
UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

ESCUELA DE INGENIERÍA EN MECATRÓNICA

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN MECATRÓNICA

TEMA:

# "ANÁLISIS E IMPLEMENTACIÓN DEL ALGORITMO DE DETECCIÓN FACIAL DE VIOLA-JONES"

*"ANALYSIS AND IMPLEMENTATION OF THE VIOLA-JONES FACE DETECTION ALGORITHM"*

AUTOR: KAREN ELIZABETH RODRÍGUEZ BAQUE

DIRECTOR: CARLOS XAVIER ROSERO

IBARRA-ECUADOR
SEPTIEMBRE 2017


# UNIVERSIDAD TÉCNICA DEL NORTE

## BIBLIOTECA UNIVERSITARIA

## AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

## IDENTIFICACIÓN DE LA OBRA

La Universidad Técnica del Norte dentro del proyecto *Repositorio Digital Institucional*, determinó la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente información:

| **DATOS DEL AUTOR** | | | |
|---|---|---|---|
| CÉDULA DE IDENTIDAD | 1003668876 | | |
| APELLIDOS Y NOMBRES | RODRÍGUEZ BAQUE KAREN ELIZABETH | | |
| DIRECCIÓN | 10 de Agosto 12-61 | | |
| EMAIL | kerodriguezb1@utn.edu.ec | | |
| TELÉFONO FIJO | 062914339 | TELÉFONO MÓVIL | 0980904503 |
| **DATOS DE LA OBRA** | | | |
| TÍTULO | “ANÁLISIS E IMPLEMENTACIÓN DEL ALGORITMO DE DETECCIÓN FACIAL DE VIOLA-JONES” | | |
| AUTOR | KAREN ELIZABETH RODRÍGUEZ BAQUE | | |
| FECHA | SEPTIEMBRE DE 2017 | | |
| PROGRAMA | PREGRADO | | |
| TÍTULO POR EL QUE OPTA | INGENIERO EN MECATRÓNICA | | |
| DIRECTOR | CARLOS XAVIER ROSERO C. | | |

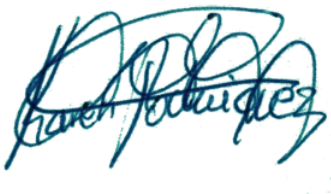
# AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, Karen Elizabeth Rodríguez Baque con cédula de identidad Nro. 1003668876, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en formato digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y uso del archivo digital en la Biblioteca de la Universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión; en concordancia con la Ley de Educación Superior, Artículo 144.

## CONSTANCIA

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló sin violar derechos de autor de terceros, por lo tanto la obra es original, y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, Septiembre de 2017



Karen Elizabeth Rodríguez Baque
C.I.: 10036688761

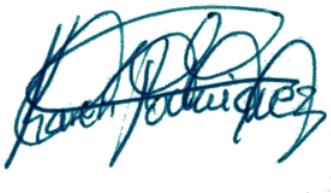
UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

# CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE GRADO A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, Karen Elizabeth Rodríguez Baque con cédula de identidad Nro. 1003668876, manifiesto mi voluntad de ceder a la Universidad Ténica del Norte los derechos patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador, artículos 4, 5 y 6, en calidad de autor (es) de la obra o trabajo de grado denominado "ANÁLISIS E IMPLEMENTACIÓN DEL ALGORITMO DE DETECCIÓN FACIAL DE VIOLA-JONES", que ha sido desarrollado para optar por el título de Ingeniero en Mecatrónica, en la Universidad Técnica del Norte, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente. En mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Técnica del Norte.

Ibarra, Septiembre de 2017



Karen Elizabeth Rodríguez Baque
C.I.: 1003668876

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

En calidad de director del trabajo de grado “ANÁLISIS E IMPLEMENTACIÓN DEL ALGORITMO DE DETECCIÓN FACIAL DE VIOLA-JONES”, presentado por el egresado KAREN ELIZABETH RODRÍGUEZ BAQUE, para optar por el título de Ingeniero en Mecatrónica, certifico que el mencionado proyecto fue realizado bajo mi dirección.

Ibarra, Septiembre de 2017

Carlos Xavier Rosero
DIRECTOR DE TESIS

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

DECLARACIÓN

Yo, Karen Elizabeth Rodríguez Baque con cédula de identidad Nro. 1003668876, declaro bajo juramento que el trabajo aquí escrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Universidad Técnica del Norte - Ibarra, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

Ibarra, Septiembre de 2017

Karen Elizabeth Rodríguez Baque
C.I.: 1003668876

# Agradecimiento

Todos los sentimientos encontrados al terminar esta etapa de mi vida son indescriptibles, cada una de las personas que han sido parte de mi formación personal y académica tienen un lugar muy importante dentro de mi corazón.

Agradezco primeramente a mis padres Felipe y Narcisa, que sé que sienten orgullosos al verme cumplir esta importante meta en mi vida, por todo el apoyo brindado, su paciencia y amor ya que sin ellos nada de esto hubiese sido posible, han sido la inspiración de todos mis sueños, me han enseñado a combatir los obstáculos que se presentan con valentia y rigor, me han cuidado en cada uno de pasos que he dado, simplemente me ense´ naron todo lo que una persona necesita saber para defenderse, salir adelante, levantar de una caida y los más importante que una persona puede tener en su vida que es la humildad. No existen palabras concretas que puedan describir todo lo que significan para mi.

Todas aquellas personas que formaron parte de mi vida y de las cuales de una u otra manera siempre aprendí algo, fue muy gratificante encontrarme en el camino con compañeros de clase con los cuales compartimos tareas, proyectos y mas que nada risas. Agradezco a mis profesores que me enseñaron todo lo necesario para poder realizar este trabajo y sobre todo a mi tutor el cual más que un tutor supo ganarse una amistad con la cual el trabajo fue mucho más gratificante y ameno.

Gracias a todos.

Karen Rodríguez Baque

# Dedicatoria

Éste trabajo quiero dedicar en primer lugar a mi abuelito Feliciano Baque ya que el ha sido para mi un ejemplo de vida, perseverancia, trabajo y amor. Todo el esfuerzo que realiza y ha realizado para salir adelante a sido una fuente de inspiración en mi vida.

Decicarles con todo el amor y todos los sentimientos encontrados a mis hermanos, Carolina y Felipe con el fin de crear en ellos la mentalidad de ser cada día mejores en todos los aspectos de su vida, sepan salir adelante y logren todo lo que se propongan y sean mil veces mejores que su hermana. Siempre estaré para apoyarles y guiarles en lo que pueda ya que a la vida hay que saber tomarla con calma y puedan cumplir todos sus sueños.

Karen Rodríguez Baque

# Resumen

La detección de objetos comprende varios campos entre los cuales se encuentra la detección de rostros. Éste es un poceso que se encarga de ubicar rostros presentes en imágenes o videos. Se constituye como el primer paso a seguir para los sistemas de reconocimiento facial aplicados en muchas áreas de la industria y la academia.

Existen diversos métodos que ayudan a la detección facial, entre ellos se encuentran la *detección facial robusta en tiempo real* propuesta por Paul Viola y Michael Jones, y los *patrones binarios locales* introducido por Jo Chang-yeon. Al encontrar diversos enfoques aparece la necesidad de verificar su desempeño computacional y capacidad de detección, para lo cual este trabajo realiza un estudio comparativo de los dos métodos más populares citados anteriormente.

Cada uno de los algoritmos ha sido implementado en *Python* con la ayuda de la librería de visíon por computador *openCV*. La base de datos utilizada para las pruebas de funcionamiento de los algoritmos contiene 50 imágenes de personas ecuatorianas de diferentes edades y etnias. Las dos aproximaciones se han evaluado considerando falsos positivos, falsos negativos y tiempo de ejecución.

Al finalizar la evaluación de todas las imágenes se obtuvo como resultado que el algoritmo de detección facial propuesto por Paul Viola y Michael Jones arroja como resultado un mayor acierto en detecciones y tiempo de ejecución que el algoritmo de patrones binarios locales.

# Abstract

Object detection consist of various fields among which is face detection. This is a process which is focused on locating faces found in images or videos. It is the first step for facial recognition systems applied in many areas in the industry and education.

There are many methods used in face detection, among them is the robust face recognition system in real time proposed by Paul Viola and Michael Jones, and local binary patterns introduced by Jo Chang Yeon. When finding diverse approaches there is the need to verify it digital performance and it detection capacity, because of such concerns this work makes a comparative study of the two most popular methods previously mentioned.

Each one of the algorithms has been implemented in Python with the help of an openCV digital visual catalogue. The data base which is used for the test runs for the algorithms has 50 images of Ecuadorian people of different race groups and ages. The two approximations have been evaluated considering false positives, false negatives and performance time.

When the evaluation of all the images finished as a result was obtained that the face detection algorithm proposed by Paul Viola and Michael Jones gives as a result a bigger success on detections and less time in the execution than the algorithm of local binary patterns.

# Índice general

# Índice de figuras

# Índice de cuadros

# Listings

# Capítulo 1

# Introducción

Este trabajo de grado ha sido realizado con el *Grupo de Investigación en Sistemas Inteligentes de la Universidad Técnica del Norte (GISI-UTN).*

## 1.1. Motivación

El rostro humano es un objeto dinámico que tiene un alto grado de variabilidad en su apariencia, lo cual hace que su detección sea un problema difícil de tratar en visión por computador. Inicialmente el problema de detección de rostros en los sistemas de reconocimiento no recibió la atención necesaria y se partía de que el rostro ya había sido detectado. Fue sólo en la década de los ochentas que surgieron los primeros algoritmos basados en técnicas heurísticas y antropométricas. En la década de los noventas cuando el desarrollo de algoritmos de detección de rostros inició su crecimiento se propusieron una gran variedad de técnicas, desde algoritmos básicos de detección de bordes hasta algoritmos compuestos de alto nivel que utilizan métodos avanzados de reconocimiento de patrones [1].

Estas técnicas se han abordado desde diferentes enfoques. En [2] se usa una base haar para la extracción de características y adaboost para la selección de estas y su clasificación, alcanzando un rendimiento alto de clasificación. Este método propuesto por Paul Viola y Michael Jones, es uno de los más usados hoy en día.

En este trabajo se presenta un análisis del algoritmo y su implementación en python, además se realiza una compración con otra técnica de detección facial para verificar su desempeño.

## 1.2. Objetivos

### 1.2.1. Objetivo Principal

- Analizar el algoritmo de VIOLA-JONES para detección de rostros.

### 1.2.2. Objetivos Específicos

- Abordar el estado del arte sobre detección facial.
- Analizar los parámetros del algoritmo de VIOLA-JONES.
- Establecer una metodología de comparación entre los diferentes enfoques utilizados para detección facial y el método usado por Viola-Jones
- Realizar pruebas de funcionamiento.

## 1.3. Antecedentes

Un detector de rostros tiene como finalidad indicar si una imagen contiene un rostro y si es así, donde esta. Un marco natural para considerar este problema es el de la clasificación binaria, en la cual se construye un clasificador para minimizar el riesgo de clasificación errónea. Ya que ninguna distribución objetiva puede describir la probabilidad previa real de que una imagen dada tenga un rostro, el algoritmo debe minimizar tanto las tasas falsas negativas como las falsas positivas para lograr un rendimiento aceptable [3].

Existen diversos métodos de detección facial, la mayoría de los cuáles no proporcionan una evaluación comparativa exhaustiva y contienen sólo un resumen del desempeño originalmente reportado entre varios algoritmos, las pruebas realizadas difícilmente pueden representar un rendimiento verdadero [4].

## 1.4. Problema

La detección facial es un paso esencial y, por lo general el primero en varias aplicaciones de visión por computador como el reconocimiento facial, las investigaciones criminales, los sistemas de acceso a la seguridad, la video vigilancia y la interacción inteligente entre computadores [5].

Dicho proceso consiste en encontrar rostros y determinar sus ubicaciones, existen muchos factores que pueden afectar el proceso de detección, tales como: escala, ubicación, orientación,

poses, expresión y condiciones de iluminación.

En la literatura hay diversos enfoques acerca de la detección facial, los mismos que se basan en inteligencia artificial, en la extracción de características como el color de piel, mientras que otros se basan en la en la integración de muchos enfoques juntos [6].

En la literatura no se han encontrado métodos de comparación referentes a algoritmos de detección facial, por lo que este trabajo consistirá en el análisis y búsqueda de una metodología de comparación entre el modelo de detector facial propuesto por Paul Viola y Michael Jones y otros tipos de detectores.

## 1.5. Alcance

El proyecto a realizarse, consistirá en el análisis e implementación de un algoritmo de detección facial, en este caso se analizará el algoritmo propuesto por Paul Viola y Michael Jones, el mismo que será sometido a comparación para establecer los parámetros que hace que sea idóneo al momento de realizar dicha acción. El algoritmo será implementado sobre una plataforma de software libre para su posterior desarrollo o implementación en distintos trabajos.

## 1.6. Justificación

La detección de rostros ha sido un área de investigación activa en la visión por computador durante años [5].

Hoy en día la técnica más popular de detección de rostros es el algoritmo de Viola Jones [2]. Paul Viola y Michael Jones en su camino hacia la detección de rostros presentaron un método que no necesita demasiado tiempo de cómputo y al mismo tiempo puede lograr una alta precisión. Este método se propuso para implementar un método de detección de rostros que es más rápido que cualquier otro [4].

Este método se ha convertido en las últimas décadas en uno de los algoritmos más importantes y ampliamente utilizados para la detección de rostros en aplicaciones en tiempo real, debido a su velocidad de detección no competitiva y precisión de detección relativamente alta [7].

La relevancia del trabajo propuesto consiste en una investigación que servirá como base para el establecimiento de una metodología de comparación entre el algoritmo de Viola-Jones y otros detectores faciales, demostrando porque este algoritmo es idóneo para la detección facial, bajo características específicas que aseguren su óptimo desempeño.

# Capítulo 2

# Visión por computador

La visión por computador es la extracción automatizada de información de imágenes. La información puede ser de modelos 3D, posición de la cámara, detección de objetos, agrupación y búsqueda de contenido de imágenes. La visión por computador en ocaciones trata de imitar a la visión humana, en ciertos casos utiliza enfoques estadísticos para resolver los problemas y en otros casos la geometría es la solución a los mismos [8].

## 2.1. OpenCV

OpenCV *Open Source Computer Vision Library* se publica bajo una licencia de BSD y por lo tanto es gratis tanto para uso académico como comercial. Tiene interfaces C++, C, Python y Java y soporta Windows, Linux, Mac OS, iOS y Android. OpenCV fue diseñado para la eficiencia computacional y con un fuerte enfoque en aplicaciones en tiempo real. Escrito en C/C++ optimizado, la biblioteca puede aprovechar el procesamiento multi-core. Habilitado con OpenCL, puede aprovechar la aceleración de hardware de la plataforma de computación [8].

### 2.1.1. OpenCV-Python

Python es un lenguaje de programación de propósito general iniciado por Guido van Rossum, que se hizo muy popular en poco tiempo principalmente debido a su simplicidad y legibilidad de código. Permite al programador expresar sus ideas en menos líneas de código sin reducir la legibilidad.

El apoyo de numpy hace que la tarea sea más fácil. Numpy es una biblioteca altamente optimizada para operaciones numéricas. Proporciona una sintaxis de estilo MATLAB. Todas las estructuras de la matriz de openCV se convierten a y desde matrices numpy. Así que cualquier operación que pueda hacer en numpy, se puede combinar con openCV, lo que aumenta el número de armas en su arsenal.

Así que OpenCV-Python es una herramienta apropiada para el prototipado rápido de problemas de visión por computador [8].

## 2.2. Algoritmos de detección facial

### 2.2.1. Algoritmo de Viola-Jones

El método de detección de objetos de Viola Jones fue propuesto por Paul Viola y Michael Jones en 2001, que fue el primer método que proporciona tasas de detección de objetos relativamente altos. Se puede utilizar para detectar objetos en tiempo real, pero se aplica principalmente a la detección de rostros. La tasa de detección de este método es relativamente alta y muy baja con relación a flasos positivos, lo que hace al algoritmo tan robusto y procesa las imágenes rápidamente. Su objetivo principal es la detección de rostros más no el reconocimiento, es decir, distiguir las caras de las no caras, este procedimiento es el primer paso para el reconocimiento facial [9].

### 2.2.2. Patrones binarios locales (Local binary patterns, LBP)

Los patrones binarios locales se utilizaron por primera vez para describir texturas ordinarias donde la relación espacial no era tan significativa como lo es para las imágenes de rostros. Un rostro puede ser visto como una composición de micro texturas dependiendo de la situación local. El LBP se divide en dos descriptores diferentes: un global y un local. El global se utiliza para discriminar la mayoría de objetos no caras para la detección [10].

### 2.2.3. Detección de rostros usando la distancia de Hausdorff

Este método se basa en bordes y funciona en imágenes fijas en escala de grises. La distancia de Hausdorff se utiliza como medida de similitud entre un modelo de cara general y las posibles instancias del objeto dentro de la imagen. Es un modelo que tiene dos pasos los cuales permiten tanto la detección gruesa como la localización exacta de las caras [11].

### 2.2.4. Detección de rostros con segmentación de tono de piel

El algoritmo utiliza un nuevo modelo de color de piel, RGB-HS-CbCr para la detección de rostros humanos, las regiones de la piel se extraen utilizando un conjunto de reglas de límites basadas en la distribución del color de la piel obtenida de un conjunto de entrenamiento. Las regiones de la cara segmentada se clasifican adicionalmente usando una convinación paralela de operaciones morfológicas simples. Los resultados experimentales en un gran conjunto de datos fotográficos han demostrado que el modelo propuesto es capaz de lograr buenas tasas de detección para caras casi frontales de diferentes orientaciones, color de la piel y entorno de fondo [12].

### 2.2.5. Detección de rostros usando redes neuronales

El sistema realiza una búsqueda de la cara guiada en regiones de interés que exhiben propiedades de color de la piel humana. Estas propiedades se detectan en una base de pixel por pixel. El sistema propuesto puede utilizarse como un módulo de sistemas de reconocimiento de rostros, sistemas de videovigilancia, sistemas de control de acceso [13].

### 2.2.6. Ventajas y desventajas de Viola-Jones

El algoritmo de detección facial propuesto por Paul Viola y Michael Jones presenta una serie de ventajas y desventajas que son detallas en los cuadros 2.1 y 2.2.

Cuadro 2.1: Ventajas

| Ventajas |
|---|
| Selección eficiente de características |
| Invariante a escala y localización |
| Utilización de filtros haar en lugar de puntos de concordancia |
| El tipo de entrenamiento se puede usar para detectar diferentes objetos |

Cuadro 2.2: Desventajas

| Desventajas |
|---|
| Más eficiente en imágenes de rostros frontales |
| Sensible a cambios de iluminación |
| La cascada deslizante puede obtener varias detecciones del mismo rostro |

# Capítulo 3

# Viola-Jones

La metodología se basa en la propuesta de [2]. Esta se divide en tres etapas como se muestra en la Figura 3.1, en la primera etapa se realiza una transformación de la imagen generando una nueva llamada imagen integral, en el segundo bloque se realiza la extracción de características usando filtros con base haar[1], y por último se usa boosting[2] para la construcción de clasificadores en cascada.

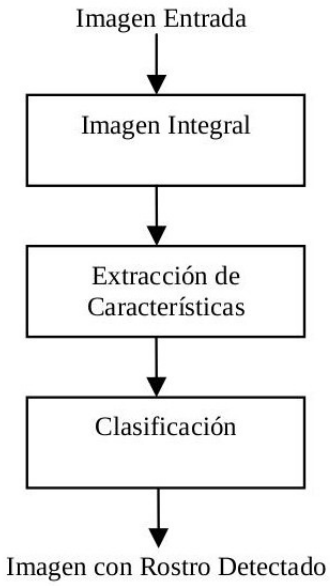



Figura 3.1: Diagrama de bloques

[1] Filtros de segmentación
[2] Algoritmo de aprendizaje

## 3.1. Características

### 3.1.1. Imagen integral

Esta nueva representación de una imagen fue introducida en [2]. Esta imagen permite extraer de forma rápida características a diferentes escalas ya que no se trabaja directamente con los valores de intensidad si no con una imagen acumulativa que se construye a partir de operaciones básicas.

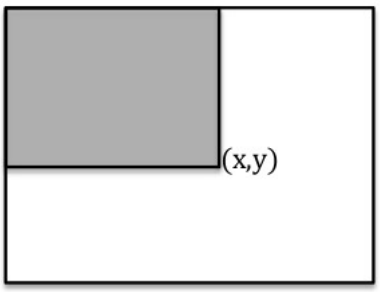



Figura 3.2: Imagen integral

La imagen integral (Figura 3.2) en la localización $x$, $y$, contiene la suma de los píxeles de la parte superior izquierda de la imagen. La suma de todos los píxeles de cualquier rectángulo en la imagen como se puede calcular a continuación

$$II(x,y) = \sum_{x'<=x,y'<=y} Im(x',y') \tag{3.1}$$

donde $II(x,y,)$ es la imagen integral e $Im(x,y)$ es la imagen original.

#### 3.1.1.1. Cálculo de imagen integral

Un cálculo muy eficiente con sólo un recorrido por toda la imagen a partir de la suma acumulada de la fila actual y el valor de la imagen integral en la fila anterior (ver Figura 3.3).

### 3.1.2. Características Haar

En imágenes las características de cada objeto se extraen al aplicar ciertas funciones que permitan la representación y descripción de los objetos de interés de la imagen (patrones). La extracción de características es un paso en el reconocimiento de patrones en el cuál las medidas u observaciones son procesadas para encontrar atributos que puedan ser usados para asignar los

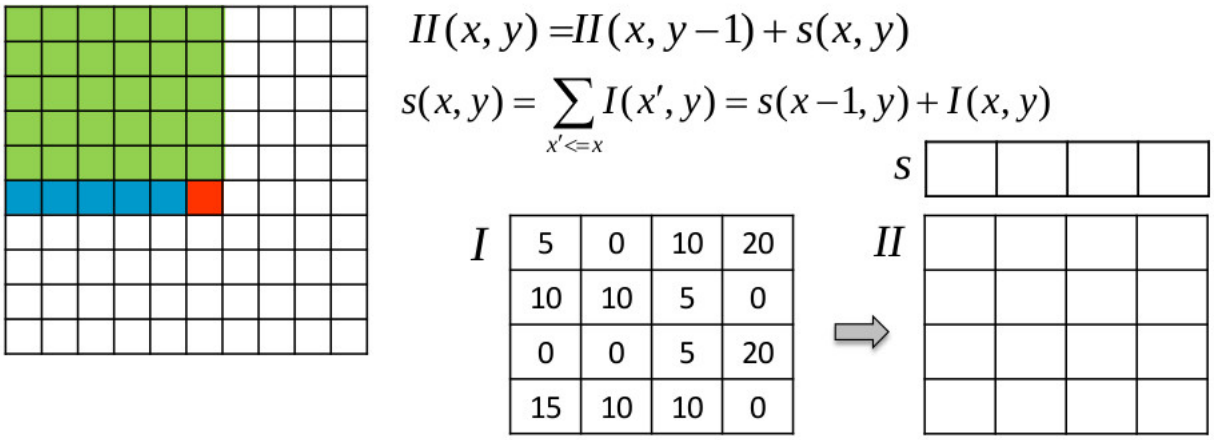



Figura 3.3: Cálculo de imagen integral

objetos a determinada clase.

En la metodología seguida, la extracción de características es realizada aplicando a la imagen filtros con bases haar. Estos filtros pueden ser calculados eficientemente sobre la imagen integral, son selectivos en la orientación espacial y frecuencia, y permiten ser modificados en escala y orientación. En la Figura 3.4 se muestran algunos de los filtros usados para la extracción de características.

De esta manera, un conjunto de características pueden ser usadas para codificar los contrastes encontrados en los rostros.

La propuesta original considera 3 tipos características: (Figura 3.5)

- Caraterística de dos rectángulos: Su valor se calcula con la diferencia entre la suma de los píxeles dentro de las dos regiones rectangulares.
- Característica de tres rectángulos: Su valor se calcula con la suma de los píxeles dentro de las dos regiones rectangulares exteriores y la sustracción de los píxeles de la región interior
- Característica de cuatro rectángulos: Su valor se calcula con la diferencia entre las diagonales de los pares de rectángulos.
- Rectángulos en negro representan zonas con una contribución positiva al filtro.
- Rectángulos en blanco representan zonas con una contribución negativa al filtro

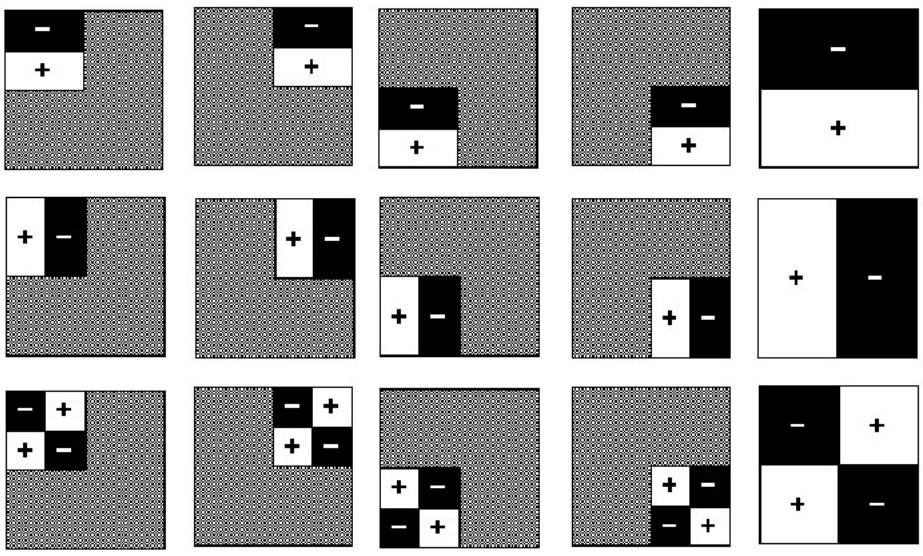


Figura 3.4: Filtros Haar rotados, trasladados y con cambios de escala

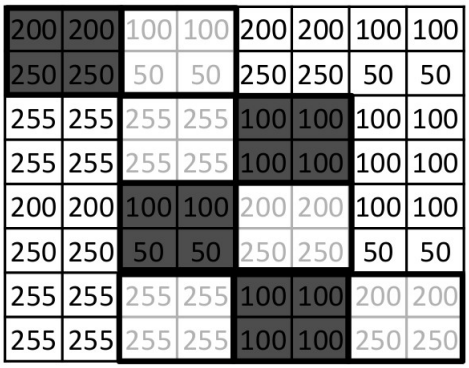



Figura 3.5: Representación de filtros haar

#### 3.1.2.1. Aplicación de filtros haar

- Cada filtro se aplica a todas las posibles escalas en horizontal y vertical.
- Cada escala se aplica en todas las posibles posiciones de la imagen.
- El resultado de aplicar cada filtro en cada escala y posición es una característica de haar.
- En imágenes de 24x24 píxeles se obtienen 162336 características.

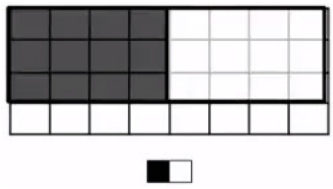


Figura 3.6: Aplicación de filtros Haar

### 3.1.3. Clasificación

Ésta etapa dentro del algoritmo de detección se encarga de asignar un conjunto de características dado a una clase con la que se encuentra una mayor similitud, de acuerdo a un modelo inducido durante el entrenamiento [14]. En la figura 3.7 se muestra un esquema de un clasificador en cascada.

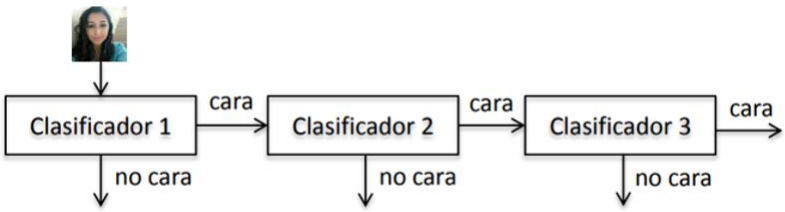



Figura 3.7: Cascada de clasificadores

#### 3.1.3.1. Adaboost

Boosting hace referencia a un tipo de algoritmo cuya finalidad es encontrar una hipótesis fuerte utilizando hipótesis débiles. Adaboost es un diseño mejorado del boosting original, es una contracción de “Adaptive Boosting”, en donde el término adaptive hace alusión a su principal

diferencia con su predecesor. En términos de funcionalidad son iguales ya que buscan crear un clasificador fuerte cuya base sea la combinación lineal de clasificadores débiles y simples. Sin embargo adaboost propone entrenar una serie de clasificadores débiles de manera iterativa de modo que, cada nuevo clasificador "weak learner" se enfoque en los datos que fueron erróneamente clasificados [15].

Para aplicar la técnica de Adaboost primero se debe establecer un algoritmo de aprendizaje sencillo (clasificador débil), que será llamado varias veces para crear diversos clasificadores débiles, para el entrenamiendo de dichos clasificadores se emplea, en cada iteración un subconjunto diferente de muestras de entrenamiento, y una distribución diferente de pesos sobre las muestras de entrenamiento [15]. A todos los datos se les asigna inicialmente el mismo peso, este peso se va actualizando con cada iteración según los ejemplos mal clasificados. Por último, se realiza la suma de todos los clasificadores débiles generando un único clasificador que se espera sea más preciso que los clasificadores débiles por separado. La figura 3.8 muestra un ejemplo del procedimiendo de Adaboost.

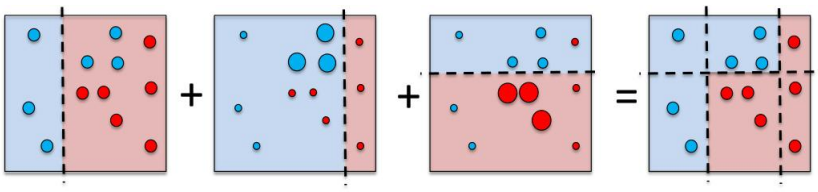


Figura 3.8: Procedimiento adaboost

El tipo de adaboost que se trata en esta investigación corresponde a un árbol de desición binario como se muestra en la figura 3.9; lo que significa que sólo trabaja con dos tipos de datos, los cuales se representan con +1 y -1, por lo tanto el resultado se expresa como *"pertenece a x clase/no pertenese a x clase"*. Cada clasificador débil depende de una sola carácteristica de haar.

Los parámetros que determinan el clasificador son dos:

- El umbral.
- El signo de la decisión.

Para fijarlos se analiza de forma exhaustiva todos los valores posibles del umbral para seleccionar el que tenga un error de clasificación mínimo.

En la figura 3.10 se tiene el conjunto de muestras que se desea clasificar y, como se ha comentado antes como resultado de los pasos previos de clasificación cada muestra va ha tener

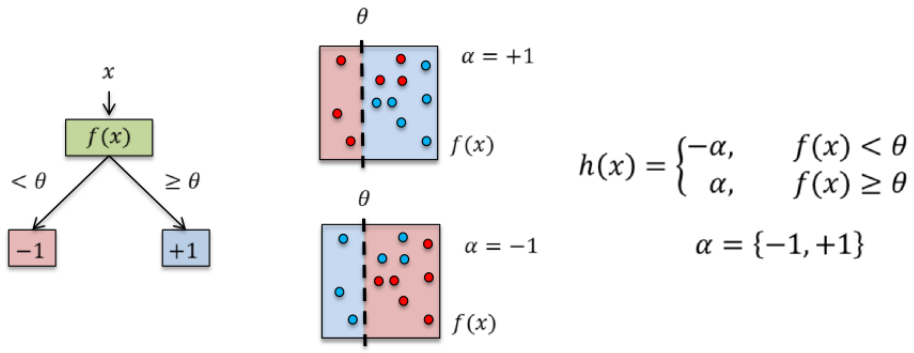



Figura 3.9: Arbol de decisión binario

asociado un peso que va a ser mayor en aquellos ejemplos que hayan sido clasificados erróneamente.

Para poder encontrar el umbral óptimo de una forma eficiente se debe ordenar las muestras de entrenamiento en función del valor de la caraterística haar que se está utilizando. El error de clasificación se calcula sumando los pesos relativos de los ejemplos mal clasificados y dividiendo para el total de ejemplos.

De esta manera se ha calculado un umbral para una característica de haar. Sin embargo en una imagen se tienen diferentes caracteríctcas, por lo tanto, en cada iteración del proceso de aprendizaje se entrenará un clasificador débil para cada una de las características de haar (tipo de filtro, escala y posición) y se seleccionarán los umbrales que den el menor error de clasificación como se muestra en la figura 3.11.

Así, de cada característica se obtiene un clasificador diferente, cada uno de ellos con un error de clasificación diferente. Por último como resultado de la iteración se escoge el clasificador débil asociado a la característica que tenga un error mínimo de clasificación.

El segundo concepto fundamental en el que se basa adaboost es la actualización del peso relativo que se da a cada ejemplo después de aprender cada nuevo clasificador débil

$$w_i(t+1)\begin{cases}\frac{1}{e_t}\frac{w_i(t)}{2} & h(x_i) \neq y_i \\ \frac{1}{1-e_t}\frac{w_i(t)}{2} & h(x_i) = y_i\end{cases} \tag{3.2}$$

, donde: $w_i(t)$ es es el peso relativo de los ejemplos y $w_i(t+1)$ es el peso relativo que se va actualizando.

Adaboost es un proceso iterativo y en cada iteración lo que se obtiene es un nuevo clasificador débil. En cada una de las iteraciones el peso relativo de cada ejemplo se va actualizar

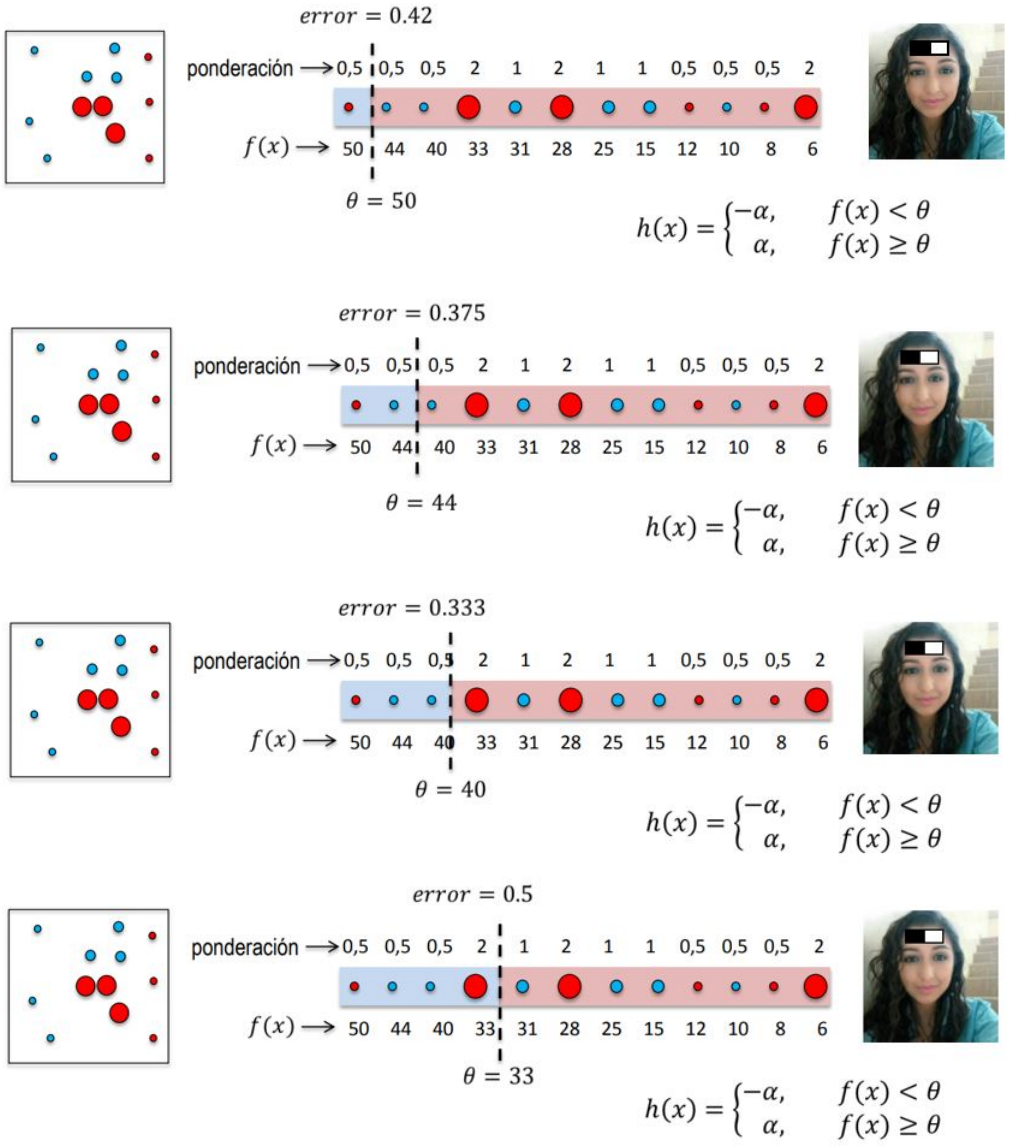



Figura 3.10: Aprendizaje

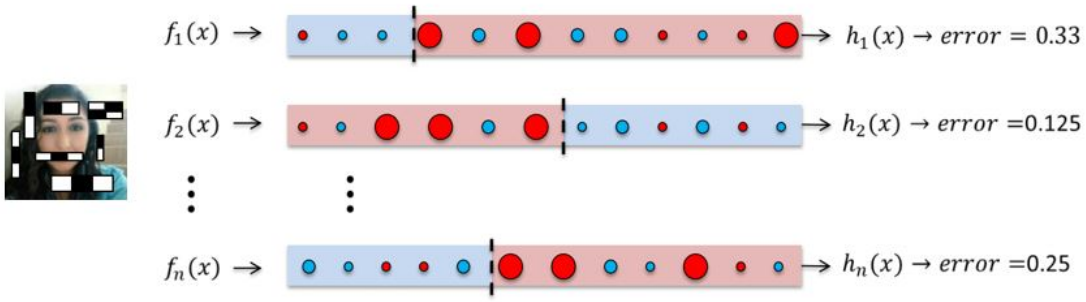



Figura 3.11: Selección de características

en función del peso relativo que tiene la iteración anterior y de un factor de actualización que dependerá del error del clasificador débil aprendido.

$$e_t < 0{,}5$$

Entonces:

- Para los ejemplos que han quedado mal clasificados $\frac{1}{2e_t} > 1$
- Para los ejemplos que han quedado bien clasificados $\frac{1}{2(1-e_t)} < 1$

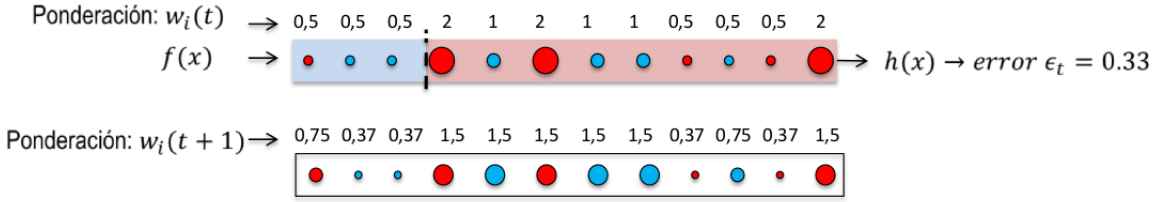



Figura 3.12: Actualización de pesos relativos

El número de iteraciones de adaboost se fija con el parámetro $T$, en cada iteración se aprende un clasificador débil en el que se selecciona la mejor característica de haar y el umbral óptimo en función del peso relativo de los ejemplos de la iteración. Como resultado final se obtiene un clasificador fuerte (figura 3.13) que se define como la suma de todos los clasificadores débiles que se han aprendido al final del número total de iteraciones.

$$H(x) = sign(\sum_{i=1}^{T} \alpha_i h_i(x)) \tag{3.3}$$

,

$$\alpha_t = \frac{1}{2}\log\frac{1-e_t}{e_t}$$

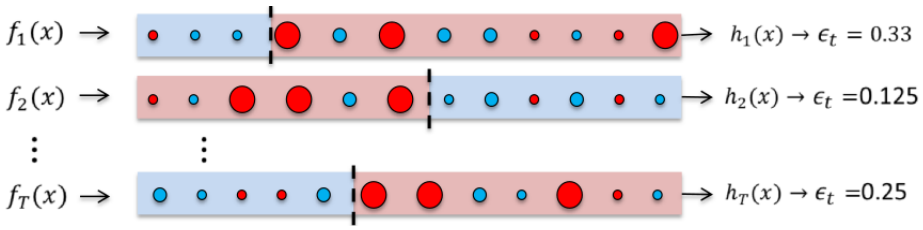



Figura 3.13: Clasificador fuerte

A continuación se muestra el algoritmo usado en la metodología. Este se encuentra planteado en [15].

- Dado un conjunto de imágenes $(x_i, y_i), ...(x_n, y_n)$ donde $y_i = 0, 1$ para muestras negativas y positivas respectivamente.
- Inicializar los pesos $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para $y_i = 0, 1$, donde $m$ es el número de muestras negativas y $l$ es el número de muestras positivas.
- Para $t = 1, ..., T$ :
  1. Normalizar los pesos $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$
  2. Seleccionar el mejor clasificador base respecto al peso del error: $e_t = \min_{f,p,\theta} \sum_i w_i |h(x_i, f, p, \theta) -$
  3. Definir $h_t(x) = h(x, f_t, p_t, \theta_t)$ son usadas para minimizar $e_t$.
  4. Actualizar los pesos: $w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$ donde $e_i = 0$ si la muestra $x_i$ es clasificada correctamente, o $e_i = 1$ en otro caso, con $\beta_t = \frac{e_t}{1-e_t}$.
  5. El clasificador robusto final es:

$$C(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2}\sum_{t=1}^{T} \alpha_t \\ 0 & en \quad otro \quad caso \end{cases} \tag{3.4}$$

#### 3.1.3.2. Cascada de clasificadores

El número de caras que se pueden encontrar en una imagen es muy limitado, sin embargo el método se analizará utilizando ventanas deslizantes por la imagen, por lo tanto se obtendrán un número considerable de hipótesis como resultado de desplazar la ventana por toda la imagen a múltiples escalas. La mayoría de las ventanas que se generan no corresponden a una cara, sólo una de las ventanas corresponde a una (figura 3.14).

Figura 3.14: Ventana deslizante

Entonces, como el objetivo es descartar el mayor número de ventanas que no contienen caras con el mínimo coste computacional posible para poder concentrar un mayor esfuerzo computacional en aquellas ventanas que tienen mayor probabilidad de ser una cara. La cascada de clasificadores (figura 3.15) permite alcanzar este objetivo mediante una combinación secuencial de clasificadores, de forma que una imagen solo será detectada como cara si realmente es reconocida de forma correcta como cara por todos los clasificadores de la cascada. Sí uno de los clasificadores rechaza la imagen como cara ésta será eliminada definitivamente.

Así, el primer clasificador tendrá como entrada todas las posibles ventanas de una imagen, todas las imágenes que rechace el primer clasificador quedarán descartadas y las que sean aceptadas serán la entrada del segundo clasificador, este proceso se repetirá hasta llegar al último clasificador. Sólo las imágenes que sean reconocidas como cara por este último clasificador y, por lo tanto, que fueron reconocidas como cara por los anteriores van a ser la detección final de caras que va a producir el detector (figura 3.15).

De esta forma, la reducción en el tiempo de cálculo se obtiene al combinar dos factores:

1. El número de imágenes que se va procesando a medida de que se pasa por la cascada se va reduciendo .

2. El número de características que se utilizan en los primeros clasificadores van a ser muy bajos, con lo que se pueden descartar muchas imágenes de forma rápida.

#### 3.1.3.3. Aprendizaje de un nivel de la cascada

Como se entrene al primer clasificador para conseguir rechazar un buen número de imágenes negativas mientras se aceptan todas o casi todas las imágenes positivas será la clave del funcionamiento de la cascada. Cada nivel de la cascada será un clasificador fuerte entrenado con adaboost el cual tiene como objetivo minimizar el error de clasificación global.

El objetivo de la cascada de clasificadores es cumplir con un número máximo de falsos positivos(falsas detecciones de caras) y falsos negativos(caras no detectadas).

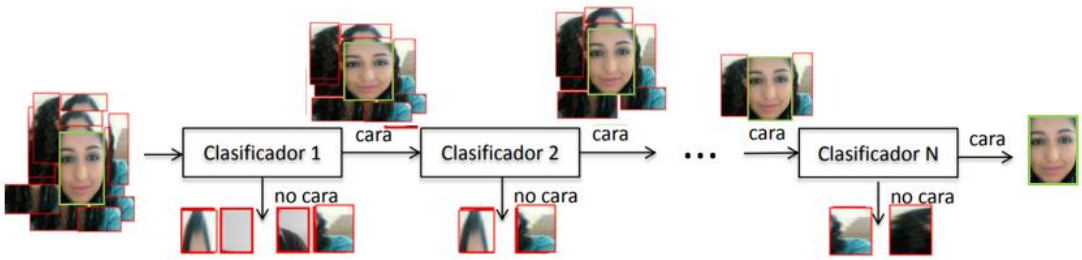



Figura 3.15: Cascada de clasificadores

Para cada clasificador se fija un número máximo respecto a falsos positivos y falsos negativos que deben producir y que se van a evaluar sobre un conjunto de validación. Sí en un clasificador fuerte el umbral de clasificación produce un valor de falsos positivos y falsos negativos, si el valor es mayor que el objetivo planteado, la solución será mover el umbral de decisión a la izquierda o derecha según convenga, para ello se introduce de forma iterativa un factor de margen $s$

$$H(x) = sign(\sum_{i=1}^{T} \alpha_i h_i(x) + s) \tag{3.5}$$

Entonces:

- Sí el número de $falsospositivos > objetivo$ entonces $s < 0$.
- Sí el número de $falsosnegativos > objetivo$ entonces $s > 0$.

De todas formas se puede encontrar la situación en que no es posible encontrar un único valor $s$ que cumpla con todos los objetivos propuestos de falsos positivos y de falsos negativos. En este caso se reentrena un nuevo clasificador fuerte añadiendo mas características de haar al clasificador.

$$H(x) = sign(\sum_{i=1}^{T'} \alpha_i h_i(x) + s) \tag{3.6}$$

, donde: $T' > T$.
Sin embargo puede pasar que por mucho que se incremente el número de características el objetivo para ese nivel de la cascada sea demasiado restrictivo y no se pueda encontrar un clasificador que pueda cumplirlo. Por lo tanto se fijará también un número máximo de características por clasificador. En la figura 3.16 se muestra el aprendizaje global de la cascada de clasificadores.

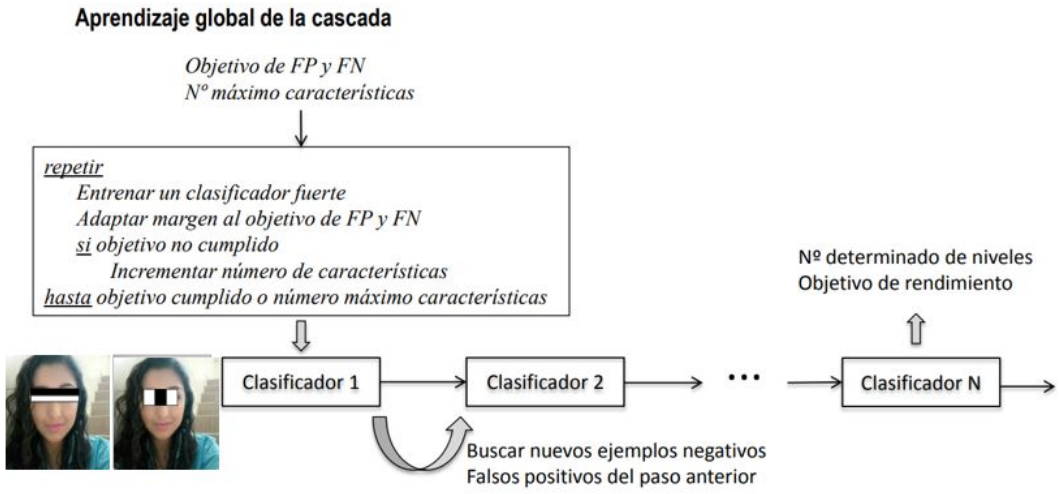



Figura 3.16: Aprendizaje global de la cascada

# Capítulo 4

# Patrones binarios locales (Local binary patterns, LBP)

Los patrones binarios locales son un operador de textura simple pero muy eficiente que etiqueta los píxeles de una imagen por vecindad de umbral de cada pixel con el valor del pixel central, y considera el resultado como un número binario. Debido a su poder de discriminación y la simplicidad de cálculo, este operador de textura se ha convertido en un método popular que se usa en varios tipos de aplicaciones. Puede ser visto como un enfoque unificador de los modelos tradicionalmente divergentes del análisis de texturas: los estadísticos versus los estructurales. Quizá la propiedad más importante del operador para aplicaciones del mundo real es su robustez frente a cambios, en una escala de grises monótona causada por las variaciones de iluminación y frente a rotaciones, en el caso de utilizar códigos circulares. Otra característica importante es su simplicidad computacional, lo que nos permite analizar las imágenes en tiempo real [16].

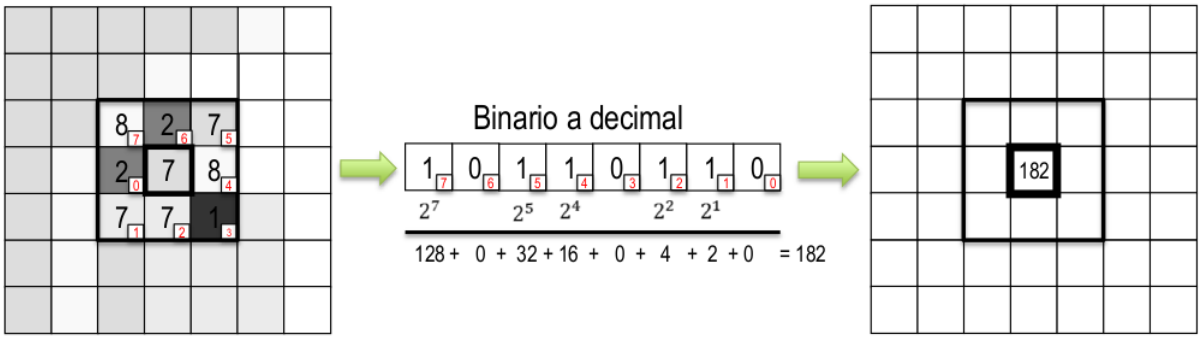
## 4.1. Cálculo de código LBP

En una imagen se tienen diferentes píxeles, se considera el cálculo del código LBP para el pixel cental, el cual se va a basar en definir una vecindad del pixel y luego ir comparando los niveles de gris del pixel central con sus vecinos. Como vecindad se definen todos aquellos píxeles que tocan al pixel central elegido. Para cada pixel vecino se aplica la siguiente regla.

$$s(k) = \begin{cases} 1 & si \quad k \geq 0 \\ 0 & si \quad k < 0 \end{cases} \tag{4.1}$$

Para ello se escoge un orden, el cual es arbitrario, pero se debe respetar el mismo para todos los píxeles de la imagen. La figura 4.1 muestra un ejemplo de cálculo de valores LBP para una imagen.

Figura 4.1: C'alculo LBP




Se ha pasado de un valor de la imagen original que va de 0-255 a un valor de imagen LBP que también va de 0-255, la diferencia es que este valor es directamente el que captura la cámara que se ha utilizado y, en este valor se codifican más cosas como el nivel de gris que había en ese pixel de la imagen original con los niveles de gris de píxeles vecinos. Por tanto la información es de alto nivel, es decir no es sólo la información del pixel si no también de como se relaciona con los píxeles vecinos.

## 4.2. Descriptor de textura

Los LBP se usaron principalmente para definir texturas como se muestra en la Figura 4.2.

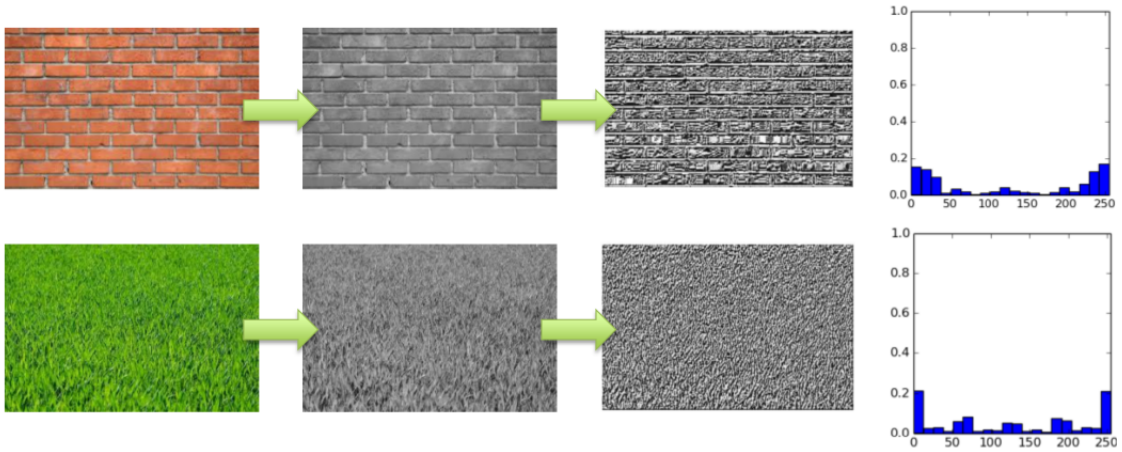


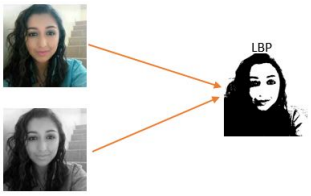
Figura 4.2: Descriptor de textura

Las texturas mostrada anteriormente son imágenes a color (RGB), pero el LBP se define para imágenes en escala de grises, para ello se transforma la imagen RGB en su correspondiente imagen de intensidad que es a la que se aplica el LBP.

Para describir las imágenes no se utiliza el array de códigos, lo que se hace es un histograma normalizado de las imágenes de códigos LBP.

## 4.3. Invarianzas

- Cambios monitónicos de nivel de gris.




- Traslación.

## 4.4. Extensión de LBP

Con el fin de tratar texturas a diferentes escalas, el operador LBP se amplió para hacer uso de barrios de diferentes tamaños. Utilizando vecindarios circulares e interpolación bilineal de los valores de píxeles, se puede manejar cualquier radio y número de muestras en el vecindario. Por lo tanto, se define la siguiente notación $(P,R)$ lo que significa $P$ puntos de muestreo en un círculo de radio $R$. La Figura 4.3 muestra algunos ejemplos de diferentes puntos de muestreo y radio

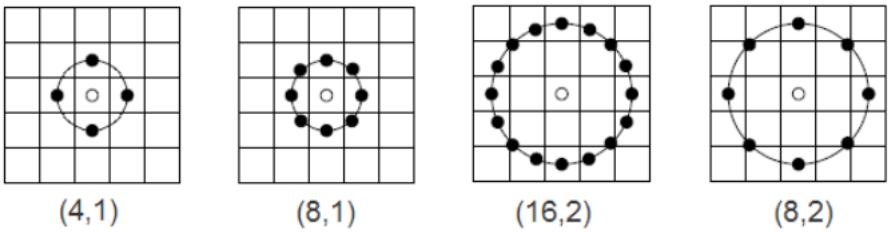



Figura 4.3: Diferentes puntos de muestreo y ejemplos de radio.

En el caso LBP (4,1), la razón por la que los cuatro puntos seleccionados corresponden a los verticales y horizontales, es que las caras contienen más bordes horizontales y verticales que diagonales [17]. Las propiedades más importantes de las características LBP son su tolerancia contra los cambios de iluminación monótona y su simplicidad computacional.

## 4.5. Patrones binarios locales aplicados a la detección de rostros

En el enfoque LBP para la clasificación de texturas, las ocurrencias de los códigos LBP en una imagen se recogen en un histograma. Estos métodos basados en características locales son más robustos contra las variaciones en la pose o la iluminación que los métodos holísticos. La metodología básica para la descripción de rosotros basada en LBP es la siguiente: La imagen facial se divide en regiones locales y los descriptores de textura LBP se extraen de cada región de forma independiente. Los descriptores se concatenan para formar una descripción global de la cara, como se muestra en la Figura 4.4 [16].

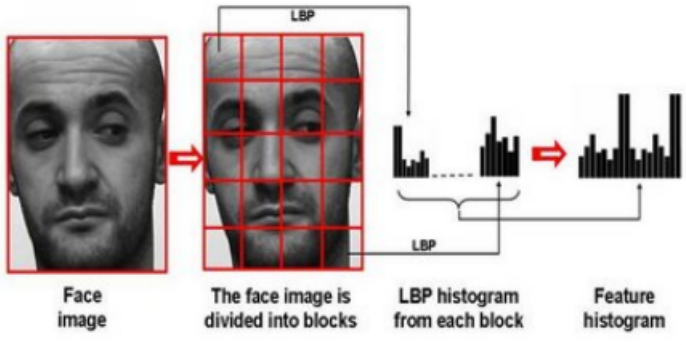



Figura 4.4: Descripción de la cara con el operador LBP.

Los histogramas LBP extraídos de cada subregión se concatenan a continuación en un único histograma de características espacialmente mejorado definido como:

$$H_{i,j} = \sum_{x,y} I(f_l(x,y) = i) I((x,y) \varepsilon R_J) \tag{4.2}$$

donde $i = 0,...,L-1,\ j = 0,...,M-1$. El histograma de características extraídas describe la textura local y la forma global de las imágenes de la cara.

Al utilizar métodos basados en histogramas las regiones no tienen que ser rectangulares, tener el mismo tamaño ni ser de la misma forma, tampoco debe cubrir toda la imagen y es posible

que diferentes regiones se puedan superponer parcialmente.

## 4.6. Aprendizaje

En este sistema, una variante de adaboost, adaboost suave se utiliza para seleccionar las características y entrenar al clasificador. Las garantías formales proporcionadas por el procedimiento de aprendizaje adaboost son bastante fuertes [10]. A continuación se detalla el algoritmo.

1. Empezar por los pesos $w_i = 1/Ni = 1,2,...,N, F(x) = 0$.
2. Repetir para $m = 1,2,...M$:
   - Estimar $f_m(x)$ ponderando un ajuste de $y$ hasta $x$.
   - Actualizar $F(x) \leftarrow F(x) + f_m(x)$.
   - Actualizar $w_i \leftarrow w_i e^{-y_i - f_m(x_i)}$

El clasificador débil está diseñado para seleccionar el único recuadro del histograma LBP que mejor separa los ejemplos positivos y negativos. Similar a [9], un clasificador débil $h_j(x)$ consiste en una característica $f_j$ que corresponde a cada compartimiento del histograma LBP, un umbral $\theta_j$ y una paridad $p_j$ que indica la dirección del signo de desigualdad:

$$h_j(x) = \begin{cases} 1 & si \quad p_j f_j(x) \leq p_j \theta_j \\ 0 & de \quad otra \quad manera \end{cases} \tag{4.3}$$

Los clasificadores débiles encontrados se utilizan para componer un clasificador fuerte.

### 4.6.1. Cascada de clasificadores

Se utiliza una cascada de clasificadores, que logra un mayor rendimiento de detección mientras que reduce la cantidad de cálculo. Los clasificadores más simples se utilizan para rechazar la mayoría de las subventanas antes de que se utilicen clasificadores más complejos para lograr bajas tasas de falsas detecciones.

Las etapas en la cascada se construyen mediante la formación de clasificadores utilizando adaboost suave. Un resultado positivo de un clasificador fuerte anterior activa el próximo clasificador fuerte que también se ha ajustado para alcanzar una tasa de detección más alta que la anterior. Un resultado negativo se rechaza inmediatamente en cualquier etapa de la cascada (figura 4.5).

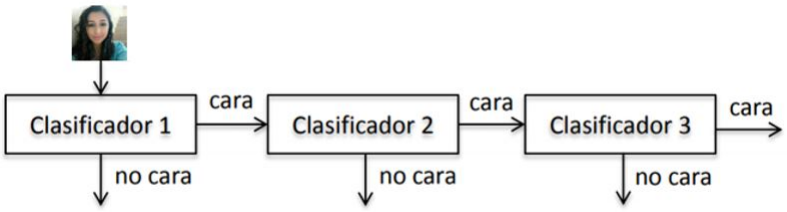



Figura 4.5: Cascada de Clasificadores

# Capítulo 5

# Metodología

La detección de rostros es el primer paso para todos los métodos de análisis facial como reconocimiento facial, alineación de rostros, verificación facial, seguimiento, etc. El objetivo de la detección de rostros es determinar si hay o no caras en la imagen y, si están presentes devolver la ubicación de la imagen y la extensión de cada cara [6]. Varios algoritmos se proponen para esta tarea. El propósito de este trabajo es evaluar la calidad de los algoritmos de detección facial y metodología de comparación. Un experimento correcto debe consistir en dos partes: aprendizaje de algoritmos en el conjunto de entrenamiento y pruebas comparativas [4]. Sin embargo esta investigación se centra en evaluar los sistemas de detección de rostros en lugar de los métodos de aprendizaje. Los algoritmos utlizados son: Viola-Jones y Patrones binarios locales.

## 5.1. Instalación openCV

### 5.1.1. Actualización del sistema

Los siguientes son los comandos utilizados para la actualización del sistema *Ubuntu 14.04*.

1. sudo apt-get autoremove libopencv-dev python-opencv
2. sudo apt-get update
3. sudo apt-get upgrade
4. sudo apt-get dist-upgrade
5. sudo apt-get autoremove

### 5.1.2. Instalación de dependencias

La intalación de dependecias necesarias para la instación de openCV se muestra a continuación.

- sudo apt-get install build-essential cmake qt5-default libvtk6-dev zlib1g-dev libjpeg-dev libwebp-dev libpng-dev libtiff5-dev libjasper-dev libopenexr-dev libgdal-dev libdc1394-22-dev libavcodec-dev libavformat-dev libswscale-dev libtheora-dev libvorbis-dev libxvidcore-dev libx264-dev yasm libopencore-amrnb-dev libopencore-amrwb-dev libv4l-dev libxine2-dev libtbb-dev libeigen3-dev ant default-jdk doxygen

### 5.1.3. Descargar openCV

Para descargar openCV se necesitan los siguientes comandos.

1. mkdir opencv
2. cd opencv
3. wget https://github.com/Itseez/opencv/archive/3.1.0.zip
4. unzip opencv-3.0.0-alpha.zip

### 5.1.4. Instalación de openCV

La instalación de la librería se realiza con los siguientes comandos.

1. cd opencv-3.0.0-alpha
2. mkdir build
3. cd build
4. cmake $-DWITH_QT = ON - DWITH_OPENGL = ON - DFORCE_VTK = ON - DWITH_TBB = ON - DWITH_GDAL = ON - DWITH_XINE = ON - DBUILD_EXAMPLES = ON$..
5. make -j4
6. sudo make install

## 5.2. Implementación

La librería de visión por computador openCV incluye estos algoritmos pre-entrenados como archivos *XML*, los mismos que pueden ser implementados y evaluados en Python.

### 5.2.1. Diagrama de flujo

El siguiente diagrama de flujo (figura **??**) describe todos los pasos seguidos para la implementación de los algoritmos en python.

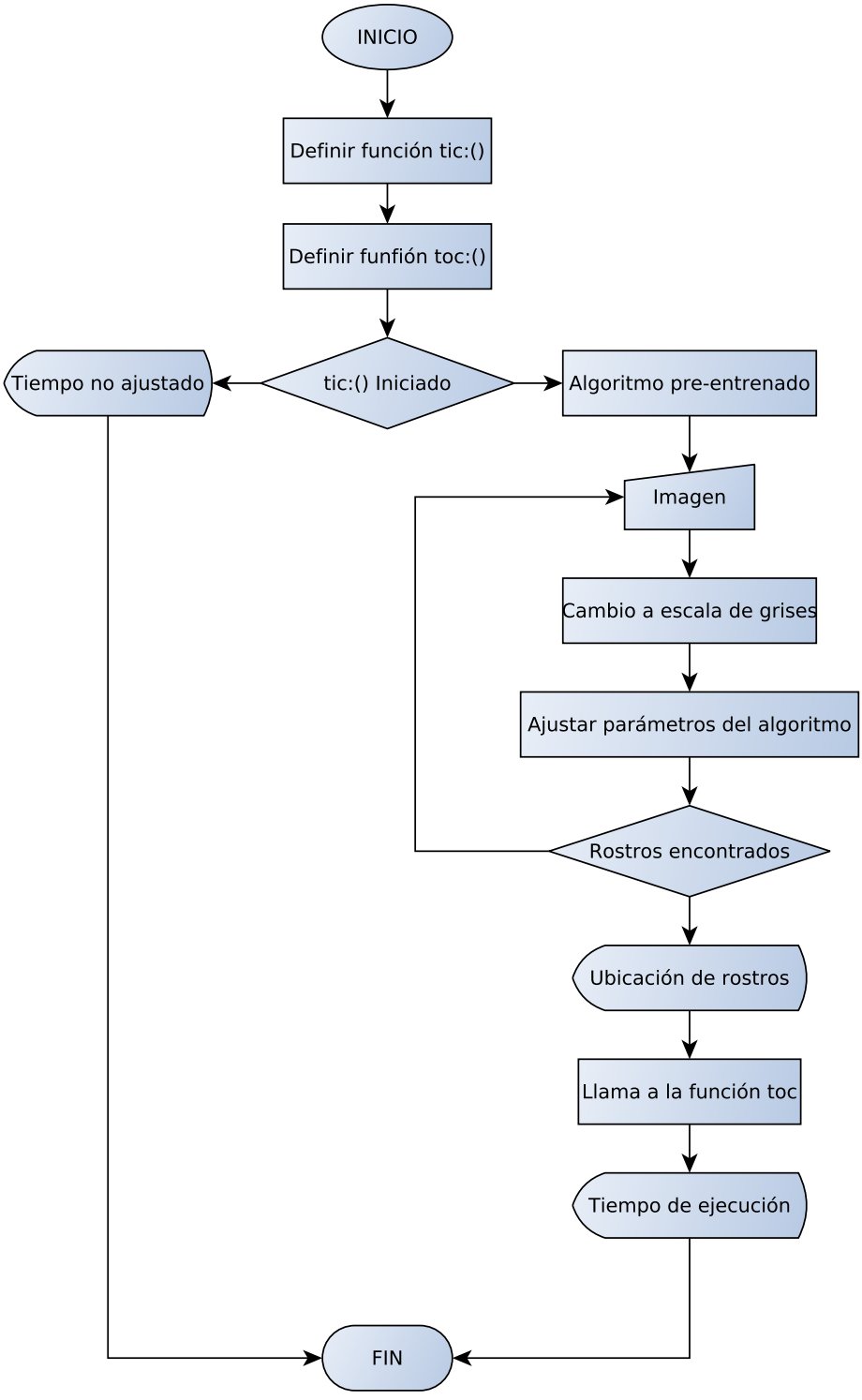



Figura 5.1: Diagrama de flujo

## 5.3. Parámetros de comparación

Los parámetros que fueron considerados para la evaluación de los algoritmos en python se denominaron: *falsos positivos* que son aquellas falsas detecciones que el algoritmo toma como rostros, *falsos negativos* a todos aquellos rostros que no son detectados y el *tiempo de ejecución* el cual permite evaluar el rendimiento del los algoritmos.

## 5.4. Base de datos

Para el desarrollo de la investigación se creó una base de datos (véase Anexo B) que contiene 50 fotografías de personas ecuatorianas de diferentes edades y etnias. Las fotografías fueron adquiridas con el uso de una cámara nikon D80 de alta resolución. Las imágenes fueron normalizadas a un mismo tamaño: ancho=724 y alto=486 píxeles.

## 5.5. Análisis comparativo

La implementación de los algoritmos fue realizada en dos computadoras con diferentes procesadores para un mejor análisis. Una vez realizada la implementación de los algoritmos en python se analizó cada imagen con los dos métodos, El cuadro 5.1 muestra las características técnicas los computadores.

Cuadro 5.1: Características técnicas de los computadores

| | Procesador | Frecuencia | RAM | Disco duro |
|---|---|---|---|---|
| Computador 1 | Core i7 | 2.4 GHz | 8 GB | 1 TB |
| Computador 2 | VIA C7-M | 1.2 GHz | 1 GB | 120 GB |

## 5.6. Resultados

En el cuadro 5.2 se pueden observar los resultados obtenidos de la evaluación realizada a los dos algoritmos sobre un computador con características de última generación. Lo que de un total de 50 imágenes de la base de datos utilizada en esta investigación con un número de rostros de 138 en total, el algoritmo de Viola-Jones tiene un mayor porcentaje de detecciones que el algoritmo de patrones binarios locales, en lo que a falsos positivos y falsos negativos se refiere, el algoritmo de Viola-Jones tiene mayor detección de no caras y menos rostros no detectados que el algoritmo lbp. El cuadro 5.3 presenta los resultados del análisis realizo sobre un computador de bajas características de procesamiento, los resultados fueron exactamente iguales, la diferencia perciste en el tiempo de ejecución como se muestra en el cuadro 5.4 de cada uno de los algoritmos en los diferentes computadores. El método lbp realiza la acción de detección en menor tiempo que el de Viola-Jones.

Al obtener un resultado general del análisis de las dos técnicas se obtiene como resultado que el algoritmo de Viola-Jones es más preciso que el algoritmo lbp al momento de detectar rostros en una imagen. los resultados han sido representados en diagramas de barras en las figuras 5.2, 5.3, 5.4.

Cuadro 5.2: Computador 1

| | Computador 1 Algoritmos | |
|---|---|---|
| | Viola-Jones | LBP |
| Total de rostros | 138 | 138 |
| Rostros detectados | 79,71 | 65,94 |
| Falsos positivos | 23,91 | 17,27 |
| Falsos negativos | 26.36 | 35,51 |

Cuadro 5.3: Computador 2

| | Computador 2 Algoritmos | |
|---|---|---|
| | Viola-Jones | LBP |
| Total de rostros | 138 | 138 |
| Rostros detectados | 79,71 | 65,94 |
| Falsos positivos | 23,91 | 17,27 |
| Falsos negativos | 26.36 | 35,51 |

Cuadro 5.4: Tiempo de ejecución

| | Tiempo de Ejecución Computadores | |
|---|---|---|
| | 1 | 2 |
| Viola-Jones | 341,074s | 402,59s |
| Patrones binarios locales | 291,59s | 322,4s |

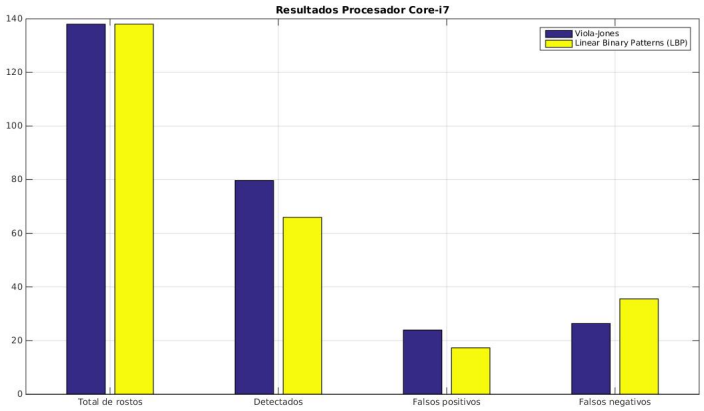



Figura 5.2: Resultados Computador 1

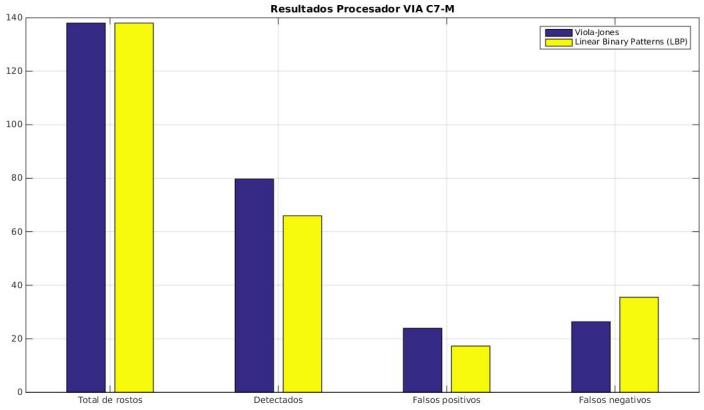



Figura 5.3: Resultados Computador 2

Al basar la selección de características en filtros haar el algoritmo de Viola-Jones presenta un mejor rendimiento al momento de detectar rostros, los histogramas utilizados en el algoritmo de patrones binarios locales al usar los vecinos más cercanos de un pixel descartan informacion útil y necesaria para la detección.

El entrenamiento utilizado en las dos aproximaciones se basan en adaboost para la selección de caracteríistivas y cascada de clasificadores para crear un clasificador fuerte que sea capaz de rechazar todas las ventanas en las cuales no exista un rostro.

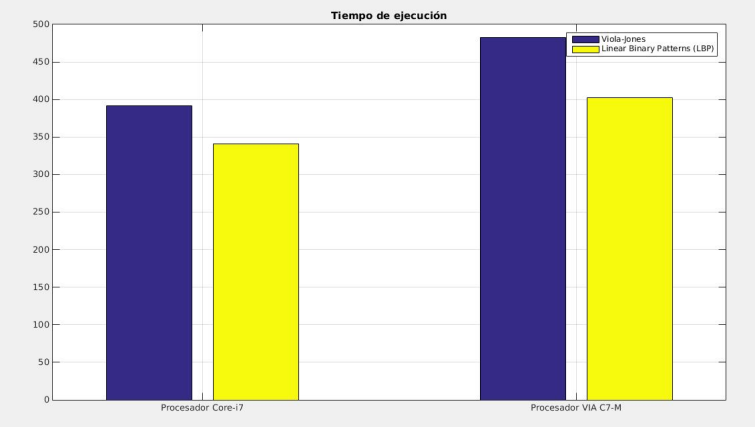



Figura 5.4: Tiempo de ejecución

# Capítulo 6

# Conclusiones, Recomendaciones y Trabajo futuro

## 6.1. Conclusiones

1. La información recopilada referente a la detección facial incluye varios métodos para poder encontrar y localizar un rostro en una imagen. Todos los algoritmos son explicados y detallados pero sólo unos cuantos de ellos son implementados y entrenados directamente en plataforma. Por lo tanto, en base a que no se encontró una gran variedad de algoritmos implementados específicamente para python se tomaron en cuenta para el desarrollo de este trabajo 2 algoritmos fundamentales en la detección de rostos que son: Viola-Jones y patrones binarios locales (LBP).

2. La literatura acerca de detección facial se refiere, en su gran mayoría al algoritmo de detección facial propuesto por Paul Viola y Michael Jones. Es por ello que, este trabajo se centró en analizar todos sus parámetros. Las características que componen el algoritmo de Viola-Jones son: Imagen integral, características haar, adaboost y cascada de clasificadores, cada una de ellas hace que este algoritmo sea más robusto y requiera de menor capacidad computacional para su procesamiento.

3. Para poder demostrar la capacidad de detección del algoritmo de Viola-Jones se sometió a comparación con otro algoritmo. Estos algoritmos se encuentran pre-entrenados en la librería openCV y fueron implementados en Python. Como resultado se obtuvo que el algoritmo de Viola-Jones es más preciso en cuando a localizar rostros se refiere que el algoritmo LBP. Cabe destacar que, en lo que se refiere a falsos positivos el algoritmo LBP arroja menor error de detección. Para un mejor resultado de evaluaron los algoritmos en diferentes procesadores pero los resultados en cuanto a detección, falsos positivos y negativos fueron los mismos. La diferencia se encuentra en el tiempo de ejecución, el algoritmo de Viola-Jones demostró ser mas eficiente en el momento de detección, y de requerir menor tiempo de ejecución.

4. Después de haber implementado los algoritmos de detección facial, las pruebas fueron realizadas y se obtubo como resultado que el algoritmo idóneo utilizado para la detección de rostros es el de Viola-Jones ya que mostró mejor capacidad de localización de caras.

## 6.2. Recomendaciones

1. Se deben obtener las imágenes con el uso de una cámara de alta resolución ya que las imágenes que se encuentran en internet son de baja calidad.
2. Para poder evaluar los algoritmos en la mismas condiciones se deben adecuar las imágenes a un mismo tamaño ya que el aumento o disminución de la resolución pueden alterar el análisis y arrojar resultados diferentes.
3. El análisis es recomendable realizarlo usando una base de datos de imágenes ya que si se usa una cámara de video en tiempo real los resultados son limitados.
4. Para un trabajo que se base en analizar el comportamiento de un algoritmo la mejor opción es usar algoritmos pre-entrenados ya que estos ahorran tiempo en lo que a implementarlos y entrenarlos se refiere.
5. En cuanto a procesadores se refiere, se recomienda el uso de procesadores de última generación, ya que la detección de rostros es el primer paso para muchas otras aplicaciones en las que es utilizada y se ahorraría tiempo de procesamiento en esta acción.
6. Los algoritmos pre-entrenados contienen tasas de fallo un poco elevadas, por lo que si se hace uso de un algoritmo de detección facial u obejtos es recomendable implementarlos y entrenarlos de acuerdo a las condiciones que se desee.

## 6.3. Trabajo futuro

En éste trabajo se ha abordado todo lo referente a detección facial. Los programas implementados para realizar dicha acción se encuentraron pre-entrenados dentro de la librería openCV. Dentro de la plataforma de python no se encontraron más algoritmos pre-entrenados, de lo que se propone como trabajo futuro lo siguiente:

- Entrenar modelos de detectores faciales para asi poder compararlos y verificar si el algoritmo de Viola-Jones sigue siendo el mejor dentro de su campo.
- Re-entrenar los algoritmos aqui estudiados de acuerdo a las necesidades que se encuentren para mejorar su desempeño.
- Análizar diferentes métodos de reconocimiento facial para de ésta manera obtener las mejores aproximaciones para que sean implementadas.

# Apéndice A

# Listado de códigos

Este apéndice incluye el software y el firmware desarrollados en el proyecto. Sólo se han anexado los archivos más importantes. Para obtener el código completo, consulte los medios adjuntos.

## A.1. Software

### A.1.1. Implementación Viola-Jones (Viola-Jones.py)

Listing A.1: Implementación Viola-Jones

```
# Importar librerias
import cv2      # libreria openCv
import sys      # libreria para proveervariables y funciones
import time     # libreria tiempo



# definir funcion tic
def tic():
    global startTime_for_tictoc
    startTime_for_tictoc = time.time()

# definir funcion toc
def toc():

    if 'startTime_for_tictoc' in globals():
        print "El tiempo de ejecucion es: " + str(time.time() - startTime_for_tictoc) + "
            segundos."
    else:
        print "Toc: tiempo de inicio no definido"

# llama a la funcion tic
tic()

#llama al archivo preentrenado
cascPath = "haarcascade_frontalface_default.xml"
```

```
#asigna a una variable con funciones de vision por computador
faceCascade = cv2.CascadeClassifier(cascPath)

#llama a la imagen que se va a analizar
image = cv2.imread('55.jpg')

#cambiar a escala de grises
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

#ajustar los parametros para la deteccion
faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(30, 30),
)

#imprimir el numero de rostros que se han encontra
print("se encontraron {0} rostros!".format(len(faces)))

#dibujar un rectangulo en la ubicacion del rostro
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

toc()
cv2.imshow("Viola-Jones", image)
cv2.waitKey(0)
```

## A.1.2. Implementación Patrones binarios locales (lbp.py)

Listing A.2: Implementación Patrones binarios locales

```
#importar librerias
import cv2
import time
import sys

#definir funfion tic
def tic():

    global startTime_for_tictoc
    startTime_for_tictoc = time.time()

#definir funcion toc
def toc():

    if 'startTime_for_tictoc' in globals():
        print "El tiempo de ejecucion es " + str(time.time() - startTime_for_tictoc) + "
            segundos."
    else:
        print "Toc: tiempo de inicio no definido"

#llamar a la funcion tic
tic()
```

```
#llamar al archivo preentrenado
cascPath = "lbpcascade_frontalface.xml"

#asignar a una variable
faceCascade = cv2.CascadeClassifier(cascPath)

#llamar a la imagen a la cual se va a analizar
image = cv2.imread('55.jpg')

#cambiar a escala de grises
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

#definir los parametros para la deteccion
faces = faceCascade.detectMultiScale(gray,1.1,5)

#imprimir el numero de rostros encontrados
print("Se encontraron {0} rostros!".format(len(faces)))

#dibujar un rectangulo en donde se encontraron rostros
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

#llamar a la funcion toc
toc()

cv2.imshow("lbp", image)
cv2.waitKey(0)
```

### A.1.3. Graficas del análisis en matlab (computador 1)

Listing A.3: Graficas del análisis en matlab (computador 1)

```
parametros={'Total de rostos','Detectados','Falsos positivos','Falsos negativos'};
y = [138 138; 79.71 65.94; 23.91 17.27; 26.36 35.51];
h = bar(y)
grid on
set(gca,'XTickLabel',parametros);
legend('Viola-Jones', 'Linear Binary Patterns (LBP)')
title('Resultados Procesador Core-i7');
```

### A.1.4. Graficas del análisis en matlab (computador 2)

Listing A.4: Graficas del análisis en matlab (computador 2)

```
parametros={'Total de rostos','Detectados','Falsos positivos','Falsos negativos'};
y = [138 138; 79.71 65.94; 23.91 17.27; 26.36 35.51];
h = bar(y)
```

```
grid on
set(gca,'XTickLabel',parametros);
legend('Viola-Jones', 'Linear Binary Patterns (LBP)')
title('Resultados Procesador VIA C7-M');
```

# Apéndice B

# Base de datos

# Apéndice C

# Resultados Viola-Jones

# Apéndice D

# Resultados Local Binary Patterns

# Bibliografía


[1] M.L. Guevara, J.D. Echeverry, and W.A. Urea, “Faces Detection in Digital Images Using Cascade Classifiers”, 2008.

[2] P. Viola, and M. Jhones, “Robust Real-Time Face Detection”, 2003.

[3] Y.-Q. Wang, “An analysis of the Viola-Jones Face detection Algorithm”,2014.

[4] N. Degtyarev, and O. Seredin, “Comparative Testing of Face Detection Algorithms”, 2001.

[5] S. Yang, A. Wiliwm, and B.C. Lovell, “To Face or Not To Face: Towards Reducing Fase Positive of Face Detection”, 2006.

[6] M.-H. Yang, D.J. Kriegman, and N. Ahuja, “Detecting Faces in Images: A Survey”, 2002.

[7] R. Makovetsky, and G. Petrosyan, “Face Detection and Tracking with Web Camera.”.

[8] J.E. Solem, “Programing computer vision with paython”, 2012.

[9] P. Viola, and M. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features”, 2001.

[10] J. Chang-yeon, “Face Detection using LBP features”, 2008.

[11] o. Jesorsky, K.J. Kirchberg, and R.W. Frischholz “Robust Face Detection Using the Hausdorff Distance”, 2001.

[12] S. Thakur, and S. Paul, “Face detection using skin tone segmentation”, 2012.

[13] A.N. Martinez-Gonzales, and V. Ayala-Ramirez, “Real Time Face Detection Using Neural Networks”, 2011.

[14] C. Papageorgiou, M. Oren, and T.Poggio “A general framework for object detection”, 1998.

[15] Y. Freud, and R.E. Schapire, “A Decision-Theoretic Generalization of On-Line Learning and an application to Boosting”, 1996.

[16] M. Pietikäinen, “Local Binary Patterns” (recuperado de: $http://www.scholarpedia.org/articleLocal-Binary-Patterns$), online.

[17] L.S. Lpez, “Local Binery Paterns applied to Face Detection an Recognition”, 2010.