UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

SEGUIMIENTO DE UNA PERSONA CON ROBOT DE SERVICIO DE RECURSOS LIMITADOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

EDUARDO ANDRÉS HORMAZÁBAL CORREA

PROFESOR GUÍA:
JAVIER RUIZ DEL SOLAR SAN MARTÍN

MIEMBROS DE LA COMISIÓN:
ANDRÉS CABA RUTTE
MAURICIO CORREA PÉREZ

SANTIAGO DE CHILE
2019

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: EDUARDO ANDRÉS HORMAZÁBAL CORREA
FECHA: 2019
PROF. GUÍA: JAVIER RUIZ DEL SOLAR SAN MARTÍN

SEGUIMIENTO DE UNA PERSONA CON ROBOT DE SERVICIO DE RECURSOS LIMITADOS

Los robots de servicio son cada vez más frecuentes dado su potencial para realizar diversos tipos de tareas. Una actividad con gran potencial es la de seguir a una persona, ya que se puede utilizar para ejecutar acciones más complejas, como aprender un recorrido predeterminado, ayudar a transportar objetos, monitorear a una persona, etc. Luego, el objetivo de este trabajo es el de implementar un sistema de seguimiento de una persona con el robot de servicio Pepper de *SoftBank Robotics*, que implica ser capaz de funcionar en tiempo real, con la presencia de múltiples personas, entre otros.

Tras investigar en la literatura distintos métodos relacionados con este trabajo, se propone e implementa un sistema modular que basa su funcionamiento en imágenes de color y profundidad adquiridas por el robot. Primero se obtienen candidatos de personas utilizando la nube de puntos generada por la imagen de profundidad para reducir el costo computacional, y luego se detectan utilizando la imagen de color. Posteriormente se asocian las detecciones de todas las personas a lo largo del *stream* de video. A continuación se entrena un clasificador adaptativo capaz de diferenciar a la persona que se está siguiendo de las demás, mediante un conjunto de características elegidas previamente. Por último, se tiene un módulo encargado de accionar al robot para realizar la rutina de movimiento, que permite seguir a la persona.

El sistema se evaluó en un conjunto de videos generados específicamente para la tarea de seguimiento. Los resultados obtenidos indican que el sistema sí se puede ejecutar en tiempo real, con una tasa aproximada de 12 fps. Por otro lado, la detección de la persona a seguir alcanzó una media del 90.49 %, además de ser capaz de reencontrarla la mayoría de las veces con una tasa promedio del 92.01 %. Estos resultados compiten con métodos tradicionales utilizados en la literatura, pero que no logran ejecutarse en tiempo real en una máquina de las mismas características a la utilizada en este trabajo.

Se concluye que el sistema cumple con los objetivos planteados, y también da la posibilidad de realizar una mejora continua de éste por la naturaleza modular de la solución propuesta. Aún así, existe un conjunto de problemáticas no resueltas que se esperan resolver con métodos modernos y hardware especializado, como por ejemplo las redes neuronales convolucionales.

Para el pequeño Eduardo del pasado.

Agradecimientos

En primer lugar, quiero agradecer a mi familia, ya que sin ellos no sería la persona que soy. Valoro mucho el apoyo recibido, y las muestras de confianza que me dan día a día.

A mi profesor guía, que me introdujo al maravilloso mundo del procesamiento de imágenes, y me permitió trabajar en este proyecto.

A los miembros del laboratorio de robótica, por el buen ambiente que generan. A Luz, que me dio el empuje necesario para este trabajo. Y por supuesto a Enma, por las incontables veces en las que me dio una mano.

A mis amigos del pasado y presente, por todos esos buenos ratos, las largas caminatas, las comilonas, esas conversaciones intensas, y las no tanto. También por los malos ratos, pues así esto se hace más interesante; siempre se puede aprender algo. Y no, no los mencionaré a todos; están en la mente.

Por último, gracias a todas esas personas agradablemente desconocidas. Esos saludos de simpatía, los diálogos con gente del metro o la micro, los deseos de suerte, las historias, las oportunidades otorgadas. Cada uno de esos detalles es importante, por lo menos para mí.

Por lo anterior y mucho más: Gracias, de verdad.

Tabla de Contenido

Índice de Ilustraciones

Índice de Algoritmos

Capítulo 1

Introducción

1.1. Motivación

La inclusión de robots como facilitadores de tareas se remonta a la edad industrial, en los procesos de manufactura en grandes fábricas. Luego se incorporaron en otras áreas, como por ejemplo la medicina [4] y los robots de servicio [5]. Estos últimos se pueden especializar en diversas tareas dependiendo de las necesidades del operador, lo que los convierte en una amplia fuente de investigación y desarrollo.

En particular, la Universidad de Chile cuenta con el robot de servicio Pepper de *SoftBank Robotics*. Pepper es un robot humanoide emocional, capaz de reconocer emociones en las personas, saludar, informar y entretener [6]. A su vez, Pepper permite el desarrollo de nuevas funcionalidades gracias a las facilidades que otorga su sistema operativo NAOqi y las herramientas de desarrollo de software que provee.

Una funcionalidad útil para añadir a Pepper es la del seguimiento de una persona a través de diversos ambientes, explotando así la movilidad que el robot ofrece. Así, esta funcionalidad permitiría ayudar al operador a transportar objetos, a vigilar el estado de la persona que sigue (en caso de caídas, por ejemplo), a aprender un recorrido enseñado por el operador, etc. Por lo tanto, desarrollar un sistema de seguimiento de una persona puede ser una base para implementar tareas más complejas en un futuro cercano, donde los robots de servicio sean cada vez más comunes.

El trabajo de título consiste en realizar un seguimiento de una persona (denominada objetivo), en específico mediante el uso de una cámara RGB-D para la percepción de ésta y un sensor Láser como ayuda para la navegación del robot. En un comienzo se detecta a la persona a seguir, de forma de tener una referencia para realizar el matching. Posteriormente se debe realizar el seguimiento al objetivo, considerando diferentes situaciones como por ejemplo la oclusión de éste por objetos móviles o estáticos, o la variación de iluminación en el ambiente.

El sistema de seguimiento se debe incorporar al robot de servicio Pepper de *SoftBank*. De

esta forma se pretende realizar una actuación con el robot para no perder de vista al objetivo. De lo contrario, se intenta detectar nuevamente al objetivo con una rutina de búsqueda.

Para verificar el correcto funcionamiento y robustez del sistema, se generan diferentes situaciones de seguimiento, variando la presencia tanto de oclusiones como de objetos adicionales en el ambiente.

1.2. Objetivos

1.2.1. Objetivos Generales

El objetivo general consiste en realizar un seguimiento de una persona en movimiento con el robot de servicio Pepper de *SoftBank*, utilizando una cámara RGB-D y un sensor Láser. El seguimiento debe ser robusto a la iluminación, breves oclusiones y a la presencia de entidades ajenas a la persona a seguir. Además, el sistema se debe implementar para funcionar con Pepper en tiempo real.

1.2.2. Objetivos Específicos

Se espera implementar un sistema capaz de realizar el tracking de una persona utilizando la información de color y profundidad otorgada por la cámara RGB-D incorporada en el robot.

Esto significa lograr una serie de objetivos específicos:

1. Comprender los conceptos que permiten la correcta navegación autónoma de un robot.
2. Comprender los sensores disponibles y cómo éstos se pueden utilizar para obtener información útil.
3. Dividir el problema en módulos de manera de resolver el problema en fases independientes.
4. Investigar métodos del estado del arte relacionados con el trabajo de título, para poder hacer una correcta planificación de la solución.
5. Implementar la solución en forma de un sistema que ejecute en tiempo real los procesos requeridos.
6. Integración de la solución con el robot basándose en sus especificaciones y características específicas.
7. Comparar los resultados con soluciones similares, ya sea en la globalidad del sistema, o por módulos.
8. Discutir el funcionamiento del sistema y plantear posibles mejoras a futuro.

Capítulo 2

Antecedentes

En este capítulo se entra en contexto con los métodos principales utilizados a lo largo del trabajo de título. Para ello se comienza con una descripción de la representación de color y distancias en el procesamiento de imágenes, luego se mencionan un conjunto de características que se pueden extraer de las imágenes captadas, para terminar con el uso de esas características en clasificadores ampliamente utilizados en la literatura.

2.1. Información de color

El color conocido por el humano se puede definir como una percepción de éste frente a estímulos de luz [7] a diferentes longitudes de onda dentro del espectro visible (ver Figura 2.1). Esta información es ampliamente utilizada en la actualidad, como por ejemplo en la vía pública, con semáforos que regulan el tránsito vehicular de acuerdo al color indicado por la luz.

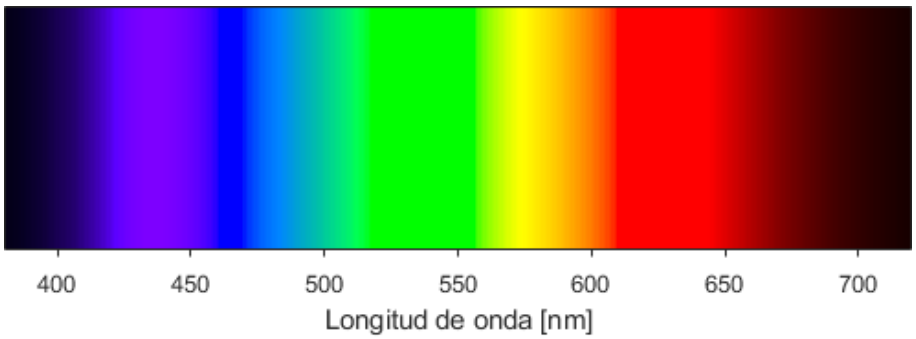


Figura 2.1: Espectro de luz visible.

Particularmente, en procesamiento digital de imágenes, esta información se obtiene mediante un conjunto de sensores agrupados que captan la intensidad y longitud de onda de la luz que reciben. Generalmente se utilizan 3 componentes para representar, según el espacio de color, la composición de un color determinado.

Un espacio de color es un modelo matemático que permite definir un color de acuerdo a las componentes del espacio. Uno de los más conocidos es el modelo RGB, usado para visualización de imágenes por computadora; YUV es utilizado para sistemas de televisión; HSV es más utilizado por artistas; etc. Particularmente en este trabajo se abordan los espacios de color RGB y HSV, pues en la literatura investigada éstas son representaciones útiles en el procesamiento de imágenes.

La imagen de color se almacena en un arreglo bidimensional por cada componente, que representa la distribución espacial. Con esto se puede definir una imagen digital a color como $I(x, y) = \{I_1(x, y), I_2(x, y), I_3(x, y)\}$, donde (x, y) son las coordenadas espaciales que representan a un pixel, y I_{i} es la intensidad en el i-ésimo canal.

2.1.1. Imagen RGB

Este modelo se basa en un sistema de coordenadas cartesianas, donde cada eje corresponde a los colores primarios Rojo, Verde y Azul, mientras que en la diagonal principal del cubo se encuentran los colores de la escala de grises (ver Figura 2.2).

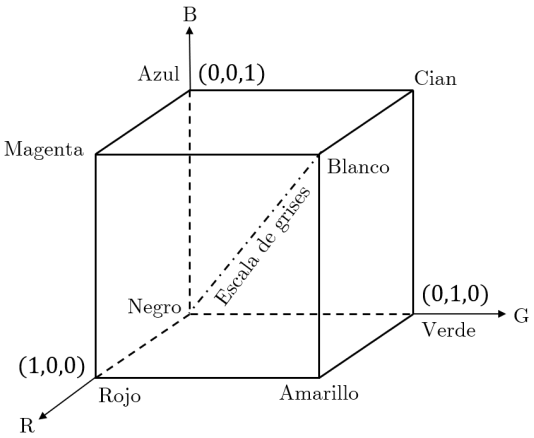


Figura 2.2: Esquemático del cubo de color RGB.

Se caracteriza por ser un modelo aditivo, pues con la suma ponderada de los tres colores se puede obtener un nuevo color. En la Figura 2.3 se muestra la descomposición de una imagen en los tres colores primarios, donde la intensidad de cada uno determina el color resultante.

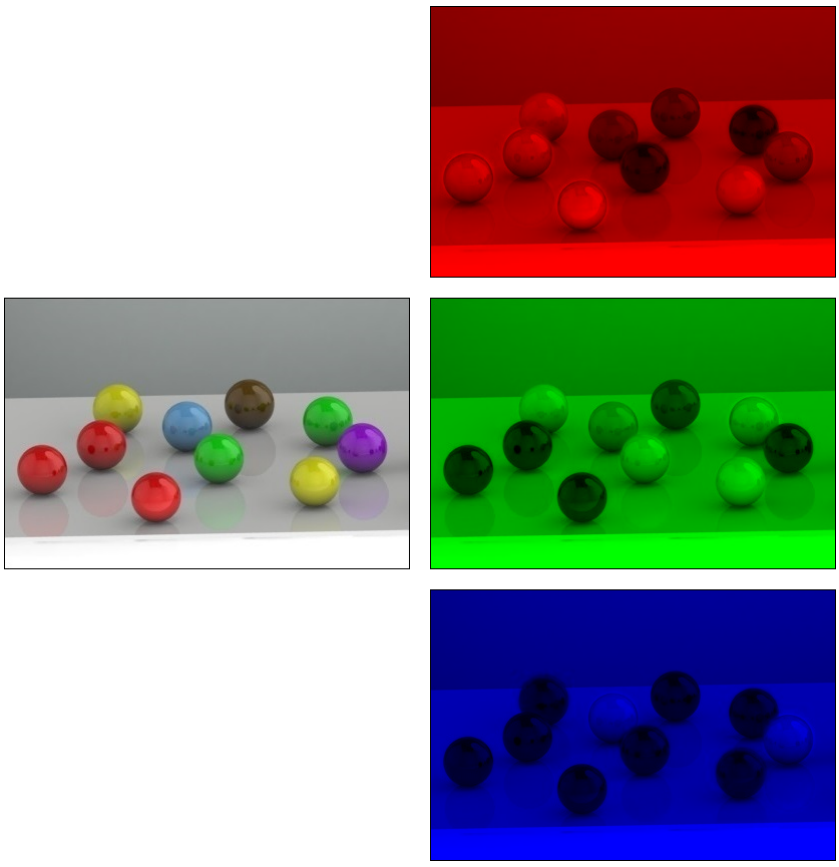


Figura 2.3: Descomposición de imagen (izquierda) en el espacio RGB (derecha). Los canales Rojo, Verde, y Azul corresponden a la imagen superior, centro-derecha, e inferior, respectivamente).

2.1.2. Imagen HSV

Este modelo se basa en un sistema de coordenadas cilíndricas [8], donde la componente angular corresponde al matiz (Hue en inglés); el radio corresponde a la saturación (Saturation); y la altura representa al brillo o valor (Value). En la Figura 2.4 se puede observar la distribución de colores en el espacio HSV.

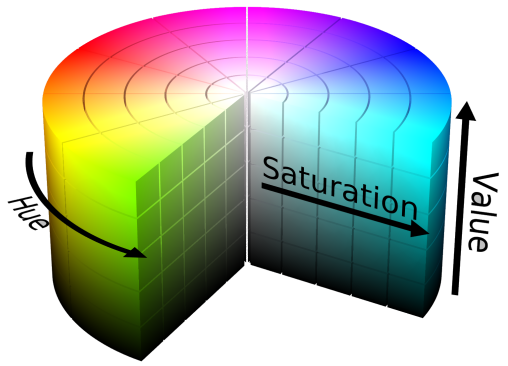


Figura 2.4: Distribución de colores en el espacio HSV.

La representación en HSV tiene como objetivo definir de manera más intuitiva la percepción del color del ojo humano en vez del modelo RGB [7]. Una de las principales ventajas del espacio HSV es que la iluminación se representa sólo por la componente Value, mientras que en el modelo RGB se necesitan los tres componentes.

Asumiendo que las componentes RGB están normalizadas en el rango $[0, 1]$, la relación entre HSV y RGB está definida por un conjunto de ecuaciones [8], como se observa a continuación:

$$C_{max} = max(R, G, B) \tag{2.1a}$$
$$C_{min} = min(R, G, B) \tag{2.1b}$$
$$\Delta = C_{max} - C_{min} \tag{2.1c}$$
$$V = C_{max} \tag{2.1d}$$
$$S = \begin{cases} 0, & \text{si } C_{max} = 0 \\ \frac{\Delta}{C_{max}}, & \text{si } C_{max} \neq 0 \end{cases} \tag{2.1e}$$
$$h = \begin{cases} 180°, & \text{si } C_{max} = 0 \\ 60°(\frac{G-B}{\Delta}), & \text{si } R = C_{max} \\ 60°(2 + \frac{B-R}{\Delta}), & \text{si } G = C_{max} \\ 60°(4 + \frac{R-G}{\Delta}), & \text{si } B = C_{max} \end{cases} \tag{2.1f}$$
$$H = \begin{cases} h, & \text{si } h \geq 0° \\ h + 360°, & \text{si } h \ngeq 0° \end{cases} \tag{2.1g}$$

Donde las componentes resultantes están en los rangos: $H \in [0°, 360°)$, $S \in [0, 1]$, y $V \in [0, 1]$.

2.2. Información de profundidad

La profundidad se puede definir como la distancia entre un punto de referencia y un punto arbitrario en el espacio. Particularmente, en procesamiento de imágenes, existen diversos métodos para obtener esta información del entorno. Por ejemplo el LADAR, SONAR, emisión y recepción de patrones infrarrojos, cámaras estéreo, etc.

De igual forma en que hay múltiples sensores para la adquisición de datos, también hay más de una representación de la información de profundidad. A continuación se dan a conocer la imagen digital de profundidad y la nube de puntos; dos de las representaciones más utilizadas en esta área.

2.2.1. Imagen de profundidad

Una imagen de profundidad es una representación bidimensional que entrega la información relativa a la distancia desde el sensor hasta los objetos visibles por éste. A diferencia

de las imágenes a color, la imagen de profundidad utiliza sólo 1 canal, con lo que puede ser representada con escala de grises o mapa de calor (ver Figura 2.5).

(a) Imagen RGB.

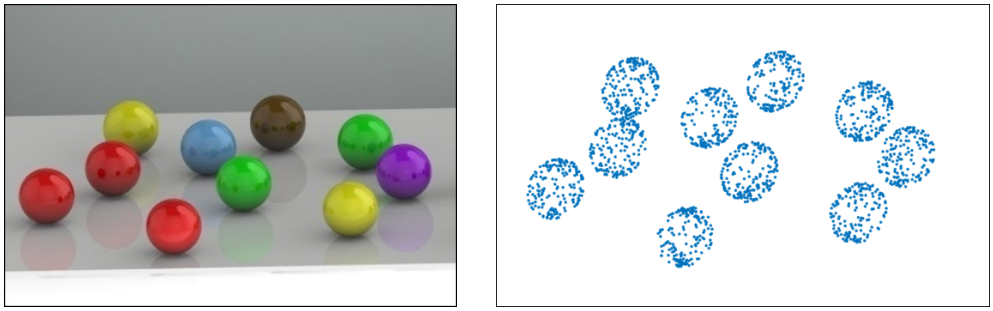
(b) Imagen de profundidad representada por un mapa de calor.

Figura 2.5: Imagen a color (izquierda) con imagen de profundidad (derecha) captadas con una cámara Asus.

Esta representación es útil para extraer características morfológicas de objetos para clasificación [9, 10] o segmentación [11, 12] . Adicionalmente puede utilizarse en conjunto con imágenes de color para identificar regiones de interés [12, 13] y descartar locaciones poco probables de encontrar el objeto, donde se observa una mejora en el tiempo de procesamiento.

2.2.2. Nube de puntos

Una nube de puntos es un conjunto de puntos distribuidos en un espacio tridimensional que contiene la información de distancia de la superficie de los objetos presentes en la escena. Son útiles para visualizar la información espacial captada por el sensor y obtener múltiples vistas de un objeto en particular (ver Figura 2.6).



(a) Imagen RGB.

(b) Nube de puntos.

Figura 2.6: Nube de puntos asociada a una imagen a color.

Es posible realizar una proyección de la nube de puntos a un plano 2D para obtener una imagen de profundidad. La relación entre ambas representaciones es la siguiente:

$$x = \mathrm{d}\frac{(u - c_x)}{f_x} \tag{2.2a}$$

$$y = \mathrm{d}\frac{(v - c_y)}{f_y} \tag{2.2b}$$

$$z = \mathrm{d} \tag{2.2c}$$

Donde (x, y, z) es la posición tridimensional de un punto; f_x, f_y son las distancias focales de la cámara en las direcciones x, y respectivamente; (c_x, c_y) es la posición de pixel principal de la cámara; d es la distancia desde el punto correspondiente al pixel (u, v) hasta la cámara.

2.3. Información de láser

Light Amplification by Stimulated Emission of Radiation (LASER) es un instrumento que envía un haz de luz, caracterizado por concentrar gran parte de su energía en un espectro de frecuencia estrecho a lo largo de su recorrido espacial, junto con su capacidad de concentrar niveles altos de energía.

Los láser tienen múltiples funcionalidades de acuerdo a la intensidad y frecuencia emitidas. Por ejemplo, en robótica, se utiliza el proceso LAser Detection And Ranging (LADAR) [14], el cual consiste en determinar la distancia del LADAR a un objeto (Figura 2.7), ya sea midiendo el tiempo de vuelo de un pulso de luz enviado por el láser, o midiendo la diferencia de fase entre el haz de luz emitido y recibido.

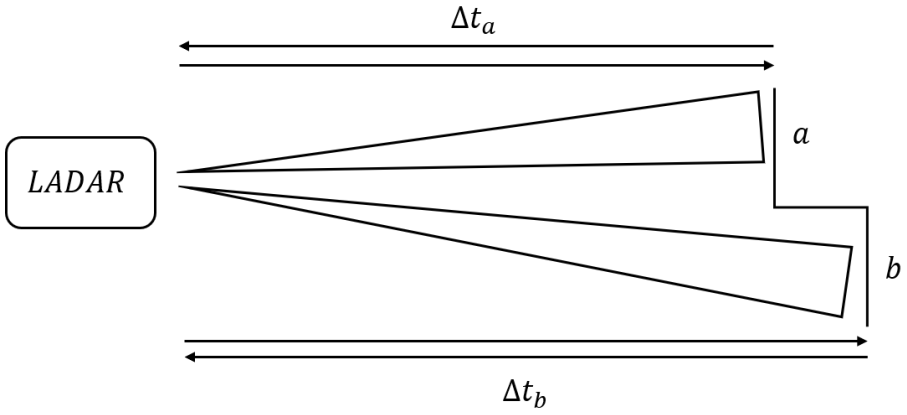


Figura 2.7: Representación gráfica del funcionamiento del LADAR de Pulsos.

La relación entre la distancia al objeto R, el tiempo de vuelo t, y la velocidad de la luz $c \approx 3 \times 10^8 [m/s]$ es la siguiente:

$$R = \frac{c \cdot t}{2} \tag{2.3}$$

En la Figura 2.8 se muestra una aplicación de LADAR utilizada para robots móbiles,

donde se realiza un barrido de mediciones en el escenario para tener un primer acercamiento a la posición de obstáculos y objetos de interés.

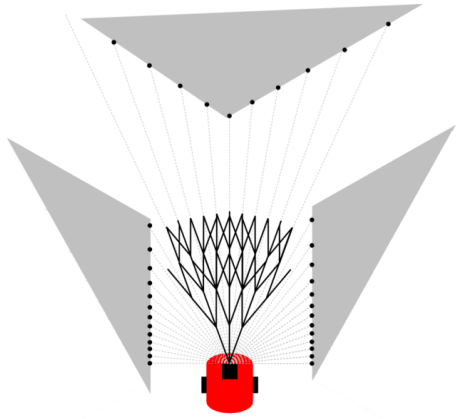


Figura 2.8: Barrido utilizando LADAR para la generación de un mapa.

2.4. Métodos para la extracción de características

La extracción de características consiste en la obtención de información útil de un objeto o una imagen. Para ello, se generan modelos matemáticos que permitan representarlos de forma concisa. De esta forma se puede disminuir notablemente la carga computacional en procesos de clasificación de objetos.

En inteligencia computacional se utilizan diversos métodos para la obtención de características, dependiendo de los requerimientos del clasificador y del tipo de objetos a clasificar para un mejor desempeño de éste.

En el caso de este trabajo, se utilizan histogramas de color con el fin de obtener una representación global de la distribución de color de cada personas, se extrae información de textura basada en la transformada LBP, se generan características locales de los gradientes entre pixeles vecinos mediante descriptores HOG para la detección de personas, se detectan bordes con el algoritmo de Canny para discriminar entre personas, etc.

2.4.1. Filtros lineales para el preprocesamiento

Un filtro se utiliza como un preprocesamiento de la imagen para modificarla o mejorarla, como por ejemplo remover el ruido o enfatizar ciertas características en ella en un procesamiento posterior. El filtrado es una operación local, ya que modifica cada pixel de acuerdo a una vecindad de pixeles. Específicamente, un filtro lineal es un filtro en que el valor resultante de cada pixel es una combinación lineal de los valores de la vecindad de pixel en la imagen original.

Para realizar un filtrado basta con computar la convolución entre la imagen y el filtro. Particularmente en imágenes, se utiliza la convolución bidimensional discreta. En la ecuación

(2.4) se muestra la operación para obtener la imagen filtrada g. Donde h es el filtro, f es la imagen original de tamaño $N \times M$, y (x, y) es la posición del pixel.

$$\begin{aligned} g(x,y) =& h(x,y) * f(x,y) \\ =& \sum_{\mathrm{i}=0}^{M-1} \sum_{j=0}^{N-1} f(\mathrm{i},j) h(x-\mathrm{i}, y-j), \quad \forall x, y \in \mathbb{Z} \quad \text{t.q.} \quad 0 \leq x < M, \quad 0 \leq y < N \end{aligned} \tag{2.4}$$

Cabe destacar que para el filtro h la posición $h(0,0)$ corresponde al centro del filtro en lugar de la primera posición. Luego, si h es de tamaño nxm, entonces se asume lo siguiente:

$$h(\mathrm{i},j) = 0, \quad \text{si} \begin{cases} \mathrm{i} < -\frac{(m-1)}{2} \\ \mathrm{i} > +\frac{(m-1)}{2} \\ j < -\frac{(n-1)}{2} \\ j > +\frac{(n-1)}{2} \end{cases} \tag{2.5}$$

En la Figura 2.9 se muestran diferentes tipos de filtros lineales aplicados a una imagen F. Estos son:

- Filtro gaussiano (Figura 2.9a): utilizado para reducir el ruido de distribución normal y suavizar la imagen. En el ejemplo se utiliza el filtro gaussiano de tamaño 5×5 y desviación estándar $\sigma = 1{,}1$.

$$G_g = \frac{1}{273} \begin{pmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{pmatrix} * F \tag{2.6}$$

- Filtro de Sobel (Figura 2.9b): utilizado para computar el gradiente en el eje X y eje Y de una imagen, útil para la detección de bordes.

$$G_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} * F \qquad G_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} * F \tag{2.7}$$

  El Gradiente aproximado se utiliza combinando los gradientes de ambos ejes, donde G es la magnitud y Σ es la orientación:

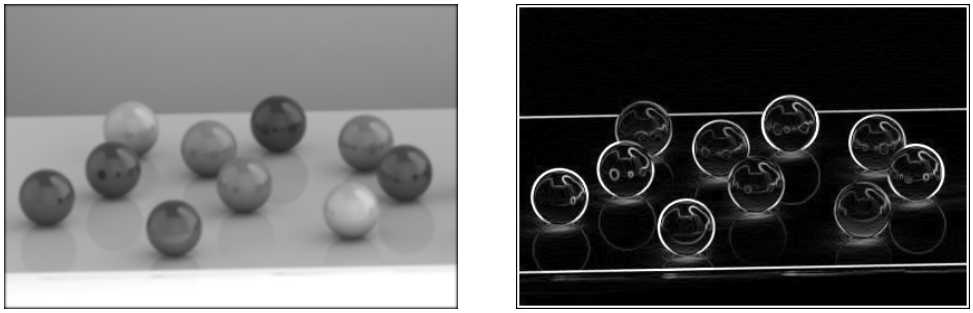
$$G = \sqrt{{G_x}^2 + {G_y}^2} \qquad \Sigma = atan\left(\frac{G_y}{G_x}\right) \tag{2.8}$$

- Filtro Shift (Figura 2.9c): utilizado para desplazamiento de la imagen. En el ejemplo la imagen se desplaza 2 pixeles a la izquierda.

$$G_{shift} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} * F \tag{2.9}$$

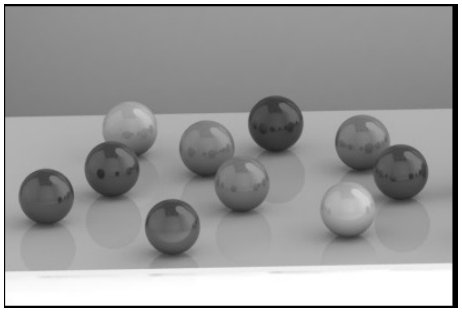
- Filtro Sharpen (Figura 2.9d): utilizado para intensificar los bordes (componentes de alta frecuencia espacial).

$$G_{sharpen} = \left(\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \right) * F \tag{2.10}$$

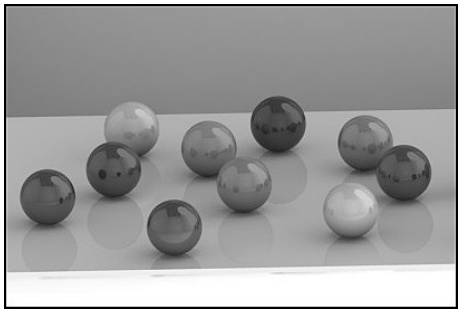


(a) Filtro Gaussiano

(b) Filtro de Sobel



(c) Filtro de Shift



(d) Filtro de Sharpen

Figura 2.9: Filtros lineales aplicados a la imagen de la Figura 2.10a.

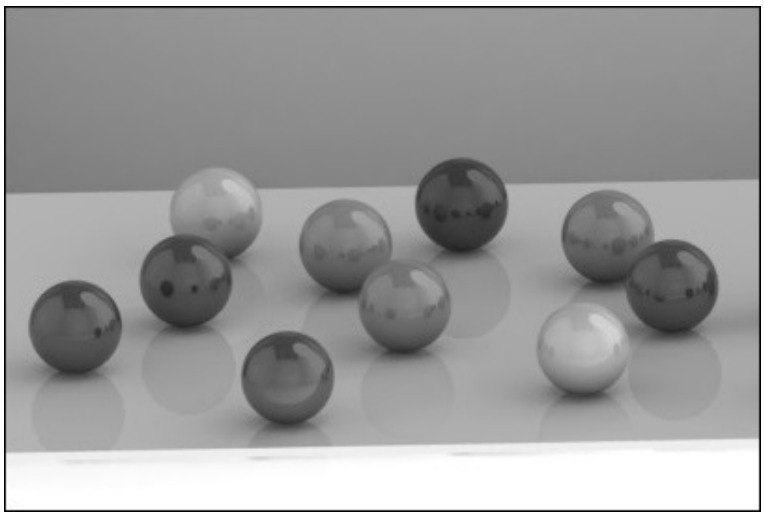
2.4.2. Extracción de histograma

Un histograma corresponde a una representación gráfica de la frecuencia de aparición de los diferentes valores que puede tomar una variable. En particular, en procesamiento de imágenes, el histograma indica el número de pixeles en la imagen que poseen una determinada intensidad. Sea $I(u, v) \in [0, K-1]$ la intensidad del pixel en la posición (u, v), K es el número de valores de intensidad posibles, se define el histograma como:

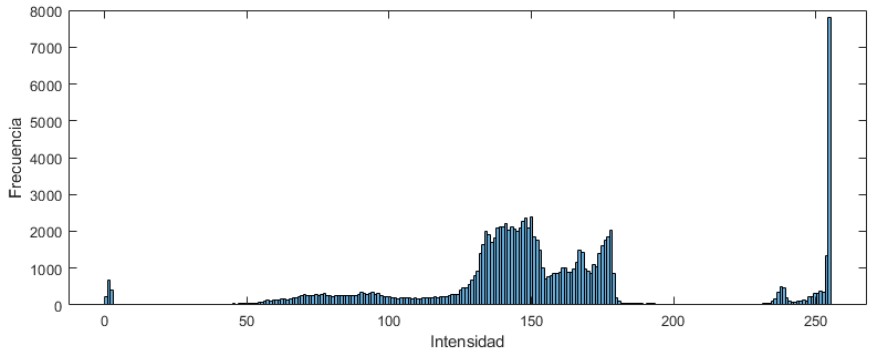
$$H(\mathrm{i}) = \sum_{j \in I_\mathrm{i}} 1, \quad \text{t.q.} \quad I_\mathrm{i} = \{(u, v) | I(u, v) = \mathrm{i}\}, \quad \forall \mathrm{i} \in [0, K-1] \tag{2.11}$$

En la Figura 2.10 se muestra una imagen junto a su histograma, donde se puede observar que existe una gran cantidad de pixeles con la intensidad máxima posible (de valor $I(u, v) = 255$ para imágenes en escala de grises de 8 bits de profundidad).

Según la descripción anterior el tamaño del histograma es proporcional a K. Esto generaría histogramas muy grandes para imágenes de mayor profundidad de bits. Sin embargo, existe



(a) Imagen en escala de grises de la Figura 2.3.



(b) Histograma con 256 bines.

Figura 2.10: Extracción de histograma a una imagen de escala de grises.

una solución a este problema llamada *binning*, que consiste en combinar valores del histograma por rangos de intensidad (bines). En la ecuación (2.12) se muestra la nueva expresión del histograma:

$$H(\mathrm{i}) = \sum_{j \in I_\mathrm{i}} 1, \quad \text{t.q.} \quad I_\mathrm{i} = \{(u,v) \mid a_\mathrm{i} \leq I(u,v) < a_{\mathrm{i}+1}\}, \quad \forall \mathrm{i} \in [0, K-1] \tag{2.12}$$

Donde $0 = a_0 < a_1 < ... < a_B = K$, siendo B el número de bines. En la Figura 2.11 se muestra la diferencia entre histogramas con distintos número de bines (notar que $B = K$ es equivalente a la definición de la ecuación (2.11)).

Hasta este punto se ha discutido sobre el histograma para imágenes monocromáticas. También se puede utilizar el mismo enfoque para imágenes multicanal, trabajando cada canal individualmente como si fuera una imagen monocromática (Figura 2.12b). No obstante, el histograma no está limitado para ser unidimensional, ya que tiene la capacidad de determinar la frecuencia de aparición de intensidades definidas por más de un valor (i.e. $I(u,v) = [\mathrm{i}_R, \mathrm{i}_G, \mathrm{i}_B]$ para una imagen RGB), como en la Figura 2.12c.

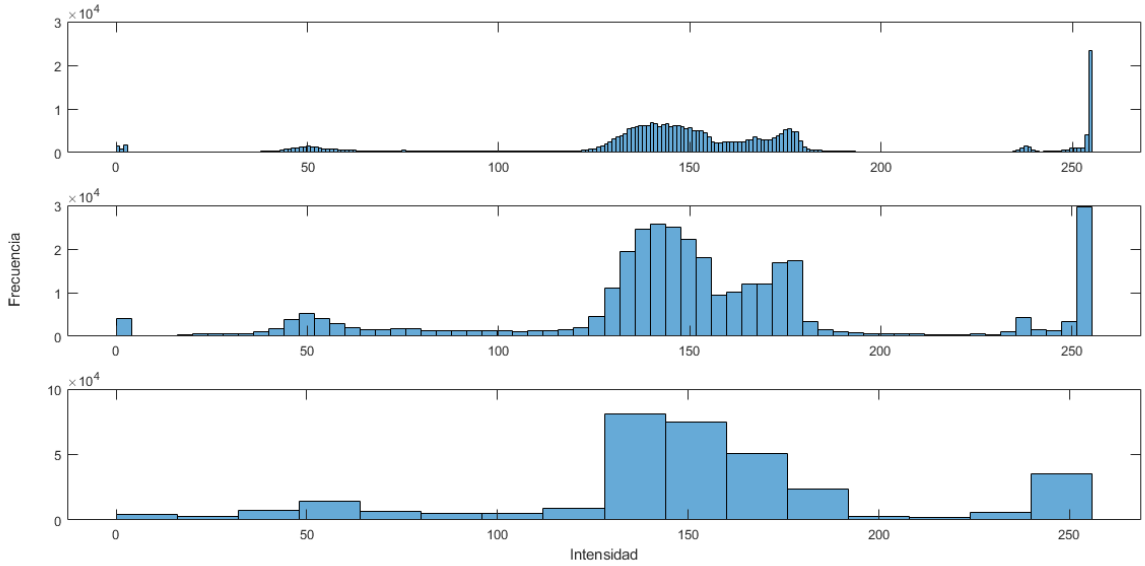
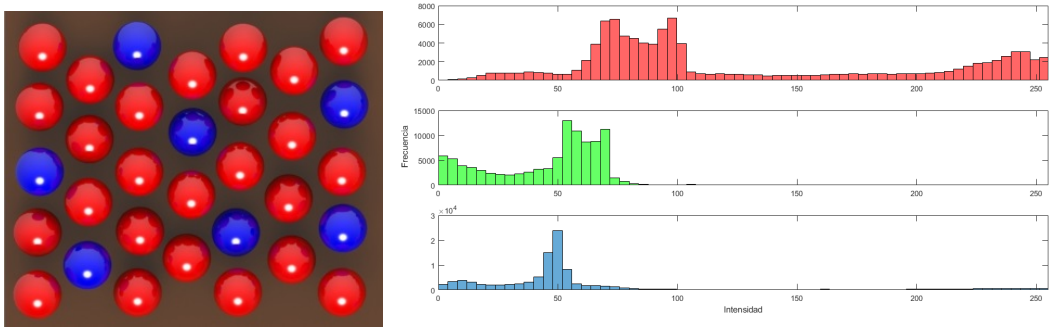
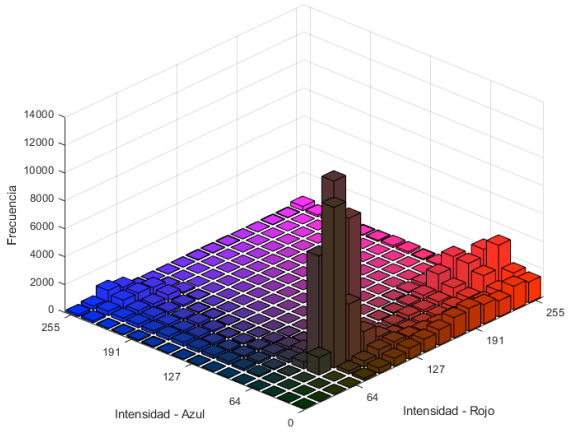


Figura 2.11: Extracción de histogramas de la Figura 2.10a para diferentes números de bines B. Superior: $B = 256$. Centro: $B = 64$. Inferior: $B = 16$.



(a) Imagen a color.

(b) Extracción de histogramas con separación de canales RGB. Superior: canal R. Centro: canal G. Inferior: canal R.



(c) Extracción de histograma bidimensional, para los canales R y B.

Figura 2.12: Extracción de histograma en imagen de múltiples canales.

2.4.3. Detección de bordes con Canny

La obtención de los bordes en una imagen puede ser útil para caracterizarla. Para ello se han propuesto diversos enfoques, siendo el detector de bordes Canny [15] uno de los más utilizados [16, 17, 18] pues, a pesar de ser un método existente por un largo periodo de tiempo, es de bajo costo computacional y posee una cantidad reducida de parámetros.

La detección de bordes de Canny consiste en un algoritmo de 4 etapas, que tienen por objetivo localizar los bordes en la imagen entregando una máscara binaria como resultado. El algoritmo es el siguiente:

1. **Reducción de ruido**: la detección de bordes es sensible al ruido, por lo que el primer paso es aplicar un filtro gaussiano (Sección 2.4.1) a la imagen. Generalmente se utiliza un filtro de 5x5 con desviación estándar $\sigma = 1{,}4$
2. **Cálculo de gradientes**: se obtienen las intensidades y orientaciones de los gradientes de la imagen utilizando el filtro de Sobel (Sección 2.4.1). Con esto se tiene que los bordes detectados son perpendiculares a la orientación del gradiente.
3. **Non-Maximum Suppression**: se utiliza para eliminar los bordes que no son el máximo. Es decir, para seleccionar solamente los bordes más representativos. Para esto se compara cada pixel en la imagen de intensidad de gradientes, y se compara con los dos adyacentes siguiendo la orientación del gradiente; si es mayor a ambos se mantiene, en caso contrario se elimina.
4. **Umbralización con histéresis**: esta etapa es para clasificar los gradientes restantes como bordes o no. Para ello se utilizan dos umbrales:
   - thr_{max}: los gradientes mayores a este umbral se clasifican inmediatamente como bordes.
   - thr_{min}: los gradientes menores a este umbral se clasifican inmediatamente como no bordes.

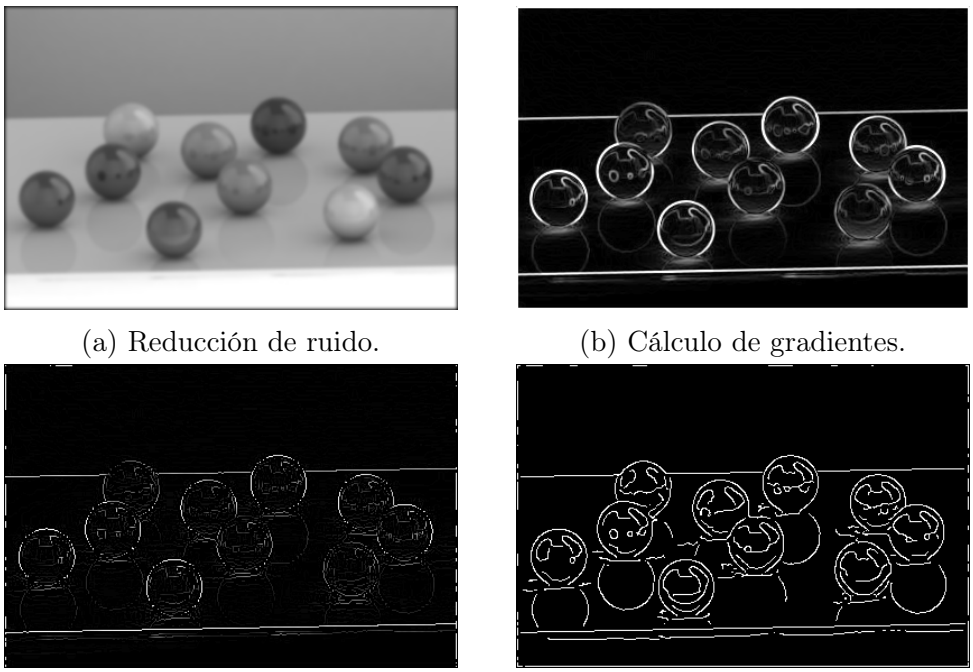
Luego, para cada gradiente que se encuentre entre ambos umbrales, se verifica si está conectado a un borde previamente clasificado. En caso de estarlo se clasifica dicho gradiente como borde. Este procedimiento se realiza hasta que no haya nuevos bordes asignados, resultando así la máscara de bordes de Canny.

En la Figura 2.13 se muestra un ejemplo del funcionamiento del algoritmo, utilizando un filtro gaussiano de 5×5 con $\sigma = \sqrt{2}$, $thr_{max} = 50$, y $thr_{min} = 20$.

2.4.4. Transformada LBP

Local Binary Pattern (LBP), introducido por Ojala *et al.* [19] es un descriptor basado en la textura de la imagen. Una de sus principales ventajas es ser computacionalmente simple de obtener, a la vez de ser ofrecer características de gran utilidad en múltiples aplicaciones, como el reconocimiento de rostros [20].

El primer paso para construir el descriptor LBP es utilizar el operador básico de LBP



(a) Reducción de ruido. (b) Cálculo de gradientes.

(c) Non-Maximum Suppression. (d) Umbralización con histéresis

Figura 2.13: Algoritmo de Canny

(Figura 2.14). Este operador le asigna un nuevo valor a los pixeles de la imagen. Para cada pixel, en una vecindad de 3×3, se utiliza el pixel central como umbral de binarización para obtener un conjunto de 8 dígitos binarios. Luego se concatenan los dígitos, resultando un patrón o número binario, que se puede representar como un número entero en el rango $[0, 255]$. En la Figura 2.15 se observa una representación visual del resultado de aplicar el operador LBP a una imagen en escala de grises.

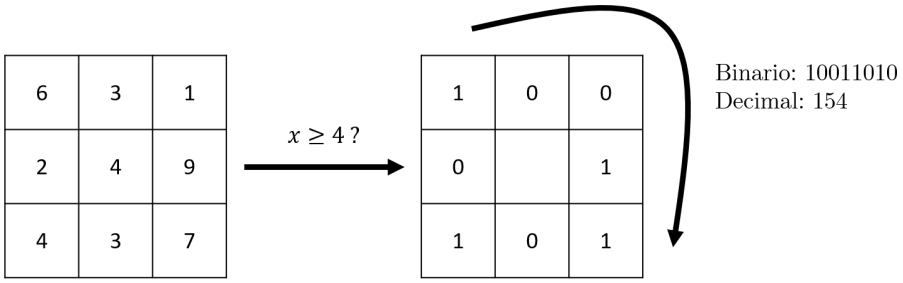


Figura 2.14: Operador LBP

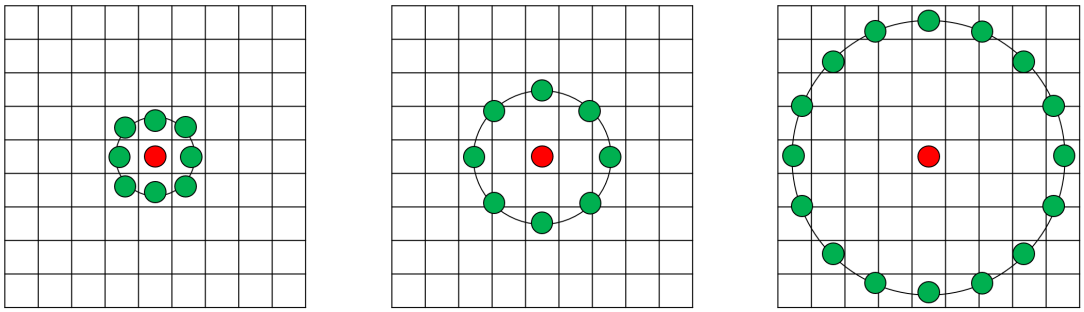
Una forma común de extraer características de la imagen procesada con el operador LBP es la de construir un histograma de 256 bines (sección 2.4.2), donde cada bin corresponde a un número binario diferente. Sin embargo Ojala *et al.* observaron que hay patrones poco probables, los que llamaron *uniform pattern* [21], los cuales son los que tienen a lo más dos transiciones entre bits (por ejemplo 00111000 tiene dos transiciones). Así, generaron un nuevo histograma donde existe un bin especial en el que consideran a todos los *uniform patterns*, transformando el histograma de 256 bines a uno de 58 bines, siendo éste el descriptor LBP.

Por otro lado, para trabajar en múltiples escalas se propuso usar vecindades circulares con

(a) Imagen a escala de grises. (b) Operador LBP$(8, 1)$ aplicado en la imagen.

Figura 2.15: Filtrado LBP.

interpolaciones de los valores de cada pixel, permitiendo variar el radio y el número de pixeles de la vecindad. Para esto se utiliza la notación $LBP(P, R)$, donde P es el número de puntos en un círculo de radio R. En la Figura 2.16 se observa una visualización del funcionamiento de ambos parámetros.



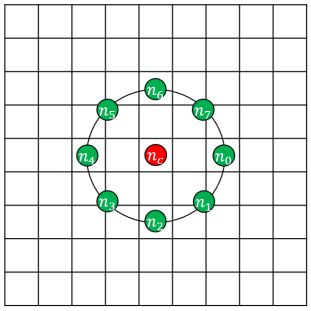
(a) LBP$(8, 1)$. (b) LBP$(8, 2)$. (c) LBP$(16, 4)$.

Figura 2.16: Funcionamiento de parámetros (P, R) en el operador LBP.

2.4.5. Transformada CSLBP

La transformada *Center-Symmetric Local Binary Pattern* (CSLBP), introducida en [22], combina el operador de textura LBP (sección 2.4.4) con el descriptor SIFT [23]. El operador CSLBP se aplica de forma similar al LBP, haciéndolo computacionalmente eficiente. Por otro lado se enfoca en representar gradientes locales en diferentes orientaciones, tal como lo hace el descriptor SIFT.

El operador CSLBP es similar al LBP, con la diferencia de que en lugar de comparar los pixeles de la vecindad con el pixel central, se comparan pixeles de la vecindad opuestos (ver Figura 2.17).



(a) Vecindad $(8, 1)$.

LBP

$s(n_0 - n_c)2^0 +$
$s(n_1 - n_c)2^1 +$
$s(n_2 - n_c)2^2 +$
$s(n_3 - n_c)2^3 +$
$s(n_4 - n_c)2^4 +$
$s(n_5 - n_c)2^5 +$
$s(n_6 - n_c)2^6 +$
$s(n_7 - n_c)2^7$

(b) Operador LBP.

CSLBP

$s(n_0 - n_4)2^0 +$
$s(n_1 - n_5)2^1 +$
$s(n_2 - n_6)2^2 +$
$s(n_3 - n_7)2^3$

(c) Operador CSLBP.

Figura 2.17: Comparación de la aplicación de los operadores LBP y CSLBP. Donde $s(x) = 1$ si $x \geq 0$, y $s(x) = 0$ si $x < 0$.

2.4.6. Descriptor HOG

Las siglas HOG provienen del término en inglés *Histogram of Oriented Gradients* [24], el cual es un descriptor que cuantiza las orientaciones de los gradientes en distintas zonas de la imagen. También está pensado para ser invariante a la iluminación, pero para ser aplicado a imágenes de un mismo tamaño.

El método de obtención del descriptor se puede resumir en un algoritmo de 5 pasos. Éste tiene un conjunto de parámetros, donde se utilizarán los que recomiendan Dalal et. al [24]. Luego, asumiendo que se tiene la imagen preprocesada de tamaño 64×128, el algoritmo es el siguiente:

1. **Calcular el gradiente de la imagen**: Se utilizan los filtros lineales unidimensionales:

$$K_X = \begin{pmatrix} -1 & 0 & 1 \end{pmatrix}, \quad K_Y = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$$

   Para cada pixel resultante se puede calcular la magnitud del gradiente G y orientación del gradiente θ a partir del resultado de los dos filtros:

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \frac{180}{\pi} arctan\left(\frac{G_y}{G_x}\right) \quad (mod\ 180)$$

   Donde G_x, G_y son las magnitudes del resultado de la imagen convolucionada con los filtros K_x, K_y, respectivamente. Cabe destacar que si la imagen posee más de 1 canal de color, la magnitud elegida es la máxima magnitud entre los canales, mientras que la orientación es la orientación correspondiente al máximo gradiente.

2. **Calcular el histograma de orientaciones de gradientes**: Para ello se divide la imagen en celdas de 8×8 (se obtienen 8x16 celdas). Por cada celda se construye un histograma de 9 bines equiespaciados entre $0°$ y $180°$, el cual cuantiza las orientaciones del gradiente donde cada orientación distribuye su magnitud proporcionalmente entre los dos bines más cercanos.

3. **Normalizar histogramas por bloques**: Para cada celda, exceptuando las del borde derecho y borde inferior, se agrupan entre 4: la celda actual, derecha, inferior, e inferior-derecha (se obtienen 7x15 bloques). Se concatenan esos 4 histogramas en uno nuevo (resulta un histograma de $9 \times 4 = 36$ bines) y éste se normaliza.

4. **Concatenar histogramas de bloques**: El descriptor HOG es el histograma que se forma al concatenar los histogramas resultantes del paso 3, el cual posee $7 \times 15 \times 36 = 3780$ bines.

En la Figura 2.18 se expresa el procedimiento explicado anteriormente.

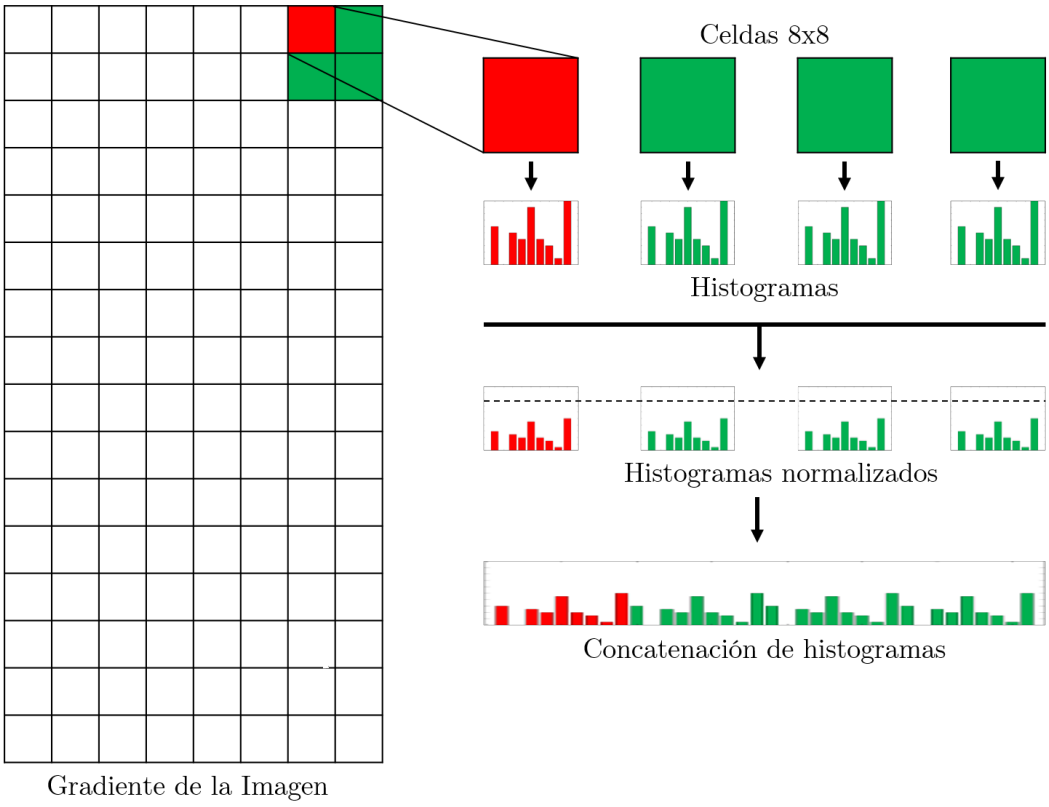


Figura 2.18: Esquema del algoritmo de HOG.

Este descriptor es utilizado para la detección de personas [24, 25, 26], ya que el performance alcanza valores cercanos al 90 % a costa de un alto costo computacional [25, 26]. Esto lo hace mediante el paradigma de ventanas deslizantes en múltiples escalas con un clasificador binario (por ejemplo SVM (Sección 2.5.1)) como se muestra en la Figura 2.19.

Para efectos de este trabajo, las personas a detectar no siempre se observan en cuerpo completo, lo que dificulta la utilización del detector HOG mencionado anteriormente. Es por ello que se utiliza el detector HOG1/2 basado en [27], que sigue el mismo funcionamiento del detector HOG, pero entrenado sólo con la parte superior de las personas.

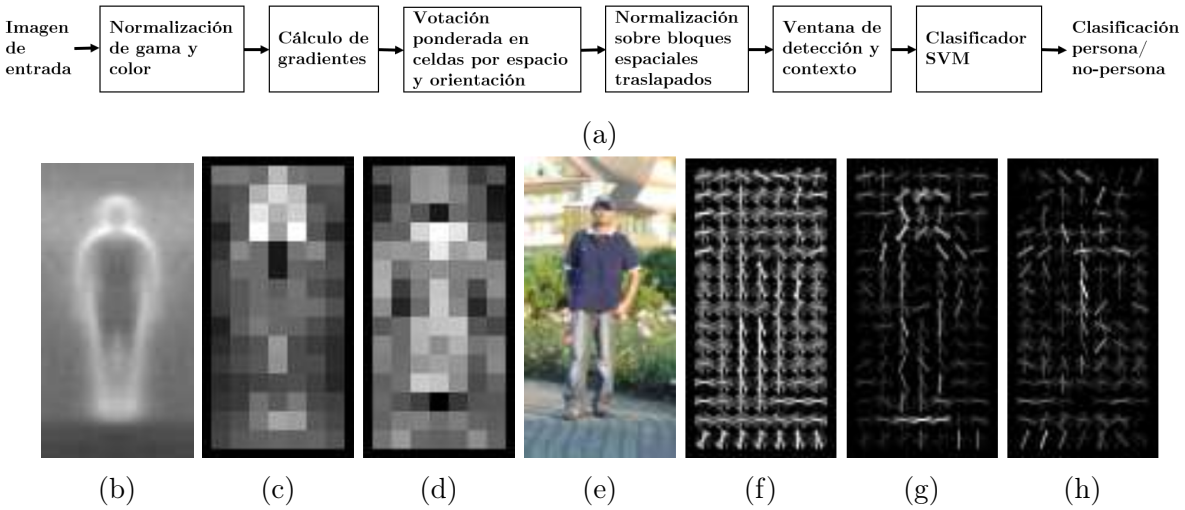


Figura 2.19: Detector de personas utilizando HOG+SVM con el paradigma de ventanas deslizantes. (a) Diagrama de bloques del detector. (b) El gradiente promedio de las imágenes utilizas en el entrenamiento. (c, d) Cada pixel muestra el peso SVM máximo de cada bloque para los pesos positivos y negativos, respectivamente. (e) Imagen de muestra. (f) Cálculo del descriptor HOG de la imagen. (g, h) El descriptor HOG ponderado con los pesos SVM positivos y negativos, respectivamente.

2.5. Clasificadores

Un objeto se puede describir por un conjunto de características, ya sea color, forma, tamaño, olor, y un sin fin de variables medibles con diferentes tipos de sensores. Particularmente en el procesamiento de imágenes, las características se pueden representar por vectores o matrices de un largo definido.

No obstante, las características por sí solas no siempre logran describir de manera óptima al objeto (muestra). Una forma de aprovecharas es mediante los clasificadores, que son los encargados de agrupar en conjuntos (clases) diferentes muestras mediante la similitud de sus características. Un clasificador aprende a clasificar mediante la observación de muestras de entrenamiento, donde luego será capaz de predecir la clase de nuevas muestras nunca antes vistas por él.

Los clasificadores se basan en el aprendizaje supervisado, que corresponde a un aprendizaje con muestras previamente etiquetadas a su clase correspondiente (e.g. algoritmo de SVM). A continuación se muestra el funcionamiento básico de 2 clasificadores con ejemplos de aplicación.

2.5.1. SVM

Una Máquina de Soporte Vectorial (SVM por sus siglas en inglés) es un clasificador discriminativo definido por un hiperplano separador óptimo entre dos clases, en base a un conjunto

de entrenamiento de ambas clases. Es decir, divide en dos el espacio de entrada donde se encuentran las muestras, como se ve en la Figura 2.20. Para construir el clasificador se definen los vectores de soporte, que son las muestras de ambas clases más cercanas al hiperplano separador (colores sólidos de la Figura 2.20). Luego, el hiperplano óptimo es el que maximiza la distancia entre vectores de soporte de clases diferentes.

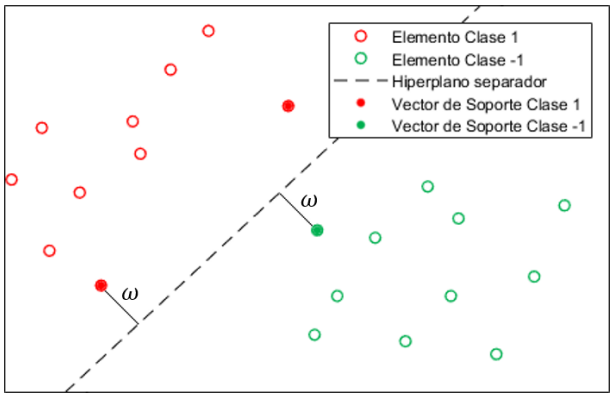


Figura 2.20: SVM Clásico.

Para obtener el hiperplano separador lineal, primero se define como

$$\boldsymbol{\omega} \cdot \mathbf{x} + b = 0 \tag{2.13}$$

Donde w es el vector de pesos, b es el bias, y x es el vector correspondiente a un elemento clasificable. Mientras que para los vectores de soporte cumplen la relación

$$\boldsymbol{\omega} \cdot \mathbf{x_i} + b = y_i \tag{2.14}$$

Donde, y_i es la etiqueta de la clase del elemento x_i del conjunto de entrenamiento que, sin pérdida de generalidad, $y_i \in -1, 1$.

Luego, con geometría se puede obtener la distancia de un vector de soporte al hiperplano, y por ende la distancia dist entre vectores de soporte de clases diferentes. El resultado de este cálculo es:

$$\mathrm{dis}t = \frac{2}{||\boldsymbol{\omega}||} \tag{2.15}$$

Finalmente, maximizar la distancia dist se traduce en el siguiente problema de optimización cuadrático:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||\boldsymbol{\omega}||^2 \\ \text{s.a.} \quad & y_i(\boldsymbol{\omega} \cdot \mathbf{x_i} + b) \geq 1 \quad \forall x_i \end{aligned} \tag{2.16}$$

Sin embargo hay casos en que no se pueden separar correctamente todos los elementos mediante el método anterior (Figura 2.21). Para ello se introduce la variable ξ que permite

clasificaciones incorrectas, pero penaliza esas instancias. Luego, el problema de optimización se convierte en el siguiente:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||\boldsymbol{\omega}||^2 + C\sum_{\mathrm{i}} \xi_{\mathrm{i}} \\ \text{s.a.} \quad & y_{\mathrm{i}}(\boldsymbol{\omega} \cdot \mathbf{x_i} + b) \geq 1 - \xi_{\mathrm{i}}, \quad \forall x_{\mathrm{i}}, \\ & \xi_{\mathrm{i}} \geq 0, \quad \forall \xi_{\mathrm{i}} \end{aligned} \tag{2.17}$$

Donde C es el *trade-off* de la penalización de clasificaciones incorrectas (i.e. a mayor C, mayor es la penalización). Este problema de optimización se puede resolver con los multiplicadores de Lagrange [28], resultando la optimización dual:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{\mathrm{i}} \alpha_{\mathrm{i}} - \frac{1}{2}\sum_{\mathrm{i}} \left(\sum_{j} \left(\alpha_{\mathrm{i}} \alpha_j y_{\mathrm{i}} y_j \mathbf{x_i}^T \mathbf{x_j} \right) \right) \\ \text{s.a.} \quad & \sum_{\mathrm{i}} \alpha_{\mathrm{i}} y_{\mathrm{i}} = 0 \\ & C \geq \alpha_{\mathrm{i}} \geq 0 \end{aligned} \tag{2.18}$$

Lo que resuelve para:

$$\boldsymbol{\omega} = \sum_{\mathrm{i}} \alpha_{\mathrm{i}} y_{\mathrm{i}} \mathbf{x_i} \tag{2.19}$$

$$b = y_k - \sum_{\mathrm{i}} (\alpha_{\mathrm{i}} y_{\mathrm{i}} \mathbf{x_k}^T \mathbf{x_i}) \tag{2.20}$$

Con k cualquiera tal que $0 < \alpha_k < C$

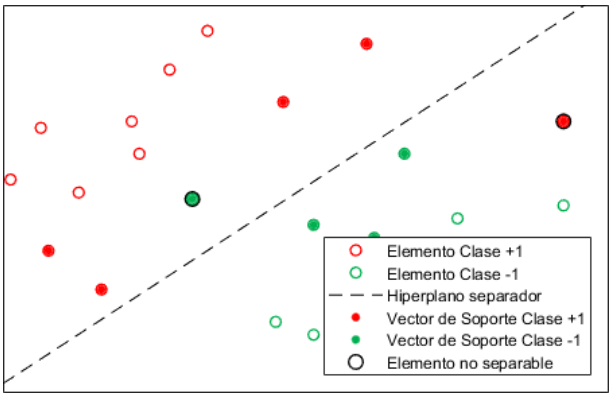


Figura 2.21: Clases linealmente no separables.

Por otro lado, hay ocasiones en que un hiperplano no lineal puede clasificar de mejor forma que el lineal propuesto en ecuación (2.13). Para abordar esto, SVM utiliza funciones de kernel no lineales para mapear las muestras en un nuevo espacio de mayor dimensionalidad, donde si es posible separar dichas muestras con la ecuación (2.13). El mapeo que aumenta la dimensionalidad se denota como $\phi(\mathbf{x_i})$, y al hacer el producto punto del mapeo de dos muestras se obtiene el kernel $k(\mathbf{x_i}, \mathbf{x_j}) = \phi^T(\mathbf{x_i})\phi(\mathbf{x_j})$. Luego, para modelar el nuevo problema

de optimización basta con reemplazar $\mathbf{x_i}^T\mathbf{x_j}$ por $k(\mathbf{x_i}, \mathbf{x_j})$ en las ecuaciones (2.18, 2.20) y $\mathbf{x_i}$ por $\phi(\mathbf{x_i})$ en la ecuación (2.19), donde el nuevo conjunto de datos de alta dimensionalidad es linealmente separable. En la Figura 2.22 se muestra el resultado de aplicar un kernel para datos no linealmente separables en el espacio original.

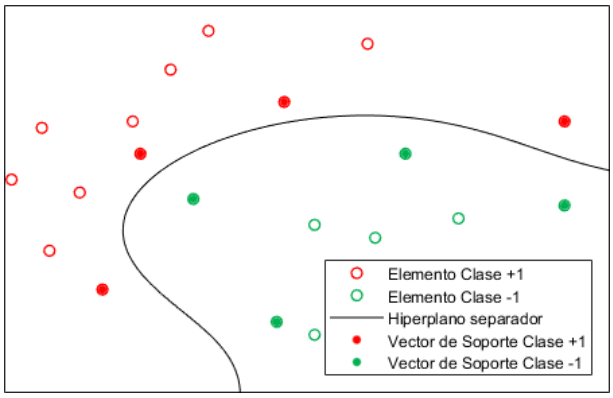


Figura 2.22: SVM con Kernel polinomial de grado 3. En el espacio $\phi(X)$ las clases son linealmente separables.

2.5.2. Adaboost

Es un algoritmo de boosting [29] propuesto por Freund y Schapire [30], que tiene como premisa combinar múltiples *clasificadores débiles* $h_i(x)$ (clasificadores de bajo desempeño) en un sólo *clasificador fuerte* $H(X)$ (clasificador de alto desempeño). Esto lo hace mediante la asignación de un peso a cada clasificador débil, mientras ayuda a elegir el conjunto de entrenamiento para cada uno. Cabe destacar que este algoritmo no genera nuevos clasificadores; solamente indica cómo agruparlos y entrenarlos.

Para obtener el clasificador fuerte, se realiza una iteración por cada clasificador débil h_t, el cual se entrena con una determinada distribución de muestras D_t. Luego, dependiendo de su tasa de error ε_t se obtiene el peso asociado y se actualiza D_t. En el Algoritmo 1 se describe el procedimiento.

Una de las aplicaciones más citadas es el detector de caras de Viola-Jones [31], que utiliza cascadas de filtros Adaboost (ver Figura 2.23) para clasificar entre caras y no-caras sin la necesidad de procesar todas las imágenes por la totalidad del detector.

Algoritmo 1: Obtención de clasificador fuerte con Adaboost.

Dados: $(x1, y1), ..., (x_m, y_m)$ donde $x_\text{i} \in X, \quad y_\text{i} \in -1, 1$

Inicializar: $D_1(\text{i}) = \frac{1}{m}$ para $\text{i} = 1, ..., m$

for $t = 1, ..., T$ **do**

Entrenar clasificadores débiles h_j, con $j = 1, ..., T$ usando distribución D_t

Elegir $h_t = h_j$ con el menor error ε_j, tal que:

$$\varepsilon_j = \sum_{\text{i}}^{m} (\omega_\text{i} | h_j(x_\text{i}) - y_\text{i}|)$$

Definir el peso para h_t:

$$\alpha_t = \frac{1}{2} ln(\frac{1 - \varepsilon_t}{\varepsilon_t})$$

Actualizar las distribuciones de muestras, para $\text{i} = 1, ..., m$:

$$D_t + 1(\text{i}) = D_t(\text{i}) \text{e} exp(-\alpha_t y_\text{i} h_t(x_\text{i}))$$

Normalizar la distribución de muestras, para $\text{i} = 1, ..., m$:

$$D_{t+1}(\text{i}) = \frac{D_{t+1}(\text{i})}{\sum_{\text{i}}^{m} (D_{t+1}(\text{i}))}$$

end

El clasificador fuerte es:

$$H(x) = sign(\sum_{t}^{T} (\alpha_t h_t(x)))$$

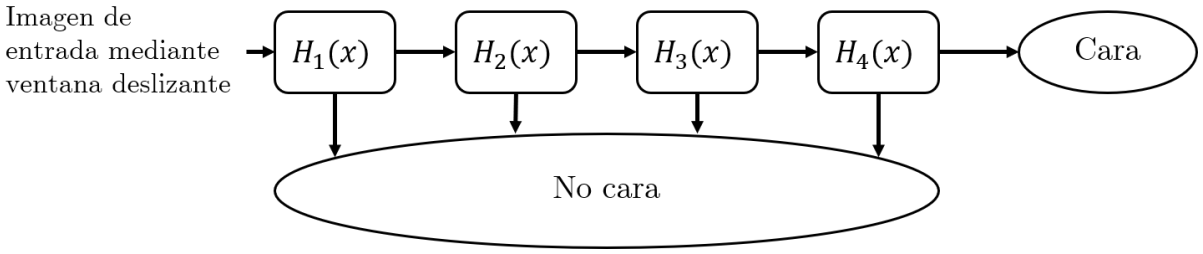


Figura 2.23: Uso de clasificadores fuertes obtenidos con Adaboost para la clasificación de caras y no-caras.

2.6. Clustering Euclidiano

El Clustering es un método que tiene por objetivo agrupar un conjunto de puntos en grupos específicos. En teoría, elementos de un mismo grupo (o cluster) tienen características similares (similitud), mientras que elementos de diferentes grupos difieren en características. Este es un método de aprendizaje no supervisado, ya que no necesita las etiquetas de cada

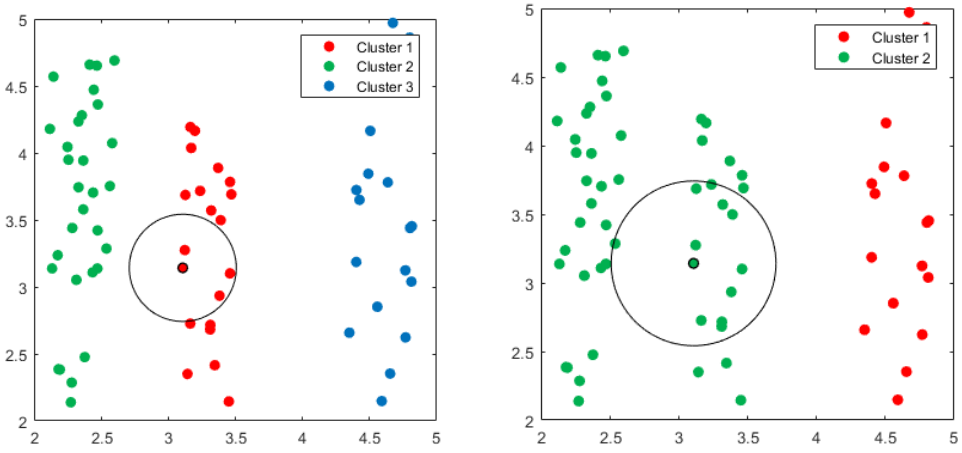
grupo para realizar la separación de elementos en grupos (clusterización).

Para el caso de una nube de puntos tridimensionales, un método simple para clusterizar es el utilizado por Rusu [32], donde utiliza la norma Euclidiana L2 como medida de similitud. El algoritmo consiste en agrupar todos los puntos con una distancia menor a d_{th} con respecto a otro punto del mismo grupo. El algoritmo es el siguiente:

Algoritmo 2: Clustering Euclidiano.

1 Crear una representación para la nube de puntos P. Se sugiere crear una representación de Kd-tree [33] para organizar los puntos.
2 Crear una lista de clusters C, y una cola de los puntos a verificar Q.
3 **for** *punto* $p_i \in P$*:* **do**
4 Agregar p_i a la cola Q.
5 **for** *punto* $p_i \in Q$*:* **do**
6 Buscar el conjunto P_i^k de puntos vecinos de p_i, en una esfera de radio $r < d_{th}$.
7 Para cada vecino $p_i^k \in P_i^k$, verificar si el punto ha sido procesado, y si no, agregar a Q.
8 **end**
9 Cuando la lista de todos los puntos en Q hayan sido procesados, agregar Q a la lista de clusters C, y reinicializar Q a una cola vacía.
10 **end**
11 El algoritmo finaliza cuando todos los puntos $p_i \in P$ hayan sido procesados y son parte de una lista de puntos en C.

El método anterior tiene como parámetro la distancia Euclidiana máxima entre dos puntos d_{th} tal que estos pertenezcan al mismo Cluster, lo que lo hace un parámetro de suma importancia pues se debe elegir dependiendo estrictamente de la aplicación específica del algoritmo. En la Figura 2.24 se muestra el resultado de aplicar el Clustering Euclidiano a un conjunto de puntos y la importancia de d_{th}.



(a) $d_{th} = 0{,}4$. Resultan 3 clusters. (b) $d_{th} = 0{,}6$. Resultan 2 clusters.

Figura 2.24: Efecto de la distancia euclidiana máxima d_{th} en el Cluster Euclidiano.

Cabe destacar que en este trabajo el clustering se utiliza para obtener candidatos a personas. Por lo tanto, una de las razones por las que se utiliza este método en particular, es

que no se necesita definir previamente el número de clusters a encontrar, pues se desconoce a priori la cantidad de personas presentes en la escena.

2.7. RANSAC

Random Sample Consensus (RANSAC) es un algoritmo propuesto por Fischler y Bolles [34] que estima los parámetros de un modelo predefinido, diseñado para tratar con una gran cantidad de outliers (ver Figura 2.25). RANSAC es un método que genera candidatos a soluciones en base a seleccionar la menor cantidad de puntos requeridos para estimar el modelo e itera esta selección.

El algoritmo consiste en lo siguiente:

Algoritmo 3: Estimación del modelo con RANSAC.

1. Seleccionar aleatoriamente el mínimo de puntos requeridos para determinar los parámetros del modelo.
2. Estimar el modelo con los puntos seleccionados.
3. Determinar cuántos puntos del conjunto total coinciden con el modelo, con una tolerancia predefinida ε.
4. Si la fracción de inliers sobre el total de puntos es mayor que un umbral predefinido τ, reestimar el modelo utilizando todos los inliers.
5. En caso contrario, repetir pasos del 1 al 4 un máximo de N veces.

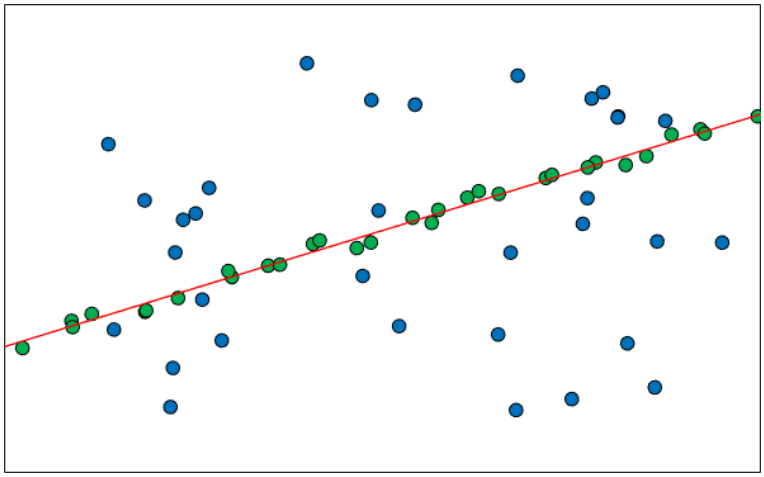


Figura 2.25: Estimación de recta con RANSAC en presencia de múltiples outliers.

2.8. Filtro de Kalman

El filtro de Kalman [35] es un estimador óptimo basado en la combinación de información ruidosa, hecho en un principio para sistemas lineales, aunque tiene su versión extendida para

sistemas no lineales [36]. Es ideal para sistemas dinámicos (en este caso se utiliza para estimar la posición y velocidad de las personas detectadas), ya que solamente necesita mediciones del estado anterior en lugar de un historial extenso, lo que además hace que sea rápido y fácil de implementar para problemas con procesamiento en tiempo real.

La premisa para combinar información proviene de generar una mejor estimación $\hat{x} \sim \mathcal{N}(\mu, \sigma)$ a partir de dos estimaciones de mayor incertidumbre $\hat{x}_1 \sim \mathcal{N}(\mu_1, \sigma_1)$, $\hat{x}_2 \sim \mathcal{N}(\mu_2, \sigma_2)$ [37] (ver Figura 2.26), de acuerdo a la siguiente ecuación:

$$\begin{aligned} k &= \frac{{\sigma_1}^2}{({\sigma_1}^2 + {\sigma_2}^2)} \\ \mu &= \mu_1 + k(\mu_2 - \mu_1) \\ \sigma^2 &= {\sigma_1}^2 - k{\sigma_1}^2 \\ \hat{x} &= \hat{x}_1 + k(\hat{x}_2 - \hat{x}_1) \end{aligned} \tag{2.21}$$

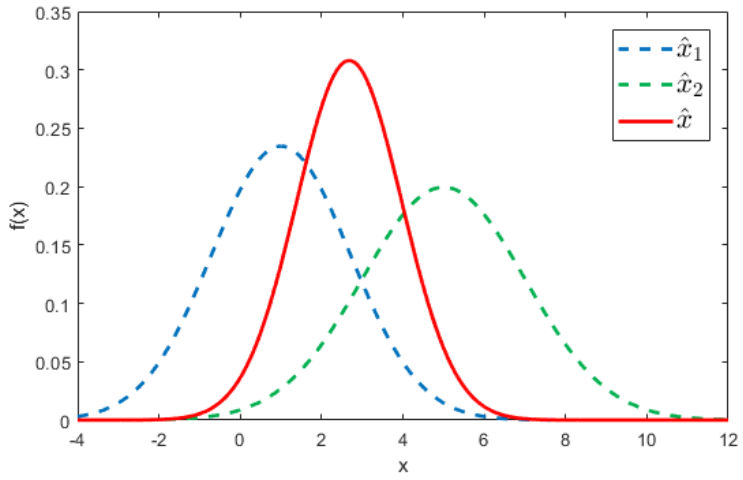


Figura 2.26: Obtención de una mejor estimación al combinar información.

Lo anterior se puede extender matricialmente, donde las medias μ_{i} se convierten en vectores $\boldsymbol{\mu_{\mathrm{i}}}$ y las varianzas σ_{i}^2 se convierten en matrices de covarianza Σ_{i}, mientras K es una matriz que se denomina ganancia de Kalman:

$$\begin{aligned} K &= \Sigma_1 \cdot (\Sigma_1 + \Sigma_2)^{-1} \\ \boldsymbol{\mu} &= \boldsymbol{\mu_1} + K(\boldsymbol{\mu_2} - \boldsymbol{\mu_1}) \\ \Sigma &= \Sigma_1 - K\Sigma_1 \\ \hat{\mathbf{x}} &= \hat{\mathbf{x_1}} + K(\hat{\mathbf{x_2}} - \hat{\mathbf{x_1}}) \end{aligned} \tag{2.22}$$

Para aplicar el filtro de Kalman se debe tener conocimiento del estado del sistema. Para ello se tiene el modelo de un sistema lineal discreto, y asume que las variables de estado son aleatorias con distribución Gaussiana:

$$\mathbf{x}(k+1) = F(k)\mathbf{x}(k) + G(k)\mathbf{u}(k) + \mathbf{v}(k) \tag{2.23}$$

Donde $\mathbf{x}(k)$ es el vector de estados en el instante k, $F(k)$ es la matriz de estado, $G(k)$ es la matriz de control, $\mathbf{u}(k)$ es el vector de control, y $\mathbf{v}(k) \sim \mathcal{N}(0, Q(k))$ son las perturbaciones al sistema de covarianza $Q(k)$. Por otro lado se tiene el modelo de observación (mediciones):

$$\mathbf{z}(k) = C(k)\mathbf{x}(k) + \mathbf{w}(k) \tag{2.24}$$

Donde $\mathbf{z}(k)$ es la observación o vector de salida en el instante k, $C(k)$ es la matriz de observación, y $\mathbf{w}(k) \sim \mathcal{N}(0, R(k))$ son las perturbaciones de covarianza $R(k)$. Luego, la finalidad del filtro de Kalman es utilizar la información combinada de ambos modelos para predecir el estado $\mathbf{x}(k+1)$.

No obstante, en la práctica sólo se tiene un estimado de los estados y las observaciones $\hat{x}(k)$ y $\hat{z}(k)$, respectivamente. Luego, utilizando álgebra se puede obtener el siguiente predictor junto con la matriz de covarianza del error $\tilde{\mathbf{x}}(k) = x(k) - \hat{\mathbf{x}}(k)$:

$$\begin{aligned} \hat{\mathbf{x}}(k+1|k) &= F(k)\hat{\mathbf{x}}(k|k) + G(k)\mathbf{u}(k) \\ P(k) &= cov(\mathbf{x}(k) - \hat{\mathbf{x}}(k)) \\ P(k+1|k) &= F(k)P(k|k)F^T(k) + Q(k) \end{aligned} \tag{2.25}$$

Por otro lado, para la estimación de la observación se tiene:

$$\begin{aligned} \hat{\mathbf{z}}(k+1|k) &= C(k+1)\hat{\mathbf{x}}(k+1|k) \\ S(k) &= cov(\mathbf{z}(k) - \hat{\mathbf{z}}(k)) \\ S(k+1|k) &= C(k+1)P(k+1|k)C^T(k+1) + R(k+1) \end{aligned} \tag{2.26}$$

Luego, utilizando la ecuación (2.22) y reemplazando $\hat{\mathbf{x}}_1$ por $\hat{\mathbf{x}}(k+1)$, $\hat{\mathbf{x}}_2$ por $\hat{\mathbf{z}}(k+1)$, Σ_1 por $P(k+1)$, y Σ_2 por $S(k+1)$, se tiene:

$$\begin{aligned} K(k+1) &= P(k+1|k)C^T(k+1)S^{-1}(k+1) \\ \hat{\mathbf{x}}_{kalman}(k+1|k+1) &= \hat{\mathbf{x}}(k+1|k) + K(k+1)(\mathbf{z}(k+1) - C(k+1)\hat{\mathbf{x}}(k+1|k)) \end{aligned} \tag{2.27}$$

Donde $\hat{\mathbf{x}}_{kalman}(k+1|k+1)$ es el mejor estado actual estimado, que se obtiene combinando la predicción hecha con el estado anterior k, con la estimación de la observación actual $k+1$. El algoritmo para implementar el filtro de Kalman se muestra a continuación:

Algoritmo 4: Filtro lineal de Kalman.

1 Inicializar el filtro: Obtener $\hat{x}(j|j)$ y $P(j|j)$ para algún j Predicción del estado estimado y su covarianza del error:

$$\hat{\mathbf{x}}(k+1|k) = F(k)\hat{\mathbf{x}}(k|k) + G(k)\mathbf{u}(k)$$

$$P(k+1|k) = F(k)P(k|k)F^T(k) + Q(k)$$

2 Medición del residuo de la observación:

$$\tilde{\mathbf{z}}(k+1|k) = \mathbf{z}(k+1) - C(k+1)\hat{\mathbf{x}}(k+1|k)$$

3 Covarianza de la observación:

$$S(k+1) = C(k+1)P(k+1|k)C^T(k+1) + R(k+1)$$

4 Ganancia de Kalman:

$$K(k+1) = P(k+1|k)C^T(k+1)S^{-1}(k+1)$$

5 Actualizar a estimación mejorada:

$$\hat{\mathbf{x}}_{kalman}(k+1|k+1) = \hat{\mathbf{x}}(k+1|k) + K(k+1)\tilde{\mathbf{z}}(k+1|k)$$

6 Actualizar covarianza estimada:

$$P(k+1|k+1) = P(k+1|k) - K(k+1)S(k+1)K(k+1)$$

7 Volver al paso 2 y calcular: $\hat{\mathbf{x}}(k+2|k+1)$, $P(k+2|k+1)$

2.9. Algoritmo de Munkres para la asociación de información

Parte de la solución propuesta requiere asociar información en distintos *frames* de la imagen. Este problema se puede formular como un conjunto de n individuos que deben realizar m tareas, donde cada individuo tiene asociado un costo para cada tarea. Particularmente para $n = m$, lo que se busca es el costo mínimo para que cada individuo realice una sola tarea, y que todos los trabajos se cumplan.

El algoritmo de Munkres (también conocido como algoritmo Húngaro) [38] propone una solución al problema mencionado, minimizando el costo obtenido por la matriz de Costos $C(\mathrm{i}, j)$ del individuo i en la tarea j. En la ecuación (2.28) se muestra un ejemplo de la matriz de costos para $n = m = 4$, $individuo = \{a, b, c, \mathrm{d}\}$, $tarea = \{p, q, r, s\}$. Una asignación posible sería la tupla $A = \{(a, s), (b, q), (c, r), (\mathrm{d}, p)\}$, con un costo total de 20.

$$C(\mathrm{i},j) = \begin{array}{c} \\ a \\ b \\ c \\ \mathrm{d} \end{array} \begin{array}{c} \begin{array}{cccc} p & q & r & s \end{array} \\ \begin{bmatrix} 1 & 2 & 3 & \mathbf{4} \\ 2 & \mathbf{4} & 6 & 8 \\ 3 & 5 & \mathbf{7} & 9 \\ \mathbf{5} & 6 & 7 & 8 \end{bmatrix} \end{array} \tag{2.28}$$

El algoritmo de Munkres, implementado en [39], describe una forma de cómo obtener la asignación óptima minimizando los costos. El algoritmo es el siguiente:

Algoritmo 5: Asignación de Munkres.

1. Crear una matriz de costo de $n \times m$. Cada elemento representa el costo de asignar uno de los n individuos a una de los m tareas. Rotar la matriz de forma de que hayan igual o más columnas que filas, y definir $K = min(n, m)$. Ir al paso 2.
2. Para cada fila de la matriz, encontrar el menor elemento y restarlo a todos los elementos de esa fila. Ir al paso 3.
3. Encontrar un cero (Z) en la matriz resultante. Si no hay un cero marcado en su fila o columna correspondiente, marcar Z. Repetir para todos los elementos en la matriz. Ir al paso 4.
4. Tapar todas las columnas que contienen un cero marcado. Si las K columnas están tapadas, los ceros marcados describen una asignación completa. En este caso ir al paso 8; si no, ir al paso 5.
5. Encontrar un cero no cubierto y llamarlo P. Si no hay un cero marcado en la fila de P, ir al paso 6. En caso contrario, tapar la fila de P y destapar la columna que contiene el cero marcado. Continuar con este procedimiento hasta que todos los ceros estén tapados. Guardar el menor valor no cubierto, luego ir al paso 7.
6. Construir una serie alternando los ceros no cubiertos obtenidos en el paso 5 y los ceros marcados, como se indica a continuación. Sea Z_0 el cero no cubierto P, Z_1 el cero marcado en la columna de Z_0 (si existe), Z_2 el cero no cubierto P en la fila de Z_1, y continuar hasta que la serie termine con un cero no cubierto que no tenga un cero marcado en su columna. Desmarcar cada cero marcado de la serie, luego marcar cada cero no cubierto P de la serie, luego borrar todos los ceros P restantes, y destapar todas las filas y columnas de la matriz. Volver al paso 3.
7. Sumar el valor almacenado en el paso 5 a todos los elementos de las filas tapadas, y restarlo de todos los elementos de las columnas destapadas. Volver al paso 5.
8. Algoritmo finalizado. Los pares asignados se indican por las posiciones de los ceros marcados en la matriz de costos. Es decir, si $C(\mathrm{i}, j)$ tiene un cero marcado, entonces el elemento de la fila i está asociado con el elemento de la columna j.

Capítulo 3

Metodología

Para lograr los objetivos planteados en la sección 1.2, se implementa un sistema compuesto de 4 módulos conectados entre sí (ver Figura 3.1). Esta solución se obtuvo en un proceso de investigación de múltiples métodos utilizados en la literatura, permaneciendo los más idóneos con respecto al problema, que se ajusten a los objetivos generales, y con un buen desempeño. Además, la solución permite modificar de manera independiente cada bloque, ya sea para corrección de errores en la etapa de desarrollo, como en la actualización del sistema con métodos con mayor *performance* que se vayan desarrollando en el tiempo.

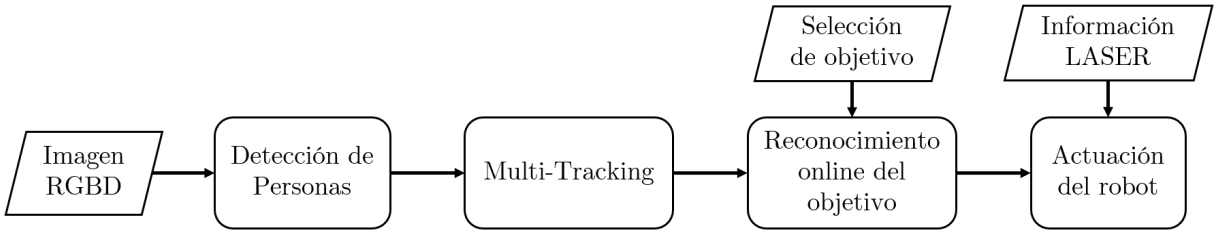


Figura 3.1: Diagrama de bloques del sistema.

El primer módulo corresponde a un detector de personas utilizando la información RGBD de la cámara. El segundo módulo se encarga de realizar un *tracking* de todas las personas detectadas en el *stream* de imágenes, con lo que es capaz de asociar a las personas en distintos instantes de tiempo, y predecir sus posiciones. El tercer módulo se encarga de reconocer a la persona a seguir (desde ahora llamada objetivo), mediante la extracción de características visuales y morfológicas, y discriminándola con respecto a las demás personas. El último módulo es el encargado de la actuación y comportamiento del robot, esto es: recibe la posición del objetivo y traslada al robot su alcance.

3.1. Detección de personas

En la literatura existe una gran variedad de métodos para detectar personas con imágenes de color. Por ejemplo en [40, 41] utilizan descriptores HOG, alcanzando tasas de *accuracy*

del 90 %, pero a un costo computacional alto para trabajar en tiempo real con una máquina de recursos limitados. Para incrementar la velocidad de detección se utilizan distintas características de menor costo, pero se reduce el desempeño del detector [42, 43].

Por otro lado, en [44, 45, 46] reducen la imagen a una región de interés (ROI por sus siglas en inglés) para evitar analizar zonas de la imagen en donde es poco probable la presencia de una persona. Para ello debe considerarse que, en su mayoría, esto requiere de hardware adicional como cámaras stereo, infrarrojas, o de profundidad. En el caso del robot Pepper, está disponible la cámara RGBD para satisfacer esta condición.

El esquema general propuesto para la detección de personas es el de la Figura 3.2. Con la cámara RGBD se obtiene una nube de puntos (sección 2.2.2) con lo que se estima el plano de tierra, correspondiente al suelo en el que el robot se encuentra. Posteriormente se realiza una segmentación a partir de los *clusters* obtenidos mediante la proyección de los puntos en el plano de tierra. Por último se utiliza un clasificador de personas para cada región de interés obtenidas por la segmentación anterior.

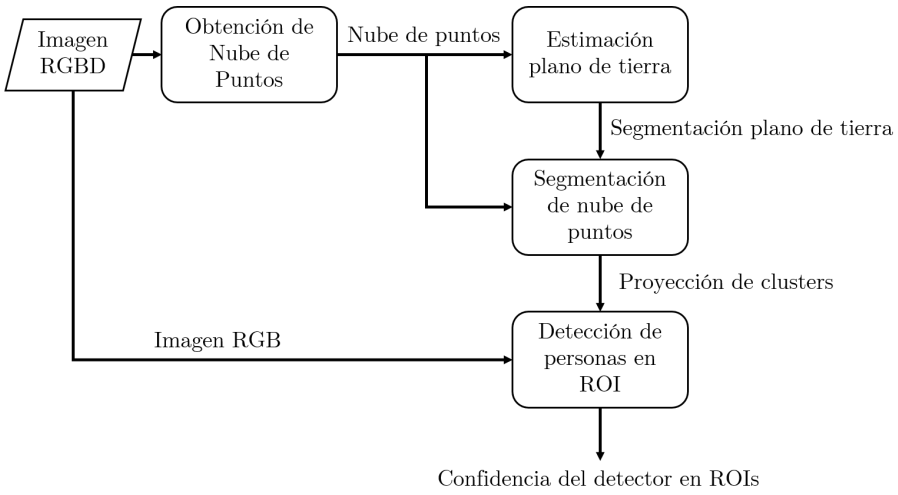
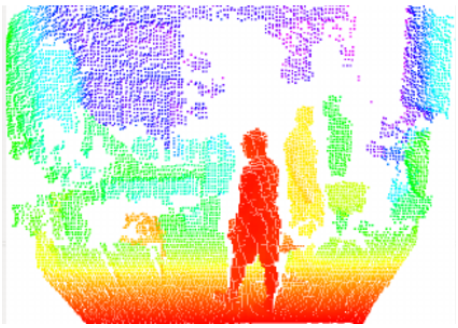


Figura 3.2: Diagrama de bloques del módulo de detección de personas.

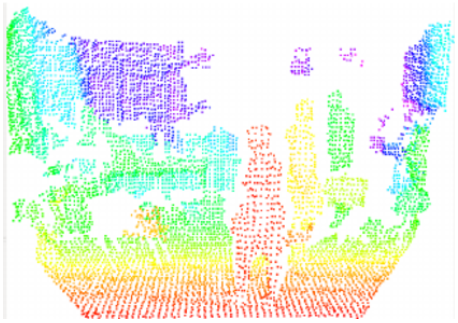
3.1.1. Estimación de plano de tierra

La nube de puntos obtenida por el sensor RGBD genera una distribución de partículas de acuerdo a la forma de los objetos dentro del campo de visión. En primera instancia, se realiza un *downsampling* a la nube de puntos mediante un *Voxel Grid Filter* [47], que consiste en dividir el espacio en celdas (*Voxels*) y reemplazar todos los puntos dentro de ellas por su centroide (ver Figura 3.3). Lo anterior se realiza con la finalidad de obtener una distribución de puntos uniforme en el espacio, además de reducir el número de puntos a procesar.

Para estimar el plano de tierra, se asume que el suelo es liso, por lo que los puntos correspondientes siguen una distribución que se aproxima al modelo matemático de un plano de la ecuación (3.1).



(a) Nube de puntos original.



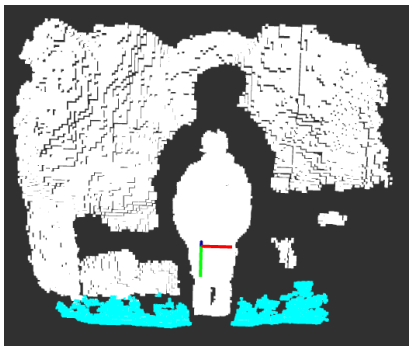
(b) Nube de puntos filtrada con un *Voxel Grid Filter*.

Figura 3.3: Efecto del *Voxel Grid Filter* en una nube de puntos [1].

$$Ax + By + Cz + D = 0 \tag{3.1}$$

Por otro lado, el robot Pepper tiene el sensor RGBD en la cabeza, alineado horizontalmente con respecto al suelo. Esto implica que, de acuerdo a la Figura 3.4, el plano de tierra observado por el sensor tiene las componentes normalizadas A y C (correspondiente a los ejes X, Z, respectivamente) cercanas a 0, mientras que la componente B (correspondiente al eje Y) es cercana a 1.

(a) Imagen de color del sensor RGBD. La ubicación del sensor no permite observar a la persona en su totalidad.



(b) Nube de puntos del sensor RGBD, con los puntos del plano de tierra segmentados. El sistema de coordenadas es X - rojo; Y - verde; Z - azul.

Figura 3.4: Segmentación del plano de tierra.

Por último, se realiza una estimación del modelo del plano de tierra de la ecuación (3.1) mediante RANSAC (sección 2.7). En el Anexo A se muestra el detalle de los parámetros utilizados.

3.1.2. Segmentación de nube de puntos

Una vez que se estima el plano de tierra se remueven los puntos pertenecientes a éste, y luego se agrupan los puntos con el método de cluster Euclidiano (sección 2.6). Se asume que las personas tienen un rango acotado de alturas, por lo que si un cluster está muy alto o muy bajo con respecto al suelo, éste es eliminado.

Con este método surge un problema, que es la agrupación de dos o más personas en un mismo cluster cuando se encuentran muy cercanas. La solución a esto es detectar cabezas con máximos locales para los clusters más anchos que una persona (definido previamente por un parámetro).

Para cada cluster se genera un mapa de alturas (ver Figura 3.5b) [1]. Éste se obtiene utilizando los puntos más altos del cluster con respecto al plano de tierra (ecuación (3.1)), a lo largo del eje X. Luego se encuentran los máximos locales correspondientes a las cabezas, para finalizar con el subclustering dependiendo de la posición de las cabezas (Figura 3.5d). El algoritmo, propuesto en [1], se muestra a continuación:

Algoritmo 6: Subclustering de personas utilizando máximos locales.

1. Para cada cluster crear un mapa de alturas a lo largo del eje X, con respecto al plano de tierra.
2. Buscar máximos locales en el mapa de alturas.
3. Se mantienen sólo los máximos locales que se encuentren separados mayor a una distancia d_{intima}, ya que las cabezas usualmente están separadas a una distancia mayor que la distancia íntima [48].
4. Por cada máximo local remanente, se crea un subcluster que contiene todos los puntos del cluster dentro de la distancia íntima, con respecto al eje X.
5. Subclusters muy altos o muy bajos con respecto al plano de tierra son descartados. Subclusters con pocos puntos también son descartados.

3.1.3. Detección de personas en Región de Interés

Cada subcluster de puntos generado en el paso anterior es proyectado en la imagen de color del sensor RGBD (Figura 3.6). Cabe destacar que Pepper tiene la posibilidad de generar una imagen RGB coincidente con la imagen de profundidad mediante los parámetros extrínsecos entre los receptores de color y profundidad, por lo que la nube de puntos del subcluster se proyecta a una imagen de profundidad utilizando las ecuaciones (2.2), y luego la imagen de profundidad obtenida se proyecta directamente a la imagen de color.

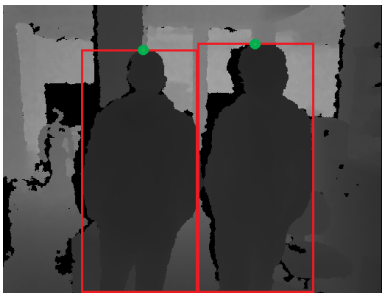
Al proyectar el subcluster a la imagen RGB, se obtiene una Región de Interés (ROI por sus siglas en inglés) en donde podría encontrarse una persona. Para ello se considera el centroide del subcluster como el centro de la ROI, el punto más alto de la nube de puntos con respecto al plano de tierra define la altura, y el ancho se estima como la mitad de la altura.

Luego, cada ROI se evalúa con el detector de personas HOG1/2 [27], el que se encuentra

(a) Imagen de color.

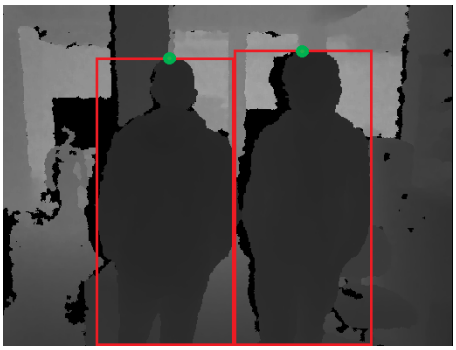
(b) Representación gráfica del Mapa de alturas.

(c) Cluster extenso.

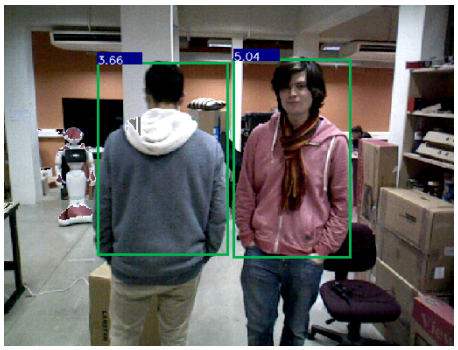


(d) Subclustering mediante máximos locales con localización de cabezas.

Figura 3.5: Subclustering para separar a personas muy cercanas.



(a) Clusters obtenidos de la etapa anterior (sección 3.1.2).



(b) Detecciones HOG1/2 obtenidas mediante la proyección de clusters en la imagen RGB. El número sobre cada ROI es la confidencia del detector.

Figura 3.6: Detección de personas en ROI.

disponible en [49] con el fin de obtener la confidencia de si el subcluster es una persona. Este enfoque evita que la evaluación del detector HOG1/2 se realice en toda la imagen y a distintas escalas, lo que imposibilitaría utilizarlo en tiempo real por el alto costo computacional que conlleva. Si la confidencia es mayor a un umbral τ_{HOG}, entonces se clasifica como persona y la ROI se recorta para delimitar solamente el torso y la cabeza, ya que en la mayoría de las veces el robot no es capaz de visualizar las piernas del objetivo con la cámara RGBD.

3.2. Tracking de personas

Las detecciones de personas realizadas en la sección 3.1 no son suficientes para discernir entre dos o más personas en diferentes instantes de tiempo. Para ello es necesaria la introducción de un módulo encargado de esta tarea, es decir, de asignar una identificación a cada detección en cada *frame*.

Un enfoque ampliamente utilizado debido a su facilidad para acoplarse a detectores creados previamente, es el *Tracking-by-Detection* [50]. Éste consiste en recibir las detecciones del objeto (en este caso los objetos en cuestión son personas), y utilizar esa información para realizar la asociación detección-persona y saber en qué lugar del siguiente *frame* se encontrará cada una de ellas.

El módulo desarrollado se basa en el *Tracking-by-Detection*, donde cada persona detectada por primera vez es asignada a una ID (desde ahora llamada *track*), y en los *frames* posteriores se intenta asignar una de las nuevas detecciones a cada *track*, mediante la asociación de información de color, posición, y confidencia del detector de personas.

Un *track* se compone de lo siguiente:

- Imagen RGBD del *frame* actual, con la ROI de la detección de la persona.
- Posición de la persona en las coordenadas X,Y,Z, con respecto al sistema de referencia de la cámara RGBD.
- Altura de la persona.
- Confidencia de ser persona, entregada por el detector HOG1/2 de la sección 3.1.3.

El módulo se divide en tres partes principales: la asociación de información para asignarle al *track* las nuevas detecciones de la misma persona, un clasificador online de color para utilizarlo a la asociación de información, y la predicción de la nueva posición de cada *track* en el *frame* siguiente.

3.2.1. Asociación de información

Asociar diferentes tipos de información es útil para encontrar las correspondencias entre un *track* ya inicializado y una nueva detección en el *frame* actual. Para ello se utiliza el método Global Nearest Neighbor [51] resuelto por el algoritmo de Munkres (sección 2.9), el que intenta maximizar la probabilidad conjunta de los siguientes 3 términos:

- **Posición y velocidad** Para cada *track* i se observa la posición de la nueva detección j, y se estima la velocidad que debería tener el *track* para encontrarse en la posición de la nueva detección. Utiliza la distancia de Mahalanobis $D_M^{i,j}$ [52], donde la matriz de covarianza se obtiene de la predicción de movimiento generada por el filtro de Kalman de la sección 3.2.3. Es útil para discriminar entre personas con colores similares, pero que existe una distancia considerable entre ellas.
- **Clasificador de color**. Se obtiene con el clasificador Adaboost (sección 2.5.2) definido

en la sección 3.2.2, donde la confidencia $c^{i,j}_{online}$ es el resultado de evaluar el histograma de la detección j en el clasificador correspondiente al *track* i. Es útil para discriminar entre personas cercanas.

- **Confidencia de detección**. Se utiliza la confidencia $c^{j}_{HOG1/2}$ entregada por el detector HOG1/2 (sección 3.1.3) de la detección j. Es útil para discriminar las personas de los objetos clasificados como persona con una baja confidencia, de colores similares al *track*.

Se busca maximizar las probabilidades de que cada *track* i tenga asociado correctamente una detección j, por lo que cada término se pondera por un parámetro (α, β, γ) elegido empíricamente [53], como se muestra en la ecuación (3.2). El resultado de la suma ponderada de los 3 términos se asigna a la matriz de costos $C_{Munkres}$ del algoritmo de Munkres [51], obteniendo así las asociaciones de los *tracks* con las nuevas detecciones. Cabe destacar que los individuos y las tareas mencionadas en la descripción del algoritmo de Munkres (sección 2.9) son representados por los *tracks* y detecciones, respectivamente.

$$C_{Munkres}(i, j) = \alpha D^{i,j}_{M} + \beta c^{i,j}_{online} + \gamma c^{j}_{HOG1/2} \tag{3.2}$$

3.2.2. Clasificador online de color para la asociación de información

El objetivo de la clasificación de los *tracks* por colores consiste en utilizar la información de color de la imagen para discernir entre distintas personas a pesar de que éstas se encuentren relativamente cercanas.

Para lo anterior se podría calcular el histograma de colores (sección 2.4.2) de cada *track* al momento de su inicialización y definir una medida de similitud para comparar entre las nuevas detecciones de personas. No obstante, debido a los cambios de iluminación, diferencias de tamaño, oclusiones, cambios en el *background*, y distintos puntos de vista de la persona (ver Figura 3.7), el histograma varía en los *frames* posteriores.

Figura 3.7: Imágenes de una misma persona en diferentes *frames*, dependiendo de las poses del robot y del objetivo. El número en la esquina superior izquierda indica la confidencia entregada por el detector de personas.

La solución utilizada para el problema mencionado anteriormente fue propuesta en [53], que consiste en el entrenamiento de un clasificador Adaboost (sección 2.5.2) por cada *track*.

La idea principal es la de aprender a diferenciar al *track* i de los demás a lo largo de los *frames*, utilizando la información de color.

Los clasificadores débiles necesarios para el método de Adaboost se obtienen a partir del histograma tridimensional $\mathbb{H}$ (sección 2.4.2) en el espacio RGB de $[0, B] \times [0, B] \times [0, B]$, donde B es el número de bines por canal. Luego, para cada *track* i se considera un clasificador Adaboost con las siguientes características:

- Para cada clasificador débil $h_{\mathrm{i},t}$ se selecciona un paralelepípedo aleatorio dentro del histograma tridimensional asociado a la detección del *track* i. El paralelepípedo resultante cubre un rango de bines B_R, B_G, B_B para los canales R, G, B, respectivamente. Luego, la característica calculada $f_{\mathrm{i},t}$ consiste en la sumatoria de todos los elementos del histograma pertenecientes al paralelepípedo.

$$f_{\mathrm{i},t}(\mathbb{H}_{\mathrm{i}}, B_R, B_G, B_B) = \sum_{r \in B_R} \sum_{g \in B_G} \sum_{b \in B_B} \mathbb{H}_{\mathrm{i}}(r, g, b) \tag{3.3}$$

  De esta forma es necesario calcular sólo una vez por *frame* el histograma de la persona detectada.
- Cada clasificador débil $h_{\mathrm{i},t}$ es una medida de disimilitud entre la característica $f_{\mathrm{i},t}$ asociada al *track* i; y la característica $f_{j,t}$ asociada a la nueva detección j. Particularmente se utiliza la distancia euclidiana normalizada

$$h_{\mathrm{i},t}(j) = \frac{\sqrt{\left(f_{\mathrm{i},t}\left(\mathbb{H}_{\mathrm{i}}, B_R, B_G, B_B\right) - f_{j,t}\left(\mathbb{H}_{\mathrm{i}}, B_R, B_G, B_B\right)\right)^2}}{\sqrt{f_{\mathrm{i},t}\left(\mathbb{H}_{\mathrm{i}}, B_R, B_G, B_B\right)^2 + f_{j,t}\left(\mathbb{H}_{\mathrm{i}}, B_R, B_G, B_B\right)^2}} \tag{3.4}$$

- Para la fase de entrenamiento se utiliza el histograma del *track* i como muestra positiva, mientras que para muestras negativas se utilizan detecciones no asociadas al *track* en cuestión. Así se seleccionan solamente histogramas de colores candidatos a asociar entre *tracks*, en lugar de histogramas provenientes de regiones aleatorias en la imagen.

3.2.3. Predicción movimiento de *tracks*

Predecir el movimiento de las personas es importante para realizar una buena asociación de información, con el fin de efectuar correspondencias correctas entre los *tracks* y las nuevas detecciones. Con el fin de ser robusto a las oclusiones parciales y totales, se utiliza un modelo de velocidad constante [54]. De esta forma, si el *track* desaparece por un corto periodo de tiempo, es posible reencontrarlo en su posición esperada antes de darlo por perdido.

Por otro lado, en lugar de utilizar el filtro de Kalman básico (sección 2.8), se utiliza el *Unscented Kalman Filter* (UKF) [55], ya que funciona para sistemas no lineales a una mayor performance que el filtro de Kalman básico y el extendido.

El estado a predecir $\hat{z}_k(\mathrm{i}, j)$ en el tiempo k, se compone de la posición del *track* i y la velocidad la detección j. Las coordenadas de posición y velocidad se proyectan en el plano

de tierra (sección 3.1.1), por lo que cada una posee dos dimensiones. Luego, el error residual asociado a la predicción del *track* i y la detección j se define como:

$$\tilde{z}_k = z_k(\mathrm{i}, j) - \hat{z}_k(\mathrm{i}, j) \tag{3.5}$$

Donde $z_k(\mathrm{i}, j)$ es la posición del *track*. Este error residual también se utiliza para calcular la distancia de Mahalanobis entre cada *track* y detección en la asociación de información.

3.3. Reconocimiento de objetivo

Este bloque es el encargado de discriminar al objetivo versus las demás personas trackeadas en el bloque de Tracking (sección 3.2). Para ello se utilizan diferentes características visuales obtenidas de la imagen RGBD, y se entrena un clasificador online en base a muestras positivas (del objetivo) y muestras negativas (de otras personas). Las muestras negativas se van acumulando en una lista de distractores [56] con el fin de entrenar constantemente el clasificador, aunque no aparezcan nuevas personas en los siguientes frames.

Para construir el clasificador online, primero se define una medida de discriminación para cada característica. En particular se utiliza la distancia inversa de Bhattacharyya [57], pues es una medida utilizada entre distribuciones de probabilidad similares [56, 58]. En la ecuación (3.6) se muestra la distancia al comparar dos histogramas normalizados $\mathbb{H}_0$, $\mathbb{H}_1$ de B bines, donde cada histograma corresponde a la f-ésima característica extraída de una muestra.

$$\mathrm{d}_f(\mathbb{H}_0, \mathbb{H}_1) = \sqrt{1 - \sum_{\mathrm{i}=0}^{B-1} \sqrt{\mathbb{H}_0(\mathrm{i})\mathbb{H}_1(\mathrm{i})}} \quad \in [0, 1] \tag{3.6}$$

Donde $\mathrm{d}_f \approx 0$ si $\mathbb{H}_0$ y $\mathbb{H}_1$ son similares, mientras que el valor unitario indica que no lo son. Con esta información se construye una medida de disimilitud entre dos muestras, considerando N características, como se observa a continuación:

$$\mathrm{di}\mathit{similitu}\mathrm{d} = \frac{1}{N} \sum_{f=1}^{N} \mathrm{d}_f(\mathbb{H}_0, \mathbb{H}_1) \quad \in [0, 1] \tag{3.7}$$

Sin embargo, la ecuación (3.7) otorga la misma importancia a cada característica al momento de discriminar dos muestras. Esto puede ser una ventaja para ambientes controlados, pero para situaciones reales a las que se expone un robot de servicio (cambios de iluminación, variación en la perspectiva, presencia de objetos deformables, etc.) es necesario incluir una componente adaptiva que proporcione de manera óptima las características. En la ecuación (3.8) se muestra la medida de disimilitud utilizada.

$$\text{di}\textit{similitu}\text{d} = \sum_{f=1}^{N} \omega_f \text{d}_f(\mathbb{H}_0, \mathbb{H}_1) \quad \in [0, 1] \tag{3.8}$$

$$\sum_{f=1}^{N} \omega_f = 1 \tag{3.9}$$

Donde ω_f corresponde al peso normalizado que se le da a la característica f, el cual se modifica de acuerdo a lo descrito en la sección 3.3.3.

Con esto se establecen las bases para el reconocimiento de un objeto en un ambiente dinámico. Específicamente para el reconocimiento de una persona se utiliza la configuración indicada subsecciones siguientes, es decir: los tipos de características a extraer, las muestras a almacenar en la base de datos, y el tipo de entrenamiento del clasificador.

3.3.1. Extracción de características

La extracción de características es una etapa importante en la clasificación, pues es donde se obtiene la información a utilizar a partir de los datos generados por la cámara RGBD. En la literatura relacionada al reconocimiento de personas se utiliza una variedad de características, por ejemplo de color, puntos de interés, de textura, morfológicas, etc. Sin embargo, debido a las restricciones de este trabajo no es posible utilizar las de mejor desempeño.

En primer lugar, el sistema debe funcionar en tiempo real (i.e. mayor a 10 fps), por lo que características con un alto costo computacional como SIFT [23] o SURF [59] son descartadas. De igual manera, características del dominio frecuencial como filtros de Gabor [60] requieren de funciones complejas y múltiples transformaciones que significan un gran costo computacional.

Otro punto importante es la pose del objetivo con respecto al robot. Se debe considerar que la persona puede estar tanto de frente como de espalda (ver Figura 3.8), implicando en la ausencia del rostro en la mayoría del tiempo, por lo que utilizar características útiles para reconocimiento facial [61, 62] no es una solución factible.

(a) Persona de frente. (b) Persona de espalda. (c) Persona lejana de frente.

Figura 3.8: Poses de una persona vistas por el robot.

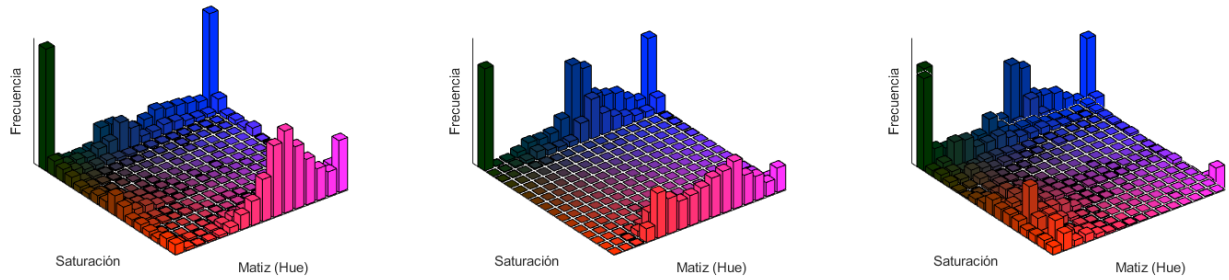
Las características utilizadas son histogramas de diversas transformaciones de la imagen correspondiente a una persona, con el fin de compararlos con la medida de disimilitud de la ecuación (3.8). Se utilizan 9 características de acuerdo al análisis de características para el seguimiento de personas realizado en [3]. En el Anexo B se presentan los resultados de dicho análisis. Las características son las siguientes:

- **Histograma HS**: Histograma bidimensional calculado para los canales H-S en el espacio de color HSV (sección 2.1.2).
- **Histograma V**: Histograma unidimensional calculado para el canal V en el espacio de color HSV (sección 2.1.2).
- **Histograma HS de la cabeza**: Histograma bidimensional calculado para los canales H-S en el espacio de color HSV (sección 2.1.2), considerando solamente la cabeza de la persona [1].
- **CSLBP**: Descriptor *Centre-Symmetric-LBP* (sección 2.4.5)[22], el cual se basa los descriptores LBP (sección 2.4.4) y SIFT [23], pero computacionalmente eficiente, siendo a su vez robusto a la iluminación.
- **CSLBP de la cabeza**: Descriptor *Centre-Symmetric-LBP*, considerando solamente la cabeza de la persona [1].
- **Densidad de bordes**: Descriptor de textura que contabiliza la cantidad de bordes en la imagen. Para obtener los bordes se utiliza el detector de Canny (sección 2.4.3). Posteriormente se utiliza una ventana deslizante de 9×9 para contabilizar la cantidad de bordes presentes. Para finalizar, se construye un histograma en base a la cantidad de bordes, donde si la cantidad de bordes es mayor que 31, se trunca a ese valor. Esto se realiza heurísticamente ya que es poco probable que haya una alta densidad de bordes dentro de la ventana.
- **Descriptor de bordes**: Descriptor que representa la distribución de 5 tipos de bordes (vertical, horizontal, diagonal-45°, diagonal-135°, no-direccional) [63]. Para obtener el descriptor, la imagen se divide en 1100 bloques; cada bloque se divide en 4 cuadrantes que representan a un superpixel obtenido con la media de los pixeles dentro del cuadrante; y finalmente, comparando los superpixeles, se obtiene el tipo de borde predominante. Con lo anterior, se genera un histograma de 5 bines (uno por cada tipo de borde), que corresponde al descriptor de bordes.
- **Histograma de Gradiente en eje X**: Histograma unidimensional obtenido del gradiente en el eje X (sección 2.4.1) de la imagen en escala de grises.
- **Histograma de Gradiente en eje Y**: Histograma unidimensional obtenido del gradiente en el eje Y (sección 2.4.1) de la imagen en escala de grises.

En la Figura 3.9 se muestran visualizaciones algunas de las características extraídas, donde se aprecia la importancia de cada una de ellas para diferenciar entre múltiples personas.

(a) Imágenes RGB de personas detectadas.

(b) Filtrado de Sobel (sección 2.4.1) para visualizar los gradientes en eje X e Y.



(c) Histogramas HS.

Figura 3.9: Características extraídas a 3 personas. Para texturas similares se observa una similitud en el filtrado de Sobel, mientras que para colores similares se observa una similitud en el histograma HS.

3.3.2. Almacenamiento de muestras

Este bloque es el encargado de almacenar las muestras de entrenamiento para el reconocimiento del objetivo. Como el clasificador se va entrenando de forma online, es necesario ir obteniendo muestras positivas (del objetivo) y negativas (distractores) a lo largo del tiempo, en los distintos *fames* obtenidos por la cámara RGBD.

Cada muestra corresponde a la imagen de color de un *track* (sección 3.2), en donde se utiliza una máscara binaria generada por la información de profundidad (sección 2.2) para

no considerar la información de color del *background*, como se muestra en la Figura 3.10. Se tiene un conjunto de reglas para seleccionar las muestras, de forma que aporten información útil para el reconocimiento:

1. Si en el *frame* actual está presente el *track* correspondiente al objetivo, y no hay muestras positivas almacenadas, entonces el *track* actual se utiliza para almacenar una muestra positiva.
2. Si en el *frame* actual está presente el *track* correspondiente al objetivo, entonces los demás *tracks* se utilizan para almacenar muestras negativas, siempre que sus imágenes de color no se traslapen con la imagen de color del objetivo.
3. Si el *track* está presente en el *frame*, y su disimilitud es mayor a un umbral τ_{low}, con respecto a la última muestra positiva almacenada, entonces se utiliza el *track* para almacenar una muestra positiva.
4. Si no está presente el objetivo en el *frame*, todos los *tracks* que tengan una medida de disimilitud (ecuación 3.8) mayor a un umbral τ_{high}, se consideran muestras negativas.
5. El número de muestras a almacenar es limitado, ya sea para las positivas como para las negativas. El criterio de eliminación de muestras para que se puedan almacenar las más recientes es el de eliminar las más antiguas, lo que permite adaptarse a los cambios de iluminación a medida que el robot se va desplazando.

(a) Persona detectada. (b) Máscara binaria de la persona. (c) Persona detectada considerando la máscara.

Figura 3.10: Utilización de máscara binaria para ignorar el *background*.

3.3.3. Entrenamiento online

El algoritmo de entrenamiento se basa en tener una lista amplia de muestras negativas (distractores), mientras que la lista de muestras positivas es pequeña. Para cada distractor se calcula la disimilitud de cada característica de la sección 3.3.1, con lo que se va entrenando el clasificador que reconoce al objetivo.

Para ello se define un puntaje $score_f$ con el que comparar por separado las características

del objetivo con las de los distractores. El objetivo de $score_f$ es maximizarlo al comparar las muestras positivas almacenadas con el *track* asociado al objetivo, y minimizarlo al comparar las las muestras positivas con las muestras negativas [56]. Con lo anterior se tiene:

$$score_f = \tilde{d}_f(\mathbb{H}_{pos}, \mathbb{H}_{distractor}) - \tilde{d}_f(\mathbb{H}_{pos}, \mathbb{H}_{obj}) \tag{3.10}$$

Donde $\tilde{d}_f(\mathbb{H}_{pos}, \mathbb{H}_{obj})$ es la media aritmética de la distancia de Bhattacharyya (ecuación (3.6)) entre los histogramas de las muestras positivas *pos* y la muestra del *track* actual del objetivo *obj*, para la característica f-ésima. De forma similar, $\tilde{d}_f(\mathbb{H}_{pos}, \mathbb{H}_{distractor}$ es la media aritmética entre los histogramas de las muestras positivas *pos*, y las muestras negativas di*stractor*.

El clasificador online corresponde a la medida de disimilitud de la ecuación (3.8). En cada *frame* se van actualizando los pesos ω_f, dándole importancia a unas características sobre otras, dependiendo de la información observada en la imagen del objetivo y las muestras negativas. Por ejemplo, si todas las personas (incluida el objetivo) presentes en el *frame* están vestidas con un mismo color pero diferentes estilos (Figura 3.11), entonces el clasificador priorizará las características de textura en lugar de las características de color.

Figura 3.11: Objetivo y distractor con colores similares.

Luego, el clasificador online se va entrenando en cada *frame* de acuerdo a lo siguiente:

1. Se actualizan las muestras positivas y negativas.
2. Se obtienen los $score_f$ para cada característica.
3. Se ajustan los pesos de acuerdo a:

$$\omega'_f = \omega_f + \alpha \cdot score_f \tag{3.11}$$

   Donde α es un parámetro de aprendizaje elegido a priori.
4. Una vez se ajusten todos los pesos, éstos se normalizan para que cumplan con la condición de la ecuación (3.9).

$$\omega_f = \frac{\omega'_f}{\sum_{f=1}^{N} \omega'_f} \quad \forall f \in \{1, ..., N\} \tag{3.12}$$

5. El clasificador resultante que reconoce al objetivo es:

$$\text{disimilitud}(track) = \sum_{f=1}^{N} \omega_f \mathrm{d}_f(\mathbb{H}_{last\_pos}, \mathbb{H}_{track}) \tag{3.13}$$

Donde $\mathbb{H}_{last\_pos}$ es el histograma de la característica f relacionado a la última muestra positiva agregada, y $\mathbb{H}_{track}$ es el histograma de la característica f asociada al *track* a evaluar.

En la Figura 3.12 se muestra el diagrama del proceso de entrenamiento para el reconocimiento del objetivo.

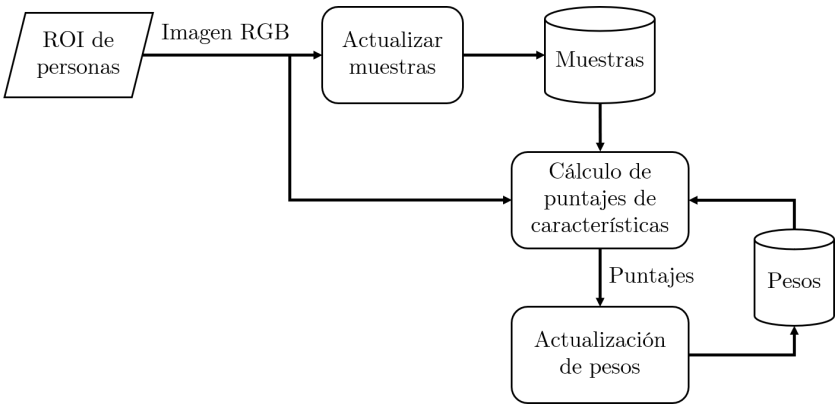


Figura 3.12: Diagrama de entrenamiento del clasificador online.

3.4. Actuación del Robot

Las secciones anteriores son utilizadas para detectar y localizar al objetivo, pero no de realizar acciones de movimiento. Para ello se tiene este módulo, siendo el encargado de decidir el comportamiento del robot al recibir el *track* del objetivo.

Con el fin de definir el comportamiento, existe un conjunto de reglas a considerar para generar un diagrama de flujo general (Figura 3.13). Éstas indican cuándo debe trasladarse el robot, si se debe permanecer a una distancia específica del objetivo, entre otras. Las principales reglas son las siguientes:

- Para inicializar el seguimiento del objetivo, primero se debe seleccionar a éste. Para ello se puede utilizar una señal externa, o un criterio predefinido por el robot. En este caso particular, se elige a la persona que esté frente a Pepper, y menor a una distancia predefinida.
- El robot está recibiendo constantemente el *track* relacionado al objetivo. Junto con esto, debe ser notificado si el objetivo se perdió.
- La cabeza de Pepper (que contiene a la cámara RGBD) siempre tiene la misma inclinación con respecto al piso, ya que así el plano de tierra (sección 3.1.1) no variará demasiado entre los *frames* recibidos por el sistema.

- La cabeza y el cuerpo de Pepper pueden rotar sobre su eje para seguir al objetivo. Esto se realiza cuando el objetivo está en un rango cercano del robot, donde no es necesario trasladarse para el seguimiento.
- Cuando se pierda al objetivo en los módulos de las secciones anteriores, se inicia el procedimiento de búsqueda:
  - Si el objetivo se pierde en un rango de distancia menor a un umbral, entonces rotar sobre su eje en la dirección correspondiente a la última detección, y verificar si encuentra al objetivo. Esto cubre los casos en que el objetivo sale del campo de visión del robot, pero no se encuentra ocluido.
  - Si no se cumple la condición anterior, o si no se detecta al objetivo con la condición anterior, entonces ir a la última posición donde detectó al objetivo.
- En caso de pérdida del objetivo, el módulo de reconocimiento del objetivo (sección 3.3) está constantemente verificando los *tracks* por si logra encontrarlo.
- Si el objetivo se perdió por más de un determinado tiempo, solicitar ayuda y reinicializar desde la selección del objetivo.

Para que Pepper se traslade por el mundo se utilizan librerías ya implementadas por el equipo de Robótica del Departamento de Ingeniería Eléctrica de la Universidad de Chile, las que poseen una capa de seguridad para evitar colisiones con obstáculos detectados con el sensor láser (sección 2.3). Una alternativa a esto es realizar el movimiento del robot mediante campos potenciales [64], asignando al objetivo como partícula atractiva, y los obstáculos como partículas repulsivas.

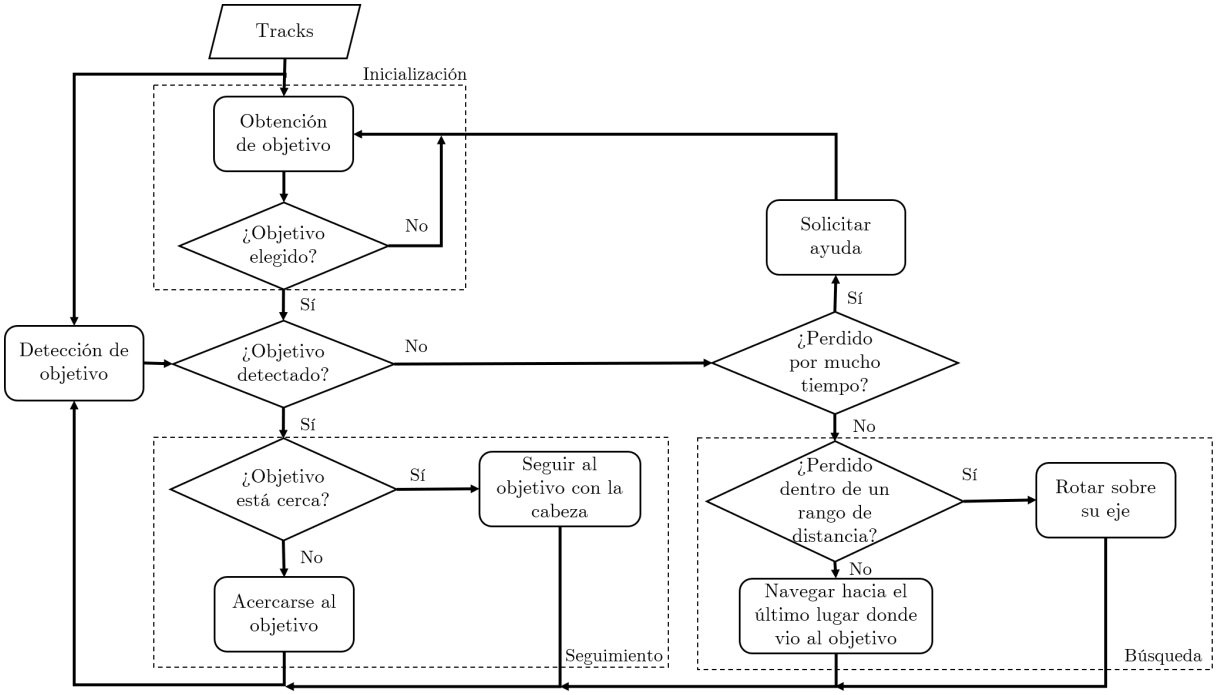


Figura 3.13: Diagrama de flujo de la actuación de Pepper.

Capítulo 4

Resultados

Por la modularidad del sistema implementado en la sección anterior (ver Figura 3.1), se pueden realizar pruebas en los diferentes bloques. A su vez, los resultados de los módulos dependen de los módulos precedentes, pues la finalidad es obtener resultados del sistema completo. También se considera que el formato de las pruebas debe ser similar a casos de seguimiento reales realizados por el robot, esto es, que sea en ambientes de interior; la persona a seguir esté a una distancia aceptable del robot ya sea de frente o de espalda; la velocidad de la persona no supera a la máxima velocidad de navegación del robot de servicio; etc.

Las pruebas se ejecutaron en un notebook que posee un procesador Intel(R) Core(TM) i5-6200U CPU 2.30GHz, 8 GB de memoria RAM, sistema operativo Ubuntu 14.04, con la distribución de ROS Indigo. El notebook se conectó vía Wi-Fi 5G al robot de servicio Pepper con el fin de enviar y recibir información por este canal. Por problemas de ancho de banda, desde Pepper hacia el notebook se enviaron la imágenes de color y profundidad en un tamaño de 160x120 pixeles, donde todo el procesamiento posterior (incluida la obtención de la nube de puntos) se realizó en el notebook, mediante código C++ ejecutado sobre ROS [65]. Posteriormente, desde el notebook se enviaron los comandos de movimiento para la navegación de Pepper.

Para probar diferentes situaciones que se pueden dar en el seguimiento de una persona, se tiene un conjunto de videos RGB-D con variadas características, pero manteniendo las restricciones de formato descritas previamente. En la sección 4.1 se muestra una descripción detallada de la base de datos.

El sistema logró una velocidad de procesamiento aproximada de 16 fps en el módulo de detección, mientras que en su totalidad tuvo una media aproximada de 12 fps, lo que para el caso particular del seguimiento de una persona, se puede considerar en tiempo real. En las secciones siguientes se muestran los resultados cuantitativos de los diferentes bloques del sistema, en base a los videos de la Tabla 4.1.

4.1. Descripción de la Base de Datos

El conjunto de videos RGB-D utilizados para la evaluación del sistema contiene 27 grabaciones. Los videos VD1, VD2 y VD3 se adquirieron en [2]; utilizando al robot de servicio Bender de la Universidad de Chile [66]. Los videos restantes, etiquetados como VAx con $x \in \{01, 02, ..., 24\}$, se adquirieron durante el proceso de implementación del sistema de este trabajo; utilizando sensores RGB-D, el robot Bender, y el robot Pepper.

En la tabla 4.1 se muestra una breve descripción de cada video, mientras que en la Figura 4.1 se muestran *frames* extraídos de la base de datos. Las situaciones recreadas se representan por un conjunto de definiciones, que se muestran a continuación:

- **PBG: Personas en el BackGround**. Corresponde a personas presentes en la escena, pero consideradas background por estar lo suficientemente lejos.
- **OC: Oclusiones**. Presencia de oclusiones hacia el objetivo. Pueden ser ocasionadas por otras personas u objetos presentes en la escena.
- **IOP: Interacción Objetivo-Personas**. El objetivo se relaciona con otras personas, ya sea manteniendo una conversación o un contacto físico.
- **MAO: Movimiento Aleatorio de Objetivo**. El objetivo no sigue una trayectoria recta de un punto a otro, si no que se mueve aleatoriamente por la escena.
- **RE: Robot Estático**. El robot no se desplaza ni rota sobre su eje.
- **DE: Desaparece de Escena en instantes**. El objetivo desaparece de la escena por breves instantes.
- **PDAO: Presencia de Distractores en Ausencia de Objetivo**. En la escena se encuentran personas, aún cuando el objetivo no lo está,
- **CVOE: Cambio de Vestimenta de Objetivo en Escena**. El objetivo cambia de vestimenta dentro de la escena del robot.
- **VSP: Vestimenta Similar entre Personas**. El objetivo y otra persona están vestidas de forma similar, según el criterio humano.

Figura 4.1: Ejemplos de *frames* RGB-D del conjunto de videos.

Tabla 4.1: Conjunto de videos utilizados para obtener resultados.

PBG: Personas en el BackGround. OC: Oclusiones. IOP: Interacción Objetivo-Personas. MAO: Movimiento Aleatorio de Objetivo. RE: Robot Estático. DE: Desaparece de Escena en instantes. PDAO: Presencia de Distractores en Ausencia de Objetivo. CVOE: Cambio de Vestimenta de Objetivo en Escena. VSP: Vestimenta Similar entre Personas.

Video	Duración [m:s]	Número de personas	Número de oclusiones de objetivo	Descripción
VD1	1:27	>20	0	PBG, obtenido de [2]
VD2	1:24	>20	7	PBG, OC, obtenido de [2]
VD3	2:19	>20	1	PGB, OC, IOP, obtenido de [2]
VA01	0:31	1	0	MAO, PGB, RE, DE
VA02	0:32	1	0	MAO, RE, DE
VA03	0:31	2	0	RE, PDAO
VA04	0:32	2	0	RE, PDAO
VA05	0:18	2	2	RE, OC
VA06	0:19	1	0	RE, CVOE
VA07	0:26	1	0	RE, CVOE
VA08	0:19	2	5	RE, MAO, OC
VA09	0:09	2	2	RE, OC
VA10	0:24	2	3	RE, DE
VA11	0:14	2	1	RE, DE, PDAO
VA12	0:32	2	2	RE, DE, PDAO
VA13	0:53	2	0	MAO, RE, DE
VA14	1:34	3	2	RE, CVOE, DE, OC, PDAO
VA15	0:39	3	2	MAO, RE, DE, OC, PDAO
VA16	0:55	1	0	DE
VA17	1:45	5	4	DE, OC, PDAO
VA18	3:52	1	0	DE
VA19	1:36	6	2	DE, OC
VA20	2:29	3	2	DE, OC
VA21	3:04	4	4	DE, OC, PDAO
VA22	1:13	4	2	DE, OC, PDAO, VSP
VA23	0:42	1	0	MAO, RE, DE
VA24	4:04	6	1	DE, OC, IOP

4.2. Desempeño en la Detección de Personas

El primer módulo, correspondiente a la detección de personas, se evaluó mediante la tasa de detección DR, complementada con la precisión PR del detector. En las ecuaciones (4.1, 4.2) se muestran las definiciones de DR y PR, respectivamente.

$$DR = \frac{TP}{TP + FN} \tag{4.1}$$

$$PR = \frac{TP}{TP + FP} \tag{4.2}$$

Donde TP es la cantidad de verdaderos positivos (personas detectadas correctamente), FN es la cantidad de falsos negativos (personas no detectadas), y FP es la cantidad de falsos positivos (detecciones incorrectas de personas). En la Tabla 4.2 se muestran los resultados obtenidos para cada video.

Se observa que la tasa de detección es inferior a las alcanzadas con los métodos tradicionales (sección 3.1), con tasas cercanas al 50 % en videos con muchas personas, mientras que la media es de un 83,63 %. Esto se contrasta con la alta velocidad alcanzada por el método implementado, ya que los métodos tradicionales requieren una gran capacidad de cómputo para detectar a la mayoría de las personas y así alcanzar grandes desempeños.

La principal causa del bajo desempeño es la reducción de regiones de interés donde se buscan personas, ya que la nube de puntos generada por el sensor RGB-D no representa a todas las personas con suficientes puntos (ver Figura 4.2), ya sea por oclusiones parciales, por estar a grandes distancias de las personas, interferencia infrarroja, etc.

Figura 4.2: Personas no seleccionadas como Región de Interés para la detección.

Por otro lado, la detección del objetivo alcanza un desempeño mucho mayor. Esto se debe a que el sistema está enfocado en detectar al objetivo en la mayoría de los casos, donde éste se encuentra frente al robot dentro de un rango de distancias acotado, en lugar de detectar a todas las personas dentro del campo de visión.

Tabla 4.2: Desempeño de la detección de personas.

Video	Tasa de Detección de Personas [%]	Precisión [%]	Tasa de Detección de Objetivo [%]
VD1	51.21	97.89	97.83
VD2	54.26	98.37	96.24
VD3	54.55	98.51	98.76
VA01	68.33	82.00	95.34
VA02	90.00	95.74	90.00
VA03	87.30	91.67	81.25
VA04	93.22	94.82	88.89
VA05	87.50	91.30	80.00
VA06	87.10	87.10	87.10
VA07	100	90.32	100
VA08	90.77	96.72	85.29
VA09	84.36	90.00	100
VA10	87.50	94.59	87.50
VA11	87.88	96.67	88.00
VA12	95.65	100	90.63
VA13	87.01	97.10	82.35
VA14	86.67	92.86	85.22
VA15	84.75	100	84.75
VA16	85.53	97.01	85.53
VA17	85.90	99.26	86.25
VA18	88.55	95.29	88.55
VA19	77.84	94.38	95.08
VA20	85.93	91.87	95.80
VA21	80.37	93.53	91.28
VA22	96.12	93.23	95.51
VA23	89.71	83.56	89.71
VA24	89.89	86.70	96.36
Media	**83.63**	**93.72**	**90.49**
Std	**12.40**	**4.78**	**5.87**

4.3. Desempeño en el Tracking

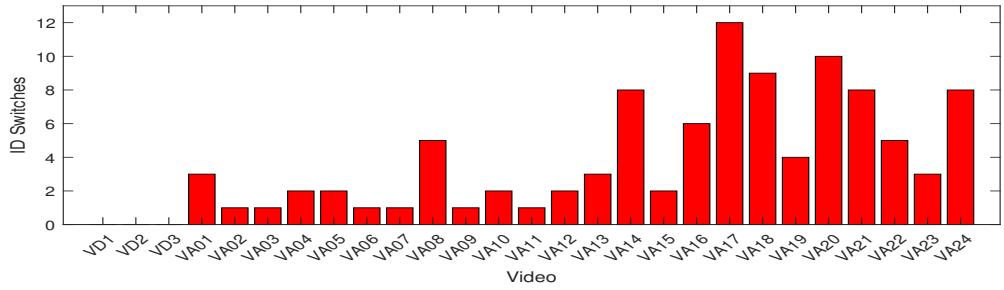
El segundo módulo, correspondiente al tracking de personas, se evaluó mediante la cantidad de *ID Switches* y la métrica MOTA [67]. Los *ID Switches* indican la cantidad de identidades diferentes asignadas a una misma persona, ya sea por un intercambio de identidad (correspondiente a los *tracks*) entre dos o más personas, o por la creación de un nuevo *track* para una persona que ya tenía un *track* asignado. Mientras que la métrica MOTA, para el caso de tracking de múltiples objetos, contabiliza los errores cometidos en el tracking (*ID Switches*), falsos positivos, y falsos negativos sobre todos los *frames* de video [68].

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + ID\_SW_t)}{\sum_t G_t} \tag{4.3}$$

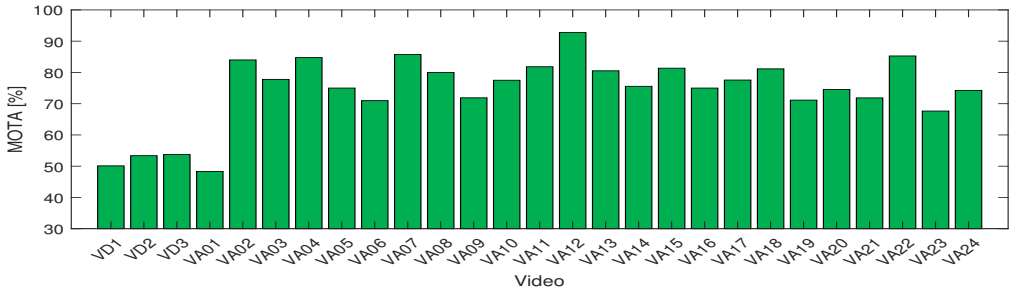
Donde para el *frame* t, G_t es el número de personas presente, FN_t el número de falsos negativos, FP_t el número de falsos positivos, y ID_SW_t el número de *ID Switches*. Un desempeño ideal es $ID\ Switches = 0$, y $MOTA = 100\,\%$.

En la Figura 4.3 se muestran los resultados obtenidos al evaluar el módulo del tracking en los videos disponibles. Se observa que generalmente existe una alta cantidad de *ID Switches*, pues este módulo no es capaz de re-encontrar personas identificadas en momentos anteriores pero que dejaron de ser vistas por instantes (e.g. oclusiones). La causa de esto es que no se puede asociar la posición y velocidad antigua con la nueva, ya que hay un lapso de tiempo en que no se tiene información de la persona. Además, rara vez puede suceder que, si dos personas muy cercanas acaban de ser detectadas y asignadas a dos nuevos *tracks*, puede que el clasificador de color (sección 3.2.2) no alcance a entrenarse correctamente, ocasionando errores en la asociación entre ambas personas.

Por otro lado, el valor del MOTA no es cercano al 100 % debido al desempeño del módulo anterior, que no es capaz de detectar a la totalidad de las personas pues está implementado con la finalidad de detectar al objetivo en lugar de todas las personas.



(a) ID Switches de cada video.



(b) MOTA obtenido para cada video.

Figura 4.3: Resultados del módulo de tracking de personas.

4.4. Desempeño en el Reconocimiento del Objetivo

El último módulo de procesamiento de imágenes, corresponde al reconocimiento del objetivo. Para este caso se evaluó si se pudo re-encontrar al objetivo luego de perderlo, junto con la cantidad de tiempo que se demoró en re-encontrarlo. Para un seguimiento exitoso del objetivo, éste debe ser reencontrado a la brevedad, y nunca confundirse con otra persona.

Las métricas utilizadas son el Ratio de reencuentro, y el tiempo medio no reconocido. La primera métrica corresponde a la fracción de re-encuentros logrados, con respecto a la cantidad de veces que se perdió el objetivo. La segunda métrica corresponde al tiempo medio desde que el objetivo perdido vuelve a aparecer en el campo de visión del robot, hasta que es re-encontrado. La tercera métrica es el Cambio de Objetivo, que indica cuántas veces se identificó a otra persona como el objetivo, ocasionando que el seguimiento del objetivo falle.

En la Tabla 4.3 se muestran los resultados obtenidos al evaluar los videos en el sistema. Se observa que en la mayoría de las veces se re-encuentra al objetivo cada vez que se pierde, y el tiempo requerido para esto es menos a 1 segundo, lo que es aceptable para la ejecución del sistema en tiempo real y una rápida reacción del robot.

Tabla 4.3: Desempeño del reconocimiento del objetivo.

Video	Ratio de Reencuentro [%]	Tiempo Medio de Objetivo No Reconocido[s]*	Cambio de Objetivo
VD1	100	0.00	0
VD2	100	0.00	0
VD3	100	0.00	0
VA01	100	1.00	0
VA02	100	0.00	0
VA03	100	1.00	0
VA04	100	0.50	0
VA05	50.00	5.50	0
VA06	100	0.00	0
VA07	100	5.50	0
VA08	100	1.00	0
VA09	100	4.00	0
VA10	100	0.50	0
VA11	100	5.00	0
VA12	100	1.75	0
VA13	100	1.50	0
VA14	75.00	7.00	0
VA15	100	1.25	0
VA16	100	1.08	0
VA17	83.33	3.83	2
VA18	100	0.50	0
VA19	100	0.50	0
VA20	100	0.69	1
VA21	100	1.57	0
VA22	100	0.50	1
VA23	100	0.50	0
VA24	100	0.81	0
Media	**92.01**	**1.90**	**0.17**
Std	**22.72**	**2.03**	**0.48**

* Las mediciones se realizaron cada 0.5 segundos, por lo que el tiempo mínimo perdido (distinto de 0) es de 0.5 [s].

Con el fin de comparar el sistema con métodos tradicionales, se generó un conjunto de gráficos que indican la detección del objetivo a lo largo de los *frames*. En particular, los videos VD1, VD2, y VD3 fueron evaluados anteriormente en [2] con diferentes métodos, alcanzando los mejores resultados con el detector de torsos de personas HOG1/2, pero a 0,41 fps (inviable para ejecutar en tiempo real) en una máquina de características similares a un Pentium Dual-Core 2.80GHz, 4 GB de memoria RAM [69].

En la Figura 4.4 se muestran las comparaciones gráficas realizadas, donde se observa que el desempeño del sistema implementado es similar al mejor resultado de [2] y al *Ground Truth*, sin considerar que el sistema implementado tiene una velocidad mucho mayor. En los anexos se presentan las gráficas generadas para los demás videos.

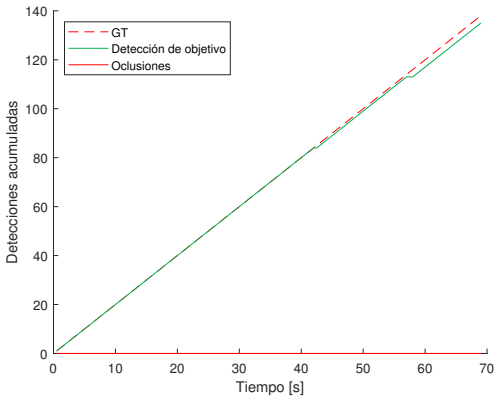
Junto con los gráficos, se compararon las tasas de detección DR del objetivo de la sección 4.2, con las obtenidas en [2] (ver Tabla 4.4). Con esto se confirma cuantitativamente que el método implementado puede competir con los mejores detectores evaluados, logrando a su vez funcionar en tiempo real.

Tabla 4.4: Tasas de detección del módulo de detección de personas implementado vs otros métodos.

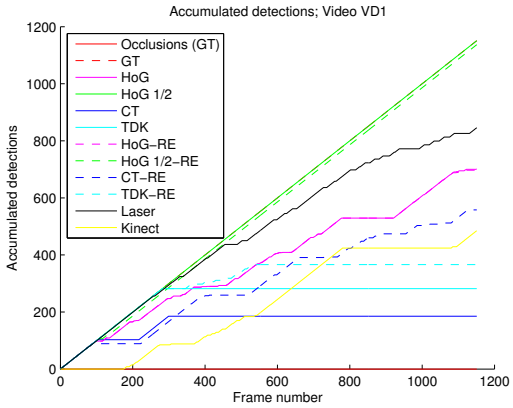
	VD1 - Tasa de Detección [%]	VD2 - Tasa de Detección [%]	VD3 - Tasa de Detección [%]	Media [%]	Std [%]	Tiempo de Procesamiento [fps]
Laser	73.50	68.42	9.10	50.34	35.81	1,507.91*
Kinect	42.05	58.08	37.39	45.84	10.85	77.52*
HOG	60.90	71.71	26.18	52.93	23.79	0.40*
HOG1/2	**100**	93.51	**98.94**	97.48	3.48	0.41*
TDK	24.50	16.82	23.32	21.55	4.14	14.40*
CT	16.07	17.67	27.18	20.31	6.01	31.12*
Sistema Implementado	97.83	**98.37**	98.76	**97.61**	1.27	12.42**

* Tiempo obtenido en una máquina de características similares a un Pentium Dual-Core CPU 2.80GHz, 4 GB de memoria RAM.

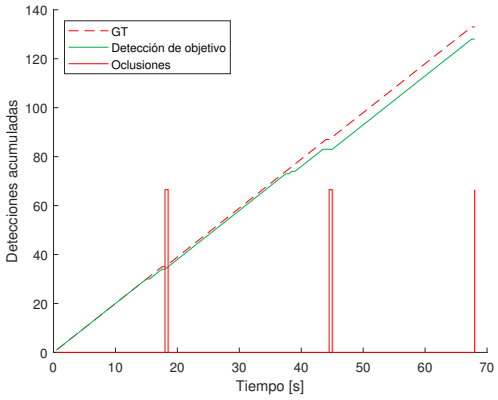
** Tiempo obtenido en una máquina Intel(R) Core(TM) i5-6200U CPU 2.30GHz, 8 GB de memoria RAM.



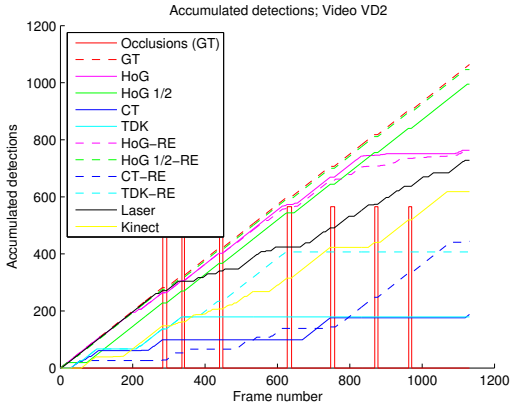
(a) VD1 con Sistema implementado.



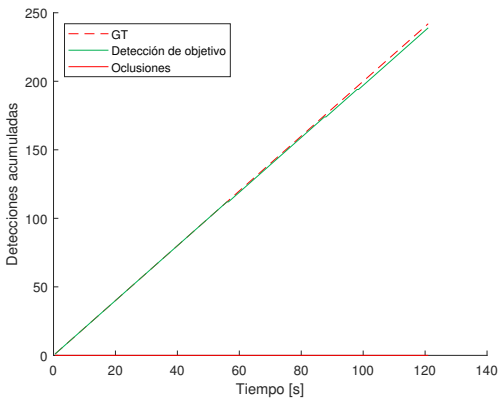
(b) VD1 con otros métodos.



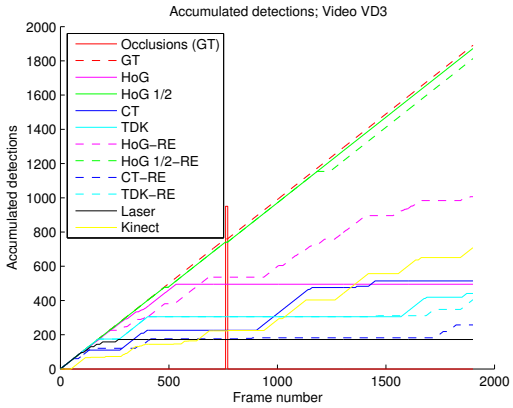
(c) VD2 con Sistema implementado.



(d) VD2 con otros métodos.



(e) VD3 con Sistema implementado.



(f) VD3 con otros métodos.

Figura 4.4: Comparación de [2] con el sistema implementado. Se indican las detecciones acumuladas del objetivo, y cuándo hay presencia de oclusiones. Los resultados varían levemente debido a la diferencia de la frecuencia de muestreo.

4.5. Actuación del Robot

El último módulo, encargado de la actuación del robot, solamente fue medido cualitativamente debido a las dificultades para realizar mediciones precisas en los escenarios. No obstante, se observó que el comportamiento definido cumplió con los requisitos para realizar un correcto seguimiento de la persona; cuando la persona se alejaba, Pepper avanzaba; cuando la persona se perdía fuera del rango de distancias cercanas, Pepper comenzaba la búsqueda e iba a la última posición donde vio a la persona; etc.

Es importante destacar que el movimiento de Pepper fue generalmente fluido, evitando la aparición de distorsiones en la imagen debido a movimientos bruscos (ver Figura 4.5). No obstante, el sistema resulta robusto a movimientos bruscos por cortos periodos de tiempo, ya que el *tracking* permite ausencia de la persona en algunos *frames*.

Figura 4.5: Distorsiones en la imagen por movimiento rápido del robot. Dificulta la detección del objetivo.

Por último, el módulo encargado de la navegación de Pepper resultó extremadamente importante, ya que es el encargado de evadir obstáculos y lograr un efectivo seguimiento del objetivo. Es por ello que, si no se tiene la capacidad de navegar a través de una gran cantidad de obstáculos, la escena por donde el robot de servicio realice el seguimiento de la persona debe adecuarse específicamente para la tarea del seguimiento.

Conclusión

El procesamiento digital de imágenes se puede utilizar para resolver múltiples problemas en diferentes industrias, como la minería, salud, manufactura, etc. Este trabajo se enfocó en el área de aplicaciones para los robots de servicio, convirtiendo a las imágenes en un sentido cognitivo de forma similar, en parte, a la del ser humano. En particular, en este trabajo se replicó la tarea de seguir, con el robot de servicio Pepper, a una persona que se mueve por el espacio, otorgándole al primero la posibilidad de, en un futuro, convertirse en un asistente personal ya sea para facilitar información, o incluso para transportar objetos.

Se logró implementar exitosamente un sistema de seguimiento de una persona, capaz de ejecutarse en tiempo real sin la necesidad de añadir una GPU como unidad de procesamiento, con una tasa de detección del objetivo que compite con los métodos clásicos del estado del arte (que no necesariamente son capaces de ejecutarse en tiempo real). Cabe destacar que el seguimiento de una persona es un problema no resuelto al 100 %, donde constantemente se están buscando nuevas soluciones. Esta es una de las razones por la que el sistema se dividió por módulos, de manera de facilitar la integración de métodos recientes con un mejor desempeño.

El problema se desglosó en 4 bloques principales, con lo que se entendió la importancia de cada uno de ellos. Por lo tanto, la solución se dividió en 4 módulos, cada uno con una función determinada. El primer módulo es el encargado de detectar al objetivo y a las demás personas; mientras más personas detecte, el siguiente bloque podrá tener un mejor desempeño. El segundo módulo asocia identidades a las detecciones obtenidas a lo largo del tiempo, siendo capaz de distinguir entre dos personas que se encuentran simultáneamente en el *frame*. El tercer módulo se enfoca en reconocer al objetivo cuando el módulo anterior perdió la identidad original del objetivo. El cuarto módulo es el que envía las instrucciones de movimiento, para que Pepper navegue en pos del objetivo. Es importante mencionar que los últimos dos módulos deben ser más estrictos pues si generan información incorrecta, pueden hacer que el seguimiento falle completamente. Por otro lado, los primeros módulos son más permisivos ya que los últimos actúan como filtros; por ejemplo, el sistema considera que el primer módulo puede fallar en la detección del objetivo en algunos *frames*, pues eventualmente volverá a detectarlo y reconocerlo.

El sistema generalmente funcionó como debía: es robusto a las oclusiones parciales y totales pues es capaz de re-identificar al objetivo aunque haya pasado una gran cantidad de tiempo, permite que haya cambios de iluminación en el recorrido de Pepper debido al entrenamiento online para discriminar al objetivo de las demás personas, y además se enfoca

en las características de objetivo que lo hacen diferente de los demás para evitar errores en el reconocimiento. Uno de los puntos críticos del sistema es la capacidad de re-identificar al objetivo, lo que generalmente se resuelve en menos de 1 segundo una vez que el objetivo vuelve a entrar en el campo de visión de Pepper, que se considera aceptable para la ejecución en tiempo real. No obstante, hubo algunas ocasiones en que no se logró reconocer al objetivo debido a variaciones considerables de iluminación (luego de reaparecer en el campo de visión de Pepper), y en un par de casos confundió al objetivo con otra persona a causa del breve tiempo en que se entrenó la discriminación del objetivo vs otras personas, y a la similitud entre sus características. Como conclusión de lo anterior, se tiene que se puede continuar iterando en el desarrollo del sistema, verificando qué conjunto de características es más útil para discriminar, re-entrenando el detector de personas sólo con imágenes obtenidas con Pepper, etc.

La solución al problema depende en gran medida de las características del robot, tanto de hardware como de software. En este caso se tiene una cámara RGBD a la altura de la cabeza de Pepper que es capaz de rotar en azimut y elevación, y sensores Laser a la altura de las pantorrillas. En base a los sensores anteriores se diseñó el sistema. Por ejemplo, se aprovechó la información de profundidad para disminuir el tiempo de procesamiento a pesar de las limitaciones de distancia e interferencia que posee.

Como trabajo futuro se propone fundamentalmente la utilización de redes neuronales convolucionales en los distintos módulos, ya que con técnicas emergentes se ha ido aumentando el *performance* con respecto a las métricas de detección y al tiempo de procesamiento requerido, sobre todo con aceleradores como la GPU. Incluso se pueden utilizar las redes para la extracción de características dedicadas al tracking y reconocimiento de objetos. La integración de esta tecnología es factible, ya que se puede aprovechar el proyecto de la mochila de Pepper que le añade capacidad de cómputo.

En este trabajo no se aprovechó la retroalimentación de la navegación de Pepper con la visión computacional que reconoce al objetivo. Esto da la posibilidad de explorar mejoras en la localización del robot con respecto al objetivo, ya sea llevando todas las posiciones relativas a un mapa global, o incluso para prevenir que el objetivo salga del campo de visión del robot.

Por último, se propone generar una gran base de datos con situaciones reales en las que se puede ver afectado Pepper, con lo que se tendría un conjunto de datos idóneo para el entrenamiento del sistema, maximizando su desempeño específicamente para el robot de servicio.

Bibliografía

[1] M. Munaro, F. Basso, and E. Menegatti. Tracking people within groups with rgb-d data. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2101–2107, Oct 2012.

[2] Wilma Pairo, Javier Ruiz-del Solar, Rodrigo Verschae, Mauricio Correa, and Patricio Loncomilla. Person following by mobile robots: Analysis of visual and range tracking methods and technologies. In Sven Behnke, Manuela Veloso, Arnoud Visser, and Rong Xiong, editors, *RoboCup 2013: Robot World Cup XVII*, pages 231–243, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[3] V. Alvarez-Santos, X. M. Pardo, R. Iglesias, A. Canedo-Rodriguez, and C. V. Regueiro. Feature analysis for human recognition and discrimination: Application to a person-following behaviour in a mobile robot. *Robot. Auton. Syst.*, 60(8):1021–1036, August 2012.

[4] Floris Ernst Achim Schweikard. *Medical Robotics*. Springer International Publishing, first edition, 2015.

[5] C. Park, S. Kang, J. Kim, and J. Oh. A study on service robot system for elder care. In *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 546–547, Nov 2012.

[6] Softbank Robotics. Pepper, the humanoid robot from softbank robotics, a genuine companion. [En línea]. Disponible: `https://www.softbankrobotics.com/emea/en/robots/pepper/find-out-more-about-pepper`. Accedido: 04-Sep-2018.

[7] M. D. Fairchild. *Color Appearance Models*. John Wiley & Sons, Ltd, doi: 10.1002/9781118653128.ch1, third edition, 2013.

[8] Keith Jack. *Video Demystified: A Handbook for the Digital Engineer*. Newnes, fifth edition, 2007.

[9] S. H. Lai H. Y. Kuo, H. R. Su and C. C. Wu. 3d object detection and pose estimation from depth image for robotic bin picking. *2014 IEEE International Conference on Automation Science and Engineering (CASE), Taipei*, pages 1264–1269, 2014.

[10] M. S. Farid M. H. Khan, K. Shirahama and M. Grzegorzek. Multiple human detection in

depth images. *2016 IEEE 18th International Workshop on Multimedia Signal Processing (MMSP), Montreal, QC*, pages 1–6, 2016.

[11] M. D. Phung D. M. Nguyen V. M. Hoang T. H. Dinh, M. T. Pham and Q. V. Tran. Image segmentation based on histogram of depth and an application in driver distraction detection. *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV), Singapore*, pages 969–974, 2014.

[12] C. Shen Y. Cao and H. T. Shen. Exploiting depth from single monocular images for object detection and semantic segmentation. *IEEE Transactions on Image Processing*, 26(2), feb 2017.

[13] H. Ren Z. Hu, H. Ai and Y. Zhang. Fast human detection in rgb-d images based on color-depth joint feature learning. *2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ*, pages 266–270, 2016.

[14] R.D. Richmond and S.C. Cain. *Direct-detection LADAR systems*. SPIE Press, 01 2010.

[15] J. Canny. Computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[16] J. Lin and C. Chiu. Low-complexity face recognition using contour-based binary descriptor. *IET Image Processing*, 11(12):1179–1187, 2017.

[17] S. K. Setarehdan and J. J. Soraghan. Automatic cardiac lv boundary detection and tracking using hybrid fuzzy temporal and fuzzy multiscale edge detection. *IEEE Transactions on Biomedical Engineering*, 46(11):1364–1378, Nov 1999.

[18] M. S. Mohd Asaari, S. A. Suandi, and B. A. Rosdi. Geometric feature extraction by ftas for finger-based biometrics system. *IET Biometrics*, 6(3):157–164, 2017.

[19] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51 – 59, 1996.

[20] Timo Ahonen, Abdenour Hadid, and Matti Pietikäinen. Face recognition with local binary patterns. In Tomás Pajdla and Jiří Matas, editors, *Computer Vision - ECCV 2004*, pages 469–481, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[21] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, July 2002.

[22] Marko Heikkilä, Matti Pietikäinen, and Cordelia Schmid. Description of interest regions with center-symmetric local binary patterns. In *Computer Vision, Graphics and Image Processing*, pages 58–69, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[23] Huiyu Zhou, Yuan Yuan, and Chunmei Shi. Object tracking using sift features and mean shift. *Computer Vision and Image Understanding*, 113(3):345 – 352, 2009. Special Issue

on Video Analysis.

[24] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005.

[25] S. Kim and K. Cho. Trade-off between accuracy and speed for pedestrian detection using hog feature. In *2013 IEEE Third International Conference on Consumer Electronics ¿ Berlin (ICCE-Berlin)*, pages 207–209, Sept 2013.

[26] W. Park, D. Kim, Suryanto, C. Lyuh, T. M. Roh, and S. Ko. Fast human detection using selective block-based hog-lbp. In *2012 19th IEEE International Conference on Image Processing*, pages 601–604, Sept 2012.

[27] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, September 2010.

[28] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 1995.

[29] R. E. Schapire. The strength of weak learnability. In *30th Annual Symposium on Foundations of Computer Science*, pages 28–33, Oct 1989.

[30] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997.

[31] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, Dec 2001.

[32] Radu Bogdan Rusu. Semantic 3d object maps for everyday manipulation in human living environments. *KI - Künstliche Intelligenz*, 24(4):345–348, Nov 2010.

[33] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.

[34] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[35] S. M. Bozic. *Digital and Kalman Filtering*. Edward Arnold, fourth edition, 1986.

[36] S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, March 2004.

[37] Peter S. Maybeck. *Stochastic Models, Estimation and Control*, volume 1. Academic Press, 1979.

[38] H. W. Kuhn and Bryn Yaw. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–97, 1955.

[39] Munkres. [En línea]. Disponible: `http://csclab.murraystate.edu/~bob.pilgrim/445/munkres.html`. Accedido: 04-Ene-2019.

[40] A. Ankit, I. R. Ahmad, and H. Shin. A cascade framework for unoccluded and occluded pedestrian detection. In *Proceedings of the 2014 IEEE Students' Technology Symposium*, pages 62–67, Feb 2014.

[41] J. Kim, J. Lee, C. Lee, E. Park, J. Kim, H. Kim, J. Lee, and H. Jeong. Optimal feature selection for pedestrian detection based on logistic regression analysis. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 239–242, Oct 2013.

[42] X. Li, X. Fang, and Q. Lu. On-road vehicle and pedestrian detection using improved codebook model. In *Proceedings of 2013 IEEE International Conference on Vehicular Electronics and Safety*, pages 1–4, July 2013.

[43] H. Yoshida, D. Suzuo, D. Deguchi, I. Ide, H. Murase, T. Machida, and Y. Kojima. Pedestrian detection by scene dependent classifiers with generative learning. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 654–659, June 2013.

[44] J. Qu and Z. Liu. Non-background hog for pedestrian video detection. In *2012 8th International Conference on Natural Computation*, pages 535–539, May 2012.

[45] Marius Cordts, Timo Rehfeld, Lukas Schneider, David Pfeiffer, Markus Enzweiler, Stefan Roth, Marc Pollefeys, and Uwe Franke. The stixel world: A medium-level representation of traffic scenes. *Image and Vision Computing*, 68:40 – 52, 2017. Automotive Vision: Challenges, Trends, Technologies and Systems for Vision-Based Intelligent Vehicles.

[46] O. H. Jafari, D. Mitzel, and B. Leibe. Real-time rgb-d based people detection and tracking for mobile robots and head-worn cameras. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5636–5643, May 2014.

[47] Documentation - point cloud library (pcl). [En línea]. Disponible: `http://pointclouds.org/documentation/tutorials/voxel_grid.php`. Accedido: 03-Ene-2019.

[48] Edward T. Hall. *The hidden dimension / Edward T. Hall.* Doubleday Garden City, N.Y, [1st ed.] edition, 1966.

[49] Vittorio Ferrari Marcin Eichner. Calvin upper-body detector v1.04. [En línea]. Disponible: `http://groups.inf.ed.ac.uk/calvin/calvin_upperbody_detector/`. Accedido: 03-Ene-2019.

[50] E. Moussy, A. A. Mekonnen, G. Marion, and F. Lerasle. A comparative view on exemplar 'tracking-by-detection' approaches. In *2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, Aug 2015.

[51] Pavlina Konstantinova, Alexander Udvarev, and Tzvetan Semerdjiev. A study of a

target tracking algorithm using global nearest neighbor approach. In *Proceedings of the 4th International Conference Conference on Computer Systems and Technologies: E-Learning*, CompSysTech '03, pages 290–295, New York, NY, USA, 2003. ACM.

[52] R. De Maesschalck, D. Jouan-Rimbaud, and D.L. Massart. The mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1):1 – 18, 2000.

[53] Matteo Munaro and Emanuele Menegatti. Fast rgb-d people tracking for service robots. *Autonomous Robots*, 37(3):227–242, Oct 2014.

[54] Nicola Bellotto and Huosheng Hu. Computationally efficient solutions for tracking people with a mobile robot: an experimental evaluation of bayesian filters. *Autonomous Robots*, 28(4):425–438, May 2010.

[55] E. A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158, Oct 2000.

[56] V. Alvarez-Santos, X. M. Pardo, R. Iglesias, A. Canedo-Rodriguez, and C. V. Regueiro. Online feature weighting for human discrimination in a person following robot. In *Foundations on Natural and Artificial Computation*, pages 222–232, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[57] N. A. Thacker, F. J. Aherne, P. I. Rockett, F. J. Aherne, and P. I. Rockett. The bhattacharyya metric as an absolute similarity measure for frequency coded data. *Kybernetika*, 1997.

[58] T. Kailath. The divergence and bhattacharyya distance measures in signal selection. *IEEE Transactions on Communication Technology*, 15(1):52–60, February 1967.

[59] David Gossow, Peter Decker, and Dietrich Paulus. An evaluation of open source surf implementations. In *RoboCup 2010: Robot Soccer World Cup XIV*, pages 169–179, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[60] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba. Coding facial expressions with gabor wavelets. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 200–205, April 1998.

[61] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, Jan 1991.

[62] J. Daugman. How iris recognition works. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):21–30, Jan 2004.

[63] Dong Kwon Park, Yoon Seok Jeon, and Chee Sun Won. Efficient use of local edge histogram descriptor. In *Proceedings of the 2000 ACM Workshops on Multimedia*, MULTIMEDIA '00, pages 51–54, New York, NY, USA, 2000. ACM.

[64] Michael A. Goodrich. 1 introduction potential fields tutorial.

[65] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[66] Bender - uchile homebreakers. [En línea]. Disponible: `http://robotica-uchile.amtc.cl/bender-index.html`. Accedido: 06-Ene-2019.

[67] Keni Bernardin, Er Elbs, and Rainer Stiefelhagen. Multiple object tracking performance metrics and evaluation in a smart room environment.

[68] Samuel Murray. Real-time multiple object tracking - A study on the importance of speed. *CoRR*, abs/1709.03572, 2017.

[69] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang. Real-time compressive tracking. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part III*, ECCV'12, pages 864–877, Berlin, Heidelberg, 2012. Springer-Verlag.

Anexos

Anexo A

Parámetros utilizados

El sistema implementado utiliza una gran cantidad de parámetros, cuyos valores se eligieron como indica la literatura, o en otros casos, empíricamente dependiendo de su desempeño. Estos se encuentran ordenados de acuerdo al módulo al que pertenecen. En las siguientes secciones se muestran las descripciones los parámetros más relevantes, junto con su valor elegido.

A.1. Estimación de plano de tierra

Tabla A.1: Parámetros para la estimación de plano de tierra.

Nombre	Descripción	Valor
$voxel\_size$	Distancia mínima (en metros) entre puntos para filtrar la nube de puntos.	0.06
$axis\_y\_thr$	Umbral para decidir si el plano de tierra estimado es lo suficientemente horizontal. Esto sucede si se cumple la condición $B > axis\_y\_thr$, de acuerdo a la ecuación (3.1).	0.70
ε	Umbral de distancia (en metros) de un punto de la nube de puntos a un modelo del plano generado en RANSAC. Si la distancia es menor al umbral, entonces se considera ese punto como perteneciente al plano en cuestión.	0.20
$consensus$	La cantidad mínima de puntos que satisfacen la ecuación del plano (3.1) de acuerdo a ε, para que el modelo del plano sea aceptado con RANSAC.	200
τ	Fracción de *inliers* vs *outliers* mínima requerida para aceptar el modelo estimado con RANSAC.	0.15
N	Número de veces que RANSAC puede repetirse.	1000

A.2. Segmentación de nube de puntos

Tabla A.2: Parámetros para la segmentación de la nube de puntos.

Nombre	Descripción	Valor
d_th	Distancia máxima (en metros) entre puntos para pertenecer al mismo cluster.	0.10
max_points	Número máximo de puntos que pertenecen a un mismo cluster.	5000
min_points	Número mínimo de puntos que pertenecen a un mismo cluster.	30
d_$intima$	Distancia íntima (en metros). Distancia mínima entre cabezas.	0.30
h_max	Altura máxima (en metros) de una persona.	2.3
h_min	Altura mínima (en metros) de una persona.	1.3

A.3. Asociación de información

Tabla A.3: Parámetros para la asociación de información.

Nombre	Descripción	Valor
α	Ponderación para la distancia de Mahalanobis $D_M^{i,j}$	0.35
β	Ponderación para la confidencia de el clasificador de color $c_{online}^{i,j}$	-0.20
γ	Ponderación para la confidencia de el clasificador de personas $c_{HOG1/2}^{j}$	-0.18

A.4. Clasificador online de color para la asociación de información

Tabla A.4: Parámetros para el clasificador online para la asociación de información.

Nombre	Descripción	Valor
B	Número de bines por canal.	16
N	Número de clasificadores débiles.	36

A.5. Extracción de características

Tabla A.5: Parámetros para la extracción de características.

Nombre	Descripción	Valor
B_H	Número de bines del canal H para los histogramas HS.	18
B_S	Número de bines del canal S para los histogramas HS.	0
B_V	Número de bines del histograma V.	8
B_CSLBP	Número de bines para cada histograma CSBLP.	16
N_CSLBP	Número de histogramas CSLBP extraídos de la imagen, que se divide en celdas de igual tamaño.	16
B_den_bordes	Número de bines del histograma de densidad de bordes.	32
B_des_bordes	Número de bines del histograma del descriptor de bordes	5
$B\_gradiente$	Número de bines de los histogramas de gradiente X e Y.	32

A.6. Almacenamiento de muestras

Tabla A.6: Parámetros para el almacenamiento de muestras.

Nombre	Descripción	Valor
τ_low	Umbral para que se almacene una muestra positiva con mayor o igual disimilitud.	0.2
τ_high	Umbral para que se almacene una muestra negativa con mayor o igual disimilitud, cuando no está presente el objetivo.	0.5
N_neg	Número de muestras negativas almacenadas.	20
N_pos	Número de muestras positivas almacenadas.	5

A.7. Entrenamiento online

Tabla A.7: Parámetros para el entrenamiento online.

Nombre	Descripción	Valor
α	Tasa de aprendizaje para el ajuste de los pesos ω_f	0.5

A.8. Actuación del Robot

Tabla A.8: Parámetros para la actuación del robot.

Nombre	Descripción	Valor
d_*max*	Distancia máxima (en metros) a la que se debe encontrar una persona para convertirse en el objetivo.	2.5
pitch	Inclinación de la cabeza de Pepper (en radianes) con respecto al suelo.	0.03
d_*lost*	Distancia máxima (en metros) de pérdida del objetivo a la que Pepper sólo debe girar sobre su eje, en primera instancia.	2.0
timeout	Tiempo máximo (en segundos) para buscar al objetivo. Una vez cumplido el tiempo, el robot solicita ayuda.	8.0

Anexo B

Análisis de características para la discriminación del objetivo

En [3] se realiza un análisis de las características útiles para discriminar a una persona (objetivo) de las demás. Bajo la premisa de utilizar un reducido conjunto de características para la discriminación, se efectúa un análisis de información mutua entre las características (Figura B.2). La información mutua está definida como:

$$MI(X,Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) log \left(\frac{p(x,y)}{p_1(x)p_2(y)} \right) \tag{B.1}$$

Donde $p_1(x)$ es la probabilidad de que la característica X tome el valor x, $p_2(y)$ es la probabilidad de que la característica Y tome el valor y, y $p(x,y)$ es la probabilidad conjunta de obtener $X = x$, $Y = y$ en la misma imagen.

Luego se agrupan incrementalmente las características para calcular el f-score del conjunto. El orden de agrupación consiste en elegir la primera característica del ranking (heurísticamente), mientras que la segunda es la con menor información mutua con respecto a la primera. Para las demás características se utiliza la misma regla, es decir, la característica n-ésima se elige dependiendo de la información mutua con respecto a la característica $(n-1)$-ésima.

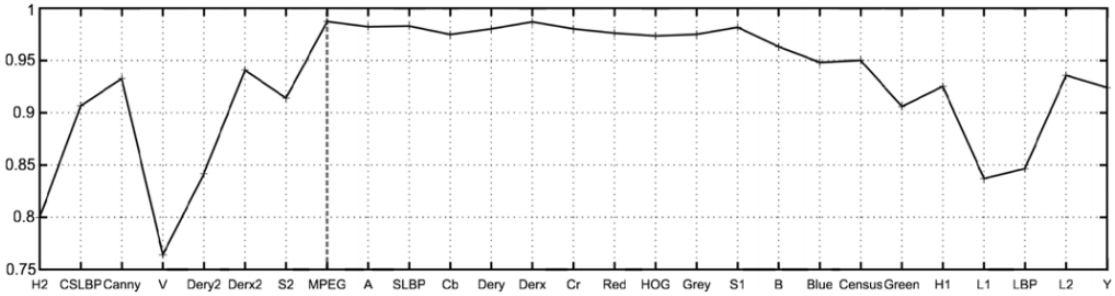


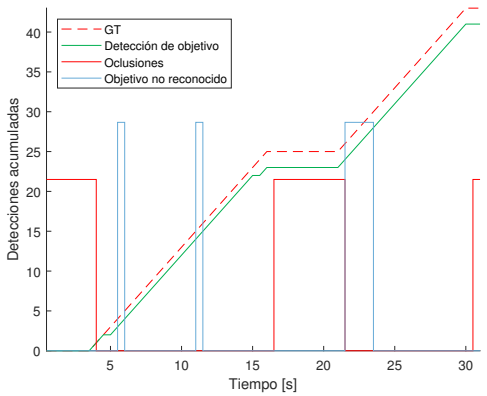
Figura B.1: f-score obtenido cuando las características son añadidas incrementalmente para discriminar al objetivo, ordenadas de acuerdo a la mínima información mutua entre la característica actual y la anterior. Figura obtenida de [3].

	H_1	L_1	S_1	L_2	A	B	Y	Cr	Cb	H_2	S_2	V	Red	Green	Blue	Grey	SLBP	LBP	Canny	HOG	Edge	Der_x	Der_y	Census	CSLBP	Der^2x	Der^2y
H_1	**5.52**	0.14	0.31	0.17	1.17	1.25	0.16	1.26	1.25	5.29	0.30	0.14	0.16	0.19	0.16	0.16	0.03	0.03	0.02	0.01	0.02	0.03	0.03	0.02	0.00	0.02	0.02
L_1	0.14	**5.80**	0.20	3.00	0.13	0.13	3.15	0.10	0.12	0.14	0.34	2.68	1.95	2.55	2.04	2.72	0.07	0.05	0.02	0.03	0.05	0.04	0.04	0.05	0.02	0.03	0.03
S_1	0.31	0.20	**5.10**	0.18	0.38	0.78	0.18	0.48	0.74	0.31	2.80	0.24	0.35	0.19	0.20	0.16	0.02	0.01	0.00	0.01	0.02	0.02	0.01	0.01	0.00	0.01	0.01
L_2	0.17	3.00	0.18	**5.87**	0.13	0.14	4.52	0.10	0.14	0.17	0.25	2.9	1.57	3.75	2.20	3.41	0.07	0.05	0.02	0.03	0.04	0.04	0.05	0.05	0.02	0.02	0.04
A	1.17	0.13	0.38	0.13	**2.95**	0.30	0.12	0.66	0.26	1.17	0.44	0.11	0.14	0.14	0.10	0.13	0.02	0.02	0.02	0.01	0.01	0.02	0.02	0.02	0.00	0.02	0.02
B	1.25	0.13	0.78	0.14	0.30	**3.67**	0.12	0.45	2.56	1.25	0.73	0.15	0.15	0.09	0.12	0.12	0.02	0.02	0.02	0.01	0.01	0.03	0.02	0.02	0.00	0.02	0.01
Y	0.16	3.15	0.18	4.52	0.12	0.12	**5.82**	0.10	0.13	0.16	0.27	2.86	1.61	3.38	2.30	3.60	0.07	0.05	0.02	0.03	0.05	0.04	0.05	0.05	0.02	0.03	0.04
Cr	1.26	0.10	0.48	0.10	0.66	0.45	0.10	**3.19**	0.49	1.26	0.48	0.12	0.11	0.09	0.12	0.10	0.01	0.01	0.00	0.00	0.01	0.01	0.00	0.01	0.00	0.00	0.00
Cb	1.25	0.12	0.74	0.14	0.26	2.56	0.13	0.49	**3.57**	1.26	0.63	0.16	0.15	0.10	0.12	0.13	0.03	0.02	0.02	0.01	0.01	0.03	0.03	0.02	0.00	0.02	0.01
H_2	5.29	0.14	0.31	0.17	1.17	1.25	0.16	1.26	1.26	**5.52**	0.30	0.14	0.16	0.18	0.16	0.16	0.03	0.03	0.02	0.01	0.02	0.03	0.03	0.02	0.00	0.02	0.02
S_2	0.30	0.34	2.80	0.25	0.44	0.73	0.27	0.48	0.63	0.30	**5.41**	0.20	0.51	0.26	0.21	0.26	0.02	0.02	0.00	0.01	0.02	0.02	0.01	0.01	0.00	0.01	0.00
V	0.14	2.68	0.24	2.9	0.11	0.15	2.86	0.12	0.16	0.14	0.20	**5.84**	2.20	2.77	3.49	2.44	0.06	0.05	0.02	0.02	0.04	0.03	0.04	0.04	0.01	0.02	0.03
Red	0.16	1.95	0.35	1.57	0.14	0.15	1.61	0.11	0.15	0.16	0.51	2.20	**5.83**	1.58	1.11	1.52	0.05	0.04	0.01	0.02	0.04	0.03	0.03	0.04	0.01	0.02	0.02
Green	0.19	2.55	0.19	3.75	0.14	0.09	3.38	0.09	0.10	0.18	0.26	2.77	1.58	**5.83**	1.87	2.86	0.07	0.05	0.02	0.03	0.04	0.04	0.05	0.05	0.02	0.03	0.04
Blue	0.16	2.04	0.20	2.20	0.10	0.12	2.30	0.12	0.12	0.16	0.21	3.49	1.11	1.87	**5.84**	2.09	0.06	0.05	0.02	0.02	0.04	0.03	0.04	0.04	0.01	0.02	0.03
Grey	0.16	2.72	0.16	3.41	0.13	0.12	3.60	0.10	0.13	0.16	0.26	2.44	1.52	2.86	2.09	**5.81**	0.07	0.06	0.03	0.03	0.05	0.05	0.06	0.05	0.02	0.03	0.04
SLBP	0.03	0.07	0.02	0.07	0.02	0.02	0.07	0.01	0.03	0.03	0.02	0.06	0.05	0.07	0.06	0.07	**5.1**	3.46	0.10	0.91	0.44	0.35	0.35	3.03	1.20	0.10	0.10
LBP	0.03	0.05	0.01	0.05	0.02	0.02	0.05	0.01	0.02	0.03	0.02	0.05	0.04	0.05	0.05	0.06	3.46	**5.66**	0.08	1.64	1.07	0.52	0.50	5.15	2.05	0.17	0.17
Canny	0.02	0.02	0.00	0.02	0.02	0.02	0.02	0.00	0.02	0.02	0.00	0.02	0.01	0.02	0.02	0.03	0.10	0.08	**3.04**	0.05	0.02	0.28	0.26	0.07	0.01	0.24	0.22
HOG	0.01	0.03	0.01	0.03	0.01	0.01	0.03	0.00	0.01	0.01	0.01	0.02	0.02	0.03	0.02	0.03	0.91	1.64	0.05	**5.29**	0.78	0.51	0.59	1.53	1.60	0.13	0.16
MPEG	0.02	0.05	0.02	0.04	0.01	0.01	0.05	0.01	0.01	0.02	0.02	0.04	0.04	0.04	0.04	0.05	0.44	1.07	0.02	0.78	**2.28**	0.27	0.26	0.96	0.83	0.07	0.06
Der_x	0.03	0.04	0.02	0.04	0.02	0.03	0.04	0.01	0.03	0.03	0.02	0.03	0.03	0.04	0.03	0.05	0.35	0.52	0.28	0.51	0.27	**3.98**	0.14	0.50	0.31	1.24	0.09
Der_y	0.03	0.04	0.01	0.05	0.02	0.02	0.05	0.00	0.03	0.03	0.01	0.04	0.03	0.05	0.04	0.06	0.35	0.50	0.26	0.59	0.26	0.14	**4.06**	0.47	0.26	0.09	1.23
Census	0.02	0.05	0.01	0.05	0.02	0.02	0.05	0.01	0.02	0.02	0.01	0.04	0.04	0.05	0.04	0.06	3.03	5.15	0.07	1.53	0.96	0.50	0.47	**5.15**	1.90	0.16	0.15
CSLBP	0.00	0.02	0.00	0.02	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.01	0.01	0.02	0.01	0.02	1.20	2.05	0.01	1.60	0.83	0.31	0.26	1.90	**3.11**	0.08	0.06
Der^2x	0.02	0.03	0.01	0.02	0.02	0.02	0.03	0.00	0.02	0.02	0.01	0.02	0.02	0.03	0.02	0.03	0.10	0.17	0.24	0.13	0.07	1.24	0.09	0.16	0.08	**2.97**	0.10
Der^2y	0.02	0.03	0.01	0.04	0.02	0.01	0.04	0.00	0.01	0.02	0.00	0.03	0.02	0.04	0.03	0.04	0.10	0.17	0.22	0.16	0.06	0.09	1.23	0.15	0.06	0.10	**3.05**

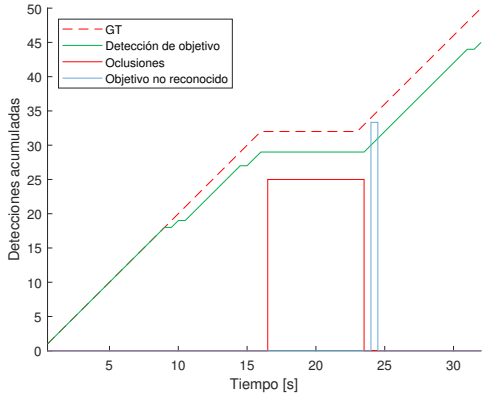
Figura B.2: Información mutua entre las 27 características analizadas para discriminar al objetivo. Figura obtenida de [3].

Anexo C

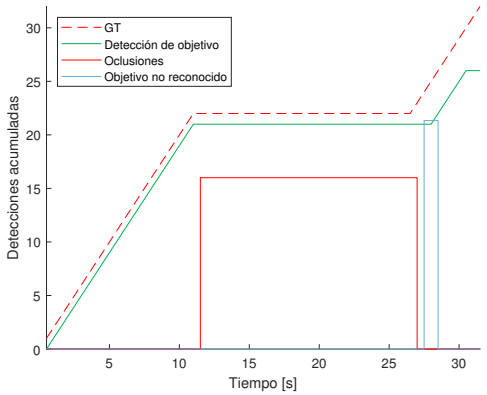
Resultados del seguimiento del objetivo



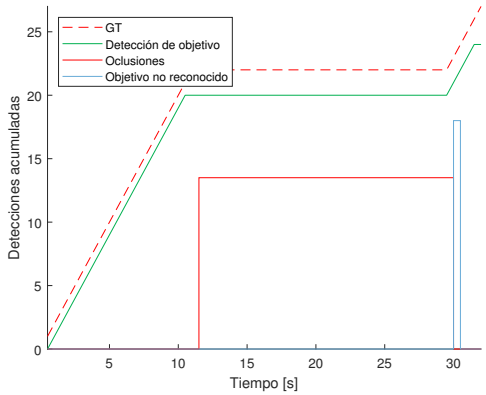
VA01



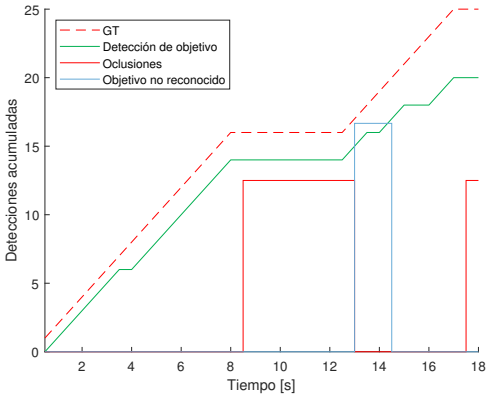
VA02



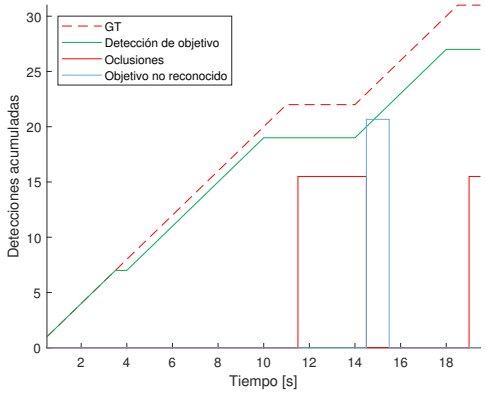
VA03



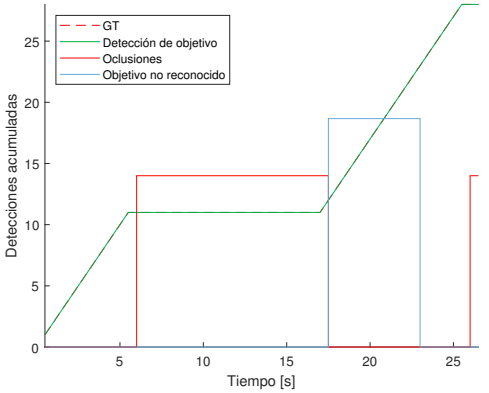
VA04



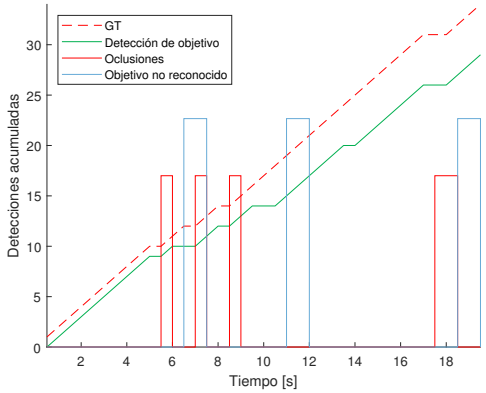
VA05



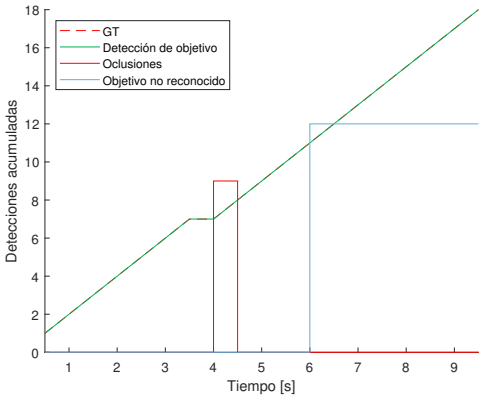
VA06



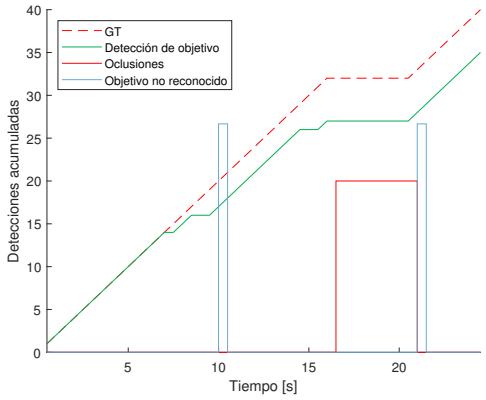
VA07



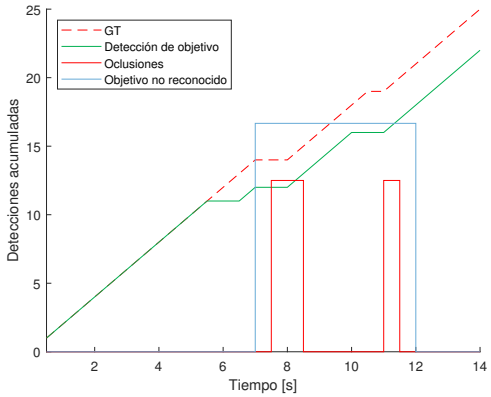
VA08



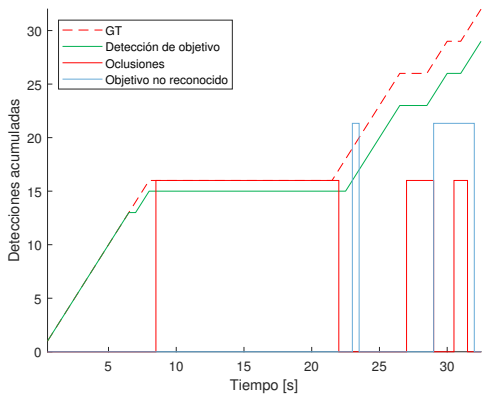
VA09



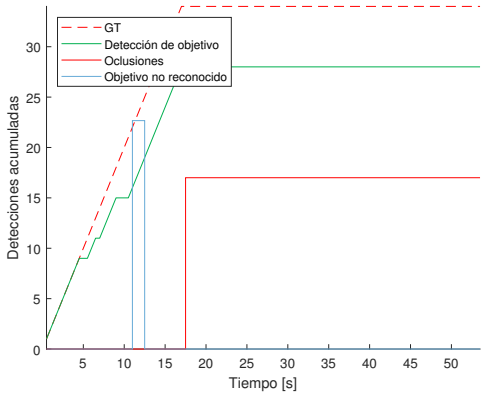
VA10



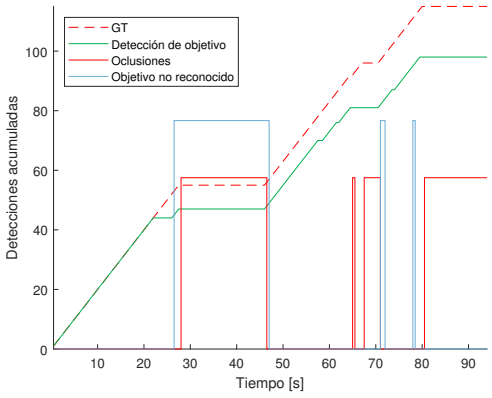
VA11



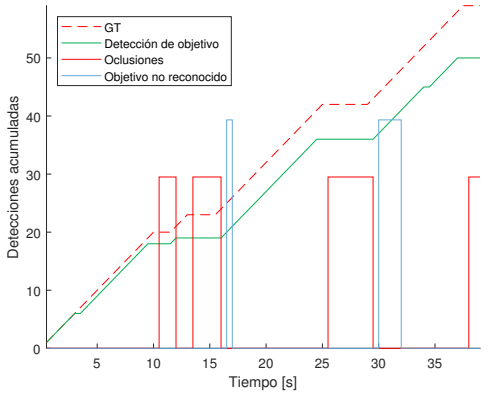
VA12



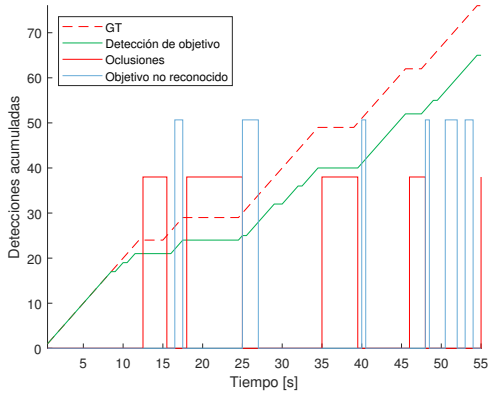
VA13



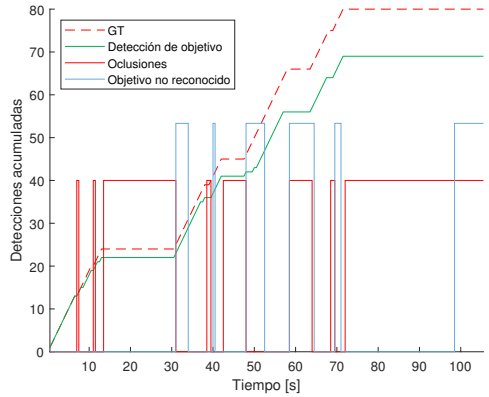
VA14



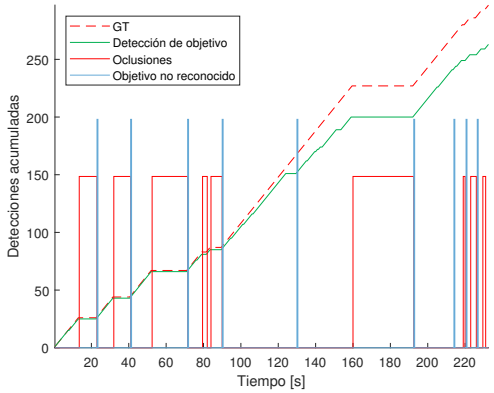
VA15



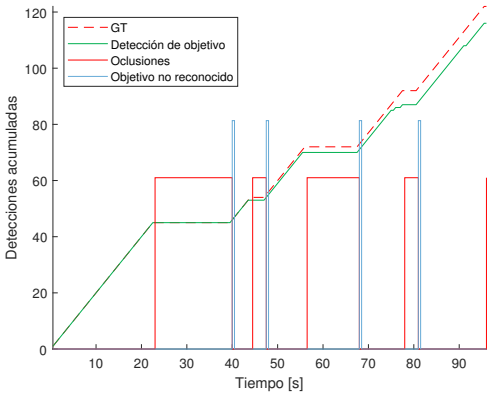
VA16



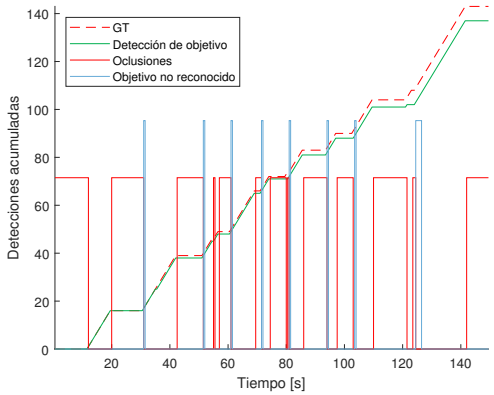
VA17



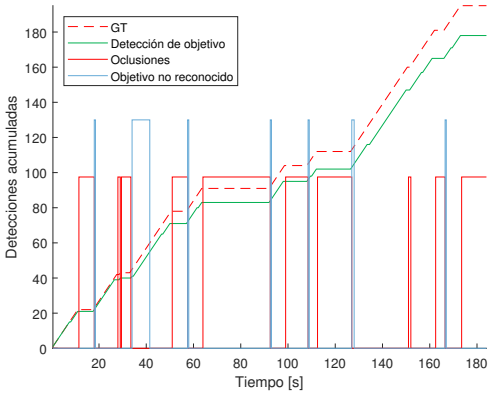
VA18



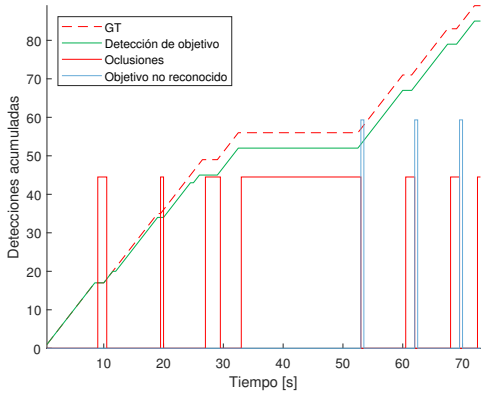
VA19



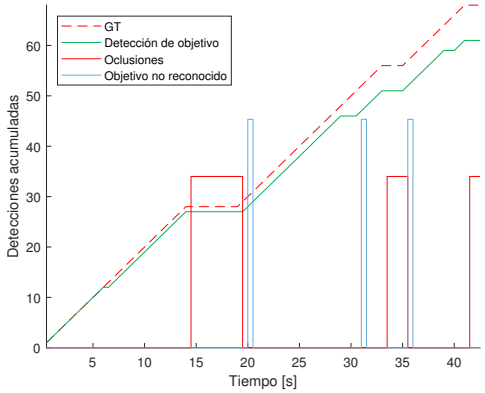
VA20



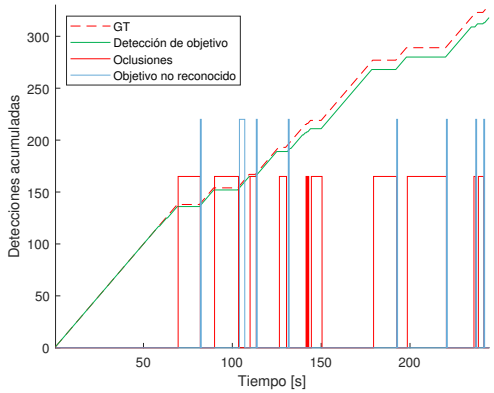
VA21



VA22



VA23



VA24

Detecciones acumuladas del objetivo, presencia de oclusiones, e instantes en que no se reconoce al objetivo.