Universidad de la República
Facultad de Ingeniería

Estimación de ocupación en el medio de transporte público incorporando tecnologías de IoT

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

Gastón García, Facundo Lezama, Juan Llaguno

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero Electricista.

Tutores

Juan Pablo González ..... Universidad de la República
Juan Pablo Garella ..... Universidad de la República

Tribunal

Juan Pablo González ..... Universidad de la República
Claudina Rattaro ..... Universidad de la República
Javier Schandy ..... Universidad de la República

Montevideo
miércoles 16 enero, 2019

Estimación de ocupación en el medio de transporte público incorporando tecnologías de IoT, Gastón García, Facundo Lezama, Juan Llaguno.

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 174 páginas.
Compilada el miércoles 16 enero, 2019.
http://iie.fing.edu.uy/

Education is not the learning of facts, it's rather the training of the mind to think.

Albert Einstein

Esta página ha sido intencionalmente dejada en blanco.

Agradecimientos

En el desarrollo de este proyecto se tuvo la colaboración de varias personas y organismos a las cuales queremos expresarles nuestro agradecimiento.

Queremos agradecer a la empresa CIEMSA por ofrecernos la oportunidad de trabajar en un proyecto tan interesante, por brindarnos un ambiente de trabajo y por la buena disposición de todos sus funcionarios involucrados.

Al Centro de Innovación en Ingeniería, a la Fundación Julio Ricaldoni y a la Agencia de Nacional de Investigación e Innovación por apoyarnos en la realización de este proyecto, impulsándonos a llevarlo más allá de una idea. Sin su apoyo económico no hubiese sido posible desarrollar los prototipos y realizar las pruebas necesarias para que este proyecto fuera más productivo.

A la empresa #Numeral por brindarnos información necesaria para el correcto análisis de WiFi.

A la empresa Chevel S.R.L. por poner a disposición sus unidades de transporte para hacer las pruebas de campo.

A los profesores del Instituto de Ingeniería Eléctrica y a los tutores por brindarnos su apoyo y conocimiento para que este proyecto salga adelante.

A todos los compañeros que nos acompañaron durante toda la carrera en largas jornadas de estudio.

A nuestras familias y amigos, por su constante apoyo a lo largo de toda la carrera, ayudando desde lo más simple hasta colaborando activamente en este proyecto. También por entender nuestra ausencia en más de una oportunidad por la dedicación prestada a esta carrera tan linda pero a la vez tan sacrificada.

Esta página ha sido intencionalmente dejada en blanco.

A nuestras familias, amigos y todos los que nos acompañaron en este camino.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

Optimizar el servicio de transporte público es necesario para brindar un mejor servicio a los usuarios. Para esto se necesitan índices de ocupación con información geográfica y de tiempo. En este proyecto se busca comenzar con un sistema de estimación de ocupación para ómnibus que permita brindar tal información en tiempo real.

En este trabajo se realiza una previa investigación de los estudios ya realizados que buscan en parte resolver el problema de contabilizar personas en un espacio físico. Posteriormente se hace una selección de las posibles tecnologías y métodos capaces de detectar personas o dispositivos móviles. Se seleccionan tecnologías y métodos tales como sensores de ultrasonido, procesamiento de imágenes obtenidas a partir de una cámara y detección de dispositivos móviles mediante la captura de paquetes de WiFi. Utilizando un ómnibus con pasajeros se genera una base de datos con el fin de validar las tecnologías seleccionadas y los distintos métodos asociados.

Se realiza la estimación de la ocupación en tres niveles: vacío, medio y lleno para todas la tecnologías, obteniéndose resultados de hasta el 84 % de precisión, así como también la estimación de ocupación por minuto y luego de cada parada.

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Capítulo 1

Introducción

1.1. Definición del problema

Para poder evaluar la eficiencia del transporte público en términos de flujo de personas y recorrido de las líneas es necesario saber la ocupación de los ómnibus en cada momento. Hoy en día en Montevideo no se cuenta con una herramienta que brinde esta información. Es por esto que resulta de gran interés lograr una estimación de tal ocupación ya que podría permitir entre otras cosas: distribuir eficientemente las líneas de ómnibus en la ciudad, disminuir el tiempo de viaje de las personas, reforzar recorridos y hasta brindar al usuario información de la ocupación en tiempo real.

Existen distintas tecnologías y productos que pueden brindar la información necesaria pero éstas tienen un precio elevado y esto hace que se opte por no utilizarlos.

1.2. Descripción del proyecto

1.2.1. Objetivo

El objetivo de este proyecto es realizar un estudio del estado del arte de las tecnologías existentes para el conteo de personas en medios de transporte público. Profundizar y validar al menos tres tecnologías orientadas a conteo de personas y dispositivos con el fin de realizar una estimación en al menos tres niveles: vacío, medio lleno y lleno.

1.2.2. Alcance

El alcance del proyecto abarca los siguientes aspectos:

- Conocer el estado del arte del área de estimación de ocupación de medios de transporte público mediante tecnologías IoT (Internet of Things).

- Seleccionar tres tecnologías de las cuales al menos una cuente personas (no dispositivos) y entender su funcionamiento en profundidad.
- Implementar prototipos para las tecnologías seleccionadas y validarlos.
- Generar una base de datos para la validación de las tecnologías.
- Realizar un estimador de ocupación de tres niveles para los prototipos implementados.

1.3. Contexto del proyecto

El crecimiento del tránsito en las grandes ciudades del mundo cada vez requiere de que éste sea más eficiente. En Montevideo una parte importante del tránsito urbano es el transporte público. Este proyecto se enmarca en la necesidad de la Intendencia de Montevideo de obtener información de la ocupación en el transporte público en su plataforma de ciudades inteligentes Fiware[1]. Junto a la empresa CIEMSA y con el apoyo de la Fundación Julio Ricaldoni y el Centro de Innovación en Ingeniería se busca estudiar las distintas tecnologías que permitan realizar un conteo de personas en tiempo real, como un comienzo en el desarrollo de un sistema capaz de brindar tal información en todo momento.

1.4. Principales contribuciones del proyecto

Este proyecto evalúa y selecciona distintas tecnologías capaces de detectar personas con el fin de iniciar el camino para el desarrollo de un sistema que integre dichas tecnologías y permita estimar la ocupación en tiempo real. Se contribuye en el análisis de algunas tecnologías en particular, utilizadas en el conteo de personas y dispositivos.

Con el fin de validar las tecnologías en necesario tener una base de datos sobre la cual probarlas y evaluar su desempeño. Debido a que no se tiene acceso a una base de ese estilo, en este proyecto se genera una base de datos a partir de pruebas de campo realizadas con público voluntario. Esta información además resulta una contribución para futuros trabajos en el área.

1.5. Organización del documento

A continuación se realiza un breve resumen del contenido de los capítulos que siguen.

[1] `https://www.fiware.org/`

Capítulo 2 - Estado del arte y selección de tecnologías

Para conocer las distintas tecnologías y métodos utilizados para el conteo de personas se realizó un estado del arte y luego se hizo una selección de las más viable.

Capítulo 3 - Generación de base de datos

Con el objetivo de validar las tecnologías seleccionadas se realiza una base datos. En este capítulo se muestra el proceso de generación y sus resultados.

Capítulo 4 - Implementación de algoritmos y experimentos

En este capítulo se explican los distintos métodos utilizados y se muestran los resultados obtenidos a partir de la base de datos.

Capítulo 5 - Análisis de resultados

Se hace un análisis profundo de los resultados y se compara el desempeño de las tecnologías seleccionadas.

Capítulo 6 - Conclusiones y trabajos a futuro

Por último se presentan las conclusiones realizadas sobre este trabajo y también los posibles trabajos que se pueden realizar a partir de lo estudiado.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 2

Estado del arte y selección de tecnologías

2.1. Estado del arte

El problema de mejorar la eficiencia del transporte público existe desde hace varias décadas con el crecimiento de las grandes ciudades en el mundo. Se pueden encontrar artículos que datan de la década de los 70 [1], donde por ejemplo en uno de ellos se muestra el trabajo realizado para crear un dispositivo capaz de contar las personas al momento de ascender y descender de los ómnibus. Existen estudios que combinan tanto medidas subjetivas mediante encuestas a los usuarios donde se les pregunta: conformidad del servicio, comodidad, precio entre otros; como medidas objetivas: ocupación, frecuencia de la línea, distancia entre paradas, etc [2].

Por ejemplo en los ómnibus, saber la ocupación en diferentes momentos del día es un parámetro importante para asignar la cantidad adecuada de unidades en cada línea de transporte. Por otro lado, la distribución correcta de los ómnibus disponibles para las diferentes líneas es fundamental para obtener una optimización de toda la red de transporte público y para poder reducir los costos.

A su vez con el avance de los teléfonos inteligentes (en inglés smartphones), los planificadores de rutas y los servicios en tiempo real se han convertido en componentes esenciales del transporte como servicio a los usuarios [3].

Por todo lo descrito anteriormente se ve la importancia de poder conocer la ocupación dentro del transporte público en tiempo real. En el caso de los ómnibus urbanos, este factor es de gran importancia al momento de evaluar si agregar o quitar una línea de servicio, lo mismo con las paradas. También brindar la información de ocupación en tiempo real a los usuarios es importante para aplicaciones del estilo de planificación de rutas, o simplemente para que el usuario decida esperar un ómnibus que arribará lleno o no.

En las siguientes dos secciones se hace un estudio del estado del arte de distintos métodos y tecnologías capaces de contabilizar personas y dispositivos móviles.

2.1.1. Conteo de personas

En la siguiente sección se explican algunos métodos que utilizan tecnologías capaces de detectar la presencia física de las personas. Estos métodos en su gran mayoría fueron pensados para poder contabilizar personas en un espacio determinado.

2.1.1.1. PIR

El sensor PIR (Passive InfraRed) es un sensor eléctrico capaz de medir la radiación electromagnética emitida por los objetos en la banda del infrarrojo. El término Passive (Pasivo en español), hace referencia a que el sensor no irradia para poder hacer la medida, sino que solamente recibe la radiación infrarroja de su entorno.

Generalmente estos sensores son utilizados en sistemas de alarmas para detectar la presencia de una fuente de calor como una persona o cualquier otro animal.

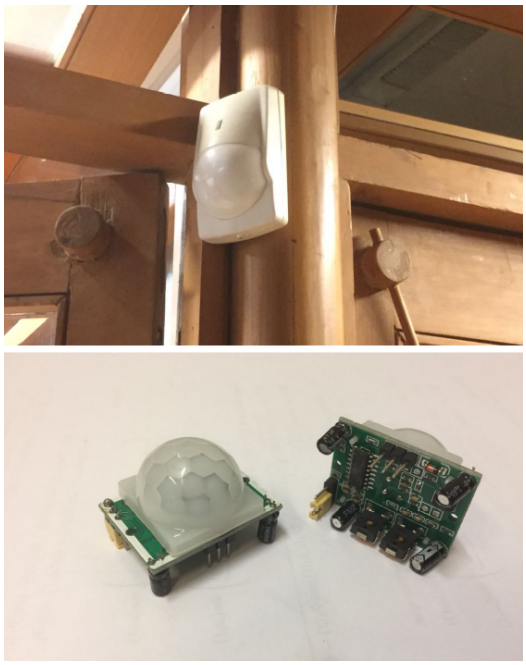


Figura 2.1: Sensor PIR.

Observando la figura 2.1, se puede ver en la parte superior una de las presentaciones del sensor bastante familiar para el común de las personas ya que éstos

se pueden ver a menudo en casas, oficinas y comercios. Éstos son producidos en masa y tienen un muy bajo costo; en contraste con otros sensores consumen muy poca potencia y requieren de muy poco mantenimiento. En la parte inferior de la figura se puede apreciar como es el sensor sin su protección habitual.

En el trabajo descrito en [4], estos sensores son utilizados para contabilizar la cantidad de personas dentro de oficinas con la idea de obtener información sobre el uso del espacio en éstas y así tener un control eficiente del consumo de energía eléctrica. El método utilizado para medir la ocupación de la oficina es detectar cuándo una persona entra o sale de ésta. Para ello se utiliza un par de sensores PIR por cada puerta de las oficinas de tal forma que uno quede del lado de adentro y el otro del lado de afuera. Así las personas que pasan a través de la puerta entrarían secuencialmente en los campos de visión de ambos PIR. La idea de utilizar dos sensores es poder diferenciar si la persona está saliendo o entrando a la oficina. Si el sensor de adentro es el primero en activarse se puede deducir que la persona está saliendo, y análogamente se puede deducir que está entrando, de esta forma es posible diferenciar la dirección de movimiento.

Las características de un sensor como el que se muestra en la parte inferior de la figura 2.1, son las siguientes:

Rango de detección	$3m$ a $7m$
Ángulo de detección	$100°$
Salida	$3{,}3V$ (Alta)
Tiempo de estado activo	configurable
Consumo en reposo	$< 50\mu A$
Voltaje de alimentación	$4{,}5V$ a $20V$ (en DC)

Tabla 2.1: Características del sensor PIR.

2.1.1.2. Radar IR-UWB

Un sensor de radar IR-UWB (Impulse Radio - Ultra WideBand) es un sensor capaz de enviar impulsos a una alta velocidad, ocupando un ancho de banda por encima de los 500MHz, para luego reconocer varias situaciones mediante el procesamiento de múltiples señales que se reciben después de ser reflejadas por múltiples humanos y/u objetos. Estos tipos de sensores tienen diferentes usos, desde detectar actividad humana por cuestiones de seguridad hasta detectar cáncer de mama [5].

En el artículo [5] se propone un sistema para contar el número de personas que pasan a través de una capa virtual usando señales de RF (Radio Frecuencia) basadas en sensores de radar IR-UWB. La configuración del sistema básicamente es con dos sensores de radar IR-UWB equipados con antenas que tienen ancho de haz estrecho en la dirección horizontal para contar el número de personas que pasan. La dirección de propagación del radar es perpendicular a la dirección de movimiento

del humano, que forma una especie de dos capas delgadas invisibles para contar simultáneamente una cantidad de transeúntes. La representación conceptual básica se muestra en la figura 2.2.

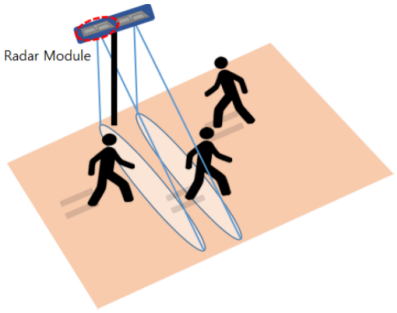


Figura 2.2: Representación conceptual del sistema de conteo. [5]

La razón de usar dos sensores de radar, al igual que el sistema que utiliza sensores PIR es reconocer la dirección del humano en movimiento. Es decir, para poder hacer un conteo en ambas direcciones.

El sistema de conteo de personas propuesto tiene varias ventajas debido a utilizar señales RF, como la solidez de la apariencia humana, la ausencia de problemas de privacidad y la capacidad para operar en un entorno con humo o sin luz. Al mismo tiempo, las buenas características de resolución de rango del sensor de radar IR-UWB proporcionan capacidad de detección para múltiples personas que pasan a diferentes distancias al mismo tiempo.

2.1.1.3. Detector de obstáculos IR

Un detector de obstáculos infrarrojo (IR) es un dispositivo que detecta la presencia de un objeto emitiendo luz infrarroja y detectando la reflexión en dicho objeto. El uso de luz infrarroja es simplemente para que ésta no sea visible para los humanos.

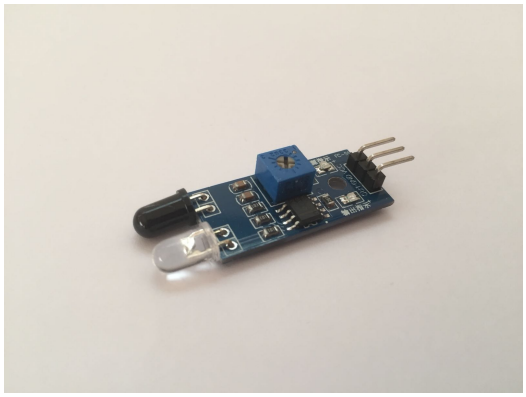


Figura 2.3: Sensor Infrarojo de Arduino.

El sensor IR que se muestra en la figura 2.3 es particularmente un dispositivo ofrecido por la marca Arduino[1], aunque conociendo sus características y funcionamiento se puede incorporar a otras tecnologías. El mismo cuenta con un perilla que permite regular la distancia de detección; una vez que el sensor tiene enfrente un objeto a una distancia menor igual a ésta, el sensor emite una señal eléctrica. Este sensor generalmente es utilizado en robots para medir la proximidad a objetos.

El sensor cuenta con las siguientes características:

Rango de detección	entre $2cm$ y $50cm$
Ángulo de detección	$35°$
Alimentación	de $3{,}3V$ a $5V$ (DC)

Tabla 2.2: Características del sensor IR.

2.1.1.4. Sensor de ultrasonido para medir distancia

Este sensor básicamente funciona de la misma forma que el anterior, emitiendo una señal y esperando el reflejo contra un objeto, para no ser detectado por el oído humano trabaja en el rango de frecuencia de ultrasonido. La gran diferencia es que éste permite medir la distancia a la que se encuentra el objeto.

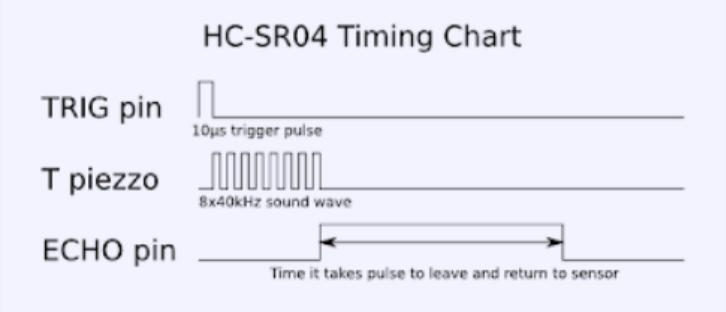


Figura 2.4: Secuencia de pulsos del sensor de ultrasonido.[2]

Para iniciar una medida el sensor necesita un pulso de largo mínimo $10\mu s$ en el pin de entrada TRIGGER, luego éste envía 8 pulsos de $40kHz$ y coloca su salida ECHO en alto. Estos pulsos se reflejan en un objeto deflector del sonido y cuando son recibidos por el sensor éste pone la salida ECHO en bajo. El tiempo en que se mantuvo la salida ECHO en alto es equivalente al tiempo que le llevó al pulso hacer el recorrido de ida y vuelta. Éste funcionamiento se puede observar de forma gráfica en la figura 2.4.

[1] `https://www.arduino.cc/`

[2] Imagen extraída de: `http://bkargado.blogspot.com/2013/09/todosobrehc-sr04.html`

Entonces, dado que la velocidad del sonido es un parámetro conocido, $v_s = 343{,}2m/s$, y se tiene el tiempo de recorrida de los pulsos (t_r), despejar la distancia al objeto (D) se hace fácilmente de la siguiente forma:

$$D = v_s \cdot \frac{t_r}{2} \tag{2.1}$$

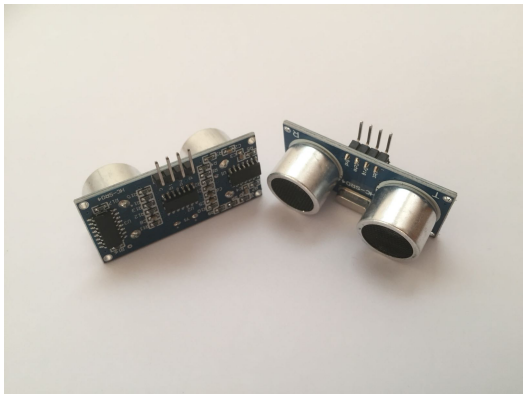


Figura 2.5: Sensor Ultrasonido de Arduino.

El funcionamiento descrito anteriormente corresponde particularmente al sensor que se muestra en la figura 2.5. Este sensor de ultrasonido es ofrecido por la marca Arduino. El mismo tiene las siguientes características:

Corriente de reposo	$< 2mA$
Corriente de trabajo	$15mA$
Ángulo máximo de medición	$30°$
Rango de medición efectiva	$2cm$ a $400cm$
Resolución	$3mm$
Frecuencia de trabajo	$40kHz$

Tabla 2.3: Características del sensor de ultrasonido.

Las dos últimas tecnologías, tanto el detector de obstáculos IR como el sensor de proximidad de ultrasonido, no fueron encontrados en ninguno de los artículos revisados, éstos simplemente fueron evaluados como variantes de sensores.

2.1.1.5. Cámara

Una de las tecnologías más utilizadas para el conteo personas es la utilización de una cámara y un software específico. En principio el desarrollo de dicho software perecería ser una tarea bastante ardua, para la cual solo estaría capacitado para hacerla gente asociada al desarrollo de software, sin embargo últimamente con el avance y la accesibilidad de Python, dicha tarea se ha simplificado notoriamente.

Esto se debe a la gran comunidad activa que desarrolla y comparte librerías para su utilización, dentro de las cuales existen funciones ya definidas que facilitan la programación para gente familiarizada en el área pero no experta.

Para el desafío propuesto existen varias técnicas en el procesamiento de imágenes que permiten realizar esta tarea [6] [7] [8]. En este trabajo se abordaron las siguientes técnicas: Background Subtraction, Face Detection y Face Recognition y por último Object Detection y Tracking.

Los métodos que se estudiaron constan principalmente de 3 etapas: detección, tracking y conteo. La detección es básicamente decir si en el cuadro del video (frame en inglés) a procesar existe el elemento que se está buscando, en este caso personas, y también decir en qué parte de la imagen se realizó la detección. Esto último generalmente se visualiza con un rectángulo dentro del cual se encuentra el elemento detectado. Una vez realizada la detección, el siguiente paso es el de seguimiento (en inglés tracking), el cual parece ser similar al anterior pero es considerablemente distinto, ya que lo que se tiene que realizar es encontrar el objeto detectado en los siguientes cuadros de video. Dicho proceso parece ser elemental pero puede resultar dificultoso, dependiendo de las condiciones del video. Por ejemplo, la mayoría de los algoritmos de tracking no son robustos a la oclusión, la cual consiste en que el objeto detectado se ve parcial o completamente tapado por otro objeto o algún otro elemento del video, y por lo tanto el algoritmo "se pierde", lo que quiere decir que no se sabe en qué parte del frame se encuentra el objeto detectado. La última etapa es la de conteo, la cual consiste en fijar un umbral a partir del cual uno decide que el objeto entró o salió del recinto en el que se hallaba. Hay varias formas de decidir cuando contar, las más comunes son: cuento si el objeto detectado supera un límite que se propuso o si el objeto sale fuera del frame. Éstas 3 etapas mencionadas, no necesariamente tienen que ser las únicas, sino que pueden existir instancias adicionales, sin embargo se considera que son las principales para realizar un contador mediante el procesamiento de video, para situaciones donde la gente entra y sale de un recinto. A continuación se detallan algunas de las técnicas utilizadas para la detección.

Detección

Background Subtraction

El método de detectar personas o vehículos mediante background subtraction es bastante conocido y útil. Tiene varias aplicaciones en muchas áreas relacionadas con la visión por computadora [8] [9]. Consiste en poder separar el fondo (Background), generalmente lo que no varía de posición en un video, de lo que sí tiene movimiento (Foreground). Esta técnica es ampliamente utilizada, por ejemplo: en una cámara de tráfico para poder detectar los vehículos en movimiento, en un local para poder contar la cantidad de gente que entra y que sale mediante los videos de cámaras de seguridad. En la mayoría de las aplicaciones donde se utiliza esta tecnología, tienen en común que están en lugares sumamente controlados, dado

que la posición de la cámara es fija y la iluminación no varía drásticamente de un momento a otro. Todo esto permite que se desarrolle un algoritmo que pueda aprender a diferenciar lo que es fondo de lo que puede estar en movimiento.

Dentro de OpenCV, que es una de las principales librerías de Python que se utilizó en este proyecto, hay 3 funciones distintas que implementan Background Subtraction: *BackgroundSubtractorMOG*, *BackgroundSubtractorMOG2* y *BackgroundSubtractorGMG*.

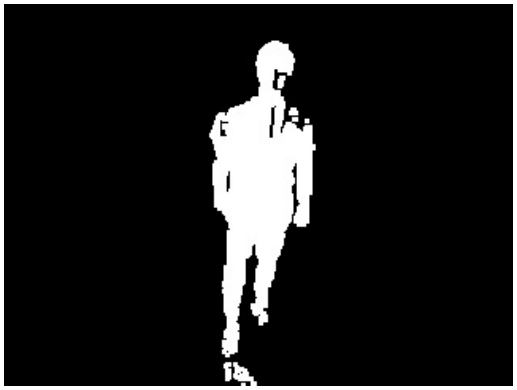
BackgroundSubtractorMOG consiste en un algoritmo de segmentación de Background/Foreground basado en un modelo mixto Gaussiano el cual fue introducido en el siguiente artículo [10]. El método que implementa el algoritmo consiste en modelar cada pixel como una mezcla de 3 a 5 distribuciones gaussianas. El peso de la mezcla representa la cantidad de tiempo que un color permanece en la escena. Por lo tanto los colores que permanecen más tiempo en la escena y no cambian son los más probables de pertenecer al fondo.

BackgroundSubtractorMOG2 es similar a *BackgroundSubtractorMOG* dado que también consiste en un algoritmo de segmentación de Background/Foreground basado en un modelo mixto Gaussiano. Una de las diferencias principales que tienen es que *BackgroundSubtractorMOG2* elige un número apropiado de distribuciones gaussianas para cada pixel, con lo cual obtiene mayor adaptabilidad a los cambios de iluminación. La base del algoritmo fue desarrollada en los siguientes artículos [11] y [12].

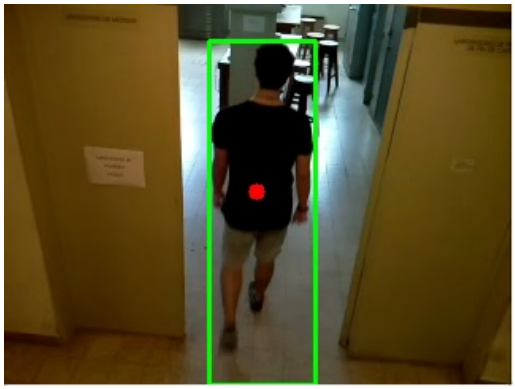
BackgroundSubtractorGMG combina una estimación estadística del Background en conjunto con una segmentacion Bayesiana por pixel. Las bases de dicho algoritmo se realizaron en el siguiente artículo [13].

El resultado que devuelve la función *BackgroundSubtractorMOG2* es una imagen en blanco, negro y gris, donde lo negro representa lo que se detectó como fondo, lo blanco lo que se detectó como Foreground y lo gris serían las sombras, esto se puede ver en la figura 2.6(a). Para poder trabajar adecuadamente con esta información lo que se tiene que hacer es aplicar una función morfológica que lo que hace es delimitar mejor las formas detectadas, de manera que a los pixeles blancos aislados los ignore y solo se quede con formas blancas de un área considerable. Esto se realiza primero dilatando la imagen, de forma de disminuir el ruido y luego erosionándola, de forma que combine las regiones blancas pequeñas, que probablemente provengan de la misma persona. El resultado devuelto por dicha función se puede observar en la figura 2.6(b). Se considera que todas las manchas blancas que aparezcan, que sean mayor a una cierta área de referencia, la cual se puede modificar, van a ser personas que se van a poder seguir. De esta forma es que se realiza la etapa de detección de personas, ver figura 2.6(c).

(a) Imagen luego de realizar Background Subtraction.



(b) Imagen filtrada.



(c) Detección de personas.

Figura 2.6: Detección mediante Background Subtraction.

Face Detection

A diferencia de la sección anterior, en la que se separaba lo que tenía movimiento de lo que se consideraba como fondo, con lo cual el método podría servir para contar cualquier tipo de objeto en movimiento, lo que se busca en esta sección es detectar caras en imágenes. Para realizar esta tarea se utilizaron dos tipos de clasificadores: Haar Classifier y LBP Classifier (Local Binary Patterns). Ambos tienen el mismo objetivo, detectar cuando en una imagen aparece una cara, pero ambos lo hacen por procedimientos distintos.

Haar Classifier es un clasificador que se basa en machine learning [14], el cual se entrena positivamente con muchas imágenes con caras y negativamente con muchas imágenes sin caras. De esta forma se extraen características con las cuales es posible detectar si en una imagen existe una cara con un porcentaje de acierto alto.

LBP Classifier es un descriptor de texturas popularizado por el siguiente artículo [15]. Este descriptor realiza una representación local de la textura, comparando cada pixel con sus vecinos y luego realiza un histograma. De esta forma

se pueden extraer características de texturas con lo cual se puede diferenciar caras de otros objetos en una imagen. A su vez, también es necesario entrenar al clasificador con muchas imágenes positivas (con caras) y negativas, y de esta forma poder detectar caras en una imagen.

Entre Haar Classifier y LBP existen muchas diferencias, pero en general se da que el Haar Classifier es más preciso que el LBP, pero demora más en procesar, de esta forma hay que hacer un balance de lo que se quiere. Si se quiere tener poco error y no importa el tiempo de procesamiento, entonces es recomendable utilizar Haar Classifier, sin embargo si lo que se quiere es hacer detección en tiempo real, entonces aunque no sea tan preciso es más recomendable utilizar LBP.

Ambos de estos métodos se denominan Cascade Classifier dado que los dos extraen características de una imagen a partir de las cuales clasifican si aparece un rostro en la imagen. Pero el número de características que extraen es bastante alto y verificar si contiene todas las características para todos los frames lo haría algo ineficiente. Es por esto que implementan un proceso en cascada, lo que quiere decir que en cada frame se busca un subconjunto de características primero, y si dichas características no son detectadas entonces se descarta el frame y se dice que no hay caras, sin embargo si el resultado es positivo se busca en el frame otro subconjunto de características y se repite el proceso. De esta forma solo un frame que contenga todas las características buscadas tendrá una cara.

En la figura 2.7 se puede observar el resultado al realizar detección de caras en una imagen con Haar Cascade Classifier.

Figura 2.7: Detección de caras mediante Haar Cascade Classifier.

Object Detection

A diferencia de los clasificadores en cascada como lo eran Haar Classifier y LBP Classifier, también se realizó la detección con un algoritmo que implantaba Redes Neuronales Convolucionales (CNN, por su sigla en inglés Convolutional Neural Network). Esto se realizó mediante la utilización de la librería TensorFlow[3] [16], la cual fue desarrollada por Google. El funcionamiento de las redes neuronales convolucionales son en principio similar al de los clasificadores en cascada debido a que es necesario un conjunto de imágenes para entrenarlos, sin embargo algunos parámetros que en los filtros de cascada son fijados manualmente, en las redes neuronales se realiza un proceso de aprendizaje donde se encuentran cuales son los mejores para la tarea. Es por esto que los clasificadores de imágenes con CNN son más precisos y son capaces de detectar personas incluso si están parcialmente ocluidas o rotadas. Existen arquitecturas de redes neuronales pre-entrenadas, las cuales se encuentran a disposición de la comunidad científica. Una de ellas está entrenada para detectar objetos, unos 90, de los cuales uno son personas. Por lo tanto se debería filtrar la detección de forma de obtener solamente las detecciones de personas que es el caso de interés. Un método similar fue implementado en el artículo [17].

En la figura 2.8 se puede observar la detección realizada con Object Detection.

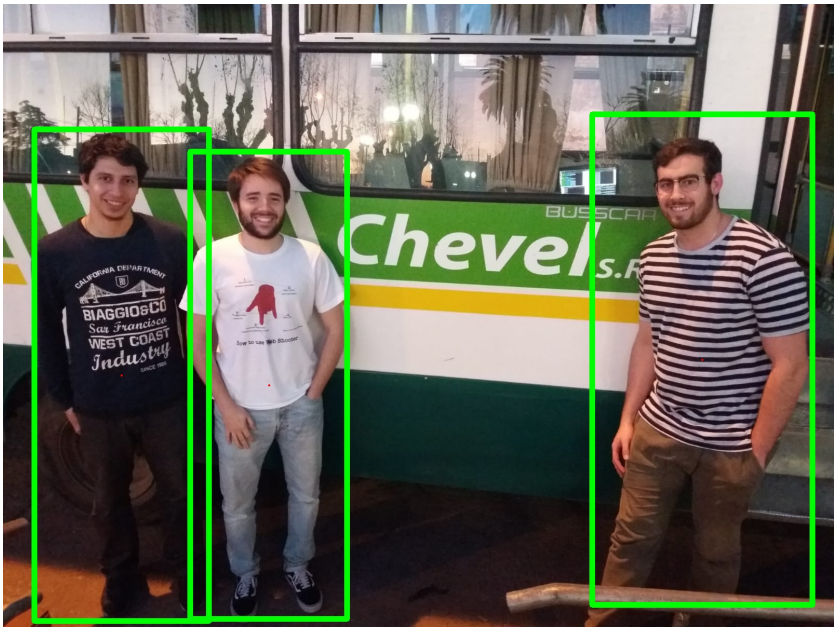


Figura 2.8: Detección de personas mediante Object Detection.

[3]Página de la librería TensorFlow: `https://www.tensorflow.org`

2.1.2. Conteo de dispositivos

El aumento del consumo de dispositivos móviles en la sociedad hace que en el año 2018 se hayan registrado mundialmente más líneas de celulares que personas [18]. Es por esto que el conteo de dispositivos para estimar la cantidad de personas que se encuentran dentro del transporte público puede ser un dato relevante y hasta concluyente.

Varios métodos son propuestos para realizar detecciones de dispositivos móviles, principalmente utilizando los protocolos de Bluetooth y WiFi. Estos protocolos tan comúnmente utilizados pueden brindar la información necesaria para permitir la detección y seguimiento de los dispositivos y así poder estimar la ocupación en el medio de transporte público.

2.1.2.1. WiFi

La tecnología comúnmente llamada WiFi, aplicada sobre Wireless Local Area Network (WLAN), está definida en el estándar IEEE 802.11. Es adoptada por los dispositivos móviles hoy en día siendo su mayor uso el de permitir el acceso a internet. Los dispositivos móviles tienen una interfaz de red para la comunicación. Ésta tiene asociada una dirección MAC única y por lo tanto la detección de esta dirección identifica al dispositivo frente al resto. Esta información está presente en los paquetes que se envían durante la comunicación de WiFi. De esta forma detectando esta información se podría distinguir y contar dispositivos dentro del ómnibus.

Existen varios estudios que presentan métodos de detección basados en WiFi. El método más recurrente es el que busca detectar las tramas que se denominan Probe Request [19] [20]. Estas tramas son enviadas por los dispositivos en busca de información respecto de las redes WiFi disponibles a su alrededor, para luego eventualmente poder conectarse como se observa en la figura 2.9. Estas tramas contienen en su encabezado Ethernet la dirección MAC asociada al dispositivo de origen.

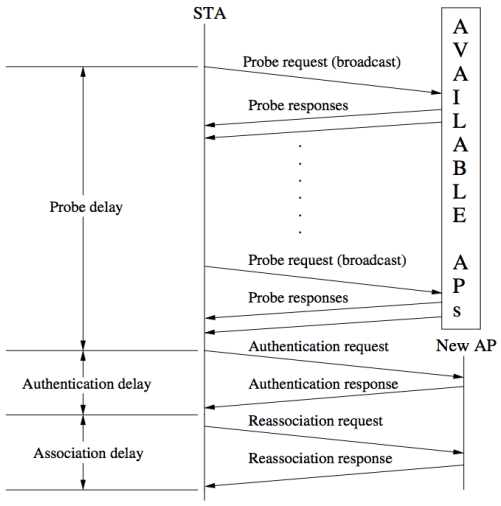


Figura 2.9: Secuencia de descubrimiento y asociación a redes WiFi [21].

Estas tramas de *Probe Request* son enviadas por los dispositivos regularmente para detectar las redes. Ya sea que el dispositivo esté siendo usado o en reposo, cada cierto tiempo envía esta trama en busca de las redes cercanas. Los tiempos de envío varían según los distintos dispositivos y no siempre son periódicos [22]. Por este motivo es que se puede capturar paquetes esperando detectar este tipo de tramas. De esta forma es de esperar que si un dispositivo se encuentra dentro del ómnibus se pueda capturar una trama *Probe Request* proveniente del mismo.

Varios métodos toman de esta trama la dirección MAC para distinguir el dispositivo y luego analizan aspectos de tiempos y niveles de señal para realizar el conteo de los mismos. Se puede en una primera instancia restringir el área de detección por el nivel de señal en recepción. Para esto se puede utilizar el indicador RSSI *(Received Signal Strength Indicator)*. RSSI es una medida del nivel de potencia que recibe la radio en recepción luego de la posible pérdida en la antena y los cables.

Trabajando con los tiempos de arribo de los paquetes detectados se puede estimar, luego de cierto tiempo, que si no se encuentra nuevamente la dirección MAC que se entendía que estaba dentro del ómnibus es porque ese dispositivo ya no se encuentra en el transporte. De esta forma se puede lograr una estimación de ascensos y descensos para luego inferir la ocupación del medio de transporte.

Justamente para evitar esta detección y seguimiento de las personas por motivos de seguridad es que ya hace algunos años varios de los fabricantes de dispositivos móviles comenzaron a realizar cambios en el envío de tramas Probe Request. Han decidido ocultar la dirección MAC de los dispositivos, enviando en su lugar una dirección MAC “aleatoria” [23]. Cambiando esta dirección regularmente se evita el seguimiento ya que se puede entender que son dispositivos distintos los que enviaron los Probe Request, dado que se detectaron MAC distintas, sin em-

bargo es el mismo dispositivo.

Por otro lado, también existen estudios sobre qué tan seguro es este procedimiento de ocultar la identidad del dispositivo cambiando la dirección MAC. Se encuentran algunas características particulares que buscan identificar y realizar un seguimiento de los dispositivos. Algunas técnicas se basan en estudiar datos básicos de los paquetes Probe Request como pueden ser el número de secuencia o el tiempo de arribo [24]. Este artículo también presenta un análisis de la frecuencia de envíos de las tramas Probe Request ya que esto puede ser un indicador relevante a la hora de analizar las direcciones MAC "aleatorias". Existe además del número de secuencia del paquete y los tiempos de arribo otro tipo de información, por ejemplo dentro del payload del Probe Request, que permite identificar los dispositivos [23]. Este análisis es mucho más profundo que el anterior y analiza características como el Wi-Fi Protected Setup (WPS) o el Service Set Identifier (SSID).

Al realizar el ocultamiento o aleatoriedad de la dirección MAC las empresas se ven en la situación de decidir qué direcciones utilizar. Una de las opciones que se encuentran en el mercado es la posibilidad de utilizar rangos de direcciones MAC locales [22]. Las direcciones MAC se dividen entre Universally Administered Addresses (UAA) o Locally Administered Addresses (LAA). Las direcciones universales son las asignadas por los fabricantes de forma única a un dispositivo. En contraposición, las direcciones MAC locales son las que no están asociadas de forma única a los dispositivos. Estos dos tipos de direcciones se diferencian simplemente con el segundo bit menos significativo del primer byte. Como se observa en la composición de la dirección MAC en la figura 2.10, si este bit es 0 entonces la dirección es universal mientras que si el bit es 1 la dirección es local.

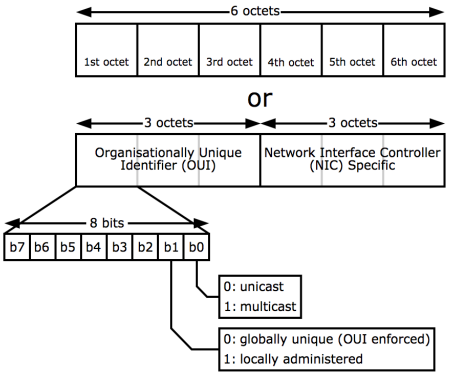


Figura 2.10: Composición de dirección MAC.[4]

Esta facilidad para distinguir entre las direcciones MAC locales o universales hace que sea también fácil encontrar los dispositivos que no realizan este tipo de

[4]Imagen extraída de: `https://en.wikipedia.org/wiki/MAC_address`.

aleatoriedad. Es un aspecto importante a la hora de realizar pruebas de detección de dispositivos ya que se pueden separar las direcciones en dos grupos y tratarlas por separado.

Uno de los aspectos necesarios para poder desarrollar todas estas ideas es poder realizar la captura de los paquetes. Para esto existen distintas herramientas tales como Wireshark, Tshark, Tcpdump, entre otras. Todas permiten capturar y filtrar los Probe Request necesarios permitiendo obtener de éstos la información de tiempo de arribo, RSSI, dirección MAC, entre otros.

2.1.2.2. Bluetooth

Buscando identificar dispositivos aparece otra tecnología que se encuentra en la mayoría de los dispositivos móviles como es el Bluetooth. Esta tecnología definida en el estándar IEEE 802.15.1 se utiliza para comunicaciones inalámbricas de corta distancia entre dispositivos móviles o fijos creando Wireless Personal Area Network (WPAN).

Varios artículos analizan la posibilidad de contar y seguir personas utilizando esta tecnología [19] [25]. De forma similar a la detección por WiFi se busca identificar los dispositivos a partir de la información de los paquetes que envían los mismos. Esta forma de identificación tiene también otros usos como puede ser el de encontrar y seguir automóviles [26].

Para la detección a través de Bluetooth es necesario que el dispositivo lo tenga encendido y que permita descubrir dispositivos cercanos. Esto hace que la cantidad de dispositivos que se pueden detectar se restrinja bastante ya que no todos tienen esta tecnología habilitada todo el tiempo. Esto también afecta a la hora de detectar dispositivos a través de WiFi, pero al ser una tecnología de acceso a internet en su mayor uso, las personas suelen tener prendido el WiFi aunque no lo estén utilizando.

Otra de las diferencias de implementación respecto del WiFi es el hecho de que la detección por Bluetooth en general no se realiza de forma pasiva. Por el contrario, se disponen de sensores que activamente buscan los distintos dispositivos que se encuentran en el rango de detección. Esto se realiza mediante el envío de *beacons* donde se busca descubrir los dispositivos cercanos y obtener de las respuestas la dirección MAC de los mismos [27].

2.2. Selección de tecnologías

A continuación se pasa a describir cuáles de las tecnologías detalladas en la sección 2.1 fueron las elegidas para desarrollarse en este trabajo y también la jus-

tificación de haber descartadas las demás.

La opción más sencilla y en un principio más viable dentro de la utilización de sensores para la detección de personas parecía ser la de los sensores PIR. Estos sensores respondían al pasaje de una persona de forma binaria. Si la detectaban respondían con la salida lógica 1 mientras que si no detectaban actividad respondían con la salida lógica 0. De esta forma resultaba muy sencillo contar la cantidad de personas ya que era igual a la cantidad de salidas lógicas 1. Se hicieron pruebas locales instalando los sensores en una puerta y se observó que la duración de la salida lógica 1 era demasiado larga, impidiendo así detectar una persona que pasa justo después de la recién detectada. La duración mínima que se logró fue de aproximadamente 7 segundos. Este problema hizo que el sensor PIR no pudiera ser tenido en cuenta para una posible implementación, teniendo que pensar en otros posibles sensores.

De forma similar al sensor PIR se buscó evaluar el sensor Detector de Obstáculos IR mencionado en la sección 2.1.1.3. Este sensor no tenía la misma dificultad que el anterior, observando que los cambios entre las salidas lógicas 1 o 0 se realizaban casi instantáneamente. Realizando las mismas pruebas que se hicieron para el sensor PIR se observó otra dificultad: estos sensores eran muy sensibles a la luz solar. Por lo tanto los sensores se encendían aún cuando no pasara ninguna persona. Pensando en la situación real donde se deberían colocar estos sensores en un ómnibus, era muy probable que la luz solar incidiera en las mediciones. Se buscó corregir este defecto pero no se tuvo resultados satisfactorios.

Por último se probaron los sensores de ultrasonido descritos en la sección 2.1.1.4. Buscando la misma lógica que con los anteriores tipos de sensores se probaron con el pasaje de personas en una puerta. A diferencia de los sensores de infrarrojo, los de ultrasonido no se veían afectados por la iluminación, dado que su funcionamiento se basa en ondas de presión en vez de ondas electromagnéticas. A su vez, el funcionamiento respecto de los dos sensores anteriores es distinto ya que el sensor de ultrasonido da como resultado una medida de distancia. Al poder tomar muestras a una tasa de, por ejemplo, 10 muestras por segundo se logra evitar el problema que surgía en la utilización del sensor PIR. Realizando estas pruebas se observó que la precisión de las medidas se encontraba dentro de un rango de aproximadamente $\pm 5cm$, lo cual se consideró aceptable. Basándose en estas pruebas se decidió seleccionar este sensor para realizar un prototipo para la estimación de ocupación en el ómnibus.

Otra de las tecnologías vistas en la sección anterior para la detección y conteo de personas es la utilización de una cámara y la detección de objetos o caras. Como se vio en la sección 2.1.1.5 existen varios artículos que respaldan la utilización de estos métodos. Se decidió utilizar esta tecnología y sus métodos, desde los más básicos como *Background Subtraction* hasta otros más complejos como *Face Detection* y *Object Detection*. De esta forma se puede comparar varias y evaluar sus

fortalezas y debilidades.

En el caso de las tecnologías relevadas para el conteo de dispositivos se elige trabajar con WiFi ya que es una tecnología adoptada y utilizada en una mayor cantidad de dispositivos móviles en comparación al Bluetooth. Por otra parte, existen varios artículos que proponen métodos sencillos de detección de dispositivos por medio de WiFi y esto hace que haya un mayor respaldo científico. Los métodos de detección por Bluetooth implican muchas veces la utilización de aplicaciones en los dispositivos o el hecho de que los dispositivos tengan el Bluetooth prendido. En el caso de WiFi esto último también es necesario pero al ser una tecnología de acceso a internet es de esperar que esté prendido en una mayor cantidad de dispositivos.

Para la implementación de todas estas tecnologías es necesario alguna unidad de procesamiento que permita utilizarlas a bordo. Es decir, debe ser capaz de integrarse por ejemplo con una cámara para grabar y luego procesar estos videos, procesar los datos de los sensores de ultrasonido, y/o permitir capturar y filtrar los paquetes *Probe Request*. Esto se puede realizar con distintas unidades o con una sola que integre a todas las tecnologías.

Existen distintas unidades de procesamiento que permiten estas integraciones que van desde una Laptop hasta otras más simples como Arduino[5], Raspberry Pi[6] o MiniPC, entre otras. Se busca validar tecnologías para estimar la ocupación del ómnibus y también que estos prototipos sean viables, tanto en lo económico como en la practicidad y capaces de procesar y enviar los resultados en tiempo real o en semi-tiempo real (con un pequeño retardo). Se optó por utilizar la unidad Raspberry Pi ya que permitía incorporar todas las tecnologías planteadas, tenía un precio al que se podía acceder y también tenía la ventaja de que el equipo de trabajo ya tenía experiencia utilizándola.

Viendo que para algunos algoritmos asociados a la utilización de la cámara podía ser necesario un poder de procesamiento mayor al que provee la Raspberry Pi fue necesario contemplar también otras soluciones. Para esto se utilizó una Laptop (ver figura 2.11) con las siguientes características:

- Modelo: Lenovo LEGION
- Procesador: i7-7700HQ
- RAM: 16GB

[5]Documentación de Arduino: `https://www.arduino.cc/en/Guide/Introduction`
[6]Documentación de Raspberry Pi: `https://www.raspberrypi.org/about/`

Figura 2.11: Laptop Lenovo LEGION.[7]

Pero con el fin de realizar un prototipo que funcione sobre un ómnibus con un costo moderado fue que también se utilizó una MiniPC (ver figura 2.12) con las siguientes características:

- Modelo: Lenovo ThinkCentre
- Procesador: i3-7100T
- RAM: 4GB

Figura 2.12: MiniPC Lenovo ThinkCentre.[8]

[7]Imagen extraída de: `https://www.lenovo.com/us/en/laptops/legion-laptops/legion-y-series/Legion-Y520/p/88GMY500808`

[8]Imagen extraída de: `https://www.lenovo.com/es/es/desktops-and-all-in-ones/thinkcentre/m-series-tiny/ThinkCentre-M710q/p/11TC1MT710Q`

Capítulo 3

Generación de base de datos

Una vez seleccionados los métodos que se creyeron convenientes con sus respectivas tecnologías, era necesario poder validar éstos en una situación lo más real posible.

Debido a que no se contaba con una base de datos que permitiera comparar los tres métodos, fue necesario generar una base de datos propia. Para esto se realizaron pruebas de campo, donde utilizando un ómnibus de recorrido se instalaron las tres tecnologías en el mismo, de manera que utilizando pasajeros voluntarios se hizo una simulación de un recorrido normal.

Estas pruebas se dividieron en dos etapas. La primera fue de acercamiento al problema real, donde se evaluó la posición de las tecnologías en el ómnibus, la cantidad de sensores a utilizar, la forma de adquisición, los problemas de espacio y de iluminación, entre otras. Teniendo en cuenta todo esto, durante la segunda etapa se focalizó en la adquisición de datos para las tres tecnologías generando la base de datos que se buscaba.

A continuación se pasa a detallar el proceso de adquisición para las dos etapas, y al final se muestra la base de datos obtenida.

3.1. Adquisición de Datos

3.1.1. Primera Etapa

En la planificación previa se decidió dividir las pruebas en dos momentos del día, desde la media mañana hasta al medio día, y desde media tarde hasta la entrada del sol. Esto permitiría dar un descanso a los pasajeros colaboradores y abarcar la mayor cantidad de momentos de luz solar del día. Por cuestiones de disponibilidad de pasajeros y por la poca iluminación con la que contaba el ómnibus en su interior, se descartó la idea de realizar pruebas en la noche.

Para todas las pruebas se recabaron los siguientes datos de los individuos: edad, altura, tipo y marca de dispositivo móvil que usaría en la prueba. Esta prueba contó con la participación de un total de 18 personas incluido el chofer.

Femeninos	Menores	2
	Adultos	7
Masculinos	Menores	5
	Adultos	4
Dispositivos móviles		13
Edad media		28 años
Altura media		1,56m

Tabla 3.1: Datos de los pasajeros durante la mañana en la primer etapa.

Para esta prueba se colocaron los sensores de ultrasonido en el techo. Si bien la frecuencia con la que se obtienen medidas es variable, la frecuencia media fue de $\frac{1}{0{,}1187s} = 8{,}42$mps (muestras por segundo). La cámara se colocó en frente de la puerta apuntando hacia ésta y fue ajustada en las dos primeras capturas de forma de que no utilizara reajuste de iluminación. El dongle a diferencia de los demás se colocó a la mitad del pasillo, para tener el mínimo radio de potencia que abarcara todo el ómnibus.

De las tres tecnologías los datos que se esperan son: un video para la cámara, una serie temporal con las medidas para los sensores de ultrasonido y capturas de paquetes Probe Request realizadas con la Raspberry Pi y un dongle WiFi. Los datos recolectados para estas pruebas son:

Captura	**Duración** (min.)	**Paradas**	**Personas al inicio**	**Datos**
1	15	9	17	video, captura WiFi
2	15	8	1	video, serie, captura WiFi
3	6	4	1	video, serie, captura WiFi

Tabla 3.2: Datos recabados durante la mañana en la primera etapa.

(a) Captura sin reajuste de iluminación.

(b) Subida sin reajuste de iluminación.

(c) Captura con reajuste de iluminación.

(d) Subida con reajuste de iluminación.

Figura 3.1: Imágenes capturadas por la cámara en la mañana.

En la figura 3.1 se puede observar la posición y el campo de visión de la cámara para estas pruebas. En 3.1(a) y 3.1(b) se muestra un ejemplo de las dos primeras capturas donde no se hizo reajuste de iluminación. Esto se hizo de esta manera debido a que el método que utilizaba Background Subtraction se veía afectado con los cambios de iluminación. En 3.1(c) y 3.1(d) donde el reajuste de iluminación estaba activo, se puede observar que hay una mejora en la calidad de las imágenes y los cambios de iluminación entre estas dos. Esta captura a diferencia de las otras dos permitiría hacer Face Detection.

En la figura 3.2 se puede observar el recorrido realizado donde también se marcan los puntos utilizados como paradas. Aproximadamente por cada vuelta completa se recorrieron $420m$.

En las pruebas de la tarde se produjeron y realizaron cambios en el público y la posición de los sensores respectivamente. Esta vez la cantidad total de personas que participaron fueron 22.

Figura 3.2: Recorrido del ómnibus en la mañana con los puntos de parada.

Femeninos	Menores	2
	Adultos	11
Masculinos	Menores	4
	Adultos	5
Dispositivos móviles		17
Edad media		26 años
Altura media		1,56m

Tabla 3.3: Datos de los pasajeros durante la tarde en la primer etapa.

Los sensores de ultrasonido se colocaron en la baranda de la escalera de forma horizontal y se mantuvo la frecuencia de muestreo media de 8,42mps. La cámara se colocó en el techo para las primeras tres capturas y la última se hizo de la misma forma que en la mañana. Para todas las capturas se mantuvo el reajuste de iluminación. El dongle mantuvo su posición a la mitad del pasillo.

Captura	**Duración** (min.)	**Paradas**	**Personas al inicio**	**Datos**
1	10	7	21	video, captura WiFi
2	10	6	4	video, serie, captura WiFi
3	10	6	4	video, serie, captura WiFi
4	10	7	2	video, serie, captura WiFi

Tabla 3.4: Datos recabados en la tarde en la primer etapa.

(a) Campo de visión para las tres primeras grabaciones de video.

(b) Campo de visión para la última grabación.

Figura 3.3: Imágenes capturadas por la cámara en la tarde.

En la figura 3.3(a) se puede observar el campo de visión de la cámara cuando ésta fue colocada en el techo. Con flechas rojas se puede ver la nueva posición de los sensores de ultrasonido. Como se puede observar, de esta forma los pasajeros pasan por enfrente de los sensores con parte del torso y con las piernas en algunos casos.

En la figura 3.3(b) se observa el cambio en el campo de visión para la última captura. Este cambio se realizó para poder capturar el rostro de los pasajeros mientras ascendían al ómnibus, con la idea de poder usar este video con métodos que permitieran detectar el rostro.

Por cuestiones de accesibilidad en la tarde se cambio el lugar donde se hizo el recorrido (ver figura 3.4). Esta vez una vuelta completa es equivalente a $400m$ de recorrido.

Figura 3.4: Recorrido del ómnibus en la tarde con los puntos de parada.

Una vez finalizada la prueba de la tarde se dio por finalizada la primera etapa de pruebas en Mercedes.

3.1.2. Segunda Etapa

Al retornar a Montevideo y analizar los datos durante tres meses, se notó que la cantidad total de datos no era suficiente y que además algunos de ellos no servían para los métodos que se estaban implementando en ese momento. Por ejemplo los videos de las dos primeras capturas de la mañana no pudieron ser usados debido a temas de iluminación, ya que los videos estaban oscuros. También para la primera captura tanto en la mañana como en la tarde no se contaba con los datos de los sensores de ultrasonido, impidiendo comparar las tres tecnologías para esa captura. Poder realizar un análisis en conjunto con los datos provenientes de las tres tecnologías era fundamental para el proyecto y por lo tanto era necesario obtener más datos. Es entonces que se decidió realizar una segunda etapa de medidas.

Al igual que la primera etapa, se dividieron las pruebas en la mañana y la tarde. En la mañana se contó con un cantidad total de 23 personas, a la cual se les pidieron los mismos datos que en la primera etapa.

Femeninos	Menores	4
	Adultos	11
Masculinos	Menores	4
	Adultos	4
Dispositivos móviles		14
Edad media		30 años
Altura media		1,63 m

Tabla 3.5: Datos de los pasajeros de la mañana en la segunda etapa.

Para esta prueba se instaló la cámara enfrente a la puerta delantera del ómnibus, de manera que el campo visual abarcara la puerta y parte del pasillo. Esto permitiría a diferencia de las pruebas de la primera etapa, percibir de mejor manera el sentido del movimiento de los pasajeros. El campo visual de la cámara se puede apreciar en la figura 3.5.

Figura 3.5: Campo visual de la cámara para las pruebas de la mañana.

En el caso de los sensores se agregaron dos sensores más a los que ya había para agregar redundancia. Este cambio se debe a las observaciones de los datos capturados en la primera etapa donde en algunos casos alguno de los dos sensores no se activaba cuando un pasajero pasaba por enfrente, haciendo imposible saber el sentido del movimiento del pasajero. A su vez se construyó una estructura para sostener los cuatro sensores y que tuvieran una mayor estabilidad contra los movimientos del ómnibus, ya que las vibraciones del ómnibus afectaban los datos generando ruido en las medidas obtenidas. Con la plataforma construída se intentó disminuir este efecto.

(a) Los cuatro sensores de ultrasonido con su soporte.

(b) Puerta de acceso con el sistema de sensores de ultrasonido.

Figura 3.6: Nueva estructura para los sensores de ultrasonido.

También se buscó que la línea del soporte de los sensores tuviera la misma inclinación que la alineación de los escalones de la puerta. De esta forma se buscaba tratar de detectar el torso de la mayoría de las personas al pasar por enfrente de los sensores. Para el dongle no hubo cambios.

Captura	**Duración** (min.)	**Paradas**	**Personas al inicio**	**Datos**
1	10	12	2	video, serie, captura WiFi
2	10	11	6	video, serie, captura WiFi
3	10	12	18	video, serie, captura WiFi
4	10	11	2	video, serie, captura WiFi
5	10	11	2	video, serie, captura WiFi
6	10	11	12	video, serie, captura WiFi

Tabla 3.6: Datos recabados en la mañana en la segunda etapa.

En la tabla 3.6 se puede apreciar los datos de las capturas. Durante todas las capturas no hubo cambios en la disposición de los sistemas abordo.

Figura 3.7: Recorrido del ómnibus en la mañana con los puntos de parada.

Para esta prueba se utilizó el mismo recorrido que en la mañana de la etapa anterior, con la diferencia de que esta vez se planeó el recorrido para que tuviera dos paradas más de forma de aprovechar más el recorrido (ver figura 3.7).

En las pruebas de la tarde se mantuvo la posición y la configuración de las tres tecnologías. En total participaron 21 personas.

Femeninos	Menores	5
	Adultos	8
Masculinos	Menores	3
	Adultos	5
Dispositivos móviles		13
Edad media		27 años
Altura media		1,70 m

Tabla 3.7: Datos de los pasajeros en la tarde en la segunda etapa.

Captura	Duración (min.)	Paradas	Personas al inicio	Datos
1	10	10	15	video, serie, captura WiFi
2	10	10	9	video, serie, captura WiFi
3	10	10	6	video, serie, captura WiFi
4	10	11	3	video, serie, captura WiFi
5	10	10	7	video, serie, captura WiFi
6	10	7	15	video, serie, captura WiFi

Tabla 3.8: Datos recabados durante la tarde en la segunda etapa.

Al igual que en la mañana, en la tarde se agregaron dos paradas más al recorrido con respecto a la etapa anterior (ver figura 3.8)

Figura 3.8: Recorrido del ómnibus con los puntos de parada.

Si se observa la columna de "Duración" en las tablas 3.2, 3.4, 3.6 y 3.8, se puede observar que en la primer etapa se realizaron un total de 76 minutos de capturas, con alguna de ellas incompletas, y en la segunda etapa se totalizaron 120 minutos de capturas para las tres tecnologías.

3.2. Base de datos obtenida y ground truth

Los resultados de la generación de base de datos y el ground truth utilizado para comparar los resultados de las distintas tecnologías se detallan en el Apéndice A.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 4

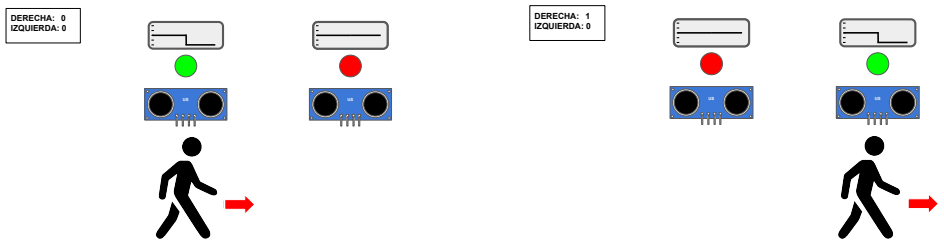
Implementación de algoritmos y experimentos

En este capítulo se detallan y analizan los distintos experimentos, métodos y algoritmos utilizados para validar las tecnologías seleccionadas en la sección 2.1. Esto se hace con el propósito de ver la viabilidad de utilizar estas tecnologías en un ómnibus, conocer sus ventajas y desventajas, y evaluar la posibilidad de enviar los datos en tiempo real.

A continuación se diferencian los distintos métodos para contar personas y dispositivos, exponiendo los resultados obtenidos en cada caso.

4.1. Sensores de Ultrasonido

Para este tipo de tecnología se diseñó un sistema capaz de distinguir el sentido del movimiento de las personas luego de pasar por enfrente de éste. La idea adoptada es la que se plantea en los artículos [4] y [5], donde utilizando dos sensores separados a una cierta distancia se puede determinar el sentido dependiendo de qué sensor se active primero. En la figura 4.1 se puede apreciar esta idea.



(a) Persona pasando por el primer sensor. (b) La misma persona pasando por el siguiente sensor.

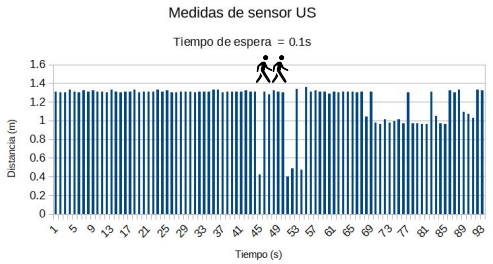
Figura 4.1: Detección del sentido con sensores de ultrasonido.

La figura 4.1 muestra la secuencia cuando una persona atraviesa el sistema de sensores hacia la derecha y como éste determinaría el sentido. En la figura 4.1(a) el individuo es detectado por el sensor que se encuentra a la izquierda, hasta ese momento los contadores "Derecha" e "Izquierda" permanecen en cero. Luego en la figura 4.1(b) el individuo es detectado por el siguiente sensor, por lo tanto se determina que el sentido del movimiento del individuo es hacia la derecha y se incrementa el contador "Derecha". De esta forma se podría contabilizar la cantidad de personas que circularon en un sentido y en el otro.

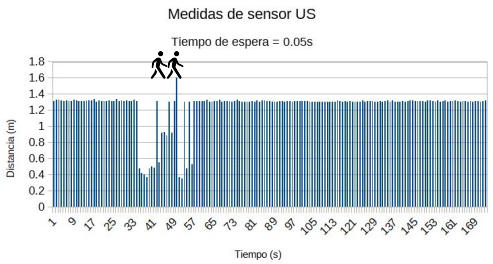
Dado que los sensores de ultrasonido están hechos para determinar la distancia a un objeto, la instalación de éstos se debe hacer de manera que enfrente a ellos haya un objeto plano fijo, de esta forma la medida de los sensores será siempre igual (estado normal) salvo cuando una persona pase por enfrente de éstos donde la medida debería ser menor (estado activo). Esta forma de instalar los sensores fue mostrada en el capítulo anterior en la figura 3.6, donde se muestra una forma de hacer esta instalación en el ómnibus. Esto también se puede ver en la figura 4.1 donde encima de los sensores de forma representativa se marca la distancia que están midiendo. Este objeto plano podría ser el piso si los sensores se colocan en el techo, o si éstos se colocan en una pared, podría ser una mampara o la pared de enfrente. También el sistema está pensado para que funcione en un lugar que solo permita que pase una persona a la vez por enfrente de los sensores. Dado que este sistema se pretende instalar en un ómnibus, el lugar debe ser en las escaleras de las puertas.

Los parámetros que se pueden variar en este sistema son: la distancia entre sensores y el tiempo de espera. La distancia entre sensores es la distancia a la cual se encuentra un sensor del otro y el tiempo de espera es el tiempo entre el fin de una medida del sensor y el inicio de la otra, necesario para que el sensor funcione correctamente. La frecuencia de muestreo es variable para el sistema de sensores ya que depende del tiempo en que demora el pulso emitido en retornar al sensor sumado al tiempo de espera. La distancia entre los sensores es necesaria para distinguir cuando un sensor pasó a estado activo antes que el otro, y el tiempo de espera define de forma parcial la resolución de la serie temporal generada por las medidas de los sensores. Si la distancia es demasiado pequeña, una persona puede pasar a una velocidad donde los sensores pasen a estado activo a la misma vez impidiendo determinar el sentido del movimiento. Tampoco la distancia puede ser muy grande dado que este sistema está pensado para ser instalado en un espacio reducido, como la escalera de un ómnibus y es necesario asegurase de que las personas pasen por ambos sensores. De la misma forma si el tiempo de espera es demasiado grande, una persona podría pasar por enfrente de los sensores con una velocidad que impida que el sensor la detecte. El tiempo de espera a su vez tiene una cota inferior, dado que está limitado por la electrónica del sensor. También la distancia entre el objeto plano y los sensores es un parámetro que se podría cambiar, pero éste está definido por las dimensiones del ómnibus, ya sea por la distancia entre el techo y el piso o el ancho de la escalera de la puerta.

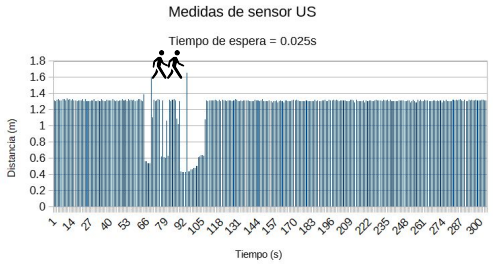
Para ver qué tiempo de espera usar es necesario hacer distintas pruebas. La primera es ver cuál es el tiempo máximo que permite detectar a una persona pasando a una velocidad normal. La otra es ver si existe un tiempo tal que permita distinguir entre dos personas que pasan muy cerca entre sí.



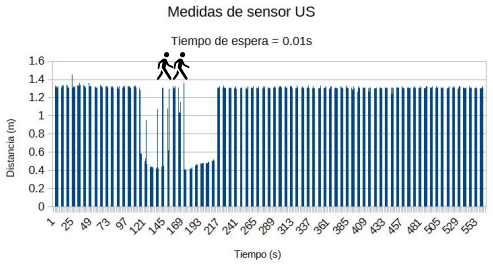
(a) Frecuencia de muestreo promedio $f =$ 9mps.



(b) Frecuencia de muestreo promedio $f =$ 17mps.



(c) Frecuencia de muestreo promedio $f =$ 30mps.



(d) Frecuencia de muestreo promedio $f =$ 78mps.

Figura 4.2: Ejemplo de medidas cuando dos personas pasan muy cerca para distintos tiempos de espera.

En la figura 4.2 se pueden ver los resultados de la prueba de dos personas pasando muy cerca entre sí por enfrente de un sensor. Variando el tiempo de espera, se observó para qué tiempo se logra diferenciar a simple vista la aparición de las dos personas. Desde la figura 4.2(a) hasta la figura 4.2(d) se va disminuyendo este tiempo, lo cual se puede ver como un aumento de la frecuencia de muestreo promedio. Se puede apreciar como para $0,01$s se puede ver el pasaje diferenciado de ambas personas y en este caso se obtiene una frecuencia de muestreo de 78mps. Para tiempos menores a éste los sensores no respondían de la misma manera, devolviendo medidas erróneas. Esta limitante hizo que debido a que se necesitan dos sensores para armar el sistema y que estos miden secuencialmente para que no se interfieran entre sí, la frecuencia del sistema está limitada a la mitad de 78mps.

Para la distancia entre sensores no fueron necesarias muchas pruebas, con una distancia mínima de 20cm se logra diferenciar la activación secuencial de los sen-

sores a una velocidad normal de pasada.

Los primeros experimentos para ver el funcionamiento del sistema se hicieron en la entrada de una sala.

Figura 4.3: Sistema de sensores instalados en la parte superior del marco de una puerta.

En la figura 4.3 se pueden ver los sensores colocados en la parte superior del marco de la puerta apuntando hacia el piso, esta configuración está pensada para que los sensores midan la distancia del marco a la cabeza o los hombros de un individuo cuando éste pasa por debajo de ellos. En la figura 4.4 se muestra un ejemplo de las medidas obtenidas cuando un individuo cruzó la puerta para entrar a la sala.

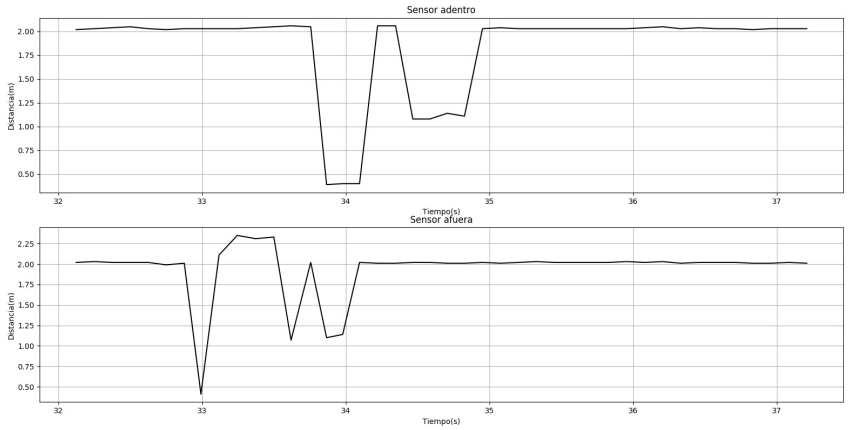


Figura 4.4: Series temporales para ambos sensores cuando alguien entró a la sala.

Al observar los datos en los casos que alguien atraviesa el sistema de sensores se observó que los resultados eran similares a los que se muestran en la figura

4.4, donde en vez de obtener una medida siempre menor mientras el individuo se encuentra debajo de los sensores, la serie también muestra retornos al valor medio (distancia al piso), y en algunos casos valores que superaban esta media. Este comportamiento es debido a que estos sensores están pensados para utilizarse contra superficies planas, y los rebotes contra superficies como el cuerpo de un ser humano generan estas medidas de distancia erróneas.

Asimismo, el comportamiento de las series temporales generadas por las medidas de los sensores permiten diferenciar cuando una persona está pasando y cuando no, dado que mientras los pulsos de los sensores se reflejan contra el piso la medida es estable. Definiendo un entorno alrededor del valor medio, se podría afirmar que una persona está pasando cuando los valores devueltos por el sensor están fuera de este entorno. En la figura 4.4 también se puede diferenciar como primero se activa un sensor y después el siguiente que se encuentra a 35cm de distancia.

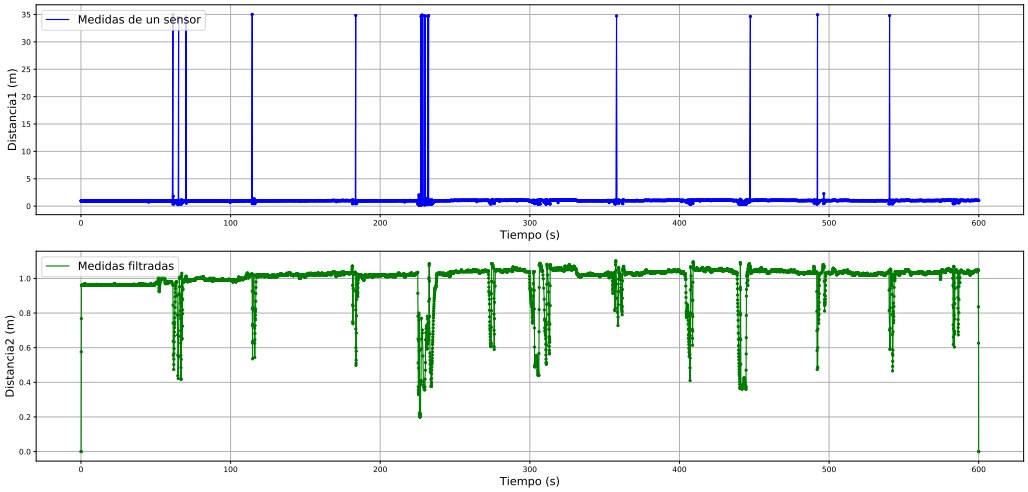
Debido a que los sensores en algunos momentos devuelven medidas fuera del entorno de la media cuando no se produce ningún evento, es necesario hacer un filtrado que elimine o disminuya el efecto de estas medidas. Para esto se utilizó un filtro de mediana móvil seguido de uno de media móvil. Este último con el objetivo de suavizar la serie. Este procesamiento también ayuda a que las medidas no sean tan cambiantes cuando una persona está cruzando por enfrente de los sensores y elimina o disminuye medidas extremadamente grandes que el sensor devuelve cuando no detecta el retorno del pulso emitido.

En la figura 4.5 se puede observar las medidas de un sensor extraídas del archivo *UsE2man1.csv* (Base de datos), junto con el resultado después de filtrarlas. Con color azul se remarcan las medidas devueltas por el sensor y con color verde las filtradas. La figura 4.5(a) muestra las medidas de un sensor en una de las pruebas de la segunda etapa. Se puede ver como las medidas exageradamente grandes desaparecen después del filtrado. La figura 4.5(b) es solamente un acercamiento de la anterior, donde se observa con más precisión como se mitigan las variaciones en las medidas cuando una persona está pasando por los sensores. Este filtrado se hizo con una ventana de 5 muestras para ambos filtros.

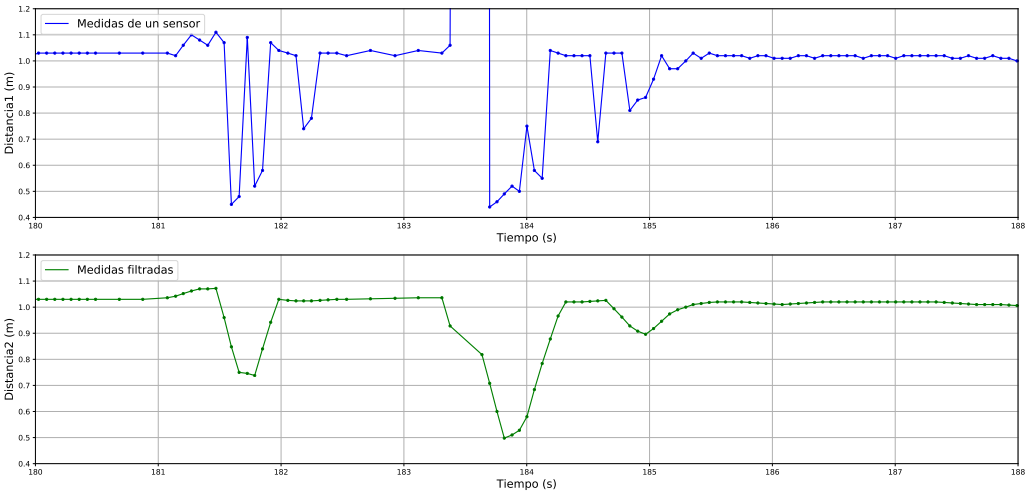
Para poder hacer el conteo de las personas en el ómnibus es necesario utilizar un algoritmo con una lógica capaz de distinguir los dos sentidos posibles de las personas, tener en cuenta cuando alguien se queda parado frente a ellos sin avanzar y poder diferenciar entre dos personas pasando muy cerca entre sí. Previo al algoritmo se planteó un diagrama de estados que resolviera el conteo.

En la figura 4.6 se puede ver dicho diagrama. Éste cuenta con cuatro estados: **Normal**, **Proceso entrando**, **Proceso entrando 2**, **Proceso saliendo** y **Proceso saliendo 2**. También cuenta con cuatro entradas distintas: cuando se activa y desactiva el sensor de afuera son dos entradas distintas, para el cual el sensor de afuera hace referencia al sensor que se encuentra fuera de la sala, y cuando se

activa y desactiva el sensor de adentro son las otras dos entradas, haciendo referencia al sensor dentro de la sala. En el caso del ómnibus donde los dos sensores van del lado de adentro del ómnibus, el sensor de afuera hace referencia al más cercano a la puerta y el sensor de adentro al más cercano al pasillo. También en el diagrama se hace referencia a tres contadores: $Cont_E$, $Cont_S$ y $Cont_{tiempo}$, donde los dos primeros se incrementan en 1 cuando se dice que una persona entró o salió del ómnibus respectivamente, y el tercero es un contador temporal de espera.



(a) Medidas de un sensor y el resultado después de filtrar para 12 paradas.



(b) Medidas de un sensor y el resultado después de filtrar para 1 parada.

Figura 4.5: Resultados del pre-procesamiento sobre las series.

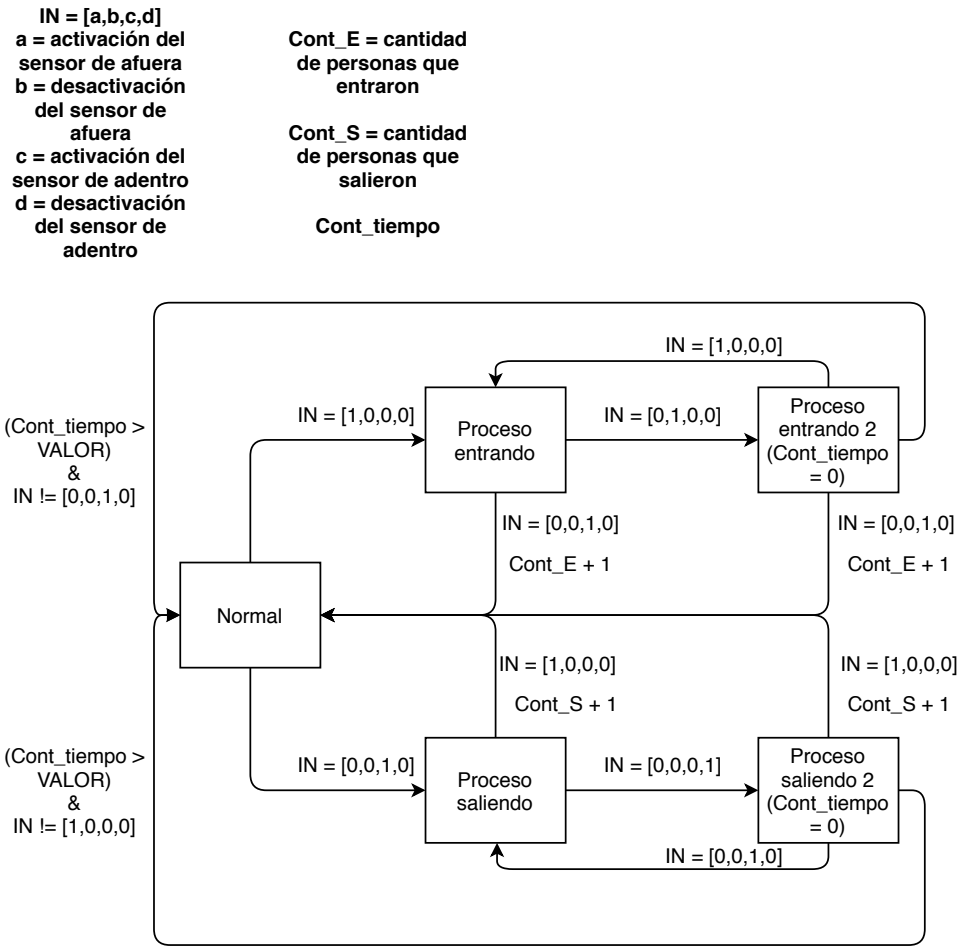


Figura 4.6: Diagrama de estados para el conteo de personas con el sistema de sensores de ultrasonido.

Cuando se hace referencia a la activación de alguno de los sensores es en el momento en el que la medida del sensor salió del rango alrededor de la media, y se dice que se desactivó en el momento que la medida del sensor pasa a estar adentro del rango nuevamente. Partiendo del estado **Normal**, el cual corresponde a cuando ninguna persona está pasando por enfrente de los sensores, solo se sale de ese estado si alguno de los dos sensores se activa. Si el sensor de afuera se activa, quiere decir que una persona empezó el proceso de entrar al ómnibus, y se va al estado **Proceso entrando**. Si antes de que se desactive el sensor de afuera se activa el de adentro, se incrementa el contador de personas que entraron y se retorna al estado **Normal**. Por el contrario si primero se desactiva el sensor de afuera se pasa al estado **Proceso entrando 2**, el cual contempla el caso de que una persona se haya parado enfrente del sensor de afuera y en vez de seguir subiendo haya retornado, en este caso pasado un tiempo pre establecido se retorna al estado **Normal**. Si antes de este tiempo el sensor de adentro se activa se incrementa el contador de personas que entraron y se vuelve al estado **Normal**, este

caso donde el sensor de afuera se activa, desactiva y luego se activa el sensor de adentro se puede dar cuando cruza un niño o una persona delgada. Siguiendo en el estado **Proceso entrando 2**, si el sensor de afuera se vuelve a activar se retorna directamente al estado **Proceso entrando**. Dado que el diagrama de estados es simétrico, el pasaje por los estados **Proceso saliendo** y **Proceso saliendo 2** es similar a lo explicado anteriormente pero para el sensor de adentro. Desde el pre-procesamiento de las medidas con los filtros de mediana y media móvil, hasta la implementación del algoritmo que implementa el diagrama de estados se hizo con el lenguaje de programación Python 2.7.

Obtenidos los datos de la primer etapa del viaje a Mercedes, se observó que el sistema tenía carencias. Muchas veces al pasar una persona solo uno de los sensores se activaba impidiendo determinar el sentido de movimiento de la persona. Este comportamiento es debido al posicionamiento de los sensores dentro del ómnibus, y a la gran presencia de público menor de edad el cual cuenta con una fisionomía menor a la de los adultos. La primer consecuencia se debe por ejemplo, en el caso en que se colocaron los sensores en el techo muchas veces las personas al ingresar al ómnibus esquivaban el campo de visión del que se encontraba del lado de adentro, en el momento que giraban para tomar el pasillo. Con respecto a los menores de edad, muchas veces al pasar por enfrente del sistema se generaba una señal de muy pocas medidas fuera del rango normal o a veces ninguna, confundiéndose con ruido o pasando desapercibido. Todo esto se tuvo en cuenta para mejorar el sistema para la segunda etapa de pruebas en Mercedes. Esta vez a cada sensor se le agregó otro sensor al lado de manera de agregar redundancia a la medida (ver figura 3.6), de esta manera el nuevo sistema cuenta con dos sensores en la parte de adentro y dos en la parte de afuera, cualquiera del par que se active habilita al cambio de estado en caso de que corresponda. También el nuevo sistema se instaló en la baranda de la escalara con el mismo ángulo que ésta de forma que las personas al pasar por el sistema pasen obligadamente por el campo de visión de los sensores en su mayoría con el torso de su cuerpo.

Los primeros errores que se encontraron para este nuevo sistema correspondían a errores en las medidas o en el pre-procesamiento. Lo primero que se observó fue que cada vez que el ómnibus arribaba a una de las paradas los sensores se activaban casi simultáneamente, probablemente debido al frenado. Esto hacía que se contara una persona de más, ya sea bajando o subiendo, por cada parada. Para solucionar esto se agregó al algoritmo una lógica donde si los sensores de adentro y de afuera se activan simultáneamente o con muy pocas medidas de diferencia, se ignoraba la cuenta. Esto también ayudo a eliminar varios falsos positivos que se daban en otras circunstancias.

En la figura 4.7 se muestra el comportamiento de los sensores en el momento que dos personas están subiendo al ómnibus.

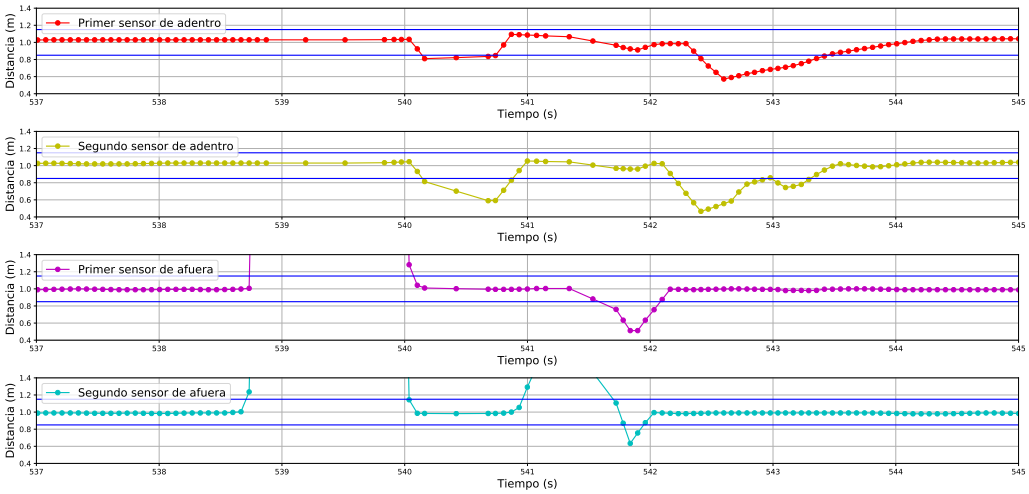


Figura 4.7: Comportamiento de los sensores cuando dos personas estan subiendo al ómnibus

Para todas las series de los cuatro sensores se muestra con líneas azules el entorno pre-establecido alrededor del valor medio. Como se puede ver primero los sensores de afuera se activan y un instante después de desactivarse se activan los sensores de adentro, en ese caso el algoritmo cuenta una persona entrando. Inmediatamente después que se desactivan los sensores de adentro se activan los sensores de afuera y cuando éstos se desactivan se activan los sensores de adentro, en ese caso se cuenta una persona más entrando. Por lo tanto el algoritmo cuenta dos personas entrando coincidiendo con lo esperado. Como se puede ver los umbrales que delimitan el entorno del valor medio, son parámetros claves para la detección. Otros parámetros importantes son los anchos de las ventanas utilizadas en los filtros para el pre-procesamiento de las series temporales. En la figura 4.7, la ventana es de 5 muestras.

En la figura 4.8 se muestra las mismas medidas que en 4.7, pero la ventana de los filtros es de 9 muestras.

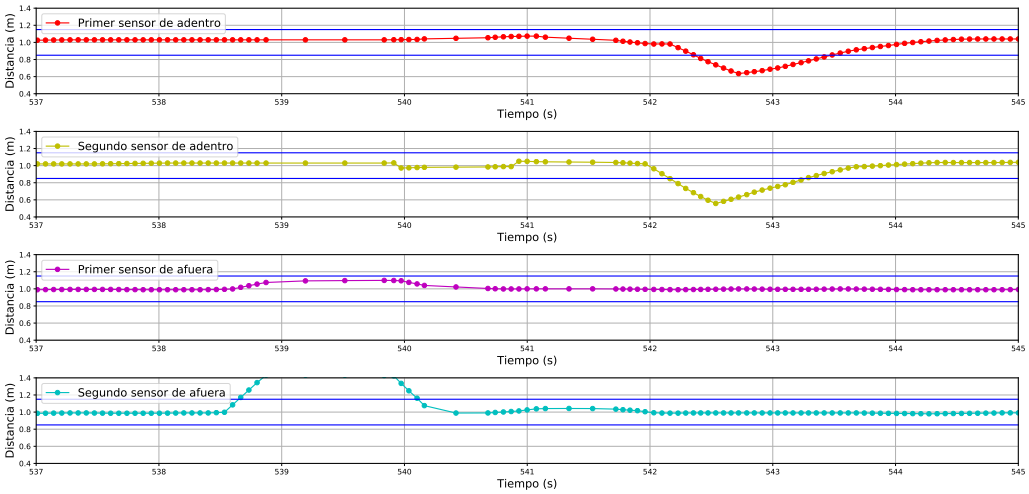


Figura 4.8: Comportamiento de los sensores cuando dos personas están subiendo al ómnibus para un pre-procesamiento distinto.

Como se puede observar en varios sensores se pierde la información cuando la persona cruza. Manteniendo los mismos umbrales solo se detecta una activación de los sensores de afuera y una de los sensores de adentro, por lo tanto se cuenta solo una persona. Cuando el sensor de afuera se desactiva, siguiendo con el diagrama de 4.6, se pasa al estado **Proceso entrando 2**, si el tiempo de espera expira antes de que se active el sensor de adentro, no se contará ni siquiera una persona. Por lo tanto el contador de los estados **Proceso entrando 2** y **Proceso saliendo 2**, puede ser determinante en algunos casos.

Debido a los cambios de los resultados con la variaciones de los parámetros, fue necesario hacer una evaluación de éstos para distintas combinaciones de parámetros. Para eso se utilizaron distintos indicadores que permiten evaluar la estimación por parada comparándola con los valores reales de estimación (True). Estos indicadores son: el coeficiente de correlación de Pearson, el promedio de ocupación (P.O.(%)) y la raíz del error cuadrático medio (RMSE). En la tabla 4.1 se muestra a modo de ejemplo la comparación entre la estimación y el True para la captura Man_1.

	Man_1	
Paradas	**True**	**Est.**
1	6	4
2	8	6
3	10	8
4	14	10
5	12	9
6	9	10
7	6	7
8	4	5
9	2	3
10	5	3
11	7	5
12	7	5
Pearson	0,88	
P.O. (%)	90,29	
RMSE	2,02	

Tabla 4.1: Comparación de la estimación por paradas con respecto al valor real utilizando un ancho de ventana de 5 muestras.

La tabla 4.1 permite comparar el valor estimado (Est.) junto al valor real (True) para cada parada de la captura Man_1. Al final de esta tabla se puede ver el valor arrojado por cada indicador. El coeficiente de Pearson permite ver la relación lineal entre los valores de True y la estimación, mientras más cercano a 1 sea el valor más fuerte es esta relación. El promedio de ocupación permite ver en cuanto se acerca la estimación por parada al valor real. Si el valor es menor al 100 % quiere decir que en promedio la estimación es menor, cuando el valor está por encima del 100 % la estimación es mayor en promedio. El valor de RMSE indica aproximadamente cuanto es la diferencia de personas en promedio por parada.

Fijando el ancho del rango del valor medio a un valor razonable, se puede ver como varían los indicadores con el ancho de la ventana usada en el filtrado previo a la estimación.

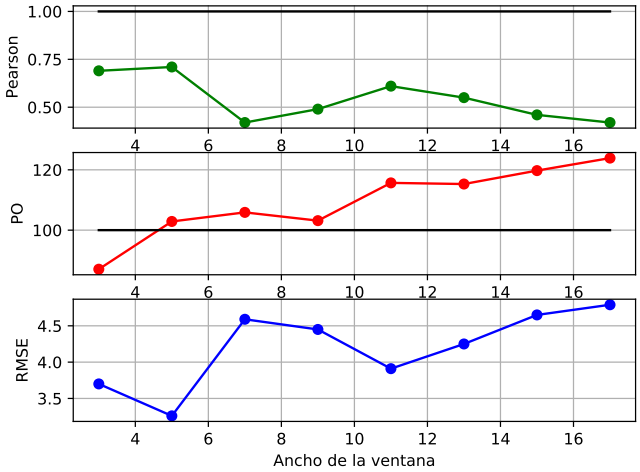


Figura 4.9: Variación de los indicadores para distintos anchos de filtro.

En la figura 4.9 se puede ver dicha variación. Cada valor de las gráficas es el promedio de los indicadores de todas las capturas de la segunda etapa. Como se puede ver, para un ancho de 5 muestras coincide que el valor es el mejor para los tres indicadores. En la tabla 4.2 se muestran los valores de los indicadores para todas la pruebas de la segunda etapa con un ancho de filtro de 5 muestras.

	Indicadores		
	Pearson	**P.O. (%)**	**RMSE**
Man_1	0,88	90,29	2,02
Man_2	0,53	93,30	1,22
Man_3	0,91	167,69	4,51
Man_4	0,74	88,38	2,00
Man_5	0,82	127,82	2,02
Man_6	0,55	104,27	2,10
Tar_1	0,47	173,07	5,32
Tar_2	0,52	45,37	4,87
Tar_3	0,74	135,38	2,41
Tar_4	0,44	49,66	3,86
Tar_5	0,97	118,28	1,82
Tar_6	0,98	170,04	2,52

Tabla 4.2: Indicadores de los resultados de ocupación por parada obtenidos a partir de la segunda etapa de pruebas.

4.2. Cámara

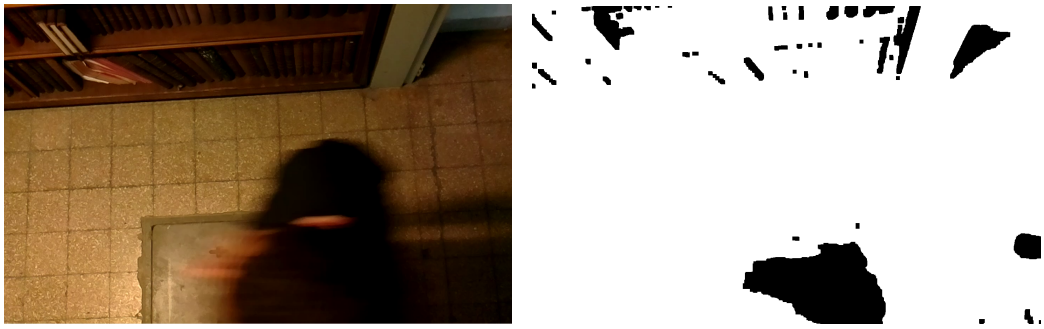
En la sección 2.1.1.5 se describieron 3 métodos distintos con los que era posible realizar la detección de personas en un video. Estos métodos fueron: Background Subtraction, Face Detection y Object Detection. A continuación se pasa a detallar el motivo por el cual fue descartado Face Detection y Background Subtraction y también porqué se procedió a trabajar con Object Detection.

4.2.1. Background Subtraction

Dentro de las funciones explicadas en la sección 2.1.1.5, para realizar Background Subtraction, la que se utilizó fue *BackgroundSubtractorMOG2* dado que ofrece un mejor desempeño frente a cambios de iluminación. Para probar el funcionamiento de la detección de personas con este método se realizaron una serie de pruebas en un ambiente controlado, como lo es la entrada a un sala.

La primera prueba realizada fue colocar un celular en el techo de un pasillo de forma de grabar un video donde los sujetos de prueba pasaban en ambas direcciones. Luego a dicho video se lo procesó con el algoritmo de Background Subtraction desarrollado. El resultado obtenido por este método no fue satisfactorio, debido al hecho de que cuando una persona pasaba por debajo de la cámara, la cabeza y los hombros ocupaban prácticamente todo el campo visual. Cuando esto sucedía la cámara del teléfono automáticamente cambiaba su balance de blancos afectando de esta forma la detección, dado que los píxeles que pertenecían al fondo ahora se detectaban como pertenecientes a objetos en movimiento debido a que cambiaron su valor. Ésto se puede observar en la figura 4.10, donde los píxeles negros corresponden a los que se detecta como fondo y los blancos a los objetos en movimiento. Para solucionar este problema se procedió a adquirir una PiCamera [1], dado que su precio era accesible y también se podían controlar las características de la grabación.

[1] `https://picamera.readthedocs.io/en/release-1.13/`



(a) Imagen capturada con celular.

(b) Imagen procesada con Background Subtraction.

Figura 4.10: Imágenes obtenidas por un celular y procesadas mediante Background Subtraction

La siguiente prueba fue similar a la anterior, solo que en vez de utilizar la cámara de un teléfono se utilizó la PiCamera. Para evitar los problemas tenidos anteriormente se dejó que la cámara ajuste sus parámetros automáticamente, como el ISO, la ganancia y la exposición[2], durante unos pocos segundos apuntando solamente al fondo y luego se fijaron esos valores de forma que no cambiaran cuando una persona ingresara al campo visual. A su vez se probó variar la posición de la cámara, estando primero en el techo apuntando hacia abajo y luego apuntando hacia la puerta. En la figura 4.11 se puede observar el funcionamiento del algoritmo, en la figura 4.11(b) se observa el contorno detectado y en la figura 4.11(a) se pueden ver las líneas utilizadas para contar, siendo la línea azul la utilizada para contar hacia abajo y la roja para contar hacia arriba. A su vez en la zona superior izquierda de la imagen también se observa la cuenta de personas. Los resultados obtenidos de esta forma fueron suficientemente satisfactorios como para que parezca posible realizar el conteo por este método. Estos resultados están en la tabla 4.3.

(a) Imagen resultado del algoritmo basado en Background Subtraction.

(b) Contorno detectado.

Figura 4.11: Resultados obtenidos mediante el algoritmo Background Subtraction.

[2] https://es.wikipedia.org/wiki/Escala_de_sensibilidad_fotogr%C3%A1fica

	Real		Est.	
	UP	**DOWN**	**UP**	**DOWN**
Cantidad de personas	8	6	6	8

Tabla 4.3: Resultados obtenidos con el algoritmo basado en Background Subtraction.

Sin embargo cuando se procedió a procesar los videos de la base de datos obtenidos en el capítulo 3 hubo inconvenientes. Por ejemplo la altura del techo del ómnibus no era lo suficientemente alta como para poder poner la cámara en el techo apuntando hacia abajo, ya que cuando una persona pasaba ocupaba todo el campo visual. Por esto es que se optó por ponerla sobre un costado apuntando hacia la puerta, pero esto trajo otro inconveniente, el fondo no era fijo, por lo tanto la detección no funcionaba como era esperado. En la figura 4.12 se puede observar este comportamiento. Por esta razón se descartó seguir trabajando con este método, ya que sus resultados no eran satisfactorios para la base de datos desarrollada.

(a) Imagen de la base de datos generada en Mercedes.

(b) Procesamiento realizado mediante Background Subtraction.

Figura 4.12: Resultados obtenidos al procesar la base de datos con el algoritmo basado en Background Subtraction.

4.2.2. Face Detection

Para estudiar el desempeño de la detección de caras mediante Face Detection también se realizaron algunas pruebas preliminares antes de procesar la base de datos. En primera instancia se intentó verificar el funcionamiento del algoritmo con una imagen, cuyo resultado se puede ver en la figura 2.7, sin embargo si algún rostro estaba levemente rotado o parcialmente ocluido entonces no era detectado. Esto se podía mejorar variando algunos parámetros del algoritmo, pero al realizar esta variación aumentaba considerablemente la cantidad de falsos positivos detectados. Esto sucedió trabajando con imágenes estáticas, cuando se intentó procesar con este método la base de datos obtenida, los resultados no fueron satisfactorios. Esto se puede observar en las figuras 4.13, donde en la figura 4.13(a) la detección

es realizada correctamente, pero en el siguiente frame aparece un falso positivo, lo cual se puede observar en la figura 4.13(b). Algunos frames después también se observa que el rostro no es detectado sin embargo aparece un falso positivo (figura 4.13(c)), a su vez en la figura 4.13(d) no se observa ninguna detección. Algunos motivos por lo que ésto puede estar sucediendo es que el modelo busca caras de frente, por lo tanto si la cara está un poco de costado ya el algoritmo no la detecta.

(a) Detección correcta de cara.

(b) Detección de cara junto con falso positivo.

(c) Falso positivo.

(d) Sin detección.

Figura 4.13: Procesamiento de imágenes mediante Face Detection.

Otro inconveniente que tiene este método es que el objetivo es contar la cantidad de personas que suben y también las que bajan, sin embargo como el Face Detection solo detecta caras de frente, serían necesarios dos cámaras, porque sino las personas que bajan no serían detectadas dado que en ningún momento estarían de frente a la cámara.

Dado el hecho de que la detección dependía fuertemente de la posición relativa de la cara con respecto a la imagen capturada por la cámara y también por el hecho de que para que funcione correctamente iban a ser necesarias dos cámaras

por cada puerta, fue que se decidió descartar este método de detección.

4.2.3. Object Detection

Con el objetivo de estudiar el desempeño del método Object Detection para realizar la detección de personas se realizaron en primera instancia pruebas con imágenes, cuyo resultado se puede observar en la figura 2.8. Al obtenerse resultados aceptables se procedió a desarrollar un algoritmo que permitiera realizar el conteo de personas. Un diagrama de bloques de este algoritmo se puede observar en la figura 4.14. El funcionamiento del algoritmo es el siguiente: se lee un frame del video y se le realiza Object Detection, si alguna persona es detectada entonces se inicializa un tracker por cada persona detectada y durante los siguientes 15 frames se realiza tracking. Si en algún momento la persona trackeada cruza la línea de izquierda a derecha entonces se cuenta que una persona subió y si la cruza de derecha a izquierda se cuenta que una persona bajó. Esto se puede observar en la figura 4.15, donde en la figura 4.15(a) el centro de la persona detectada, que está subiendo, se encuentra a la izquierda de la línea y en el frame siguiente, figura 4.15(b), se encuentra a la derecha por lo tanto cruzó la línea. Se puede observar que el contador *UP* se incrementó en 1, por lo tanto se contó una persona subiendo. Luego de esto se vuelve a leer otro frame y realizar Object Detection.

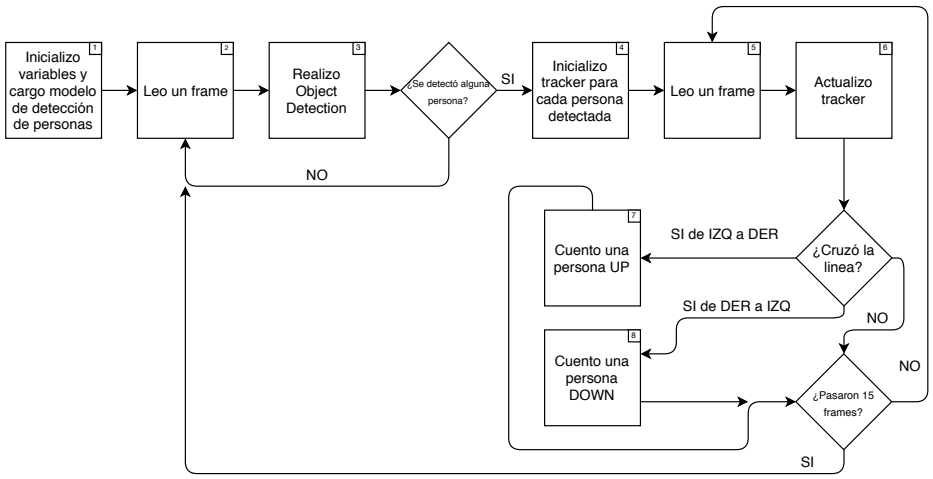


Figura 4.14: Diagrama de bloques del algoritmo utilizado para contar basado en Object Detection.

(a) Persona antes de cruzar la línea. (b) Persona después de cruzar la linea.

Figura 4.15: Proceso de conteo de personas.

Con el algoritmo desarrollado se procedió a procesar la base de datos generada y se encontró que era necesario realizarle algunas modificaciones para que éste funcione con un mejor grado de precisión. En primera instancia fue necesario realizar un pre-procesamiento a los videos obtenidos durante la segunda etapa, ya que estaban bastante oscuros y eso afecta el desempeño de Object Detection. Para esto lo que se utilizó fue una función que realiza Corrección Gamma, lo cual consiste en mejorar el nivel de iluminación para los píxeles más oscuros, ampliando su espectro. Ésto en el diagrama de bloques de la figura 4.14 se realiza entre los bloques 2 y 3. Los resultados de ésto se pueden ver en las figuras 4.17(a) y 4.17(b). Otra de las modificaciones realizadas fue agregar una bandera para saber si una persona que esta siendo trackeada ya se la contó como que subió o bajó, de forma que no se tengan múltiples subidas y bajadas por la misma persona. Dicha bandera se pierde en el proceso de detección ya que cuando se realiza la detección, a todas las personas detectadas se las toma como personas nuevas. Para disminuir los errores provenientes de esta lógica lo que se realizó fue fijarse si los centros de las nuevas personas detectadas estaban cerca de los centros de las personas trackeadas en el frame anterior, si así lo era se la consideraba como la misma persona y de esa forma se mantenía la bandera. Un diagrama de bloques del algoritmo modificado se puede ver en la figura 4.16, donde los elementos en rojo representan las modificaciones realizadas.

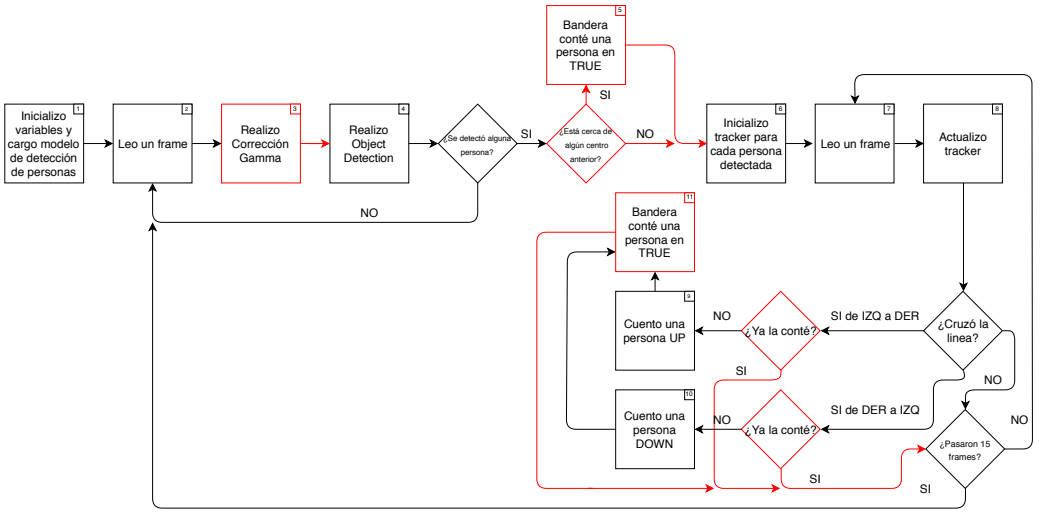


Figura 4.16: Diagrama de bloques del algoritmo utilizado para contar basado en Object Detection modificado.

(a) Sin Corrección Gamma. (b) Con Corrección Gamma.

Figura 4.17: Corrección Gamma.

Algunos de los parámetros que se podían modificar para mejorar el desempeño del algoritmo eran por ejemplo, la ubicación de la línea con la que se cuenta si una persona sube o baja, la cual se decidió ubicar a la derecha de la puerta, como se puede observar en la figura 4.18. El siguiente parámetro que se modificó para estudiar el desempeño fue la cantidad de frames durante los cuales se realizo trackeo, entre detección y detección. En el diagrama de bloques de la figura 4.16 este parámetro fue de 15 cuadros, pero también se probó con 30.

Figura 4.18: Imagen donde se puede observar la línea utilizada para contar.

El código en el que se basa el diagrama de bloques de la figura 4.16 se adjunta en el apéndice B, el cual fue desarrollado en Python 2.7. Como se mencionó en la sección 2.1, la librería utilizada para realizar Object Detection fue Tensorflow. Más específicamente los modelos utilizados para realizar la detección de objetos fueron: *ssd_mobilenet_v1_coco_11_06_2017* y *faster_rcnn_resnet101_coco_11_06_2017*, siendo el primero con el que se obtuvieron mejores resultados tanto de detección como en tiempo de procesamiento. La función utilizada para realizar la detección fue: *(boxes, scores, classes, num) = sess.run([detection_boxes, detection_scores, detection_classes, num_detections], feed_dict = image_tensor: image_np_expanded)*, donde los parámetros proporcionados a la función *sess.run* fueron inicializados previamente y el parámetro *image_np_expanded* corresponde al frame que se está leyendo del video, modificado para que el algoritmo funcione correctamente. Lo que devuelve la función son cuatro variables, *classes*: donde se indican las clases detectadas en el frame, *scores*: donde aparece el porcentaje de confianza de que el objeto detectado pertenezca a la clase indicada, *num*: número de detecciones y por último *boxes*: donde se devuelve el rectángulo que delimita al objeto detectado. Como el objetivo del algoritmo es detectar personas, se filtran las detecciones realizadas de modo que solo permanezcan los rectángulos que pertenezcan a la clase *person* (persona en inglés) y cuyo *score* sea mayor al 50 %. De esta forma solo se trabaja con la detección de personas cuyo porcentaje de confianza sea considerablemente bueno.

A su vez se puede observar que el tracker utilizado pertenece a la librería *dlib* la cual es una librería que contiene algoritmos de Machine Learning y herramientas para crear programas complejos para resolver problemas del mundo real. Dentro de esa librería la función que se utiliza es: *dlib.correlation_tracker()*, a la cual hay que inicializarla pasándole un frame de video junto con el rectángulo donde se detectó una persona, el cual fue generado por la función de detección explicada anteriormente. Una vez hecho esto con las funciones *update(image_np)* y *get_position()* se obtiene otro rectángulo donde la función predice que se encuentra la persona de-

tectada anteriormente, en el nuevo frame.

El algoritmo funciona a grandes rasgos de la siguiente forma: se realiza una detección y luego por una cierta cantidad de frames se realiza trackeo a las personas detectadas, cuando se termina de trackear se vuelve a realizar una detección y así sucesivamente. Esto se lleva a cabo de esta forma debido a que realizar trackeo es menos costoso, más que nada en tiempo de ejecución, que realizar detección en todos los frames. Es por esto que uno de los parámetros que se varía fue la cantidad de frames durante los cuales se realiza trackeo, probando principalmente con 15 y 30, ya que se probaron con otros valores intermedios pero los resultados no diferían demasiado. Idealmente lo mejor sería realizar siempre trackeo, ya que es más rápido, pero si aparece alguna persona mientras estoy realizando trackeo no se detecta, entonces no se contaría. Es por esto que existe un compromiso entre realizar detección lo suficientemente seguido como para no perder personas pero no tanto como para que el algoritmo demore mucho en su ejecución. Los parámetros elegidos fueron 30 y 15, ya que como la cámara graba a 30fps (frames per second), entonces 30 frames corresponden a 1 segundo. Por lo tanto se realizaría detección cada 1 segundo, si se utiliza el parámetro 30 y cada medio segundo si se utiliza 15. Esos tiempos se consideran aceptables para la implementación deseada.

Para analizar el desempeño de los algoritmos desarrollados se procedió a implementarlos sobre la base de datos desarrollada. Para esto lo que se hizo fue cortar los videos obtenidos en videos más cortos, uno por cada parada realizada por el ómnibus. Esto se consideró necesario, ya que en los videos la mayor parte del tiempo el ómnibus estaba en movimiento y no había pasaje de personas, por lo cual el algoritmo no realizaba ninguna acción. Sin embargo con los videos cortados por parada, siempre hay pasaje de personas, entonces de esta forma se centró en el correcto funcionamiento del algoritmo.

Los resultados obtenidos para el algoritmo correspondiente a la figura 4.14 implementado sobre uno de los videos de la base de datos, se pueden observar en la tabla 4.4. La primer columna corresponde al nombre asignado a cada video cortado, correspondiente a cada parada. La segunda corresponde a la cantidad de personas que subieron (UP), bajaron ($DOWN$) y el nivel de ocupación ($Ocup.$) contado manualmente a partir de los videos, con el objetivo de tener una base con qué comparar el comportamiento de los algoritmos. La siguiente columna corresponde a los resultados obtenidos por el algoritmo, donde el 30 representa la cantidad de frames durante los cuales se realiza trackeo, el cual era uno de los parámetros que se tenía para modificar. La siguiente columna es análoga, solo que cambiando el parámetro antes mencionado a 15. En las últimas filas se realizó la correlación de Pearson y RMSE entre las columnas de **True** y las columnas correspondientes de **Obj Det (30)** y de **Obj Det (15)**, con el objetivo de tener una medida cuantitativa del funcionamiento del algoritmo.

	True			Obj Det (30)			Obj Det (15)		
	UP	**DOWN**	**Ocup.**	**UP**	**DOWN**	**Ocup.**	**UP**	**DOWN**	**Ocup.**
Man_1_1	4	0	6	4	0	6	3	0	5
Man_1_2	2	0	8	0	0	6	1	0	6
Man_1_3	2	0	10	0	0	6	1	0	7
Man_1_4	4	0	14	1	0	7	3	1	9
Man_1_5	0	2	12	0	0	7	0	0	9
Man_1_6	0	3	9	0	4	3	0	3	6
Man_1_7	0	3	6	1	2	2	0	1	5
Man_1_8	0	2	4	0	1	1	0	0	5
Man_1_9	0	2	2	0	0	1	0	1	4
Man_1_10	3	0	5	2	0	3	3	0	7
Man_1_11	2	0	7	1	0	4	0	0	7
Man_1_12	1	1	7	2	1	5	0	1	6
Pearson	-	-	-	0,63	0,73	0,81	0,91	0,60	0,87
RMSE	-	-	-	1,32	0,96	3,80	0,87	1,08	2,38

Tabla 4.4: Resultados obtenidos con el algoritmo desarrollado

En la tabla 4.5 se pueden observar los resultados obtenidos con el algoritmo modificado, correspondiente al diagrama de bloques de la figura 4.16 y el mismo video utilizado para obtener los resultados de la tabla 4.4. Se puede observar que tanto la correlación de Pearson como el RMSE, en promedio dio mejor en los resultados obtenidos con el algoritmo modificado. Es por esto que el algoritmo utilizado de ahora en adelante es éste.

	True			Obj Det Mod. (30)			Obj Det Mod. (15)		
	UP	**DOWN**	**Ocup.**	**UP**	**DOWN**	**Ocup.**	**UP**	**DOWN**	**Ocup.**
Man_1_1	4	0	6	3	0	5	2	0	4
Man_1_2	2	0	8	0	0	5	0	0	4
Man_1_3	2	0	10	1	0	6	1	0	5
Man_1_4	4	0	14	2	0	8	2	0	7
Man_1_5	0	2	12	0	0	8	0	2	5
Man_1_6	0	3	9	0	2	6	0	3	2
Man_1_7	0	3	6	0	2	4	0	2	0
Man_1_8	0	2	4	0	2	2	0	1	-1
Man_1_9	0	2	2	0	2	0	0	2	-3
Man_1_10	3	0	5	3	0	3	3	0	0
Man_1_11	2	0	7	2	0	5	2	0	2
Man_1_12	1	1	7	2	0	7	2	0	4
Pearson	-	-	-	0,82	0,84	0,91	0,77	0,93	0,89
RMSE	-	-	-	0,96	0,76	2,99	1,08	0,50	5,30

Tabla 4.5: Resultados obtenidos con el algoritmo modificado.

Con el objetivo de cuantificar el desempeño de este algoritmo se generó la tabla 4.6, donde se puede observar la correlación de Pearson, el RMSE y el porcentaje de ocupación obtenidos para la ocupación para cada video de la base de datos de la segunda etapa.

	Obj Det Mod. (30)			Obj Det Mod. (15)		
	Pearson	**RMSE**	**PO(%)**	**Pearson**	**RMSE**	**PO(%)**
Man_1	0,91	2,99	62	0,89	5,30	16
Man_2	0,58	1,04	99	0,37	2,56	66
Man_3	0,26	5,97	193	0,73	5,15	183
Man_4	0,52	2,84	135	0,79	1,91	118
Man_5	0,95	1,17	100	0,91	1,57	88
Man_6	-0,19	3,61	130	0,32	2,11	89
Tar_1	0,63	4,20	161	0,96	2,77	143
Tar_2	0,41	6,27	240	0,78	1,64	115
Tar_3	0,71	3,15	156	0,92	1,26	117
Tar_4	0,65	4,36	172	0,97	1,00	92
Tar_5	0,99	1,53	125	0,98	1,11	136
Tar_6	0,93	2,21	160	0,93	3,00	182

Tabla 4.6: Resultados obtenidos para la ocupación con el algoritmo modificado sobre la base de datos generada.

Los resultados de la tabla 4.6 se realizaron para dos valores de cantidad de frames entre detecciones, sin embargo en la sección 5.2 se analizarán los resultados y se elegirá el parámetro con el que se obtenga el mejor desempeño.

4.3. Wifi

Tomando como punto de partida el relevamiento realizado para la base de datos se procedió a trabajar estos datos para lograr el objetivo final de estimar la cantidad de personas en el ómnibus. Basándose en los artículos estudiados en la sección 2.1.2.1 se procedió a utilizar algunos de estos conceptos de filtrado y tratamiento de las capturas de *Probe Request*. Básicamente se utilizaron dos modelos distintos: el continuo y el discreto. Se estudian en profundidad los modelos buscando evaluar el rendimiento de los mismos a la hora de detectar dispositivos. Luego se busca evaluar la posibilidad de estimar a partir de estos modelos la ocupación de personas en los ómnibus que es el problema de interés.

4.3.1. Modelo continuo

Este modelo continuo es la forma más intuitiva de realizar la estimación de ocupación, buscando realizar la detección y conteo de dispositivos a medida que se van capturando los paquetes. La idea principal es relevar los datos de los paquetes que se capturan realizando una tabla que contenga la dirección MAC y el tiempo del último paquete capturado asociado a esa dirección de origen. La lógica de este modelo se explica con el diagrama de flujo de la figura 4.19.

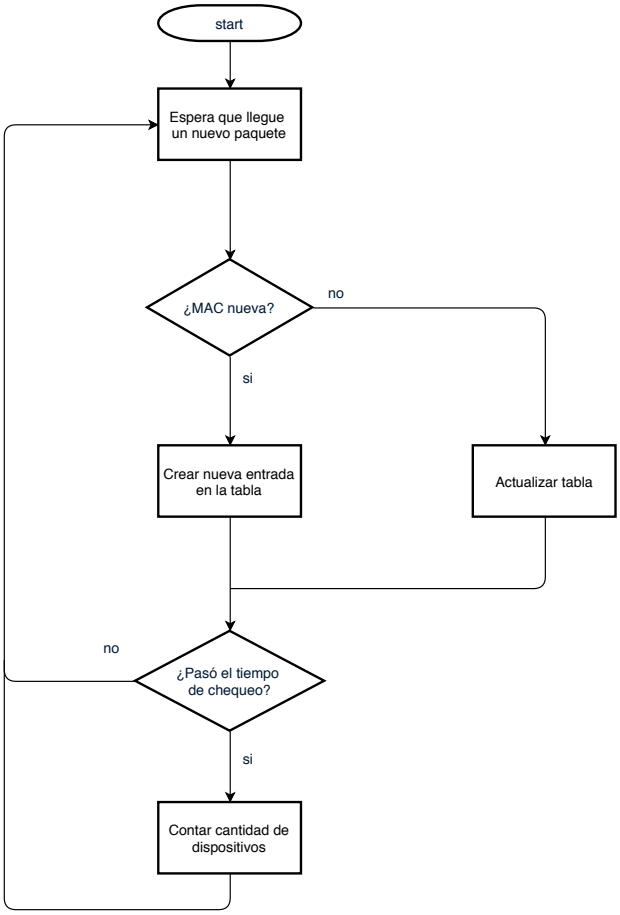


Figura 4.19: Diagrama de flujo del modelo continuo.

La figura anterior explica la lógica a la hora de poder armar la tabla con las direcciones MAC. Se espera por la captura de un paquete y una vez que llega se analiza la dirección MAC de origen. Si esta dirección ya se encuentra en la tabla simplemente se actualiza la tabla con el dato del tiempo de llegada del paquete. Por el contrario, si la dirección MAC de origen del paquete no se encuentra en la tabla entonces se crea una entrada con esa dirección y el tiempo del paquete.

Periódicamente se revisa la tabla buscando aquellas direcciones MAC que excedieron un determinado umbral de tiempo, entendiendo así que ese dispositivo ya no se encuentra dentro del ómnibus. Se remueven estas entradas de la tabla y se procede a contar la cantidad de entradas que quedan. Este dato es tomado como la cantidad de dispositivos que se encuentran dentro del ómnibus en ese momento.

La aplicación de este modelo no tuvo resultados satisfactorios. Utilizando la base de datos obtenida anteriormente, se realizaron pruebas y se observó que la estimación dada por este modelo es aproximadamente seis o siete veces mayor a la ocupación real del ómnibus. Estos resultados reflejan que es necesario un mayor filtrado de los datos, ya que se están contando dispositivos que no se encuentran

dentro del transporte.

Una variante de este modelo es utilizar la información del RSSI que se obtiene en las capturas. A partir de esta información se puede aplicar un umbral de decisión sobre este dato, siendo una referencia de la distancia a la que se encuentra el dispositivo. Es una referencia pero no es un dato de distancia preciso ya que depende, entre otras cosas, de la potencia de transmisión de los distintos dispositivos y la ganancia de las antenas y éstas no son siempre iguales para todos. Antes de agregar o actualizar una entrada de la tabla se revisa el RSSI del paquete y solamente se tiene en cuenta aquellos que contengan un valor por encima del umbral utilizado. Esta lógica se puede ver en el siguiente diagrama.

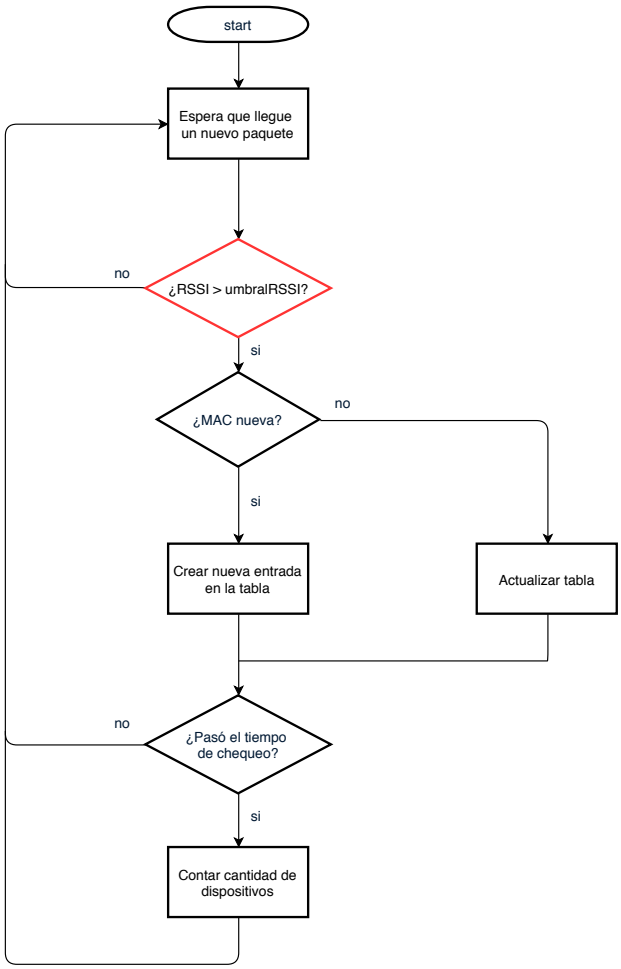


Figura 4.20: Diagrama de flujo del modelo continuo modificado con RSSI.

De la figura 4.20 se desprende que si se captura un paquete cuyo valor de RSSI no supera cierto umbral es descartado, entendiendo que el dispositivo está fuera del ómnibus. Así se puede acotar la cantidad de dispositivos que se cuentan, buscando restringir solamente a los que se encuentran dentro del ómnibus. Se entiende

que esta variante es realmente útil ya que permite obtener un dato más acertado de la ocupación del ómnibus, no teniendo en cuenta aquellos dispositivos que se encuentren fuera del mismo.

Estudiando otras posibilidades de filtrado se evaluó agregar una variante de decisión que considere solo aquellas direcciones MAC que se encuentren al menos una cierta cantidad de veces. Para esto es necesario agregar otra columna en la tabla donde se lleve la cuenta de la cantidad de veces que lleva apareciendo la misma dirección MAC. Con este dato a la hora de realizar el conteo de dispositivos se debe tener en cuenta solamente aquellas entradas que superen cierto umbral respecto a la cantidad de veces que han aparecido. Este dato permite descartar aquellos dispositivos que se encuentran una o dos veces, entendiendo que si se encuentran pocas veces puede ser un indicador de que el dispositivo no se encuentra en el ómnibus, sino que pudo haber sido detectado mientras éste pasaba cerca.

Otra posibilidad es la de utilizar el tiempo en que llegan los paquetes para determinar la duración durante la cual se detectó cierta dirección MAC. De forma similar a la variante anterior busca eliminar detecciones puntuales de dispositivos que pudieran estar cerca del ómnibus. Si solamente se cuentan aquellas que aparecen durante un tiempo determinado se puede ser más preciso en la estimación de la ocupación.

Un aspecto importante discutido en la sección 2.1.2.1 es el tema de la aleatoriedad de las direcciones MAC. Esta alternativa que utilizan algunos fabricantes puede incidir directamente en la cantidad de dispositivos detectados y contados. Esto implica que se deben tomar consideraciones a la hora de contar este tipo de direcciones MAC, ya que sino podría llegar a contarse varias de estas direcciones como distintos dispositivos cuando realmente es uno solo. Para esto es necesario diferenciar en la tabla estas direcciones de las reales y luego aplicar la misma lógica pero utilizando, por ejemplo, umbrales de decisión distintos. Una forma rápida de diferenciar estas direcciones aleatorias es la explicada en la sección 2.1.2.1, observando el bit que indica si la dirección es local o universal. Un ejemplo de la tabla con todos estos datos se puede observar en la tabla 4.7.

Dirección MAC	**Tiempo desde el último reporte (s)**	**Reportes**	**Random**
c4:84:66:f2:1e:c6	4	6	No
c0:cc:f8:ee:10:e0	67	10	No
a8:66:7f:41:f7:a7	12	3	No
da:a1:19:9f:1a:5e	107	1	Si

Tabla 4.7: Ejemplo de tabla utilizada en el modelo continuo.

El filtrado de los paquetes por RSSI se realiza utilizando distintos umbrales. La definición de los mismos se basó en unas pruebas realizadas localmente donde

se relevó el RSSI de los paquetes enviados por algunos celulares conocidos. Conociendo la distancia a la que se encontraba el celular se realizó una asociación entre distancia y RSSI medido. A partir de este relevamiento y conociendo el largo promedio de los ómnibus es que se utilizaron umbrales en el rango de -75dBm a -65dBm.

Trabajando con distintos valores de umbrales se utilizó el algoritmo planteado sobre la base de datos obtenida. El resultado de utilizar umbrales de duración no fue el mejor, ya que se obtuvo valores de ocupación de hasta 50 personas cuando el máximo fue de 18. Observando los resultados se encuentra que este valor se compone casi exclusivamente de direcciones MAC aleatorias. Sucede que este tipo de direcciones se mantienen durante varios minutos haciendo que el algoritmo las considere en la cuenta. Una explicación de este resultado es que los dispositivos que utilizan esta aleatoriedad también utilizan distintas direcciones en la búsqueda de redes y las mantienen a lo largo del tiempo.

Utilizando el algoritmo con la variante del umbral de cantidad de paquetes se obtuvo mejores resultados. Aquí la elección de los umbrales se realizó de forma empírica observando que el promedio de paquetes capturados en cada transmisión fue de 2 en todos los archivos de la base de datos. Es por esto que se utilizaron umbrales por encima de este valor y a su vez distintos para las direcciones reales y las aleatorias. La tabla 4.8 muestra algunos resultados representativos de lo obtenido a partir de este algoritmo[3].

	Man_1		**Man_2**		**Man_3**		**Tar_1**		**Tar_2**		**Tar_3**	
Minuto	**Est**	**Real**	**Est**	**Real**	**Est**	**Real**	**Est**	**Real**	**Est**	**Real**	**Est**	**Real**
1	2	2	2	5	9	11	7	10	3	3	1	3
2	3	7	7	3	10	9	18	8	11	7	5	2
3	4	7	4	7	5	8	9	6	6	4	6	3
4	6	12	0	6	6	7	6	5	8	2	6	7
5	5	10	2	5	8	4	3	3	7	3	6	7
6	6	8	3	5	3	6	5	6	11	2	9	8
7	3	3	2	4	5	7	6	6	8	3	8	5
8	5	2	4	7	9	7	4	5	6	3	8	3
9	6	4	6	5	10	6	4	5	5	2	5	4
10	3	6	5	5	7	3	3	4	5	3	3	3
P.O. (%)	62,75		81,29		119,11		91,25		228,45		141,89	
RMSE	3,5		2,6		2,8		3,0		3,8		2,4	
Pearson	0,87		-0,22		0,21		0,56		0,50		0,51	

Tabla 4.8: Resultados del modelo continuo.

Para cada una de las capturas se observa la estimación de la ocupación en cada minuto y la ocupación real en los mismos minutos. Se obtiene el promedio de los porcentajes de ocupación en cada minuto así como el coeficiente de correlación de

[3]Los resultados completos se pueden observar en el Apéndice C

Pearson para cada captura y el valor de RMSE.

Se desprende de los resultados de la tabla anterior que, si bien son más acertados que los de la variante evaluada anteriormente, no son precisos. Algunos de los datos relevantes a la hora de evaluar el rendimiento del modelo son el porcentaje de ocupación y el coeficiente de Pearson. Si bien en algunas capturas se observa un comportamiento que se podría considerar como el deseado, en otras se observan diferencias que dejan en evidencia que el modelo no logra ser consistente. No se observan porcentajes similares entre capturas; por el contrario se observan valores de porcentajes muy distantes entre algunas. Otro dato que evidencia esto es el coeficiente de Pearson que tampoco logra ser aceptable ni consistente entre las diferentes capturas. En algunas se observa una correlación positiva considerada aceptable mientras que en otras capturas el coeficiente no da ningún indicio de que la estimación esté correlacionada con la realidad.

4.3.2. Modelo discreto

El modelo discreto busca dividir las capturas en intervalos cortos, por ejemplo de dos minutos. A partir de estas capturas cortas se busca contar todos los dispositivos que estuvieron dentro del ómnibus en esos minutos. Esto quiere decir que si una persona estuvo en el ómnibus en algún momento de ese intervalo, aunque haya bajado, se debería encontrar y contar.

Este modelo no da como resultado la ocupación en un momento determinado, sino que cada cierta cantidad de minutos busca tener el resultado de la cantidad de personas que estuvieron en el ómnibus durante ese intervalo. Lo que se intenta es analizar si esta situación, con el intervalo adecuado, se acerca al valor de la ocupación. Esta idea surge de pensar que si los intervalos fueran instantáneos el resultado sería el de la ocupación en cada momento. Al realizar el análisis con intervalos más grandes entonces el resultado del algoritmo difiere de lo que se busca pero gana en precisión ya que tiene más tiempo para detectar los dispositivos. Hay un compromiso entre estos aspectos y se evalúan distintas variantes de forma de encontrar la mejor opción.

A diferencia del modelo continuo en este modelo se trabaja sobre archivos de capturas realizadas durante un intervalo determinado, no se trabaja con las capturas que van llegando en tiempo real, siendo entonces una estimación en semi tiempo real (conocido por su nombre en inglés *Near Real Time*). Una primera aproximación de la ocupación del ómnibus es tomar y contar todas las direcciones MAC distintas que aparecen en el archivo. Esto introduce errores como encontrar dispositivos que se encuentran fuera del ómnibus o contar varios dispositivos erróneamente debido a la aleatoriedad de direcciones MAC discutida en la sección 2.1.2.1. Estos errores se buscan mitigar de forma similar a lo explicado en el modelo continuo. El diagrama de flujo de la figura 4.21 muestra la lógica de este modelo.

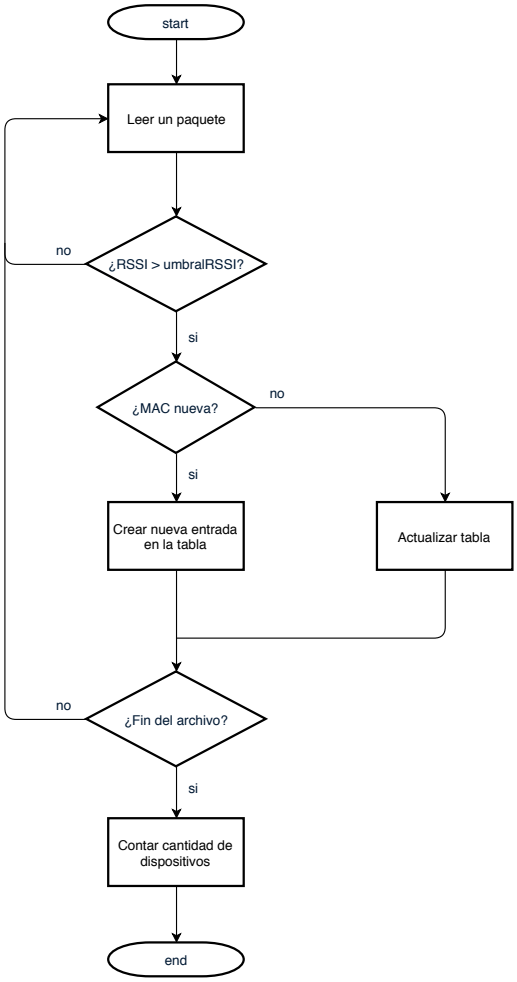


Figura 4.21: Diagrama de flujo del modelo discreto.

Las acciones de crear una nueva entrada en la tabla y de actualizar la tabla son idénticas a las del modelo continuo. La mayor diferencia surge a la hora de contar la cantidad de dispositivos, ya que en este caso no se borran entradas de la tabla porque se desea contar todos los dispositivos que estuvieron en algún momento del intervalo estudiado. Luego al igual que en el modelo anterior se utilizan los umbrales de RSSI y duración con la diferencia que en este modelo el umbral de duración solamente se utiliza para las MAC aleatorias. Esto se debe a que realizando un análisis sobre capturas de Probe Request se observó que los dispositivos envían estos paquetes en períodos que pueden ir desde 15 segundos hasta 3 minutos. Si se utilizara este umbral en las direcciones reales se podría estar obviando algunos dispositivos por el simple hecho de que utilizan un período mayor al tiempo de captura. También se diferencian las direcciones aleatorias y se utilizan distintos umbrales.

A partir de esta información y de la aplicación del último modelo descrito es que se realizaron pruebas sobre las capturas obtenidas en la base de datos. Se

utilizaron los datos recabados en la primera etapa para ajustar los parámetros del modelo y observar los cambios en los resultados. A partir de estos resultados se tomaron los siguientes umbrales como base para realizar la validación sobre los datos recabados en la segunda etapa:

- Duración de las capturas: 2 minutos
- Umbrales de RSSI: -75 dBm, -70 dBm y -65 dBm
- Umbral de duración: 5 segundos

Algunos resultados de la aplicación del algoritmo del modelo discreto se observan en la tabla 4.9 [4].

			Estimación			**Porcentaje**		
Número	**Sub**	**Dispositivos**	**-75 dBm**	**-70 dBm**	**-65 dBm**	**-75dBm**	**-70 dBm**	**-65 dBm**
Man_1	1	7	7	7	4	100 %	100 %	57 %
	2	12	10	10	9	83 %	83 %	75 %
	3	12	9	7	7	75 %	58 %	58 %
	4	8	7	7	7	88 %	88 %	88 %
	5	7	4	4	4	57 %	57 %	57 %
Man_2	1	7	5	5	4	71 %	71 %	57 %
	2	7	7	6	5	100 %	86 %	71 %
	3	9	9	7	5	100 %	78 %	56 %
	4	8	7	5	5	88 %	63 %	63 %
	5	9	7	7	4	78 %	78 %	44 %
Tar_1	1	12	14	12	11	117 %	100 %	92 %
	2	8	13	11	11	163 %	138 %	138 %
	3	10	11	7	6	110 %	70 %	60 %
	4	9	9	8	7	100 %	89 %	78 %
	5	7	10	7	6	143 %	100 %	86 %
Tar_2	1	11	12	11	10	109 %	100 %	91 %
	2	7	19	13	9	271 %	186 %	129 %
	3	5	10	8	6	200 %	160 %	120 %
	4	6	9	7	6	150 %	117 %	100 %
	5	6	10	9	8	167 %	150 %	133 %

Tabla 4.9: Resultados del modelo discreto.

Como una primera observación se desprende que los resultados de la aplicación de este modelo son más precisos que los del modelo continuo. Esto se ve reflejado en los porcentajes de ocupación en la estimación. En la siguiente tabla se resumen los resultados de los porcentajes de ocupación, los coeficientes de correlación de Pearson y los valores de RMSE obtenidos para las capturas de la base de datos en la segunda etapa.

[4]Los resultados completos se pueden observar en el Apéndice C

	P.O. (%)			Pearson			RMSE		
Captura	**-75 dBm**	**-70 dBm**	**-65 dBm**	**-75 dBm**	**-70 dBm**	**-65 dBm**	**-75 dBm**	**-70 dBm**	**-65 dBm**
Man_1	81	77	67	0,86	0,68	0,83	2,14	2,79	3,25
Man_2	87	75	58	0,70	0,75	0	1,34	2,09	3,54
Man_3	84	77	72	0,95	0,85	0,73	1,78	2,60	3,25
Man_4	90	78	70	0,46	0,46	0,29	1,94	2,60	3,28
Man_5	111	97	80	0,36	0,52	0,11	2,40	2,14	3,76
Man_6	106	90	80	-0,52	-0,60	-0,24	2,44	1,84	2,32
Tar_1	126	99	91	0,54	0,50	0,39	2,79	1,94	2,48
Tar_2	179	142	115	0,24	0,53	0,83	6,24	3,31	1,41
Tar_3	216	166	105	0,90	0,48	0,91	7,14	4,60	1,26
Tar_4	157	127	86	0,66	0,48	-0,20	4,49	2,40	2,14
Tar_5	165	119	89	0,53	0,61	0,29	4,44	2,14	3,16
Tar_6	136	104	87	0,51	0,72	0,72	3,89	1,61	2,00

Tabla 4.10: Resumen de resultados del modelo discreto.

De la tabla 4.10 se desprenden resultados más consistentes que los obtenidos mediante el modelo continuo. Sin embargo existe una diferencia apreciable entre los valores obtenidos para las capturas realizadas en la tarde respecto de las de la mañana. Los resultados de las capturas de la tarde muestran una sobre estimación en la ocupación en la mayoría de los casos. Esto puede deberse al hecho de que el recorrido fue distinto en la mañana y en la tarde. En la tarde se dieron vueltas a una plaza más pequeña y es posible que los filtros de RSSI de -75 dBm y -70 dBm no hayan sido lo suficientemente restrictivos. De esta forma se pudo haber detectado personas en la plaza de forma constante ya que pudieron nunca haber salido del rango de detección al dar vueltas alrededor de la plaza. La correlación muestra en su mayoría resultados positivos, dando lugar a pensar que el rango de dos minutos para las detecciones puede ser acertado.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 5

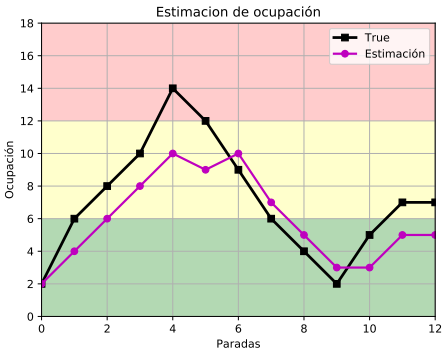
Análisis de resultados

En el presente capítulo se comparan y analizan los resultados obtenidos en el capítulo anterior para las distintas tecnologías evaluadas. Se destacan las fortalezas y debilidades de las distintas tecnologías y métodos implementados a la hora de estimar la ocupación en el medio de transporte público. Más allá del resultado de la estimación se toman en cuenta aspectos como la posibilidad de procesar y enviar los datos en Near Real Time (NRT) y el precio de las distintas tecnologías.

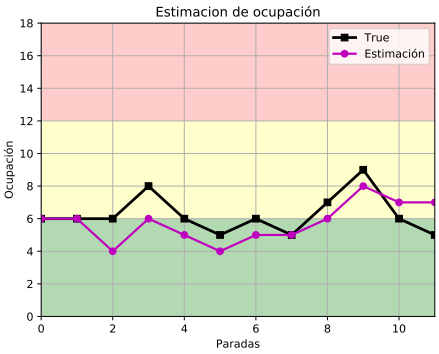
En la sección anterior se presentaron como resultados el coeficiente de correlación de Pearson, el RMSE (Root Mean Square Error) y P.O. (Porcentaje de Ocupación) para evaluar el desempeño de los algoritmos desarrollados. En el Apéndice D se explica cuál es el significado de cada uno.

5.1. Ultrasonido

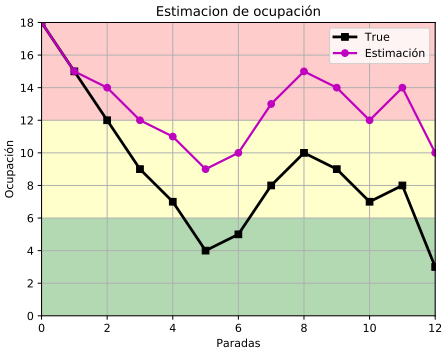
Analizando los datos expuestos en las tablas 4.1 y 4.2 se observa que para las estimaciones de todas las capturas existe error. A continuación se analiza alguno de estos errores. De forma gráfica se puede observar cómo varía la ocupación junto con la estimación para cada parada de todas las capturas de la segunda etapa en las figuras 5.1 y 5.2. En estas figuras se pueden ver estos errores como la separación de la gráfica de Estimación con la gráfica de True.



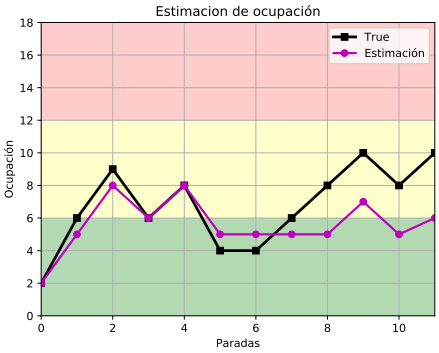
(a) Ocupación Man_1.



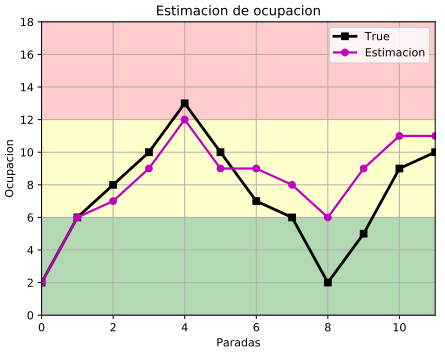
(b) Ocupación Man_2.



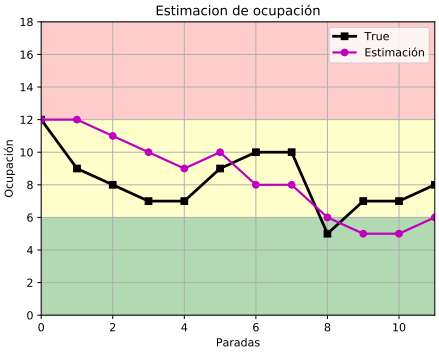
(c) Ocupación Man_3.



(d) Ocupación Man_4.

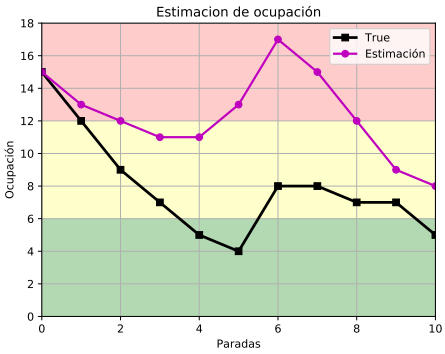


(e) Ocupación Man_5.

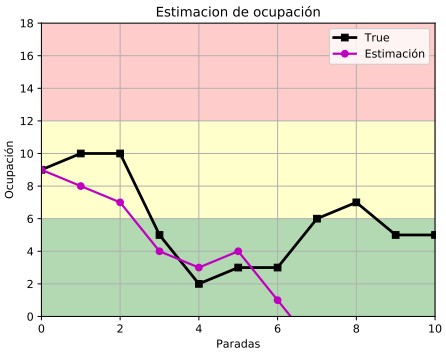


(f) Ocupación Man_6.

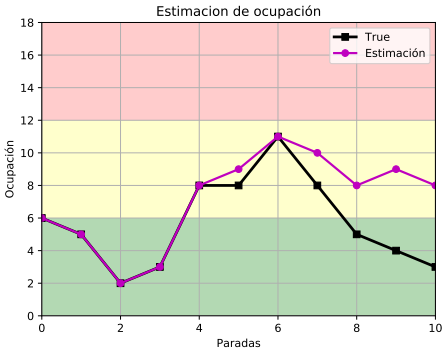
Figura 5.1: Gráficas de ocupación para todas las capturas de la mañana de la segunda etapa.



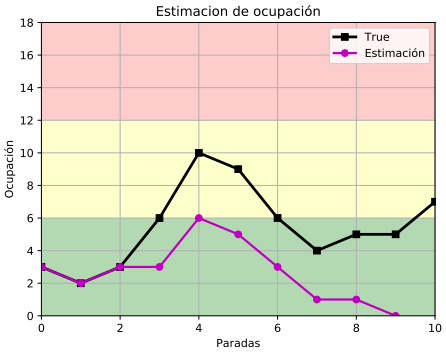
(a) Ocupación Tar_1.



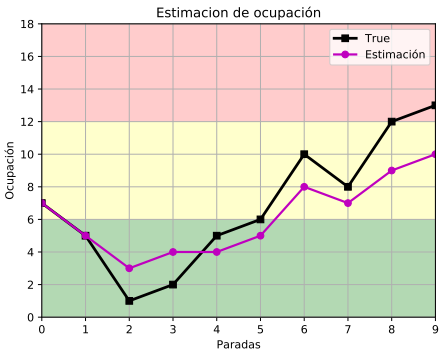
(b) Ocupación Tar_2.



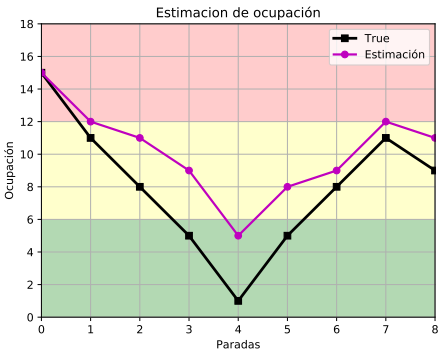
(c) Ocupación Tar_3.



(d) Ocupación Tar_4.



(e) Ocupación Tar_5.



(f) Ocupación Tar_6.

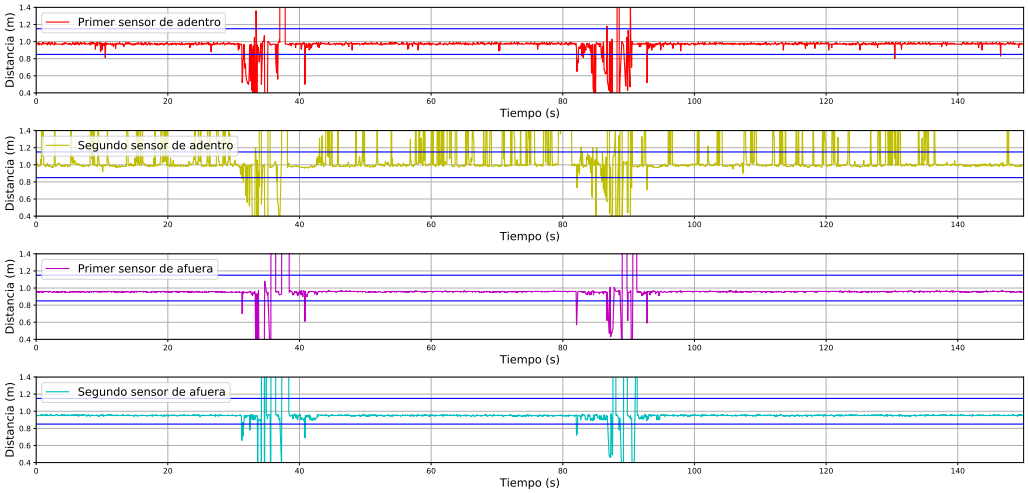
Figura 5.2: Gráficas de ocupación para todas las capturas de la tarde de la segunda etapa.

Las franjas roja, amarilla y verde representan la ocupación en niveles como: lleno, medio lleno y vacío respectivamente.

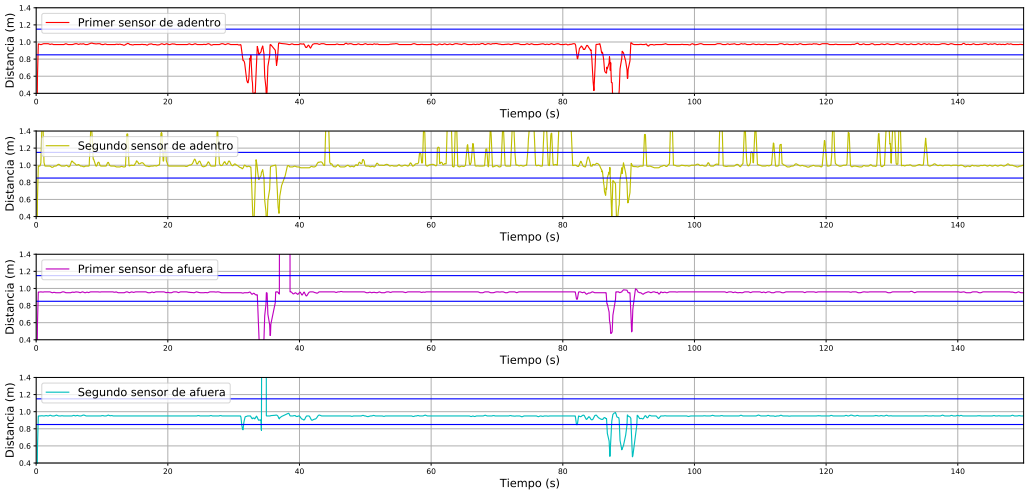
Analizando exhaustivamente las señales de los sensores antes y después de ser filtradas, se observan los distintos motivos de porqué en algunos casos la estimación

difiere con el valor real. A continuación se muestran ejemplos de las principales razones.

Durante las pruebas en Mercedes se probaron varios sensores de ultrasonido ya que algunos de estos fallaban devolviendo medidas erróneas durante largos períodos de tiempo. Así mismo para la captura Tar_1, durante los primeros minutos el segundo sensor de adentro falló, esto se puede ver en la figura 5.3.



(a) Medidas sin filtrar de los sensor en la captura Tar_1.



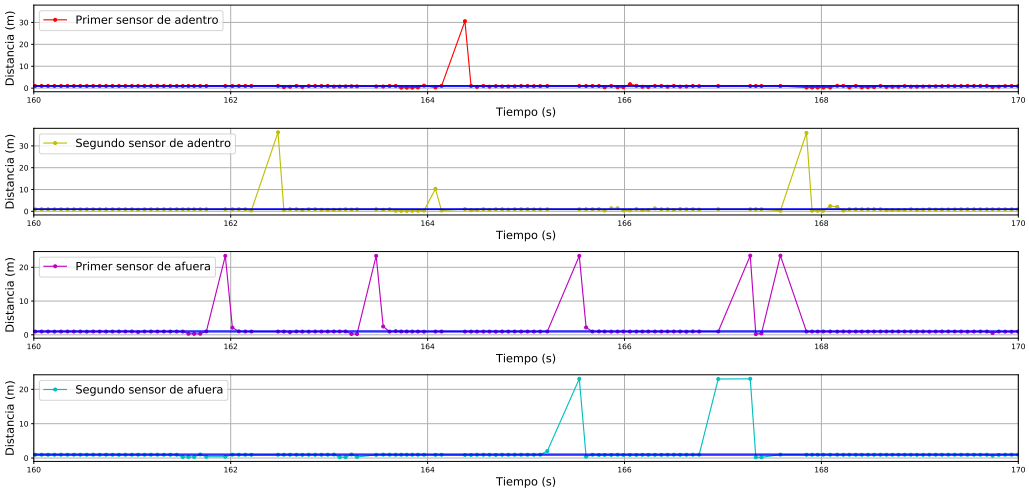
(b) Medidas filtradas.

Figura 5.3: Ejemplo donde uno de los sensores falla.

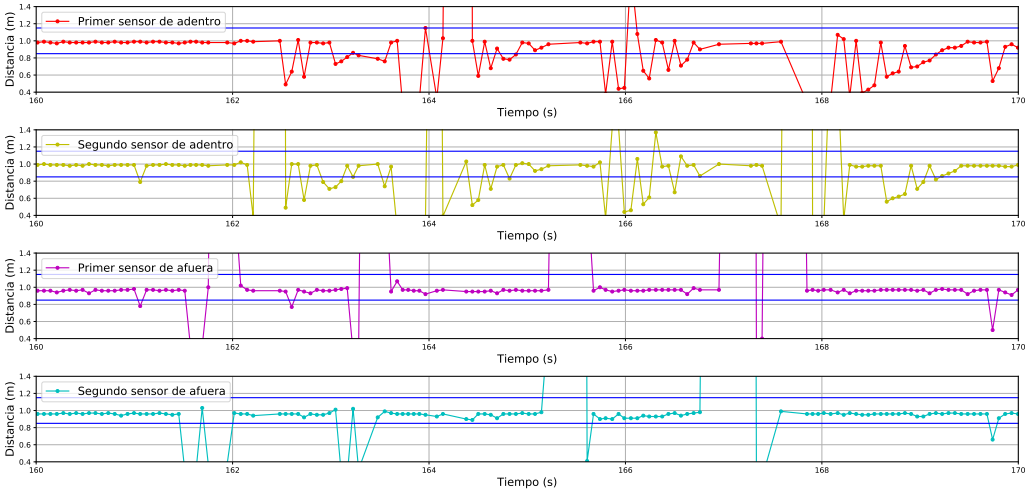
Como se puede ver en la figura 5.3(a) el segundo sensor de adentro sale del rango de la media en lugares donde los otros sensores se mantienen dentro. En la figura 5.3(b) se puede ver como quedan las medidas después de ser filtradas, donde

se aprecia que las medidas erróneas de dicho sensor no son mitigadas por el filtro de mediana. Esto se debe a la gran densidad de estas medidas fuera de rango, lo cual provoca errores en la estimación. Sin embargo este caso solo se vio en esta captura.

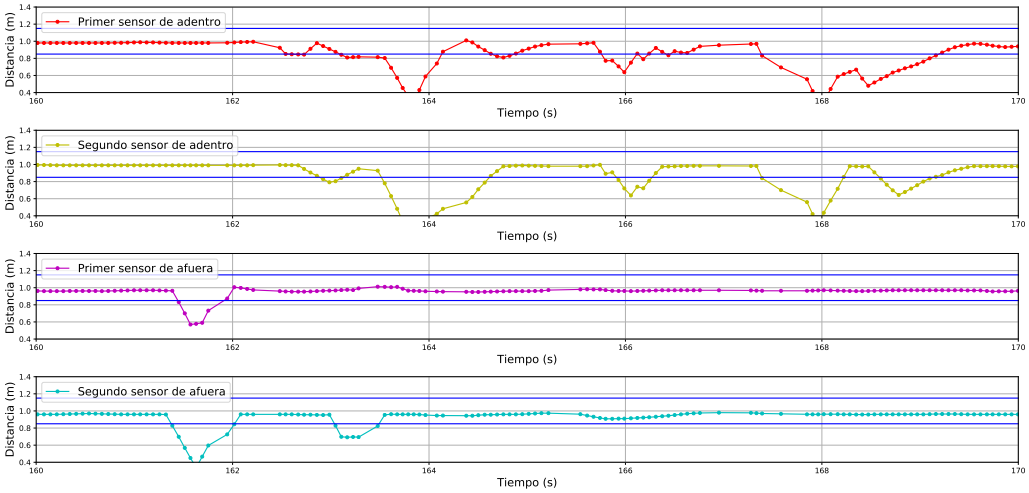
Como contraparte, existen varios casos donde el filtrado suprime medidas relevantes. En la figura 5.4 si se observa primero 5.4(b) y luego 5.4(c) se puede apreciar como los filtros suprimen medidas entre 164s y 170s para los sensores de afuera donde efectivamente cruzaron personas, impidiendo la detección.



(a) Medidas sin filtrar de los sensores en la captura Tar_4.



(b) Acercamiento de las medidas sin filtrar de los sensores en la captura Tar_4.



(c) Medidas filtradas.

Figura 5.4: Ejemplo donde los filtros suprimen medidas relevantes.

Este caso corresponde a 4 subidas en la parada 3 de la captura Tar_4, donde la estimación solo contó 1 persona subiendo. Esta situación es más frecuente y es la principal razón por la cual se tienen errores en la estimación. Si se observa con cuidado la zona donde estas medidas relevantes son mitigadas en las tres subfiguras de la figura 5.4 se puede ver que el tiempo entre muestras es mayor que en otras partes de la gráfica. Esto se debe a que la medida que observó el sensor es de más de 20m, la cual corresponde a cuando una persona pasa y el sensor no recibe el pulso reflejado. Esta baja densidad de muestras hace que el filtro de mediana suprima estas medidas. Se probaron otro tipo de filtros como máximo y mínimo móvil tratando de evitar estos errores pero los resultados en las capturas no tuvieron mejoras significativas.

Los errores de estimación en su gran mayoría se dan cuando la cantidad de personas pasando son más de una, ya sea al subir o al bajar. Por ejemplo siguiendo con el caso que se muestra en la figura 5.4, donde hay dos situaciones donde las medidas de los sensores de afuera fueron suprimidas (figura 5.4(c)), de no haber sido así estos se hubiesen activado antes que los sensores de adentro llevando al sistema al estado **Proceso entrando** y luego cuando se activaran los sensores de adentro se contaría una persona subiendo al ómnibus para cada situación. Sin embargo al no pasar esto, los primeros sensores que el sistema ve activarse son los de afuera, por lo tanto se pasa al estado **Proceso saliendo**. Si inmediatamente después de las personas que entraron y generaron estas medidas entra otra más que active los sensores de afuera, se contará una persona saliendo en esa parada, debido a que se está en el estado **Proceso saliendo** y los sensores de afuera se activaron. Para ver como varían estos errores dependiendo de la cantidad de personas, se hizo una análisis de los eventos por parada desde 0 personas bajando o subiendo hasta cuando pasan 6 personas bajando o subiendo en el mismo sentido.

En la siguiente tabla se muestra el desempeño del método en eventos por paradas, esto quiere decir, independientemente de si las personas bajan o suben, se muestra la cantidad de acierto cuando la cantidad de personas que pasan es 0, 1, 2, 3, 4, 5 ó 6. Por ejemplo si en una parada suben 2 personas y no baja ninguna y la estimación es de 2 personas subiendo y 0 bajando, se tomará como un acierto con error cero para los dos eventos 2 y 0. Por el contrario si la estimación fue de 1 persona subiendo y 1 bajando dado que las medidas de los sensores permitieron tal confusión como se explicó en el ejemplo del párrafo anterior, se dice que hubo un desacierto en los eventos de 2 y 0, con un error de una persona en cada evento.

	Indicadores		
# Personas	**# Eventos**	**P.A. (%)**	**RMSE**
0	76	78,94	1,17
1	44	81,81	1,32
2	63	53,97	1,19
3	39	38,46	1,67
4	22	18,18	1,89
5	3	33,33	1,58
6	3	0	2,71

Tabla 5.1: Indicadores de los resultados obtenidos por evento para cada parada.

La columna **# Personas**, representa la cantidad de personas que pasan por evento, **# Eventos** es la cantidad total de veces que se da el evento, por ejemplo, el evento en que solo una persona baja o sube se repite 44 veces en el total de capturas. **P.A. (%)** es el porcentaje de aciertos, que indica cual es la cantidad de eventos en que la estimación fue acertada sobre la cantidad total y el valor **RMSE** es una aproximación de la diferencia en personas cuando hay un error.

Analizando la tabla 5.1 empezando de abajo, se puede ver que los eventos donde pasan 5 y 6 son muy pocos, 6 en total, y que la estimación no fue buena para ninguno de los dos casos. Para los 3 eventos donde subieron o bajaron 6 personas, la estimación nunca fue de 6 exactamente y en promedio hubo 3 personas de diferencia aproximadamente. Para el caso de 5 personas solo en uno de los eventos la estimación fue acertada, en el resto el error fue de 2 personas aproximadamente. Para 4 y 3 personas donde ya se tienen una mayor cantidad de eventos, 22 y 39 respectivamente, el error es de más del 80 % y 60 % y en cada error la estimación tiene una diferencia de 2 personas siendo más perjudicado el evento de 3. Para los eventos de 2 y 1, el porcentaje de acierto aumenta con respecto a los demás y el error **RMSE** disminuye a 1 persona aproximadamente. Lo cual sigue siendo un problema ya que para el evento de 2 personas el porcentaje de acierto es un poco más del 50 % y en cada desacierto el error es del 50 % de la cantidad. Sin embargo los errores en eventos de poca cantidad de personas a veces se ven afectados por los de mayor cantidad de personas, debido al conteo en sentido erróneo. Es por esto último que el evento 0 no tiene un acierto del 100 %, si en una parada solo bajan o suben personas y se estima una persona en el sentido contrario de forma errónea, el evento 0 se verá afectado. En total se contabilizan 250 eventos de los cuales en 100 el valor de la estimación no coincidió con el valor esperado.

Otra forma de evaluar el método es ver la exactitud estimando la ocupación en tres niveles: lleno (rojo), medio lleno (amarillo), y vacío (verde), como se muestra en las figuras de ocupación 5.1 y 5.2.

	Mañana	**Tarde**
Estimador de Niveles	48,65 %	66,67 %

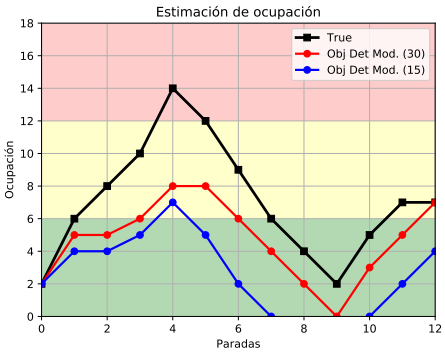
Tabla 5.2: Promedio de estimación por niveles para el sistema de sensores.

De igual forma en la tabla 5.2 se puede ver que los resultados no son buenos, las estimaciones de la mañana solo el 48,65 % de las veces comparten el mismo nivel de ocupación que el True, y en la tarde el 66,67 % de las veces.

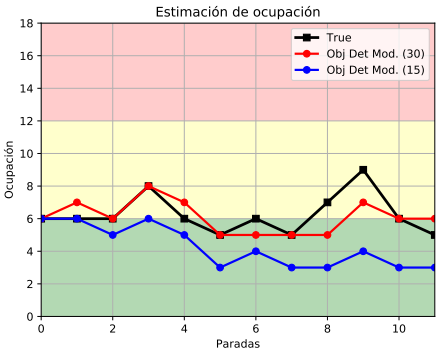
Otro análisis necesario es el tiempo de procesamiento. Utilizando la unidad de procesamiento elegida (Raspberry Pi), en promedio el tiempo de procesamiento de cada captura es de 3,2s. Si se divide este tiempo por la cantidad promedio de paradas que contienen cada captura, el tiempo es de 0,26s, lo cual es una cota superior del tiempo de procesamiento de una sola parada.

5.2. Cámara

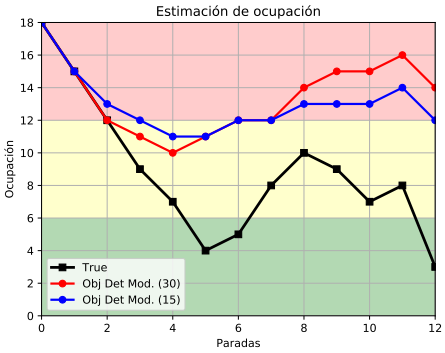
A partir de los resultados expuestos en la tabla 4.5 se procedió a graficar el nivel de ocupación real junto con el estimado con el algoritmo Object Detection Modificado, utilizando el parámetro cantidad de frames entre detecciones en 30 y en 15. Esto se puede observar en la figura 5.5(a). A su vez se graficaron los resultados obtenidos para todos los videos de la segunda etapa, tanto en la mañana (figura 5.5) como en la tarde (figura 5.6).



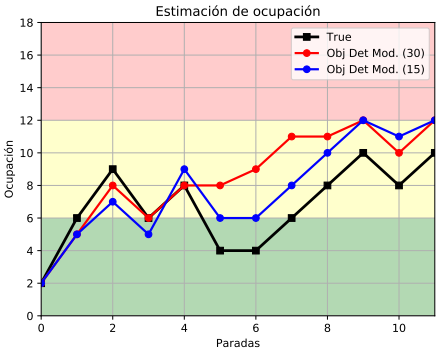
(a) Ocupación Man_1.



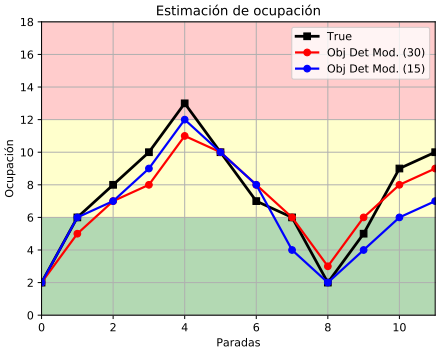
(b) Ocupación Man_2.



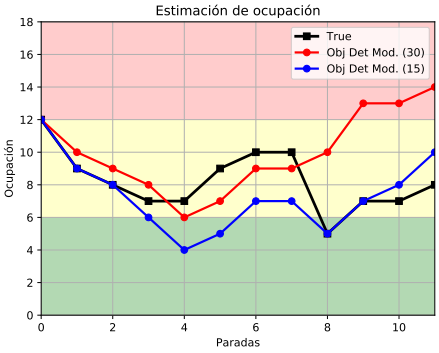
(c) Ocupación Man_3.



(d) Ocupación Man_4.

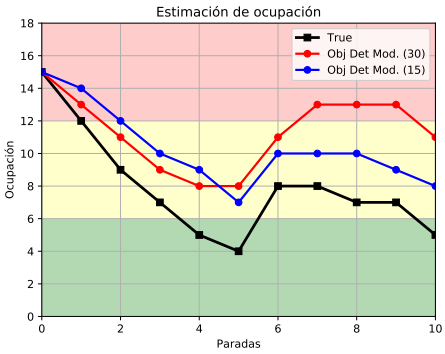


(e) Ocupación Man_5.

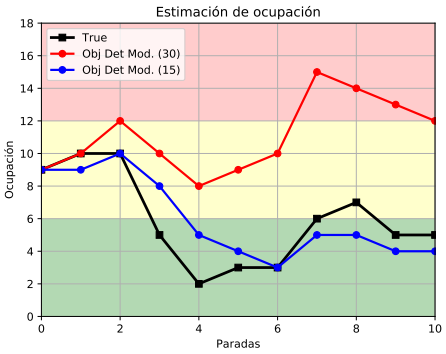


(f) Ocupación Man_6.

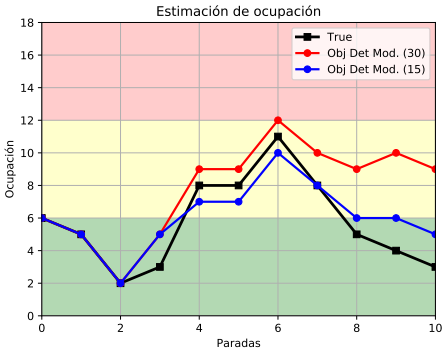
Figura 5.5: Gráficas de ocupación para todos los videos de la mañana de la segunda etapa.



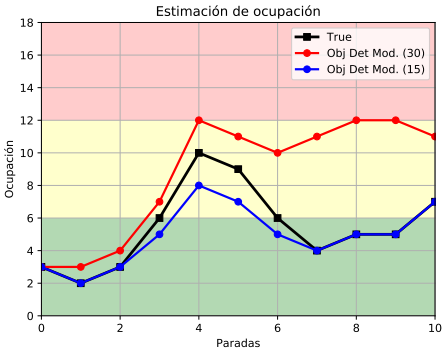
(a) Ocupación Tar_1.



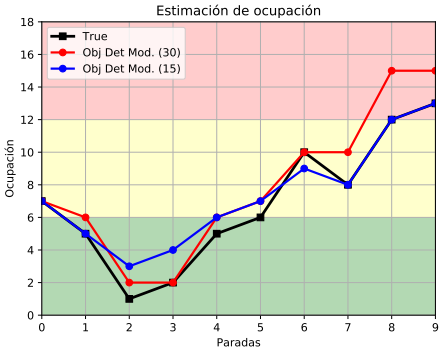
(b) Ocupación Tar_2.



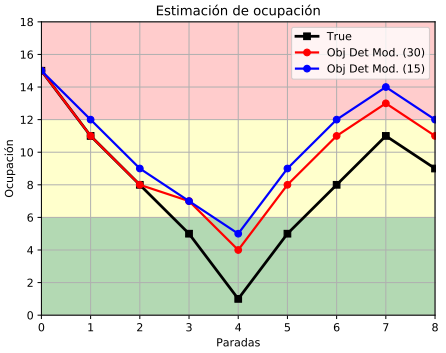
(c) Ocupación Tar_3.



(d) Ocupación Tar_4.



(e) Ocupación Tar_5.



(f) Ocupación Tar_6.

Figura 5.6: Gráficas de ocupación para todos los videos de la tarde de la segunda etapa.

Se puede observar que en algunos casos los valores estimados se alejan del valor real, sin embargo en otros la estimación se acerca bastante al valor real, siendo en algunos casos igual. Con el objetivo de realizar un análisis cuantitativo de los resultados obtenidos fue que se calculó el coeficiente de Pearson, el RMSE y el P.O. para todos los videos de la base de datos generada en la segunda etapa. Estos resultados se pueden observar en la tabla 5.3. Analizando la columna de Pearson se espera que cuanto más cercano a 1 sean los valores, mejores serán los resultados, dado que un Pearson de 1, implica que existe una correlación lineal entre los datos, con lo cual el comportamiento de los datos reales es similar al de los estimados. Se puede observar que para el parámetro 30 (Obj Det Mod. (30)) en un 41.7 % de los casos el Pearson dio mayor a 0.7 sin embargo con el parámetro 15 (Obj Det Mod. (15)) en un 83.3 % de los casos el Pearson dio mayor a 0.7. Esto también se puede observar en la figura 5.7(a). Por esto se puede decir que con el parámetro 15, los resultados estimados tienen un comportamiento más similar a los valores reales que con el parámetro 30. Sin embargo esto no necesariamente quiere decir que el modelo con el parámetro 15 es mejor que con el parámetro 30, para esto es necesario observar los resultados obtenidos para RMSE y P.O.. Dado que como el coeficiente de Pearson es invariante bajo cambios de posición, si el modelo devolviera exactamente los mismos valores que los reales pero con una diferencia constante positiva arbitraria, el Pearson seguiría siendo 1 pero sin embargo para la aplicación deseada el resultado no es bueno porque se está sobrestimando en todo momento. Con el fin de estudiar mejor los dos modelos también se pasa a analizar los valores de RMSE y P.O.

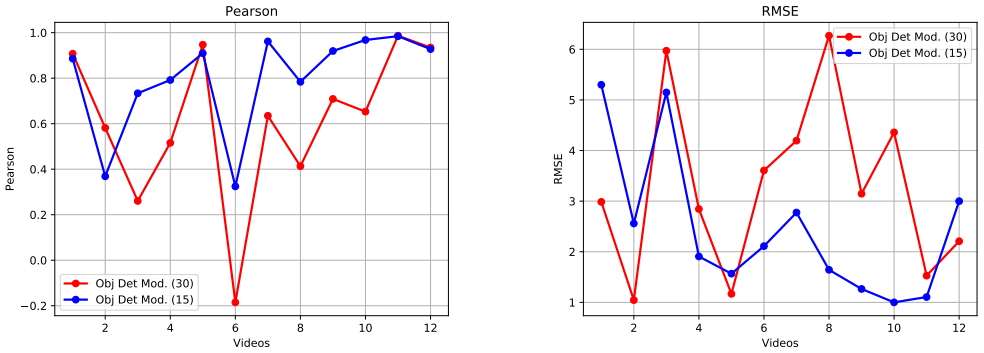
	Obj Det Mod. (30)			**Obj Det Mod. (15)**		
	Pearson	**RMSE**	**PO(%)**	**Pearson**	**RMSE**	**PO(%)**
Man_1	0,91	2,99	62	0,89	5,30	16
Man_2	0,58	1,04	99	0,37	2,56	66
Man_3	0,26	5,97	193	0,73	5,15	183
Man_4	0,52	2,84	135	0,79	1,91	118
Man_5	0,95	1,17	100	0,91	1,57	88
Man_6	-0,19	3,61	130	0,32	2,11	89
Tar_1	0,63	4,20	161	0,96	2,77	143
Tar_2	0,41	6,27	240	0,78	1,64	115
Tar_3	0,71	3,15	156	0,92	1,26	117
Tar_4	0,65	4,36	172	0,97	1,00	92
Tar_5	0,99	1,53	125	0,98	1,11	136
Tar_6	0,93	2,21	160	0,93	3,00	182

Tabla 5.3: Resultados obtenidos para la ocupación con el algoritmo modificado sobre la base de datos generada. (Copia de la tabla 4.6 para comodidad del lector)

El RMSE se puede tomar como una estimación del promedio del error cometido en cada parada. Por lo tanto cuanto menor sea el valor de RMSE mejor serán los

resultados obtenidos, ya que en promedio se cometen errores menores en la estimación de la ocupación. Observando la columna de RMSE de la tabla 4.6, se observa que en un 66.7 % de los casos el RMSE dio menor para el parámetro 15 que para el parámetro 30. Esto se puede ver en la figura 5.7(b). A su vez si se observa la columna P.O.(%) también se observa que para el parámetro 15 los valores dan más cercanos al 100 % si se lo compara con la columna correspondiente al parámetro 30.

Por todo lo analizado antes es que se puede concluir que se obtuvieron mejores resultados con el parámetro 15, por lo tanto este será el modelo elegido.



(a) Gráfica del coeficiente de correlación de Pearson.

(b) Gráfica de RMSE.

Figura 5.7: Gráficas del coeficiente de Pearson y RMSE para todos los videos de la base de datos.

Como uno de los objetivos era lograr realizar un estimador de tres niveles de ocupación, fue por eso que en las gráficas de las figuras 5.5 y 5.6 el fondo contiene 3 colores, representando cada color un nivel de ocupación. Dichos niveles se definieron tomando el máximo de ocupación, de todas las capturas, y tomando tres intervalos iguales. Con el objetivo de evaluar el desempeño para el estimador de tres niveles fue que se realizó el promedio de las veces que el estimador coincidía en el nivel con el valor real y se obtuvieron los resultados de la tabla 5.4. Se puede observar que en la tarde los resultados obtenidos fueron aproximadamente un 15 % mejores.

	Mañana	**Tarde**
Estimador de Niveles	67,57 %	84,13 %

Tabla 5.4: Promedio de estimación por niveles.

A continuación se pasan a analizar los principales errores cometidos durante el procesamiento con el objetivo de reconocer su procedencia y evaluar alguna posibilidad de mitigarlos. Éstos se pueden ubicar en principio en 4 categorías: errores en el seguimiento, errores en la detección, errores por oclusión y errores por pérdida

del tracker.

Errores en el seguimiento

Estos errores provienen de que en el último frame donde se hace trackeo el centro de la persona detectada se encuentra a la izquierda de la línea, pero en el siguiente frame, donde se hace detección, el centro se encuentra luego de la línea. Con el objetivo de mitigar esos errores fue que se realizó lo explicado en la sección 4.2.3, lo cual consistía en fijarse si el centro del último frame del tracker se encontraba cerca del centro del rectángulo de detección siguiente. Sin embargo esa lógica no mitigó todos los errores sino que algunos se siguieron dando como se puede observar en la figura 5.8. En la figura 5.8(b) se puede observar en rojo la zona donde se considera que si el centro anterior cae dentro, entonces es la misma persona. Por un pequeño margen el centro de la figura 5.8(a) no cayó dentro de esa zona por lo tanto en la detección de la figura 5.8(b) las personas detectadas se las consideró como nuevas personas, por lo tanto no se contó la subida. Para mitigar esto se podría aumentar el radio de la zona roja pero tampoco puede ser excesivamente grande dado que se podrían confundir personas.

Para mitigar este tipo de errores se podrían extraer características de los píxeles de la persona trackeada durante el último frame de trackeo, y luego en el frame de detección fijarse en las personas detectadas, si alguna comparte la características extraídas entonces debe ser la misma persona. Esta solución si bien puede ser más precisa también lentificaría el algoritmo dado que para extraer características y compararlas es necesario mayor poder de cómputo.

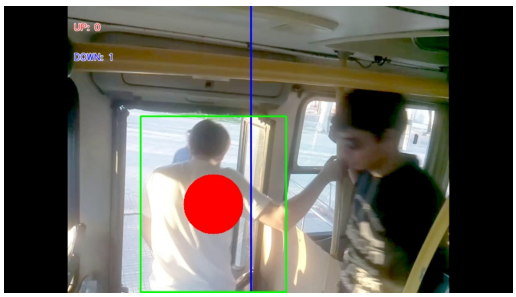
(a) Último frame del tracker. (b) Primer frame de detección.

Figura 5.8: Visualización de errores provenientes del seguimiento.

Errores en la detección

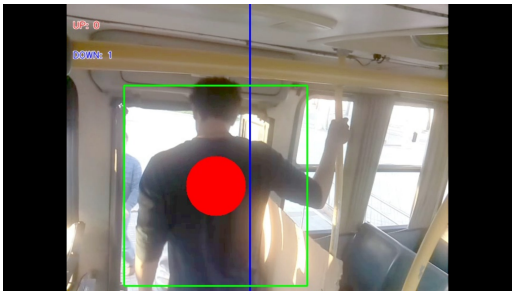
Los errores en la detección provienen de que en el frame de detección el algoritmo detecta menos personas de las que en realidad hay en la imagen, esto se puede observar en la figura 5.9. Donde en la figura 5.9(a) y en la 5.9(b) se pueden observar dos personas, sin embargo el algoritmo solo detecta una. Luego en la figura 5.9(c) finalmente se detecta la persona pero ya es tarde debido a que la persona ya pasó la línea, por lo tanto no se la contó.

Esto se podría mitigar cambiando el modelo utilizado por la red neuronal por uno más preciso de forma de tener más seguridad a la hora de realizar la detección. Sin embargo no cualquier modelo serviría ya que también es necesario que el algoritmo no requiera mucho tiempo de procesamiento, dado que es necesario que funcione en NRT, y generalmente cuanto mejor es la precisión del modelo mayor es el tiempo de procesamiento que necesita.



(a) Fallo en la detección.

(b) Fallo en la detección.



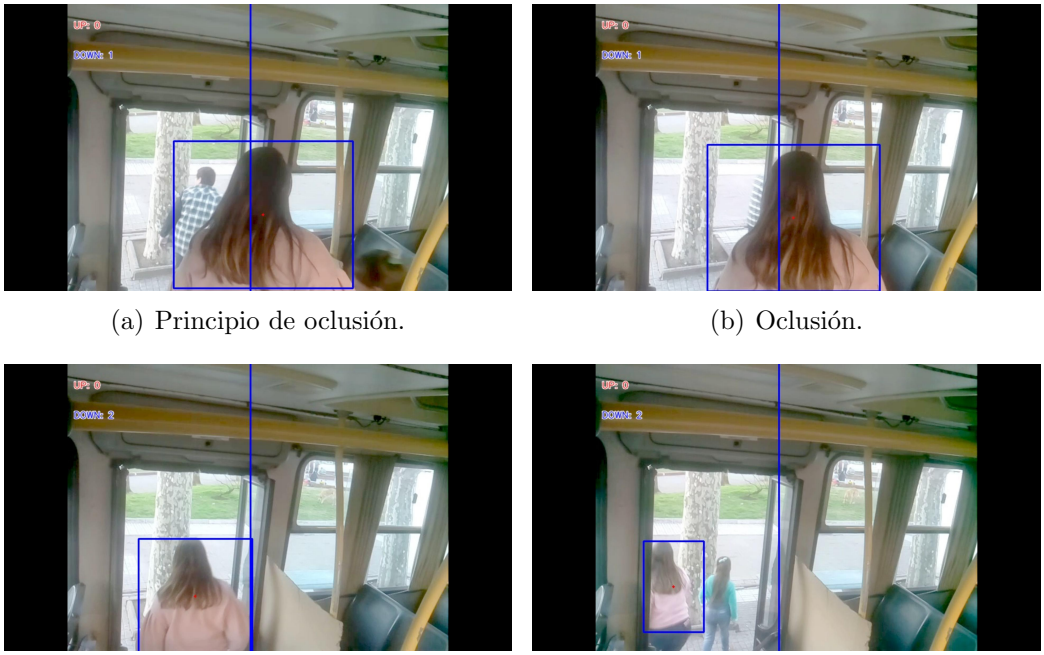
(c) Realización de la detección luego de pasar la linea.

Figura 5.9: Visualización de errores provenientes de la detección.

Errores por oclusión

Los errores por oclusión provienen de que una persona pase delante de otra y de esta forma la cámara no las pueda captar a ambas. Un ejemplo de esto se puede observar en la figura 5.10, donde en la figura 5.10(a) se puede observar una persona detectada y la cabeza de otra persona, sin embargo en las figuras 5.10(b) y 5.10(c) solo se puede observar la persona detectada. Recién en la figura 5.10(d) se puede observar que pasaron dos personas pero ya es tarde porque ambas ya pasaron la línea.

Mitigar este tipo de errores es muy difícil, ya que provienen del comportamiento de los pasajeros, lo que sí se podría hacer es colocar la cámara de forma tal que esta situación se pueda dar la menor cantidad posible de veces.



(a) Principio de oclusión. (b) Oclusión.

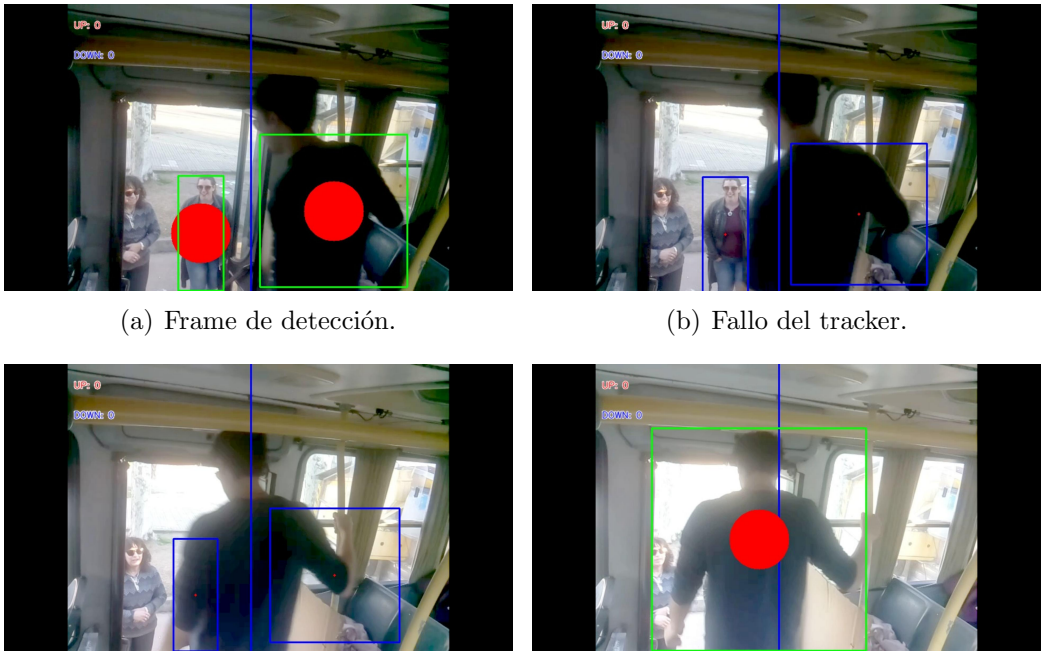
(c) Oclusión. (d) Se observa que pasaron dos personas.

Figura 5.10: Visualización de errores provenientes de oclusión.

Errores por pérdida del tracker

Los errores de esta categoría provienen del incorrecto funcionamiento del tracker. Un ejemplo de ésto se puede observar en la figura 5.11. Como se puede observar en la figura 5.11(a) se realiza la detección de la persona sin embargo en los frames siguientes (figuras 5.11(b) y 5.11(c)) la persona se mueve pero el tracker no la sigue. Luego en la figura 5.11(d) se realiza detección nuevamente pero la persona ya pasó la línea por lo tanto no se la contó.

Una forma de mitigar este tipo de errores podría ser cambiar el tracker por otro que sea más preciso, pero también hay que realizar un compromiso entre precisión y tiempo de procesamiento, dado que es probable que un tracker más preciso tenga un tiempo de procesamiento mayor.



(a) Frame de detección. (b) Fallo del tracker.

(c) Fallo del tracker. (d) Detección luego de pasar la línea.

Figura 5.11: Visualización de errores provenientes de fallos del tracker.

Uno de los objetivos es que el algoritmo funcione en tiempo real o al menos en Near Real Time, es por esto que la forma en la que se procede a contar es: durante cada parada se graba video y luego que termina se pasa el video al algoritmo y cuando éste termine se mandan los datos de ocupación obtenidos. Es por esto que es necesario que el tiempo de procesamiento sea lo menor posible. Con el fin de estudiar esto se procedió a realizar un análisis del tiempo de procesamiento en función del hardware utilizado. Inicialmente se pretendía que el algoritmo funcione sobre una Raspberry Pi, pero para realizar Object Detection mediante redes neuronales es necesario un hardware superior. Por esto los hardwares utilizados son:

Tipo de disopsitivo	**Modelo**	**Procesador**	**RAM**
Laptop	Lenovo LEGION Y520	Intel Core i7-7700HQ@2.8GHz	16GB
MiniPC	Lenovo ThinkCentre	Intel Core i3-7100T@3.4GHz	4GB

Con el fin de comparar los tiempos se realizó la gráfica de la figura 5.12 donde se puede observar la duración del video, el tiempo de procesamiento con la Laptop y el tiempo de procesamiento con la MiniPC. Se puede observar que el tiempo de procesamiento para la Laptop es menor a la duración del video en la gran mayoría de los casos y el de la MiniPC es igual o inferior a la duración del video. Por lo tanto el tiempo de la MiniPC es mayor al de la Laptop. Por esto es que cuanto mejor sea el hardware, menor será el tiempo de procesamiento, pero también más caro es el equipo. Por esto existe un compromiso entre velocidad de procesamiento y costo.

Sin embargo los tiempos obtenidos con la MiniPC se consideran suficientemente buenos como para evaluar una implementación con este hardware.

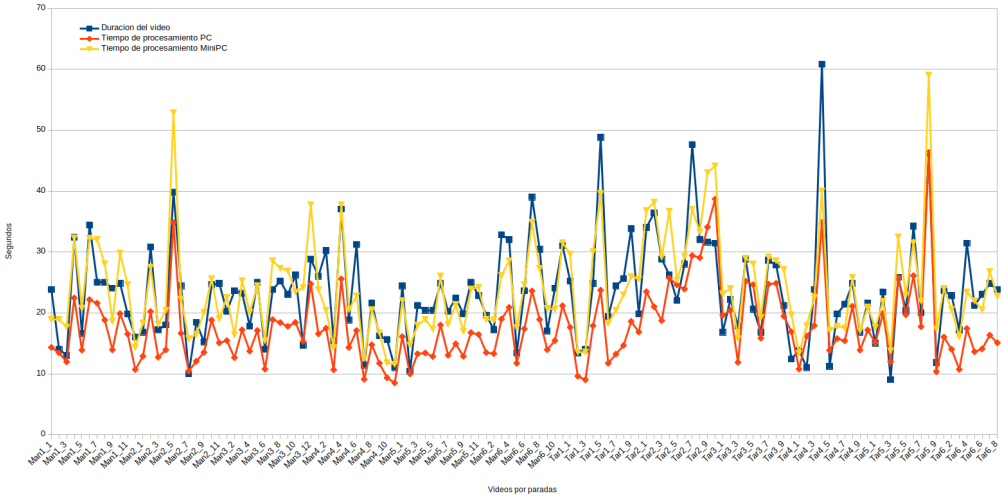


Figura 5.12: Comparación de tiempos.

5.3. Wifi

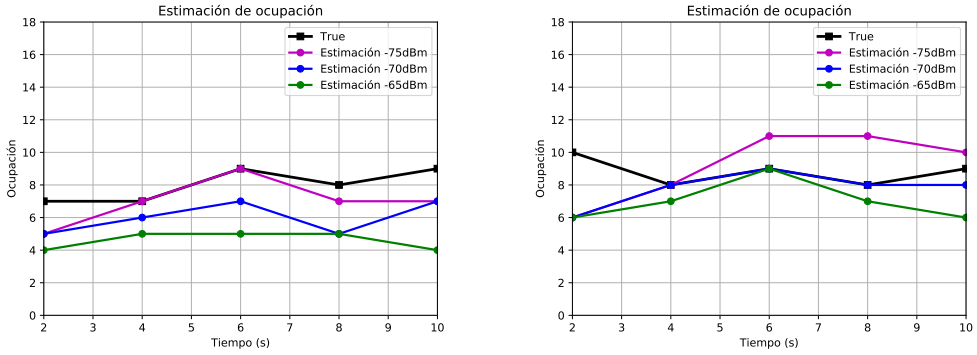
A partir de los resultados expuestos en la sección 4.3 se observó que la aplicación del modelo discreto tuvo mejores resultados que el modelo continuo. Es por esto que se analizan los resultados más en profundidad para este modelo.

Uno de los parámetros más relevantes a la hora de determinar el modelo es el umbral de RSSI que limita el conteo de direcciones MAC por distancia. Se analizan las variantes de umbrales de RSSI y su impacto en la estimación de la ocupación exponiendo los resultados obtenidos para estas variantes. Los primeros resultados se pueden observar en la tabla 5.5 donde se encuentra para cada captura de la base de datos el promedio de porcentaje de ocupación, el coeficiente de correlación de Pearson y el valor de RMSE. Como primera observación el coeficiente de Pearson indica que en la mayoría de las capturas existe una correlación positiva entre las estimaciones y la ocupación real. Estos valores muestran una menor correlación para el umbral de RSSI de $-65dBm$, pudiendo deberse a una restricción muy grande a la hora de filtrar los paquetes. Otro aspecto que refleja en general un buen comportamiento aunque también algunas discrepancias son los valores de RMSE. Teniendo en general valores cercanos a 2 hace pensar que la estimación es acertada. El umbral de RSSI de $-75dBm$ es el que refleja valores más altos y por lo tanto más alejados de los datos reales, mientras que los umbrales de $-70dBm$ y $-65dBm$ arrojan resultados similares entre sí.

	P.O. (%)			Pearson			RMSE		
Captura	**-75 dBm**	**-70 dBm**	**-65 dBm**	**-75 dBm**	**-70 dBm**	**-65 dBm**	**-75 dBm**	**-70 dBm**	**-65 dBm**
Man_1	81	77	67	0,86	0,68	0,83	2,14	2,79	3,25
Man_2	87	75	58	0,70	0,75	0	1,34	2,09	3,54
Man_3	84	77	72	0,95	0,85	0,73	1,78	2,60	3,25
Man_4	90	78	70	0,46	0,46	0,29	1,94	2,60	3,28
Man_5	111	97	80	0,36	0,52	0,11	2,40	2,14	3,76
Man_6	106	90	80	-0,52	-0,60	-0,24	2,44	1,84	2,32
Tar_1	126	99	91	0,54	0,50	0,39	2,79	1,94	2,48
Tar_2	179	142	115	0,24	0,53	0,83	6,24	3,31	1,41
Tar_3	216	166	105	0,90	0,48	0,91	7,14	4,60	1,26
Tar_4	157	127	86	0,66	0,48	-0,20	4,49	2,40	2,14
Tar_5	165	119	89	0,53	0,61	0,29	4,44	2,14	3,16
Tar_6	136	104	87	0,51	0,72	0,72	3,89	1,61	2,00

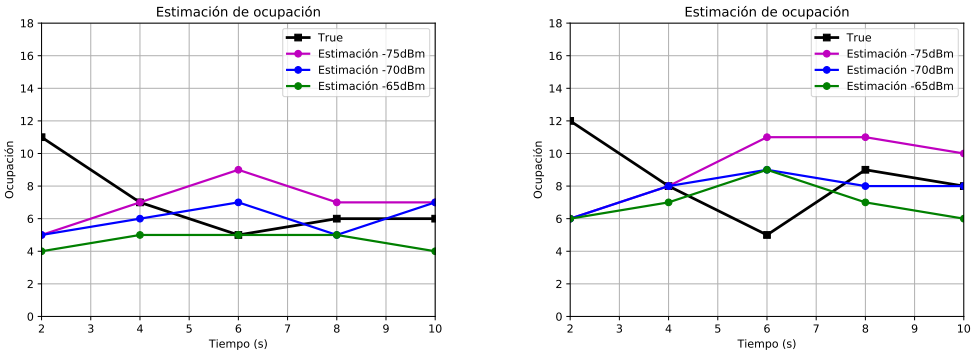
Tabla 5.5: Resumen de resultados del modelo discreto. (Copia de la tabla 4.10 para comodidad del lector)

En cuanto al valor de promedio de porcentaje de ocupación se observa que existe una diferencia apreciable entre los resultados obtenidos en la etapa de la mañana y los de la tarde. Esto puede deberse al cambio de recorrido efectuado entre las dos etapas. Como se puede apreciar en la figura 3.8 el recorrido de la tarde se realizó alrededor de una plaza pudiendo afectar al resultado las personas que se encontraban en la misma. La diferencia con la situación de la mañana es que no se encontraban muchas personas en la plaza y por lo tanto se pudo evitar este comportamiento. Cabe destacar que en una situación normal de recorrido de ómnibus no se tendría esta dificultad ya que el ómnibus no da vueltas sobre un mismo lugar. Se podría detectar algún dispositivo cercano pero con los distintos filtros y umbrales es de esperar que estas situaciones no deseadas se mitiguen. Estas diferencias de estimación entre las capturas de la mañana y la tarde se pueden observar gráficamente comparando las figuras 5.13 y 5.14.



(a) Ocupación Man_2. (b) Ocupación Man_6.

Figura 5.13: Gráficas de ocupación para algunas capturas de la mañana.



(a) Ocupación Tar_2. (b) Ocupación Tar_6.

Figura 5.14: Gráficas de ocupación para algunas capturas de la tarde.

Como era de esperar a partir de los resultados de promedios de porcentaje de ocupación de la tabla 4.10, en las figuras anteriores se observa gráficamente que las estimaciones de las capturas de la tarde se posicionan por encima de la ocupación real. Por otro lado las gráficas de la figura 5.13 muestran una mejor estimación respecto de las de la tarde. A partir de los resultados analizados, más allá de que algunos umbrales de RSSI tienen indicadores similares, en general se observa que el que tiene mejor comportamiento es el umbral de $-70dBm$.

Una vez definido este umbral de RSSI se busca analizar la estimación de ocupación de personas en el ómnibus, contrario a lo que se venía analizando que era la estimación de dispositivos. Para realizar este análisis se utilizan dos estimadores: uno donde se asume que la cantidad de dispositivos detectados es igual a la cantidad de personas y el otro donde se toma en cuenta el factor de que las personas tengan un dispositivo pero con el WiFi apagado. A este primer estimador le llamaremos Estimador 1 y mantiene los mismos resultados de estimación que se venían trabajando. La segunda variante, Estimador 2, busca contemplar las personas que tienen un dispositivo pero no tienen activado el WiFi, las cuales no son tenidas en cuenta por el Estimador 1. Para esto se utilizó como referencia una encuesta realizada por la empresa $\#Numeral$ donde se relevan datos de personas, la cantidad de dispositivos por persona y el dato que indica si tienen el WiFi prendido o no. A partir de una muestra de 675 personas se observó que aproximadamente el 77 % tienen el WiFi prendido. Es por esto que las estimaciones del Estimador 2 toman los datos del Estimador 1 y los divide por 0,77.

La primera aproximación es la de comparar los resultados de los estimadores en las capturas de 2 minutos contra la cantidad de personas que estuvieron dentro del ómnibus en ese tiempo. Estos resultados se resumen en la siguiente tabla.

	P.O. (%)		Pearson		RMSE	
Captura	**Estimador 1**	**Estimador 2**	**Estimador 1**	**Estimador 2**	**Estimador 1**	**Estimador 2**
Man_1	67	86	0,68	0,68	4,15	2,68
Man_2	64	81	0,56	0,47	3,55	2,24
Man_3	60	74	0,81	0,81	5,57	3,71
Man_4	62	80	0,49	0,43	4,49	3,07
Man_5	85	108	0,73	0,69	2,49	1,95
Man_6	61	79	0,14	0,27	5,20	3,16
Tar_1	81	104	0,38	0,43	3,41	2,79
Tar_2	98	126	0,35	0,34	3,03	3,85
Tar_3	122	157	0,43	0,39	3,44	5,83
Tar_4	109	144	0,59	0,63	1,90	3,92
Tar_5	98	128	0,57	0,55	2,72	3,03
Tar_6	86	110	0,33	0,32	3,32	3,26

Tabla 5.6: Resultados de los estimadores en relación a la cantidad de personas cada 2 minutos

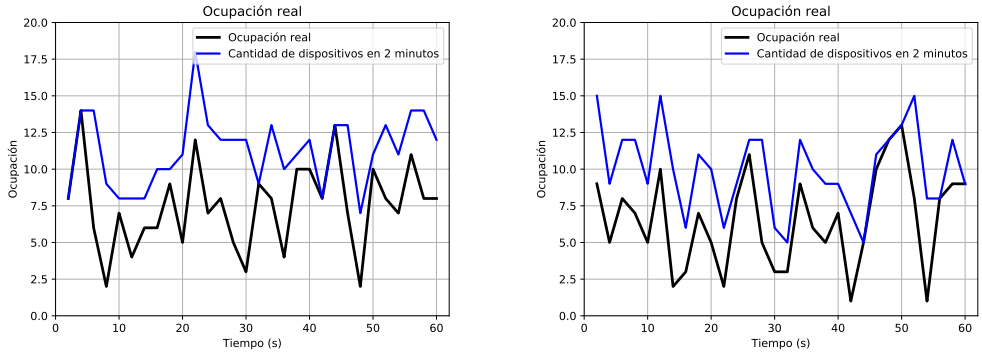
En la tabla 5.6 se observan valores de correlación positiva para todas las capturas en ambos estimadores. A su vez se puede observar que la diferencia entre ambos estimadores respecto de este indicador es casi nula, lo cual era de esperar dado que se diferencian en una constante multiplicativa y el coeficiente de correlación de Pearson es invariante bajo cambios de escala. Los valores para la mañana de promedio de porcentaje de ocupación y RMSE indican que el Estimador 2 logra mejores resultados que el Estimador 1. En la tarde es diferente y esto puede deberse a la misma razón que fue explicada al analizar los resultados de la tabla 4.10. Es por esto que se entiende que el Estimador 2 logra determinar la cantidad de personas que estuvieron dentro del ómnibus cada 2 minutos de mejor forma que el Estimador 1.

Luego de realizado este análisis se procedió a analizar el resultado de utilizar el Estimador 2 para estimar la ocupación de personas en el ómnibus. Como el modelo discreto tiene la limitante de trabajar sobre capturas de dos minutos para estimar la ocupación los resultados se comparan cada ese período de tiempo. Estos resultados se pueden observar en la tabla 5.7.

Captura	P.O. (%)	Pearson	RMSE
Man_1	175	0,57	3,58
Man_2	136	-0,26	2,90
Man_3	157	0,94	3,19
Man_4	122	-0,11	3,44
Man_5	181	0,95	3,44
Man_6	120	0,70	1,90
Tar_1	179	0,29	5,59
Tar_2	338	-0,19	8,28
Tar_3	288	0,86	7,87
Tar_4	225	0,45	6,78
Tar_5	336	0,31	5,27
Tar_6	359	-0,31	6,16

Tabla 5.7: Resultados del Estimador 2 en relación a la ocupación real cada dos minutos.

Se observan resultados irregulares a partir de los datos de la tabla anterior. Por un lado el coeficiente de Pearson arroja resultados de correlación positiva en la mayoría de las capturas. En las capturas de la mañana existe una sobrestimación que se refleja en el promedio de porcentaje de ocupación así como en los valores de RMSE altos. Esto se puede explicar a partir de la gráfica de la figura 5.15.



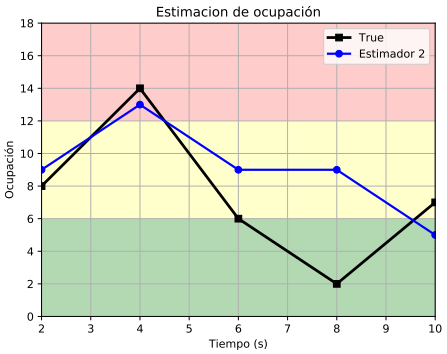
(a) Mañana. (b) Tarde.

Figura 5.15: Diferencia entre la ocupación real y la cantidad de personas que pasaron en 2 minutos.

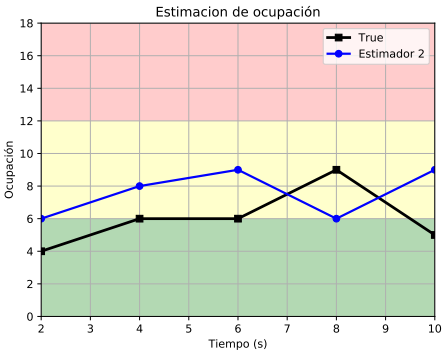
Se observa que existe una diferencia entre los valores de ocupación real muestreados cada dos minutos y la cantidad de personas que estuvieron en el ómnibus cada dos minutos. Esta diferencia es el punto de partida de la implementación del algoritmo, es decir que es un error conocido. Se analizó el resultado de estimar las curvas azules de las figuras 5.15(a) y 5.15(b), y logrando un buen resultado era de esperar tener una diferencia respecto de los valores de ocupación. Si a los

resultados de la estimación se le resta el promedio de la separación entre las curvas de las figuras 5.15(a) y 5.15(b) se obtienen mejores resultados, disminuyendo los valores de RMSE y aproximándose los valores de P.O. a un entorno del 100 %.

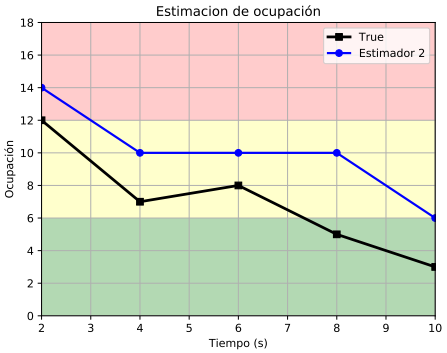
En las figuras 5.16 y 5.17 se observan gráficamente los resultados de la aplicación del algoritmo del modelo discreto con el Estimador 2 para las capturas de la segunda etapa de la base de datos. A su vez, al igual que para las otras dos tecnologías se utiliza un código de colores en las gráficas para permitir visualizar el resultado del modelo sobre la estimación de tres niveles.



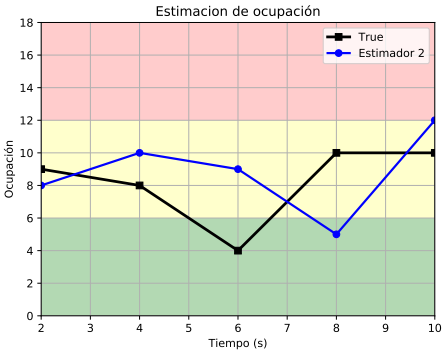
(a) Ocupación Man_1.



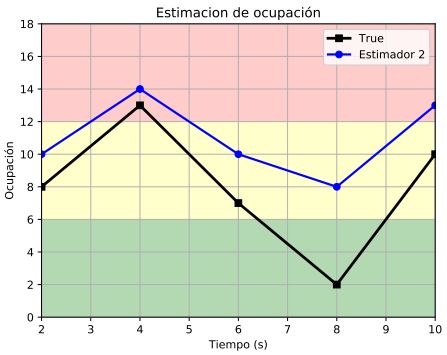
(b) Ocupación Man_2.



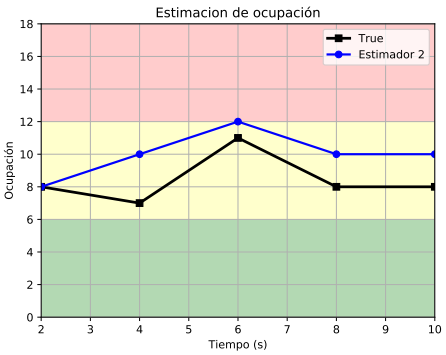
(c) Ocupación Man_3.



(d) Ocupación Man_4.

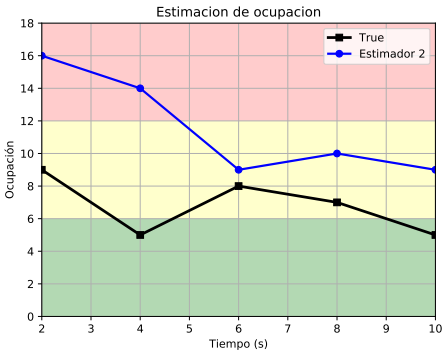


(e) Ocupación Man_5.

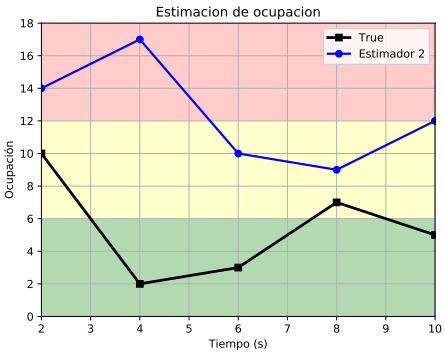


(f) Ocupación Man_6.

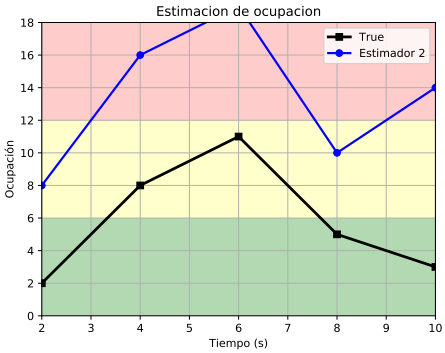
Figura 5.16: Gráficas de ocupación para todas las capturas de la mañana de la base de datos.



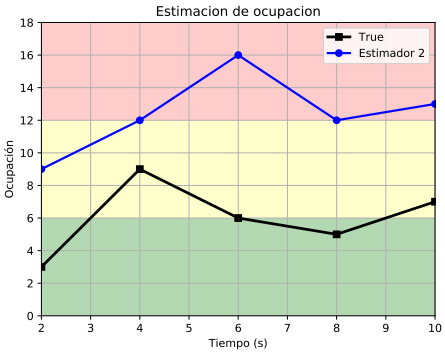
(a) Ocupación Tar_1.



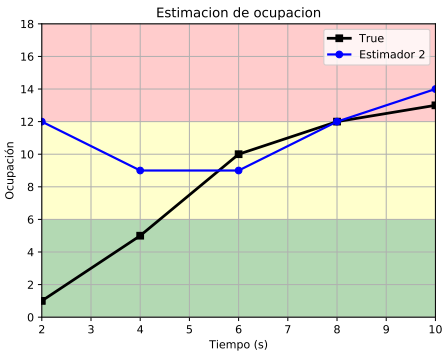
(b) Ocupación Tar_2.



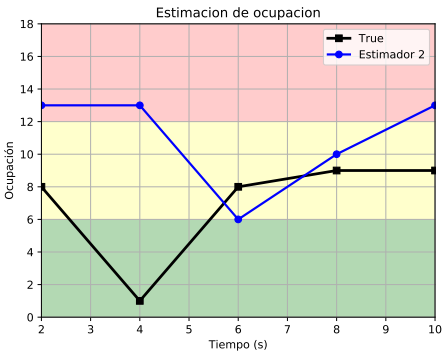
(c) Ocupación Tar_3.



(d) Ocupación Tar_4.



(e) Ocupación Tar_5.



(f) Ocupación Tar_6.

Figura 5.17: Gráficas de ocupación para todas las capturas de la tarde de la base de datos.

En las gráficas de la figura 5.16 y 5.17 se observan las estimaciones analizadas en la tabla 5.7. A partir de estas estimaciones se desprenden los siguientes resultados para el estimador de tres niveles:

	Mañana	**Tarde**
Estimador de Niveles	56,67 %	26,67 %

Tabla 5.8: Promedio de estimación por niveles.

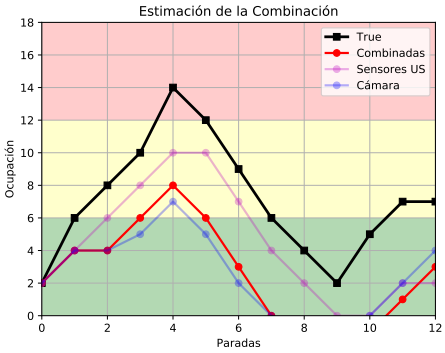
No se logra un resultado preciso y esto puede deberse a varios factores. Por un lado al realizar capturas cortas de dos o tres minutos no se logra detectar todos los dispositivos que se encuentran en el ómnibus ya que la frecuencia con la que se envían los Probe Request no es constante entre los distintos modelos de dispositivos. Estudiando este aspecto se observaron valores de períodos de tiempo que van desde 15 segundos hasta 3 minutos. Es por esto que al realizar capturas de 2 o 3 minutos es difícil lograr la detección de todos los dispositivos que se encuentran abordo. Si se extendiera el tiempo de la captura se podría tener resultados más precisos sobre la cantidad de gente que estuvo abordo en ese tiempo, pero por otro lado la estimación se apartaría del valor real de ocupación ya que es probable que se estuvieran contando dispositivos que ya no se encontraran en el ómnibus. Además al aumentar el tiempo de captura se estaría perdiendo la posibilidad de estimar en NRT. Este enfoque podría tomar mayor sentido en el análisis de ocupación de medios de transporte como puede ser el tren donde el tiempo entre paradas es de varios minutos, pudiendo así capturar y contar durante un mayor período de tiempo, logrando resultados más estables.

Analizando el tiempo de procesamiento que implica el algoritmo del modelo discreto se observó que utilizando una Raspberry Pi se logra procesar las capturas de dos minutos en un promedio de $5,2ms$. Esto entonces no es un problema a la hora de evaluar el requerimiento de estimar en NRT. La mayor diferencia con un estimador en tiempo real se da por la lógica del algoritmo en sí, ya que procesa los últimos 2 minutos y por lo tanto no se logra una resolución menor que esa.

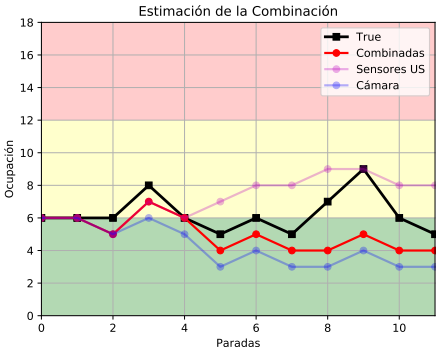
5.4. Combinación de tecnologías

Otra opción para estimar la cantidad de personas que suben y bajan en cada parada es combinar los resultados de las diferentes tecnologías. Analizando los resultados obtenidos para el sistema de sensores de ultrasonido y la cámara, se puede observar que muchas veces los cambios de ocupación entre una parada y la otra en lo estimado son menores que en el True. Esto lleva a pensar en una combinación de estos dos métodos donde en cada parada una vez procesada y obtenido un valor de estimación para cada método, el valor de estimación devuelto será el que tenga una mayor diferencia en ocupación con respecto a la parada anterior, ya sea positiva o negativa. En algunos casos esta diferencia es positiva para un método y negativa para el otro, por lo tanto es necesario decidir que sentido tomar. Si se observa las columnas de Pearson de las tablas 4.2 y 4.6, se puede observar que en general este valor es más cercano a 1 para la estimaciones de Object Detection con 15 frames, esto indica que tiene sentido utilizar este método para determinar si la ocupación debe incrementar o disminuir.

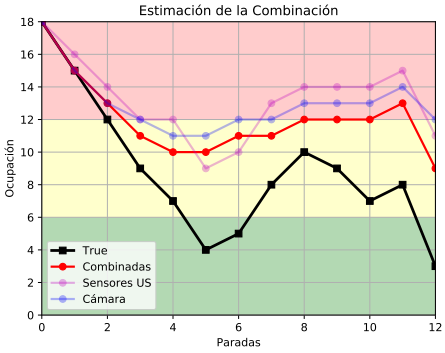
En las figuras 5.18 y 5.19, se puede observar los resultados de la estimación de los dos métodos combinados junto con la estimación de cada método por separado.



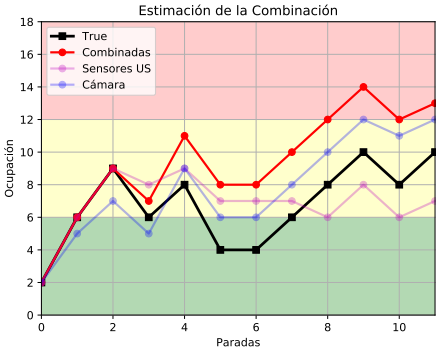
(a) Ocupación Man_1.



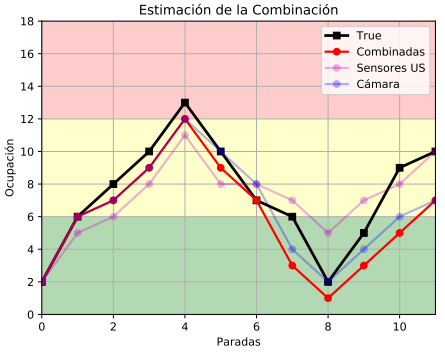
(b) Ocupación Man_2.



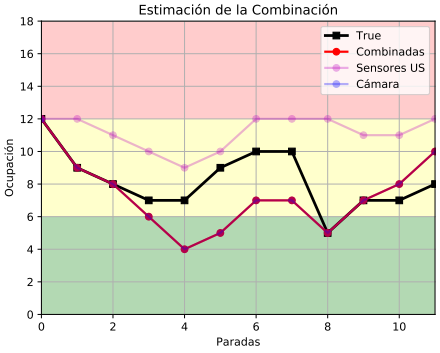
(c) Ocupación Man_3.



(d) Ocupación Man_4.



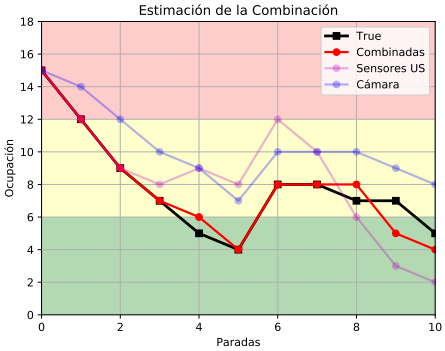
(e) Ocupación Man_5.



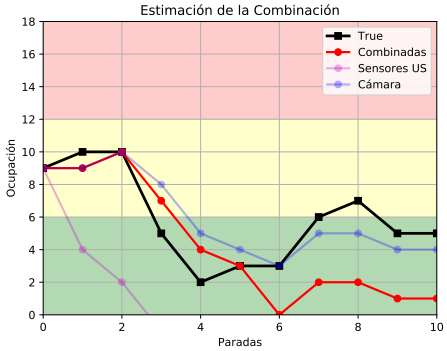
(f) Ocupación Man_6.

Figura 5.18: Gráficas de ocupación de las tecnologías combinadas para las capturas de la mañana.

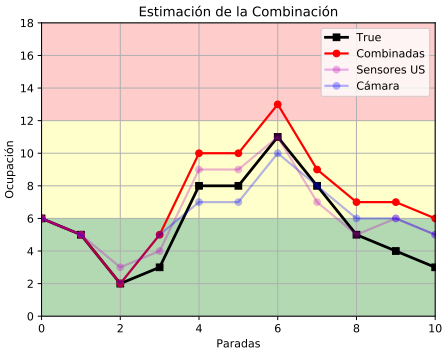
En las figuras 5.18(b) y 5.18(c), se puede observar que la gráfica de la combinación se acerca más al True, sin embargo en el resto no se logra una mejora.



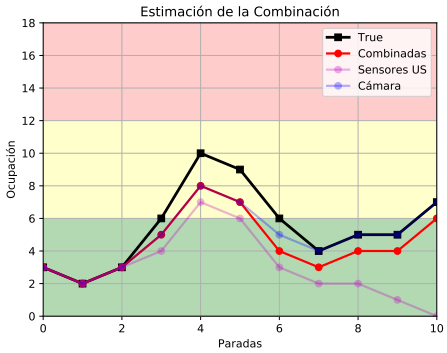
(a) Ocupación Tar_1.



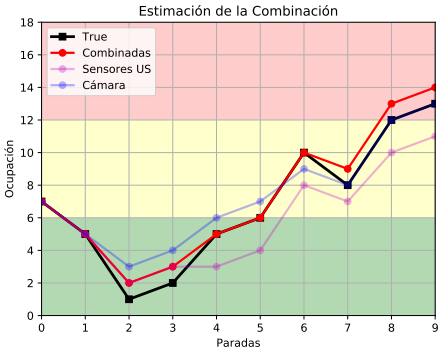
(b) Ocupación Tar_2.



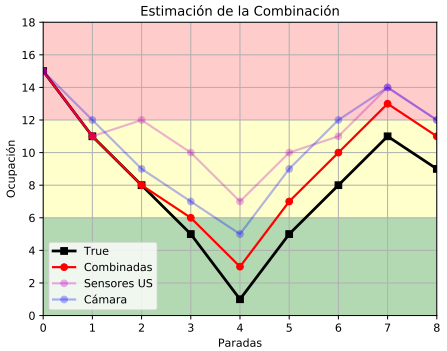
(c) Ocupación Tar_3.



(d) Ocupación Tar_4.



(e) Ocupación Tar_5.



(f) Ocupación Tar_6.

Figura 5.19: Gráficas de ocupación de las tecnologías combinadas para las capturas de la tarde.

En las figuras 5.19(a), 5.19(e) y 5.19(f) se puede observa también que la gráfica de ocupación de las tecnologías combinadas se acerca más a la gráfica del True que

las estimaciones de las tecnologías por separado.

En la tabla 5.9 se exponen los resultados de los indicadores para la combinación de las tecnologías. Si bien se observa una buena correlación entre las estimaciones y la ocupación real mejorando en algunos casos lo obtenido para la cámara, en algunas capturas no se logra mejorar el resultado.

	Mediciones		
	Pearson	**P.O. (%)**	**RMSE**
Man_1	0,85	35,37	4,26
Man_2	0,72	72,50	2,00
Man_3	0,88	173,25	4,81
Man_4	0,84	108,66	1,68
Man_5	0,88	104,41	1,58
Man_6	0,72	95,27	1,55
Tar_1	0,73	146,22	3,71
Tar_2	0,79	80,13	2,39
Tar_3	0,93	117,34	1,12
Tar_4	0,79	81,26	1,88
Tar_5	0,98	131,28	1,10
Tar_6	0,94	171,23	2,72

Tabla 5.9: Indicadores de los resultados obtenidos a partir de la combinación de los resultados del sistema de sensores con la cámara.

En la tabla 5.10 se pueden observar los resultados obtenidos para el estimador de tres niveles.

	Mañana	**Tarde**
Estimador de Niveles	51,35 %	71,42 %

Tabla 5.10: Promedio de estimación por niveles para la combinación de tecnologías.

5.5. Comparación de resultados

A partir de los resultados obtenidos se pasa a realizar una comparación de las tecnologías implementadas. De las tablas 5.2, 5.4, 5.8 y 5.10 se puede observar que los mejores resultados se obtienen con la cámara. En la tabla 5.11 se exponen los resultados obtenidos para las tres tecnologías y la combinación. También se exponen otros datos como el precio, hardware utilizado y tiempo de procesamiento.

	Estimador de Niveles		Hardware Utilizado	Precio	Tiempo de procesamiento promedio por parada
	Mañana	Tarde			
WiFi	56,67 %	26,67 %	Raspberry Pi + Dongle	∼U$S110	0,0032s
Cámara	67,57 %	84,13 %	MiniPC + Cámara	∼U$S600	24,03s
Ultrasonido	48,65 %	66,67 %	Raspberry Pi + Sensores	∼U$S150	0,267s
Combinación	51,35 %	71,42 %	Raspberry Pi + Sensores + MiniPC + Cámara	∼U$S750	24,30s

Tabla 5.11: Comparación de las tecnologías.

Si bien el estimador de niveles dio mejores resultados para la cámara, se puede observar que esta tecnología no solo es la que tiene un costo mayor sino que además es la que requiere un mayor tiempo de procesamiento. Tanto para los sensores de ultrasonido como para la detección por WiFi, si bien estas características son mejores en el sentido de que son más baratas y con pocos requerimientos de procesamiento, los resultados no se acercan a los valores que se esperan para tener una buena estimación. Para el caso de WiFi como ya se dijo anteriormente no es posible obtener una estimación buena cuando el tiempo de captura es muy corto, por lo tanto no es la mejor opción para implementar en un escenario de este estilo. En el caso del sistema de sensores es necesario mejorar las señales que permiten la detección, realizando un mejor pre-procesamiento o cambiando los sensores. El resultado de la combinación de la utilización de la cámara con los sensores de ultrasonido no logra ser mejor que el de la cámara sola, teniendo también la desventaja de tener un precio más elevado.

Capítulo 6

Conclusiones y trabajo a futuro

En este capítulo se exponen las conclusiones del proyecto, incluyendo los aportes principales, se analiza el cumplimiento de los objetivos planteados y se incluye una sección donde se indican los posibles líneas de trabajos a futuro.

6.1. Conclusiones

Se logró probar dos tecnologías capaces de detectar personas junto con una que permita detectar dispositivos en distintos escenarios y evaluar sus ventajas y desventajas. Como las tecnologías utilizadas fueron de bajo costo y de fácil acceso, es que los resultados obtenidos pueden ser reproducidos, dado que en los apéndices se adjuntan los códigos utilizados para la realización del procesamiento y los esquemáticos de los circuitos realizados.

Uno de los aportes realizados durante este trabajo fue la realización de una base de datos la cual contiene videos, series temporales e información de las direcciones MAC detectadas todo esto debidamente etiquetado. Se destaca la posibilidad de haber realizado pruebas con público real en un ómnibus donde se ven manifestadas todas las asperezas de las tecnologías utilizadas. A su vez la simulación se hizo lo más real posible ya que en ningún momento se condicionó a la gente de qué forma subir al ómnibus y cada uno participó con su dispositivo móvil personal, sin ninguna modificación realizada por el equipo. Se contó con un público diverso, contando con hombres, mujeres y niños. La unidad de ómnibus utilizada es una de las unidades que se encuentra en funcionamiento en la ciudad de Mercedes actualmente, lo cual aporta legitimidad a la prueba realizada.

Para la estimación realizada con el sistema de sensores en la mayoría de los casos se logró una buena detección, de igual forma los resultados no son los esperados, debido al incorrecto conteo de personas en una cantidad considerable de casos. El sistema fue pensado para funcionar con componentes de bajo costo pero los utilizados sufrían bastantes variaciones durante el pasaje de las personas. Esto se buscó mitigar realizando un pre-procesamiento pero como contrapartida en al-

gunos casos se perdió información relevante del pasaje de personas. Por otro lado se probó que el tiempo de procesamiento para esta tecnología en una Raspberry Pi logra ser lo suficientemente rápido para implementarse en casi tiempo real (en inglés Near Real Time). Otra de las ventajas de este sistema es que conserva la privacidad de los pasajeros.

Para el conteo de personas mediante el procesamiento de imágenes una de los aspectos a destacar es que se logró realizar la detección y conteo de personas que suben y bajan por la puerta delantera del ómnibus con una sola cámara de video de bajo costo. Reduciendo así el costo del sistema y problemas de instalación, por esto mismo es que solo es necesario el procesamiento de un video. Se logró realizar la detección de personas con un algoritmo que implementa redes neuronales convolucionales. De las tres tecnologías implementadas, con la cámara fue que se obtuvieron los mejores resultados llegando a un 84 % de precisión en la estimación de tres niveles para los videos de la tarde de la base de datos. Es por esto que parecería ser la tecnología mas prometedora, pero tiene como desventaja que es la más cara ya que para las tecnologías de ultrasonido y detección de dispositivos el procesamiento fue hecho en una Raspberry Pi, sin embargo para el algoritmo de detección de personas fue necesario utilizar un hardware superior, por ejemplo una MiniPC, cuyo costo es considerablemente superior. A su vez también es la que más tiempo de procesamiento requiere, lo cual es una limitante ya que uno de los objetivos del proyecto es que el algoritmo funcione en tiempo real o al menos en casi tiempo real (en inglés Near Real Time). Es por esto que hay un compromiso entre precisión, tiempo de procesamiento y costo.

Aunque le detección de dispositivos no fue el método que arrojó los mejores resultados, se puede decir que se logró realizar una estimación de la ocupación utilizando solo los paquetes Probe Request enviados por los dispositivos móviles, sin necesidad de ninguna acción externa por parte del equipo. Es por esto que se considera una tecnología no invasiva. Una de las principales dificultades encontradas fue que los dispositivos más actuales implementan una randomizacion de su dirección MAC con el fin de mejorar la seguridad de los usuarios. No se encontró demasiada información sobre el tema, y muchos de los artículos que implementan tecnologías similares no tienen en cuenta esta variante. Por otra parte se observó que utilizando capturas de dos o tres minutos no se logra detectar todos los dispositivos en el ómnibus, por lo que se entiende que es necesario un intervalo de tiempo mayor. Esto dificulta tener una estimación en tiempo real, ya que hay que realizar un compromiso entre el tiempo durante el que se detectan direcciones MAC y el período en que se reporta el nivel de ocupación.

Se cree que esto es un primer paso para mejorar la calidad del servicio brindado por las empresas de transporte público, ya que tener datos de ocupación junto con información de posición geográfica y tiempo puede ayudar a mejorar la eficiencia del servicio.

6.2. Trabajo a futuro

En esta sección se pasa a describir cuáles son las posibles lineas de trabajo a futuro que se podrían desarrollar a partir de lo estudiado.

Para lograr un sistema que permita adquirir los datos, procesarlos y enviarlos a una plataforma de IoT de una forma eficiente, se cree que es necesario desarrollar un sistema integrador automático donde se realice la adquisición de datos solamente cuando el ómnibus tenga la puerta abierta. Una vez cerrada la puerta los datos adquiridos son procesados por el algoritmo de estimación para luego enviar los resultados obtenidos a la plataforma de IoT como por ejemplo Fiware [1]. Un resumen del funcionamiento del envío de los datos se puede observar en el Apéndice E. Para lograr saber el estado de la puerta solo se necesita un sensor que devuelva una señal cuando la puerta se abra y se cierre, donde esta señal será la que dé paso a la adquisición o al procesamiento y enviado según corresponda. Esto permitiría tener mayor ahorro de energía, mitigar detecciones falsas mientras no hay personas circulando y hacer un procesamiento off line de los datos con los beneficios que esto conlleva.

Como modificaciones a los sistemas de estimación implementados una posibilidad es utilizar sensores en modo barrera para el sistema de sensores. Este modo consiste en instalar dos sensores enfrentados uno a cada lado de la escalera del ómnibus, donde uno esté solo en modo emisor y el otro en modo receptor. Como trabajo a futuro se puede evaluar si los sensores utilizados en este trabajo son capaces de funcionar en este modo o es necesario adquirir otro. De esta forma se podría aumentar la frecuencia de muestreo al doble, ganando en resolución.

El área de reconocimiento de objetos mediante redes neuronales convolucionales, o el aprendizaje profundo en general, es un área que últimamente ha tenido un desarrollo considerable, por lo tanto constantemente se desarrollan nuevos métodos o modelos que resultan ser más precisos, rápidos y/o eficientes. Por lo tanto si se cambia el modelo, utilizado por la red neuronal, por uno más preciso pero cuyo tiempo de procesamiento sea menor o igual al utilizado actualmente los resultados obtenidos probablemente sean mejores. A su vez otra de las limitantes con las que se encontró fue el hardware necesario para que el algoritmo funcione, ya que para realizar detección de personas mediante redes neuronales es necesario un alto poder de procesamiento. Un elemento que podría solucionar tanto el tiempo de procesamiento como el costo moderado del equipo es una placa FPGA (Field-Programmable Gate Array). Actualmente estas placas se están utilizando para implementar algoritmos aprendizaje profundo (en inglés Deep Learning). Las características de las FPGA hacen que el procesamiento de la imagen sea en paralelo, por eso el tiempo de procesamiento es considerablemente menor que si se implementa en una máquina con un microprocesador, ya que este tipo de equipos se caracterizan por realizar el procesamiento de forma secuencial. Las FPGA no

[1] `https://www.fiware.org/`

fueron estudiadas en este trabajo dado que la curva de aprendizaje necesaria para desarrollar algoritmos sobre tipo de equipos es bastante pronunciada, y por lo tanto excedía los objetivos del trabajo.

Apéndice A

Resultados de base de datos

A.1. Base de datos obtenida

A continuación se da detallada la base da datos generada para las tres tecnologías en las dos etapas.

Archivo	**Duración** (min.)	**Campo de Visión**	**Iluminación**	**Momento del día**
CamE1man1.h264	15	Puerta	Sin Ajuste	Mañana
CamE1man2.h264	15	Puerta	Sin Ajuste	Mañana
CamE1man3.h264	6	Puerta	Con Ajuste	Mañana
CamE1tar1.h264	10	Escalera	Con Ajuste	Tarde
CamE1tar2.h264	10	Escalera	Con Ajuste	Tarde
CamE1tar3.h264	10	Escalera	Con Ajuste	Tarde
CamE1tar4.h264	10	Puerta	Con Ajuste	Tarde
CamE2man1.h264	10	Puerta y pasillo	Con Ajuste	Mañana
CamE2man2.h264	10	Puerta y pasillo	Con Ajuste	Mañana
CamE2man3.h264	10	Puerta y pasillo	Con Ajuste	Mañana
CamE2man4.h264	10	Puerta y pasillo	Con Ajuste	Mañana
CamE2man5.h264	10	Puerta y pasillo	Con Ajuste	Mañana
CamE2man6.h264	10	Puerta y pasillo	Con Ajuste	Mañana
CamE2tar1.h264	10	Puerta y pasillo	Con Ajuste	Tarde
CamE2tar2.h264	10	Puerta y pasillo	Con Ajuste	Tarde
CamE2tar3.h264	10	Puerta y pasillo	Con Ajuste	Tarde
CamE2tar4.h264	10	Puerta y pasillo	Con Ajuste	Tarde
CamE2tar5.h264	10	Puerta y pasillo	Con Ajuste	Tarde
CamE2tar6.h264	10	Puerta y pasillo	Con Ajuste	Tarde

Tabla A.1: Base de datos adquirida a partir de la cámara.

Archivo	**Duración** (min.)	**Cantidad de sensores**	**Posición**	**Frecuencia media de muestreo (mps)**	**Distancia entre sensores (cm.)**
UsE1man2.csv	15	2	Techo	8,4	35
UsE1man3.csv	6	2	Techo	8,4	35
UsE1tar2.csv	10	2	Baranda	8,4	35
UsE1tar3.csv	10	2	Baranda	8,4	35
UsE1tar4.csv	10	2	Baranda	8,4	35
UsE2man1.csv	10	4	Escalera	14,7	35
UsE2man2.csv	10	4	Escalera	14,7	35
UsE2man3.csv	10	4	Escalera	14,7	35
UsE2man4.csv	10	4	Escalera	14,7	35
UsE2man5.csv	10	4	Escalera	14,7	35
UsE2man6.csv	10	4	Escalera	14,7	35
UsE2tar1.csv	10	4	Escalera	14,7	35
UsE2tar2.csv	10	4	Escalera	14,7	35
UsE2tar3.csv	10	4	Escalera	14,7	35
UsE2tar4.csv	10	4	Escalera	14,7	35
UsE2tar5.csv	10	4	Escalera	14,7	35
UsE2tar6.csv	10	4	Escalera	14,7	35

Tabla A.2: Base de datos adquirida a partir de los sensores de ultrasonido.

Directorio	**Duración de captura (min.)**	**Duración de archivo (min.)**
MacE1man1	15	3
MacE1man2	15	3
MacE1man3	6	3
MacE1tar1	10	2
MacE1tar2	10	2
MacE1tar3	10	2
MacE1tar4	10	2
MacE2man1	10	2
MacE2man2	10	2
MacE2man3	10	2
MacE2man4	10	2
MacE2man5	10	2
MacE2man6	10	2
MacE2tar1	10	2
MacE2tar2	10	2
MacE2tar3	10	2
MacE2tar4	10	2
MacE2tar5	10	2
MacE2tar6	10	2

Tabla A.3: Base de datos adquirida a partir del Dongle.

Con respecto a la tabla A.2, todos los archivos indican el tiempo relativo para cada medida de los sensores. Para la base de datos especificada en A.3, a diferencia de las dos tablas anteriores se especifica el nombre del directorio y no del archivo, debido a que cada directorio contiene archivos (.txt) de una duración de 2 ó 3 minutos, por lo tanto cada directorio contiene una cantidad de archivos igual a la mitad de la duración de la captura. A su vez en cada archivo se proporciona la dirección MAC capturada, el RSSI y la hora con la que llego el paquete que contenía la respectiva dirección.

A.2. Ground Truth

Con el objetivo de poder comparar los distintos modelos es necesario tener una referencia de la ocupación real. A continuación se detallan los valores de ocupación de personas y dispositivos para todos los archivos de la base de datos generados en la segunda etapa, dado que se entendió que los datos de la primera etapa no eran de buena calidad. Diferenciando el conteo por paradas realizadas por el ómnibus y también cada un minuto.

	Paradas											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Man_1	6	8	10	14	12	9	6	4	2	5	7	7
Man_2	6	6	8	6	5	6	5	7	9	6	5	-
Man_3	15	12	9	7	4	5	8	10	9	7	8	3
Man_4	6	9	6	8	4	4	6	8	10	8	10	-
Man_5	6	8	10	13	10	7	6	2	5	9	10	-
Man_6	9	8	7	7	9	10	10	5	7	7	8	-
Tar_1	12	9	7	5	4	8	8	7	7	5	-	-
Tar_2	10	10	5	2	3	3	6	7	5	5	-	-
Tar_3	5	2	3	8	8	11	8	5	4	3	-	-
Tar_4	2	3	6	10	9	6	4	5	5	7	-	-
Tar_5	5	1	2	5	6	10	8	12	13	-	-	-
Tar_6	11	8	5	1	5	8	11	9	-	-	-	-

Tabla A.4: Nivel de ocupación luego de cada parada

	Minutos									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Man_1	2	7	7	12	10	8	3	2	4	6
Man_2	5	3	7	6	5	5	4	7	5	5
Man_3	11	9	8	7	4	6	7	7	6	3
Man_4	2	6	4	6	3	3	4	7	5	7
Man_5	5	7	9	12	12	7	6	2	5	7
Man_6	8	7	6	4	3	6	4	5	5	7
Tar_1	10	8	6	5	3	6	6	5	5	4
Tar_2	3	7	4	2	3	2	3	3	2	3
Tar_3	3	2	3	7	7	8	5	3	4	3
Tar_4	3	3	4	7	7	6	4	5	5	5
Tar_5	6	1	2	4	5	8	7	9	10	11
Tar_6	12	8	8	1	3	5	8	8	8	8

Tabla A.5: Cantidad de dispositivos por minuto en el ómnibus

	Minutos									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Man_1	2	8	8	14	12	6	4	2	5	7
Man_2	6	4	8	6	5	6	5	9	6	5
Man_3	15	12	9	7	4	8	10	5	8	3
Man_4	6	9	6	8	4	4	5	10	8	10
Man_5	6	8	10	13	10	7	6	2	9	10
Man_6	9	8	7	7	7	11	6	8	8	8
Tar_1	12	9	7	5	4	8	8	7	5	5
Tar_2	10	10	5	2	3	3	6	7	5	5
Tar_3	5	2	3	8	8	11	5	5	4	3
Tar_4	1	3	6	9	9	6	4	5	5	7
Tar_5	7	1	2	5	6	10	8	12	12	13
Tar_6	15	8	8	1	5	8	11	9	9	9

Tabla A.6: Nivel de ocupación por minuto

Apéndice B

Códigos

Sensores de ultrasonido: Adquisición de medidas

```
#!/usr/bin/python3

import time
import RPi.GPIO as GPIO
import matplotlib.pyplot as  plt
import numpy as np

# Configurar el GPIO con convenio de numerado BCM
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

#Setea pines de entrada y salida
# Trigger
Trigger1 = 24
GPIO.setup(Trigger1, GPIO.OUT)
Trigger2 = 27
GPIO.setup(Trigger2, GPIO.OUT)


# Echo
Echo1 = 23
GPIO.setup(Echo1, GPIO.IN)
Echo2 = 17
GPIO.setup(Echo2, GPIO.IN)

# inicializa  variables
cont_med = 0
gente = 0
total = 0
```

```
media = 0
bandera = False
start5=time.time()
x=list()
y1=list()
y2=list()

try:
    #while True:
    for i in range(1,300):
        start1 = 0
        end1 = 0
        # Configura el sensor
        GPIO.output(Trigger1, False)
        time.sleep(0.05)  # Tiempo a esperar entre medidas consecutivas

        # Empezamos a medir. Enviamos pulso de 10 microsegundos en el
            trigger
        GPIO.output(Trigger1, True)
        time.sleep(10*10**-6) #10 microsegundos
        GPIO.output(Trigger1, False)


        # Medimos el tiempo del pulso en Echo
        # Flanco de 0 a 1 = inicio
        while GPIO.input(Echo1) == GPIO.LOW:
            start1 = time.time()
        # Flanco de 1 a 0 = fin
        while GPIO.input(Echo1) == GPIO.HIGH:
            end1 = time.time()




        start2 = 0
        end2 = 0
        # Configura el sensor
        GPIO.output(Trigger2, False)
        time.sleep(0.05)  # Tiempo a esperar entre medidas consecutivas

        # Empezamos a medir. Enviamos pulso de 10 microsegundos en el
            trigger
        GPIO.output(Trigger2, True)
        time.sleep(10*10**-6) #10 microsegundos
```

```
        GPIO.output(Trigger2, False)


        # Medimos el tiempo del pulso en Echo
        # Flanco de 0 a 1 = inicio
        while GPIO.input(Echo2) == GPIO.LOW:
            start2 = time.time()
        # Flanco de 1 a 0 = fin
        while GPIO.input(Echo2) == GPIO.HIGH:
            end2 = time.time()




        end5=time.time()
        # el tiempo que devuelve time() esta en segundos
        distancia1 = (end1-start1) * 343 / 2  # 343 es la velocidad del sonido
            y se divide entre 2 porque el tiempo toma el camino de ida y
            vuelta
        distancia2 = (end2-start2) * 343 / 2  # 343 es la velocidad del sonido
            y se divide entre 2 porque el tiempo toma el camino de ida y
            vuelta


        print (" %.2f  %.2f  %f" % (distancia1, distancia2,end5-start5))

except KeyboardInterrupt:
    print ("\nFin del programa")
    GPIO.output(Trigger1, False)
    GPIO.output(Trigger2, False)
    GPIO.cleanup()
```

Sensores de ultrasonido: Estimación

```
import numpy as np
import matplotlib.pyplot as plt


def Mediana(arr, side):
    fil = np.zeros_like(arr)
    for j in range(side, len(arr)-side):
        window =arr[j-side:j+side+1]
        fil[j] = np.sort(window, axis=None)[window.size/2]

    return fil

def Media(arr, side):
    fil = np.zeros_like(arr)
    for j in range(side, len(arr)-side):
        window = arr[j-side:j+side+1]
        fil[j] = np.sum(window)/len(window)

    return fil

def MinMax(arr, side, media):
    fil = np.zeros_like(arr)
    for j in range(side, len(arr)-side):
        window = arr[j-side:j+side+1]
        if np.abs(np.min(window)-media) > np.abs(np.max(window)-media):
            fil[j] = np.min(window)
        else:
            fil[j] = np.max(window)
    return fil

# Ground Truth
#================================
#Manana 1
Man1=
    [[62,74,4,'s' ,1],[116,119,2, 's' ,2],[183,187,2, 's' ,3],[225,238,4, 's' ,4],[276,280,2, 'b' ,5],
[305,318,3, 'b' ,6],[360,367,3, 'b' ,7],[409,414,2, 'b' ,8],[442,450,2, 'b' ,9],[492,500,3, 's' ,10],
[541,546,2, 's' ,11],[585,587,1, 's' ,12],[589,591,1, 'b' ,12]]

#Manana 2
Man2=
    [[48,52,2,'b' ,1],[55,58,2, 's' ,1],[114,119,2, 'b' ,2],[122,126,2, 's' ,2],[171,175,2, 's' ,3],
[211,219,2, 'b' ,4],[261,268,3, 'b' ,5],[272,276,2, 's' ,5],[321,324,1, 'b' ,6],[330,335,2, 's' ,6],
[383,384,1, 'b' ,7],[424,428,2, 's' ,8],[468,472,2, 's' ,9],[526,534,4, 'b' ,10],[536,537,1, 's' ,10],
```

```
[579,587,2, 'b' ,11],[590,592,1, 's' ,11]]

#Manana 3
Man3=
    [[47,53,3,'b' ,1],[99,107,3, 'b' ,2],[150,157,3, 'b' ,3],[198,200,1, 's' ,4],[203,208,3, 'b' ,4],
[244,254,4, 'b' ,5],[256,258,1, 's' ,5],[306,308,1, 's' ,6],[342,351,3, 's' ,7],[379,383,1, 'b' ,8],
[386,392,3, 's' ,8],[425,430,2, 'b' ,9],[433,436,1, 's' ,9],[479,486,4, 'b' ,10],[489,493,2, 's' ,10],
[536,539,1, 's' ,11],[578,590,5, 'b' ,12]]

#Manana 4
Man4=
    [[49,60,4,'s' ,1],[109,117,3, 's' ,2],[170,180,3, 'b' ,3],[215,227,4, 's' ,4],[233,239,2, 'b' ,4],
[269,276,4, 'b' ,5],[319,326,1, 's' ,6],[327,332,2, 'b' ,6],[335,338,1, 's' ,6],[381,386,2, 's' ,7],
[417,419,1, 'b' ,8],[421,431,3, 's' ,8],[470,475,2, 's' ,9],[524,529,2, 'b' ,10],[562,568,2, 's' ,11]]

#Manana 5
Man5=
    [[23,35,4,'s' ,1],[70,74,2, 's' ,2],[124,129,2, 's' ,3],[203,213,3, 's' ,4],[300,309,3, 'b' ,5],
[344,354,3, 'b' ,6],[387,404,1, 'b' ,7],[431,442,4, 'b' ,8],[491,500,3, 's' ,9],[536,547,4, 's' ,10],
[579,581,1, 'b' ,11],[583,591,2, 's' ,11]]

#Manana 6
Man6=
    [[45,52,3,'b' ,1],[85,88,2, 'b' ,2],[90,93,1, 's' ,2],[133,142,2, 'b' ,3],[144,149,1, 's' ,3],
[183,192,2, 'b' ,4],[193,198,2, 's' ,4],[261,267,2, 's' ,5],[295,300,2, 'b' ,6],[302,308,3, 's' ,6],
[337,350,1, 'b' ,7],[355,358,2, 's' ,7],[390,401,5, 'b' ,8],[403,406,1, 's' ,8],[406,408,1, 'b' ,8],
[467,471,2, 's' ,9],[502,507,2, 'b' ,10],[511,516,2, 's' ,10],[564,572,2, 'b' ,11],[576,586,2, 's' ,11]]

#==============================


#Tarde1
Tar1 =
    [[36,47,3, 'b' ,1],[90,100,3, 'b' ,2],[160,167,2, 'b' ,3],[207,213,2, 'b' ,4],[258,267,4, 'b' ,5],
[270,277,3, 's' ,5],[310,320,4, 's' ,6],[382,383,1, 'b' ,7],[383,389,1, 's' ,7],[389,390,1, 'b' ,7],
[390,394,1, 's' ,7],[437,445,3, 'b' ,8],[446,451,2, 's' ,8],[495,499,2, 'b' ,9],[501,505,2, 's' ,9],
[540,546,2, 'b' ,10]]

#Tarde2
Tar2 =
    [[42,50,3, 'b' ,1],[50,60,4, 's' ,1],[99,105,2, 'b' ,2],[106,113,2, 's' ,2],[170,181,5, 'b' ,3],
[227,237,3, 'b' ,4],[282,284,1, 'b' ,5],[285,295,2, 's' ,5],[324,330,2, 's' ,6],[331,339,2, 'b' ,6],
[395,403,3, 'b' ,7],[404,428,6, 's' ,7],[468,470,1, 'b' ,8],[472,480,2, 's' ,8],[526,534,3, 'b' ,9],
[537,540,1, 's' ,9],[571,575,2, 'b' ,10],[578,583,2, 's' ,10]]
```

```
#Tarde3
Tar3 =
    [[50,53,1,'b' ,1],[79,89,3, 'b' ,2],[148,150,1, 'b' ,3],[152,156,2, 's' ,3],[202,217,5, 's' ,4],
[258,260,1,'b' ,5],[261,263,1, 's' ,5],[301,308,3, 's' ,6],[367,376,4, 'b' ,7],[379,382,1, 's' ,7],
[420,432,3,'b' ,8],[481,484,1, 's' ,9],[485,489,2, 'b' ,9],[541,543,1, 'b' ,10]]

#Tarde4
Tar4 =
    [[60,62,2,'b' ,1],[63,66,1, 's' ,1],[94,96,1, 's' ,2],[159,161,1, 'b' ,3],[164,173,4, 's' ,3],
[208,215,2,'b' ,4],
    [220,240,5,'s' ,4],[248,252,1, 's' ,4],[296,298,1, 'b' ,5],[329,338,3, 'b' ,6],
[400,409,3,'b' ,7],[411,414,1, 's' ,7],[451,456,1, 'b' ,8],[458,463,2, 's' ,8],[509,514,1, 'b' ,9],
[516,518,1,'s' ,9],[547,549,1, 'b' ,10],[553,559,3, 's' ,10]]

#Tarde5
Tar5 =
    [[64,67,2,'b' ,1],[102,111,4, 'b' ,2],[173,175,1, 's' ,3],[231,233,1, 's' ,4],[239,245,2, 's' ,4],
[281,284,1,'b' ,5],[286,291,2, 's' ,5],[323,339,4, 's' ,6],[406,411,2, 'b' ,7],[457,459,1, 's' ,8],
[472,474,1,'s' ,8],[480,487,2, 's' ,8],[541,543,1, 's' ,9]]

#Tarde6
Tar6 =
    [[65,76,4,'b' ,1],[110,119,3, 'b' ,2],[188,195,3, 'b' ,3],[232,247,4, 'b' ,4],[288,299,4, 's' ,5],
[329,337,3,'s' ,6],[403,412,3, 's' ,7],[459,469,3, 'b' ,8],[471,474,1, 's' ,8]]

GroundTruth =
    [Man1,Man2,Man3,Man4,Man5,Man6,Tar1,Tar2,Tar3,Tar4,Tar5,Tar6]


#
tiempo = []
sensor_1 = []
sensor_2 = []
sensor_3 = []
sensor_4 = []
SubBaj = []

#Manana

for m in range(6):

  indx = m + 1
  csv = np.genfromtxt(r'./MercedesDomingoManana%d.csv' % indx,
      delimiter=',', names=True, dtype=None)
```

```
    tiempo.append(csv['tiempo'])
    sensor_1.append(csv['sensor1'])
    sensor_2.append(csv['sensor2'])
    sensor_3.append(csv['sensor3'])
    sensor_4.append(csv['sensor4'])

#Tarde

for t in range(6):

    indx = t + 1
    csv = np.genfromtxt(r'./MercedesDomingoTarde%d.csv' % indx,
        delimiter=',', names=True, dtype=None)

    tiempo.append(csv['tiempo'])
    sensor_1.append(csv['sensor1'])
    sensor_2.append(csv['sensor2'])
    sensor_3.append(csv['sensor3'])
    sensor_4.append(csv['sensor4'])

#%%
Personas_incio =  [2,6,18,2,2,12,15,9,6,3,7,15]

for p in range(12):

    #Paso los datos a arreglos
    sen1 = np.array(sensor_1[p])
    sen2 = np.array(sensor_2[p])
    sen3 = np.array(sensor_3[p])
    sen4 = np.array(sensor_4[p])
    time = np.array(tiempo[p])

    #Los filtro
    side = 2
     sen1_fil  = Mediana(sen1,side)
     sen1_fil  = Media(sen1_fil,side)
#  sen1_fil  = MinMax(sen1,side,0.98)
     sen2_fil  = Mediana(sen2,side)
     sen2_fil  = Media(sen2_fil,side)
#  sen2_fil  = MinMax(sen2,side,0.98)
     sen3_fil  = Mediana(sen3,side)
     sen3_fil  = Media(sen3_fil,side)
#     sen3_fil  = MinMax(sen3,side,0.98)
     sen4_fil  = Mediana(sen4,side)
     sen4_fil  = Media(sen4_fil,side)
```

```
# sen4_fil = MinMax(sen4,side,0.98)



  #Diagrama de Estados
  #Umbrales
  th_inf = 0.85
  th_sup = 1.15

  bef1 = sen1_fil[side]
  bef2 = sen2_fil[side]
  bef3 = sen3_fil[side]
  bef4 = sen4_fil[side]

  Input = [0,0,0,0] #[Act A, Des A, Act B, Des B]

  state = 'normal'
  cont = 0
  cont_in = 0
  cont_out = 0
  person_in = 0
  person_out = 0 #Cantidad de entradas y salidas totales
  equal = 3
  time_out = 15 #Tiempo de espera en los estados proces 2

  sb_time = []

  for i in range(side+1,len(sen1_fil)-side):
    # Entradas
    antes12 = ((bef1 > th_inf) and (bef1 < th_sup)) and ((bef2 > th_inf) and
        (bef2 < th_sup))
    antes34 = ((bef3 > th_inf) and (bef3 < th_sup)) and ((bef4 > th_inf) and
        (bef4 < th_sup))
    actual12 = ((sen1_fil[i] > th_inf) and (sen1_fil[i] < th_sup)) and
        ((sen2_fil[i] > th_inf) and (sen2_fil[i] < th_sup))
    actual34 = ((sen3_fil[i] > th_inf) and (sen3_fil[i] < th_sup)) and
        ((sen4_fil[i] > th_inf) and (sen4_fil[i] < th_sup))

    if (antes12 and not actual12): #Activacion de los Sensors adentro
      Input[2] = 1
    else:
      Input[2] = 0

    if (not antes12 and actual12): #Desactivacion de los Sensores adentro
      Input[3] = 1
```

```
else:
  Input[3] = 0

if (antes34 and not actual34): #Activacion de los Sensores afuera
  Input[0] = 1
else:
  Input[0] = 0

if (not antes34 and actual34): #Desactivacion de los Sensores afuera
  Input[1] = 1
else:
  Input[1] = 0



# Transiciones
if Input != [0,0,0,0]:

  if state == 'normal':

    if (Input[0] == 1) and (Input[2]!=1):
      state = 'tran_input'
      cont_in += 1

    elif (Input[2] == 1) and (Input[0]!=1):
      state = 'tran_output'
      cont_out += 1

  elif state == 'tran_input':

    if (Input[2] == 1):
      state = 'normal'
      if cont_in > equal:
        person_in = person_in + 1
        cont_in = 0
        sb_time.append(['s',time[i ]])

      else:
        cont_in = 0

    elif (Input[1] == 1):
      state = 'tran_input_2'
      cont_in += 1

    else:
```

```
            cont_in += 1

        elif state == 'tran_input_2':
            if (Input[0] == 1):
                state = 'tran_input'
                cont_in += 1
            elif (Input[2] == 1):
                state = 'normal'
                if cont_in > equal:
                    person_in = person_in + 1
                    cont_in = 0
                    sb_time.append(['s',time[i]])

                else:
                    cont_in = 0
            else:
                cont_in += 1

        elif state == 'tran_output':
            if (Input[0] == 1):
                state = 'normal'
                if cont_out > equal:
                    person_out = person_out + 1
                    cont_out = 0
                    sb_time.append(['b',time[i]])

                else:
                    cont_out = 0
            elif (Input[3] == 1):
                state = 'tran_output_2'
                cont_out += 1
            else:
                cont_out += 1

        elif state == 'tran_output_2':
            if (Input[2] == 1):
                state = 'tran_output'
                cont_out += 1
            elif (Input[0] == 1):
                state = 'normal'
                if cont_out > equal:
                    person_out = person_out + 1
                    cont_out = 0
                    sb_time.append(['b',time[i]])
```

```
            else:
              cont_out = 0

        else:
          if (state != 'normal'):
            if cont_in != 0:
              cont_in += 1
            if cont_out != 0:
              cont_out += 1
          if (state != 'normal') and (actual12 and actual34):
            cont = cont + 1
            if cont == time_out:
              state = 'normal'
              cont = 0
              cont_in = 0
              cont_out = 0
          else:
            cont = 0


        bef1 = sen1_fil[i]
        bef2 = sen2_fil[i]
        bef3 = sen3_fil[i]
        bef4 = sen4_fil[i]

      SubBaj.append(sb_time)


#%%
#Subidas y bajadas estimadas por parada
ResultBS_paradas = []
for i in range(len(SubBaj)):
  max_paradas = GroundTruth[i][-1][4]
  subidas_paradas = np.zeros(max_paradas)
  bajadas_paradas = np.zeros(max_paradas)
  for j in range(len(SubBaj[i])):
    bandera = False
    for k in range(len(GroundTruth[i])):
      t = SubBaj[i][j][1]
      t_start = GroundTruth[i][k][0]-10
      t_fin = GroundTruth[i][k][1]+10

      if ((t_start <= t) and (t < t_fin)):
        parada = GroundTruth[i][k][4]
```

```
            if SubBaj[i][j ][0] == 's':
              subidas_paradas[parada-1] +=1
              bandera = True

            else :
              bajadas_paradas[parada-1] +=1
              bandera = True

            break

    ResultBS_paradas.append([subidas_paradas,bajadas_paradas])

sub_Man1 =  [4,2,2,4,0,0,0,0,0,3,2,1]
sub_Man2 =  [2,2,2,0,2,2,0,2,2,1,1]
sub_Man3 =  [0,0,0,1,1,1,3,3,1,2,1,0]
sub_Man4 =  [4,3,0,4,0,2,2,3,2,0,2]
sub_Man5 =  [4,2,2,3,0,0,0,0,3,4,2]
sub_Man6 =  [0,1,1,2,2,3,1,1,2,2,2]

sub_Tar1 =  [0,0,0,0,3,4,2,2,2,0]
sub_Tar2 =  [4,2,0,0,2,2,6,2,1,2]
sub_Tar3 =  [0,0,2,5,1,3,1,0,1,0]
sub_Tar4 =  [1,1,4,6,0,0,1,2,1,3]
sub_Tar5 =  [0,0,1,3,2,4,0,4,1]
sub_Tar6 =  [0,0,0,0,4,3,3,1]

baj_Man1 =  [0,0,0,0,2,3,3,2,2,0,0,1]
baj_Man2 =  [2,2,0,2,3,1,1,0,0,4,2]
baj_Man3 =  [3,3,3,3,4,0,0,1,2,4,0,5]
baj_Man4 =  [0,0,3,2,4,2,0,1,0,2,0]
baj_Man5 =  [0,0,0,0,3,3,1,4,0,0,1]
baj_Man6 =  [3,2,2,2,0,2,1,6,0,2,1]

baj_Tar1 =  [3,3,2,2,4,0,2,3,2,2]
baj_Tar2 =  [3,2,5,3,1,2,3,1,3,2]
baj_Tar3 =  [1,3,1,0,1,0,4,3,2,1]
baj_Tar4 =  [2,0,1,2,1,3,3,1,1,1]
baj_Tar5 =  [2,4,0,0,1,0,2,0,0]
baj_Tar6 =  [4,3,3,4,0,0,0,3]

GT_SB = []
GT_SB.append([sub_Man1,baj_Man1])
GT_SB.append([sub_Man2,baj_Man2])
GT_SB.append([sub_Man3,baj_Man3])
GT_SB.append([sub_Man4,baj_Man4])
```

```
GT_SB.append([sub_Man5,baj_Man5])
GT_SB.append([sub_Man6,baj_Man6])

GT_SB.append([sub_Tar1,baj_Tar1])
GT_SB.append([sub_Tar2,baj_Tar2])
GT_SB.append([sub_Tar3,baj_Tar3])
GT_SB.append([sub_Tar4,baj_Tar4])
GT_SB.append([sub_Tar5,baj_Tar5])
GT_SB.append([sub_Tar6,baj_Tar6])



Cantidad_sb = np.zeros(7)
Cantidad_sb_acc = np.zeros(7)
Cantidad_sb_error = np.zeros(7)


for a in range(len(GT_SB)):
  for b in range(len(GT_SB[a])):
    for c in range(len(GT_SB[a][b])):
      if GT_SB[a][b][c] == 0:
        Cantidad_sb[0] += 1
        if GT_SB[a][b][c] <= ResultBS_paradas[a][b][c]:
          Cantidad_sb_acc[0] += 1
        else:
          Cantidad_sb_error[0] += (GT_SB[a][b][c] -
              ResultBS_paradas[a][b][c])**2

      elif GT_SB[a][b][c] == 1:
        Cantidad_sb[1] += 1
        if GT_SB[a][b][c] <= ResultBS_paradas[a][b][c]:
          Cantidad_sb_acc[1] += 1
        else:
          Cantidad_sb_error[1] += (GT_SB[a][b][c] -
              ResultBS_paradas[a][b][c])**2

      elif GT_SB[a][b][c] == 2:
        Cantidad_sb[2] += 1
        if GT_SB[a][b][c] <= ResultBS_paradas[a][b][c]:
          Cantidad_sb_acc[2] += 1
        else:
          Cantidad_sb_error[2] += (GT_SB[a][b][c] -
              ResultBS_paradas[a][b][c])**2

      elif GT_SB[a][b][c] == 3:
```

```
            Cantidad_sb[3] += 1
            if GT_SB[a][b][c] <= ResultBS_paradas[a][b][c]:
              Cantidad_sb_acc[3] += 1
            else:
              Cantidad_sb_error[3] += (GT_SB[a][b][c] -
                  ResultBS_paradas[a][b][c])**2

          elif GT_SB[a][b][c] == 4:
            Cantidad_sb[4] += 1
            if GT_SB[a][b][c] <= ResultBS_paradas[a][b][c]:
              Cantidad_sb_acc[4] += 1
            else:
              Cantidad_sb_error[4] += (GT_SB[a][b][c] -
                  ResultBS_paradas[a][b][c])**2

          elif GT_SB[a][b][c] == 5:
            Cantidad_sb[5] += 1
            if GT_SB[a][b][c] <= ResultBS_paradas[a][b][c]:
              Cantidad_sb_acc[5] += 1
            else:
              Cantidad_sb_error[5] += (GT_SB[a][b][c] -
                  ResultBS_paradas[a][b][c])**2

          elif GT_SB[a][b][c] == 6:
            Cantidad_sb[6] += 1
            if GT_SB[a][b][c] <= ResultBS_paradas[a][b][c]:
              Cantidad_sb_acc[6] += 1
            else:
              Cantidad_sb_error[6] += (GT_SB[a][b][c] -
                  ResultBS_paradas[a][b][c])**2

Cantidad_sb_error = np.sqrt(Cantidad_sb_error/(Cantidad_sb-Cantidad_sb_acc))
Cantidad_sb_acc_porc = 100*(Cantidad_sb_acc/Cantidad_sb)

#%%
#Ocupacion por parada
Ocupacion = []
Personas_incio =  [2,6,18,2,2,12,15,9,6,3,7,15]
for i in range(len(ResultBS_paradas)):
  Ocup_aux = np.zeros(len(ResultBS_paradas[i][0])+1)
  Ocup_aux[0] = Personas_incio[i]
  for j in range(1,len(Ocup_aux)):
    Ocup_aux[j] = Ocup_aux[j-1] + ResultBS_paradas[i][0][j-1] -
        ResultBS_paradas[i][1][j-1]
```

```
    Ocupacion.append(Ocup_aux)

#%%

from scipy.stats.stats import pearsonr
from sklearn.metrics import mean_squared_error

Indices = []

GTO_Man1 = [2,6,8,10,14,12,9,6,4,2,5,7,7]
GTO_Man2 = [6,6,6,8,6,5,6,5,7,9,6,5]
GTO_Man3 = [18,15,12,9,7,4,5,8,10,9,7,8,3]
GTO_Man4 = [2,6,9,6,8,4,4 ,6,8,10,8,10]
GTO_Man5 = [2,6,8,10,13,10,7,6,2,5,9,10]
GTO_Man6 = [12,9,8,7,7,9,10,10,5,7,7,8]

GTO_Tar1 = [15,12,9,7,5,4,8,8,7,7,5]
GTO_Tar2 = [9,10,10,5,2,3,3,6,7,5,5]
GTO_Tar3 = [6,5,2,3,8,8,11,8,5,4,3]
GTO_Tar4 = [3,2,3,6,10,9,6,4,5,5,7]
GTO_Tar5 = [7,5,1,2,5,6,10,8,12,13]
GTO_Tar6 = [15,11,8,5,1,5,8,11,9]

GTO =
    [GTO_Man1,GTO_Man2,GTO_Man3,GTO_Man4,GTO_Man5,GTO_Man6,
GTO_Tar1,GTO_Tar2,GTO_Tar3,GTO_Tar4,GTO_Tar5,GTO_Tar6]

#graficas
for i in range(12):
    plt.figure('Captura_%d' %i)
    plt.title("Estimacion_de_ocupacion")
    plt.plot(GTO[i], 'ks-', linewidth=2.5, label='True')
    plt.plot(Ocupacion[i], 'mo-', linewidth=2, label='Estimacion')

    plt.ylim(0,18)
    plt.xlim(0,len(GTO[i])-1)
    plt.ylabel("Ocupacion")
    plt.xlabel("Paradas")
    plt.fill_between(range(len(GTO[i])),0,6, facecolor='green',alpha=0.3)
    plt.fill_between(range(len(GTO[i])),6,12, facecolor='yellow',alpha=0.2)
    plt.fill_between(range(len(GTO[i])),12,18, facecolor='red',alpha=0.2)
    plt.legend()
    plt.grid()

    plt.savefig('./Ocupacion/Ocupacion_%d_mm.pdf' %i)
```

```
    plt.close()

    Pearson = pearsonr(Ocupacion[i],GTO[i])
    PO = np.sum((Ocupacion[i])*100/GTO[i])/len(GTO[i])
    RMSE = np.sqrt(mean_squared_error(Ocupacion[i],GTO[i]))

    Indices.append([i+1,Pearson[0],PO,RMSE])


with open("./Ocupacion/Indices.txt", "w") as file:
    file.write(str(Indices))

#%%
prom_Pearson = 0
prom_PO = 0
prom_RMSE = 0
for i in range(12):
    prom_Pearson += Indices[i][1]
    prom_PO += Indices[i][2]
    prom_RMSE += Indices[i][3]

prom_Pearson = prom_Pearson/12
prom_PO = prom_PO/12
prom_RMSE = prom_RMSE/12

print prom_Pearson
print prom_PO
print prom_RMSE


#%% Niveles de colores

tot_paradas = 0
acc_niveles = 0

sup = 12
inf = 6

for i in range(6,12):
    for j in range(len(GTO[i])):
        tot_paradas += 1
        if (GTO[i][j] > sup) and (Ocupacion[i][j] > sup):
            acc_niveles += 1
        elif (GTO[i][j] <= sup) and (Ocupacion[i][j] <= sup) and (GTO[i][j] >
            inf) and (Ocupacion[i][j] > inf):
```

```
            acc_niveles += 1
        elif (GTO[i][j] < inf) and (Ocupacion[i][j] < inf):
            acc_niveles += 1

print 100*float(acc_niveles)/tot_paradas
```

Cámara: Grabado de video en la Raspberry Pi

```
from time import sleep
from picamera import PiCamera

camera = PiCamera(resolution=(640, 480), framerate=30)

camera.start_preview()
camera.start_recording('video.h264')
camera.wait_recording(600)
camera.stop_recording()
camera.stop_preview()
camera.close()
```

Cámara: Object Detection Modificado

```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile
import time
import dlib
import cv2
from collections import defaultdict
from io import StringIO
from PIL import Image
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as vis_util

start = time.time()

MODEL_NAME='ssd_mobilenet_v1_coco_11_06_2017'#fast
#MODEL_NAME='faster_rcnn_resnet101_coco_11_06_2017'#medium speed
MODEL_FILE=MODEL_NAME+'.tar.gz'
DOWNLOAD_BASE='http://download.tensorflow.org/models/object_detection/'
PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'
PATH_TO_LABELS =
    os.path.join('/home/llagu/Escritorio/ObjectDetection/tensorflow/models/research/object_det
    'mscoco_label_map.pbtxt')

NUM_CLASSES = 90

IMAGE_SIZE = (12, 8)

fileAlreadyExists = os.path.isfile(PATH_TO_CKPT)

if not fileAlreadyExists:
  print('Downloading frozen_inference_graph')
  opener = urllib.request.URLopener()
  opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)
   tar_file = tarfile.open(MODEL_FILE)
  for file in tar_file.getmembers():
    file_name = os.path.basename(file.name)
    if 'frozen_inference_graph.pb' in file_name:
      tar_file.extract(file, os.getcwd())
```

```
detection_graph = tf.Graph()
with detection_graph.as_default():
  od_graph_def = tf.GraphDef()
  with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
    serialized_graph = fid.read()
    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map,
    max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)

tracker = []
j = 0
auxCentros = []
auxBanderasConteo = []
RadioNear = 50


###-----------------------------------------------------###
#Voy a hacer una funcion para contar
#Subida
def goingUP(antes,ahora,lineUP):
  if antes[0]<lineUP and ahora[0]>=lineUP:
    return True
  else:
    return False

#Bajada
def goingDOWN(antes,ahora,lineDOWN):
  if antes[0]>lineDOWN and ahora[0]<=lineDOWN:
    return True
  else:
    return False
###-----------------------------------------------------###
#Voy a hacer una funcion para ver si dos puntos estan cerca
def IsNear(punto,centro,radio):
  if ((punto[0]-centro[0])**2+(punto[1]-centro[1])**2<radio**2):
    return True
  else:
    return False
###-----------------------------------------------------###
```

```
#Ajuste de iluminacion
def adjust_gamma(image, gamma=1.0):

  invGamma = 1.0 / gamma
  table = np.array([((i / 255.0) ** invGamma) * 255
  for i in np.arange(0, 256)]).astype("uint8")

  return cv2.LUT(image, table)
####-----------------------------------------------------------------------


entrada = sys.argv[1]
salida = sys.argv[2]

cap = cv2.VideoCapture(entrada)
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
out = cv2.VideoWriter(salida,cv2.VideoWriter_fourcc('M','J','P','G'), 25,
    (frame_width,frame_height))

centroAntes = []
BanderasConteo = []
contUP = 0
contDOWN = 0
cx = 0
cy = 0
lineaUP = int(48*frame_width/100)
lineaDOWN = int(48*frame_width/100)
pt1 = [lineaDOWN, 0];
pt2 = [lineaDOWN, frame_height];
pts_L1 = np.array([pt1,pt2], np.int32)
pts_L1 = pts_L1.reshape((-1,1,2))
pt3 = [lineaUP, 0];
pt4 = [lineaUP, frame_height];
pts_L2 = np.array([pt3,pt4], np.int32)
pts_L2 = pts_L2.reshape((-1,1,2))

Primera = True

gamma = 1.5

with detection_graph.as_default():
  with tf.Session(graph=detection_graph) as sess:
    while(cap.isOpened()):
      ret, frame = cap.read()
```

```
if(ret):
  frame1 = cv2.resize(frame, (frame_width,frame_height))
  frame1 = adjust_gamma(frame, gamma=gamma)
  image_np = np.array(frame1)

  # Expand dimensions since the model expects images to have shape: [1,
      None, None, 3]
  image_np_expanded = np.expand_dims(image_np, axis=0)

  # Definite input and output Tensors for detection_graph
  image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

  # Each box represents a part of the image where a particular object
      was detected.
  detection_boxes =
      detection_graph.get_tensor_by_name('detection_boxes:0')

  # Each score represent how level of confidence for each of the objects.
  # Score is shown on the result image, together with the class label.
  detection_scores =
      detection_graph.get_tensor_by_name('detection_scores:0')

  detection_classes =
      detection_graph.get_tensor_by_name('detection_classes:0')

  num_detections =
      detection_graph.get_tensor_by_name('num_detections:0')

  print('Running_detection..')
  (boxes, scores, classes, num) = sess.run(
    [detection_boxes, detection_scores, detection_classes,
        num_detections],
    feed_dict={image_tensor: image_np_expanded})

  taco = [category_index.get(value) for index,value in
      enumerate(classes[0]) if scores[0,index] > 0.5]
  print('Numero_de_cara:_', len(taco))  ##Cantidad de personas que se
      detectan por frame len(taco)
  detecciones = []
  for i in range(len(taco)):
    detecciones.append(taco[i]['name'])
  print(detecciones)
  tracker = []
  auxCentros = centroAntes[:]
  centroAntes = []
```

```
Antes = []
auxBanderasConteo = BanderasConteo[:]
BanderasConteo = []
cv2.polylines(image_np,[pts_L2],False,(0,0,255),thickness=2)
cv2.polylines(image_np,[pts_L1],False,(255,0,0),thickness=2)
str_up = 'UP: '+ str(contUP)
str_down = 'DOWN: '+ str(contDOWN)
cv2.putText(image_np, str_up
    ,(10,40),cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,255,255),2,cv2.LINE_AA)
cv2.putText(image_np, str_up
    ,(10,40),cv2.FONT_HERSHEY_SIMPLEX,0.5,(0,0,255),1,cv2.LINE_AA)
cv2.putText(image_np, str_down
    ,(10,90),cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,255,255),2,cv2.LINE_AA)
cv2.putText(image_np, str_down
    ,(10,90),cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,0,0),1,cv2.LINE_AA)

for i in range(len(taco)):                    #hago el for por cada persona
    que encuentra
  if (taco[i]['name']=='person'):
    frame1_height,frame1_width,aux = frame1.shape #obtengo el largo
        y ancho del frame despues del resize
    ymin = boxes[0][i][0]*frame1_height
    xmin = boxes[0][i][1]*frame1_width
    ymax = boxes[0][i][2]*frame1_height
    xmax = boxes[0][i][3]*frame1_width
    bbox = (int(xmin),int(ymin),int(xmax-xmin),int(ymax-ymin))
        ##Inicializo el tracker con las cajas que devuelve Object
        Detection
    p1 = (int(bbox[0]), int(bbox[1]))
    p2 = (int(bbox[0])+int(bbox[2]), int(bbox[1])+int(bbox[3]))
    cx = int((xmax+xmin)/2)
    cy = int((ymax+ymin)/2)
    cv2.circle(image_np,(cx,cy), RadioNear, (0,0,255), -1)
    cv2.rectangle(image_np, p1, p2, (0,255,0), 2, 1)
    tracker.append(dlib.correlation_tracker())
    rect = dlib.rectangle(int(bbox[0]), int(bbox[1]), int(bbox[0] +
        bbox[2]), int(bbox[1] + bbox[3]))
    tracker[i].start_track(frame1, rect)
    Antes.append('MEDIO')

    if Primera == False:
      Bandera = True
      for k in range(len(auxCentros)):
        if IsNear(auxCentros[k],(cx,cy),RadioNear):
          centroAntes.append(auxCentros[k])
```

```
                BanderasConteo.append(auxBanderasConteo[k])
                Bandera = False
                break
            if Bandera:
              centroAntes.append((cx,cy))
              BanderasConteo.append(False)
          else:
            centroAntes.append((cx,cy))
            BanderasConteo.append(False)

          if Primera:
            Primera = False

        else:
          tracker.append(dlib.correlation_tracker())
          BanderasConteo.append(False)
          centroAntes.append((cx,cy))
          Antes.append('MEDIO')
        ###-----------------------------###
      out.write(image_np)
      cv2.imshow('object_detection', image_np)
      if cv2.waitKey(25) & 0xFF == ord('q'):
        cv2.destroyAllWindows()
        break

      if 'person' in detecciones:    ##me fijo si lo que detecte son
          personas, sino lo son paso a detectar de nuevo
        for i in range(15):
          ret, frame = cap.read()
          if(ret):
            frame1 = cv2.resize(frame, (frame_width,frame_height))
            frame1 = adjust_gamma(frame, gamma=gamma)
            image_np = np.array(frame1)

            for j in range(len(taco)):
              if (taco[j]['name']=='person'):
                tracker[j].update(image_np)
                pos = tracker[j].get_position()
                startX = int(pos.left())
                startY = int(pos.top())
                endX = int(pos.right())
                endY = int(pos.bottom())
                p1 = (startX, startY)
                p2 = (endX, endY)
                cx = int((startX+endX)/2)
```

```
                cy = int((startY+endY)/2)
                cv2.rectangle(image_np, p1, p2, (255,0,0), 2, 1)
                cv2.circle (image_np,(cx,cy), 2, (0,0,255), -1)
                cv2.polylines(image_np,[pts_L2],False,(0,0,255),thickness=2)
                cv2.polylines(image_np,[pts_L1],False,(255,0,0),thickness=2)

                if i == 1:
                  if cx < lineaDOWN:
                    Antes[j] = 'ABAJO'
                  elif cx > lineaUP:
                    Antes[j] = 'ARRIBA'

                if BanderasConteo[j] == False:
                  if not((0,0)==(cx,cy)):
                    if goingUP(centroAntes[j],(cx,cy),lineaUP) and Antes[j]
                        != 'ARRIBA':
                      contUP = contUP + 1
                      print('Centro Antes:',centroAntes[j], 'Centro
                          despuesn:',(cx,cy))
                      BanderasConteo[j] = True
                    elif goingDOWN(centroAntes[j],(cx,cy),lineaDOWN)
                        and Antes[j] != 'ABAJO':
                      contDOWN = contDOWN + 1
                      print('Centro Antes:',centroAntes[j], 'Centro
                          despuesn:',(cx,cy), 'LineDown:', lineaUP )
                      BanderasConteo[j] = True
                    centroAntes[j] = (cx,cy)

                print('UP:',contUP,'DOWN:',contDOWN)

                str_up = 'UP:'+ str(contUP)
                str_down = 'DOWN:'+ str(contDOWN)
                cv2.putText(image_np, str_up
                    ,(10,40),cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,255,255),2,cv2.LINE_A
                cv2.putText(image_np, str_up
                    ,(10,40),cv2.FONT_HERSHEY_SIMPLEX,0.5,(0,0,255),1,cv2.LINE_AA)
                cv2.putText(image_np, str_down
                    ,(10,90),cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,255,255),2,cv2.LINE_AA
                cv2.putText(image_np, str_down
                    ,(10,90),cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,0,0),1,cv2.LINE_AA)

            cv2.polylines(image_np,[pts_L2],False,(0,0,255),thickness=2)
            cv2.polylines(image_np,[pts_L1],False,(255,0,0),thickness=2)
            str_up = 'UP:'+ str(contUP)
            str_down = 'DOWN:'+ str(contDOWN)
```

```
                cv2.putText(image_np, str_up
                    ,(10,40),cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,255,255),2,cv2.LINE_AA)
                cv2.putText(image_np, str_up
                    ,(10,40),cv2.FONT_HERSHEY_SIMPLEX,0.5,(0,0,255),1,cv2.LINE_AA)
                cv2.putText(image_np, str_down
                    ,(10,90),cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,255,255),2,cv2.LINE_AA)
                cv2.putText(image_np, str_down
                    ,(10,90),cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,0,0),1,cv2.LINE_AA)

                out.write(image_np)
                cv2.imshow('object␣detection', image_np)
                if cv2.waitKey(25) & 0xFF == ord('q'):
                  cv2.destroyAllWindows()
                  break
              else:
                print('exiting')
                end = time.time()
                print('Duracion␣',end-start)
                text = salida[:-4]
                extension = '.txt'
                with open(text+extension, 'w') as file:
                  file.write(text + extension + '␣Tiempo␣procesamiento:␣' +
                      str(end-start) + '␣UP:␣' + str(contUP) + '␣DOWN:␣' +
                      str(contDOWN))
                cap.release()
                out.release()
                cv2.destroyAllWindows()


        else:
          print('exiting')
          end = time.time()
          print('Duracion␣',end-start)
          text = salida[:-4]
          extension = '.txt'
          with open(text+extension, 'w') as file:
            file.write(text + extension + '␣Tiempo␣procesamiento:␣' +
                str(end-start) + '␣UP:␣' + str(contUP) + '␣DOWN:␣' +
                str(contDOWN))
          cap.release()
          out.release()
          cv2.destroyAllWindows()

    cap.release()
    out.release()
```

```
cv2.destroyAllWindows()
```

WiFi: Captura de paquetes Probe Request

```
#!/bin/bash

for i in {1..5}
do
    sudo timeout 120 tcpdump -i wlan1 -e -s 0 -l type mgt subtype
        probe-req | grep -P --line-buffered -o
        '(([0-9]){2}:){2}(([0-9]){2}.)(([0-9]){6})|(?<=11b
        )(-([0-9]){2}dBm)|(?<=SA:)(([a-f]|[0-9]){2}:){5}([a-f]|[0-9]){2}' >
        prueba_captura_wifi$i.txt
done
```

WiFi: Modelo Discreto

```
import numpy as np
import sys

#Funcion que convierte el tiempo (hora minuto segundo) en segundos
def time_conv(t_orig):
    t = int(t_orig[:2])*360 + int(t_orig[3:5])*60 + float(t_orig[6:])
    return t

#Funcion que calcula el largo del archivo
def file_len (fname):
    with open(fname) as f:
        for i, l in enumerate(f):
            pass
    return i + 1


archivo = sys.argv[1]    #Argumento con el nombre del archivo a analizar
dur = sys.argv[3]        #Argumento con la duracion en segundos (por ej 5)
umb = sys.argv[2]        #Argumento con el umbral en dBm (por ej -70)

umbral = int(umb)   #Umbral de RSSI

file = open(archivo, "r")    #abro el archivo para leerlo
flen= file_len (archivo)     #calculo el largo del archivo

DireccionesMAC = []
DireccionesMACRand = []
RSSI = True
MACtiempos = []
tiempos_ini = []
tiempos_fin = []

# Recorro el archivo y obtengo la cantidad de direcciones MAC tanto reales
    como random
for i in range(flen):    #recorro el archivo linea por linea
    linea = file.readline()

    if len(linea) == 16:    #verifico que sea una linea de tiempo
        time = linea[:15]     #Guardo el tiempo del paquete

    elif "dBm" in linea:    #verifico que sea una linea de RSSI
        if int(linea[:-4]) >= umbral:
            RSSI = True
```

```
            else:
                RSSI = False

        elif RSSI:    #analizo el paquete si tiene un RSSI mayor que el umbral
            if len(linea) == 18:  #verifico que sea una direccion MAC
            aux = linea[0:2]
            aux_bin = bin(int(aux,16))[2:].zfill(8)
            aux_bin = aux_bin[6]

            if aux_bin == '1':    #verifico que la MAC sea random
              DirMACRand = linea[:17]
              DireccionesMACRand.append(DirMACRand) #lista con las
                  direcciones MAC random (puede haber repetidas)

                  if linea[:17] in MACtiempos:
                      tiempos_fin[MACtiempos.index(linea[:17])] =
                          time_conv(time) -
                          tiempos_ini[MACtiempos.index(linea[:17])]
                  else:
                      tiempos_ini.append(time_conv(time))
                      tiempos_fin.append(0)
                      MACtiempos.append(linea[:17])

            DireccionesMAC.append(linea[:17])  #lista con todas las direcciones
                MAC (puede haber repetidas)

Counter = set(DireccionesMAC)
CounterRand = set(DireccionesMACRand)

file.close()      #Cierro el archivo

# Resultados
duracion = float(dur)
contador_random = 0
for mac in CounterRand:
    if (tiempos_fin[MACtiempos.index(mac)] > duracion):
        contador_random = contador_random + 1

print "Cantidad con umbral" + umb + "dBm y duracion > " + dur + ": "
    + str(contador_random + (len(Counter)-len(CounterRand)) )
```

Apéndice C

Resultados completos

C.1. Ultrasonido

	True			**Estimación**		
	UP	**DOWN**	**Ocup.**	**UP**	**DOWN**	**Ocup.**
Man_1_1	4	0	6	3	1	4
Man_1_2	2	0	8	2	0	6
Man_1_3	2	0	10	2	0	8
Man_1_4	4	0	14	2	0	10
Man_1_5	0	2	12	0	1	9
Man_1_6	0	3	9	2	1	10
Man_1_7	0	3	6	0	3	7
Man_1_8	0	2	4	0	2	5
Man_1_9	0	2	2	0	2	3
Man_1_10	3	0	5	1	1	3
Man_1_11	2	0	7	2	0	5
Man_1_12	1	1	7	0	0	5
Pearson	-	-	-	0,73	0,74	0,86
RMSE	-	-	-	1,08	0,82	2,10
Man_2_1	2	2	6	1	1	6
Man_2_2	2	2	6	1	3	4
Man_2_3	2	0	8	2	0	6
Man_2_4	0	2	6	0	1	5
Man_2_5	2	3	5	2	3	4
Man_2_6	2	1	6	2	1	5
Man_2_7	0	1	5	0	0	5
Man_2_8	2	0	7	1	0	6
Man_2_9	2	0	9	2	0	8
Man_2_10	1	4	6	1	2	7
Man_2_11	1	2	5	1	1	7
Pearson	-	-	-	0,83	0,78	0,54

RMSE	-	-	-	0,52	0,90	1,28
Man_3_1	0	3	15	0	3	15
Man_3_2	0	3	12	0	1	14
Man_3_3	0	3	9	0	2	12
Man_3_4	1	3	7	0	1	11
Man_3_5	1	4	4	1	3	9
Man_3_6	1	0	5	1	0	10
Man_3_7	3	0	8	3	0	13
Man_3_8	3	1	10	3	1	15
Man_3_9	1	2	9	1	2	14
Man_3_10	2	4	7	1	3	12
Man_3_11	1	0	8	2	0	14
Man_3_12	0	5	3	0	4	10
Pearson	-	-	-	0,89	0,90	0,86
RMSE	-	-	-	0,50	1,00	4,69
Man_4_1	4	0	6	4	1	5
Man_4_2	3	0	9	3	0	8
Man_4_3	0	3	6	0	2	6
Man_4_4	4	2	8	4	2	8
Man_4_5	0	4	4	0	3	5
Man_4_6	2	2	4	2	2	5
Man_4_7	2	0	6	1	1	5
Man_4_8	3	1	8	1	1	5
Man_4_9	2	0	10	2	0	7
Man_4_10	0	2	8	0	2	5
Man_4_11	2	0	10	1	0	6
Pearson	-	-	-	0,90	0,92	0,58
RMSE	-	-	-	0,74	0,60	2,09
Man_5_1	4	0	6	4	0	6
Man_5_2	2	0	8	1	0	7
Man_5_3	2	0	10	2	0	9
Man_5_4	3	0	13	3	0	12
Man_5_5	0	3	10	0	3	9
Man_5_6	0	3	7	1	1	9
Man_5_7	0	1	6	0	1	8
Man_5_8	0	4	2	0	2	6
Man_5_9	3	0	5	3	0	9
Man_5_10	4	0	9	2	0	11
Man_5_11	2	1	10	1	1	11
Pearson	-	-	-	0,87	0,87	0,77
RMSE	-	-	-	0,80	0,85	2,11
Man_6_1	0	3	9	1	1	12
Man_6_2	1	2	8	1	2	11

Man_6_3	1	2	7	1	2	10
Man_6_4	2	2	7	1	2	9
Man_6_5	2	0	9	1	0	10
Man_6_6	3	2	10	2	4	8
Man_6_7	1	1	10	1	1	8
Man_6_8	1	6	5	1	3	6
Man_6_9	2	0	7	0	1	5
Man_6_10	2	2	7	2	2	5
Man_6_11	2	1	8	2	1	6
Pearson	-	-	-	0,39	0,59	0,41
RMSE	-	-	-	0,85	1,28	2,20
Tar_1_1	0	3	12	0	2	13
Tar_1_2	0	3	9	1	2	12
Tar_1_3	0	2	7	0	1	11
Tar_1_4	0	2	5	1	1	11
Tar_1_5	3	4	4	4	2	13
Tar_1_6	4	0	8	4	0	17
Tar_1_7	2	2	8	0	2	15
Tar_1_8	2	3	7	0	3	12
Tar_1_9	2	2	7	0	3	9
Tar_1_10	0	2	5	0	1	8
Pearson	-	-	-	0,68	0,65	0,36
RMSE	-	-	-	1,22	1,00	5,58
Tar_2_1	4	3	10	3	4	8
Tar_2_2	2	2	10	0	1	7
Tar_2_3	0	5	5	0	3	4
Tar_2_4	0	3	2	0	1	3
Tar_2_5	2	1	3	1	0	4
Tar_2_6	2	2	3	0	3	1
Tar_2_7	6	3	6	2	5	-2
Tar_2_8	2	1	7	2	1	-1
Tar_2_9	1	3	5	1	2	-2
Tar_2_10	2	2	5	0	1	-3
Pearson				0,68	0,56	0,42
RMSE				1,73	1,34	5,11
Tar_3_1	0	1	5	0	1	5
Tar_3_2	0	3	2	0	3	2
Tar_3_3	2	1	3	2	1	3
Tar_3_4	5	0	8	5	0	8
Tar_3_5	1	1	8	1	0	9
Tar_3_6	3	0	11	2	0	11
Tar_3_7	1	4	8	1	2	10
Tar_3_8	0	3	5	0	2	8

Tar_3_9	1	2	4	1	0	9
Tar_3_10	0	1	3	0	1	8
Pearson	-	-	-	0,98	0,78	0,75
RMSE	-	-	-	0,32	1,00	2,53
Tar_4_1	1	2	2	1	2	2
Tar_4_2	1	0	3	1	0	3
Tar_4_3	4	1	6	1	1	3
Tar_4_4	6	2	10	4	1	6
Tar_4_5	0	1	9	0	1	5
Tar_4_6	0	3	6	0	2	3
Tar_4_7	1	3	4	1	3	1
Tar_4_8	2	1	5	1	1	1
Tar_4_9	1	1	5	0	1	0
Tar_4_10	3	1	7	1	2	-1
Pearson	-	-	-	0,86	0,81	0,51
RMSE	-	-	-	1,38	0,55	4,05
Tar_5_1	0	2	5	0	2	5
Tar_5_2	0	4	1	0	2	3
Tar_5_3	1	0	2	1	0	4
Tar_5_4	3	0	5	0	0	4
Tar_5_5	2	1	6	2	1	5
Tar_5_6	4	0	10	3	0	8
Tar_5_7	0	2	8	0	1	7
Tar_5_8	4	0	12	3	1	9
Tar_5_9	1	0	13	1	0	10
Pearson	-	-	-	0,79	0,85	0,98
RMSE	-	-	-	1,11	0,82	1,91
Tar_6_1	0	4	11	0	3	12
Tar_6_2	0	3	8	0	1	11
Tar_6_3	0	3	5	0	2	9
Tar_6_4	0	4	1	0	4	5
Tar_6_5	4	0	5	3	0	8
Tar_6_6	3	0	8	1	0	9
Tar_6_7	3	0	11	3	0	12
Tar_6_8	1	3	9	1	2	11
Pearson	-	-	-	0,91	0,91	0,96
RMSE	-	-	-	0,79	0,94	2,67

Tabla C.1: Resultados obtenidos con el sistema de sensores

C.2. Cámara

	True			Obj Det Mod. (30)			Obj Det Mod. (15)		
	UP	**DOWN**	**Ocup.**	**UP**	**DOWN**	**Ocup.**	**UP**	**DOWN**	**Ocup.**
Man_1_1	4	0	6	3	0	5	2	0	4
Man_1_2	2	0	8	0	0	5	0	0	4
Man_1_3	2	0	10	1	0	6	1	0	5
Man_1_4	4	0	14	2	0	8	2	0	7
Man_1_5	0	2	12	0	0	8	0	2	5
Man_1_6	0	3	9	0	2	6	0	3	2
Man_1_7	0	3	6	0	2	4	0	2	0
Man_1_8	0	2	4	0	2	2	0	1	-1
Man_1_9	0	2	2	0	2	0	0	2	-3
Man_1_10	3	0	5	3	0	3	3	0	0
Man_1_11	2	0	7	2	0	5	2	0	2
Man_1_12	1	1	7	2	0	7	2	0	4
Pearson	-	-	-	0,82	0,84	0,91	0,77	0,93	0,89
RMSE	-	-	-	0,96	0,76	2,99	1,08	0,50	5,30
Man_2_1	2	2	6	2	1	7	2	2	6
Man_2_2	2	2	6	1	2	6	1	2	5
Man_2_3	2	0	8	2	0	8	1	0	6
Man_2_4	0	2	6	0	1	7	0	1	5
Man_2_5	2	3	5	1	3	5	1	3	3
Man_2_6	2	1	6	0	0	5	2	1	4
Man_2_7	0	1	5	0	0	5	0	1	3
Man_2_8	2	0	7	0	0	5	0	0	3
Man_2_9	2	0	9	2	0	7	1	0	4
Man_2_10	1	4	6	0	1	6	1	2	3
Man_2_11	1	2	5	1	1	6	1	1	3
Pearson	-	-	-	0,55	0,72	0,58	0,60	0,86	0,37
RMSE	-	-	-	1,00	1,13	1,04	0,85	0,74	2,56
Man_3_1	0	3	15	0	3	15	0	3	15
Man_3_2	0	3	12	0	3	12	0	2	13
Man_3_3	0	3	9	0	1	11	0	1	12
Man_3_4	1	3	7	1	2	10	1	2	11
Man_3_5	1	4	4	2	1	11	2	2	11
Man_3_6	1	0	5	1	0	12	1	0	12
Man_3_7	3	0	8	0	0	12	0	0	12
Man_3_8	3	1	10	3	1	14	3	2	13
Man_3_9	1	2	9	1	0	15	1	1	13
Man_3_10	2	4	7	1	1	15	1	1	13
Man_3_11	1	0	8	1	0	16	1	0	14
Man_3_12	0	5	3	0	2	14	0	2	12
Pearson	-	-	-	0,55	0,63	0,26	0,55	0,68	0,73
RMSE	-	-	-	0,96	1,73	5,97	0,96	1,58	5,15
Man_4_1	4	0	6	3	0	5	3	0	5

Man_4_2	3	0	9	3	0	8	2	0	7
Man_4_3	0	3	6	0	2	6	0	2	5
Man_4_4	4	2	8	3	1	8	4	0	9
Man_4_5	0	4	4	0	0	8	0	3	6
Man_4_6	2	2	4	2	1	9	2	2	6
Man_4_7	2	0	6	2	0	11	2	0	8
Man_4_8	3	1	8	1	1	11	3	1	10
Man_4_9	2	0	10	1	0	12	2	0	12
Man_4_10	0	2	8	0	2	10	0	1	11
Man_4_11	2	0	10	2	0	12	1	0	12
Pearson	-	-	-	0,89	0,53	0,52	0,95	0,88	0,79
RMSE	-	-	-	0,80	1,31	2,84	0,52	0,80	1,91
Man_5_1	4	0	6	3	0	5	4	0	6
Man_5_2	2	0	8	2	0	7	1	0	7
Man_5_3	2	0	10	1	0	8	2	0	9
Man_5_4	3	0	13	3	0	11	3	0	12
Man_5_5	0	3	10	0	1	10	0	2	10
Man_5_6	0	3	7	0	2	8	0	2	8
Man_5_7	0	1	6	0	2	6	0	4	4
Man_5_8	0	4	2	0	3	3	0	2	2
Man_5_9	3	0	5	3	0	6	2	0	4
Man_5_10	4	0	9	2	0	8	2	0	6
Man_5_11	2	1	10	2	1	9	2	1	7
Pearson	-	-	-	0,91	0,87	0,95	0,91	0,64	0,91
RMSE	-	-	-	0,74	0,80	1,17	0,74	1,17	1,57
Man_6_1	0	3	9	0	2	10	0	3	9
Man_6_2	1	2	8	0	1	9	0	1	8
Man_6_3	1	2	7	1	2	8	1	3	6
Man_6_4	2	2	7	0	2	6	0	2	4
Man_6_5	2	0	9	1	0	7	1	0	5
Man_6_6	3	2	10	3	1	9	3	1	7
Man_6_7	1	1	10	2	2	9	2	2	7
Man_6_8	1	6	5	2	1	10	1	3	5
Man_6_9	2	0	7	3	0	13	2	0	7
Man_6_10	2	2	7	1	1	13	2	1	8
Man_6_11	2	1	8	1	0	14	2	0	10
Pearson	-	-	-	0,48	0,37	-0,19	0,65	0,73	0,32
RMSE	-	-	-	1,00	1,68	3,61	0,80	1,17	2,11
Tar_1_1	0	3	12	0	2	13	0	1	14
Tar_1_2	0	3	9	0	2	11	0	2	12
Tar_1_3	0	2	7	0	2	9	0	2	10
Tar_1_4	0	2	5	0	1	8	0	1	9
Tar_1_5	3	4	4	2	2	8	2	4	7

Tar_1_6	4	0	8	3	0	11	3	0	10
Tar_1_7	2	2	8	3	1	13	1	1	10
Tar_1_8	2	3	7	2	2	13	2	2	10
Tar_1_9	2	2	7	2	2	13	1	2	9
Tar_1_10	0	2	5	0	2	11	0	1	8
Pearson	-	-	-	0,93	0,78	0,63	0,97	0,80	0,96
RMSE	-	-	-	0,55	0,95	4,20	0,63	0,95	2,77
Tar_2_1	4	3	10	4	3	10	4	4	9
Tar_2_2	2	2	10	2	0	12	2	1	10
Tar_2_3	0	5	5	0	2	10	1	3	8
Tar_2_4	0	3	2	0	2	8	0	3	5
Tar_2_5	2	1	3	2	1	9	1	2	4
Tar_2_6	2	2	3	1	0	10	1	2	3
Tar_2_7	6	3	6	6	1	15	5	3	5
Tar_2_8	2	1	7	1	2	14	2	2	5
Tar_2_9	1	3	5	1	2	13	1	2	4
Tar_2_10	2	2	5	1	2	12	2	2	4
Pearson	-	-	-	0,97	0,34	0,41	0,94	0,56	0,78
RMSE	-	-	-	0,55	1,55	6,27	0,63	0,95	1,64
Tar_3_1	0	1	5	0	1	5	0	1	5
Tar_3_2	0	3	2	0	3	2	0	3	2
Tar_3_3	2	1	3	3	0	5	3	0	5
Tar_3_4	5	0	8	4	0	9	2	0	7
Tar_3_5	1	1	8	1	1	9	1	1	7
Tar_3_6	3	0	11	3	0	12	3	0	10
Tar_3_7	1	4	8	1	3	10	1	3	8
Tar_3_8	0	3	5	0	1	9	0	2	6
Tar_3_9	1	2	4	2	1	10	1	1	6
Tar_3_10	0	1	3	0	1	9	0	1	5
Pearson	-	-	-	0,94	0,85	0,71	0,78	0,93	0,92
RMSE	-	-	-	0,55	0,84	3,15	1,00	0,63	1,26
Tar_4_1	1	2	2	1	1	3	0	1	2
Tar_4_2	1	0	3	1	0	4	1	0	3
Tar_4_3	4	1	6	4	1	7	3	1	5
Tar_4_4	6	2	10	6	1	12	5	2	8
Tar_4_5	0	1	9	0	1	11	0	1	7
Tar_4_6	0	3	6	0	1	10	0	2	5
Tar_4_7	1	3	4	1	0	11	1	2	4
Tar_4_8	2	1	5	2	1	12	2	1	5
Tar_4_9	1	1	5	1	1	12	1	1	5
Tar_4_10	3	1	7	0	1	11	3	1	7
Pearson	-	-	-	0,88	0,00	0,65	0,97	0,90	0,97
RMSE	-	-	-	0,95	1,22	4,36	0,55	0,55	1,00

Tar_5_1	0	2	5	0	1	6	0	2	5
Tar_5_2	0	4	1	0	4	2	0	2	3
Tar_5_3	1	0	2	0	0	2	1	0	4
Tar_5_4	3	0	5	4	0	6	2	0	6
Tar_5_5	2	1	6	2	1	7	2	1	7
Tar_5_6	4	0	10	3	0	10	2	0	9
Tar_5_7	0	2	8	0	0	10	0	1	8
Tar_5_8	4	0	12	5	0	15	4	0	12
Tar_5_9	1	0	13	0	0	15	1	0	13
Pearson	-	-	-	0,93	0,87	0,99	0,91	0,92	0,98
RMSE	-	-	-	0,75	0,75	1,53	0,75	0,75	1,11
Tar_6_1	0	4	11	0	4	11	0	3	12
Tar_6_2	0	3	8	0	3	8	0	3	9
Tar_6_3	0	3	5	0	1	7	0	2	7
Tar_6_4	0	4	1	0	3	4	0	2	5
Tar_6_5	4	0	5	4	0	8	4	0	9
Tar_6_6	3	0	8	3	0	11	3	0	12
Tar_6_7	3	0	11	2	0	13	2	0	14
Tar_6_8	1	3	9	1	3	11	1	3	12
Pearson	-	-	-	0,98	0,91	0,93	0,98	0,92	0,93
RMSE	-	-	-	0,35	0,79	2,21	0,35	0,87	3,00

Tabla C.2: Resultados obtenidos con el algoritmo Object Detection Modificado

C.3. WiFi

		Minuto									
		1	2	3	4	5	6	7	8	9	10
Man_1	Est	2	3	4	6	5	6	3	5	6	3
	Real	2	7	7	12	10	8	3	2	4	6
Man_2	Est	2	7	4	0	2	3	2	4	6	5
	Real	5	3	7	6	5	5	4	7	5	5
Man_3	Est	9	10	5	6	8	3	5	9	10	7
	Real	11	9	8	7	4	6	7	7	6	3
Man_4	Est	4	6	5	4	4	3	5	3	3	5
	Real	2	6	4	6	3	3	4	7	5	7
Man_5	Est	4	6	4	5	3	4	3	4	5	3
	Real	5	7	9	12	12	7	6	2	5	7
Man_6	Est	1	7	5	4	3	6	6	6	7	6
	Real	8	7	6	4	3	6	4	5	5	7
Tar_1	Est	7	18	9	6	3	5	6	4	4	3
	Real	10	8	6	5	3	6	6	5	5	4
Tar_2	Est	3	11	6	8	7	11	8	6	5	5
	Real	3	7	4	2	3	2	3	3	2	3
Tar_3	Est	1	5	6	6	6	9	8	8	5	3
	Real	3	2	3	7	7	8	5	3	4	3
Tar_4	Est	4	7	7	5	5	7	8	5	2	6
	Real	3	3	4	7	7	6	4	5	5	5
Tar_5	Est	5	11	7	3	2	3	8	6	4	5
	Real	6	1	2	4	5	8	7	9	10	11
Tar_6	Est	3	12	10	6	4	1	3	2	3	5
	Real	12	8	8	1	3	5	8	8	8	8

Tabla C.3: Resultados del modelo continuo para la segunda etapa de la base de datos.

			Estimación			Porcentaje		
		Disp.	**-75dBm**	**-70dBm**	**-65dBm**	**-75dBm**	**-70dBm**	**-65dBm**
Man_1	1	7	7	7	4	100 %	100 %	57 %
	2	12	10	10	9	83 %	83 %	75 %
	3	12	9	7	7	75 %	58 %	58 %
	4	8	7	7	7	88 %	88 %	88 %
	5	7	4	4	4	57 %	57 %	57 %
Man_2	1	7	5	5	4	71 %	71 %	57 %
	2	7	7	6	5	100 %	86 %	71 %
	3	9	9	7	5	100 %	78 %	56 %
	4	8	7	5	5	88 %	63 %	63 %
	5	9	7	7	4	78 %	78 %	44 %
Man_3	1	14	13	11	9	93 %	79 %	64 %
	2	10	8	8	8	80 %	80 %	80 %
	3	10	9	8	8	90 %	80 %	80 %
	4	9	8	8	7	89 %	89 %	78 %
	5	9	6	5	5	67 %	56 %	56 %
Man_4	1	6	7	6	6	117 %	100 %	100 %
	2	10	9	8	7	90 %	80 %	70 %
	3	9	8	7	5	89 %	78 %	56 %
	4	9	5	4	4	56 %	44 %	44 %
	5	10	10	9	8	100 %	90 %	80 %
Man_5	1	7	11	8	7	157 %	114 %	100 %
	2	12	12	11	9	100 %	92 %	75 %
	3	12	9	8	5	75 %	67 %	42 %
	4	7	7	6	4	100 %	86 %	57 %
	5	8	10	10	10	125 %	125 %	125 %
Man_6	1	10	6	6	6	60 %	60 %	60 %
	2	8	8	8	7	100 %	100 %	88 %
	3	9	11	9	9	122 %	100 %	100 %
	4	8	11	8	7	138 %	100 %	88 %
	5	9	10	8	6	111 %	89 %	67 %
Tar_1	1	12	14	12	11	117 %	100 %	92 %
	2	8	13	11	11	163 %	138 %	138 %
	3	10	11	7	6	110 %	70 %	60 %
	4	9	9	8	7	100 %	89 %	78 %
	5	7	10	7	6	143 %	100 %	86 %
Tar_2	1	11	12	11	10	109 %	100 %	91 %
	2	7	19	13	9	271 %	186 %	129 %
	3	5	10	8	6	200 %	160 %	120 %
	4	6	9	7	6	150 %	117 %	100 %
	5	6	10	9	8	167 %	150 %	133 %
Tar_3	1	4	9	6	3	225 %	150 %	75 %
	2	8	15	12	9	188 %	150 %	113 %
	3	9	18	15	11	200 %	167 %	122 %

	4	9	15	8	8	167 %	89 %	89 %
	5	4	12	11	5	300 %	275 %	125 %
Tar_4	1	5	8	7	6	160 %	140 %	120 %
	2	9	13	9	6	144 %	100 %	67 %
	3	8	14	12	6	175 %	150 %	75 %
	4	9	11	9	6	122 %	100 %	67 %
	5	7	13	10	7	186 %	143 %	100 %
Tar_5	1	6	11	9	8	183 %	150 %	133 %
	2	4	11	7	5	275 %	175 %	125 %
	3	9	9	7	4	100 %	78 %	44 %
	4	10	13	9	6	130 %	90 %	60 %
	5	11	15	11	9	136 %	100 %	82 %
Tar_6	1	12	12	10	9	100 %	83 %	75 %
	2	8	15	10	9	188 %	125 %	113 %
	3	5	6	5	4	120 %	100 %	80 %
	4	9	10	8	6	111 %	89 %	67 %
	5	8	13	10	8	163 %	125 %	100 %

Tabla C.4: Resultados del modelo discreto para la segunda etapa de la base de datos.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice D

Indicadores

Coeficiente de correlación de Pearson

El coeficiente de correlación de Pearson es una medida de la correlación lineal entre dos variables X e Y. Los valores posibles que puede tomar se encuentran entre -1 y 1, siendo 1 una correlación lineal positiva total, -1 una correlación lineal negativa total y 0 que no existe correlación lineal. Se define como la covarianza de las dos variables dividida por el producto de sus desviaciones estándar, lo cual se puede escribir como se observa en la ecuación D.1.

$$\rho_{xy} = \frac{N\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{N\sum x_i^2 - (\sum x_i)^2}\sqrt{N\sum y_i^2 - (\sum y_i)^2}} \tag{D.1}$$

Siendo:

- N el número total de muestras.
- x_i e y_i los valores que toman las variables X e Y respectivamente.

Una propiedad matemática fundamental del coeficiente de correlación de Pearson es que es invariante bajo distintos cambios en posición y escala de las variables. Esto quiere decir que si la variable X es transformada por $aX + b$ e Y por $cY + d$, el valor de ρ_{xy} no cambia, siendo a y c mayores a 0.

Un $\rho_{xy} = 1$ implica que una ecuación lineal describe la relación entre X e Y perfectamente, quedando todos los puntos sobre una linea, de modo que a medida que X crece también lo hace Y. Si $\rho_{xy} = -1$ también implica que la relación entre X e Y es descrita perfectamente por una ecuación lineal, pero ahora a medida que X crece Y decrece. Si $\rho_{xy} = 0$ entonces no existe una correlación lineal entre las variables X e Y. Esto se puede observar en la figura D.1.

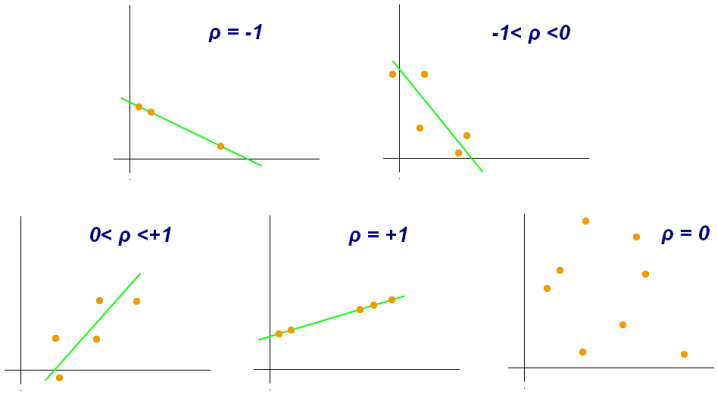


Figura D.1: Distintos valores de coeficientes de correlación de Pearson.[1]

RMSE (Root Mean Square Error)

El RMSE generalmente se utiliza para medir la diferencia entre los valores estimados u obtenidos mediante un modelo, con los valores observados. El RMSE es una medida de exactitud para comparar errores entre distintos modelos implementados sobre un mismo conjunto de datos, dado que es dependiente de la escala. Realizar comparaciones entre distintos conjuntos de datos no es válido ya que la medida depende de la escala de los números usados. Cuanto menor sea el RMSE más cercanos serán los valores del modelo a los valores observados.

El RMSE es la raíz cuadrada del promedio de los errores cuadráticos, como se puede observar en la ecuación D.2. El efecto de cada error sobre el RMSE depende del valor al cuadrado, por lo tanto errores grandes producen un efecto considerablemente grande sobre el RMSE. Es por esto que el RMSE es sensible a valores atípicos.

$$RMSE = \sqrt{\frac{\sum_{n=1}^{N}(\hat{y}_n - y_n)^2}{N}} \tag{D.2}$$

Siendo:

- N el número total de muestras.
- y_n los valores observados.
- $\hat{y}_n$ los valores estimados.

P.O. (Porcentaje de Ocupación)

El porcentaje de ocupación es un promedio de los porcentajes de estimación en comparación con el valor real. Este parámetro da una idea de cuán acertadas son las estimaciones realizadas. Se calcula de la siguiente manera:

[1]Imagen extraída de: `https://es.wikipedia.org/wiki/Coeficiente_de_correlaci%C3%B3n_de_Pearson#/media/File:Correlation_coefficient.png`

$$P.O.(\%) = \frac{1}{N} \sum_{i=1}^{i=N} \frac{x_i}{y_i} * 100 \tag{D.3}$$

Siendo:

- N la cantidad de eventos
- x_i los resultados estimados por evento
- y_i los valores observados por evento

Esta página ha sido intencionalmente dejada en blanco.

Apéndice E

Comunicación

Un aspecto importante a la hora de desarrollar prototipos para el conteo de personas en el medio de transporte público es el envío de los datos. El prototipo no sólo debe tener la posibilidad de medir, independientemente de la tecnología, sino que también debe tener una forma de enviar los resultados de la estimación.

Para esto se resolvió conectar los sensores mediante 3G a internet, utilizando un router TP-LINK 3020 y un módem HUAWEI E3131. Esto se realiza de la siguiente manera: se crea una WLAN con la que se conectan los distintos dispositivos al router, y éste tiene acceso a internet mediante una conexión 3G a partir del módem. La siguiente figura muestra una ilustración de la conexión de los prototipos a internet.

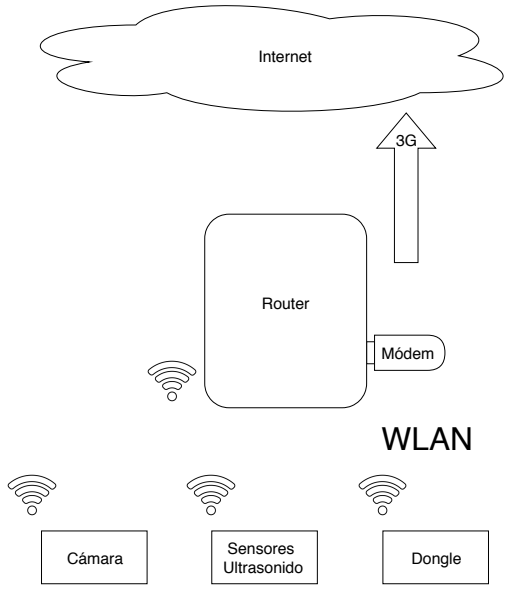


Figura E.1: Esquema de la conexión de las tecnologías a Internet.

Una vez resuelta la conectividad es necesario contar con una plataforma que

permita recibir los resultados y enviarlos a donde corresponda. Para esto la opción analizada es la plataforma de ciudades inteligentes llamada Fiware utilizada hoy en día en varias ciudades, entre ellas la ciudad de Montevideo. Se utiliza el Fiware-Orion [1] que implementa una API que permite el manejo de suscripciones, registros, consultas y actualizaciones. El Orion trabaja en base a un mecanismo de tipo "publish/subscribe" que es una característica especialmente diseñada para tiempo real. De esta forma se puede saber cuando hay datos nuevos de estimación de ocupación en forma fiable y efectiva sin tener que estar haciendo consultas aleatorias en ciertos períodos de tiempo con alto riesgo de perder información. Los sensores deberían enviar los datos mediante el protocolo HTTP con el método POST al Orion cada vez que tengan una actualización en la estimación (cada un cierto tiempo o luego de cada parada). Del otro lado quien desee recibir esta información debe subscribirse a este canal también utilizando el protocolo HTTP con el método POST. Luego cada vez que exista una actualización de la información de ese canal el Orion la enviaría a cada suscriptor.

[1] `https://fiware-orion.readthedocs.io/en/latest/index.html`

Apéndice F

Conexión de sensores de ultrasonido

En este apéndice se muestra como hacer la conexión para que los sensores de ultrasonido de la marca Arduino funcionen en la unidad de procesamiento Raspberry Pi.

Los sensores HC-SR04 y HY-SRF05 son sensores que funcionan con una tensión VCC de 5V. En ese caso la pata marcada con las letras VCC debe ir conectada al pin de la Raspberry Pi que marca esta tensión. Luego para tener un mismo punto de referencia de tensión la pata maracada en el sensor como GND debe ser conectada al pin GND de la Raspberry Pi. Siguiendo con las conexiones, la pata TRIG debe ir conectada directamente a uno de los pines GPIO de la unidad, estos pines sirven tanto de entrada como de salida para la Raspberry Pi. En este caso se usará como una salida para dar el pulso de Trigger necesario para que el sensor envié el pulso de ultrasonido. Por último se debe conectar la pata ECHO del sensor. En la figura F.1[1] se puede ver que entre la pata ECHO y la Raspberry Pi hay un divisor de tensión, esto se debe a que el sensor por la pata ECHO devuelve una señal de 5V y la Raspberry Pi en sus pines GPIO trabaja con señales de 3,3V. Los sensores HY-SRF05 cuentan con un pata más llamada OUT, que para esta aplicación debe quedar desconectada. Luego para hacer funcionar el sensor es necesario enviar los comandos necesarios a través de la Raspberry Pi. En este proyecto se utilizó un script en Python 2.7 como se muestra en el apéndice B

[1]Imagen extraída de: `https://robologs.net/2015/07/31/tutorial-de-raspberry-pi-y-hc-sr04/`

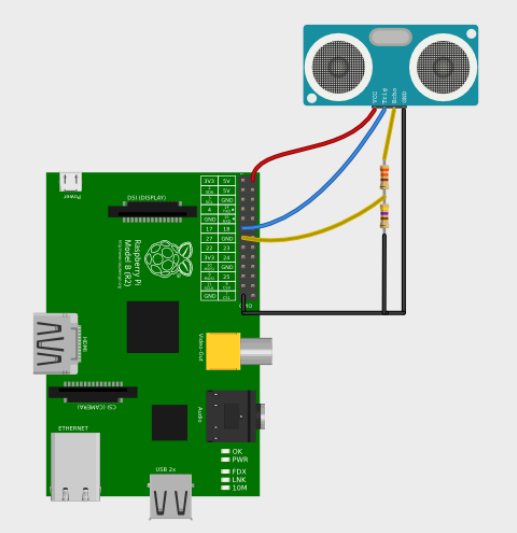


Figura F.1: Conexión de sensor de ultrasonido Arduino con Raspberry Pi.

Referencias

[1] Ulf Lindholm. Apparatus for the automatic counting of passengers, February 22 1977. US Patent 4,009,389.

[2] Laura Eboli and Gabriella Mazzulla. A methodology for evaluating transit service quality based on subjective and objective measures from the passenger's point of view. *Transport Policy*, 18(1):172–181, 2011.

[3] Neal Lathia, Jon Froehlich, and Licia Capra. Mining public transport usage for personalised intelligent transport systems. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 887–892. IEEE, 2010.

[4] F Wahl, M Milenkovic, and O Amft. A green autonomous self-sustaining sensor node for counting people in office environments. In *Education and Research Conference (EDERC), 2012 5th European DSP*, pages 203–207. IEEE, 2012.

[5] Jeong Woo Choi, Xuanjun Quan, and Sung Ho Cho. Bi-directional passing people counting system based on ir-uwb radar sensors. *IEEE Internet of Things Journal*, 5(2):512–522, 2018.

[6] Zheng Ma and Antoni B Chan. Counting people crossing a line using integer programming and local features. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(10):1955–1969, 2016.

[7] Chao-Ho Chen, Yin-Chan Chang, Tsong-Yi Chen, and Da-Jinn Wang. People counting system for getting in/out of a bus based on video processing. In *Intelligent Systems Design and Applications, 2008. ISDA'08. Eighth International Conference on*, volume 3, pages 565–569. IEEE, 2008.

[8] Jau-Woei Perng, Ting-Yen Wang, Ya-Wen Hsu, and Bing-Fei Wu. The design and implementation of a vision-based people counting system in buses. In *System Science and Engineering (ICSSE), 2016 International Conference on*, pages 1–3. IEEE, 2016.

[9] Jianzhao Cao, Liangliang Sun, Manfred Gilbert Odoom, Fangjun Luan, and Xiaoyu Song. Counting people by using a single camera without calibration. In *Control and Decision Conference (CCDC), 2016 Chinese*, pages 2048–2051. IEEE, 2016.

[10] Pakorn KaewTraKulPong and Richard Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Video-based surveillance systems*, pages 135–144. Springer, 2002.

[11] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 28–31. IEEE, 2004.

[12] Zoran Zivkovic and Ferdinand Van Der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern recognition letters*, 27(7):773–780, 2006.

[13] Andrew B Godbehere, Akihiro Matsukawa, and Ken Goldberg. Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation. In *American Control Conference (ACC), 2012*, pages 4305–4312. IEEE, 2012.

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

[15] Timo Ojala, Matti Pietikainen, and Topi 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, 2002.

[16] Lingbo Liu, Hongjun Wang, Guanbin Li, Wanli Ouyang, and Liang Lin. Crowd counting using deep recurrent spatial-aware network. *arXiv preprint arXiv:1807.00601*, 2018.

[17] George Papandreou, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, Chris Bregler, and Kevin Murphy. Towards accurate multi-person pose estimation in the wild. In *CVPR*, volume 3, page 6, 2017.

[18] Global System for Mobile Communications. The mobile economy 2018. 2018.

[19] Samy El-Tawab, Raymond Oram, Michael Garcia, Chris Johns, and B Brian Park. Data analysis of transit systems using low-cost iot technology. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2017 IEEE International Conference on*, pages 497–502. IEEE, 2017.

[20] Woramate Pattanusorn, Itthisek Nilkhamhang, Somsak Kittipiyakul, Kittipong Ekkachai, and Atsushi Takahashi. Passenger estimation system using wi-fi probe request. In *Information and Communication Technology for Embedded Systems (IC-ICTES), 2016 7th International Conference of*, pages 67–72. IEEE, 2016.

[21] Sangho Shin, Andrea G Forte, Anshuman Singh Rawat, and Henning Schulzrinne. Reducing mac layer handoff latency in ieee 802.11 wireless lans. In *Proceedings of the second international workshop on Mobility management & wireless access protocols*, pages 19–26. ACM, 2004.

[22] Jerome Henry. *Optimise your Wi-Fi Network for iPhones and iPads, Cisco Live! Conference, Melbourne, Australia*. 2017.

[23] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S Cardoso, and Frank Piessens. Why mac address randomization is not enough: An analysis of wi-fi network discovery mechanisms. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 413–424. ACM, 2016.

[24] Julien Freudiger. How talkative is your mobile device?: an experimental study of wi-fi probe requests. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, page 8. ACM, 2015.

[25] Matthew Dunlap, Zhibin Li, Kristian Henrickson, and Yinhai Wang. Estimation of origin and destination information from bluetooth and wi-fi sensing for transit. *Transportation Research Record: Journal of the Transportation Research Board*, (2595):11–17, 2016.

[26] Ali Haghani, Masoud Hamedi, Kaveh Sadabadi, Stanley Young, and Philip Tarnoff. Data collection of freeway travel time ground truth with bluetooth sensors. *Transportation Research Record: Journal of the Transportation Research Board*, (2160):60–68, 2010.

[27] Emanuele Frontoni, Adriano Mancini, Roberto Pierdicca, Mirco Sturari, and Primo Zingaretti. Analysing human movements at mass events: A novel mobile-based management system based on active beacons and avm. In *Control and Automation (MED), 2016 24th Mediterranean Conference on*, pages 605–610. IEEE, 2016.

Esta página ha sido intencionalmente dejada en blanco.

Índice de tablas

Índice de figuras

Índice de figuras

Esta es la última página.
Compilado el miércoles 16 enero, 2019.
http://iie.fing.edu.uy/