Universidad de la República
Facultad de Ingeniería

AUTOBOL
Detección Automática de Bólidos para Cámaras AllSky

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

Juan Pedro Ballestrino, Cecilia Deandraya y Cristian Uviedo

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero/a Electricista.

Tutores

Dr. Ing. Ignacio Ramírez ..... Facultad de Ingeniería, UdelaR
Dr. Gonzalo Tancredi ..... Facultad de Ciencias, UdelaR

Tribunal

Dr. Gonzalo Tancredi ..... Facultad de Ciencias, UdelaR
Mag. Ing. Manuel Caldas ..... Facultad de Ciencias, UdelaR
Mag. Ing. Guillermo Carbajal ..... Facultad de Ingeniería, UdelaR

Montevideo
viernes 29 abril, 2022

AUTOBOL

Detección Automática de Bólidos para Cámaras AllSky, Juan Pedro Ballestrino, Cecilia Deandraya y Cristian Uviedo.

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 125 páginas.
Compilada el viernes 29 abril, 2022.
http://iie.fing.edu.uy/

"...durante setenta y cinco días continuos, se vio en los cielos un enorme cuerpo de fuego similar a una nube llameante, que no descansaba en un lugar sino que se movía con movimientos intrincados e irregulares, de modo que los fragmentos de fuego que se desprendían de ella por su curso errático y en picada eran llevados en todas direcciones y lanzaban destellos de fuego, al igual que las estrellas fugaces. Pero cuando cayó en esa parte de la Tierra y los habitantes, después de recuperarse de su miedo y asombro, se reunieron a su alrededor, no se vio ninguna acción de fuego, ni siquiera un rastro de ella, sino una piedra que yacía allí, de gran tamaño, que no guardaba casi ninguna proporción con la masa ardiente que se veía en los cielos."

Plutarco - Vidas Paralelas [96-117 d.C]

Esta página ha sido intencionalmente dejada en blanco.

Agradecimientos

A nuestras familias, que han sido un pilar importante durante toda la carrera, donde siempre nos alentaron a seguir adelante en todo momento, apoyándonos en nuestras horas y jornadas de estudio.

A nuestros tutores, Nacho y Gonzalo, por guiarnos a lo largo del proyecto, brindarnos sus conocimientos técnicos y opiniones críticas.

A Manuel y Álvaro por la colaboración y disposición aportada de su parte para la realización del proyecto.

A nuestros respectivos trabajos por la paciencia y consideración que nos tuvieron.

A todos nuestros amigos y compañeros por estar junto a nosotros en este camino.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

En este proyecto se estudiará el problema de predecir la presencia de un objeto celeste denominado bólido a partir de eventos registrados por grabaciones nocturnas realizadas por cámaras All Sky gestionadas por el proyecto BOCOSUR de la Facultad de Ciencias, UDELAR las cuales se encuentran desplegadas en la zona sur de Uruguay. Se discutirá el comportamiento del objeto y se ahondará en los distintos algoritmos que logran calcular características que permitan la distinción de estos eventos frente a otros de distintas índoles, como pueden ser: aviones, pájaros, insectos luminiscentes, entre otros. A partir de estas características se utilizará aprendizaje automático para llevar a cabo la clasificación.

Esta página ha sido intencionalmente dejada en blanco.

Acrónimos

DM - Division Model
FN - Falsos Negativos
FP - Falsos Positivos
FPS - Frames Per Second
FWHM - Full-Width at Half Maximum
MAD - Median Absolute Deviation
MAE - Median Absolut Error
RANSAC - Random Sample Consensus
UAI - Unión Astronómica Internacional
XGBoost - Extreme Gradient Boosting

Esta página ha sido intencionalmente dejada en blanco.

Palabras Clave

Cenit: Punto del hemisferio celeste situado sobre la vertical del observador.

Clusters: Conjunto de puntos que tienen características similares y por lo tanto se pueden agrupar.

Dataset: Conjunto de datos de trabajo.

Horizonte: Círculo máximo de la esfera celeste, normal a la vertical del lugar y a 90° del cenit. En las cámaras all-sky, hace referencia al círculo (o arco del círculo) que limita el cielo visible, y por fuera de este círculo se ubica la zona a descartar de la imagen.

Inlier: Punto que pertenece al modelo planteado.

Ensemble: Conjunto de clasificadores.

Evento: Suceso detectado por el sistema de adquisición.

Folds: Separación del conjunto de entrenamiento en subconjuntos que se utilizan para validación cruzada.

Outlier: Punto que no pertenece al modelo planteado.

Percentil: "Es una medida de posición usada en estadística que indica, una vez ordenados los datos de menor a mayor, el valor de la variable por debajo del cual se encuentra un porcentaje dado de observaciones en un grupo." [Wikipedia contributors, 2022]

Splits: Separación de puntos o datos en distintos conjuntos.

Toma: Todo lo que logra captar la cámara de su campo visual.

Tracking: Seguimiento del evento.

Trade-off: "Es la decisión tomada en una situación en la cual se debe perder, reducir cierta cualidad a cambio de otra cualidad." [Wikipedia contributors, 2021e]

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Esta página ha sido intencionalmente dejada en blanco.

Conceptos Generales

1. Meteoroides

Debido al paso de algún cometa, asteroide o restos de la formación del Sistema Solar, partículas de polvo (formadas por materiales rocoso, o mezcla de hielo y rocas) se encuentran dispersos en el vecindario espacial de la Tierra. La UAI[1] definió como **meteoroide** a un cuerpo proveniente del Sistema Solar que tuviera unas dimensiones de entre 100 μm hasta 50 m de diámetro.

Cuando un meteoroide toma contacto con la atmósfera terrestre se produce un fenómeno luminoso debido a la fricción y se le denomina **meteoro**. Si éste llega a ser lo suficiente luminoso como para adquirir una luminosidad de magnitud aparente[2] de -4 (luminosidad equivalente al del planeta Venus) se le considera una **bola de fuego** o **bólido** (en inglés fireball o bolide). El nombre **bólido** también hace referencia al momento en que la **bola de fuego** produce una detonación producto del contacto con la atmósfera.

Si el **meteoroide** logra traspasar la atmósfera sin desintegrarse completamente y fragmentos del mismo llegan a la superficie terrestre, entonces dichos fragmentos serán denominados **meteoritos**. Una ilustración de esta terminología puede observarse en la Figura 1.

En este trabajo se estudiarán imágenes de bólidos, aunque, dependiendo de la calidad de los sistemas de detección usados, se podría llegar a detectar meteoros de menor brillo. Uno de los objetivos de la red de cámaras, que se describe más adelante, es la determinación de la posible caída de un meteorito en la superficie, y su ubicación.

[1] La Unión Astronómica Internacional (UAI) es la asociación mundial de los astrónomos profesionales. Es la encargada de definir los estándares en la Astronomía.

[2] La magnitud aparente cuantifica el brillo de una estrella o cuerpo celeste observado desde la Tierra.

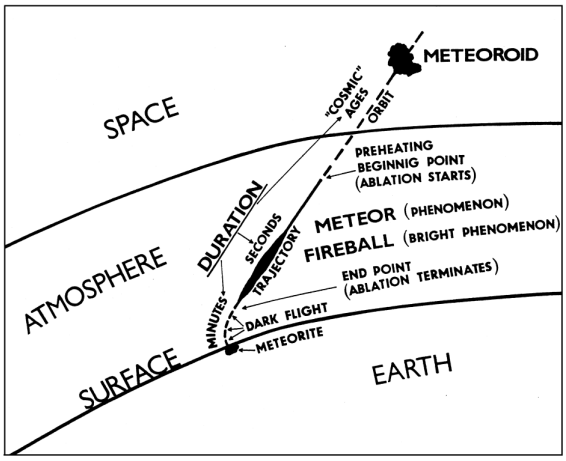


Figura 1: Terminología básica de meteoros. Fuente: [Ceplecha et al., 1998].

2. Lentes Ojo de Pez y Cámaras All-Sky

El lente ojo de pez es una lente de ángulo ultra ancho que produce una distorsión visual fuerte con la intención de crear una imagen panorámica o hemisférica ancha. En una lente de ojo de pez circular, el círculo de la imagen está inscrito en la película o área del sensor.

Su primer uso práctico fue en 1920 para uso en meteorología para estudiar la formación de las nubes, de donde se les dio el nombre de “lentes de cielo entero”(en inglés “all-sky lens” o simplemente All-Sky).

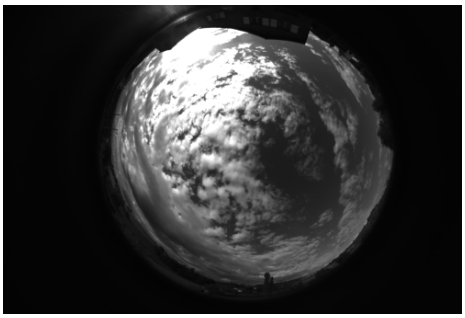
En los visores de las puertas de mirilla se puede encontrar este tipo de lente ya que brinda al usuario un amplio campo de visión.

En las Figuras 2a y 2b se pueden observar ejemplos de fotografías que utilizan este tipo de lente.

Las cámaras All-Sky son utilizadas en la meteorología y astronomía ya que mediante éstas se puede visualizar todo el cielo en una sola toma, utilizando un lente ojo de pez que capta una imagen hemisférica de gran angular.

3. Redes All-Sky para estudio de bólidos

El estudio de meteoritos es de gran importancia científica debido a que éstos como la Tierra y el resto de los componentes del Sistema Solar, están compuestos de los mismos materiales presentes en la nebulosa primitiva de donde se originaron. Los principales componentes son materiales rocosos (como silicatos), minerales (como hierro y níquel), agua y compuestos orgánicos simples. Una teoría que intenta explicar el origen de la vida es la Teoría de la Panspermia que propone que cuerpos como los cometas o asteroides transportan formas de vida simple como bacterias o microorganismos en un estado de hibernación, durante los largos viajes interestelares



(a) Cámara all-sky usada para meteorología. [Alcor System, 2022]

(b) Primera imagen de ojo de pez conocida registrada en 1905 con el aparato de baldes de Wood. [Wood, 1906]

Figura 2: Ejemplos de fotos con lentes de ojo de pez.

o interplanetarios. La llegada de meteoritos con estas formas de vida a la superficie de un planeta como la Tierra, haría posible el desarrollo e intercambio de vida entre los componentes del Universo.

De todas formas, el impacto del estudio de los mismos se ve aplacado si no se posee contexto espacial para interpretar los datos obtenidos de su composición.

"Imagínese intentar comprender la geología de un continente si todo lo que tiene para trabajar es una colección de rocas al azar tiradas en su patio. En eso estamos con los meteoritos. Sabemos que los meteoritos proceden del espacio, sobre todo del cinturón de asteroides. Nosotros podemos señalar su origen preciso en el Sistema Solar determinando la órbita en la que se encontraban antes de chocar con nuestra atmósfera. Sin embargo, esto sólo es posible si somos capaces de seguir a el bólido antes de que aterrice. La combinación de la órbita con el meteorito recuperado es un paso importante para interpretar el registro de los primeros procesos del Sistema Solar que contienen los meteoritos" [Bland et al., 2015].

En los últimos 70 años, distintas organizaciones han sido establecidas con el afán de estudiar las trayectorias de estos cuerpos celestes ya sea para poder predecir dónde caerán y con eso poder estudiar su composición como también lograr determinar la trayectoria de procedencia. En la Figura 3 se ilustra las actuales redes de detección que se encuentran a lo largo del planeta, y en la Tabla 1 se despliega información sobre las mismas indicando país de origen, cantidad de cámaras y ubicación geográfica.

De acuerdo al mapamundi (Figura 3) las zonas coloreadas son mayoritariamente en el Hemisferio Norte por lo que la concentración e investigación de este tipo de redes se realiza en dicho hemisferio. Esto denota la importancia del proyecto Bocosur para impulsar las redes de detección de bólidos en el hemisferio sur y en especial en Latinoamérica.

Tabla de contenidos

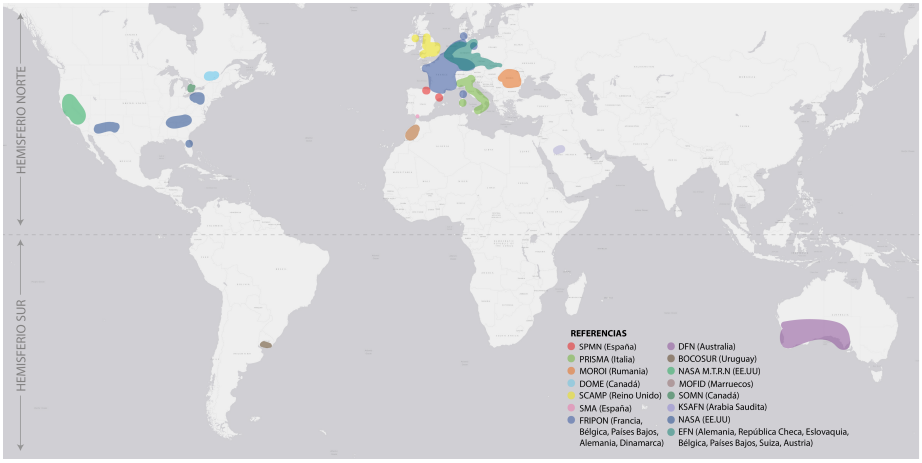


Figura 3: Mapa con las ubicaciones de las redes de detección de bólidos de todo el mundo. Fuente: Elaboración propia con información recabada de distintas fuentes [Aznar et al., 2016], [Wikipedia contributors, 2021a], [Wikipedia contributors, 2021b], [Oberest et al., 1998], [Bland et al., 2015], [Colas et al., 2020], [FRIPON: Database web frontend, 2021], [Space Science and Technology Centre at Curtin University, 2020], [NASA Meteoroid Environment Office (MEO), 2022], [Ames Research Center, 2021].

Organización	País Origen	Ubicación de estaciones/ Redes que la conforman	Total de Estaciones	Hemisferio Sur/Norte
UKFall	Reino Unido	UKMON, SCAMP (Componente UK de Fripon), GFO (Componente UK), Nemetode y AllSky7	>30	Norte
Desert Fireball Network (DFN)	Australia	Ubicado en región oeste y sur de Australia. Incluyendo la Llanura de Nullarbor, el Cinturón de Trigo de Australia Occidental y el desierto australiano.	50	Sur
European Fireball Network (EFN)	Europa Central	Alemania, República Checa, Bélgica, Luxemburgo, Suiza, Eslovaquia y Austria.	34	Norte
GLOBAL FIREBALL OBSERVATORY (GFO)	Colaboración multiinstitucional, con conexiones en todo el mundo	Desert Fireball Network (AU), NASA Meteorite Tracking and Recovery Network (EU), Moroccan Observatory for Fireball Detection (MA), MORP 2.0 (CA), Souther Ontario Meteor Network (CA), UK Fireball Network (GB), Southest Australian Fireball Network (AU), Kingdom of Saudi Arabia Fireball Network (SAU), Tierra del Fuego Fireball Network (ARG), Omani-Swiss Meteorite Project.	$\sim$ 100	Sur y Norte
FRIPON	Francia	Fripon (Francia, Alemania, Suiza y Países Bajos), SPNM (Fripon-España), PRISMA (Fripon-Italia), MOROI (Fripon-Rumania), DOME (Fripon-Canadá), SCAMP (Fripon-Reino Unido)	150	Norte

Tabla 1: Lista de los organismos más grandes y las redes que conforman cada uno de ellos.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 1

Introducción

A continuación, se expondrán los antecedentes del proyecto, se explicará de forma concisa el sistema de detección base de comienzo del proyecto y se culminará la sección introducción con un texto sobre la motivación que llevó a realizar este trabajo.

Uruguay es un país pequeño y con una densidad de población baja lo que implica que las grandes ciudades son escasas y que la contaminación lumínica está concentrada en pocas zonas. Esto permite que uno no deba desplazarse demasiado para poder encontrar zonas donde el cielo nocturno pueda ser contemplado sin distorsiones. Mensualmente se registran reportes de avistamiento de bólidos en nuestro país, y de forma más esporádica entregas de muestras de presuntos meteoritos encontrados en el territorio.

"Estas muestras son entregadas a la Facultad de Ciencias de la UDELAR para su estudio ya que se deben de realizar distintos tipos de análisis para poder verificar su naturaleza. En la casi totalidad de los casos, se trata de de rocas terrestres de apariencia extraña, pero en algún caso se identificó como meteorito o material asociado a un impacto (impactitas)" (G. Tancredi, comunicación personal, 14 de Abril de 2022).

Este tipo de reportes y hallazgos físicos de materiales del espacio exterior son escasos con respecto a los que se podrían obtener si uno observara el cielo constantemente. Las redes de detección son la clave para hacer este proceso más frecuente y que la recolección pueda a llegar a ser suficiente como para poder disponer de una base de datos de elementos extraterrestres.

Los sistemas de adquisición que permiten registrar eventos en el cielo nocturno consisten en estaciones con cámaras all-sky distribuidas por todo el territorio. A este sistema se les incorpora un software para que las mismas tengan cierta autonomía para disparar (trigger) la grabación del evento. Este software suele ser computacionalmente menos exigente que el procesamiento llevado a cabo en la postadquisión. Esto permite que la verificación se realice de forma offline, lo cual tiene ventajas a la hora de no saturar los canales de red y que el sistema no se vea comprometido a procesar una gran cantidad de información al mismo tiempo.

1.1. Antecedentes

El presente trabajo será una incorporación a un proyecto que se encuentra en actividad impulsado por el Departamento de Astronomía, Instituto de Física, Facultad de Ciencias de la Universidad de la República denominado Bocosur "Red de Detección de Bólidos del Cono Sur".

"Bocosur es un proyecto de Ciencia Ciudadana donde estudiantes y docentes del DA-FCIEN, junto a integrantes de comunidades educativas del Interior, participan activamente en la generación de conocimiento científico, colaborando en la operación y mantenimiento de la red, proponiendo mejoras a los prototipos (interacción con docentes/estudiantes de electrónica, robótica, TIC, etc)." (Fuente: *Sitio Oficial de Bocosur*).

En la realización del proyecto Bocosur se destacan los siguientes trabajos los cuales permitieron al equipo responsable adquirir experiencia y conocimiento a lo largo de los años.

- 2002-2013: Diferentes prototipos de cámaras all-sky desarrolladas e instaladas en el Observatorio Astronómico Los Molinos (Ministerio de Educación y Cultura), con la participación de los astrónomos y técnicos del staff y colaboradores del Dpto. de Astronomía.
- 2003: Tesis de Lic. De Astronomía del Lic. J.C Tulic. [J. C. Tulic, 2003]
- 2005: Trabajos finales de los Ing. en Computación F. Kugelmass y G. Chelle.
- 2012: Tesis de Maestría en Física opción Astronomía del Mag. Manuel Caldas. [J. M. Caldas, 2012]
- 2013: Sistema de Nowcasting Solar (SNS) a partir de cámaras todo cielo financiado a través del Fondo Sectorial de Energía de ANII-2013.

1.2. Sistema de adquisición

BOCOSUR al momento cuenta con siete estaciones operativas que se encuentran como se indica en la Tabla 1.1.

Al momento cuentan con tres prototipos de estaciones:

- Prototipo 1: Basado en cámara CCTV Watec 902H2 de alta sensibilidad, lente ojo de pez Arecont Vision (f/2.0, 1.55mm).
- Prototipo 2: Basada en cámara CMOS ASI 178 mm, lente ojo de pez provista por el fabricante de la cámara.
- Prototipo 3: Basado en cámara DSLR profesional Canon EOS 6D y lente ojo de pez Sigma Fisheye (8 mm, f/3.5).

Número de estación	Ubicación	Institución	Fecha de instalación
1	Montevideo	Facultad de Ciencias	Instalación: 2019 Actualización: 10/02/2022
2	San Carlos, Maldonado	Liceo "Monseñor Mariano Soler"	Instalación: 11/03/2020 Actualización: 25/03/2022
3	Casupá, Florida	Liceo "Ramón Goday"	Instalación: 21/02/2022 Actualización: 11/02/2022
4	Castillos, Rocha	Liceo "José Aldunate Ferreira"	Instalación: 10/09/2021 Actualización: 25/03/2022
5	Rosario, Colonia	Liceo "Agustín Urbano Indart Curuchet"	Instalación: 30/09/2021
6	Dolores, Soriano	Liceo "Roberto Taruselli"	Instalación: 01/10/2021 Actualización: 04/2022
7	Trinidad, Flores	Liceo "Carlos Brignoni Mosquera"	Instalación: 15/12/2021 (ya con última versión)

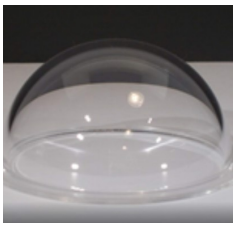
Tabla 1.1: Estaciones actuales de la red. [G. Tancredi, 2022]

Los distintos prototipos utilizados pueden observarse en las Figuras 1.1, 1.2 y 1.3.

El software desarrollado para el control de las estaciones denominado *bolidos*, fue desarrollado en MATLAB por parte del equipo del proyecto BOCOSUR. Esta aplicación permite controlar todos los aspectos vinculados a la detección en una estación en particular, desde el hardware (seteo de parámetros de la cámara y GPS) a parámetros vinculados a la estación, como su nombre, ID y la máscara de horizonte que el usuario puede definir manualmente a partir de una imagen nocturna. Este sistema posee una primera máscara para evitar la generación de videos espurios, debido a luces artificiales y otros artefactos (árboles, edificios, etc).

Toda la información expuesta en la sección de sistema de adquisición fue obtenida explícitamente del *Sitio Oficial de Bocosur*. Ver especialmente los videos del Taller BOCOSUR (*Link al enlace*), donde se describe el sistema y su operación.

En el correr del 2022 se piensa concretar la instalación de 13 estaciones más a ubicarse en la zona centro y norte de Uruguay, totalizando 20 estaciones de la red.



(a) Domo de la cámara

(b) Cámara CCTV

(c) Lente utilizado

(d) Estación

(e) Interior de la estación

Figura 1.1: Elementos, estación e interior de Prototipo 1.

Figura 1.2: Cámara CMOS ASI 178 mm, prototipo 2.

1.3. Motivación

La realización del presente proyecto partió de la búsqueda de un área de investigación que combinara las inquietudes que encontraban los estudiantes en el área de la Astronomía y la Ingeniería Eléctrica.

Poder investigar y buscar una solución a algún problema que se tuviera en el área de la Astronomía que pudiera ser resuelto a partir de los conocimientos adquiridos en Ingeniería Eléctrica fue la principal motivación. Fue así como los estudiantes encontraron al grupo de investigación para el proyecto BOCOSUR, liderado por el Dr. Gonzalo Tancredi de la Facultad de Ciencias. BOCOSUR es un proyecto con muchos años de trayectoria que mezclaba diversas áreas de conocimiento,

(a) Lente utilizada

(b) Cámara dentro de la estación

Figura 1.3: Elementos e interior de Prototipo 3.

Astronomía principalmente, pero también una gran implementación del área de Ingeniería Electrónica impulsada en la actualidad por el Mag. Ing. Manuel Caldas.
¿En qué podía entonces contribuir el grupo? El proyecto tenía una dificultad, que era la clasificación manual de las grabaciones que se obtenían mediante el sistema de adquisición. Esto implicaba horas de trabajo en verificar que si lo que se detectó era un bólido o no. La propuesta era clara, se necesitaba realizar un programa que lograse a partir de un video decidir si el evento detectado es un bólido o no.
De aquí nacieron las posteriores motivaciones: aplicar conocimientos de procesamiento de imágenes, programación, visión por computadora y machine learning; áreas de la Ingeniería Eléctrica en que los estudiantes poseen conocimiento y que en la actualidad son ramas que están en continuo avance y exploración. Como también la motivación en investigar cómo realizar el programa que debiera incorporar todo lo mencionado, de lo cual no se tenía conocimiento y dió gran satisfacción poder incluirla como parte final del proyecto.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 2

Objetivos y Especificaciones

2.1. Objetivos Generales

Desarrollar un sistema de detección automática basado en aprendizaje automático y tratamiento de imágenes capaz de detectar y localizar un evento dentro de una secuencia de video y clasificarlo -como bólido o no-. De esta manera se pretende ahorrar la clasificación manual que actualmente se está realizando que será inviable en el futuro con el incremento progresivo de la cantidad de estaciones. El mismo debe priorizar la tasa de falsos negativos frente a la tasa de falsos positivos ya que la pérdida de un bólido es de mayor importancia frente a la clasificación de algún otro evento como bólido, la cual puede ser descartada de forma manual a posteriori.

2.2. Objetivos Específicos

- El clasificador debe tener una tasa de falsos negativos menor al 5 % sobre la muestra a clasificar, con una probabilidad de éxito de clasificación del 95 %, esto debe ser medible al final de la ejecución cuando el sistema esté en marcha.
- El sistema debe ser capaz de procesar un video en un tiempo no superior a 10 veces su duración.

2.3. Alcance

Se limitará a recibir un video en un formato estándar específico a acordar previamente; no es necesario que maneje más de un formato. El sistema desarrollado deberá ser capaz de detectar y localizar un evento dentro de la secuencia de video. Se generará un archivo de salida, el formato del archivo será único y acordado con la parte interesada. No competen al sistema aspectos como la captura, codificación, el almacenamiento, o la medición de otros datos que no sean los que conciernen a la presencia y localización de bólidos en los videos. No hay requisitos a priori en cuanto

a la eficiencia computacional del método.

2.4. Criterios de Éxito

Los criterios de éxito planteados para lograr un excelente desempeño del proyecto son:

1. La tasa de falsos negativos por debajo del 5 % sin generar más de 10 % de falsos positivos.
2. El sistema deberá de ser capaz de procesar un video en un tiempo no superior 10 veces su duración. Este criterio se medirá utilizando un procesador Intel i5 11400 2,60 GHz.

Se medirá la eficacia sobre un subconjunto de los datos de referencia apartado para tal fin.

2.5. Restricciones

- La ganancia de las cámaras All-Sky está en modo manual para que la ganancia se mantenga constante, si estuviera en automático variaría cuando ingresa un bólido en la toma de la cámara.
- El formato de video es MJPG, no se puede controlar la compresión de los datos.
- Cada estación presenta diferentes tipos de contaminación lumínica de la ciudades donde está instalada la estación, lo que hace imposible definir umbrales/máscaras generales.

Capítulo 3

Marco Teórico

3.1. Movimiento de objetos en cámaras con lente ojo de pez

La trayectoria de un bólido en su pasaje por la atmósfera se puede asumir, en una primera aproximación, como rectilínea, si bien existe una desaceleración diferente en las tres componentes espaciales del vector velocidad. No obstante, debido a la deformación óptica en una lente ojo de pez, es necesario analizar cuál es la proyección de una trayectoria rectilínea en el espacio en la imagen.

Cuando se obtiene una imagen a partir de una cámara digital, ésta es el resultado de un mapeo de un objeto en el espacio tridimensional a una imagen en el espacio bidimensional. Vale aclarar que un video es una composición sucesiva de imágenes.

El modelo referente al principio más simple es el "pinhole" [Wikipedia contributors, 2021c] en donde cada punto del espacio se proyecta en el plano de la imagen sin ningún tipo de distorsión e implica que un punto en el espacio, el centro óptico y el punto proyectado son colineales, con lo cual se puede afirmar que una recta se verá como una recta en la imagen proyectada [E. E. Orozco Guillén, 2007].

No es así para el caso de cámaras con lente ojo de pez, las cuales tienen un modelo no-lineal para su proyección, en donde se puede observar el fenómeno de distorsión radial. La distorsión radial causa esta pérdida de "rectilineidad" de los objetos en su transformación de las coordenadas en el espacio real al plano de la imagen [Hughes et al., 2008].

La Figura 3.1 ejemplifica el esquema de proyección para el modelo ojo de pez donde el ángulo θ corresponde al ángulo de incidencia del haz de luz proveniente del punto P que se quiere capturar con la cámara; debido a que ésta es una transformación no-lineal el ángulo que se forma entre la horizontal y la recta que pasa por el punto p (proyección del punto P en el plano imagen) es $\theta' \neq \theta$. Esto no sería así en el caso ideal de "pinhole" ya que la transformación sería lineal llevando a que $\theta = \theta'$.

Entonces, si las rectas no se transforman en rectas ¿en qué se transforman?. Para

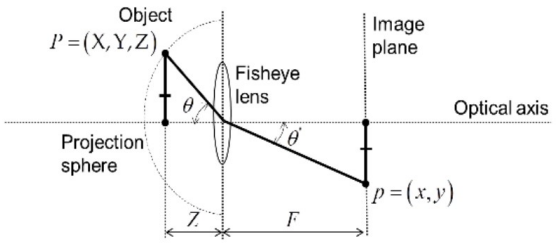


Figura 3.1: Modelo de Proyección para lentes ojo de pez [Lee et al., 2019].

responder a esta pregunta se realizará un planteo matemático a partir de un modelo que logra representar la distorsión radial de forma fidedigna llamado "Modelo de División"[1] (DM).

Sean (x_d, y_d) y (x_u, y_u) las coordenadas en el plano imagen distorsionadas y no distorsionadas respectivamente. El DM describe la distorsión radial mediante la siguiente expresión:

$$r_u = \frac{r_d}{1 + \lambda_1 r_d^2 + \lambda_2 r_d^4 + ...} \tag{3.1}$$

Siendo r_d y r_u el valor del módulo de los puntos (x_d, y_d) y (x_u, y_u) con respecto al origen de coordenadas del plano imagen respectivamente y λ_i son los coeficientes del desarrollo polinómico de distorsión radial. A modo simplificar las cuentas, se usará como modelo el expuesto a continuación ya que es suficiente para la mayoría de las cámaras.

$$r_u = \frac{r_d}{1 + \lambda r_d^2} \tag{3.2}$$

De esta forma las coordenadas de la imagen se expresan como sigue:

$$x_u = \frac{x_d}{1 + \lambda r_d^2}; y_u = \frac{y_d}{1 + \lambda r_d^2} \tag{3.3}$$

con

$$r_d^2 = x_d^2 + y_d^2 \tag{3.4}$$

Considerar la ecuación de una recta para el punto sin distorsión $y_u = kx_u + b$ y sustituyendo en la Ecuación (3.3) se obtiene la siguiente igualdad:

$$\frac{y_d}{1 + \lambda r_d^2} = k\frac{x_d}{1 + \lambda r_d^2} + b \tag{3.5}$$

Reacomodando los términos se llega a la expresión matemática de una circunferencia:

$$x_d^2 + y_d^2 + \frac{k}{b\lambda}x_d - \frac{1}{b\lambda}y_d + \frac{1}{\lambda} = 0 \tag{3.6}$$

La recta que al comienzo fue propuesta fue transformada en una circunferencia en el plano imagen [Wang et al., 2009].

Muchos artículos que hablan sobre cómo tratar con este problema de la distorsión

[1] Modelo de División (Division Model) sugerido por Fitzgibbon [Fitzgibbon, 2001]

radial indican que uno puede utilizar los puntos cercanos al centro de la imagen ya que esta es la zona menos afectada por la distorsión radial debido a que su efecto aumenta conforme uno se aleje del centro.

Los bólidos que ingresen a la atmósfera a la altura del cenit serán visualizados en el centro de la toma de la cámara; de acuerdo a este razonamiento, eventos que realicen una trayectoria rectilínea por el cenit seguirán observándose de la misma forma siempre y cuando su extensión no sea tal que llegue al horizonte de la cámara. Cuando un bólido ingresa a la atmósfera casi a nivel del horizonte, en la cámara su trayectoria será observada con una curvatura debido a este fenómeno.

Estas observaciones serán de importancia para el abordaje del problema a través de la implementación del algoritmo RANSAC (será abordado más adelante en este capítulo).

3.2. Modelo cuadrático

La expresión matemática que representa un modelo cuadrático se expone en la Ecuación (3.7) donde los parámetros $(a_0, ..., a_5)$ se hallarán mediante una regresión lineal.

$$f(x, y) = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2 \tag{3.7}$$

Dado un conjunto de puntos en donde se conoce el valor de z dado por la Ecuación (3.8).

$$z = f(x, y) : (x_j, y_j, z_j : j = 1, ..., n) \tag{3.8}$$

Se convierten los pares (x_j, y_j) en variables auxiliares de dimensión $m = 6$ de acuerdo al mapeo de la Ecuación (3.9).

$$(x, y) \to \mathbf{w} = (1, x, y, x^2, xy, y^2) \tag{3.9}$$

Luego se puede escribir el problema de regresión en términos de los pares (w_j, z_j) según Ecuación (3.10),

$$\mathbf{a}^* = \arg\min_{\mathbf{a}} \sum_j \|\mathbf{w}_j^T \mathbf{a} - \mathbf{z}_j\|_2^2 \tag{3.10}$$

o escrito matricialmente según la Ecuación (3.11).

$$\mathbf{a}^* = \arg\min_{\mathbf{a}} \|\mathbf{W}\mathbf{a} - \mathbf{z}\|_2^2 \tag{3.11}$$

Donde *argmin* referiere al valor de a que minimiza la expresión.

El desarrollo anterior se logra resolver de manera exacta mediante la Ecuación (3.12).

$$\mathbf{a}^* = \left(\mathbf{W}^T \mathbf{W}\right)^{-1} \mathbf{W}^T \mathbf{z} \tag{3.12}$$

El paquete `numpy` incluye una forma eficiente de resolver lo anterior mediante la función `numpy.linalg.lstsq`

Se utilizará este modelo para solucionar problemas de saturación en videos a causa de la configuración manual de la ganancia de la cámara como se verá en el Capítulo 5 Sección 5.1.2.

3.3. RANSAC

RANSAC es la abreviatura de "RANdom SAmple Consensus" [Fishcler and Bolles, 1981] es un algoritmo iterativo para la estimación de los parámetros de un modelo matemático de un conjunto de datos que contiene valores atípicos que se suelen denominar *outliers*.

Para muchos problemas que deben de realizar una estimación de parámetros este tipo de supuestos no se mantienen y se debe considerar que los datos contienen valores atípicos no compensados. RANSAC se diferencia de las técnicas de estimación de parámetros clásicas (por ejemplo, mínimos cuadrados) en la cual no se dispone de mecanismos internos que permitan detectar y rechazar valores atípicos. En las técnicas clásicas se asume que la desviación máxima esperada es función directa a la cantidad de datos y sin importar esta cantidad siempre habrá suficientes valores "buenos" para suavizar cualquier desviación considerable.

Una breve descripción del procedimiento que emplean estas técnicas típicas es comenzar utilizando todo el conjunto de datos para poder obtener los parámetros del modelo. Se localiza el dato más alejado en concordancia con el modelo establecido para el cual se asume que es un valor irregular, se elimina, y se itera este proceso hasta que la máxima desviación es menor a un umbral establecido o hasta que no haya más datos que procesar. Un solo valor atípico incluido en un conjunto de datos favorable puede causar que el procedimiento descrito anteriormente falle como se puede ver en Figura 3.3.

El algoritmo de RANSAC se desprende del pensamiento convencional y parte de un conjunto de datos inicial tan pequeño como sea posible y lo amplía con datos coherentes cuando es posible.

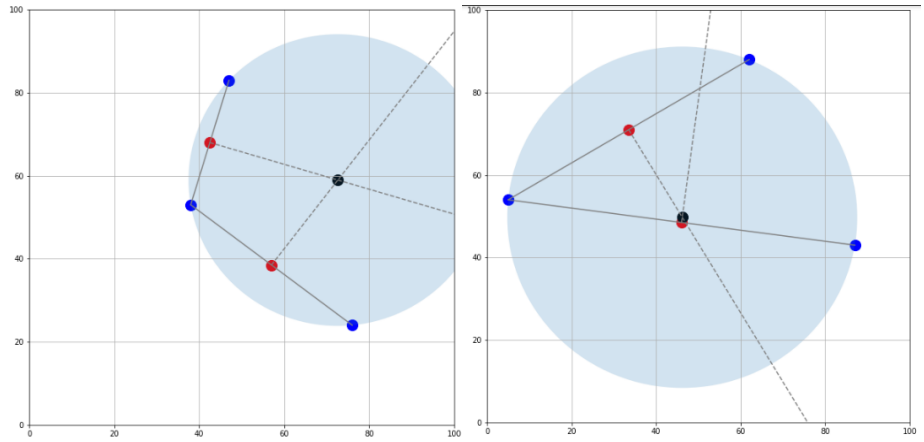
3.3.1. Modelo de circunferencia

Una circunferencia se determina a partir de tres puntos no alineados, y la intersección de las mediatrices de los segmentos de rectas determinados por la unión dos a dos de dichos puntos determinan el centro como se puede ver en la Figura 3.2.

Además, el centro equidista de los extremos de los segmentos de rectas por lo tanto el centro pertenece a la mediatriz de ésta.

Concluyendo que la intersección de dos mediatrices de dos segmentos de rectas determinan el centro de la circunferencia.

Luego el radio se calcula como la distancia del centro a uno de los puntos.



(a) (b)

Figura 3.2: Centro (punto negro) y radio de la circunferencia determinada por tres puntos no alineados (puntos azules). Además se puede visualizar el punto medio (en rojo) de los segmentos de recta determinados.

3.3.2. Ajuste de arco de circunferencia

¿Cómo realiza RANSAC el ajuste de arco de circunferencia? Dado un conjunto de puntos bidimensionales RANSAC necesita seleccionar solamente tres puntos no alineados del conjunto. Esto es porque RANSAC primero busca utilizar la cantidad mínima posible de puntos para comenzar a iterar y segundo la cantidad mínima necesaria para poder determinar una circunferencia son tres puntos.

Luego de seleccionar de forma aleatoria tres puntos del conjunto se calcula el centro y radio de la circunferencia circunscrita que describen. Se realiza el conteo de puntos que estén lo suficientemente cercanos a la circunferencia de forma de plantear una compatibilidad entre los mismos. Si la compatibilidad es suficiente entonces RANSAC empleará técnicas de estimación como mínimos cuadrados para calcular una estimación mejorada de los parámetros de la circunferencia dado que ahora se identificó un grupo mutuamente consistente.

3.3.3. Algoritmo RANSAC

El algoritmo RANSAC procede de la siguiente forma [Flores and Braun, 2011]:

1. Dado un modelo que requiere un mínimo de n puntos para determinar sus parámetros, y un conjunto de datos P tal que el número de puntos en P es mayor que n, se sortea un subconjunto $S1$ de n puntos de P para instanciar el modelo. Con el modelo instanciado $M1$ se determina el subconjunto de decisión S_1^* de puntos de P que están a menos de una distancia t de M_1.
2. Si la cantidad de puntos en S_1^* es mayor que un umbral T entonces se elige el subconjunto de decisión S_1^* para computar el nuevo modelo M_1^*.

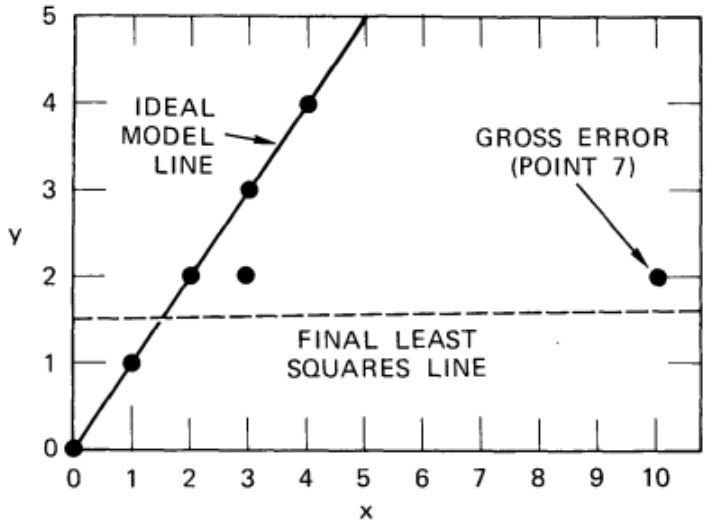


Figura 3.3: Ejemplo de falla de mínimos cuadrados. [Fishcler and Bolles, 1981]

3. Si la cantidad de puntos en S_1^* es menor que T, se sortea un nuevo subconjunto S_2 y se repite el proceso. Si luego de una cantidad de N números de pruebas no se obtiene un subconjunto de decisión que cumpla con el umbral T, se resuelve el modelo con el subconjunto de decisión más grande obtenido, o se termina sin devolver modelo.

3.3.4. Parámetros a determinar

3.3.4.1. Umbral para parámetro de distancia t

t: parámetro que determina si un punto es compatible con el modelo o no.
Este parámetro se determina a partir de los requerimientos que deba cumplir la aplicación, también suele ser determinado de forma empírica. Si se quiere hallar para que cumpla los requerimientos una forma suele ser considerar que la probabilidad de que un punto es compatible con el modelo es α.

3.3.4.2. Cantidad de iteraciones: N

N: cantidad de iteraciones para buscar el subconjunto de decisión.

Sea p probabilidad que el subconjunto que se elige aleatoriamente esté libre de *outliers*. Si se tiene que la probabilidad de que un punto dentro del conjunto de datos sea un inlier es ω, entonces ω se puede expresar como:

$$\omega = \frac{\#\text{de inliers en el conjunto}}{\#\text{de puntos en el conjunto}} \tag{3.13}$$

Suponiendo que la cantidad n necesaria para determinar el modelo son seleccionados de forma independiente, entonces si se considera la probabilidad ω^n, ésta será la probabilidad de que todos los n puntos sean *inliers*. De la misma forma que el valor de probabilidad $1 - \omega^n$ será el correspondiente a la probabilidad de que al menos uno de los n puntos sea un valor atípico. Si a su vez, se eleva toda la expresión a la N, es decir $(1 - \omega^n)^N$ será entonces la probabilidad de que el algoritmo nunca seleccione un subconjunto de n datos *inliers* y por lo tanto será igual a la probabilidad $1 - p$. A partir de lo expresado se tiene la Ecuación (3.14).

$$1 - p = (1 - \omega^n)^N \tag{3.14}$$

Aplicando logaritmo a ambos lados de la Ecuación (3.14) y despejando N, se obtiene la Ecuación (3.15).

$$N = \frac{log(1 - p)}{log(1 - \omega^n)} \tag{3.15}$$

Se deberá tener en consideración el supuesto que los n puntos deben de ser seleccionados de forma independiente ya que de seleccionar dos veces el mismo valor generará problemas, como ejemplo para el caso lineal, si RANSAC elige dos valores en cada iteración y calcula la recta entre los dos puntos es fundamental que éstos no sean el mismo valor ya que sino no sería realizable [Wikipedia contributors, 2021d] [Flores and Braun, 2011].

3.3.4.3. Umbral T

T: valor que corresponde a la cantidad mínima de puntos compatibles para considerar una buena estimación del modelo.

Esta cantidad deberá de ser similar a la cantidad de *inliers* que se supone que hay en el conjunto de datos. Dado que se necesitan n valores para establecer el modelo y ω la probabilidad de un punto del conjunto sea *inlier* entonces T se tomará como $T = \omega n$. [Flores and Braun, 2011]

3.4. FWHM - Intensidad de brillo

3.4.1. FWHM

Se define Full-Width at Half Maximum como la distancia al centro de masa a la cual su intensidad decae un 50 % respecto a la intensidad de brillo del centro de masa.

El FWHM nos permite calcular el área cubierta por el evento en la imagen; lo que se usará para calcular la intensidad total del evento como la integral bajo el área (ver Capitulo 3 Sección 3.4.2).

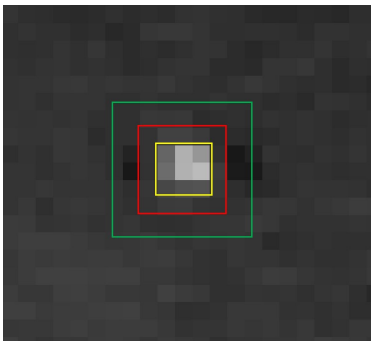


Figura 3.4: Ejemplo de aplicación del método de Fotometría de apertura a partir de la Ecuación (3.16) y las diferentes regiones de interés. Región verde y rojo: fondo de cielo. Región roja y amarilla: zona donde aún puede estar contaminada por la extensión del evento detectado. Región amarilla zona del evento detectado.

3.4.2. Fotometría de apertura

Con la fotometría de apertura hacemos una estimación digital de la intensidad de brillo de forma análoga a lo que se realizaba con un fotómetro[2]. Se mide la contribución de luz en la región donde está el evento y se le resta la contribución del cielo de fondo como se puede ver en la Ecuación (3.16) y la Figura 3.4.

$$I = \sum_i \sum_j I_{ij} - n_{pix} I_{cielo} \tag{3.16}$$

Parámetros en Ecuación (3.16):

I_{ij}: Valor de intensidad de brillo para las posición de píxel de coordenadas (i, j).

n_{pix}: Cantidad de píxeles en el área del evento detectado. En la Figura 3.4 es la región dentro del cuadrado amarillo.

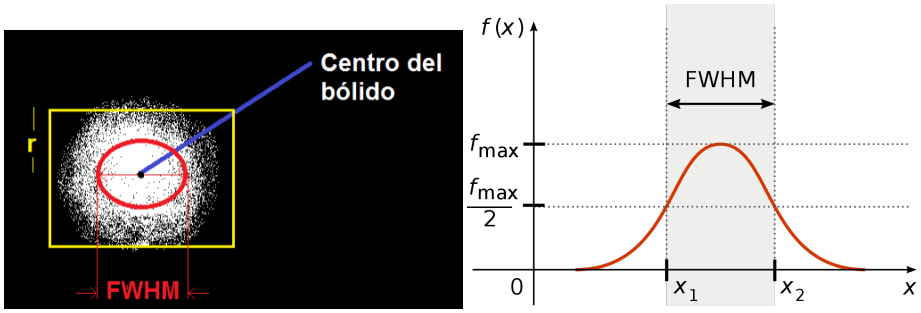
I_{cielo}: Brillo del fondo de cielo por píxel.

La Figura 3.4 muestra el caso de un evento donde se le definirán las regiones tomadas en cuenta para el cálculo de la Ecuación (3.16).

- Región delimitada por el cuadrado verde y rojo es lo que se considera “fondo de cielo”.
- Región delimitada por el cuadrado rojo y amarillo es una región intermedia que sólo se utiliza a modo de referencia para poder indicar la zona donde aún puede estar contaminada por la extensión del evento detectado.
- Región por dentro del cuadrado amarillo es el obtenido mediante el FWHM y un múltiplo de éste que indica la región donde comprendería el evento detectado.

La gran ventaja de este método para calcular la intensidad del brillo es que solo se necesita una imagen para poder aplicarlo. En contraposición el problema que presenta es que la estimación de la intensidad de “fondo de cielo” puede verse afectada por

[2]Un fotómetro es una herramienta que permite medir la luz en un punto concreto.



(a) (b)

Figura 3.5: Izquierda: Estimación de FWHM de un bólido, imagen extraída de la Tesis de Maestría de Manual Caldas [J. M. Caldas, 2012]. Derecha: Exposición del valor FWHM sobre un perfil de una curva gausiana - f(x) -.

la interferencia de objetos cercanos, lo que tiene como consecuencia es un valor de intensidad menor al real.

Este inconveniente es evitable cuando se cuenta con una secuencia de imágenes —video— del evento. Al tener mayor información se puede calcular de forma precisa el valor de "fondo de cielo" para toda la toma en la que está presente el evento. Por lo tanto obteniendo así un valor más preciso del valor real de la intensidad del brillo del evento.

3.5. Centro de Masas

Otro de los parámetros que caracterizan al objeto es su centro de masas, el cual representa la ubicación del centro del objeto tomando en cuenta, en forma ponderada, las contribuciones de brillo ("masas") de los píxeles que ocupa la imagen del objeto. Un concepto similar es el de centroide, éste da un valor de centro geométrico mientras que el centro de masas da un valor central tomando en cuenta la distribución de masa. En nuestro caso, se optó por utilizar la referencia de centro de masa pero aplicado a una imagen, en la cual en vez de contar con una distribución de masa se utiliza la distribución de intensidad de brillo para calcularlo.

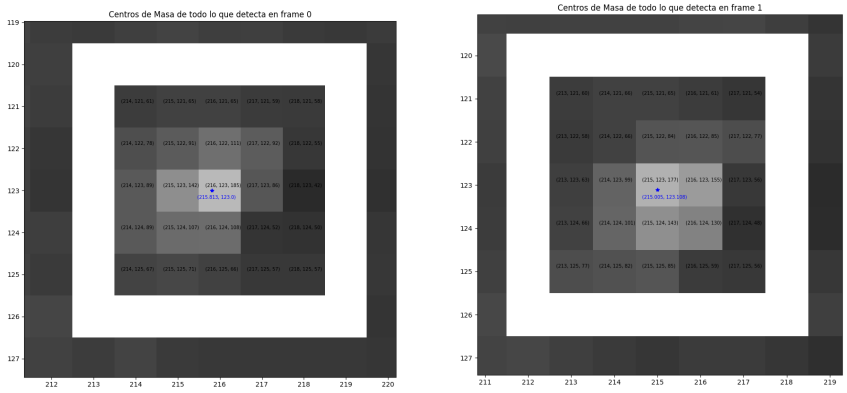
En las Ecuaciones (3.17) y (3.18) se expone las ecuaciones que se utilizan para calcular la componente horizontal y vertical respectivamente, dando por centro de masa al par dado por estos dos valores, (x_c, y_c).

Hay que tener en cuenta que una imagen, es un conjunto de píxeles donde cada píxel tiene un valor de coordenada correspondiente en un plano cartesiano, y donde el punto inicial $(0, 0)$ se encuentra en la esquina superior izquierda.

$$x_c = \frac{\sum_i x_i \left(\sum_j I_{ij}\right)}{\sum_i \sum_j I_{ij}} \tag{3.17}$$

$$y_c = \frac{\sum_j y_j \left(\sum_i I_{ij}\right)}{\sum_i \sum_j I_{ij}} \tag{3.18}$$

Parámetros en Ecuaciones (3.17) y (3.18):
- x_c: valor de centro de masa correspondiente a la coordenada horizontal.
- y_c: valor de centro de masa correspondiente a la coordenada vertical.
- x_i: valor de coordenada i-ésima horizontal de los píxeles del evento detectado.
- y_i: valor de coordenada i-ésima vertical de los píxeles del evento detectado.
- I_{ij}: valor de intensidad de brillo para la coordenada {i,j}.



(a) (b)

Figura 3.6: Ejemplos de cálculo de centros de masa. En cada píxel se puede apreciar impreso x_i, y_i e I_{ij} `Video: Station_1_2020-11-01-01-52-50.avi`

3.6. Suavidad

Definiremos un indicador de suavidad de una curva, a partir de la longitud de la misma, que se calcula según la Ecuación (3.19),

$$L = \sum_{k=0}^{N-1} l_k \tag{3.19}$$

Donde l_k se obtiene de la Ecuación (3.20):

$$l_k^2 = ((k+1) - k)^2 + (f[k+1] - f[k])^2 \tag{3.20}$$

Siendo k el índice del frame y $f[k]$ el valor del indicador (por ejemplo, el FWHM o la intensidad) en el k-ésimo frame.

$$l_k^2 = 1 + |f[k+1] - f[k]| \tag{3.21}$$

Por lo tanto la suma cuadratica de cada segmento l_k de la curva f cumple la

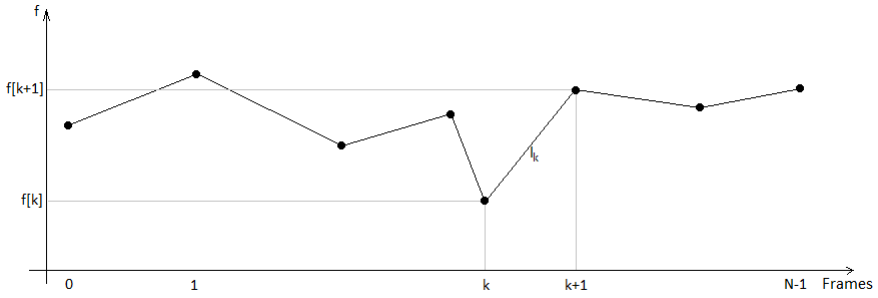


Figura 3.7: Ejemplo gráfico para el cálculo del indicador suavidad según la Ecuación (3.20).

siguiente igualdad:

$$\sum_{k=0}^{N-1} 1 + |f[k+1] - f[k]| = N + \sum_{k=0}^{N-1} |f[k+1] - f[k]|| \tag{3.22}$$

Si se opta por normalizar los datos para que todos estén referenciados a la misma cantidad de puntos N -como referencia los fps- se puede tomar como indicador de suavidad S:

$$S = \sum_{k=1}^{N-1} f[k+1] - f[k] \tag{3.23}$$

3.7. Árboles de decisión

Los árboles de decisión son un método de aprendizaje supervisado usado para regresión o clasificación. Para entender como funciona y hace las predicciones se presenta un ejemplo a continuación.

Dado un conjunto de datos sobre la flor de iris, se utiliza un árbol de decisión para clasificar entre los distintos tipos que existen, setosa, virginica y versicolor. Dentro de las características que se incluyen están el largo y ancho de los pétalos. A partir de estas dos características y los datos etiquetados al entrenar el árbol éste va creando reglas que le permite diferenciar los tipos de iris [Aurélien Géron, 2019].

Luego de entrenar el modelo se obtiene el árbol de decisión de la Figura 3.8, suponiendo que se quiere clasificar una flor de iris, se empieza por la raíz del árbol (profundidad cero) y se observa la longitud del pétalo y se pregunta si es menor a 2.45 cm, si es verdadero se moverá a la hoja izquierda (profundidad 1), donde ya la clasifica como clase setosa. En caso contrario, se deberá fijar en el ancho del pétalo donde se tiene como umbral 1.75 cm, de ser menor se clasifica a la flor como versicolor mientras que si es mayor como virginica. Un mapeo de los umbrales y las decisiones se puede observar en la Figura 3.9.

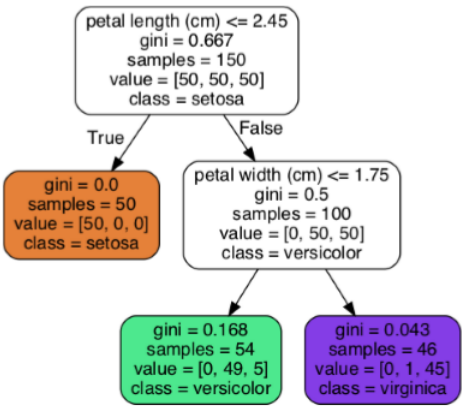


Figura 3.8: Árbol de decisión [Aurélien Géron, 2019].

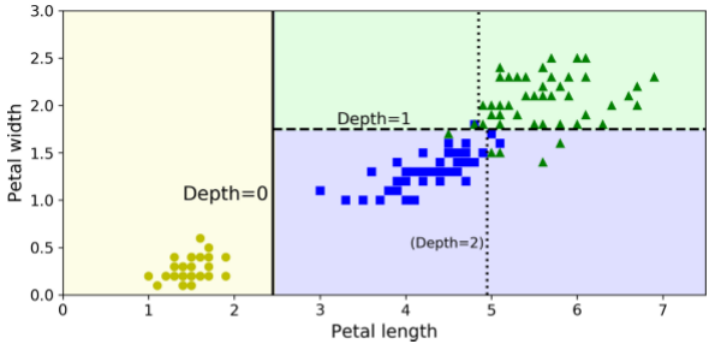


Figura 3.9: Mapa de decisión [Aurélien Géron, 2019].

3.8. Gradient Boosting

Gradient Boosting es una técnica de aprendizaje supervisado utilizada para problemas de regresión y clasificación. El modelo final es un ensemble de clasificadores que típicamente son árboles de decisiones, conocido como gradient-boosted trees.

Boosting hace referencia al método de ir entrenando los árboles de manera secuencial, donde cada árbol luego del modelo inicial tiene como objetivo el residuo del anterior, ver Figura 3.10.

La idea general detrás de esto es que se pondrá énfasis en las instancias que son difíciles de predecir correctamente (casos "difíciles") durante el aprendizaje, de modo que el modelo aprenda de los errores del pasado. La predicción final se forma con todos los clasificadores.

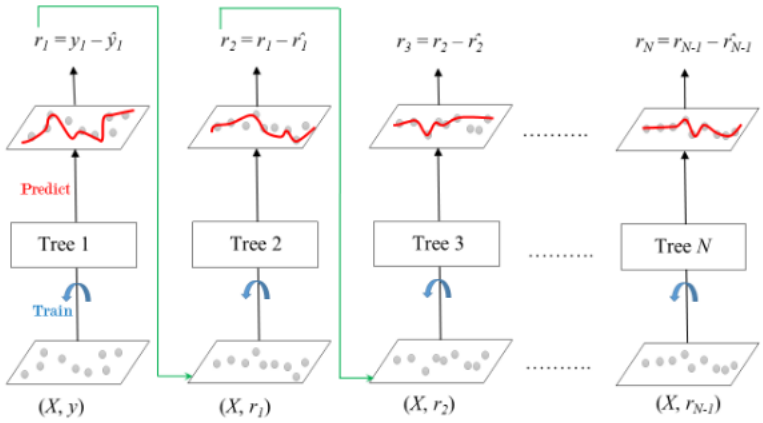


Figura 3.10: Diagrama del algoritmo Gradient Boosting.

3.9. Precision y recall

Muchas veces la tasa de aciertos del clasificador (accuracy) no es una métrica confiable, sobre todo frente a problemas con clases desbalanceadas. De esta forma se definen métricas que permiten definir de manera más precisa la efectividad del clasificador.

Precision, es la tasa de aciertos en las predicciones positivas.

$$precision = \frac{TP}{TP + FP} \tag{3.24}$$

Donde TP es la cantidad de verdaderos positivos mientras que FP la cantidad de falsos positivos.

El **recall** por otro lado es el ratio de instancias positivas que son correctamente detectadas por el clasificador.

$$recall = \frac{TP}{TP + FN} \tag{3.25}$$

Donde FN hace referencia a la cantidad de falsos negativos [Aurélien Géron, 2019].

Se utilizarán estas medidas para medir el desempeño del clasificador ya que en este trabajo se presenta un problema con clases desbalanceadas.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 4

Problema

El problema parte de analizar en una serie de grabaciones las características que diferencian a un bólido de otro evento, con el fin de idear e implementar algoritmos que permitan obtener indicadores para distinguir la presencia o no de un bólido.

4.1. Abordaje del Problema

El abordaje del problema se llevó a cabo en varias etapas las cuales fueron plasmadas en el diagrama de bloques de la Figura 4.1.

Se comienza por entender y analizar la salida del sistema de adquisición descrito en el Capítulo 1: Sección 1.2.

La salida del sistema es una grabación de video en formato .avi en escala de grises que no dura más de una decena de segundos. Cada pixel es un valor entero correspondiente a un nivel de luminosidad en el rango de 0 a 255, donde el valor 0 corresponde a una luminosidad mínima -negro- y 255 a una luminosidad máxima -blanco-. Cada valor en el rango corresponde a un porcentaje entre blanco y negro que genera la gama de escala de grises.

Es necesario dividir la grabación en frames que serán imágenes de tamaño 640x480 píxeles en tiempos consecutivos a lo largo de la duración de éste para poder hacer un estudio más simple logrando eliminar la componente temporal. Esto es así porque una grabación es un elemento de tres dimensiones, como se explicó en el Capítulo 3: Sección 3.1, cada frame es una imagen bidimensional tomada a tiempos t_i diferentes.

Una vez obtenidos los frames, se localiza dónde está ocurriendo el evento que hizo disparar la grabación y por tanto es necesario ubicarlo calculando su centro de masa "frame a frame".

Con esta información es posible crear un vector de posiciones de centros de masas para ser introducido a un algoritmo que permita ajustar de forma lo más fidedigna posible la trayectoria del evento. Esto permitirá descartar puntos espurios que puedan ocasionar cálculos erróneos.

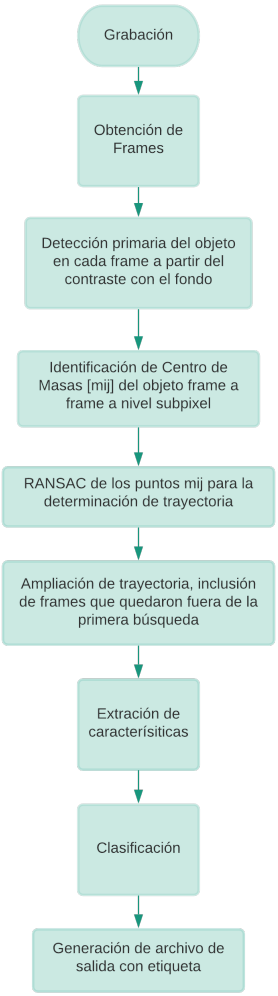


Figura 4.1: Diagrama de bloques del proyecto.

Una vez estimada la trayectoria se buscan puntos adicionales sobre ella que puedan ser parte del evento y que hayan sido descartados en la primera etapa.

Localizados todos los puntos del evento, el siguiente paso es determinar el tipo en cuestión. Esto se aborda mediante una metodología de aprendizaje automático. Según este esquema, se extraen características relevantes (producto de la observación realizada por humanos) y se entrena un modelo de clasificación.

4.2. Características del evento

Para comenzar a entender cómo generar una solución al problema es necesario la visualización de las grabaciones, para poder a partir de ellas extraer conclusiones sobre, por ejemplo, ¿qué particularidad tiene éste frente a la grabación de un avión, o de un insecto luminiscente?, ¿qué tan brillante es?, ¿tiene un movimiento errático?, ¿suelen aparecer en ciertas zonas más que en otras?

La primera pregunta que se hizo el grupo fue cómo es que un humano empieza a aprender a diferenciar cuándo es un bólido y cuándo no. De la misma forma en que un humano aprende a distinguir mediante la experiencia y las características particulares del evento, el aprendizaje automático ha demostrado ser una herramienta muy capaz de realizar esta tarea.

De forma preliminar, las observaciones que se obtuvieron fueron las siguientes:

- Bólidos: muy luminosos, por lo general de corta extensión espacial como también temporal. Siempre se "encienden" y se "apagan". Hace referencia a que el bólido no comienza brillante en la grabación. El mismo llega a su brillo máximo a la mitad de su trayecto y luego se desvanece. En algunos casos estallan, generando picos de luminosidad muy marcados.
- Aviones: luminosos, trayectorias largas, por lo general a nivel de horizonte de la cámara, su velocidad a simple vista es constante y la baliza del avión genera un patrón de intensidad repetitivo y fácilmente identificable.
- Insectos: no muy luminosos, movimientos erráticos en la mayoría de los casos, tamaños grandes debido a que muchos son grabados pasando por arriba del domo de la cámara.

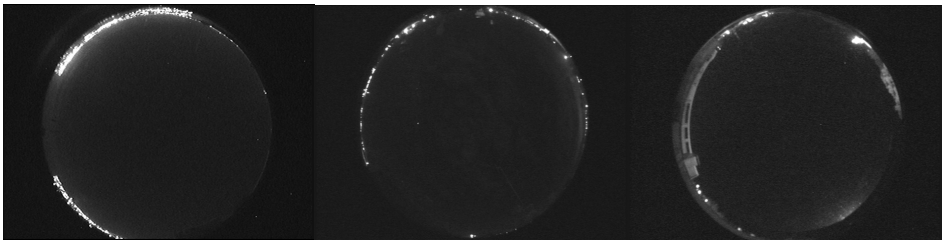
4.3. Desafíos

4.3.1. Ruido en el horizonte de la toma

El primer desafío a resolver es que cada video presenta en la toma un horizonte luminoso y variable, en el cual se ven desde autos, edificios, alumbrado público, etc, generando un montón de puntos candidatos que no pertenecen al objeto de interés o simplemente disparando la adquisición del video sin un evento propiamente dicho debido a variaciones en la intensidad de luz sobre el fondo "estático" del video.

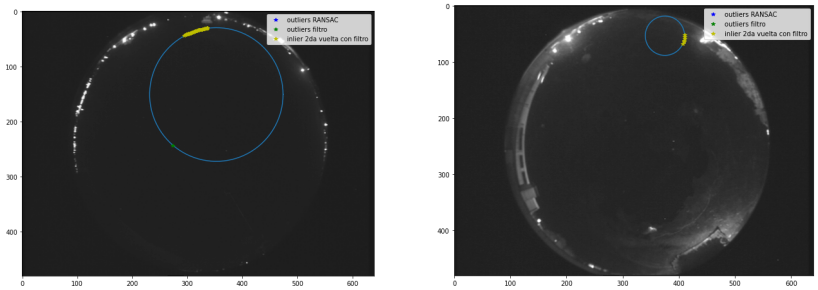
Como las estaciones están en diferentes locaciones, presentan horizontes diferentes, por lo que se tuvo que diseñar un enmascarado que sea particular de cada estación. Para ello se tuvo en consideración hacer que el enmascarado además de particular que pueda ser dinámico e interactivo con el usuario.

En la Figura 4.2 se pueden apreciar los diferentes horizontes, en el caso de la primera estación (Figura 4.2a) se puede ver mayor densidad de luces sobre el horizonte pero no se aprecian edificaciones debido a que está situada en el techo de la Facultad de Ciencias en Montevideo a una altura considerable. La segunda estación (Figura 4.2b) situada en el liceo de San Carlos al igual que la primera se puede apreciar un horizonte



(a) Toma de la estación 1 (b) Toma de la estación 2 (c) Toma de la estación 3

Figura 4.2: Frames aleatorios de distintos videos en donde se pueden apreciar los diferentes horizontes luminosos para las tomas de las estaciones 1, 2 y 3.

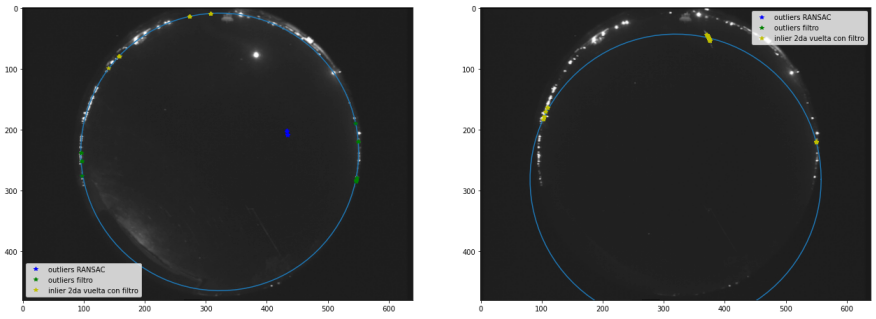


(a) (b)

Figura 4.3: En ambos casos se puede visualizar la cercanía del bólido al horizonte, de ser demasiado laxos con la construcción de la máscara se podrá perder parte o la totalidad del bólido.

contaminado por luces pero en menor medida, ya que es una ciudad mucho más chica. En tercer y último lugar (Figura 4.2c) la estación situada en el liceo de Casupá se puede apreciar en el horizonte edificaciones cercanas que deben de ser descartados de la zona de interés por posible intervención humana. Además solo se aprecian un par de focos cercanos ya que es un pueblo pequeño.

El enmascarado debe ser lo más preciso posible ya que un exceso en el mismo reduce la zona de interés en la que ocurren los eventos, pudiéndose así verse afectada la visualización de todo el evento e interferir luego en la clasificación del mismo por pérdida de información. En casos extremos incluso podría no llegarse a detectar el evento. En la Figura 4.3 se puede ver dos casos de bólidos cercanos al horizonte. Por el contrario, si se realiza una máscara donde no se logre eliminar el ruido se podría generar detecciones innecesarias que generen errores en la estimación de la trayectoria del evento de interés y como consecuencia una mala clasificación como se puede ver en la Figura 4.4.



(a) Bólido excluido del modelo determinado (b) Bólido incluido en el modelo determinado

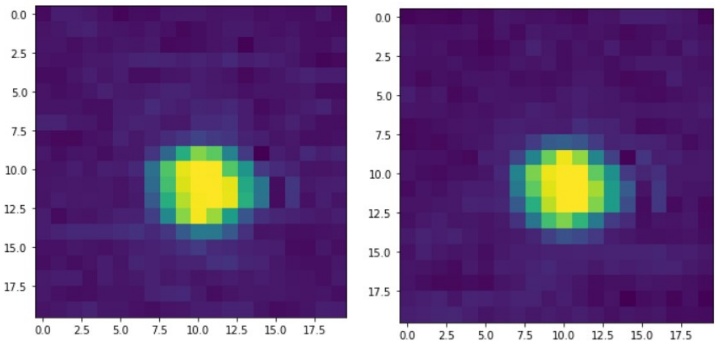
Figura 4.4: En ambos casos luego el clasificador lo etiquetaría como no bólido como consecuencia de la dispersión entre los puntos detectados.
En azul los outliers por distancia del modelo determinado por RANSAC, en amarillo los puntos candidatos a evaluar.

4.3.2. Saturación de las cámaras

Otro de los desafíos a resolver es la saturación que produce el sensor de la cámara en algunos frames del video (ver Figura 4.5). Si la luminosidad del objeto supera el valor 255, el valor digital registrado saturará en dicho valor.

La saturación tiene como consecuencia una mala estimación del FWHM debido a que el calculo parte del máximo valor del objeto. Como es un indicador del tamaño es fundamental estimarlo de forma precisa ya que luego se utilizará como característica de clasificación.

Otra consecuencia es en el cálculo de la intensidad de brillo del bólido que es afectado de forma directa ya que se pierde información de los valores reales producto de la ganancia manual de las cámaras.



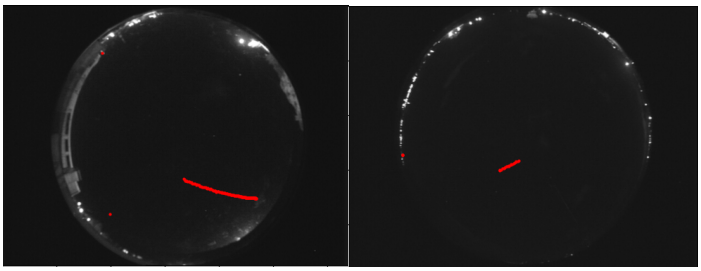
(a) Frame 84 de video ejemplo (b) Frame 85 de video ejemplo

Figura 4.5: Caso Ejemplo: sin reconstrucción de punto saturado. Se puede apreciar que en el centro de la figura en color amarillo permanece constante por varios pixeles.

4.3.3. Modelo de la trayectoria

Dado un estado estacionario del cielo capturado por la cámara donde no ocurre ningún evento, se entiende como evento de brillo positivo al evento que ocurre con una intensidad superior al del estado estacionario. Dado un evento con brillo positivo registrado por el sistema de adquisición, el tracking tiene como objetivo extraer las coordenadas del centro de masa del objeto junto a su marca de tiempo, y su seguimiento mediante los frame. Para poder hallar los eventos se debe definir un umbral para la intensidad lo suficientemente bajo para detectar el recorrido del bólido desde que entra a la atmósfera hasta que se está apagando, pero no tanto como para que las fluctuaciones del fondo tomen protagonismo. Si se toma un umbral demasiado alto, se perderá parte del comienzo y fin del bólido ya que estos instantes del recorrido pueden ser menos intensos, por lo contrario si se toma un umbral muy bajo se detectan cambios sobre el fondo estático sin relación al objeto que se deben descartar.

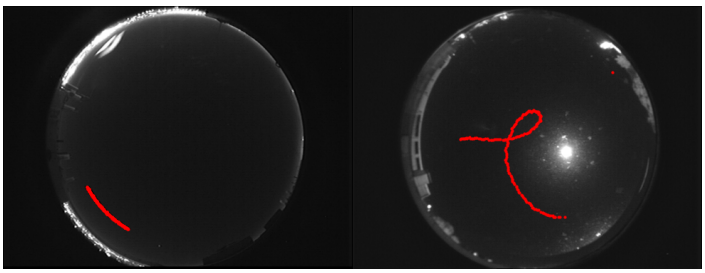
Una vez que se tiene el seguimiento del evento en el tiempo y espacio, se debe distinguir entre los puntos que forman parte de éste y los que no, o sea se debe poder distinguir los casos de inliers/outliers y así simplificar la extracción de características propias para luego realizar la clasificación.



(a) Apreciación de la curva consecuencia de la deformación de la cámara ojo de pez.

(b) En este caso al ser una trayectoria corta y cerca del cenit no se visualiza la deformación.

Figura 4.6: Ejemplos de eventos capturados.



(a) Trayectoria curva de un avión cerca del horizonte de la toma.

(b) Trayectoria de un animal volador reflejado por la luna.

Figura 4.7: Ejemplos de eventos capturados.

Capítulo 5

Solución

5.1. Funciones Implementadas

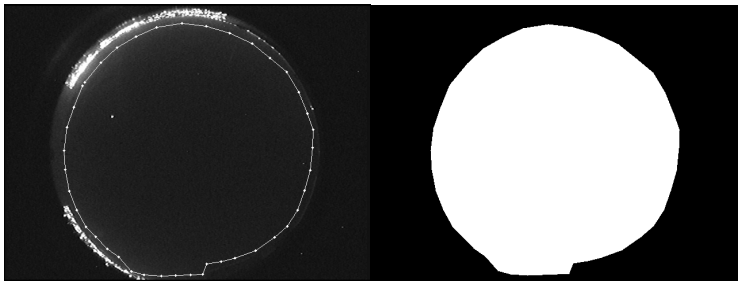
Los pseudocódigos y diagramas de flujo de las funciones implementadas para la resolución de estos desafíos así como las funciones implementadas para la obtención de características se pueden ver en el Apéndice B.

5.1.1. Enmascaramiento

Como se adelantó en el capitulo anterior para solucionar este desafío se recurrió a un enmascaramiento de los frames, determinando una zona de interés en la que se espera que esté el evento, excluyendo la variabilidad del horizonte. Para llegar a la implementación final, se pasó por un proceso evolutivo de varias versiones de máscaras. La primera versión consistía simplemente en un círculo con centro y radio fijo para todas las estaciones, el objetivo no era resolver el enmascaramiento de forma precisa, particular ni dinámica, su objetivo era permitirnos analizar la detección del evento en el frame.

Las siguientes dos versiones, se basaron en buscar un círculo centrado en la toma, pero cuyo radio busque optimizar la zona de interés a partir del estudio de la intensidad de brillo de la toma con respecto al fondo estático. Si bien se mejoraron los objetivos del enmascaramiento, aún se trabajaba sobre supuestos como que el centro de la toma era el centro del frame, lo cual no era correcto.

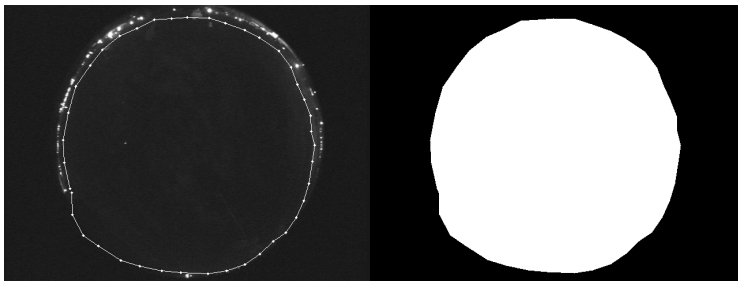
En la cuarta versión se utilizó la transformada de Hough para buscar el "círculo" formado por el horizonte, generando nuevamente una máscara circular de centro x_c, y_c y radio r. En este momento se había logrado un enmascaramiento particular para cada estación y dinámico bajo variaciones del horizonte, pero seguía careciendo de precisión bajo el supuesto de que siempre se enmascara con un círculo, por lo que en conjunto con el equipo de Bocosur se decidió cambiar de un proceso automático a un proceso interactivo llegando a la versión final.



(a) Frame con máscara delimitada punto a punto.

(b) Máscara generada.

Figura 5.1: Enmascaramiento de la estación 1.

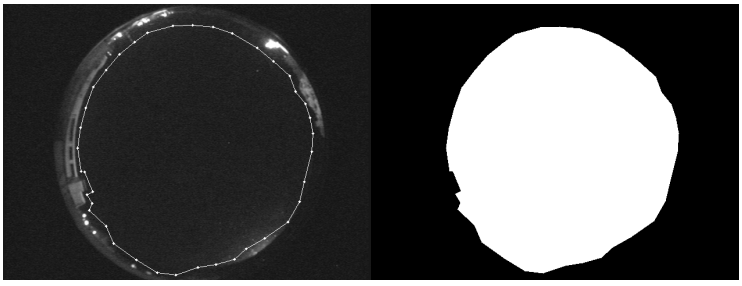


(a) Frame con máscara delimitada punto a punto.

(b) Máscara generada.

Figura 5.2: Enmascaramiento de la estación 2.

Por último se implementó un enmascaramiento donde se despliega en una ventana emergente un frame aleatorio del video seleccionado, y el usuario debe ir delimitando con ayuda del mouse la máscara deseada. Con este nuevo proceso se logró resolver el desafió con sus objetivos como se puede ver en las Figuras 5.1, 5.2 y 5.3.



(a) Frame con máscara delimitada punto a punto.

(b) Máscara generada.

Figura 5.3: Enmascaramiento de la estación 3.

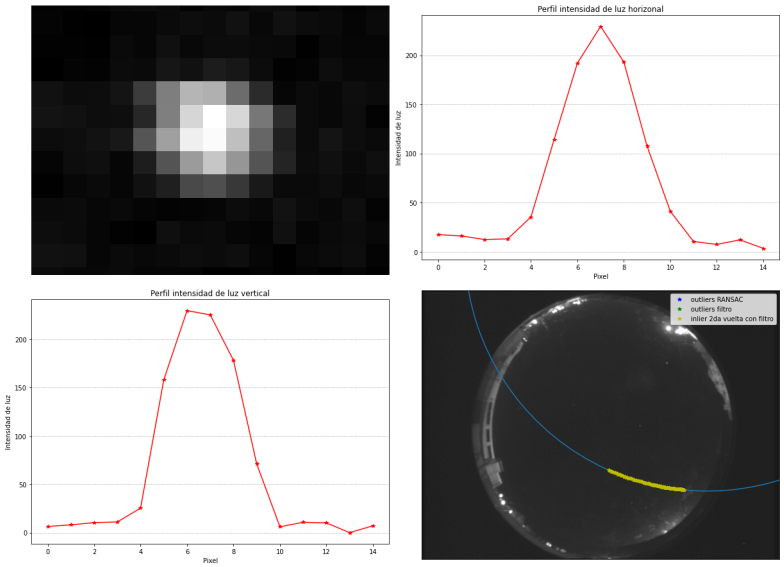


Figura 5.4: Arriba Izquierda: Bólido amplificado. Arriba Derecha: Perfil de la intensidad de luz horizontal por el medio del bólido. Abajo Izquierda: Perfil de la intensidad de luz vertical por el medio del bólido. Abajo Derecha: Trayectoria del bólido de ejemplo.

5.1.2. Reconstrucción de brillo

El desafío de reconstruir el brillo en una región saturada fue resuelto utilizando el modelo cuadrático del Capítulo 3: Sección 3.2 de una parábola invertida en $\mathbb{R}^2$, dado que se observó que los perfiles de intensidad de los eventos de interés no saturados seguían esta forma como se puede ver en la Figura 5.4.

Para la reconstrucción se utilizan los píxeles cercanos que superan un umbral de intensidad ajustable y que no están saturados. Con esta información se puede determinar los coeficientes de la parábola. Finalmente los valores saturados se sustituyen por los valores de la parábola en los puntos correspondientes.

En las Figuras 5.5 y 5.6 se pueden visualizar dos ejemplos donde hay presencia de saturación y su reconstrucción. Esta reconstrucción se realiza frame a frame, por lo que el modelo se calcula siempre y cuando hayan píxeles saturados.

5.1.3. Tracking

El seguimiento del objeto se realizó en dos instancias, en la primera el objetivo consistió en registrar el evento, detectando las coordenadas del punto con más brillo positivo sobre el fondo estático del video cuando supera un umbral igual al percentil 20 % de la intensidad de todo el frame. Con estos puntos detectados se determina su centro de masa para realizar su seguimiento en los frames. Una vez creado el vector de posiciones del centro de masa en los frames se ajusta a un modelo de trayectoria.

Como segunda instancia, y con un modelo de la trayectoria descrita por el objeto, se puede realizar una segunda búsqueda más especifica, con el objetivo de obtener más

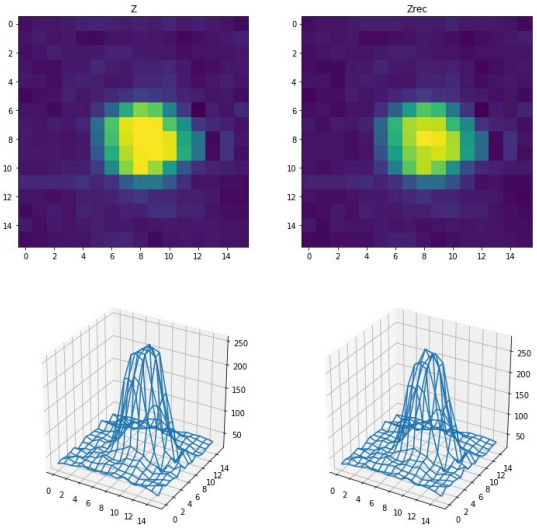


Figura 5.5: Caso Ejemplo 1: Reconstrucción de píxeles saturados. Izq. caso original, Der. caso reconstruido.

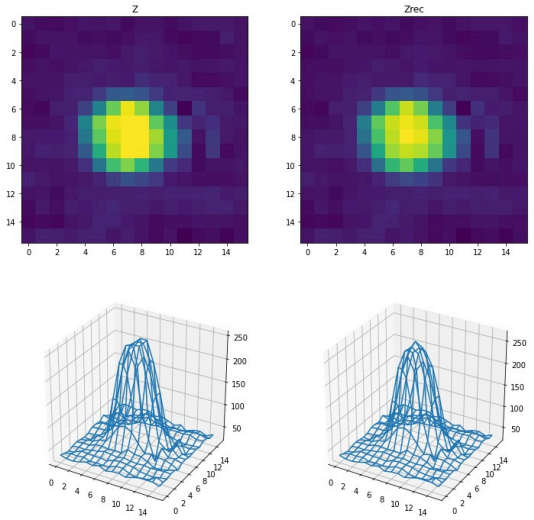


Figura 5.6: Caso Ejemplo 2: Reconstrucción de píxeles saturados. Izq. caso original, Der. caso reconstruido.

puntos del objeto que se nos escaparon en la primera búsqueda reduciendo el umbral al percentil 5 %.

Para determinar el modelo de la trayectoria descrita se utilizó el algoritmo RANSAC [Fishcler and Bolles, 1981] el cual es un método iterativo para calcular los parámetros de un modelo matemático de un conjunto de datos observados que contiene valores atípicos.

Al principio se tomó un modelo lineal, el cual funciona relativamente bien para

eventos cercanos al cenit y que no son lo suficientemente largos como para empezar a deformarse al acercase al horizonte como se mencionó en el marco teórico, pero es inexacto para los restantes eventos. Esto llevó a implementar el algoritmo tomando como modelo el arco de circunferencia.

El algoritmo RANSAC cataloga los puntos obtenidos hasta ahora como *inlier/outlier*. Como pueden haber puntos en el arco de circunferencia que sean *outliers* y RANSAC no los reconoce como tal se filtra una vez más calculando la distancia media entre puntos y descartando los que superan un umbral.

El algoritmo va muestreando de a tres puntos aleatoriamente, si no están alineados calcula la circunferencia que pasa por los tres puntos.

Para evaluar el desempeño de esta circunferencia en los puntos se define una función de costo:

$$\text{loss}_i = |\text{dist}(c, p_i) - r| \tag{5.1}$$

Siendo c es el centro de la circunferencia y p_i el punto i-ésimo del conjunto. La Ecuación (5.1) mide qué tan alejado está cada punto del conjunto de datos a la circunferencia. Si la sumatoria de los costos es menor a la del modelo anterior, se reemplaza y se sigue buscando mejores modelos.

Para evaluar si es *inlier* se utiliza la desviación mediana absoluta (MAD) del vector loss:

$$\text{MAD} = \text{median}(|\text{loss}_i - \text{median}(\text{loss})|) \tag{5.2}$$

Si $\text{loss}_i > \text{MAD} \times$ residual_treshold el punto i-ésimo se considera *outlier*. Donde residual_treshold es un parámetro elegido y ajustado según la prueba.

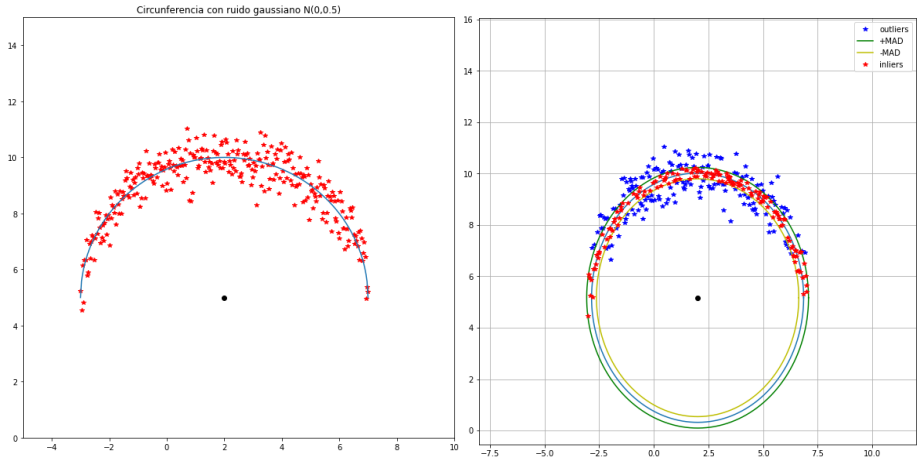
En la Figura 5.7 se ejemplificará el funcionamiento del algoritmo RANSAC. De forma teórica se tomará un arco de circunferencia de centro $(2, 5)$ y radio 5, además se contaminarán las muestras con ruido gaussiano de media cero y varianza 0.5 para luego correr RANSAC 3[1] con residual_treshold $= 1$.

En las Figuras 5.8 y 5.9 se puede apreciar la diferencia entre utilizar un modelo lineal o un arco de circunferencia para determinar el modelo de trayectoria del evento.

5.2. Dataset y preprocesamiento

El dataset final que se obtuvo puede observarse en la Tabla 5.1. Como primer paso hay que separar los datos a utilizar para la validación y una vez elegido el modelo final se procede a la evaluación en campo. Claramente se está ante un problema de clases desbalanceadas, por lo que se hará una separación estratificada manteniendo la proporción de positivos/negativos tanto en el conjunto de entrenamiento como en el de test. El tamaño del conjunto de test elegido fue del 15 %.

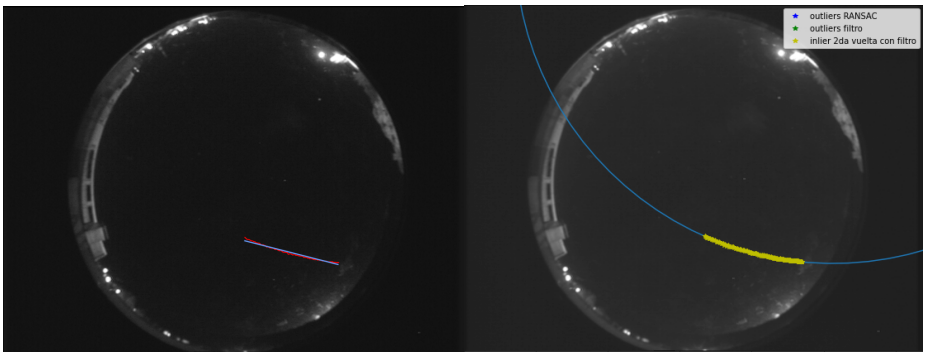
[1]RANSAC 3 es el nombre del algoritmo diseñado para este trabajo basado en RANSAC.



(a) Muestras ruidosas con las que se determinará el arco de circunferencia

(b) Modelo encontrado cuyos parámetros son $x_c = 2{,}00$, $y_c = 5{,}16$, $r = 4{,}85$

Figura 5.7: Ejemplificación del algoritmo RANSAC, donde se puede ver el modelo encontrado y la categorización inliners/outliers de los puntos.

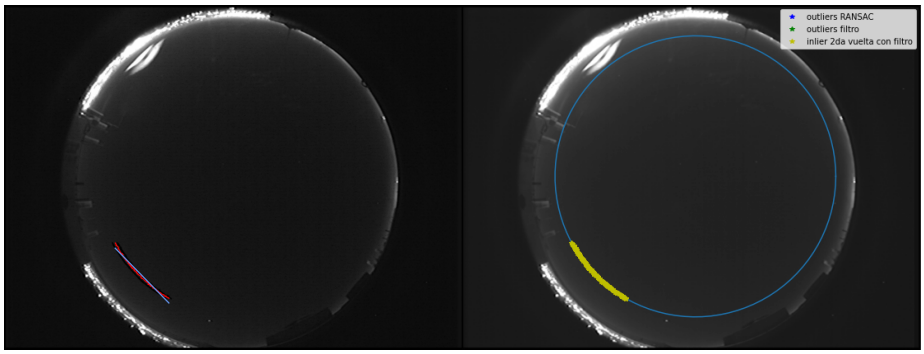


(a) Trayectoria modelada de forma lineal.

(b) Trayectoria modelada a partir de un arco de circunferencia.

Figura 5.8: Ejemplo de bólido utilizando RANSAC.

La razón por la cual la cantidad de datos disponibles disminuye luego del pre-procesamiento es debido a que muchas veces el sistema de adquisición de videos se dispara por ruidos en el horizonte. Al procesar el video con su máscara correspondiente este ya no contiene eventos, por lo que se devuelve características nulas. Por tal razón, el pre-procesamiento elimina estos eventos dejando solo un evento nulo ya que los demás no aportarían nada al modelo. También se sustituyen valores NaN que puedan aparecer en las características por valores nulos.



(a) Trayectoria modelada de forma lineal. (b) Trayectoria modelada a partir de un arco de circunferencia.

Figura 5.9: Ejemplo de un avión utilizando RANSAC.

	Bólidos	No Bólidos
Antes del preprocesamiento	181	3321
Después del preprocesamiento	181	2404

Tabla 5.1: Datos disponibles luego de la clasificación

5.3. Análisis de características

Casi el 85 % del dataset está constituido por aviones, animales, bólidos y lluvia/nubes, el restante son eventos varios, desde personas pasando con linternas hasta fuegos artificiales, es por eso que si se logra encontrar un conjunto de características que pueda distinguir la mayoría de estos eventos de los bólidos gran parte del problema estaría resuelto. En la Tabla 5.2 se describe cada una de las características disponibles que se obtuvieron.

Utilizando la correlación de Pearson entre las características se puede encontrar varias que están fuertemente correlacionadas, un ejemplo claro es el de FWHM y la intensidad media; tiene sentido la correlación alta ya que el FWHM es directamente proporcional con el brillo del evento. Aún así, se decidió no descartar ninguna característica ya que el modelo que se utilizará tiene la capacidad de detectar qué características son más importantes y no utilizar las redundantes o que no sirven.

A continuación se resumirá las mayores diferencias que se lograron notar de la observación de los datos y se intentó demostrar que al menos una característica logra diferenciar un bólido de los demás eventos.

Los aviones tienen menor velocidad, mayor cantidad de puntos (N) y la luminosidad (intensidad) del evento suele ser más constante. En la Figura 5.10 se pueden observar características graficadas de a pares. Es claro que hay clusters que permiten diferenciar aviones de bólidos en su gran mayoría, como es la gráfica de velocidad media vs cantidad de puntos. Sin embargo, no todas las características son tan útiles, como por

Feature	Descripción
Video	Nombre que identifica el video que se procesó
N	Cantidad de puntos del evento
dispersion	Medida de dispersión de los puntos. Ver B.4.2
max_vel	Velocidad instantánea máxima del evento. Ver B.4.6
min_vel	Velocidad instantánea mínima del evento. Ver Ver B.4.6
velocidad_media	Velocidad instantánea media del evento. Ver B.4.6
var_vel	Varianza de la velocidad instantánea. Ver B.4.6
mean_intensidad	Intensidad media del evento. Ver B.4.4
max_int	Intensidad máxima del evento. Ver B.4.4
min_int	Intensidad mínima del evento. Ver B.4.4
var_int	Varianza de la intensidad. Ver B.4.4
mean_fwhm	FWHM media del evento. Ver B.4.3
var_fwhm	Varianza del FWHM. Ver B.4.3
sensibilidad_i_1	Sensibilidad de la intensidad normalizada del evento. Ver B.4.5
sensibilidad_i_2	Sensibilidad de la intensidad sin normalizar. Ver B.4.5
sensibilidad_v_1	Sensibilidad de la velocidad normalizada del evento. Ver B.4.5
sensibilidad_v_2	Sensibilidad de la velocidad sin normalizar. Ver B.4.5
sensibilidad_fwhm_1	Sensibilidad del FWHM normalizada del evento. Ver B.4.5
sensibilidad_fwhm_2	Sensibilidad del FWHM sin normalizar. Ver B.4.5
recorrido	Desplazamiento del evento. Ver B.4.1
loss	Indica qué tan bien se ajusta el evento al modelo de un arco. Ver B.3.4

Tabla 5.2: Descripción resumida de cada característica

ejemplo recorrido vs velocidad máxima.

Analizando la Figura 5.11 se tiene a los animales (insectos, pájaros, murciélagos y otros) vs Bólidos, por lo general tienen mayor cantidad de puntos (N) y su trayectoria no se ajusta bien a un arco como sí lo hace el bólido por lo que se observa mayor loss.

Por último y no menos importante está la lluvia y nubes que generan ruido muy diverso y son parte importante de los videos generados en un mes. Estos eventos generan puntos dispersos en el lente y por lo tanto loss y dispersion alta pero hay casos donde los puntos son estáticos, donde la velocidad_media, recorrido y dispersion son nulos. Como se ve en la Figura 5.12 algunas características como el recorrido y la velocidad_minima permiten diferenciar la mayoría de los bólidos. Por lo tanto se ha visto que si bien hay una gran variedad de eventos posibles se logra identificar una característica entre cada grupo permitiendo diferenciar. Esto si bien es útil para analizar cada tipo de evento, a la hora de entrenar el modelo; éste se va a enfrentar a un problema de clasificación binaria donde estarán todos los aviones, animales y lluvia/nubes clasificados como no bólidos. Viendo la Figura 5.13 se observa que ya no es tan sencillo definir umbrales que logren separar los eventos de interés. Afortunadamente existen algoritmos que pueden manejar diversas características y lidiar con este tipo de problemas en los que se ahondará en las siguientes secciones.

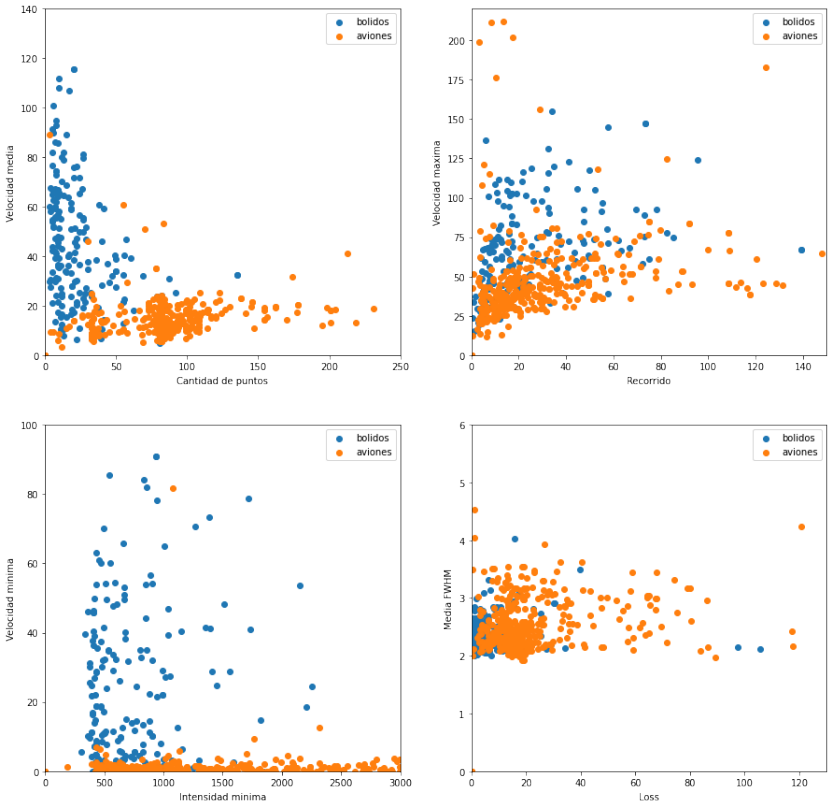


Figura 5.10: Características de bólidos vs aviones.

5.4. Modelo

Del análisis de características se desprende que no es tan sencillo poder definir umbrales que permitan clasificar de forma clara bólido/no bólido. Es aquí donde entran los modelos de aprendizaje automático.

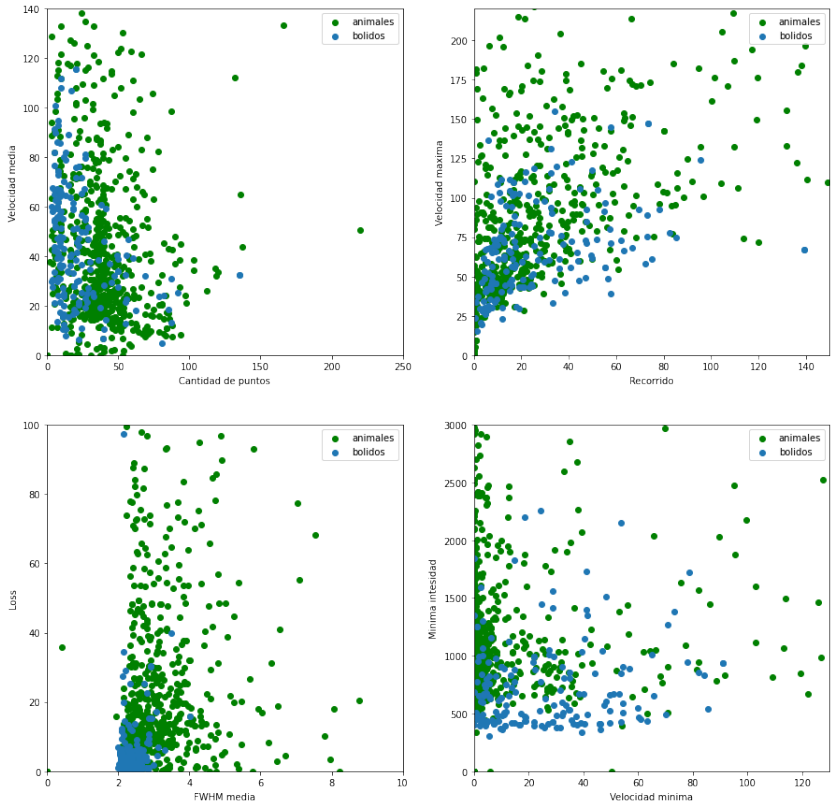


Figura 5.11: Características de bólidos vs animales.

En primer lugar se pensó en Árboles de Decisiones y Random Forest debido a la simplicidad y a la eficiencia en este tipo de problemas. Por otro lado XGBoost [Tianqi Chen, Carlos Guestrin, 2016] como es descrito en su paper es una biblioteca de Gradient Boosting optimizada y diseñada para ser altamente eficiente, flexible y portátil.

Este algoritmo introduce ciertas ventajas frente a Gradient Boosting, en primer lugar permite que el entrenamiento se pueda paralelizar haciéndolo más eficiente. Además permite generalizar mejor ya que introduce parámetros de regularización (L1 & L2).

Entrenando sin búsqueda de hiper-parámetros se obtuvieron los resultados de la Tabla 5.3, viendo que XGBoost supera a los demás modelos más sencillos e inclinando la balanza a favor de la elección de este modelo.

Para lidiar con el problema de desbalance de clases, XGBoost introduce un parámetro que denomina `scale_pos_weight` y recomienda el valor de la Ecuación (5.3).

$$\text{scale\_pos\_weight} = \frac{\sum(\text{instancias\_negativas})}{\sum(\text{instancias\_positivas})} \tag{5.3}$$

Introducir este valor permite que una instancia de la clase minoritaria tenga mayor influencia para ser clasificada correctamente.

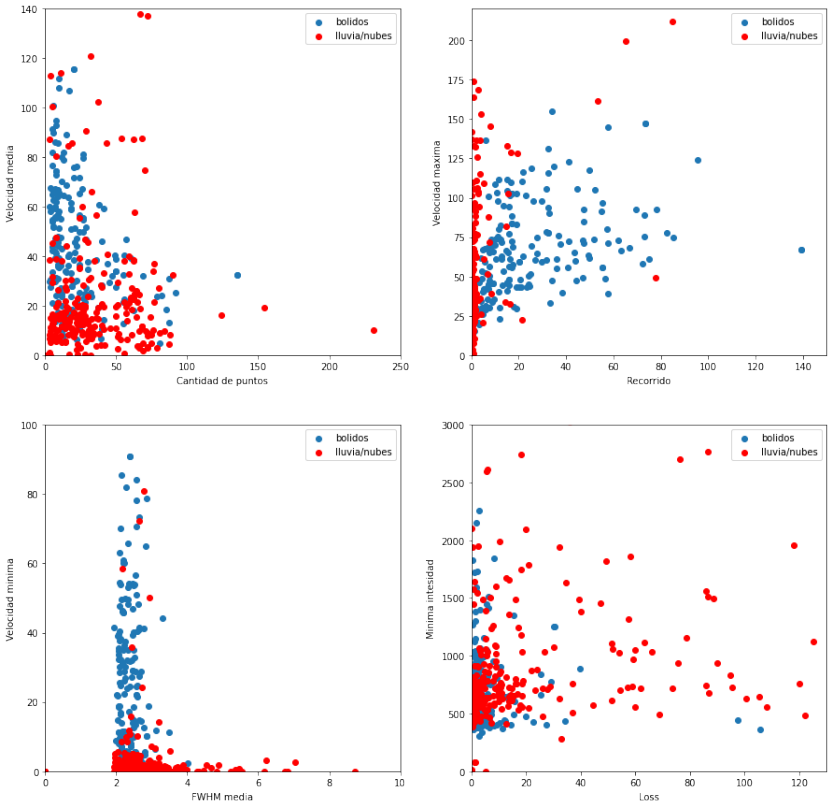


Figura 5.12: Características de bólidos vs lluvia/nubes.

Modelo	Accuracy	Precision	Recall
Decision Tree	95.0 %	64.4 %	66.5 %
Random Forest	96.7 %	86.9 %	62.3 %
XGBoost	96.7 %	81.8 %	83.9 %

Tabla 5.3: Comparación entre modelos

La cantidad de árboles a entrenar secuencialmente se controla con el parámetro `num_boost_round`; para asegurar de obtener el número adecuado de arboles se fijó el parámetro `num_boost_round` a un valor alto y se agregó `early_stopping=20`, es decir, si luego de veinte rondas el algoritmo no obtiene mayores avances en términos de la función de pérdida el algoritmo dejará de iterar.

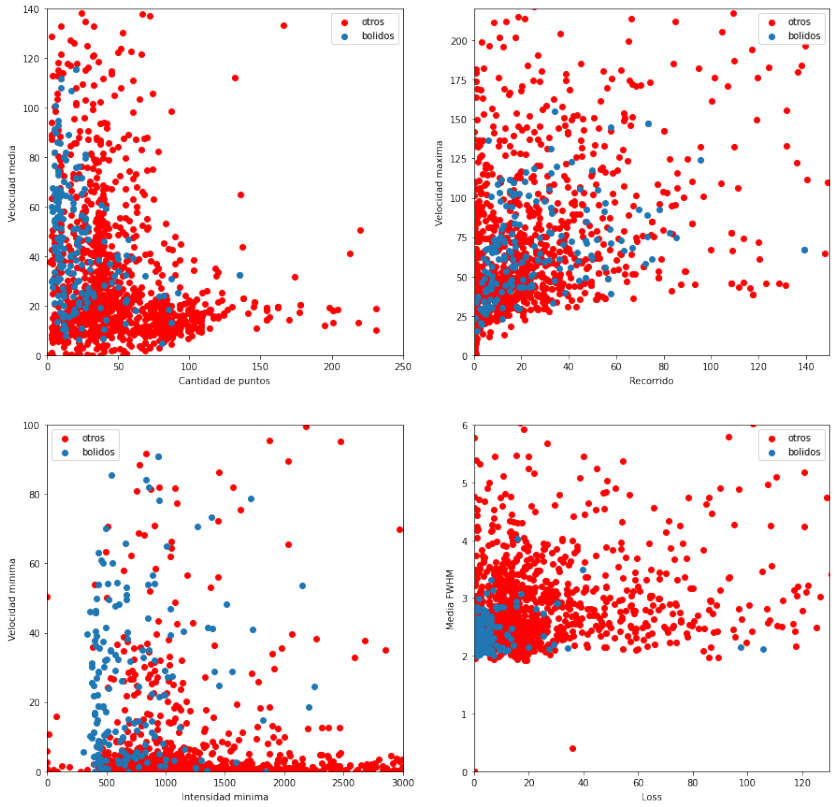


Figura 5.13: Características de bólidos vs lluvia/nubes aviones y animales.

La función de pérdida utilizada fue MAE:

$$MAE = \sum_{i=0}^{n} \frac{|y_i - x_i|}{n} \tag{5.4}$$

Donde y_i es la predicción perteneciente al intervalo $[0, 1]$, es decir, el modelo devolverá la probabilidad de pertenencia a la clase positiva y x_i los valores verdaderos que pueden adoptar solo dos valores posibles [0,1], correspondientes a Bólido (1) y No Bólido (0).

5.4.1. Búsqueda de hiperparámetros

La grilla en donde se buscó hiperparámetros está descrita en la Tabla 5.4. Esta búsqueda requiere entrenar y evaluar 648 modelos utilizando validación cruzada con 5 folds. Una búsqueda más exhaustiva si bien podría implicar mejores resultados hay que tener en cuenta el tiempo de entrenamiento que actualmente se lleva a cabo en 12-15 minutos[2]. En el futuro con un dataset mayor debido a la instalación de nuevas estaciones y clasificación de nuevos videos el tiempo podría escalar rápidamente.

[2]En una PC con CPU Intel i5-11400.

max_depth	6	7	8	9	10	11
min_child_weight	1	5	10			
colsample_bytree	0.6	0.7	0.8			
subsample	0.6	0.7	0.8			
eta	0.1	0.2	0.3	0.4		

Tabla 5.4: Búsqueda de hiperparámetros.

Cada hiperparámetro está descrito según la documentación de XGBoost [XGBoost developers, 2021] como se describirá en las subsecciones siguientes.

5.4.1.1. max_depth y min_child_weight

Los parámetros max_depth y min_child_weight agregan limitaciones a la arquitectura de los árboles, max_depth es el número máximo de nodos permitidos desde la raíz hasta la hoja más alejada, a mayor profundidad los splits son menos relevantes, complejizando el modelo y muchas veces sobre ajustando. Por otro lado, min_child_weight es el número de muestras mínimas requeridas para crear otro split en el árbol. Disminuyendo este parámetro el árbol se complejiza ya que permite crear nuevos nodos con menos muestras.

Ajustando estos parámetros se permite encontrar un buen trade-off entre el sesgo-varianza del modelo.

5.4.1.2. subsample y colsample

Los parámetros subsample y colsample controlan la fracción del dataset que se usa para entrenar en cada round. En vez de utilizar todo el conjunto de entrenamiento para cada round, se puede utilizar una fracción diferente tanto del dataset como de features para entrenar cada árbol, lo que hace que el modelo se sobreajuste menos al conjunto de entrenamiento. El parámetro subsample corresponde al muestreo en eventos del dataset, mientras que colsample corresponde al muestreo de características.

5.4.1.3. eta

El parámetro eta controla la corrección de los pesos luego de cada round, en la práctica, tener un eta bajo implica más iteraciones para llegar a la convergencia del modelo, mientras que un eta más alto si bien requiere menos rounds para converger pueden surgir problemas de oscilaciones entorno a los óptimos no llegando a converger nunca.

5.4.2. Elección del umbral

La parte interesada puso énfasis en no perder videos de bólidos pero sin contener exceso de falsos positivos, en otros términos, se desea maximizar el recall sin descuidar

el precision. Para lograrlo se observó el $f_\beta - $ score definido en la Ecuación (5.5).

$$f_\beta = (1 + \beta^2) \frac{\text{precision} \times \text{recall}}{(\beta^2 \times \text{precision}) + \text{recall}} \tag{5.5}$$

Donde β es elegido de forma tal que el `recall` es β veces más importante que el `precision`. Como el modelo creado devuelve la probabilidad de pertenencia a la clase positiva, se utilizó el f_2-`score` para la elección del umbral adecuado de decisión y de esta forma cumplir el objetivo.

Capítulo 6

Resultados

En esta sección se expondrán los resultados obtenidos del modelo de clasificación junto a una evaluación de campo para determinar el desempeño del programa implementando.

6.1. Evaluación del modelo

Para determinar el mejor modelo de clasificación se realizó la búsqueda de hiperparámetros pertenecientes a XGBoost. Se puede ver en la Tabla 6.1 los valores que lo optimizan. Una vez entrenado el modelo, se procede a graficar el $f2-score$, precision y recall en función del umbral de decisión como se ve en la Figura 6.1, dando como resultado que el umbral que maximiza el $f2-score$ es igual a 0.116 con los resultados de la Tabla 6.2.

La librería XGBoost proporciona las características que fueron más relevantes para el modelo y se pueden ver en la Figura 6.2. El eje horizontal indica la cantidad de veces que cierta característica aparece en los árboles del modelo. Por lo visto en el capítulo anterior no existe una característica única que logre separar todos los casos. La característica loss que es un indicador de qué tan bien se ajusta la trayectoria a un arco de circunferencia y la intensidad mínima parecieron ser las más relevantes a la hora de clasificar el dataset mientras que la varianza del FWHM y la sensibilidad de la velocidad las que menos. Por último se muestra la matriz de confusión obtenida

Parámetro	Valor
max depth	10
min child weight	1
colsample	0.8
subsample	0.8
eta	0.3

Tabla 6.1: Resultados de la búsqueda de hiperparámetros del modelo XGBoost.

Métrica	Score obtenido
Accuracy	98.2 %
Recall	92.6 %
Precision	83.3 %
f2-score	90.5 %
FP	7.4 %
FN	1.4 %

Tabla 6.2: Resultados finales de la clasificación de 388 videos, de los cuales 27 son bólidos y 361 no bolidos.

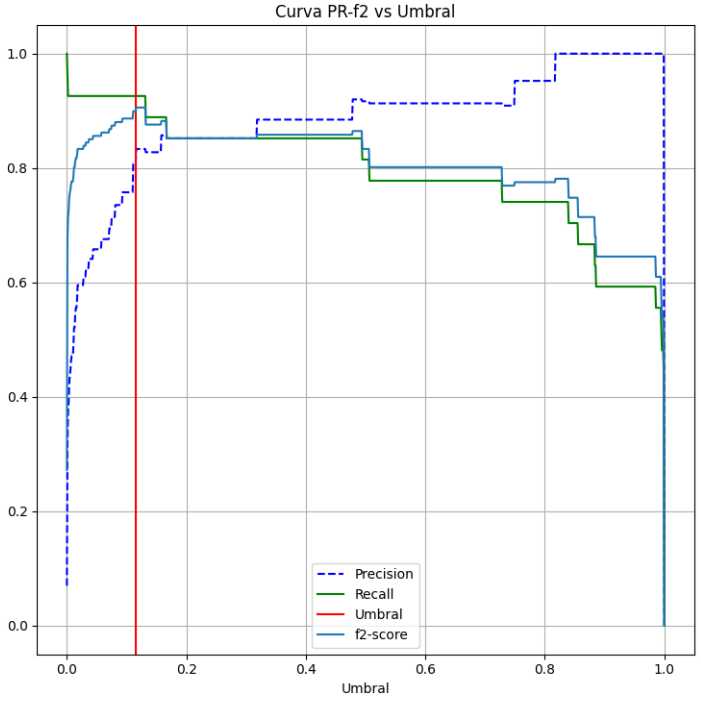


Figura 6.1: Curvas PR-f2. Curva de Precision, Recall y f2 − score

en la Figura 6.3.

Estos resultados son una estimación de como se comportaría el modelo en producción. Se observa que uno de los resultados fue levemente mayor al objetivo planteado ya que se obtuvo una tasa de falsos negativos mayor a 5 %, donde se ubica en 7.4 % pero se cumplió con satisfacción la tasa de falsos positivos, donde se ubica en 1.3 %.

La causa principal de la alta tasa de falsos negativos es consecuencia del gran

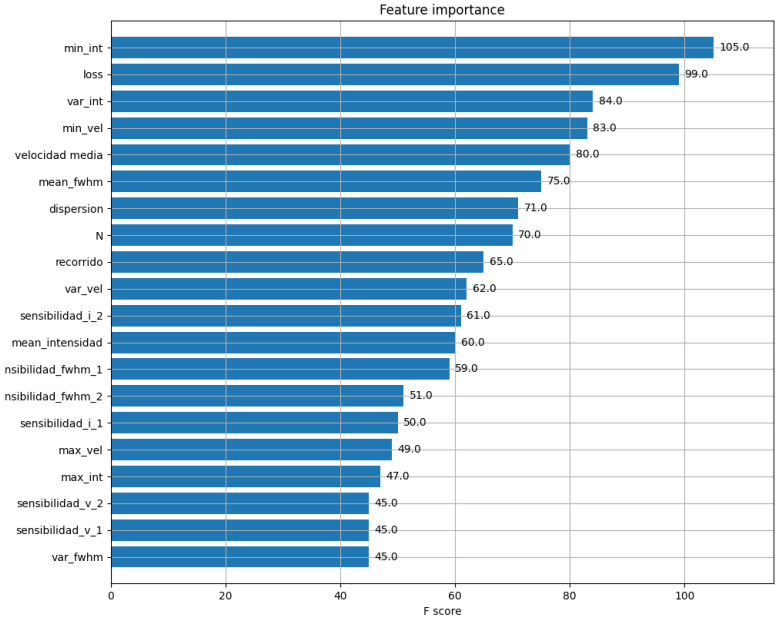


Figura 6.2: Importancia de las características de la librería XGBoost.

desbalance del dataset, el tener pocos bólidos genera poco margen para cumplir el objetivo generando que la tasa de falsos negativos sea muy sensible con respecto a la cantidad de bólidos. Por ejemplo, si se toma las cantidades obtenidas a continuación para la evaluación de campo; si tenemos 17 bólidos y se detecta 1 erróneamente se obtiene una tasa de 5,9 %, en cambio sin detección errónea la tasa baja drásticamente a 0 %, en conclusión para cumplir el objetivo se acepta hasta 0.85 bólidos lo que corresponde a no detectar erróneamente, aumentando la exigencia considerablemente en un dataset desbalanceado.

Una manera de forzar a que se cumpla el objetivo pero aumentando la tasa de falsos positivos sería aumentar el β en el $f_\beta - score$, dando aún más importancia al recall y bajando el umbral. Sin embargo se cree que el problema está por el lado de la cantidad de datos, un mayor número sobre todo de instancias positivas lograría el cometido de reducir el valor de la tasa de FN.

6.2. Evaluación en campo

Una parte importante es ver cómo se desempeña el modelo en datos no utilizados para entrenar y testear el modelo. De la clasificación de los últimos meses se obtuvo 17 bólidos y 210 no bólidos, resultando en 1 falso negativo y 2 falsos positivos como

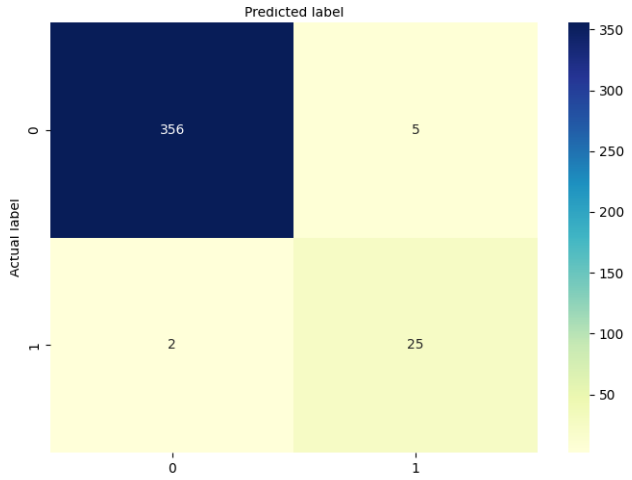


Figura 6.3: Matriz de confusión obtenida para el modelo. Donde 0 es No Bólido y 1 es Bólido.

Métrica	Score obtenido
Accuracy	98.7 %
Recall	94 %
Precision	88 %
f2-score	90.5 %
FP	5.9 %
FN	1.0 %

Tabla 6.3: Resultados finales de la prueba de campo de un total de 227 videos, de los cuales 17 son bólidos y 210 no bólidos.

se puede ver en la matriz de confusión de la Figura 6.6. Los valores de la Tabla 6.3 son del orden de los obtenidos en el conjunto de validación del modelo descartando un posible sobre ajuste del modelo al dataset de entrenamiento.

Se analizará el caso de falso negativo a través del análisis de características, el rastro o imagen máxima se puede ver en la Figura 6.4 [1].

Como se observa en la Figura 6.5 el evento que resultó en falso negativo muestra una velocidad máxima poco usual en comparación con los bólidos que se usaron como datos de entrenamiento. Este desvío presente en la velocidad máxima en conjunto con alguna otra característica no presente en el análisis pueden ser las claves para el algoritmo clasificarlo como No Bólido ya que se observa que en el resto de las características analizadas éste no se aleja demasiado del conjunto de entrenamiento.

[1] Ejemplo de lo que devuelve la función predict -localize del programa Autobol

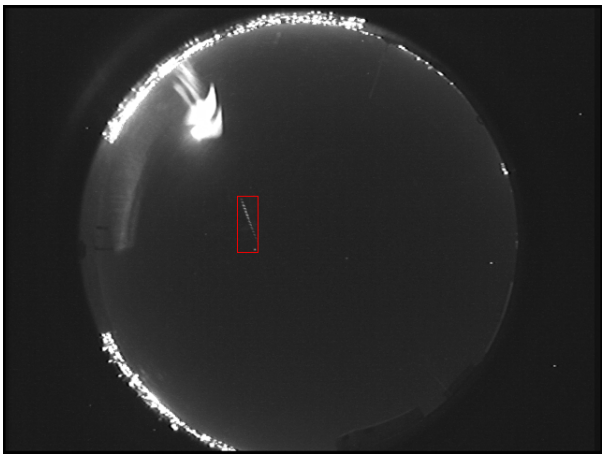


Figura 6.4: Trayectoria del bólido clasificado como FN.

6.3. Evaluación de tiempos

Para evaluar la eficiencia del algoritmo al predecir se mide el tiempo promedio que demoró en clasificar los 227 vídeos obtenidos en la evaluación en campo. En la Figura 6.7 se muestran los histogramas de la duraciones de los videos y el tiempo que lleva el procesamiento, desde la extracción de características hasta la predicción[2]. La duración promedio de los videos fue de 1.44 segundos mientras que el tiempo de procesamiento promedio fue de 1.46 segundos. Por lo tanto se puede concluir el tiempo de procesamiento es similar a la duración del video, satisfaciendo el objetivo de que sea menor a 10 veces la duración del mismo.

[2]Ejecutado en i5-11400 2.6 Ghz

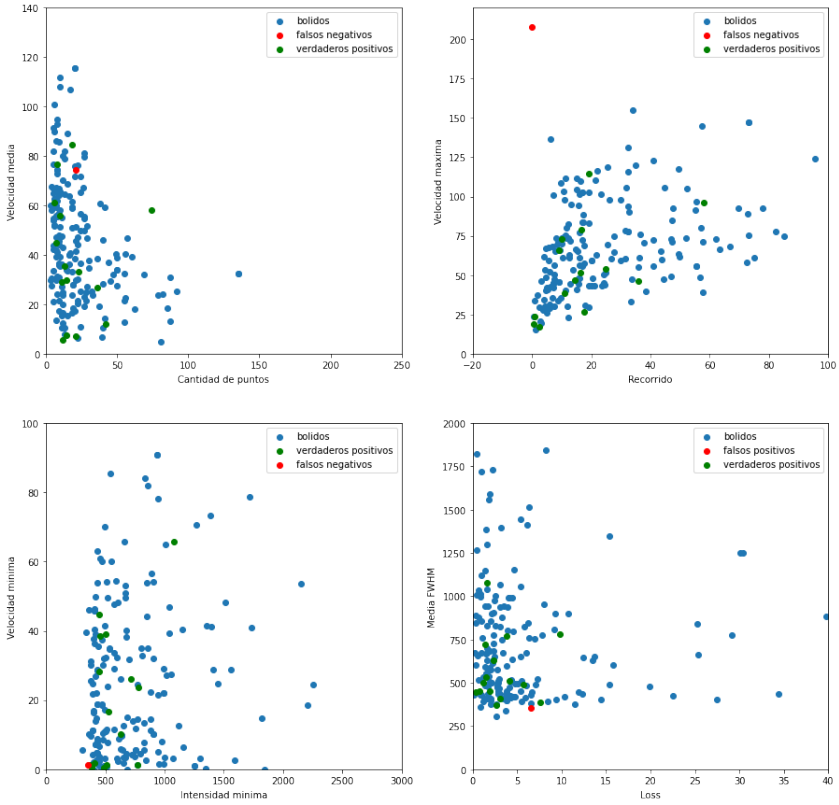


Figura 6.5: Resultado de evaluación de campo. En AZUL los bólidos del conjunto entrenamiento, en ROJO el bólido clasificado erróneamente y en VERDE de forma correcta.

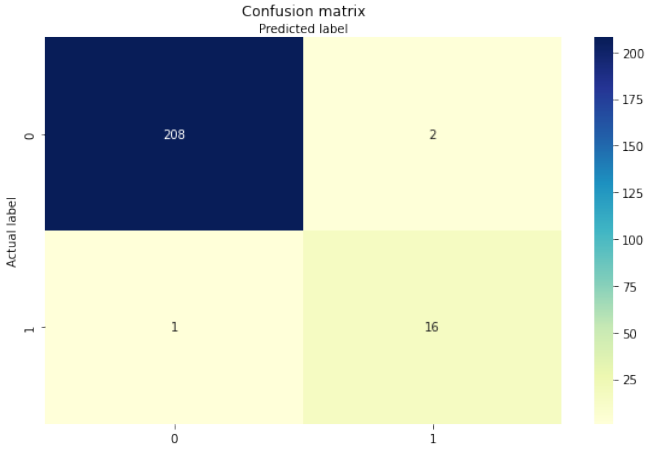


Figura 6.6: Matriz de confusión para la evaluación en campo. Donde 0 es No Bólido y 1 es Bólido.

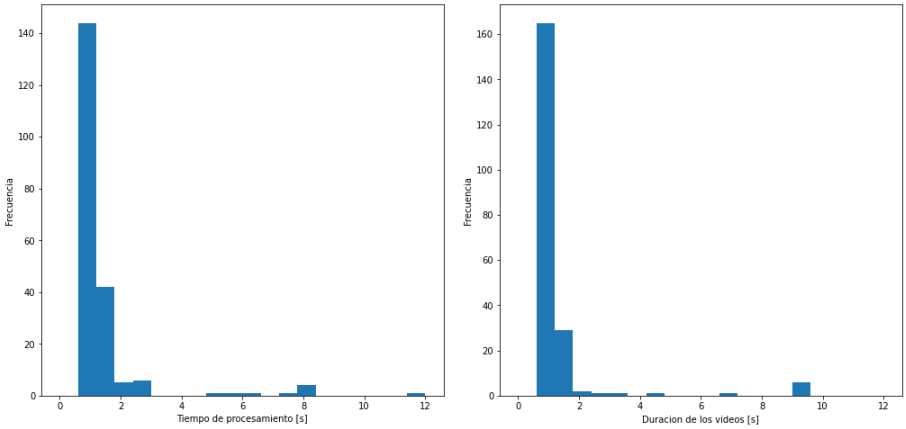


Figura 6.7: Histogramas de duración video y tiempo de procesamiento.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 7

Conclusiones

Se logró abordar el problema de detección de bólidos mediante grabaciones de cámaras All Sky de manera satisfactoria. Desde la extracción y análisis de características hasta la clasificación de eventos. Se entrega a Bocosur un programa capaz de clasificar, predecir, localizar eventos y re-entrenar el modelo de manera eficiente y rápida junto con un manual de usuario vasto para hacer uso del mismo sin necesidad de haber participado en el desarrollo. Cabe destacar que la funcionalidad principal del programa era clasificar y predecir pero al surgir necesidades funcionales apartadas del plan de proyecto inicial, se decidió implementar nuevos módulos que fuesen de ayuda para solventar futuras adversidades, aportándole integridad al producto final.

El continuo intercambio de diálogo con la parte interesada fue de gran valor en el desarrollo del producto ya que permitió un constante desarrollo y una ágil evacuación de dudas enriqueciendo y perfeccionando el producto.

Tener un leve alejamiento del valor establecido como objetivo de la tasa de falsos negativos por debajo del 5 % no se interpreta en un logro fallido, demuestra que se fue demasiado exigentes a la hora de definir los objetivos previo a conocer el problema y los desafíos que conlleva enfrentarse a un dataset con un gran desbalance de clases. Sin embargo se estima que en el futuro con más datos y con la capacidad de re-entrenar el modelo este requerimiento se cumpla con facilidad.

Destacamos el cumplimiento del resto de los objetivos, la tasa de falsos positivos fue ampliamente alcanzada estableciéndose en un porcentaje que se encuentra alrededor del $1{,}0\,\% - 1{,}5\,\%$ la cual no es superior al 10 % propuesto al inicio del proyecto. Así como también el tiempo de ejecución del modo predicción del programa desarrollado no supera 10 veces la duración del video, teniendo como valor promedio 1.46 segundos.

La realización del proyecto aporto al grupo un gran avance en el plano personal como también profesional al tratarse de una experiencia de mayor integración de los conocimientos adquiridos en la formación académica tanto técnicos, como comerciales y humanos dando un primer cierre a esta etapa educativa para adentrarnos al ámbito profesional.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 8

Trabajo a futuro

Como continuación de este trabajo de tesis y como en cualquier otro proyecto de investigación, existen diversas líneas de investigación que quedan abiertas y en las que es posible continuar trabajando. Las siguientes son áreas en las que se podría darle seguimiento al presente trabajo y en que el grupo Bocosur podría interesarse.

- Incorporación de nuevas cámaras: La incorporación de nuevas estaciones en otros puntos del país se encuentra en ejecución actualmente por parte de los integrantes de Bocosur. La integración de éstas es algo que el grupo contempló y que bastaría con agregar la nueva máscara a las existentes. Para clasificar el video se debe respetar el formato acordado. Si bien el modelo actual debería clasificar los videos generados sería recomendable relevar una cantidad significativa de datos -que incluyan las estaciones nuevas- y determinar un nuevo modelo de clasificación, que se puede realizar con las herramientas que ofrece el software actualmente.
- Identificación de similares: Ya determinados los bólidos en cada estación, sería importante la identificación del mismo en otros sitios cuyo campo de visión lo permita. Esto daría acceso a realizar validación cruzada entre estaciones dando certezas sobre la clasificación, y permitiendo despejar dudas en casos con incertidumbre. Además permitirían una mejor estimación de características propias del objeto como su tamaño o luminosidad.
- Estimación de trayectoria: Un objetivo algo ambicioso es determinar la trayectoria del bólido con los datos procedentes de varias estaciones mediante triangulación. Conocer la trayectoria de este podría permitir -en caso que corresponda- estimar o reducir el lugar de impacto facilitando las tareas de rastreo y recolección del mismo. Además se podría estudiar su trayectoria pre-atmósferica permitiendo conocer la región de procedencia y realizar una base de datos con esta información.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice A

Manual de Usuario

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

A.1. Derechos de autor

Este documento y su contenido son propiedad de Facultad de Ingeniería de la Universidad de la República, y están protegidos por el Derecho de Autor garantizado en la Constitución de la República Oriental del Uruguay y por leyes especiales que reconocen al autor un derecho de dominio sobre las producciones de su pensamiento.

Los usuarios estarán autorizados, por los medios puestos a su disposición, a visualizar, imprimir y descargar el material únicamente para uso personal y sin fines comerciales. Sin embargo, no les permite borrar o corregir el documento sin previo consentimiento escrito por parte de los miembros involucrados del proyecto AutoBol y BOCOSUR.

Se reservan el resto de los derechos

Cualquier pregunta con relación al uso particular autorizado y todas las solicitudes de permiso para publicar, modificar, distribuir, etc., del documento deberán dirigirla a los siguientes correos: juanpballestrino@gmail.com, cdeandraya@gmail.com , cuviedo1@gmail.com y redbolidos@gmail.com.

A.2. Prefacio

Este programa está implementado en Python 3, software de uso libre y gratuito en el marco del proyecto de fin de carrera de Ingeniería Eléctrica y como parte de un

proyecto madre denominado BOCOSUR. El programa AUTOBOL tiene 5 funciones implementadas que permiten predecir sobre videos, extraer características para clasificación manual, crear máscara para estaciones nuevas o modificarlas y por último volver a entrenar el modelo de clasificación.

El programa AUTOBOL cuenta con 4 archivos principales y necesarios para su ejecución:

- `autobol.py`: Es el módulo principal con el que se accede a las distintas funciones, en este archivo se definen los requerimientos para ejecutar cada modo implementado.
- `algoritmos.py`: Donde se encuentran todos los algoritmos relacionados a la extracción de características, crear máscara y clasificación.
- `abyo.py`: Módulo que tiene como objetivo separar el input/output de los demás archivos y tenerlos en un mismo lugar.
- `functions.py`: En este archivo se definen los cuatro modos de funcionamiento.

A.3. Propósito

Este documento tiene como objetivo facilitar la tarea de conocimiento, uso y aprendizaje del sistema desarrollado para detección y clasificación de bólidos a partir de cámaras all-sky. Contiene información acerca de todas las operaciones básicas que el programa ofrece, así como imágenes útiles para el seguimiento de la explicación y su uso.

A.4. Audiencia

Este documento está dirigido a personal técnico y operativo, encargados de ejecutar el software desarrollado.

Si bien su operativa es simple se recomienda conocimientos básicos de líneas de comando en consola. Aunque no es un requerimiento excluyente porque se detallaran sus funcionalidades y opciones a continuación.

A.5. Alcance

Se pretende detallar las funciones de forma que cualquier persona encargada del proyecto así como colaboradores puedan manipular y usar las funcionalidades del programa sin conocer los detalles del funcionamiento. No es parte del alcance, explicar como se implementó, estructuró, desarrolló, etc, el software propiamente dicho. Por más información sobre la implementación se puede visitar el github desarrollado (*Link al Enlace en GitHub*).

A.6. Guía de uso

A.6.1. Consola

Lo primero que se debe hacer es abrir una ventana de línea de comandos.

- En Windows:

  Para la versión de Windows en idioma Español se deberá seguir los siguientes pasos:

  - Pulsa a la vez la tecla de Windows (normalmente tiene el logotipo de Microsoft, está ubicada cerca de la tecla `Alt`) y la letra `R`.
  - Se abrirá una pequeña ventana con el título de "Ejecutar". Dentro del rectángulo para introducir texto que se desplegará, escribir `cmd` y pulsar sobre el botón de "Aceptar".
  - La ventana de comandos `cmd.exe` estará iniciada.

  Otra forma es utilizar el buscador del escritorio con las palabras `cmd` o `símbolo de sistema`.

- En Linux:

  Hay varios métodos para acceder a una ventana de línea de comandos: Uno de ellos es pulsa a la vez la siguiente combinación de teclas `Ctrl + Alt + T`.

  Otra de ellas es buscar usando el tablero Ubuntu.

A.6.2. Directorio de trabajo

Lo siguiente a realizar es ir al directorio de trabajo. En la línea de comandos ubicarse en la carpeta donde se tenga el archivo `autobol.py` y sus archivos relacionados.

- Uno de los comandos más esenciales de la consola tanto en Windows como en Linux es: `cd`. Función para cambiar de directorio.

  Sintaxis: `cd <RutaDirectorio>` (donde `RutaDirectorio` es la ubicación a la cual se quiere mover) o `cd..` para salir de una carpeta e ir al nivel superior o carpeta donde estaba alojada.

- Comando para obtener el contenido del directorio o carpeta donde uno se encuentra, mostrando todas las subcarpetas o archivos que posee. Con este comando se podrá saber si el archivo que se busca se encuentra ahí o a qué subcarpeta navegar.

  - En Windows: `dir`
  - En Linux: `ls`

A.6.3. Módulos de trabajo

En el directorio de trabajo se debe contar con las siguientes carpetas o archivos:

- Carpeta `Mascaras`: Donde se guardaran las mascaras ya guardadas o a crear/modificar.
- Carpeta `<Directorio_Salida>`: Directorio donde se quiere guardar los archivos de salida.
- Archivo `abio.py`.
- Archivo `algoritmos.py`.
- Archivo `functions.py`.
- Un modelo es solo necesario si se desea predecir, en este caso es `model.bin`.

Seguido de ejecutar en consola el archivo `autobol` con `python` y teniendo en cuenta los parámetros de cada módulo de trabajo, que se describirán a continuación:

A.6.3.1. Train

El módulo `train` es utilizado para reentrenar el modelo `XGBoost`, para mejorar la eficiencia del algoritmo o para entrenar sobre otra base de datos.

Los requerimientos son:

- `-m train`: Se entra a la función `train`.
- `-i <Directorio_Videos>`: Ruta donde se encuentra el archivo .csv con los datos de entrenamiento.

  Observación 1: El archivo `.csv` corresponde al archivo `Clasificacion.csv`, salida del modo `predict verbose`, luego de que haya sido editado por el usuario con las etiquetas 0 o 1.
- `-o <Directorio_Salida>`: Directorio donde se quiere guardar el archivo de salida. Tendrá el nombre por defecto `model_umbral.bin`.

  Siendo `_umbral` el umbral óptimo encontrado donde maximiza el `f2 score` del modelo.

  Observación 2: Este valor es el recomendado a utilizar en el modo `predict` en el parámetro `-u <umbral>`, aunque se puede utilizar cualquiera.
- `-t <%_Test>`: Indica el porcentaje de los datos que se utilizan para `test`, permite evaluar el modelo en datos que no se usaron para el entrenamiento. Donde `<%_Test>` es el índice porcentual, $\%\_Test \in [0, 1]$.

Un ejemplo de ejecución del módulo `train`, se puede ver en la Figura A.1. Donde la línea de comandos ingresada fue:
`python autobol.py -m train -i Clasificacion.csv -o salida-prueba -t 0.15`
Al presionar enter, el software devuelve un mensaje con la cantidad de bólidos y no bólidos en los conjuntos de `entrenamiento` y `test`. Preguntará si se desea continuar, esperando una respuesta válida `(Y, y, N o n)`. En caso afirmativo se ingresa `Y` o `y`, y se empezará a ejecutar el entrenamiento. De lo contrario se ingresa `N` o `n` y

Figura A.1: Ejemplo de ejecución del módulo test.

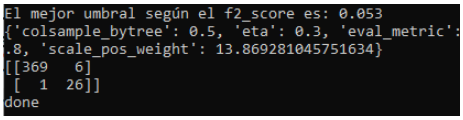


Figura A.2: Salida del ejemplo de ejecución del módulo train con parámetros de entrada vistos.

saldrá del programa.

Observación 3: El módulo de entrenamiento puede llegar a demorar unos minutos. Para el ejemplo mencionado que cuenta con 2122 videos etiquetados como "No Bólido" y unos 153 videos como "Bólido" demoró unos 19 minutos con 30 segundos.

La salida en consola se puede ver en la Figura A.2. Donde se devolverá el valor del mejor umbral que optimice el modelo, junto con la matriz de confusión del modelo aplicado al conjunto de `test`. Estos y otros datos útiles quedarán guardados en un archivo `.txt` en la misma ruta.

La matriz de confusión está constituida como:

$$\begin{bmatrix} \text{verdaderos negativos} & \text{falsos positivos} \\ \text{falsos negativos} & \text{verdaderos positivos} \end{bmatrix}$$

A.6.3.2. Predict

El modulo `predict`, tiene como objetivo clasificar los videos que se encuentran en la ruta especificada entre las categorías "Bólido" y "No Bólido".
Los requerimientos son:

- `-m predict`: Se entra a la función `predict`.
- `-i <Directorio_Videos>`: Directorio donde se encuentra/n el/los video/s que se desea/n predecir.
- `-o <Directorio_Salida>`: Directorio donde se quiere guardar el archivo de salida, en este caso un `.csv` con el formato:

  Encabezado: Nombre_Archivo, Probabilidad_De_Ser_Bólido, Etiqueta
  Fila 1: Video1, Probabilidad1, Etiqueta1
  Fila 2: Video2, Probabilidad2, Etiqueta2
  Fila 3: Video3, Probabilidad3, Etiqueta3

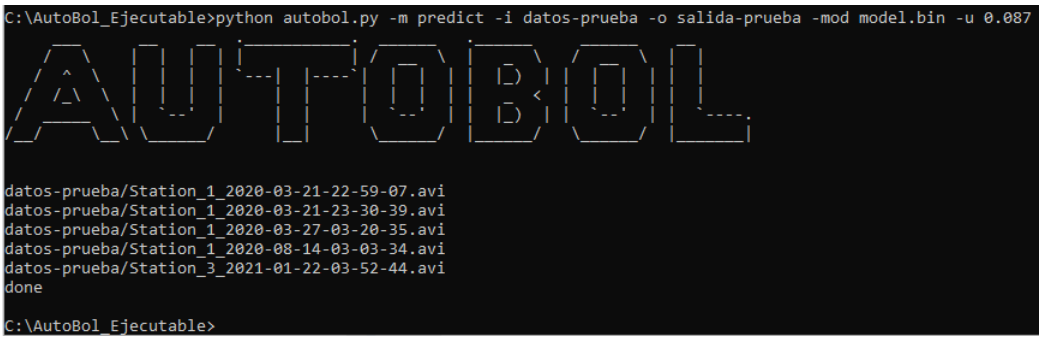


Figura A.3: Ejemplo de ejecución del módulo predict.

	A	B	C
1	Nombre Archivo	Probabilidad de Bólido	Etiqueta
2	datos-prueba/Station_1_2020-03-21-22-59-07.avi	99,96	Bólido
3	datos-prueba/Station_1_2020-03-21-23-30-39.avi	98,78	Bólido
4	datos-prueba/Station_1_2020-03-27-03-20-35.avi	99,98	Bólido
5	datos-prueba/Station_1_2020-08-14-03-03-34.avi	99,99	Bólido
6	datos-prueba/Station_3_2021-01-22-03-52-44.avi	0	No Bólido

Figura A.4: Salida del ejemplo de ejecución del módulo predict con parámetros de entrada vistos.

Nombre: Referido a la raíz común de los archivos.
Probabilidad: Valor p absoluto porcentual, $p \in [0, 100]$ %.

Etiqueta:

$$Etiqueta = \begin{cases} \text{Bólido} & si \quad p \geq u \\ \text{No Bólido} & si \quad p < u \end{cases} \tag{A.1}$$

Donde u es el umbral que se pasa como parámetro.

Observación 4: Si el archivo `.csv` no está creado en `<Directorio_Salida>`, el software lo crea con el encabezado y formato mencionado, con el nombre `Predicciones.csv`.

Observación 5: Cada vez que se ejecuta el módulo `predict`, se concatenan en el archivo de salida al final del mismo n filas, donde n corresponde a la cantidad de videos en `<Directorio_Videos>`.

- `-mod <Modelo.bin>`: Modelo que se quiere usara para predecir la clasificación de los videos en `<Directorio_Videos>`.
- `-u <Umbral>`: Umbral de decisión u usado para determinar la etiqueta correspondiente como se explicó en A.1. Se recomienda usar el umbral que esta referenciado en el `.txt` del modelo.

Un ejemplo de ejecución del módulo `predict`, se puede ver en la Figura A.3. Donde la línea de comandos ingresada fue:
`python autobol.py -m predict -i datos-prueba -o salida-prueba -mod model.bin -u 0.087`, cuya salida es el archivo `Predicciones.csv` ubicado en `<Directorio_Salida>`.

A.6.3.3. Predict Verbose

El módulo `predict verbose`, tiene como objetivo obtener las características de los videos indicados en una ruta especificada, de esta manera permite crear nuevos datos de entrenamiento.

Los requerimientos son:

- `-m predict -verbose`: Se entra a la función `predict verbose`.
- `-i <Directorio_Videos>`: Directorio donde se encuentran los videos que se desean predecir.
- `-o <Directorio_Salida>`: Directorio donde se quiere guardar el archivo de salida, en este caso un `.csv` con el formato:

  Encabezado: Nombre_Archivo, Característica1,.., Característica20, Clasificación
  Fila 1: Video_1, Característica1_1,..., Característica1_20, NaN
  Fila 2: Video_2, Característica2_1,..., Característica2_20, NaN
  Fila 3: Video_3, Característica3_1,..., Característica3_20, NaN

  Nombre: Referido a la raíz común de los archivos.
  Características: Valor real que abstrae una característica del video.
  Clasificación: Valor por defecto `NaN`.

  Luego el usuario debe editar el archivo y sustituir el `NaN` con el siguiente criterio:

$$Clasificacin = \begin{cases} 1 & si \quad \text{Bólido} \\ 0 & si \quad \text{No Bólido} \end{cases} \tag{A.2}$$

  Observación 6: Si el archivo `.csv` no está creado en `<Directorio_Salida>`, el programa lo crea con el encabezado y formato mencionado, con el nombre `Clasificacion.csv`.

  Observación 7: Cada vez que se ejecuta el módulo `predict verbose` se concatenan en el archivo de salida al final del mismo n filas, donde n corresponde a la cantidad de videos en `<Directorio_Videos>`.

Un ejemplo de ejecución del módulo `predict verbose`, se puede ver en la Figura A.5. Donde la línea de comandos ingresada fue:
`python autobol.py -m predict -verbose -i datos-prueba -o salida-prueba`, cuya salida es el archivo `Clasificacion.csv` ubicado en `<Directorio_Salida>`.

A.6.3.4. Predict Localize

El módulo `predict localize`, tiene como objetivo brindar una retroalimentación visual al usuario. A la vez de predecir, también devuelve una imagen con el evento identificado encerrado en un cuadrado, esto permite al usuario ver qué detecta el algoritmo.

Los requerimientos son:

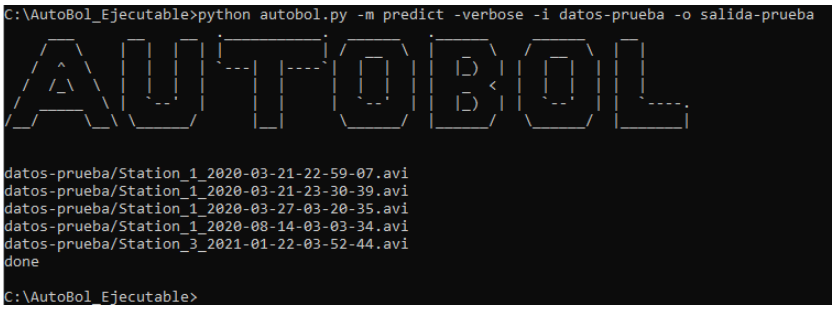


Figura A.5: Ejemplo de ejecución del módulo predict verbose.

	A	B	C	D	T	U	V
1	Video	velocidad media	mean_intensidad	N	loss	recorrido	Clasificación
2	datos-prueba/Station_1_2020-03-21-22-59-07.avi	45,948350019	5104,1621312626	40	350,860844151	57,1054429673	NaN
3	datos-prueba/Station_1_2020-03-21-23-30-39.avi	20,527577229	1148,775	22	97,5008567932	12,0274772255	NaN
4	datos-prueba/Station_1_2020-03-27-03-20-35.avi	41,4488000575	631,430952381	21	105,4922961999	25,3835826398	NaN
5	datos-prueba/Station_1_2020-08-14-03-03-34.avi	39,1987445908	957,4983333333	60	292,2970702623	74,8302540869	NaN
6	datos-prueba/Station_3_2021-01-22-03-52-44.avi	53,9508676283	50850,4440789474	228	338,3185222315	210,7419988516	NaN

Figura A.6: Salida del ejemplo de ejecución del módulo predict verbose con parámetros de entrada vistos.

- `-m predict -localize`: Se entra a la función `predict localize`.
- `-i <Directorio_Videos>`: Directorio donde se encuentran los videos que se desean predecir y devolver la imagen.
- `-o <Directorio_Salida>`: Directorio donde se quiere guardar el archivo de salida y la imagen .png.

Ejemplo de ejecución:

`python autobol.py -m predict -u 0.053 -o salida -i datos-prueba -mod model_0.053.bin -localize`, cuya salida es el archivo `Clasificacion.csv` y una imagen ubicados en `<Directorio_Salida>`.

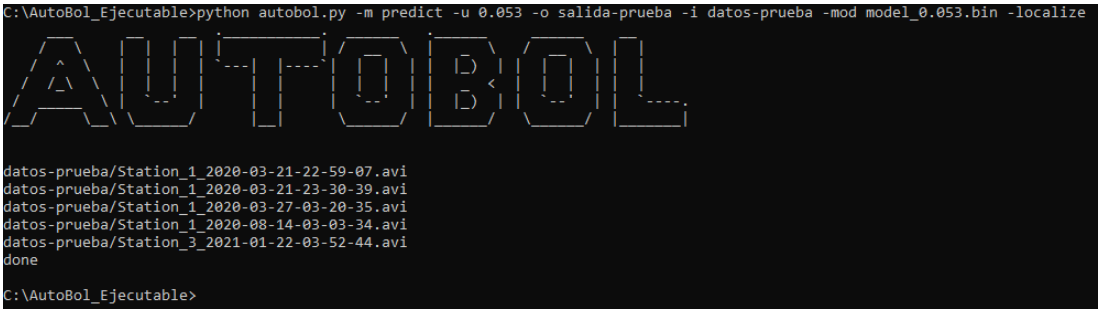


Figura A.7: Ejemplo de ejecución del módulo predict localize.

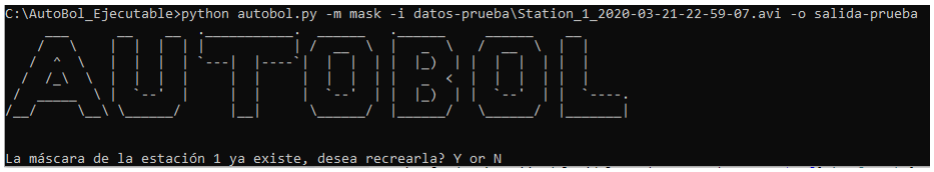


Figura A.8: Ejemplo de ejecución del módulo mask.

A.6.3.5. Mask

El módulo `mask`, se utiliza para crear la máscara de una estación. Este se ejecuta automáticamente en caso de que se intente procesar un video de una estación que no tenga máscara. Por ejemplo, al ejecutar el módulo `predict` con un video de una estación X, donde no se tiene el archivo `Mascara_Station_X`. También se puede ejecutar este módulo por intervención del usuario si quiere o necesita crear una nueva máscara de una estación ya existente. Los requerimientos para ejecutarlo son:

- `-m mask:` Se entra a la función `mask`.
- `-i <Directorio_Video>`: Directorio donde se encuentra el video de la estación de interés que se quiere crear la máscara.

  El formato a seguir debe de ser:

  `Station_X_Año-Mes-Día-Hora-Minutos-Segundos`
  Donde X es el número de la estación.
  Ejemplo: `Station_1_2020-03-21-22-59-07`

  Recordar las estaciones existentes hasta el momento:

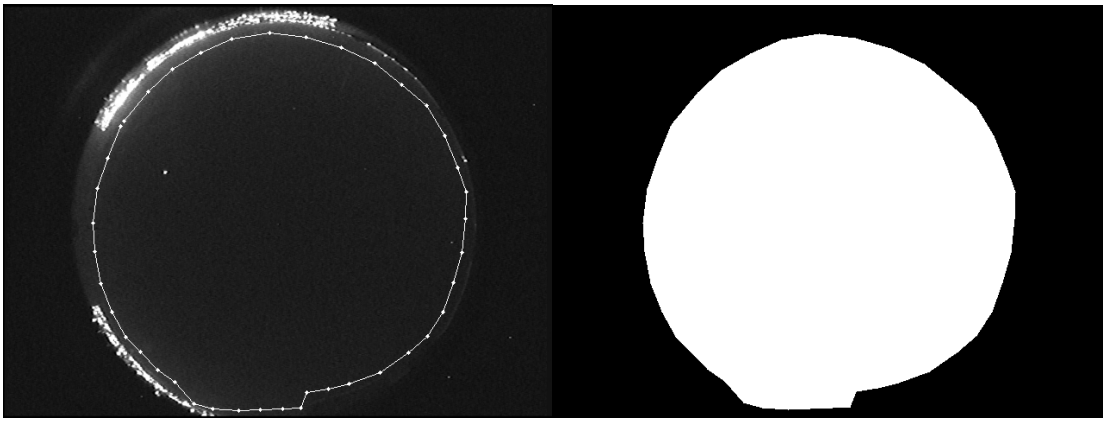
  - Estación 1: Montevideo – Facultad de Ciencias
  - Estación 2: Maldonado –Liceo de San Carlos
  - Estación 3: Florida – Liceo de Casupá

- `-o <Directorio_Salida>`: Directorio donde se guardan las máscaras. Está predefinido a la carpeta Mascaras y se guarda con el nombre `Mascara_Station_E.png`.

Un ejemplo de ejecución del módulo `mask`, se puede ver en la Figura A.8, donde la línea de comandos ingresada fue:
`python autobol.py -m mask -i datos-prueba\Station_1_2020-03-21-22-59-07.avi -o salida-prueba`

Al presionar enter, el software devolverá el mensaje: `La máscara de la estación X ya existe, desea recrearla? Y or N`.

Esperando una respuesta válida `(Y, y, N o n)` por un máximo de 3 intentos. Si no se ingresa una respuesta válida en los primeros 3 intentos o se ingresa `N` o `n`, el software se cerrará y habrá que ejecutar el módulo de nuevo.



(a) Frame con máscara delimitada punto a punto. (b) Máscara generada.

Figura A.9: Ejemplo de ejecución del módulo mask para video de la estación 1.

Si se presiona `Y` o `y`, se abrirá una ventana como se puede ver en la Figura A.9 con un frame aleatorio del video que fue utilizado como input, es decir `<Directorio_Video>`. Una vez que nos aparece el frame, se tiene que ir haciendo click sobre la imagen para delimitar la máscara sobre el horizonte de la toma. Los únicos botones que funcionan en la ventana que está el frame son los siguientes:

- Botón izquierdo del mouse: Agregar punto.
- Botón derecho del mouse: Quitar último punto agregado.
- Tecla `Esc` del teclado: Salir del módulo y guardar la nueva máscara.

Observación 8: Cuando se está delimitando la máscara se pueden ingresar tantos puntos como sean necesarios, se recomienda un polígono convexo.

Observación 9: La cantidad mínima de puntos válidos para crear la máscara es de 3. Si se ingresan una cantidad menor de puntos y se presiona `Esc`, el software saldrá del módulo con el mensaje: `Ingrese una cantidad de puntos superior a 2`.

Observación 10: Se recomienda actualizar la máscara cada vez que se modifique el lugar de la cámara all-sky o aparezcan elementos en la periferia o deformaciones de la cámara que agreguen ruido.

A.7. Solución de problemas

A continuación se listan algunos errores conocidos con su solución:

- Si la respuesta de consola al ejecutar cualquier modulo es:
  `usage: autobol.py [-h] -i INDIR -o OUTDIR -m MODE [-u UMBRAL] [-t TEST_SZ] [-mod MODEL] [-verbose] autobol.py: error: the following arguments are required: Parámetro`
  Con Parámetro:

1. `-i --indir`
2. `-o --outdir`
3. `-m --mode`

Solución: Ejecute nuevamente el módulo agregándole el parámetro faltante:

1. Directorio donde se encuentra/n el/los video/s.
2. Directorio donde se va a guardar la salida.
3. El modulo a utilizar: `predict`, `predict verbose`, `predict localize`, `train` o `mask`.

- Si la respuesta de consola al ejecutar el módulo `predict` es:
  `PermissionError: [Errno 13] Permission denied: 'salida-prueba\\Predicciones.csv'`
  Solución: El archivo `Predicciones.csv` se encuentra abierto, ciérrelo y vuelva a intentarlo.
- Si la respuesta de consola al ejecutar el módulo `predict verbose` es:
  `PermissionError: [Errno 13] Permission denied: 'salida-prueba\\Clasificacion.csv'`
  Solución: El archivo `Clasificacion.csv` se encuentra abierto, ciérrelo y vuelva a intentarlo.

A.8. Preguntas Frecuentes

- ¿En qué sistema operativo se ejecuta el software?
  Se puede ejecutar tanto en Windows como en Linux, teniendo los siguientes requisitos:

  - certifi==2021.10.8
  - charset-normalizer==2.0.7
  - colorama==0.4.4
  - cycler==0.11.0
  - decorator==4.4.2
  - fonttools==4.28.1
  - idna==3.3
  - imageio==2.11.1
  - imageio-ffmpeg==0.4.5
  - joblib==1.1.0
  - kiwisolver==1.3.2
  - matplotlib==3.5.0
  - moviepy==1.0.3
  - networkx==2.6.3
  - numpy==1.21.4
  - opencv-python==4.5.4.58
  - packaging==21.3
  - pandas==1.3.4
  - Pillow==8.4.0
  - proglog==0.1.9
  - pyfiglet==0.8.post1
  - pyparsing==3.0.6
  - python-dateutil==2.8.2
  - python-math==0.0.1
  - pytz==2021.3
  - PyWavelets==1.2.0
  - requests==2.26.0
  - scikit-image==0.18.3
  - scikit-learn==1.0.1
  - scipy==1.7.2
  - setuptools-scm==6.3.2
  - six==1.16.0
  - sklearn==0.0
  - threadpoolctl==3.0.0
  - tifffile==2021.11.2
  - tomli==1.2.2
  - tqdm==4.62.3
  - urllib3==1.26.7
  - xgboost==1.5.0

- ¿Cómo salgo del programa?
  Desde la consola se debe de apretar la combinación de teclas: `Ctrl + C`, esto

termina el proceso que se esté ejecutando, útil para recuperar el control de la consola y luego cerrarla.

- ¿Se puede cambiar la carpeta de las máscaras?
  No se puede, la ubicación de las máscaras esta predefinida en la carpeta `Mascaras` ubicada en la raíz junto a los archivos código del software. Para cambiarla hay que modificar el código.
- ¿Es necesario que los parámetros de los módulos especificados en las secciones A.6.3.1, A.6.3.2, A.6.3.3, A.6.3.4 y A.6.3.5 tengan el orden descrito?
  No es necesario que se respete el orden, es suficiente con que estén todos los parámetros mencionados en el módulo correspondiente.
- ¿Cómo creo un nuevo modelo?
  1. Disponer de un dataset de videos lo suficientemente grande en una carpeta cuyos nombres tengan el formato:
     `Station_E_Año-Mes-Día-Hora-Minutos-Segundos`.
  2. Ejecutar el modo `predict verbose` con esta carpeta de videos, del cual se obtendrá un archivo `Clasificacion.csv`.
  3. Editar la última columna del archivo `Clasificacion.csv` con el criterio descrito en A.2.
  4. Ejecutar el modo `train` pasandole el archivo `Clasificacion.csv` y se obtendrá un nuevo modelo con el umbral recomendado.

En el caso de que usted quiera conocer más acerca de este proyecto o exista alguna duda, es posible contactarse con el equipo de AutoBol y Bocosur a través de los siguientes e-mail de contacto: juanpballestrino@gmail.com, cdeandraya@gmail.com , cuviedo1@gmail.com.

Apéndice B

Funciones Implementadas

B.1. Enmascaramiento

B.1.1. numero_estacion

Pseudocódigo

Algortimo 1 `numero_estacion(file)`

Require: `file`, string con el nombre del archivo.
Ensure: f, entero con el número de estación correspondiente.

```
f = 0
Encontro ← False
while (f < 100) and (Encontro == False) do
  palabra = 'Station_' + str(f)
  if file.find(palabra) != -1 then
    Encontro ← True
  end if
  if file.find(palabra) == -1 then
    f = f + 1
  end if
end while
return f
```

Diagrama de Flujo

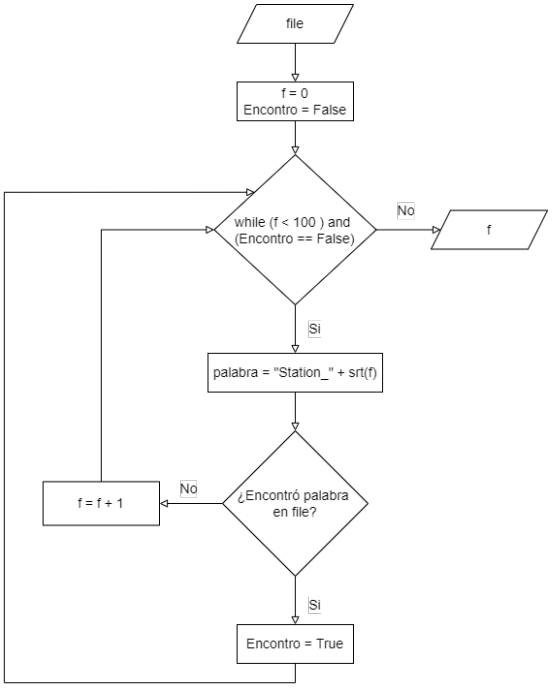


Figura B.1: Diagrama de Flujo de la función numero_estacion.

B.1.2. MousePoint

Pseudocódigo

Algortimo 2 `MousePoint(event,x,y)`

Require: `event`, código identificador de los botones del mouse o teclado. `x`, posición horizontal del evento. `y`, posición vertical del evento.

```
global counter
if event == cv2.EVENT_LBUTTONDOWN then
  colll.append(x)
  filll.append(y)
  counter = counter + 1
end if
if event == cv2.EVENT_RBUTTONDOWN then
  colll.pop( len(colll) - 1)
  filll.pop( len(filll) - 1)
  counter = counter - 1
end if
```

Diagrama de Flujo

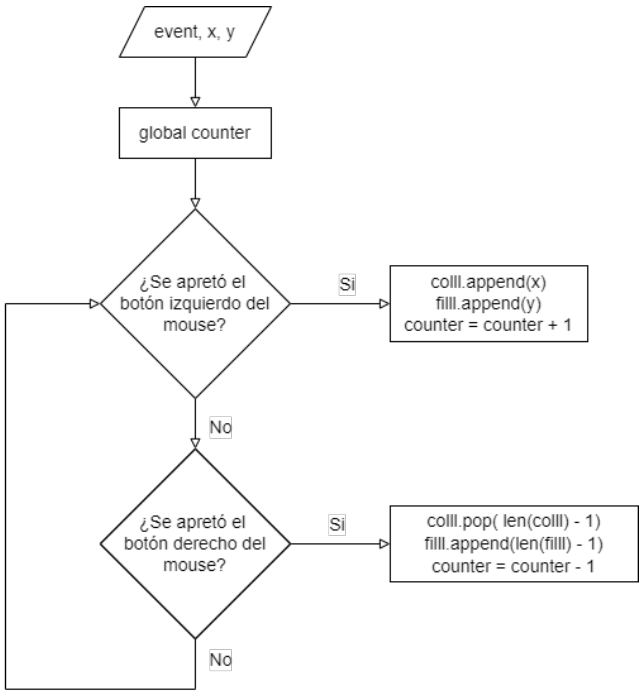


Figura B.2: Diagrama de Flujo de la función MousePoint.

B.1.3. CrearMascara

Pseudocódigo

Algortimo 3 `CrearMascara(file)`

Require: `file`, string con el nombre del archivo.
Ensure: Guarda la máscara creada en la carpeta de mascaras.

```
global filll
global colll
global counter
filll = [ ]
colll = [ ]
counter = -1
f = numero_estacion(file)
Imprimo mensaje: Creando máscara para la estación f...
frames,fps = get_frames(file)
Obtengo un frame aleatorio de frames
_,filas,columnas = frames.shape
while True do
   k ← Código de boton del mouse o teclado
  if k == 27 (Esc) then
    Termino procedimiento
  end if
  if k ≠ 27 (Esc) then
    Ploteo el frame aleatorio.
    Llamo a la función MousePoint para obtener los puntos y crear
    la máscara.
  end if
end while
if counter < 3 then
  return     Mensaje informando que la cantidad de puntos debe ser
  superior a 2
end if
if counter > 2 then
  Creo mascara con 1s adentro y 0s afuera, de dimenciones filas x
  columnas.
  Guarda la máscara creada en la carpeta de mascaras.
end if
```

Diagrama de Flujo

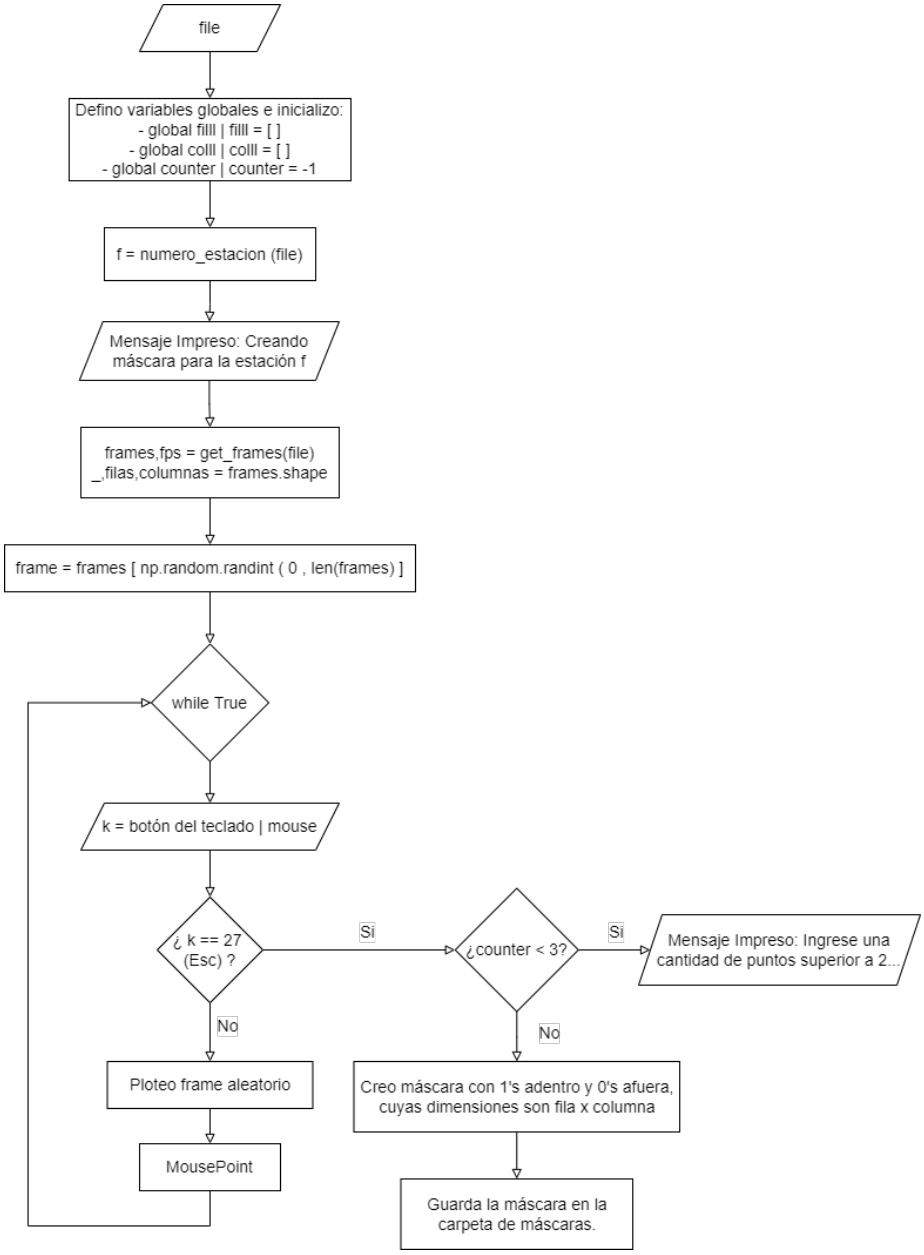


Figura B.3: Diagrama de Flujo de la función CrearMascara.

B.2. Reconstrucción del brillo

B.2.1. xq_to_quadratic

Pseudocódigo

Algortimo 4 `xy_to_quadratic(xy)`

Require: `xy`, tupla con dos elementos.

Ensure: `w=`$(1, x, y, x^2, xy, y^2)$

```
y = xy[0]
x = xy[1]
n = len(x)
m = 6
X_poly = np.ones((n , m))
X_poly [:, 1] = x
X_poly [:, 2] = y
X_poly [:, 3] = x^2
X_poly [:, 4] = xy
X_poly [:, 5] = y^2
return X_poly
```

B.2.2. recostruir_brillo

Pseudocódigo

Algortimo 5 `reconstruccion_brillo(img,sat)`

Require: `img`, imagen donde se encuentra el objeto con brillo saturado ; `sat`, matriz con True donde img esta saturado

Ensure: imagen con brillo reconstruido

```
xy_sat ← np.nonzero(saturated)
xy_train ← np.nonzero(np.logical_and(np.greater(img,140),np.logical_not(sat)))

z_train ← img[xy_train]
X_sat ← xy_to_quadratic(xy_sat)
X_train ← xy_to_quadratic(xy_train)
res ← np.linalg.lstsq(X_train,z_train)
a ← res[0]
z_pred ← np.dot(X_sat,a)
rec ← np.copy(img)
rec[xy_sat] ← z_pred
return img_rec
```

Diagrama de Flujo

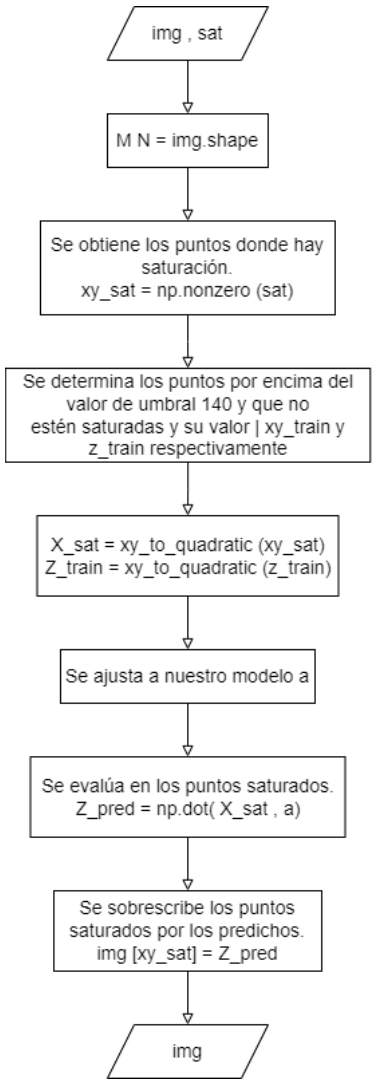


Figura B.4: Diagrama de Flujo de la función reconstruir_brillo.

B.3. Tracking

B.3.1. tracking

Pseudocódigo

Algortimo 6 `tracking(ruta_entrada)`

Require: `ruta_entrada`, string con el nombre del archivo.

Ensure: `ruta_entrada`, string con el nombre del archivo. `tiempo`, arreglo numpy con tiempos. `X_Model`, arreglo numpy con las posiciones horizontales. `Y_Model`, arreglo numpy con las posiciones verticales. `LC`, arreglo numpy con el valor de intensidad relativo a la intensidad de fondo. `N`, entero con el numero de frames en el que el objeto fue detectado. `total_dispersion`, entero con el valor de dispersion de los puntos. `FWHMarray`, arreglo numpy con el valor de fwhm frame a frame. `loss`, entero con el valor del error cuadrático medio entre la trayectoria que hace el objeto y el modelo de una circunferencia. `d`, entero con el valor de la distancia recorrida.

```
frames,fps = get_frames(ruta_entrada)
cantidad_frames,cantidad_filas,cantidad_columnas = frames.shape
fondo = numpy.percentile(frames, 5, axis=0)
maximo = numpy.percentile(frames, 95, axis=0)
if no está creada la máscara? then
  mask = CrearMascara(ruta_entrada)
end if
for i in range (len(cantidad_frames)) do
  resta = | frames[i] - fondo |
  tracking[i] = resta
  fila[i],columna[i] = argumentos del máximo valor de intensidad en
  video_tracking[i]
end for
umbral: Se calcula para detectar eventos con la parte no nula
embascarada. Percentil 20%
for i in range (len(cantidad_frames)) do
  if El máximo del frame es mayor a umbral then
    x,y=center_mass(video_tracking[i],fila[i],columna[i],win)
    X_detect[i] = x
    Y_detect[i] = y
    tiempo.append(i/fps)
  end if
end for
```

```
if len(X_detect <3) then
  print(Evento con pocos puntos)
  return ruta_entrada,nan,nan,nan,nan,nan,nan,nan,nan,nan
end if
if len(X_detect >2) then
  pocket = RANSAC(X_detect,Y_detect,residual_treshold)
  Xc = pocket[0][0][0]
  Yc = pocket[0][1][0]
  r = pocket[0][2][0]
  pertenecen = pocket[0][3]
  loss = pocket[0][4]
  no_pertenecen = np,logical_not(pertenecen)
  umbral2: Se calcula con los puntos que pertenecen al modelo de la
  cefa. Percentil 5%
  for i in range (len(cantidad_frames)) do
    if El máximo del frame es mayor a umbral2 then
      x,y=center_mass(video_tracking[i],fila[i],columna[i],win)
      X_Model[i] = x
      Y_Model[i] = y
    end if
  end for
  Se filtran los puntos que pasaron las 2 vueltas X_Model e Y_Model

  Se calcula: distancia recorrida (d) y total_dispersion
  Int = [ ]
  FWHM = [ ]
  LC = [ ]
  for p in range (len(X_frames)) do
    if El objeto satura then
      Se reconstruye el brillo
    end if
    FWHM[p]
    Int[p]
    LC[p]
  end for
end if
return   ruta_entrada, tiempo, X_model, Y_model, LC, len(X_model),
total_dispersion, FWHM, loss, d
```

Diagrama de Flujo

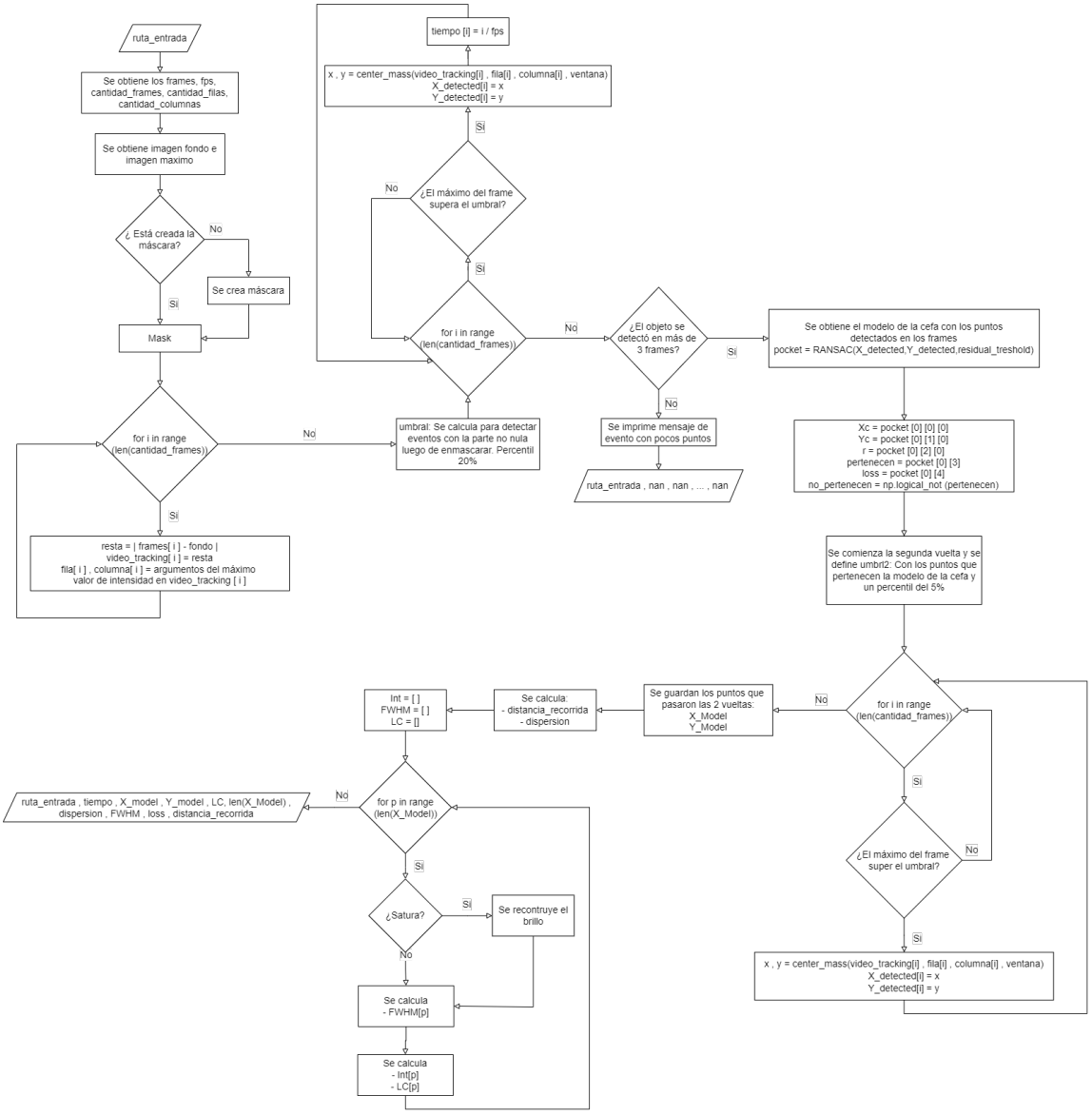


Figura B.5: Diagrama de Flujo de la función tracking.

B.3.2. get_frame

Pseudocódigo

Algortimo 7 `get_frames(ruta_entrada)`

Require: `ruta_entrada`, string con la ubicación del video.

Ensure: Arreglo numpy con los frames del video y un entero con el valor de fps utilizado.

```
video ← video.iter.frames()
frames = [ ]
for i in range video.iter.frames() do
  frames.append(i[:,:,0])
end for
 frames = np.array(frames)
return frames, video.fps
```

Diagrama de Flujo

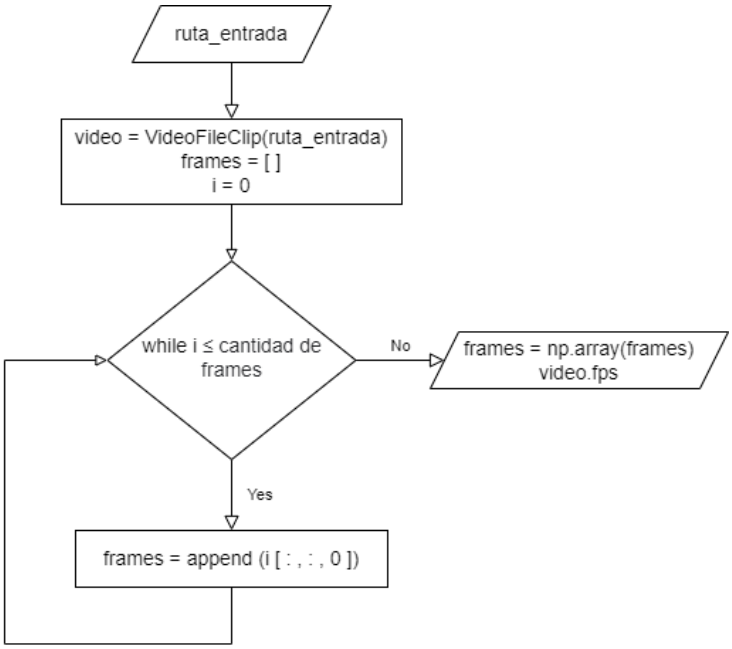


Figura B.6: Diagrama de Flujo de la función get_frames.

B.3.3. center_mass

Pseudocódigo

Algortimo 8 `center_mass(frame,i_obj,j_obj,window)`

Require: `frame`, frame donde se encuentra el objeto ; `i_obj`: coordenada horizontal del objeto ; `j_obj`, coordenada vertical del objeto ; window, tamaño de la ventana donde se calcula el centro de masa .

Ensure: Coordenadas del centro de masa del objeto detectado.

```
ventana_j← window
ventana_i← window
Ancho, Alto ← frame.shape
while (i_obj - window <0) OR (i_obj + window >Ancho) do
  ventana_i← ventana_i-1
end while
while (j_obj - window <0) OR (j_obj + window >Alto) do
  ventana_j← ventana_j-1
end while
x_c ← eq(1)
y_c ← eq(2)
return x_c,y_c
```

Diagrama de Flujo

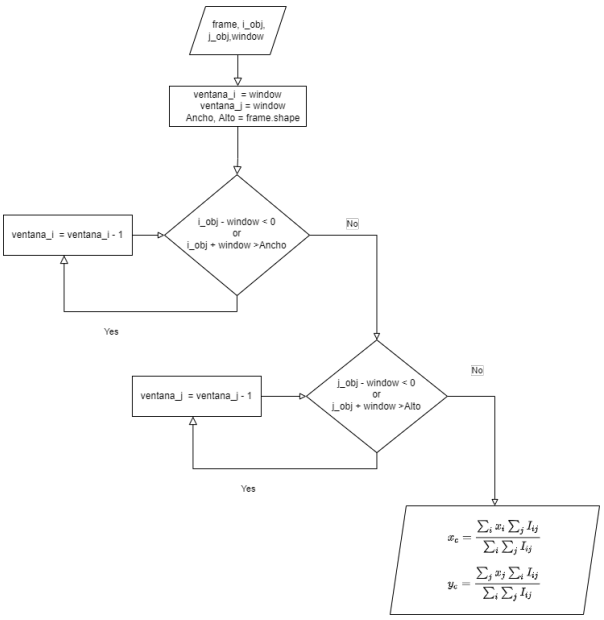


Figura B.7: Diagrama de Flujo de la función center_mass.

B.3.4. RANSAC_3

Pseudocódigo

Algortimo 9 RANSAC_3(x,y,residual_treshold)

Require: x, coordenadas horizontales del evento; y: coordenadas verticales del evento; residual_threshold: parámetro que determina tolerancia del algoritmo para clasificar un punto como inlier.

Ensure: Centro y radio del mejor modelo encontrado. Lista de inliers. Loss: MAD de cada uno de los puntos con respecto al modelo.

```
actual_loss , k ← 50000 , 0
while (k ≤ 900 and actual_loss ≥ length(x)/4) do
  a,b,c ← Muestreo 3 puntos aleatoriamente
  if a,b,c estan alineados then
    k ← k+1
  end if
  if a,b,c no están alineados then
    x_c,y_c,r ← Circunferencia que pasa por los 3 puntos
    loss=[]
    for all (x,y) do
      loss.append(|dist((x_c,y_c),(x,y))-r|)
    end for
    MAD← median_absolute_deviation(loss)
    if sum(loss)<actual_loss then
      inliers=[]
      for all k in loss do
        if loss[k] >residual_threshold * MAD then
          inliers.append(False)
        end if
        if loss[k] <residual_threshold * MAD then
          inliers.append(True)
        end if
      end for
      pocket=[]
      actual_loss ← sum(loss)
      pocket.append([xc,yc,r,inliers,actual_loss])
    end if
  end if
  k ← k+1
end while
return pocket
```

Diagrama de Flujo

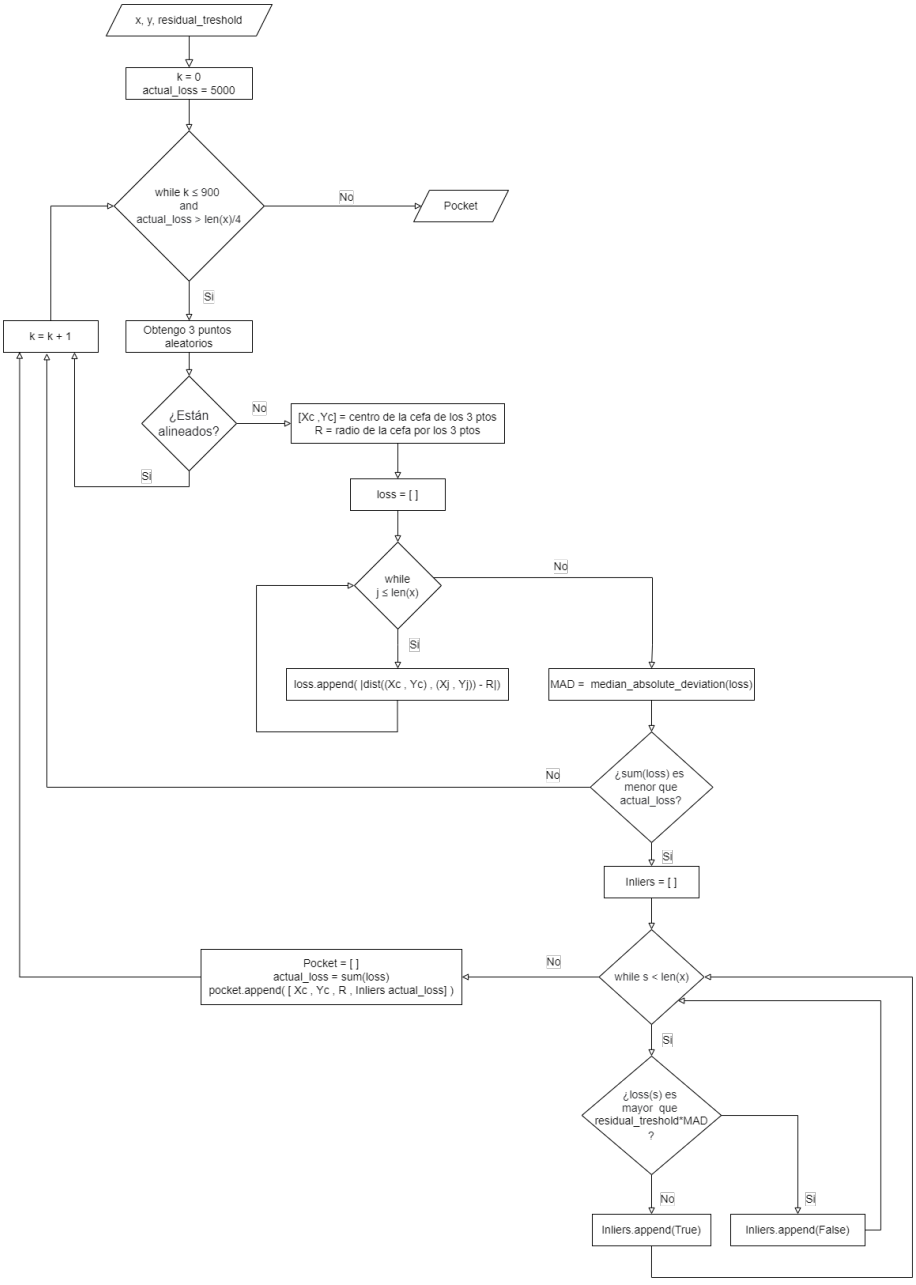


Figura B.8: Diagrama de Flujo de la función RANSAC.

B.3.5. filtro_extra

Pseudocódigo

Algortimo 10 `filtro_extra(X_Bol,Y_Bol,pertenecen)`

Require: `X_Bol`, coordenadas horizontal del objeto ; `Y_Bol`, coordenadas vertical del objeto ; `pertenecen` vector booleano con la información de si el punto pertenece al modelo de la cefa o no.

Ensure: vector booleano sin outliers

```
cantidad = 2
while cantidad >0 do
  X_ord ← vector X_Bol ordenado
  Y_ord ← vector Y_Bol ordenado
  n ← punto medio de X_Bol
  X_Q1 ← mediana X_ord[:n]
  X_Q3 ← mediana X_ord[n:]
  X_IQR = X_Q3 - X_Q1
  X_IQR = X_Q3 - X_Q1
  X_High_outliers ← X_Q3 + 1.5X_IQR
  X_Low_outliers ← X_Q1 - 1.5X_IQR
  Y_Q1 ← mediana Y_ord[:n]
  Y_Q3 ← mediana Y_ord[n:]
  Y_IQR = Y_Q3 - Y_Q1
  Y_IQR = Y_Q3 - Y_Q1
  Y_High_outliers ← Y_Q3 + 1.5Y_IQR
  Y_Low_outliers ← Y_Q1 - 1.5Y_IQR
  for i in range(len(pertenecen)) do
    if X_Bol[i] < X_Low_outliers or X_Bol[i] > X_High_outliers or
    Y_Bol[i] < Y_Low_outliers or Y_Bol[i] > Y_High_outliers then
      pertenecen[i] = False
    end if
  end for
  X_Bol = X_Bol[pertenecen]
  Y_Bol = Y_Bol[pertenecen]
  cantidad = cantidad - 1
end while
return pertenecen
```

Diagrama de Flujo

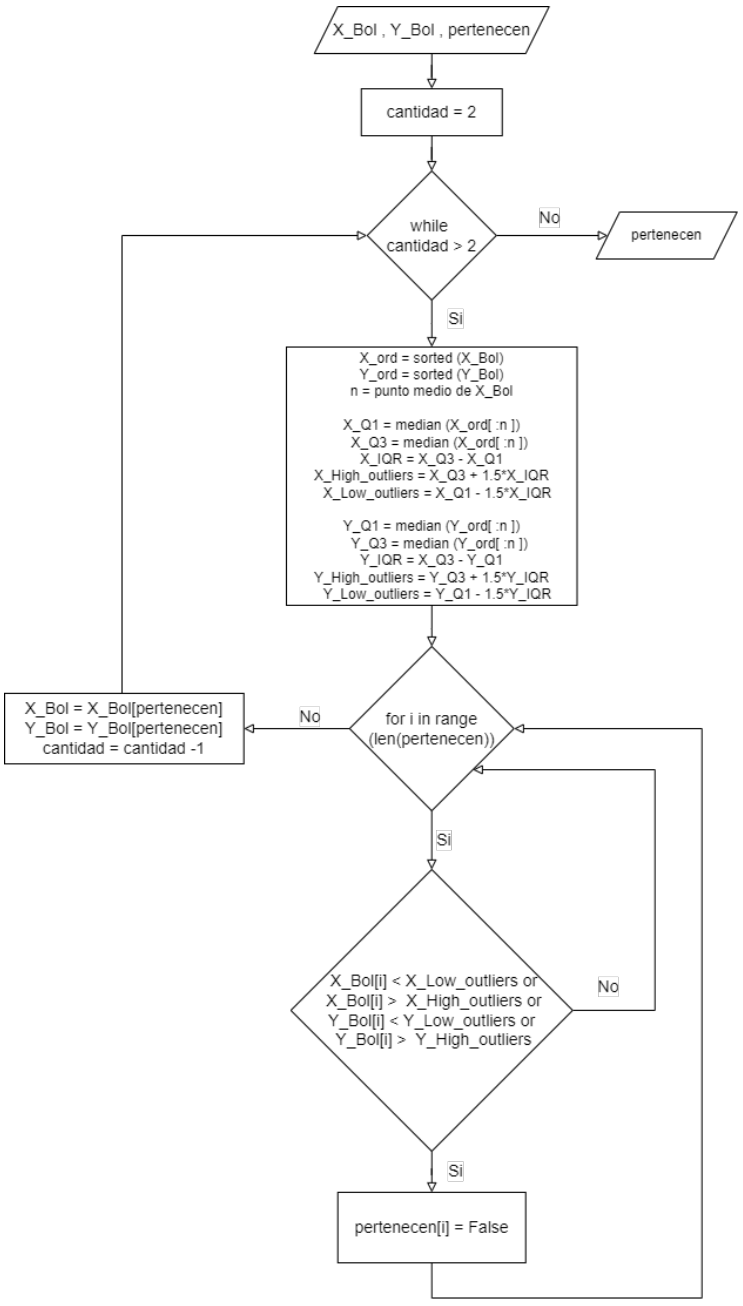


Figura B.9: Diagrama de Flujo de la función filtro_extra.

B.4. Características

B.4.1. distancia_recorrida

Pseudocódigo

Algortimo 11 `distancia_recorrida(r,ang)`

Require: `r`, radio del modelo de circunferencia descrito por el evento ; `ang`, angulo de la circunferencia recorrida descrito por el evento.

Ensure: Distancia recorrida por el evento.

return `r * ang`

B.4.2. dispersion

Pseudocódigo

Algortimo 12 `dispersion(x,y)`

Require: `x`, coordenadas del eje horizontal del evento; `y`: coordenadas del eje vertical del evento;

Ensure: `disper`, entero con el valor el valor de dispersión de los puntos.

```
x_mean = np.mean(x)
y_mean = np.mean(y)
c = [x_mean , y_mean]
disper ← 0
for i in range(len(x)) do
  a = x[i]
  b = x[i]
  disper = disper + 1/(len(x)*sqrt((a-c[0])^2+(b-c[1])^2))
end for
return disper
```

Diagrama de flujo

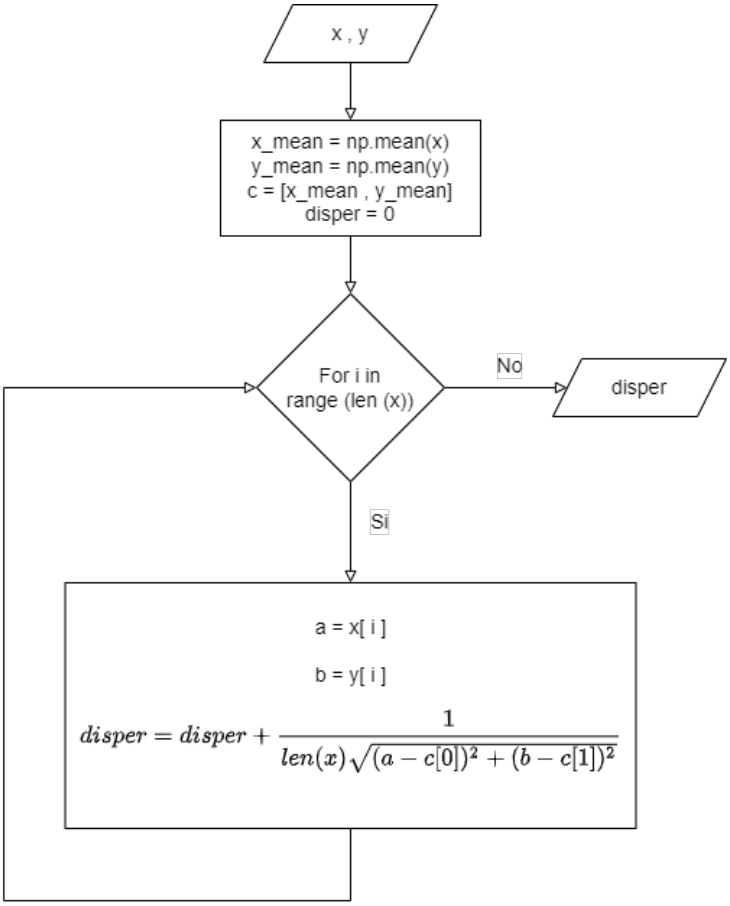


Figura B.10: Diagrama de Flujo de la función dispersion.

B.4.3. FWHM

Pseudocódigo

Algortimo 13 `FWHM(frame,CM_f,CM_c)`

Require: `frame`, frame en presencia del objeto ; `CM_f`: coord. del centro de masa horizontal del objeto ; `CM_c`, coord. del centro de masa vertical del objeto.

Ensure: FWHM de la intensidad de brillo del evento detectado. Valor entero y real.

```
w ← 1 Cuadrado de lado 2*w
Borde , Diff , Entro ← 1000 , 100 , False
Int ← Frame[0][round(CM_f),round(CM_c)]
while (Borde ≤ Int/2  and  Dif > 0.01) do
  if El perímetro de interés se va fuera de los límites then
    return 255 , 255
  end if
  if El perímetro de interés está dentro de los límites then
    Aux ← Int. promedio de píxeles en el perímetro del cuadrado 2*w
    Diff ← |Borde-Aux|
    Anterior ← Borde
    Borde ← Aux
    if Borde > 1.2 * Anterior then
      Diff , Entro ← 0 , True
    end if
    w ← w+1
  end if
end while
if Entro == True then
  a = Anterior - Borde
  if a == 0 then
    return w,w
  end if
end if
if Entro == False then
  a = Borde - Anterior
  if a == 0 then
    return w,w
  end if
end if
fwhm_spix = (Int/2−Anterior−a*w)/a
return fwhm, fwhm_spix
```

Diagrama de Flujo

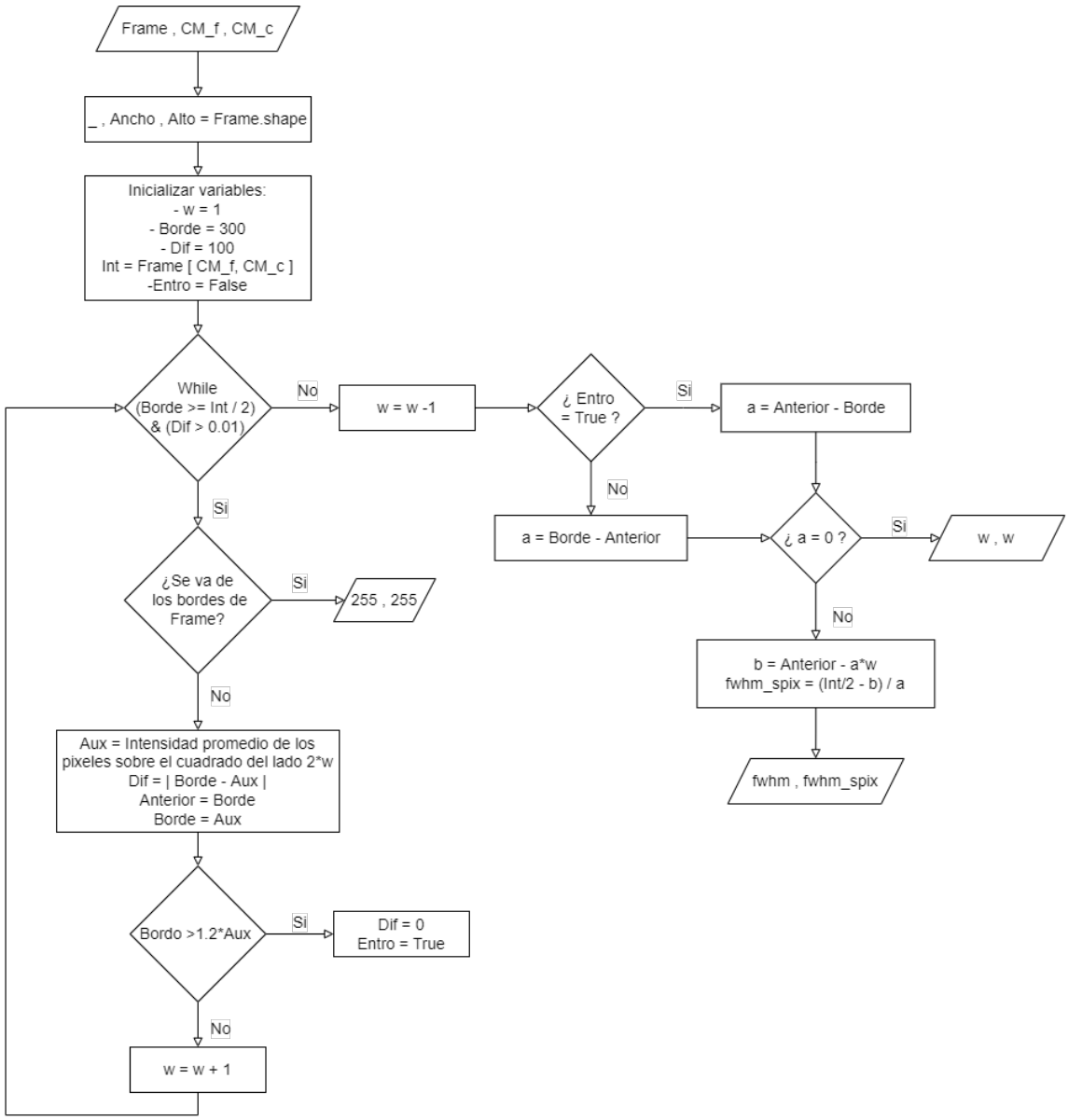


Figura B.11: Diagrama de Flujo de la función FWHM.

B.4.4. CurvaDeLuz

Pseudocódigo

Algortimo 14 `CurvaDeLuz(frame,Fondo,CM_f,CM_c,Size_Win)`

Require: `frame`, frame donde se encuentra el objeto ;`fondo`, fondo de la toma sin el objeto ; `CM_f`: coordenada del centro de masa horizontal del objeto ; `CM_c`, coordenada del centro de masa vertical del objeto ; `Size_Win` Tamaño de la ventana de interes.

Ensure: Intensidad del brillo del objeto.

`Región_Interes` ← Cuadrado de lado 2 veces `Size_Win` centrado en `CM_f`,`CM_c`

`lc` ← $\sum$(`frame`[`Región_Interes`])
`fon` ← $\sum$(`fondo`[`Región_Interes`)
`I` ← `lc` - `fon`
return `I`

Diagrama de Flujo

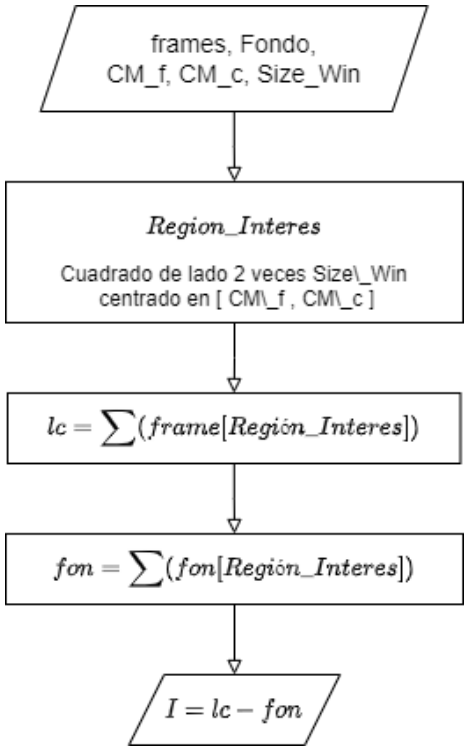


Figura B.12: Diagrama de Flujo de la función CurvaDeLuz.

B.4.5. V_feature

Pseudocódigo

Algortimo 15 `V_feature(array)`

Require: `array`, Numpy array.

Ensure: `feature 1`, entero con la suma de la distancia entre puntos de la curva resampleada al valor de fps. `Feature 2`, entero con la suma de la distancia entre puntos de la curva normalizada.

```
M ← fps
N = len(array)
R ← resample(array , M)
F1 = 0
F2 = 0
for i in range (1 , M) do
  F1 = F1 + |R[i]-R[i-1]|
end for
for i in range (1 , N) do
  F2 = F2 + |array[i]-array[i-1]|
end for
return F1, F2
```

Diagrama de Flujo

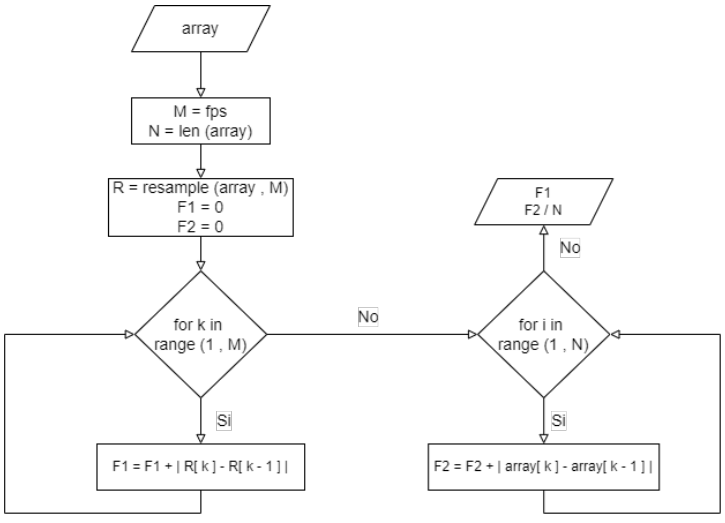


Figura B.13: Diagrama de Flujo de la función V_feature.

B.4.6. velocidad

Pseudocódigo

Algortimo 16 `velocidad(t,x,y)`

Require: `t`, vector de tiempos del video; `x`: vector con la posición horizontal del evento en el video; `y`, vector con la posición vertical del evento en el video.

Ensure: Velocidad instantánea del objeto frame a frame.

```
v = [ ]
i ← 0
while i ≤ len(t) do
  if i==0 then
    d ← Distancia entre posición ``i'' e ``i+1''
    t ← Tiempo entre posición ``i'' e ``i+1''
  end if
  if i > 0 then
    d ← Distancia entre posición ``i'' e ``i-1''
    t ← Tiempo entre posición ``i'' e ``i-1''
  end if
  v.append(d/t)
  i ← i+1
end while
return v
```

Diagrama de Flujo

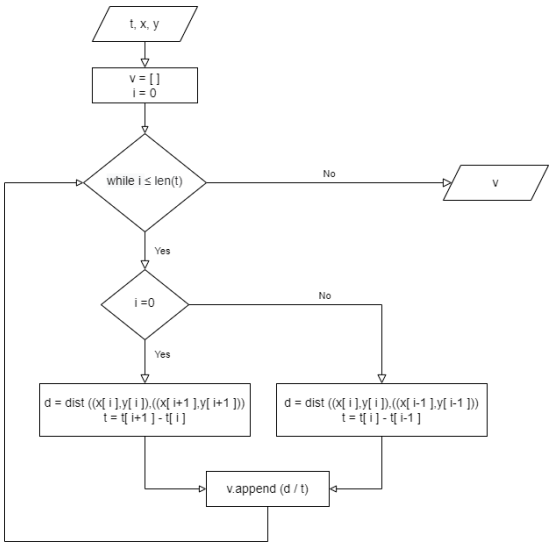


Figura B.14: Diagrama de Flujo de la función velocidad.

Esta página ha sido intencionalmente dejada en blanco.

Referencias

[Alcor System, 2022] Alcor System (2022). Alphea - All-Sky Camera. *Página oficial: Alcor-System* [Fecha de acceso: 2022-04-19].

[Ames Research Center, 2021] Ames Research Center (2021). Nasa meteorite tracking and recovery network. *NASA Meteorite Tracking and Recovery Network* [Fecha de acceso: 2022-01-22].

[Aurélien Géron, 2019] Aurélien Géron (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly, 2nd edition.

[Aznar et al., 2016] Aznar, J., A.Castellón, Gálvez, F., Martínez, E., Troughton, B., Núñez, J., and Villalba, A. (2016). The meteor and fireball network of the sociedadmalagueña de astronomía. *Revista Mexicana de Astronomía y Astrofísica*, v. 48:99–102.

[Bland et al., 2015] Bland, P., Benedix, G., and Team, D. F. N. (2015). Catching a falling star (or meteorite): Fireball camera networks in the 21st century. *Elements: An international magazine of Mineralogy, Geochemistry, and Petrology*, v. 11(3):160–161.

[Ceplecha et al., 1998] Ceplecha, Z., Borovicka, J., Elford, W. G., Revelle, D., Hawkes, R., Porubcan, V., and Simek, M. (1998). Meteor phenomena and bodies. *Space Science Reviews*, v. 84:1–5.

[Colas et al., 2020] Colas, F., Zanda, B., S.Bouley, Jeanne, S., Malgoyre, A., Birlan, M., Blanpain, C., Gattacceca, J., Jorda, L., Lecubin, J., Marmo, C., Rault, J., Vaubaillon, J., Vernazza, P., Yohia, C., Gardiol, D., Nedelcu, A., Poppe, B., and J. Rowe, ..., A. Z. (2020). Fripon: a worldwide network to track incoming meteoroids. *Astronomy and astrophysics*, v. 644(A53).

[E. E. Orozco Guillén, 2007] E. E. Orozco Guillén (2007). Formación de imágenes panorámicas para aplicaciones arqueológicas.

[Fishcler and Bolles, 1981] Fishcler, M. A. and Bolles, R. C. (1981). Random sample consensus. *Communications of the ACM*, v. 24(6):381–395.

[Fitzgibbon, 2001] Fitzgibbon, A. (2001). Simultaneous linear estimation of multiple view geometry and lens distortion. volume 1, pages I–125.

[Flores and Braun, 2011] Flores, P. and Braun, J. (2011). Algortimo ransac: fundamento teórico. Technical report, Facultad de Ingeniería, UDELAR.

Referencias

[FRIPON: Database web frontend, 2021] FRIPON: Database web frontend (2021). Página oficial de la organización. *Fireball Recovery and InterPlanetary Observation Network* [Fecha de acceso: 2022-01-24].

[G. Tancredi, 2022] G. Tancredi (2022). Informe sobre la situación de la "red de detección de bólidos del cono sur" bocosur.

[Hughes et al., 2008] Hughes, C., Glavin, M., Jones, E., and Denny, P. (2008). Review of geometric distortion compensation in fish-eye cameras. In *IET Irish Signals and Systems Conference (ISSC 2008)*, pages 162–167.

[J. C. Tulic, 2003] J. C. Tulic (2003). Detección Automática de Meteoros en Cámaras All-Sky mediante la transformada de Hough.

[J. M. Caldas, 2012] J. M. Caldas (2012). Sistema Todo Cielo para la Detección y Procesamiento de Bólidos y Determinación de Fracción de Nubosidad Diurna.

[Lee et al., 2019] Lee, M., Kim, H., and Paik, J. (2019). Correction of barrel distortion in fisheye lens images using image-based estimation of distortion parameters. *IEEE Access*, 7:45723–45733.

[NASA Meteoroid Environment Office (MEO), 2022] NASA Meteoroid Environment Office (MEO) (2022). Nasa's all sky fireball network. *NASA's All SKy Fireball Network* [Fecha de acceso: 2022-01-22].

[Oberest et al., 1998] Oberest, J., Molau, S., Heinlein, D., Gritzner, C., Schindler, M., Spurny, P., Ceplecha, Z., Rendtel, J., and Betlem, H. (1998). The "european fireball network": Current status and future prospects. *Meteoritics & Planetary Science*, v. 33(1):49–56.

[Space Science and Technology Centre at Curtin University, 2020] Space Science and Technology Centre at Curtin University (2020). Global fireball observatory. *Global Fireball Observatory - Partners*[Fecha de acceso: 2022-01-22].

[Tianqi Chen, Carlos Guestrin, 2016] Tianqi Chen, Carlos Guestrin (2016). Xgboost: A scalable tree boosting system. *XGBoost: A Scalable Tree Boosting System.* [Online; accessed 13-March-2022].

[Wang et al., 2009] Wang, A., Qiu, T., and Shao, L.-T. (2009). A simple method of radial distortion correction with centre of distortion estimation. *Journal of Mathematical Imaging and Vision*, 35:165–172.

[Wikipedia contributors, 2021a] Wikipedia contributors (2021a). Desert fireball network — Wikipedia, the free encyclopedia. *Desert Fireball Network*[Fecha de acceso: 2022-01-22].

[Wikipedia contributors, 2021b] Wikipedia contributors (2021b). European fireball network — Wikipedia, the free encyclopedia. *European Fireball Network* [Fecha de acceso: 2022-01-22].

[Wikipedia contributors, 2021c] Wikipedia contributors (2021c). Pinhole camera model — Wikipedia, the free encyclopedia. *Pinhole* [Fecha de acceso: 2022-03-11].

[Wikipedia contributors, 2021d] Wikipedia contributors (2021d). Random sample consensus – Wikipedia, la enciclopedia libre. *Random sample consensus* [Fecha de acceso: 2022-01-15].

[Wikipedia contributors, 2021e] Wikipedia contributors (2021e). Trade-off — wikipedia, la enciclopedia libre. [*Trade-off*[Fecha de acceso: 2022-04-19]].

[Wikipedia contributors, 2022] Wikipedia contributors (2022). Percentil — wikipedia, la enciclopedia libre. [*Percentil*[Fecha de acceso: 2022-04-19]].

[Wood, 1906] Wood, R. W. (1906). Fish-eye views, and vision under water. pages 160 and 621–622.

[XGBoost developers, 2021] XGBoost developers (2021). Xgboost documentation. *XGBoost documentation*. [Online; accessed 13-March-2022].

Esta página ha sido intencionalmente dejada en blanco.

Índice de tablas

Esta página ha sido intencionalmente dejada en blanco.

Índice de figuras

Esta es la última página.
Compilado el viernes 29 abril, 2022.
http://iie.fing.edu.uy/