



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Propuesta de detección de incendios utilizando procesamiento de imagen, en sistemas de control aéreo.

Trabajo de Titulación para optar al Título de Técnico
Universitario en ROBÓTICA Y MECATRÓNICA.

Alumno:

Esteban Godoy Anfossi.

Profesor Guía:

Rodrigo Méndez.

Enero-2022

AGRADECIMIENTOS

Quiero agradecer a todas las personas que me acompañaron en el camino durante la carrera y mención especial a esas personas que motivaron , ayudaron y criticaron cuando era necesario gracias por todo .

Resumen

El procesamiento de imágenes, o también llamado Visión Artificial es la capacidad de analizar imágenes del mundo real adquiriendo información relevante de estas, para poder resolver tareas en algún problema específico del escenario de donde se están obteniendo.

Hoy en día las industrias cuentan con distintos tipos de monitoreo contra incendios en su interior, pero ¿cómo controlamos el exterior ? Para esto se ideó un filtro antiincendios el cual podría ser implementado en dispositivos aéreos para el monitoreo en vivo de nuestro exterior .

En el siguiente proyecto de título se propondrá una opción de control y monitoreo contra incendios el cual será programado gracias a la librería OpenCV .mediante esta lograremos procesar la imagen en vivo entregada por nuestro dispositivo aéreo de esta manera mejoraremos seguridad y eficiencia de respuesta ante estos posibles siniestros gracias al filtro programado.

ÍNDICE

CONTENIDO

AGRADECIMIENTOS	1
Resumen	2
ÍNDICE	3
INDICE DE FIGURAS	4
SIGLA Y SIMBOLOGIA	5
INTRODUCCION (cambiar)	6
OBJETIVOS	7
OBJETIVO GENERAL	7
OBJETIVOS ESPECIFICOS	7
CAPITULO N°1 CONTEXTUALIZACION DEL PROBLEMA Y CONCEPTOS GENERALES	8
CONTEXTUALIZACION DEL PROBLEMA	9
INCENDIOS	9
UNA SOLUCION	10
CONCEPTOS GENERALES	11
PYTHON	11
OPEN CV (THE OPEN COMPUTER VISION LIBRARY)	11
DRON DJI TELLO	12
VISUAL STUDIO	13
Capítulo 2 ETAPAS DE PROCESAMIENTO DE IMAGEN E IMPLEMENTACION	14
PROGRAMAS NECESARIOS	15
LIBRERIAS	15
PARAMETROS DE CONEXIÓN DRON DJI TELLO	15
CONFIGURACION DE FOTOGRAMA CAPTURADO PARA SER PROCESADO	16
ETAPAS DE UNA APLICACIÓN DE VISION POR COMPUTADOR	16
PROCESAMIENTO DE IMAGEN	17
ESPACIOS DE COLOR	18
ESPACIO HSV	18
HUE-MATIZ	19
SATURACION	20
VALUE-VALOR	20
RANGO DE ALTOS Y BAJOS DE IMAGEN	21
CREACION DE MASCARA DE COLOR	21
FILTRADO DE RUIDO EN UNA IMAGEN	22

APLICACIÓN DE FILTRO GAUSSIANO PARA LA ELIMINACION DE RUIDO EN IMAGEN 23
FUSION FINAL DE IMÁGENES 26
PROCESADO FINAL PARA LA DETECCION DE SINIESTROS 26
PROGRAMACION EN CASO DE DETECCION DE SINIESTRO 26
ANALISIS DE RESULTADOS.....28
CONCLUSION Y RECOMENDACIONES.....28

INDICE DE FIGURAS

Figura 1.1 : Dron sobrevolando área perimetrada 6
Figura 1.2 : Incendio forestal 9
Figura 1.3 : Triangulo del fuego..... 10
Figura 1.4: Vista área industria enap..... 10
Figura 1.5 : Logo Python 11
Figura 1.6 : Logo OpenCV..... 12
Figura 1.7 : Dron Dji Tello 13
Figura 1.8 : Logo Visual Studio..... 13
Figura 2.1 : Librerías a utilizar 15
Figura 2.2 : Parámetros de conexión Dron dji tello..... 16
Figura 2.3 : Definición de frame 16
Figura 2.4 : Definición de frame para opencv 16
Figura 2.5 : Definición de visión 17
Figura 2.6 : Matriz G..... 17
Figura 2.7 : Matriz R,G,B 18
Figura 2.9 : Definición variable HSV 19
Figura 2.10 : representación de espacio de color utilizado..... 19
Figura 2.11 : Variación de matiz en HSV..... 20
Figura 2.12 : Saturación en HSV..... 20
Figura2.13 : Brillo en HSV 21
Figura 2.14 : Espacio HSV 21
Figura 2.15 : Rangos de altos y bajos 21
Figura 2.16 : Especificación de Valor..... 21
Figura2.17 : Ejemplo de mascara de color 22
Figura 2.19 : Kernel 3x3 y 4x4..... 23
Figura 2.20 : Grafico función Gaussiana 24
Figura 2.21 : Ejemplo filtro Gaussiano 24
Figura 2.22 : Definición filtro Gaussiano 25
Figura 2.23 : Filtro gassiano usado 25
Figura 2.24 : Definición mascara final 26
Figura 2.25 : Contador de pixeles distintos a negros 26
Figura 2.26 : Definición de Función para detectar siniestros 26
Figura 2.27 : Resultados de filtros utilizados en proyecto..... 27
Figura 2.28 : Definición de función para enviar correos 28

SIGLA Y SIMBOLOGIA

HSV	: Hue, Saturation, Value (Tono o matiz, Saturación, Brillo o Valor)
OPENCV	: Open Computer Vision Library (Librería de Visión por Computador)
RGB	: Red, Green, Blue (Rojo, Verde, Azul), Matriz de una imagen
IA	: Inteligencia Artificial
IOT	: Internet of Things (Internet de las cosas)

INTRODUCCION

Hoy en día la industria de la seguridad es algo común en la mayoría de los sitios, debido a que la gente necesita tener controlado su entorno de trabajo, generalmente cuentan con cámaras de vigilancia y alarma contra incendios, todo esto para el control interior, pero ¿monitoreamos y controlamos nuestro exterior? El objetivo del trabajo a realizarse consta de diseñar y programar un filtro de imagen el cual pueda ser simulado en dispositivos aéreos como propuesta de control y monitoreo exterior de tu recinto.

Mediante la implementación de esta propuesta se busca implementar medida de control de bajo costo ,versátil y adaptable a las necesidades de hoy en día, aumentando la seguridad del entorno logrando un control objetivo Se utilizará un dron DJI Tello además de un computador la cual procesará la imagen a través de la librería open CV de esta podremos obtener una imagen procesada por filtros la cual nos determinará la existencia de algún siniestro el cual al ser detectado se enviara un correo a un emisor específico.

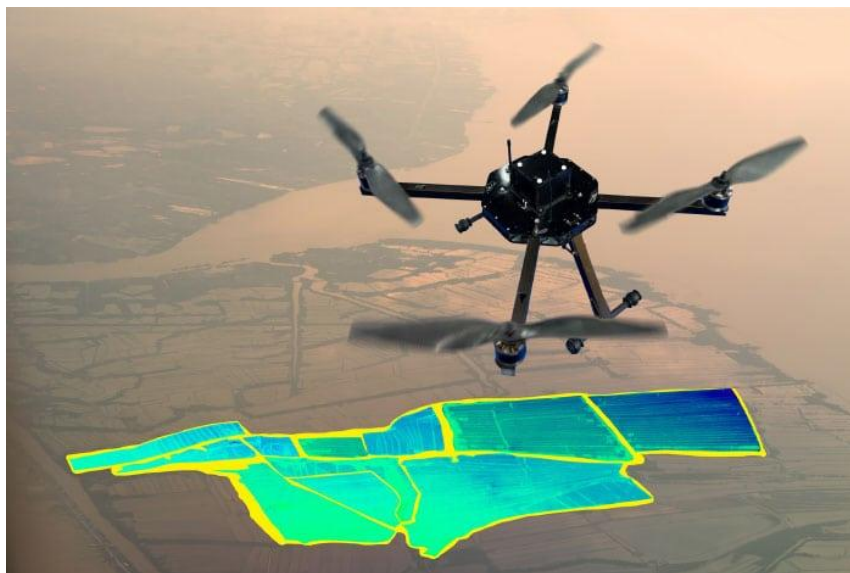


Figura 1.1 : Dron sobrevolando área perimetrada

OBJETIVOS

OBJETIVO GENERAL

Generar una propuesta de monitoreo y control exterior de incendios para reducir pérdidas materiales con la entrega de información en vivo, a la vez mejorando la rapidez de envío de esta al receptor específico.

OBJETIVOS ESPECIFICOS

Proponer una idea de control que pueda ser mejorada y usada a futuro.

Programar y crear un filtro de imagen que pueda ser utilizado en drones de gama media en adelante.

Estudiar las ventajas y desventajas proporcionadas por el sistema a crear para lograr su máxima fiabilidad.

Lograr disminuir los tiempos de respuestas ante a estos eventos.

Lograr un mayor control de estos a corto plazo.

Estudiar las librerías a usar .

**CAPITULO N°1 CONTEXTUALIZACION DEL PROBLEMA Y CONCEPTOS
GENERALES**

CONTEXTUALIZACION DEL PROBLEMA

INCENDIOS

Un incendio es una ocurrencia de fuego no controlada que puede afectar o abrasar algo que no está destinado a quemarse. Puede afectar a estructuras y a seres vivos. La exposición de los seres vivos a un incendio puede producir daños muy graves hasta la muerte, generalmente por inhalación de humo o por desvanecimiento producido por la intoxicación y posteriormente quemaduras graves.

Para que se inicie un fuego es necesario que se den conjuntamente tres componentes: combustible, oxígeno y calor o energía de activación, lo que se llama triángulo del fuego.



Figura 1.2 : Incendio forestal

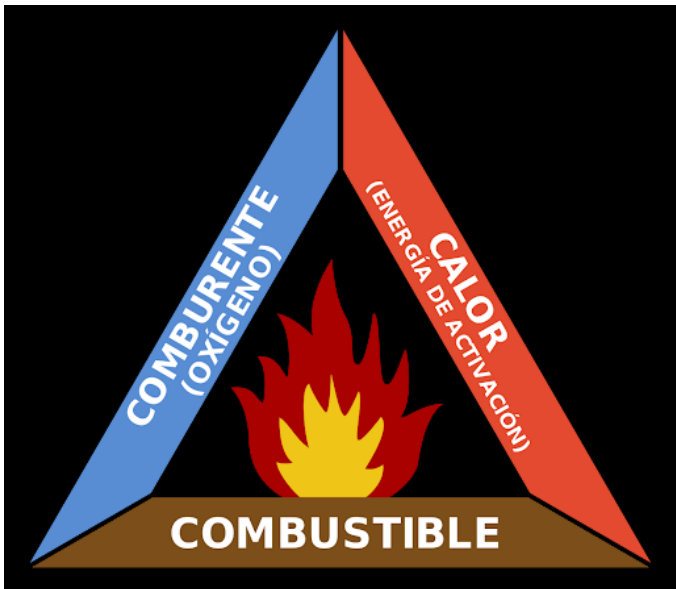


Figura 1.3 : Triangulo del fuego

UNA SOLUCION

Hoy en día las industrias cuentan con distintos tipos de prevención de incendio en su interior, tales como redes secas contra incendios, redes húmedas contra incendios, detectores de humo, etc. ¿Pero ¿qué sucede con el exterior?



Figura 1.4: Vista área industria enap

Actualmente las industrias o empresas constan de cámaras de seguridad las cuales a veces no cubren todo el perímetro , Para contrarrestar esa necesidad se ideó un filtro el cual se

simulará mediante un dron donde obtendremos imagen en vivo la cual será procesada para detectar posibles siniestros en nuestra área perimetrada específica

CONCEPTOS GENERALES

PYTHON

Python es un lenguaje simple usado en varias áreas de la tecnología, procesamiento de datos, web, redes, visión artificial, etc.

Es un lenguaje sencillo de usar y de código abierto, Python es un lenguaje de programación de scripting.

Los lenguajes scripting son aquellos lenguajes que usan un intérprete en vez de ser compilados.

La mayoría de los usuarios de Python le consideran como un lenguaje más limpio y elegante a la hora de programar.

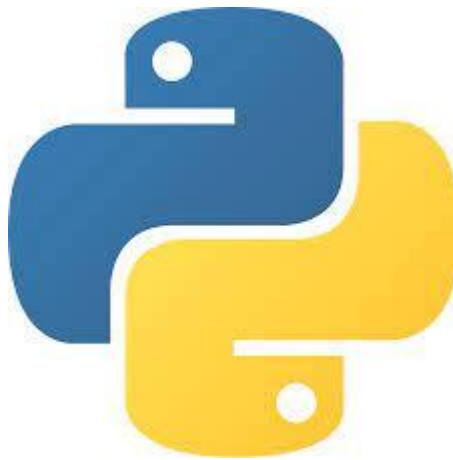


Figura 1.5 : Logo Python

OPEN CV (THE OPEN COMPUTER VISION LIBRARY)

OpenCV es una librería de visión artificial creada por Intel, su publicación es bajo licencia BSD que permite que sea utilizada libremente para propósitos comerciales y de investigación.

La librería tiene más de 2500 algoritmos, que incluye algoritmos de machine learning y de visión artificial para usar.

Estos algoritmos permiten identificar objetos, caras, clasificar acciones humanas en vídeo, extracción de modelos 3D, comparar imágenes, eliminar ojos rojos, seguir el movimiento de los ojos, reconocer escenarios.

Esta fue creada el año 1999 se ha utilizado en una gran cantidad de aplicaciones, y hasta 2020 se la sigue mencionando como la biblioteca más popular de visión artificial.



Figura 1.6 : Logo OpenCV

DRON DJI TELLO

Este pequeño dron está diseñado con el diseño de DJi y la tecnología de la start-Up Ryze Tech para poder grabar casi en cualquier lugar mediante un teléfono smartphone y una App dedicada de forma fácil e intuitiva.

Su reducido tamaño permite grabar videos con una resolución de 720p y hacer fotos de hasta 5 Megapíxeles.

La autonomía de este drone es de 13 minutos de vuelo y la distancia máxima a la que puedes operarlo es de hasta 100 metros

El dron ofrece la facilidad de programarlo en Python a través de sus librerías facilitadas por sus creadores.



Figura 1.7 : Dron Dji Tello

VISUAL STUDIO

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para Windows y macOS. Es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web.

El entorno de desarrollo integrado de Visual Studio es un panel de inicio creativo que se puede usar para editar, depurar y compilar código y, después, publicar una aplicación. Un entorno de desarrollo integrado (IDE) es un programa con numerosas características que se pueden usar para muchos aspectos del desarrollo de software. Más allá del editor estándar y el depurador que proporcionan la mayoría de IDE, Visual Studio incluye compiladores, herramientas de finalización de código, diseñadores gráficos y muchas más características para facilitar el proceso de desarrollo de software.



Figura 1.8 : Logo Visual Studio

Capítulo 2 ETAPAS DE PROCESAMIENTO DE IMAGEN E IMPLEMENTACION

PROGRAMAS NECESARIOS

Para la programación del filtro se necesitará tener instalado Visual Studio o algún compilador que acepte Python 3.7.

LIBRERIAS

Para poder interactuar con algunos componentes y funciones se necesitarán las siguientes librerías:

- **DJItellopy:** Esta librería nos permite interactuar con el dron y establecer comunicación vía wifi, asegurando una transmisión de video lo más estable posible mediante programación en Python.
- **Cv2:** OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. OpenCV significa Open Computer Vision (Visión Artificial Abierta). Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en una gran cantidad de aplicaciones, y hasta 2020 se la sigue mencionando como la biblioteca más popular de visión artificial.¹ Detección de movimiento, reconocimiento de objetos, reconstrucción 3D a partir de imágenes, son sólo algunos ejemplos de aplicaciones de OpenCV.
- **Numpy:** es una biblioteca para el lenguaje de programación Python que da soporte para crear vectores y matrices grandes multidimensionales, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas.
- **Smtplib:** El módulo smtplib define un objeto de sesión de cliente SMTP que se puede usar para mandar correo a cualquier máquina de Internet con un demonio de escucha SMTP o ESMTP.

```
from djitellopy import Tello
import cv2
import time
import numpy as np
import smtplib
```

Figura 2.1 : Librerías a utilizar

PARAMETROS DE CONEXIÓN DRON DJI TELLO

Para el proyecto se utilizarán los siguientes parámetros definidos por la librería **djitellopy**.

```
me = Tello() #Asignamos a la palabra me la variable principal de llamado de la librería
Tello
me.connect() #Luego estableceremos conexión con nuestro dron mediante wifi
connect() nos servirá para establecer enlace con la red correspondiente del dron para
hacer efectivo el traspaso de información
print(me.get_battery()) # se escribirá en consola la batería actual de nuestro dron
me.streamoff()# se llamará al apagado de stream del dron por definición de librería para
evitar posibles errores
me.streamon() # luego iniciaremos stream para poder utilizar la imagen mostrada por
nuestra Cámara
```

Figura 2.2 : Parámetros de conexión Dron dji tello

CONFIGURACION DE FOTOGRAMA CAPTURADO PARA SER PROCESADO

Para lograr la captura de video del dron se definirá lo siguiente :

```
frame_read = me.get_frame_read()
```

Figura 2.3 : Definición de frame

En este caso estamos definiendo que **frame_read** corresponderá a la captura de imagen recibida por nuestro dron .

Para lograr que la librería OpenCV detecte la lectura de nuestra transmisión ya definida como **frame_read** se configurara lo siguiente :

```
FrameF = frame_read.frame
```

Figura 2.4 : Definición de frame para opencv

ETAPAS DE UNA APLICACIÓN DE VISION POR COMPUTADOR

Las etapas de programación y procesamiento de imágenes de visión por computador son variadas y dependen del resultado o finalidad a ejecutar, para el siguiente proyecto de reconocimiento y selección de imágenes por color y forma se requirió de distintos procesos de Visión de bajo nivel.

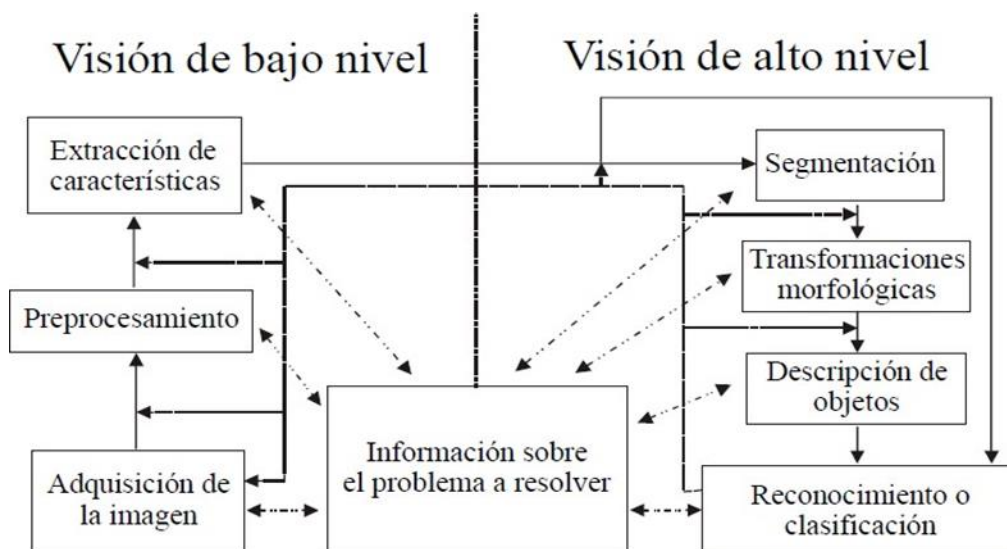


Figura 2.5 : Definición de visión

PROCESAMIENTO DE IMAGEN

Una imagen se conforma digitalmente por una serie de valores en una matriz, por ejemplo, una matriz como la de la ecuación (Fig. 2.6), podría representar una imagen de escala de grises, donde cada valor dentro de esta es considerado un pixel y donde axy toma valores entre un rango de 0 y 255, además cada pixel de la matriz se considera de 1 byte (8 bits). Teniendo en cuenta esta información, se puede decir que, si un valor axy se acerca a 0, significa que ese pixel es más oscuro, mientras que si se acerca a 255, el pixel es más claro. Cuando solo existen esos dos valores (0 y 255), se considera una imagen binaria, es decir de solo blanco y negro. La resolución de una imagen es conocida como los valores de m y n, es decir, las dimensiones de la matriz G.

$$G = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}$$

Figura 2.6 : Matriz G

Una imagen de color está compuesta por tres matrices con la forma de G (Fig. 2.7), en donde cada matriz R, G y B, representan la intensidad en la imagen de los colores rojo, verde y azul respectivamente, que, al combinarse estos colores, se puede generar toda una

gama de colores que percibe el ser humano. En este caso, un pixel es representado por tres valores (rxy; gxy; bxy) indicando la intensidad de cada color, tal como se muestra en la ecuación (Fig.2.7).

$$RGB = \left[\begin{matrix} R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1n} \\ r_{21} & r_{22} & r_{23} & \dots & r_{2n} \\ r_{31} & r_{32} & r_{33} & \dots & r_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & r_{m3} & \dots & r_{mn} \end{bmatrix} \quad G = \begin{bmatrix} g_{11} & g_{12} & g_{13} & \dots & g_{1n} \\ g_{21} & g_{22} & g_{23} & \dots & g_{2n} \\ g_{31} & g_{32} & g_{33} & \dots & g_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{m1} & g_{m2} & g_{m3} & \dots & g_{mn} \end{bmatrix} \quad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2n} \\ b_{31} & b_{32} & b_{33} & \dots & b_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & b_{m3} & \dots & b_{mn} \end{bmatrix} \end{matrix} \right]$$

Figura 2.7 : Matriz R,G,B

Bajo este concepto de estructura de una imagen podemos entender más conceptos de procesamiento de imágenes.

ESPACIOS DE COLOR

Hay varias maneras en que se puede describir una imagen, una de ellas es la que se describió anteriormente y es el espacio RGB, el cual se utiliza bastante debido a que físicamente cada pixel en una pantalla es representado por la cantidad de intensidad que se le aplique para cada color, y la pantalla completa se describe como la matriz de la ecuación (Fig. 2.7). Sin embargo, esta matriz puede transformarse de varias maneras digitalmente, modificando los valores RGB y transformando las matrices R, G y B, para obtener otros espacios de color y así representar la imagen con otra matriz que contenga los valores del nuevo espacio de color. A continuación, se describirá el espacio de color con el que trabajamos en este proyecto.

ESPACIO HSV

El modelo HSV es un espacio de color que al igual que el espacio RGB descrito en la ecuación (Fig. 2.7), es representado con tres componentes que son H, S y V, que por sus siglas en ingles corresponden al tono o matiz (Hue), saturación (Saturation) y brillo o valor (Vale). Estas tres magnitudes se representan por medio de un diagrama cilíndrico que contiene la información necesaria para hallar el color deseado (Fig.2.8). El tono o matiz está representado por un ángulo que recorre el cilindro sobre su eje, los valores de brillo están representados el eje vertical del cilindro y la saturación por el eje horizontal.

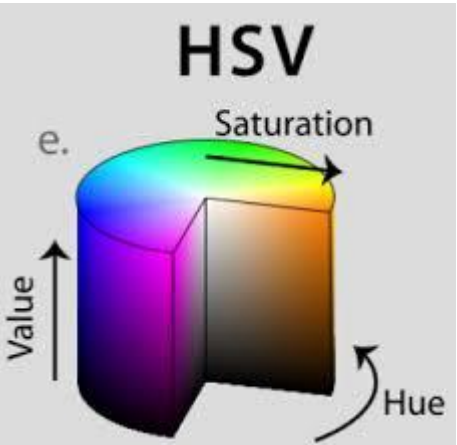


Figura 2.8 : Espacio color HSV

En este caso se definió la variable hsv de la siguiente manera :

```
hsv = cv2.cvtColor(blur, cv2.COLOR_BGR2HSV)
```

Figura 2.9 : Definición variable HSV

cv2.cvtColor nos ayudará a transformar el espacio de color de BGR A HSV utilizando el siguiente comando : **cv2.COLOR_BGR2HSV** .

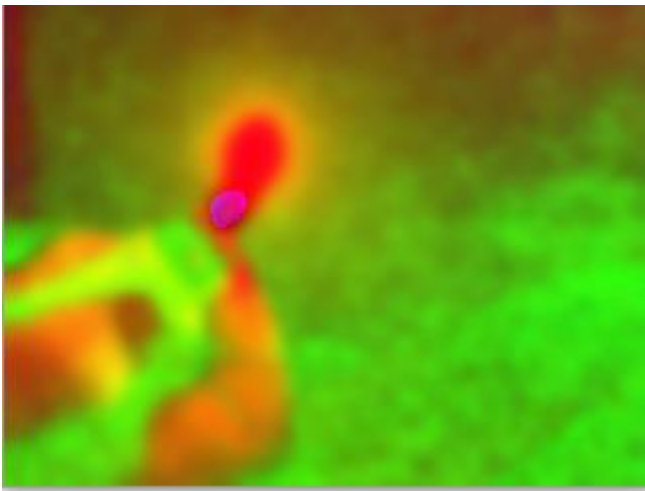


Figura 2.10 : representación de espacio de color utilizado

HUE-MATIZ

La gama cromática se representa por medio del valor del ángulo del matiz respecto al eje del cilindro, donde cada ángulo representa un color diferente. En el caso de ser programado en Python con la biblioteca de OpenCV este valor varía entre 0° a 180° puesto

que si se trabajara con 360°, este valor supera el valor de 255, que es el valor máximo de un pixel en el formato llamado uint8 que utiliza únicamente 8 bits. Se puede observar en la (Fig.2.11), como varía el color al variar el ángulo, manteniendo constante la saturación y el brillo en 255 (100 %).



Figura 2.11 : Variación de matiz en HSV

SATURACION

La saturación indica la escala de grises en el espacio de color. Los valores varían de 0% a 100 %. Al aumentar este parámetro, se obtiene un color más vivo, mientras que, al disminuirlo, el color tiende al blanco.

En la figura 2.12 se puede apreciar cómo se comportan los cambios de este componente, tomando como ejemplo un tono rojo de matiz cero (0) y un brillo (value) máximo (255) y variando la componente de saturación.



Figura 2.12 : Saturación en HSV

Estos valores en OpenCV varían entre 0 y 255, para representar el rango mencionado anteriormente de 0% a 100% de saturación.

VALUE-VALOR

El valor o brillo es el grado de luminosidad del color que al igual que la saturación, varía entre 0% y 100 %. Cuando lo valores están cercanos a 0 el espacio de color estará en la tonalidad oscura en cambio, mientras más grande sea el valor, va disminuyendo la oscuridad. En la figura 2-13 se puede apreciar cómo aumenta el brillo del azul, que corresponde a un ángulo de matiz de 90% y utilizando una saturación del 100 %.



Figura2.13 : Brillo en HSV

RANGO DE ALTOS Y BAJOS DE IMAGEN

En este proyecto se determinó utilizar los siguientes rangos de altos y bajos de imagen los cuales nos servirán para determinar el color específico a detectar.

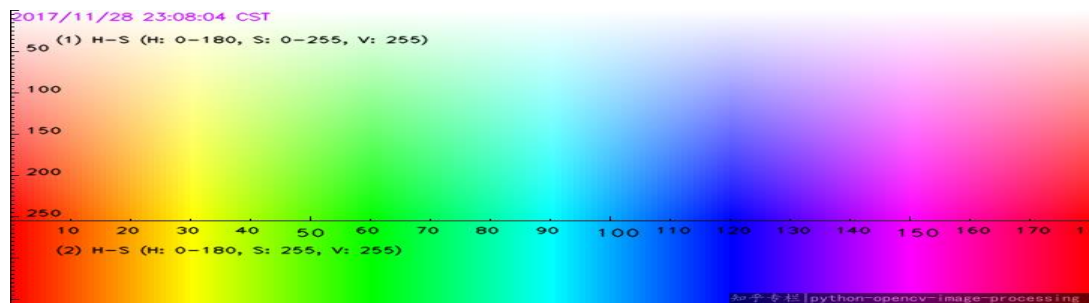


Figura 2.14 : Espacio HSV

Luego de determinar los rangos de altos y bajos de imagen necesitaremos crear un array con Numpy donde se especificarán los componentes en H,S y V , además especificaremos con **dtype="uint8"** que nuestro valores será un entero sin signo representado en 8 bits .

```
bajos_img = [0,120,120] #BAJOS DE IMAGEN H,S,V
altos_img = [45, 220, 255] #ALTOS DE IMAGEN H,S,V
```

Figura 2.15 : Rangos de altos y bajos

```
bajos_img = np.array(bajos_img, dtype="uint8")
altos_img = np.array(altos_img, dtype="uint8")
```

Figura 2.16 : Especificación de Valor

CREACION DE MASCARA DE COLOR

Una Capa Máscara, es un tipo de capa usada para esconder o “enmascarar” parte del contenido de otra capa, en el caso de este proyecto se realiza para acotar el color a seleccionar separándolo del entorno donde se encuentran objetos de diferente color y forma que debemos descartar.

El comando utilizado es “mask = cv2.inRange(hsv, bajos, altos)”



Figura2.17 : Ejemplo de mascara de color

En el proyecto se definió de la siguiente forma :

```
mask = cv2.inRange(hsv, bajos_img, altos_img)
```

Figura 2.18 : Definición de mascara

FILTRADO DE RUIDO EN UNA IMAGEN

Las imágenes digitales no son perfectas. El ruido inherente de los propios dispositivos o los efectos contraproducentes por iluminación alteran la realidad.

De acuerdo a los diferentes entornos los cambios de luz, reflejos y brillos alteran la imagen, por lo consiguiente se deben tener en cuenta soluciones a estos problemas.

Es algo que no solo sucede en los entornos visuales. Cualquier señal digital o analógica está expuesta a diferentes fuentes de ruido.

En el tratamiento digital de imágenes hay diferentes métodos para eliminar el ruido (promediado, mediana, Gaussiano o bilateral).

Para realizar los filtros hemos utilizado un método llamado convolución. Este consiste en recorrer píxel a píxel una imagen con una máscara o kernel de $N \times N$.

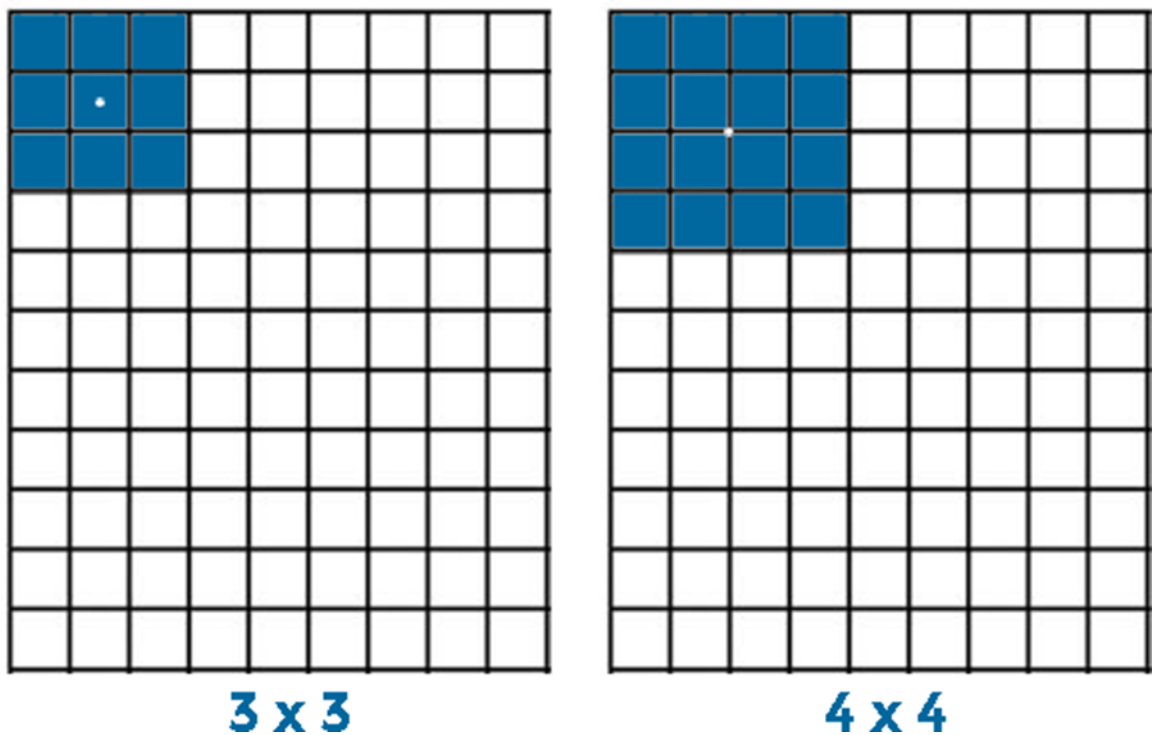


Figura 2.19 : Kernel 3x3 y 4x4

Es importante que el tamaño del kernel sea impar para tener un pixel central que será el pixel a tratar, en el caso de nuestro proyecto el Kernel es de 21x21.

Gracias al kernel podemos suavizar la imagen y eliminar los detalles, se puede comparar con el desenfoque de una cámara fotográfica. Para poder detectar los bordes en nuestra imagen aplicaremos el filtro Gaussiano.

APLICACIÓN DE FILTRO GAUSSIANO PARA LA ELIMINACION DE RUIDO EN IMAGEN

El filtrado Gaussiano es igual que un promediado pero ponderado. Esta ponderación se hace siguiendo la Campana de Gauss. Con esto se consigue dar más importancia a los píxeles que están más cerca del centro de los que están más alejados.

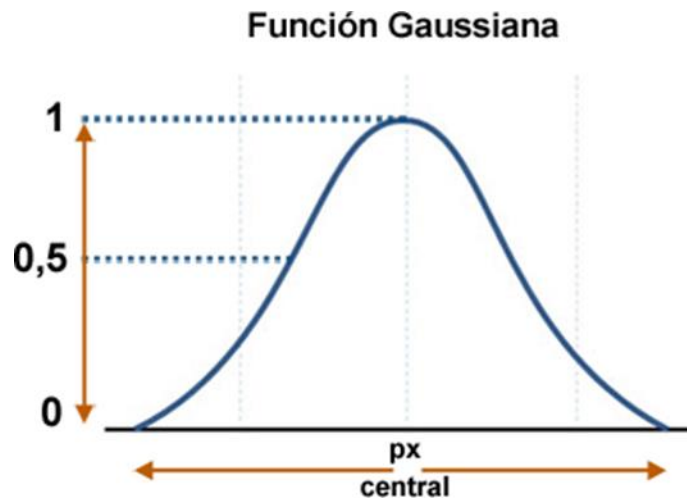


Figura 2.20 : Grafico función Gaussiana

En términos simples la campana de Gauss es una aproximación de cómo ve el ojo humano, si lo vemos en comparación a algo natural.

La función Gaussiana o Distribución Normal se utiliza en muchas áreas de la ciencia, la altura de la población o el coeficiente intelectual se puede representar a través de la Distribución Normal. Gracias a su forma acampanada y simétrica, hace que los elementos que están más cerca del centro sean los más comunes. Por el contrario, los elementos más lejanos serán los más raros o diferentes.

Para poder aplicarlo en una imagen debemos hacerlo en dos dimensiones. Esto lo hacemos a través de una máscara o kernel de convolución.

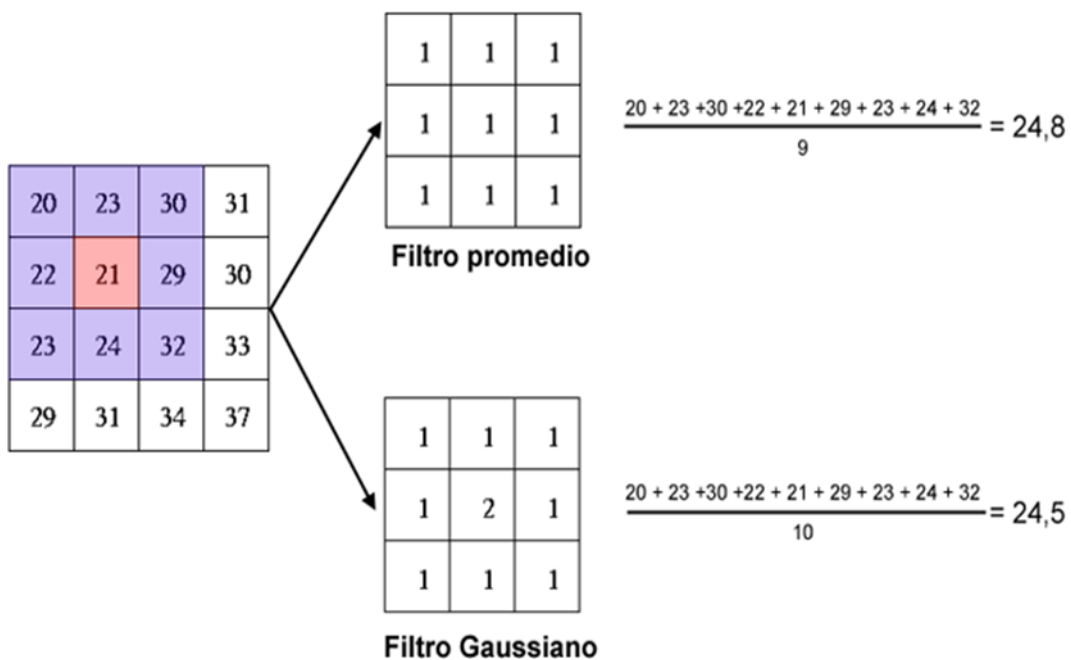


Figura 2.21 : Ejemplo filtro Gaussiano

Como se puede comprobar en la imagen anterior, se da más importancia al píxel central que a los píxeles que están alrededor de éste.

La configuración en OpenCV para el filtro Gaussiano es como sigue: `gaussiana = cv2.GaussianBlur(imagen, (n, n),  $\sigma$ )`

Donde

- `gaussiana`: es la imagen desenfocada resultante.
- `imagen`: es la imagen original, la que queremos suavizar o desenfocar.
- `n X n`: es el tamaño del kernel o máscara de convolución. Recuerda que debe ser impar.
- σ : sigma (σ) representa la desviación estándar en el eje X es decir, la anchura de la campana de Gauss. Si se pone un 0, OpenCV se encarga automáticamente de calcular ese valor para el kernel o máscara que se ha elegido. Esta es la opción aconsejable.

En nuestro caso la aplicación del filtro Gaussiano con un kernel de 21×21 es de la siguiente manera.

```
blur = cv2.GaussianBlur(img, (21, 21), 0)
```

Figura 2.22 : Definición filtro Gaussiano



Figura 2.23 : Filtro gaussiano usado

FUSION FINAL DE IMÁGENES

Ya sabiendo que filtros se usaron se necesita unificar las distintas capas realizadas lo cual se realiza con el siguiente operador :

bitwise_and(matriz_fuente1, matriz_fuente2, matriz_destino, máscara)

bitwise_and :Nos permite combinar las matrices correspondientes a las dos imágenes dando como resultado la fusión de las dos imágenes . La operación bit wise en las dos imágenes devuelve una imagen con la fusión realizada según la especificación.

En nuestro caso el operador se definió de la siguiente manera:

```
output = cv2.bitwise_and(img, hsv_mask=mask)
```

Figura 2.24 : Definición mascara final

En este caso **output** será nuestra variable que represente los resultados de nuestro filtro final .

PROCESADO FINAL PARA LA DETECCION DE SINIESTROS

Para lograr procesar la información entregada por el filtro realizado se definirá primero una variable que nos diga exactamente cuantos pixeles blancos se encuentran en la imagen actual procesada.

```
valor = cv2.countNonZero(mask)
```

Figura 2.25 : Contador de pixeles distintos a negros

cv2.countNonZero se utiliza para poder detectar todos los pixeles que no son negros , en este caso nuestra variable **valor** será equivalente a los pixeles que no son negros en nuestra mascara final .

PROGRAMACION EN CASO DE DETECCION DE SINIESTRO

Para la detección de siniestros se utilizará la información entregada de la variable **valor** .

```
if int(valor) >=4500 :  
    enviar_correo()  
else :  
    print("no hay fuego")
```

Figura 2.26 : Definición de Función para detectar siniestros

Donde tenemos que si nuestra variable **valor** es igual o mayor a 4500 pixeles distintos de negro este envíe un correo ,se determinó este valor debido a pruebas realizadas anteriormente. Si no se cumple la condición se escribirá en consola que no se ha detectado nada .Adicionalmente se estará mostrando cada capa realizada.

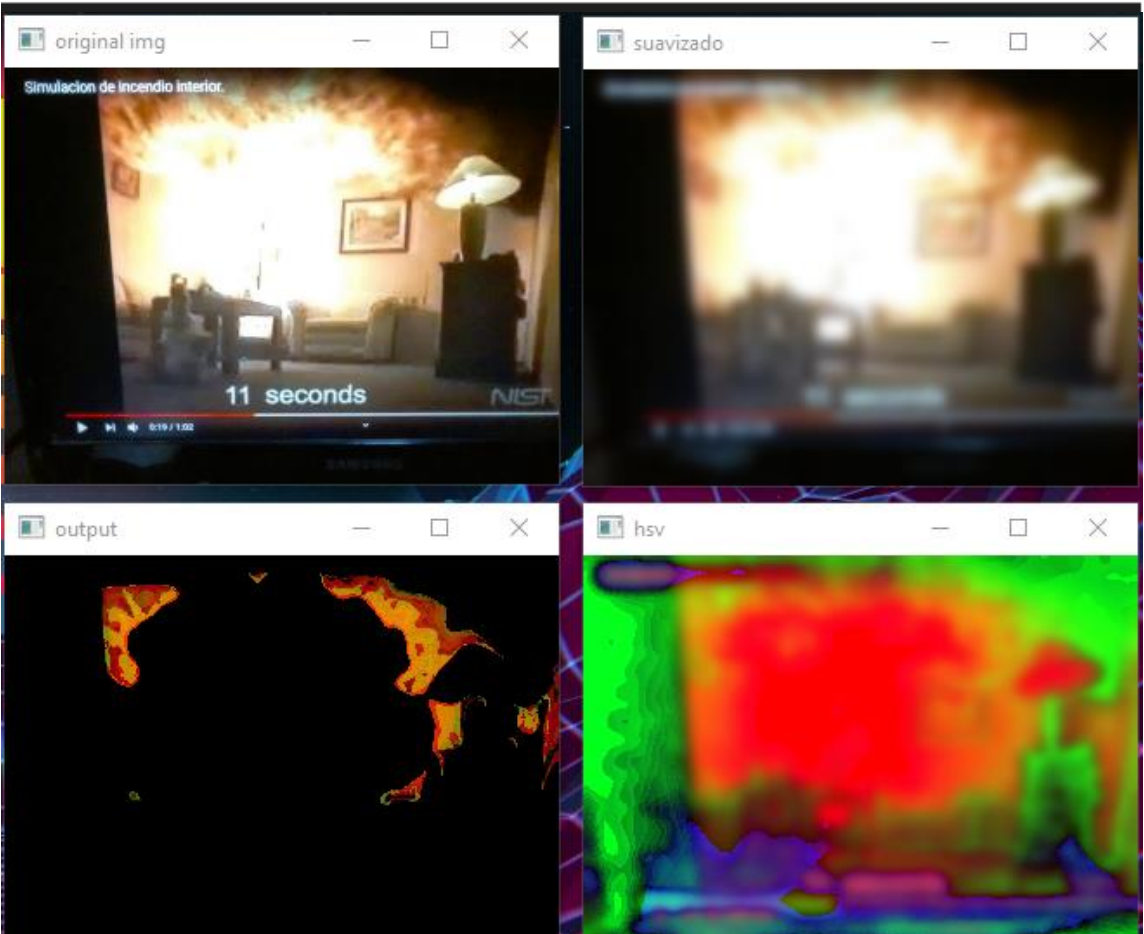


Figura 2.27 : Resultados de filtros utilizados en proyecto

Para el enviado de correo se creó una función predeterminada por el módulo **Smtplib**.

```
def enviar_correo():

    mensaje = "Incendio detectado" #se ingresa el mensaje que se quiere transmitir
    server = smtplib.SMTP('smtp.gmail.com', 587) #se define objeto smtp, se le entregan
dos valores ("servidor de correo", "puerto")
    server.starttls() #inicio de protocolo tls
    server.login('incendioutfsm@gmail.com', 'Eslegoan13') #LOGUEO EN CORREO E
MISOR
    server.sendmail('incendioutfsm@gmail.com', 'incendioutfsm2@gmail.com', mensaje )
# SE DEFINE CORREO EMISOR Y RECEPTOR
    server.quit() #FINALIZACION DEL SERVER
    print("correo enviado exitosamente!") #SE ESCRIBE MENSAJE DE |
CONFIRMACION
```

Figura 2.28 : Definición de función para enviar correos

ANALISIS DE RESULTADOS

Se realizaron pruebas en distintos ambientes de siniestros para lograr que el proyecto sea lo más versátil posible .Se determino un porcentaje de efectividad equivalente a 55 % de efectividad ,lo cual nos confirma que es un sistema que es posible trabajarlo y mejorarlo a nivel industrial , debido a que hoy en día se considera un proyecto desarrollable sobre 45% de efectividad .

CONCLUSION Y RECOMENDACIONES

La visión artificial o el procesamiento de imágenes es una gran herramienta informática la cual puede ser aplicada en distintos sistemas ,lo cual la, muy poderosa y efectiva, puede ser muy útil según el contexto que se le aplique. En este caso en la detección de un espectro de color específico. Como sentido adicional a los sensores normales es un complemento con buenas capacidades y versatilidad, para todo tipo de aplicación.

En si es una herramienta completa, pero no perfecta. Depende mucho de las condiciones donde se aplique como por ejemplo el tipo de escenario, tipo de cámara utilizada, problemática a ser resuelta, Espectro de color a procesar, etc. Si en algunas de estas

especificaciones mencionadas anteriormente se presenta en tu proyecto será difícil la fiabilidad de la imagen a procesar obteniendo posibles errores o falsos positivos.

En proyecto realizado se logro mayor fiabilidad de lo esperado gracias a la correcta configuración de espacio de color HSV superando el 50% de fiabilidad .

Se recomienda mejorar e implementar sistema mediante IOT logrando alcanzar un avance hacia las nuevas tecnologías creadas ,así será un sistema versátil y de tecnología actual logrando que no se quede atrás en el tiempo.

Posibles mejoras a este proyecto serian la implementación de una inteligencia artificial para el procesamiento de imagen mejorando fiabilidad , filtrado de imagen y detección de estos eventos logrando la obtención de un sistema con una fiabilidad reconocida ante la detección de siniestros.

Se recomienda la mejora de dispositivo aéreo de simulación mejorando calidad de imagen y fotogramas por segundo para evitar posibles falsos de lectura. Otra recomendación factible sería una mejora del envío del correo en vez de solo enviar a un destinatario, preferible incorporar nube de correos para así la entrega de la información sea masiva.

REFERENCIAS

1. DESARROLLO DE SISTEMA RECOLECTOR DE COLILLAS DE CIGARROS CONTAMINANTES, PARA VEHICULOS ROBOTICOS AUTONOMOS TERRESTRES[en línea]. <
<https://repositorio-usm-cl.usm.idm.oclc.org/handle/11673/50325>>
[Consulta: 05 Junio 2021].
2. Cursos de detección de colores con OpenCV
[en línea]. <<https://omes-va.com/deteccion-de-colores/>> [Consulta: 07 junio2021].
3. ALGORITMOS DE PROCESAMIENTO DE IMAGEN APLICADOS A LA DETECCIÓN DE FIGURAS GEOMETRICAS Y SUS PROPIEDADES ESPACIALES. [en línea]. <
http://opac.pucv.cl/pucv_txt/Txt-4500/UCE4968_01.pdf>
[Consulta: 05 junio 2021].
4. Procesado de imagen [en línea].
< <https://volaya.github.io/libro-sig/chapters/Imagenes.html>>
[Consulta: 29Mayo 2021].
5. Filtrado de imagen [en línea]. <
<http://www.dccia.ua.es/dccia/inf/assignaturas/Vision/vision-tema2.pdf>> [Consulta: 10 junio 2021].
6. Librería DJI TELLOPY [en línea].
<https://github.com/damiafuentes/DJITelloPy/tree/aad98c1e8d8fd916112e7e52eba1318398adb6ad> > [Consulta: 27Mayo 2021].
7. Explicación de programación de librería OpenCV [en línea]. <
<https://www.youtube.com/watch?v=2uxfqIDbVV4&t=1748s>>
[Consulta: 15 junio 2021].
8. Programa realizado [en línea]. <
<https://github.com/Esteban1/Deteccion-de-incendios-a-traves-de-opencv.git>> [Consulta: 05 enero 2022].