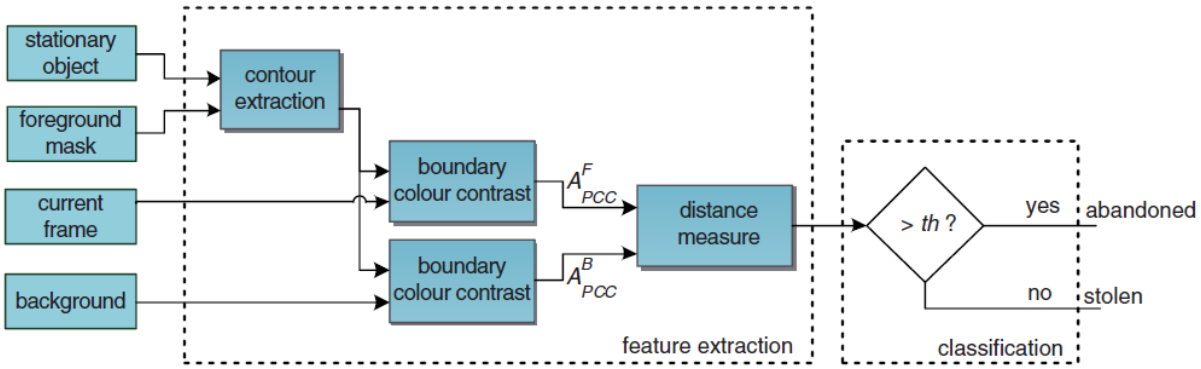



Figura 3.2: Esquema propuesto para la discriminación entre objetos robados y abandonados

El procedimiento seguido comienza por encontrar los bordes de la región estática por medio de un detector de Canny. Del contorno hallado $C_t$ se eligen $N$ puntos y se traza por cada uno de ellos una recta normal (a la línea de contorno) de longitud $2L + 1$ (figura 3.3). En cada extremo de la recta se elige una ventana de dimensiones $MxM$ centrada en los píxeles $P_o$ y $P_i$, respectivamente.

Tanto para el *frame* actual como para el modelo de fondo se calcula un parámetro que mide la similitud entre los pixeles de los extremos. El criterio elegido para medir dicha similitud es la distancia en contraste entre los puntos $P_o$ y $P_i$, *boundary spatial color contrast (BSCC)*, y se define de la siguiente manera:

$$BSCC(F_t, C_t, i) = \frac{\left\| W^i_O(t) - W^i_I(t) \right\|}{\sqrt{3 \times 255^2}} \tag{3.3}$$

Donde $W_O$ y $W_I$ son los valores de color promedio calculados en la vecindad $M \times M$ de los puntos de $P_O$ y $P_I$ (en el espacio de color RGB) para el i-ésimo punto del contorno $C_t$ de

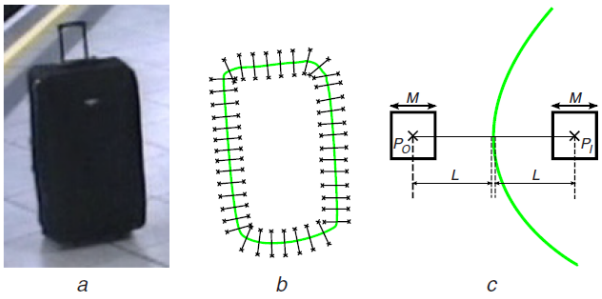



Figura 3.3: a) Objeto estático del frente. b) Puntos análizados a lo largo del contorno c) Punto analizado del contorno

la imagen $F_t$, que puede ser el *frame* actual o el modelo de fondo. Este parámetro se calcula para todos los $N$ puntos y se obtiene el valor medio de $BSCC$ como:

$$z(F_t, C_t) = \frac{1}{K_t} \sum_{i=1}^{K_t} BSCC(F, C_t, i) \tag{3.4}$$

Donde $BSCC$ es la distancia en contraste de color del i-ésimo punto, $K_t$ es el total de puntos válidos de los $N$ elegidos inicialmente. Se consideran puntos no válidos aquellos cuya vecindad $M{\times}M$ de los puntos $P_O$ y $P_I$ caen fuera de las fronteras de la imagen.

Esta media se calcula para el modelo de fondo y para el *frame* actual, lo que produce $z(B_t, C_t)$ y $z(F_t, C_t)$ , respectivamente. Para el primer caso, $z(B_t, C_t)$ , se espera que su valor sea próximo a cero cuando se trate de un objeto abandonado y un valor mayor para el caso de objetos robados, debido a la diferencia de contraste a los alrededores del contorno $C_t$. Para el segundo caso se espera que ocurra justamente lo contrario. La diferencia de estos valores medios genera otro parámetro $SPCC$, que es la puntuación final que será comparada con el umbral $th$ indicado en la figura 3.2.

$$S_{PCC} = z(B_t, C_t) - z(F_t, C_t) \tag{3.5}$$

# Capítulo 4

# Integración del Sensor Kinect

## 4.1. Introducción

El propósito de utilizar el Sensor Kinect es conseguir un sistema robusto para la detección del robo o abandono de objetos. El funcionamiento del mismo está restringido solamente a espacios interiores, dadas las limitaciones de la tecnología del dispositivo utilizado (sección 2.4.2). El sistema de partida es el prototipo existente en el grupo VPU-Lab. La figura 2.1 es un esquema de las etapas de funcionamiento de dicho prototipo.

Una vez obtenidos los *frames* de color y profundidad proporcionadas por el sensor Kinect (sección 4.2), se identifican las etapas donde es útil aprovechar las bondades de la información de profundidad. En nuestro caso, éstas son las etapas de Detección del primer plano y Discriminación de objetos.

En una primera aproximación, se realiza una segmentación basada únicamente en imágenes de profundidad (sección 4.3), se estudian sus deficiencias, se analizan sus causas y se propone una solución. Esta segmentación da como resultado una máscara binaria con todos o gran parte los *blobs* del frente. Como se verá más adelante, la tendencia de esta máscara es a presentar un mayor número de falsos negativos que positivos (es decir, no sobresegmenta).

En una segunda aproximación se combinan las máscaras de *foreground* de color y profundidad para obtener otra, cuya fiabilidad es mayor (sección 4.4). El objetivo de esta máscara compuesta es integrar, en la mayor medida de lo posible, toda la información real de cada una de las máscaras individuales.

Finalmente, para la última etapa, Discriminación de objetos, la información de profundidad se utiliza de manera complementaria a las técnicas basadas en información de color (sección 4.5). La idea es realizar una primera comprobación utilizando información de profundidad. Si la decisión con este proceso de discriminación no es lo suficientemente determinante, se pasa a realizar una segunda comprobación con los métodos tradicionales

basados en color.

El resto de módulos, aunque no directamente, también se benefician de la información de profundidad (como se ve en los resultados de la sección 5.4.3), pues parten de la máscara de *foreground* obtenida en la primera etapa. La figura 4.1 ilustra los módulos donde la información de profundidad se incluye de forma directa.

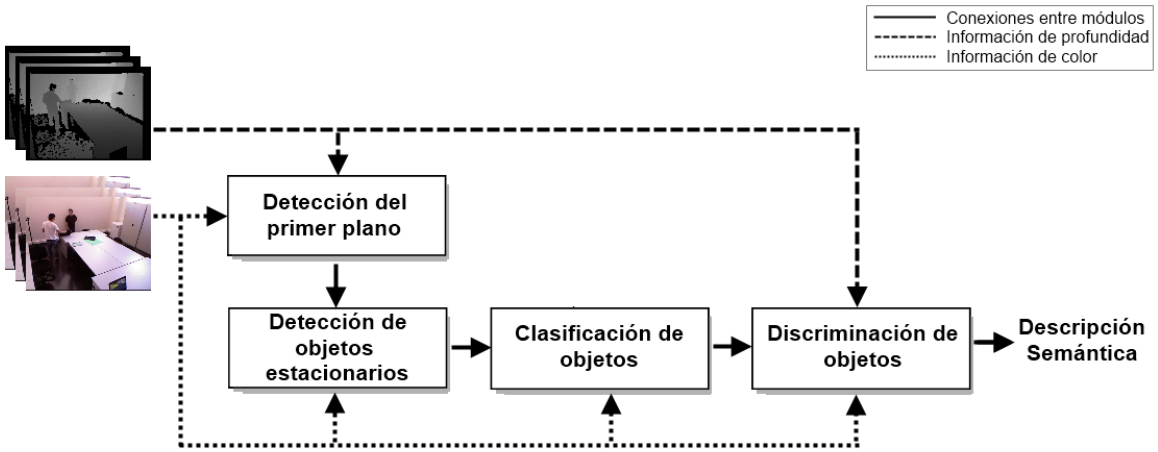



Figura 4.1: Sistema inicial modificado

## 4.2. Adquisición de *frames* de color y profundidad

Para obtener los *frames* de color y profundidad es necesario tener instalado el *software* adecuado para poder manipular los datos entregados por el Sensor Kinect. La instalación de dicha herramienta y su integración con otras orientadas al tratamiento de imágenes se explican detalladamente en el apéndice A. En este caso, de los SDKs disponibles para manejar el dispositivo (ver sección 2.4.3), se ha optado por el entorno de desarrollo OpenNI, cuyas prestaciones se ajustan bastante bien a las necesidades surgidas en el desarrollo de este proyecto. Asimismo, para el tratamiento de las imágenes se utilizan las librerías proporcionadas por OpenCV[1]. Su instalación e integración con el entorno elegido se explican en el apéndice B.

## 4.3. Detección del primer plano con profundidad

En la extracción del primer plano se separaran todos aquellos elementos de una imagen ($I$) que no aparecen en otra de referencia ($B$), considerada como el modelo de fondo.

[1]OpenCV (Open Source Computer Vision): `http://opencv.willowgarage.com/wiki/`

El procedimiento seguido para modelar el fondo en imágenes de profundidad utiliza la misma filosofía que el utilizado en imágenes de color (sección 4.3). No obstante, debido a las limitaciones tecnológicas del sensor Kinect (sección 2.4.2), es necesario considerar las imperfecciones que éstas ocasionan sobre las imágenes de profundidad. La subsección siguiente trata en detalle el proceso seguido para obtener el modelo de fondo, así como las estrategias empleadas para compensar las imperfecciones mencionadas anteriormente.

### 4.3.1. Obtención del modelo de fondo

El modelo de fondo (o *background*) es, en este caso, una imagen de profundidad que refleja la distancia inicial de los elementos de la escena bajo análisis. Este modelo se puede elegir como un *frame* de profundidad cualesquiera o como la contribución de muchas de ellas por medio de una determinada operación, como puede ser una media aritmética de un conjunto de ellas.

Una particularidad muy común en las imágenes de profundidad proporcionadas por el sensor Kinect es la presencia de regiones "intermitentes"; es decir, zonas en las que la profundidad no se detecta de forma permanente, sino que en determinados momentos no se puede estimar. Esto ocurre así para áreas de la escena donde las propiedades o disposición de la superficie de los objetos condicionan una interpretación correcta del patrón de luz reflejado hacia el sensor de IR (ver sección 2.4.2). Esto hace que la elección de un solo *frame* en un determinado instante como modelo de fondo no sea una opción válida, debido a la aparición de muchas regiones donde la profundidad es indeterminada, como se puede apreciar en la figura 4.2. La regiones de color blanco y verde en las imágenes de profundidad y color, respectivamente, se corresponden con zonas donde no se pudo estimar la profundidad en un instante dado

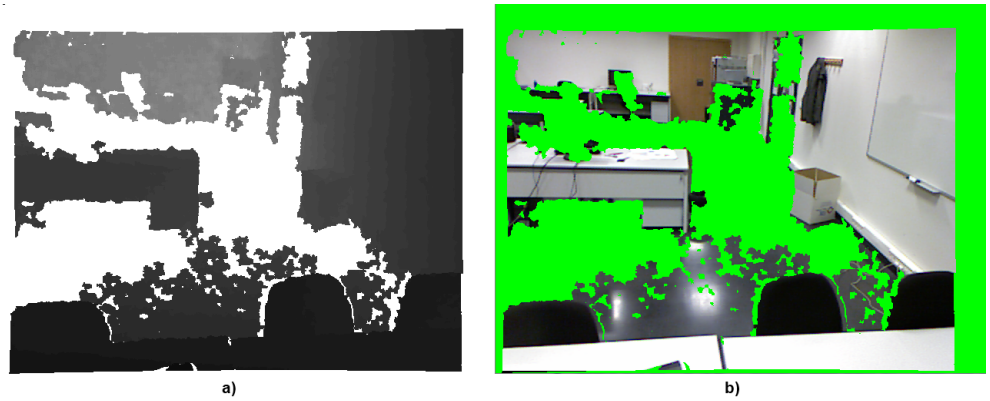



Figura 4.2: Imagen de profundidad capturada en un instante concreto. a) Imagen de profundidad. b) Imagen de color. En blanco y verde, las regiones de profundidad nula, respectivamente.

La profundidad de algunas de estas regiones no se puede calcular definitivamente, pero hay otras en las que para ciertos instantes de tiempo se da un valor de distancia válido. Así pues, cada *frame* consecutivo será distinto de la anterior, dado que hay valores de profundidad que fluctúan entre un valor válido y otro indefinido. Este comportamiento puede ser aprovechado para intentar obtener el mayor número de píxeles de profundidad válidos. Tal aprovechamiento se consigue almacenando $N$ *frames* de forma consecutiva y posteriormente aplicando un operador sobre ellas. Si tras $N$ *frames* consecutivos, el valor de un pixel de profundidad es siempre nulo (valor 0), entonces ese pixel es indefinido de forma permanente. Un valor de $N$ en torno a 50 es suficiente para determinar la validez de un pixel de profundidad. En la figura 4.3 se muestra el fondo de una escena, donde cada pixel es el último valor válido de los últimos 50 frames ($N = 50$).

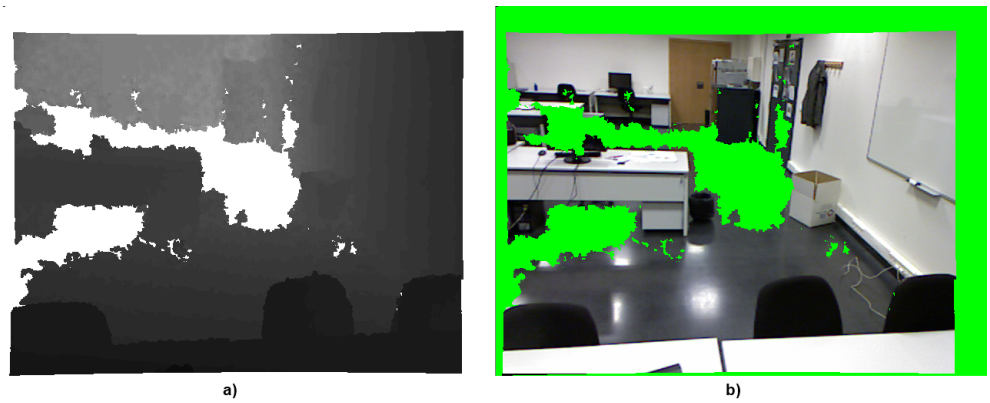



Figura 4.3: Fondo modelado con varios *frames* consecutivos. a) Imagen de profundidad. b) Imagen de color. En blanco y verde, las regiones de profundidad nula, respectivamente.

En la figura 4.3 se puede apreciar claramente (respecto a la figura 4.2) que el número de pixeles de profundidad nula es bastante menor. El criterio empleado para elegir el valor final del pixel del modelo de fondo es más acertado si se tiene en cuenta que, para los pixeles intermitentes, los valores devueltos en cada instante no son fijos, sino que siguen una distribución gaussiana (figura 4.4).

Se puede ver en el histograma de la figura 4.4 que prácticamente la mitad de las muestras tomadas se corresponden con valores nulos o indefinidos. Además, hay que tener en cuenta que mientras que hay píxeles con valores válidos siempre constantes, también los hay que van cambiando entre valores concretos transcurrido cierto tiempo, como se puede apreciar en la figura 4.5. En ambos casos (píxeles intermitentes y píxeles permanentes), la variación de los valores de profundidad se puede considerar gaussiana.

Teniendo esto en cuenta, una mejor aproximación para obtener el modelo de fondo de profundidad consiste en modelar cada pixel con una media $\mu$ y una desviación típica $\sigma$ a partir de un subconjunto de valores válidos de profundidad $\widetilde{D}_i = \left\{ i = 1, 2, 3, ...\widetilde{N} / D_i \neq 0 \right\}$

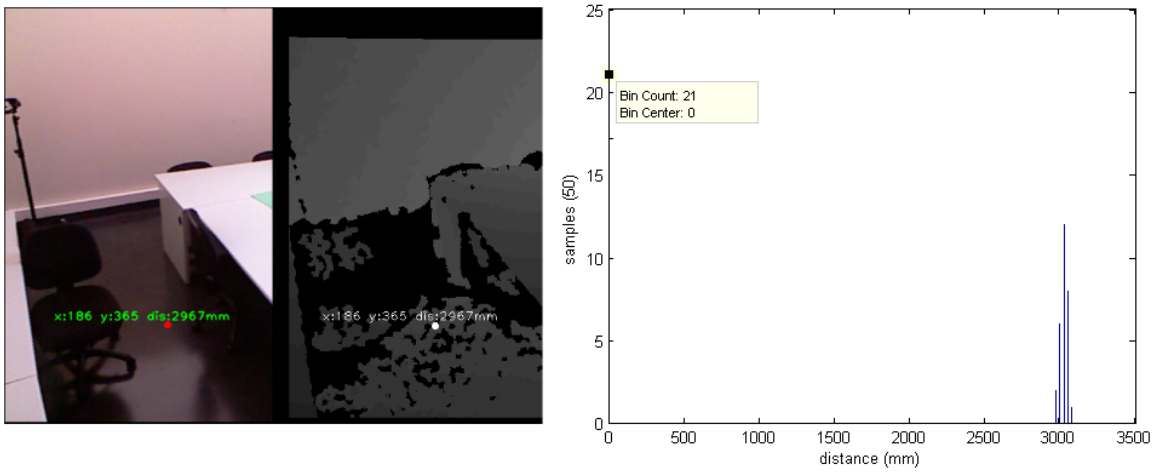



Figura 4.4: Distribución de valores de píxeles intermitentes

del conjunto total de $N$ muestras tomadas $D_i = \{i = 1, 2, 3, ...N\}$. Así, el fondo (o *background*) queda modelado de acuerdo a la expresión 4.1.

$$B(x,y) = \mu(x,y) = \begin{cases} \frac{1}{\widetilde{N}} \sum_{i=1}^{\widetilde{N}} \widetilde{D}_i(x,y) & if\ \widetilde{N} > 0 \\ 0 & otherwhise \end{cases} \tag{4.1}$$

Donde $\widetilde{N}$ es el número de muestras válidas de las $N$ muestras iniciales y $\widetilde{D}_i(x,y)$ es el valor de profundidad i-ésimo del pixel localizado en$(x,y)$.

De igual manera, la desviación típica asociada a cada pixel del modelo de fondo se calcula según la expresión 4.2.

$$\sigma(x,y) = \begin{cases} \sqrt{\frac{1}{\widetilde{N}} \sum_{i=1}^{\widetilde{N}} (\widetilde{D}_i(x,y) - \mu(x,y))^2} & if\ \widetilde{N} > 0 \\ 0 & otherwise \end{cases} \tag{4.2}$$

Si $N \neq \widetilde{N}$, entonces el valor del pixel correspondiente es fluctuante. Este hecho queda reflejado en una máscara binaria $FL$ descrita por la ecuación 4.3, la cual será utilizada en la obtención del primer plano.

$$FL = \begin{cases} 1 & if\ \widetilde{N} \neq N \wedge \widetilde{N} > 0 \\ 0 & otherwise \end{cases} \tag{4.3}$$

Nótese en la ecuación 4.3 que $\widetilde{N}$ será igual a 0 cuando el valor del pixel correspondiente sea nulo para las $N$ muestras consideradas.

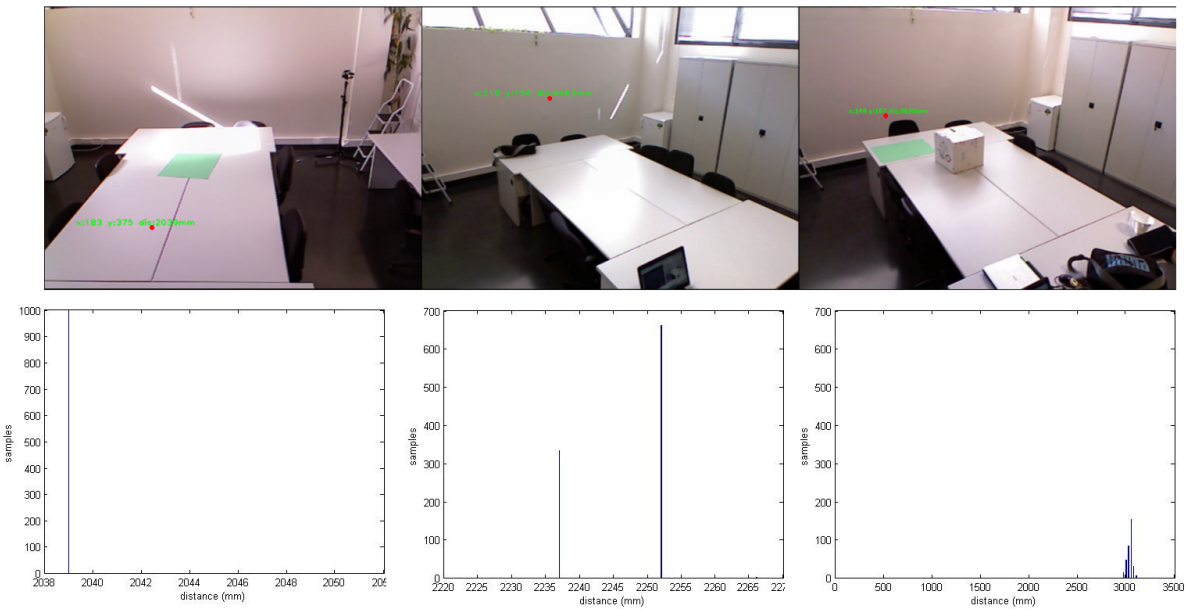



Figura 4.5: Distribución de píxeles válidos constantes (izquierda) y variables (centro y derecha) para 1000 muestras

### 4.3.2. Obtención del primer plano

Una vez obtenido el modelo de fondo, se realiza una diferenciación entre el *frame* actual $F$ y dicho modelo, caracterizado por una Gaussiana de media $\mu$ y una varianza $\sigma^2$. La umbralización de esta diferenciación produce una máscara binaria que revela justamente los nuevos elementos aparecidos o desaparecidos de la escena.

Para llevar a cabo la diferenciación hay que tener en cuenta que gran parte de los píxeles de profundidad son "fluctuantes"; es decir, que su valor oscila entre uno válido y otro que no lo es (profundidad 0), que son indicados por la máscara $FL$. Además, se sabe que estos píxeles tienen un valor fijo en el modelo fondo, que es justamente el valor medio de píxeles válidos en una secuencia sucesiva de $N$ *frames* (sección 4.3.1). De este modo, solamente se realiza la diferencia entre fondo y frente, $Diff$, cuando los píxeles del *frame* actual tienen valores válidos ($F(x,y) > 0$) o, si no lo tiene ($F(x,y) = 0$), cuando el fondo lo tiene y no corresponde a un pixel oscilante ($FL = 0$). Este último caso ocurre cuando el fondo tiene valores válidos permanentemente, pero algún elemento del frente no lo tiene, como puede ser el cabello (negro) de una persona, por ejemplo. La ecuación 4.4 resume el proceso de diferenciación utilizado.

$$Diff(x,y) = \begin{cases} ||F(x,y) - \mu(x,y)|| & if\, F(x,y) > 0 \vee (F(x,y) = 0 \wedge FL = 0) \\ 0 & otherwise \end{cases} \tag{4.4}$$

Donde $F$ es la profundidad del frame actual, $\mu$ es la profundidad media del modelo de fondo y $FL$ es una máscara descrita por la ecuación 4.3, que representa los pixeles fluctuantes ($FL = 1$) y los que no lo son ($FL = 0$).

Un pixel igual a cero en $Diff$ no significa que en esa región no haya frente, más bien se trata de un pixel de valor "indeterminado" que puede ocurrir en las siguientes situaciones:

- Fondo y frente con pixel válido ($\mu(x,y) = F(x,y)$). Este caso tampoco se puede considerar fiable en toda regla, ya que puede tratarse de un objeto plano; es decir, que su distancia respecto a la del fondo sea muy pequeña. En esta situación se tendría un falso negativo.

- Fondo y frente con pixel no válido ($\mu(x,y) = 0 \wedge F(x,y) = 0$). Esta situación se da cuando en el modelo de fondo existe una región con valores de profundidad nulos constantemente (p. ej. una superficie reflectante) y sobre el pasa un objeto (o persona) con valores nulos también (p. ej. pelo negro). Esta situación en concreto, produciría falsos negativos. Sin embargo, en el caso de no haber frente alguno, los valores de $Diff$ serían correctos.

De estas dos situaciones queda claro que no se pueden producir falsos positivos. Como se verá más adelante (sección 4.3.3), esto no es del todo cierto debido a determinadas condiciones de la escena y restricciones del sensor Kinect. También se deduce que todo valor 0 en $Diff$ es propenso a pertenecer a un falso negativo.

El resultado de diferenciar un pixel del *frame* actual respecto a uno del modelo de fondo proporciona un valor que, en el caso de un fondo estable y preciso, debería ser cero para regiones donde no hay frente. Sin embargo, como ya se mencionó en la sección 4.3.1, los valores de profundidad para una misma distancia pueden ser distintos. Por ello se permite un margen de variación determinado por el parámetro $\sigma$. Dicho parámetro se multiplica por un número entero $k$ elegido de manera experimental. El número resultante es el umbral que indica la variación máxima permitida en la diferenciación anterior. Todo lo que esté por encima de este umbral corresponderá a zonas del *foreground* $FG$ y lo que no al *background*. La siguiente expresión muestra el método seguido para la obtención del primer plano.

$$FG(x,y) = Diff(x,y) > k\sigma(x,y) \tag{4.5}$$

El valor de $k$ se debe elegir teniendo en cuenta que los objetos muy cercanos al fondo

pueden no ser detectados como frente. Un valor muy pequeño haría que zonas del *background* sean detectados como *foreground*, mientras que un valor muy alto provocaría la situación inversa. En la figura 4.6 se pueden ver ilustrados estos dos casos.

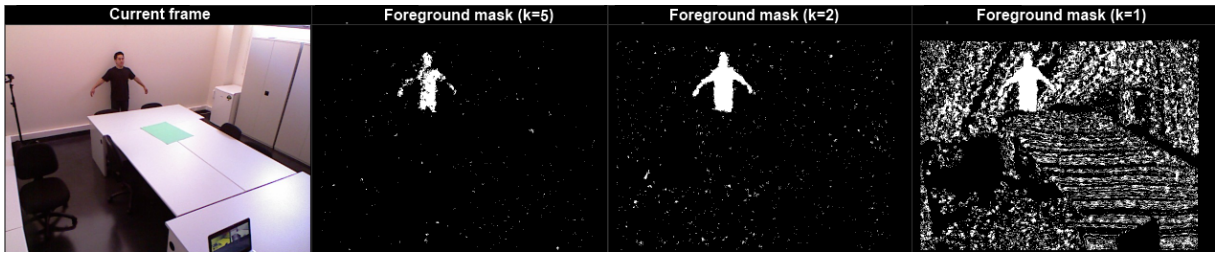



Figura 4.6: *Foreground* en función de $k$

En la figura 4.6 se observa que conforme va aumentando el valor de $k$, el ruido es menor; sin embargo, se pierden ciertas regiones del *foreground*. Un valor de $k = 2$ produce una máscara en la que gran parte de los detalles del *foreground* se conservan, donde el ruido es aceptable y puede ser eliminado con una operación morfológica. En la figura 4.7 se puede ver el resultado de la máscara del frente, tras aplicar una apertura de una iteración sobre la misma escena de la figura 4.6.


Figura 4.7: Máscara del frente con ruido filtrado tras una apertura morfológica

### 4.3.3. Actualización del modelo de fondo

Las restricciones de la tecnología utilizada por el sensor Kinect hacen que imágenes de profundidad consecutivas difieran bastante entre sí incluso cuando no hay frente. Esto significa que el modelo de fondo debe ser capaz de absorber todas esas variaciones ruidosas que pueden dar lugar a falsos positivos en el *foreground*.

Además del comportamiento de los píxeles mostrados en la figura 4.5, existen otros provocados por alteraciones en las condiciones iniciales de la escena o cambios repentinos durante el transcurso del tiempo que no fueron contemplados a la hora de modelar el fondo.

El primer caso se produce cuando hay un cambio muy fuerte de luminosidad (luz solar) que impide que la luz infrarroja pueda ser detectada por el Sensor IR, y el segundo tiene que ver con la propia tecnología del sensor Kinect en la estimación de la profundidad.

Un cambio de luminosidad intenso como la luz solar, hace que los píxeles de profundidad de algunas regiones sean nulas. Cuando esta luminosidad desaparece de la escena aparece entonces un valor de profundidad válido que se deberá tener en cuenta para no confundirlo con un posible pixel del *foreground*. En la figura 4.8 se ilustra esta situación.

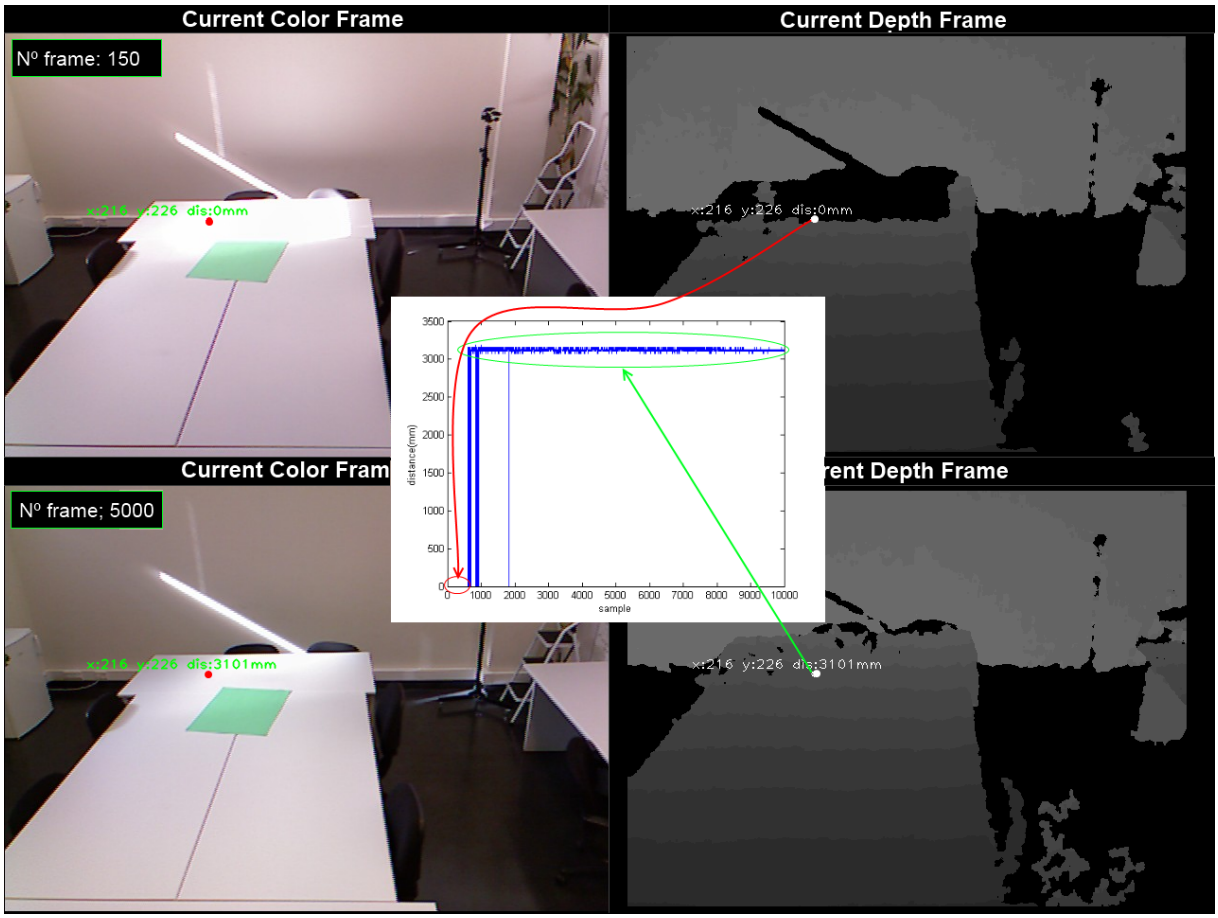



Figura 4.8: Efecto de la luz solar en la detección de profundidad

Para intentar solucionar este problema se procede de la siguiente manera. En el instante que un pixel del *foreground* cambia de estado (de 1 a 0 ó viceversa) se empieza a contar el número de *frames* en los que el pixel permanece consecutivamente en ese nuevo estado. Si el contador supera un umbral $\tau_c$, entonces se comprueba el valor de sus 8 vecinos en el modelo de fondo (figura 4.9). Si todos los vecinos son nulos, entonces no se actualiza el modelo en ese píxel. Si existe al menos un vecino con valor válido, entonces se crea una máscara binaria, $Diff_{D\_binary}$, que decide si el pixel debe incorporarse al fondo ($Diff_{D\_binary} \geqslant 1$) o

no. La ecuación 4.6 describe como se obtiene la máscara $Diff_{D\_binary}$.

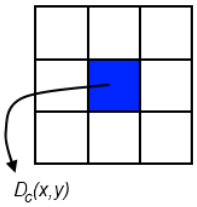



Figura 4.9: Vecindad considerada para un pixel afectado por la luz solar.

$$Diff_{D\_binary}(x,y) = \sum_m \sum_n |D_c(x,y) - \tilde{\mu}(m,n)| > k\sigma(m,n) \tag{4.6}$$

Donde $D_c(x,y)$ es la profundidad de pixel bajo análisis, $\tilde{\mu}(m,n)$ es un pixel válido (profundidad no nula) de la vecindad de $D_c(x,y)$ en el modelo de fondo $\mu$, y $k\sigma(m,n)$ es un parámetro de umbralización igual al utilizado en la ecuación 4.5.

La figura 4.10 muestra el resultado del método propuesto para la correción del problema de la luz solar.

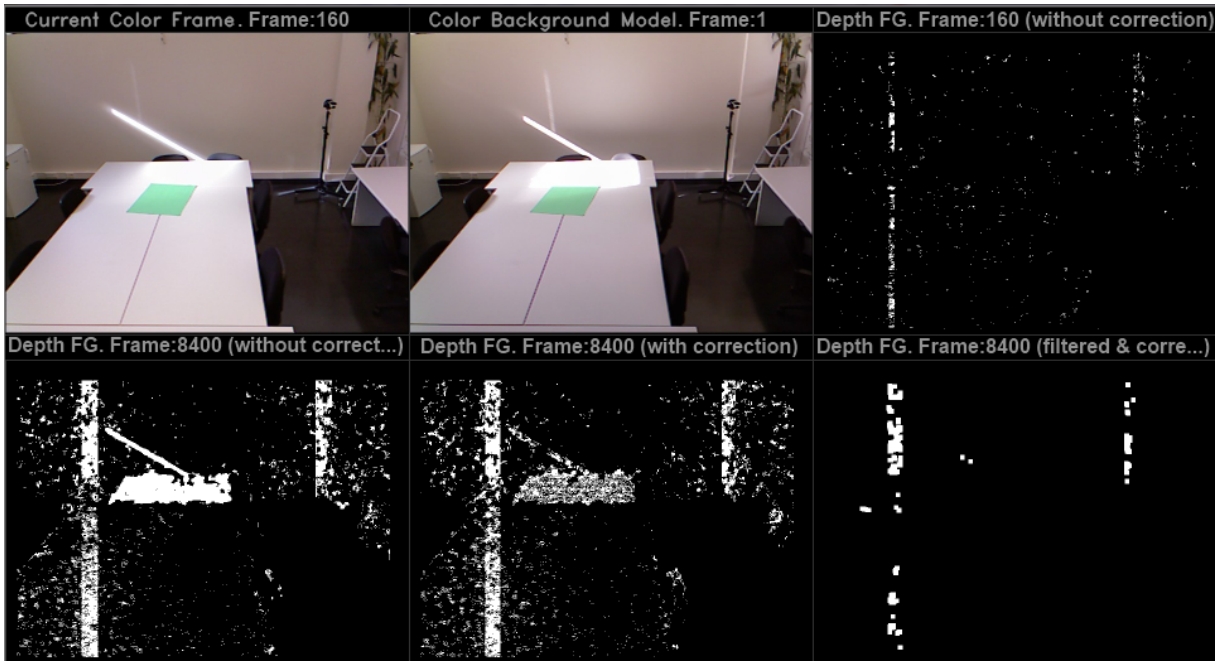



Figura 4.10: Corrección del *foreground* afectado por la luz solar

El segundo mayor problema ocurre cuando después de haber obtenido el *background*, una gran cantidad de píxeles adyacentes cambian de valor considerablemente sin haber

frente alguno. Este problema se manifiesta con la aparición de *blobs* extensos, pero poco densos (en forma de nube), o a través de franjas "blancas" verticales en la máscara de *foreground* (figura 4.11). Este problema debe corregirse, ya que las franjas pueden llegar a alcanzar una densidad mayor formando *blobs* indeseados de un tamaño considerable.

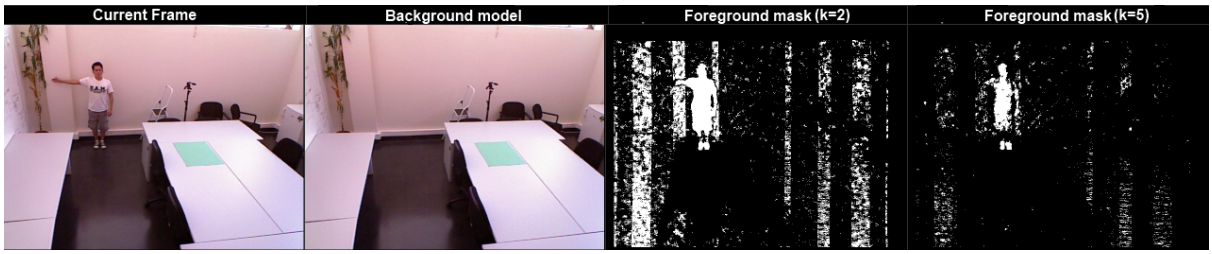



Figura 4.11: Franjas en el *foreground* de profundidad

En la figura 4.11 se muestra el efecto de la franjas para valores de $k = 2$ y $k = 5$. En el segundo caso se consigue apaciguar este efecto, pero a cambio se pierden gran parte de los detalles de los *blobs* del *foreground*. En el primer caso (*k=2*), el *blob* de interés aparece prácticamente entero; sin embargo, el efecto de las franjas es más acentuado. Con todo, se puede ver que es más conveniente encontrar un método para palear este efecto en el primer caso. El procedimiento seguido para corregir este problema se explica en el párrafo siguiente.

El método seguido para actualizar el fondo empieza por detectar las regiones donde suceden las franjas o "*blobs* ruidosos" (Q en la figura 4.12). Para ello, solamente se consideran los píxeles semiestáticos del *foreground* que se mantienen activos durante *m* (*m*=5 en nuestro caso) *frames* consecutivos, lo que produce la máscara $FGS_i(x, y)$, cuya obtención se resume en la ecuación 4.7.

$$FGS_i(x, y) = FG_i(x, y) \times FG_{i-1}(x, y) \times FG_{i-2}(x, y) \times ... \times FG_{i-m}(x, y) \tag{4.7}$$

Donde $FG_i(x, y)$ es la máscara de *foreground* de profundidad i-ésima.

Posteriormente, se detectan los *blobs* grandes (R en la figura 4.12), los que corresponden a personas u objetos presentes en la escena, y se retiran de la máscara del *foreground*. La detección de estos *blobs* se realiza mediante un filtrado por reconstrucción y se obtiene la máscara $R$ (ecuación 4.8).

$$R = \gamma^{rec}(FG_i(x, y)\,;\, FG_i^{erosion}(x, y)) \tag{4.8}$$

Donde $FG_i(x, y)$ es la máscara de *foreground* i-ésima y es la imagen donde se buscarán las componentes conexas a partir de la imagen marcador $FG_i^{erosion}(x, y)$. La imagen marcador se obtiene aplicando una fuerte erosión (p. ej. 3 iteraciones con *kernel* cuadrado de 5x5)

sobre $FG_i(x, y)$ a fin de conservar solo parte de los *blobs* grandes. Esta operación además descarta los *blobs* dispersos pertenecientes a las franjas.

La imagen binaria (S en la figura 4.12) que se utiliza para analizar las franjas se obtiene de acuerdo a la ecuación 4.9. Esta máscara contiene solamente los píxeles correspondientes a las franjas.

$$S = \overline{R} \times FGS_i(x, y) \tag{4.9}$$

La máscara $S$ se divide a su vez en $N$ franjas verticales $C_j(x, y)/j\epsilon\{1, 2, 3...N\}$ de dimensiones $\frac{W}{N}xH$, donde $W$ y $H$ son las dimensiones de $S$. Para cada franja se calcula el porcentaje $p$ de píxeles respecto al total considerado. Esta operación se realiza según la ecuación 4.10.

$$p_j = \frac{\sum_{W'} \sum_{H} C_j(x, y)}{W'xH}, C_j(x, y) = 1 \tag{4.10}$$

Donde $H$ es la altura y $W' = \frac{W}{N}xH$ el ancho de la franja.

Si cualquiera de los $p_j$ supera un umbral $\tau_{act}$ (presencia de franja) entonces se procede a la actualización del fondo. Para ello, se empiezan a almacenar *n frames* de profundidad consecutivos, los que se utilizarán para la actualización, y se inicializa una máscara de movimiento $M$ que marcará la trayectoria de los *blobs* grandes durante la adquisición de los *n frames*. Una vez adquiridos los *frames* se actualiza todo el fondo, excepto las regiones que corresponden a píxeles activos de la máscara $M$. La actualización se realiza siguiendo el procedimiento descrito en la sección 4.3.1. La figura 4.12 ilustra las máscaras involucradas en todo este proceso.

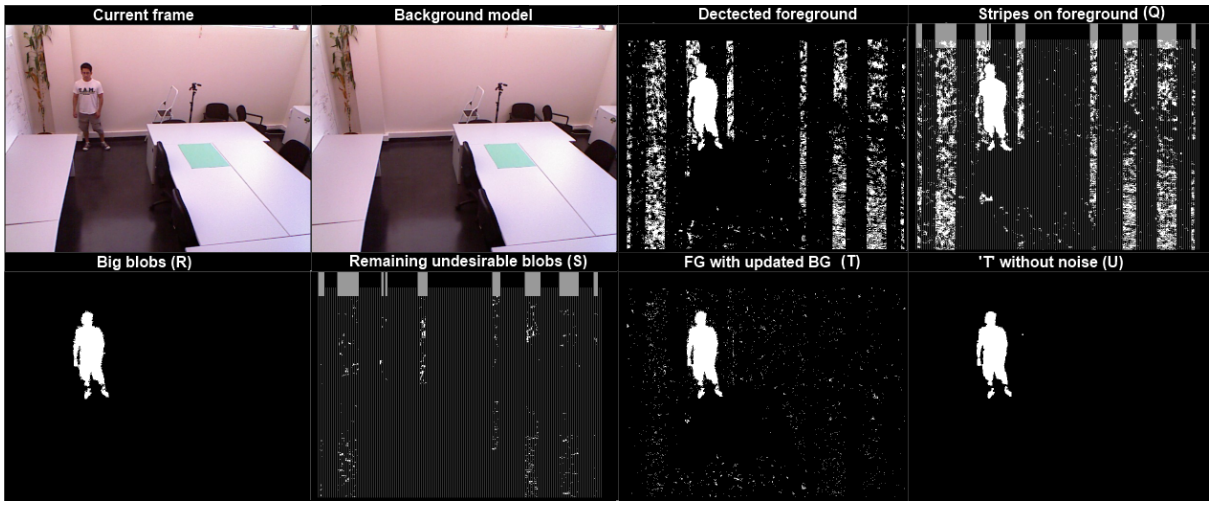



Figura 4.12: Actualización del modelo de fondo

Se actualiza todo el fondo y no solo el área correspondiente a la franja, debido a que pueden existir otras franjas, menos marcadas, que no hayan superado el umbral $\tau_{act}$.

En este caso, si alguno de los *blobs* de la máscara $S$ perteneciese a un objeto pequeño, éste sería también incorporado al fondo, excepto cuando la diferencia de su profundidad con respecto a la del fondo sea bastante elevada, en cuyo caso, el *blob* se añade a la máscara $M$. Sin embargo, de acuerdo al esquema seguido para la obtención de la máscara final (sección 4.4), esto no supone un problema grave, pues los objetos pequeños serán considerados utilizando las imágenes de color.

El problema del párrafo anterior se podría evitar teniendo en cuenta la información de color y las dimensiones del *blob*. Una aproximación bastante fiable consistiría en realizar una diferenciación absoluta de cada uno de los canales HSV entre fondo y frente en las imágenes de color y comprobar que difieren lo suficiente como para que en esa región haya aparecido o desaparecido un objeto. Además, se podrían filtrar los *blobs* atendiendo a sus dimensiones.

Las imágenes $T$ y $U$ de la figura 4.12 representan el *foreground* tras haber actualizado el fondo, con y sin el ruido filtrado, respectivamente. Las imágenes $R$ y $U$ son prácticamente iguales, con la única diferencia de que en $U$, los contornos de los *blobs* están mejor definidos, ya que con el filtrado se elimina el posible ruido que pudo pasar a $R$ en la operación descrita en la ecuación 4.8.

## 4.4. Combinación de máscaras de *foreground* de color y profundidad

Hay que decir que, tal como dice el título de este PFC, las escenas que se muestran a continuación (y las mostradas anteriormente) han sido grabadas en interiores, donde además, la distancia respecto al sensor Kinect es razonable, de modo que éste sea capaz detectar su profundidad (~7m). Asímismo, las imágenes de color y profundidad están alineadas mediante las herramientas que proporciona el entorno de desarrollo elegido (OpenNI). En la figura 4.13 se ilustra una escena en las condiciones antes mencionadas.

De la escena se obtienen las máscaras del *foreground* tanto para la información de color como para la de profundidad. En las máscaras de las primeras dos filas, se puede observar que la de profundidad prácticamente refleja todos los objetos "grandes" (con profundidad) y sin los problemas de sombras o camuflajes, que sí ocurren en la máscara de color. Asimismo, en las dos últimas máscaras, se ve claramente como la máscara de profundidad es inmune a los cambios de iluminación (encendido de luces). No obstante, los objetos con poca profundidad (bolsa de portátil y carpeta) no se reflejan en la máscara de profundidad.

Una primera aproximación para combinar las dos máscaras (similar a la utilizada en

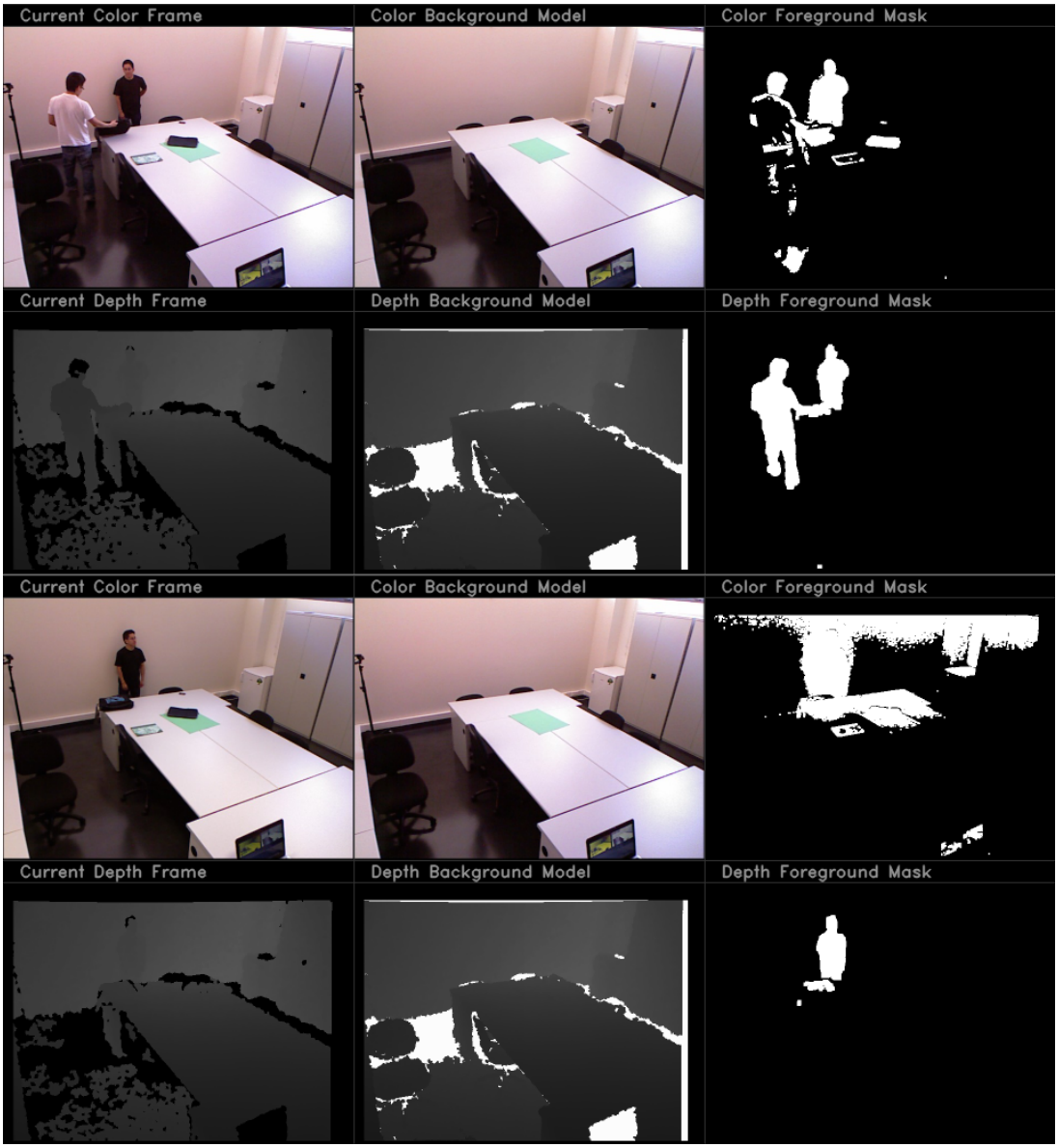



Figura 4.13: Máscaras de *foreground* de color y profundidad.

[60]) consiste en realizar una operación binaria OR entre ambas. Este método es bastante sencillo, pero solamente corrige uno de los problemas típicos y es el camuflaje (figura 4.14). Los problemas provenientes de la máscara de color, como son las sombras y cambios de iluminación siguen persistiendo.

Antes de fusionar ambas máscaras, se puede aplicar previamente un preprocesado a la máscara de color para intentar eliminar las sombras. Sin embargo, esto supone una carga computacional adicional, lo que no es apropiado para un sistema que ha de trabajar en tiempo real. Además, los resultados de dicho preprocesado tienen una efectividad

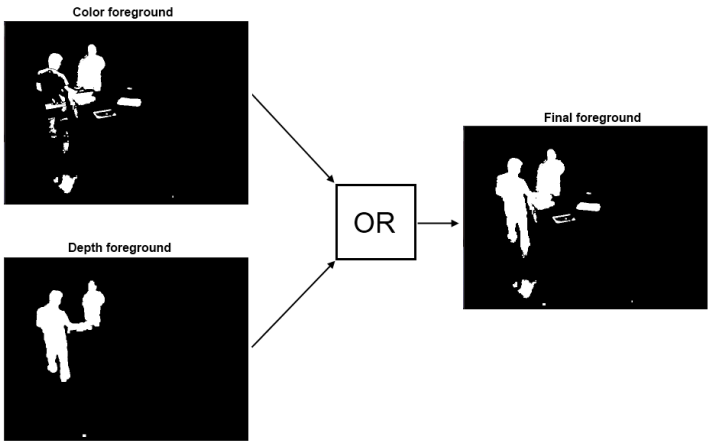



Figura 4.14: OR lógico entre el *foreground* de color y profundidad

relativamente aceptable: las sombras muy marcadas no serán filtradas y aquellos objetos cromáticamente similares al fondo sí lo serán. Se dispone de información adicional procedente de la máscara de profundidad, con lo que un procesado basado en color no parece lo más acertado, puesto que se intenta reducir procesos, no incrementarlos.

La aproximación que se explica a continuación es por la que se ha optado por ser la que mejores resultados proporciona al sistema global. No obstante, eso no significa que el procedimiento seguido no sea proclive a errores, cuya naturaleza también se explica.

Observando la figura 4.13, parece más razonable trasladar los elementos de la máscara de color a la máscara de profundidad que no aparecen en ésta. Es decir, en este caso en concreto (figura 4.13), se trataría de incluir en la máscara de profundidad los *blobs* correspondientes a objetos de muy poca profundidad (considerando la resolución del sensor) tales como la carpeta, la bolsa del portátil y el teléfono móvil. La inclusión directa de todos los *blobs* ausentes conduciría al mismo problema reflejado en la figura 4.14, ya que se estarían añadiendo también los *blobs* correspondientes a las sombras o interreflexiones. Este problema se aplaca teniendo en cuenta dos asunciones relacionadas con la profundidad de los objetos y los eventos que el sistema está destinado a detectar. La primera de ellas considera que todas las personas y objetos grandes de la escena serán detectadas por la máscara de profundidad y la segunda asume que si un *blob* no aparece en la máscara de profundidad, pero sí lo hace en la máscara de color, es porque no tiene profundidad y por tanto, tampoco sombras. Así, solamente se añaden a la máscara de profundidad los *blobs* de objetos pequeños y estáticos que aparecen en la máscara de color. Estas consideraciones dan como resultado una máscara compuesta (color + profundidad) donde todos los *blobs* corresponden a objetos o personas sin sombras. La figura 4.15 resume esquemáticamente el procedimiento seguido.

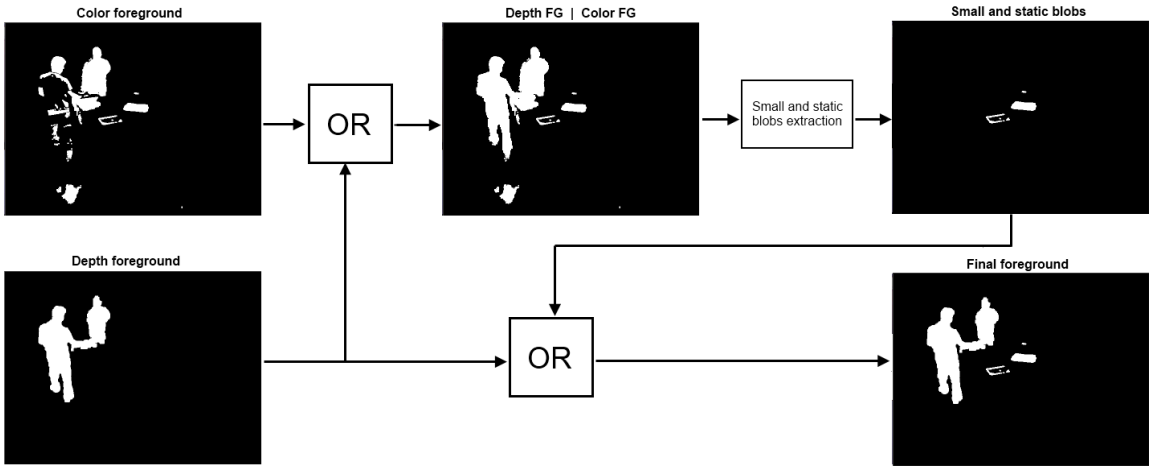



Figura 4.15: Combinación del *foreground* de color y de profundidad

Las imagen en color correspondientes a la escena mostrada en el esquema 4.15 es la que aparece en la primera fila de la figura 4.13. Como se puede ver en la figura 4.15, al extractor de objetos pequeños y estáticos entra la máscara resultado del OR lógico entre las máscaras de *foreground* de color y de profundidad. Esta operación evita que se considere y analice cada *blob* perteneciente a un mismo objeto, lo que se traduce en un ahorro de coste computacional. En este caso en concreto, se evita analizar cada *blob* de la imagen *"Color foreground"*, que han sido originados por un problema de camuflaje.

El módulo extractor de objetos pequeños y estáticos tiene en cuenta el *bouding box* del *blob* correspondiente. Si la posición del *bounding box* se mantiene fija durante $n$ frames consecutivos, entonces el *blob* se considera estático. De igual manera, si el área del *blob* considerado como estático está por debajo de un umbral $\tau_{small}$, entonces se considera pequeño. El *blob* que cumpla con estas dos condiciones será el que entregue el módulo "*small and static blobs extraction* ".

En el *foreground* final, las sombras asociadas a las personas se desechan, a menos que ésta aparezca fraccionada (en *blobs* pequeños) y sea estática. Por ejemplo, en la imagen de la esquina superior izquierda de la figura 4.15, la sombra que aparece en el suelo (perteneciente a la primera persona) pasaría el filtro "*small and static blobs extraction* " si la persona asociada permaneciese sin movimiento alguno. Si hubiese un cambio de iluminación fuerte que produjese *blobs* que se solapen con los de los objetos pequeños (en la máscara de color), entonces éstos también se descartarían. Otra de las desventajas que tiene el método propuesto ocurre con objetos planos extensos, como puede ser un mantel. En este caso, el *blob* correspondiente sería detectado por la máscara de color, pero sería descartado en la máscara final por no ser pequeño.

## 4.5. Discriminación de objetos

La otra contribución realizada sobre el sistema global tiene lugar en esta etapa. Hay que mencionar, que no se ha hecho ninguna modificación interna del módulo existente en el sistema prototipo. Más bien lo que se ha hecho es crear un módulo adicional, muy rápido en sus operaciones, pero cuyo funcionamiento depende enteramente de la profundidad de los objetos. Este nuevo módulo tiene una precisión altamente fiable y su funcionamiento se basa en la siguiente premisa: si la distancia media del objeto correspondiente al *blob* bajo análisis es menor que la distancia del fondo en esa misma región, entonces el evento corresponde a un robo; en caso contrario, se trata de un abandono, tal como se ilustra en la figura 4.16. Si el valor de profundidad está por debajo de $\mu - k\sigma$, entonces el pixel se contabiliza como el perteneciente a un objeto abandonado; si está por encima de $\mu + k\sigma$, como el perteneciente a un objeto robado. Si el valor está entre $\mu - k\sigma$ y $\mu + k\sigma$, entonces el pixel correspondiente es considerado parte de fondo.

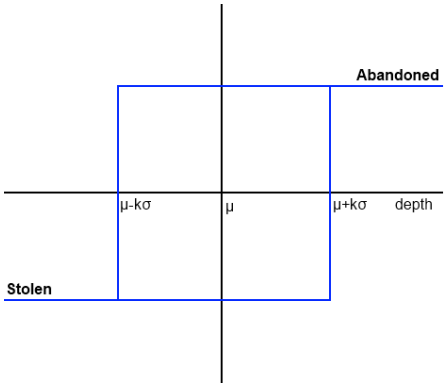



Figura 4.16: Condición de robo o abandono

La comparación entre distancias implicada en el proceso de detección del tipo de evento no se puede realizar con un solo pixel elegido arbitrariamente del *blob*, debido a la falta de estabilidad que presentan algunos de ellos. Dicha inestabilidad suele presentarse sobre todo en los bordes del *blob,* donde los píxeles de profundidad tienden a ser intermitentes; es decir, que algunos píxeles del *blob,* algunas veces aparecen como activos y otras no. Concretamente, lo que se hace es considerar todos los píxeles del *blob* ($PX_{active}$) que se quiere analizar . Se obtiene la distancia de cada uno de ellos y se ve cuántos de ellos están por encima ($PX_{up}$) o por debajo ($PX_{down}$) de un valor, que es la distancia al fondo con su respectiva varianza (figura 4.16). La decisión final se toma en función una puntuación ($score$), que se calcula según la ecuación 4.11.

$$score = \frac{PX_{down}}{PX_{active}} \tag{4.11}$$

Donde $PX_{down}$ es la cantidad de píxeles que están por debajo de $(\mu - k\sigma)$ y $PX_{active}$ es la cantidad de píxeles activos del *blob* bajo análisis. Si el *score* es próximo a cero, entonces el objeto ha sido retirado de la escena (robado), de lo contrario, si es próximo a uno, el objeto ha sido abandonado. En la figura 4.17 se ilustran ambos casos.

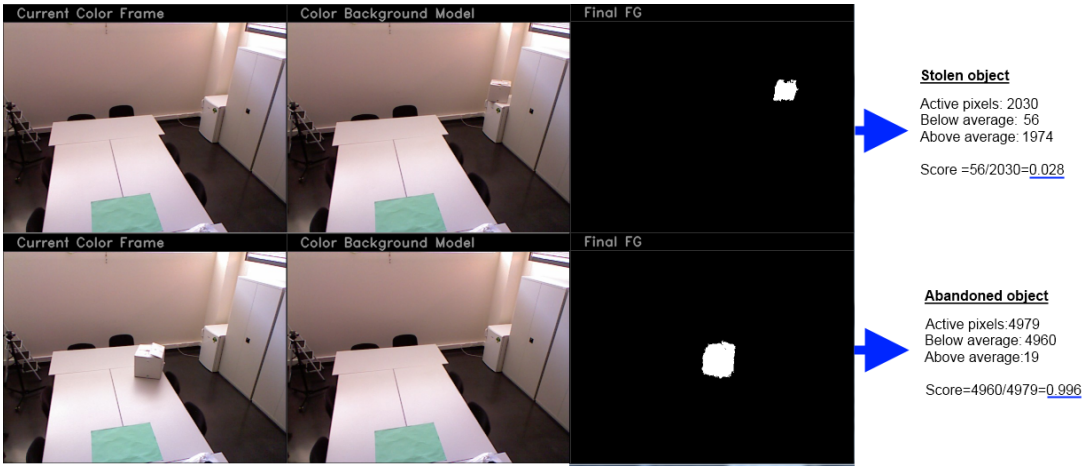



Figura 4.17: Objeto robado (arriba) y objeto abandonado (abajo)

Por último, como ya se dijo, este nuevo módulo trabaja de manera alterna con el módulo ya existente. El módulo nuevo entra en funcionamiento solamente para aquellos *blobs* de la máscara final que también son detectados por la máscara de *foreground* de profundidad. Además, el análisis se realiza sobre el *blob* de la máscara de profundidad, ya que éste tiene en cuenta el margen de error (varianza) de la profundidad del fondo. Este hecho hace que un *blob* en la máscara de profundidad sea más pequeño que el objeto real al que corresponde, pues los bordes del éste, al estar cerca de la superficie del fondo, pueden también confundirse con el fondo. En el caso de que el *blob* solamente esté presente en la máscara de *foreground* de color, entonces el análisis que se realiza para la detección del evento es el que ya se utiliza por el sistema prototipo en esta etapa (sección 3.5). El esquema de la figura 4.18 ilustra la conmutación que se produce entre un módulo y otro.

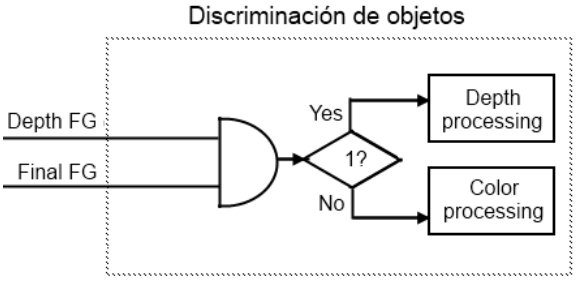



Figura 4.18: Conmutación entre procesamiento basado en color o profundidad

# Capítulo 5

# Experimentos

## 5.1. Introducción

En este capítulo se describen los experimentos llevados a cabo para comprobar la efectividad de las mejoras añadidas (capítulo 4) al sistema inicial (capítulo 3). Para realizar las pruebas se han creado una serie secuencias de video (sección 5.2), tanto en color como en profundidad, donde se reflejan situaciones desfavorables que podrían complicar el funcionamiento del sistema inicial. El sistema se prueba con y sin las mejoras, a fin de obtener una comparativa visual y cuantitativa (sección 5.4), en función de los parámetros que se describen en la sección 5.3.

## 5.2. *Dataset*

El set de datos (secuencias de video) que se presenta en esta sección ha sido obtenido por medio del sensor Kinect de Microsoft (sección 2.4). Las secuencias se almacenan en formato ONI, un formato propio de OpenNI [72], que integra la información de color y profundidad en un mismo fichero (con la extensión *.oni*). Los escenarios elegidos son interiores e intentan reflejar todo tipo de situaciones que puedan entorpecer el funcionamiento de un sistema con las características del estudiado en este proyecto. En ellos se pueden analizar eventos de robo/abandono, realizar segmentaciones en diferentes condiciones, *tracking* de personas e interacciones entre las mismas. A continuación se describen cada una de las secuencias de *dataset*.

- Secuencia *stolen_box.oni*. Ésta refleja un escenario inicial (primera fila de la figura 5.1), donde las únicas limitaciones (de las descritas en la sección 2.4.2) son la reflectancia del suelo, el color negro de las sillas y las superficies cuasi-paralelas a los rayos de luz infrarroja (i.e. armario del lado derecho). El evento a detectar en esta secuen-

cia es el robo de una caja de color blanco situada en la esquina derecha del fondo (segunda fila de la figura 5.1), ésta también de color blanco.

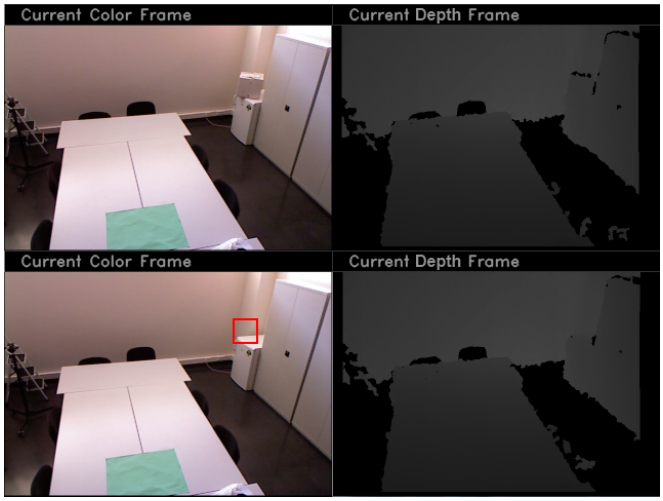



Figura 5.1: Instantáneas de la secuencia *stolen_box.oni*

- Secuencia *abandoned_box.oni*. En esta secuencia, las características de la escena inicial (primera fila de la figura 5.2) son las mismas que las de *stolen_box.oni.* El evento que se desea detectar en este caso es una caja abandonada (segunda fila de la figura 5.2). Ésta es de color blanco y se deja sobre una superficie del mismo color.

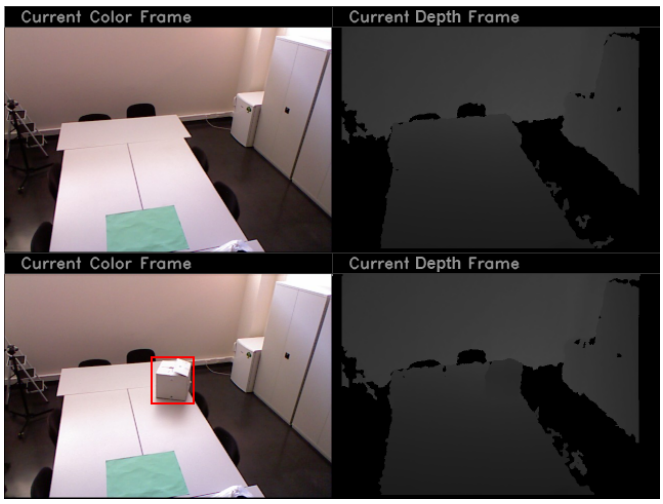



Figura 5.2: Instantáneas de la secuencia *abandoned_box.oni*

- Secuencia *meeting_room.oni.* Aquí, el escenario inicial es similar a los anteriores. En esta secuencia el objetivo es detectar el abandono de cuatro objetos: un maletín, una

carpeta, una bolsa de portátil y un teléfono móvil (figura 5.3). Además, estos dos últimos tienen una profundidad poco significativa. A lo largo de la grabación se encienden las luces para hacer variar las condiciones del entorno drásticamente. También hay una pantalla de ordenador encendida con imágenes cambiantes.

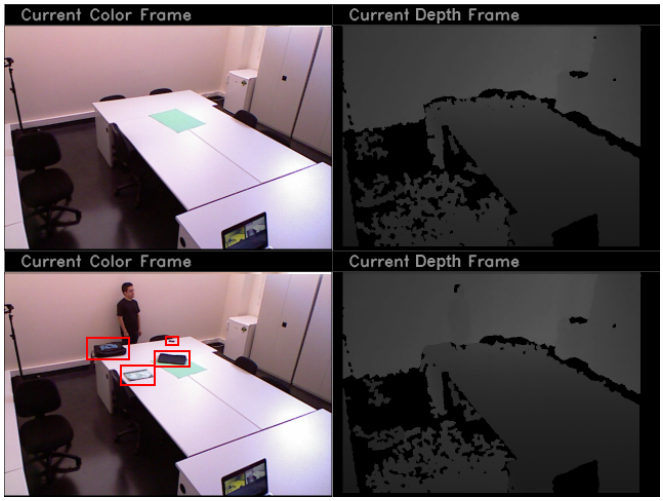



Figura 5.3: Instantáneas de la secuencia *meeting_room.oni*

- Secuencia *stolen_bin.oni*. Esta secuencia muestra un espacio interior, donde las principales dificultades para el sensor son las grandes dimensiones de dicho espacio y superficies altamente reflectantes (suelo). El objetivo es detectar el robo de una papelera de color negro situada a una distancia razonable del sensor (figura 5.4).


Figura 5.4: Instantáneas de la secuencia *stolen_bin.oni*

- Secuencia *abandoned_bin.oni*. La escena en este video es la misma que la anterior (*stolen_bin.oni*). Esta vez lo que se pretende detectar es el abandono de la misma papelera (figura 5.5).


Figura 5.5: Instantáneas de la secuencia *abandoned_bin.oni*

- Secuencia *illumination_change.oni*. En esta escena se deja un par de libros sobre una mesa (de color similar a los libros) y, posteriormente, se encienden las luces de modo que varíe la iluminación en la escena.

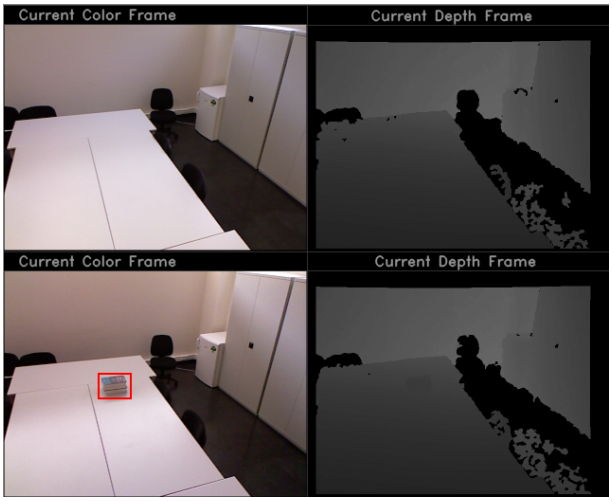



Figura 5.6: Instantánea de la secuencia *illumination_change.oni*

- Secuencia *interaction.oni*. Esta secuencia no refleja ningún evento de robo o abandono. Su objetivo es mostrar las posibilidades que ofrece el sensor Kinect para el

análisis de interacciones entre personas en espacios relativamente amplios y con condiciones desfavorables. En el video se puede observar dos personas interactuando a diferentes distancias del sensor. Como se puede apreciar en las imágenes extraídas del video en la figura 5.7, cuanto mayor es la distancia al sensor, menores son los detalles de las personas detectadas.

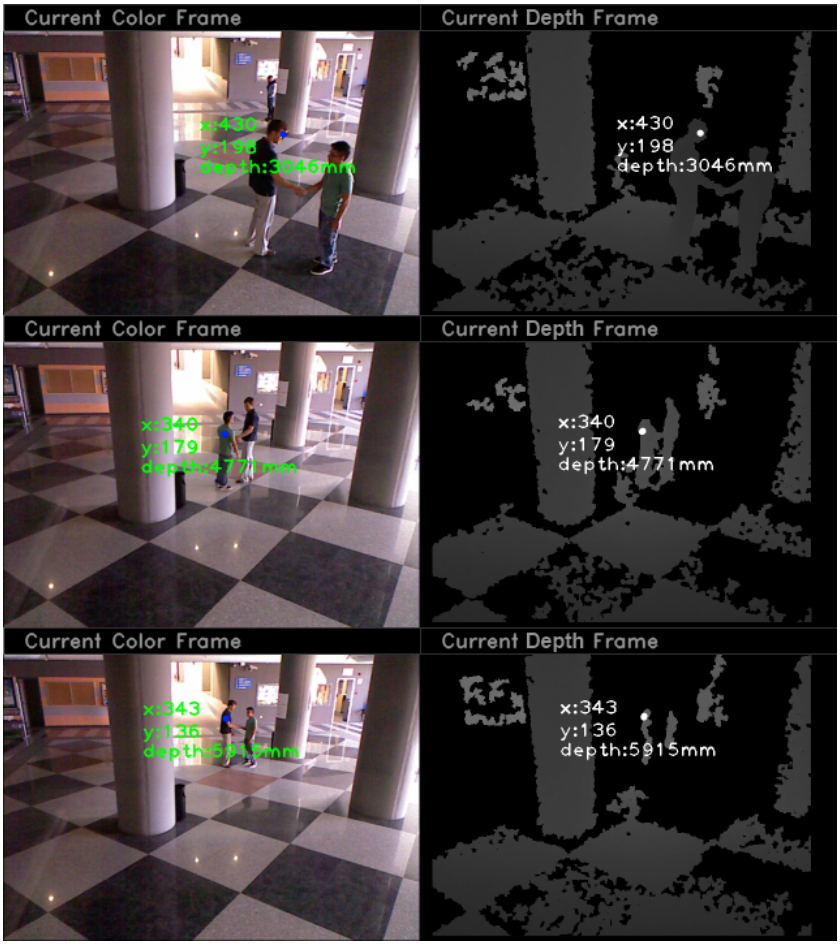



Figura 5.7: Instantáneas de la secuencia *interaction.oni*

- Secuencia *entrance.oni*. El propósito de esta secuencia es el mismo que la anterior: mostrar las posibilidades de la Kinect en otras aplicaciones. En este caso se trata del conteo de personas que acceden y salen de un recinto (figura 5.8).


Figura 5.8: Instantáneas de la secuencia *entrance.oni*

- Secuencia *sunshine.oni*. Igualmente, en esta secuencia no existe ningún evento. El objetivo aquí es observar la estabilidad de las imágenes de profundidad en el tiempo (~5 minutos). Además, en el video existe un cambio gradual de iluminación provocado por una intensa luz solar, lo que permite estudiar su efecto sobre las imágenes de profundidad (figura 5.9).

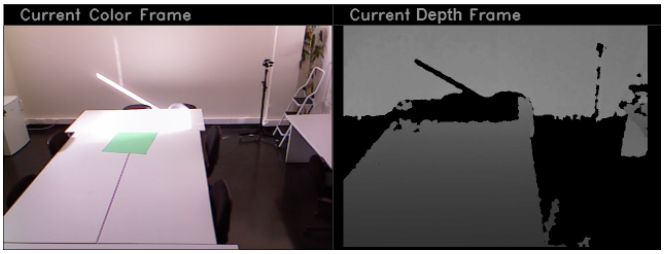



Figura 5.9: Instantánea de la secuencia *sunshine.oni*

La siguiente tabla resume las características más relevantes de los eventos mostrados en cada una de las secuencias descritas anteriormente.

| **Secuencia** | **Nº abandonos** | **Nº robos** | **Nº *Frames*** | **Descripción** | **Problemas en color** | **Problemas en profundidad** |
|---|---|---|---|---|---|---|
| *stolen_box.oni* | 0 | 1 | 914 | Robo de una caja | Camuflaje y sombras | Ninguno |
| *abandoned_box.oni* | 1 | 0 | 776 | Abandono de una caja | Camuflaje y sombras | Ninguno |
| *stolen_bin.oni* | 0 | 1 | 1120 | Robo de una papelera | Camuflaje y sombras | Camuflaje |
| *abandoned _bin.oni* | 1 | 0 | 767 | Abandono de una papelera | Camuflaje y sombras | Camuflaje |
| *meeting_room.oni* | 4 | 0 | 1407 | Abandono de varios objetos (carpeta, maletín, bolso de portátil y teléfono móvil) | Camuflaje, sombras, oclusiones y cambios de iluminación | Camuflaje |
| *Illumination _change.oni* | 1 | 0 | 550 | Abandono de una pila de libros en un entorno con iluminación variable. | Camuflaje y cambios de iluminación | Ninguno |
| *interaction.oni* | 0 | 0 | 2658 | Interacción entre dos personas | Camuflaje, sombras, oclusiones y cambios de iluminación | Camuflaje y oclusiones |
| *entrance.oni* | 0 | 0 | 718 | Personas accediendo y saliendo de un recinto | Camuflaje, sombras, oclusiones y cambios de iluminación | Oclusiones |
| *sunshine.oni* | 0 | 0 | 10113 | Salón interior con fuerte iluminación solar | Cambio de iluminación (solar) | Cambio de iluminación (solar) |

Tabla 5.1: Resumen de características de las secuencias del *Dataset*

## 5.3. Métricas de evaluación

En la evaluación final de eventos se definen tres parámetros para medir el grado de efectividad tanto del sistema inicial como del mejorado. Estos parámetros son:

$$P = \frac{TP}{TP + FP} \qquad (Precision)$$

$$R = \frac{TP}{TP + FN} \qquad (Recall)$$

$$F = \frac{2 \times P \times R}{P + R} \quad (Final - Score)$$

Donde, en función del tipo de evaluación realizado (*foregrounds* o eventos), $TP$, $FP$ y $FN$ representan:

- Evaluación de *foregrounds*
  - TP: cantidad de píxeles a uno que están en el *foreground* y en el *ground truth*
  - FP: cantidad de píxeles a uno que están en el *foreground*, pero no en el *ground truth*
  - FN: cantidad de píxeles a cero que están en el *foreground*, pero no en el *ground truth*
- Evaluación de eventos
  - TP: número de detecciones correctas
  - FP: número de detecciones incorrectas
  - FN: número de detecciones perdidas

Estas cantidades se relacionan por los parámetros $P$ y $R$, que a su vez quedan relacionados por el parámetro $F$. El valor de este último da una idea global de la eficacia del sistema a través de una única cifra que va de 0 (pésimo) a 1 (óptimo).

## 5.4. Resultados

Para los ejemplos mostrados en esta sección se utilizan los videos del *Dataset* descrito en la sección 5.2. Concretamente, se analizan cinco secuencias*: stolen_box.oni*, *abandoned_box.oni*, *stolen_bin.oni*, *abandoned_bin.oni, illumination_change.oni, interaction.oni* y *meeting_room.oni*.

Esta sección se divide a su vez en tres subsecciones. En la primera (sección 5.4.1) se evalúan visual y numéricamente (en función de los parámetros mencionados en la sección 5.3) la exactitud de las máscaras de *foreground* de color, profundidad y la combinación de ambas, respecto al *ground truth*. En la segunda (sección 5.4.2) se evalúan de forma visual y cuantitativa una serie de eventos de robo/abandono. Finalmente, en la subsección 5.4.3 se analizan brevemente los efectos de la profundidad en otros módulos del sistema.

### 5.4.1. Evaluación de *foregrounds* de color, profundidad y combinado

En este apartado se evalúa la similitud de las máscaras de *foreground* de color, profundidad, y la combinación de las dos (sección 4.4), con respecto al *ground truth* correspondiente. Dicha similitud se cuantifica en función de los parámetros de la sección 5.3.

En concreto, se analizan 4 secuencias de video (*meeting_room.oni*, *abandoned_box.oni*, *illumination_change.oni* e *interaction.oni*, del Dataset de la sección 5.2) y de cada una de ellas se toman 8 muestras.

Las figuras 5.10, 5.11, 5.12, 5.13 y las tablas 5.2, 5.3, 5.4, 5.5, muestran de manera visual y numérica los resultados de la evaluación de *foregrounds* para cada muestra de las secuencias.

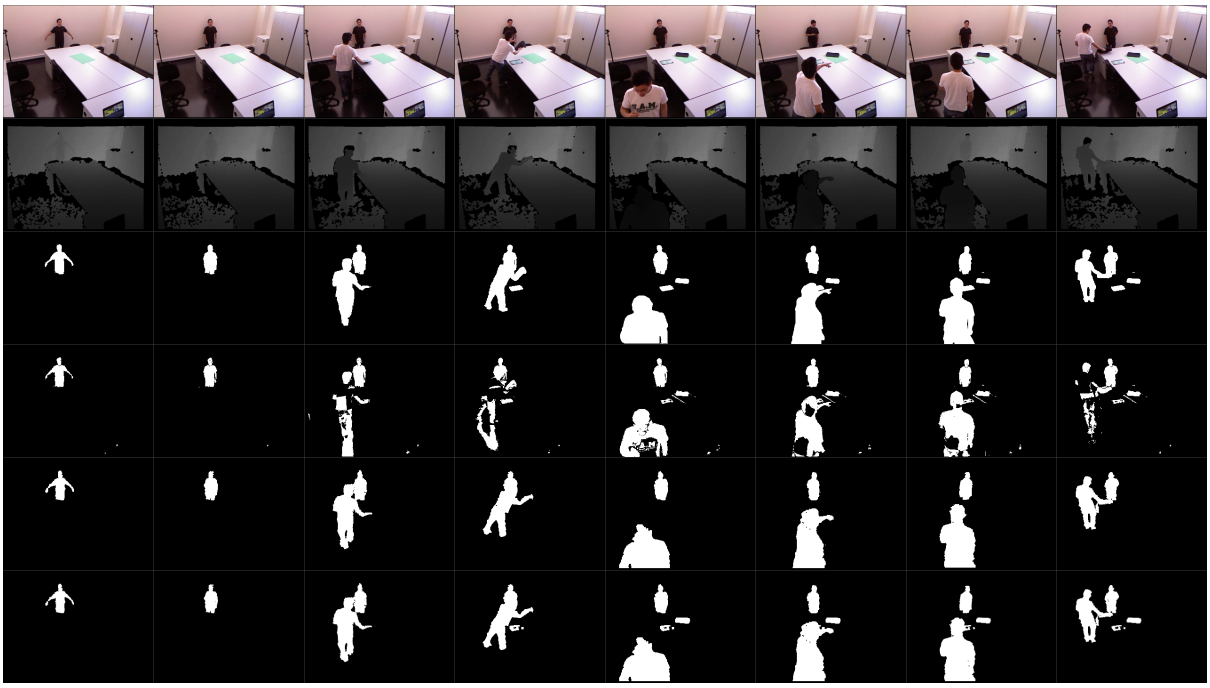


Figura 5.10: Evaluación de *foregrounds* para la secuencia *meeting_room.oni.* Fila 1) Imagen del frente. Fila 2) Imagen de profundidad. Fila 3) *Ground truth.* Fila 4) *Foreground* de color. Fila 5) *Foreground* de profundidad. Fila 6) *Foreground* combinado.

| Color | | | | Profundidad | | | | Combinación | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **#frame** | **P** | **R** | **F** | **#frame** | **P** | **R** | **F** | **#frame** | **P** | **R** | **F** |
| 0 | 0.969 | 0.884 | **0.924** | 0 | 0.889 | 0.882 | 0.885 | 0 | 0.889 | 0.882 | 0.885 |
| 1 | 0.983 | 0.877 | **0.927** | 1 | 0.930 | 0.905 | 0.917 | 1 | 0.930 | 0.905 | 0.917 |
| 2 | 0.746 | 0.592 | 0.660 | 2 | 0.913 | 0.902 | **0.908** | 2 | 0.913 | 0.902 | **0.908** |
| 3 | 0.669 | 0.531 | 0.592 | 3 | 0.875 | 0.813 | 0.843 | 3 | 0.872 | 0.852 | **0.862** |
| 4 | 0.965 | 0.788 | 0.868 | 4 | 0.958 | 0.807 | 0.876 | 4 | 0.946 | 0.857 | **0.899** |
| 5 | 0.970 | 0.761 | 0.853 | 5 | 0.930 | 0.871 | 0.900 | 5 | 0.924 | 0.911 | **0.917** |
| 6 | 0.969 | 0.810 | 0.882 | 6 | 0.941 | 0.882 | 0.910 | 6 | 0.931 | 0.927 | **0.929** |
| 7 | 0.782 | 0.548 | 0.644 | 7 | 0.891 | 0.809 | 0.848 | 7 | 0.880 | 0.899 | **0.889** |

Tabla 5.2: Resultados de evaluación de *foregrounds* de la secuencia *meeting_room.oni*.

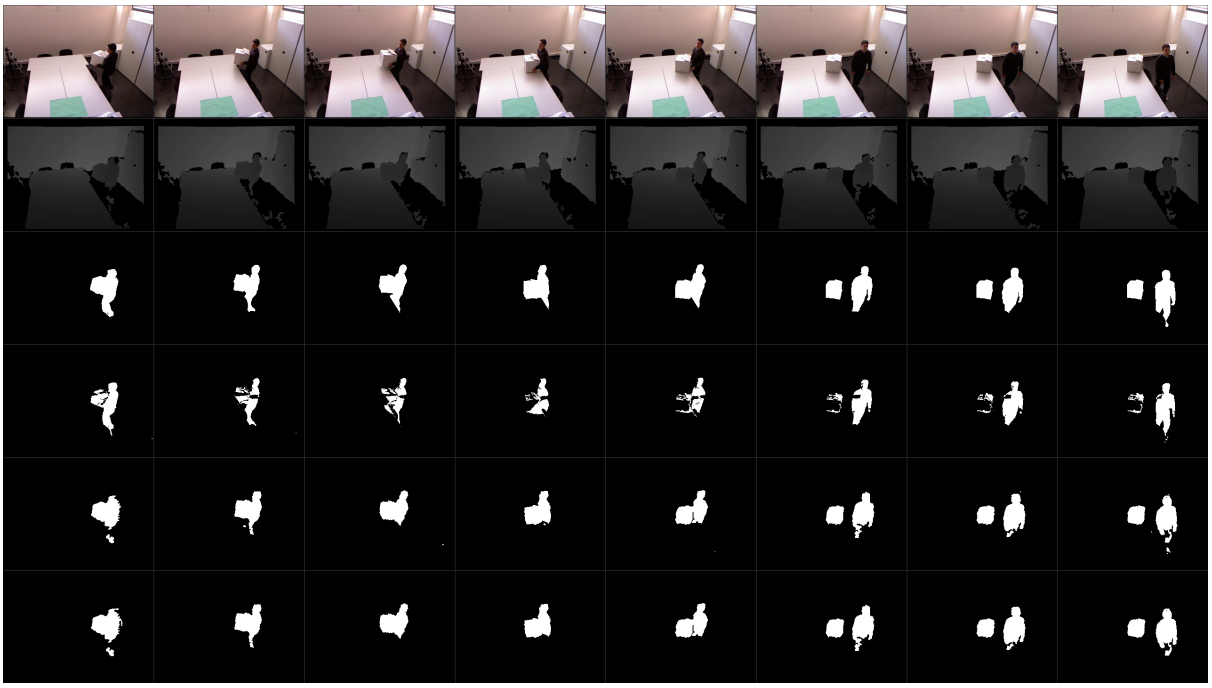


Figura 5.11: Evaluación de *foregrounds* para la secuencia *abandoned_box.oni*. Fila 1) Imagen del frente. Fila 2) Imagen de profundidad. Fila 3) *Ground truth*. Fila 4) *Foreground* de color. Fila 5) *Foreground* de profundidad. Fila 6) *Foreground* combinado.

| Color | | | | Profundidad | | | | Combinación | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **#frame** | **P** | **R** | **F** | **#frame** | **P** | **R** | **F** | **#frame** | **P** | **R** | **F** |
| 0 | 0.981 | 0.858 | **0.915** | 0 | 0.893 | 0.811 | 0.850 | 0 | 0.877 | 0.811 | 0.843 |
| 1 | 0.969 | 0.683 | 0.801 | 1 | 0.860 | 0.844 | 0.852 | 1 | 0.867 | 0.844 | **0.855** |
| 2 | 0.926 | 0.576 | 0.710 | 2 | 0.901 | 0.824 | 0.861 | 2 | 0.903 | 0.824 | **0.862** |
| 3 | 0.905 | 0.524 | 0.664 | 3 | 0.901 | 0.836 | **0.867** | 3 | 0.900 | 0.831 | 0.864 |
| 4 | 0.962 | 0.516 | 0.671 | 4 | 0.901 | 0.840 | **0.869** | 4 | 0.901 | 0.840 | **0.869** |
| 5 | 0.985 | 0.671 | 0.798 | 5 | 0.886 | 0.852 | **0.869** | 5 | 0.886 | 0.852 | **0.869** |
| 6 | 0.981 | 0.675 | 0.800 | 6 | 0.879 | 0.873 | **0.876** | 6 | 0.879 | 0.873 | **0.876** |
| 7 | 0.973 | 0.704 | 0.817 | 7 | 0.875 | 0.819 | 0.846 | 7 | 0.885 | 0.815 | **0.848** |

Tabla 5.3: Resultados de evaluación de *foregrounds* de la secuencia *abandoned_box.oni*.

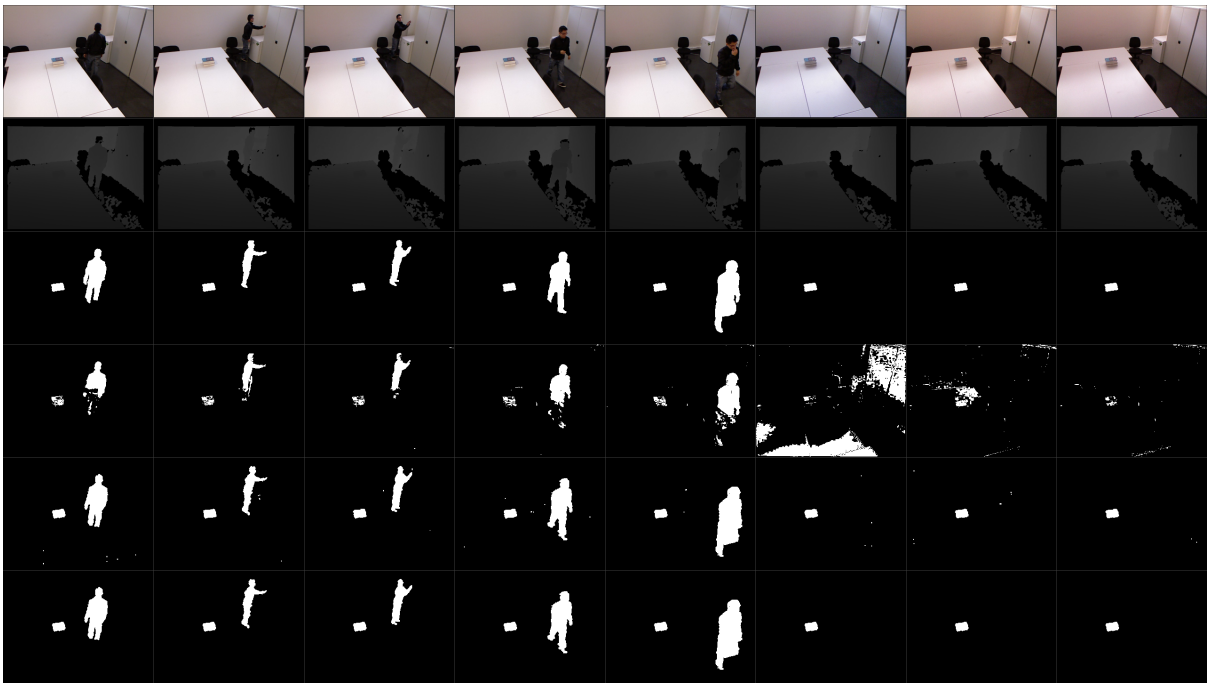


Figura 5.12: Evaluación de *foregrounds* para la secuencia *illumination_change.oni.* Fila 1) Imagen del frente. Fila 2) Imagen de profundidad. Fila 3) *Ground truth.* Fila 4) *Foreground* de color. Fila 5) *Foreground* de profundidad. Fila 6) *Foreground* combinado.

| Color | | | | Profundidad | | | | Combinación | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **#frame** | **P** | **R** | **F** | **#frame** | **P** | **R** | **F** | **#frame** | **P** | **R** | **F** |
| 0 | 0.961 | 0.623 | 0.756 | 0 | 0.836 | 0.860 | 0.848 | 0 | 0.843 | 0.860 | **0.851** |
| 1 | 0.894 | 0.760 | 0.821 | 1 | 0.800 | 0.861 | 0.829 | 1 | 0.807 | 0.861 | **0.833** |
| 2 | 0.949 | 0.743 | **0.834** | 2 | 0.810 | 0.858 | 0.833 | 2 | 0.815 | 0.854 | **0.834** |
| 3 | 0.931 | 0.707 | 0.803 | 3 | 0.807 | 0.878 | 0.841 | 3 | 0.812 | 0.877 | **0.843** |
| 4 | 0.955 | 0.687 | 0.799 | 4 | 0.837 | 0.899 | **0.867** | 4 | 0.839 | 0.898 | **0.867** |
| 5 | 0.009 | 0.318 | 0.018 | 5 | 0.811 | 0.908 | **0.857** | 5 | 0.821 | 0.889 | 0.853 |
| 6 | 0.136 | 0.533 | 0.217 | 6 | 0.774 | 0.891 | 0.828 | 6 | 0.821 | 0.889 | **0.853** |
| 7 | 0.327 | 0.355 | 0.341 | 7 | 0.824 | 0.900 | **0.861** | 7 | 0.821 | 0.889 | 0.853 |

Tabla 5.4: Resultados de evaluación de *foregrounds* de la secuencia *illumination_change.oni*.

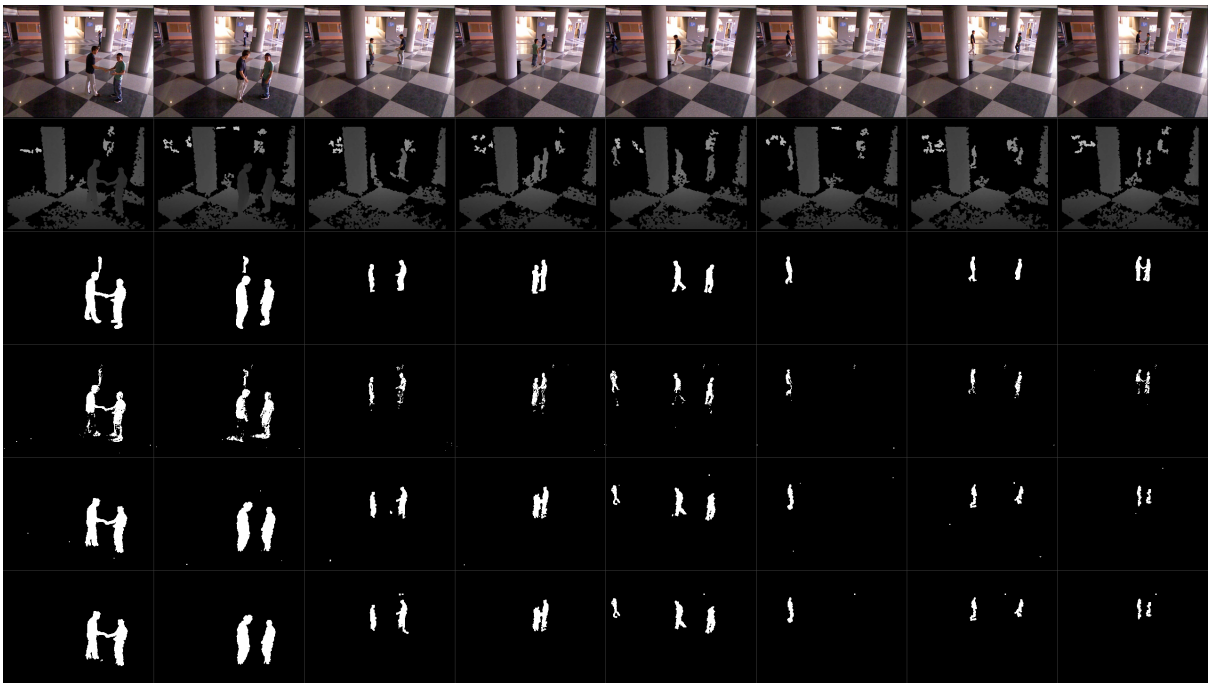


Figura 5.13: Evaluación de *foregrounds* para la secuencia *interaction.oni*. Fila 1) Imagen del frente. Fila 2) Imagen de profundidad. Fila 3) *Ground truth*. Fila 4) *Foreground* de color. Fila 5) *Foreground* de profundidad. Fila 6) *Foreground* combinado.

| Color | | | | Profundidad | | | | Combinación | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **#frame** | **P** | **R** | **F** | **#frame** | **P** | **R** | **F** | **#frame** | **P** | **R** | **F** |
| 0 | 0.953 | 0.688 | 0.799 | 0 | 0.826 | 0.757 | 0.790 | 0 | 0.826 | 0.758 | **0.791** |
| 1 | 0.677 | 0.608 | **0.641** | 1 | 0.574 | 0.683 | 0.623 | 1 | 0.576 | 0.679 | 0.623 |
| 2 | 0.917 | 0.604 | 0.728 | 2 | 0.624 | 0.536 | 0.577 | 2 | 0.627 | 0.536 | **0.578** |
| 3 | 0.881 | 0.730 | **0.798** | 3 | 0.660 | 0.573 | 0.614 | 3 | 0.673 | 0.573 | 0.619 |
| 4 | 0.970 | 0.532 | **0.687** | 4 | 0.779 | 0.403 | 0.532 | 4 | 0.776 | 0.403 | 0.531 |
| 5 | 0.906 | 0.671 | 0.771 | 5 | 0.823 | 0.766 | 0.794 | 5 | 0.826 | 0.766 | **0.795** |
| 6 | 0.918 | 0.622 | **0.741** | 6 | 0.717 | 0.716 | 0.717 | 6 | 0.699 | 0.716 | 0.707 |
| 7 | 0.932 | 0.661 | **0.773** | 7 | 0.749 | 0.710 | 0.729 | 7 | 0.751 | 0.707 | 0.728 |

Tabla 5.5: Resultados de evaluación de *foregrounds* de la secuencia *interaction.oni.*

| Secuencia | Parámetro | Color | Profundidad | Combinación |
|---|---|---|---|---|
| *meeting_room.oni* | **P** | 0.882±0.016 | 0.916±0.001 | 0.911±0.001 |
| | **R** | 0.724±0.021 | 0.859±0.002 | 0.892±0.001 |
| | **F** | 0.794±0.019 | 0.886±0.001 | **0.901±0.000** |
| *abandoned_box.oni.* | **P** | 0.960±0.001 | 0.887±0.000 | 0.887±0.000 |
| | **R** | 0.651±0.013 | 0.837±0.000 | 0.836±0.000 |
| | **F** | 0.772±0.007 | **0.861±0.000** | **0.861±0.000** |
| *illumination_change.oni* | **P** | 0.645±0.171 | 0.812±0.000 | 0.822±0.000 |
| | **R** | 0.591±0.030 | 0.882±0.000 | 0.877±0.000 |
| | **F** | 0.574±0.108 | **0.846±0.000** | 0.848±0.000 |
| *interaction.oni* | **P** | 0.894±0.008 | 0.719±0.009 | 0.719±0.009 |
| | **R** | 0.640±0.004 | 0.643±0.016 | 0.642±0.016 |
| | **F** | **0.742±0.003** | 0.672±0.010 | 0.672±0.010 |
| Total | **P** | 0.845±0.019 | 0.834±0.008 | 0.835±0.007 |
| | **R** | 0.652±0.003 | 0.805±0.012 | 0.812±0.013 |
| | **F** | 0.721±0.010 | 0.816±0.010 | **0.821±0.010** |

Tabla 5.6: Resultados globales de evaluación de *foregrounds*

En todas las secuencias se espera que el *foreground* obtenido combinando las máscaras de color y profundidad dé resultados mejores o iguales que cada una por separado. Sin embargo, hay casos en los que no ocurre así. Por ejemplo, en la tabla 5.2, la puntuación final *(F)* del *frame* 0 es mayor en la máscara de color. Esto es así porque, de acuerdo al esquema de fusión de la figura 4.15, el *foreground* final (el de la combinación) se queda con los *blobs* de la máscara de profundidad, la cual, en este caso, es de peor calidad que la máscara de color (en la máscara de profundidad, el *blob* es más "delgado"). Lo mismo ocurre con el *F-score (F)* de*l frame* 0 en la tabla 5.3 . La máscara final es igual a la máscara de profundidad, que, este caso, tiene sus *blobs* en peores condiciones que la máscara de color.

Otro ejemplo donde el *F-score (F)* es mejor en la máscara de color que en la combinada,

ocurren en el *frame* 1 de la tabla 5.5. Lo que ocurre aquí es que la persona que está más al fondo (~8 metros) no es detectada por el sensor de profundidad. Al ser considerado el *blob* correspondiente a dicha persona como un *blob* no estático ni pequeño, entonces no pasa a la máscara final, de acuerdo con el esquema de la figura 4.15.

No obstante, en la mayoría de los casos los resultados mejoran considerablemente. Por ejemplo, el *F-score* del frame 4 en la tabla 5.3 es mayor en la máscara combinada (0.869) que la máscara de color (0.671). Esto ocurre así, porque se corrige el problema de camuflaje de la máscara de color. Otra de las mejoras se puede observar en el *frame* 5 de la figura 5.12, donde el *F-score* es mayor el máscara combinada. En este caso, la mejora se produce debido a la inmunidad de la profundidad a los cambios de iluminación.

Con todo, se puede observar en la tabla 5.6, en general, que en todas las secuencias (excepto en *interaction.oni*) el *foreground* de la combinación presenta una puntuación final (*F*) mucho mejor que el *foreground* basado únicamente en color (sistema inicial).

### 5.4.2. Evaluación de eventos de robo/abandono

En este apartado se evalúan de forma visual y cuantitativa los eventos de robo y abandono, tanto en el sistema inicial como en el mejorado. Para ello se utilizan secuencias del *Dataset* que son evaluadas acorde con la sección 5.3.

Para la secuencia *stolen_box.oni,* figura 5.14, el evento a detectar es un robo. Lo más destacable es la presencia de un problema acentuado de camuflaje , donde el sistema inicial difícilmente puede detectar la caja robada. El sistema inicial detecta el evento de robo, pero solamente lo hace para parte del objeto, mientras que gran parte de mismo queda sin detectar. Esto se puede considerar como dos eventos (de robo) por separado, uno que es detectado (que denominaremos evento *stolen_box-a*) y otro que no lo es (evento *stolen_box-b*). Por su lado, el sistema mejorado detecta el evento sin problemas, con lo que se considerará que éste ha detectado los dos eventos correctamente. Además, las probabilidades de acierto para el sistema mejorado superan en un 20 % al inicial (0.6 y o 0.79, respectivamente).

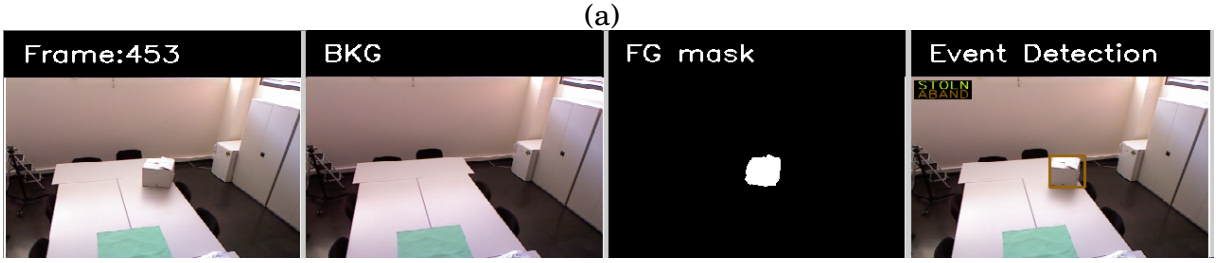

(a)


(b)

Figura 5.14: Detección de objeto robado en la secuencia *stolen_box.oni* con el sistema (a) inicial y (b) mejorado

Para la secuencia *abandoned_box.oni,* figura 5.15, el evento a detectar es un abandono. En este caso, el problema principal es un fuerte camuflaje, lo que provoca que los resultados para el sistema inicial sean nefastos. En este caso, el evento analizado se puede considerar como tres por separado, que, para el caso del sistema inicial, el primero correspondería a una detección correcta del abandono (evento *abandoned_box-a*), el segundo a una detección errónea (evento *abandoned_box-b*) y el tercero a una no detección (evento *abandoned_box-c*). Por su parte, el sistema mejorado detecta correctamente el evento con una probabilidad del 99 % y, por tanto, los tres considerados.


(a)

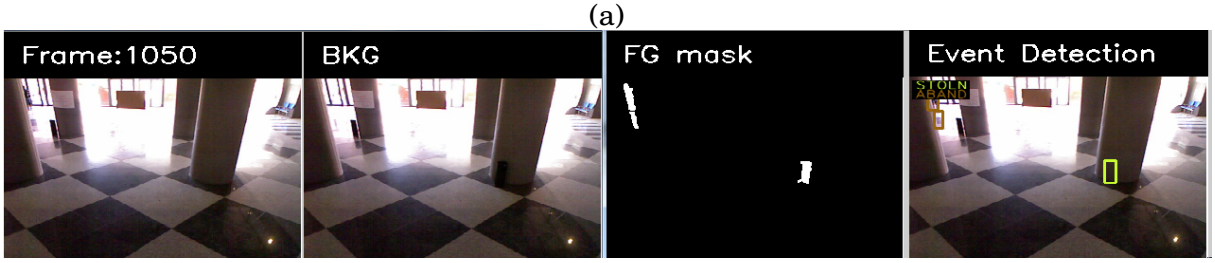



(b)

Figura 5.15: Detección de objeto abandonado en la secuencia *abandoned_box.oni* con el sistema (a) inicial y (b) mejorado

Para la secuencia *stolen_bin.oni*, figura 5.16, el evento a detectar es un robo. La detección del evento se realiza correctamente por parte de los dos sistemas. Sin embargo, en ambos casos, la última etapa (discriminación de objetos) se realiza utilizando únicamente la información de color (sección 3.5). Esto es así, porque el modelo de fondo para la secuencia de profundidad tiene valores nulos (no profundidad) justamente en el objeto a detectar (figura 5.4). Cuando tales valores nulos son remplazados por unos válidos del frente, estos pueden deberse a la aparición o desaparición de un objeto (como es el caso). Un análisis similar al de la sección 3.5 aplicado a imágenes de profundidad podría desvelar el tipo de evento (robo o abandono); sin embargo, en este trabajo, ante una situación así, el análisis se lleva a cabo únicamente utilizando información de color, como lo hace el sistema inicial. Esto no significa que la profundidad no aporte nada en este caso, todo lo contrario, da una mayor definición al que el *blob* correspondiente (sección 4.4). Por ejemplo, si el cubo de basura que aparece en la imagen tuviese líneas horizontales de un color similar a la columna junto a la que está, el *blob* asociado sería troceado y la detección se realizaría sobre varios objetos inexistentes, como ocurre en la secuencia de la imagen 5.15, por ejemplo. En este caso, el evento es detectado tanto por el sistema inicial como por el mejorado.


(a)




(b)

Figura 5.16: Detección de objeto robado en la secuencia *stolen_bin.oni* con el sistema (a) inicial y (b) mejorado

Para la secuencia *abandoned_bin.oni*, figura 5.17, el evento a detectar es un abandono. Aquí ocurre el caso contrario al anterior (secuencia *stolen_bin.oni*). Esta vez, el fondo tiene valores válidos, y es el objeto (la papelera) la que introduce valores no válidos. Al igual que antes, el análisis para la detección del evento se realiza con los procedimientos ya existentes basados en color (sección 3.5). El evento es detectado por ambos sistemas.


(a)

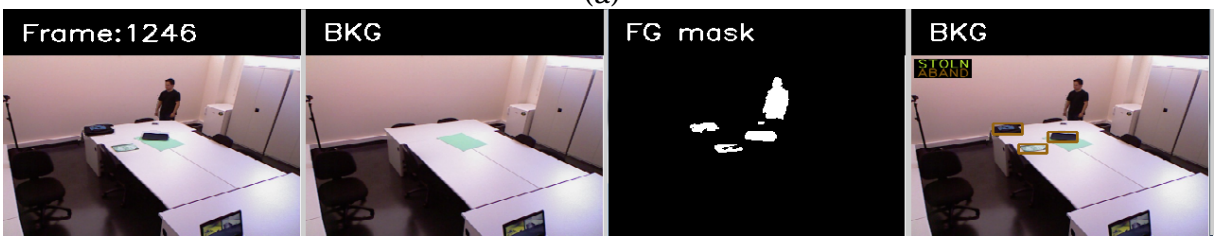

(b)

Figura 5.17: Detección de objeto abandonado en la secuencia *abandoned_bin.oni* con el sistema (a) inicial y (b) mejorado

Para la secuencia *meeting_room.oni*, figura 5.18, evento a detectar es un abandono. En

esta secuencia se pretende detectar el abandono de cuatro objetos. Ambos sistemas detectan solamente tres de los eventos y desechan uno, el del teléfono móvil abandonado. Esto se debe a un filtrado de *blobs* atendiendo a su tamaño. La carpeta y la funda del portátil no tienen suficiente volumen, con lo que serán ignorados por el sensor de profundidad. Entonces, la detección se lleva a cabo por el módulo de discriminación que ya viene en el sistema inicial (sección 3.5). Se puede ver que la detección en el sistema mejorado la precisión es mayor. Esto es debido a que los parámetros de segmentación en color han sido ajustados en el sistema mejorado, de modo que las formas de los objetos "pequeños" sean alteradas lo menos posible. Concretamente, se ha disminuido el tamaño de la ventana de análisis y disminuido el umbral de ruido (sección 3.2). Con todo, lo que se pretende demostrar con este ejemplo es que a pesar de que dos de los objetos son imperceptibles para el sensor de profundidad, el sistema sigue ofreciendo las mismas prestaciones que el sistema inicial. De aquí se deduce que los valores de los parámetros mencionados en la sección 5.3 serán mayores o iguales en el sistema mejorado respecto al sistema de partida, el prototipo del grupo VPU-Lab. En este caso, se distinguen tres eventos de abandono: el correspondiente a la carpeta (*Meeting_room-carpeta*), al maletín *(Meeting_room-maletín*), a la bolsa del portátil *(Meeting_room-funda*) y al teléfono móvil. Ambos sistemas detectan todos los eventos, a excepción del abandono del teléfono móvil (*Meeting_room-móvil*).


(a)




(b)

Figura 5.18: Detección de objetos abandonados en la secuencia *meeting_room.oni* con el sistema (a) inicial y (b) mejorado

Para la secuencia *illumination_change.oni*, figura 5.19, el evento a detectar es un abandono de una pila de libros en un entorno con iluminación variable. El sistema inicial no detecta el evento, puesto que el *blob* correspondiente al objeto es solapado por el genera-

do en el cambio de iluminación, cuyo tamaño es variable (siendo no estático). El sistema mejorado detecta el evento correctamente, pero además detecta un robo inexistente. Esto se debe al cambio de iluminación que se produce en una región aislada (la nevera, en este caso). Al no tratarse las sombras ni la sobre-iluminación, el *blob* correspondiente es considerado como válido.


(a)


(b)

Figura 5.19: Detección de objetos abandonados en la secuencia *illumination_change.oni* con el sistema (a) inicial y (b) mejorado

Las tablas siguientes evalúan los resultados de eventos acorde a la sección 5.3.

| | **Sistema inicial** | | | | | | **Sistema mejorado** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TP | FP | FN | P | R | F | TP | FP | FN | P | R | F |
| Stolen_box-a | 1 | 0 | 0 | | | | 1 | 0 | 0 | | | |
| Stolen_box-b | 0 | 0 | 1 | | | | 1 | 0 | 0 | | | |
| abandoned_box-a | 1 | 0 | 0 | | | | 1 | 0 | 0 | | | |
| abandoned_box-b | 0 | 0 | 1 | | | | 1 | 0 | 0 | | | |
| abandoned_box-c | 0 | 0 | 1 | | | | 1 | 0 | 0 | | | |
| Stolen_bin | 1 | 0 | 0 | | | | 1 | 0 | 0 | | | |
| abandoned_bin | 1 | 0 | 0 | | | | 1 | 0 | 0 | | | |
| Meeting_room-carpeta | 1 | 0 | 0 | | | | 1 | 0 | 0 | | | |
| Meeting_room-maletín | 1 | 0 | 0 | | | | 1 | 0 | 0 | | | |
| Meeting_room-funda | 1 | 0 | 0 | | | | 1 | 0 | 0 | | | |
| Meeting_room-móvil | 0 | 0 | 1 | | | | 0 | 0 | 1 | | | |
| Illumination_change | 0 | 0 | 1 | | | | 1 | 0 | 0 | | | |
| **Total** | **7** | **0** | **5** | **1** | **0.58** | **0.73** | **11** | **0** | **1** | **1** | **0.92** | **0.96** |

Tabla 5.7: Resultados del sistema inicial y mejorado

Se puede apreciar que, para los eventos seleccionados, la puntuación final del el sistema

mejorado (**0.96**) es considerablemente mayor que el sistema inicial (**0.73**).

Por su parte, el módulo discriminador basado en profundidad (cuando entra en funcionamiento) da probabilidades mayores de que un evento sea el detectado. Por ejemplo, para las secuencias *abandoned_box.oni* (evento: abandono) y *stolen_box.oni* (evento: robo), si se evalúa el *blob* generado en la máscara final, por un lado, con el discriminador inicial (basado en color); y por otro, con el discriminador basado en profundidad, se obtienen los resultados de la tabla 5.8.

| | **Color** | | **Profundidad** | |
|---|---|---|---|---|
| Secuencia | $P_{abandono}$ | $P_{robo}$ | $P_{abandono}$ | $P_{robo}$ |
| *abandoned_box.oni* | - | 0.66 | 0.99 | - |
| *stolen_box.oni* | - | 0.78 | - | 0.93 |

Tabla 5.8: Resultados de discriminación basado en color y basado en profundidad

En la tabla 5.8 se puede ver que las probabilidades de detección de los evento son mayores del **90 %** para el discriminador basado en profundidad. Sin embargo, en el discriminador basado en color, solamente se detecta de manera correcta el evento de robo, con una probabilidad del **78 %**. El evento de abandono se detecta como uno de robo y con una probabilidad del **66 %**. Esto ocurre así, porque el discriminador basado en color (sección 3.5) requiere que el *blob* sea lo más fiel posible al objeto correspondiente, sobre todo, en los bordes. El *blob* de la máscara final, al tomarse de la máscara de profundidad, tiene imprecisiones en los bordes, lo que provoca esa detección errónea.

### 5.4.3. Efecto de la profundidad en otros módulos del sistema

Los módulos en los que no se aplica la información de profundidad también varían los resultados que producen, ya que dependen de otros que si lo hacen. En concreto estos módulos son los de Detección de objetos estacionarios y Clasificación de objetos.

#### Detección de objetos estacionarios

La máscara binaria obtenida en la etapa de Detección del primer plano (sección 4.3.2) se filtra para eliminar el ruido granulado. A diferencia del procesado en color, esta vez no hay que eliminar sombras. Si los problemas relacionados con luz solar y las franjas indicados en la sección 4.3.3 no aparecen o son mitigados, entonces es muy probable que la máscara de *foreground* de no presente falsos positivos (sección 4.3.2). No obstante, el caso opuesto (falsos negativos) sí que es muy propenso a ocurrir.

Inicialmente, la máscara que recibía este módulo se obtenía de la etapa de segmentación con imágenes de color (sección 3.2). Sin embargo, la precisión y fiabilidad de esta

máscara es bastante cuestionable debido a los problemas inherentes a la hora de operar con imágenes de color. Estos problemas se puede ver ilustrados claramente en la figura 2.2.

Esta vez, este módulo recibe es una combinación de las máscaras de *foreground* de color y profundidad (sección 4.4), en la cual se han suprimido la mayoría de los problemas antes mencionados. Esto reduce considerablemente el análisis de *blobs* para determinar cuál de ellos es estático, lo que se traduce en un ahorro en coste computacional bastante importante. En la figura 5.20 se pueden apreciar las diferencias en el número de *blobs* considerados en las máscaras de *foreground* de color y de color más profundidad, respectivamente.

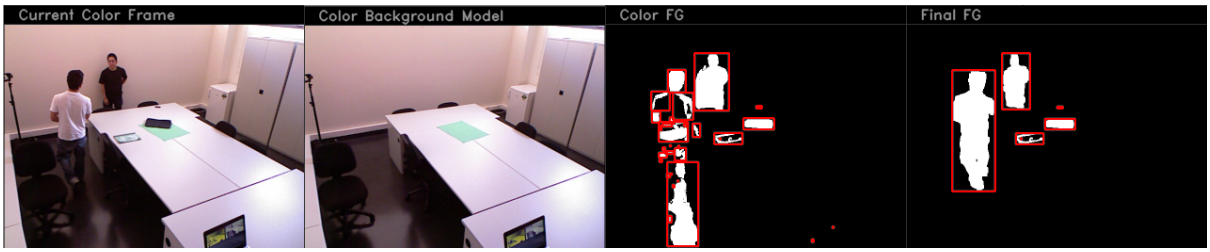



Figura 5.20: *Bounding boxes* analizados en el *foreground* de color (color FG) y en el combinado (Final FG)

**Clasificación de objetos**

Esta vez, la clasificación es más precisa, puesto que los *blobs* correspondientes a personas entran sin sombras y, bajo ciertas condiciones, sin el problema de camuflaje. Estas condiciones se resumen en que al menos una de las imágenes de profundidad (frente o fondo) tenga un valor de profundidad válido en la región bajo análisis. En otras palabras, si en el modelo de fondo existe un pixel con profundidad nula (valor no válido) y sobre esa región pasa una persona, cuya indumentaria también produjese valores nulos, en este caso, habría camuflaje en la máscara de *foreground* final (figura 5.21), de acuerdo al esquema de la figura 4.15.

Como se puede apreciar, la chaqueta y el pantalón negros y con propiedades reflectivas producen camuflaje cuanto solapan al suelo de superficie también reflectante.

Para llevar a cabo la clasificación es importante considerar el problema del camuflaje. Por ejemplo, si la persona que viste de blanco en la figura 5.20, permaneciese mucho tiempo inmóvil, el sistema inicial (capítulo 3), trocearía el *blob* correspondiente y lo consideraría como varios objetos estáticos (color FG). Sin embargo, el sistema mejorado, considera el *blob* integro (Final FG) y lo descarta como un posible objeto estático.

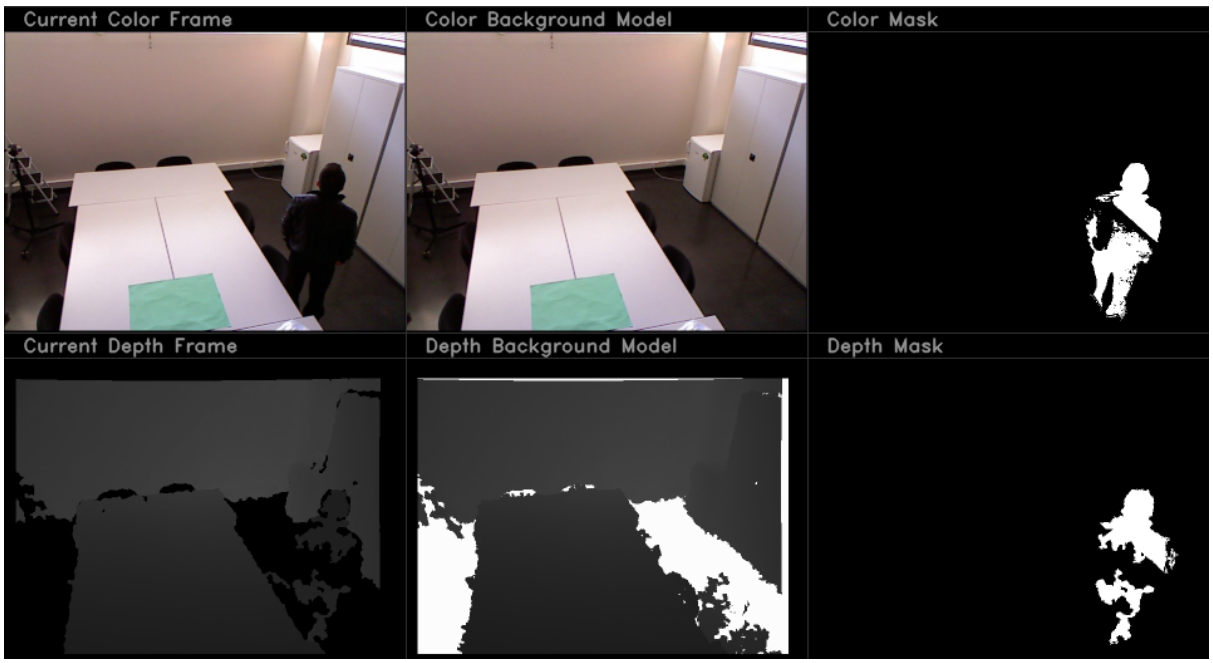



Figura 5.21: Camuflaje en imágenes de profundidad

# Capítulo 6

# Conclusiones y trabajo futuro

## 6.1. Introducción

En este capítulo se resumen (sección 6.2) los principales objetivos y logros conseguidos en este proyecto. Asimismo, se evalúan las ventajas e inconvenientes del uso de la información de profundidad y se extraen unas conclusiones (6.3). Por último se plantean nuevos desafíos para un trabajo futuro (sección 6.4), a fin de extender el funcionamiento del sistema a otros entornos.

## 6.2. Resumen

En un sistema de video vigilancia como lo es el estudiado en este proyecto, una de las etapas más delicadas es la de segmentación (o detección del primer plano); aquella donde se marcan todas las alteraciones producidas en el modelo de fondo, debidas a la presencia de nuevos elementos o ausencia de los que ya estaban. Las etapas posteriores a la segmentación basan su funcionamiento en el resultado (máscara de *foreground*) producido por ésta. Por ello interesa que esta máscara sea lo más precisa y fiable posible.

Concretamente, el sistema que se analiza en este proyecto consta de cuatro etapas: detección del primer plano, detección de objetos estacionarios, clasificación de objetos y discriminación de objetos. En este trabajo, los esfuerzos se centran principalmente en mejorar la primera y la cuarta etapa. Estas aportaciones son posibles gracias al aprovechamiento de la información de profundidad proporcionada por un nuevo y novedoso dispositivo: el sensor Kinect (sección 2.4). De entre sus múltiples ventajas destacan su bajo coste y la razonable calidad de información que proporciona. A continuación se resume las contribuciones realizadas a cada etapa.

- **Detección del primer plano** (sección 4.3). Las mayores contribuciones se han rea-

lizado sobre todo en esta etapa. Se realiza una segmentación basada en imágenes de color y otra basada en información de profundidad en paralelo. Posteriormente se combinan ambas máscaras resultantes, intentando extraer solamente la información auténtica de cada una de ellas. Se aprovecha el hecho de que en la mayoría de los casos los problemas presentes en una máscara son complementarios a la otra. Así, por ejemplo, en la máscara de profundidad, las sombras no suponen ningún obstáculo a la hora de segmentar, en cambio, en la máscara de profundidad, esto es un factor crítico a tener en cuenta. Sin embargo, los bordes de los *blobs* presentes en la máscara de profundidad no tienen la definición que presentaría el mismo *blob* en la máscara de color. Esto puede suponer un hándicap en etapas posteriores, como la etapa de discriminación de objetos, puesto que para su correcto funcionamiento los bordes del *blob* deben ser lo más fieles posible al del objeto correspondiente. Con todo, lo que se ha hecho es procurar integrar en la máscara de profundidad los *blobs* de los objetos, cuya profundidad no es posible estimar, pero que sí aparecen en la máscara obtenida con imágenes de color (sección 4.4).

- **Discriminación de objetos** (sección 4.5). En este módulo se hace una adaptación de los procesos ya existentes para hacer uso de la información de profundidad. De todos los *blobs* estáticos y pertenecientes a objetos que llegan a este módulo, se eligen aquellos que aparecen también en la máscara de profundidad y se calcula la distancia a la que están. Esta distancia es luego comparada con la distancia al fondo en esta misma región. Si es menor, entonces se trata de un objeto abandonado, en caso contrario, el objeto ha sido retirado de la escena. Para los *blobs* que no aparecen en la máscara de profundidad, el análisis que realiza para determinar el tipo de evento es el mismo que ya está implementado el prototipo del VPU-Lab, como se describe en la sección 2.2.

Por otro lado, para poder testear el funcionamiento del sistema global se han creado una serie de secuencias de video (sección 5.2) donde se reflejan los problemas típicos que perjudican el funcionamiento de un sistema de estas características. Este *Dataset* se ha probado tanto en el sistema inicial como el sistema mejorado. Los *Datasets* "tradicionales" disponibles para el testeo de este tipo de eventos no sirven para probar las mejoras realizadas, dado que carecen de las imágenes de profundidad correspondientes. El conjunto de datos de video creados exponen los problemas más comunes en imágenes de color (sombras, cambios de iluminación, camuflajes, etc.) y de profundidad (camuflaje) en todas sus formas posibles.

## 6.3. Conclusiones

Una de las grandes ventajas de segmentar utilizando imágenes de profundidad es que en la máscara de *foreground* de profundidad no presenta falsos positivos (sección 4.3.2). Sin

embargo, el caso opuesto (falsos negativos), si es muy propenso a ocurrir debido a ciertas características de los objetos. Una primera aproximación para mejorar el prototipo existente en el grupo VPU-Lab es la descrita en la sección 4.3.2, que cosiste en una operación binaria OR entre esta máscara y la máscara de *foreground* que el sistema existente produce. Esta operación introduce claramente una mejora al eliminar el problema de camuflaje casi en su totalidad (salvos ciertas condiciones) como se puede observar en la figura 4.14. Sin embargo, seguiría siendo necesario un procesado adicional para eliminar sombras y cambios de iluminación, lo que computacionalmente es costoso. Por otro lado, estos problemas no suponen un problema para las imágenes de profundidad, lo que conduce a considerar mejor la máscara de profundidad y extraer los objetos, cuya profundidad no es posible estimar, de la máscara de color. El procedimiento seguido para esta fusión se describe en la sección 4.4. La máscara resultante no necesita eliminación de sombras, pero si puede sufrir de camuflajes que provengan tanto de las máscaras de color como la de profundidad. De acuerdo al método para combinar ambas máscaras, puede ocurrir que mientras que en la máscara de profundidad hay camuflajes, en la máscara de color no las haya. Detectar cuando ocurre esto no es tarea fácil, pues en otras ocasiones dicho camuflaje puede ser falseado por la presencia de una sombra, por ejemplo.

Por otro lado, hay tareas como la actualización del fondo de profundidad que son específicas de este tipo de aplicación. La necesidad de actualizar surge precisamente por intentar detectar los objetos con profundidades pequeñas; es decir, objetos que están muy cerca del fondo. Esta tarea no sería necesaria, por ejemplo, en aplicaciones, donde las condiciones del entorno están bien controladas (sin luz solar, por ejemplo) y las distancia entre el fondo y frente es considerablemente alta, pues en este caso los márgenes de error son muy inferiores en comparación con dichas distancias y no hay lugar posibles ambigüedades.

El sensor Kinect proporciona información muy fiable bajo ciertas condiciones del entorno y bastaría con la información de profundidad para clasificar un determinado evento. Por ejemplo, si se supone una escena donde el modelo de fondo tiene una profundidad válida en todos sus píxeles y en la que se quiere detectar el robo o abandono de unas cajas de unas dimensiones suficientes como para detectar su profundidad, entonces un análisis basada en información de profundidad sería suficiente, además de ser más barato y rápido computacionalmente.

En aplicaciones de otra índole como el conteo de personas que entran y salen de un recinto, por ejemplo, el sensor Kinect puede funcionar con bastante fiabilidad. En la figura 6.1 se ve, por ejemplo, que si la superficie del marco del puerta devuelve siempre una profundidad válida, entonces bastaría trabajar solamente con las imágenes de profundidad para tener permanentemente custodiada la entrada.

Dada la alta de densidad y precisión de la información del sensor Kinect con personas,

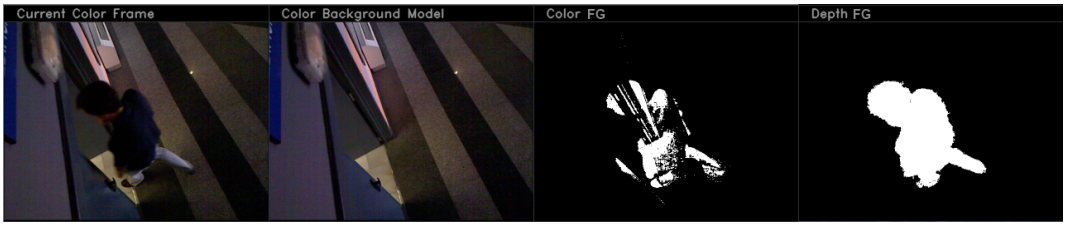



Figura 6.1: Conteo de personas que acceden a un recinto

su uso se puede extender también al análisis de interacciones entre personas. Además, en este caso, el problema de *blobs* solapados es menos crítico, ya que al disponer de la información de profundidad se puede hacer una distinción entre ellos, como se puede observar en la figura 6.2.

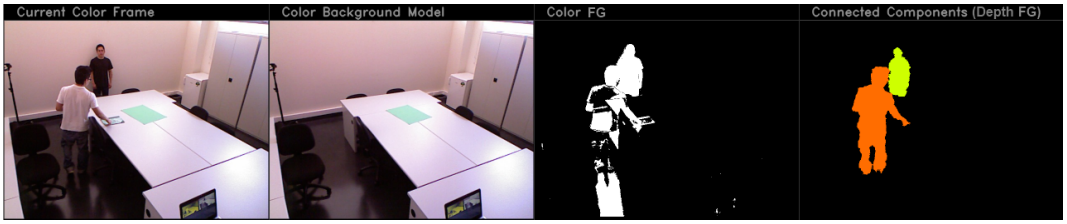



Figura 6.2: Interacción entre personas

## 6.4. Trabajo futuro

El trabajo desarrollado a lo largo de este proyecto ha estado enfocado esencialmente a buscar una manera óptima de combinar la información de color y la información de profundidad. Y esto se ha conseguido integrando la información de profundidad en las etapas de detección del primer plano (sección 4.3) y discriminación de objetos (sección 4.5). El método elegido para combinar ambos tipos de información, explicado en la sección 4.4, es uno de los muchos posibles. En este caso, el análisis principalmente se ha realizado sobre imágenes de profundidad y se han elegido las imágenes de color como apoyo a éstas. Sin embargo, queda aún por investigar el proceso inverso; es decir, utilizar las imágenes de profundidad como apoyo a las imágenes de color. Una posible vía consistiría en ajustar los parámetros de segmentación o del procesado de sombras en función de la máscara del frente obtenida con profundidad. Esto sería útil sobre todo para definir bien los bordes de los objetos, aislándolos de las sombras que producen. Por ejemplo, si una región que aparece como activa en el *foreground* de color, pero no aparece en el *foreground* de profundidad, entonces es más probable que corresponda a una sombra; en tal caso, los parámetros para el procesado de sombras serían menos tolerantes.

Si se sigue por el mismo camino para combinar la información de color y profundidad (sección 4.4), entonces el trabajo a realizar se debería centrar en conseguir que todos los *blobs* de la máscara de color (que no aparecen en la de profundidad) se reflejen en el *foreground* final. En la implementación actual, no pasarán aquellos *blobs* pertenecientes a objetos grandes y planos, ni aquellos (planos también) solapados por *blobs* grandes.

Para la actualización del modelo de fondo se propone una aproximación que, en principio, es útil solamente para cambios bruscos de iluminación (i.e. encendido de luces). Cuando un cambio de iluminación brusco ocurre, la máscara de *foreground* de color presenta un exceso de pixeles activos repentinos, respecto a los de la máscara de *foreground* de profundidad, que permanece sin alteraciones. Aprovechando esta circunstancia se puede actualizar todo el modelo de fondo, excepto en las zonas donde hay frente, que posteriormente serán actualizados de forma progresiva. Este mismo enfoque puede ser estudiado para cambios de iluminación progresivos.

Por último, en este trabajo se han analizado imágenes con escenas estáticas; es decir, que el objetivo de la cámara siempre permanece en el mismo sitio. Sin embargo, para analizar un escenario más amplio, que no sea solamente un recinto cerrado, se puede colocar el sensor Kinect sobre un vehículo que recorra todas las zonas de interés. Esto es precisamente lo que el grupo HARL [73] tiene en sus *Datasets.* En una de las secuencias disponibles para detección de eventos de robo/abandono, el sensor Kinect, adosado a un robot móvil, recorre un pasillo, cubriendo así todas las áreas a las que el sensor no podría alcanzar desde un punto fijo. El desafío aquí es claro: conseguir que nuestro sistema también funcione en esas circunstancias. El modo de proceder aquí parece ser modelar en fondo en modo panorámico para cada posición del robot. Cada vez que el robot volviese a esa posición, se intentaría buscar dentro del modelo la correspondencia más apropiada (por reconocimiento de patrones, por ejemplo) a la imagen actual entregada por el sensor Kinect. Seguidamente, el procedimiento para detectar el evento sería similar al desarrollado en este proyecto.

# Bibliografía


[1] C.S. Regazzoni. K.N. Plataniotis. Visual-centric surveillance networks and services. In *IEEE Signal Processing Magazine*, pages 22(2):12–15, 2005.

[2] G. Gera S. Ferrando and C. Regazzoni. Classification of unattended and stolen objects in videosurveillance system. In *In Video and Signal Based Surveillance, 2006. AVSS '06. IEEE International Conference on*, page 21, Nov 2006.

[3] Pi Robot. `http://www.pirobot.org/blog/0017/`.

[4] The Kinect Effect. `http://www.xbox.com/en-US/kinect/kinect-effect`.

[5] Darrell Pentland Wren, Azarbayejani. Real-time tracking of the human body. In *PAMI*, 1997.

[6] W. E. L. Stauffer, C; Grimson. *Adaptive background mixture models for realtime tracking*. Proc. of CVPR, 1999.

[7] Steiger O. Ebrahimi T. Cavallaro, A. Semantic vídeo análisis for adaptive content delivery and automatic description. In *IEEE Transactions of Circuits and Systems for Vídeo Technology*, pages 15(10):1200–1209, Octubre 2005.

[8] S.M. Cheng, S.; Xingzhi Luo; Bhandarkar. A multiscale parametric background model for stationary foreground object detection. In *Proc. of Motion and Vídeo Computing*, page 8 pp, 2007.

[9] J.; Orrite C. Martínez, J.; Herrero. Automatic left luggage detection and tracking using a multi-camera ukf. In *Proc. of PETS*, pages 59–66, 2006.

[10] S.M. Cheng, S.; Xingzhi Luo; Bhandarkar. A multiscale parametric background model for stationary foreground object detection. In *Proc. of Motion and Vídeo Computing*, page 8 pp, 2004.

[11] H. Huwer, S.; Niemann. Adaptive change detection for real-time surveillance applications. In *Proc. of Visual Surveillance*, pages 37–46, 2000.

[12] Z.; Zhang J. Mathew, R.; Yu. Detecting new stable objects in surveillance vídeo. In *Annals of the BMVA Vol. 2010, No. 3*, pages 1–16, 2010.

[13] J.A.; Pushee I.H. Guler, S.; Silverstein. Stationary objects in multiple object tracking. In *Proc. of AVSS 2007*, pages 248–253, 2007.

[14] J-Y.; Chen L-G. Liao, H-H.; Chang. A localized approach to abandoned luggage detection with foreground - mask sampling. In *Proc. of AVSS*, pages 132–139, 2008.

[15] Y.; Haga-T. Porikli, F.; Ivanov. Robust abandoned object detection using dual foregrounds. *Journal on Advances in Signal Processing, art. 30*, page 11 pp, November 2008.

[16] M. Harville T. Darrell, G. Gordon and J. Woodfill. Integrated person tracking using stereo, color, and pattern detection. *International Journal of Computer Vision, 37:175185, 2000. 10.1023/A:1008103604354.*

[17] M. Harville T. Darrell, G. Gordon and J. Woodfill. Robust people detection by fusion of evidence from multiple methods in image analysis for multimedia interactive services. *WIAMIS '08. Ninth International Workshop on*, page 55, May 2008.

[18] V. et al. Fernandez-Carbajales. Robust people detection by fusion of evidence from múltiple methods. *Proc of Int Workshop on image Analysis for Multimedia Interactive Services*, pages 55–58, 2008.

[19] A. Hampapur Y.-L. Tian L. Brown J. Connell, A.W. Senior and S. Pankanti. Detection and tracking in the ibm peoplevision system. In *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on, volume 2*, pages 1403–1406, June 2004.

[20] J.C. San Miguel and J.M. Martínez. Robust unattended and stolen object detection by fusing simple algorithms. In *In Advanced Video and Signal Based Surveillance, 2008. AVSS '08. IEEE Fifth International Conference on*, pages 18–25, Sept 2008.

[21] Andrew Witkin Michael Kass and Demetri Terzopoulos. Active contour models. international journal of computer vision. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, pages 1:321331, 1988. 10.1007/BF00133570.

[22] L. Caro J.C. SanMiguel and J.M. Martínez. Pixel-based colour contrast for abandoned and stolen object discrimination in video surveillance. In *Electronics Letters*, pages 86–87, 2012.

[23] P. Besl. Active optical range imaging sensors. In *Advances in Machine Vision*, pages 1–63, 1989.

[24] B. Curless. Overview of active vision techniques. In *SIGGRAPH 99 Course on 3D Photography*, 1999.

[25] D. Poussart and D. Laurendeau. 3-d sensing for industrial computer vision. In *in Advances in Machine Vision, chapter 3*, pages 122–159, 1989.

[26] T. C. Strand. Optical three-dimensional sensing for machine vision. In *Optical Engineering*, pages 24(1):33–40, 1985.

[27] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *IJCV*, pages 47(1):7–42, 2002.

[28] U. Dhond and J. Aggarwal. Structure from stereo-a review. In *IEEE Transactions on Systems, Man and Cybernetics*, 1989.

[29] Epipolar geometry. http://en.wikipedia.org/wiki/Epipolar_geometry.

[30] P. Besl. Active optical range imaging sensors. In *Machine Vision, chapter 1*, pages 1–63, 1989.

[31] J. Davis and X. Chen. A laser range scanner designed for minimum calibration complexity. In *In Third International Conference on 3D Digital Imaging and Modeling*, 2001.

[32] Seungkyu Lee. Depth camera image processing and applications. *Samsung Advanced Institute of Technology*.

[33] S. Lee K. Lee J.D.K. Kim B. Kang, SJ. Kim and CY. Kim. Harmonic distortion free distance estimation in tof camera. *SPIE EI*, 2011.

[34] J. Mure-Dubois and H. Hugli. Real-time scattering compensation for time-of-flight camera. In *CVS*, 2007.

[35] G. Alenya S. Foix and C. Torras. Lock-in time-of-flight (tof) cameras: A survey. *IEEE Sensors Journal*, 11(9):1917–1926, 2011.

[36] A. Hermanski S. Hussmann and T. Edeler. Realtime motion artifact suppression in tof camera systems. *IEEE Transactions on Instrumentation and Measurement*, 60(5):1682–1690, may 2011.

[37] Richard Szeliski. Computer vision: Algorithms and applications. Springer, 2010.

[38] O. Hall-Holt and S. Rusinkiewicz. Stripe boundary codes for realtime structured-light range scanning of moving objects. In *ICCV*, pages 359–366, 2001.

[39] K. Sato S. Inokuchi and F. Matsuda. Range-imaging for 3d object recognition. In *ICPR*, pages 806–808, 1984.

[40] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *CVPR*, 2003.

[41] Tao Peng. Algorithms and models for 3-d shape measurement using digital fringe projections. Ph.D. Dissertation, University of Maryland, USA. 2007.

[42] Peter Kuhmstedt Ingo Schmidt Matthias Heinze Gunther Notni Christian Bräuer-Burchardt, Andreas Breitbarth. Fringe projection based high-speed 3d sensor for real-time measurements.

[43] Yakov Frayman Asim Bhatti, Saeid Nahavandi. 3d depth estimation for visual inspection using wavelet transform modulus maxima. *Computers and Electrical Engineering*, 33:48–57, January 2007.

[44] Kinect Components. http://www.waybeta.com/news/58230/microsoft-kinect-somatosensory-gamedevice-full-disassembly-report-_microsoft-xbox.

[45] Resources and documentation of Kinect. http://www.microsoft.com/en-us/kinectforwindows/develop/resources.aspx.

[46] Technical aspects of the Kinect device and its calibration. http://www.ros.org/wiki/kinect_calibration/technical.

[47] Kinect Depth vs. Actual Distance. http://mathnathan.com/2011/02/depthvsdistance/.

[48] Kinect Depth vs. Actual Distance. http://futuretheater.net/wiki/Kinect_Workshop.

[49] Structured light 3D scanner: limitations. http://en.wikipedia.org/wiki/Structured-light_3D_scanner#Limitations.

[50] Eron Steger and Kiriakos N. Kutulakos. *A Theory of Refractive and Specular 3D Shape by Light-Path Triangulation*. Int. J. Computer Vision, vol. 76, no. 1., 2008.

[51] Ashok Veeraraghavan Mohit Gupta, Amit Agrawal and Srinivasa G. Narasimhan. *Measuring Shape in the Presence of Inter-reflections, Sub-surface Scattering and Defocus*. Proc. CVPR., 2011.

[52] OpenNI group (from google groups). http://groups.google.com/group/opennidev/browse_thread/thread/a6028d5bacc3fe5f?pli=1.

[53] Primesense Natrual Interaction. http://www.primesense.com.

[54] OpenKinect. http://openkinect.org/wiki/Main_Page.

[55] Kinect operation. http://www.ros.org/wiki/kinect_calibration/technical.

[56] Kinect calibration. http://nicolas.burrus.name/index.php/Research/KinectCalibration.

[57] Alessandro Leone Cosimo Distante, Giovanni Diraco. Active range imaging dataset for indoor surveillance. In *Proc. of Multimedia Signal Processing*, pages 1–4, 2005.

[58] Ariel Shamir Daniel Cohen-Or Meir Johnathan Dahan, Nir Chen. Combining color and depth for enhanced image segmentation and retargeting. *Springer-Verlag 2011*.

[59] Josep R. Casas b Johannes R. Sveinsson c Sigurjon Arni Gudmundsson, Montse Pardas b. Improved 3d reconstruction in smart-room environments using tof imaging. *Computer Vision and Image Understanding*, 2008.

[60] M. Harville J. Woodfill G. Gordon, T. Darrell. Background estimation and removal based on range and color. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 1999.

[61] Marc Van droogenbroeck Sebastien Pierard, Jerome Leens. Techniques to improve the foreground segmentation with a 3d camera and a color camera. *INTELSIG, Laboratory for Signal and Image Exploitation*, 2009.

[62] I. Tashev. Recent advances in human-machine interfaces for gaming and entertainment. In *Information Technology and Security vol. 3, no. 3*, pages 69–76, 2011.

[63] Pierre Moulin Bingbing Ni, Gang Wang. Rgbd-hudaact: A color-depth video database for human daily activity recognition. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1147 – 1153, 2011.

[64] Pietro Azzari Alessandro Bevilacqua, Luigi Di Stefano. People tracking using a time-of-flight depth sensor. *ARCES - DEIS (Department of Electronics, Computer Science and Systems)*, 2006.

[65] Chia-Chih Chen Lu Xia and J. K. Aggarwal. Human detection using depth information by kinect. *The University of Texas at Austin: Department of Electrical and Computer Engineering*, 2011.

[66] Kinect Hacks. `http://www.kinecthacks.net/`.

[67] O. Steiger A. Cavallaro and T. Ebrahimi. Semantic video analysis for adaptive content delivery and automatic description. *Circuits and Systems for Video Technology, IEEE Transactions on*, pages 15(10):1200–1209, Oct 2005.

[68] Massimo Piccardi Rita Cucchiara, Costantino Grana and Andrea Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 25:1337–1342, 2003.

[69] P. Salembier and J. Ruiz. On filters by reconstruction for size and motion simplifcation. In *In Proc. of Int. Symposium in Mathematical Morphology*, pages 425–434, 2002.

[70] José M. Martínez Álvaro Bayona, Juan C. SanMiguel. Stationary foreground detection using background subtraction and temporal difference in video surveillance. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 4657–4660, Hong Kong (China), 2010.

[71] G.; Regazzoni C. Ferrando, S.; Gera. Classification of unattended and stolen objects in video-surveillance system. In *Proc. of AVSS*, pages 21–27, 2006.

[72] OpenNI. `http://www.openni.org/`.

[73] HARL (Human Activities Recognition and Localization). `http://liris.cnrs.fr/harl2012/`.

# Apéndice A

# Entornos de desarrollo: OpenNI y MSDK

## A.1. OpenNI (Primesense/NITE)

- **Instalación (para Windows 7, 32 bits, *stable release*)**
  - Pasos a seguir (son necesarios 4 ficheros por separado):
    - Descargar el *driver* para el Sensor Kinect en en siguiente enlace: *https://github.com/avin2/SensorKinect*. Buscar el controlador adecuado en la carpeta *bin*, en nuestro caso, uno para plataformas de 32 bits *(SensorKinect-Win-OpenSource32-5.0.3.4.msi)*. Nota: las versiones de los ficheros cambian constantemente, por lo que no necesariamente los nombres de ficheros que aquí se mencionan tienen que ser los mismos.
    - Descargar el módulo *OpenNI binaries (OpenNI-1.4.0.2)* desde el sitio *web* official: *http://www.openni.org/Downloads/OpenNIModules.aspx*
    - Descargar el módulo *OpenNI compliant hardware binaries (Sensor-5.0.5.1)* del mismo sitio.
    - Descargar el módulo *OpenNI compliant middleware binaries (NITE-1.5.0.1)* del mismo sitio. NITE (*Natural Interaction Technology for End-user*) es el *middleware* que traduce, basándose en OpenNI, el mundo real a datos significativos, que permiten, por ejemplo, identificar usuarios y rastrearlos.
    - Instalar los ficheros anteriores en el orden descargado en el directorio deseado.
    - En caso de haber instalado anteriormente un SDK diferente, cambiar el controlador de la kinect (sólo las cámaras):

        - ◇ Ejecutar el administrador de dispositivos (Win+R -> «devmgmt.msc»)
        - ◇ Buscar el disposivo *Microsoft Kinect*, desplegar sus componentes y seleccionar *Microsoft Kinect Camera*.
        - ◇ Click derecho -> «Actualizar software del controlador...» -> «Buscar software de controlador en el equipo».
        - ◇ El el cuadro dialogo, asegurarse de que el *checkbox* «Mostrar el hardaware compatible» está seleccionado. Seleccionar el nuevo controlador de la lista.

- **Configuración del entorno (Microsoft Visual Studio 2010)**

    - ***Libs***
        - ○ Con el proyecto seleccionado (en el lado izquierdo), clicamos sobre *project -> references... -> configuration properties -> VC++ directories*.
        - ○ En el cuadro de diálogo, añadir «C:\Program Files\PrimeSense\NITE\Lib» y «C:\Program Files\OpenNI\Lib» al campo *Lybrary Directories*.
        - ○ En el cuadro de dialogo *Linker->input* añadir «openNI.lib» en *Aditional Dependencies*.
    - ***Includes***
        - ○ En la ventana de diálo anterior (*project -> references... -> configuration properties -> VC++ directories*) añadir «C:\Program Files\PrimeSense\NITE\Include» y «C:\Program Files\OpenNI\Include» al campo *Include Directories*.
        - ○ Aceptamos y listo.
    - ***DLLs***
        - ○ Asegurarse de que en la variable de entorno *path* están los siguientes directorios: «C:\Program Files\OpenNI\Bin» y «C:\Program Files\PrimeSense\NITE\bin»
        - ○ Para ello, ejecutamos el siguiete comando «sysdm.cpl» (Win + R -> «sysdm.cpl»). El ventana que aparece, seleccionamos la pestaña «Opciones avanzadas» y luego pulsamos sobre el boton «variables de entorno». Editar la variable deseada (*path*).

  **Nota:** los directorios anteriores son sólo válidos si en la instalación se eligieron los directorios por defecto.

- **Pros**

    - Licencia para uso comercial.

- Incluye un *framework* para el *tracking* de manos.
- Incluye un *framework* para el *tracking* gestos con manos.
- **Alineamiento automático de las imagenes de color y profundidad**.
- *Cropping* para trabajar sobre una cierta area de un determinado *frame*.
- *Tracking* completo del cuerpo.
  - Calcula, además, las rotaciones de las articulaciones.
- Soporte para múltiples sensores (entre ellos el sensor de profundidad *PrimeSense/ASUS WAVI Xtion*).
- Válido para Windows (incluyendo al Vista y XP), Linux y MacOSX.
- Permite grabar/reproducir al/del disco.
- Hay eventos cuando un nuevo usuario entra en la escena o sale de ella.
- Existe un «escaparate» (http://arena.openni.org/) para exponer aplicaciones propias, basadas en *OpenNI*.

- **Contras**
  - No hay controladores para el audio.
  - No hay controladores para el motor.
  - El el *tracking* del cuerpo las articulaciones ocultas no se estiman.
  - La configuración para multiples sensores no es inmediata.
  - No hay un único instalador. Se necesitan cuatro.

## A.2. MSDK (Microsoft Software Development Kit)

- **Instalación (para Windows 7 u 8, 32 bits)**
  - Requisitos hardware:
    - Procesador Dual-core 2.66-GHz.
    - Bus USB 2.0 dedicado.
    - 2 GB de memoria RAM
    - Sistema operativo: Windows 7 o Windows 8
  - Requisitos software:
    - *Microsoft® Visual Studio® 2010 Express* u otra edición de *Visual Studio 2010*

- *.NET Framework 4.0*

- Pasos a seguir para la instalación:
  - Descargar el SDK (32 bits) de la página oficial: *http://www.microsoft.com/en-us/kinectforwindows/develop/developer-downloads.aspx*
  - Doble click sobre el fichero descargado y seguir las indicaciones hasta finalizar la instalación. Para realizar este paso, el Sensor Kinect debe estar desconectado del PC. Asimismo, cualquier versión anterior del SDK, debe haber sido desinstalada.
  - Conectar el sensor Kinect y los *drivers* se cargarán automáticamente. En este momento el SDK debería estar listo para usarse. Si se tuviese abierto el entorno Visual Studio durante la instalación, cerrarlo y volverlo abrir, para que se actualizen las variables de entorno (KINECTSDK_DIR) necesarias para la Kinect.
- Se puede encontrar más detalles en: *http://www.microsoft.com/en-us/kinectforwindows/develop/*

- **Configuración del entorno (Microsoft Visual Studio 2010).** Antes de empezar con la configuración del entorno debemos tener creado un proyecto en *Microsoft Visual C++ 2010 (Express).* Este entorno (la versión *express*) se puede descargar de manera gratuita en: *http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-cpp-express.* Con el proyecto creado, procedemos a añadir las referencias necesarias:

  - ***Libs***
    - Con el proyecto seleccionado (en el lado izquierdo), clicamos sobre *project -> references... -> configuration properties -> VC++ directories*.
    - En el cuadro de diálogo, añadir «C:\Program Files\Microsoft SDKs\Kinect\v1.0 Beta2\lib\x86» al campo *Lybrary Directories.*
    - En el cuadro de dialogo *Linker->input* añadir la librería «MSRKinectNUI.lib» en el campo *Aditional Dependencies.*
  - ***Includes***
    - En la ventana de diálogo anterior (*project -> references... -> configuration properties -> VC++ directories*) añadir «C:\Program Files\Microsoft SDKs\Kinect\v1.0 Beta2\inc» al campo *Include Directories.*
    - Aceptamos y listo.

**Nota:** los directorios anteriores son sólo válidos si en la instalación se han elegido los directorios por defecto.

- **Pros**
  - *Driver* para usar los microfonos.
  - *Driver* para controlar el motor (su inclinación).
  - *Tracking* completo de todo el cuerpo.
    - No necesita una pose para calibrar.
    - Incluye la cabeza, manos, pies y clavicula.
    - Estimación para las articulaciones ocultas.
  - Soporta **multiples sensores** a la vez.
  - Instalador único.
  - Eventos para cuando un nuevo *frame* de video o profundidad está disponible.
  - Resolución de imagen de color de hasta **1280x1024** (~1.3 Mpix)
- **Contras**
  - Solo *«traquea»* el cuerpo entero. No hay modo para *«traquear»* solo las manos.
  - **No ofrece alineamiento de las imagenes** de color y profundidad
    - Aunque hay metodos que permiten alinear cada pixel por individual.
  - En el *Tracking* completo de todo el cuerpo:
    - Solo calcula las posiciones de las articulaciones, no rotaciones.
    - Siempre *«traquea»* todo el cuerpo, no hay opción para elgir solo la parte superior del cuerpo o solo las manos.
  - No reconoce gestos
  - SDK válido sólo para el sensor Kinect.
  - Válido solamente para **Windows 7 y Windows 8** (32 y 64 bits).
  - No se puede grabar/reproducir al/del disco.
  - No hay eventos para cuando un nuevo usuario entra en la escena o sale de ella.

# Apéndice B

# Integración de OpenCV con SDKs

OpenCV es una biblioteca de funciones de programación para visión computacional en tiempo real. Permite manipular imagenes de una gran variedad de formatos así como applicar sobre ellas los algoritmos más comunes del tratamiento de imágenes.

## B.1. Instalación y configuración del entorno

Antes de integrar OpenCV (v2.3) en nuestro proyecto debemos descargarnos los ficheros necesarios. Para ello:

- Descargargamos el *superpack* de OpenCV 2.3 (*OpenCV-2.3.0-win-superpack.exe*) en la página oficial: *http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.3/*. Éste contendrá las librerías necesarias ya compiladas para Miscrosoft Visual Studio 2010.
- Doble click sobre el archivo descargado y lo instalamos siguiendo las indicaciones. Se nos habrá creado una carpeta en el directorio especificado con todos los archivos necesarios para integrarlo en nuestro proyecto.

Una vez instalado el paquete de OpenCV (v2.3), vamos a nuestro proyecto (el que creamos en la configuración del entorno en la sección A.1) y referenciamos los archivos necesarios de la siguiente manera:

- ***Libs***
  - Con el proyecto seleccionado (en el lado izquierdo), clicamos sobre *project -> references... -> configuration properties -> VC++ directories*.
  - En el cuadro de diálogo, añadir «C:\OpenCV2.3\build\x64\vc10\lib» al campo *Lybrary Directories*.

- En el cuadro de dialogo *Linker->input* añadir «opencv_core230.lib, opencv_highgui230.lib, opencv_video230.lib, opencv_ml230.lib, opencv_legacy230.lib, opencv_imgproc230.lib» en *Aditional Dependencies*.

- ***Includes***

  - En la ventana de diálogo anterior (*project -> references... -> configuration properties -> VC++ directories*) añadir «C:\OpenCV2.3\build\include\opencv» y «C:\OpenCV2.3\build\include» al campo *Include Directories*.
  - Aceptamos y listo.

- ***DLLs***

  - Asegurarse de que el la variable de entorno *path* están los siguientes directorios: «C:\OpenCV2.3\build\x64\vc10\bin» y «C:\OpenCV2.3\build\bin».
  - Para ello, ejecutamos el siguiete comando «sysdm.cpl» (Win + R -> «sysdm.cpl»). El ventana que aparece, seleccionamos la pestaña «Opciones avanzadas» y luego pulsamos sobre el boton «variables de entorno». Editar la variable deseada (*path*).

**Nota:** los directorios anteriores son sólo válidos si en la instalación se ha elegido el directorio «C:\OpenCV2.3». Si el objetivo del proyecto es crear una aplicación para plataformas de 32 bits, cambiar «x64» del directorio anterior por «x86».

Más información sobre la instalación así como la configuración del entorno para diferentes plataformas se puede encontrar en: *http://opencv.willowgarage.com/wiki/InstallGuide.*

### B.1.1. OpenCV y MSDK

El objetivo de esta sección es conseguir los datos de profundidad y de color ofrecidos por la Kinect a través de MSDK y convertirlos a imágenes IPL, que son las que puede manipular OpenCV.

Para poder obtener los datos de los sensores de la Kinect (color o profundidad), es necesario, en la aplicación, crear un *stream* de datos, es decir, un *buffer* donde se almacenarán los datos procedentes del sensor correspondiente. El **formato de los datos** almacenados en dicho *buffer* depende del sensor que se esté utilizando:

- **Sensor de profundidad**

  - La información de profundidad se devuelve en un mapa de pixeles (VGA o QVGA).

- Cada pixel representa, teoricamente, la distancia en milimetros desde el sensor hasta el objeto al que pertenece el pixel.
- Cada pixel ocupa 16 bits (**2 bytes**).
- Sólo los 11 bits LSB contienen datos significativos.
- Los datos se devuelven en bytes distribuidos secuencialmente en un array.
- Se obtiene el puntero a esos datos, y ya se puede trabajar con ellos.

Con los *bytes* de profundidad disponibles, es inmediato obtener a una imagen IPL. En OpenCV, crearemos un *header* para una imagen, y la rellenaremos su campo de datos con los bytes de profundidad. El tipo de imagen creado será una imagen de grises de 1 canal y 16 bits por pixel.

- **Sensor de color (cámara RGB)**

  - La información de color se devuelve en un mapa de pixeles, de resoluciones indicadas en la sección 2.4.1.
  - Cada pixel representa el color (en formato RGB o YUV) del objeto al que pertenece el pixel.
  - Cada pixel ocupa 32 bits (**4 bytes**).
  - Sólo los 24 bits LSB contienen datos significativos, siguiendo el formato 0xXXRRGGBB. Por ejemplo, el rojo sería: 0xXX00ff00.
  - Los datos se devuelven bytes distribuidos secuencialmente en un array.
  - Se obtiene el puntero a esos datos, y ya se puede trabajar con ellos.

  En este caso, el tipo de imagen creado será una imagen de color de 3 canales y 8 bits por cada canal, es decir 1 *Byte* por canal. Aquí, el mapeo es practicamente inmediato, salvo porque una imagen IPL tiene los datos en formato BGR. Una simple inversión soluciona el problema. En otras palabras, habrá que recorrer el array devuelto por la Kinect cada 4 *Bytes* e ir extrayendo los tres bytes menos significativos, al tiempo que los introducimos en la zona de datos de la cabecera de la imagen IPL.

### B.1.2. OpenCV y OpenNI

El objetivo de esta sección es conseguir los datos de profundidad y de color entregados por la Kinect a través de OpenNI y convertirlos a imágenes IPL, que son las que puede manipular OpenCV.

Como se explicó en la sección 3.2 B.1.1, asumimos que los streams de datos ya has sido creados. Pasamos ahora a analizar el formato de los datos devueltos por los sensores en los *buffers:*

- **Sensor de profundidad**
  - El formato de los datos de profundidad y su manipulación es exactamente igual que en el descrito anteriormente, en la sección B.1.1.
- **Sensor de color**
  - Esta vez, cada pixel ocupa 24 bits (**3 bytes**), siguiendo el formato RGB
  - Los datos se devuelven bytes distribuidos secuencialmente en un array.
  - Se obtiene el puntero a esos datos, y ya se puede trabajar con ellos.

  Aquí, tambien se procedería igual que en el caso anterior, salvo porque el recorrido del array de *Bytes* del *buffer* de color se realiza de 3 en 3.

# Apéndice C

# Presupuesto

1. **Ejecucion Material**
   - Compra de ordenador personal (Software incluido) ..... 2.000 €
   - Alquiler de impresora láser durante 6 meses ..... 260 €
   - Material de oficina ..... 150 €
   - Total de ejecución material ..... 2.400 €
2. **Gastos generales**
   - 16 % sobre Ejecucion Material ..... 352 €
3. **Beneficio Industrial**
   - 6 % sobre Ejecucion Material ..... 132 €
4. **Honorarios Proyecto**
   - 1800 horas a 15 €/ hora ..... 27.000 €
5. **Material fungible**
   - Gastos de impresión ..... 280 €
   - Encuadernación ..... 200 €
6. **Subtotal del presupuesto**
   - Subtotal Presupuesto ..... 32.774 €
7. **I.V.A. aplicable**

- 21 % Subtotal Presupuesto ..... 6.882,5 €

8. **Total presupuesto**

- Total Presupuesto ..... 38.673,3 €

Madrid, Diciembre del 2012

El Ingeniero Jefe de Proyecto

Fdo.: Fabricio Córdova Lucero

Ingeniero Superior de Telecomunicación

# Apéndice D

# Pliego de condiciones

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un sistema para la detección de objetos robados/abandonados en interiores utilizando cámaras de profundidad. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

**Condiciones generales**

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero

Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4 % del presupuesto y la provisional del 2 %.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea

debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad “Presupuesto de Ejecución de Contrata” y anteriormente llamado ”Presupuesto de Ejecución Material” que hoy designa otro concepto.

### Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.