Sistema de vigilancia para hogares de bajo costo con cámaras fijas y notificaciones a través de mensajería instantánea

Alumno: Lic. Carlos Sebastián Castañeda

Director: Dra. María José Abásolo

Trabajo Final presentado para obtener el grado de Especialista en Computación Gráfica, Imágenes y Visión por Computadora

Facultad de Informática, UNLP
Septiembre 2017

Agradecimientos

Este trabajo culmina 2 años de un gran esfuerzo, en los cuales decidí volver a la Facultad luego de 5 años de haber recibido mi título de grado. Llegar hasta aquí no podría haber sido posible sin la ayuda y apoyo de muchas personas que forman parte de mi vida.

En primer lugar, quiero agradecerle a mi directora de trabajo final, la Dra. Maria Jose Abasolo, que desde el primer dia que la contacte para explicarle mi idea me dio su apoyo, y me fue guiando y aconsejando durante todo el desarrollo de este trabajo.

A Emiliano Dalla Verde Marcozzi, quien en una charla de trabajo me recomendó comprar una cámara de seguridad y me sugirió la idea inicial de la que parte la inspiración para realizar este trabajo.

A todos y cada uno de los profesores que tuve a lo largo del Postgrado. De todos aprendí y me lleve algo. Realmente, hicieron que me sienta muy agradecido de haber vuelto al mundo Académico luego de 5 años.

A Paula Barbieri, por su ayuda y apoyo incondicional desde el primer dia en el que decidí inscribirme en este Posgrado.

Por último, no quiero dejar de agradecer a mi Familia y Amigos. Sin ellos todo sería mucho más complicado.

A todos, muchas gracias.

Lic. Carlos Sebastián Castañeda

Indice

Resumen 1

Capítulo 1 - Introducción 3

1.1 Motivación 3

1.2 Objetivos y contribución esperada 4

1.3 Organización del presente informe 5

Capítulo 2 - Detección de Movimiento en Video 7

2.1 Detección de movimiento y cámaras de seguridad 7

2.2 Método de Sustracción de fondo 8

2.3 Sustracción de fondo con OpenCV 10

2.3.1 Algoritmos implementados en la librería 10

2.3.2 Implementación con bajo costo computacional 12

Capítulo 3 - Detección de Personas en Imagenes y Video 15

3.1 Introducción a la detección de personas 15

3.2 Histograma de Gradientes Orientados 15

3.3 Detección de personas en OpenCV 20

3.3.1 Descripción del algoritmo 20

3.3.2 Parámetros del algoritmo 23

Capítulo 4 - Sistema de Vigilancia Analizado 27

4.1 Descripción de la Cámara TP-LINK nc220 27

4.2 Puntos débiles y falencias del sistema analizado 29

4.2.1 Configuración Web con soporte limitado en los navegadores Web más populares 29

4.2.2 Activación y desactivación de los eventos de detección de movimiento y sonido 30

4.2.3 Falsos positivos en la detección de movimientos 31

Capítulo 5 - Sistema de Vigilancia Mejorado 33

5.1 Objetivos 33

5.2 Descripción de la arquitectura 33

5.3 Componentes del Sistema 35

5.3.1 Aplicación WEB 35

5.3.2 Servidor FTP 36

5.3.3 Mensajería Instantánea 39

5.3.4 Detector de intrusos 45

5.4 Host para ejecutar la aplicación 47
5.5 Recursos necesarios y costos del sistema 48
Capítulo 6 - Pruebas Realizadas y Resultados Obtenidos 51
6.1 Usabilidad 51
6.1.1 Definición y evaluación 51
6.1.2 Prueba de usabilidad 52
6.2 Evaluación de la detección de movimiento embebida en la cámara 57
6.2.1 Área monitoreada en las pruebas 57
6.3 Evaluación de la detección de movimiento mejorada con el detector de intrusos 61
6.3.1 Ajuste de parámetros 61
6.3.2 Métricas utilizadas 63
6.3.3 Resultados obtenidos 64
6.3.4 Evaluación en tiempo real 66
Capítulo 7 - Conclusiones y Trabajos a Futuro 69
7.1 Conclusiones 69
7.2 Trabajos a futuro 70
Bibliografía 73

Resumen

Muchos de los dispositivos para monitoreo de video, actualmente disponibles en el mercado, cuentan con herramientas automáticas, para alertar ante eventos o posibles amenazas, entre las cuales se incluye software de visión por computadora.

En este trabajo, se evaluará las características y desempeño de una cámara de vigilancia de bajo costo que posee entre sus funciones la detección de movimiento en video. Se propone, para mejorar sus prestaciones generales, la utilización de una capa adicional de software que interactúa con el hardware y software provisto por el sistema comercial. El sistema implementado permite recibir las alertas enviadas por la cámara en una aplicación de mensajería instantánea, así como también la activación y desactivación de las alertas.

Por otra parte, se introducen la detección de movimiento en video y detección de personas en imagenes. Se estudiaron algoritmos para la detección de movimiento, profundizando en la técnica de sustracción de fondo, la cual es ampliamente utilizada en cámaras de seguridad y vigilancia. También, se estudiaron algoritmos para la detección de personas en imágenes, describiendo el método de histogramas de gradientes orientados. Se implementó una versión de este algoritmo utilizando la librería OpenCV. Este algoritmo, se incorporó al sistema implementado, para mejorar las prestación de la cámara comercial, de tal manera que al recibir una imagen se pueda analizar y en caso de detectar personas se aumente la imagen original encuadrando las personas detectadas y aumentado la semántica del alerta enviada a la aplicación de mensajería instantánea.

Por último, se realizaron diferentes pruebas con el objetivo de:

- Medir la eficacia del sistema de detección de movimiento provisto por el sistema comercial.
- Medir la exactitud y precisión del algoritmo de detección de personas implementado.
- Evaluar la usabilidad de los nuevos medios para activar el sistema de alarmas y recibir las notificaciones.

Los logros obtenidos fueron:

- Una mejora en la usabilidad general del sistema de vigilancia.
- Una mejor clasificación de las alertas enviadas por el sistema comercial.

Palabras clave: Servicios en la nube (Cloud services), Aplicaciones Web, Telegram Bots, Detección de movimiento, Videovigilancia, Detección de personas, Visión por computadora, Sistemas de vigilancia, Seguridad

Capítulo 1 - Introducción

1.1 Motivación

El monitoreo y control de diferentes áreas, tales como lugares públicos, bancos, centros comerciales, comercios, aeropuertos e incluso viviendas por medio de sistemas de seguridad; se ha vuelto una necesidad en estos tiempos. Existen diversos dispositivos para el control de seguridad como por ejemplo cámaras de video, alarmas y sensores de movimiento. En general, los sistemas de seguridad se componen de dispositivos de hardware y la interacción de personas físicas, que los controlan, supervisan y actúan ante alguna potencial amenaza. Con los avances tecnológicos en el área de seguridad y la disponibilidad de hardware de bajo costo, la utilización de este tipo de dispositivos ha dejado de ser propiedad exclusiva de grandes corporaciones, bancos, terminales aéreas o instituciones gubernamentales, para convertirse en algo de uso habitual en hogares y pequeños comercios. Hoy en día es común que una vivienda cuente con cámaras de seguridad, alarmas y sensores para detectar y disuadir potenciales actividades criminales que atenten contra la propiedad privada.

En particular, los sistemas de vigilancia a través de cámaras de video, son útiles para detectar actividades criminales, en aeropuertos, centros comerciales y estacionamientos entre otros. Por este motivo, están en constante desarrollo, siendo uno de los principales objetivos, contar con sistemas de vigilancia automatizados que necesiten poca interacción humana para su funcionamiento y detección de potenciales amenazas. En general, la instalacion de camaras de seguridad, no implica un gran gasto económico, pero si la contratación de personas que puedan monitorear esas cámaras durante las 24 horas del día [1].

Es posible adquirir cámaras de seguridad fijas, en el mercado local, como por ejemplo la cámara TP-LINK NC220 [2]. Estas cámaras tienen un costo relativamente bajo en relación a otros sistemas de seguridad hogareños disponibles, tales como sistemas de alarmas con sensores de movimientos o la contratación de empresas de seguridad privada. Entre sus varias funciones este tipo de cámaras proveen: detección de movimiento seleccionando un sub área de la zona monitoreada, envío de los screenshots por email o a un servidor FTP, detección de sonido y visión nocturna. Por otra parte, el fabricante provee varias aplicaciones para observar el stream de video desde una computadora o un teléfono móvil.

Un problema que presentan estos dispositivos es que la detección de movimiento, está basada en las técnicas de sustracción de fondo [3] [4] [5] y diferencia entre cuadros [6]. Estos algoritmos son en general muy sensibles a los cambios del entorno, tales como la luminosidad o proyección de sombras en el campo visual de la cámara. Esto produce que muchas veces se detecten "falsos positivos", es decir, detectar un movimiento cuando en realidad no lo hay.

Por otra parte, existen en la bibliografía, diferentes técnicas o algoritmos para detectar personas en imágenes o stream de videos, entre ellos los histogramas de gradientes orientados [7]. En este algoritmo se extraen descriptores de la imagen y se entrena un clasificador supervisado. De esta manera, dada una imagen es posible determinar si la misma contiene personas. Existen implementaciones disponibles de este algoritmo, como por ejemplo la de Adrián Rosebrock en [8] y Dahams en [9], que explican cómo utilizar la implementación de este algoritmo en la librería OpenCV [10] utilizando el lenguaje de programación Python.

Otro de los problemas que presenta el sistema comercial es que las notificaciones por email no son instantáneas, puede haber un delay considerable entre la detección de un evento y la recepción del email. Existen, por otra parte, programas de mensajería instantánea como Telegram [11], el cual es de uso gratuito y provee una API para desarrollar lo que se conocen como "chatbots" [12].

1.2 Objetivos y contribución esperada

El objetivo general del presente trabajo es mejorar las capacidades de notificaciones y análisis de imágenes provistos por cámaras para seguridad hogareñas disponibles en el mercado.

Se tiene como objetivos específicos:

- Analizar el comportamiento y robustez como sistema de vigilancia de video, de una cámara de seguridad hogareña disponible en el mercado, en lo que respecta a la detección de movimiento y notificaciones. Se entiende por sistema de vigilancia robusto, aquel que puede manejar adecuadamente diferentes tipos de situaciones como cambios de luminosidad en el entorno; ruido en la imagen; y efectos de brillo, sombras y reflejos.
- Diseñar un algoritmo que, en base a la detección de movimiento provista por la cámara, analice las imágenes para detectar la presencia de intrusos en el área monitoreada.
- Implementar un nuevo sistema de notificaciones por mensajería instantánea alternativo a las notificaciones ya provistas por el sistema comercial (basado en e-mail).

El aporte que se espera obtener es un sistema de seguridad para hogares con capacidades mejoradas con respecto a los ya provistos por el sistema comercial, tanto en el procesamiento de las imágenes para detectar movimiento como en las notificaciones de detección de movimiento.

Se propone dar solución, a los problemas antemencionados, mediante un sistema de vigilancia con las siguientes características:

- Una o varias cámaras conectadas a la red WI-FI hogareña.
- Las cámaras envían por FTP las imágenes, al detectar un movimiento, utilizando el software y hardware provisto por el fabricante.
- Las imágenes son recibidas en el FTP y procesadas nuevamente por un módulo de procesamiento de imágenes a fin de detectar la presencia de intrusos (personas), pudiendo de esta manera "filtrar" los falsos positivos o aumentar el nivel del alerta recibido.
- Las imágenes son enviadas a través de la API de mensajería instantánea y recibidas en un dispositivo móvil, siendo posible recibir las alertas y screenshots en tiempo real.
- A su vez, a través del mensajero será posible activar y desactivar el sistema de notificaciones como se haría con un sistema de alarmas tradicional.

Todo este sistema puede ser puesto en ejecución, adquiriendo las cámaras necesarias (puede ser tan solo una o más dependiendo de las áreas que se desea monitorear), una conexión a internet, un host que permita ejecutar la aplicación de notificaciones y procesamiento de las imágenes, y un teléfono celular para recibir las notificaciones; que a su vez también permitirá activar y desactivar el

sistema de alertas. La aplicación puede ejecutarse en una computadora propia conectada a la red hogareña o un servidor en la nube.

1.3 Organización del presente informe

El resto del presente informe se estructura de la siguiente manera:

En el Capítulo 2, se hace un resumen de las diferentes técnicas que existen para detectar movimiento en streams de vídeo, cuáles de estas técnicas son más ampliamente utilizadas en los sistemas comerciales existentes en el mercado y se introduce con mayor detalle el método de detección conocido como sustracción de fondo, ampliamente utilizado en dispositivos de video para seguridad.

En el Capítulo 3, se introduce el tema de detección de personas en imagenes y video; y en particular, se describe el algoritmo conocido como Histograma de Gradientes Orientados el cual es utilizado para reconocer objetos en imagenes y resulta efectivo al ser aplicado específicamente a la detección de personas.

Con el marco teórico introducido en los capítulos 2 y 3, el Capítulo 4 describe en primer lugar las prestaciones generales de la cámara TP-LINK NC220 [2], utilizada como parte de este trabajo, y se hace especial énfasis en algunas de las falencias y puntos débiles que presenta dicho sistema.

El Capítulo 5 se detalla la aplicación de software implementada que interactúa con el sistema comercial, con el objetivo de mejorar las falencias descritas en el Capítulo 4 incluyendo la experiencia de usuario, y proveyendo un algoritmo de detección de personas para enriquecer las notificaciones enviadas por la cámara; mediante su software de detección de movimiento embebido.

El Capítulo 6, describe los criterios que pueden utilizarse para medir las prestaciones de un sistema de seguridad por cámaras de video, las pruebas realizadas y los resultados obtenidos con el sistema de seguridad propuesto en el Capítulo 5.

Por último, el Capítulo 7, presenta las conclusiones y potenciales trabajos a futuro que se derivan de este trabajo.

Capítulo 2 - Detección de Movimiento en Video

Este capítulo, introduce la detección de movimiento en secuencias de video. En la Sección 2.1, se describen las técnicas más comúnmente utilizadas para detectar movimiento en video y en particular, su aplicación en cámaras de seguridad; se mencionan varios de las investigaciones relacionadas con el tema en la bibliografía. En la Sección 2.2, se describe en mayor detalle la técnica de sustracción de fondo, ampliamente utilizada para detectar movimiento en cámaras de seguridad estacionarias. Por último, en la Sección 2.3 se describen las implementaciones disponibles de la técnica de sustracción de fondo en la librería OpenCV y también se presenta una implementación sencilla de esta técnica; adecuada para utilizar en dispositivos con bajo poder de cómputo.

2.1 Detección de movimiento y cámaras de seguridad

En los últimos tiempos la detección de movimiento ha ganado un gran interés, principalmente por la gran cantidad de aplicaciones en diferentes áreas tales como el monitoreo de seguridad por video, control de tráfico o reconocimiento de lenguaje de señales. La detección de objetos en movimiento en una secuencia de video es el más desafiante y la vez el primer y más relevante paso de extracción de información, en una gran cantidad de aplicaciones de visión por computador, tales como vigilancia por vídeo, detección de objetos, detección de humanos y control de robots.

Existen diferentes técnicas para separar los objetos en movimiento (primer plano de la imagen), del resto de la escena (fondo de la imagen), en una secuencia de video. Segmentar los objetos en primer plano del resto de la imagen de manera precisa, es una tarea compleja debido a diferentes anomalías como ruido en la imagen, falso movimiento, variaciones en la luminosidad, efectos fantasmales, aperturas en primer plano y entornos no controlados. Entre las técnicas, más ampliamente utilizadas para detectar moviendo entre dos cuadros sucesivos de un video, se encuentran la sustracción de fondo [4], [13]–[16], la diferencia entre cuadros [6], [17], [18] y el flujo óptico [19]–[21].

En la técnica de sustracción de fondo se utiliza la diferencia entre el cuadro actual y el cuadro que representa el fondo de la imagen, para detectar objetos en movimiento. En este método, es necesario mantener un modelo que represente el fondo de la imagen. Un modelo del fondo podría obtenerse por ejemplo, de la imagen promedio en un cierto periodo de entrenamiento. Los objetos en movimiento son fáciles de detectar mediante esta técnica, pero el modelo del fondo debe ser actualizado regularmente. En su versión más simple, esta técnica es sencilla de implementar pero tiene la desventaja de ser muy sensible a los cambios generados por el entorno, haciendo difícil aislar las interferencias del movimiento real de los objetos en la imagen. La sustracción de fondo es la técnica más comúnmente utilizada en cámaras de seguridad fijas. El modelo estadístico basado en la sustracción de fondo es flexible y rápido, pero tanto el modelo del fondo como la cámara deben ser estacionarios cuando se aplica.

En la diferencia entre cuadros, la presencia de objetos en movimiento, es determinada a través de la diferencia entre cuadros consecutivos de la imagen. Se calcula la diferencia absoluta en cada píxel, entre 2 a 3 cuadros consecutivos de la secuencia de video, y se aplica un umbral para obtener los objetos en movimiento. Este método es altamente adaptativo a los entornos dinámicos y computacionalmente menos complejo, pero generalmente no es preciso en la tarea de extraer la forma

completa de ciertos tipo de objetos en movimiento. En particular, los principales causantes de detecciones incorrectas se deben a imágenes fantasmales, aperturas en primer plano con objetos que tiene un movimiento lento o rápido de un color homogéneo.

Por último, la técnica de flujo óptico, es utilizada para encontrar el movimiento aparente o trayectoria de los objetos entre cuadros sucesivos. Sin embargo, la detección basada en esta técnica, es computacionalmente compleja y se pueden obtener resultados poco precisos debido a ruido en la imagen, falso movimiento y variaciones en la luminosidad del entorno. Al ser computacionalmente más costosa, que las técnicas de diferencias entre cuadros y sustracción de fondo, no es en general, apta para ser utilizada en aplicaciones de tiempo real.

Un sistema de monitoreo y vigilancia basado en video no debería depender de la cuidadosa ubicación de las cámaras. Tambien deberia ser robusto ante la presencia de objetos y efectos de luz que puedan ocurrir en su campo visual. Debe ser capaz de detectar movimiento en áreas desordenadas, objetos que se superponen en el campo visual, sombras, cambios de iluminación, efectos de elementos móviles de la escena (por ejemplo, árboles que se mueven), objetos que se mueven lentamente y objetos que son introducidos o removidos de la escena. En general, las técnicas basadas en la sustracción de fondo fallan al ser expuestas a esta gran variedad de efectos.

Es común, que muchos sistemas de monitoreo de video, incluyendo el equipo utilizado en este trabajo, requieran la configuración manual del nivel de sensibilidad para la detección de movimiento para generar alarmas de movimiento. La perfomance de los algoritmos de detección de movimiento, embebidos en cámaras y grabadoras de video digital, usualmente depende del nivel de sensibilidad preseleccionado, que funciona para todas las posibles condiciones del entorno. Por este motivo, estos sistemas tienen una tasa de error que incluye la detección de falsas alarmas (falsos positivos) y la omisión de verdaderas alarmas (falsos negativos).

2.2 Método de Sustracción de fondo

Como se mencionó anteriormente, la sustracción de fondo es una técnica ampliamente utilizada en los sistemas de vigilancia basados en cámaras fijas. Cualquier sistema de detección de movimiento, basado en esta técnica, debería ser capaz de manejar adecuadamente, algunas situaciones críticas, tales como: ruido en la imagen, debido a una calidad pobre de la fuente de transmisión; variaciones graduales en las condiciones de luz de la escena; pequeños movimientos de objetos no estáticos como las ramas de un árbol o arbustos movidos por el viento; y regiones sombreadas que son proyectadas por objetos en primer plano y que son detectadas como objetos en movimiento.

Cuando se trabaja en detección de movimiento, se podría realizar la siguiente asunción:

"El fondo de cualquier stream de video es principalmente estático y no sufre cambios en cuadros consecutivos. Por ende, si se puede modelar el fondo de la imagen, se podrán detectar cambios sustanciales cuando estos se produzcan en el video".

Sin embargo, esta asunción puede fácilmente ser contradicha en el mundo real por diversas situaciones tales como reflejos, cambios de luz y otros cambios en el entorno que pueden ocasionar que el fondo de la imagen modelado luzca diferente entre los cuadros consecutivos del stream de

video. Si el fondo cambia, resultará en una tasa de error alta del algoritmo de detección. Por esta razón, los sistemas de monitoreo más efectivos, basados en sustracción de fondo, son aquellos que utilizan cámaras fijas en condiciones luz controladas.

Se han propuesto una gran variedad de algoritmos para segmentar los objetos en primer plano del fondo de una secuencia de video. Sobral et al. en [13] realizan una reseña y evaluación de varios de los algoritmos, para detectar movimiento mediante la sustracción de fondo, disponibles en la bibliografía. En general, todos estos algoritmos comparten el mismo esquema, que consiste de los siguientes pasos:

1. **Inicialización del fondo:** el primer objetivo es construir un modelo que represente el fondo de la imagen en base a un conjunto fijo de cuadros de la secuencia. Este modelo puede ser diseñado de varias maneras, incluyendo: métodos estadísticos, lógica borrosa o redes neuronales.
2. **Detección de los objetos en primer plano**: en los siguientes cuadros, se realiza un proceso de comparación entre el cuadro actual y el modelo que representa el fondo de la escena. De este proceso, se obtiene el primer plano de la escena en cuestión.
3. **Mantenimiento del modelo del fondo:** durante el proceso de detección, las imágenes también son analizadas a fin de actualizar el modelo del fondo de la escena, construido en el paso de inicialización, en relación a una tasa de aprendizaje preestablecida. Así, por ejemplo, un objeto que no se mueve durante un periodo prolongado de tiempo, debería ser integrado al fondo de la escena.

Además, los autores clasifican los diferentes algoritmos, según la técnica utilizada para inicializar y actualizar el modelo del fondo de la escena, en 4 grupos principales:

Modelos Basicos

Básicamente, el proceso de sustracción de fondo consiste en crear un modelo del fondo de la escena. La manera más simple de realizar esta tarea, es configurar manualmente una imagen estática que represente el fondo de la imagen y que no posea objetos en movimiento (por ejemplo: tomar como referencia el primer cuadro de la secuencia de video). Para cada cuadro del video, se computa la diferencia absoluta entre el cuadro actual y la imagen estática. Este método, es conocido como diferencia de cuadros estática. Sin embargo, la elección de una imagen estática, no es la mejor alternativa si el entorno sufre de cambios de luminosidad, ya que la segmentación del primer plano y el fondo puede fallar drásticamente. Un posible alternativa, es utilizar el cuadro anterior de la secuencia, en vez de una imagen estática.

Luego de construir el modelo del fondo de la escena, se procede a la dirección de los objetos en primer plano. La primer y más común forma de realizar esta tarea es computar la diferencia absoluta entre el cuadro actual y el modelo del fondo. de manera similar a la diferencia de cuadros estática. Sin embargo, en este caso el modelo del fondo es continuamente adaptado en vez de ser una imagen fija.

Modelos estadisticos

Una de los métodos más populares, para implementar la sustracción de fondo, esta basado en un modelo del fondo probabilístico parametrizado. En este algoritmo, la distribución de color de cada

pixel es representado por una suma de distribuciones Gaussianas pesadas definidas en un espacio de color dado: GMM (Gaussian Mixture Model).

Los modelos basados en GMM han demostrado tener un buen desempeño para el análisis de escenas en exteriores, y se ha convertido en un algoritmo popular de sustracción de fondo. Aun, cuando este método es capaz de manejar variaciones graduales de luz, tiene deficiencias para manejar efectos de sombras y variaciones repentinas de luminosidad. Además, la etapa de aprendizaje puede ser ineficiente si es realizada con cuadros del video que contienen ruido.

Muchos autores, han propuesto diferentes formas de atenuar estos problemas y mejorar la sustracción de fondo basada en GMM. En particular, Kaewtrakulpong et al. en [5], propone modificar la ecuación de actualización del modelo para mejorar la adaptación del sistema a las variaciones de iluminación.

Modelos borrosos, métodos neuronales y neuro-borrosos

Recientemente, algunos autores han introducido conceptos de lógica borrosa en las diferentes etapas del proceso de sustracción de fondo. Por ejemplo, realizando la segmentación de la imagen, por la medida de similaridad de las características de color y textura en la imagen de entrada y el modelo del fondo, utilizando diferentes integrales (Sugeno y Choquet).

Por otra parte, es posible utilizar redes neuronales, en donde la red neuronal aprende a clasificar cada píxel de una imagen. Para cada píxel, la red neuronal determina si el mismo pertenece al fondo o el primer plano de la imagen.

2.3 Sustracción de fondo con OpenCV

2.3.1 Algoritmos implementados en la librería

OpenCV [10] es un librería de código abierto para procesamiento de imágenes y visión por computadora. Esta librería, fue utilizada en este trabajo para implementar un módulo de detección de personas en imágenes. Entre el amplio conjunto de utilidades que provee, se encuentran las implementaciones de varios de los algoritmos para sustracción de fondo presentados en la bibliografía. A continuación, se presenta una breve reseña de 3 algoritmos de sustracción de fondo disponibles en la librería OpenCV.

BackgroundSubtractorMOG

Es un algoritmo de segmentación, basado en GMM. Implementa el algoritmo presentado por Kaewtrakulpong et al. en [5]. Utiliza un método para modelar cada píxel del fondo de la imagen mediante una mezcla de K distribuciones Gaussianas (K = 3 a 5). Los pesos de esta mezcla representan las proporciones de tiempo en que dichos colores han permanecido como parte de la escena. Los colores de fondo más probables, son aquellos que permanecen más estáticos durante un tiempo más prolongado. Este sustractor de fondo se encuentra disponible en OpenCV mediante la sentencia:

```
cv2.bgsegm.createBackgroundSubtractorMOG() [22]
```

BackgroundSubtractorMOG2

Es también, un algoritmo de segmentación basado en GMM. El algoritmo es presentado por Zivkovic et al. en [3], [4]. Una característica importante de este algoritmo, es que selecciona el número apropiado de distribuciones Gaussianas para cada pixel. A diferencia del algoritmo MOG, descrito anteriormente, en donde se utilizan K distribuciones para cada pixel a través de todo el algoritmo. Este algoritmo, provee mejor adaptabilidad a las variaciones causadas por ejempl,o por cambios en la luminosidad del entorno. Este sustractor de fondo se encuentra disponible en OpenCV mediante la sentencia:

```
cv2.bgsegm.createBackgroundSubtractorMOG2() [22]
```

En este caso, además es posible especificar un parámetro adicional en el constructor, para especificar si se desea detectar las sombras de los objetos en movimiento. En el caso que `detectShadows = True` (valor por defecto), el sustractor detectara y marcará las sombras en color gris. Como contrapartida, este proceso adicional decrementará la velocidad de cómputo del algoritmo.

BackgroundSubtractorGMG

En este algoritmo, se combina la estimación estadísticas del fondo de la imágenes y una segmentación Bayesiana por pixel. Este algoritmo, es presentado por Godbehere et al. en [23]. Según los autores, el sistema obtuvo muy buenos resultados en un sala de arte interactiva llamada "Are We There Yet?" en exposición desde el 31 de marzo al 31 de julio de 2011 en el Museo Judío Contemporáneo ubicado en la ciudad de San Francisco, California.

El algoritmo, utiliza algunos pocos cuadros iniciales para modelar el fondo de la imagen (120 por defecto), y emplea un algoritmo de segmentación probabilístico para el primer plano de la imagen, que identifica los posibles objetos utilizando inferencia Bayesiana. La estimaciones son adaptativas, ya que a las nuevas observaciones se le asignan pesos mayores que a las menos recientes, logrando así una adaptación a las condiciones variables de iluminación del entorno. Este sustractor de fondo se encuentra disponible en OpenCV mediante la sentencia:

```
cv2.bgsegm.createBackgroundSubtractorGMG() [22]
```

Es recomendable aplicar la operación morfológica de apertura al resultado obtenido para eliminar el ruido existente.

La *Imagen 1*, muestra una pequeña demostración donde se compara los resultados de estos 3 algoritmos de segmentación basados en sustracción de fondo, sobre una secuencia de video, utilizando la librería OpenCV.

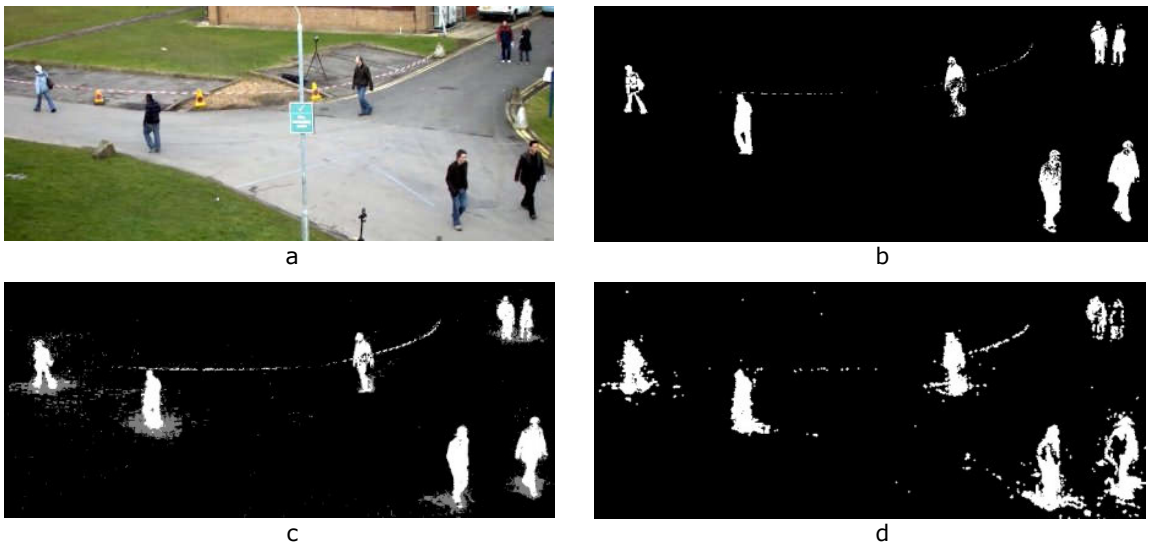


Imagen 1: Prueba de los algoritmos de sustracción de fondo (MOG, MOG2 y GMG) implementados por OpenCV en una secuencia de video. a) Cuadro número 200 del video original b) Resultado de aplicar el sustractor de fondo MOG c) Resultado de aplicar el sustractor de fondo MOG2 (en color gris se visualizan las sombras de los objetos)d) Resultado de aplicar el sustractor de fondo GMG (se elimina el ruido aplicando la operación de apertura morfológica) (Fuente: [22])

2.3.2 Implementación con bajo costo computacional

Si bien los algoritmos antes mencionados, son capaces de adaptarse a los cambios del entorno y filtrar con una menor tasa de error falsos positivos y falsos negativos, tienen como contrapartida ser computacionalmente costosos. Esto puede ser un limitante al momento de ser embebidos en algunos dispositivos para monitoreo de video. Adrian Rosebrock en [24], propone una alternativa más simple, para detectar movimiento utilizando sustracción de fondo, que consiste en tomar como modelo del fondo el primer cuadro de una secuencia de video y detectar movimiento simplemente haciendo una diferencia entre el fondo y el cuadro actual y configurando un cierto límite para descartar áreas en movimiento pequeñas.

La *Imagen 2* muestra un ejemplo de esta implementación, en donde puede observarse la diferencia entre el fondo y el cuadro actual del video. En este ejemplo, se detecta la presencia de una persona, y se muestra la diferencia entre el fondo y el cuadro actual, y la imagen binaria que representa el área en movimiento. Obviamente, por lo antes expuesto, un algoritmo con estas caracteristicas, será muy propenso a tener una tasa de error alto, debido a la no actualización y adaptación del modelo del fondo a lo largo del tiempo. Por otro lado, puede llegar a ser lo suficientemente efectivo, si el área a monitorear esta luminicamente controlada.

Imagen 2: Prueba de detección de movimiento, realizada sobre un stream de video, mediante la implementación mas básica de sustracción de fondo con OpenCV y Python (Fuente: propia)

Capítulo 3 - Detección de Personas en Imagenes y Video

Este capítulo, introduce la detección de personas en imagenes y video. En la Sección 3.1, se presenta el tema y se mencionan varios de los trabajos disponibles en la bibliografía. En la Sección 3.2, se describe la técnica de Histogramas de Gradientes Orientados, la cual es ampliamente utilizada para detectar personas en imagenes y videos, obteniendo muy buenos resultados. Por último, en la Sección 3.3, se presenta una implementación completa utilizando la librería OpenCV y los descriptores de histogramas de gradientes orientadas combinados con una máquina vectorial de soporte lineal (LSVM).

3.1 Introducción a la detección de personas

Una rama importante de investigación en el área de visión por computadora, que ha ganado gran importancia en los últimos años, es el entendimiento e interpretación de la actividad humana en streams de video. El interés creciente por la detección de movimiento de personas, está fuertemente motivado por las mejoras recientes en las técnicas de visión por computadora y la disponibilidad de hardware de bajo costo, tales como cámaras de video y una gran variedad de aplicaciones prometedoras como la identificación personal y los sistemas de vigilancia visual automáticos. El objetivo, de la detección de movimiento, es detectar los objetos en movimiento. La detección de un objeto puede contribuir al siguiente paso, que es el reconocimiento de dichos objetos [6].

Con el correr de los años se han aplicado a la detección de personas una amplia variedad de métodos, con continuas mejoras en la performance. Algunos métodos se focalizan en la mejora de las características base utilizadas, mientras que otros se enfocan en los algoritmos de aprendizaje, y otras técnicas incorporan modelos de partes deformables (deformable parts models) o la utilización de un contexto.

Viola et al. en [25] propone utilizar clasificadores en cascada, un método ampliamente utilizado en aplicaciones de tiempo real. Este método, ha sido extendido para emplear diferentes tipos de características y técnicas, pero fundamentalmente el concepto de cascada, ha sido utilizado para lograr una detección en tiempo real. Quizás, la característica más popular utilizada para la detección de personas (y varias otras tareas basadas en la detección sobre imágenes) son los histogramas de gradientes orientados, desarrollados por Dalal et al. [7]. Este método, será explicado en detalle en las siguientes secciones y fue utilizado para implementar el detector de intrusos incorporado al sistema de vigilancia comercial.

Hamdoun en [26] propone utilizar algoritmos para la obtención de puntos claves, como SIFT [27] y SURF [28], en combinación con la sustracción de fondo para detectar personas. Mientras que Angelova et al. en [29], combina la eficiencia de los clasificadores en cascada con la exactitud de las redes neuronales.

3.2 Histograma de Gradientes Orientados

El método de reconocimiento de objetos mediante descriptores HOG (sigla de histogramas de gradientes orientados), utilizando principalmente en las áreas de visión por computadora y procesamiento de imágenes, fue propuesto inicialmente en 2005 en el artículo seminal de Dalal y Triggs [7]. A pesar de tener más de una década, es un método que aún hoy sigue siendo ampliamente

utilizado y con el que se obtienen excelentes resultados. En su paper, Dalal y Triggs, demuestran que los descriptores de histogramas de gradientes orientados de una imagen, en conjunto con una máquina vectorial de soporte lineal (LSVM), pueden ser utilizados para entrenar clasificadores con una alta precisión. Los autores, en su trabajo, lo aplicaron a la detección de personas en imagenes.

La técnica de HOG cuenta las ocurrencias de la orientación de los gradientes localizados en un área de una imagen. El fundamento esencial, detrás de los descriptores de histograma de gradientes orientados, es que la apariencia y la forma de los objetos dentro de una imagen pueden describirse mediante la distribución de gradientes de intensidad o direcciones de los bordes. La imagen es dividida en peqũenas regiones llamadas celdas, y para los píxeles dentro de cada celda, se compila un histograma de direcciones de gradientes. Los descriptores son la concatenación de estos histogramas. Para mejorar la precisión, los histogramas locales pueden ser normalizados en contraste, mediante el cálculo de la intensidad en una área mayor de la imagen, llamada bloque; y luego utilizar este valor para normalizar las celdas dentro del bloque. En general, esta normalización, resulta en la obtención de mejores invariantes a cambios en la luminosidad y reflejos que se puedan presentar en las imágenes.

Los descriptores HOG tiene algunas ventajas claves sobre otro tipo de descriptores. Ya que operan sobre celdas locales, son invariantes a las transformaciones geométricas (por ejemplo: espejar la imagen) o fotométricas (ejemplo: aumentar el contraste o convertir a escala de grises), excepto por la orientación de los objetos (ejemplo: rotar la imagen en 90 grados). Tales cambios sólo serian notorios en regiones espaciales más amplias. Además, como describen Dalal y Triggs, el muestreo espacial amplio, el muestreo de orientación más fino y la normalización fotométrica localizada permiten que los movimientos corporales individuales de las personas puedan ser ignorados siempre y cuando mantengan una posición los mas cercana a estar erguidas. Por este motivo, es que esta técnica es muy efectiva para la detección de personas en imagenes. En la *Imagen 3* puede observarse una representación visual de los descriptores HOG de una imagen, realizada mediante Matlab [30]. Se puede notar que, en la zona de la imagen donde se encuentra una persona, la orientación de los gradientes es diferente al resto de la imagen.

El algoritmo general para entrenar un detector de objetos, utilizando histogramas de gradientes orientados, implica los siguientes pasos:

1. Del conjunto de datos de entrenamiento tomar P muestras positivas (muestras que contengan los objetos que se desea detectar) y extraer los descriptores HOG de estas muestras.
2. Del conjunto de datos de entrenamiento tomar N muestras negativas (muestras que no contengan los objetos que se desea detectar) y extraer los descriptores HOG de estas muestras. En la práctica es esperable que N >> P.
3. Entrenar una máquina de con soporte para vectores lineales sobre las muestras negativas y positivas.
4. Aplicar minería de datos negativa. Es decir, para cada imagen y cada posible escala de cada imagen en el conjunto de entrenamiento negativo, aplicar la técnica de ventana deslizante deslizando dicha ventana sobre la imagen (*Imagen 4*). En cada ventana computar los descriptores HOG y aplicar el clasificador. Si el clasificador clasifica

(incorrectamente) una ventana como un objeto, registrar el vector de características asociado con el falso positivo junto con la probabilidad de clasificación.

5. Tomar los falsos positivos encontrados durante el paso 4 y ordenarlos por su confidencia (probabilidad) y re entrenar el clasificador utilizando estos ejemplos negativos (Es posible aplicar los pasos 4 y 5 en forma iterativa, pero en la práctica con una sola etapa suele ser suficiente).
6. En este punto el clasificador está entrenado y puede ser aplicado al conjunto de datos de prueba. Una vez más, al igual que en el paso 4, para cada imagen en el conjunto de datos de prueba, y para cada escala de la imagen, aplicar la técnica de ventana deslizante. En cada ventana extraer los descriptores HOG y aplicar el clasificador. Si el clasificador detecta un objeto con una alta probabilidad, registrar el marco de la ventana. Luego de terminar de escanear la imagen, será necesario aplicar alguna alguna tecnica para eliminar aquellas ventanas que se solapen y obtener el área del objeto detectado.

Imagen 3: Visualización de los descriptores de Histogramas de Gradientes Orientados en imágenes con personas utilizando Matlab a.1 b.1) Imagen original a.2 b.2) Gradientes Orientados de la imagen original a.3 b.3) Superposición de los gradientes sobre la imagen original a.4 b.4) Detalle de la orientación de los gradientes en la zona donde se visualiza una persona (Fuente: propia)

Estos son los pasos mínimos requeridos, para entrenar y construir un detector de objetos basado en histograma de gradientes orientados.

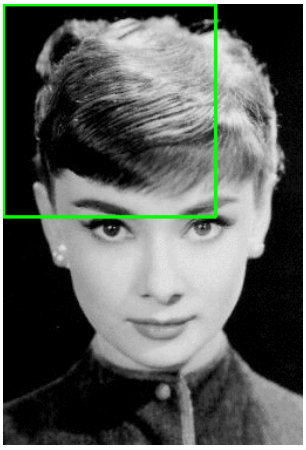


Imagen 4: En la técnica de ventana deslizante, se desliza la ventana sobre toda la imagen (de arriba hacia abajo y de izquierda a derecha) extrayendo los descriptores HOG en cada posición. Aquí solo se muestra una escala de la imagen, pero en la práctica hay que realizar este procedimiento para múltiples escalas de la imagen (Fuente: [31])

Al trabajar con detección de objetos en imágenes, es muy probable encontrarse con el problema de obtener múltiples recuadros sobre el objeto que se desea detectar. Básicamente, se necesita un procedimiento para eliminar estos recuadros redundantes y obtener el recuadro que mejor represente el área del objeto detectado. Existen diferentes técnicas para realizar esta tarea. Rosebrock A en [31], propone utilizar el método de supresión no máxima (*non maximum suppression*) que tiene por objetivo eliminar los múltiples recuadros detectados y fusionarlos en uno solo. En la *Imagen 5* se muestra los múltiples recuadros obtenidos, al aplicar un detector de rostros a una imagen, los cuales son fusionados en un único recuadro luego de aplicar la técnica de supresión no máxima.

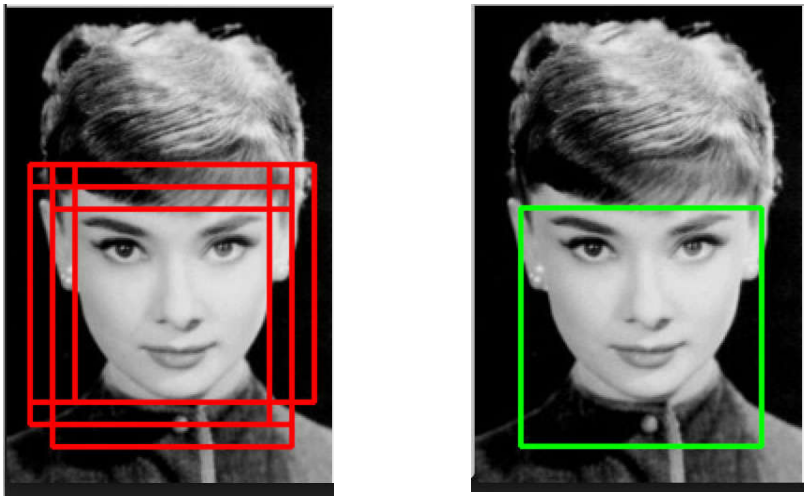


Imagen 5: Luego de aplicar un detector de objetos a una imagen se pueden obtener múltiples recuadros para el mismo objeto (a) siendo necesario aplicar alguna mecanismo para fusionar los recuadros y obtener el área del objeto detectado (b) (Fuente: [31])

3.3 Detección de personas en OpenCV

3.3.1 Descripción del algoritmo

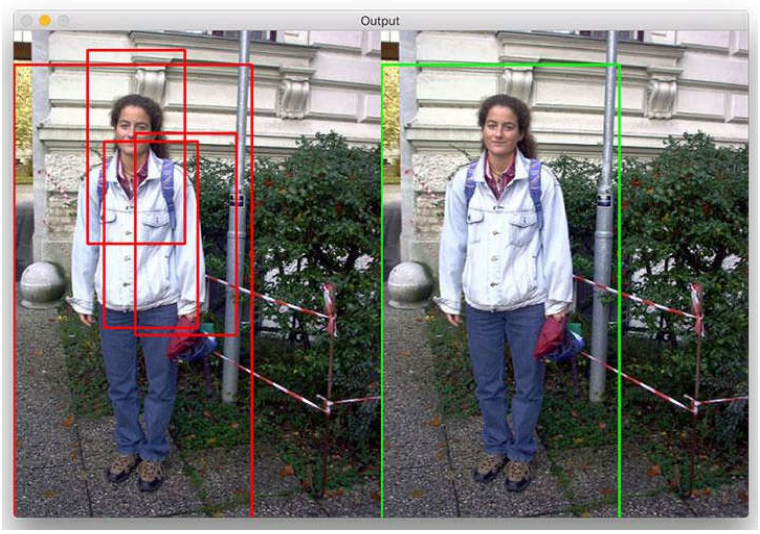
OpenCV [10] es un librería de código abierto para procesamiento de imágenes y visión por computadora. Entre las numerosas características que provee, se encuentra un clasificador SVM con histograma de gradientes orientados, pre entrenado para detectar personas tanto en imágenes como streams de video. Esta implementación está basada en el algoritmo descrito por Dalal y Triggs en su artículo [7]. La *Imagen 6* muestra el código que implementa la detección de personas en imágenes utilizando OpenCV y Python, basada en la solución propuesta por Adrián Rosebrock en [31]. A continuación, se detallara cada sección del código presentado, el cual se utilizará para implementar el detector de intrusos que forma parte del sistema de vigilancia desarrollado en este trabajo:

- Se inicializa el clasificador con los descriptores de gradientes orientados para detectar personas. Luego de este punto el detector de personas ya se encuentra inicializado para procesar imágenes (líneas 2-3).
- Se itera sobre un conjunto de imágenes de entrada en un directorio y para cada imagen la misma es cargada en memoria y redimensionada (líneas 6-11). El objetivo de reducir el tamaño de las imágenes a un ancho no mayor a 400 pixeles tiene 2 objetivos:
    1. Asegurar que sea necesario evaluar menos ventanas deslizantes en la pirámide de la imagen. Es decir, extraer los descriptores HOG y pasarlos al clasificador lineal SVM. De esta manera, se reduce el tiempo de detección.
    2. También mejora la exactitud general del detector, que se traduce en una menor cantidad de falsos positivos.
- Se realiza efectivamente la detección de personas sobre la imagen, mediante la llamada al método *detectMultiScale* (línea 14). Este método, construye una pirámide de la imagen con escala igual a 1.05 y una ventana deslizante con pasos de tamaño 4x4 píxeles en la dirección x e y respectivamente. El tamaño de la ventana está fijo en 64x128 pixeles, tal como se sugiere en [7]. Este método, retorna una tupla con las coordenadas que delimitan el rectángulo de cada persona detectada sobre la imagen, y el valor de confidencia retornado por el clasificador para cada detección. En la siguiente sección, se describirán en mayor detalle cada parametros del metodo *detectMultiScale,* y como influye en los resultados de detección y tiempos de ejecución.
- Se dibujan sobre la imagen original los rectángulos obtenidos en el paso anterior (líneas 17-18).

```
1  # initialize the HOG descriptor/person detector
2  hog = cv2.HOGDescriptor()
3  hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
4
5  # loop over the image paths
6  for imagePath in paths.list_images(args["images"]):
7      # load the image and resize it to (1) reduce detection time
8      # and (2) improve detection accuracy
9      image = cv2.imread(imagePath)
10     image = imutils.resize(image, width=min(400, image.shape[1]))
11     orig = image.copy()
12
13     # detect people in the image
14     (rects, weights) = hog.detectMultiScale(image, winStride=(4, 4), padding=(8, 8), scale=1.05)
15
16     # draw the original bounding boxes
17     for (x, y, w, h) in rects:
18         cv2.rectangle(orig, (x, y), (x + w, y + h), (0, 0, 255), 2)
19
20     # apply non-maxima suppression to the bounding boxes using a
21     # fairly large overlap threshold to try to maintain overlapping
22     # boxes that are still people
23     rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])
24     pick = non_max_suppression(rects, probs=None, overlapThresh=0.65)
25
26     # draw the final bounding boxes
27     for (xA, yA, xB, yB) in pick:
28         cv2.rectangle(image, (xA, yA), (xB, yB), (0, 255, 0), 2)
29
30     # show some information on the number of bounding boxes
31     filename = imagePath[imagePath.rfind("/") + 1:]
32     logger.info("%s: %d original boxes, %d after suppression", filename, len(rects), len(pick))
33
34     # show the output images
35     cv2.imshow("Before NMS", orig)
36     cv2.imshow("After NMS", image)
37     cv2.waitKey(0)
```

Imagen 6: Codigo base que implementa la detección de personas utilizado un clasificador SVM con histogramas de gradientes orientados utilizando OpenCV y Python (Fuente: propia)

- Se aplica la técnica de supresión no máxima, para eliminar los recuadros solapados y obtener una única área rectangular para cada persona detectada (líneas 23-24). En este caso particular, se fusionan aquellos rectángulos que estén solapados en un área igual o mayor a 0.65.
- Se dibujan sobre una copia de la imagen, los rectángulos obtenidos en el paso previo (líneas 27-28).
- Por último, se visualizan las imágenes con los rectángulos dibujados sobre cada persona detectada, antes y después de aplicar el método para eliminar los rectangulos solapados (líneas 35-36). La *Imagen 7*, muestra un ejemplo de cómo se visualizan los rectángulos de las personas detectadas antes y después de aplicar la técnica de supresión no máxima [32]. La *Imagen 8* y *Imagen 9* muestran la utilización del algoritmo presentado sobre un stream de video y un par de imágenes tomadas con la cámara utilizada en este trabajo, respectivamente.



a b

Imagen 7: Ejemplo de detección de personas utilizando el algoritmo HOG + SVM.
a) Visualización de los rectángulos detectados b) Visualización de los rectángulos detectados luego de aplicar el método de supresión no máxima (Fuente: [31])

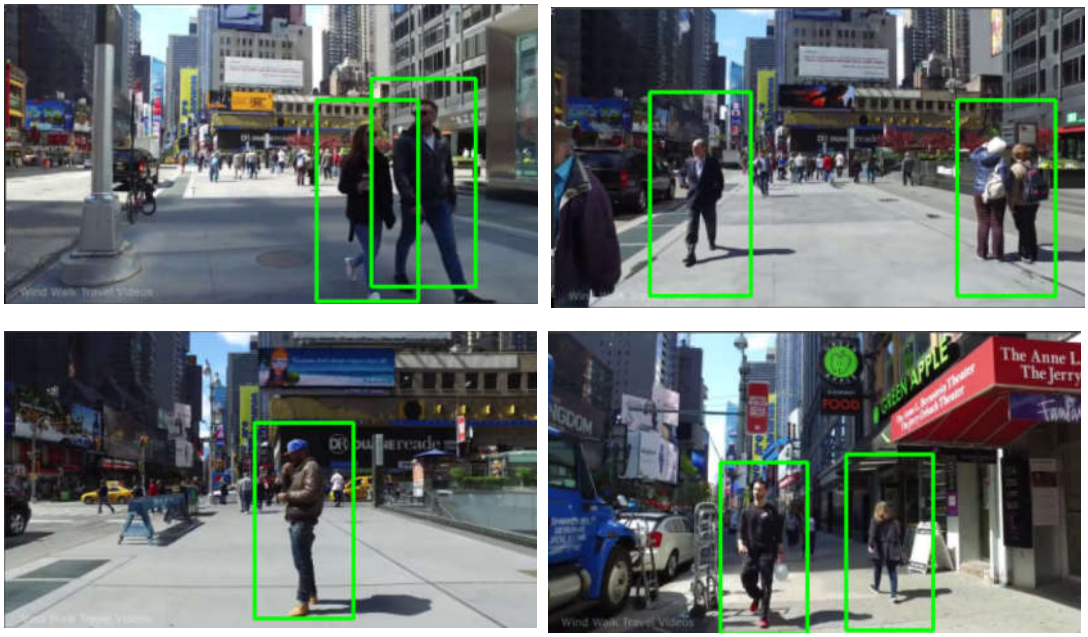


Imagen 8: Prueba de detección de personas con HOG + LSVM en OpenCV, sobre un stream de video obtenido de youtube[1] (Fuente: propia)

[1] Walking tour around Times Square in Midtown Manhattan, New York City
https://www.youtube.com/watch?v=u68EWmtKZw0&t=108s

Imagen 9: Prueba de detección de personas con HOG + LSVM en OpenCV, sobre imágenes capturadas con la cámara utilizada en este informe (Fuente: propia)

3.3.2 Parámetros del algoritmo

La función detectMultiScale presentada en la sección previa es la que efectivamente realiza la detección de objetos sobre imágenes. En esta sección se analizará cómo los parámetros de este método pueden:

- Incrementar el número de falsos positivos (ejemplo: reporta que una parte de una imagen contiene una persona, pero en realidad no existe)
- Resultar en la pérdida de detecciones
- Afectar dramáticamente la velocidad del proceso de detección.

A continuación se describirán cada parámetros, en base al análisis realizado por Adrian Rosebrock en [33]:

- **img (requerido)**: es la imagen en donde se realizará la detección de los objetos. La imagen puede ser a color o en escala de grises.
- **hitThreshold (opcional)**: este parámetro no es utilizado por defecto en la implementación de OpenCV, con lo cual no tiene ningún efecto en el proceso de detección.
- **winStride (opcional)**: es una tupla que determina el tamaño en pixeles del desplazamiento de la ventana deslizante en las direcciones x e y respectivamente.
- **padding (optional)**: es una tupla que indica el espacio en píxeles en x e y de margen interno (padding), que se aplica al área de la ventana deslizante, previo a extraer los descriptores de HOG. En Dalal et. al [7] los autores sugieren que, agregar un poco de padding previo a la extraccion y clasificacion, incrementa la exactitud del clasificador. Los valores usuales para esta parámetro variante entre (8,8), (16,16), (24, 24) y (32,32).
- **scale (optional)**: la pirámide de una imagen es una representación en diferentes escalas de dicha imagen (*Imagen 10*). En cada capa de la pirámide, la imagen es redimensionada a una escala menor y opcionalmente suavizada mediante un filtro Gaussiano. El parámetro *scale* controla el factor en el cual la imagen es redimensionada en cada capa de la pirámide, influenciando de esta manera la cantidad de niveles que tendrá la pirámide.

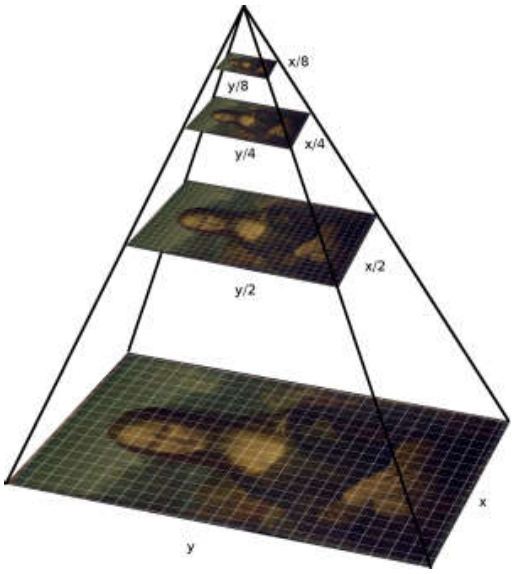


Imagen 10: La pirámide de una imagen es la representación en varias escalas de dicha imagen (fuente: [33])

- **finalThreshold (opcional)**: en los enlaces de la librería OpenCV con Python, este parámetro no tiene ninguna utilidad. Es decir, no es utiliza dentro del código.
- **useMeanShiftGrouping (opcional)**: este parámetro indica si debe agruparse los rectángulos de los objetos detectados en la imagen, para evitar posibles solapamientos. Adrian Rosebrock en [33], sugiere que se obtienen mejores resultados aplicando la técnica de supresión no máxima [32], para eliminar rectangulos solapados luego del proceso de detección. Por lo tanto, este parámetro se utiliza con el valor booleano false.

Los parámetros *winStride* y *scale* son dos de los parámetros más importantes, de la función *detectMultiScale* descrita en la sección previa, y deben ser seteados adecuadamente; ya que de otra manera afectarán notoriamente la exactitud del detector y la velocidad de procesamiento necesaria para realizar la detección.

En el contexto de la detección de objetos, una ventana deslizante es una región rectangular con un alto y ancho fijos que se desliza sobre una imagen (*Imagen 4*). En cada posición de la ventana deslizante, y por cada capa en la pirámide de la imagen (determinadas por el valor del parámetro scale), se extraen los descriptores HOG y se pasan a la LSVM para realizar el proceso de clasificación. El proceso de extracción de descriptores y clasificación es un proceso costoso; con lo cual, evaluar la menor cantidad de ventanas posibles permitirá que el detector se ejecute en una velocidad que permita aplicarlo en tiempo real. Cuanto más chico sea el valor de *winStride*, será necesario evaluar más ventanas. Por el contrario, a medida que se incrementa su valor el tiempo de detección disminuye. Sin embargo, si el valor es muy alto, se producirá la pérdida de detecciones. Un buena alternativa es comenzar con una valor de 4x4, y gradualmente ir elevando su valor hasta conseguir un buen balance entre velocidad y exactitud del detector.

Por otra parte, un valor pequeño para el parámetro *scale*, incrementará el número de capas en la pirámide de la imagen y por ende el tiempo del proceso de detección. Mientras, que un valor mayor decrementará el número de capas en la pirámide, junto con el tiempo necesario para detectar un

objeto; a costa de perder algunas detecciones. Los valores usuales para el parámetro *scale* van en el rango de 1.01 a 1.5 si se desea poder realizar una detección en tiempo real o al menos cercana a dicha cota.

Por lo explicado anteriormente se pueden proponer los siguientes tips, para lograr un buen balance entre exactitud de detección y velocidad de procesamiento, que permitan llevar a cabo la detección de objetos lo más cercano a tiempo real que sea posible:

1. Redimensionar las imágenes para que sean lo más pequeñas posible sin sacrificar exactitud de detección. Cuanto más chicas sean las imagenes mas rapido sera el proceso de detección.
2. Asignar adecuadamente los valores de los parámetros *scale* y *winStride*. Como se mencionó anteriormente estos dos parámetros son los que mayor impacto tienen sobre la velocidad de detección. Los parámetros *scale* y *winStride*, deberían ser lo más grandes posibles sin sacrificar exactitud en la detección.

Capítulo 4 - Sistema de Vigilancia Analizado

Este capítulo, describe las especificaciones y características de la cámara comercio TP-LINK nc220 utilizada como parte de las pruebas de este trabajo. En la Sección 4.1, se resumen las características y especificaciones del equipo; y en la Sección 4.2 se especifican los puntos débiles y falencias que presenta el sistema comercial en cuestión, y que luego se intentarán mejorar con el software implementado en este trabajo.

4.1 Descripción de la Cámara TP-LINK nc220

La cámara, en la cual se basa este trabajo, es una camara TP-LINK modelo NC220 [2]. Esta cámara cuenta con las siguientes características y prestaciones:

- Visión nocturna hasta 18 pies en oscuridad total.
- Detección de movimiento y sonido.
- Envío de notificaciones por email o FTP ante eventos de movimiento y sonido.
- Conectividad inalámbrica.
- Funciona también como amplificador de señal WIFI.
- Provee aplicaciones para visualizar en vivo, el stream de video, para las plataformas Android , IOS, Windows [2] o desde un navegador Web a través de la plataforma TP-LINK cloud [34] (*Imagen 11*).

Otras características importantes relacionadas directamente con el desempeño del sistema de detección de movimiento embebido en la cámara son:

- Se puede configurar el nivel de sensibilidad para el detector de movimiento, se proveen tres niveles: bajo, medio y alto. Un nivel de sensibilidad bajo puede ocasionar la pérdida de alarmas positivas, mientras que un nivel de sensibilidad algo puede ocasionar una mayor cantidad de falsas alarmas.
- También es posible configurar la visión de la cámara en modo diurno, nocturno y automático. En modo diurno, la cámara filma en colores pero es necesario que haya buena luz en el ambiente; en modo nocturno la cámara filma en escala de grises y es capaz de registrar video con muy poca o directamente sin luz en el ambiente, por último en modo automático la cámara cambia entre el modo diurno y nocturno según la cantidad de luz que presenta el ambiente en cada instante. En un ambiente donde las condiciones lumínicas no son constantes, la mejor opción será configurar la cámara en modo automático.

La *Imagen 12* muestra el esquema de funcionamiento del sistema original. La cámara es conectada a una red wifi, y al activar la detección de movimiento, las alertas pueden ser enviadas a un email, el cual puede ser visualizado en un dispositivo móvil o computadora, o un servidor FTP, al cual es posible acceder posteriormente para inspeccionar las imágenes. La activación de las alertas se realiza accediendo a la aplicación Web embebida en la cámara, únicamente desde la red interna donde está instalada la cámara.

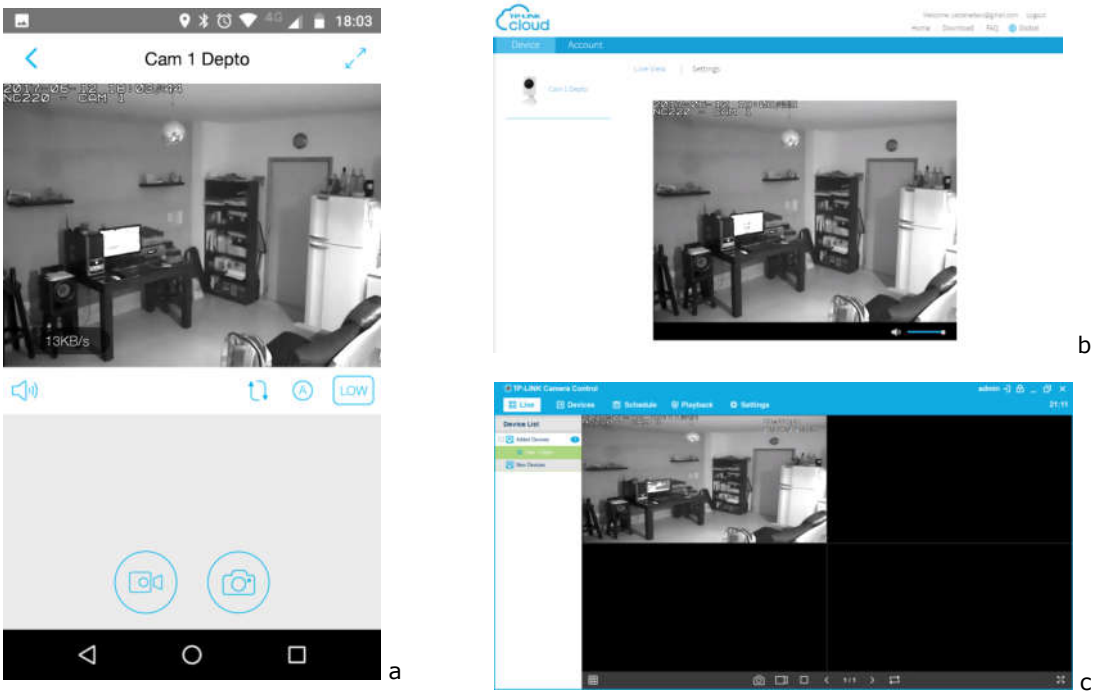


Imagen 11: Diferentes aplicaciones provistas por TP-LINK para observar el stream de video en vivo de la cámara NC220. a) Aplicación para dispositivos móviles Android b) Plataforma Web TP-LINK Cloud c) Aplicación de escritorio para sistemas Windows (Fuente: propia)

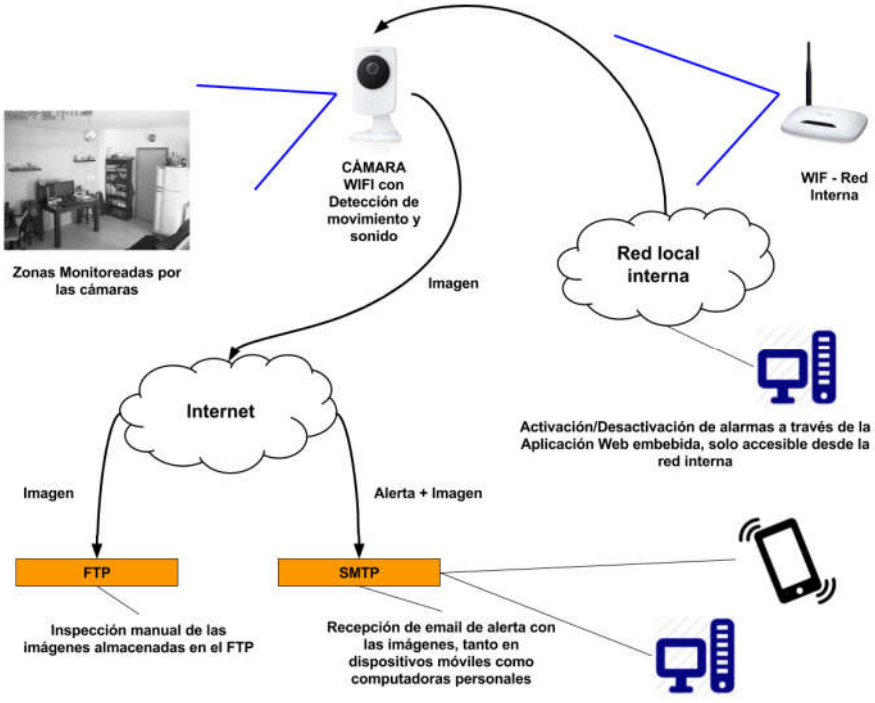


Imagen 12: Esquema de funcionamiento del sistema de vigilancia comercial con cámaras TP-LINK nc220 (Fuente: propia)

Esta cámara puede adquirirse por alrededor de 800 a 1000 pesos argentinos (fuente: Mercado

Libre[2]), lo cual es un precio relativamente bajo en relación a las prestaciones que ofrece. Sin embargo, la cámara tal como es ofrecida por el fabricante tiene varios puntos débiles, que hacen que su uso como sistema de monitoreo no sea el más efectivo. Entre estos puntos, se pueden mencionar los siguientes:

- Configuración con soporte limitado en algunos navegadores Web.
- Procedimiento complejo para la activación y desactivación de las alertas de movimiento y sonido.
- El algoritmo de detección de movimientos tiene una tasa alta de falsos positivos. En muy sensible a los cambios lumínicos y reflejos del entorno.

  En las siguientes secciones se detallará cada uno de los puntos antes mencionados.

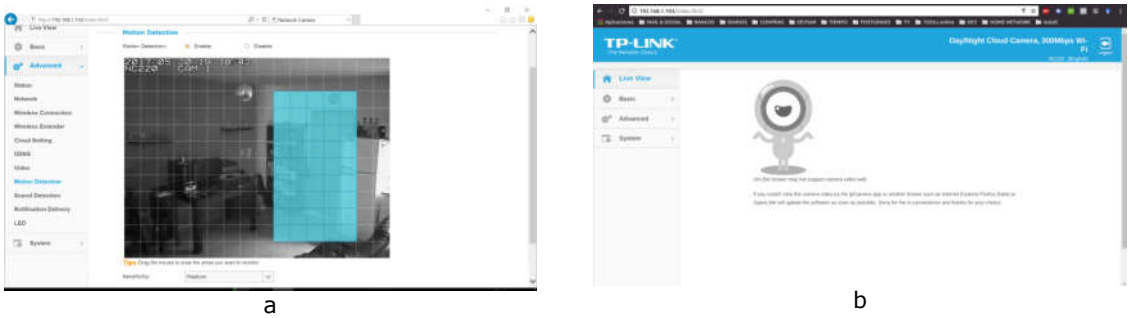
En general, dispositivos como la cámara NC220 y routers WIFI entre otros, poseen actualizaciones provistas por los fabricantes tanto para su firmware como para las aplicaciones de software necesarias para interactuar con los equipos. Usualmente, estas actualizaciones corrigen bugs, mejoran algún aspecto de funcionalidad existente o incluso, puede agregar nuevas funcionalidades. Es probable que algunos de los problemas mencionados sobre la usabilidad y prestaciones de la cámara, especialmente los de usabilidad, sean resueltos en futuras actualizaciones del firmware y software. Al momento de finalización de este trabajo, TP-LINK no reportó ninguna actualización que permita la activación de las alarmas, mediante la aplicación para dispositivos móviles o la plataforma TP-LINK Cloud; así como tampoco una solución al problema con los plugins propietarios para administrar la camara via navegador Web. Para el desarrollo de este trabajo, se utilizó la última versión del firmware y aplicación para dispositivos Android publicada por TP-LINK.

4.2 Puntos débiles y falencias del sistema analizado

4.2.1 Configuración Web con soporte limitado en los navegadores Web más populares

La cámara provee una aplicación de configuración Web, que permite ver y modificar diferentes aspectos de su funcionamiento, como por ejemplo: visualizar el stream de video, configuración de red, configuración de video y audio, notificaciones ante eventos de detección de movimiento y sonido, entre otros. Sin embargo, esta aplicación utiliza plugins propietarios que solo están disponibles en la plataforma Windows y funcionan únicamente con los navegadores Web Firefox e Internet Explorer, lo cual limita bastante la configuración y puesta a punto desde otros navegadores y plataformas. El caso más notorio, donde esta falta de soporte complica la configuración, es la sección de configuración de detección de movimiento, donde el usuario puede seleccionar una sub área, del área monitoreada por la cámara (*Imagen 13*).

[2] Mercado Libre: http://electronica.mercadolibre.com.ar/camaras-ip/tp-link-nc220



a b

Imagen 13: a) Sección para la configuración de detección de movimiento de la cámara b) Algunos navegadores Web como Google Chrome no tienen soporte para este plugin propietario, no siendo posible realizar esta configuración (Fuente: propia)

En [35] se provee un script que permite realizar las mismas configuraciones desde una aplicación implementada en Perl. Con esta aplicación es posible configurar la detección de movimiento desde una plataforma Linux (*Imagen 14*).

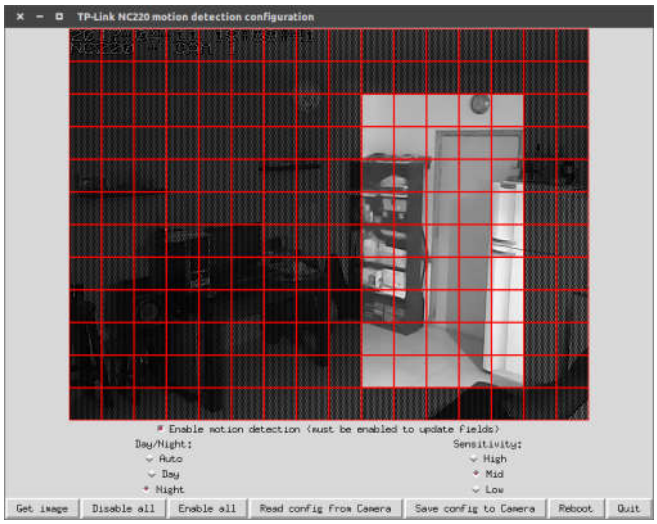


Imagen 14: Aplicación independiente implementada en Perl que permite realizar la configuración de detección de movimiento de la cámara TPLINK NC220 desde otras plataformas como Linux, a causa del limitado soporte del fabricante (Fuente: propia)

4.2.2 Activación y desactivación de los eventos de detección de movimiento y sonido

La camara TP-LINK nc220 permite detectar movimiento y sonido en el área monitoreada (el campo visual registrado por la cámara). Una vez configuradas estas dos opciones, la cámara permite enviar una alarma a un email o a un servidor FTP. La principal limitación que se observa, de esta funcionalidad, es que la activación y desactivación de los eventos son únicamente posibles de realizar a través de la aplicación de configuración Web. Esta aplicación es accesible únicamente dentro de la red local donde la cámara ha sido instalada. Todas las demás aplicaciones (mobile, desktop y la plataforma TP-LINK Cloud), provistas por el fabricante para tener acceso a la configuración de la cámara, no permiten visualizar y/o modificar la configuración de detección de movimiento y sonido y las notificaciones ante dichos eventos.

Sería deseable que la activación y desactivación de las alertas se pueda realizar en forma análoga a la manera en que se activa una alarma hogareña (ingresando un código numérico en una consola), o la alarma de un automóvil (utilizando un control remoto). Sin dudas, este es uno de los puntos más débiles que presenta la cámara, como dispositivo de monitoreo y seguridad. Ya que será necesario ingresar vía navegador Web a la aplicación de configuración cada vez que se desee activar o desactivar las notificaciones.

4.2.3 Falsos positivos en la detección de movimientos

Como se explicó anteriormente la mayoría de las cámaras para monitoreo, como es el caso de la cámara utilizada en este trabajo, utiliza como algoritmo para detección de movimiento la sustracción de fondo o la diferencia entre cuadros. El principal, problema de estos algoritmos es que son muy susceptibles a diferentes cambios del entorno tales como cambios en la luminosidad del ambiente, reflejos y sombras. Lo cual conlleva a obtener lo que se conoce como "Falsos Positivos". En otras palabras, la cámara detecta un movimiento y emite un alerta, cuando en realidad no hubo un movimiento real en el campo visual de la cámara. En una de las pruebas realizadas con el sistema, utilizando únicamente la funcionalidad provista por el fabricante, se obtuvo aproximadamente un 20% de falsos positivos, sobre el total de alertas registradas por el equipo; tal como se documenta en el Capítulo 6.

Si bien parte de código de la cámara es de acceso público bajo la licencia GPL, el fabricante no especifica cual es el algoritmo de detección de movimiento incorporada en la misma. Sin embargo, mediante su uso y resultados, todo parece indicar que el algoritmo está basado en la sustracción de fondo, que a su vez puede ser limitada a un sub área de la imagen capturada por el rango de visión de la cámara.

Capítulo 5 - Sistema de Vigilancia Mejorado

Este capítulo, describe el sistema de software implementado para mejorar las prestaciones generales, del sistema comercial presentado en el capítulo anterior. En la Sección 5.1, se especifican los objetivos que se intentan lograr con esta nueva implementación. La Sección 5.2, describe la arquitectura general. En la Sección 5.3, se detalla la implementación y tecnologías utilizadas en cada componente. La Sección 5.4, enumera las posibles alternativas para ejecutar el sistema implementado. Por último, la Sección 5.5 resume los recursos y costos que implican poner en funcionamiento todo el sistema de vigilancia propuesto.

5.1 Objetivos

Habiendo explicado las características de la cámara a utilizar. Uno de los objetivos de este trabajo es intentar mejorar los puntos débiles del sistema comercial. En particular, se intentará realizar mejoras en 3 puntos:

1. El procedimiento para activar y desactivar la detección de movimiento, de manera tal que sea similar a la activación de una alarma tradicional para hogares o la alarma de un automóvil. El sistema comercial solo permite realizar esta acción ingresando a la aplicación Web de configuración, la cual sólo es accesible desde la red local donde está instalada.
2. Proveer un nuevo canal, para recibir las notificaciones, ante un evento de detección de movimiento. El sistema comercial solo provee la recepción de estas notificaciones vía email, o el envío a un servidor FTP para realizar una inspección manual posteriormente.
3. Intentar disminuir la cantidad de falsos positivos registrados por el hardware y software del sistema comercial. En particular, se intentará determinar la existencia de intrusos (personas) en el campo visual de la cámara, mediante la utilización del algoritmo de histograma de gradientes orientados pre entrenado para detectar personas.

Una alternativa para llevar a cabo esta serie de modificaciones, sería modificar el software embebido en la cámara, también conocido como firmware. Sin embargo, esto implicaría una gran complejidad y con una alta probabilidad de dejar el equipo inutilizable. Dada esta situación, la otra alternativa posible es agregar una capa de software adicional que permita lograr las mejoras en la experiencia de usuario y prestaciones, sin alterar en ningún modo el hardware y software del equipo.

Esta capa de software permitirá activar, desactivar y recibir las notificaciones de eventos de movimiento en una aplicación de mensajería instantánea desde un dispositivo móvil; y por otra parte postprocesar las imágenes recibidas por la cámara, ante un evento de detección de movimiento, para detectar intrusos y categorizar el evento de acuerdo a la presencia o no de personas en la imagen recibida por la cámara.

5.2 Descripción de la arquitectura

A los componentes de la arquitectura original se le suma una capa de software adicional, compuesta por una aplicación Web, un servidor FTP personalizado, un módulo de detección de intrusos en imagenes, y un módulo para enviar y recibir mensajes e imágenes desde y hacia un programa de mensajería instantánea. La *Imagen 15* muestra un esquema de la nueva arquitectura, la cual se puede contrastar con la arquitectura del sistema original mostrada en la *Imagen 12*. Las diferencias más

notorias son que ahora será posible activar y desactivar las alarmas a través de un programa de mensajería instantánea desde Internet; recibir las notificaciones en este mismo programa de mensajería en vez de utilizar un correo electrónico; y además, se agrega un módulo para detectar personas en las imágenes originales, recibidas desde la o las cámaras conectadas al sistema.

En líneas generales, el sistema funcionara de la siguiente manera:

1. La cámara envía las imágenes, cuando se produce un evento de detección de movimiento, a un servidor FTP.
2. Las imágenes son reprocesadas, con una librería de procesamiento de imágenes para detectar intrusos, y enviar en caso positivo una imagen aumentada con esta información.
3. Las imágenes recibidas son enviadas, junto con una notificación de texto, a un programa de mensajería instantánea. La mensajería instantánea puede ser leída desde casi cualquier dispositivo móvil actual, que puede ser un teléfono móvil o una tablet.
4. Desde el programa de mensajería instantánea, también será posible enviar un comando para activar y desactivar la recepción de notificaciones.

Una vez que las imágenes son procesadas por el detector de intrusos, en el caso que el resultado sea positivo se enviará una notificación semánticamente más fuerte que en los casos en donde la detección resulte negativa, adjuntando además la imagen original o la imagen aumentada según sea el caso.

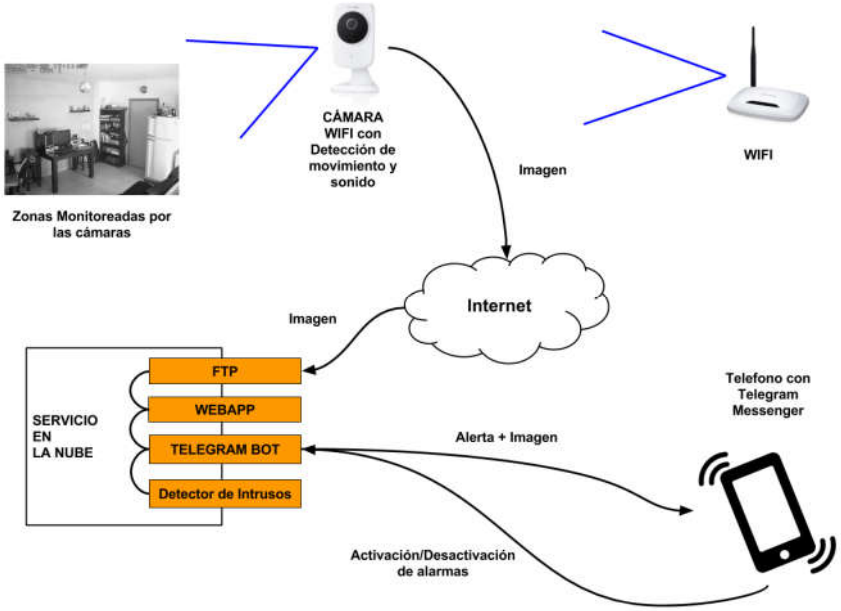


Imagen 15: Arquitectura general del sistema de software que interactúa con la cámara de seguridad TP-LINK nc220 (Fuente: propia)

El sistema como está planteado, soporta la utilización de más de una cámara. Cada cámara, enviará las alertas al mismo servidor FTP. Se pueden utilizar un usuario de conexión por FTP para cada cámara en forma individual o utilizar un único usuario y password compartido entre todos los equipos. También es posible que más de un usuario reciba las notificaciones de movimiento en su dispositivo móvil. Obviamente cuantos más dispositivos y usuarios se conecten al sistema, crecerá la demanda de

recursos por parte del software subyacente, el cual adicionalmente, tiene algunas limitaciones por la forma en que está implementado.

El código fuente de toda la aplicación, los scripts de las diferentes pruebas realizadas con la librería OpenCV y uno de los datasets utilizados en las pruebas de detección de personas, descritas en el Capítulo 5, están disponibles en un repositorio de acceso público en la cuenta personal de Github del autor del trabajo [36].

En las siguientes secciones se describirán en mayor detalle cada componente; explicando su función, las tecnologías de software elegidas, y las diferentes interacciones con el resto de los módulos que componen la aplicación.

5.3 Componentes del Sistema

5.3.1 Aplicación WEB

Todos los módulos del sistema están contenidos en una aplicación WEB implementada con el Framework DJANGO [37]. DJANGO es un framework para el desarrollo de aplicaciones WEB implementado en Python [38], que brinda una gran cantidad de utilidades para desarrollar una aplicación WEB de manera organizada y ágil. Entre las muchas características que brinda se pueden nombrar: el modelado y persistencia de entidades, la autenticación y administración de permisos de usuarios, la renderización de templates y la abstracción del motor de persistencia utilizado. Posee una gran comunidad de usuarios y soporte, lo cual lo hacen un producto maduro y robusto. Otro punto fuerte de DJANGO es la cantidad de módulos adicionales, que es posible agregar como dependencia a la base del framework, para implementar nuevas características y funcionalidades. En el caso, del sistema implementado para este trabajo, se hace uso de las siguientes utilidades de DJANGO:

- **Modelos (Models)**: para representar y persistir las entidades que componen el sistema.
- **Autenticación y permisos**: para validar y autenticar el usuario FTP utilizado por la cámara.
- **Comandos (commands)**: para ejecutar la API de mensajería instantánea y el servidor FTP.
- **Admin**: DJANGO provee una aplicación básica para administrar y editar las entidades que componen el sistema. Se hace uso de esta aplicación para configurar los settings necesarios para que todos los componentes del sistema se puedan ejecutar.

Un proyecto DJANGO se organiza en aplicaciones. Cada aplicación, contiene comportamiento y definiciones agrupadas bajo algún criterio. Dentro de cada aplicación se encuentra la definición de los modelos, que luego se persistirán en la base de datos; comandos; y toda la lógica de negocios relacionada a los modelos. La *Imagen 30* muestra un esquema de cómo está estructurado el proyecto DJANGO implementado para este trabajo.

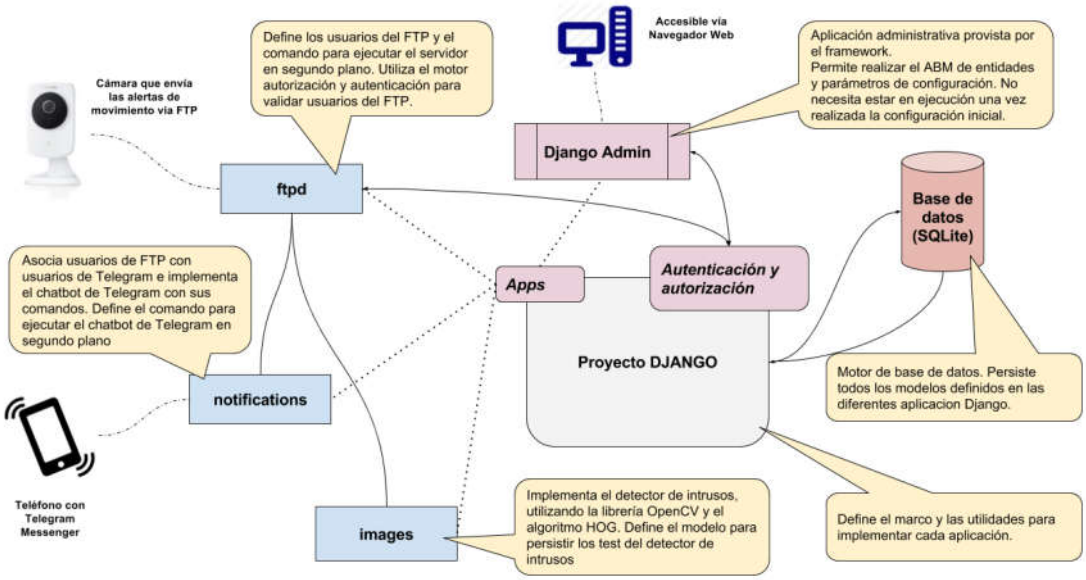


Imagen 16: Organización del proyecto Django implementado para este trabajo (Fuente: propia)

El proyecto Django implementado, se compone de 4 aplicaciones:

- **FTPD**: implementa el servidor FTP que se puede ejecutar a través de un comando. Define el modelo que representa los usuarios del servidor e interactúa con el motor de autenticación y autorización del framework para validar las conexiones entrantes.
- **Notifications**: implementa el chatbot de Telegram que permite recibir comandos desde usuarios de Telegram registrados en el sistema, y enviar las alertas cuando se recibe una imagen en el servidor FTP. También, define un comando para dejar en ejecución el chatbot de Telegram en segundo plano.
- **Images**: implementa el detector de intrusos, y también provee los modelos para registrar las pruebas realizadas en el Capítulo 6. Se estructuró el código de tal manera que sea posible agregar nuevos algoritmos de detección y hacerlos intercambiables.
- **Django Admin**: esta utilidad es provista por el framework, y posibilita con una mínima configuración de código, registrar los modelos definidos en las diferentes aplicaciones del proyecto y generar una Aplicación Web Administrativa que permite realizar el ABM básico de las entidades.

5.3.2 Servidor FTP

La cámara provee dos tipos de notificaciones, ante la detección de eventos de movimiento: email y FTP. Para la aplicación a desarrollar, se utilizó la opción de enviar los screenshots a un servidor FTP. Es necesario un mecanismo que permita disparar un evento, ante la recepción de una nueva imagen enviada por la cámara. A tal fin, se utilizó la librería pyftpdlib [39], que permite implementar de manera personalizable un servidor FTP. Esta librería está desarrollada íntegramente en Python y permite redefinir métodos hooks (*Imagen 17*) que representan los diferentes eventos significativos que pueden ocurrir en un servidor FTP. Los hooks que la librería permite redefinir son los siguientes:

- **on_connect**: se recibe una nueva petición de conexión al servidor.

- **on_disconnect**: se desconecta una conexión previa al servidor.
- **on_login**: un usuario se autentica en el servidor.
- **on_logout**: un usuario termina la sesión actual con el servidor.
- **on_file_sent**: se realiza la descarga de un archivo desde servidor.
- **on_file_receive**: se realiza la subida de un nuevo archivo al servidor.
- **on_incomplete_file_sent**: se realiza la descarga de un archivo desde el servidor.
- **on_incomplete_file_receive**: se realiza la descarga parcial de un archivo desde el servidor.

```python
class NotificationFTPHandler(FTPHandler):

    def __init__(self, conn, server, ioloop=None):
        logger.debug("Initializing FTP Notification handler...")
        super(NotificationFTPHandler, self).__init__(conn, server, ioloop)

    def get_ftp_user(self):
        return FTPUser.objects.get(user__username=self.username)

    def on_connect(self):
        logger.debug("%s:%s connected", self.remote_ip, self.remote_port)

    def on_disconnect(self):
        logger.debug("%s:%s disconnected", self.remote_ip, self.remote_port)

    def on_login(self, username):
        logger.debug("%s logged in!", username)

    def on_logout(self, username):
        logger.debug("%s logged out!", username)

    def on_file_received(self, file):
        logger.info("File received %s", file)
        result = self._pre_process(file)
        self._send_notifications(result)

    def on_incomplete_file_received(self, file):
        logger.info("Incomplete file received %s", file)
        result = self._pre_process(file)
        self._send_notifications(result)
```

Imagen 17: pyftpdlib permite definir hooks para cada evento de un servidor FTP (Fuente: propia)

Un servidor FTP necesita poder autenticar a los usuarios que realizan una petición para subir o descargar un archivo, con lo cual se implementó esta autenticación utilizando el sistema de autenticación provisto por DJANGO. Así, los usuarios definidos mediante la aplicación DJANGO admin, son los mismos usuarios utilizados para autenticarse con el servidor FTP implementado. En la *Imagen 18* se muestra un diagrama de flujos de los eventos que se producen en el servidor cuando la cámara envía una alerta de movimiento, mientras que la imagen *Imagen 19* muestra el fragmento de código donde el servidor FTP valida las conexiones, invocando al motor de autorización y autencionacion provisto por el framework DJANGO.

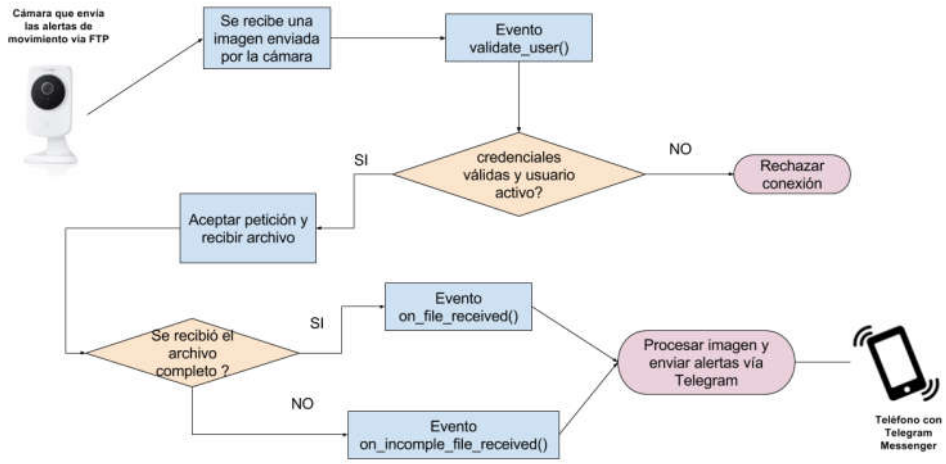


Imagen 18: Diagrama de flujos de los eventos generados por el servidor FTP cuando la cámara envía una alerta de movimiento (Fuente: propia)

```python
class FTPDjangoUserAuthorizer(DummyAuthorizer):
    """
    FTPD Authorizer that use DJANGO users decorated with FTPUser model
    """

    def __init__(self, rootdir):
        self.root = rootdir
        super(FTPDjangoUserAuthorizer, self).__init__()

    def validate_authentication(self, username, password, handler):
        logger.info("Authenticating user %s...", username)
        user = authenticate(username=username, password=password)
        msg = "Authentication failed."
        if not user:
            logger.error(msg)
            logger.debug("Django user does not exist!")
            raise AuthenticationFailed(msg)
        try:
            ftp_user = FTPUser.objects.get(user=user)
            homedir = ftp_user.homedir
            perm = ftp_user.ftpd_perm
            root_homedir = os.path.join(self.root, homedir)
            os.makedirs(root_homedir, exist_ok=True)
            if not self.has_user(username):
                self.add_user(username, password, root_homedir, perm)
        except FTPUser.DoesNotExist:
            logger.error(msg)
            logger.debug("FTP user does not exist!")
            raise AuthenticationFailed(msg)
```

Imagen 19: Integración del sistema de autenticación de DJANGO con pyftpdlib, para autenticar usuarios en el servidor FTP (Fuente: propia)

Por último, se implementó un comando de DJANGO, para ejecutar en segundo plano el servidor FTP, de tal manera que pueda atender los request de conexión y envío de imágenes por parte de la cámara (*Imagen 20*). Un vez recibida una imagen, mediante la redefinición de los hooks apropiados, se podrá realizar acciones sobre la misma. Entre ellas, analizar la imagen para detectar intrusos y enviar las correspondientes notificaciones.

```python
class Command(BaseCommand):
    help = 'Run FTPD server'

    def add_arguments(self, parser):
        parser.add_argument('-t', '--type',
                            default=['async'],
                            dest='stype',
                            choices=NotificationFTPServer.SERVER_TYPE,
                            type=str,
                            help="Concurrency model used by the FTP server",
                            nargs=1,
                            )

    def handle(self, *args, **options):

        stype = options['stype'].pop()
        logger.debug("stype: %s", stype)

        host = settings.FTPD_HOST
        port = settings.FTPD_PORT
        rootdir = settings.FTPD_ROOT
        ftp_server = NotificationFTPServer(host, port, rootdir, stype=stype)
        ftp_server.run()
```

Imagen 20: Comando de DJANGO para ejecutar en background el servidor FTP, que atiende las solicitudes enviadas por la cámara (Fuente: propia)

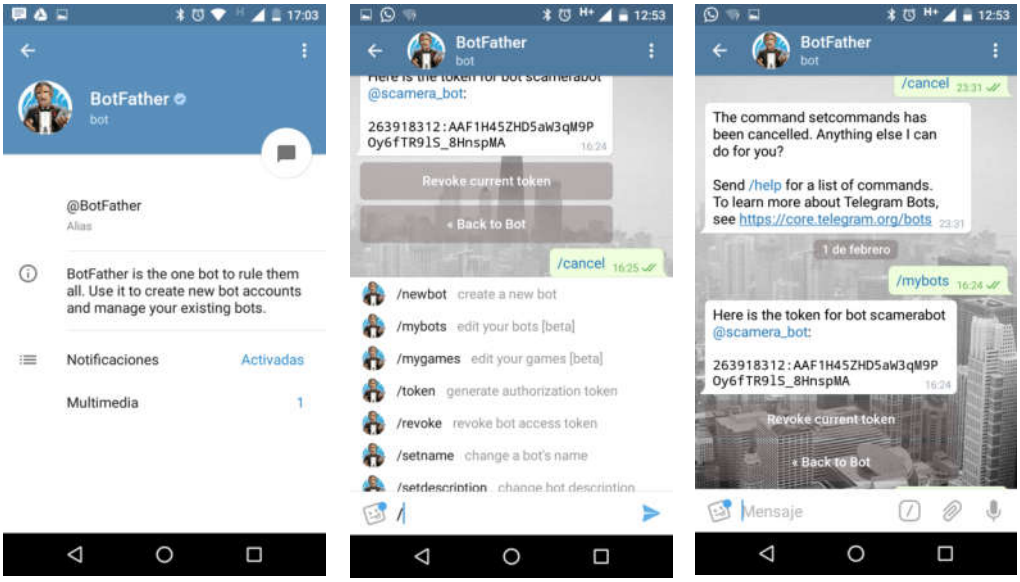
5.3.3 Mensajería Instantánea

Como alternativa al email, como medio para recibir las notificaciones de alerta, se utilizó la posibilidad de recibir dichas alertas en un aplicación de mensajería instantánea. Entre las aplicaciones más populares para mensajería instantánea se pueden nombrar a Whatsapp [40] y Facebook Messenger [41]. Así, el sistema desarrollado permite que las imágenes sean recibidas en Telegram [11]. Telegram es una aplicación de mensajería instantánea, muy similar en características y prestaciones, al popularmente conocido Whatsapp, pero presenta la ventaja sobre su par, de ser un software de código abierto y proveer un conjunto de APIs para implementar entre otras cosas lo que se conoce como chatbots [12]. Un chatbot es básicamente, un programa que responde simulando ser una persona. Es posible implementarlos con distinto nivel de complejidad, ya sea desde un chatbot que es capaz de procesar texto en lenguaje natural y simular que se habla con otra persona, a un chatbot más simple que puede responder a un conjunto de comandos predefinidos.

Para crear un Telegram chatbot es necesario bajar y configurar la aplicación Telegram en un dispositivo móvil y luego acceder al chatbot predefinido @BotFather [42]. Este chatbot predefinido, es un meta chatbot, que permite crear nuevos chatbots de Telegram. A través de la lista de comandos que provee es posible crear un nuevo chatbot, especificando su nombre, un alias, una descripción, la lista de comandos (en caso de que se desee un chatbot que responda a una serie de comandos predefinidos) y finalmente configurar una clave de API o token, el cual servirá para conectarse a la API de Telegram desde las librerías de los diferentes lenguajes de programación. La *Imagen 21* muestra la pantalla inicial del chatbot @BotFather, la lista de comandos que provee para crear y configurar chatbots y la obtención del token para un chatbot particular. Por último, es necesario que cada usuario de Telegram, otorgue permiso a un chatbot para recibir y enviar mensajes.

Para interactuar con la API de chatbots de Telegram se utilizó la librería python-telegram-bot [43], provista por los mismos desarrolladores de Telegram. En este punto, es necesario dar una implementación para cada comando definido cuando se creó el chatbot con el @BotFather. La *Imagen 25* muestra un fragmento del código que implementa el envío de las imágenes recibidas por la cámara. Para recibir actualizaciones desde el chatbot de Telegram, se proveen dos mecanismos mutuamente

excluyentes "getUpdates" y "Webhooks" [12]. Con *getUpdate* se reciben las notificaciones entrantes utilizando long polling; es decir, se ejecuta el código del chatbot en un loop que periódicamente verifica si hay nuevos mensajes para el chatbot en el servidor de Telegram. Mientras que con "Webhooks", es necesario definir una url o endpoint en donde el servidor de Telegram redirigirá las nuevas notificaciones, lo cual implica que dicho endpoint este en un url accesible desde Internet sobre el protocolo HTTPS. En este último caso, es el servidor de Telegram el que redirige los nuevos mensajes recibidos por el chatbot. Por una cuestión de simplicidad, en este trabajo se utilizó la primera opción, y mediante un comando de Django se deja ejecutando en background el chatbot, que periódicamente consulta el servidor de Telegram para obtener nuevas notificaciones.



a b c

Imagen 21: Telegram @BotFahter a) Pantalla inicial b) lista de comandos para crear y configurar chatbots c) Obtención del token para interactuar con la API de Telegram (Fuente: propia)

Para realizar el envío de notificaciones, desde el código del chatbot hacia un usuario de Telegram, es necesario contar con el identificador de Telegram de dicho usuario. El identificador de Telegram se puede obtener mediante algún chatbot implementado por terceros, como @userinfobot o consultado el "chat_id" que se recibe cuando un usuario es el que inicia la conversación con el chatbot. En el sistema implementado es necesario registrar los usuarios de Telegram mediante la aplicación administrativa Django Admin.

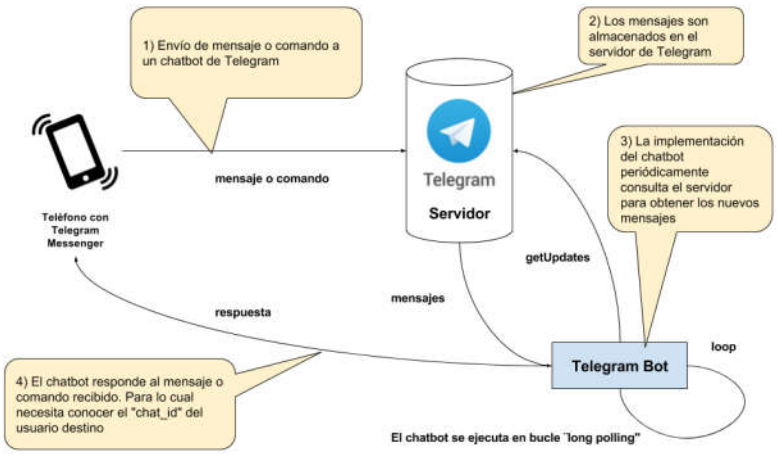
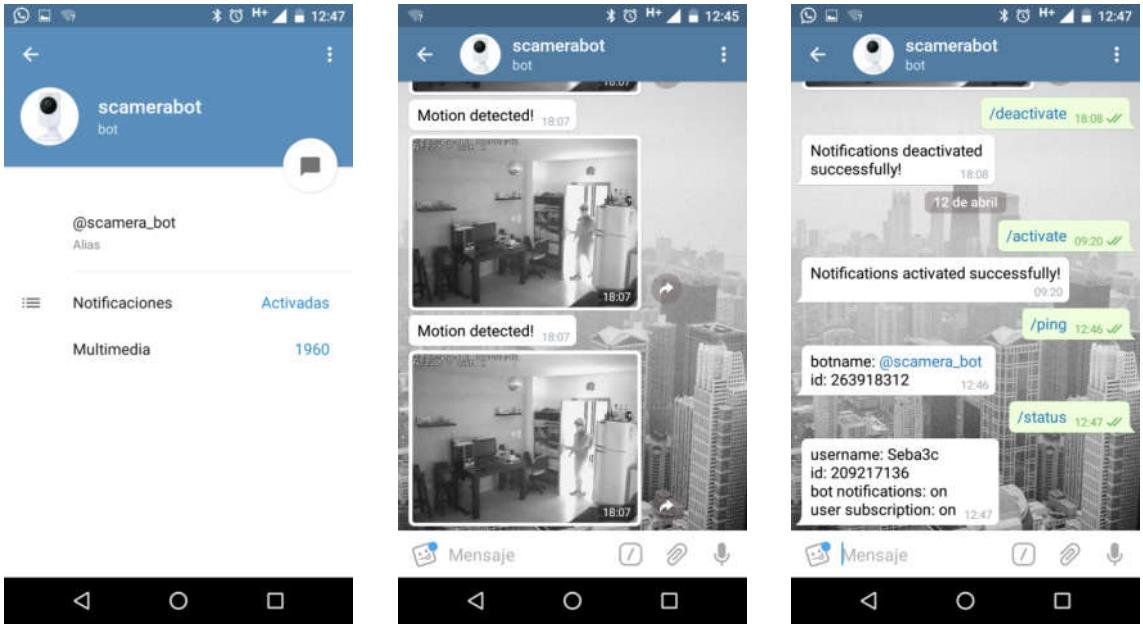


Imagen 22: Interacciones entre la aplicación, el servidor y un chatbot de Telegram ejecutado con long polling (Fuente: propia)

La *Imagen 22* muestra las interacciones que se suceden para enviar mensajes y recibir notificaciones con un chatbot de Telegram, utilizando la técnica de "getUpdates" o long polling. Cabe recordar que todas estas interacciones son posibles si y sólo si el usuario de Telegram, agendo entre sus contactos al chatbot y le concedió permisos para recibir notificaciones.

Entrando más en detalle en el funcionamiento de la aplicación, el servidor FTP al recibir una nueva imagen la procesa utilizando el detector de intrusos, para luego enviar un mensaje junto con la imagen registrada por la cámara. La *Imagen 23* muestra el el aspecto general del chatbot implementado y cómo se visualizan las alertas recibidas.



a b c

Imagen 23: Telegram chabot, para recibir las notificaciones de detección de movimiento de la cámara a) Pantalla inicial del chatbot donde se muestra diferente información del mismo (ej: alias, archivos multimedia recibidos, etc) b) Recepción de un alerta de movimiento en el chatbot c) Interacción con el chatbot a través de los comandos que provee (Fuente: propia)

El chatbot de Telegram, cumplira ademas la funcion de enviar los comandos de activación y desactivación de las notificaciones (*Imagen 26*). Dado que no es posible interactuar con la cámara desde internet para activar y desactivar el envío de notificaciones, la alternativa que se utilizó es la de rechazar las conexiones por parte del servidor FTP. Así, la cámara siempre seguirá activa y enviando notificaciones, con la diferencia que al anular la autenticación desde el servidor FTP las conexiones realizadas por la cámara serán rechazadas. Como el servidor FTP implementado, utiliza el sistema de autenticación de usuarios de DJANGO, lo único que se necesita para lograr el rechazo de las conexiones es setear el flag 'active' a falso en la entidad que representa al usuario dentro de la aplicación DJANGO. Un usuario inactivo, equivale a que el usuario no exista; con lo cual la autenticación será rechazada. El procedimiento inverso, permite volver a activar las notificaciones y recibirlas en el chatbot de Telegram. La desventaja de esta solución, es que no es eficiente desde el punto de vista de utilización de la red de datos y conexión a Internet, puesto que la cámara no deja de enviar peticiones de conexión al servidor en ningún momento, generando un overhead en la red en la cual se encuentra instalada, lo cual puede empeorar si el sistema cuenta con más de una cámara conectadas a la aplicación. La *Imagen 24* muestra un diagram de flujos del proceso de activación y desactivación de alarmas mediante el chatbot.

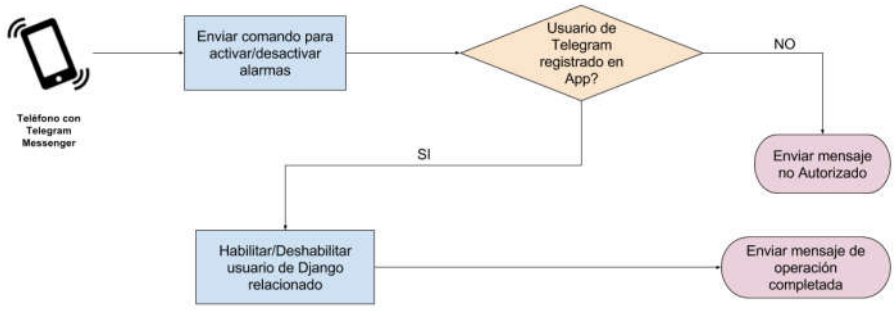


Imagen 24: Diagram de flujos del proceso de activación de desactivación de alarmas mediante el chatbot de Telegram (Fuente: propia)

```
class SCameraBotTelegramHandlers():
    """
    Commands - /setcommands BotFather message
    ping - check availability
    status - current notification status
    register - start to receive notifications
    subscribe - subscription on
    unsubscribe - subscription off
    activate - notifications on
    deactivate - notifications off
    """

    def ping(self, bot, update):

    def status(self, bot, update):

    def subscribe(self, bot, update):

    def unsubscribe(self, bot, update):

    def register(self, bot, update):

    def activate(self, bot, update):

    def deactivate(self, bot, update):

    def _build_updater(self):
        logger.debug("Building telegram bot updater...")
        updater = Updater(self.telegrambot.token)
        updater.dispatcher.add_handler(CommandHandler('ping', self.ping))
        updater.dispatcher.add_handler(CommandHandler('status', self.status))
        updater.dispatcher.add_handler(CommandHandler('subscribe', self.subscribe))
        updater.dispatcher.add_handler(CommandHandler('unsubscribe', self.unsubscribe))
        updater.dispatcher.add_handler(CommandHandler('register', self.register))
        updater.dispatcher.add_handler(CommandHandler('activate', self.activate))
        updater.dispatcher.add_handler(CommandHandler('deactivate', self.deactivate))
        return updater
```

Imagen 25: Código base del Telegram chabot implementado con la librería python-telegram-bot (Fuente: propia)

El nombre del chatbot implementado es "scamerabot", tiene como alias "@scamerabot" y responde a la lista de comandos especificada en la *Tabla 1*.

Comando	Descripción	Explicación
/ping	*check availability*	Permite verificar si el chatbot esta online
/status	*current notification status*	Informa el estado de las notificaciones y en caso de estar en modo DEBUG muestra las estadísticas del test del detector de intrusos en tiempo real (ver Sección 6.2.3)
/register	*start to receive notifications*	No implementado. Permitirá registrar nuevos usuarios en el sistema de alertas en vez de tener que hacerlo íntegramente desde la aplicación administrativ.
/subscribe	*subscription on*	Activa la recepción de mensajes (Las alarmas pueden seguir activadas, pero el usuario no recibirá las notificaciones)
/unsubscribe	*subscription off*	Desactiva la recepción de mensajes
/activate	*notifications on*	Activa el sistema de alarmas (habilita el usuario FTP en DJANGO con lo cual las conexión de la cámara serán aceptadas. Ningún usuario suscrito al chatbot recibirá alertas)
/deactivate	*notifications off*	Desactiva el sistema de alarmas

Tabla 1: Lista de comandos del Telegram chatbot @scamerabot implementado

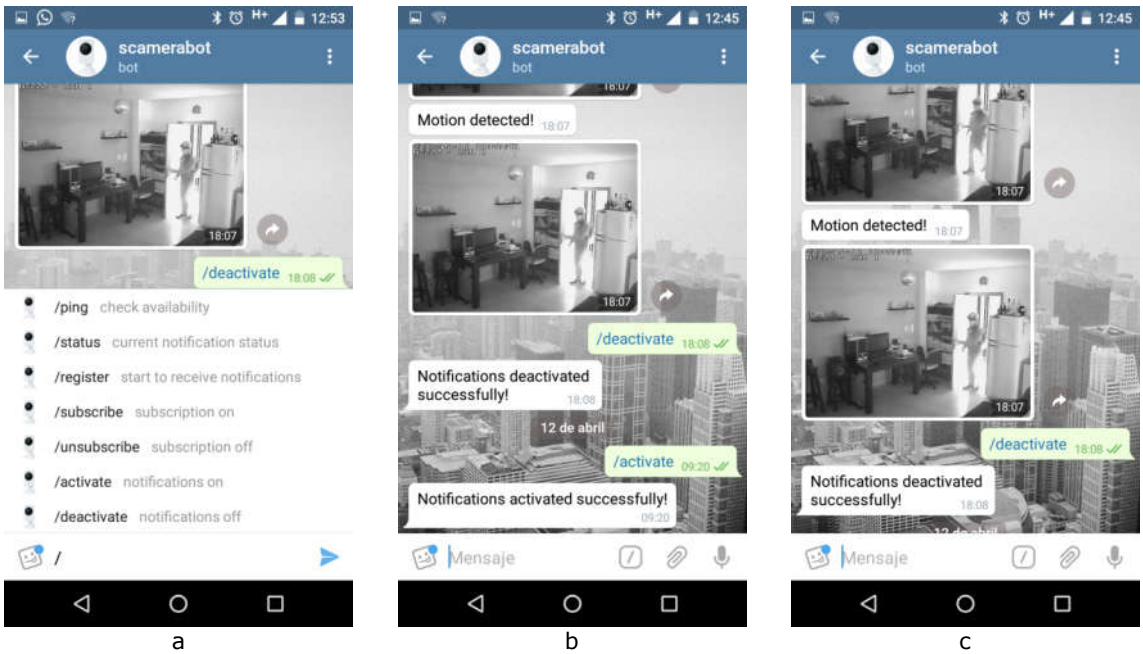


Imagen 26: Comandos del Telegram chatbot a) Lista de comandos disponibles en el chatbot b) Envío del comando para activar las notificaciones c) Envío del comando para desactivar las notificaciones (Fuente: propia)

Al igual que el módulo del servidor FTP, se implementó un comando de DJANGO que deja en ejecución el programa que atiende los mensajes enviados desde Telegram, hacia y desde la aplicación (*imagen 27*).

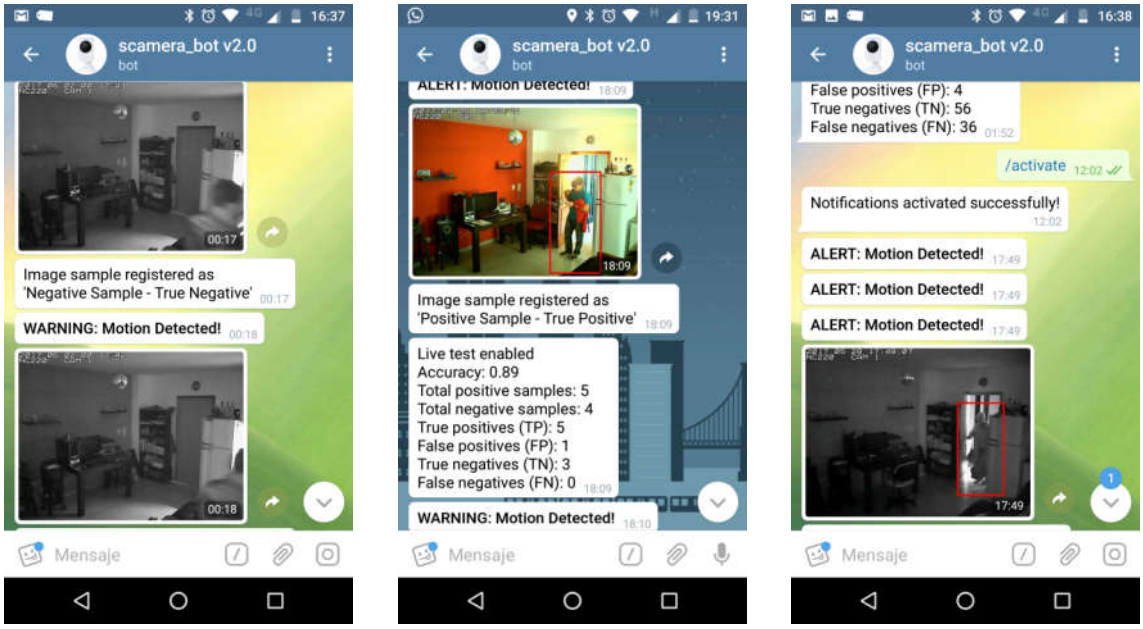
```
class Command(BaseCommand):
    help = 'Run A telegram bot registered in this app'

    def add_arguments(self, parser):
        parser.add_argument('-b', '--botname',
                            dest='botname',
                            type=str,
                            help="Telegram botname to run",
                            nargs=1,
                            required=True
                            )

    def handle(self, *args, **options):

        botname = options['botname'].pop()
        logger.info("Running telegram bot %s", botname)
        TelegramBotRunner.run_telelgram_bot(botname)
```

Imagen 27: Comando de DJANGO para ejecutar en background el telegram chatbot (Fuente: propia)

5.3.4 Detector de intrusos

Finalmente, el sistema contiene un módulo que post procesa las imágenes recibidas, con el objetivo de detectar intrusos (personas) en la misma, y enriquecer la imagen enviada al sistema de notificaciones encuadrando las personas detectadas. En el caso, de detectar intrusos en forma positiva, a su vez se emite un mensaje de mayor importancia al chatbot de Telegram. Como se explicará más adelante, no es viable la opción de ignorar la alerta en el caso que este módulo no detecte intrusos, ya que su precisión no es del 100%, lo cual podría ocasionar el filtrado de un verdadero positivo. Si bien, ambos casos no son deseables en un sistema de seguridad, es preferible enviar una alerta de falso positivo que ignorar una alerta de un verdadero positivo.

Para postprocesar las imágenes, se utilizó la librería OpenCV [10]. Las imágenes recibidas por el servidor FTP son procesadas con este algoritmo, y en caso positivo se envía un mensaje de mayor relevancia al chatbot de Telegram, junto con la imagen aumentada en la cual se resalta la o las personas detectadas (*Imagen 28*).



a b c

Imagen 28: Imágenes recibidas en el chatbot de Telegram.
a) Notificación de falso positivo donde se envía un mensaje de advertencia
b,c) Aquellas imágenes donde efectivamente se detectaron intrusos, son aumentadas encuadrando las personas, y cambiado el mensaje adjunto a la notificación por uno semánticamente más fuerte (Fuente: propia)

El código que implementa la detección de intrusos (*Imagen 29*) toma como base el código presentado por Adrian Rosebrok en [8] y explicado en detalle el sección 3.3, realizando las adaptaciones necesarias, para que sea posible, mediante la aplicación administrativa (*Imagen 30*), configurar los parámetros de la función detectMultiScale (*winStride, scale, padding*); el umbral para eliminar detecciones solapadas de un mismo objeto, mediante la técnica de supresión no máxima; y el ancho máximo para redimensionar las imágenes, previo a realizar el proceso de detección sobre la misma. El código de detección está acoplado, al resto del sistema, de manera tal que sea sencillo agregar nuevos algoritmos de detección de personas, ya sea provistos por OpenCV, otra librería, o incluso implementados desde cero. Esto permitiría eventualmente realizar comparaciones de resultados o elegir el algoritmo más apropiado según los recursos de cómputo disponible o la naturaleza y características de las imágenes a procesar de cada caso de uso particular.

```python
class HOGPeopleDetectorImagePreProcessor(ImagePreProcessor):

    def __init__(self, save_enhaced_imgs=None, ouput_path=None):

        super(HOGPeopleDetectorImagePreProcessor, self).__init__(save_enhaced_imgs,
                                                                 ouput_path)
        # initialize the HOG descriptor/person detector
        self.hog = cv2.HOGDescriptor()
        self.hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
        self.max_width_size = imsettings.image_max_width_size
        self.winStride = (imsettings.win_stride_x, imsettings.win_stride_y)
        self.padding = (imsettings.padding_x, imsettings.padding_y)
        self.scale = imsettings.scale
        self.rect_line_color = imsettings.rect_line_color
        self.rect_line_width = imsettings.rect_line_width
        self.overlapThresh = imsettings.non_maxima_suppression_thresh

    def process(self, path):
        logger.info("Analyzing image: '%s'", path)
        new_path = path
        image = cv2.imread(path)  # IMREAD_GRAYSCALE
        image = imutils.resize(image, width=min(self.max_width_size, image.shape[1]))

        start_time = timeit.default_timer()
        # detect people in the image
        (rects, weights) = self.hog.detectMultiScale(image,
                                                     winStride=self.winStride,
                                                     padding=self.padding,
                                                     scale=self.scale)

        # apply non-maxima suppression to the bounding boxes using a
        # fairly large overlap threshold to try to maintain overlapping
        # boxes that are still people
        rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])
        pick = non_max_suppression(rects, probs=None, overlapThresh=self.overlapThresh)
        object_detected_count = len(pick)
        elapsed_time = timeit.default_timer() - start_time

        if object_detected_count > 0:
            # draw the final bounding boxes
            for (xA, yA, xB, yB) in pick:
                cv2.rectangle(image, (xA, yA), (xB, yB),
                              self.rect_line_color,
                              self.rect_line_width)
            if self.save_enhaced_imgs:
                filename = os.path.basename(path)
                new_path = os.path.join(self.ouput_path, filename)
                cv2.imwrite(new_path, image)

        return ImagePreProcessorResult(new_path, object_detected_count, elapsed_time)
```

Imagen 29: Código que implementa la detección de personas en imágenes mediante Histograma de gradientes orientados utilizando la librería Open CV (Fuente: propia)

5.4 Host para ejecutar la aplicación

Es necesario un computador para ejecutar la aplicación y servicios antes descritos. En este punto existen 2 alternativas:

1. Ejecutar la aplicación en una computadora dentro de la red local donde se encuentra la cámara. Podría ser una PC convencional, un servidor dedicado o incluso una computadora de placa única, como por ejemplo Raspberry PI. Raspberry Pi es un computador de placa reducida, computador de placa única o computador de placa simple (SBC) de bajo coste desarrollado en Reino Unido por la Fundación Raspberry Pi [44], con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas. La única restricción es que dichos dispositivos sean capaces de ejecutar el software implementado.
2. Ejecutar la aplicación en un servidor en la nube, que puede ser propietario o de terceros.

Para este trabajo se utilizó la segunda alternativa, por cuestiones de simplicidad, disponibilidad y costos. Ya que la primer alternativa implica:

- Contar con el equipo adecuado para instalar la aplicación.
- Configurar dicho equipo con el Sistema Operativo y dependencias necesarias para ejecutar la aplicación.
- Mantener el equipo constantemente encendido, lo cual deriva en un mayor costo en el suministro eléctrico; y en caso de que no se utilice un equipo especialmente diseñados para funcionar como servidor, se corre el riesgo de acortar la vida útil de dicho equipo.

Por lo antes expuesto, la aplicación fue instalada en la plataforma provista por Digital Ocean [45]. Esta plataforma, provee un servicio de cloud hosting que cubre diferentes necesidades. Para este caso particular, en un único host se ejecuta el servidor FTP, el bot de Telegram, y eventualmente la aplicación DJANGO admin para realizar la configuración inicial y/o modificar algun parametro de configuración, luego no es necesario que esta última aplicación siga en ejecución para el funcionamiento de todo el sistema de seguridad. Cabe destacar, que en esta primera versión, los componentes desarrollados no están preparados para ejecutarse en forma distribuida, ya que el servidor FTP y el chatbot de telegram, comparten las imágenes a través del sistema de archivos local, y el motor de base de datos utilizado por DJANGO es SQLite [46], el cual guarda los datos en un archivo local.

El plan más económico ofrecido por Digital Ocean, al momento de la realización de este trabajo, es de 5 dólares mensuales por un host con un solo procesador, un disco de 20GB y 6GM de memoria RAM, lo cual son más que suficiente para ejecutar el software implementado, con una única cámara conectada. En caso de tener una mayor cantidad de cámaras conectadas, usuarios suscriptos y cantidad de eventos que se generan, posiblemente sea necesario tener un mayor poder de cómputo y por ende ampliar el plan de suscripción.

5.5 Recursos necesarios y costos del sistema

Como sugiere el título de este trabajo, el sistema de vigilancia propuesto no es costoso de instalar y mantener en términos monetarios. El fundamento utilizado, sin entrar en detalles particulares y subjetivos de qué parámetros determinan si un bien o servicio como es económico o costoso, fue comparar el costo de este sistema contra otros posibles dispositivos de seguridad, como puede ser un kit inicial de alarma con 2 sensores de movimiento. Si se considera que tanto la instalación de una fuente eléctrica, como la conexión a internet son preexistentes en el sitio donde se realizará el monitoreo mediante cámaras, los gastos que implica este sistema son la inversión inicial de la compra de las cámaras, el consumo eléctrico de las mismas y la infraestructura para mantener en ejecución el sistema de software implementado.

Como se mencionó anteriormente, el costo de una camara similar a la utilizada en este trabajo, ronda los mil pesos argentinos (aproximadamente 62 dólares por cámara, al momento de realización de este trabajo).

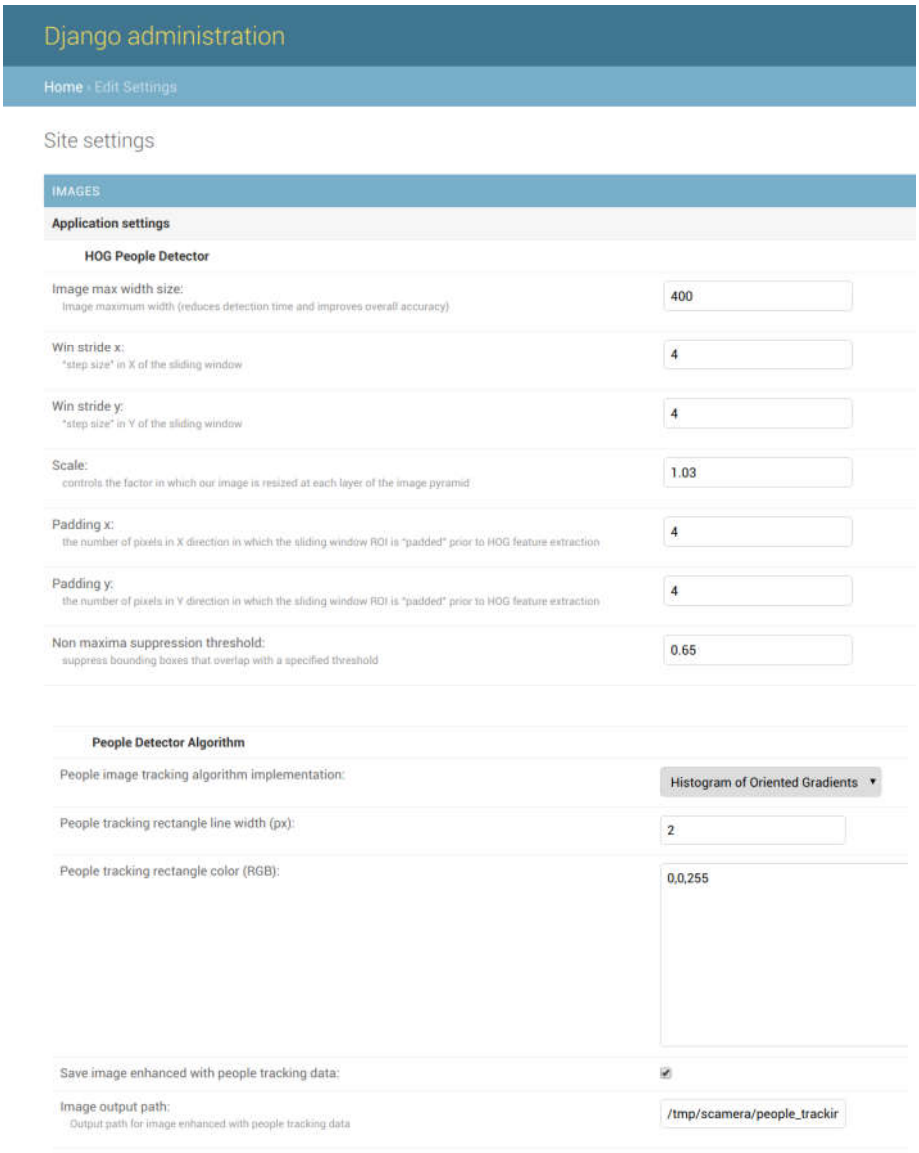


Imagen 30: A través de la aplicación DJANGO admin, es posible setear el valor de cada parámetro del algoritmo de detección de personas HOG implementado con OpenCV (Fuente: propia)

El consumo eléctrico de cada cámara es ínfimo, con lo cual casi es imperceptible en el costo total de consumo eléctrico del área monitoreada. Este dato se pudo comprobar, durante la realización de las pruebas descritas en el Capítulo 6, en donde durante los meses de prueba, no se notó un incremento en las facturas del servicio eléctrico del departamento, por tener la cámara conectada durante las 24 horas del dia.

Por último, es necesario mantener la aplicación de software implementada en ejecución, para que la cámara pueda interactuar con la misma. Si la aplicación es ejecutada en una computadora, servidor o minicomputadora en forma local solo se tendrá el costo adicional de consumo eléctrico de dicho dispositivo. Si por otra parte, la aplicación es instalada en un servicio de terceros, se tendrá como costo extra el pago al proveedor externo del servicio. En este trabajo, se utilizó para desplegar el software adicional, un Droplet de Digital Ocean [45], el cual tiene un costo mensual de 5 dólares.

En relación al costo monetario que insumen otros sistemas de seguridad tales como la compra de una alarma con sensores de movimiento convencional o la contratación de un servicio de seguridad privado, este sistema tiene costo significativamente menor. Cabe destacar que cada alternativa, independientemente de los costos, cubre necesidades de seguridad de diferentes niveles y será necesario evaluar para cada caso particular cuáles son las mejores opciones para aumentar la seguridad de las áreas de interés, según el presupuesto disponible y requerimientos específicos. Por ejemplo, es inviable cubrir los requerimientos de seguridad de un banco o entidad gubernamental con el equipo y sistema utilizados en este trabajo, ya que los requerimientos y niveles de seguridad están muy por encima de los que el sistema puede brindar.

Capítulo 6 - Pruebas Realizadas y Resultados Obtenidos

Este capítulo, describe las diferentes pruebas realizadas para verificar el funcionamiento y desempeño del software adicional implementado; así como también el desempeño y funcionalidad provistas por el sistema comercial original. En la Sección 6.1, se presenta el concepto de usabilidad y se comparan mediante un test de usabilidad las interacciones del usuario con el sistema comercial y con el sistema mejorado. En la Sección 6.2, se realiza una prueba para evaluar el desempeño del detector de movimiento embebido en el sistema comercial, también se describen las características del entorno donde se realizaron las pruebas. Por último, la Sección 6.3 presenta las diferentes pruebas realizadas para evaluar el desempeño del detector de intrusos, que forma parte del conjunto de mejoras introducidas por el software implementado y descrito en el capítulo anterior.

6.1 Usabilidad

6.1.1 Definición y evaluación

En el contexto específico de la ingeniería de software, se puede definir a la usabilidad de la siguiente manera:

"La usabilidad es el grado en el cual un software puede ser utilizado por un conjunto de usuarios específicos para alcanzar objetivos específicos con efectividad, eficiencia y satisfacción dentro de un contexto de uso específico." [47]

Existen una gran variedad de metodos y tecnicas de diversa complejidad, para evaluar la usabilidad de un sistema de software. Entre ellas se encuentra test de usabilidad SUS (*System Usability Scale*) descrito por John Brooke en [48]. SUS es un test de usabilidad simple y rápido que consiste de 10 preguntas, que aceptan como respuesta un valor dentro de una escala que puede ir de 1 a 5 o 1 a 7 (*Tabla 2*).

El usuario responde las 10 preguntas, luego de interactuar con el sistema para luego poder calcular la puntuación SUS (*SUS Score*) que varía entre 0 y 100. Esta puntuación, permite obtener una visión global de las valoraciones subjetivas de usabilidad de un sistema. Para calcular la puntuación SUS, en primer lugar se suman las puntuaciones de cada ítem. Cada ítem tendrá una puntuación que va en el rango de 0 a 4. Para los ítems 1,3,5,7 y 9 la puntuación es la posición de la escala menos 1. Mientras que para los ítems 2,4,6,8 y 10 la puntuación está dada por 5 menos la posición de la escala. Por último, se multiplica la suma de las puntuación de cada ítem por 2.5 para obtener así el valor total para la usabilidad del sistema.

	Preguntas	Totalmente en desacuerdo	1	2	3	4	5	Totalmente de acuerdo
1	Pienso que me gustaría usar este sistema frecuentemente							
2	Encuentro que el sistema es innecesariamente complejo							
3	Encuentro que el sistema es fácil de usar							
4	Pienso que necesitaría la ayuda de una persona técnica para usar el sistema							
5	Encuentro que las diferentes funciones del sistema están bien integradas							
6	Pienso que hay mucha inconsistencia en el sistema							
7	Me imagino que mucha gente aprendería a usar el sistema muy rápido							
8	Encuentro que el sistema es muy engorroso para usar							
9	Me sentí muy confiado al usar el sistema							
10	Necesite aprender muchas cosas antes de poder usar el sistema							

Tabla 2: Cuestionario para el test de usabilidad SUS (*System Usability Scale*) descrito por John Brooke en [48]

6.1.2 Prueba de usabilidad

Se pueden identificar dos interacciones que los usuarios realizarán habitualmente, tanto con el sistema comercial original como con la versión mejorada descrita en este informe:

1. Activar y desactivar el sistema de alertas.
2. Visualizar una alerta recibida.

La *Tabla 3*, muestra una comparación de las acciones que un usuario debe llevar a cabo para activar y desactivar el sistema de alarmas, utilizando la aplicación Web embebida en la cámara (*Imagen 31*) y a través del chatbot de Telegram (*Imagen 32*). Mientras, que en la *Tabla 4* se muestra la comparación de las acciones a realizar por un usuario, para visualizar una alarma de movimiento utilizando un correo electrónico (*Imagen 33*), y a través el chatbot de Telegram (*Imagen 34*).

Para poder comparar la usabilidad de estas 2 acciones, mediante los mecanismos provistos por el sistema comercial y los nuevos mecanismos agregados con el software implementado, se realizó un test de usabilidad SUS.

La prueba realizada consistió en que 6 usuarios, realicen las dos interacciones antes mencionadas, con los mecanismos disponibles en el sistema comercial original y mediante el uso del chatbot de Telegram. Luego, de interactuar con cada alternativa, se solicitó a cada usuario, que complete el cuestionario de usabilidad SUS con valores en la escala de 1 a 5. La *Tabla 5*, muestra el valor promedio asignado para cada pregunta del cuestionario y la puntuación SUS promedio para cada

alternativa de interacción.

De los resultados obtenidos, se puede observar que los usuario manifestaron, una mejor valoración subjetiva de la usabilidad del sistema, mediante la interacción con el chatbot de Telegram con una puntuación SUS promedio de 95; sobre la aplicación Web embebida en la cámara y la revisión de una casilla de correo electrónico, que obtuvo una puntuación SUS promedio de 57.08.

		Aplicación Web embebida en la cámara	**Telegram Bot**
Precondiciones		Estar conectado con una computadora o dispositivo móvil a la WIFI donde está instalada la cámara. Estar dentro del alcance de la señal de la red WIFI.	Tener un dispositivo móvil con conexión a internet y que posea Telegram y esté registrado para operar con el chatbot *@scamerabot*.
Acciones	**1**	Abrir un navegador WEB	Abrir la aplicación Telegram
	2	Tipear la dirección IP de la cámara	Buscar y seleccionar el chatbot *@scamerabot* dentro de los contactos
	3	Ingresar usuario y password para acceder a la aplicación Web embebida en la cámara	Clickear en la lista de comandos [/]
	4	Clickear Login	Clickear */activate* para activar y */deactivate* para desactivar las alertas
	5	Ir a la sección Advanced	
	6	Ir a la sección Motion Detection	
	7	Seleccionar entre las opciones Enabled y Disabled para activar y desactivar las alertas respectivamente	
	8	Clickear Save	

Tabla 3: Acciones necesarias para llevar a cabo la activación y desactivación de las alertas del sistema, utilizando la aplicación Web embebida en la cámara y utilizando el chatbot de Telegram

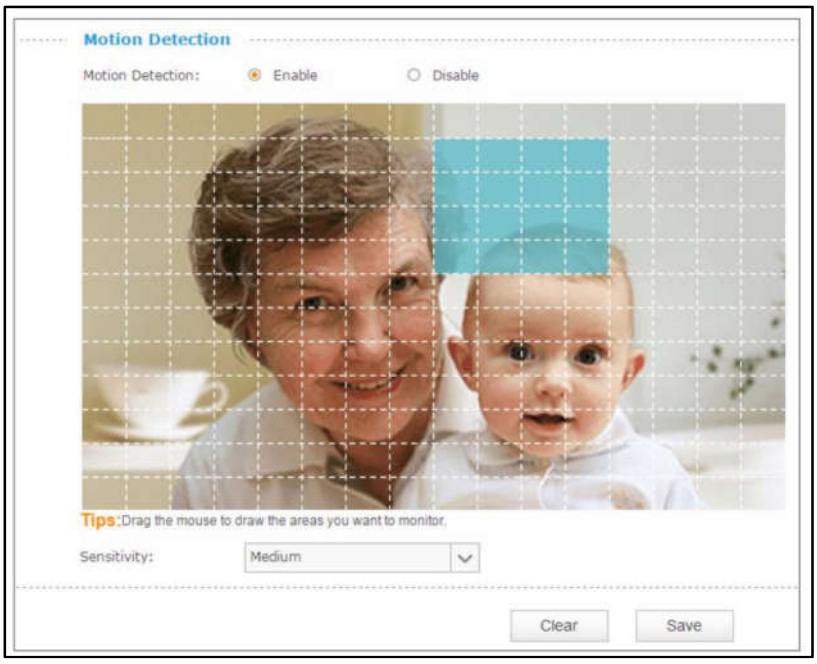
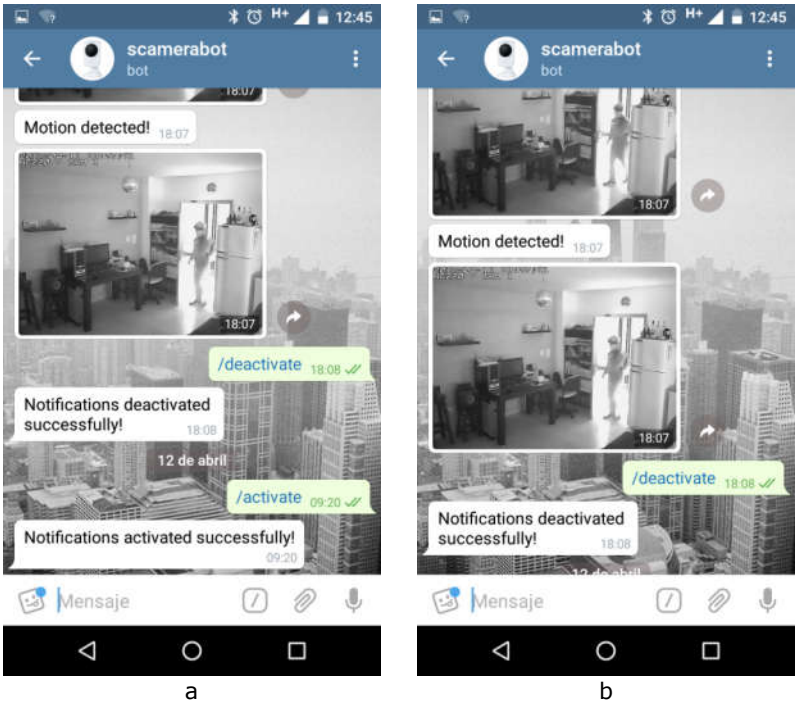


Imagen 31: Sección de la aplicación WEB embebida en la cámara para activar y desactivar las notificaciones por email y FTP (Fuente: [49])

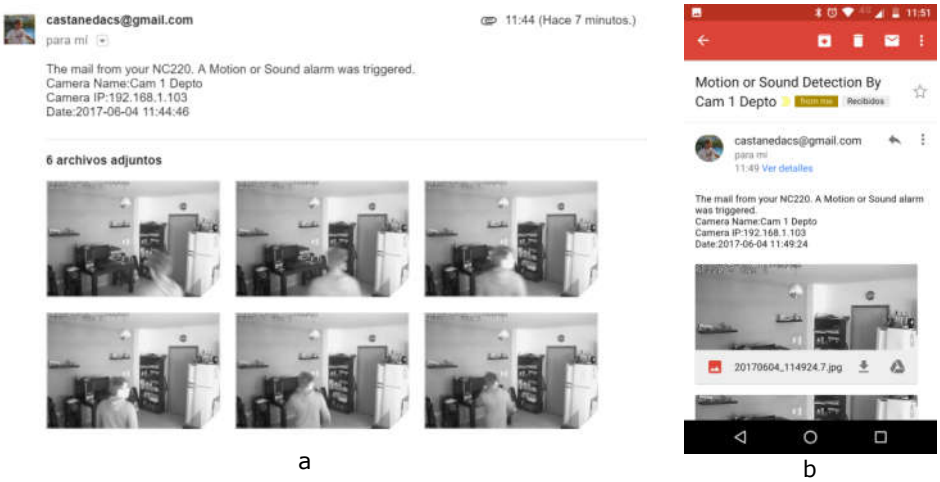


a b

Imagen 32: Activación y desactivación de las notificaciones a través del Telegram chatbot a) Activación b) Desactivación (Fuente: propia)

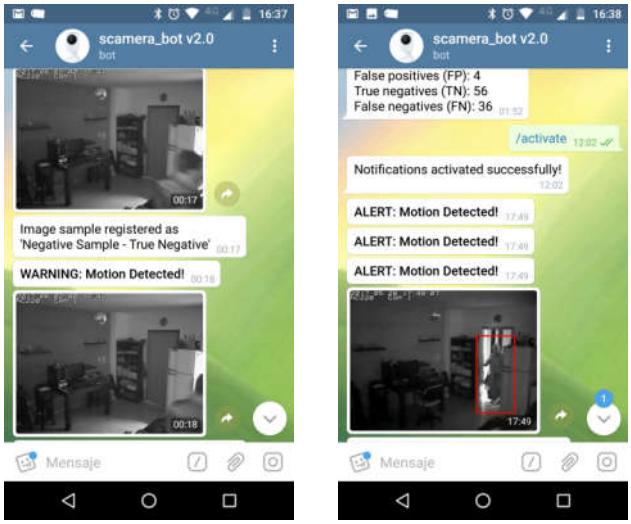
		Email	Telegram Bot
Precondiciones		Tener un dispositivo con conexión a internet con acceso a la casilla de correo electrónico configurada para recibir las alertas de la cámara	Tener un dispositivo móvil con conexión a internet y que posea Telegram y esté registrado para operar con el chatbot @scamerabot.
Acciones	**1**	Abrir la aplicación de correo electrónico	Abrir la aplicación Telegram
	2	Identificar el email de alerta en la bandeja de entrada	Buscar y seleccionar el chatbot @scamerabot dentro de los contactos
	3	Abrir el email	Visualizar los últimos mensajes recibidos en el chatbot
	4	Abrir los archivos adjuntos en el email	

Tabla 4: Acciones necesarias para visualizar un alerta de movimiento, utilizando una casilla de email y utilizando el chatbot de Telegram



a b

Imagen 33: Emails enviados por la cámara ante un evento de detección de movimiento o sonido. a) Email recibido en webmail b) el mismo email visualizado desde un teléfono móvil (Fuente: propia)



a b

Imagen 34: Imágenes recibidas en el chatbot de Telegram. a) Notificación de falso positivo b) Notificación de verdadero positivo (Fuente: propia)

Cantidad de usuarios encuestados		6	**Aplicación Web (Provista por el sistema comercial)**	**Telegram Bot (Implementado en este trabajo)**
1	Pienso que me gustaría usar este sistema frecuentemente		3.5	4.83
2	Encuentro que el sistema es innecesariamente complejo		4.0	1.5
3	Encuentro que el sistema es fácil de usar		3.0	5.0
4	Pienso que necesitaría la ayuda de una persona técnica para usar el sistema		1.33	1.0
5	Encuentro que las diferentes funciones del sistema están bien integradas		3.33	4.83
6	Pienso que hay mucha inconsistencia en el sistema		3.0	1.33
7	Me imagino que mucha gente aprendería a usar el sistema muy rápido		2.16	4.33
8	Encuentro que el sistema es muy engorroso para usar		3.66	1.0
9	Me sentí muy confiado al usar el sistema		4.0	4.83
10	Necesite aprender muchas cosas antes de poder usar el sistema		1.16	1.0
Score Promedio			22.83	38
SUS Score Promedio			**57.08**	**95**

Tabla 5: Resultados del test de usabilidad SUS realizado para evaluar la usabilidad del sistema comercial original y el sistema comercial mejorado con el software implementado

6.2 Evaluación de la detección de movimiento embebida en la cámara

6.2.1 Área monitoreada en las pruebas

El software de detección de movimiento de la camara TP-LINK nc220, es sensible al entorno, en especial a cambios de luminosidad en el ambiente. Es decir, que el desempeño de la cámara está determinado en gran medida por las características del entorno en donde es instalada, con lo cual es necesario describir el lugar de instalación, utilizado para desarrollar las pruebas. Esto ayudará a mejorar la interpretación de los resultados obtenidos.

La *Imagen 35*, muestra un croquis del área donde se instalo la camara utilizada en este trabajo. Como puede observarse, la cámara fue instalada en un departamento de un ambiente, que posee una puerta y dos ventanas, el lente de la cámara apunta hacia la puerta del departamento y se configuró la detección de movimiento en el área circunscripta a la puerta con una sensibilidad media con el fin de evitar una alta tasa de alertas de falsos positivos y la omisión de alertas de verdaderos positivos. Además, la cámara se configuró en modo automático, ya que como se explicó anteriormente el ambiente donde se realizaron la pruebas no tiene una luminosidad constante, con lo cual resultará más efectivo, al momento de monitorear el área, que la cámara adapte su lente en forma automatica segun la cantidad de luz presente en el ambiente. Otro punto a tener en cuenta, es que durante gran parte del día la luz del sol ingresa al departamento, cambiando la luminosidad dentro del mismo, en muchas ocasiones de manera brusca principalmente cuando se trata de días soleados con grandes bancos de nubes.

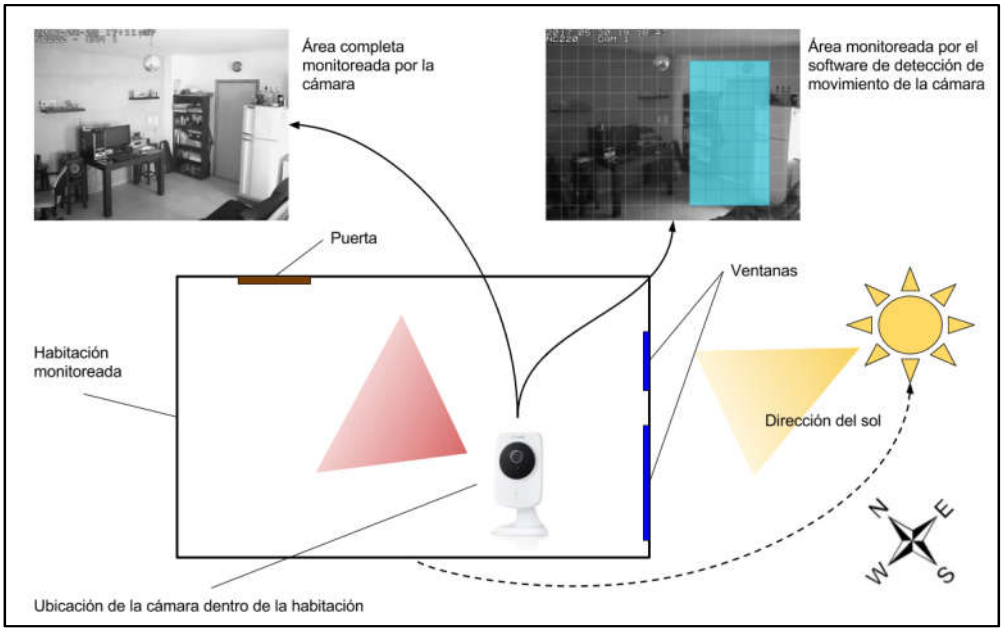


Imagen 35: Croquis del ambiente donde se instaló la cámara, para las pruebas del sistema de vigilancia desarrollado (Fuente: propia)

6.2.2 Prueba de detección de movimiento

La primer prueba realizada, consistió en contabilizar la cantidad de falsos positivos registrados por la cámara. En esta prueba se utilizó el software implementado sólo para recibir las notificaciones en Telegram, sin hacer uso del módulo para detección de personas. En otras palabras, en el chatbot de Telegram, se reciben exactamente las mismas imágenes que la cámara envió originalmente al servidor FTP sin realizar ningún post procesamiento.

La prueba se ejecutó durante 185 días, recolectando las imágenes enviadas por la cámara en los momentos en que el departamento estaba vacío y con el sistema activado, y también forzando la activación de detección de movimiento ingresando al departamento antes de proceder a su desactivación vía Telegram. Durante este periodo se recolectaron 2149 imágenes.

Dado que, ante un evento de detección de movimiento, la cámara envía en promedio entre 4 a 6 screenshots consecutivos, se consideró cada secuencia como una única muestra, reduciendo así las 1683 imágenes a 309 secuencias o muestras. Luego de una inspección en forma manual (observando una a una cada secuencia de imágenes), se obtuvieron los siguientes resultados:

- 199 muestras positivas: Son aquellas secuencias en donde se observan personas, hubo un movimiento real de algún objeto (ejemplo: la apertura de la puerta del departamento, pero sin el ingreso de ninguna personas), y secuencias con un cambio repentino de luminosidad a causa de encender la luz del departamento. En estas últimas secuencias no es posible visualizar nada ya que las imagenes estan completamente quemadas por la luz, pero se conoce de antemano que se originaron por un cambio forzado al encender la luz del departamento, con lo cual se consideran como una muestra positiva.
- 110 muestras negativas: Son aquellas muestras donde no hubo un movimiento real en la escena, y las alertas fueron ocasionadas por cambios en la luminosidad del ambiente, brillos y reflejos de los objetos de la escena monitoreada.

De los datos recolectados, en esta primer prueba, se pueden realizar las siguientes observaciones:

- La cámara notificó un 35,60% de falsos positivos, siendo el 64,40% restante notificaciones de verdaderos positivos. La *Imagen 39* muestra cuatro ejemplos de muestras de verdaderos positivos; es decir, momentos en los cuales hubo un movimiento real en el área monitoreada, generados por presencia de personas, movimiento real de algún objeto, o cambio de luminosidad repentino generado por luz artificial. Este último, caso se genera por la tardanza que tiene el lente de la cámara en adaptarse a la nueva intensidad de luz y cambiar entre modo nocturno y diurno.

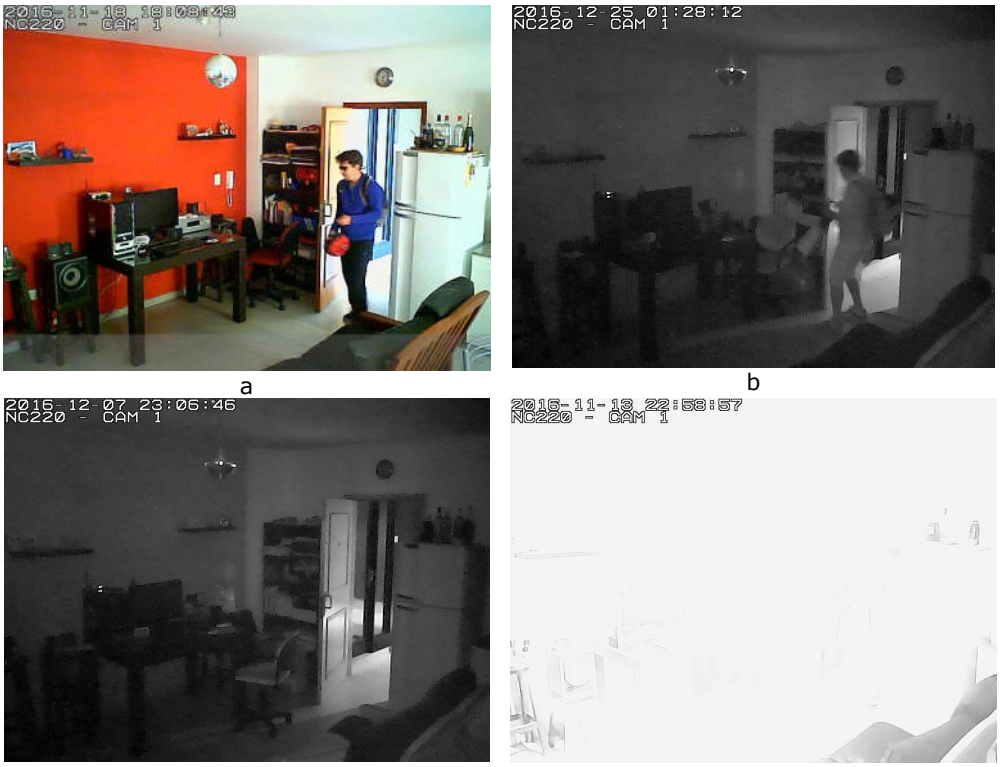


Imagen 36: Ejemplos de verdaderos positivos registrados por la cámara al detectar movimiento en el área monitoreada a) Presencia de personas (muestra tomada con la cámara en modo diurno) b)Presencia de personas (muestra tomada con la cámara en modo nocturno) c) Movimiento real de objetos (apertura de la puerta) d) Cambio de la luz artificial del área monitoreada (Fuente: propia)

- De un total de 110 muestras negativas, 86 se registraron en horario diurno, principalmente por cambios bruscos en la luminosidad del ambiente. Estos cambios fueron más comunes en días soleados con bancos de nubes, donde abruptamente el sol es cubierto y la cantidad de luz que ingresa en el ambiente cambia repentinamente. Se registraron 14 falsos negativos, en horario nocturno, debido al brillo reflejado sobre alguno de los objetos de la escena monitoreada. Por último, las 10 muestras restantes también se produjeron en horario nocturno, pero no es apreciable visualmente en la imagen una causa específica, como en los casos anteriores donde el brillo de los objetos quedó registrado como parte de la imagen. La *Imagen 37* muestra 3 ejemplos de falsos negativos, de cada uno de los motivos antes mencionados.

a b c

Imagen 37: Ejemplos de falsos positivos registrados por la cámara al detectar movimiento en el área monitoreada a) Falso positivo registrado durante la luz del día b) Falso positivo registrado durante la noche generado por el brillo reflejado sobre alguno de los objetos de la imagen c) Falso positivo registrado en horario nocturno (Fuente: propia)

Dada la cantidad de falsos negativos registrados y teniendo en cuenta las causas que los generaron, es posible concluir que el algoritmo de detección embebido en la cámara carece de robustez, como sistema de vigilancia basado en video, ya que es sensible o se ve afectado por los cambios que se producen en el entorno, como cambios en la luminosidad y el brillo, reflejo y sombras que pueden producir los objetos en el campo visual de la cámara.

La *Tabla 6* resumen toda la información recolectada y explicada en esta sección.

Días de recolección de datos	**185**	
Imágenes recolectadas	**1681**	
Secuencias (secuencia de imágenes que forman 1 alerta)	**309**	**100%**
Secuencias positivas (presencia de personas o movimiento real de algún objeto)	**199**	**64.40%**
presencia de personas	176	88.44%
movimiento real de objetos (apertura de puerta)	11	5.52%
cambio brusco de luminosidad (encendido de luces)	12	6.03%
Secuencias negativas (sin presencia de personas o movimiento real de algún objeto)	**110**	**35.60%**
diurnas (cambio brusco de luz natural)	86	78.18%
nocturnas (brillo o reflejo de objetos)	14	12.72%
nocturnas (sin visualización clara de la causa)	10	9.09%

Tabla 6: Resumen de los datos obtenidos en la prueba de evaluación del algoritmo de detección de movimiento embebido en la cámara.

6.3 Evaluación de la detección de movimiento mejorada con el detector de intrusos

6.3.1 Ajuste de parámetros

La siguiente prueba consistió en evaluar la exactitud del detector de intrusos. Como se explicó en el Capítulo 5, cuando la cámara activa la alarma de detección de movimiento, las imágenes recibidas en el servidor FTP, son analizadas por el detector de intrusos, y en caso positivo se envía la imagen enriquecida reencuadrando la o las personas detectadas (*Imagen 38*).

Antes de utilizar el algoritmo de detección de intrusos en conjunto con el resto del sistema de vigilancia mejorado, fue necesario ajustar el valor de los parámetros utilizados por el mismo, los cuales fueron explicados en detalle en la sección 3.3.1. El objetivo asignar el valor más adecuado para scale, padding, winStride, ancho máximo de las imágenes redimensionadas y umbral para eliminar solapamientos en las detecciones positivas; para obtener el mejor balance entre velocidad de ejecución y exactitud de detección.

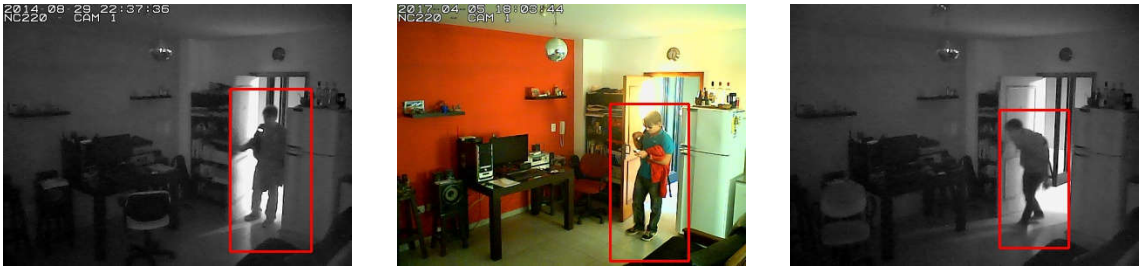


Imagen 38: Imágenes procesadas con el detector de intrusos
donde se resaltan las personas presentes en cada imagen (Fuente: propia)

Como primera acción fue necesario armar un conjunto de test sobre el cual ejecutar el detector de intrusos y que permita ir ajustando el valor de cada parámetro hasta llegar al mejor balance entre exactitud de detección y velocidad de procesamiento posible. Con lo cual se armaron dos conjuntos de prueba, compuestos de la siguiente manera:

Conjunto de muestras 1	Conjunto de muestras 2
25 muestras positivas	786 muestras positivas
25 muestras negativas	530 muestras negativas

Donde:

- muestra positiva = presencia de una o más personas en la imagen; en forma total, parcial y en diferentes poses (paradas, caminando, agachadas, etc). Con presencia se quiere significar que son visibles al ojo humano.
- muestra negativa = ausencia de personas en la imagen, no visibles al ojo humano.

En este caso las 1681 muestras recolectadas fueron clasificadas en muestras positivas y negativas según o no contengan personas visibles al ojo humano. Algunas muestras que se habían

considerado como positivas en la prueba anterior, aquí pasan a ser muestras negativas, tal es el caso de aquellas muestras donde la imagen está completamente saturada de luz artificial y no es posible distinguir ninguń objeto (*imagen 39*.d). El primer conjunto de muestras se compone de 25 muestras positivas y 25 muestras negativas, con muestras bien diferentes en luminosidad, color, contraste, ubicación de las personas en el caso de las muestras positivas, presencia de brillos y reflejos en las muestras negativas, y tiene como objetivo hacer una primera prueba que no demande procesar una cantidad grande de muestras, para encontrar los potenciales mejores valores de los parámetros y luego confirmarlos en una segunda pasada utilizando el segundo conjunto, el cual está compuesto por un número mayor de muestras extraídas también del total de 1681 muestras recolectadas en la anterior prueba.

Para poder ejecutar y luego visualizar los resultados sobre cada conjunto de pruebas, dentro de la aplicación de DJANGO "images" se implementó un comando, que recibe como parámetro un directorio con imágenes positivas y otro con imágenes negativas y ejecuta el algoritmo de detección guardando los resultados en un modelo de DJANGO que luego puede visualizarse desde la aplicación administrativa (*imagen 40*). Para cada ejecución se registra el tiempo tiempo total, el tiempo por imagen, y todas las métricas para evaluar un clasificador binario propuestas por Tom Fawcett en [50] (ver sección 6.4.2).

De esta manera, se realizó la configuración de los parámetros del detector de intrusos con los valores iniciales sugeridos por Adrián Rosebrock en [8], y se ejecutó iterativamente el algoritmo sobre el conjunto de muestras más pequeño, variando gradualmente los valores para los parámetros scale, padding, winStrinde y tamaño máximo de redimensionado de imágenes; observando en cada caso, si la exactitud aumenta o disminuye y el tiempo de ejecución por imagen se incrementa o decrementa, respectivamente. La *Tabla 7*, muestra los valores iniciales utilizados, y la unidad de incremento y decremento para cada parámetro, entre cada prueba ejecutada. Luego, para aquellas pruebas en donde se obtuvo la mejor relación entre exactitud y tiempo de ejecución, se ejecutó el algoritmo nuevamente sobre el conjunto de muestras más grande, a fin de confirmar los resultados obtenidos. En la sección 6.4.3, se muestran los resultados obtenidos para las pruebas y cuáles fueron los valores finalmente usados para los parámetros del algoritmo en la integración del detector de intrusos con el resto del sistema de vigilancia.

Parametro	Valor Inicial	Unidad de incremento y decremento
image_max_width_size	400	+-10 pixels
win_stride	(4,4)	+-1 pixel
scale	1.05	+-0,01
padding	(8,8)	+-1 pixel
non_maxima_suppression_thresh	0.65	no modificado

Tabla 7: Valores iniciales utilizados para realizar el ajuste de los parámetros del detector de intrusos y unidad de incremento y decremento utilizada en cada iteración

El parámetro *non_maxima_suppression_thresh*, se mantuvo siempre con el valor 0.65, ya que no está directamente relacionado con el desempeño del detector, sino que solo sirve a fines de mejorar la visualización de las detecciones sobre cada imagen, tal como se explicó en la sección 3.3.1.

ID	TITLE	STATE	EXECUTED	TIME TOOK	ACCURACY	AVG TIME	POSITIVE SAMPLES	NEGATIVE SAMPLES	DISCARDED SAMPLES	TOTAL SAMPLES	TP	FP	TN	FN
18	Test	finished	May 23, 2017, 3:52 a.m.	139.28 secs	0.76	0.11 secs (9.25 samples/s)	786	503	0	1289	559	227	420	83
19	Test	finished	May 23, 2017, 3:55 a.m.	174.93 secs	0.75	0.14 secs (7.37 samples/s)	786	503	0	1289	580	206	392	111
14	Test	finished	May 23, 2017, 3:24 a.m.	119.23 secs	0.75	0.09 secs (10.81 samples/s)	786	503	0	1289	532	254	431	72
17	Test	finished	May 23, 2017, 3:50 a.m.	85.98 secs	0.74	0.07 secs (14.99 samples/s)	786	503	0	1289	500	286	453	50
16	Test	finished	May 23, 2017, 3:37 a.m.	661.86 secs	0.74	0.51 secs (1.95 samples/s)	786	503	0	1289	484	302	466	37
13	Test	finished	May 23, 2017, 3:21 a.m.	66.32 secs	0.73	0.05 secs (19.44 samples/s)	786	503	0	1289	464	322	483	20
15	Test	finished	May 23, 2017, 3:27 a.m.	471.94 secs	0.71	0.37 secs (2.73 samples/s)	786	503	0	1289	657	129	255	248
4	Test	finished	May 23, 2017, 3:13 a.m.	2.33 secs	0.86	0.05 secs (21.48 samples/s)	25	25	0	50	19	6	24	1
8	Test	finished	May 23, 2017, 3:16 a.m.	2.35 secs	0.86	0.05 secs (21.28 samples/s)	25	25	0	50	19	6	24	1
6	Test	finished	May 23, 2017, 3:14 a.m.	2.36 secs	0.86	0.05 secs (21.21	25	25	0	50	19	6	24	1

Imagen 42: Visualización de resultados del algoritmo de detección de personas en DJANGO admin (Fuente: propia)

6.3.2 Métricas utilizadas

Para medir el desempeño del algoritmo se utilizaron las métricas para evaluar un clasificador binario, presentadas por Tom Fawcett en [50], ya que justamente el algoritmo utilizado tiene dos posibles salidas o categorías: la presencia o la ausencia de intrusos en la imagen analizada. La *Tabla 8*, muestra todas las métricas registradas para las pruebas de ajuste de parámetros, y también para las pruebas de integración con el resto del sistema de vigilancia.

De todas las métricas disponibles, se tomó como referencia principal la exactitud (accuracy) de la clasificación. En caso de tener más de una prueba con el mismo valor de exactitud, para diferentes valores de los parámetros del algoritmo de detección, se considera como mejor resultado la que haya insumido menos tiempo de ejecución.

Metrica	Formula	Descripción
Accuracy ACC	(TP + TN)/(P + N)	Exactitud. Fracción de todas las instancias correctamente clasificadas.
Balanced accuracy (BACC)	(TP/P + TN/N)/2	Exactitud balanceada.
Sensitivity (TPR)	TP/P = TP/(TP + FN)	Tasa de verdaderos positivos (true positive rate)
Specificity (SPC)	TN/N = TN/(FP + TN)	Tasa de verdaderos negativos (false negative rate)
Precision (PPV)	TP/(TP + FP)	Valor predictivo positivo (positive predictive value)
Negative predictive value (NPV)	TN/(TN + FN)	Valor predictivo negativo (negative predictive value)
Fall-out (FPR)	FP/N = FP/(FP + TN) = 1 - TNR	Tasa de falsos positivos (false positive rate)
False-discovery rate (FDR)	FP/(FP + TP) = 1 - PPV	Tasa de descubrimiento falso (false discovery rate)
Miss rate (FNR)	FN/(FN + TP) = 1 - TPR	Tasa de omisiones (miss rate - false negative rate)
Referencia		
TP	Verdaderos Positivos (True Positives)	Aciertos
TN	Verdaderos Negativos (True Negatives)	Rechazos correctos
FP	Falsos Positivos (False Positives)	Falsa alarma
FN	Falsos Negativos (False Negatives)	Alarmas omitidas
P	TP + FN	
N	FP + TN	

Tabla 8: Métricas utilizadas para evaluar un clasificador binario [50]

6.3.3 Resultados obtenidos

La *Tabla 9* muestra los valores obtenidos, para las métricas de evaluación de un clasificador binario, en varias de las pruebas realizadas para ajustar los parámetros del detector de intrusos. Para cada conjunto, se muestran los valores obtenidos para las métricas:

- los valores iniciales de los parámetros,
- la prueba con la que se obtuvo la tasa más alta de aciertos (accuracy),
- los valores obtenidos invirtiendo la mejor prueba de cada conjunto de muestras disponible.

Finalmente, luego de analizar los valores resultantes en estas pruebas, se configuró el detector de intrusos utilizando los parámetros con lo que se obtuvo la mejor marca de exactitud (0.76) sobre el conjunto de muestras más grande. Sin embargo, aumentado tan solo en 0.01 el valor del parámetro scale se obtuvo un 0.75 de exactitud y un decremento en el tiempo promedio de procesamiento por imagen. La *Tabla 10*, muestra los valores finales para los parámetros del algoritmo de detección de intrusos a utilizar en el siguiente paso, que consiste en integrarlo al resto del sistema de vigilancia propuesto.

En base a los resultados obtenidos (tasa de aciertos de 0.75), y como se sugirió al describir el software implementado, no es posible filtrar las notificaciones en donde la detección sea negativa, porque existe el riesgo de filtrar un verdadero positivo. En cambio la solución utilizada, consistió en enviar una notificación, semánticamente más fuerte, en caso que la detección de intrusos sea positiva. En la implementación del chatbot de Telegram, se envía una advertencia (1 solo mensaje con la imagen adjunta) cuando la detección es negativa, y una alerta (3 mensajes consecutivos más la imagen adjunta) en caso de una detección positiva.

Las pruebas de ajustes de parámetros se ejecutaron en una Notebook Asus modelo Q325UA-BI7T18 [51], con un procesador Intel Core i7, 16GB de memoria RAM y un disco de estado sólido de 512GB.

Cantidad de Muestras	**Conjunto de muestras 1**			**Conjunto de muestras 2**			
Positivas	25			786			
Negativas	25			530			
Parámetros del detector de intrusos	**Valores Iniciales**	**Mejor Resultado**	**Mejor Resultado Conjunto 2**	**Valores Iniciales**	**Mejor Resultado**	**Mejor Balance (Ex/Vel)**	**Mejor Resultado Conjunto 1**
image_max_width_size	400	400	400	400	400	400	400
winStride	(4,4)	(4,4)	(4,4)	(4,4)	(4,4)	(4,4)	(4,4)
scale	1.05	1.1	1.04	1.05	1.04	1.05	1.1
padding	(8,8)	(4,4)	(4,4)	(8,8)	(4,4)	(4,4)	(4,4)
non_maxima_suppression_thresh	0.65	0.65	0.65	0.65	0.65	0.65	0.65
Métricas y tiempo de ejecución							
Tiempo total	4.49s	2.54s	5.21s	117.28s	139.27s	116.82s	68.07s
Tiempo promedio por imagen	0.09s	0.05s	0.10s	0.09s	0.11s	0.09s	0.05s
Accuracy ACC	0.74	0.86	0.76	0.73	0.76	0.75	0.73
Balanced accuracy BACC	0.74	0.88	0.76	0.73	0.76	0.75	0.78
TP	19	19	19	547	559	532	464
TN	18	24	19	394	420	431	483
FP	6	6	6	239	227	254	322
FN	7	1	6	109	83	72	20
P	26	20	25	656	642	604	484
N	24	30	25	633	647	685	805
Sensitivity (TPR)	0.73	0.95	0.76	0.83	0.87	0.88	0.96
Specificity (SPC)	0.75	0.80	0.76	0.62	0.65	0.63	0.60
Precision (PPV)	0.76	0.76	0.76	0.70	0.71	0.68	0.59
Negative predictive value (NPV)	0.72	0.96	0.76	0.78	0.83	0.86	0.96
Fall out (FPR)	0.25	0.20	0.24	0.38	0.35	0.37	0.40
False discovery rate (FDR)	0.24	0.24	0.24	0.30	0.29	0.32	0.41
Miss rate (FNR)	0.27	0.05	0.24	0.17	0.13	0.12	0.04

Tabla 9: Valores de todas las métricas obtenidos en el procedimiento de ajuste de parámetros sobre los conjuntos de muestras utilizados

image_max_width_size	400
win stride	(4,4)
scale	1.05
padding	(4,4)
non_maxima_suppression_thresh	0.65

Tabla 10: Valores de los parámetros del detector de intrusos con los que se obtuvo el mejor balance entre tasa de aciertos y velocidad de detección, sobre los conjunto de muestras disponibles.

6.3.4 Evaluación en tiempo real

La siguiente prueba consistió en constatar estos resultados activando el módulo de detección de intrusos y evaluarlo en conjunto con el resto del sistema. A fin de poder medir el desempeño del algoritmo de detección en tiempo real, se agregó al chatbot de Telegram la posibilidad de indicar manualmente el resultado de cada alerta recibida. Así, cada vez que el chatbot recibe una imagen se muestra una botonera con las siguientes opciones (*Imagen 41*):

- **PS-TP (muestra positiva - verdadero positivo):** el algoritmo detectó intrusos en la imagen, y efectivamente se observan personas en la imagen (*Imagen 42*.a).
- **PS-FN (muestra positiva - falso negativo):** el algoritmo no detectó intrusos en la imagen, pero visualmente si son observables personas en la imagen (*Imagen 42*.b).
- **NS-FP (muestra negativa - falso positivo):** en la imagen no se observan intrusos, pero sin embargo el algoritmo detecta en forma positiva (*Imagen 42*.c).
- **NS-TN (muestra negativa - verdadero negativo):** el algoritmo no detecta intrusos y efectivamente no se observan personas en la imagen (*Imagen 42*.d).
- **DISCARD (descartar la muestra)**: esta opción si agregó para ciertos casos donde la muestra no puede ser evaluada y no se desea que afecte a las métricas que se están recolectando. Esta situación se presenta por ejemplo cuando el sensor de la cámara no logra adaptarse a un cambio repentino de luz y la imagen registrada está completamente quemada (*Imagen 42*.e), imposibilitando incluso visualmente distinguir personas u objetos.

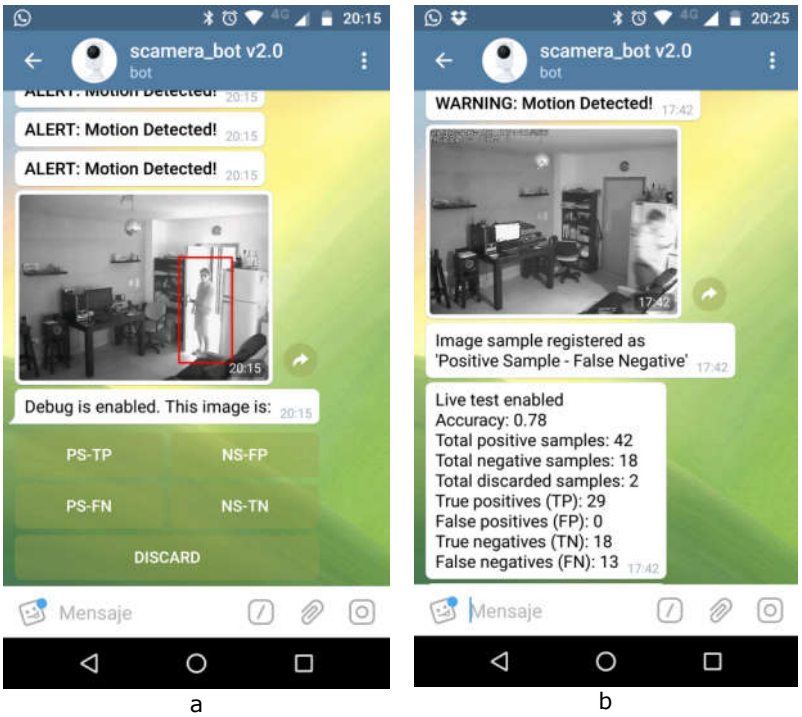


Imagen 41: Evaluación en tiempo real de las alertas de movimiento recibido, a través del chatbot de Telegram a) Botonera mostrada debajo de cada imagen para especificar manualmente el resultado b) Luego de seleccionar una opción se muestran las estadísticas del test (Fuente: propia)

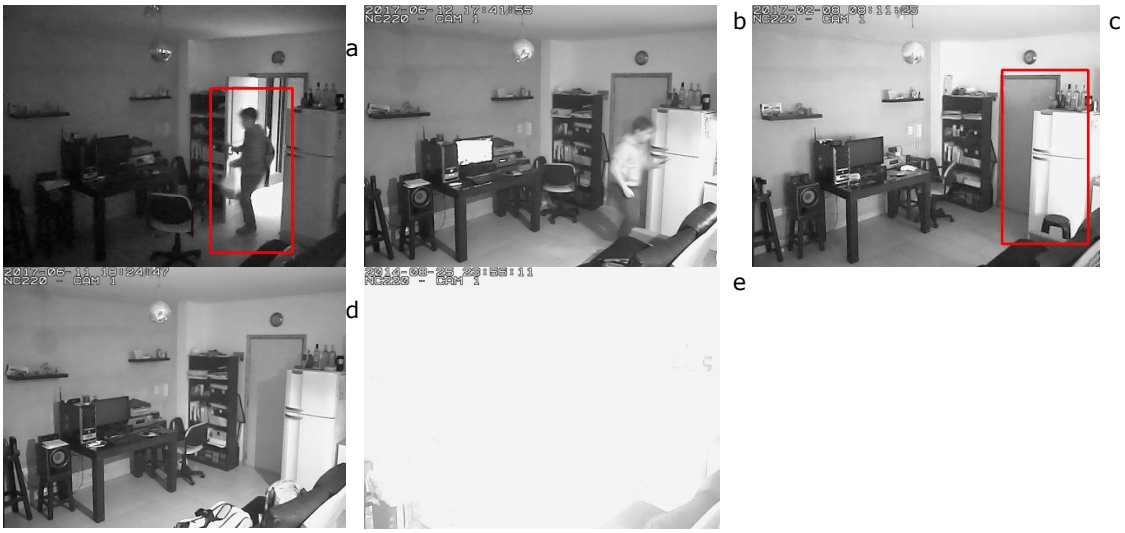


Imagen 42: a) muestra positiva - verdadero positivo b) muestra positiva - falso positivo c) muestra negativa - verdadero positivo d) muestra negativa - verdadero negativo e) muestra descartada (Fuente: propia)

Estos resultados, se guardan en la misma estructura definida para las pruebas descritas anteriormente. La botonera, para evaluar en tiempo real cada imagen recibida en el chatbot de Telegram, puede ser desactivada desde la aplicación DJANGO admin. En la *Tabla 11* se resumen los resultados obtenidos de las pruebas realizadas mediante esta funcionalidad.

Se puede observar que la exactitud ("accuracy") obtenida para el detector de intrusos, fue muy similar a la obtenida en las pruebas de ajustes de parámetros con las muestras preexistentes. Esto

tiene sentido ya que las muestras obtenidas en ambas pruebas tienen las mismas características. Es decir, la cámara tomó muestras bajo las mismas condiciones de entorno y estando exactamente en la misma posición. Con lo cual se puede afirmar, que para este caso de uso en particular, el detector de intrusos tiene una exactitud que ronda el valor de 0.75. Otra observación importante, es que el detector tiene más falencias al detectar verdaderos positivos, que indicar una alarma verdadera en una muestra negativa. Es decir, en un conjunto reducido de casos, detecta intrusos en imágenes que no registran presencia de personas. Este comportamiento se ve reflejado en el valor de la tasa de detección de falsos positivos ("False Discovery Rate"), que varió entre 0 y 0.8 y una tasa de omisiones ("Miss rase") que osciló entre 0.32 y 0.37, en las tres pruebas realizadas. Se podrían seguir haciendo pruebas, moviendo gradualmente los diferentes parámetros de configuración, la sensibilidad de la cámara, y el área de detección del campo visual de la cámara; y llegar a obtener algun punto mas de exactitud en los resultados, pero todo indicaría que difícilmente se pueda superar una exactitud de alrededor de 0.80, en base a los resultados obtenidos en las pruebas de ajustes de parámetros y las pruebas en tiempo real, utilizando el detector de intrusos implementado con LSVM y HOG, y con las características que tienen las imágenes utilizadas en este caso de uso en particular.

	Prueba 1	**Prueba 2**	**Prueba 3**
Configuración de la cámara	Detección en área de la puerta Modo: nocturno	Detección en área de la puerta Modo: automatico	Detección en toda el área de visión Modo: automatico
Parámetros del detector de intrusos			
image_max_width_size		400	
winStride		(4,4)	
scale		1.05	
padding		(4,4)	
non_maxima_suppression_thresh		0.65	
Cantidad de muestras recolectadas			
Positivas	97	44	56
Negativas	64	18	41
Descartadas	0	2	16
Métricas y tiempo de ejecución			
Accuracy ACC	0.75	0.77	0.76
Balanced accuracy BACC	0.78	0.84	0.78
TP	61	30	36
TN	60	18	38
FP	4	0	3
FN	36	14	20
P	97	44	56
N	64	18	41
Sensitivity (TPR)	0.63	0.68	0.64
Specificity (SPC)	0.94	1.00	0.93
Precision (PPV)	0.94	1.00	0.92
Negative predictive value (NPV)	0.62	0.56	0.66
Fall out (FPR)	0.06	0.00	0.07
False discovery rate (FDR)	0.06	0.00	0.08
Miss rate (FNR)	0.37	0.32	0.36

Tabla 11: Valores obtenidos para las métricas de evaluación del detector de intrusos obtenidos en las pruebas realizadas en tiempo real, mediante la botonera de evaluación de imágenes provista por el chatbot de Telegram

Capítulo 7 - Conclusiones y Trabajos a Futuro

7.1 Conclusiones

En este trabajo se introdujo el tema de la aplicación del procesamiento de imágenes en el área de vigilancia. En particular, se describen algoritmos de detección de movimiento en video y el reconocimiento de personas en imágenes, que en combinación con dispositivos de captura de video permiten tener un sistema de monitoreo automático capaz de detectar potenciales amenazas y emitir las alertas o avisos correspondientes en tales situaciones.

Teniendo este marco teórico como inicio, se evaluó un sistema de cámara de seguridad comercial de bajo costo, TP-LINK NC220, el cual puede ser útil para monitorear hogares y comercios. Se analizaron las características del equipo y se marcaron algunos problemas en la experiencia de interacción con el sistema, en particular:

- El mecanismo para activar y desactivar las alarmas de movimiento,
- Los medios disponibles para recibir las alertas ante un evento de movimiento.

Se propuso la implementación de un sistema de software, que interactúa con el equipo comercial, mejorando por una parte la experiencia de interacción entre el usuario y el equipo y; por otra parte, añadir niveles de alarmas basados en la detección de intrusos utilizando un algoritmo de reconocimiento de personas mediante histogramas de gradientes orientados (HOG) en conjunto con una máquina vectorial de soporte lineal (LSVM).

Se realizaron test de usabilidad mediante la técnica SUS, para verificar que la funcionalidad agregada al sistema comercial, mejora la usabilidad e interacciones de los usuarios con el sistema. Los resultados obtenidos de estas pruebas confirmaron que la valoración subjetiva de usabilidad de sistemas mejorado fue mayor (puntuación SUS promedio de 95) que la obtenida por el sistema comercial original (puntuacion SUS promedio de 57.08) .

En las pruebas realizadas, al evaluar el sistema de detección de movimiento embebido en la cámara se obtuvo un 64.40% de alarmas positivas y un 35.60% de falsas alarmas. Por otra parte, el algoritmo de detección de personas mediante histogramas de gradientes orientados tiene un rendimiento con una tasa de aciertos de alrededor del 75%. Con lo cual se pudo mejorar la semántica de notificación de las alertas en el 75% de las alarmas reportadas por la cámara. Sin embargo, en ocasiones puede omitir o filtrar alarmas verdaderas dado que se reportó una tasa de omisiones o "miss rate" de alrededor del 25%.

Por este último motivo, equipos comerciales como los utilizados en este trabajo, permiten monitorear un área de interés y aumentar el nivel de control que se tiene sobre el área, no pueden garantizar un alto nivel de seguridad. Si el objetivo es monitorear un área de una vivienda o comercio y recibir alertas ante algún evento inusual, el sistema cumple con los requerimientos de mejorar el sistema comercial. Por otra parte, si la seguridad es una necesidad crítica, este sistema por sí solo no puede garantizar un alto nivel de seguridad y será necesario acudir a otros sistemas que brinden un nivel mayor, o complementar el sistema con otros dispositivos como sensores de movimiento y alarmas o incluso mediante la supervisión humana de las cámaras de video.

El sistema de seguridad, descrito en este trabajo, depende de varios factores externos que no

son controlables por el usuario ya que el sistema depende de:

- el suministro de energía eléctrica,
- una conexión a internet y,
- en el caso de desplegar la aplicación implementada en un servicio externo, de la disponibilidad de dicho servicio por parte del proveedor.

Es decir, que existen tres recursos, no controlables, para que el sistema de seguridad pueda operar. Durante los meses en los cuales se realizaron las pruebas, explicadas en el Capítulo 6, se produjeron algunos cortes del suministro eléctrico y también del servicio de internet. En ambos casos, el sistema de vigilancia quedó fuera de servicio.

Otro tema, a tener en cuenta en este sistema, es la seguridad en el acceso a la cámara y la red donde está conectada. Si la red WIFI a la que se conecta el equipo no está debidamente configurada, dejaría al equipo expuesto a accesos no autorizados a la aplicación de configuración, con el posible riesgo de que el sistema sea anulado completamente en forma intencional. Lo mismo aplica para la aplicación implementada, la cual se debería desplegar en un servidor que cumpla con las medidas necesarias de seguridad para evitar accesos no autorizados.

7.2 Trabajos a futuro

En lo que respecta a la parte práctica realizada en este trabajo, son varios los puntos en los que se pueden realizar mejoras y nuevos desarrollos. Entre ellos se pueden citar:

- Realizar los cambios necesarios para que los componentes del software implementado puedan ejecutarse en forma distribuida, e interactúen entre sí mediante una API Rest. Cada aplicación del proyecto DJANGO debería exponer una API que permita la interacción con la lógica y modelos que define. De esta manera, las tres aplicaciones definidas en el proyecto: notifications, ftp, images podrían ejecutarse en nodos diferentes y comunicarse a través de estas APIs. Esto, también requeriría cambiar el motor de base de datos por uno que pueda ejecutarse en forma distribuida, como por ejemplo MySQL [52]. Por último, como parte de este refactoring podría cambiarse la implementación del chatbot de Telegram, para que las nuevas notificaciones recibidas en el servidor de Telegram, sean redirigidas a un webhook publicado por la aplicación, en vez de utilizar la técnica de long polling descrita en la sección 5.2.3. Todas estas APIs, pueden implementarse utilizando el framework django REST framework [53], el cual es un framework que se monta sobre DJANGO y brinda el marco para la definición e implementación de APIs REST. Estos cambios contribuyen también a que el software implementado soporte un mayor nivel de carga de trabajo y sea más escalable.
- Agregar implementaciones de otros algoritmos de detección de personas en imágenes~~,~~ como redes neuronales o métodos basados en machine learning o que utilicen algún otro tipo de descriptores. Lo cual permitirá comparar los resultados obtenidos con cada algoritmo, y elegir el más adecuado para cada caso de uso.
- Desplegar el sistema en forma local, mediante el uso de una computadora o minicomputadora, eliminando así la necesidad de contar y depender de la disponibilidad de un servicio externo para el despliegue y funcionamiento de la aplicación.
- Realizar pruebas, con mayor carga en el sistema. En otra palabras, monitorear varias áreas, lo cual implica conectar más cámaras y tener más usuarios suscriptos al chatbot de Telegram que reciban las notificaciones de alertas.

- Probar otros modelos de cámaras similares en prestaciones a la utilizada en este trabajo, ya se del mismo fabricante u otro diferente. Esto permitirá evaluar y comparar el desempeño de cada equipo testeado.
- Permitir el registro y activación de nuevos usuarios, que reciben las notificaciones, desde el chatbot de Telegram.
- Reducir el tiempo fuera de servicio del sistema en caso de un corte del suministro eléctrico, conectado todos los componentes del sistema (cámara, y router wifi) a una unidad UPS para que puedan funcionar por un lapso de tiempo adicional en caso de un corte de energía.

Bibliografía

[1] R. S. Shirbhate, N. D. Mishra, y R. Pande, "Video Surveillance System Using Motion Detection: A Survey", Advanced Networking and Applications, 2012.

[2] TP-LINK, "TP-LINK: NC220", Cámara Cloud Diurna/Nocturna, 300Mbps Wi-Fi NC220. [En línea]. Disponible en: http://www.tp-link.es/products/details/cat-19_NC220.html. [Accedido: 13 de Julio, 2017].

[3] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction", en Proceedings of the 17th International Conference on Pattern Recognition, Cambridge, UK, 26 de agosto, 2004.

[4] Z. Zivkovic y van der H. F., "Efficient adaptive density estimation per image pixel for the task of background subtraction", Pattern Recognition Letters, vol. 27, nº 7, pp. 773–780, may 2006.

[5] P. Kaewtrakulpong y R. Bowden, "An Improved Adaptive Background Mixture Model for Realtime time Tracking with Shadow Detection", en Proceedings 2nd European Workshop on Advanced Video Based Surveillance Systems, 2001.

[6] N. Singla, "Motion Detection Based on Frame Difference Method", International Journal of Information & Computation Technology, 2014.

[7] N. Dalal y B. Triggs, "Histograms of oriented gradients for human detection", en IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA, 2005.

[8] A. Rosebrock, "Pedestrian Detection OpenCV", 9 de Noviembre, 2015. [En línea]. Disponible en: http://www.pyimagesearch.com/2015/11/09/pedestrian-detection-opencv. [Accedido: 13 de Julio, 2017].

[9] C. Dahms, "OpenCV Tutorial 8: Pedestrian Detection using Histogram of Oriented Gradients". [En línea]. Disponible en: https://www.youtube.com/watch?v=AKLEuAtFDXQ. [Accedido: 13 de Julio, 2017].

[10] OpenCV, "OpenCV", Open Source Computer Vision. [En línea]. Disponible en: http://opencv.org. [Accedido: 13 de Julio, 2017].

[11] Telegram, "Telegram", Telegram - a new era of messaging. [En línea]. Disponible en: https://telegram.org. [Accedido: 13 de Julio, 2017].

[12] Telegram, "Telegram", Telegram Bot API. [En línea]. Disponible en: https://core.telegram.org/bots/api. [Accedido: 13 de Julio, 2017].

[13] A. Sobral y A. Vacavant, "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos", Computer Vision and Image Understanding, nº 122, pp. 4–21, 2014.

[14] J. Dou, Q. Qin, y Z. Tu, "Background subtraction based on circulant matrix", J. VLSI Signal Process. Syst. Signal Image Video Technol., vol. 11, nº 3, pp. 1–8, 2017.

[15] S. Brutzer, B. Hferlin, y G. Heidemann, "Evaluation of background subtraction techniques for video surveillance", en IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 1937–1944.

[16] R. Cucchiara, C. Grana, M. Piccardi, y A. Prati, "Detecting moving objects, ghosts, and shadows in video streams", IEEE Trans. Pattern Anal. Mach. Intell., vol. 25, nº 10, pp. 1337–1342, 29 de Septiembre, 2003.

[17] S. S. Sengar y S. Mukhopadhyay, "A novel method for moving object detection based on block based frame differencing", en In 3rd International Conference on Recent Advances in Information Technology, 2016, pp. 462–472.

[18] M. Fei, J. Li, y H. Liu, "Visual tracking based on improved foreground detection and perceptual

hashing", Neurocomputing, vol. 152, nº C, pp. 413–428, Marzo, 2015.

[19] S. S. Sengar y S. Mukhopadhyay, "Moving object area detection using normalized self adaptive optical flow", Optik - International Journal for Light and Electron Optics, vol. 127, nº 16, pp. 6258–6267, Agosto, 2016.

[20] M. A. Mahraz, J. Riffi, y H. Tairi, "High accuracy optical flow estimation based on PDE decomposition", J. VLSI Signal Process. Syst. Signal Image Video Technol., pp. 1409–1418, 2015.

[21] A. A. Shafie, H. Fadhlan, y M. H. Ali, "Motion Detection Techniques Using Optical Flow", World Academy of Science, Engineering and Technology, vol. 56, pp. 559–561, 2009.

[22] OpenCV, "Background Subtraction", OpenCV-Python Tutorials. [En línea]. Disponible en: http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_bg_subtraction/py_bg_subtraction.html. [Accedido: 29 de Julio, 2017].

[23] A. B. Godbehere, A. Matsukawa, y K. Goldberg, "Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation", presentado en American Control Conference (ACC), Montreal, QC, Canada, 2012.

[24] A. Rosebrock, "Basic motion detection and tracking with Python and OpenCV". [En línea]. Disponible en: http://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv. [Accedido: 13 de Julio, 2017].

[25] Viola, Jones, y Snow, "Detecting pedestrians using patterns of motion and appearance", en Proceedings Ninth IEEE International Conference on Computer Vision, Nice, France, 2005.

[26] O. Hamdoun y F. Moutarde, "Keypoints-based background model and foreground pedestrian extraction for future smart cameras", en 3rd ACM/IEEE International Conference on Distributed Smart Cameras, Como, Italy, Agosto de 2009.

[27] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", International Journal of Computer Vision, vol. 60, pp. 91–110, 2004.

[28] H. Bay, T. Tuytelaars, y L. V. Gool, "SURF:Speeded Up Robust Features", en Proceedings of the 9th European Conference on Computer Vision, 2006, pp. 404–417.

[29] A. Angelova, A. Krizhevsky, V. Vanhoucke, A. Ogale, y D. Ferguson, "Real-Time Pedestrian Detection With Deep Network Cascades", en Proceedings of BMVC, 2015.

[30] MathWorks, "Matlab", The Language of Technical Computing. [En línea]. Disponible en: https://www.mathworks.com/products/matlab.html. [Accedido: 13 de Julio, 2017].

[31] A. Rosebrock, "Histogram of Oriented Gradients and Object Detection". [En línea]. Disponible en: http://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection.

[32] A. Rosebrock, "Non-Maximum Suppression for Object Detection in Python", 17 de Noviembre, 2014. [En línea]. Disponible en:

http://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/. [Accedido: 18 de Julio, 2017].

[33] A. Rosebrock, "HOG detectMultiScale parameters explained", 16 de Noviembre, 2015. [En línea]. Disponible en:

http://www.pyimagesearch.com/2015/11/16/hog-detectmultiscale-parameters-explained/. [Accedido: 13 de Julio, 2017].

[34] "TP-LINK Cloud". [En línea]. Disponible en: https://www.tplinkcloud.com/. [Accedido: 13 de Julio, 2017].

[35] "TP-Link NC220", Some stuff about TP-Link NC220 IP Camera. [En línea]. Disponible en:

https://github.com/reald/nc220. [Accedido: 13 de Julio, 2017].

[36] C. S. Castañeda, "GitHub Repo", GitHub. [En línea]. Disponible en:

https://github.com/seba3c. [Accedido: 13 de Julio, 2017].

[37] Django, "Django", The Web framework for perfectionists with deadlines. [En línea]. Disponible en: https://www.djangoproject.com. [Accedido: 13 de Julio, 2017].

[38] "Python". [En línea]. Disponible en: https://www.python.org/. [Accedido: 13 de Julio, 2017].

[39] G. Rodla, "pyftpdlib", Python FTP server library. [En línea]. Disponible en:

https://github.com/giampaolo/pyftpdlib. [Accedido: 13 de Julio, 2017].

[40] "Whatsapp". [En línea]. Disponible en: https://www.whatsapp.com/. [Accedido: 13 de Julio, 2017].

[41] "Facebook Messenger". [En línea]. Disponible en: https://es-es.messenger.com/. [Accedido: 13 de Julio, 2017].

[42] Telegram, "Bots: An introduction for developers". [En línea]. Disponible en: https://core.telegram.org/bots. [Accedido: 17 de julio, 2017].

[43] Telegram, "python-telegram-bot". [En línea]. Disponible en:

https://pypi.python.org/pypi/python-telegram-bot. [Accedido: 13 de Julio, 2017].

[44] RASPBERRY PI FOUNDATION, "RASPBERRY PI", TEACH, LEARN AND MAKE WITH RASPBERRY PI. [En línea]. Disponible en: https://www.raspberrypi.org/. [Accedido: 13 de Julio, 2017].

[45] Digital Ocean, "Digital Ocean", Digital Ocean-Cloud Computing, Designed For Developers. [En línea]. Disponible en: https://www.digitalocean.com. [Accedido: 13 de Julio, 2017].

[46] "SQLite". [En línea]. Disponible en: https://www.sqlite.org/.

[47] ISO, "Ergonomic Requirements for Office Work with Visual Display Terminals", ISO, 9241-11, Marzo, 1998.

[48] J. Brooke, "SUS-A quick and dirty usability scale", Usability evaluation in industry, vol. 189, nº 194, pp. 4–7, 11 de Junio, 1996.

[49] TP-LINK, "TP-LINK User Guide - NC220 Day/Night Cloud Camera, 300Mbps Wi-Fi".

[50] T. Fawcett, "Introduction to ROC analysis", Pattern Recognition Letters, vol. 27, nº 8, pp. 861–874, jun. 2006.

[51] ASUS, "ASUS 2-in-1 Q325UA", Specifications. [En línea]. Disponible en:

https://www.asus.com/us/Laptops/Q325UA/specifications/. [Accedido: 23 de Julio, 2017].

[52] Oracle, "MySQL". [En línea]. Disponible en: https://www.mysql.com/. [Accedido: 23 de Julio, 2017].

[53] "django REST framework". [En línea]. Disponible en:

http://www.django-rest-framework.org/. [Accedido: 23 de Julio, 2017].