

**UNIVERSIDAD DE COSTA RICA**  
**Facultad de Ingeniería**  
**Escuela de Ingeniería Eléctrica**

**IE0499 – Proyecto Eléctrico**

**Sistema de reconocimiento de imágenes  
para diagnóstico de un conmutador de red**

por

**Timna Belinda Brown Ramírez**

**Ciudad Universitaria Rodrigo Facio**

Julio de 2020



# **Sistema de reconocimiento de imágenes para diagnóstico de un conmutador de red**

por

**Timna Belinda Brown Ramírez**

B61254

**IE0499 – Proyecto Eléctrico**

Aprobado por

---

Ing. Marvin Coto Jiménez, PhD

*Profesor guía*

---

Ing. Lochi Yu Lo, PhD

*Profesor lector*

---

Ing. Rafael Esteban Badilla Alvarado

*Profesor lector*

Julio de 2020



## **Resumen**

# **Sistema de reconocimiento de imágenes para diagnóstico de un conmutador de red**

por

**Timna Belinda Brown Ramírez**

Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

Profesor guía: Ing. Marvin Coto Jiménez, PhD

Julio de 2020

Este documento presenta el desarrollo de un sistema capaz de reconocer mediante imágenes y por medio de un video remoto, la estructura y características del comportamiento de un conmutador de red. El objetivo principal del proyecto se centra en revisar de manera automatizada los LEDs de un conmutador y notificar si existen cambios en el comportamiento, debido a que muchas veces se presentan anomalías en conmutadores de red, los cuales se tienden a ubicar en lugares destinados al almacenamiento de servidores o laboratorios. Un enfoque de detección de intensidad de color resulta ser adecuado para las condiciones en las cuales se presenta el equipo ya que el propósito del algoritmo es el análisis señales en una imagen o video. Finalmente, se utilizan herramientas como GitHub, con la intención de llevar un control adecuado del desarrollo del proyecto y otras Open Source: Python como lenguaje principal y entre algunos paquetes importantes para la elaboración del mismo se pueden mencionar numpy, OpenCV.

**Palabras claves:** *Python, reconocimiento, imágenes, video, conmutador, Aruba 2930M.*

---

### **Acerca de IE0499 – Proyecto Eléctrico**

El Proyecto Eléctrico es un curso semestral bajo la modalidad de trabajo individual supervisado, con el propósito de aplicar estrategias de diseño y análisis a un problema de temática abierta de la ingeniería eléctrica. Es un requisito de graduación para el grado de bachiller en Ingeniería Eléctrica de la Universidad de Costa Rica.



**Abstract**

**Sistema de reconocimiento de imágenes  
para diagnóstico de un conmutador de red**

Original in Spanish. Translated as: “Image recognition system for diagnosis  
of a network switch”

by

**Timna Belinda Brown Ramírez**

University of Costa Rica  
Department of Electrical Engineering  
Tutor: Ing. Marvin Coto Jiménez, PhD  
July of 2020

This document presents the development of a system capable of recognizing by means of images and by means of a remote video the structure and characteristics of the behavior of a network switch. The main objective of the project is focused on automatically reviewing the LEDs of a switch and notifying if there are changes in behavior, because many times there are anomalies in network switches which tend to be located in places intended for the storage of servers or laboratories. A color intensity detection approach turns out to be suitable for the conditions in which the equipment is presented since the purpose of the algorithm is the analysis of signals in an image or video. Finally, tools such as GitHub are used, with the intention of taking adequate control of the development of the project and other Open Source: Python as the main language and among some important packages for the elaboration of it, numpy, OpenCV can be mentioned.

**Keywords:** *Python, recognition, images, video, switch, Aruba 2930M.*

---

**About IE0499 – Proyecto Eléctrico (“Electrical Project”)**

The “Electrical Project” is a course of supervised individual work of one semester, with the purpose of applying design and analysis strategies to a problem in an open topic in electrical engineering. It is a requisite of graduation for the Bachelor of Science in Electrical Engineering, granted by the University of Costa Rica.



*Dedicado a mi familia y amigos.*

## **Agradecimientos**

A todas aquellas personas que han sido parte de mi aprendizaje, quienes han sido mi guía, un hombro en quién apoyarme, a todo aquel quién ha influido en mi vida personal y académica. Quisiera agradecerles por ser parte de mi vida, por ser mi motivación y por acompañarme que de alguna u otra manera nunca me dejaron sola. Finalmente, quisiera agradecerles por permitirme e influenciar de manera positiva en mi crecimiento integral tanto personal, académico como laboral.



# Índice general

Índice general	xii
Índice de figuras	xiii
Índice de tablas	xiii
<b>1 Introducción</b>	<b>1</b>
1.1. Alcances del proyecto . . . . .	1
1.2. Justificación . . . . .	1
1.3. Problema a resolver . . . . .	2
1.4. Objetivos . . . . .	2
1.4.1. Objetivo general . . . . .	2
1.4.2. Objetivos específicos . . . . .	2
1.5. Metodología . . . . .	2
1.6. Cronograma . . . . .	3
<b>2 Marco teórico</b>	<b>5</b>
2.1. Hardware . . . . .	5
2.1.1. Comutador de red . . . . .	5
2.1.2. ARUBA - 2930M . . . . .	5
2.1.3. Puerto RJ45 . . . . .	7
2.2. Contratiempos . . . . .	8
2.3. Cámara . . . . .	8
2.4. Software . . . . .	9
2.5. Reconocimiento de imágenes/video . . . . .	10
2.5.1. Python . . . . .	10
2.5.2. OpenCV . . . . .	10
2.5.3. NumPy . . . . .	11
2.5.4. Glob . . . . .	11
2.5.5. OS . . . . .	11
2.5.6. Collections . . . . .	12

<b>3 Modelado y funcionamiento</b>	<b>13</b>
3.1. Construcción . . . . .	13
3.1.1. Diagrama de flujo . . . . .	14
3.2. Funcionamiento . . . . .	15
3.2.1. Análisis de imágenes . . . . .	15
3.2.2. Análisis de video . . . . .	16
3.3. Historial de decisiones . . . . .	17
3.3.1. Historial del proceso para el análisis de imágenes . . . . .	17
3.3.2. Historial del proceso para el análisis de videos . . . . .	28
<b>4 Resultados y análisis</b>	<b>29</b>
4.1. Resultados del análisis de imágenes . . . . .	29
4.1.1. Resultados en la terminal . . . . .	29
4.1.2. Resultados de Python IDLE . . . . .	30
4.2. Resultados del análisis de videos . . . . .	33
<b>5 Conclusiones y recomendaciones</b>	<b>37</b>
5.1. Conclusiones . . . . .	37
5.2. Recomendaciones . . . . .	37
<b>6 Anexos</b>	<b>39</b>
<b>Bibliografía</b>	<b>43</b>

# Índice de figuras

2.1. ARUBA 2930M estructura física [3] . . . . .	6
2.2. Puerto RJ45 [1] . . . . .	7
2.3. HPE X121 1G SFP RJ45 T Transceiver (J8177C) [4] . . . . .	8
2.4. Variantes en las imágenes . . . . .	9
2.5. Matiz de color [9] . . . . .	11
3.1. Diagrama de flujo del algoritmo a realizar [9] . . . . .	14
3.2. Imagen base de análisis con un led en verde. . . . .	18
3.3. Muestra de la prueba del método del contorno . . . . .	19
3.4. Muestra de la prueba del método del contorno con reducción de ruido . . . . .	20
3.5. Muestra de la prueba del método de contorno con dos filtros de reducción de ruido . . . . .	21

3.6.	Muestra de la prueba del método de contorneado contemplando áreas . . . . .	22
4.1.	Resultado final en reconocimiento de imágenes . . . . .	33
4.2.	Muestra de las imágenes capturadas de los videos . . . . .	34
4.3.	Muestra del resultado de reconocimiento por video . . . . .	35
6.1.	Muestra del resultado en donde se reconocen varias coordenadas en una misma posición, mediante el método de plantilla en una imagen. . . . .	39
6.2.	Comparación del resultado en donde se reconocen varias coordenadas en una misma posición, mediante el método de plantilla en dos imágenes distintas. . . . .	40
6.3.	Muestra del resultado en donde se observa el mismo fenómeno utilizando otros tipos de plantillas. . . . .	41
6.4.	Acercamiento de la muestra del resultado en donde se observa el mismo fenómeno utilizando otros tipos de plantillas. . . . .	42

# Índice de tablas

2.1.	Características ARUBA 2930M [3] . . . . .	6
2.2.	Código del color en los LEDs respecto a su estado global en las imágenes [5] . . . . .	7
2.3.	Ventajas del lenguaje Python [10] . . . . .	10



# Capítulo 1

## Introducción

El siguiente proyecto se lleva a cabo utilizando una idea de HPE para el proyecto de graduación de ingeniería eléctrica. El objetivo principal es desarrollar un algoritmo usando Python y otras herramientas como OpenCV para la revisión automática de los LED de un interruptor, usando una transmisión de video - imágenes. Este proyecto se realiza con la intención de optimizar el tiempo de la persona en cargo de la verificación del estado de cada puerto RJ-45 mediante los indicadores de sus LEDs.

### 1.1. Alcances del proyecto

El proyecto estaba delimitado en tres partes principales, de manera que se desarrolle el algoritmo de forma funcional utilizando video: En la primera parte se pretende realizar un reconocimiento del estado de las luces LED por medio de imágenes estáticas. Dado que existen distintos factores que pueden alterar los datos, se plantea una segunda etapa en donde se analizará una entrada en formato de video. En una tercera etapa, se pone a prueba lo desarrollado en las dos secciones anteriores con video en vivo, dado que para realizar las pruebas se debe tener acceso a las cámaras, las cuales se encuentran en el laboratorio de la compañía. Como no es posible acceder a la cámara en vivo debido a la pandemia mundial del coronavirus, la compañía delimita el alcance del proyecto.

El proyecto espera realizar el reconocimiento del estado de X (cantidad) de luces LED del equipo de HPE/ARUBA, el cual consta de un commutador de red que posee una extensión generalizada de doce puertos RJ-45, con algunas diferencias de las armaduras si se compara con el commutador 2930M. Por medio de cualquier tipo de cámara se plantea el análisis de los distintos patrones, con el fin de encontrar el estado en el que se encuentra cada puerto para verificar de manera automatizada las distintas pruebas que genera la industria en su proceso de desarrollo.

### 1.2. Justificación

La importancia de este proyecto recae en la investigación que se llevará a cabo, en donde se analizan distintos métodos para el tratamiento de imágenes, los cuales se pretenden explicar teóricamente y

evidenciar experimentalmente. Puesto que el tratamiento de imágenes a nivel de automatización de procesos es una rama que no se ha desarrollado mucho a nivel de algoritmos presentes en la comunidad de Open Source, se considera este proyecto como un aporte a la misma.

### **1.3. Problema a resolver**

Un algoritmo resuelve un problema en particular, en este caso se presenta el escenario de un laboratorio, el cual, según lo indicado, no posee cambios en la iluminación y las imágenes capturas pueden contemplar cierto tipo de inclinación, así mismo puede existir modificaciones en el hardware dado que se presente que pueda analizar diferentes conmutadores.

### **1.4. Objetivos**

#### **1.4.1. Objetivo general**

Diseñar un sistema de reconocimiento de imágenes para diagnóstico de un conmutador de red.

#### **1.4.2. Objetivos específicos**

- Conocer las diferentes bibliotecas de OpenCV para el reconocimiento de patrones en imágenes.
- Entrenar un sistema para identificar puertos de red, luces de un puerto y determinar colores de las mismas.
- Identificar la presencia de uno o varios puertos de red en una imagen o frame de video.
- Reconocer las luces LEDs presentes en cada puerto.
- Determinar si las luces LEDs se encuentran apagadas o prendidas, así como el color de las mismas cuando están encendidas.

### **1.5. Metodología**

Para el desarrollo del proyecto y el alcance de los objetivos se siguió una serie de pasos, los cuales se detallan a continuación.

- Revisión de artículos y proyectos relacionados con el tema, con el fin de obtener información para realizar un diseño optimizado del sistema.
- Análisis de los datos disponibles para estudiar las características y necesidades del algoritmo.
- Desarrollo de la estructura de un algoritmo capaz de reconocer los parámetros necesarios, con el fin de identificar el estado del conmutador.

- Elaboración de pruebas preliminares para la depuración de errores en el código.
- Modificación de las incongruencias entre lo deseado y lo obtenido, para la ejecución correcta de las acciones deseadas.

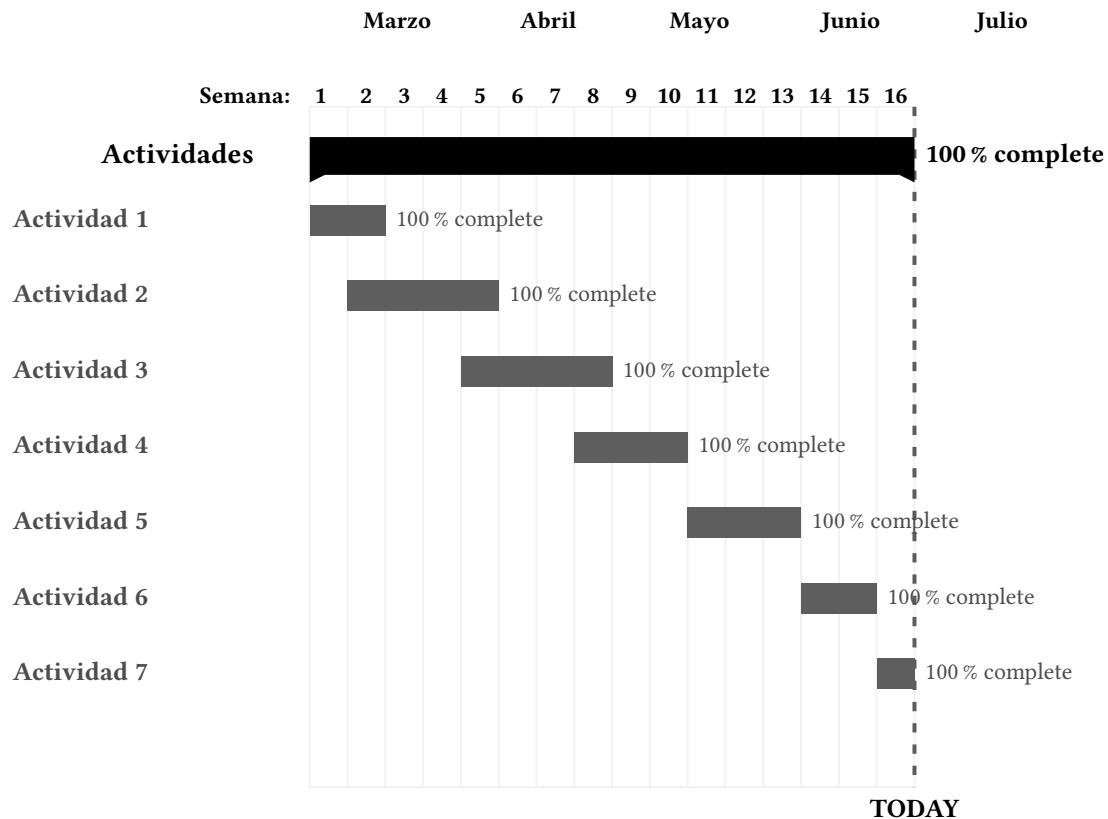
A continuación se detallará la distribución del trabajo escrito en cuanto a secciones del proyecto. Primeramente, en el capítulo 1 se detalla la introducción. En el capítulo 2 se desarrolla la explicación de la distinta teoría relacionada al tratamiento de imágenes, con el fin de reconocer patrones y tomar decisiones. Seguidamente, en el capítulo 3 se detalla el modelado y funcionamiento respecto a la elaboración del algoritmo. En el capítulo 4 se evidencian los resultados obtenidos junto con su debido análisis. Después, en el capítulo 5 se exponen las conclusiones y recomendaciones. Finalmente, en el capítulo 6 se encuentran los distintos anexos, en los cuales se agregan secciones del código.

Asimismo, el tiempo invertido es distribuido mayoritariamente en resolver el reconocimiento por medio de imágenes, debido a que las otras secciones se pueden analizar como una sucesión de imágenes por segundo.

## 1.6. Cronograma

Para completar con los dos modos de operación a implementar, así como la elaboración de la documentación, se consideran las siguientes actividades como subprocessos requeridos para completar las diferentes etapas.

- Actividad 1: Análisis de la original.
- Actividad 2: Analizar las posiciones de los LEDs en imágenes.
- Actividad 3: Categorizar los estados de cada LED en imágenes.
- Actividad 4: Modificar la imagen con los estados encontrados.
- Actividad 5: Imprimir en consola la cantidad de puertos en cierto estado.
- Actividad 6: Aplicarlo a video.
- Actividad 7: Ajuste de las guías de desarrollador y de usuario, así como ajustes en el reporte del proyecto.

**Sistema de reconocimiento de imágenes para diagnóstico de un conmutador de red**



## Capítulo 2

# Marco teórico

A nivel de estructura se pretende descomponerla en diferentes secciones, con el fin de entender cómo funcionan los dispositivos electrónicos y las herramientas de lenguaje empleadas.

### 2.1. Hardware

Se considera necesario entender qué se está analizando, por lo que se procede a definir distintas estructuras físicas.

#### 2.1.1. Comutador de red

Comutador se define como un instrumento capaz de interconectar varios segmentos de una red con el fin de vincular datos, lo que permite una conexión entre dos sitios, análogo a un puente. Esto funciona con base al ordenamiento y sistematización de datos con el fin de multiplicar las redes y los datos que se desean transmitir. Una de las cualidades más importantes es que permite mejorar la seguridad y el rendimiento de la red. Se considera que el comutador posee la característica de almacenar y asimilar las distintas rutas de los dispositivos dentro de su rango de accesibilidad por medio de los puertos [2].

#### 2.1.2. ARUBA - 2930M

Es un comutador diseñado para espacios en donde se trabaje y se diseñe inteligencia digital, son optimizados para usuarios con movilidad mediante un cable integrado lo que permite que su conexión sea inalámbrica. Se caracteriza por ser fáciles de implementar y administrar mediante herramientas propias como Aruba ClearPass Policy Manager y Aruba AirWave and cloud-based Aruba Central [3]. Su estructura física se observa en la figura 2.1.

Además, sus características principales se muestran en la tabla 2.1:



Figura 2.1: ARUBA 2930M estructura física [3]

<b>ARUBA 2930M</b>
Velocidad de cable de 10 GbE.
Enlaces de subida 40 GbE.
Fuentes de alimentación modulares dobles redundantes para hasta 1440 V de PoE.
Nuevos modelos con el estándar industrial IEEE 802.3bt clase 6 que proporciona hasta 60 W de PoE por puerto.

Tabla 2.1: Características ARUBA 2930M [3]

Respecto al código de colores presente en los LEDs de la estructura se presenta la siguiente tabla.

Estado	Significado
Verde	El puerto se encuentra en modo operacional.
Anaranjado o Anaranjado Oscuro	Puede significar una falla o bien un error al realizar la autocomprobación. Del mismo modo, si este se observara durante un periodo de tiempo prolongando en estos colores significa que hay un error fatal en el hardware.
Sin color	Representa el estado de off, por lo que no se está recibiendo corriente.

Tabla 2.2: Código del color en los LEDs respecto a su estado global en las imágenes [5]

### 2.1.3. Puerto RJ45

Un RJ45 por sus siglas en inglés Registered Jack, es el conector más usado en tarjetas de red para área local en computadores. Es importante considerar que los distintos tipos de conexión surgen de acuerdo a la forma en la que se realizan el alambrado en los cables. Este puerto se ejemplifica con la siguiente figura.



Figura 2.2: Puerto RJ45 [1]

#### HPE X121 1G SFP RJ45 T Transceiver (J8177C)

De acuerdo a lo solicitado por la empresa, se menciona que se trabaja con puertos de ethernet de 10M/100M/1G de cobre. Dado que la construcción de la estructura a analizar no es objetivo en este

proyecto no se pretende profundizar en la misma, sin embargo, esta estructura se ve como se muestra en la figura 2.3.



Figura 2.3: HPE X121 1G SFP RJ45 T Transceiver (J8177C) [4]

## 2.2. Contratiempos

El contratiempo que se pretende solucionar es poder conocer de manera automatizada el estado en el que se encuentra cada puerto, debido a que en el presente se requiere de una persona encargada de revisar estos estados ya que no se posee algún sistema interno que permita notificarlos. Por este motivo, a la compañía le surgió la idea de colocar una *webcam* (cámara web) lo que lleva a la necesidad de generar algún algoritmo de reconocimiento de patrones mediante imágenes con el fin de solucionarlo.

## 2.3. Cámara

Se indica que las imágenes serán capturas por una *webcam* sin realizar ninguna especificación, por lo que se toma en cuenta que pueden variar los siguientes aspectos:

- Grados de inclinación en relación con una imagen a otra.
- Ancho y largo de la imagen capturada, dependiendo de la cámara.
- Nitidez.
- Contraste en la imagen.
- Grado de exposición y contraste a la luz.
- Brillos y sombras en cada foto.
- Nivel de saturación, temperatura, tinta y sepia presente.

En la figura 2.4 se muestra un ejemplo de los aspectos que pueden ser modificados en cada imagen.



Figura 2.4: Variantes en las imágenes

## 2.4. Software

Se entiende como software al fundamento lógico de un conjunto informático, que comprende el acumulado de los componentes lógicos necesarios que hacen viable la ejecución de tareas específicas, en oposición a los componentes físicos que son llamados hardware. La relación entre el software y el hardware hace operativo un ordenador (u otra unidad), es decir, el software envía disposiciones que el hardware ejecuta, haciendo factible su funcionamiento.

## 2.5. Reconocimiento de imágenes/video

Con lo que respecta a lenguajes de programación para procesar imágenes se cuenta con algoritmos realizados en C, C++, Python y algunos otros. Debido a la naturaleza del proyecto y las recomendaciones realizadas por la compañía, se considera desarrollar el programa mediante el lenguaje de Python, así como la utilización de las diversas bibliotecas que posee, las cuales se detallaran en el presente documento.

### 2.5.1. Python

Python es un lenguaje de programación interpretado, cuya ideología recae en la legibilidad de su código [10]. Debido a que es un lenguaje interpretado se considera de alto nivel. Entre sus ventajas principales se encuentran las mencionadas en la tabla 2.3.

Python: Lenguaje interpretado
Es muy útil en scripting.
Posee muchas herramientas para análisis de datos y tratamiento de señales.
Python es un lenguaje portable.
Es capaz de utilizar bibliotecas libres de inteligencia digital, las cuales en su mayoría poseen una mejor velocidad en reacción.

Tabla 2.3: Ventajas del lenguaje Python [10]

Considerando lo mencionado anteriormente, se analiza la utilización de Python como lenguaje base para este proyecto.

### 2.5.2. OpenCV

OpenCV es una biblioteca de código abierto enfocada en el tema de visión artificial, inicialmente desarrollada por Intel, desde el inicio de su primera versión alfa en el mes de enero de 1999. Dentro del campo de sus aplicaciones se encuentra el área de seguridad con identificación de movimiento, inclusive aplicaciones de sistemas de control de procesos en el cual se requiere reconocimiento de objetos. Debido a que está construido bajo la licencia BSD permitiendo su uso libre en fines comerciales o de investigación, se puede utilizar este diseño. Una de sus ventajas principales es su multiplataforma ya que puede trabajarse en ambientes de desarrollo de: Windows, Android, GNU/Linux y Mac OS X. Proporciona un espacio de desarrollo fácil de utilizar y muy eficiente [12].

A continuación se explica brevemente algunas funciones presentes en esta biblioteca:

#### MedianBlur

Funciona mediante el reemplazo del elemento central de la imagen por el valor de la mediana de todos los píxeles que están en área que encierra el núcleo, al mismo tiempo permite filtrar la imagen

eliminando el ruido [9].

#### Hue value

También conocido como el valor de matiz, el cual se entiende como la diferencia de un color a otro. Se define como la variación de tono que hace un color en la lista cromática, esta variación se obtiene mediante el análisis de la longitud de onda [9]. Se mide mediante grados de  $0^\circ$  a  $360^\circ$ , o bien normalizadas de 0 % a 100 %, como se muestra en la figura 2.5.

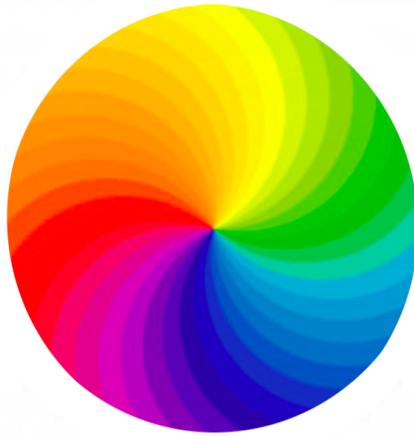


Figura 2.5: Matiz de color [9]

#### 2.5.3. NumPy

NumPy es definido como una expansión de las funciones de Python, las cuales son de código abierto y permiten implementar funciones que manipulan de una manera más fácil para el usuario matrices y vectores, permitiendo ampliar las herramientas que se poseen para manejar funciones matemáticas más complejas [11].

#### 2.5.4. Glob

Dado que para reconocer patrones así como la implementación del algoritmo debe realizarse de manera cíclica con el fin de analizar todos los archivos de la base de datos ingresada, se requiere de alguna herramienta que permita empaquetar de manera accesible las imágenes o bien el video ingresado. Por esto se considera el módulo de Glob, que permite accesar a todos los datos presentes en una ruta específica, es decir, realiza un acomodamiento en conjunto de los elementos en cierta ruta [7].

#### 2.5.5. OS

Respecto a la necesidad de modificar datos almacenados se considera el módulo de OS, que permite leer archivos, accesar rutas, así como guardar o escribir imágenes [8].

### **2.5.6. Collections**

El módulo de collections facilita el uso general de datos almacenados en contenedores especializados. Específicamente OrderedDict de la subclase dict permite poder tener presentes las entradas que fueron agregadas [6].



## Capítulo 3

# Modelado y funcionamiento

En este capítulo se recopila la información referente al modelado del algoritmo, en sus diferentes secciones y su funcionamiento.

### 3.1. Construcción

De manera general, un algoritmo se puede definir como un conjunto de pasos ordenados con instrucciones definidas y finitas de manera lógica tal, que permiten llegar a un estado final para realizar una acción determinada. En este caso, se presenta un diagrama de flujo relacionado al desarrollo de las distintas etapas con la intención de llegar a reconocer los diferentes patrones presentes en varias imágenes o bien videos.

### 3.1.1. Diagrama de flujo

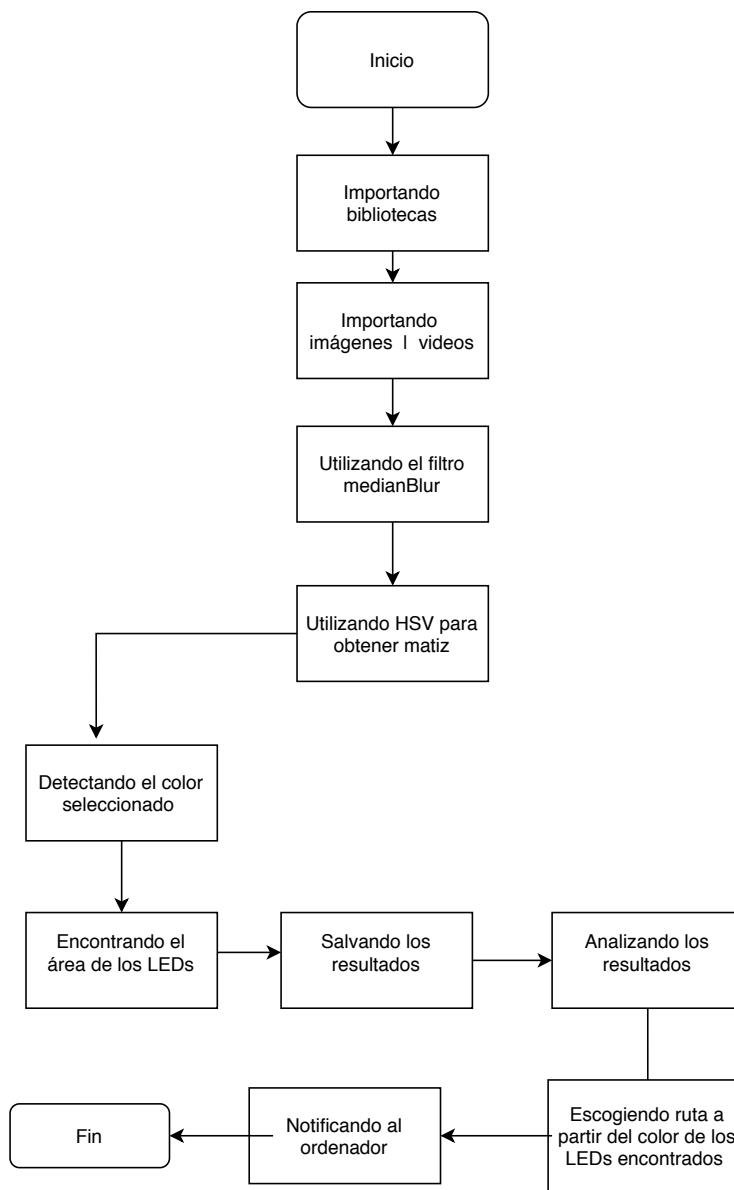


Figura 3.1: Diagrama de flujo del algoritmo a realizar [9]

## 3.2. Funcionamiento

### 3.2.1. Análisis de imágenes

#### Cambio en rutas absolutas de plantillas

Se debe modificar las rutas absolutas de las plantillas a utilizar. Esto se ubica en la sección del archivo *main.py* en donde se encuentra una parte del algoritmo similar a lo mostrado a continuación.

```
1 ##### Read the template
2 template_green = cv2.imread ('', 0)
3 template_orange = cv2.imread ('', 0)
4 template_dark_orange = cv2.imread ('', 0)
```

Se debe ingresar la dirección correspondiente a su máquina entre paréntesis, por ejemplo:  
*/Users/belindabrown/Desktop/Folder1/Images\_Recognition/Templates/ledOrange.jpg*

#### Cambio en la ubicación de las imágenes para verificar

Se debe modificar la ruta donde se encuentra la carpeta que contiene las imágenes que desea analizar. Esta parte del código es el bloque correspondiente:

```
1 ##### Directory with images verify
2 img_dir = ''
```

Debe ingresar la dirección correspondiente a su máquina entre paréntesis, por ejemplo:  
*/Users/belindabrown/Desktop/Folder1/Images\_Recognition/ImgtoVerify*

#### Algoritmo de ejecución

Para que el intérprete de Python ejecute el algoritmo, se debe ir al terminal o consola que posee su máquina, ubicar el archivo que se ejecutará, que en este caso es *main.py*, y poner las siguientes instrucciones:

```
1 <PathWhereIsMain.py USER>$ python3 main.py
```

Si se considera el ejemplo que se ha utilizado, sería el siguiente:

Ubicándonos en: */Users/belindabrown/Desktop/Folder1/Images\_Recognition*

Su computadora cambiará el nombre de lo que viene antes de \$, en este caso en particular utilizado en el ejemplo es:

```
1 Belindas-MacBook-Air: Images_Recognition belindabrown $
```

Entonces se vería así.

```
1 Belindas-MacBook-Air: Images_Recognition belindabrown $ python3
main.py
```

### 3.2.2. Análisis de video

Este algoritmo se divide en dos secciones, considerando que una de las partes del algoritmo es el reconocimiento por medio de imágenes se considera lo explicado anteriormente.

1. La primera sección consiste en dividir el video en cuadros por segundo de acuerdo con lo solicitado por el usuario. Es decir, el usuario inserta la frecuencia con la que desea capturar una imagen del video completo.
2. La segunda sección es el análisis que se implementó para el modo de operación de análisis de imagen, que se encuentra en la carpeta *Images\_Recognition*.

#### Para correr el algoritmo:

**Importante:** realice los cambios que se mencionan en los cambios de dos secciones.

Para facilitar el desarrollo del programa, se implementó un archivo makefile que sigue las siguientes instrucciones:

- Para dividir el video en cuadros por segundo, debe ingresar a la consola en una ruta como la siguiente:

```
1 <Donde se almacenan las carpetas/Video\_Recognition> $ make frames
```

- Para reconocer el estado de los puertos, se sigue lo siguiente:

```
1 <Donde se almacenan las carpetas/Video\_Recognition> $ make analize
```

#### Cambios en frame\_videos.py:

- 1. Cambiar las rutas absolutas del video que se analizará:

```
1 ##### Directory with images verify
2 img_dir = ''
3
4 Example:
5 /Users/belindabrown/Desktop/EIE_Project_stream_aruba_recognition-
   master/Video_Recognition/VdstoVerify/
6
7 img_dir = '/Users/belindabrown/Desktop/
   EIE_Project_stream_aruba_recognition-master/Video_Recognition/
   VdstoVerify/'
```

- 2. Cambiar la ruta absoluta del resultado de los cuadros:

```
1 cv2.imwrite('/%s.jpg'%for_name, img) #Need path
2
3 Example:
4 /Users/belindabrown/Desktop/EIE_Project_stream_aruba_recognition-
   master/Video_Recognition/FramestoVerify
5
6 cv2.imwrite(/Users/belindabrown/Desktop/
   EIE_Project_stream_aruba_recognition-master/Video_Recognition/
   FramestoVerify/%s.jpg'%for_name, img) #Need path
```

**Cambios en main.py:**

Antes de utilizar las instrucciones del archivo makefile, se debe realizar los cambios pertinentes con respecto a las direcciones donde se encuentra la información con la que va a trabajar.

### 3.3. Historial de decisiones

El historial del proceso de diseño del algoritmo adecuado se puede dividir en dos secciones, de acuerdo al orden cronológico de lo desarrollado primero se trabajó en el reconocimiento de patrones en imágenes, ya que esta es la base para poder analizar video. Se recuerda que el video es el conjunto de varias imágenes por segundo de algún suceso capturado, por lo que para el reconocimiento en video se consideró este hecho y se utilizó la base de reconocimiento por imágenes, ya que se implementó un algoritmo que toma un cuadro de imagen cada cierto tiempo de acuerdo a lo ingresado por el usuario.

#### 3.3.1. Historial del proceso para el análisis de imágenes

##### Métodos para caracterizar la imagen

Un algoritmo resuelve un problema particular, en este caso se presenta el escenario de un laboratorio que (según lo indicado) no presenta cambios en la iluminación y las imágenes capturadas pueden contemplar un cierto tipo de inclinación, así como puede haber modificaciones en el hardware ya que se presenta que puede analizar diferentes interruptores. Aunque una vez que se han llevado a cabo las pruebas iniciales se observa que el comportamiento de las imágenes con respecto a los colores presentes en la imagen original y considerando lo que la teoría menciona, tiende a ser un resultado inesperado. Estas pruebas se pueden resumir en la aplicación de diferentes filtros con la intención de conocer las características experimentales de las imágenes: valor de saturación de tono, filtrado con umbral superior e inferior (rango de color limitado), escala de grises, escala a gris con umbral y binario `cv2.THRESH_BINARY`.

### Métodos de identificación

Siguiendo lo anterior, se probaron diferentes métodos de identificación, entre los cuales se puede mencionar los siguientes junto con su resultado:

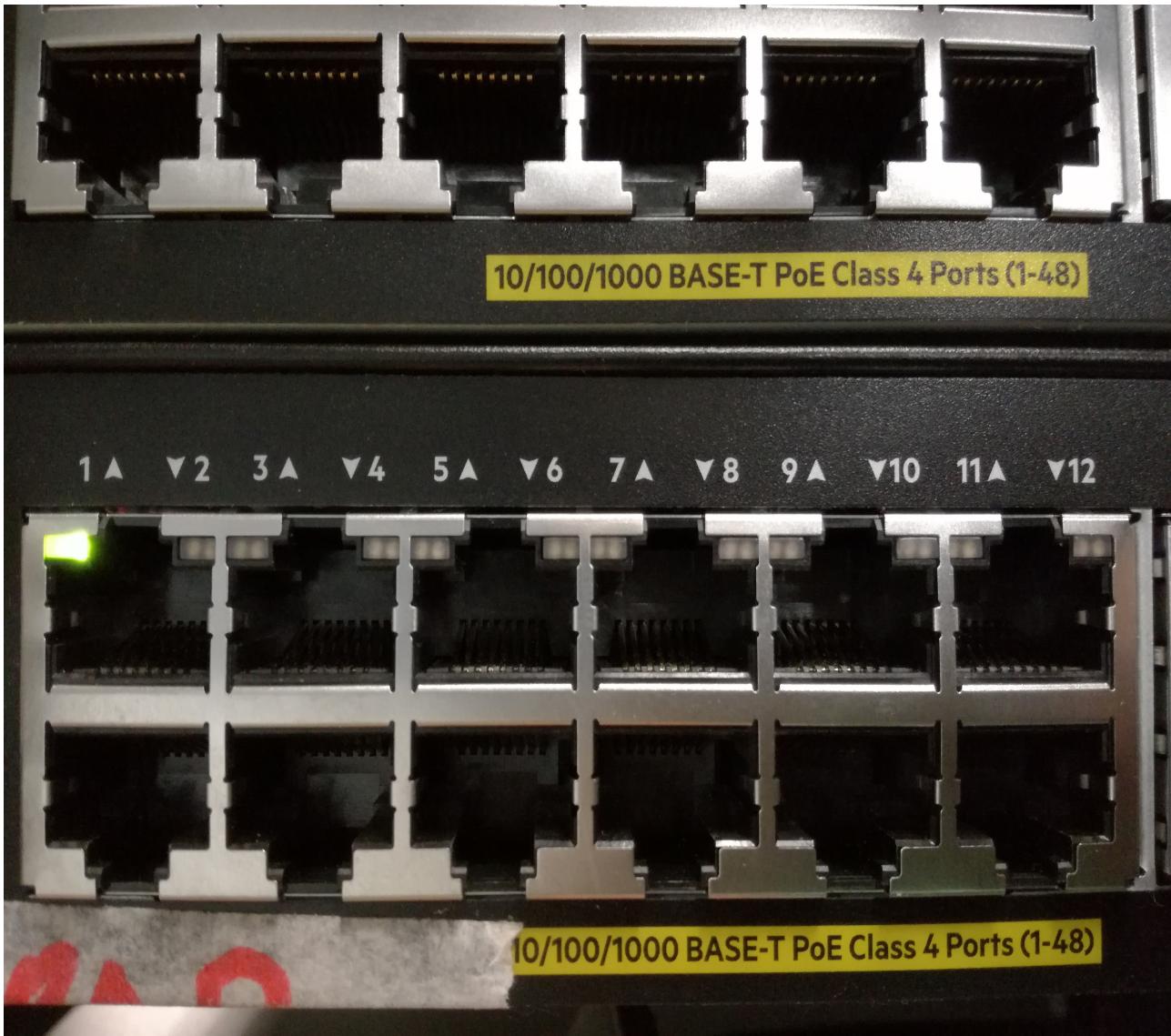


Figura 3.2: Imagen base de análisis con un led en verde.

- Por contorno:

Este método utiliza `cv2.findContours` y `cv2.drawContours` para el que obtenemos un resultado como el que se muestra en la figura 3.3:



Figura 3.3: Muestra de la prueba del método del contorno

- Contorno ajustado con eliminación de ruido:

Si se intenta mejorar el método anterior, se opta por tratar de filtrar ligeramente el ruido de la imagen, para esto se aplicó un filtro de `cv2.fastNlMeansDenoisingColored`, en donde se obtiene el resultado mostrado en la figura 3.4:



Figura 3.4: Muestra de la prueba del método del contorno con reducción de ruido

- Usando el método de contorno con dos filtros (*fastNlMeansDenoisingColored*):

Cuando se visualiza esta prueba, se compara este resultado con el resultado de método **contorno ajustado con eliminación de ruido** 3.3.1, a partir de esto se aprecia que se pierde nitidez al mismo tiempo que se pierde calidad en la imagen, como se muestra en la figura 3.5.



Figura 3.5: Muestra de la prueba del método de contorno con dos filtros de reducción de ruido

- Contorneado pero capturando áreas:

En este caso se realizan ciertas consideraciones, cuyo resultado se muestra en la figura 3.6:

\* 1 \* El umbral de la imagen por medio de `cv2.threshold` y `cv2.findContours`.

\* 2 \* Un área mínima haciendo uso de `cv2.minAreaRect` que devuelve un rectángulo giratorio, y por medio de `cv2.boxPoints` se puede acceder a los píxeles que se forman dentro del rectángulo, puesto que

devuelve las coordenadas de los puntos de esquina del rectángulo girado.

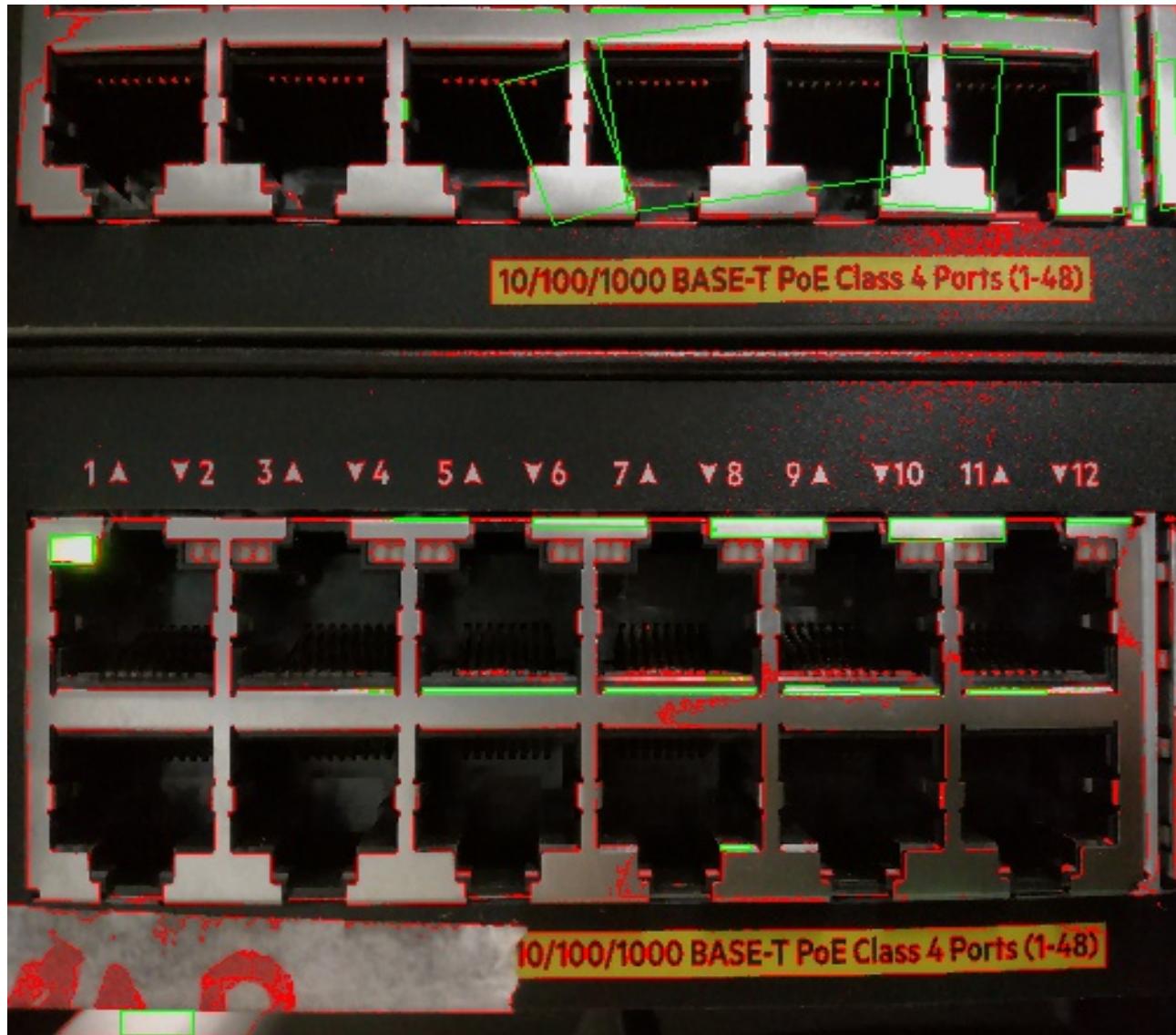


Figura 3.6: Muestra de la prueba del método de contorneado contemplando áreas

- Método de plantilla:

Este proceso se puede dividir en dos etapas:

#### Etapa 1

No funcionó como se esperaba, ya que la documentación recomendaba ciertos valores en diferentes parámetros que no permitían ajustar el reconocimiento necesario, por esto los parámetros fueron probados tanto experimentalmente, como por historial de los métodos anteriores. Se finalizó con la obtención de los valores óptimos para estas imágenes, como es el caso del valor de umbral.

Se recuerda que el valor de este umbral debe ser un valor normalizado, es decir, entre 0 y 1, donde un valor más cercano a uno nos da una mayor precisión.

### **Etapa 2**

Lo que hace este método es considerar en cada píxel y el umbral establecido los valores que cumplen las condiciones. De esto se obtiene como resultado una gran cantidad de píxeles que calzan con la identificación del template de LED en cierto estado. Por ejemplo, aplicando este método sobre la imagen mostrada en la figura 3.2 considerada como base de explicación, para el primer LED en el cual se puede observar que los pares ordenados de la localización del estado ENCENDIDO para el primer puerto son: (145,1678), (146, 1678), (147,1679), (148,1677), (149,1677), (150,1676), (151,1677), (152,1677), (153,1678), (154,1675), (1515,1677), (156,1677) esto no es deseado, por lo tanto, se realizaron varios cálculos aritméticos con el fin de poder filtrar las coordenadas obtenidas a un solo par por localización.

Finalmente, se generó el algoritmo presente en main.py, capaz de analizar los diferentes estados de los LED desarrollados a partir del estudio y la implementación, ya sea para resultados positivos como pruebas no concluyentes, quienes brindaron conocimiento sobre las características del reconocimiento de los objetos deseados. Se considera que los colores presentes en la imagen no presentan cambios abruptos, excepto por las posibles intensidades de luz que pueden ser recogidas cuando un puerto está activado.

Para explicar algunas decisiones tomadas y profundizar en la explicación en el archivo main.py, se procederá a contextualizar diferentes decisiones.

**\*\*\* Filtrar ruido / Hacer que la imagen sea nítida:** Usando el umbral de Kittler usando la herramienta previamente creada en [https://github.com/brown9804/Image\\_Segmentation\\_Project](https://github.com/brown9804/Image_Segmentation_Project)- se obtiene una variación de 1: 15.39, por lo que se utilizan valores de 15 para las imágenes, puesto que se consideran de laboratorio sin cambios en la iluminación.

Teniendo en cuenta las diferentes pruebas, se consideran los valores 15 15 7 15 (aproximadamente 30 s), a pesar de que la literatura leída recomienda valores de 10 10 7 21 (aproximadamente 1:42 minutos). Esto se mantiene en el código general porque puede ser necesario analizar imágenes que, debido a la falta de nitidez, hagan que el programa no funcione completamente.

```

1 # def denoising_sharpening(input):
2 #     sin_ruido= cv2.fastNLMeansDenoisingColored(input, None
3 #         ,15,15,7,15)
4 #     kernel=np.array([[ -1, -1, -1, -1, -1],

```

```

5 # [-1, 2, 8, 2, -1],
6 # [-2, 2, 2, 2, -1],
7 # [-1, -1, -1, -1, -1]])/8.0
8 # sin_ruido = cv2.filter2D(sin_ruido, -1, kernel)
9 # return sin_ruido

```

**\*\*\* Operación de comparación usando cv2.matchTemplate ():** La coincidencia de plantillas (*cv2.matchTemplate*) es un método para buscar y encontrar la ubicación de una imagen de plantilla en una imagen más grande. OpenCV viene con una función *cv2.matchTemplate()* para este propósito. Funciona por medio de deslizar la imagen de la plantilla sobre la imagen de entrada (como en convolución 2D), comparando la plantilla y el parche de la imagen de entrada debajo de la imagen de la plantilla. Combinando varios métodos de comparación de OpenCV, se obtiene una imagen en escala de grises, donde cada píxel indica qué tan cerca coincide el vecindario de ese píxel con la plantilla. *.TM\_CCOEFF\_NORMED* hace el coeficiente de correlación. El método simplemente se usa para:

- a) Hacer que la plantilla y la imagen sean cero.
- b) Hacer las partes oscuras de los valores negativos de la imagen y las partes brillantes de los valores positivos de la imagen.

Por ejemplo, dentro del código encontrará:

```

1 res_matching_green = cv2.matchTemplate(<image in gray>, <template
2     that is going to used>, cv2.TM_CCOEFF_NORMED)
3
4 Example:
5 res_matching_green = cv2.matchTemplate(img_gray, template_green, cv2.
6     TM_CCOEFF_NORMED)

```

Cuyos parámetros son: imagen analizada convertida a escala de grises, imagen del objeto que desea encontrar, método a utilizar.

#### **\*\*\* Obtenga la posición donde se obtuvo un píxel similar al objeto:**

Esto se hace usando *np.where ()*, quien devuelve elementos donde se cumple la condición implementada, en este caso devuelve una tupla con dos matrices, una para la coordenada x y otra para la coordenada Y.

#### **\*\*\* Ordenar los pixeles obtenidos para relacionarlos con las posiciones de los LEDs de cada puerto:**

*Sorted ()* es un método que devuelve una lista ordenada del objeto iterable especificado, por lo que se aplica a la coordenada x.

```

1 for itergreenx in sorted(location_green[0]):
2     if itergreenx not in Green_x:
3         Green_x.append(itergreenx)

```

Dado que la intención de este método es comparar la diferencia entre la coordenada anterior, se genera una matriz que contiene el mismo número de elementos que la matriz original, se elimina la primera coordenada y se copia la última, lo que permite la siguiente lógica, la cual se verá ejemplificada, tomando como base de análisis la imagen mostrada en la figura 3.2:

Se obtiene un arreglo con las coordenadas (144, 145, 146, 147, 767, 768, 769, 771, 998, 1000, 1001) con la siguiente sección de código, se genera un segundo arreglo con las coordenadas (145, 146, 147, 767, 768, 769, 771, 998, 1000, 1001). Si se copia la primera posición entregada por el método en un nuevo arreglo, eliminando este valor del arreglo original podemos comparar dos arreglos con la misma cantidad de items y calcular la diferencia entre las coordenadas existentes.

Esto sucede en la coordenada x:

```

1 ##### Copying the vector without repetitions to generate the
   second to compare
2 X_Green_before_filtered = Geen_x.copy()
3 ##### Obtaining the first coordinate
4 xGreen0 = Geen_x[0]
5 ##### Deleting the first coordinate
6 X_Green_before_filtered.pop(0)
7 ##### The deleted coordinate is added to the result
8 X_Green_Filtered.append(xGreen0)
```

Se aplica a la coordenada Y, el mismo método de sorted () .

```

1 ##### Green before filtering for y - basically vector obtained
   minus repeated coordinates
2 for itergreeny in sorted(location_green[1]):
3     if itergreeny not in Green_y:
4         Green_y.append(itergreeny)
```

Al igual que en la coordenada x, se implementa el mismo procedimiento para la coordenada y solo en este caso, valores de (1670,1671,1660,1670...), con diferencias muy bajas entre ellos, teniendo sentido ya que estas diferencias son un producto de la ángulo de inclinación existentes en la toma de las imágenes.

```

1 ##### Copying the vector to generate the second
2 Y_Green_before_filtered = Green_y.copy()
3 ##### Gets the first coordinate obtained from the list of elements
   without repetitions
4 yGreen0 = Green_y[0]
5 ##### Deleting the first element to be able to subtract with the
   complete list
6 Y_Green_before_filtered.pop(0)
7 ##### The deleted coordinate is added to the result
8 Y_Green_Filtered.append(yGreen0)
```

Teniendo en cuenta que la estructura del algoritmo ya logra filtrar las coincidencias (el motivo por el cual se requiere filtrar es ya que solo debe existir un píxel por posición de LED), esta parte del código muestra dónde se calcula la diferencia y compara esta diferencia entre píxeles adyacentes con el valor del ancho y la longitud de cada plantilla.

```

1 ##### GREEN COORDENATES FILTERS ONE FOR LOCATION
2 ######
3 #For green Y
4 for e, i in sorted(zip(Green_y, Y_Green_before_filtered)):
5     #Difference between y coordinates
6     Diff_Y_Green = i - e
7     if h_green < abs(Diff_Y_Green):
8         Y_Green_Filtered.append(i)
9 Y_Green_Filtered = list(OrderedDict.fromkeys(Y_Green_Filtered))
# Filter for the X
10 for ee, ii in sorted(zip(Green_x, X_Green_before_filtered)):
11     #Difference between x coordinates
12     Diff_X_Green = ii - ee
13     if w_green < abs(Diff_X_Green):
14         X_Green_Filtered.append(ii)
15 X_Green_Filtered = list(OrderedDict.fromkeys(X_Green_Filtered))

```

Como se obtienen valores muy similares para la coordenada Y, contamos el número de elementos filtrados en el arreglo de coordenadas X y duplicamos la primera coordenada Y hasta que la cantidad de los elementos sea igual al tamaño del arreglo de coordenadas de Y y de las coordenadas X sea el mismo.

Finalmente se obtiene las coordenadas filtradas, siendo esto lo deseado.

```

1 #Counting the number of pixels each coordinate
2 number_X_Green = len(X_Green_Filtered)
3 number_Y_Green = len(Y_Green_Filtered)
4 HowManyGreen = 0
5 #Equalizing in order the number of pixels Y
6 while HowManyGreen < number_X_Green - 1:
7     HowManyGreen = HowManyGreen + 1
8     if number_Y_Green != number_X_Green and number_Y_Green <
9         number_X_Green:
10         Y_Green_Filtered.append(yGreen0)
10 ##### Joining the two x, y coordinates
11 Green_filtered_coordenates = sorted(zip(X_Green_Filtered,
Y_Green_Filtered))

```

**\*\*\* Dibujar en la imagen al mismo tiempo permite contar la cantidad de LED en un estado:**

```

1 ##### Draw the rectangle and label in that case where it finds
   green
2 for ptGreen in Green_filtered_coordenates:
3 #### cv2.rectangle(image where draw, place , color BGR, thick
   line drawing)
4 cv2.rectangle(img, ptGreen, (ptGreen[0] + w_green, ptGreen[1] +
   h_green), (0,255,255), 4)
5 ### In this function the color goes BGR, what it does is put
   the text where it found the led
6 cv2.putText(img, 'GREEN', ptGreen, cv2.FONT_HERSHEY_TRIPLEX, 1, (0,
   255, 255), 4)
7 ##### Count the number of LEDs you found in this state
8 Quantity_Leds_Green = Quantity_Leds_Green +1
9 print("The number of LEDs in Green Green status (on / on) found is:
   ", Quantity_Leds_Green)

```

Con el fin de explicar los parámetros de `cv2.putText()`, se sigue que los parámetros corresponden a lo siguiente: imagen donde se dibujará, texto para agregar, píxel reconocido de la plantilla para identificar, fuente, escala de fuente, color (BGR), grosor de la línea.

**\*\*\* Cambiar el rango del eje x reconocimiento:** Esta etapa (mostrada a continuación) se usa para relacionar la posición de los puertos, en donde se toma una plantilla de dos puertos y para obtener su cantidad total se multiplica por dos. Como la plantilla se identifica comenzando en la posición x0, la esquina superior izquierda se considera un rango ajustable para esta relación.

```

1 print("Positions of the leds ON found", leds_on_fnd)
2 number_label = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
3 loc_led_templ = sorted(zip(loc_full_port, number_label))
4 diff = [] # For the result of the sub created for filter positions
5 for i in sorted(loc_led_templ):
6     for j in sorted(leds_on_fnd):
7         z = j + 20
8         x = i[0] - z
9         if x in range(-150,70):
10            if j in X_Green_Filtered:
11                print("Port", i[1], "status:
12                                Green")
13                elif j in X_YellowOrange_Filtered:
14                    print("Port", i[1], "status:
15                                Orange")
16                elif j in X_OrangeOrange_Filtered:
17

```

15                   print("Port", i[1], "status:

Para ajustar los parámetros, los valores se modifican en:

1 if x in range(-150,70):

Se recomienda modificar el primer valor siendo este -150, disminuyendo este valor hacia  $-\infty$ .

### 3.3.2. Historial del proceso para el análisis de videos

Se toma en cuenta el mismo historial antes mencionado, dado que la diferencia es el valor de ajuste de los parámetros de umbral y el archivo llamado como *frames.py*

## Capítulo 4

# Resultados y análisis

## 4.1. Resultados del análisis de imágenes

Es necesario contemplar que, dada la forma en que se planteó el problema, la imagen analizada y sus características cualitativas se muestran en IDLE, además, en el terminal se muestra la descripción cuantitativa de sus propiedades.

### 4.1.1. Resultados en la terminal

Dependiendo de lo que contenga la imagen, se obtendrán diferentes resultados que se presentan en un formato como el siguiente. Esto sucede después de aplicar lo descrito en la sección **Algoritmo de ejecución** 3.2.1.

```
1 /Users/belindabrown/Desktop/EIE_Project_stream_aruba_recognition-
    master/Images_Recognition/ImgtoVerify/3YellowOrange.jpg
2 Image loaded, analyzing patterns ...
3 The number of LEDs in Yellow Orange state (on / problem) found is:
    3
4 The number of ports are:           12
5 Positions of the leds ON found [149, 814, 2124]
6 Port 1 statuts:                  Orange
7 Port 3 statuts:                  Orange
8 Port 7 statuts:                  Orange
9
10 /Users/belindabrown/Desktop/EIE_Project_stream_aruba_recognition-
     master/Images_Recognition/ImgtoVerify/1Mixed.jpg
11 Image loaded, analyzing patterns ...
12 The number of LEDs in Green Green status (on / on) found is:
    1
13 The number of LEDs in Yellow Orange state (on / problem) found is:
    1
```

```

14 The number of LEDs in Orange Orange status (problem / problem)
    found is:      1
15 The number of ports are:      12
16 Positions of the leds ON found [140, 800, 1446]
17 Port 1 statuts:           Green
18 Port 3 statuts:           Orange
19 Port 5 statuts:           Dark Orange
20
21 /Users/belindabrown/Desktop/EIE_Project_stream_aruba_recognition-
     master/Images_Recognition/ImgtoVerify/4YellowOrange.jpg
22 Image loaded, analyzing patterns ...
23 The number of LEDs in Yellow Orange state (on / problem) found is:
     4
24 The number of ports are:      12
25 Positions of the leds ON found [149, 814, 2124, 3434]
26 Port 1 statuts:           Orange
27 Port 3 statuts:           Orange
28 Port 7 statuts:           Orange
29 Port 11 statuts:          Orange

```

#### 4.1.2. Resultados de Python IDLE

Unos segundos más tarde se mostrará (dependiendo de los casos y la cantidad de LEDs que están encendidos) una ventana con una imagen analizada y para continuar con la siguiente imagen se debe presionar cualquier tecla.

\*\* Nota: \*\* Si se muestra una imagen ampliada, considere hacer doble clic en la barra superior de la ventana para que la imagen se reajuste.

#### Configuraciones

Si consideramos el algoritmo desarrollado a nivel de reconocimiento se pueden hacer varios cambios:

##### \*\*\* Cambio del objeto u objetos a reconocer:

- **1.1 La plantilla se modifica:** Una vez que se ha decidido qué objetos se van a identificar, estas imágenes se almacenan en una carpeta exclusiva para plantillas, estas imágenes en todo el código se denominan plantillas.
- **1.2 Modificar imágenes para verificar:** Si desea cambiar lo que se identifica en las imágenes, se recomienda almacenar primero todas las imágenes en una carpeta. Siguiendo el modelo de reconocimiento presente en esta sección.

- **1.3 Cantidad de objetos:** El número de objetos que identifica son 3 (estados diferentes), pero si desea aumentar o disminuir el número de elementos, la idea sería eliminar o agregar. En cualquiera que sea el caso, llame a la función nuevamente pero ingrese los parámetros correspondientes.

```

1 res_matching_green = cv2.matchTemplate(<image in gray>,<template
2   that is going to used>,cv2.TM_CCOEFF_NORMED)
3
3 Example:
4 res_matching_green = cv2.matchTemplate(img_gray,template_green,cv2.
5   TM_CCOEFF_NORMED)
```

Agregue esta línea siguiendo el orden:

```
1 location_green = np.where(res_matching_green >= threshold)
```

Llame a la función como este ejemplo:

```
1 X_Green_Filtered0 = color_filter(location_green, w_green, h_green,
2   img, 'GREEN')
```

Se verifica si la matriz está vacía en:

```

1 # Checking empty
2 if X_Green_Filtered0:
3     X_Green_Filtered = list(OrderedDict.fromkeys(
4       X_Green_Filtered0))
5 else:
6     X_Green_Filtered = []
```

Se agregan los píxeles en la lista ordenada para reconocer las posiciones de los puertos y relacionarlos con los LEDs para obtener el estado de cada puerto.

```
1 leds_on_fnd = sorted(list(OrderedDict.fromkeys(
2   X_YellowOrange_Filtered + X_Green_Filtered +
3   X_OrangeOrange_Filtered)))
```

Finalmente, se debe implementar otro elif como el del siguiente ejemplo:

```
1 elif j in X_OrangeOrange_Filtered:
2     print("Port", i[1], "status:
```

**\*\*\*\* Ajuste del umbral:** Ejecute el programa y verifique si se identifica bien, si no ajusta el umbral. Actualmente está en 0.90 recordando que su valor está en un rango entre 0 - 1, cuanto más se acerca a la precisión.

```
1 ##### Specifying (threshold)
2 threshold= 0.90
3 thresholdport= 0.70
```

### \*\*\* Modificaciones a la imagen

- **3.1. Texto (etiquetas de objeto reconocidas):** Puede modificar el formato del texto que se agrega en la imagen, para los casos anteriores tenemos GREEN, YELLOW ORANGE, ORANGE, estas etiquetas que se agregan en la imagen tienen un formato de color, letra y grosor de línea.

```

1 ##### In this function the color goes BGR, what it does is put
   the text where it found the led
2     cv2.putText(img, 'GREEN', ptGreen, cv2.FONT_HERSHEY_TRIPLEX
               , 1, (0, 255, 255), 4)

```

Cuando se sigue el orden en que se encuentra la sintaxis de `cv2.putText()`, se deduce que los parámetros corresponden a lo siguiente: *imagen donde se dibujará, texto para agregar, píxel reconocido de la plantilla para identificar, fuente, escala de fuente, color (BGR), grosor de la línea*.

- **3.2. Formato de redondeo:** En este caso, dada la forma de los objetos a reconocer, se decidió implementar un contorno rectangular. Por la razón anterior, se utiliza `cv2.rectangle`, aunque si desea que sea circular, se recomienda buscar en la documentación de OpenCV `cv2.circle`.

```

1 ##### cv2.rectangle(image where draw, place , color BGR, thick
   line drawing)
2 cv2.rectangle(img, ptGreen, (ptGreen[0] + w_green, ptGreen[1] +
   h_green), (0,255,255), 4)

```

- **Mensajes en pantalla:** Imprime los resultados de 3 secciones del proceso, nombre de la imagen analizada, un anuncio de que la imagen ha sido cargada, esto nos muestra que el proceso de adquisición de los datos de dicha imagen fue exitoso y finalmente imprime el resultado de cuántos LEDs hay en cada estado, siendo estos:

#### \*\*\* Primera sección \*\*\*

```
1 print("\n", f1) #picture name
```

#### \*\*\* Segunda sección \*\*\*

```
1 ##### Announces every time an image is reviewed
2 print("Image loaded, analyzing patterns ...")
```

#### \*\*\* Tercera sección \*\*\*

```

1 print("The number of LEDs in Green Green status (on / on) found is:
          ", Quantity_Leds_Green)
2 print("The number of LEDs in Yellow Orange state (on / problem)
          found is:      ", Quantity_Leds_YellowOrange)
3 print("The number of LEDs in Orange Orange status (problem /
          problem) found is:      ", Quantity_Leds_OrangeOrange)

```

Resultando en:

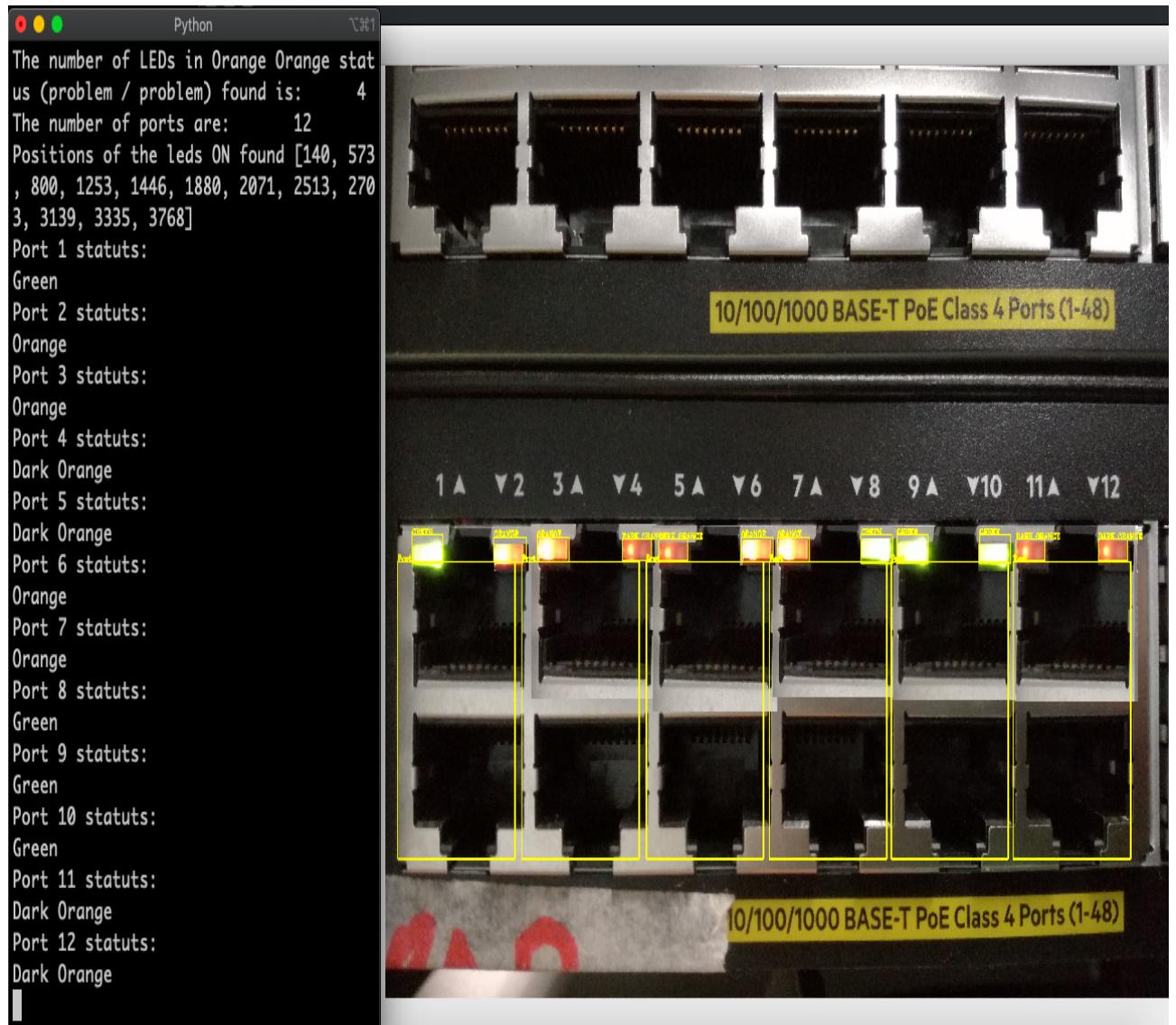


Figura 4.1: Resultado final en reconocimiento de imágenes

## 4.2. Resultados del análisis de videos

Se obtuvieron como resultados un folder con n cantidad de imágenes las cuales fueron tomadas de cada video, como se muestra a continuación:

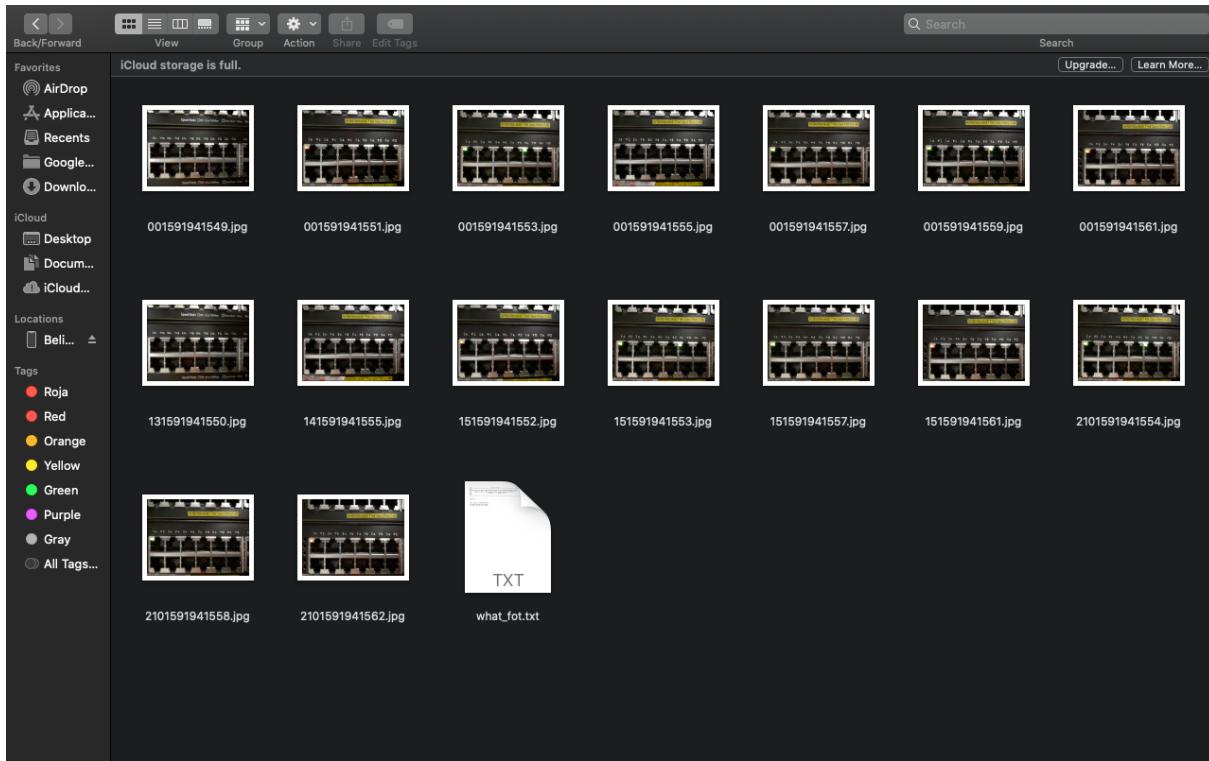


Figura 4.2: Muestra de las imágenes capturadas de los videos

Finalmente se obtiene el siguiente reconocimiento:

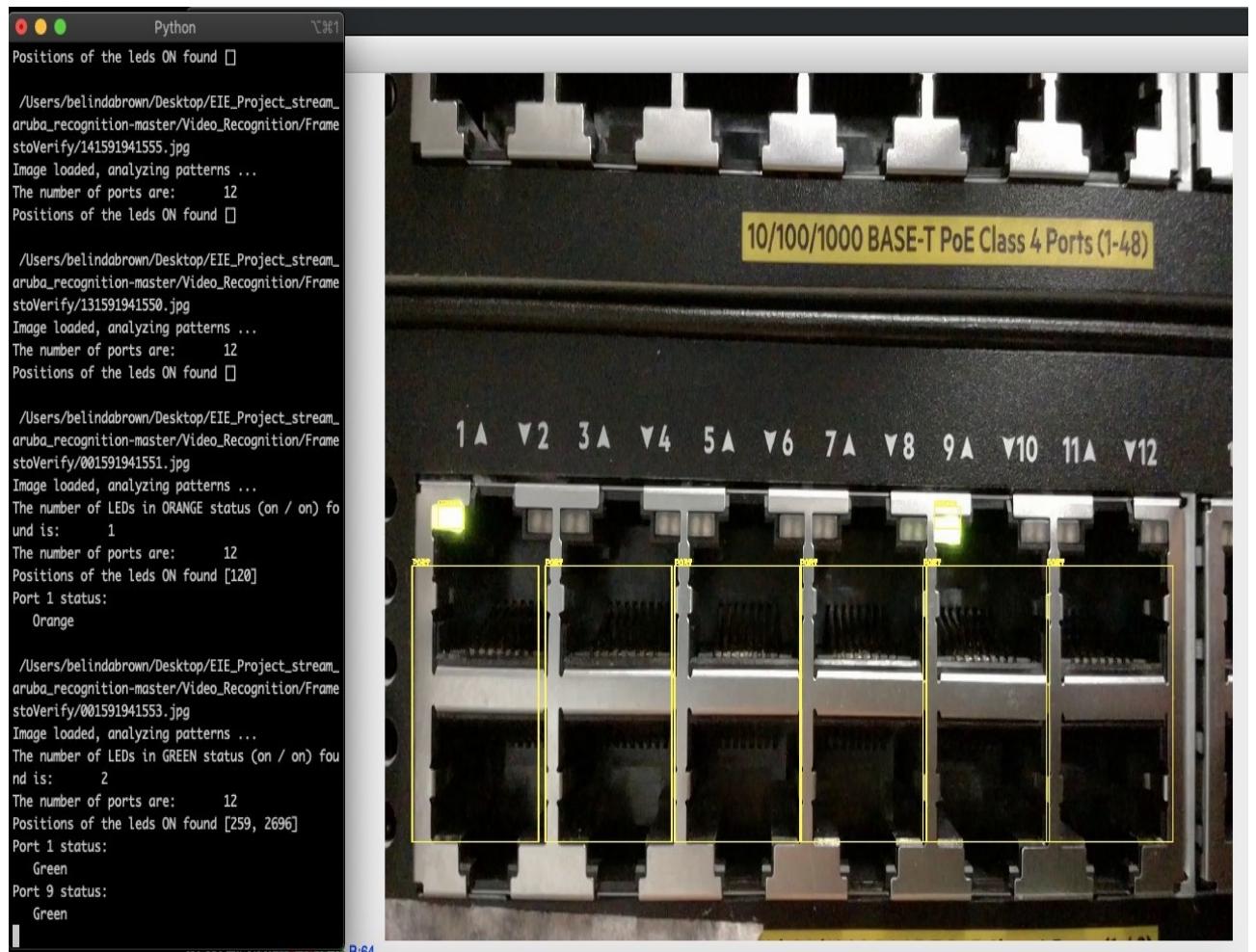


Figura 4.3: Muestra del resultado de reconocimiento por video

### Consideraciones necesarias

- Lectura detallada del **capítulo 3** de este documento, debido a que en este capítulo se comenta con detalle cada decisión o procedimiento fundamental para un análisis adecuado de lo solicitado.
- Es necesario considerar el umbral como principal parámetro de ajuste, así como el rango mencionando en el **capítulo 3** con la finalidad de adecuar las condiciones de la imagen a un correcto análisis.



## Capítulo 5

# Conclusiones y recomendaciones

### 5.1. Conclusiones

- Se diseñó un sistema de reconocimiento para diagnóstico de un conmutador de red, el cual considera ángulos de inclinación respectos al marco de referencia entre una imagen y otra, métodos de automatización capaces de calcular posiciones describiendo el estado de cada puerto y ajuste a la iluminación variable del entorno.
- La versión 4.1.2 de OpenCV fue la empleada para el reconocimiento de algunas características o acciones sobre las imágenes. Sin embargo, posee un alto grado de error con relación a la cantidad de objetos reconocidos en una imagen (error de conteo).
- Con respecto al entrenamiento de un sistema para identificar luces y puertos de red se entrenó por medio del método de OpenCV *matchTemplate*, se realizó una cascada de filtros para contar de manera adecuada y se relacionó el comportamiento de los LEDs respecto a la localización del puerto y para describir de manera satisfactorio el estado del puerto.
- El reconocimiento tanto de los puertos como LEDs se realizó de manera exitosa por medio del algoritmo implementado.
- Para determinar si las LEDs se encontraban apagadas o encendidas se empleó el método de *matchTemplate*, y se crearon los siguientes parámetros: rango asociado a la localización y el valor de umbral respecto a las características del entorno, además se diseñaron los filtros en cascada.

### 5.2. Recomendaciones

- En primer lugar, ante el resultado de las diferentes funciones de reconocimiento de imágenes de la versión 4.1.2 de OpenCV se recomienda el uso de los rangos y los filtros de cascada que fueron diseñados.
- En segundo lugar, es necesario ajustar el umbral y el rango elegido, los cuales corresponden a la última sección del archivo con el nombre *main.py* en cualquier modo de operación (imágenes o video).

- En tercer lugar, durante el desarrollo de un proyecto para un cliente es necesario mantener una comunicación constante en ambos sentidos, en donde se comuniquen las necesidades, inquietudes y metodologías específicas para entregar un producto acorde con las necesidades planteadas.
- Finalmente, es pertinente acotar que dentro de este proyecto se utilizaron conocimientos básicos de la ingeniería eléctrica, obtenidos a lo largo de la carrera; sin embargo, el desarrollador debe investigar por su cuenta todo aquello que se necesite para la construcción de un proyecto.

# Capítulo 6

# Anexos

```
bash ● #1 bash
Pares de coordenadas para punto naranja [(172, 1551), (171, 1553), (174, 1552), (173, 1550), (170, 1552), (170, 1550), (173, 1554), (171, 1551), (175, 1552), (171, 1552), (174, 1553), (176, 1552), (173, 1553), (171, 1550), (172, 1552), (176, 1550), (177, 1552), (177, 1551), (176, 1553), (168, 1550), (169, 1551), (172, 1549), (174, 1549), (170, 1551), (173, 1553), (171, 1550), (172, 1552), (176, 1550), (177, 1552), (177, 1551), (176, 1553), (168, 1550), (169, 1551), (170, 1549), (174, 1549), (173, 1553), (172, 1552), (173, 1553), (169, 1553), (176, 1551), (174, 1550), (171, 1549), (168, 1552), (175, 1551), (168, 1551), (171, 1549), (169, 1550), (172, 1550), (170, 1549), (173, 1551), (172, 1554), (169, 1552), (174, 1554), (174, 1551), (175, 1553)]Pares de coordenadas para punto naranja set()
Leds en VERDE: 95
Leds en NARANJA: 160
Leds en Naranja OSCURO: 29
/Users/belindabrown/Desktop/Reconocimiento13/ImgAVerificar/off5.jpg

Imagen cargada, Analizando patrones...
Pares de coordenadas para punto verde set()
Pares de coordenadas para punto naranja [(181, 1582), (176, 1579), (178, 1581), (182, 1581), (177, 1578), (179, 1583), (183, 1580), (177, 1581), (179, 1579), (181, 1581), (176, 1581), (178, 1578), (180, 1580), (178, 1582), (179, 1582), (182, 1582), (177, 1580), (184, 1580), (179, 1578), (181, 1580), (178, 1579), (176, 1581), (180, 1581), (175, 1581), (178, 1583), (180, 1578), (181, 1579), (182, 1579), (179, 1581), (176, 1582), (180, 1582), (177, 1583), (175, 1580), (181, 1583), (178, 1580), (182, 1580), (177, 1579), (180, 1579), (175, 1579), (181, 1578), (179, 1580), (180, 1583), (183, 1581), (177, 1582)]Pares de coordenadas para punto naranja set()
Leds en VERDE: 95
Leds en NARANJA: 204
Leds en Naranja OSCURO: 29
/Users/belindabrown/Desktop/Reconocimiento13/ImgAVerificar/off6.jpg

Imagen cargada, Analizando patrones...
Pares de coordenadas para punto verde [(152, 1582), (154, 1585), (153, 1583), (149, 1585), (148, 1583), (151, 1586), (149, 1582), (157, 1585), (158, 1584), (153, 1589), (148, 1585), (154, 1586), (150, 1588), (149, 1588), (156, 1585), (151, 1585), (157, 1588), (150, 1582), (156, 1583), (151, 1583), (153, 1584), (155, 1589), (150, 1585), (160, 1585), (151, 1588), (152, 1586), (159, 1586), (147, 1584), (153, 1587), (155, 1584), (146, 1584), (150, 1586), (158, 1588), (152, 1583), (159, 1585), (147, 1583), (153, 1582), (149, 1584), (157, 1585), (151, 1582), (156, 1583), (158, 1585), (153, 1588), (147, 1584), (155, 1587), (156, 1586), (154, 1587), (156, 1586), (154, 1582), (151, 1584), (157, 1587), (155, 1583), (150, 1583), (151, 1582), (158, 1583), (158, 1585), (153, 1588), (159, 1588), (148, 1586), (154, 1586), (150, 1587), (153, 1586), (155, 1587), (146, 1585), (150, 1587), (156, 1588), (152, 1584), (159, 1584), (146, 1586), (148, 1587), (154, 1584), (149, 1586), (160, 1586), (152, 1587), (153, 1586), (155, 1587), (146, 1585), (150, 1587), (156, 1588), (152, 1584), (159, 1584), (146, 1586), (148, 1587), (154, 1584), (149, 1586), (156, 1587), (154, 1583), (151, 1587), (149, 1583), (157, 1586), (155, 1582), (158, 1587), (148, 1584), (154, 1589), (156, 1584), (160, 1587), (156, 1582), (157, 1583), (152, 1583), (157, 1587), (152, 1588), (147, 1586), (153, 1585), (155, 1586), (150, 1584), (156, 1589), (152, 1585), (159, 1587), (147, 1585), (155, 1585)]Pares de coordenadas para punto naranja set()
Pares de coordenadas para punto naranja set()
Leds en VERDE: 188
Leds en NARANJA: 204
Leds en Naranja OSCURO: 29
/Users/belindabrown/Desktop/Reconocimiento13/ImgAVerificar/on2.jpg
```

Figura 6.1: Muestra del resultado en donde se reconocen varias coordenadas en una misma posición, mediante el método de plantilla en una imagen.

The screenshot displays two side-by-side software interfaces, likely from different versions of a program. Both interfaces have a top bar with various icons for View, Zoom, Share, Highlight, Rotate, Markup, and Search.

**Left Interface:**

- Shows a list of coordinates with their values: p[242]: 0.000571, p[243]: 0.000578, p[244]: 0.000561, p[245]: 0.000528, p[246]: 0.000532, p[247]: 0.000532, p[248]: 0.000581, p[249]: 0.000624, p[250]: 0.000828, p[251]: 0.001161, p[252]: 0.002237, p[253]: 0.006889, p[254]: 0.005739, p[255]: 0.002321.
- Statistics: Valor medio: 66.109304, Valor medio cuadrático: 8587.359177, Varianza: 4216.919056.
- Section header: Datos obtenidos para umbralización Kittler.
- Values: c1: 0.000596, c2: 0.999404, varianza1: 0.000000, varianza2: 4216.825980, m1: 0.000000, m2: 66.148716, thOp: 0.
- Message: Saving unsigned char image in: output/imagenSegmentada.yuv
- Command: rm a.exe
- Path: Belindas-MacBook-Air:Image\_Segmentation\_Project--master belindabrown\$

**Right Interface:**

- Shows a list of coordinates with their values: p[242]: 0.000464, p[243]: 0.000436, p[244]: 0.000425, p[245]: 0.000418, p[246]: 0.000442, p[247]: 0.000506, p[248]: 0.000588, p[249]: 0.000568, p[250]: 0.000722, p[251]: 0.001125, p[252]: 0.002061, p[253]: 0.008193, p[254]: 0.004810, p[255]: 0.000945.
- Statistics: Valor medio: 65.277709, Valor medio cuadrático: 8352.159630, Varianza: 4090.980394.
- Section header: Datos obtenidos para umbralización Kittler.
- Values: c1: 0.000512, c2: 0.999488, varianza1: 0.000000, varianza2: 4090.892148, m1: 0.000000, m2: 65.311126, thOp: 0.
- Message: Saving unsigned char image in: output/imagenSegmentada.yuv
- Path: Belindas-MacBook-Air:Image\_Segmentation\_Project--master belindabrown\$

Figura 6.2: Comparación del resultado en donde se reconocen varias coordenadas en una misma posición, mediante el método de plantilla en dos imágenes distintas.

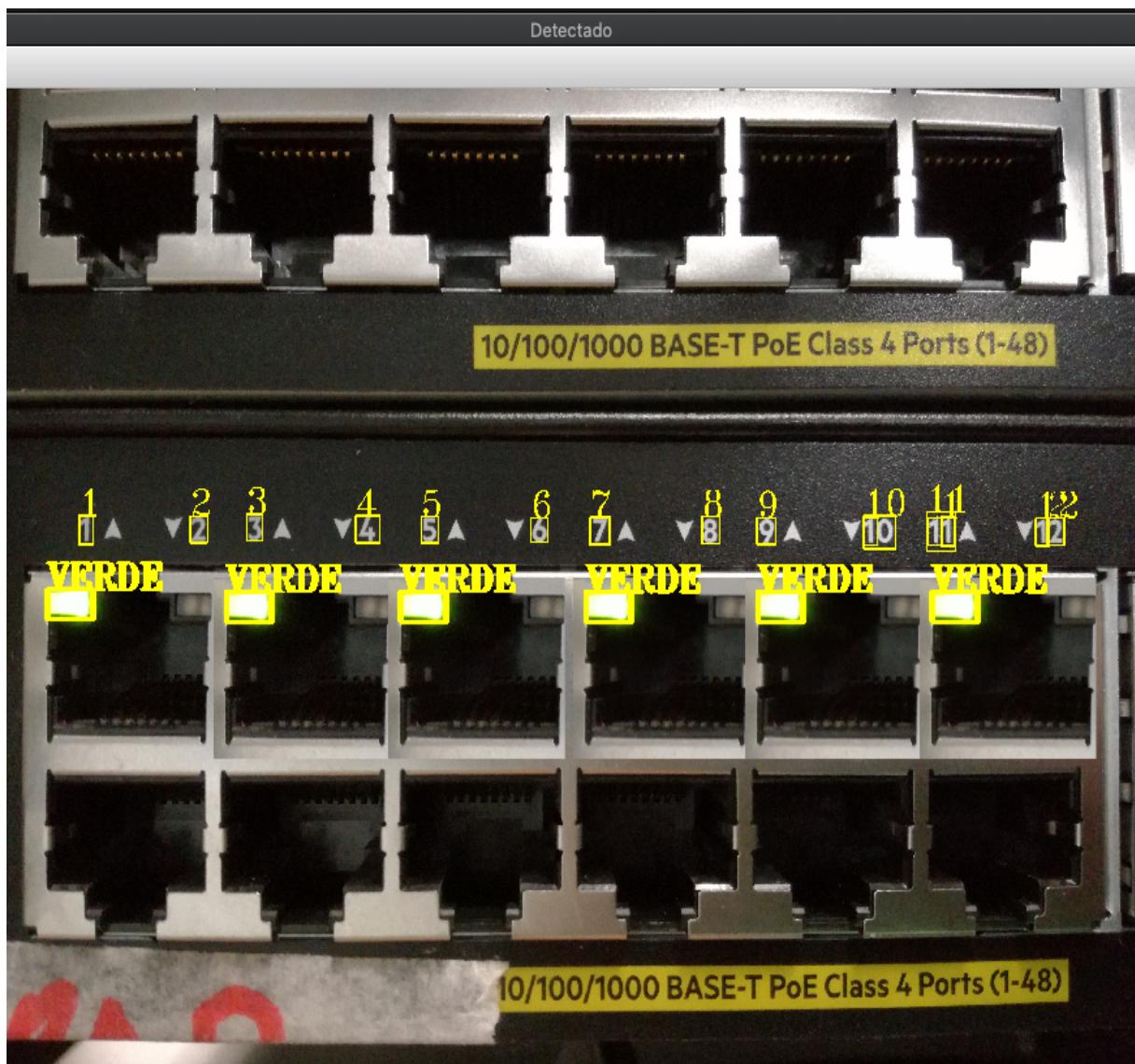


Figura 6.3: Muestra del resultado en donde se observa el mismo fenómeno utilizando otros tipos de plantillas.

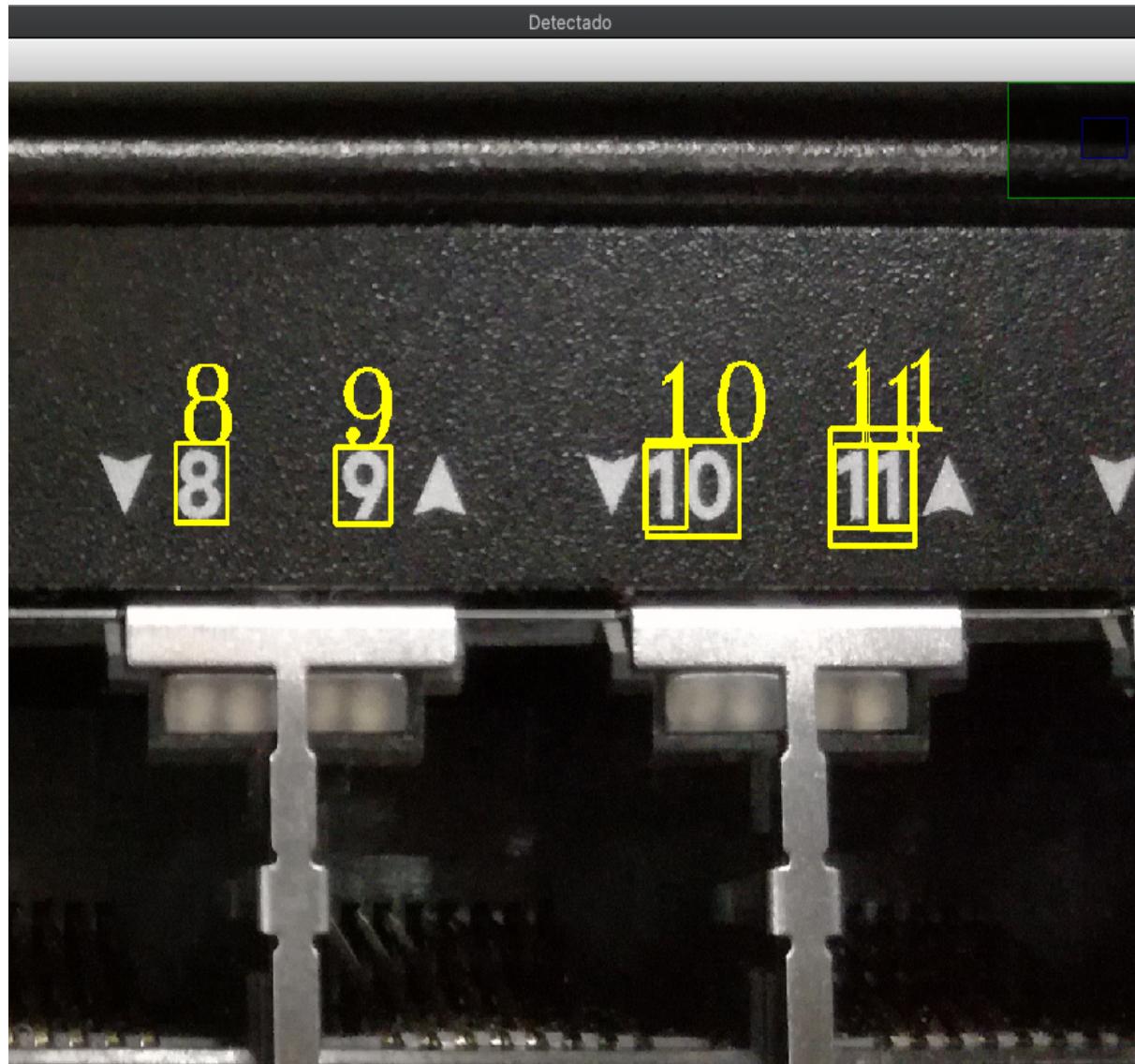


Figura 6.4: Acercamiento de la muestra del resultado en donde se observa el mismo fenómeno utilizando otros tipos de plantillas.

# Bibliografía

- [1] Kordz. Connectivity Assured. Rj45 cat6 field termination connector. data sheet, Obtenido el 30 de mayo del 2020.
- [2] J.M Caballero. *Redes de banda ancha*. MARCOMBO, S. A. Barcelona, España, 1998.
- [3] Hewlett Packard Enterprise company. Data sheet. aruba 2930m switch series, Obtenido el 13 de febrero del 2020.
- [4] Hewlett Packard Enterprise company. Hpe x121 1g sfp rj45 t transceiver (j8177c), Obtenido el 30 de mayo del 2020.
- [5] Hewlett Packard Enterprise Centro de soporte. Aruba 2930m switch series - indicators, Obtenido el 7 de febrero del 2020.
- [6] Python Software Foundation. collections – high-performance container datatypes, Obtenido el 31 de mayo del 2020.
- [7] Python Software Foundation. glob – unix style pathname pattern expansion, Obtenido el 31 de mayo del 2020.
- [8] Python Software Foundation. os – miscellaneous operating system interfaces, Obtenido el 31 de mayo del 2020.
- [9] Rodriguez R. & Sossa J. *Procesamiento y Análisis Digital de Imágenes*. IRA-MA S.A. Editorial y Publicaciones, 2011.
- [10] J Knowlton. *Python*. Anaya Multimedia-Anaya Interactiva, 2009.
- [11] NumPy project and community. Numpy, Obtenido el 31 de mayo del 2020.
- [12] Chen Q. *A Basic Introduction to OpenCv for Image Processing*. DiscoverLab. School of Information Technology Engineering. University of Ottawa, 2007.