UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS

DESARROLLO DE UN SOFTWARE PARA LA DETECCIÓN DE CADENAS NACIONALES MEDIANTE LA IDENTIFICACIÓN DE PATRONES DE IMÁGENES

MALPICA P GUSTAVO A
MOGOLLON G NELSON G

Bárbula, 14 de diciembre del 2015

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS

DESARROLLO DE UN SOFTWARE PARA LA DETECCIÓN DE CADENAS NACIONALES MEDIANTE LA IDENTIFICACIÓN DE PATRONES DE IMÁGENES

TRABAJO ESPECIAL DE GRADO PRESENTADO ANTE LA ILUSTRE UNIVERSIDAD DE CARABOBO PARA OPTAR AL TÍTULO DE INGENIERO DE TELECOMUNICACIONES

MALPICA P GUSTAVO A
MOGOLLON G NELSON G

Bárbula, 14 de diciembre del 2015

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS

CERTIFICADO DE APROBACIÓN

Los abajo firmantes miembros del jurado asignado para evaluar el trabajo especial de grado titulado «DESARROLLO DE UN SOFTWARE PARA LA DETECCIÓN DE CADENAS NACIONALES MEDIANTE LA IDENTIFICACIÓN DE PATRONES DE IMÁGENES», realizado por los bachilleres MALPICA P GUSTAVO A, cédula de identidad 19.523.122, MOGOLLON G NELSON G, cédula de identidad 19.653.098, hacemos constar que hemos revisado y aprobado dicho trabajo.

Firma
Prof. ING. ANTONIO FEDÓN,PHD
TUTOR

Firma
Prof. ING. FABIAN ROBLEDO
JURADO

Firma
Prof. ING. CARLOS GUTIERREZ
JURADO

Bárbula, 14 de diciembre del 2015

Dedicatoria

A mis padres, mis hermanos, amigos y familiares. En especial a Manene.

MALPICA P GUSTAVO A

Primeramente a Dios ante todo.
Luego a mis padres, familiares y amigos.
También a todos los profesores que formaron parte de mi formación.

MOGOLLON G NELSON G

Agradecimientos

A dios por darnos todo en esta vida, la capacidad para realizar los trabajos, la salud para seguir adelante y la fuerza para enfrentarlos.
A nuestros padres, por ser los pilares en nuestra formación, gracias a ellos somos lo que somos hoy en día. A ellos les debemos nuestra vida y nuestra formación.
A nuestros hermanos, por siempre escucharnos en los momentos que necesitamos hablar, por ayudarnos cuando lo necesitamos, y sencillamente por ser nuestros hermanos.
A nuestra familia, abuelos, tíos, primos. Por estar allí cuando lo necesitamos y formar parte de nuestro crecimiento como personas.
A nuestros amigos y compañeros de estudios, por todos esos buenos momentos que compartimos, formaron parte de esta gran etapa que nos tocó emprender y estamos muy agradecidos por todos los recuerdos que nos quedaran.
A nuestros profesores, por enseñarnos todos los conocimientos que manejamos hoy en día, por haber creado a profesionales con gran moral y ética de trabajo. A ellos les debemos gran parte de nuestra carrera, eso es algo que siempre debemos de agradecer.
Al profesor Antonio Fedón, por habernos guiado durante esta investigación y haber confiado en nosotros todo este tiempo.
Por último a nuestra Alma Mater, la Universidad de Carabobo, por habernos brindado los espacios para nuestra formación. Además del orgullo que representa ser ucista. ...

Índice general

Índice de figuras

Indice de tablas

Acrónimos

CONATEL	**C**omisión **N**acional de **T**elecomunicaciones
DIMETEL	**D**irección de **M**edios **E**lectrónicos y **T**elemática de la Universidad de Carabobo
DSP	**D**igital **S**ignal **P**rocessing
LOT	**L**ey **O**rgánica de **T**elecomunicaciones
OPENCV	**O**pen **S**ource **C**omputer **V**ision Library
RESORTE	**R**esposabilidad **S**ocial de **R**adio y **T**elevisión

DESARROLLO DE UN SOFTWARE PARA LA DETECCIÓN DE CADENAS NACIONALES MEDIANTE LA IDENTIFICACIÓN DE PATRONES DE IMÁGENES

por

MALPICA P GUSTAVO A y MOGOLLON G NELSON G

Presentado en el Departamento de Señales y Sistemas
de la Escuela de Ingeniería en Telecomunicaciones
el 14 de diciembre del 2015 para optar al Título de
Ingeniero de Telecomunicaciones

RESUMEN

Con este trabajo de investigación se busca corregir los errores del sistema automatizado para la transmisión de cadenas nacionales con el que cuentan actualmente las estaciones FMUC y UCTV. Para ello se pretende desarrollar un software para la detección e identificación de cadenas nacionales mediante la comparación de patrones de video. Haciendo uso de las librerías de Opencv para el procesamiento digital de imágenes, se pretende analizar las coletillas de entrada y salida presentes en una cadena nacional e identificar los patrones presentes en la misma. En caso tal de cambiar la coletilla de las cadenas nacionales; se diseñará una interfaz gráfica que le permita al operador administrar los patrones de referencia del programa y poder hacer ajustes en el programa de ser necesario.

Palabras Claves: Python, OpenCV, Visión artificial, Procesamiento digital de imágenes, Frame

Tutor: ING. ANTONIO FEDÓN,PHD
Profesor del Departamento de Señales y Sistemas
Escuela de Telecomunicaciones. Facultad de Ingeniería adscrito al Laboratorio X

Capítulo I

Introducción

1.1. Motivación

En Venezuela el cumplimiento de la Ley de Responsabilidad Social de Radio y Televisión (RESORTE) es un requisito que deben acatar los servicios de radiodifusión sonora en amplitud modulada (AM), radiodifusión sonora en frecuencia modulada (FM), radiodifusión sonora comunitaria de servicio público, televisión UHF, televisión VHF, servicios de difusión por suscripción, entre otros servicios de medios de comunicaciones.

En los casos más comunes de los medios de comunicación nacionales de radio y TV para prestar el servicio de transmisión gratuita de mensajes del Estado, se cuenta con un operador manual el cual se encarga de enlazar la señal del canal del Estado, manipulando un sistema que depende exclusivamente de la capacidad humana. Por consiguiente esto hace que se puedan producir retrasos en la conexión de la cadena y sean detectados por el ente regulador CONATEL, el cual tendrá la potestad de recurrir a sanciones por incumplimiento de la ley a todo medio prestador de servicios de radio y TV que no se encuentre difundiendo en su respectivo espectro radioeléctrico las alocuciones del Estado.

La radio Universitaria 104,5 FM (FMUC) y Televisión Universitaria (UCTV) cuentan actualmente con un circuito automatizado que enlaza la señal de la radio

con la señal transmitida por el Estado en cadena nacional, este circuito fue desarrollado en el año 2015 como tesis de grado por los autores Wilder Jiménez y Luz María Penagos, dicho circuito funciona mediante la detección y comparación de patrones de una señal de audio, el cual activará un sistema de conmutación que conectará la señal de la radio universitaria con la señal transmitida por el Estado en cadena nacional. Además cuenta con el operador de consola para la conmutación manual de la señal en caso de fallar el sistema con el que cuentan. Sin embargo, hoy en día el circuito de detección de cadenas nacionales implementado en Dimetel se encuentra generando errores en la conmutación, conectando la señal de la radio universitaria con la señal transmitida por el Estado aun cuando no se encuentren transmitiendo en cadena nacional. Por tal motivo se requiere de otro método que identifique las coletillas de entrada y salida que presentan las cadenas nacionales de radio y TV, mediante la detección de patrones de imágenes basado en el procesamiento digital de señales (DSP) y de esta manera disminuir el error introducido por el detector de cadenas nacionales implementado en Dimetel. Y así poder cumplir con el artículo 10 de la ley RESORTE y el artículo 192 de la Ley Orgánica de Telecomunicaciones los mensajes y alocuciones transmitidos por el Estado.

Para la realización del presente trabajo se consideró conveniente hacer un análisis de algunos proyectos de grado y de trabajos de investigación los cuales guardan cierta similitud con la investigación en cuestión. Los mismos fueron los siguientes:

- L.M. Penagos, W. D. Jimenez. "Diseño e implementación de un sistema de detección de patrones de audio utilizando el dispositivo Raspberry Pi modelo B, para la identificación de cadenas nacionales de radio y TV para ser integrado en la conmutación automatizada de la programación de las estaciones FMUC y UCTV". Tesis de Pregrado de la Universidad de Carabobo de Valencia. 2015. En este trabajo se realizó un sistema de detección de cadenas nacionales mediante la comparación de señales de audio, en el cual mediante la correlación y la transformada de Fourier lograron diseñar un algoritmo que reconociera el contenido espectral de la coletilla de una cadena nacional [1].

- S. Benzaquen, "Sistema de detección de caras en imágenes de video" Tesis de Pregrado de la Universidad Simón Bolívar. 2006. En este proyecto se realizó

un sistema para la detección de patrones de caras en imágenes de video, se realizó mediante las librerías de Maquinas de Vectores Soporte. Es de gran utilidad debido a la información que proporciona sobre la detección de patrones [2].

- T, Gumaa. "Automated Inspection of an Apple Moth" Tesis de Pregrado de la Universidad de Ciencias Aplicadas Hamk. 2014. En este trabajo se realizó un sistema para la detección de polillas, utilizando Python, las librerías de Opencv e implementado sobre la Raspberry Pi, es un trabajo que maneja muchas de las herramientas que se necesitaran en este proyecto de investigación y por tal motivo es de gran utilidad [3].

Por lo anterior descrito, las empresas de telecomunicaciones de radiodifusión y teledifusión, en búsqueda de la mejora y optimización de sus servicios, requieren la necesidad de reducir el error por factor humano en su máxima posibilidad ó como en el caso de la radio Universitaria reducir el error por el sistema automatizado que poseen y el factor humano en su máxima posibilidad. Este software será basado en el procesamiento digital de señales de video y en el reconocimiento de los patrones de las mismas. A su vez, el proyecto puede beneficiar a todo el sector de las empresas prestadoras de servicio de radiodifusión y teledifusión, ya que puede ser implementado para todo tipo de medio de comunicación, tal como teledifusoras UHF, VHF, radios comunitarias AM y FM, entre otras.

El planteamiento de un sistema de detección de patrones de imágenes basado en el procesamiento digital de señales, es un método que podría disminuir la problemática que presenta la radio Universitaria al momento de transmitir una cadena. Aunque esta empresa de telecomunicaciones ya cuenta con un buen sistema para la detección de cadenas nacionales a base del reconocimiento de patrones de audio, este nuevo método busca corregir los posibles errores que pueda generar el que se encuentra implementado y reducir aún más el factor humano.

Por medio de una capturadora de video [4] el programa analizará las coletillas de entrada y salida que son transmitidas por el canal del Estado al momento de entrar y salir de una cadena mediante el proceso de digitalización de la señal de

video, el programa comparará la señal de entrada con los patrones almacenados en su memoria y por medio de la correlación se verificará si existe coincidencia entre las señales.

Por medio de esta investigación se generan diversos tipos de beneficios, ya que busca consolidar al sector de los medios de comunicación ante cualquier imposición de tipo económico o social. Además, los estudios utilizados en el procesamiento digital de señales y reconocimiento de patrones de imágenes y video son de gran utilidad para otros tipos de sistemas, tales como de seguridad, medicina, tecnología, fines académicos y otros. Ya que se plantea obtener la detección de una señal de video mediante la comparación de señales utilizando la correlación. A la vez contribuye al desarrollo tecnológico en esta área de la radiodifusión y teledifusión a nivel nacional, debido a la realización de un software que monitoree y detecte cadenas nacionales de radio y TV.

1.2. OBJETIVOS

1.2.1. OBJETIVO GENERAL

Desarrollo de un software para la detección de cadenas nacionales de radio y TV mediante la identificación de patrones de imágenes.

1.2.2. OBJETIVOS ESPECIFICOS

- Identificar los parámetros de una señal de video que permitan generar criterios para el reconocimiento y detección de una cadena nacional.
- Plantear un algoritmo utilizando las librerías de Opencv, que permita identificar el contenido de la señal de video de una cadena nacional.
- Diseñar una interfaz gráfica para la administración del programa, de ser modificados los patrones de video.
- Probar la eficiencia del programa de detección de cadenas nacionales.

1.3. ALCANCE

Mediante esta investigación se procura desarrollar un software para la detección de cadenas nacionales mediante la identificación de patrones de videos, con la finalidad de mejorar el desempeño del sistema con el que cuenta actualmente FMUC y UCTV, el cual se encuentra a cargo de un sistema de reconocimiento de señales de audio y un operador de consola. Además el desarrollo de una interfaz gráfica que le permitirá al operador la administración del programa como también la modificación de los patrones de videos preestablecidos en la memoria del dispositivo para posibles modificaciones futuras.

Capítulo II

Marco conceptual

2.1. COMPARACIÓN DE PATRONES

La combinación de diferentes datos dan forma a un arreglo de patrones; estos datos pueden ser numéricos, pixeles, sonido, entre otros. Los patrones son importantes en la visión por computadora porque determinan la representación de un objeto en una imagen, si esos patrones son detectados, entonces es un indicador que ese objeto está ocurriendo en esa imagen. Lamentablemente estos patrones tienden a ser afectados por el ruido, la iluminación, la orientación, entre otros; es por ello que se tiende a extraer características de la imagen y realzarla para que sobresalga, y con ello utilizarla como patrón para la detección de estos mismos en otras imágenes [5].

2.2. CORRELACIÓN

Sabiendo que las imágenes tienen señales dependientes del dominio espacial, en donde, f(x,y) representa la función de una imagen; podemos definir la correlación

para el tratamiento de imágenes de un dominio espacial como [6]:

$$f(x,y) * h(x,y) = \frac{1}{MN} \sum_{0}^{M-1} \sum_{0}^{N-1} f^*(m,n) \quad h(x+n, y+m) \tag{2.1}$$

De donde f^* representa la conjugada de la función f. La correlación nos permite comparar dos imágenes por medio de sus histogramas, de la ecuación 2.1, Opencv realiza una correlación unidimensional la cual está definida como [7]:

$$d(H_1, H_2) = \frac{\sum_I ((H_1(I) - H_1^-)(H_2(I) - H_2^-))}{\sqrt{\sum_I (H_1(I) - H_1^-)^2 \sum_I (H_2(I) - H_2^-)^2}} \tag{2.2}$$

Donde:
$H_k^- = \frac{1}{N} \sum_J H_k(J)$
y N es el número total de píxeles del histograma.

2.3. VISIÓN POR COMPUTADORA

Es un campo que se encarga de la adquisición, procesamiento y análisis de imágenes. Por tal motivo el propósito de la visión por computadora o visión artificial, es que el computador "entienda" una escena o las características de ella, para luego aplicar una determina acción, dependiendo de cuál sea el caso.

La visión artificial la componen un conjunto de procesos destinados a realizar el análisis de imágenes. Estos procesos son: captación de imágenes, memorización de la información, procesado e interpretación de los resultados. Con la visión artificial se puede [8]:

- Automatizar tareas repetitivas de inspección realizadas por operadores.
- Automatizar tareas repetitivas de inspección realizadas por operadores.
- Identificación e inspección de objetos.
- Determinar posición de objetos en el espacio.

- Establecimiento de relaciones espaciales entre varios objetos.
- Entre otros.

2.4. OPENCV

Es una librería de visión artificial escrita en lenguaje C y C++, corre bajo las plataformas de Linux, Windows y Mac OS X. el diseño y la estructura de esta librería lo convierte en una gran aplicación para análisis de imágenes o videos en tiempo real. Esto es debido a que contienen más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos, calibración de cámaras, visión estérea y visión robótica. Opencv está publicado bajo la licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas [9].

2.5. Capturadora de Video

Por el motivo de que la mayoria de los ordenadores no poseen una entrada de video físca disponible, es necesario contar con una tarjeta de video para poder adquirir la señal de videos requerida. Para ello dicha tarjeta cuenta con un convertidor RCA a USB, este dispositivo se incorpora al ordenador mediante conexión USB y hará la conversión de analógico/digital de las tramas de video analizadas [4].

Figura 2.1: Convertidor RCA/USB.[4]

2.6. IMAGEN DIGITAL

Una imagen capturada a través de un sensor es expresada por una función continua bidimensional f(x,y). El proceso de digitalización de la imagen significa que la función f(x,y) es muestreada en un arreglo matricial de M filas por N columnas. Por otro lad la imagen digital tiene otro proceso de cuantización, el cual asigna a cada valor muestreado un valor entero dentro del arreglo matricial [10].

En la Figura se observa la imagen de un automóvil. En la imagen podemos observar que el vehículo posee un espejo retrovisor del lado del piloto. Sin embargo lo que la computadora .ºbserva.ᵉˢ solo un arreglo de números. Cualquier valor numérico en dicho arreglo posee un ruido asociado. Por lo tanto la tarea consiste en agarrar este arreglo afectado por el ruido y entenderlo para poder identificarlo como un " espejo retrovisor " [9].

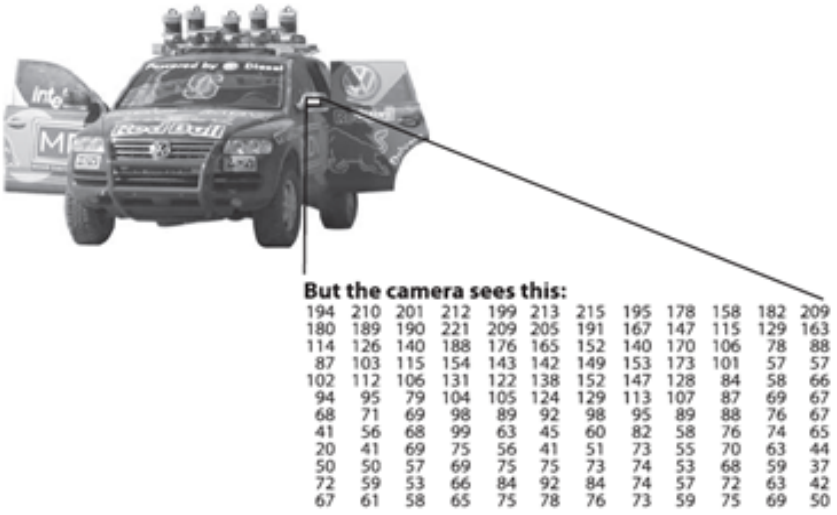

Figura 2.2: Imagen Digital.[9]

2.7. NTSC

El Comité Nacional de Estándar de Televisión (NTSC) por sus siglas en inglés, es (valga la redundancia) un estándar para la transmisión de la señal de televisión, inventado en 1941 en los Estados Unidos para la transmisión de TV a blanco y negro; para luego en los años 50 adaptar el estándar para TV a color. Este estándar es implementado en los Estados Unidos, Japón, México, Canadá y gran parte de Sur América. Dentro de las especificaciones que posee NTSC podemos destacar las siguientes: [11]

- Tiene una tasa de 29,27 cuadros por segundo (29,27frames per second).
- La imagen se forma a partir de un escaneo a una velocidad constante de 525 líneas por cuadro.
- Las dimensiones de la imagen visual de los usuarios es de 480 líneas verticales por 720 líneas horizontales.
- Para la transmisión se requiere una polarización horizontal [12].

En Venezuela de acuerdo al ente regulador Conatel los canales de televisión abierta deben transmitir de acuerdo a las estándares de una señal NTSC, por tal motivo la señal de TV que se estará procesando responde a tales especificaciones [13].

2.8. SEGMENTACIÓN

Es uno de las principales herramientas para el procesamiento de la información en imágenes. La segmentación subdivide una imagen en múltiples regiones u objetos, el cual, el nivel de segmentación o método de segmentación proviene del problema que quiere ser solucionado. Los algoritmos de segmentación generalmente se basan en uno de dos propiedades básicas: por discontinuidades o por similitud.

La primera de ellas se basa en la segmentación por cambios abruptos en la intensidad, como por ejemplo la segmentación por bordes dentro de la imagen. La segunda categoría se basa en la partición de la imagen en regiones, de acuerdo a criterios previamente definidos [6].

2.9. HISTOGRAMAS

Los histogramas son un puente entre las imágenes y un descriptor probabilístico, es decir, definiendo una función de probabilidad $p_1(z; x, y)$ en donde (x,y) representa la posición de un pixel de la imagen y posse una intensidad z; esta representación lo hacen por medio de un gráfico de barra [10], en la figura 2.3 se observa el histograma de la imagen.

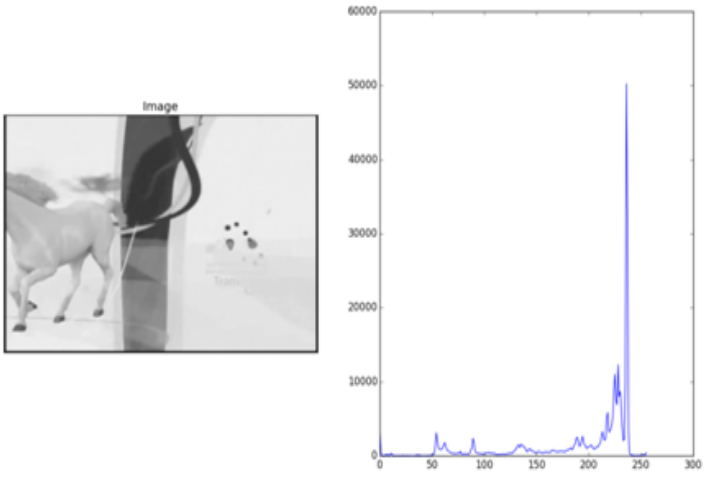

Figura 2.3: Histograma de un frame de la Cadena Nacional.

Los histogramas representan un arreglo unidimensional de L elementos, en donde, L depende de los niveles de gris (para el caso de imágenes monocromáticas), por tal motivo tomando de ejemplo la figura 2.3, la mayor parte de la función del histograma tiende a concentrarse hacia el lado derecho del arreglo, ya que en la imagen existe un predominio de pixeles con intensidades muy altas; por otro lado si tuviésemos una imagen donde las intensidades de los pixeles fuesen muy bajas,

la función que representa al histograma tendería a ubicarse hacia la parte más baja del arreglo. Los histogramas son unas de las herramientas más útiles dentro del procesamiento de imágenes, sobre todo para implementaciones en tiempo real [6].

2.10. USB-Relay

Es una tarjeta impresa con un relay controlado por comunicación USB, dependiendo de la configuración que se le asigne a activará el relay realizando la conmutación. Es alimentado por medio del puerto USB, por tal motivo no se requiere ninguna otra fuente adicional. El relay está impreso en una tarjeta y funciona con 5V DC y hasta 2A para la conmutación. La comunicación se ejecuta mediante un PIC18F14K50, el cual contiene los comandos y funciones que genera los cambios de estado del relay [14]. Este dispositivo se utilizará para poder realizar una conmutación física cuando el programa detecte la cadena nacional, podemos observar una imagen del mismo en la figura 2.4.

Figura 2.4: USB-Relay.

Capítulo III

Procedimientos de la investigación

En el presente trabajo se planteó el desarrollo de un software para la detección de cadenas de radio y TV mediante el procesamiento de imágenes. Por lo cual se desarrollaron los objetivos específicos en fases de la siguiente manera:

3.1. Fase I: DOCUMENTACIÓN ACERCA DEL PROCESAMIENTO DE IMÁGENES Y VIDEOS

Esta sección fue una etapa de recopilación de bibliografía como libros de texto, publicaciones, entre otras. Con la finalidad de conocer las características que hay dentro de una imagen y poder identificar dichos elementos dentro de la cadena nacional y así poder tomarlos como criterio para la detección de la misma. Podemos destacar como parte de esta recopilación de información la realización de un curso por internet sobre detección de objetos a partir del procesamiento digital de imágenes de la Universidad Autónoma de Barcelona, a través, de la página Coursera.org.

Una imagen digital se forma a partir de un arreglo matricial, en donde, la menor unidad de dicho arreglo representa el pixel. Por tal motivo, la información que podamos extraer de estos pixeles de la imagen, es la que nos servirá de criterio para la detección de objetos o en este caso la coletilla de una cadena nacional[15].

Un esquema general para un sistema de detección de objetos en una imagen podemos apreciarlo en la figura 3.1. Este esquema muestra las etapas que debe de poseer un sistema de detección de objetos dentro de una imagen. Cómo desde una imagen podemos extraer características de ella (por ejemplo un segmento de imagen o la intensidad de los pixeles de la misma) y a partir de estas características podemos analizar la imagen y generar candidatos que se encuentren dentro de la imagen, es decir, tomar la información que suministra tales características. También se puede realizar este procedimiento de manera inversa, en donde, en primera instancia generamos candidatos dentro de la imagen, para luego extraer las características de los mismos. Todo con la finalidad de poder llegar a una última etapa de decisión sobre la información que poseemos de la imagen.

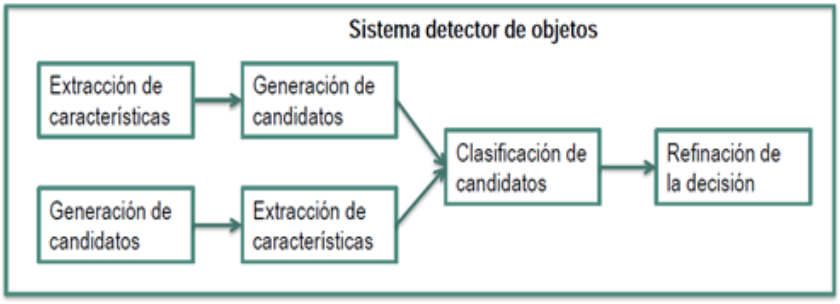

Figura 3.1: Esquema General de Detección de Objetos.

Teniendo en cuenta que al pasar los años la coletilla de la cadena nacional ha sido cambiada en varias ocasiones y la intensión del software es que pueda ser utilizado para cualquier coletilla que posea la cadena nacional. Se requiere de unos descriptores amplios y básicos dentro de la detección de objetos que sirvan de criterio para la detección de la cadena nacional, para que en el supuesto caso de cambiar la coletilla de la cadena nacional se actualicen dichos descriptores y el software se mantenga operativo. Por lo tanto dentro del esquema de detección de objetos de la figura 3.1 primero extraeremos las características de la imagen, luego generaremos candidatos a partir de estas características y por ultimo un criterio de decisión.

Dentro de las herramientas y procesos para la detección de objetos, nos llamaron la atención dos de ellos, usar la segmentación para conseguir una referencia o

template [15], y partir de esta referencia llevar la imagen a escala de grises para luego calcular su histograma y compararlo mediante la correlación [9]. De esta manera se logrará cumplir con las características del software.

3.2. FASE II: PLANTEAMIENTO DEL ALGORITMO USANDO LAS LIBRERIAS DE OPENCV

Teniendo definidos las herramientas dentro del procesamiento de imágenes que se utilizará para la detección de la cadena nacional; se procedió a la documentación y manipulación de las librerías de Opencv para definir un algoritmo que detecte la señal de video deseada.

3.2.1. Descripción del algoritmo desarrollado

Para la correcta selección del entorno de trabajo y el lenguaje a utilizar, es necesario el planteamiento del algoritmo que se eligió como fundamento del proyecto. Para el desarrollo del programa encargado del cómputo de comparación de señales de video basado en la correlación, se diseñó el siguiente flujograma de manera general. En la figura **??**f9) se observa el esquema general del algoritmo de detección de las cadenas nacionales, podemos dividir dicho esquema en tres etapas:

3.2.2. Descripción Etapa de Referencia

En esta primera etapa del esquema de la figura 3.2 se genera la referencia o base de datos, la cual se usara de base como criterio para la comparación de la señal de video en tiempo real con dicha base de datos, y así poder identificar si la información que se transmite por la señal de televisión corresponde a una cadena nacional.

Esta recopilación de imagenes de refenrencia se generó a partir de la digitalización de la coletilla de la cadena nacional por medio de la capturadora de video, en

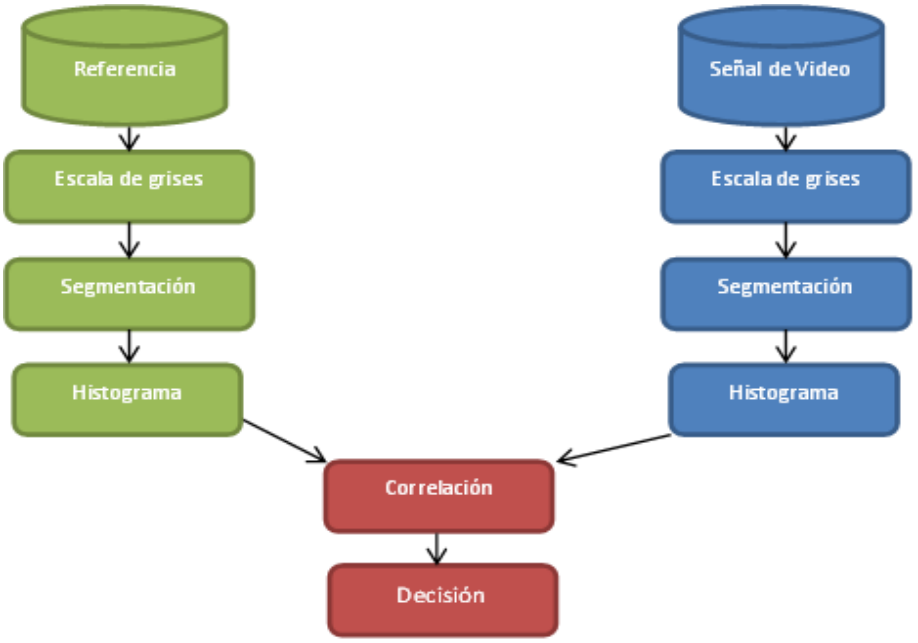

Figura 3.2: Esquema general algoritmo detección de cadenas nacionales.

donde se extrajeron 12 frames, donde cada frame de referencia se encuentra distanciado del otro por 30 frames. A partir de estos frames y mediante la segmentación se extrajeron dos patrones por frame, que servirán de referencia para la detección de la cadena nacional, en la figura 3.3 se observa uno de los cuadros y la extracción de los patrones mediante la segmentación.

La imagen (a) de la figura 3.3 vemos el frame almacenado de referencia a partir de la digitalización de la coletilla de la cadena nacional, esta imagen posee unas dimensiones de 720 pixeles de ancho por 480 pixeles de alto, de acuerdo a los estándares de la señal NTSC; mientras que la imagen (b) y (c) representan los patrones extraídos de dicho frame mediante la segmentación, los cuales tienen unas dimensiones de 360 pixeles de ancho por 120 pixeles de alto. Los 12 frames extraídos como referencia como el que se muestra imagen (a) de la figura 3.3 son almacenados como archivos de imagen .jpg para luego extraer la información de estos archivos y proceder a su comparación en tiempo real.

Una vez almacenados los 12 frames de referencia, se procede a degradar todos

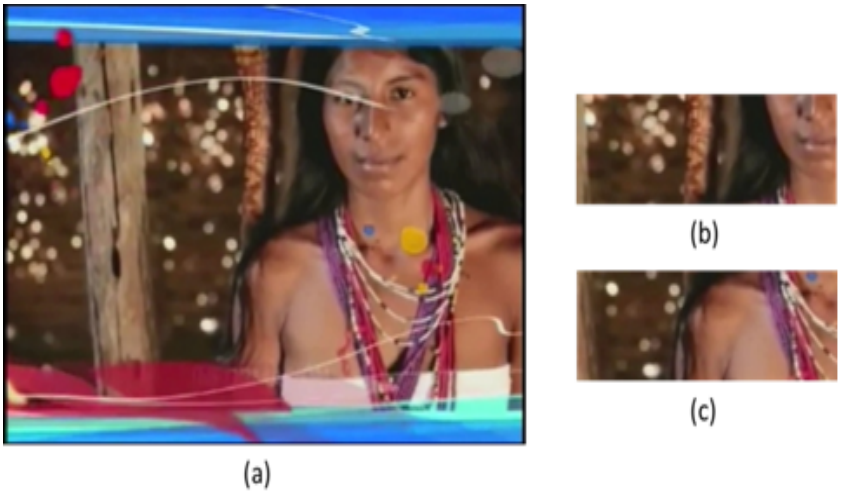

Figura 3.3: (a)Muestra un frame de referencia de la coletilla de la cadena nacional. (b)y(c) Patrones que se utilizan para la detección.

los cuadros con diferentes niveles de ruido. A cada una de las 12 imágenes se almaceno de manera adicional tres imágenes del mismo frame con diferentes tipos de ruido y diferentes intensidades en la figra 3.4 podemos observar un ejemplo de estas imágenes. Estas imágenes adicionales también forman parte de la referencia, es decir, también se le aplica el proceso de segmentación y son utilizadas para el proceso de decisión para identificar la coletilla de la cadena nacional. En vista a que los frames capturados pueden poseer diferentes niveles de ruido y usar la referencia sin ningún tipo reforzamiento en la información en ella o en la captura de los cuadros, puede ocasionar problemas y que no sea identificable la coletilla debido al mismo ruido mencionado. Esto ayuda al algoritmo a poseer una mayor información sobre lo que debe de identificar como una cadena nacional.

En vista de que la intención de este programa es que pueda identificar cualquier coletilla de la cadena nacional sin importar que esta cambie con el pasar del tiempo, fue necesario observar las presentaciones anteriores y ver que la duración de estas tenga como mínimo una duración de 12 segundos, ya que la base de datos de nuestro software almacena como se mencionó anteriormente 12 cuadros de 12 segundos consecutivos de video, al observar las presentaciones las más antigua encontrada y

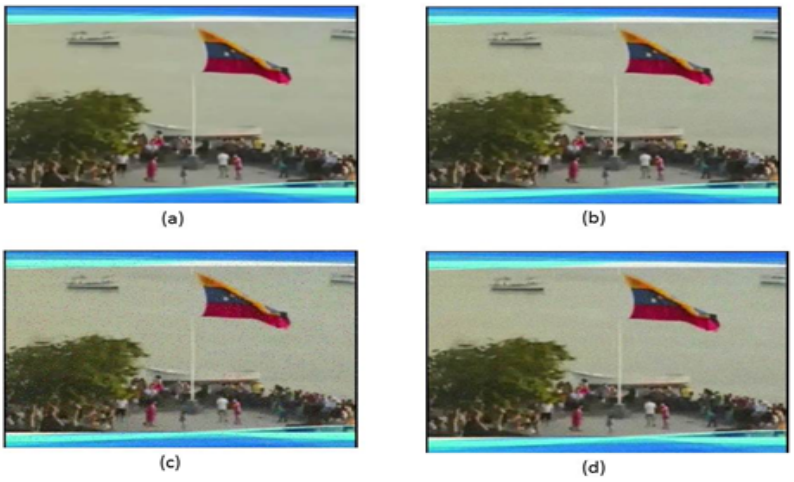

Figura 3.4: (a)Imagen Original; (b)Imagen Ruido Gaussiano al 20 %; (c)Imagen Ruido Salt and Pepper al 30 % y (d)Imagen Ruido Gaussiano al 40 %.

con menor duración data del año 1999 y posee una duración de 19 segundos, por lo tanto la duración de la base de datos no será mayor que la coletilla de la cadena nacional, garantizando de esta manera que toda la información almacenada en la base de datos corresponda a la coletilla de una cadena nacional.

Una vez obtenido las referencias el siguiente paso dentro de la etapa es pasar las mismas a escala de grises para luego poder obtener los histogramas de las mismas y poder compararlos con la señal de televisión digitalizada.

3.2.3. Descripción Etapa señal de video

En esta etapa se procesara la señal de estándar NTSC proveniente de la televisión, en donde por medio de la capturadora de video se digitalizara y se procesara como una señal con las características del estándar ya mencionado en el capítulo II; cuadro por cuadro de la señal se le aplicara el procedimiento descrito en el esquema de la figura 3.2, en donde a cada cuadro se le extrae dos referencias mediante la segmentación, para posteriormente llevarlas a un espacio de color monocromático, para calcular los histogramas de dichas referencias y compararlos por la correlación

con la base de datos e identificar las coincidencias, un ejemplo de los histogramas obtenidos por la referencia de un cuadro podemos obsérvalo en la figura 3.5.

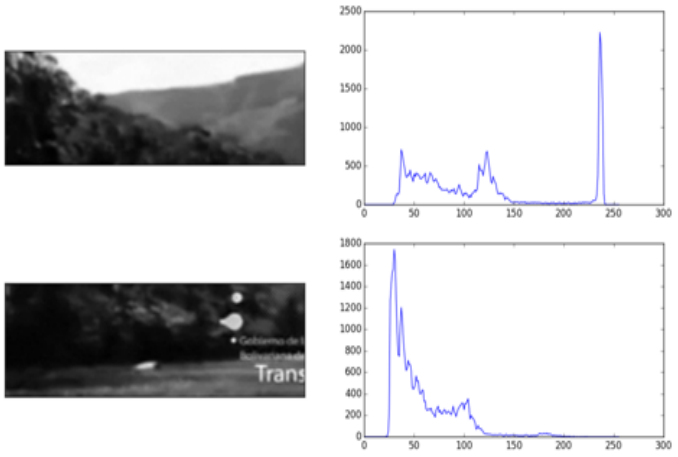

Figura 3.5: (a)Imagen Original; (b)Imagen Ruido Gaussiano al 20 %; (c)Imagen Ruido Salt and Pepper al 30 % y (d)Imagen Ruido Gaussiano al 40 %.

Teniendo los histogramas de un cuadro procesado, se procede a comparar con la base de datos y ver si existe coincidencia, de no ser así, se procede a realizar la misma operación con el siguiente cuadro procesado y así sucesivamente hasta encontrar coincidencia entre los cuadros; es procedimiento de análisis fue tomado de un método que utiliza el Open Source Mythtv, en donde, uno de los métodos que usan para detectar comerciales de televisión es el identificar un cuadro negro que se genera entre comerciales [16], en nuestro caso debido que no queremos detectar comerciales sino un tipo de video en específico, se procede a procesar cuadro por cuadro la programación y detectar coincidencia que existen entre ellos y la base de datos.

3.2.4. Descripción Etapa de Decisión

En esta última etapa planteado en el esquema de la 3.2 es el procedimiento de la toma de decisión, es decir, es la forma en como el algoritmo piensa para identificar si lo que está procesando corresponde a una cadena nacional.

Primero que nada antes de poder entrar dentro de una etapa de decisión debe de pasar por un proceso de correlación, en donde, se compararan los cuadros de la señal de televisión provenientes de la capturadora de video, con los cuadros almacenados con la base de datos; las librerías de Opencv nos ofrecen funciones para realizar el proceso de correlación de los histogramas de la base de datos con los obtenidos de la señal digitalizada [9], en la figura 3.6 podemos observar cómo se realiza el proceso de comparación con los patrones de la base de datos, con los patrones extraídos de un cuadro digitalizado.

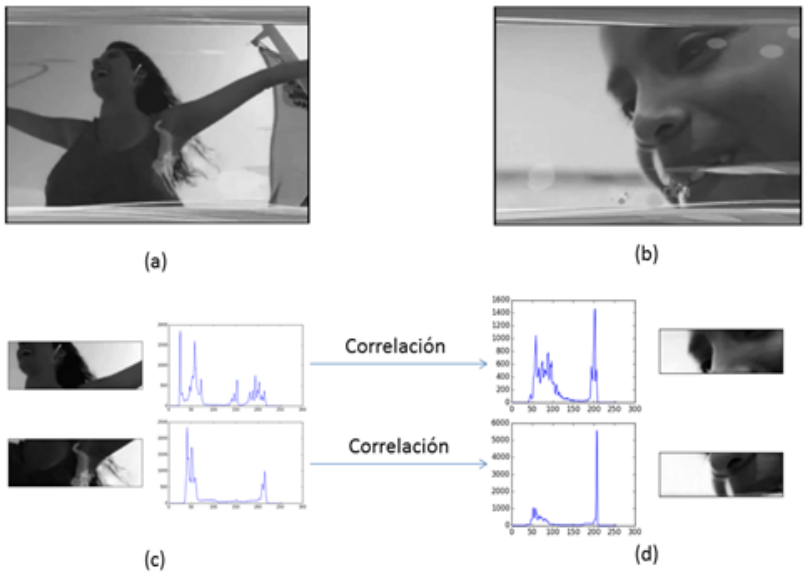

Figura 3.6: Comparación de Histogramas por patrones.

Las imágenes (a) y (c) de la figura 3.6 corresponden al cuadro y a los patrones e histogramas extraídos de ese mismo cuadro en la base de datos, mientras que las imágenes (b) y (d) son un cuadro procesado por la capturadora, al cual se le extrajeron dos patrones y los histogramas de sus respectivos patrones así como se describió en la etapa anterior. Ahora bien se procede a realizar la correlación de los cuatro histogramas y verificar si existe coincidencias entre ambos pares, de ser así se procede a entrar en una etapa de verificación que podemos observar en la figura 3.7.

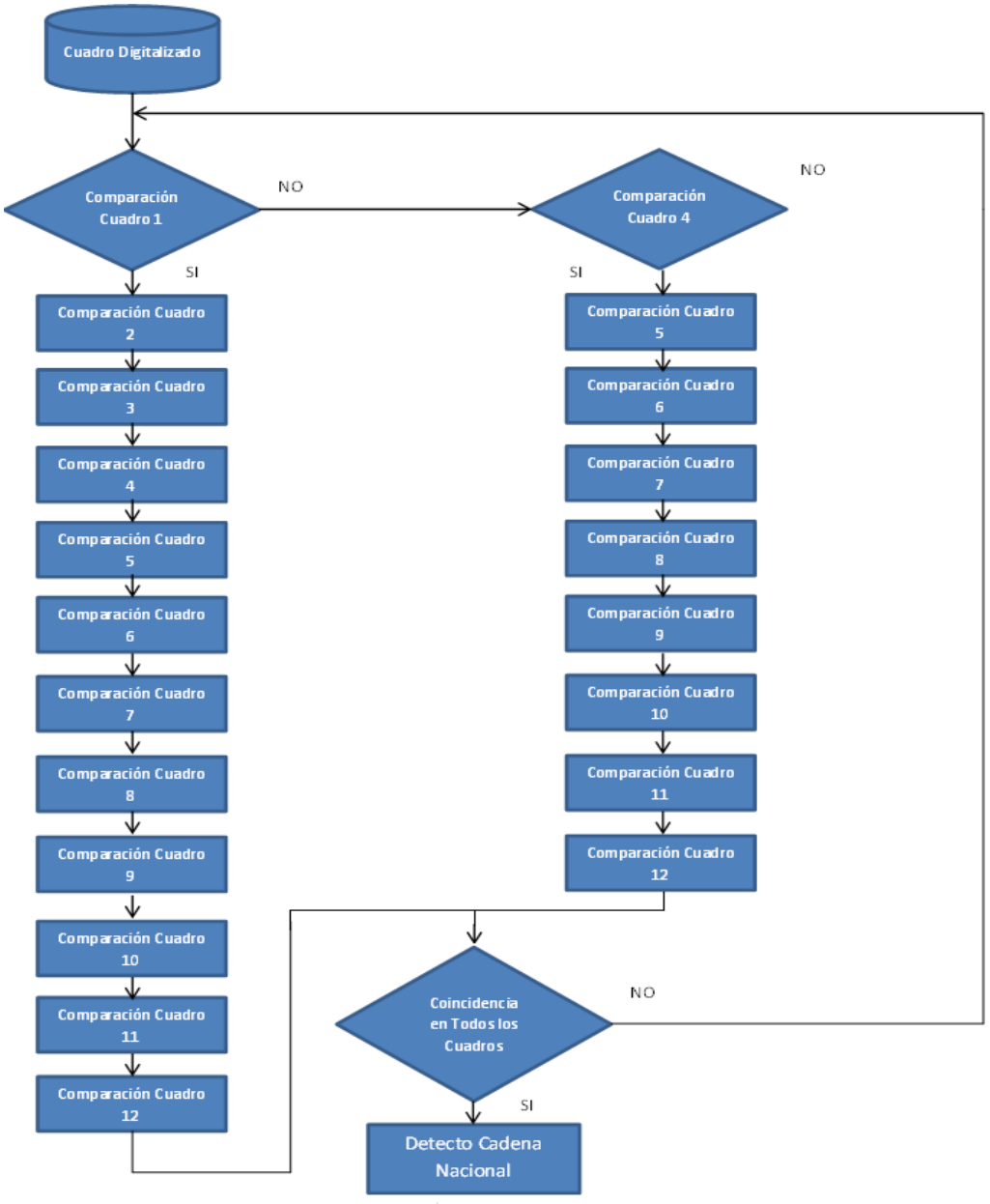

Figura 3.7: Esquema de decisión.

Para entender el esquema de la figura 3.7 debemos saber que el algoritmo carga todos los 12 cuadros de la base de datos junto con sus tres copias con diferentes niveles de ruido de cada uno, y tomara el cuadro del primer segundo junto con otro cuadro separado cuatro segundos entre ellos, es decir, si imaginamos los 12 cuadros

almacenados de la coletilla tomaremos los cuadros que equivalen al segundo 1 y 4 para entrar al proceso de verificación que mostramos en el esquema de la figura 3.7. Ahora bien, un cuadro procesado se compara con el cuadro 1 y 4 junto con sus respectivas copias con ruido almacenados como base de datos, de existir coincidencia de ese cuadro procesado con cualquiera de los cuadros previamente mencionados de la base de datos, se procede a comparar el siguiente cuadro procesado con el siguiente en la lista de la base de datos. Para aclarar mejor este punto tomemos de ejemplo que entro un cuadro y existió coincidencia realizando la correlación de acuerdo al procedimiento descrito anteriormente entre ese cuadro capturado y el primer cuadro en la base de datos, ahora se procede a comparar por correlación el siguiente cuadro capturado con el segundo cuadro de la base de datos, de existir coincidencia se procede a procesar el próximo cuadro capturado y el tercer cuadro de la base de datos, y así sucesivamente hasta realizar el recorrido por todos los cuadros de la base de datos.

La intención de que al principio del algoritmo se compararan los cuadros 1, y 4 con el mismo cuadro procesado, fue debido a que si llega existir algún problema, ya sea de ruido, o que la señal no se procesó bien justo en el momento de que la programación transmitida sea la coletilla de una cadena nacional, este análisis le da tiempo al procesamiento de poder verificar si lo procesado corresponde a una cadena nacional.

3.3. FASE III: DESARROLLO DE UNA INTERFAZ GRAFICA

En esta etapa se realizó una interfaz gráfica, la cual tiene como propósito activar la detección cadena nacional siguiendo con el esquema descrito anteriormente, así como poder cambiar la información de la base de datos para poder mantener el programa operativo por cualquier cambio que surja en la coletilla de presentación de una cadena nacional.

Esta interfaz gráfica se realizó con las librerías de Python PyQt, la cual nos ofrece un entorno sencillo y básico para trabajar de acuerdo a las necesidades que se requieren, en la tabla 3.1 observamos las librerías utilizadas para el desarrollo de esta interfaz.

Tabla 3.1: Librerias Utilizadas.

Librería	Descripción
Numpy	Librería con gran cantidad de paquetes para el cómputo científico , arreglo de vectores, matrices, entre otros.
Opencv	Librería para el procesamiento de imágenes, detección de objetos, reconocimiento de patrones, máquinas de aprendizaje, entre otros.
PyQt	Librería para el desarrollo de interfaces gráficas.
Time	Librería con variables de tiempo.
Sys	Librería para el manejo de funciones y objetos definas por el intérprete.
Serial	Librería de python para el manejo de comunicación serial.
Smtplib	Librería para el envío de correos electrónicos.

Al momento en que la interfaz inicia el proceso de comparación, la capturadora de video procede a digitalizar la señal de video que se encuentra recibiendo; cuando el programa consigue detectar una cadena nacional, el programa envía por el puerto USB la activación de un relay como el descrito en el capítulo II, el cual cambia de estado de acuerdo a la detección de la coletilla de entrada o salida de la cadena nacional.

El programa funciona como una máquina de estados, es decir, tiene un estado de si se encuentra conectado a cadena nacional o no; por lo tanto lleva un registro en un archivo .txt en donde se van almacenando hora, fecha y si detecto o salió de cadena nacional. A partir de este archivo el programa es capaz de recordar cual fue su último estado. Para una mejor explicación de la interfaz gráfica desarrollada consultar el Apéndice F.

3.4. FASE IV: PROBAR LA EFICIENCIA DEL PROGRAMA DE DETECCION DE CADENAS NACIONALES

Con la finalidad de hacer las respectivas evaluaciones en el sitio donde será instalado el programa de detección de cadenas nacionales, se realizaron visitas a DIMETEL. De estas visitas se definieron varios puntos y consideraciones relevantes para la instalación del proyecto. Además se realizaron las mediciones necesarias para determinar la eficiencia del programa de detección.

Para realizar estas mediciones se adiciono ruido digital con diferentes modelos e intensidades a las coletillas de entrada y salida y posteriormente se procedió a reproducir este ruido de manera analógica, para posteriormente digitalizarlo mediante la capturadora de video. Todo esto se realizó con la finalidad de poder someter al programa a diferentes escenarios y poder determinar la eficiencia del mismo.

Capítulo IV

Análisis, interpretación y presentación de los resultados

4.1. PRUEBAS REALIZADAS

Las pruebas que se realizaron para ver el funcionamiento y eficiencia del programa consistieron en tomar la coletilla de entrada y salida de la cadena nacional, degradar adicionándole ruido digital a la coletilla y posteriormente reproducir las coletillas de manera analógica en el estándar NTSC, para posteriormente digitalizar los cuadros de video por medio de la capturadora y realizar la comparación. Se realizaron 11 pruebas, una se tomó el video de las coletillas de entrada sin ruido, 5 pruebas adicionando ruido gaussiano desde 10 % de ruido en la imagen hasta 50 % de ruido en la imagen. Y 5 pruebas son un ruido salt & pepper, igualmente, adicionando el ruido desde 10 % en la totalidad de la imagen hasta un 50 % de ruido. Cada una de las pruebas se realizaron 10 mediciones en total, y se tomaron el número de coincidencias, el tiempo de detección y si detecto o no la coletilla tanto para la coletilla de entrada como para la de salida.

4.1.1. Prueba 1 Coletilla de cadena nacional sin ruido

Para esta prueba se realizaron 10 mediciones en la coletilla de entrada como en la coletilla de salida; en la tabla 4.1 podemos observar los resultados de las pruebas, en donde vemos que el número de coincidencias para que tuvo el programa con la de coletilla de entrada fueron de 12, coincidiendo todas las imágenes de la base de datos con los cuadros capturados de la coletilla logrando detectar la coletilla de entrada de la cadena nacional, además podemos observar que el tiempo de respuesta en promedio fue de 10.4749 segundos.

Por otro lado si observamos la misma tabla 4.1 vemos que se realizaron otras 10 mediciones de la coletilla de salida, observamos que al igual que en la coletilla de entrada, sin presencia de ruido tuvo un numero de coincidencias entre cuadros de 12, detectando con éxito la coletilla de la cadena nacional, además tuvo un tiempo promedio de respuesta de 10,598.

Con estos resultados podemos decir que el programa detecta de manera satisfactoria la coletilla de la cadena nacional tanto de entrada como de salida, coincidiendo en su totalidad todos los cuadros dentro de la referencia.

4.1.2. Prueba 2 Coletilla de cadena nacional con ruido Gaussiano al 10 %

Para esta prueba se adiciono un ruido gaussiano afectando en un 10 % la imagen, como ya se mencionó se realizaron 10 pruebas con el programa para ver su detección tanto como para la coletilla de entrada como para la de salida. Los resultados de estas pruebas los podemos observar en la tabla 4.2. Podemos destacar de acuerdo a los resultados obtenidos que para las 10 pruebas tanto en la coletilla de entrada como en la de salida, el programa logro coincidir los 12 cuadros almacenados, logrando detectar la coletilla de entrada como la de salida de la cadena nacional. Además observar que el tiempo promedio que obtuvo entre las 10 mediciones de la coletilla de entrada fue de 9.4892 segundos, mientras que el tiempo promedio para la coletilla de salida fue de 10.528 segundos.

Tabla 4.1: tabla de las mediciones de la Colettilla sin ruido.

Coletilla de entrada sin ruido			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	12	Si	10.714
2	12	Si	10.412
3	12	Si	10.476
4	12	Si	10.411
5	12	Si	10.479
6	12	Si	10.412
7	12	Si	10.479
8	12	Si	10.411
9	12	Si	10.476
10	12	Si	10.479
Tiempo promedio			10.4749
Coletilla de salida sin ruido			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	12	Si	10.546
2	12	Si	10.544
3	12	Si	10.544
4	12	Si	10.579
5	12	Si	10.611
6	12	Si	10.543
7	12	Si	10.578
8	12	Si	10.619
9	12	Si	10.598
10	12	Si	10.488
Tiempo promedio			10.565

De acuerdo a los resultados obtenidos con las pruebas con ruido gaussiano al 10 % podemos decir que el programa tiene un buen desempeño en la detección de la cadena nacional tanto en la entrada como en la salida.

4.1.3. Prueba 3 Coletilla de cadena nacional con ruido Gaussiano al 20 %

En esta nueva prueba se aumentó la intensidad del ruido gaussiano sobre la coletilla, afectando al video de entrada y de salida en un 20 %, los resultados de estas pruebas podemos observarlos en la tabla 4.3.

Tabla 4.2: tabla de las mediciones de la coletilla con ruido gaussiano a 10 %.

Coletilla de entrada con ruido gaussiano a 10 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	12	Si	9.374
2	12	Si	9.362
3	12	Si	9.335
4	12	Si	9.624
5	12	Si	9.553
6	12	Si	9.344
7	12	Si	9.348
8	12	Si	9.322
9	12	Si	10.018
10	12	Si	9.612
Tiempo promedio			9.4892
Coletilla de salida con ruido gaussiano a 10 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	12	Si	10.567
2	12	Si	10.459
3	12	Si	10.566
4	12	Si	10.518
5	12	Si	10.459
6	12	Si	10.574
7	12	Si	10.534
8	12	Si	10.535
9	12	Si	10.524
10	12	Si	10.544
Tiempo promedio			10.528

Para la coletilla de entrada en las 10 mediciones se obtuvieron 12 coincidencias en total y un tiempo promedio en la detección de 10.170 segundos. En el caso de la coletilla de salida obtuvimos en las 10 mediciones 12 coincidencias y un tiempo promedio de detección de 10.549 segundos.

Con estos resultados podemos observar que para ambos casos en la entrada y la salida, logramos obtener el máximo de coincidencias en la detección de la coletilla, por lo tanto el programa de detección sigue teniendo buen desempeño con este nivel de ruido.

Tabla 4.3: tabla de las mediciones de la coletilla con ruido gaussiano a 20 %.

Coletilla de entrada con ruido gaussiano a 20 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	12	Si	10.188
2	12	Si	10.127
3	12	Si	10.197
4	12	Si	10.201
5	12	Si	10.203
6	12	Si	10.159
7	12	Si	10.123
8	12	Si	10.190
9	12	Si	10.177
10	12	Si	10.135
Tiempo promedio			10.170
Coletilla de salida con ruido gaussiano a 20 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	12	Si	10.528
2	12	Si	10.489
3	12	Si	10.987
4	12	Si	10.458
5	12	Si	10.457
6	12	Si	10.485
7	12	Si	10.476
8	12	Si	10.589
9	12	Si	10.452
10	12	Si	10.567
Tiempo promedio			10.549

4.1.4. Prueba 4 Coletilla de cadena nacional con ruido Gaussiano al 30 %

En este caso se adiciono ruido gaussiano de 30 % en el video de la coletilla de la cadena nacional de entrada y de salida, los resultados obtenidos después de realizar 10 mediciones con esta nueva intensidad de ruido podemos observarlos en la tabla 4.4.

De las 10 mediciones realizadas en la coletilla de entrada como en la de salida podemos ver que ambas tuvieron el máximo número de coincidencias posibles, detectando de manera exitosa la cadena nacional, la entrada de la cadena tuvo un tiempo promedio de detección de 10.228 segundos mientras que la salida tuvo un

tiempo promedio de 10.520.

El programa con este nivel de ruido detecta sin ningún problema la cadena nacional de entrada y salida.

Tabla 4.4: tabla de las mediciones de la coletilla con ruido gaussiano a 30 %.

Coletilla de entrada con ruido Gaussiano 30 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	12	Si	10.076
2	12	Si	10.111
3	12	Si	10.144
4	12	Si	10.211
5	12	Si	10.212
6	12	Si	10.478
7	12	Si	10.143
8	12	Si	10.678
9	12	Si	10.145
10	12	Si	10.078
Tiempo promedio			10.228
Coletilla de salida con ruido Gaussiano 30 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	12	Si	10.478
2	12	Si	10.612
3	12	Si	10.611
4	12	Si	10.477
5	12	Si	10.479
6	12	Si	10.478
7	12	Si	10.477
8	12	Si	10.498
9	12	Si	10.479
10	12	Si	10.611
Tiempo promedio			10.520

4.1.5. Prueba 5 Coletilla de cadena nacional con ruido Gaussiano al 40 %

Para esta prueba se adiciono ruido gaussiano de 40 % sobre el video, nuevamente se realizaron 10 mediciones en la coletilla de entrada y de salida. Los resultados obtenidos en esta nueva prueba pueden apreciarse en la tabla 4.5.

En el caso de la coletilla de entrada de las 10 mediciones una tuvo 8 coincidencias, mientras que el resto tuvo un máximo de 9 coincidencias de los 12 cuadros almacenados como referencia, pero aun así logrando detectar el contenido de la coletilla de entrada de la cadena nacional en todas las pruebas; el programa tuvo un tiempo de detección de 10.4184 segundos.

Por otro lado de las 10 mediciones realizadas en la coletilla de salida, se obtuvo una medición en la cual no se logró detectar la salida, teniendo un máximo de coincidencias entre cuadros igual a 2, para el resto de las mediciones en esta prueba se obtuvieron 8 coincidencias entre cuadros en el video de salida y un tiempo promedio de detección de 10.528 segundos.

Con estos resultados podemos apreciar que el desempeño del programa empieza a ser afectado por los niveles del ruido gaussiano. Sin embargo, aún puede detectar de manera satisfactoria la cadena nacional.

4.1.6. Prueba 6 Coletilla de cadena nacional con ruido Gaussiano al 50 %

Los resultados de las pruebas realizadas con ruido gaussiano al 50 % sobre los videos de las coletillas de entrada y salida podemos apreciarlos en la tabla 4.6.

Vemos que para la coletilla de entrada disminuyo el número de cuadros de coincidencia a 8 pero logrando la detección de la coletilla de entrada, tuvo un tiempo de detección de 10.682 segundos. Por otro lado, la coletilla de la cadena nacional tuvo, al igual que la de entrada, 8 coincidencias entre los cuadros, con excepción de una medición la cual se registró 9 coincidencias, el tiempo promedio en la detección fue de 10.744 segundos.

Con estas mediciones vemos que con mayor intensidad del ruido gaussiano sobre el video el programa va disminuyendo su eficiencia en la detección. Sin embargo, con el ruido a 50 % el programa aun logra detectar el contenido de la cadena nacional.

Tabla 4.5: tabla de las mediciones de la coletilla con ruido gaussiano a 40 %.

Coletilla de entrada con ruido Gaussiano 40 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	9	Si	10.584
2	9	Si	11.241
3	9	Si	10.987
4	9	Si	9.919
5	9	Si	9.937
6	8	Si	9.852
7	9	Si	10.862
8	9	Si	10.851
9	9	Si	9.993
10	9	Si	9.958
Tiempo promedio			10.4184
Coletilla de salida con ruido Gaussiano 40 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	2	No	No tiene
2	8	Si	11.356
3	8	Si	10.676
4	8	Si	10.712
5	8	Si	10.517
6	8	Si	10.048
7	9	Si	10.547
8	8	Si	10.121
9	8	Si	10.097
10	8	Si	10.676
Tiempo promedio			10.528

4.1.7. Prueba 7 Coletilla de cadena nacional con ruido salt & pepper al 10 %

En estas nuevas pruebas se varió el modelo de ruido que afecta al video a Salt & Pepper, afectando a la coletilla en un 10 % de intensidad, los resultados de las pruebas realizadas pueden apreciarse en la tabla 4.7.

Para el caso de la coletilla de entrada en las 10 mediciones que se hicieron se lograron obtener el máximo número de coincidencias igual a 12, con un tiempo promedio de detección de la cadena de entrada de 9.363 segundos.

En el caso de la coletilla de salida también se lograron detectar los 12 cuadros de

Tabla 4.6: tabla de las mediciones de la coletilla con ruido gaussiano a 50 %.

Coletilla de entrada con ruido Gaussiano 50 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	8	Si	10.746
2	8	Si	10.379
3	8	Si	10.591
4	8	Si	10.121
5	8	Si	10.525
6	8	Si	11.035
7	8	Si	10.872
8	8	Si	10.756
9	8	Si	11.122
10	8	Si	10.676
Tiempo promedio			10.682
Coletilla de salida con ruido Gaussiano 50 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	8	Si	10.646
2	8	Si	10.754
3	8	Si	11.043
4	8	Si	10.587
5	8	Si	10.953
6	8	Si	10.512
7	9	Si	10.676
8	8	Si	10.716
9	8	Si	10.683
10	8	Si	10.871
Tiempo promedio			10.744

referencia haciendo que el programa cambie de estado y salga de cadena nacional, además, tuvo un tiempo promedio de detección de 10.499 segundos.

Con estas pruebas podemos decir que el programa tiene un buen desempeño con la presencia de este tipo de ruido.

4.1.8. Prueba 8 Coletilla de cadena nacional con ruido salt & pepper al 20 %

Para esta prueba se aumentó el nivel de ruido Salt & Pepper sobre el video de la coletilla, y nuevamente se realizaron 10 mediciones con la coletilla de entrada y

Tabla 4.7: tabla de las mediciones de la coletilla con ruido salt & pepper a 10 %.

Coletilla de entrada con ruido salt & pepper 10 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	12	Si	9.451
2	12	Si	9.391
3	12	Si	9.352
4	12	Si	9.335
5	12	Si	9.319
6	12	Si	9.322
7	12	Si	9.375
8	12	Si	9.385
9	12	Si	9.319
10	12	Si	9.385
Tiempo promedio			9.363
Coletilla de salida con ruido salt & pepper 10 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	12	Si	10.454
2	12	Si	10.525
3	12	Si	10.524
4	12	Si	10.482
5	12	Si	10.514
6	12	Si	10.471
7	12	Si	10.533
8	12	Si	10.522
9	12	Si	10.455
10	12	Si	10.506
Tiempo promedio			10.499

la coletilla de salida, los resultados de estas mediciones podemos observarlos en la tabla 4.8.

Con este nivel de ruido las coletillas de entrada y salida lograron detectar todos los cuadros de la referencia, logrando que el programa cambie de estado en ambos casos, se obtuvo un tiempo promedio de detección de 9.711 segundos para la coletilla de entrada y de 10.395 segundos para la salida.

Con los resultados de esta prueba el programa sigue siendo funcional bajo estas condiciones.

Tabla 4.8: tabla de las mediciones de la coletilla con ruido salt& pepper a 20 %.

Coletilla de entrada con ruido salt& pepper 20 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	12	Si	9.651
2	12	Si	9.654
3	12	Si	9.716
4	12	Si	9.842
5	12	Si	9.717
6	12	Si	9.688
7	12	Si	9.775
8	12	Si	9.724
9	12	Si	9.656
10	12	Si	9.687
Tiempo promedio			9.711
Coletilla de salida con ruido salt& pepper 20 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	12	Si	10.424
2	12	Si	10.392
3	12	Si	10.399
4	12	Si	10.394
5	12	Si	10.388
6	12	Si	10.401
7	12	Si	10.391
8	12	Si	10.384
9	12	Si	10.389
10	12	Si	10.388
Tiempo promedio			10.395

4.1.9. Prueba 9 Coletilla de cadena nacional con ruido salt & pepper al 30 %

Para esta prueba se aumentó el nivel de ruido Salt & Pepper a 30 %, los resultados de las mediciones realizadas pueden apreciarse en la tabla 4.9.

En estas mediciones se obtuvieron el máximo de coincidencias tanto para la coletilla de entrada y la de salida, cada uno tuvo un tiempo de detección de 10.311 y 10.405 respectivamente. Teniendo este nivel de ruido el programa aun logra desempeñar el máximo de su rendimiento.

Tabla 4.9: tabla de las mediciones de la coletilla con ruido salt& pepper a 30 %.

Coletilla de entrada con ruido salt& pepper 30 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	12	Si	10.279
2	12	Si	10.344
3	12	Si	10.345
4	12	Si	10.344
5	12	Si	10.345
6	12	Si	10.343
7	12	Si	10.276
8	12	Si	10.277
9	12	Si	10.279
10	12	Si	10.278
Tiempo promedio			10.311
Coletilla de salida con ruido salt& pepper 30 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	12	Si	10.345
2	12	Si	10.410
3	12	Si	10.478
4	12	Si	10.344
5	12	Si	10.410
6	12	Si	10.411
7	12	Si	10.410
8	12	Si	10.411
9	12	Si	10.411
10	12	Si	10.418
Tiempo promedio			10.405

4.1.10. Prueba 10 Coletilla de cadena nacional con ruido salt & pepper al 40 %

Los resultados de las mediciones realizadas pueden apreciarse en la tabla 4.10. En donde podemos ver que al igual que en los casos anteriores con este modelo de ruido, se obtuvieron todas las coincidencias en la coletilla de entrada y en las de salida para las 10 mediciones realizadas; además se registró un tiempo promedio de detección de 10.115 y 11.982 respectivamente. A pesar del aumento del ruido en los videos de las coletillas, el programa aún posee una buena respuesta en la detección.

Tabla 4.10: tabla de las mediciones de la coletilla con ruido salt& pepper a 40 %.

Coletilla de entrada con ruido salt& pepper 40 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	12	Si	10.181
2	12	Si	10.088
3	12	Si	10.123
4	12	Si	10.019
5	12	Si	10.135
6	12	Si	10.121
7	12	Si	10.157
8	12	Si	10.151
9	12	Si	10.166
10	12	Si	10.009
Tiempo promedio			10.115
Coletilla de salida con ruido salt& pepper 40 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	12	Si	11.056
2	12	Si	10.791
3	12	Si	11.059
4	12	Si	10.803
5	12	Si	10.788
6	12	Si	10.085
7	12	Si	10.802
8	12	Si	10.780
9	12	Si	11.059
10	12	Si	11.600
Tiempo promedio			11.982

4.1.11. Prueba 11 Coletilla de cadena nacional con ruido salt & pepper al 50 %

Esta fue la última prueba que se realizó sobre el programa de detección de cadenas nacionales, la cual tuvo un modelo de ruido Salt & Pepper y una intensidad del 50 % sobre las coletillas, los resultados de esta prueba puede verse en la tabla 4.11.

La coletilla de entrada en las 10 mediciones realizadas logro detectar el máximo de cuadros de la referencia; mientras que la coletilla de salida si disminuyo el número de cuadros de detección a 8, sin embargo en ambos casos se logró detectar

la cadena nacional en todas las mediciones. El tiempo de detección fue de 10.014 segundos para la entrada y de 10.914 segundos para la salida.

Con esta última prueba se puede observar que el programa empieza a reducir su desempeño, es decir, que para mayores niveles de este modelo de ruido el programa ira siendo menos eficiente.

Tabla 4.11: tabla de las mediciones de la coletilla con ruido salt& pepper a 50 %.

Coletilla de entrada con ruido salt& pepper 50 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	12	Si	9.713
2	12	Si	10.612
3	12	Si	9.741
4	12	Si	11.211
5	12	Si	9.744
6	12	Si	10.077
7	12	Si	9.811
8	12	Si	9.878
9	12	Si	9.677
10	12	Si	9.677
Tiempo promedio			10.0141
Coletilla de salida con ruido salt& pepper 50 %			
Medición	Número de coincidencia	Detección	Tiempo[s]
1	8	Si	10.840
2	8	Si	11.909
3	8	Si	10.705
4	8	Si	11.840
5	8	Si	11.178
6	8	Si	10.741
7	8	Si	10.574
8	8	Si	10.539
9	8	Si	10.407
10	8	Si	10.408
Tiempo promedio			10.9141

Capítulo V

Conclusiones y recomendaciones

5.1. CONCLUSIONES

El aporte principal de este trabajo de investigación, era el desarrollo de un software para la detección de la coletilla de entrada y salida de la cadena nacional, para lograr automatizar el proceso de transmisión de cadenas nacionales con el que cuenta Dimetel. El software fue diseñado en Python y se usaron diferentes librerías para el procesamiento de imágenes, desarrollo de interfaz gráfica para administración del mismo y librerías para comunicación serial y envió de correos electrónicos.

Se realizó un estudio sobre las características de las imágenes, criterios para la detección de objetos, algoritmos para detección, entre otras cosas. Se comprobó el tipo de análisis que debía poseer el programa, es decir, implementar un análisis de cuadro a cuadro y no un detector de objetos de acuerdo a los requerimientos básicos que debía de poseer el programa, como por ejemplo una base de datos modificable. Se determinaron las herramientas que debía de poseer el programa haciendo uso de la librería para procesamiento de imágenes Opencv, tales como, la segmentación, escala de grises y comparación por histogramas de patrones cuadro a cuadro.

Se diseñó el algoritmo de comparación y decisión. El cual, analiza cuadro a cuadro procesado por medio de la capturadora de video, aplica las herramientas para procesamiento de imágenes previamente mencionadas y realiza la comparación de

histogramas de una señal digitalizada y otra de 12 imágenes tanto de la coletilla de entrada como de salida, en donde, se realiza 3 copias de cada imagen de referencia con diferentes intensidades y modelos de ruido. Con la finalidad de que la búsqueda en la comparación sea más certera por si la señal digitalizada se encuentra dañada por ruido. Además se diseñó un método de verificación de llegar a encontrar coincidencia entre algunos cuadros de la referencia y la NTSC digitalizada.

Se desarrolló una interfaz gráfica la cual administra el programa, por dicha interfaz funciona como de una máquina de estados, en la cual dependiendo del estado en el que se encuentre buscara detectar la entrada o salida de la coletilla de la cadena nacional, al momento de ser iniciado el proceso de detección de la cadena nacional, la interfaz muestra en su pantalla en tiempo real la señal digitalizada por medio de la capturadora de video, además se estableció una comunicación vía USB que controla la activación de un Relay al momento de entrar y salir de una cadena nacional, por otro lado también se configuro que la interfaz envié un correo electrónico al momento de entrar y salir de cadena nacional a una lista de contactos la cual puede ser modificada. También se desarrolló un proceso para poder cambiar las imágenes que son usadas como referencia para la detección de la cadena nacional, en la cual al momento que requiera un cambio, se abrirá una ventana emergente la cual se utilizara para poder cambiar la base de datos utilizada por una nueva. Por ultimo con ayuda de un programa podemos controlar el escritorio de la computadora de manera remota, pudiendo controlar y administrar el estado del programa.

Se realizaron pruebas del programa, en donde, se introdujo ruido digital dentro de los videos de la coletilla de entrada y salida de la cadena nacional y se procedió a digitalizarlos por medio de la capturadora de video. Se utilizaron dos modelos de ruido, ruido Gaussiano y ruido Salt & Pepper, los cuales fueron aumentando la intensidad del ruido sobre los videos de las coletillas y se procedió a realizar mediciones, las cuales se anotó el número de coincidencias entre cuadros que tenía el programa con la señal digitalizada, si logro o no la detección y el tiempo promedio que tardaba la detección. De estas mediciones podemos resaltar que el ruido gaussiano afecta en mayor cantidad el desempeño del programa, que el ruido salt &

pepper, ya que con menor intensidad se logró perder más información de los cuadros digitalizados, dificultando de esta forma la detección de la cadena nacional.

5.2. RECOMENDACIONES

Se recomienda que el ministerio del poder popular para la comunicación e información transmita un patrón, que permita identificar la coletilla de la cadena nacional, y que dicho patrón no sea modificado sin importar si se realizó algún cambio en la coletilla de la cadena nacional. O que dicho ente coloque un portal web para que los medios que utilicen una detección automatizada de la coletilla de la cadena nacional puedan descargar el video de la nueva coletilla y de esta manera cambiar los cuadros o frame de referencia para lograr una detección satisfactoria.

Se recomienda realizar un algoritmo de aprendizaje que detecte la información de coletilla de la cadena nacional para así fortalecer el proceso de detección de la cadena nacional. En vista a que la coletilla de la cadena nacional es cambiada tan seguida, el uso de un algoritmo de aprendizaje le daría mayor robustez y fortalecimiento al proceso de detección.

También se recomienda realizar una interfaz web para la administración del programa, por medio de los lenguajes pertinentes, y no usar un servido VNC para controlar el programa. Es decir, que la información que se encuentra dentro del programa sea posible acceder a ella por medio de una interfaz web.

Se recomienda emigrar el software a una FPGA o un miniordenador, para así no depender del uso de una computadora.

Apéndice A

Código del programa principal

```
#_____ importando todas las librerías, métodos y clases a utilizar _____################
import sys                                                                                ##
import cv2                                                                                ##
import numpy as np                                                                        ##
import time                                                                               ##
#_____librería para la ventana principal de la interfaz gráfica                           ##
from Interfaz_Detector import *                                                           ##
#_____Método que calcula la coincidencia con las imágenes de referencia                   ##
from buscar_coincidencia2 import *                                                        ##
#_____ ___Método que calcula los histogramas de todas las imágenes de referencia          ##
from Calculo_histogramas import *                                                         ##
#_____Metodo para el guardado y la lectura de los históricos                              ##
from historico import *                                                                   ##
#_____Metodo para enviar el correo electronico                                            ##
from MailPrueba import *                                                                  ##
#_____metodo de conmutación con la USB-Relay                                              ##
from USB_Relay import *                                                                   ##
import serial                                                                             ##
############################################################################################################################################
#_____Declaración de variables_____#######################
USUARIO = 'dimetel'                                                                       ##
PASSWORD = 'ipv6'                                                                         ##
n = w = h = i = j = k = t = resp = XXX = RR = rr = me = vi = tiempoI = t_i_m= 0           ##
Up_grisE = []                                                                             ##
Down_grisE = []                                                                           ##
Up_grisS = []                                                                             ##
Down_grisS = []                                                                           ##
Up_grisE30s = []                                                                          ##
Dowm_grisE30s = []                                                                        ##
Up_grisE20g = []                                                                          ##
Dowm_grisE20g = []                                                                        ##
Up_grisE40g = []                                                                          ##
Dowm_grisE40g = []                                                                        ##
Up_grisS30s = []                                                                          ##
Down_grisS30s = []                                                                        ##
Up_grisS20g = []                                                                          ##
Down_grisS20g = []                                                                        ##
Up_grisS40g = []                                                                          ##
Down_grisS40g = []                                                                        ##
Z = []                                                                                    ##
Z1 = []                                                                                   ##
#####################################################################################################################################################
#_____Lectura del archivo txt para levantar el ultimo estado_____###################
tabla = leertxt()                                                                          ##
if (len(tabla) == 0 ):                                                                     ##
  modo = 0                                                                                 ##
else :                                                                                     ##
  modo= ((tabla[len(tabla)-1])[0])                                                         ##
#############################################################################################################################################################
#_____Cargar cálculo de los histogramas de las imágenes de referencia_____ ########################
Ma1E = calculo_histogramas_referencia_input()                                              ##
Up_grisE = Ma1E.a_up                                                                       ##
Down_grisE = Ma1E.a_down                                                                   ##
Ma1Es30 = calculo_histogramas_referencia_input30s()                                        ##
Up_grisE30s = Ma1Es30.a_up                                                                 ##
Down_grisE30s =Ma1Es30.a_down                                                              ##
Ma1Eg20 =calculo_histogramas_referencia_input20g()                                         ##
Up_grisE20g =Ma1Eg20.a_up                                                                  ##
Down_grisE20g= Ma1Eg20.a_down                                                              ##
Ma1Eg40 = calculo_histogramas_referencia_input40g()                                        ##
Up_grisE40g =Ma1Eg40.a_up                                                                  ##
Down_grisE40g= Ma1Eg40.a_down                                                              ##
Ma1S = calculo_histogramas_referencia_output()                                             ##
Up_grisS = Ma1S.a_up                                                                       ##
Down_grisS = Ma1S.a_down                                                                   ##
Ma1S30s = calculo_histogramas_referencia_output30s()                                       ##
Up_grisS30s =Ma1S30s.a_up                                                                  ##
Down_grisS30s = Ma1S30s.a_down                                                             ##
Ma1S20g= calculo_histogramas_referencia_output20g()                                        ##
Up_grisS20g= Ma1S20g.a_up                                                                  ##
Down_grisS20g = Ma1S20g.a_down                                                             ##
Ma1S40g= calculo_histogramas_referencia_output40g()                                        ##
Up_grisS40g= Ma1S40g.a_up                                                                  ##
```

```
Down_grisS40g = Ma1S40g.a_down                                                                                   ##
####################################################################################################
# ______________________________Clase para mostrar los frames que digitaliza la capturadora__________________________________###########
class Video():                                                                                                   ##
  def __init__(self,capture):                                                                                    ##
    self.capture = capture                                                                                       ##
    self.currentFrame = np.array([])                                                                             ##
  def captureNextFrame(self):                                                                                    ##
    #capture frame and reverse RBG BGR and return opencv image                                                   ##
    ret, readFrame = self.capture.read()                                                                         ##
    Z = readFrame                                                                                                ##
    if(ret==True):                                                                                               ##
      self.currentFrame = cv2.cvtColor(readFrame,cv2.COLOR_BGR2RGB)                                              ##
    return Z                                                                                                     ##
  def convertFrame(self):                                                                                        ##
    #converts frame to format suitable for QtGui                                                                 ##
    try:                                                                                                         ##
      height,width = self.currentFrame.shape[:2]                                                                 ##
      img = QtGui.QImage(self.currentFrame,width,height,QtGui.QImage.Format_RGB888)                              ##
      img = QtGui.QPixmap.fromImage(img)                                                                         ##
      self.previousFrame = self.currentFrame                                                                     ##
      return img                                                                                                 ##
    except:                                                                                                      ##
      return None                                                                                                ##
####################################################################################################
#__________________________________________ _________Clase de la interfaz principal__________________________#################################
class Mi_Interfaz(QtGui.QDialog):                                                                                ##
  def __init__ (self, parent=None):                                                                              ##
    global rr                                                                                                    ##
    QtGui.QWidget.__init__(self,parent)                                                                          ##
    self.ui = Ui_Dialog()                                                                                        ##
    self.ui.setupUi(self)                                                                                        ##
    QtCore.QObject.connect(self.ui.Button_detector, QtCore.SIGNAL('clicked()'),self.mostarmensaje)               ##
    QtCore.QObject.connect(self.ui.Button_inicio, QtCore.SIGNAL('clicked()'),self.video_start)                   ##
    QtCore.QObject.connect(self.ui.Button_cambio_coletilla, QtCore.SIGNAL('clicked()'), self.mostrar_VentanaE)   ##
    QtCore.QObject.connect(self.ui.Button_reset, QtCore.SIGNAL('clicked()'),self.Resetear)                       ##
    QtCore.QObject.connect(self.ui.Button_continuar, QtCore.SIGNAL('clicked()'),self.Verificar)                  ##
    tabla = leertxt()                                                                                            ##
    if (rr == 0):                                                                                                ##
      if (modo == '1'):                                                                                          ##
        self.ui.label_entro_cadena.setText('      Se encuentra en Cadena Nacional')                              ##
        self.ui.label_entro_cadena.show()                                                                        ##
        #self.ui.label_entro_cadena.setVisible(False)                                                            ##
      else:                                                                                                      ##
        self.ui.label_entro_cadenaS.setText('          Salió de Cadena Nacional')                                ##
        self.ui.label_entro_cadenaS.show()                                                                       ##
  def Verificar(self):                                                                                           ##
    if (((self.ui.lineEdit_usuario.text()) == USUARIO)&((self.ui.lineEdit_password.text())==PASSWORD)):          ##
      self.ui.Button_inicio.setEnabled(True)                                                                     ##
      self.ui.Button_reset.setEnabled(True)                                                                      ##
      self.ui.Button_cambio_coletilla.setEnabled(True)                                                           ##
      self.ui.Button_detector.setEnabled(True)                                                                   ##
      self.ui.label_31.setVisible(False)                                                                         ##
      self.ui.label_32.setVisible(False)                                                                         ##
      self.ui.lineEdit_usuario.setVisible(False)                                                                 ##
      self.ui.lineEdit_password.setVisible(False)                                                                ##
      self.ui.Button_continuar.setVisible(False)                                                                 ##
  def Resetear(self):                                                                                            ##
    global myapp                                                                                                 ##
    global modo                                                                                                  ##
    global tabla                                                                                                 ##
    tabla = leertxt()                                                                                            ##
    if (len(tabla) == 0):                                                                                        ##
      modo = 0                                                                                                   ##
    else:                                                                                                        ##
      modo = ((tabla[len(tabla)-1])[0])                                                                          ##
    myapp.deleteLater()                                                                                          ##
    myapp = Mi_Interfaz()                                                                                        ##
    myapp.show()                                                                                                 ##
    tabla = leertxt()                                                                                            ##
  def mostarmensaje(self):                                                                                       ##
    global tabla                                                                                                 ##
    global t_i_m                                                                                                 ##
```

```
    modo = calculo_modo()                                                                                      ##
    #print(modo)                                                                                               ##
    if(modo == '0'):                                                                                           ##
      self.ui.label_entro_cadenaS.setVisible(False)                                                            ##
      self.ui.label_entro_cadena.setVisible(False)                                                             ##
      self.ui.label_entro_cadena.setText("    Acaba de entrar a Cadena Nacional")                              ##
      self.ui.label_entro_cadena.show()                                                                        ##
      modo = 1                                                                                                 ##
      respuesta = "on"                                                                                         ##
      conmuta(respuesta)                                                                                       ##
      t_i_m = (float(time.strftime("%H"))*3600) + (float(time.strftime("%M"))*60) + float(time.strftime("%S"))  ##
      grabartxt(str(str(modo)+" "+str('Acaba de entrar a Cadena Nacional')+" "+str(time.strftime("%H:%M:%S"))+" "+str(time.strftime("%d/%m/%Y"))))  ##
    elif (modo == '1'):                                                                                        ##
      self.ui.label_entro_cadena.setVisible(False)                                                             ##
      self.ui.label_entro_cadenaS.setVisible(False)                                                            ##
      self.ui.label_entro_cadenaS.setText("    Acaba de salir de Cadena Nacional")                             ##
      self.ui.label_entro_cadenaS.show()                                                                       ##
      modo = 0                                                                                                 ##
      grabartxt(str(str(modo)+" "+str('Acaba de salir de Cadena Nacional')+" "+str(time.strftime("%H:%M:%S"))+" "+str(time.strftime("%d/%m/%Y"))))  ##
      respuesta = "off"                                                                                        ##
      conmuta(respuesta)                                                                                       ##
      t_i_m = (float(time.strftime("%H"))*3600) + (float(time.strftime("%M"))*60) + float(time.strftime("%S"))  ##
  def video_start(self):                                                                                       ##
    global XXX                                                                                                 ##
    global vi                                                                                                  ##
    global t_i_m                                                                                               ##
    self.ui.label_28.setVisible(False)                                                                         ##
    self.ui.label_29.show()                                                                                    ##
    self.ui.label_30.show()                                                                                    ##
    self.video = Video(cv2.VideoCapture(1))                                                                    ##
    if (XXX == 0):                                                                                             ##
      vi = cv2.getTickCount()                                                                                  ##
      cv2.setUseOptimized(True)                                                                                ##
      XXX = 1                                                                                                  ##
    self._timer = QtCore.QTimer(self)                                                                          ##
    self._timer.timeout.connect(self.play)                                                                     ##
    self._timer.start(30)                                                                                      ##
    self.update()                                                                                              ##
  def play(self):                                                                                              ##
    try:                                                                                                       ##
      global t_i_m                                                                                             ##
      global  n                                                                                                ##
      global  w                                                                                                ##
      global  resp                                                                                             ##
      global  h                                                                                                ##
      global Up_grisE                                                                                          ##
      global Down_grisE                                                                                        ##
      global Up_grisE30s                                                                                       ##
      global Down_grisE30s                                                                                     ##
      global Up_grisE40s                                                                                       ##
      global Down_grisE40s                                                                                     ##
      global Up_grisE10g                                                                                       ##
      global Down_grisE10g                                                                                     ##
      global Up_grisE20g                                                                                       ##
      global Down_grisE20g                                                                                     ##
      global Up_grisE30g                                                                                       ##
      global Down_grisE30g                                                                                     ##
      global Up_grisE40g                                                                                       ##
      global Down_grisE40g                                                                                     ##
      global Up_grisS                                                                                          ##
      global Down_grisS                                                                                        ##
      global Up_grisS30s                                                                                       ##
      global Down_grisS30s                                                                                     ##
      global Up_grisS20g                                                                                       ##
      global Down_grisS20g                                                                                     ##
      global Up_grisS40g                                                                                       ##
      global Down_grisS40g                                                                                     ##
      global modo                                                                                              ##
      global tiempo                                                                                            ##
      global me                                                                                                ##
      global a1                                                                                                ##
      global tabla                                                                                             ##
      global vi                                                                                                ##
```

```
modo = calculo_modo()                                                                                          ##
tabla = leertxt()                                                                                              ##
linea = agrupa((tabla[len(tabla)-1]))                                                                          ##
horai = (int(linea.hora[0:2])*3600) + (int(linea.hora[3:5])*60) + int(linea.hora[6:8])                         ##
horaf = (int(time.strftime("%H"))*3600) + (int(time.strftime("%M"))*60) + int(time.strftime("%S"))             ##
#print horai                                                                                                   ##
self.video.captureNextFrame()                                                                                  ##
Z1 = self.video.captureNextFrame()                                                                             ##
if (Z1 == None):                                                                                               ##
   pass                                                                                                        ##
else:                                                                                                          ##
   Z2 = cv2.cvtColor(Z1,cv2.COLOR_BGR2GRAY)                                                                    ##
   y1 = Z2[120:240,180:540]                                                                                    ##
   y2 = Z2[241:361,180:540]                                                                                    ##
   self.ui.label_Ventana.setPixmap(self.video.convertFrame())                                                  ##
   self.ui.label_Ventana.setScaledContents(True)                                                               ##
   if ((modo == '0')):                                                                                         ##
   a1 = detectar_coincidencia_paralelo(y1, y2, Up_grisE, Down_grisE, n, w, resp, Up_grisE30s, Down_grisE30s, Up_grisE20g, Down_grisE20g, Up_grisE40g, Down_grisE40g)    ##
   elif ((modo == '1')):                                                                                       ##
     a1 = detectar_coincidencia_paraleloS(y1, y2, Up_grisS, Down_grisS, n, w, resp, Up_grisS30s, Down_grisS30s, Up_grisS20g, Down_grisS20g, Up_grisS40g, Down_grisS40g)    ##
   w = a1.a                                                                                                    ##
   resp = a1.b                                                                                                 ##
   if (resp == 1):                                                                                             ##
      vi = cv2.getTickCount()                                                                                  ##
   tiempoF = (float(time.strftime("%H"))*3600) + (float(time.strftime("%M"))*60) + float(time.strftime("%S"))  ##
   if (resp == 1):                                                                                             ##
      self.ui.label_1.show()                                                                                   ##
      self.ui.label_13.show()                                                                                  ##
   elif (resp == 2):                                                                                           ##
      self.ui.label_2.show()                                                                                   ##
      self.ui.label_14.show()                                                                                  ##
   elif (resp == 3):                                                                                           ##
      self.ui.label_3.show()                                                                                   ##
      self.ui.label_15.show()                                                                                  ##
   elif (resp == 4):                                                                                           ##
      self.ui.label_4.show()                                                                                   ##
      self.ui.label_16.show()                                                                                  ##
   elif (resp == 5):                                                                                           ##
      self.ui.label_5.show()                                                                                   ##
      self.ui.label_17.show()                                                                                  ##
   elif (resp == 6):                                                                                           ##
      self.ui.label_6.show()                                                                                   ##
      self.ui.label_18.show()                                                                                  ##
   elif (resp == 7):                                                                                           ##
      self.ui.label_7.show()                                                                                   ##
      self.ui.label_19.show()                                                                                  ##
   elif (resp == 8):                                                                                           ##
      self.ui.label_8.show()                                                                                   ##
      self.ui.label_20.show()                                                                                  ##
   elif (resp == 9):                                                                                           ##
      self.ui.label_9.show()                                                                                   ##
      self.ui.label_21.show()                                                                                  ##
   elif (resp == 10):                                                                                          ##
      self.ui.label_10.show()                                                                                  ##
      self.ui.label_22.show()                                                                                  ##
   elif (resp == 11):                                                                                          ##
      self.ui.label_11.show()                                                                                  ##
      self.ui.label_23.show()                                                                                  ##
   elif (resp == 12):                                                                                          ##
      self.ui.label_12.show()                                                                                  ##
      self.ui.label_24.show()                                                                                  ##
   if ((resp == 12)&(modo == '0')):#resp=a1.b                                                                  ##
      self.ui.label_entro_cadenaS.setVisible(False)                                                            ##
      self.ui.label_entro_cadena.setVisible(False)                                                             ##
      self.ui.label_entro_cadena.setText("    Acaba de entrar a Cadena Nacional")                              ##
      self.ui.label_entro_cadena.show()                                                                        ##
      modo = 1                                                                                                 ##
      resp = 0                                                                                                 ##
      grabartxt(str(str(modo)+" "+str('Acaba de entrar a Cadena Nacional')+" "+str(time.strftime("%H:%M:%S"))+" "+str(time.strftime("%d/%m/%Y"))))    ##
      tiempoI = (float(time.strftime("%H"))*3600) + (float(time.strftime("%M"))*60) + float(time.strftime("%S"))    ##
      body = str('El dispositivo')+" "+ str('Acaba de entrar a Cadena Nacional')+" "+str(time.strftime("%H:%M:%S"))+" "+str(time.strftime("%d/%m/%Y"))    ##
      enviar_correo(body)                                                                                      ##
      respuesta = "on"                                                                                         ##
```

```
            conmuta(respuesta)                                                                                          ##
            t_i_m = (float(time.strftime("%H"))*3600) + (float(time.strftime("%M"))*60) + float(time.strftime("%S"))    ##
            self.ui.label_33.show()                                                                                     ##
        elif (((resp == 12)&(modo == '1'))or ((modo == '1')&((horaf - horai)>= 300))):                                  ##
            self.ui.label_entro_cadena.setVisible(False)                                                                ##
            self.ui.label_entro_cadenaS.setVisible(False)                                                               ##
            self.ui.label_entro_cadenaS.setText("    Acaba de salir de Cadena Nacional")                                ##
            self.ui.label_entro_cadenaS.show()                                                                          ##
            modo = 0                                                                                                    ##
            resp = 0                                                                                                    ##
            grabartxt(str(str(modo)+" "+str('Acaba de salir de Cadena Nacional')+" "+str(time.strftime("%H:%M:%S"))+" "+str(time.strftime("%d/%m/%Y"))))  ##
            tiempoI = (float(time.strftime("%H"))*3600) + (float(time.strftime("%M"))*60) + float(time.strftime("%S"))  ##
            body = str('El dispositivo')+" "+ str('Acaba de Salir de cadena Nacional')+" "+str(time.strftime("%H:%M:%S"))+" "+str(time.strftime("%d/%m/%Y"))  ##
            enviar_correo(body)                                                                                         ##
            respuesta = "off"                                                                                           ##
            conmuta(respuesta)                                                                                          ##
            t_i_m = (float(time.strftime("%H"))*3600) + (float(time.strftime("%M"))*60) + float(time.strftime("%S"))    ##
            self.ui.label_33.show()                                                                                     ##
        if ((resp == 0)&((float(tiempoF))-(float(tiempoI))>= 30.00)):                                                   ##
            self.ui.label_1.setVisible(False)                                                                           ##
            self.ui.label_2.setVisible(False)                                                                           ##
            self.ui.label_3.setVisible(False)                                                                           ##
            self.ui.label_4.setVisible(False)                                                                           ##
            self.ui.label_5.setVisible(False)                                                                           ##
            self.ui.label_6.setVisible(False)                                                                           ##
            self.ui.label_7.setVisible(False)                                                                           ##
            self.ui.label_8.setVisible(False)                                                                           ##
            self.ui.label_9.setVisible(False)                                                                           ##
            self.ui.label_10.setVisible(False)                                                                          ##
            self.ui.label_11.setVisible(False)                                                                          ##
            self.ui.label_12.setVisible(False)                                                                          ##
            self.ui.label_13.setVisible(False)                                                                          ##
            self.ui.label_14.setVisible(False)                                                                          ##
            self.ui.label_15.setVisible(False)                                                                          ##
            self.ui.label_16.setVisible(False)                                                                          ##
            self.ui.label_17.setVisible(False)                                                                          ##
            self.ui.label_18.setVisible(False)                                                                          ##
            self.ui.label_19.setVisible(False)                                                                          ##
            self.ui.label_20.setVisible(False)                                                                          ##
            self.ui.label_21.setVisible(False)                                                                          ##
            self.ui.label_22.setVisible(False)                                                                          ##
            self.ui.label_23.setVisible(False)                                                                          ##
            self.ui.label_24.setVisible(False)                                                                          ##
            self.ui.label_33.setVisible(False)                                                                          ##
        n = n + 1                                                                                                       ##
        if ((modo == '0') & ((float(tiempoF) - t_i_m)>= 30.00)):                                                        ##
            self.ui.label_entro_cadenaS.setVisible(False)                                                               ##
            self.ui.label_entro_cadenaS.setText("           Salió de Cadena Nacional")                                  ##
            self.ui.label_entro_cadenaS.show()                                                                          ##
            #self.ui.label_33.setVisible(False)                                                                         ##
        elif((modo == '1') & ((float(tiempoF) - t_i_m)>= 30.00)):                                                       ##
            self.ui.label_entro_cadena.setVisible(False)                                                                ##
            self.ui.label_entro_cadena.setText("      Se encuentra en Cadena Nacional")                                 ##
            self.ui.label_entro_cadena.show()                                                                           ##
            #sel.ui.label_33.setVisible(False)                                                                          ##
    except TypeError:                                                                                                   ##
        pass                                                                                                            ##
###############################################################################################################################
#_________________________________________Ventana emergente para el cambio de la coletilla____________________________________________#########
  def mostrar_VentanaE(self):                                                                                           ##
    global ventana                                                                                                      ##
    ventana = Secundaria().exec_()                                                                                      ##
###________Ventana emergente                                                                                            ##
class Secundaria(QtGui.QDialog):                                                                                        ##
  def __init__(self, parent=None):                                                                                      ##
    global label_Frame                                                                                                  ##
    global label_pantalla                                                                                               ##
    global Button_Cap                                                                                                   ##
    global Button_Cap1                                                                                                  ##
    global Button_Cap2                                                                                                  ##
    global Button_exit                                                                                                  ##
    QtGui.QDialog.__init__(self, parent)                                                                                ##
    contenedor = QtGui.QWidget()                                                                                        ##
```

```
        self.setWindowTitle('Cambio de Coletilla de CN')  ##
        self.setWindowIcon(QtGui.QIcon('C:\Users\Cristina\Desktop\Gustavo Tesis\Imagenes\Icon1.png'))  ##
        self.setGeometry(40,40,722,572)  ##
        label_Frame = QtGui.QLabel(self)  ##
        label_Frame.setGeometry(QtCore.QRect(1, 1, 720, 480))  ##
        label_Frame.setText((""))  ##
        Button_Cap = QtGui.QPushButton("Play", self)  ##
        Button_Cap.setGeometry(QtCore.QRect(60, 530, 75, 40))  ##
        Button_Cap1 = QtGui.QPushButton("Capturar entrada", self)  ##
        Button_Cap1.setGeometry(QtCore.QRect(195, 530, 100, 40))  ##
        Button_Cap2 = QtGui.QPushButton("Capturar salida", self)  ##
        Button_Cap2.setGeometry(QtCore.QRect(427, 530, 100, 40))  ##
        Button_exit = QtGui.QPushButton("exit", self)  ##
        Button_exit.setGeometry(QtCore.QRect(587, 530, 75, 40))  ##
        label_pantalla = QtGui.QLabel(self)  ##
        label_pantalla.setGeometry(QtCore.QRect(323, 482, 250, 40))  ##
        label_pantalla.setText((""))  ##
        self.connect(Button_exit, QtCore.SIGNAL("clicked()"), self.close)  ##
        self.connect(Button_Cap, QtCore.SIGNAL("clicked()"), self.captura)  ##
        self.connect(Button_Cap1, QtCore.SIGNAL("clicked()"),self.grabarE)  ##
        self.connect(Button_Cap2, QtCore.SIGNAL("clicked()"),self.grabarS)  ##
    def captura(self):  ##
        i = 0  ##
        Button_Cap.setEnabled(False)  ##
        self.video = Video(cv2.VideoCapture(1))  ##
        self._timer = QtCore.QTimer(self)  ##
        self._timer.timeout.connect(self.playe)  ##
        self._timer.start(30)  ##
        self.update()  ##
    def playe(self):  ##
        try:  ##
            global RR  ##
            global i  ##
            global j  ##
            global t  ##
            global K  ##
            global Z  ##
            Z1 = self.video.captureNextFrame()  ##
            if (RR == 1):  ##
                if ((j == WW+2)or(j == WW+32)or(j == WW+62)or(j == WW+92)or(j == WW+122)or(j == WW+152)or(j == WW+182)or(j == 212)or(j == WW+242)or(j == WW+272) or (j == WW+302) or (j == WW+332)):  ##
                    Z.insert(K,Z1)  ##
                    K = K + 1  ##
                    #print str('cargo')+str(j)  ##
            if (K == 12) & (t == 0):  ##
                for i in range(12):  ##
                    cv2.imwrite('C:\Users\Cristina\Desktop\Gustavo Tesis\Imagenes\Imagen_ref_entrada\Ref'+str(i)+'.jpg',Z[i])  ##
                t = 1  ##
                label_pantalla.setText(("Cargo Imagen de referencia de entrada "))  ##
            elif (RR == 2):  ##
                if ((j == QQ+2)or(j == QQ+32)or(j == QQ+62)or(j == QQ+92)or(j == QQ+122)or(j == QQ+152)or(j == QQ+182)or(j == QQ+212)or(j == QQ+242)or(j == QQ+272) or (j == QQ+302) or (j == QQ+332)):  ##
                    Z.insert(K,Z1)  ##
                    K = K + 1  ##
                if (K == 12) & (t == 0):  ##
                    for i in range(12):  ##
                        cv2.imwrite('C:\Users\Cristina\Desktop\Gustavo Tesis\Imagenes\Imagen_ref_salida\Ref'+str(i)+'.jpg',Z[i])  ##
                    t = 1  ##
                    label_pantalla.setText(("Cargo Imagenes de referencia para la salida"))  ##
            j = j + 1  ##
            label_Frame.setPixmap(self.video.convertFrame())  ##
            label_Frame.setScaledContents(True)  ##
        except TypeError:  ##
            pass  ##
    def grabarE(self):  ##
        global RR  ##
        global WW  ##
        Button_Cap1.setEnabled(False)  ##
        RR = 1  ##
        WW = j  ##
    def grabarS (self):  ##
        global RR  ##
        global QQ  ##
        Button_Cap2.setEnabled(False)  ##
        RR = 2  ##
        QQ = j  ##
```

```
    i = 0                                                                                                                              ##
  ####################################################################################################################################
#_________________________________________________Llamado a la interfaz principal_____________________________________#####################
if __name__ == "__main__":                                                                                                             ##
  app = QtGui.QApplication(sys.argv)                                                                                                   ##
  myapp = Mi_Interfaz()                                                                                                                ##
  myapp.show()                                                                                                                         ##
  sys.exit(app.exec_())                                                                                                                ##
######################################################################################################################################
```

Apéndice B

Algoritmo de toma de decisión

```
####____________________________________________Método para la toma de decisión ____________________________________________________________##
class objeto (object):                                                                                                                        ##
    def __init__(self,a,b,c):                                                                                                                 ##
        self.a = a                                                                                                                            ##
        self.b = b                                                                                                                            ##
        self.c = c                                                                                                                            ##
###_________________________________Método que busca coincidencia de las imágenes de referencia de entrada________________________________________##
def detectar_coincidencia_paralelo(template1,template2,Histo_up,Histo_down,contador1,contador2,decide,Histo_up30s,Histo_down30s,Histo_up20g,     ##
Histo_down20g,Histo_up40g,Histo_down40g):                                                                                                     ##
    import cv2                                                                                                                                ##
    a1 = objeto(contador2,decide,contador1)                                                                                                   ##
    hist2_up_gris = cv2.calcHist([template1], [0], None, [256], [0,256])                                                                      ##
    hist2_down_gris = cv2.calcHist([template2], [0], None, [256], [0,256])                                                                    ##
    if ((decide == 0)):                                                                                                                       ##
        Correlation_Hist_up_gris = cv2.compareHist(Histo_up[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)                                     ##
        Correlation_Hist_Down_gris = cv2.compareHist(Histo_down[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)                               ##
        if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                                           ##
            a1.b = decide + 1                                                                                                                 ##
            a1.a =  contador1                                                                                                                 ##
        else:                                                                                                                                 ##
            Correlation_Hist_up_gris = cv2.compareHist(Histo_up30s[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)                              ##
            Correlation_Hist_Down_gris = cv2.compareHist(Histo_down30s[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)                        ##
            if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                                       ##
                a1.b = decide + 1                                                                                                             ##
                a1.a =  contador1                                                                                                             ##
            else:                                                                                                                             ##
                Correlation_Hist_up_gris = cv2.compareHist(Histo_up20g[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)                          ##
                Correlation_Hist_Down_gris = cv2.compareHist(Histo_down20g[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)                    ##
                if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                                   ##
                    a1.b = decide + 1                                                                                                         ##
                    a1.a =  contador1                                                                                                         ##
                else:                                                                                                                         ##
                    Correlation_Hist_up_gris = cv2.compareHist(Histo_up40g[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)                      ##
                    Correlation_Hist_Down_gris = cv2.compareHist(Histo_down40g[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)                ##
                    if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                               ##
                        a1.b = decide + 1                                                                                                     ##
                        a1.a =  contador1                                                                                                     ##
                    else:                                                                                                                     ##
                        decide = 4                                                                                                            ##
                        a1.b = decide                                                                                                         ##
                        Correlation_Hist_up_gris = cv2.compareHist(Histo_up[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)                     ##
                        Correlation_Hist_Down_gris = cv2.compareHist(Histo_down[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)               ##
                        if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                           ##
                            a1.b = decide + 1                                                                                                 ##
                            a1.a =  contador1                                                                                                 ##
                        else:                                                                                                                 ##
                            Correlation_Hist_up_gris = cv2.compareHist(Histo_up30s[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)              ##
                            Correlation_Hist_Down_gris = cv2.compareHist(Histo_down30s[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)        ##
                            if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                       ##
                                a1.b = decide + 1                                                                                             ##
                                a1.a =  contador1                                                                                             ##
                            else:                                                                                                             ##
                                Correlation_Hist_up_gris = cv2.compareHist(Histo_up20g[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)          ##
                                Correlation_Hist_Down_gris = cv2.compareHist(Histo_down20g[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)    ##
                                if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                   ##
                                    a1.b = decide + 1                                                                                         ##
                                    a1.a =  contador1                                                                                         ##
                                else:                                                                                                         ##
                                    Correlation_Hist_up_gris = cv2.compareHist(Histo_up40g[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)      ##
                                    Correlation_Hist_Down_gris = cv2.compareHist(Histo_down40g[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)##
                                    if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                               ##
                                        a1.b = decide + 1                                                                                     ##
```

```
                a1.a =  contador1                                                                                            ##
              else:                                                                                                          ##
                decide = 0                                                                                                   ##
                a1.b = decide                                                                                                ##
  elif((0 < decide <= 12)):                                                                                                  ##
    if((contador1 - contador2)<= 50):                                                                                        ##
      Correlation_Hist_up_gris = cv2.compareHist(Histo_up[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)                      ##
      Correlation_Hist_Down_gris = cv2.compareHist(Histo_down[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)                ##
      if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                            ##
        a1.b = decide + 1                                                                                                    ##
        a1.a = contador1                                                                                                     ##
      else:                                                                                                                  ##
        Correlation_Hist_up_gris = cv2.compareHist(Histo_up30s[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)                 ##
        Correlation_Hist_Down_gris = cv2.compareHist(Histo_down30s[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)           ##
        if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                          ##
          a1.b = decide + 1                                                                                                  ##
          a1.a =  contador1                                                                                                  ##
        else:                                                                                                                ##
          Correlation_Hist_up_gris = cv2.compareHist(Histo_up20g[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)               ##
          Correlation_Hist_Down_gris = cv2.compareHist(Histo_down20g[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)         ##
          if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                        ##
            a1.b = decide + 1                                                                                                ##
            a1.a =  contador1                                                                                                ##
          else:                                                                                                              ##
            Correlation_Hist_up_gris = cv2.compareHist(Histo_up40g[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)             ##
            Correlation_Hist_Down_gris = cv2.compareHist(Histo_down40g[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)       ##
            if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                      ##
              a1.b = decide + 1                                                                                              ##
              a1.a =  contador1                                                                                              ##
    elif((contador1 - contador2)>50):                                                                                        ##
      decide = 0                                                                                                             ##
      a1.b = 0                                                                                                               ##
  return a1                                                                                                                  ##
##########################################################################################################################
###________________________________Método que busca coincidencia de las imágenes de referencia de salida _________________________________________##
def detectar_coincidencia_paraleloS(template1,template2,Histo_up,Histo_down,contador1,contador2,decide,Histo_up30s,Histo_down30s,Histo_up20g,      ##
Histo_down20g,Histo_up40g,Histo_down40g):                                                                                    ##
  import cv2                                                                                                                 ##
  a1 = objeto(contador2,decide,contador1)                                                                                    ##
  hist2_up_gris = cv2.calcHist([template1], [0], None, [256], [0,256])                                                       ##
  hist2_down_gris = cv2.calcHist([template2], [0], None, [256], [0,256])                                                     ##
  if ((decide == 0)):                                                                                                        ##
    Correlation_Hist_up_gris = cv2.compareHist(Histo_up[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)                        ##
    Correlation_Hist_Down_gris = cv2.compareHist(Histo_down[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)                  ##
    if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                              ##
      a1.b = decide + 1                                                                                                      ##
      a1.a =  contador1                                                                                                      ##
    else:                                                                                                                    ##
      Correlation_Hist_up_gris = cv2.compareHist(Histo_up30s[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)                   ##
      Correlation_Hist_Down_gris = cv2.compareHist(Histo_down30s[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)             ##
      if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                            ##
        a1.b = decide + 1                                                                                                    ##
        a1.a =  contador1                                                                                                    ##
      else:                                                                                                                  ##
        Correlation_Hist_up_gris = cv2.compareHist(Histo_up20g[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)                 ##
        Correlation_Hist_Down_gris = cv2.compareHist(Histo_down20g[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)           ##
        if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                          ##
          a1.b = decide + 1                                                                                                  ##
          a1.a =  contador1                                                                                                  ##
        else:                                                                                                                ##
          Correlation_Hist_up_gris = cv2.compareHist(Histo_up40g[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)               ##
          Correlation_Hist_Down_gris = cv2.compareHist(Histo_down40g[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)         ##
          if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                        ##
            a1.b = decide + 1                                                                                                ##
```

```
          a1.a =  contador1                                                                                                    ##
        else:                                                                                                                  ##
          decide = 4                                                                                                           ##
          a1.b = decide                                                                                                        ##
          Correlation_Hist_up_gris = cv2.compareHist(Histo_up[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)                    ##
          Correlation_Hist_Down_gris = cv2.compareHist(Histo_down[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)              ##
          if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                          ##
            a1.b = decide + 1                                                                                                  ##
            a1.a =  contador1                                                                                                  ##
          else:                                                                                                                ##
            Correlation_Hist_up_gris = cv2.compareHist(Histo_up30s[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)               ##
            Correlation_Hist_Down_gris = cv2.compareHist(Histo_down30s[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)         ##
            if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                        ##
              a1.b = decide + 1                                                                                                ##
              a1.a =  contador1                                                                                                ##
            else:                                                                                                              ##
              Correlation_Hist_up_gris = cv2.compareHist(Histo_up20g[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)             ##
              Correlation_Hist_Down_gris = cv2.compareHist(Histo_down20g[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)       ##
              if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                      ##
                a1.b = decide + 1                                                                                              ##
                a1.a =  contador1                                                                                              ##
              else:                                                                                                            ##
                Correlation_Hist_up_gris = cv2.compareHist(Histo_up40g[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)           ##
                Correlation_Hist_Down_gris = cv2.compareHist(Histo_down40g[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)     ##
                if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                    ##
                  a1.b = decide + 1                                                                                            ##
                  a1.a =  contador1                                                                                            ##
                else:                                                                                                          ##
                  decide = 0                                                                                                   ##
                  a1.b = decide                                                                                                ##
  elif((0 < decide <= 12)):                                                                                                    ##
    if((contador1 - contador2)<= 50):                                                                                          ##
      Correlation_Hist_up_gris = cv2.compareHist(Histo_up[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)                        ##
      Correlation_Hist_Down_gris = cv2.compareHist(Histo_down[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)                  ##
      if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                              ##
        a1.b = decide + 1                                                                                                      ##
        a1.a = contador1                                                                                                       ##
      else:                                                                                                                    ##
        Correlation_Hist_up_gris = cv2.compareHist(Histo_up30s[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)                   ##
        Correlation_Hist_Down_gris = cv2.compareHist(Histo_down30s[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)             ##
        if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                            ##
          a1.b = decide + 1                                                                                                    ##
          a1.a =  contador1                                                                                                    ##
        else:                                                                                                                  ##
          Correlation_Hist_up_gris = cv2.compareHist(Histo_up20g[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)                 ##
          Correlation_Hist_Down_gris = cv2.compareHist(Histo_down20g[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)           ##
          if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                          ##
            a1.b = decide + 1                                                                                                  ##
            a1.a =  contador1                                                                                                  ##
          else:                                                                                                                ##
            Correlation_Hist_up_gris = cv2.compareHist(Histo_up40g[decide],hist2_up_gris, cv2.cv.CV_COMP_CORREL)               ##
            Correlation_Hist_Down_gris = cv2.compareHist(Histo_down40g[decide],hist2_down_gris, cv2.cv.CV_COMP_CORREL)         ##
            if ((Correlation_Hist_up_gris >= 0.7)&(Correlation_Hist_Down_gris >= 0.7)):                                        ##
              a1.b = decide + 1                                                                                                ##
              a1.a =  contador1                                                                                                ##
    elif((contador1 - contador2)>50):                                                                                          ##
      decide = 0                                                                                                               ##
      a1.b = 0                                                                                                                 ##
  return a1                                                                                                                    ##
############################################################################################################################################
```

Apéndice C

Código para cargar los valores de los Histogramas de las imágenes de referencia

```
##__________________________________________Método para cargar la base de datos_______________________________##############################
import cv2                                                                                                  ##
class Matrices(object):                                                                                     ##
  def __init__(self,a_up,a_down):                                                                           ##
    self.a_up = a_up                                                                                        ##
    self.a_down = a_down                                                                                    ##
##________________Cargamos los parámetros de las imágenes de referencia tanto de la coletilla de entrada y salida de la cadena ____________################
def calculo_histogramas_referencia_input():                                                                 ##
  i = 0                                                                                                     ##
  Up_gris = []                                                                                              ##
  Down_gris = []                                                                                            ##
  for i in range(12):                                                                                       ##
    frame = cv2.imread('C:\Users\Cristina\Desktop\Gustavo Tesis\Imagenes\Imagen_ref_entrada\Ref'+str(i)+'.jpg')   ##
    frame1 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)                                                        ##
    frame2 = frame1[120:240,180:540]                                                                        ##
    frame3 = frame1[241:361,180:540]                                                                        ##
    hist_up_gris = cv2.calcHist([frame2],[0],None,[256],[0,256])                                            ##
    hist_down_gris = cv2.calcHist([frame3],[0],None,[256],[0,256])                                          ##
    Up_gris.insert(i,hist_up_gris)                                                                          ##
    Down_gris.insert(i,hist_down_gris)                                                                      ##
  M1 = Matrices(Up_gris, Down_gris)                                                                         ##
  return M1                                                                                                 ##
def calculo_histogramas_referencia_input30s():                                                              ##
  i = 0                                                                                                     ##
  Up_gris = []                                                                                              ##
  Down_gris = []                                                                                            ##
  for i in range(12):                                                                                       ##
    frame = cv2.imread('C:\Users\Cristina\Desktop\Gustavo Tesis\Imagenes\Imagen_ref_entrada\Refs30_'+str(i)+'.jpg')   ##
    frame1 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)                                                        ##
    frame2 = frame1[120:240,180:540]                                                                        ##
    frame3 = frame1[241:361,180:540]                                                                        ##
    hist_up_gris = cv2.calcHist([frame2],[0],None,[256],[0,256])                                            ##
    hist_down_gris = cv2.calcHist([frame3],[0],None,[256],[0,256])                                          ##
    Up_gris.insert(i,hist_up_gris)                                                                          ##
    Down_gris.insert(i,hist_down_gris)                                                                      ##
  M1 = Matrices(Up_gris, Down_gris)                                                                         ##
  return M1                                                                                                 ##
def calculo_histogramas_referencia_input20g():                                                              ##
  i = 0                                                                                                     ##
  Up_gris = []                                                                                              ##
  Down_gris = []                                                                                            ##
  for i in range(12):                                                                                       ##
    frame = cv2.imread('C:\Users\Cristina\Desktop\Gustavo Tesis\Imagenes\Imagen_ref_entrada\Refg20_'+str(i)+'.jpg')   ##
    frame1 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)                                                        ##
    frame2 = frame1[120:240,180:540]                                                                        ##
    frame3 = frame1[241:361,180:540]                                                                        ##
    hist_up_gris = cv2.calcHist([frame2],[0],None,[256],[0,256])                                            ##
    hist_down_gris = cv2.calcHist([frame3],[0],None,[256],[0,256])                                          ##
    Up_gris.insert(i,hist_up_gris)                                                                          ##
    Down_gris.insert(i,hist_down_gris)                                                                      ##
  M1 = Matrices(Up_gris, Down_gris)                                                                         ##
  return M1                                                                                                 ##
def calculo_histogramas_referencia_input40g():                                                              ##
  i = 0                                                                                                     ##
  Up_gris = []                                                                                              ##
  Down_gris = []                                                                                            ##
  for i in range(12):                                                                                       ##
    frame = cv2.imread('C:\Users\Cristina\Desktop\Gustavo Tesis\Imagenes\Imagen_ref_entrada\Refg40_'+str(i)+'.jpg')   ##
    frame1 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)                                                        ##
    frame2 = frame1[120:240,180:540]                                                                        ##
    frame3 = frame1[241:361,180:540]                                                                        ##
    hist_up_gris = cv2.calcHist([frame2],[0],None,[256],[0,256])                                            ##
    hist_down_gris = cv2.calcHist([frame3],[0],None,[256],[0,256])                                          ##
    Up_gris.insert(i,hist_up_gris)                                                                          ##
```

```
        Down_gris.insert(i,hist_down_gris)                                                                                    ##
    M1 = Matrices(Up_gris, Down_gris)                                                                                         ##
    return M1                                                                                                                 ##
def calculo_histogramas_referencia_output():                                                                                  ##
    i = 0                                                                                                                     ##
    Up_gris = []                                                                                                              ##
    Down_gris = []                                                                                                            ##
    for i in range(12):                                                                                                       ##
        frame = cv2.imread('C:\Users\Cristina\Desktop\Gustavo Tesis\Imagenes\Imagen_ref_salida\Ref'+str(i)+'.jpg')             ##
        frame1 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)                                                                      ##
        frame2 = frame1[120:240,180:540]                                                                                      ##
        frame3 = frame1[241:361,180:540]                                                                                      ##
        hist_up_gris = cv2.calcHist([frame2],[0],None,[256],[0,256])                                                          ##
        hist_down_gris = cv2.calcHist([frame3],[0],None,[256],[0,256])                                                        ##
        Up_gris.insert(i,hist_up_gris)                                                                                        ##
        Down_gris.insert(i,hist_down_gris)                                                                                    ##
    M2 = Matrices(Up_gris, Down_gris)                                                                                         ##
    return M2                                                                                                                 ##
def calculo_histogramas_referencia_output30s():                                                                               ##
    i = 0                                                                                                                     ##
    Up_gris = []                                                                                                              ##
    Down_gris = []                                                                                                            ##
    for i in range(12):                                                                                                       ##
        frame = cv2.imread('C:\Users\Cristina\Desktop\Gustavo Tesis\Imagenes\Imagen_ref_salida\Refs30_'+str(i)+'.jpg')         ##
        frame1 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)                                                                      ##
        frame2 = frame1[120:240,180:540]                                                                                      ##
        frame3 = frame1[241:361,180:540]                                                                                      ##
        hist_up_gris = cv2.calcHist([frame2],[0],None,[256],[0,256])                                                          ##
        hist_down_gris = cv2.calcHist([frame3],[0],None,[256],[0,256])                                                        ##
        Up_gris.insert(i,hist_up_gris)                                                                                        ##
        Down_gris.insert(i,hist_down_gris)                                                                                    ##
    M2 = Matrices(Up_gris, Down_gris)                                                                                         ##
    return M2                                                                                                                 ##
def calculo_histogramas_referencia_output20g():                                                                               ##
    i = 0                                                                                                                     ##
    Up_gris = []                                                                                                              ##
    Down_gris = []                                                                                                            ##
    for i in range(12):                                                                                                       ##
        frame = cv2.imread('C:\Users\Cristina\Desktop\Gustavo Tesis\Imagenes\Imagen_ref_salida\Refg20_'+str(i)+'.jpg')         ##
        frame1 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)                                                                      ##
        frame2 = frame1[120:240,180:540]                                                                                      ##
        frame3 = frame1[241:361,180:540]                                                                                      ##
        hist_up_gris = cv2.calcHist([frame2],[0],None,[256],[0,256])                                                          ##
        hist_down_gris = cv2.calcHist([frame3],[0],None,[256],[0,256])                                                        ##
        Up_gris.insert(i,hist_up_gris)                                                                                        ##
        Down_gris.insert(i,hist_down_gris)                                                                                    ##
    M2 = Matrices(Up_gris, Down_gris)                                                                                         ##
    return M2                                                                                                                 ##
def calculo_histogramas_referencia_output40g():                                                                               ##
    i = 0                                                                                                                     ##
    Up_gris = []                                                                                                              ##
    Down_gris = []                                                                                                            ##
    for i in range(12):                                                                                                       ##
        frame = cv2.imread('C:\Users\Cristina\Desktop\Gustavo Tesis\Imagenes\Imagen_ref_salida\Refg40_'+str(i)+'.jpg')         ##
        frame1 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)                                                                      ##
        frame2 = frame1[120:240,180:540]                                                                                      ##
        frame3 = frame1[241:361,180:540]                                                                                      ##
        hist_up_gris = cv2.calcHist([frame2],[0],None,[256],[0,256])                                                          ##
        hist_down_gris = cv2.calcHist([frame3],[0],None,[256],[0,256])                                                        ##
        Up_gris.insert(i,hist_up_gris)                                                                                        ##
        Down_gris.insert(i,hist_down_gris)                                                                                    ##
    M2 = Matrices(Up_gris, Down_gris)                                                                                         ##
    return M2                                                                                                                 ##
```

Apéndice D

Código para el control del USB-Relay

```
####_________________________________________Método para la administración de la USB-Relay ________________________________________##
def conmuta(respuesta):                                                                                                            ##
   import sys                                                                                                                      ##
   import serial                                                                                                                   ##
###______________________________el puerto depende del ordenador en el cual se trabaja                                             ##
   portName = "COM8"                                                                                                               ##
   relayNum = "0"                                                                                                                  ##
   relayCmd = respuesta                                                                                                            ##
   #_______Open port for communication                                                                                             ##
   serPort = serial.Serial(portName, 19200, timeout=1)                                                                             ##
   #____________Send the command                                                                                                   ##
   serPort.write("relay "+ str(relayCmd) +" "+ str(relayNum) + "\n\r")                                                             ##
   #Close the port                                                                                                                 ##
   serPort.close()                                                                                                                 ##
####################################################################################################################################
```

Apéndice E

Código para el envío del correo electrónico

```
##_____________________________________Método para enviar el correo electrónico__________________________________________________________####
def enviar_correo(cuerpo):                                                                                                                 ##
  Correo = ['gustavomalpicapineros@gmail.com','nelsonmogollon18@gmail.com','angelogasparini@hotmail.com']                                  ##
  i = 0                                                                                                                                    ##
  for i in range(len(Correo)):                                                                                                             ##
    import smtplib                                                                                                                         ##
    from email.mime.text import MIMEText                                                                                                   ##
    from email.mime.multipart import MIMEMultipart                                                                                         ##
    fromaddr = cbrandt@uc.edu.ve                                                                                                           ##
    toaddr = Correo[i]                                                                                                                     ##
    msg = MIMEMultipart()                                                                                                                  ##
    msg['From'] = fromaddr                                                                                                                 ##
    msg['To'] = toaddr                                                                                                                     ##
    msg['Subject'] = "Detector de Cadenas Nacionales"                                                                                      ##
    body = cuerpo                                                                                                                          ##
    msg.attach(MIMEText(body,'plain'))                                                                                                     ##
    server = smtplib.SMTP('nsmtp.uc.edu.ve',25)                                                                                            ##
    server.starttls()                                                                                                                      ##
    server.ehlo()                                                                                                                          ##
    server.login("cbrandt@uc.edu.ve", "cebc.19*")                                                                                          ##
    text = msg.as_string()                                                                                                                 ##
    server.sendmail(fromaddr, toaddr, text)                                                                                                ##
#############################################################################################################################################
```

Apéndice F

Manual de usuario para el software de Detección de la Cadena Nacional de radio y televisión

Bienvenidos

Bienvenido al manual de usuario para el software de Detección de la Cadena Nacional de radio y televisión. En este manual se será de utilidad para el correcto uso del software, así como la administración del mismo para futuros cambios que se requiera (de ser necesario). Recomendamos leerlo en su totalidad.

Requerimientos

Este programa fue desarrollado bajo el lenguaje Python versión 2.7.9, por tal motivo se requiere la instalación del mismo sobre la plataforma donde será instalado; además se requiere las siguientes librerías de Python para su correcto funcionamiento:

- Numpy versión 1.8.0

- Opencv versión 2.4.10
- PyQt versión 4.11.4
- SMTPlib
- PySerial

El resto de las librerías que requiere el programa ya vienen predeterminadas junto con el instalador de Python. Importante. De usar versiones diferentes a las mencionadas no podemos garantizar el funcionamiento del programa. Además de Python junto con sus librerías se requiere una capturadora de video como se muestra en la figura 6.1, dicha capturadora digitalizara la programación de la señal que esté recibiendo y posteriormente se hará el análisis de detección de la cadena nacional, por tal motivo es vital tener este dispositivo para el uso del programa.

Figura 6.1: Capturadora de video.

Por ultimo también se requiere de un dispositivo USB – Relay, como el que muestra la figura 6.2 que envía la señal de 5 voltios por via USB cada vez que el programa entra o sale de cadena nacional, activando el relay que realiza la conmutación.

Figura 6.2: USB-Relay.

Uso del Programa ULead

La capturadora de video al momento de adquirirla vino con un programa de edición de video llamado ULead, dicho programa es necesario instalarlo si el sistema en que se corre es Windows, debido a que por medio de ese programa podemos inicializar la capturadora de video, es decir, el programa posee algunos protocolos que permiten que la tarjeta capturadora realice el proceso de digitalización de una señal de video en estándar NTSC, PAL, etc. De no realizar este procedimiento, la interfaz realizada generara un error al momento de inicializar la detección de la cadena nacional. El programa ULead viene en un CD junto con el paquete de la capturadora de video, el serial para la instalación del programa esta ubicado en la parte de atrás del estuche y lo repetimos aquí para mayor información:

- Serial Software ULead: 783A2 – 8A000 – 05566663

Además de utilizar el programa ULead para inicializar los protocolos de la capturadora de video; también nos apoyamos en él para el proceso de cambiar la re-

ferencia que es utilizada para detectar la cadena nacional. Este procedimiento se explicara más adelante.

Proceso de Instalación

El programa fue probado y desarrollado sobre una laptop con un sistema Windows 7, un procesador Intel Core 2 Duo y 4 Gb de memoria RAM. Una de las ventajas que ofrece Python, es que es un lenguaje multiplataforma así que puede ser instalado en cualquier otro sistema, así como también la librería de PyQt; sin embargo las librerías de Numpy y Opencv se requiere la versión del sistema donde será instalado el programa.

Instalación en Windows

Cuando Python es instalado en Windows crea una carpeta en el disco local c con el siguiente nombre: " C:
Python27 " . El programa consta de varios archivos .py que deben de ubicarse dentro de esa carpeta para su funcionamiento, el archivo "Interfaz_ def_ prueba.py " es el archivo que genera la interfaz gráfica, mientras que el archivo " buscar_ coincidencias.py " es el código que contiene el algoritmo de detección y decisión. Además cuenta con otro archivo " MailPrueba.py ", este archivo se encarga de enviar un correo electrónico cada vez que el programa entre o salga de cadena nacional. Además tenemos otro archivo "USB_ Relay.py" , este archivo envía la activación del relay conectado mediante USB a la computadora. Es importante resaltar que deben de estar ubicados dentro de la carpeta de Python en el disco c. Teniendo los archivos bien ubicados debemos de ejecutar el archivo "Interfaz_ def_ prueba.py" para generar la interfaz gráfica y poder correr el programa para la detección de la cadena nacional. Es importante que se cree una carpeta dentro de la carpeta de Python en el disco C, con el nombre de "Imágenes" y otras dos dentro de esa misma con los nombres de "Imágenes Referencia Entrada" e " Imágenes Referencia Salida" , dentro de estas carpetas estarán las imágenes utilizadas como referencia

para la comparación y detección de la cadena nacional. Por tal motivo es de vital importancia que creen para su funcionamiento.

Interfaz Gráfica

En la figura 6.3 podemos observar la interfaz gráfica desarrollada. La interfaz consta de una pantalla en la que se podrá observar en tiempo real la programación digitalizada por la capturadora de video, en la parte inferior de la pantalla observamos un anuncio que indica el estado del programa, es decir, si se encuentra o no en cadena nacional; además consta de un gráfico de coincidencias que se procede a llenar a medida que van aumentando las coincidencias entre cuadros y por ultimo consta de 5 botones que administran la inicialización, administración y estado del programa. Además en la parte superior de la pantalla observamos que es necesario ingresar una contraseña para tener control sobre el programa, de esta manera garantizamos que no todo el mundo tendrá acceso o manipulación sobre el programa. A continuación se procede a explicar de manera más detallada cada uno de estos puntos mencionados.

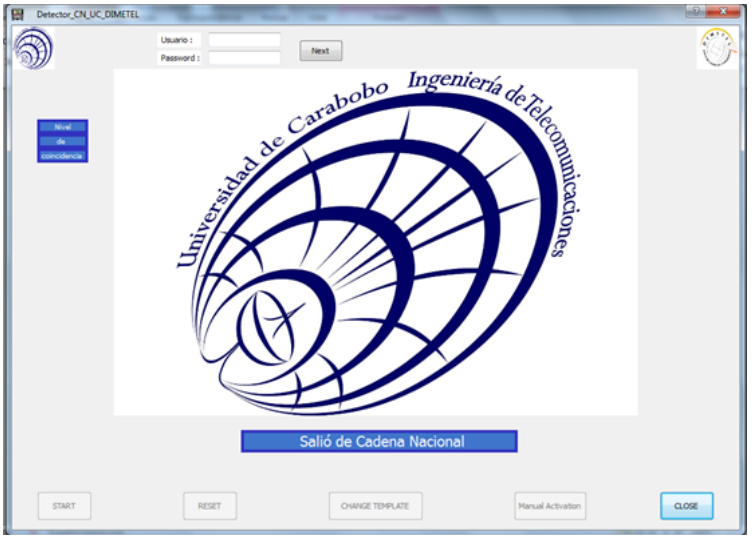

Figura 6.3: Interfaz Gráfica Detector de Cadenas.

Acceso al Programa

En la parte superior de la pantalla de la interfaz podemos apreciar que se requiere de un usuario y contraseña para acceder a la interfaz, al agregar los datos correspondientes se activara el resto de las opciones que ofrece el programa. En la figura 6.4 podemos apreciar el espacio de acceso.

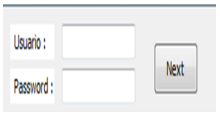

Figura 6.4: Usuario y Contraseña para activar programa.

Pantalla de la Interfaz

Cuando el programa se inicializa, se mostrara en la pantalla del programa la programación digitalizada mediante la capturadora de video en tiempo real, así como podemos apreciar en la figura 6.5.

Figura 6.5: Pantalla del programa de Detección de Cadena Nacional.

Niveles de Coincidencia

Del lado izquierdo de la pantalla de la interfaz podemos observar los niveles de coincidencias que el programa va teniendo, a medida que va comparando los cuadros procesados con los cuadros de referencia, el máximo de coincidencias que puede tener el programa es de 12 coincidencias y un mínimo de 8. En la figura 6.6 podemos observar un ejemplo del gráfico.

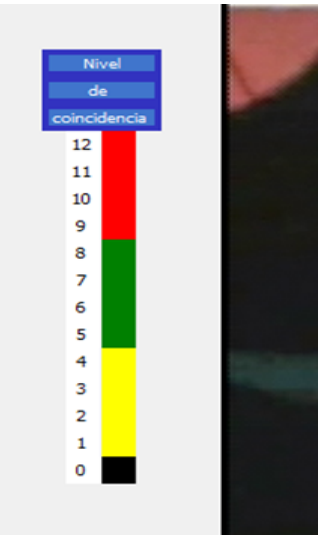

Figura 6.6: Niveles de Coincidencias.

Estado del Programa

El estado del programa indica si se encuentra en cadena nacional o no; en la parte de debajo de la pantalla del programa se muestra el estado, el programa genera un archivo .txt ubicado en la carpeta de Python en el disco C, en el cual, el programa va generando un registro de la hora, el día y si entro o salió de cadena nacional. A partir de este registro el programa tiene la capacidad de recordar cual fue su último estado, es decir, si se encontraba conectado a cadena o no. En la figura 6.7 podemos apreciar los mensajes generados por el programa en su respectivo estado.

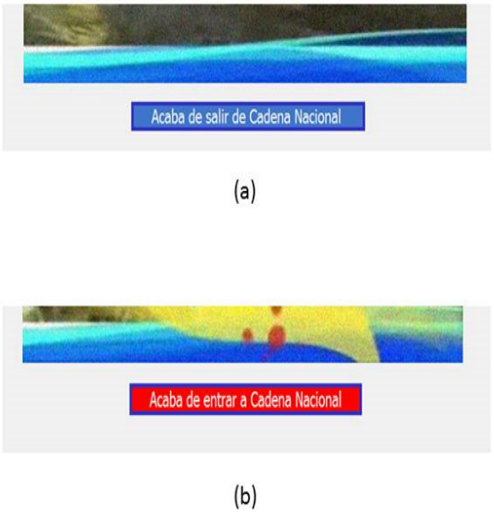

Figura 6.7: Estado del Programa Detector de Cadenas Nacionales.

Botones del Programa

El programa contiene 5 botones los cuales serán explicados a continuación:

- START: El botón de START inicia el programa de detección, es decir, que sin presionarlo la pantalla ni los niveles de coincidencias se activaran, por tal motivo se requiere de su activación para que el programa pueda arrancar su análisis. En la figura 6.8 se muestra una imagen del botón START.

Figura 6.8: Botón START.

- RESET: El botón de RESET reinicia el programa en su totalidad. En la figura 6.9 observamos una imagen del mismo.
- CHANGE TEMPLATE:

  Con este botón podemos cambiar la base de datos del programa, es decir, cambiamos la referencia que se utiliza para la comparación entre los cuadros

Figura 6.9: Botón RESET.

que se buscan detectar y la programación que se encuentra procesando. En la figura 6.10 observamos una imagen de este botón.

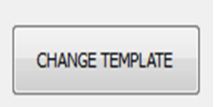

Figura 6.10: Botón CHANGE TEMPLATE.

A continuación procedemos a explicar el proceso de cambiar la referencia. Primero como se mencionó anteriormente requerimos el uso del software ULead para el proceso de cambiar la base de datos. Esto se debe, a que la base de datos almacenada debe de ser digitalizada por medio de la misma capturadora de video, así podemos garantizar que las imágenes almacenadas poseen una misma tonalidad de color, intensidad de bits y mismas dimensiones que los cuadros capturados cuando se realiza el proceso de detección de la cadena nacional. Podemos seguir los siguientes pasos descritos para el proceso de cambiar la base de datos.

1. El canal del estado siempre transmite la coletilla de la cadena nacional y como se mencionó anteriormente es necesario que la referencia se obtenga por medio de la capturadora de video. Se utiliza el software ULead y se procede a grabar varias horas de programación del canal del estado, una vez verificado que se posee una cadena nacional digitaliza por medio de la capturadora procedemos al siguiente paso.
2. Teniendo una cadena nacional ya digitalizada, usamos nuevamente el software ULead y haciendo uso de las herramientas que ofrece sobre edición de video, eliminamos toda la información que no corresponda a la coletilla de la cadena nacional. Teniendo esto procedemos a crear dos

archivos de video (.avi preferiblemente); uno con la coletilla de entrada y de nombre "Coletilla de Entrada" y el otro con la coletilla de salida y el archivo con nombre "Coletilla de Salida". Ambos archivos los ubicamos en una carpeta que creamos de nombre "Videos Coletillas Cadena Nacional" dentro de la carpeta de Python en el disco c.

3. Teniendo estos archivos procedemos a ir al programa y oprimimos el botón de CHANGE TEMPLATE, al hacer esto se abrirá una venta emergente como lo muestra la figura 6.11, en donde observamos dos botones, uno con el nombre "Capturar entrada" y el otro "Capturar Salida"; como su nombre lo dice, al oprimir el botón de "Capturar entrada" se muestra en la pantalla emergente el video de la coletilla de entrada de la cadena nacional y se inicia el proceso de cambio de referencia, cuando se son cargadas todas las imágenes de referencia se muestra un mensaje en la ventana. Después se repite el mismo procedimiento para la coletilla de salida de la cadena nacional. Cuando almacenamos una nueva base de datos se grabaran 12 imágenes .jpg (dentro de la carpeta de Imágenes que fue creada en la carpeta de Python en el disco c), tanto como para la coletilla de entrada como para la de salida, cada una de estas imágenes se grabaran 3 copias por cada una de ellas, estas copias vienen degradas por ruido; esto se realizó con la finalidad de que la información sea mas amplia y por tal motivo poder aumentar el porcentaje de detección del programa por si la señal digitalizada por medio de la captura viene corrupta por ruido.

- Manual Activation: El botón Manual Activation cambia el estado del programa de manera manual, es decir, si ocurrió algún error en la detección y el programa no logro identificar una cadena nacional, al oprimir este botón el programa cambia de estado y reconoce que la programación que está procesando corresponde a una cadena nacional, por el contrario, supongamos que el programa no detecto la salida de una cadena, al oprimir el botón vuelve a cambiar de estado y reconoce que la información que procesa corresponde a programación regular. Cada uno de estos cambios de estados se almacenan en

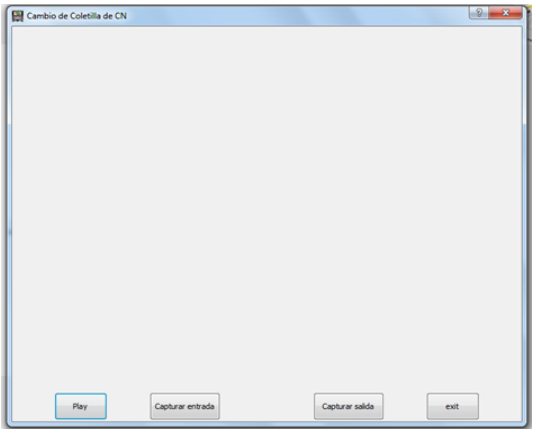

Figura 6.11: Ventana Emergente del botón CHANGE TEMPLATE.

el archivo .txt que se mencionó anteriormente. En la figura 6.12 observamos una imagen del botón.

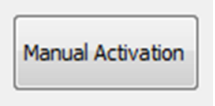

Figura 6.12: Botón Manual Activation.

- CLOSE: Por ultimo tenemos el botón de CLOSE, el cual sirve para salir del programa.

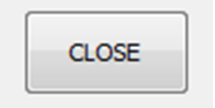

Figura 6.13: Botón CLOSE.

Activación Remota

Recomendamos la instalación del programa TeamViewer si la plataforma donde se instalara es Windows, este programa es un Virtual Network Computer (VNC), es decir, el programa toma la información del escritorio de la computadora y permite

que otras personas puedan controlarlo de manera remota. También existe la aplicación Andriod, de esta manera uno puede acceder a la computadora y al programa de manera remota.

Correo Electrónico

Por medio de las librerías SMTPlib, el programa de detección cada vez que entre o salga de una cadena nacional enviara un correo electrónico a una lista de contactos que posee almacenado. Dentro de la carpeta de Python en el disco c tenemos el archivo de "MailPrueba.py", al abrir este archivo podemos fijarnos que en la parte superior de código hay una variable con nombre "correo", en esta variable podemos agregar o quitar a los cuales el programa contactara cada vez que cambie de estado. Podemos observar una imagen de la misma en la figura 6.14.

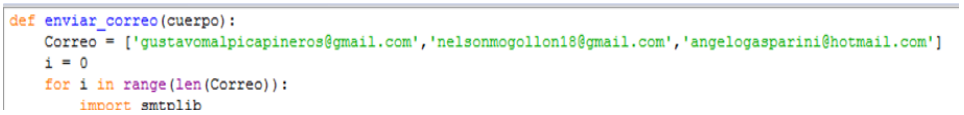

Figura 6.14: Correos electrónicos dentro de MailPrueba.py.

USB – Relay

Los drivers de instalación de este dispositivo pueden conseguirse en su página de internet: [14]. Teniendo los drivers del relay instalados es necesario señalar algo para su funcionamiento. El modo en que el programa activa y desactiva el relay controlado por USB es mediante un código de Python ubicado dentro de la carpeta de Python en el disco c con nombre "USB_ Relay.py". Este archivo da la orden al relay que se active, sin embargo el USB debe de estar conectado a un puerto específico del computador, y como el hardware varía entre computadora y computadora es más sencillo modificar el puerto en el archivo.

1. Conectamos el dispositivo en el puerto USB que queremos que funcione.

2. Por medio de Windows podemos determinar que puerto es el que está ubicado el dispositivo.

3. Teniendo el nombre del puerto en donde está conectado el relay; abrimos el archivo "USB_ Relay.py" y la variable "PortName" la modificamos con el nombre del puerto en donde esté conectado el USB-Relay. En la figura 6.15 observamos un ejemplo.

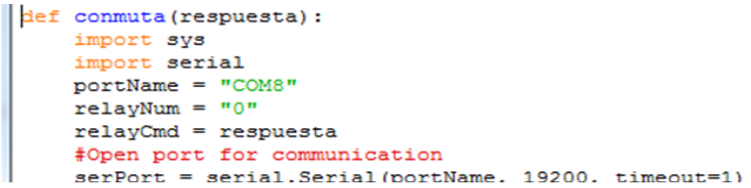

Figura 6.15: Puerto de conexión del USB - Relay.

Contáctenos

Para cualquier duda o soporte avanzado del sobre el programa de detección de cadenas nacionales de radio y televisión contáctenos por los correos: nelsonmogollon18@gmail.com, gustavomalpicapineros@gmail.com

Referencias Bibliográficas

[1] Luz Penagos Wilder Jimenez. *Diseño e implementación de un sistema de detección de patrones de audio utilizando el dispositivo Raspberry Pi Modelo B, para la identificación de cadenas nacionales de Radio y TV para ser integrado en la conmutación automatizada de la programación de las estaciones FMUC y UCTV.* Inf. téc. UNIVERSIDAD DE CARABOBO, 2015.

[2] S. Benzaquen. *Sistema de detección de caras en imágenes de video*. Inf. téc. Universidad Simón Bolívar, 2006.

[3] Gumaa T. *Automated Inspection of an Apple Moth*. Inf. téc. Universidad de Ciencias Aplicadas Hamk, 2014.

[4] *Encore Electronics, Capturador de audio video*. URL: http://www.encore-usa.com/jm/product/ENMVG.

[5] J. R. Parker. *Algorithms For Processing and Computer Vision*. Ed. por Second Edition. Wiley Publishing, Inc., 2011.

[6] Rafael C. Gonzalez. *Digital Image Processing*. Ed. por Second Edition. Prentice Hall, 2002.

[7] *Tutotial Opencv*. URL: http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_comparison/histogram_comparison.html.

[8] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Ed. por First Edition. Springer, 2010.

[9] Adrian Kaebler Gary Bradski. *Learning OpenCV Computer Vision with the OpenCV Library*. Ed. por First Edition. O'REILLY, 2008.

[10] Roger Boyle Milan Sonka Vaclav Hlavac. *Image Processing, Analysis, and Machine Vision*. Ed. por Third Edition. Thomson, 2008.

[11] Mick Hurbis. *The following discussion of NTSC video standards is from the first edition*. Ed. por first Edition. Mick Hurbis-Cherrier y Focal Press, 2007.

[12] FCC. *NTSC SIGNAL SPECIFICATIONS*. FEDERAL COMNUNICATIONS COMMISSION, 1953. URL: WWW.FCC.gov.

[13] *Comisión Nacional de Telecomunicaciones*. URL: www.Conatel.gob.ve.

[14] *Numato Lab*. URL: http://numato.com.

[15] *Coursera*. URL: https://www.coursera.org.

[16] *Open Source DVR*. URL: https://www.mythtv.org.