DEPARTAMENTO DE ELECTRICA Y ELECTRÓNICA

CARRERA DE ELECTRÓNICA Y TELECOMUNICACIONES

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

TEMA: "DESARROLLO DE UN SISTEMA EMBEBIDO PARA DETECTAR EN TIEMPO REAL LA PRESENCIA DE PERSONAS EN AMBIENTES CONTROLADOS".

AUTOR: SOTO ESPINOZA, JESSICA JACQUELINE

DIRECTOR: ING. SILVA TAPIA, RODRIGO

SANGOLQUÍ

2018

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE ELECTRÓNICA Y TELECOMUNICACIONES

CERTIFICACIÓN

Certifico que el trabajo de titulación, **"DESARROLLO DE UN SISTEMA EMBEBIDO PARA DETECTAR EN TIEMPO REAL LA PRESENCIA DE PERSONAS EN AMBIENTES CONTROLADOS"** fue realizado por la señorita Soto Espinoza, Jessica Jacqueline con CI: 172178621-6 e ID: L00009715, el mismo que ha sido revisado en su totalidad, la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, razón por la cual me permito acreditarlo y autorizar para que lo sustente públicamente.

Sangolquí, 05 de Marzo del 2018

Firma:

ING. RODRIGO SILVA TAPIA

C.C. 0602199523

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

AUTORÍA DE RESPONSABILIDAD

Yo, **SOTO ESPINOZA, JESSICA JACQUELINE**, con cédula de identidad N° 172178621-6, declaro que el contenido, ideas y criterios del trabajo de titulación: **"DESARROLLO DE UN SISTEMA EMBEBIDO PARA DETECTAR EN TIEMPO REAL LA PRESENCIA DE PERSONAS EN AMBIENTES CONTROLADOS"** es de mi autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, respetado los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Consecuentemente el contenido de la investigación mencionada es veraz.

Sangolquí, 05 de Marzo del 2018

JESSICA JACQUELINE SOTO ESPINOZA

C.C: 172178621-6

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

AUTORIZACIÓN

Yo, **SOTO ESPINOZA, JESSICA JACQUELINE** autorizo a la Universidad de las Fuerzas Armadas ESPE el trabajo de titulación: **"DESARROLLO DE UN SISTEMA EMBEBIDO PARA DETECTAR EN TIEMPO REAL LA PRESENCIA DE PERSONAS EN AMBIENTES CONTROLADOS"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 05 de Marzo del 2018

JESSICA JACQUELINE SOTO ESPINOZA

C.C: 172178621-6

DEDICATORIA

Quiero dedicar este trabajo de investigación a una de las personas más importantes de mi vida mi madre María, la cual me apoyo cuando muchos me dieron la espalda, gracias porque nunca me dejaste rendir, siempre me alentaste a tu manera pero lo hiciste sin tu apoyo no estaría culminando esta etapa de mi vida, gracias por ser un ejemplo a seguir y por ser una persona que nunca se deja vencer sin importar los problemas que tenga.

A mi padre Carlos y mis hermanos Pedro y Carla, que han estado apoyándome y nunca me dejaron sola, su amor y sus consejos han sido muy importantes en cada paso y decisión que he tomado en mi vida, gracias a los tres por siempre cuidarme y no dejarme caer.

A mi abuela Carmen por ser siempre un apoyo en mi vida, gracias por estar conmigo en toda mi vida brindándome tu amor incondicional y tus ánimos, tú fuiste una de las pocas personas que nunca dudo de mis decisiones sin importar lo locas que sean, tu presencia y apoyo fue fundamental para culminar mi carrera profesional por todo lo que has hecho por mi te dedico este trabajo.

Jessica Jacqueline Soto Espinoza

AGRADECIMIENTO

Agradezco a mi madre María por siempre luchar conmigo, por apoyarme en todo aunque a veces no te parecía correctas mis decisiones, gracias por tus palabras de aliento cuando tenía ganas de rendirme, nunca tendré la manera de agradecerte todo lo que has hecho por mí en toda mi carrera, por el momento gracias madre por apoyarme sin ti este paso no sería posible, te amo.

Gracias a mi hermana Carla, sabes que a veces tú pareces más madura que yo por eso te agradezco por cuidarme en estos años por siempre apoyarme, quererme y aconsejarme.

A mi padre Carlos que su apoyo fue fundamental para culminar mi carrera, por tu apoyo te agradezco, y en general le agradezco a toda mi familia por estar conmigo siempre y cuidarme cuanto estaba muy ocupada, ustedes han sido uno de los pilares fundamentales en toda mi carrera universitaria, gracias por toda su ayuda.

A Patricio y Alicia gracias por su apoyo y su cariño, su presencia siempre fue de mucha ayuda en las largas noches de estudio, gracias por estar ahí brindándome una palabra de aliento y ayudarme en lo que podían.

A Cristina que siempre me apoyo en todo y se convirtió en una gran amiga a largo de todos estos años de carrera, gracias por luchar conmigo en todos los semestres que estuvimos juntas, por ser un apoyo y una gran compañera en todos los obstáculos que se presentaron en este camino.

Jessica Jacqueline Soto Espinoza

LISTADO DE ACRÓNIMOS

LBP:	Local Binary Patterns
HOG:	Histogramas de Gradientes Orientados
SVM:	Support Vector Machines
VGA:	Video Graphics Array
LCD:	Liquid Cristal Display
CCTV:	Circuito Cerrado de Televisión
IP:	Internet Protocol
ACC:	Accuracy
PPV:	Positive Predictive Value
TPR:	True Positive Rate
TNR:	True Negative Rate
Wi-Fi:	Wireless Fidelit
INRIA:	Institut National de Recherche en Informatique et en Automatique
BMP:	Windows Bitmap
JPG:	Joint Photographers Experts Group
PNG:	Portable Network Graphics

GPIO:	General Purpose Input/Output
HDMI:	High-Definition Multimedia Interface
AdaBoost:	Adaptive Boosting
FPGA:	Field Programmable Gate Array

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

La visión por computadora es un área de conocimiento de mucho interés tanto en el ámbito académico como en el comercial. En los últimos años se ha desarrollado significativamente gracias al uso de múltiples herramientas computacionales que han permitido la implementación de técnicas de procesamiento de imágenes y algoritmos de aprendizaje automático para realizar la detección, identificación y seguimiento de objetos observados dentro de imágenes fijas ó de video. El propósito del presente trabajo es diseñar e implementar un prototipo de un sistema inteligente de bajo costo, para detectar personas en un video capturado en tiempo real de un ambiente con iluminación controlada. Para el efecto, en un sistema embebido Raspberry Pi 3 configurado con herramientas OpenCV-Python se implementaron tres algoritmos detectores: LBP-AdaBoost, Haar AdaBoost y HOG-SVM. La captura de video se realizó a través de una cámara de video VGA. El prototipo implementado tiene como objetivo alertar al operador por medio de un tono audible de la presencia de una persona en el video capturado. Para evaluar la precisión de los detectores se efectuaron pruebas considerando entornos controlados con distintos niveles de iluminación y distancias entre la cámara y los objetos a ser detectados. Los resultados muestran que el detector Haar-AdaBoost es el más preciso, mientras que el detector LBP-AdaBoost es el que tiene menor consumo computacional.

PALABRAS CLAVE:

- **PROCESAMIENTO DE IMÁGENES DIGITALES**
- **ALGORITMOS DETECTORES**
- **HERRAMIENTAS COMPUTACIONALES**
- **SISTEMAS EMBEBIDOS**

ABSTRACT

Computer vision domain is interesting for both the academic and commercial fields. In recent years, this domain has developed significantly thanks to the use of multiple computational tools. These tools have allowed implementing image processing techniques and automatic learning algorithms to perform the detection, identification and monitoring of objects observed in fixed or video images. This work aims at designing and implementing a low-cost intelligent-system prototype in order to detect people in a real-time captured video of a controlled lighting environment. For this purpose, an embedded Raspberry Pi 3 system configured with OpenCV-Python tools was used. Three detection algorithms were implemented: LBP-AdaBoost, Haar AdaBoost and HOG-SVM. The video was captured through a VGA video camera. The implemented prototype alerts the operator of the presence of a person in the captured video by means of an audible tone. To evaluate the accuracy of the proposed detectors, a set of tests were carried out considering controlled environments with different levels of illumination and distances between the camera and the detected objects. Results show that the Haar-AdaBoost detector has more precision while the LBP-AdaBoost has the lowest computational consumption.

KEY WORDS:

- **DIGITAL IMAGE PROCESSING**
- **ALGORITHMS DETECTORS**
- **COMPUTATIONAL TOOLS**
- **EMBEDDED SYSTEMS**

CAPÍTULO I

INTRODUCCIÓN

En la actualidad el uso de sistemas de visión artificial, denominada también visión por computadora ha adquirido una gran importancia en múltiples ámbitos como la detección de objetos. Este crecimiento se debe principalmente a dos factores: el abaratamiento de las herramientas de captura de vídeo y el avance que ha experimentado el procesamiento digital de imágenes en los últimos años. Un ámbito en el que es ampliamente utilizado son los sistemas de videovigilancia. Un sistema de videovigilancia tradicional permite controlar mediante un video, un área de forma manual ya que un guardia de seguridad verifica las cintas de grabación por lo tanto no se obtiene una respuesta autónoma.

Los sistemas de videovigilancia son muy costosos ya que están constituidos por varios elementos complejos como son cámaras de seguridad, un videograbador que es el encargado de grabar los videos, un disco duro para el almacenamiento de las grabaciones, una fuente de alimentación, cableado y en caso de que se necesite de un sistema en tiempo real una pantalla LCD que permita el control de los videos. Además de personal especializado como lo son los guardias de seguridad. En consecuencia, estos equipos requieren de una gran cantidad de dinero para su instalación y uso.

El uso masivo de sistemas de videovigilancia en la sociedad actual y el costo del personal asociado, hace necesaria la implementación de sistemas que realicen la detección automática de objetos, seguimiento, reconocimiento de acciones y demás tecnologías aplicadas al análisis y comprensión de la imagen digital. El análisis de imágenes (Pajares Saenz, 2008), (Rodriguez

Morales & Sossa Azuela, 2011) es fundamental para poder extraer información sobre un determinado ambiente. Mediante el uso de visión artificial, el procesamiento de una imagen es más sencillo ya que se logra diferenciar de manera eficiente los objetos que se desean analizar, ya sea una persona, o un determinado objeto de forma autónoma.

El uso de librerías de visión artificial actualmente facilita la creación de algoritmos que permiten controlar procesos que requieren la presencia de personas para realizar una determinada acción. Al implementar un algoritmo de detección de personas en conjunto con un sistema embebido, se puede realizar un sistema de videovigilancia de bajo costo.

El uso de detectores de objetos ha sido fundamental para el análisis de una imagen. Con el pasar del tiempo estos algoritmos han ido perfeccionándose para tener un menor consumo computacional y una mayor eficiencia. Existen métodos holísticos que utilizan algoritmos para la detección de objetos tales como el algoritmo clasificador AdaBoost, que permite entrenar una serie de clasificadores débiles de manera iterativa de modo que cada nuevo clasificador se enfoque en los datos que fueron erróneamente clasificados por su predecesor. De esta manera, el algoritmo se adapta y logra mejores resultados (Puebla, s.f.). Esta clase de algoritmos es usado en la detección de rostros (Luna Gallegos, Palacios Hernandéz, & Marín Hernández, 2014) (Guevara, Echeverry, & Urueña, 2008), ya que al ir analizando la cantidad de falsos positivos obtiene un buen resultado en sus detecciones. En la publicación realizada por Rama y Tarrés (Un nuevo método para la detección de caras basado en Integrales Difusas, 2007) se realiza la detección de rostros mediante un clasificador no lineal.

Otro método holístico para la detección de objetos es el SMV (Support Vector Machine), el cual usa información de los descriptores (Waring & Liu, 2005). Por medio de un descriptor se

extraen las características de un conjunto de imágenes de entrenamiento (muestras), se puede etiquetar las clases y entrenar una SVM para construir un modelo. El modelo predecirá la clase de una nueva muestra de imágenes obtenidas al separar los espacios mediante un hiperplano denominado vector de soporte (Hernández Calandria, Cañas, & Diaz, 2007).

En la literatura se puede encontrar varios trabajos enfocados a la detección de rostros de personas (Felzenszwalb, McAllester, & Ramanan, 2008). Una de los principales aplicaciones es en el uso de sistemas de seguridad (Sanchez Matilla, 2014). Gallardo Calvopiña & Sánchez Quevedo (2016) realizaron la construcción de un sistema biométrico que compara una fotografía con una base de datos. Además se desarrolló un proyecto en tiempo real para la detección de peatones (Manlises, Martinez, & Belenzo, 2015), en el cual al detectar la presencia de una persona y se emite una señal para evitar accidentes de tránsito. Todos estos trabajos facilitan la posibilidad de desarrollar un sistema de detección de personas por medio de un sistema embebido, el cual no necesitará la presencia de una persona para controlar las cámaras de seguridad ya que este sistema será autónomo.

La detección de objetos al ser realizada por medio de visión artificial necesita de una plataforma que permita su funcionamiento. Actualmente se puede realizar este proceso mediante varios lenguajes de programación, siendo Python un lenguaje que con la ayuda de la librería OpenCV, permite desarrollar un programa en conjunto de visión artificial, que permite la detección de objetos. La tarjeta Raspberry Pi 3 cuenta con un sistema operativo que tiene preinstalado Python, y además permite la instalación de la librería OpenCV, lo que facilita la implementación de un sistema de videovigilancia.

1.1. Objetivos del Proyecto

1.1.1. Objetivo General

- Implementar un prototipo de sistema de video en tiempo real para la detección de personas por medio de un algoritmo detector de objetos.

1.1.2. Objetivos Específicos

- Analizar la arquitectura y configuración de un sistema embebido tipo Raspberry Pi.
- Analizar y seleccionar un algoritmo clasificador de imágenes para la detección de personas.
- Implementar un detector de personas en tiempo real utilizando el sistema embebido.
- Realizar pruebas del prototipo en ambientes controlados.

CAPÍTULO II

MARCO TEÓRICO

2.1. Introducción

La detección de objetos mediante visión artificial busca identificar ciertas características que permitan diferenciar el objeto de interés de todo el entorno en el que se encuentre. Las características que ayudan a la distinción del objeto podrían ser la forma, el color o el tamaño, sin embargo estos rasgos no suelen ser suficientes. Por ejemplo al momento de detectar un objeto, si se analiza el color varios de los elementos del entorno analizado podrían compartir la misma característica ocasionando problemas en la detección. Al analizar la forma de un objeto en sistemas de video podrían existir problemas si se utilizan métodos clásicos como la sustracción de fondo, ya que si cualquier objeto está en movimiento sería considerado como un elemento de interés. En el caso de determinar a una persona como parte de la detección el uso del método de extracción de bordes podría ser poco eficientes, ya que cualquier objeto en movimiento podría ser considerado como una persona. Con el pasar del tiempo se han desarrollado herramientas más complejas que permiten solucionar los inconvenientes que ya se mencionaron.

Por lo tanto el uso de descriptores para realizar la extracción de características de una imagen, y el entrenamiento de clasificadores es primordial para desarrollar un detector eficiente. En este capítulo se explicarán las técnicas que permiten desarrollar de manera exitosa la detección de personas. Además se incursionará en las herramientas que ayudan al funcionamiento del detector de objetos.

Los descriptores de objetos son los encargados de extraer las características de una imagen, para así poder distinguir ciertos rasgos de un objeto seleccionado. Por medio de un descriptor de características, se puede describir la forma de un objeto. Los descriptores ayudan al entrenamiento de un clasificador, para que se pueda realizar la detección de un objeto.

2.2. Descriptores Haar, LBP, HOG

2.2.1. Características Haar

Las características Haar son utilizadas en la detección de objetos principalmente en rostros y peatones. Actualmente el algoritmo desarrollado por Paul Viola y Michael Jones es uno de los métodos que más se utilizan para la detección de rostros (Viola, Jones, & Snow, Detecting Pedestrians Using Patterns of Motion and Appearance, 2005). Los detectores Haar buscan identificar las peculiaridades de un objeto, basándose en los niveles de intensidad que poseen los pixeles en una determinada imagen. Estos rasgos serán extraídos de la imagen analizada, utilizando métodos que ofrecen las características Haar. Este tipo de detectores permite analizar la estructura de objetos aunque estos no sean uniformes.

Los descriptores Haar son definidos como una ventana de pixeles, que posee un tamaño y una orientación variable. Esta ventana estará dividida en regiones rectangulares como se observa en la Figura 1. Las regiones de las características Haar pueden ser de dos tipos: serán positivas las regiones de color blanco, y negativas las regiones de color negro. Esta ventana de pixeles va analizando la imagen, y las características que pertenecen al objeto deseado serán consideras como positivas y las que no contienen al objeto serán negativas. La diferencia de la suma de las regiones positivas y la suma de las regiones negativas, es lo que determinará el valor de la característica Haar. Mediante la resta de los dos valores se obtendrá un mapa, que indicará la posición de la

característica Haar. Esto permite determinar un valor umbral que sirve para realizar la clasificación de objetos.

En un descriptor Haar pueden considerarse varios parámetros, que facilitan la extracción de características estos pueden ser: el tamaño, la orientación, y la distribución de regiones positivas y negativas de la imagen. Estas características son específicas de los rasgos del objeto a detectar, y dependen de la distribución de intensidad de los pixeles analizado, ya que al desarrollar la selección de características del descriptor Haar, se necesita que este se asemeje a la estructura de los objetos que serán detectados. Por lo tanto las características Haar pueden ser utilizadas para la detección de contornos, líneas, rostros, etc.

Existen tres tipos de características Haar propuestas por Viola-Jones (Viola & Jones, Robust Real-Time Face Detection, 2004), las cuales serán clasificadas según el número de regiones rectangulares que las componen, y son de: dos, tres y cuatro regiones. Estas regiones se utilizan en distintas áreas de la imagen. Las características Haar de dos regiones, serán utilizadas para la detección de bordes, las cuales pueden tener orientaciones de 0° y 90° tal como muestra la Figura 1a. Las características Haar de tres regiones son aquellas que pueden ser utilizadas para detectar líneas verticales de 0° y 90°, y se encuentran representadas en la Figura 1b. Las características Haar de cuatro regiones son las que pueden utilizar para detectar líneas diagonales y se ilustra en la Figura 1c.

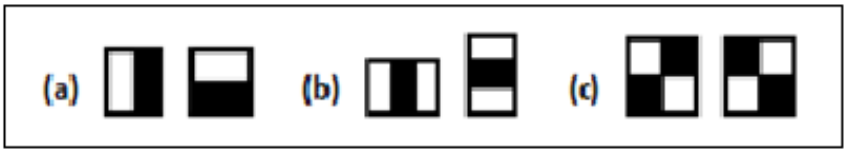

Figura 1. Características Haar (a) Dos regiones de 0° y 90°
(b) Tres regiones 0° y 90° (c) cuatro regiones

Fuente: (Mendieta, 2013)

Los valores que pertenecerán a las características Haar serán determinadas de acuerdo al tipo de región utilizada. Si se decide analizar la imagen con dos regiones, se obtiene el valor mediante realizando la suma de todos los pixeles positivos, y restando la suma de todos los pixeles negativos, que se encuentran en sus regiones rectangulares correspondientes. En el caso de tres regiones se obtiene el valor de las características realizando la suma de los rectángulos exteriores que pertenecen a los pixeles positivos, y a dicho valor se le resta el rectángulo intermedio que pertenece a los pixeles negativos. Para el caso de una característica de cuatro regiones, se procede a calcular la diferencia entre las diagonales de los rectángulos que pertenecen a los pixeles positivos y negativos. Como se puede comprobar las características Haar son sencillas al momento de realizar su cálculo, pero el consumo computacional que necesita es excesivo, ya que se analizará cada pixel de una determinada imagen. Este problema ocasiona que se busque una manera de solucionarlo, y de allí el concepto de imagen integral.

2.2.1.1. Imagen Integral

La imagen integral es una transformación de la imagen original, que como resultado, dará una nueva imagen del mismo tamaño, con la diferencia que los pixeles de la imagen integral serán la suma de todos los pixeles de la imagen original. El uso de regiones rectangulares puede no ser muy efectivo al momento de realizar la extracción de características, como ya se mencionó en la

sección anterior. Por lo que se introduce el concepto de imagen integral (Viola & Jones, Robust Real-Time Face Detection, 2004), la cual permite que las regiones rectangulares puedan analizar las características en tiempo real. Esto facilita el proceso de adquisición de rasgos del objeto a analizar, disminuyendo el consumo computacional. La transformación a una imagen integral consiste en crear una matriz, que será conformada por la suma de todos los pixeles que conforman la imagen y quedará encima y en la izquierda de la imagen original.

Con el uso de una imagen integral se busca realizar el procesamiento de los descriptores de imagen. Mediante el uso de características Haar, se extraen los rasgos de una forma más sencilla, y ya no se trabaja con la intensidad en los pixeles de la imagen. Estas imágenes trabajarán en modo escala de grises y por lo tanto se pueden presentar problemas, no se tendrá información de color es la imagen solo cuenta con un canal y es de color negro. Este único canal tendrá una gama de 256 tonos de gris (0-255). En donde m será la posición del pixel en x, n es la posición del pixel en y, la intensidad es el valor de la gama en tono de gris que tendrá el pixel. En la Figura 2 se puede observar las partes que conforman el valor de una imagen integral. El área pintada de azul serán los valores que se sumarán para el cálculo de es pixel y el área de color anaranjado será la intensidad la línea en la que se encuentre el pixel. Se puede utilizar la siguiente ecuación para interpretar la suma de los pixeles en una imagen integral.

$$\sum_{i=1}^{m}\sum_{j=1}^{n} intensidad_{ij} \tag{1}$$

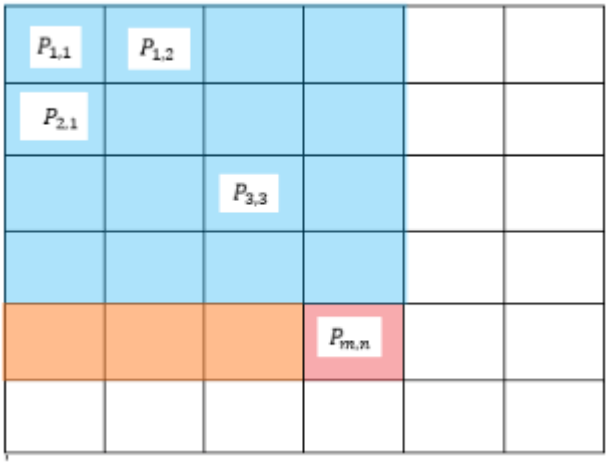

Figura 2. Imagen Integral

Como ejemplo se utilizará a la Figura 3 y se realizan los cálculos para obtener el valor que tendrá cada pixel de la imagen integral. Para el cálculo de los valores de la imagen integral se hará un barrido de la imagen original los valores de sus pixeles se encuentran en la matriz I como se indica a continuación:

1. Se realizará un cálculo de la intensidad de la fila en donde se encuentre el pixel y sus valores se ubicarán en la matriz s1. Cada valor será acumulativo sumando la intensidad de sus antecesores. Como ejemplo en la segunda fila se acumulan los valores de la imagen original en este caso será el valor 5 ubicado en la posición (1, 1) sumado el valor de 0 ubicado en la posición (1, 2), estos valores se ubicarán en la matriz s1 en la posición (1, 2), y así sucesivamente con cada uno de los valores de los pixeles.
2. Para el cálculo de la primera fila de la imagen integral se suman los valores de la fila anterior de la matriz II, con los valores de la matriz s1, ya que no existen valores anteriores se procede a copiar los mismos de s1.

3. Para calcular los valores de intensidad de la matriz s2, se acumularán los valores de la fila en la que se encuentre y pixel. Por ejemplo para el valor de intensidad s1 de la tercera columna, se sumarán los valores de la matriz I y estos son 10+10+5 y es 25.
4. Para el valor de la fila en la imagen integral se suman los valores de s2 más el valor de la posición del pixel analizado en la primera final de la matriz de II. Como por ejemplo para la cuarta columna de la matriz II el valor se obtiene sumando 35 de la posición (1, 4) de la matriz II, más 25 de la posición (1, 4) de la matriz s1. Se obtiene el valor de 60 que será ubicado en la posición (2, 4) de la matriz II.
5. Se realiza este procedimiento hasta que se obtenga el valor de toda la imagen integral.

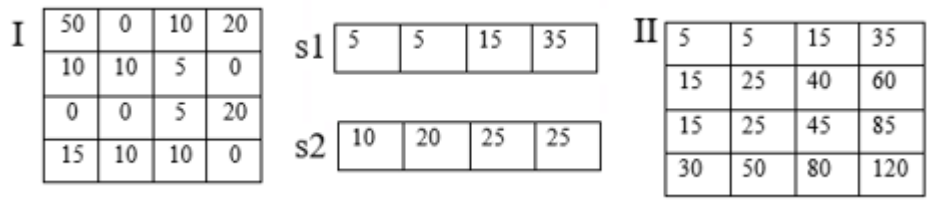

Figura 3. Ejemplo de imagen integral

2.2.2. Características LBP (*Local Binary Patterns*)

Los descriptores LBP son operadores de textura, los cuales fueron desarrollados inicialmente por D. He y L. Wan (Texture Unit, Texture Spectrum, and Texture Analysis, 1990). Este tipo de técnica evalúa el contraste de los pixeles en espacios de regiones pequeñas, para obtener un espectro de textura en una determinada imagen. Posteriormente T. Ojala (Performance Evaluation of Texture Measures, 1994) propone que la textura de las imágenes son analizadas mediante un número binario, identificando de manera más sencilla el tipo de textura. Por medio de esta investigación se desarrollaron las características LBP (*Local Binary Patterns),* y su funcionamiento se maneja sobre entornos de pixeles de un tamaño de 3x3. La propuesta compara el nivel de

intensidad de un pixel central, comparándolo con sus pixeles vecinos. En caso de que el valor del pixel central sea mayor o igual a la intensidad del su pixel vecino, se asignará el valor de uno, y cuando es menor se asignará el valor de 0. Finalmente obtiene una matriz binaria de 0 y 1 como se observa en la Figura 4, finalmente se procede a multiplicar la posición de cada número binario por su valor decimal, y se suman obteniendo de ese modo su característica LBP del pixel central. En la Figura 4 se puede observar el procedimiento para la obtención de esta característica. Las características LBP son invariantes cuando están en escala de grises. Tiene como objetivo asignar un código binario que describirá, el patrón de textura local por medio de un umbral determinado por el valor de intensidad de gris del valor central.

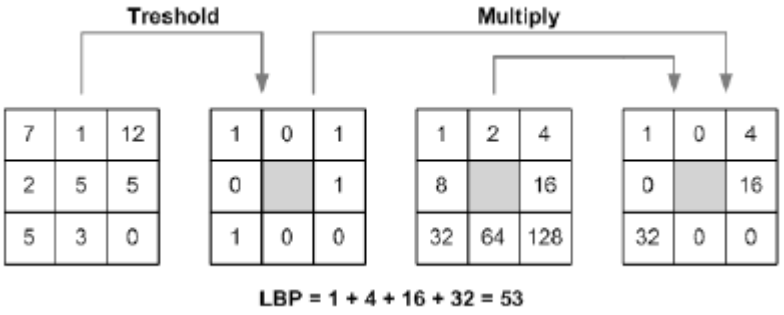

Figura 4. Funcionamiento del operador LBP
Fuente: (Mendieta, 2013)

2.2.3. Características HOG

Los *Histograms Of Oriented Gradients* (HOG) o denominadas Histogramas de Gradientes Orientados, es un métodos extractor de características que tiene como fundamento, la orientación de los gradientes de la imagen. Este tipo de descriptor cambia de acuerdo a la intensidad de la imagen, además que la dirección de sus gradientes depende del contraste de las imágenes. La idea principal de este método es que se puedan extraer características de un objeto tales como: apariencia, forma, o la intensidad de la iluminación. Los cambios en la dirección en una imagen

puede ser descrita por los vectores gradiente. En la Figura 5 se ilustra la forma en que es analizada la imagen. Primeramente se divide la imagen en bloques y a partir de ahí serán divididos en celdas de los cuales se calculará su histograma y su gradiente. Se transforma a la imagen a escala de grises.

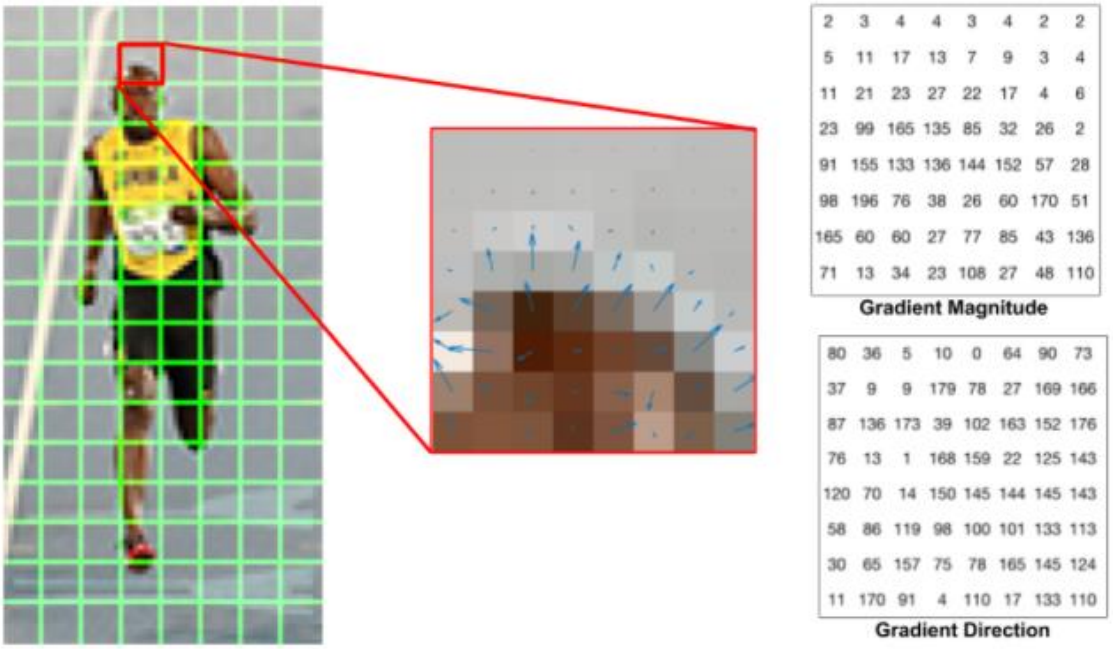

Figura 5. Descriptor HOG
Fuente: (Mallick, 2016)

2.3. Clasificadores

Clasificar un objeto consiste en asignarlo a una de las dos clases disponibles, la primera clase será la que contiene al objeto deseado, y la segunda la que no contiene al objeto. Los objetos se pueden definir por una serie de características. Para poder clasificar objetos es necesario definir las fronteras entre las diferentes clases. Normalmente estas fronteras se calculan mediante un proceso de entrenamiento. Este entrenamiento utiliza las características extraídas de una imagen, y por

medio de estas se puede definir las clases. Las fronteras pueden definir los límites que ayudan al clasificador a determinar la decisión, de los objetos que serán clasificados de manera exitosa. Además que permite establecer las reglas de decisión que se considerarán durante el entrenamiento (Garrido Satué, 2013). El uso de clasificadores es fundamental para poder separar de manera eficiente los objetos deseados. El proceso de entrenamiento del clasificador, es primordial ya que separa las características de un determinado objeto, permitiendo desarrollar de ese modo un clasificador eficiente.

El uso de descriptores en conjunto con los clasificadores, permiten desarrollar un algoritmo detector de objetos. En la Sección 2.2 se explica cómo se desarrolla la extracción de características de un objeto. Por lo tanto se necesita seleccionar aquellos rasgos que permitan identificar al objeto deseado del resto de la imagen, realizando el entrenamiento de un clasificador.

Un clasificador puede ser desarrollado de una manera sencilla, determinando un valor umbral, el cual determinará los objetos a ser detectados de manera exitosa. Los objetos que se detecten de forma correcta se encontraran sobre el valor umbral, mientras que si se encuentra abajo los objetos no serán detectados. Existen clasificadores binarios como son: AdaBoost y SVM. En este trabajo se utilizarán clasificadores para ser utilizados en una plataforma digital, y así se puedan determinar, cuáles objetos serán detectados como personas y cuáles no. Por lo tanto en esta sección se describirán los clasificadores AdaBoost y SVM, que son los que se adaptan de manera exitosa a las características deseadas en este proyecto.

2.3.1. Clasificadores débiles

Los clasificadores débiles son aquellos que están definidos por una función h(x), la cual depende de una característica *f* utilizada en una región x, además de un valor umbral *t* y una paridad

p la que representa el sentido de la desigualdad siendo 0 ó 1. De ese modo se busca que la salida del clasificador sea un número binario, que indicará si las características se encuentran sobre o bajo del valor umbral. Determinando si se detecta de manera eficiente un objeto.

$$h(x) = \begin{cases} 1, & p * f > p * t \\ 0, & si\ no \end{cases} \quad (2)$$

Un problema de este clasificador es encontrar el valor exacto del umbral *t*, con el cual se pueda clasificar de forma correcta, la mayor cantidad de características de un objeto. Existen varios algoritmos que ayudan a encontrar el valor del umbral óptimo, basándose en una serie de muestras de un objeto a ser detectado. Estos clasificadores pueden ser definidos como un algoritmo de decisión. Ya que dependerá directamente del valor de las características g(x), como se observa en la Figura 6 esta función determinarán si la detección seguirá un camino u otro.

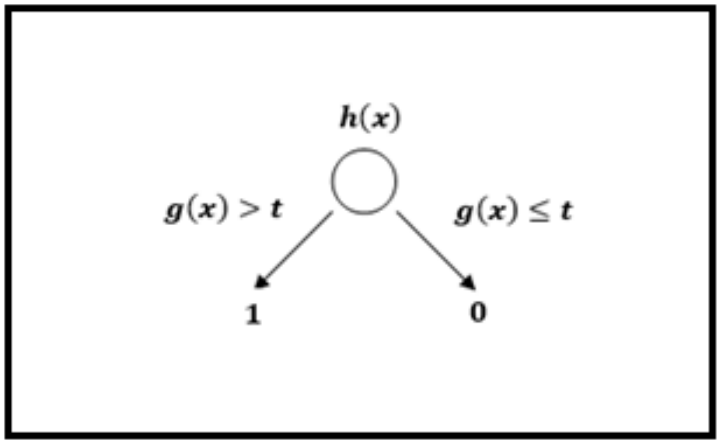

Figura 6. Esquema de un clasificador de decisión

2.3.2. Clasificadores Fuertes

Estos clasificadores ayudan a mejorar el procesamiento de los clasificadores débiles. En la Figura 7 se puede observar que un clasificador fuerte es un conjunto de clasificadores débiles. Esta

característica permite mejorar, la selección de objetos detectados de manera eficiente. Para desarrollar un clasificador fuerte se combinan varios clasificadores débiles, el valor de este tipo de clasificadores se puede definir de la siguiente manera:

$$H(x) = \begin{cases} 1, & p * F(x) > p * t \\ 0, & si\ no \end{cases} \quad (3)$$

$$F(x) = \sum_{i=1}^{N} w_i h_i(x) \quad (4)$$

En donde $h_i(x)$ son los clasificadores débiles, w_i son los pesos de cada clasificador, un peso es la importancia que se le da a cada clasificador, al inicio todos tienen el mismo valor, N es el número de clasificadores débiles involucrados, t será el valor final del umbral y p es el sentido de la desigualdad. Debido a que están involucrados varios clasificadores débiles, se deberá realizar un ajuste en los valores de los pesos y umbrales para poder determinar un solo valor que será el del clasificador fuerte.

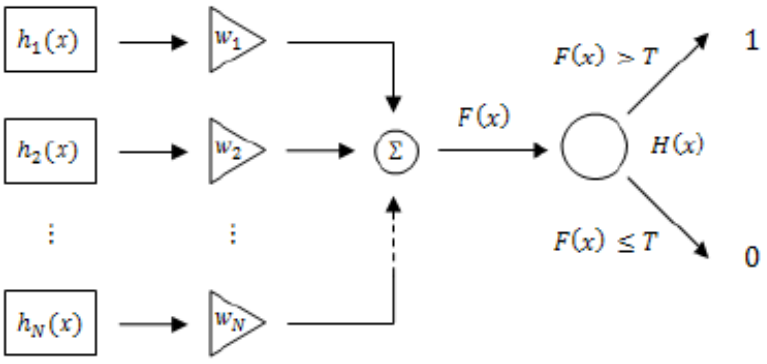

Figura 7. Esquema de un clasificador de fuerte
Fuente: (Mendieta, 2013)

2.3.3. Clasificadores en Cascada

2.3.3.1. Experimentos Iniciales

En la sección anterior se observó que las características Haar, son utilizadas para extraer información que permitan la detección de un objeto. Al realizar un algoritmo detector de objetos, si se considera una sola característica no será suficiente en la distinción del objeto en el entorno. Considerando que en este trabajo la detección de personas debe ser precisa, y en tiempo real la combinación de varias características Haar, ayudaría a unir varios rasgos de un mismo objeto proporcionando una detección más exacta.

Los clasificadores en cascada que fueron planteados por Paul Viola y Michael Jones (Robust Real-Time Face Detection, 2004), buscan desarrollar un clasificador fuerte en base a un conjunto de clasificadores débiles. Cada clasificador débil está compuesto por un descriptor Haar. Las características que cada objeto necesite para ser detectado, serán determinadas por el algoritmo clasificador, mediante valores que han sido asignadas por un grupo de muestras positivas y negativas. Las muestras positivas son imágenes que indican el objeto de interés, y las muestras negativas son imágenes en las que no está presente el objeto.

Cada una de las características Haar ayudarán al entrenamiento del clasificador en cascada y se seguirán los siguientes pasos:

1. Seleccionará la imagen que va a ser analizada, de un conjunto de imágenes.
2. Se realizará una evaluación con una ventana que realizará, un barrido en toda la imagen buscando el objeto de interés.
3. Si la ventana no detecta ningún objeto se modificará el tamaño de la ventana hasta que se obtenga al objeto de interés.

En la Figura 8 se observa el funcionamiento de un clasificador en cascada con un solo clasificador fuerte. Los investigadores Viola-Jones al inicio de su investigación explicaron el funcionamiento de este tipo de clasificador, demostrando no ser muy eficiente debido a que este algoritmo se enfoca en analizar en mayor profundidad las imágenes negativas, ya que poseen una mayor cantidad de espacio en la imagen. Las ventanas que poseen las características Haar al evaluar la imagen en distintas escalas consumían demasiados recursos computacionales en otros procesos.

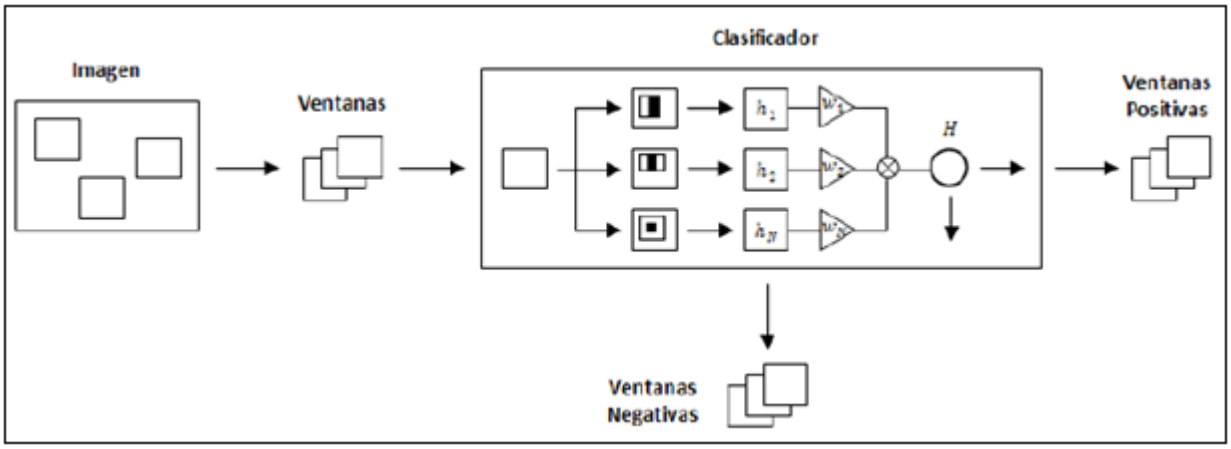

Figura 8. Esquema de un clasificador en cascada con un solo clasificador fuerte.
Fuente: (Mendieta, 2013)

2.3.3.2. Descripción de los clasificadores

Después de varias pruebas Viola-Jones (Robust Real-Time Face Detection, 2004) desarrollaron los clasificadores en cascada. Los que permiten combinar varios clasificadores fuertes en solo algoritmo, con estructura de árbol jerárquico o en cascada. Los clasificadores en cascada analizan por etapas a una imagen, descartando inicialmente los lugares en donde no está el objeto de interés. Un clasificador en cascada estará conformado por varios clasificadores fuertes formando varias etapas, es decir, cada clasificador estará ubicado en secuencia del otro. De ese modo, cada clasificador fuerte es más complejo que su antecesor. Las etapas iniciales serán las

encargadas de analizar la imagen y descartar el fondo. Las últimas etapas al ser más complejas son las encargadas de analizar los objetos deseados. Al utilizar clasificadores en cascada se disminuye el consumo computacional y el tiempo de análisis.

El funcionamiento de los clasificadores será igual que en sus experimentos iniciales. Con la diferencia que en cada etapa del clasificador se va descartado imágenes negativas, de modo que en la siguiente etapa ya no serán consideradas. Al final del algoritmo una imagen positiva será aquella que pasó por todas las etapas de este algoritmo. En la Figura 9 se indica el funcionamiento de un clasificador en cascada.

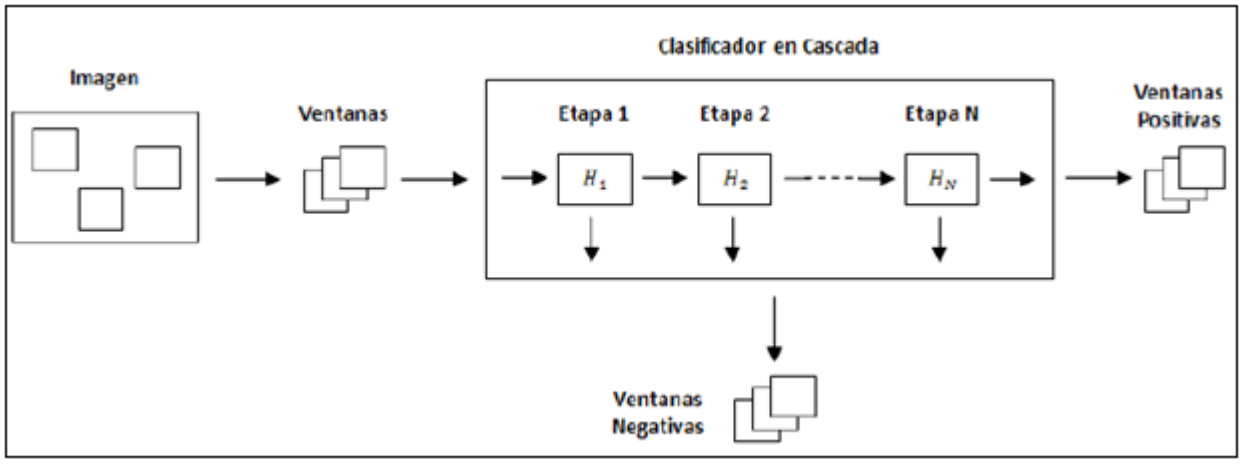

Figura 9. Esquema de un clasificador en cascada
Fuente: (Mendieta, 2013)

2.3.4. Clasificadores SVM (*Support Vector Machines*)

Las Support Vector Machines (SVM) o denominadas Máquinas de Vectores de Soporte, son clasificadores que utilizan los descriptores de características HOG para realizar el entrenamiento de una SVM. Este clasificador podrá reconocer patrones y realizar una detección binaria. Se considerarán dos clases las positivas y las negativas que serán ubicadas en dos planos como se indica en la Figura 10, el primer plano será el que se encuentre bajo la línea divisoria y serán

considerados como valores positivos, mientras que las muestras que se encuentren sobre la línea serán los valores negativos. Todos los vectores de entrada serán considerados en un espacio n-dimensional. Después se construirá un hiperplano que dividirá en dos regiones las muestras. En la Figura 9 se puede observar que un hiperplano es la distancia de separación que está pintada de color amarillo. Los vectores de soporte son los que determinan que la distancia entre los planos que contienen a las muestras positivas y negativas sea máxima. Mientras mayor sea la distancia de separación mejor será la detección ya que no existirá una mala ubicación de las muestras.

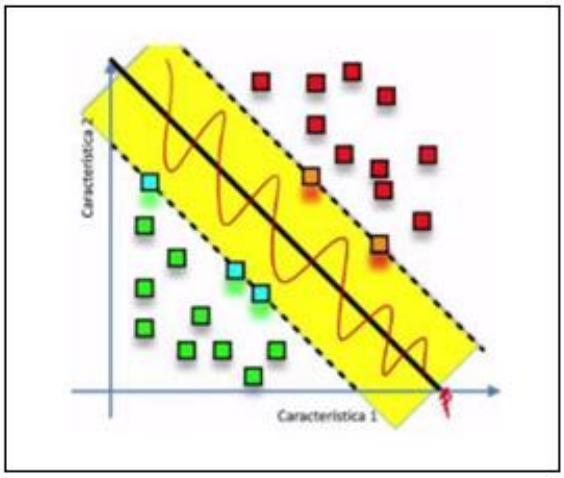

Figura 10. Hiperplano en SVM
(Waring & Liu, 2005)

2.4. Entrenamiento Clasificador en Cascada

Para que un clasificador en cascada pueda detectar un objeto específico, debe pasar por un entrenamiento que le permitirá distinguir las características del objeto deseado. En base a un conjunto de datos se seleccionarán los rasgos de los objetos. En el caso de este trabajo de investigación los objetos a ser detectados serán personas, y por lo tanto se realizarán los ajustes necesarios en los parámetros del algoritmo para que pueda realizar una buena detección.

2.4.1. *Boosting*

El algoritmo de entrenamiento *Boosting* fue propuesto por Schapire (The Strength of Weak Learnability, 1989). Y esta metodología fue el origen del entrenamiento de los clasificadores en cascada. El algoritmo *Boosting* está compuesto por varios clasificadores débiles de aprendizaje computacional. Estos algoritmos son aquellos encargados de un entrenamiento de bajo nivel, y son supervisados por clasificadores fuertes también de aprendizaje computacional los cuales, están un nivel más arriba (Freud & Schapire, 1996). Una de las características del entrenamiento *Boosting* es que los clasificadores débiles son creados de manera iterativa, de modo que el clasificador predecesor es capaz de corregir los errores de su antecesor teniendo una mejor clasificación. En la Figura 11 se puede observar la estructura de funcionamiento de un entrenamiento *Boosting,* el cual está constituido por 3 etapas.

2.4.2. Entrenamiento AdaBoost

Una de las aplicaciones del entrenamiento Boosting es el algoritmo AdaBoost, el cual fue desarrollado por Y. Freud y R.E. Schapire (Experiments with a New Boosting Algorithm, 1996). Este entrenamiento fue capaz de unir el método *Boosting* con varios clasificadores débiles. Este tipo de entrenamiento usa un conjunto de imágenes del objeto a detectar, y este grupo será denominado como muestras positivas. También se utiliza un conjunto de imágenes negativas en las cuales no se encuentra el objeto. Este algoritmo es iterativo, es decir, que repetirá varias veces un mismo procedimiento hasta alcanzar una meta deseada. En el caso del presente proyecto el entrenamiento no se detendrá hasta que se detecten personas únicamente.

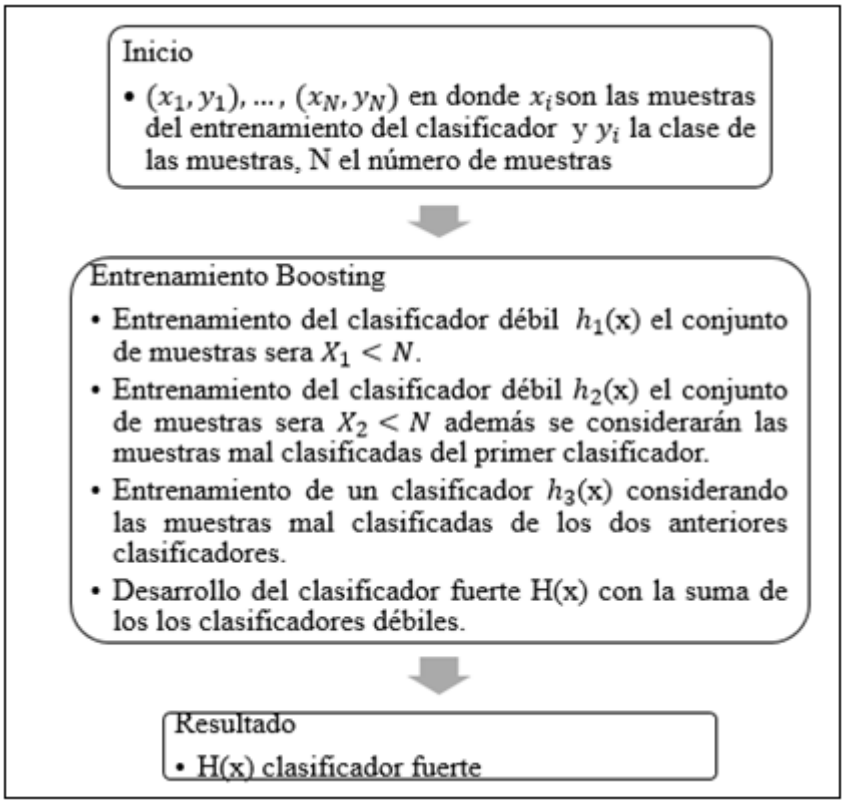

Figura 11. Esquema de un clasificador en cascada

Mediante un clasificador débil de aprendizaje computacional, se desarrollará un clasificador que se encargará de las características del objeto. El algoritmo escogerá el clasificador con menor error de clasificación, para después desarrollar un clasificador fuerte, y se irán actualizando los pesos de las imágenes. En el caso de que sean muestras mal detectadas se les dará un mayor peso mientras que en las muestras bien clasificadas se disminuirá su peso. Al modificar los pesos se está haciendo que el algoritmo sea más robusto, ya que este se enfoca en las muestras que presenten mayor dificultad de ser clasificadas. En la Figura 12 se puede visualizar de mejor manera el funcionamiento del algoritmo AdaBoost.

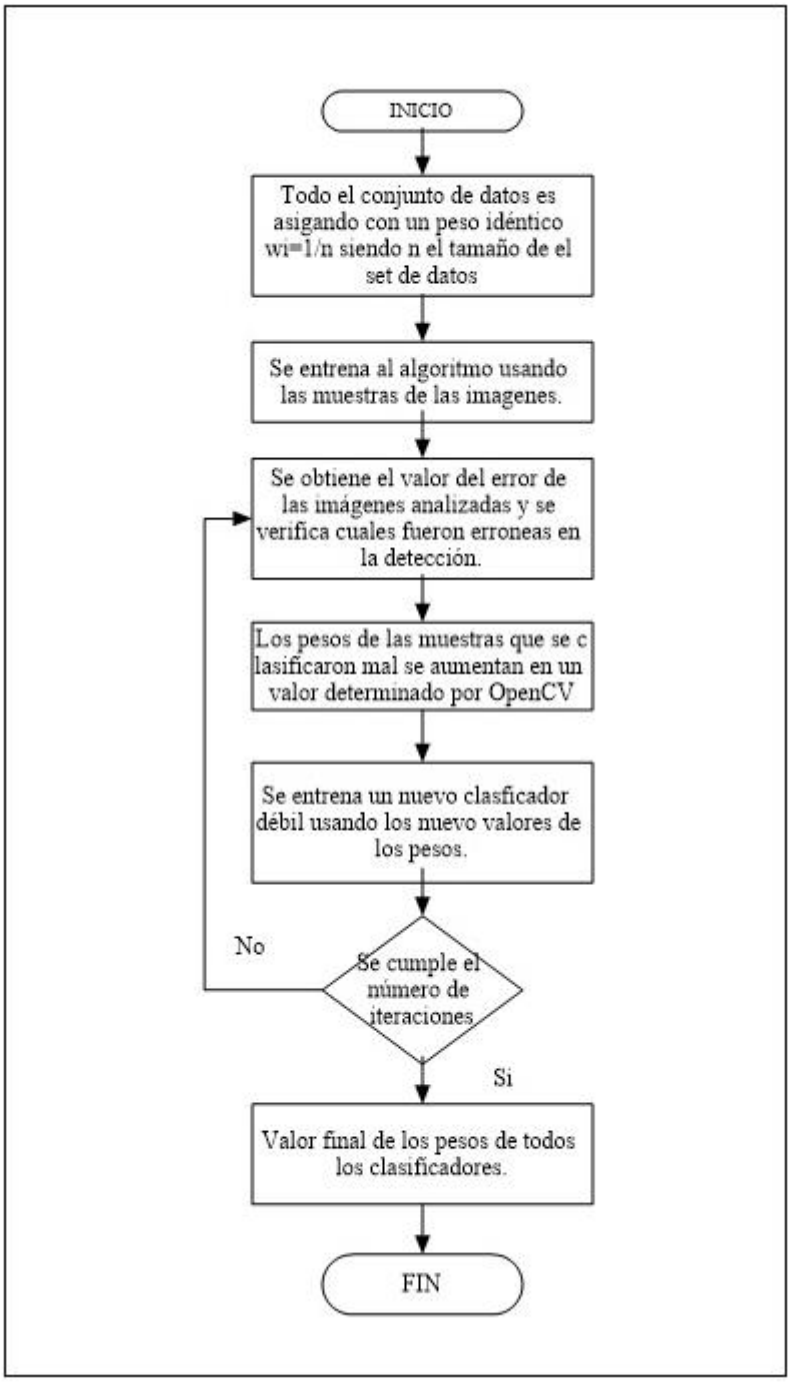

Figura 12. Esquema de entrenamiento AdaBoost

2.4.3. Entrenamiento de Clasificadores en Cascada

Para el entrenamiento de clasificadores en cascada Viola-Jones (Robust Real-Time Face Detection, 2004) se consideran dos parámetros fundamentales: la tasa de verdaderos positivos (T*rue Positive Rate*), que es la razón del conjunto de imágenes bien clasificadas, es decir, que siendo positivas son consideradas como positivas. La tasa de falsos positivos (*False Positive Rate*), que es la razón de imágenes que siendo negativas son consideradas como positivas. Cada una de las etapas será considerada como un clasificador fuerte. El cual será evaluado a partir de la precisión que se desee alcanzar, para eso se definen TPR y FPR que son calculadas de la siguiente forma:

$$TPR = \prod_{i=1}^{N} tpr_i \tag{5}$$

$$FPR = \prod_{i=1}^{N} fpr_i \tag{6}$$

Siendo N el número de etapas que tendrá el detector de objetos, tpr_i será la tasa de verdaderos positivos de cada una de las etapas del algoritmo, fpr_i es la tasa de falsos positivos de cada una de las etapas. Los valores tpr_i y fpr_i serán considerados independientemente en cada clasificador fuerte, de ese modo se puede analizar cada etapa por separado, consiguiendo que al final el algoritmo obtenga un solo valor en base a todas a las etapas.

En la Figura 13 se puede observar el procedimiento para el entrenamiento de un clasificador en cascada. Cada etapa será construida con la información del clasificador que estuvo antes. Con el conjunto de muestras de imágenes se calcularán las tasas tpr y fpr los que se compararán con un valor predeterminado que se desea para todo el algoritmo.

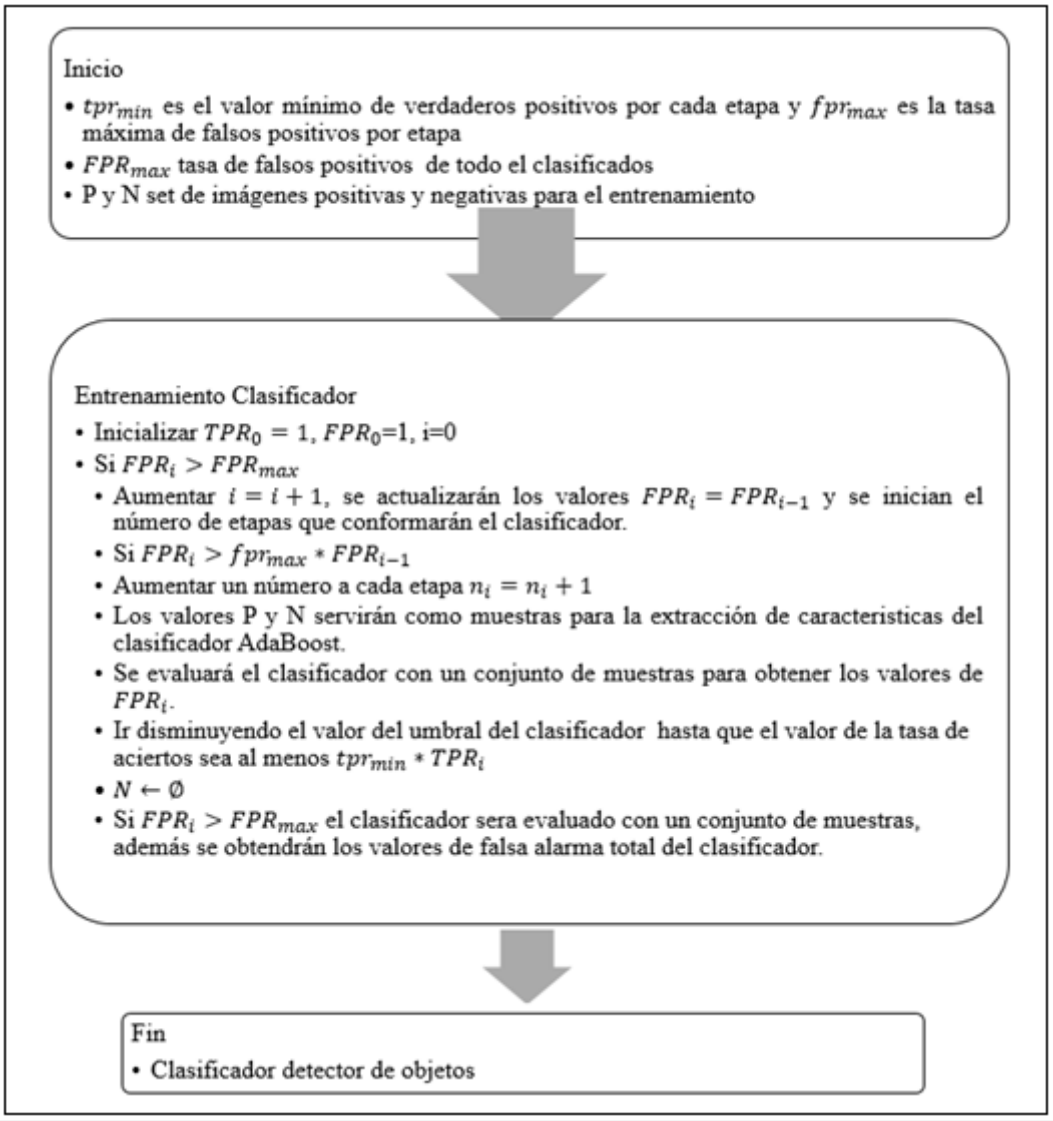

Figura 13. Algoritmo de entrenamiento de un clasificador en cascada.

2.5. Evaluación de Clasificadores en Cascada.

En los anteriores capítulos se explicó como extraer las características de un objeto, para después proceder a conocer lo que es un clasificador y como construir uno. Con las herramientas anteriormente explicadas, sería suficiente para el entrenamiento de un detector de objetos. Pero al

momento de aplicar estos algoritmos en la vida real se necesita realizar una evaluación del rendimiento del clasificador.

El rendimiento de un clasificador puede ser analizado mediante un conjunto de imágenes positivas y negativas. Las cuales pueden determinar los resultados en el momento de la detección de objetos. Al realizar el análisis de los resultados se determinará el desempeño del detector. Existen varios parámetros que ayudan en este proceso como lo son: la tasa de falsos positivos, la tasa de verdaderos negativos, el tipo de características a analizar en el clasificador, que tipo de clasificador será utilizado, el número de etapas, el número de imágenes, etc. En este capítulo se analizarán las herramientas que ayudan en el análisis del comportamiento del clasificador.

2.5.1. Matriz de confusión.

La matriz de confusión (Fawcett, 2003) es una herramienta fundamental, en el campo de visión artificial que ayuda a visualizar, el nivel de confusión que puede presentar el clasificador. Este método ayuda a analizar el rendimiento del detector de objetos sobre un conjunto de imágenes. Se debe considerar que esta matriz trabaja con un algoritmo binario, es decir, solo considera dos clases la clase *"persona"* y la clase *"no persona"*. Además se considera que el clasificador no es exacto en su detección, por lo tanto está sujeto a ciertas confusiones. Por lo tanto se crean dos nuevos grupos que nacen, de la combinación de las dos primeras clases que determinan la equivocación o el acierto en la selección del objeto clasificado.

En este trabajo se considerará en la clasificación de objetos, que cuando una muestra es clasificada como positiva quiere decir que es una persona, mientras que si es negativa quiere decir que no es una persona. En la Figura 14 se visualizará cómo funciona la matriz de confusión de la cual se pueden describir sus conceptos de la siguiente manera:

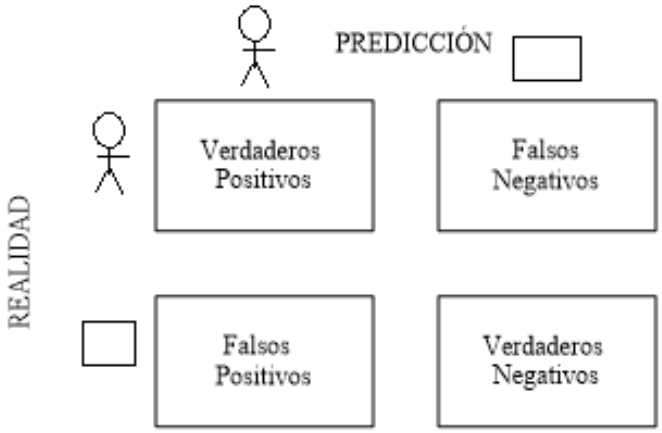

Figura 14. Matriz de Confusión

- Verdadero Positivo (*True Positive*).- Es cuando en la realidad una muestra es una persona, y en la predicción es una persona.
- Falso Positivo (*False Positive*).- Este parámetro es considerado cuando en la realidad no es una persona, y en la predicción es considerada una persona.
- Falso Negativo (*False Negative*).- Es considerado cuando en la realidad es una persona, y en la predicción no es una persona.
- Verdadero Negativo (*True Negative*).- Es cuando en la realidad no es una persona, y en la predicción no es considerado como una persona.

Para poder definir el rendimiento del clasificador se relacionan las clases utilizadas de la siguiente manera:

- Exactitud (*ACC*).- Es el parámetro que indica la capacidad que tiene el clasificador en detectar de forma correcta los objetos.

$$Exactitud = ACC = \frac{Verdades\ Positivos + Verdaderos\ Negativos}{Predicciones\ totales} \quad (7)$$

- Precisión (*PPV*).- Este valor indica las muestras positivas que fueron clasificadas exitosamente, también es conocido como valor predictivo positivo.

$$Precisión = PPV = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Positivos} \quad (8)$$

- Sensibilidad (*TPR*).- Este parámetro indica la capacidad de clasificar correctamente a las muestras positivas, también es conocido como tasa de verdaderos positivos.

$$Sensibilidad = TPR = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos} \quad (9)$$

- Especificidad (*TNR*).- Este valor indica la capacidad que tiene el clasificador de detectar muestras negativas, también es conocido como tasa de verdaderos negativos.

$$Especificidad = TNR = \frac{Verdaderos\ Negativos}{Verdaderos\ Negativos + Falsos\ Positivos} \quad (10)$$

Cada parámetro permite determinar el rendimiento del clasificador, y cada uno de ellos representa un papel importante. Pero que este trabajo tiene como principal objetivo la detección de personas, la precisión será el parámetro que ayudará a determinar el funcionamiento del clasificador. La precisión al ser el valor que determina la cantidad de personas detectadas de forma exitosa, verificando el número de personas cuantas personas que se clasificaron correctamente.

2.6. Herramientas utilizadas para la elaboración de clasificador

2.6.1. Python

Python es un lenguaje de programación de alto nivel independiente multiplataforma, lo que permite que funcione en cualquier sistema operativo que permita instalar un intérprete de este

leguaje. Una de sus principales características es que es de licencia libre. Además que está orientado a objetos, lo que permite trabajar con funciones.

Este lenguaje de programación cuenta con la característica, de que al ser de código abierto ha sido depurado y editado a lo largo de los años. Con cada cambio que se ha hecho se han creado varias bibliotecas, que complementan este lenguaje. En el caso del uso de visión artificial solo se necesita la instalación de la librería OpenCV, la cual permitirá desarrollar un programa que facilite el uso de visión artificial y de ese modo poder verificar el funcionamiento del clasificador.

2.6.2. OpenCV

OpenCV es una librería de código abierto utilizada para trabajos que necesitan de visión artificial lo que facilita el uso de algoritmos detectores de objetos. Fue creada por Intel para desarrollo comercial o de investigación. A lo largo de los años se ha ido perfeccionando en el ámbito de detección de objetos, ya que al ser una librería de libre acceso, que permite desarrollar herramientas que sirven en la detección de objetos.

OpenCV (OpenCV.org, s.f.) es compatible con varios sistemas operativos como: Windows, Linux, Mac, entre otros. Los lenguajes de programación que pueden utilizarlo son: C++, C, Python y Java. El uso de OpenCV en este trabajo es fundamental, ya que puede ser utilizado en aplicaciones que trabajan en tiempo real y permite que se desarrolle un detector de personas utilizando un sistema embebido.

2.7. Sistema Embebido

Un sistema embebido es un sistema que se utiliza para procesamiento de datos, es la combinación del hardware y software de una computadora, los componentes de estos sistemas están

ubicados en una sola placa, lo que reduce su tamaño. Es utilizado para realizar actividades específicas, y su mayor ventaja es que son flexibles, ya que si se desea realizar un cambio en su configuración se pueden modificar las líneas de código del sistema.

Los sistemas embebidos pueden ser programados de dos maneras: por lenguaje ensamblador si se decide programar el microcontrolador que está en la tarjeta o por medio de lenguajes de programación como: Python, C++. El uso de un sistema embebido se orienta a:

- Reducir el tamaño, consumo y costo.
- Aumentar la confiabilidad.
- Mejorar el desempeño.

El uso de sistemas embebidos es muy común en sistemas de tiempo real ya que facilita la interacción con el entorno físico.

2.7.1. Sistema Embebido en tiempo real

Los sistemas embebidos en tiempo real son aquellos en los que el tiempo de ejecución es fundamental para su funcionamiento, estos dependen directamente del tiempo y es necesario determinar un límite o por realizar cálculos para determinar el límite. Estos sistemas interaccionan varias veces con el entorno físico en el que se encuentre, y depende de estímulos que recibe en un plazo de tiempo.

2.8. Tiempo Real

En algunos casos los sistemas no solo son controlados por su resultado lógico, sino que su resultado también depende del tiempo en el que este es ejecutado. Ya que este sistema se está desarrollando en tiempo real, determina que este reaccionará a los eventos que se están produciendo

en ese momento. En algunos casos se puede determinar que el procesamiento realizado se produce como una respuesta instantánea aunque exista un pequeño retraso. Para la transmisión de video se acepta como tiempo real el uso de 12 cuadros por segundo (Tiempo, 2000).

2.9. Ambiente controlado

Un ambiente controlado es aquel lugar en donde se cuidan todos los detalles ambientales, con medios no naturales. En estos ambientes se pueden controlar factores como: luz, temperatura, agua, etc. En el caso de este trabajo de investigación el factor que será analizado es la iluminación de tres escenarios.

2.10. Raspberry

La Raspberry es una mini computadora de bajo costo creada en el 2011. Sus principales características son: un procesador ARM, las dos últimas versiones poseen 1GB de RAM, 4 puertos USB y puerto Ethernet. Su última versión cuenta con Wi-Fi y Bluetooth. Para que esta tarjeta pueda funcionar cuenta con la opción de tener varios sistemas operativos como Linux, siendo Raspbian su sistema operativo oficial, el cual está basado en Debian lo que facilita el uso de Python y OpenCV. La instalación del sistema operativo se realiza por medio de una tarjeta SD de 32 GB. La velocidad de procesamiento de esta tarjeta es de 700MHz y el consumo de potencia es 2.5W. En la Figura 15 se puede observar la estructura de la tarjeta Raspberry Pi 3.

2.10.1. Requerimientos para la selección de la tarjeta

La tarjeta Raspberry que se seleccionó es la Pi 3, ya que cuenta con parámetros que facilitan la implementación de un sistema de videovigilancia. Esta microcomputadora cuenta con las siguientes características que ayudaron en su selección:

- 4 puertos USB 2.0
- 40 pines GPIO
- 1 puerto HDMI
- 1 puerto Ethernet
- Módulo *Wi-Fi* b/g/n en la banda 2.4
- Puerto para cámara
- RAM: 1GB
- CPU*: Quad-Core Cortex* A7 a 900MHZ
- GPU: *VideoCore* IV de doble núcleo

Figura 15. Tarjeta Raspberry Pi 3
(Raspberry.org, 2017)

2.11. Sistemas de Videovigilancia

Los sistemas de videovigilancia actualmente son de ayuda para el monitoreo de un área determinada. En empresas se busca contratar estos servicios para proteger sus instalaciones, mientras que en un domicilio sirve para salvaguardar sus pertenencias.

El uso de estos sistemas en algunos casos no se utiliza personal para que esté atento a las cámaras ya que la información puede ser registrada en grabaciones, las cuales pueden ser analizadas posteriormente en busca de información. Se debe considerar que las cámaras de videovigilancia no son autónomas, es decir, puede existir un robo y las cámaras no alertarán de la presencia de una persona, aunque estas grabaciones son realizadas en tiempo real, nada asegura que estos sistemas impida un robo. Por tal motivo se necesita de una herramienta que pueda avisar si existe la presencia de personal no autorizado.

Un sistema de videovigilancia está constituido por varias partes: cámaras de video vigilancia, estas pueden ser IP o analógicas, un monitor y un medio de transmisión de las imágenes, por lo tanto la adquisición de este sistema puede ser costoso.

Todos los conceptos que mencionaron en este capítulo, ayudan a entender cómo se desarrollará el detector de objetos. Además que se explicaron las herramientas que se utilizarán para la implementación física del prototipo, sin la compresión de estos temas no se podría desarrollar este trabajo de investigación.

CAPÍTULO III

DISEÑO E IMPLEMENTACIÓN

3.1. Introducción

En este trabajo de investigación se desarrolla un sistema embebido para detectar en tiempo real la presencia de personas en ambientes controlados. Una parte fundamental de este proyecto será desarrollar un detector de personas, que sea capaz de realizar la detección en tiempo real. En el capítulo anterior se explicó las características que debe poseer este detector, es decir, constará de dos partes fundamentales un descriptor y un clasificador. El descriptor será el que extrae las características de los objetos deseados, y necesitará de un conjunto de imágenes que contengan personas. Por medio de estas características se realizará el entrenamiento del clasificador. Finalmente se necesita de una herramienta que permita realizar la detección de personas en tiempo real. En este capítulo se explicarán las herramientas que se utilizaron para el desarrollo del descriptor y el clasificador, además se indica la implementación física del proyecto.

3.2. Metodología

Para el desarrollo del detector de personas se siguieron los pasos indicados en la Figura 16, en la cual se realiza adquisición de un conjunto de imágenes positivas y negativas de bases de datos certificadas. Se realiza la conversión de las imágenes para realizar la extracción de muestras positivas del conjunto de imágenes mediante el uso del programa *objectmarker*, se crea una matriz que contenga la ubicación de las personas en las imágenes positivas. Se desarrolla un vector mediante OpenCV que ayudará al entrenamiento del clasificador, y se procede a realizar el entrenamiento. Posteriormente se procede a realizar un programa que permita utilizar el

clasificador para realizar la detección, y se implementará físicamente el detector de personas para realizar la detección en tiempo real.

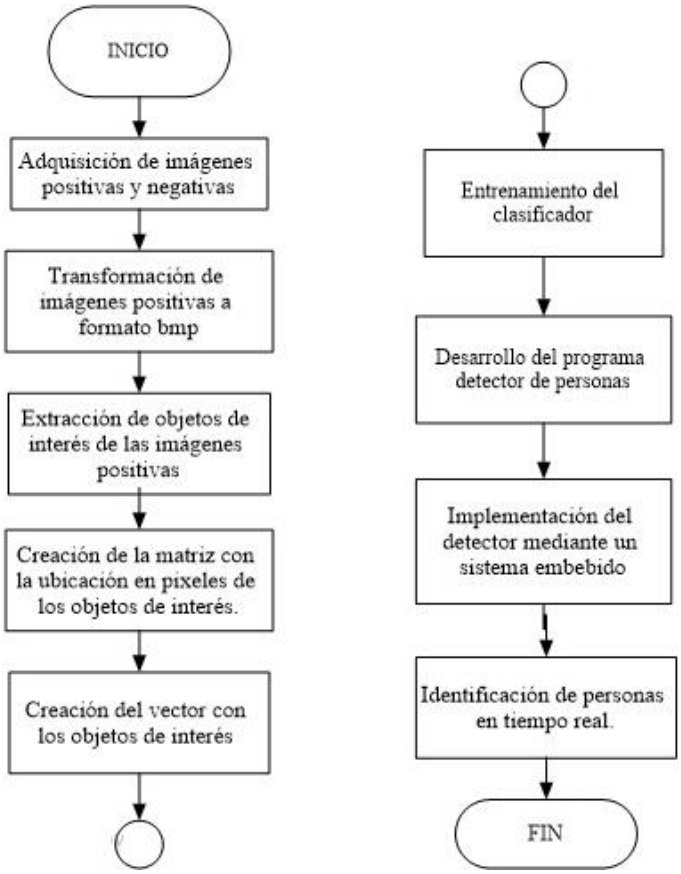

Figura 16. Metodología

3.3. Recolección de imágenes de muestra

La recolección de imágenes es un proceso complejo, ya que se necesitan muestras positivas y negativas, que faciliten el entrenamiento del clasificador. Por lo tanto se necesita una cantidad considerable de imágenes que contengan figuras de personas. Por este motivo se descargaron bases de datos certificadas como: INRIA (INRIA Person Dataset, 2005), PASCAL (Penn-Fudan Database for Pedestrian Detection and Segmentation, s.f.). Además se utilizó información provista

en la referencia (Cheng, Li, & Loy, 2016), para realizar un análisis de las imágenes. En total se utilizaron 1068 imágenes positivas y 1096 imágenes negativas, los set que se utilizaron no poseen las mismas dimensiones ya que provienen de diferentes autores, este factor no afecta en la selección de personas ya lo que importa es la postura en la que encuentren más no su tamaño en pixeles.

3.3.1. Muestras Positivas

Para que las imágenes positivas tengan una mejor calidad se las transformará a formato BMP (España, s.f.), ya que no se tienen pérdidas en la calidad de la imagen y es fácil de manipular. Además que, se puede guardar mayor cantidad de información. La conversión se realizó mediante el programa XnView, el cual permite transformar varias imágenes a la vez. Para la transformación al abrir el programa XnView se selecciona la conversión de imágenes como se observa en el Anexo C1.

En el Anexo C2 se puede observar la conversión de las imágenes, y es fundamental que se realice este paso ya que no se podrá realizar el descriptor si todas las muestras no poseen el mismo formato.

3.3.2. Muestras Negativas

Las muestras negativas no necesitan ser convertidas a formato BMP, solo se necesita que todas se encuentren en una misma carpeta, ya que no todas estas imágenes fueron extraídas de una misma base de datos, se realiza una transformación de los formatos de las imágenes, adicionalmente algunas muestras fueron tomadas en uno de los escenarios de prueba a los que fue sometido el detector. En la Figura 17 se pueden observar unas muestras de imágenes negativas.

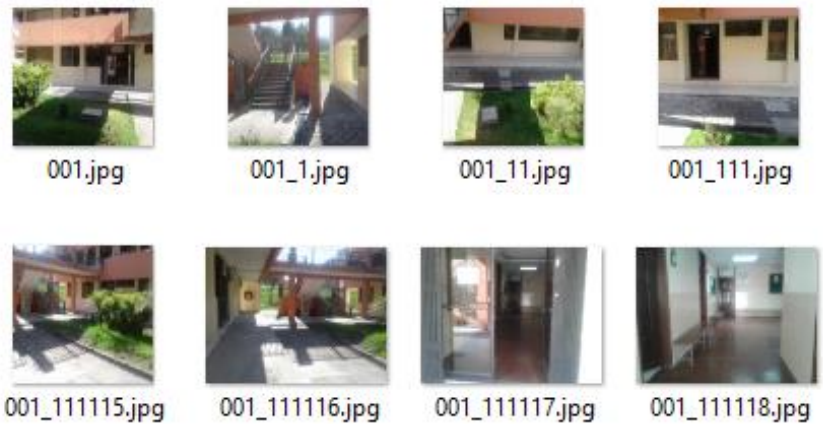

Figura 17. Imágenes Negativas

3.4. Extracción de características

3.4.1. Imágenes positivas

Para el entrenamiento del clasificador se usan varias características que ofrece OpenCV. Uno de los requisitos para acceder a estas herramientas es se cree un archivo con la información de las imágenes positivas, en el que se indique las posiciones de los pixeles en los que se encuentre ubicado el objeto de interés. Para facilitar este proceso se utilizó *objectmarker*, el cual crea un archivo que contiene la ubicación de las imágenes más la posición de los mismos. En la Figura 18 se puede observar un ejemplo. En la imagen se encuentran dos personas, al seleccionar a uno de los objetos se obtendrá: la coordenada en x (x), la coordenada en y (y), el ancho (width) y el alto (height), de cada persona seleccionada. El tiempo para realizar este proceso puede tardar dependiendo de: las muestras que se necesiten y del número de personas. En este trabajo fueron utilizadas un total de 1068 imágenes y el tiempo en que se realizó este proceso fue de una semana. En las imágenes analizadas se encontraba más de una persona, en algunas muestras existían grupos de 10 personas. En el Anexo C3 se tiene un ejemplo del contenido de las imágenes positivas. Este

documento es fundamental en la creación de un vector que contenga las características. Si existe un error en la información el vector no se creará, y no se contarán con las muestras necesarias para crear el clasificador.

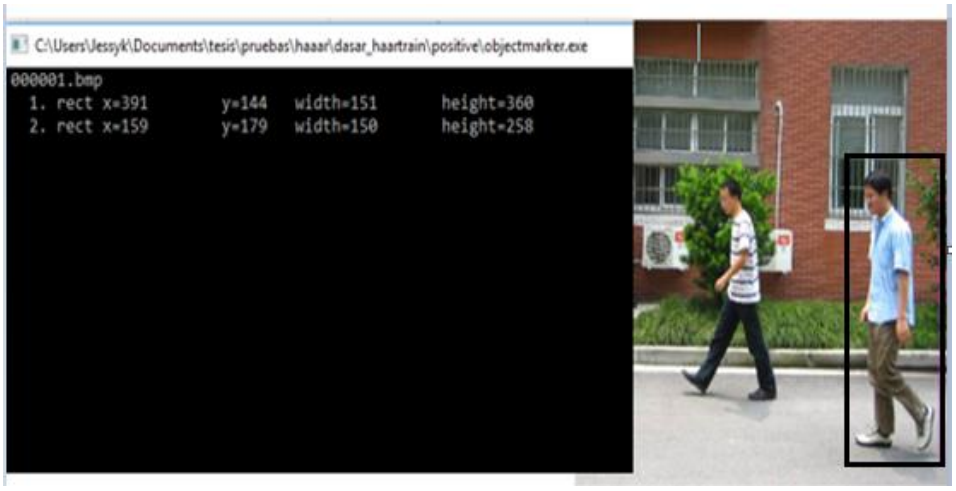

Figura 18. *objectmarker* selección de personas

3.4.2. Imágenes negativas

Para las imágenes negativas solo se necesita un documento que indique la ubicación de las muestras, ya que el algoritmo fue creado en el sistema operativo Ubuntu. Se utilizó un comando que permite extraer esta información de manera rápida:

find ./negative d >bg.txt

Si no se cuenta con el sistema operativo Ubuntu, se necesita crear un archivo que contenga la ubicación de las imágenes de forma manual o descargando un programa que facilite el proceso. Finalmente se obtienen dos documentos: bg.txt que es el tendrá la información de las imágenes negativas, y info.txt que es el que contiene la información de las muestras positivas.

3.5. Entrenamiento del Clasificador de Personas.

Después del proceso de recolección de imágenes y de la extracción de objetos, se procede a realizar el entrenamiento del clasificador. Este proceso es realizado por un algoritmo de visión artificial. Se utilizarán las herramientas que ofrecen las librerías OpenCV. Las características de la computadora que se utilizará para el desarrollo son muy importantes, debido a que el entrenamiento de los detectores Haar-AdaBoost y LBP-AdaBoost, es un proceso que puede tardar alrededor de dos días en entrenarse. Se podría acelerar este proceso si se realiza un procesamiento paralelo, el cual puede ser realizado si se cuenta con una tarjeta NVIDIA instalada en el computador, en este trabajo se realizó el entrenamiento con un computador que no cuenta con estas características por lo que el proceso tardo varios días en finalizarse. Las características que condicionan el tiempo de entrenamiento, dependen del vector que contiene las imágenes y del número de etapas seleccionadas.

3.5.1. Creación del vector que contiene imágenes de personas

Antes de empezar el entrenamiento se procede a crear un vector que contenga las características de las imágenes positivas. La aplicación *opencv_createsamples* es una herramienta de OpenCV, que permite la creación de un vector que las imágenes positivas de los objetos que se seleccionaron anteriormente. Luego se procede a guardar las imágenes en un archivo .vec. Los parámetros con los que se trabaja son los siguientes:

- -vec <*vec_file_name*>: es el nombre que se le dará al vector para el entrenamiento.
- -w: el ancho que tendrán las imágenes positivas.
- -h: la altura que tendrán las muestras positivas.
- -num: el número de las muestras positivas que en este caso son 1068.

- -info: ubicación del documento que contiene la posición de las imágenes.

Una de las consideraciones más importantes es el tamaño de las imágenes que en este caso será de 32 x 32 pixeles. Ya que si se mantiene el tamaño original de las muestras el consumo computacional al momento de desarrollar el clasificador será excesivo. En la Figura 19 se observa cómo se genera el vector.

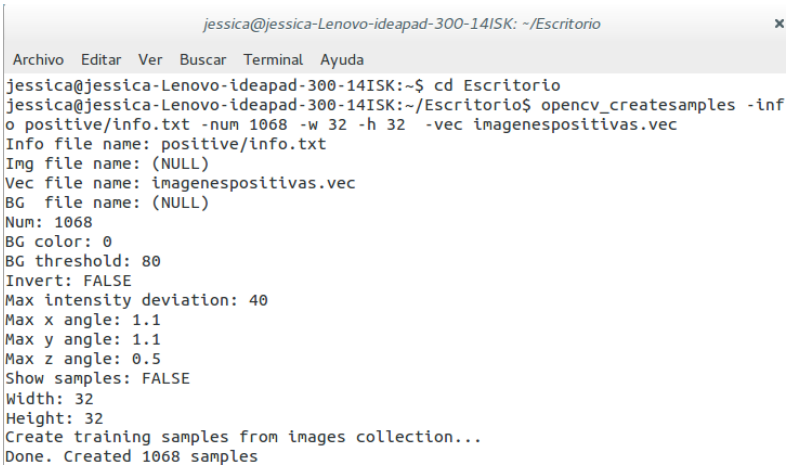

Figura 19. Creación vector de características

3.5.2. Entrenamiento clasificador

Para el entrenamiento del clasificador se utilizarán otra de las herramientas que posee OpenCV. En este caso es el comando *opencv_traincascade,* esta aplicación es la encargada de implementar el algoritmo detector. Es una parte fundamental del desarrollo ya que está diseñada, específicamente para el entrenamiento de clasificadores en cascada. Se puede trabajar con el método *Boosting,* el cual está formado por varios clasificadores débiles en forma jerárquica, y a partir de este método se pueden seleccionar el entrenamiento para el algoritmo AdaBoost, se trabajará con el clasificador *AdaBoost.* Este algoritmo necesitará de las imágenes positivas y negativas para su ejecución, ya que por medio de ellas se ira entrenado al clasificador para detectar

los objetos de interés del resto de imágenes analizadas. Esta herramienta de *OpenCV* proporciona la facilidad, que para la extracción de características se pueda usar dos técnicas como lo son las características *Haar* y *LBP*. Los parámetros considerados son:

- -*data:* carpeta donde se guardará el clasificador.
- -*vec:* vector de muestras positivas.
- -*bg:* fichero que contiene la ubicación de las imágenes negativas.
- -*numPos:* número de muestras positivas.
- -*numNeg:* número de muestras negativas.
- -*numStages:* número de etapas del clasificador.
- -*mode: ALL* usa todas las características rotadas.
- -*w:* es el ancho de las muestras.
- -*h:* es la altura de las muestras.
- -*featureType:* indica el tipo de descriptor a utilizar *Haar* y *LBP*.
- -*precalcValBufSize:* específica el espacio de memoria RAM que se utiliza para precalcular los valores de las características *Haar* y *LBP*, en este trabajo se utilizó de 512 MB.
- -*precalcIdxBufSize:* específica el espacio de memoria RAM que se utiliza para precalcular los índices de las características *Haar* y *LBP*, en este trabajo se utilizó de 512 MB.
- -*minHitRate:* es la tasa de éxito que tendrá el clasificador en cada etapa, hasta que el clasificador no tenga este valor continuará entrenándola.
- -*maxFalseAlarm:* es la tasa de falsas alarmas que el clasificador admitirá en cada etapa.

Antes de ejecutar esta línea de comando se calcula la tasa de falsa alarma que admitirá el algoritmo, y la tasa de éxito, para lo que aproximadamente se tendrán estos valores:

$$minHitRate = \frac{1000}{1068} = 0.95 \quad (11)$$

$$maxFalseAlarm = \frac{540}{1078} = 0.50 \quad (12)$$

Se implementarán dos clasificadores uno con un descriptor LBP y otro con un descriptor Haar por lo que desarrollarán dos algoritmos. Posteriormente se comprobará cuál de los dos algoritmos es más eficiente en la detección de personas. A continuación se encuentran los comandos que se utilizaron para el entrenamiento.

- Clasificador 15 etapas Haar

opencv_traincascade -data data -vec imagenespositivas1.vec -bg bg.txt –numPos 1000 –numNeg 1000 –numStages 15 -precalcValBufSize 512 –precalcIdxBufSize 512 -minHitRate 0.999 -maxFalseAlarm 0.5 -w 32 -h 32 -mode ALL

- Clasificador 15 etapas LBP

opencv_traincascade -data data -vec imagenespositivas1.vec -bg bg.txt –numPos 1000 –numNeg 1000 –numStages 15 –featureType LBP -precalcValBufSize 512 –precalcIdxBufSize 512 -minHitRate 0.999 -maxFalseAlarm 0.5 -w 32 -h 32 -mode ALL

Se pueden observar los parámetros obtenidos durante el entrenamiento, se encuentran detallados a continuación en el Anexo C4 este proceso demoró alrededor de un día y medio en el caso del detector Haar-AdaBoost. Mientras que el entrenamiento del detector LBP-AdaBoost se

generó en 10 minutos. La diferencia que en procesamiento de estos detectores de debe al análisis en las características que requiere cada uno de ellos.

En la Figura 20 se pueden observar los elementos que se obtienen después de finalizar el entrenamiento del clasificador. Se crean quince archivos que corresponden al número de etapas entrenadas. Adicionalmente se generan dos archivos en los cuales se encuentran los parámetros que necesita el clasificador para realizar el entrenamiento y otro que es el archivo que contiene al clasificador ya entrenado. El archivo que es utilizado para realizar la detección es el denominado cascade.xml, y para su uso se cambia su nombre ya que por defecto OpenCV denomina el archivo de ese modo.

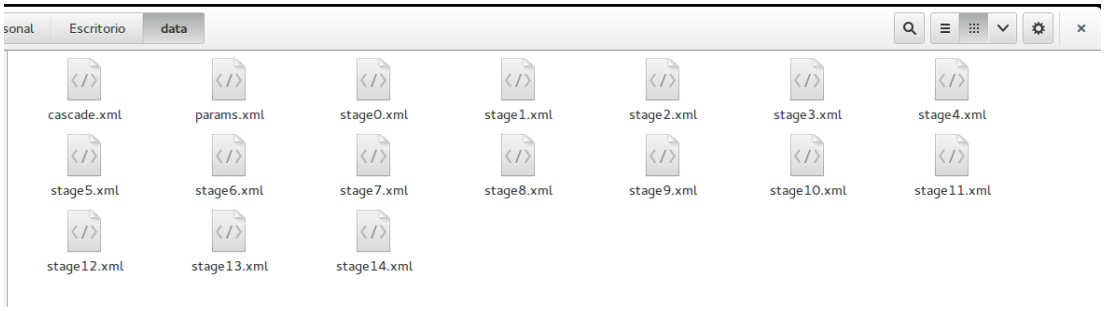

Figura 20. Elementos del clasificador Haar-AdaBoost

3.6. Desarrollo de la aplicación para la detección de personas

Una vez que se entrenó el clasificador se necesita un programa que permita al algoritmo estar en funcionamiento. Este trabajo busca realizar la detección de personas en tiempo real con un sistema embebido de bajo consumo computacional, razón por la cual se decidió analizar cual algoritmo es el que menos recursos computacionales necesitaba al momento de realizar la detección. En esta sección se explicará las herramientas que se utilizaron para ejecutar la detección.

3.6.1. Tarjeta Raspberry

Se escogió la tarjeta Raspberry Pi 3 por las características que ofrece, ya que cuenta con un sistema operativo propio. Este sistema operativo permite que sea un micro computador que facilita la instalación de la librería OpenCV. También cuenta preinstalado el lenguaje de programación Python 2.7, que es donde se realizó el programa de detección. Se debe realizar una instalación del sistema operativo en la tarjeta Raspberry Pi 3, el cual esta explicado en el Anexo A.

3.6.2. Instalación de OpenCV

La instalación de OpenCV, es uno de los pasos más importantes en este trabajo, ya que si no está instalado de manera correcta no se podrá verificar el funcionamiento del clasificador, ya que no se podrán acceder a las librerías de OpenCV. El uso de estas librerías son fundamentales en el desarrollo del programa, por defecto ya viene en el sistema operativo Raspbian viene instalado Python 2.7, el procedimiento para instalar estas librerías se explica en el Anexo B.

3.6.3. Implementación del programa detector de personas

El lenguaje de programación que se utilizó para el desarrollo del programa fue Python 2.7. Adicionalmente se utilizan las librerías de visión artificial que ofrece OpenCV, en conjunto estas dos herramientas ayudan a complementar al clasificador en cascada. En esta sección se explicará el proceso de desarrollo del código que permite la detección de personas en tiempo real por medio de la tarjeta Raspberry Pi 3 y una cámara que fue creada específicamente para trabajar con la Raspberry.

3.6.3.1. Diseño del código detector de personas

En la Figura 21 se ilustra el diagrama de flujo del código que será implementado. Estará constituido por dos secciones la detección y la alarma. La parte que más relevancia posee es la que

utiliza el clasificador para analizar los frames de video, y realizar la extracción de características. Se dibujará un rectángulo en la persona detectada y emitirá una señal de alarma. Si no se detecta ningún objeto de interés se volverá a realizar nuevamente el análisis en cada frame de video hasta identificar una persona.

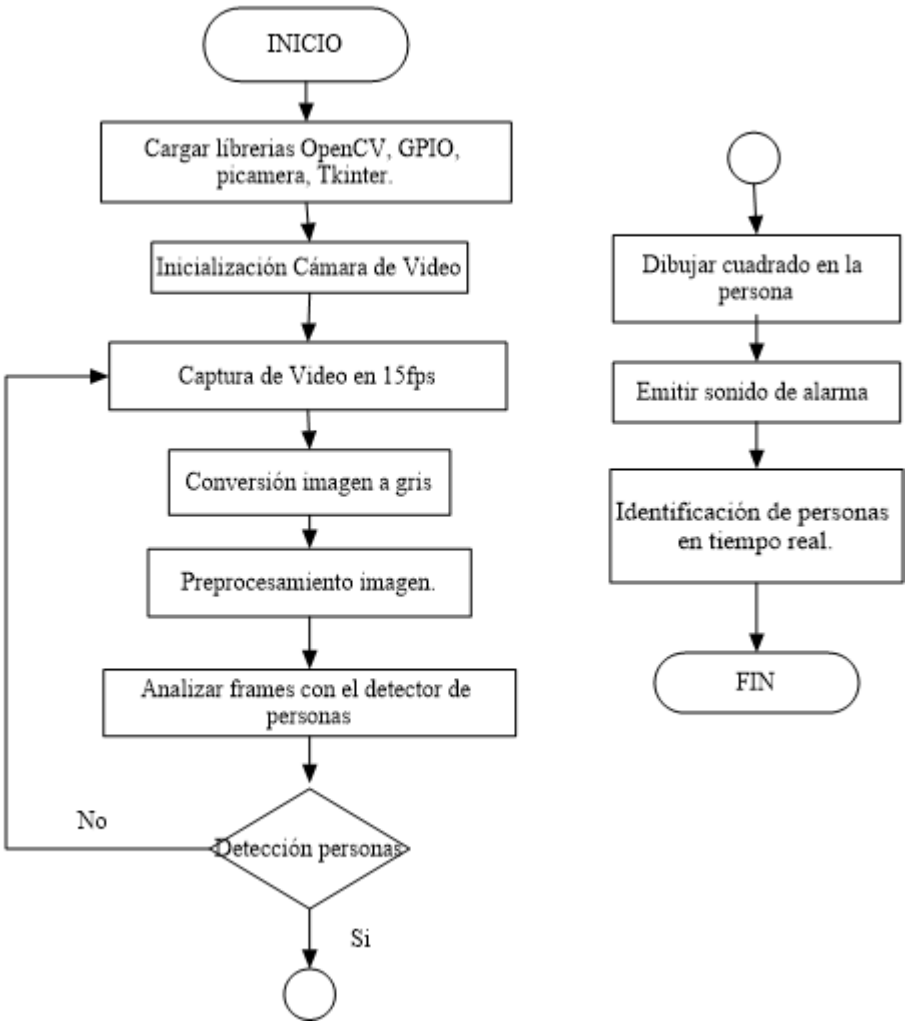

Figura 21. Diagrama de flujo del código

3.6.3.2. Desarrollo del código

El programa fue dividido en dos funciones: la primera es la encargada de emitir la señal de alarma al detectar la presencia de una persona, y la segunda es la que realiza la detección de personas. Antes de empezar con estos procesos se necesita realizar un análisis al video que se estará recibiendo. El tamaño del video a analizarse tiene resolución VGA que es de 640x480, ya que si se analiza con un tamaño mayor pueden generarse problemas en el consumo computacional de la tarjeta. Además solo se utilizarán 15 frames por segundo, para evitar que la raspberry tenga que analizar más imágenes de las necesarias, ya que si se analizan todos los frames del video que se presentan en la grabación, el proceso sería demasiado lento y consumiría recursos computacionales. Las líneas de código utilizadas son:

```
camera = PiCamera()

camera.resolution = (640, 480)

camera.framerate = 15

rawCapture = PiRGBArray(camera, size=(640, 480))
```

Al analizar con una cantidad menor a 15 frames y mayor a 10 frames por segundo, el detector de personas analiza las imágenes más rápido, y la detección es más eficiente ya que cuenta, con una menor cantidad de imágenes a analizar. El sistema no se verá afectado en el video, ya que no presenta retardo y el video será considerado en tiempo real.

OpenCV es la librería que ayuda en la detección de personas, en la función de detección de personas se usó la función *CascadeClassifier,* el cual es el encargado de cargar el clasificador en Python. El archivo denominado cascade15e_1000.xml es el clasificador creado, y solo se copia ese

archivo en la carpeta donde se encuentre ubicado el algoritmo. La línea de comando utilizada es la siguiente:

```
persona = cv2.CascadeClassifier('cascade15e_1000.xml')
```

Para poder realizar la detección se necesita que la imagen analizada sea convertida a escala de grises. Lo que permite mejorar el funcionamiento del clasificador. Para realizar la conversión se utilizó la función *cvtColor*:

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Después de que se trata a la imagen se procede a utilizar *detectMultiScale,* que en conjunto con el clasificador desarrollado detectará una persona. Para que se pueda visualizar de mejor manera la detección, se procede a dibujar un cuadrado en la ubicación del objeto de interés de la siguiente forma:

```
personas = persona.detectMultiScale(gray, 1.3,100)

for (x, y, w, h) in personas:

    cv2.rectangle(image,(x,y),(x+w,y+h),(255,255,0),4)
```

La función alarma es la que encarga de emitir un sonido cuando se detecte la presencia de una persona. Se realiza mediante los puertos GPIO que ofrece la Raspberry, se utilizará el puerto 27 como salida. La alarma sonará por un segundo antes de ser silenciada. Esta función es llamada cuando el algoritmo dibuje un rectángulo sobre la persona. Por lo tanto si no se detectó no sonará la alarma.

```
def alarma():
```

GPIO.setmode(GPIO.BCM)

GPIO.setup(27, GPIO.OUT)

GPIO.output(27, GPIO.HIGH)

Finalmente se realizó una interfaz gráfica que permite una mejor interacción al momento de trabajar con el programa. Para ello se utilizó la librería *Tkinter* que ofrece Python. Primero se importan las librerías de *Tkinter* que es la que permite crear una GUI. Después se procede a crear una ventana la cual se realiza de la siguiente manera:

*from Tkinter import **

vent = Tk()

vent.title("Detección de personas en tiempo real")

vent.geometry("1600x900")

vent.config(bg="white")

La creación de las etiquetas que contienen los títulos y los botones es muy similar. Solamente se utilizan las palabras *label* y *button.* El uso de estas se determina de acuerdo a como se desee que se visualice la aplicación. Se debe tomar en cuenta que cada etiqueta o botón tendrá un lugar diferente, por lo tanto se utiliza la palabra *place* y después se procede, a poner la ubicación en pixeles tanto en el eje x como en el eje y. La creación de un botón se realiza debido a que por medio de este se va a llamar a la función persona, que es donde se realizará todo el proceso de detección. La palabra *command* es la encargada de realizar esta acción. Al dar *click* sobre este botón

inmediatamente se inicializa la cámara de video y se procede con la detección. Un ejemplo de la creación de etiquetas y botones se realiza de la siguiente manera:

txt1=Label(vent,text="UNIVERSIDAD DE LAS FUERZAS ARMADAS-ESPE",bg="white",font=("helvetica",17,"bold"),fg="black").place(x=60,y=17)

botond = Button (vent,text="Iniciar Detección", command=detectar). place(x=300, y=450)

La interfaz gráfica está constituida por seis etiquetas y dos botones. Además de una pantalla denominada Detección Personas que es la que realiza todo el proceso de clasificación y detección de personas. En la Figura 22 se puede observar la interfaz creada para poder manipular de mejor manera la detección.

Figura 22. Interfaz programa

CAPÍTULO IV

PRUEBAS EXPERIMENTALES Y ANÁLISIS DE RESULTADOS

4.1. Introducción

En esta sección se analizará el rendimiento de tres clasificadores LBP-AdaBoost, Haar-AdaBoost, y HOG-SVM, las imágenes analizadas en los algoritmos son extraídas del video de una cámara que estará sometida a tres ambientes que estarán sometidos a cambios es la iluminación, además se ubicará al prototipo en distintas posiciones. Se realizarán pruebas tanto en ambientes interiores como en ambientes exteriores, para determinar cuál algoritmo es el mejor. Adicionalmente se debe tomar en consideración que este trabajo, se centrará específicamente en la detección de personas. Por lo tanto el parámetro que se analizará en el sistema embebido utilizado, será el consumo computacional que existe en al utilizar los tres detectores. Además que al ser un prototipo que busca emitir una señal audible en la detección de personas, realizará la captura de imágenes en tiempo real por medio de una cámara de video. Esto permite que al momento de visualizar la grabación, se considere que el sistema está trabajando en tiempo real.

Se debe tomar en consideración que para este proyecto un ambiente controlado es aquel que determina la intensidad de luz, en un ambiente exterior o interior, ya que al ser un detector de personas está sujeto a errores, y las condiciones lumínicas afectan en la detección. Por lo tanto para este trabajo se establecen ciertos escenarios, que permiten ver el comportamiento del algoritmo y verificar su funcionamiento.

4.2. Conjunto de sistemas de prueba

Las pruebas para verificar el funcionamiento del detector de personas se realiza mediante una tarjeta Raspberry Pi 3 y un computador. Se accede de manera remota a la tarjeta para verificar la detección, ya que la pantalla del computador será en donde se visualizará el video que este grabando la cámara. Se tiene que tomar en consideración que uno de los factores, que influye en la aparición de falsos positivos en la mayoría de los casos es debido a los niveles en la iluminación. La ubicación y la calibración de la cámara de video son muy importante para que existan menos errores en la detección.

La calidad del video de la cámara de la Raspberry para este trabajo es VGA, es decir, de 640x480 y es uno de los factores que puede influir en la detección, ya que al no ser tan claro el video el algoritmo puede llegar a confundirse, y generar falsos positivos y falsos negativos. En las pruebas se analiza la cantidad de verdaderos positivos que se tiene en la detección. Se utilizaron los algoritmos Haar-AdaBoost, LBP-AdaBoost, HOG-SVM para realizar los experimentos, y ver cuál de los tres clasificadores es más eficiente y confiable para este trabajo.

Para el desarrollo del clasificador se utilizaron imágenes de personas ubicadas de diferentes posturas, ya que se necesita detectar a una persona en varias posturas como son: de frente, de espaldas o de lado. Debido a que se desea realizar la detección de personas es fundamental que no solo detecte una sola postura. En las pruebas realizadas se verificó la precisión del clasificador sometiéndola a distintas situaciones.

4.2.1. Consideraciones Físicas

La toma de datos se realizó ubicando la cámara de video en una posición que facilita la detección, tratando de abarcar la mayor cantidad de espacio posible. En la Figura 23 a) se puede observar la ubicación de la cámara de video y en la Figura 23 b) se visualiza el espacio que se utilizó para verificar el comportamiento de los clasificadores. Se utilizó este tipo de escenario ya que los cambios en la iluminación son constantes, y el clasificador se comportaba de diferente modo en una hora específica del día.

Figura 23. Escenario ambiente exterior

Las pruebas se realizaron en los exteriores de los laboratorios de electrónica de la Universidad de las Fuerzas Armadas-ESPE, al ser una Institución Educativa la cantidad de personas que transitan por esta área es considerable. Por lo tanto se puede exponer el detector a un ambiente real. Con la cámara de video ubicada a una altura de 3m se obtiene una gran visibilidad del lugar. Lo que permitió analizar el alcance máximo al que el detector puede detectar sin equivocarse.

En este trabajo la distancia a la que el detector de personas alcanza a realizar la detección, es de aproximadamente 15m para el detector Haar-AdaBoost. Tomando en cuenta la iluminación del ambiente, si el día se encuentra nublado el alcance es mejor en comparación con un el ambiente exterior soleado. Estos problemas se presentan más al utilizar el clasificador LBP-AdaBoost.

En la Figura 24 están los tres escenarios en los que se realizaron las pruebas. Las condiciones lumínicas afectan los niveles de luz ocasionando inconvenientes. Un día soleado genera más falsos positivos debido a los reflejos de las ventanas, y las sombras que pueden producirse. En un ambiente cerrado iluminado con luz artificial la detección es más precisa con los algoritmos Haar-AdaBoost y HOG-SVM que con el detector LBP-AdaBoost. Los cambios muy bruscos en la intensidad de la luz generan problemas en la detección. Al trabajar con ambientes controlados se puede determinar cuál es el mejor escenario para el detector, determinando que si la intensidad de luz fuera constante no existieran problemas.

Figura 24. Ambientes de prueba

Tabla 1
Alcance de detección

Detector	Día Soleado	Día Nublado	Espacio Cerrado
Haar-AdaBoost	10m	15m	15m
LBP-AdaBoost	2m	10m	10m
HOG-SVM	1m	10m	17m

En la Tabla 1 se indica la distancia máxima de detección de cada detector en los escenarios analizados, utilizando la cámara de la tarjeta Raspberry Pi 3. Cuando se utilizó el algoritmo LBP-AdaBoost si el día estaba extremadamente soleado en ocasiones no detectaba, y todas las personas eran considerados cómo falsos negativos, ocasionando que el detector no sea efectivo. Esto se debe a las características que cada algoritmo pose. Como se mencionó en el Capítulo II, el algoritmo LBP es un algoritmo mucho más eficiente que Haar, ya que trabaja con integrales que hacen su análisis más sencillo teniendo un menor consumo computacional. Incluso en el entrenamiento de estos algoritmos la diferencia es que LBP, se desarrolló en 10 minutos y Haar se desarrolló en un día. Por lo tanto en su detección puede pasar por alto algunas características que Haar-AdaBoost considera. Hay que tomar en cuenta que los dos fueron entrenados con la misma cantidad de muestras positivas y negativas, además que el tamaño del vector utilizado en el entrenamiento fue de 32x32. Es importante mencionar que el detector HOG-SVM, fue utilizado solamente para realizar la comparación con los otros dos detectores, por lo tanto este no fue entrenado, y se utilizó un algoritmo creado por otro autor (Dalal, Triggs, & Schmid, 2006).

Cuando se realizaron las pruebas con los tres detectores, el consumo computacional de la tarjeta aumentaba dependiendo del detector que se estaba utilizando en ese momento. En la Tabla

2 se puede observar como aumentaba o disminuía el consumo computacional, si se utilizaba un filtro o no.

Tabla 2
Consumo Computacional del detector

Detector	Sin filtro	Con filtro
Haar-AdaBoost	50%	55%
LBP-AdaBoost	45%	50%
HOG-SVM	80%	85%

4.2.2. Pruebas de funcionamiento del sistema en ambientes exteriores.

Antes de empezar con la explicación de las pruebas en la Figura 25 se puede observar un ejemplo de un frame de video que contiene un verdadero positivo, un falso negativo y un falso positivo. Para obtener los valores de la matriz de confusión se realizó de forma manual la contabilización de todos los elementos mencionados en el Capítulo II para de ese modo determinar el desempeño del detector.

Se analizaron los frames de video para ir contabilizando los verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos para luego utilizar esta información y determinar la eficiencia del detector. Como herramienta de evaluación se utilizó la matriz de confusión, la cual nos permitirá obtener el desempeño de cada uno de los algoritmos propuestos, y determinar cuál de ellos posee un mejor rendimiento, ya que los tres clasificadores seleccionados tienen un consumo computacional bajo, y no sobrecargan la Raspberry. La selección del mejor detector se realizó al determinar cual posee una mayor precisión ya que es el parámetro que determina la cantidad de personas detectadas.

La matriz de confusión en este trabajo funciona con dos clases, la clase del objeto *persona* y la clase del objeto *no persona*., por lo tanto la matriz es de tipo binaria y permitirá obtener el valor de la precisión. Como ya se mencionó la precisión es el parámetro más importante en el análisis de este trabajo, ya que permitirá realizar la comparación de cada algoritmo y permitirá determinar su efectividad.

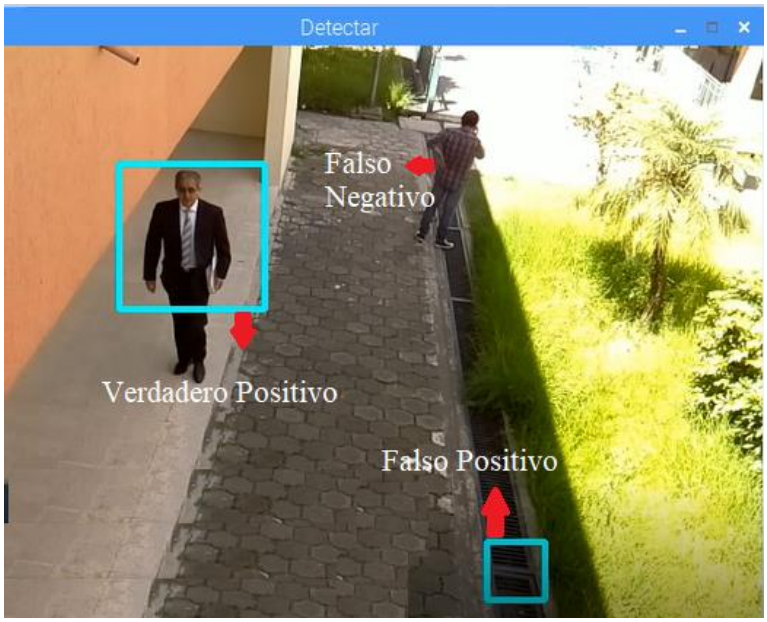

Figura 25. Elementos de matriz de confusión

4.2.2.1. Pruebas en un día soleado

Los detectores al tener problemas con el nivel de intensidad de luz se comportan de diferente manera, ya que en algunos casos existe la presencia de falsos positivos y falsos negativos, esto se produce debido a que se generan sombras en algunos lugares por la ubicación del sol confundiendo al detector.

4.2.2.1.1. Detector con Haar-AdaBoost

Cuando se realizaron pruebas con un día soleado, no solo se disminuía la distancia de detección, sino que también aparecían falsos positivos. En la Figura 26a) se puede observar un falso positivo. Debido a la intensidad de la luz, el detector distingue la forma de una persona ya que existen reflejos, y la cámara confunde esta sombra como a una persona. Al comparar la figura 26a) y 26b) se puede notar que aun cuando existe la presencia de un falso positivo este no está presente todo el tiempo y se realiza la detección sin ningún inconveniente

Figura 26. Detección día soleado Haar-AdaBoost

En este ambiente se tiene que tomar en consideración que dos detectores no pudieron ser evaluados, ya que no detectaban ninguna persona teniendo de ese modo un mal desempeño. En la Tabla 3 explica el uso de la matriz de confusión del algoritmo Haar-AdaBoost.

Tabla 3
Matriz de confusión detector Haar-AdaBoost

		Predicción	
		Persona	No persona
Realidad	Persona	Verdaderos Positivos 51	Falsos Negativos 15
	No Persona	Falsos Positivos 28	Verdaderos Negativos 90

4.2.2.1.2. Detector LBP-AdaBoost

Este detector presentó demasiados problemas en este tipo de ambiente. En un inicio no detectaba verdaderos positivos, inclusive no existía la presencia de falsos positivos. Se intentó colocar la cámara en una posición distinta, y no se obtuvieron resultados favorables. También se trató de detectar a una persona a una distancia muy corta consiguiendo la detección a medio metro de distancia lo que no era factible. En la Figura 27 se puede observar que aunque, la distancia es muy corta no se detecta ninguna persona por lo tanto en este tipo de ambiente el clasificador LBP-AdaBoost no es eficiente.

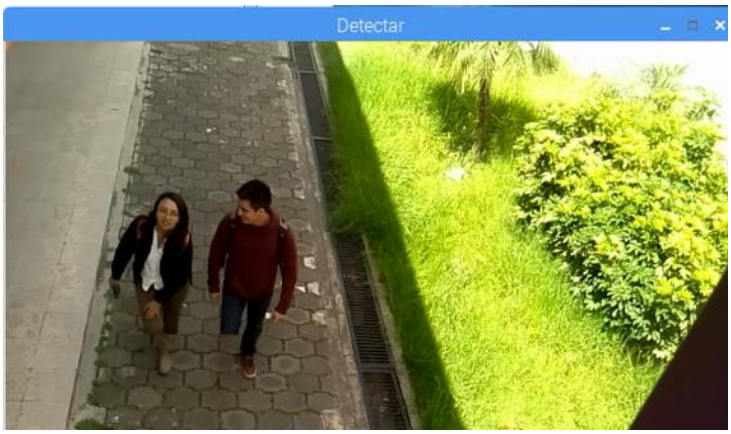

Figura 27. Detección día soleado LBP-AdaBoost

4.2.2.1.3. Detector HOG-SVM

Para las pruebas en este detector se presentaron varios problemas, debido a la iluminación y la ubicación de la cámara. No se detectaron personas y se ubicó la cámara en varios ángulos sin tener el resultado deseado, aunque no presentó falsos positivos, tampoco presentaba verdaderos positivos. En otros escenarios la detección si se produce pero en el caso de este escenario se obtuvo varias pérdidas no siendo un detector apto para estas condiciones. Este detector es uno de los más utilizados en detección de objetos para este tipo de escenario no fue óptimo. En la Figura 28 se observa como no existe detección, ni falsos positivos.

Figura 28. Detector día soleado HOG-SVM

4.2.2.1.4. Comparación de los resultados de los detectores en un ambiente exterior soleado

En base a los valores obtenidos de la matriz de confusión a continuación se explica el uso de los valores de cálculo utilizando las ecuaciones (7), (8), (9) (10). En la Tabla 4 se pueden visualizar los datos tabulados de los tres algoritmos. El parámetro que más importancia tendrá es la precisión ya que es el que determina las personas que fueron detectadas exitosamente.

$$Exactitud = \frac{Verdades\ Positivos + Verdaderos\ Negativos}{Predicciones\ totales}$$

$$Exactitud = \frac{51 + 90}{184} = 0.77$$

$$Precisión = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Positivos}$$

$$Precisión = \frac{51}{51 + 28} = 0.65$$

$$Sensibilidad = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos}$$

$$Sensibilidad = \frac{51}{51 + 15} = 0.77$$

$$Especificidad = \frac{Verdaderos\ Negativos}{Verdaderos\ Negativos + Falsos\ Positivos}$$

$$Especificidad = \frac{90}{90 + 28} = 0.76$$

En la Tabla 4 se puede observar el comportamiento del detector Haar-AdaBoost, el cual posee un 65% de precisión. Estos resultados muestran que este algoritmo tiene un comportamiento eficiente en su detección, ya que cuenta una de detección de personas que sobrepasa al 50%. Adicionalmente se debe considerar que los otros dos algoritmos no detectaron ninguna persona. Por lo tanto el detector Haar-AdaBoost es el más eficaz para este ambiente.

Tabla 4
Parámetros de los detectores en ambiente exterior soleado

Parámetros de evaluación	Haar-AdaBoost
Exactitud	0.77
Precisión	0.65
Sensibilidad	0.77
Especificidad	0.76

4.2.2.2. Pruebas en un día nublado

La detección en un día nublado no ocasiona grandes problemas, ya que el nivel de luz es más bajo y no se generan tantos reflejos ni sombras. El detector se comporta de manera eficaz en los tres casos. Las distancias máximas de detección también aumentan proporcionando un mayor alcance. En la sección anterior se mencionó que los algoritmos LBP-AdaBoost y HOG-SVM no funcionaron debido a la intensidad de luz en este caso los tres detectores funcionan correctamente.

4.2.2.2.1. Detector Haar-AdaBoost

Las pruebas que se realizaron en un día nublado fueron más eficientes ya que el detector pudo alcanzar una distancia de 17m. En la Figura 29a) se observa que el alcance del algoritmo aumenta, sin que la luz sea un inconveniente en la detección. La cantidad de falsos positivos disminuyó ya que no se presentaban sombras ni reflejos, eso no quiere decir que no exista la presencia de falsos

positivos o falsos negativos. En la Figura 29b) se puede observar que la detección disminuye su distancia. Debido al aumento el nivel de luz, en esta imagen se puede observar un falso negativo, ya que una persona no es detectada.

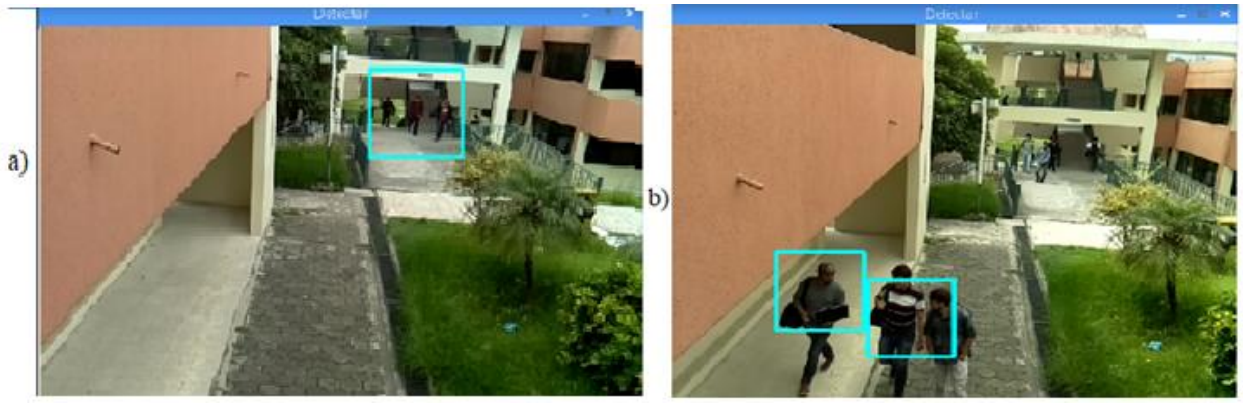

Figura 29. Detector Haar-AdaBoost en un Día Nublado

Para realizar la matriz de confusión se tomó una serie de datos, los cuales fueron analizados de los videos captados por la cámara de video. Los resultados obtenidos fueron utilizados para construir la matriz de confusión y se observan en la Tabla 5.

Tabla 5
Matriz de confusión detector Haar-AdaBoost

		Predicción	
		Persona	No persona
Realidad	Persona	Verdaderos Positivos 39	Falsos Negativos 10
	No Persona	Falsos Positivos 28	Verdaderos Negativos 57

4.2.2.2.2. Detector LBP-AdaBoost

En un día soleado no se pudo comprobar la eficiencia de este algoritmo ya que no se detectaron personas. En este caso ya se pudo realizar una serie de pruebas que permitieron determinar el rendimiento de este clasificador. En la toma de datos se tuvieron varios inconvenientes, ya que este tipo de algoritmo detecta a verdaderos negativos como verdaderos positivos, disminuyendo de ese modo su predicción. En la Figura 30a), se puede observar la detección de dos personas siendo una de ellas un verdadero negativo, por lo tanto el clasificador al poseer menos características asume que la silueta del árbol es una persona realizando mal la detección. En la Figura 30b) se puede observar un falso positivo, ya que se está detectando una persona y en realidad no hay ninguna persona. Por lo tanto este detector es más inestable que el algoritmo Haar-AdaBoost ya que produce más errores. La distancia de detección aumento aunque todavía no alcanza el nivel del clasificador Haar-AdaBoost.

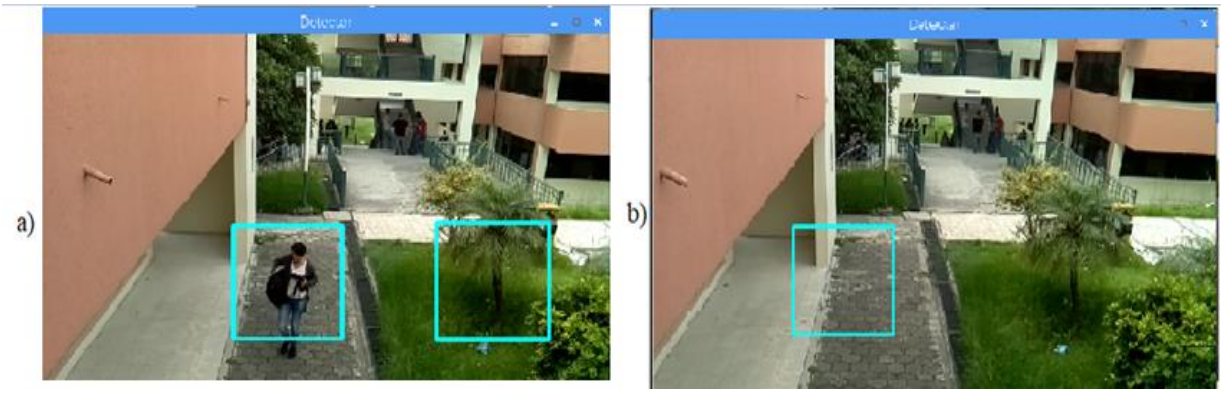

Figura 30. Detector LBP-AdaBoost en un Día Nublado

Para realizar la matriz de confusión se tomó una serie de datos, los cuales fueron analizados de los videos captados por la cámara de video. Los resultados obtenidos fueron utilizados para construir la matriz de confusión y se observan en la Tabla 6.

Tabla 6
Matriz de confusión detector LBP-AdaBoost

		Predicción	
		Persona	No persona
Realidad	Persona	Verdaderos Positivos 19	Falsos Negativos 10
	No Persona	Falsos Positivos 23	Verdaderos Negativos 68

4.2.2.2.3. Detector HOG-SVM

El funcionamiento de este algoritmo ya pudo ser verificado. En este tipo de escenario las condiciones en la iluminación ya cambiaron, y los niveles de luz son distintas. Se buscó una posición diferente de la cámara lo que permitió al detector una mejor visualización para realizar la detección. En la Figura 31a) se puede observar la detección con cuatro personas existe un falso negativo, ya que una persona no es considerada como persona por lo tanto ya existen pérdidas. Además la distancia de detección disminuye a 10m al igual que con LBP-AdaBoost. En la Figura 31b) se puede observar como la detección no es tan precisa con algunas posturas de las personas, al ser un detector desarrollado por otras autores este algoritmo no cuenta con algunas características, que permitan realizar la detección de personas en varias posiciones. Al pasar una persona junto a un grupo ocasiona confusión haciendo que estas personas no sean detectadas.

Figura 31. Detector HOG-SVM en un Día Nublado

Para realizar la matriz de confusión se tomó una serie de datos, los cuales fueron analizados de los videos captados por la cámara de video. Los resultados obtenidos fueron utilizados para construir la matriz de confusión y se observan en la Tabla 7.

Tabla 7
Matriz de confusión detector HOG-SVM

		Predicción	
		Persona	No persona
Realidad	Persona	Verdaderos Positivos 19	Falsos Negativos 10
	No Persona	Falsos Positivos 1	Verdaderos Negativos 70

4.2.2.2.4. Comparación de los resultados de los detectores en ambiente exterior nublado

En la Figura 32 se pueden observar los resultados de la detección en un escenario nublado la sensibilidad de 95% es del detector HOG-SVM, siendo capaz de detectar la mayor cantidad de personas cabe recalcar que este algoritmo fue desarrollado por otro autor (Dalal, Triggs, & Schmid,

2006), y adaptado para trabajar con la raspberry. El detector Haar-AdaBoost cuenta con un 63% de precisión, que permitió realizar la detección de una gran cantidad de verdaderos positivos siendo más estable para este ambiente. Al ser un algoritmo entrenado con un conjunto de características específicas permite detectar posiciones de personas que el algoritmo HOG-SVM no pudo. La cámara de video estaba ubicada a una altura de 3m que en ocasiones confunde la postura de una persona, siendo considerada como un falso negativo por el detector HOG-SVM. El algoritmo Haar-AdaBoost pudo realizar la detección sin tantos inconvenientes por la postura de personas. Ambos algoritmos podrían ser considerados como eficaces para este ambiente.

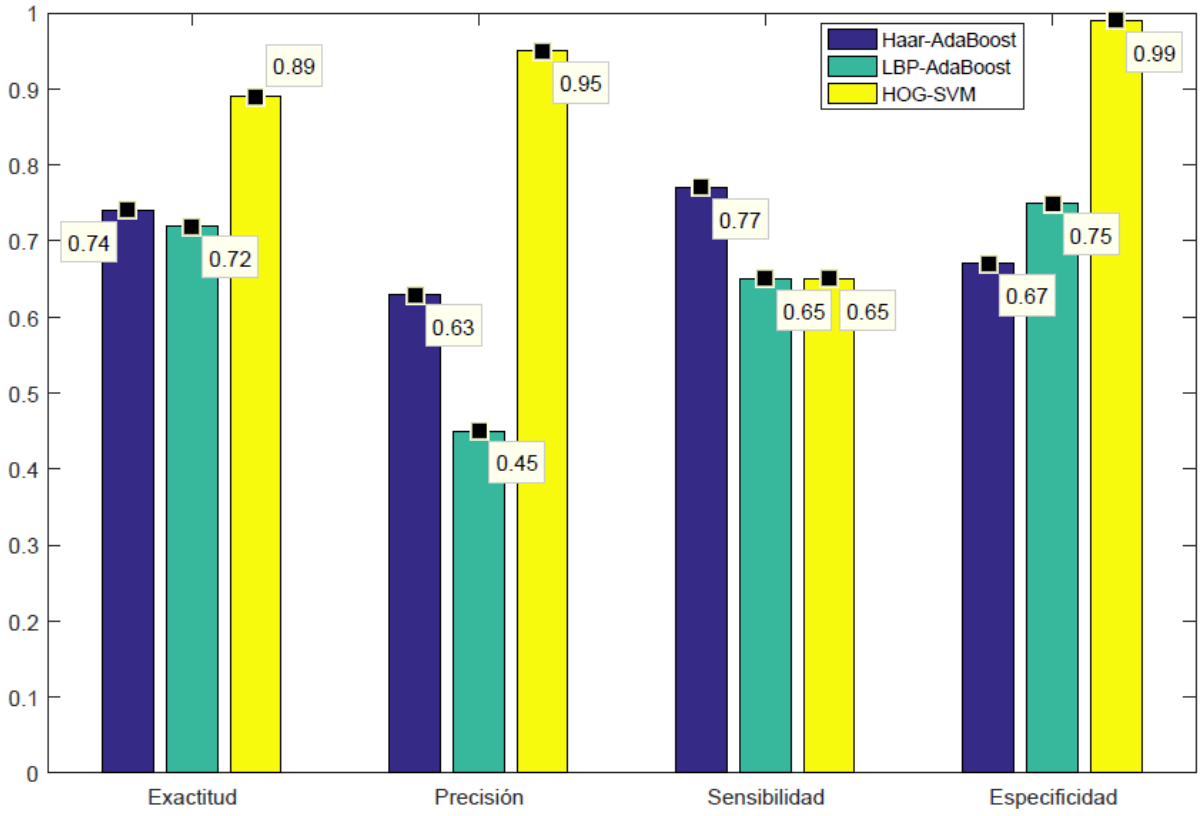

Figura 32. Ambiente Exterior Nublado

4.2.2.3. Pruebas en un ambiente interior controlado

Después de realizar pruebas exponiendo al detector a un ambiente exterior en el que la intensidad de la luz no puede ser controlada. Se buscó verificar el funcionamiento del detector en un ambiente controlado, es decir, en donde el nivel de luz ya es determinado por la mano humana. Se realizó la toma de datos en una habitación con la iluminación que ofrece un foco ahorrador. En este ambiente fue en el que mejor se comportaron todos los clasificadores ya que no tenían reflejos ni objetos que puedan hacer que el clasificador se confunda. Se realizaron pruebas en distintas posiciones de la cámara, aunque al ser una habitación no se pudo ubicar la cámara de video en un sitio muy alto.

4.2.2.3.1. Detector Haar-AdaBoost

El comportamiento del detector en este ambiente fue mejor que en los anteriores, ya que la intensidad de luz no es muy fuerte por lo que el algoritmo no tiene sombras con las que confundirse. Debido a la facilidad que presentó la detección en este tipo de ambiente se realizaron pruebas en distintas posiciones, como se puede observar en la Figura 33, se puede observar que sin importar la posición de la persona el clasificador la detectó sin problemas. En ciertas ocasiones se presentaban reflejos de una ventana o un objeto de cristal, ocasionando confusión en el algoritmo, también existían problemas en la detección cuando el algoritmo encontraba figuras con la silueta de una persona. La cantidad de falsos positivos es menor en comparación con los ambientes exteriores.

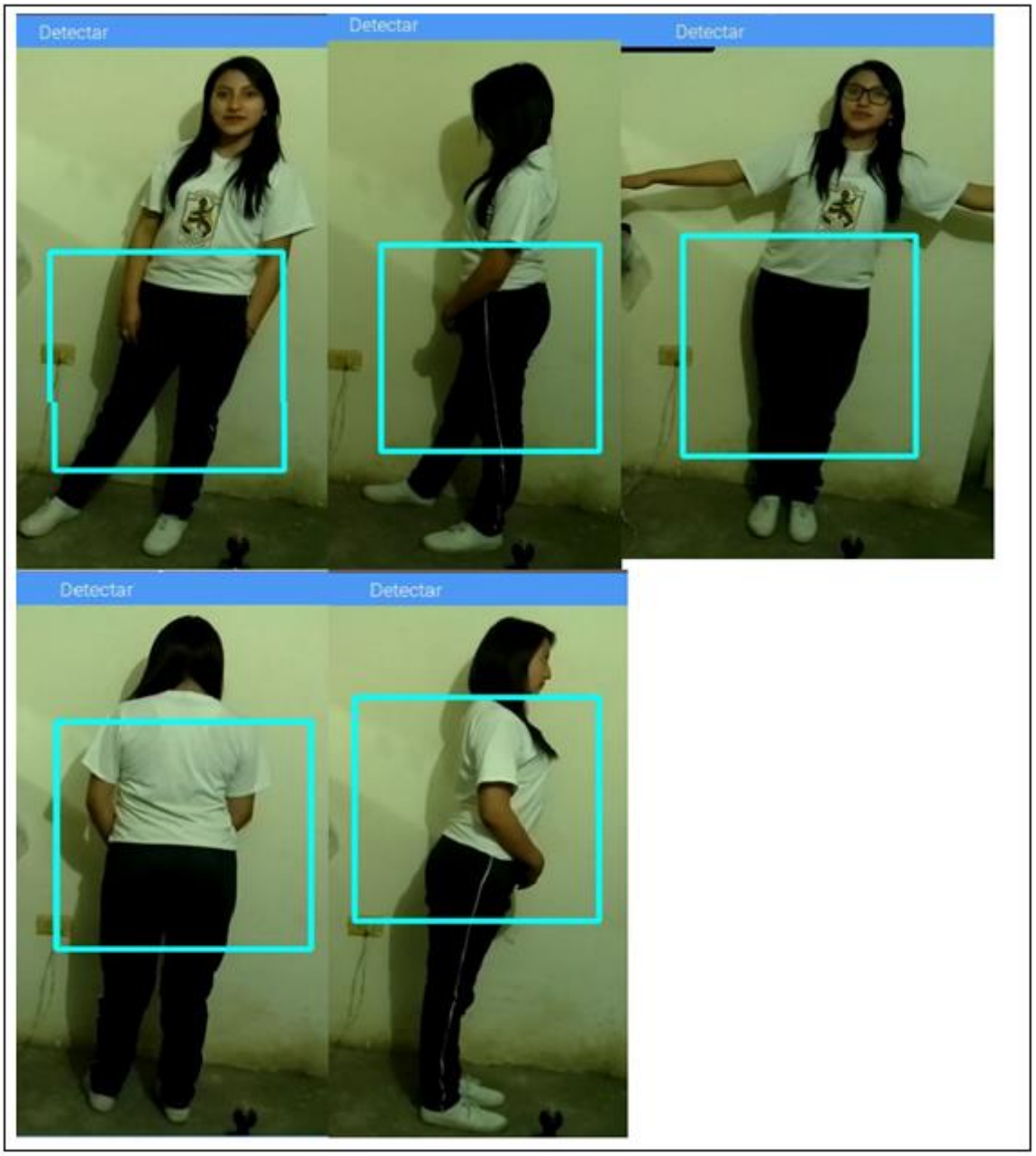

Figura 33. Detector Haar-AdaBoost en ambiente interior controlado

Para realizar la matriz de confusión se tomó una serie de datos, los cuales fueron analizados de los videos captados por la cámara de video. Los resultados obtenidos fueron utilizados para construir la matriz de confusión y se observan en la Tabla 8.

Tabla 8
Matriz de confusión detector Haar-AdaBoost

		Predicción	
		Persona	No persona
Realidad	Persona	Verdaderos Positivos 21	Falsos Negativos 11
	No Persona	Falsos Positivos 1	Verdaderos Negativos 20

4.2.2.3.2. Detector LBP-AdaBoost

Este detector en un ambiente controlado tuvo una nivel de detección mayor, aun presentaba ciertos inconvenientes como fue que dibujaba más de un rectángulo sobre una persona, determinando que esa misma persona representaba a dos. En todas las mediciones siempre se mostró constante un falso negativo, al igual que con el anterior clasificador se procedió a verificar el funcionamiento en varias posiciones de una persona, de ese modo se puede verificar que tan eficaz es el clasificador. En la Figura 34 se puede observar cómo se detecta más de una persona en una sola imagen. En ciertas posiciones se generan menos errores que en otras. Aunque se disminuyó la cantidad de falsos positivos mal interpretados como fue en el caso del clasificador en un ambiente nublado.

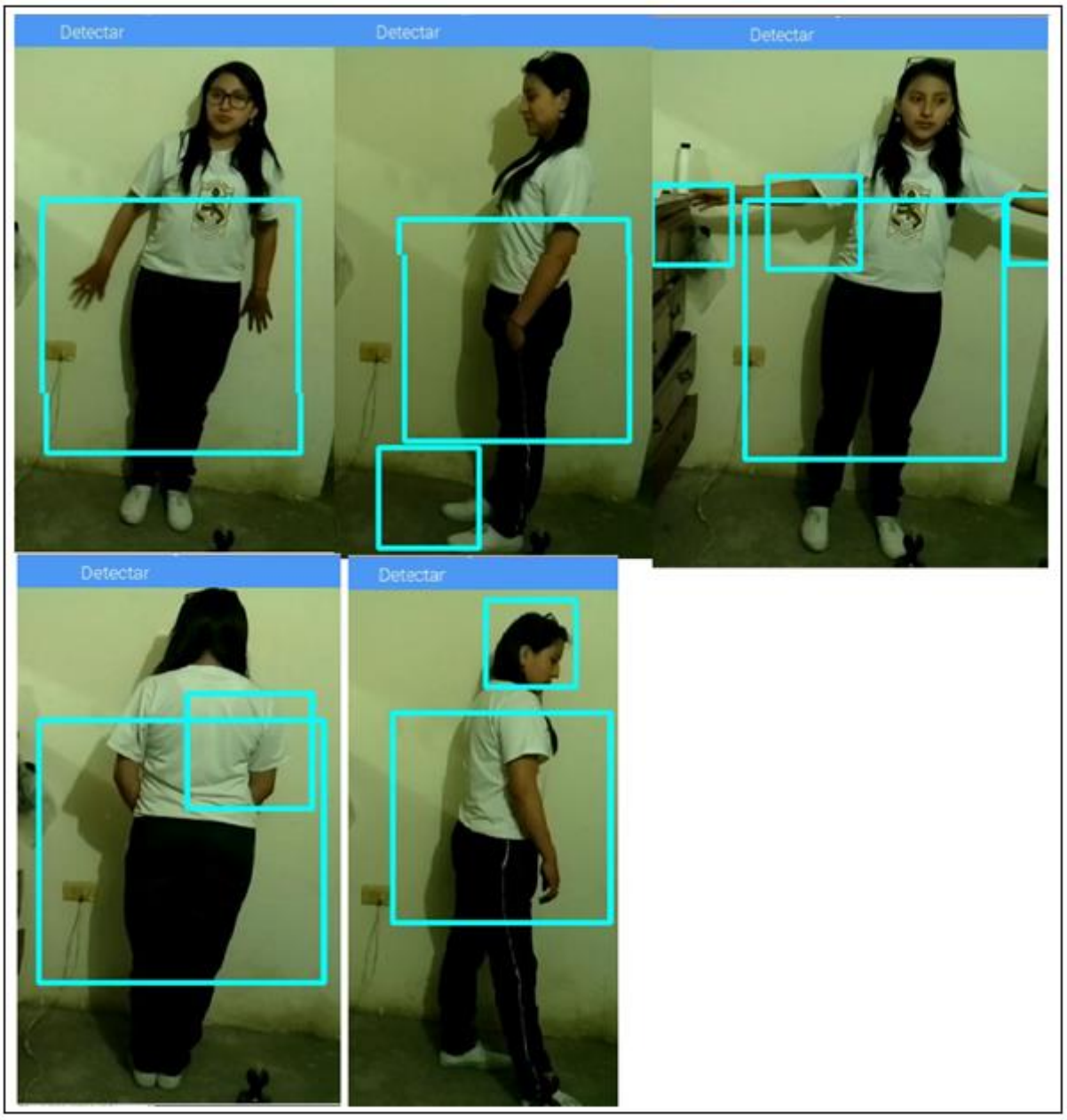

Figura 34. Detector LBP-AdaBoost en un ambiente interior controlado

Para realizar la matriz de confusión se tomó una serie de datos, los cuales fueron analizados de los videos captados por la cámara de video. Los resultados obtenidos fueron utilizados para construir la matriz de confusión y se observan en la Tabla 9.

Tabla 9
Matriz de confusión detector LBP-AdaBoost

		Predicción	
		Persona	No persona
Realidad	Persona	Verdaderos Positivos 18	Falsos Negativos 5
	No Persona	Falsos Positivos 10	Verdaderos Negativos 29

4.2.2.3.3. Detector HOG-SVM

Este detector en este escenario tuvo una mejor eficiencia, aunque tiene problemas en la detección al identificar a una sola persona como varias ya que se dibuja varios rectángulos en esa persona. Aunque no existen falsos positivos la detección también puede resultar poco eficaz al presentar varios objetos en uno mismo. En la Figura 35 se puede observar que se presentan varias detecciones, no en todas las posiciones se tienen estos inconvenientes pero en los casos más relevantes como son las posiciones de frente y espalda se producen estos problemas, esto se debe a se necesita que el detector sea más preciso al momento de analizar el video. No existe la presencia de falsos positivos algo que es muy importante en la detección de personas.

Para realizar la matriz de confusión se tomó una serie de datos, los cuales fueron analizados de los videos captados por la cámara de video. Los resultados obtenidos fueron utilizados para construir la matriz de confusión y se observan en la Tabla 10.

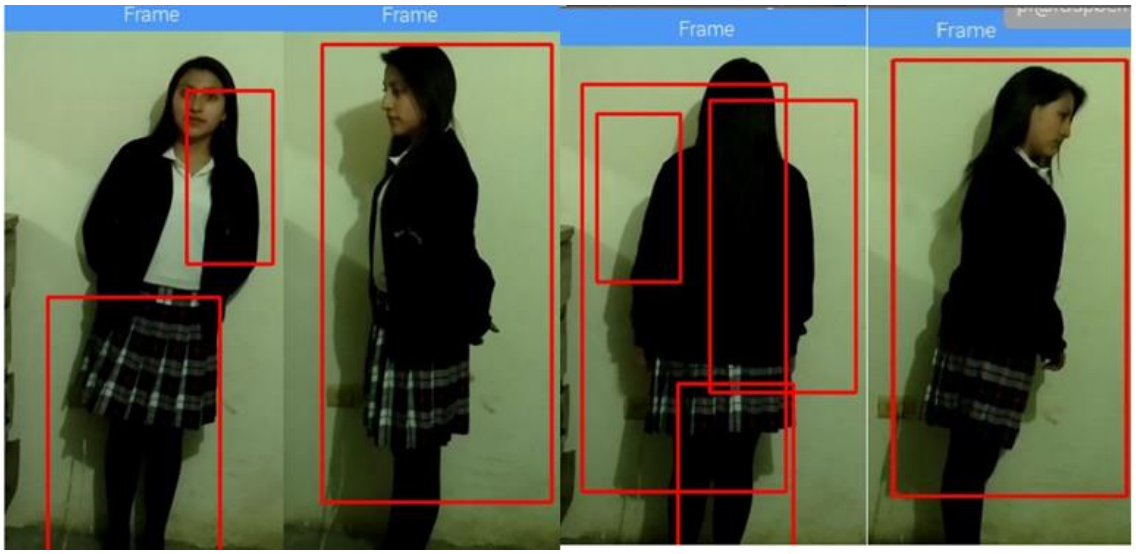

Figura 35. Escenario Interior Controlado Detector HOG-SVM

Tabla 10
Matriz de confusión detector HOG-SVM

		Predicción	
		Persona	No persona
Realidad	Persona	Verdaderos Positivos 25	Falsos Negativos 2
	No Persona	Falsos Positivos 2	Verdaderos Negativos 20

4.2.2.3.4. Comparación de los resultados de los detectores en ambiente interior controlado

Para realizar la tabla comparativa se utiliza la matriz de confusión, para después obtener los valores de los parámetros. En este ambiente la intensidad de luz es constante, y no sufre de cambios bruscos en su iluminación por lo tanto la cantidad de falsos positivos disminuye. En la Figura 36 se pueden observar los resultados de la detección en un escenario interior controlado, la

sensibilidad de 92% es del detector HOG-SVM, el algoritmo Haar-AdaBoost posee una sensibilidad del 88%. La iluminación en este ambiente permite que la detección sea más eficaz.

De acuerdo con los valores analizados anteriormente se puede determinar que para este trabajo de investigación el mejor detector es Haar-AdaBoost, ya que al ser entrenado con un grupo de imágenes seleccionadas, se puede realizar una mayor detección de personas en distintas posturas. El consumo computacional de la Raspberry con el detector Haar-AdaBoost no es excesivo, siendo de un 50% por lo tanto no sobrecarga el procesamiento de la tarjeta, y se evita que se sobrecaliente sin importar el tiempo que está encendida.

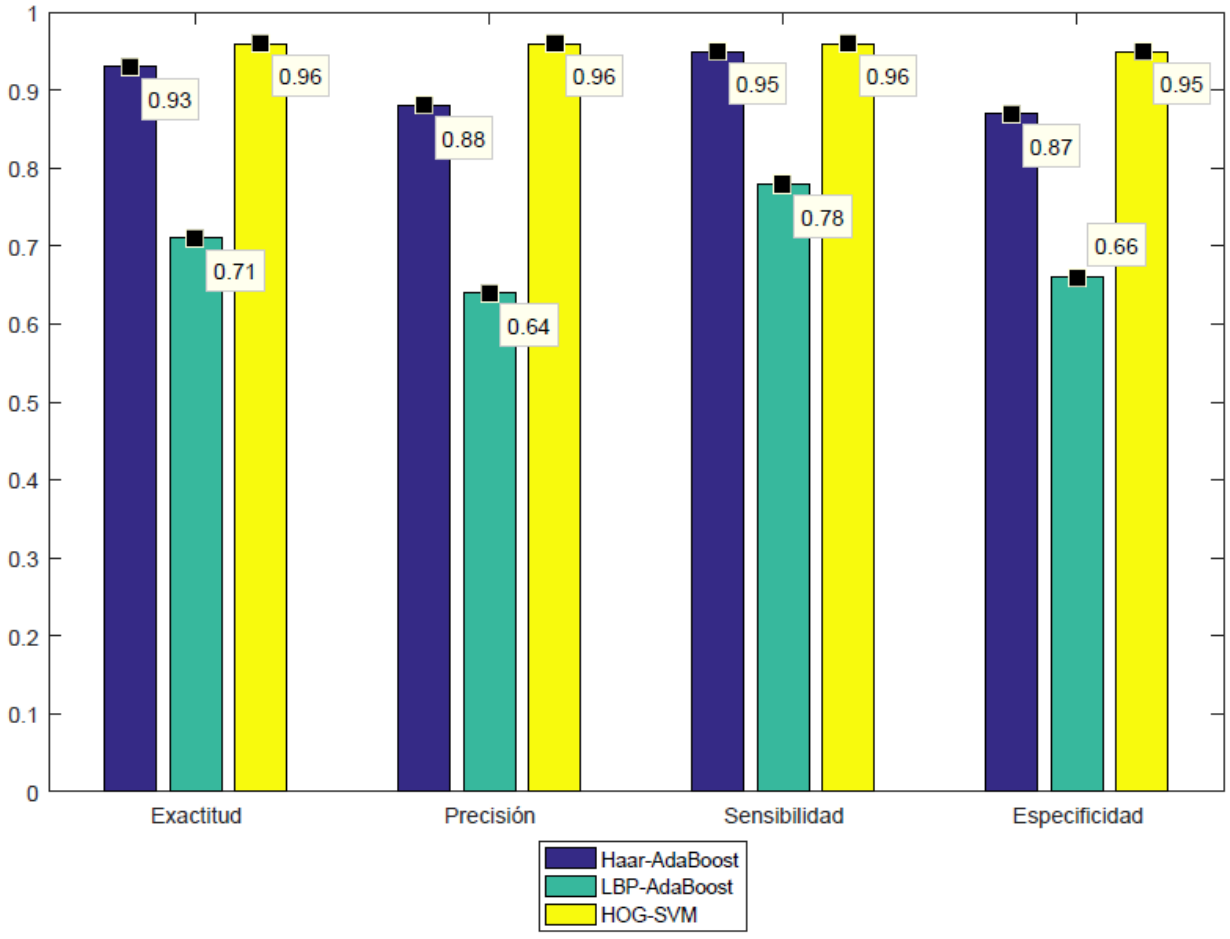

Figura 36. Ambiente Interior Controlado

El comportamiento de este algoritmo en los tres escenarios analizados fue superior al 60% en cada uno de los parámetros. Demostrando que es un algoritmo eficaz ya que realizó la detección sin muchos inconvenientes, ya que presenta una mayor cantidad de personas detectadas en cada ambiente, aunque presenta falsos positivos estos no afectan directamente en la detección. Adicionalmente se puede determinar que el detector más eficiente es el LBP-AdaBoost, ya que posee un menor consumo de recursos computacionales en su entrenamiento. Aunque no es recomendable trabajar con este detector ya que en las pruebas no fue muy estable en comparación con los otros detectores.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

La implementación se facilitó gracias a que la tarjeta Raspberry Pi 3 cuenta con un sistema operativo, que tiene preinstalado el lenguaje de programación Python. Facilitó la implementación del prototipo ya que se desarrolló que el código sea óptimo, mediante el uso de librerías de visión artificial (OpenCV). Si bien es cierto, el código desarrollado no es extenso el uso de las librerías de OpenCV fue de gran ayuda, para determinar la aplicabilidad que tendrá el programa.

Debido a las variaciones en la iluminación del entorno se pueden generar falsos positivos y falsos negativos en la detección, por lo tanto fue necesario el uso de filtros especiales para el preprocesamiento de la imagen, para mejorar la precisión de los detectores. Gracias a las múltiples pruebas realizadas por varios filtros que realizaron para el preprocesamiento de un frame de video, se pudo observar que la mejor respuesta fue del el *filtro de mediana*. Este filtro permitió acentuar los valores en la intensidad de la imagen resaltando la silueta y mejorando la detección de verdaderos positivos. Sin embargo, cabe recalcar que este filtraje incrementa el uso de recursos computacionales en la Raspberry en un 5%.

Dado que el entrenamiento de los clasificadores requieren un conjunto de imágenes compuesto por miles de imágenes, se utilizaron bases de datos certificadas con imágenes positivas y negativas tales como *INRIA* y *Penn-Fudan Database* cuyas imágenes positivas están compuestas por al menos una persona en diferentes posturas. De esta manera se evitan problemas por el derecho de privacidad de las personas, al momento de tomar fotos en la calle sin su consentimiento.

Los resultados obtenidos muestran que el detector de personas Haar-AdaBoost, tiene una mayor precisión alcanzó un 80% en un ambiente interior controlado. Aunque el detector de menor consumo computacional resultó ser el LBP-AdaBoost ya que consume un 45% de la capacidad de la Raspberry. Por lo tanto se puede determinar que mientras mayor precisión tenga el detector, existirá un mayor consumo computacional y viceversa.

El detector de personas Haar-AdaBoost fue el que presentó, un mayor alcance en la detección con 15m en un ambiente nublado, y 10 en un ambiente soleado. Por lo tanto se puede determinar que mientras mayor es la intensidad de luz la distancia de detección disminuye.

Este prototipo puede servir como elemento de un laboratorio de procesamiento de imágenes y video, ya que cuenta con características que facilitan el uso de librerías y lenguajes de programación de fácil la creación de algoritmos como fue el caso del detector de objetos, además que la construcción fue de un costo accesible.

5.2. Recomendaciones

Es recomendable realizar un estudio previo del uso las herramientas de OpenCV, para poder tener una mejor comprensión al momento de desarrollar el código.

Como objetivo en este trabajo de investigación, se planteó el uso de una cámara desarrollada específicamente para trabajar con la Raspberry Pi 3. Sin embargo se recomienda el uso de otra cámara con mayor resolución, para mejorar la precisión en la detección de objetos. Sin embargo, es necesario que se verifique si el consumo computacional aumenta, y si se puede seguir trabajando con este sistema embebido o si se necesita realizar una migración a otro sistema.

Es recomendable realizar pruebas para poder determinar el número de imágenes, que permitirán realizar el entrenamiento de los clasificadores. Adicionalmente se recomienda verificar si al utilizar un mayor número de imágenes el consumo computacional no se ve afectado.

Se recomienda que este prototipo sea utilizado como elemento de un laboratorio procesamiento de imágenes y video en proyectos académicos y de desarrollo tecnológico en aplicaciones de videovigilancia y seguridad física, ya que el desarrollo de este prototipo es de bajo costo, lo que permite que este tipo de tarjetas sean de fácil acceso.

Como trabajos futuros se recomienda implementar detectores de personas, y otros tipos de objetos utilizando técnicas de aprendizaje profundo como es el uso de redes neuronales, con el fin de mejorar la precisión de los mismos. Esto requerirá también el uso de herramientas computacionales y sistemas embebidos más potentes que la Raspberry Pi3, tales como el uso de FPGAs.

BIBLIOGRAFÍA

Andriluka, M., Roth, S., & Schiele, B. (2015). Pictorial structures revisited: People detection and articulated pose estimation. *IEEE Xplore*, 1-6.

Cheng, Z., Li, X., & Loy, C. C. (2016). *Color Naming via Convolutional Neural Network*. Obtenido de http://mmlab.ie.cuhk.edu.hk/projects/PCN.html

Dalal, N., Triggs, B., & Schmid, C. (2006). Human Detection Using Oriented Histograms of Flow and Appearance. *European Conference on Computer Vision*, (págs. 428-441).

España, M. d. (s.f.). *Diseño de Materiales Multimedia*. Obtenido de http://www.ite.educacion.es/formacion/materiales/107/cd/imagen/imagen0105.html

Fawcett, T. (2003). ROC Graphs: Notes and Practical Considerations for Data Mining Researchers. *HP Laboratories Technical Report*, 1-25.

Felzenszwalb, P., McAllester, D., & Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. *IEEE Xplorer*, 1-5.

Freud, Y., & Schapire, R. (1996). Experiments with a New Boosting Algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning*, (págs. 1-9).

Gallardo Calvopiña, E. S., & Sánchez Quevedo, E. X. (2016). *"Diseño y construcción de un sistema de autentificación con reconocimiento facial mediante procesamiento de imágenes con la utilización de software libre y tecnología raspberry pi"*. Latacunga: Universidad de Las Fuerzas Armadas.

Garrido Satué, M. (2013). *Reconocimiento de señales de tráfico para un sistema de ayuda a la conducción* . Universidad de Sevilla.

Guevara, M. L., Echeverry, J. D., & Urueña, W. A. (2008). Detección de rostros en imágenes digitales usando clasificadores en cascada. *Scientia et Technica*, 4.

He, D.-c., & Wang, L. (1990). Texture Unit, Texture Spectrum, and Texture Analysis. *IEEE Trans Geoscience and Remote Sensing*, 509-512.

Hernández Calandria, D., Cañas, A., & Diaz, A. (Enero de 2007). Detección de rostros humanos mediante SVM y generación de fotografías de tipo carné. *Researchgate*, 8. Obtenido de Researchate: https://www.researchgate.net/publication/267712643_Deteccion_de_rostros_humanos_mediante_SVM_y_generacion_de_fotografias_de_tipo_carne

INRIA. (2005). *INRIA Person Dataset*. Obtenido de http://pascal.inrialpes.fr/data/human/

Luna Gallegos, K. L., Palacios Hernandéz, E. R., & Marín Hernández, A. (2014). Detección y seguimiento de personas con analisis de color en datos RGB-D. *Detección y seguimiento de personas con analisis de color en datos RGB-D* (pág. 6). México: Congreso Latinoamericano de Control Automático.

Mallick, S. (6 de Diciembre de 2016). *Learn OpenCV*. Obtenido de Histogram of Oriented Gradients: https://www.learnopencv.com/histogram-of-oriented-gradients/

Maniscalco Martín, F. (2011)." *Identificación visual en ambientes subacuáticos".* Universidad de las Palmas de Gran Canaria.

Manlises, C., Martinez, J., & Belenzo, J. (2015). Real-time integrated CCTV using face and pedestrian detection image processing algorithm for automatic traffic light transitions. *IEEE Xplorer*, 1-4.

Mendieta, V. A. (2013). "*Detección y reconocimiento de semáforos por visión artificial".* Madrid: Universodad de Madrid.

Navarro Ubeda, R. (2004). *Aplicación de técnicas de Boosting para detección de matriculas.* Valencia: Universidad de Valencia.

Ojala, T., Pietikainen, M., & Harwood, D. (1994). Performance Evaluation of Texture Measures. *IEEE ICPR*, 582-585.

OpenCV. (20 de Diciembre de 2017). *OpenCV*. Obtenido de https://opencv.org/

OpenCV. (s.f.). *OpenCV.org*. Obtenido de http://docs.opencv.org/3.1.0/d6/d00/tutorial_py_root.html

Pajares Saenz, G. M. (2008). *Vision por computador: imagenes digitales y aplicaciones.* RA-MA.

Pang , Y., Yuan, Y., Xuelong , L., & Pan, J. (2011). Efficient HOG human detection. *Scient Direct*, 77-781.

PASCAL. (s.f.). *Penn-Fudan Database for Pedestrian Detection and Segmentation*. Obtenido de https://www.cis.upenn.edu/~jshi/ped_html/#pub1

Puebla, U. d. (s.f.). *Universdad de las Américas Puebla*. Obtenido de http://catarina.udlap.mx/u_dl_a/tales/documentos/lmt/morales_s_aa/capitulo3.pdf.

Rama, A., & Tarrés, F. (2007). Un nuevo método para la detección de caras basado en Integrales Difusas. *Simposium Nacional de la Unión Científica Internacional de Radio*, 4.

Raspberry.org, F. (20 de Noviembre de 2017). *Raspberry Pi*. Obtenido de https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Rodriguez Morales, R., & Sossa Azuela, J. H. (2011). *Procesamiento y análisis digital de imágenes.* Ra-ma .

Sanchez Matilla, R. (2014). *Detección Jerárquica de Grupos de Personas.* Madrid: Universidad Autónoma de Madrid.

Schapire, R. E. (1989). The Strength of Weak Learnability. *30th Annual Symposium on Foundations of Computer Science*, 28-33.

Tiempo, R. e. (02 de Mayo de 2000). *Qué son y cómo funcionan las cámaras web*. Obtenido de http://www.eltiempo.com/archivo/documento/MAM-1285331

Viola, P., & Jones, M. (2004). Robust Real-Time Face Detection. *International Journal of*, 137-154.

Viola, P., Jones, M., & Snow, D. (2005). Detecting Pedestrians Using Patterns of Motion and Appearance. *International Journal of Computer Vision*, 153-161.

Waring, C., & Liu, X. (2005). Face detection using spectral histograms and SVMs. *IEEExplore*, 467-476.