UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Implementación y programación de un algoritmo de detección de incendios forestales para México utilizando imágenes satelitales GOES 16/ABI con software libre.

TESIS

Que para obtener el título de

Ingeniero Geomático

P R E S E N T A

Colvert Gomez Rubio

DIRECTORA DE TESIS

Dra. Lilia de Lourdes Manzo Delgado

Ciudad Universitaria, Cd. Mx., 2022

Agradecimientos

A mis amados padres, Irene Rubio Herrera y Demetrio Gomez Aguilar, por su amor incondicional, consejos y valores que me enseñaron. Por darme su vida y su tiempo, sin ustedes, no lo habría logrado.

A mis hermanos, Kenia Gomez Rubio y Marco Alonso Gomez Rubio, siempre han estado para mí cuando lo he necesitado, escuchándome y brindarme su ayuda cuando fue necesario. Sé que siempre podre contar con ustedes y ustedes conmigo.

A toda mi familia y en especial a mi tía Maria Feliz Gomez Aguilar, por tus consejos y porque, aun estando lejos, siempre te has preocupado por mi bienestar y mi formación.

A la Universidad Nacional Autónoma de México (UNAM), a la Faculta de Ingeniería (FI) y a todos los docentes que estuvieron en el camino de mi formación, por todo el conocimiento y experiencias que llevaré en mi memoria para siempre.

A la Dra. Lilia de Lourdes Manzo Delgado, por aceptar dirigir el presente trabajo, siempre teniendo tiempo para aconsejarme, corregirme y motivarme. Una de las personas e investigadora que más admiro por su vocación y dedicación.

Al Instituto de Geografía (IG) y en especial al Dr. Jorge Prado Molina por sus consejos y el apoyo de una beca para el desarrollo de este trabajo.

A mis profesores: M.I. Adolfo Reyes Pizano, Ing. Roberto Carlos de la Cruz Sánchez, M.C.T. Maria Elena Osorio Tai e Ing. Neith Moreno Rodríguez por aceptar ser sinodales del presente trabajo y todo el conocimiento que me brindaron.

A mis amigos: Aníbal Cortez, Areli Jonas, Jesús Hernández, Karen Vázquez, Ricardo Martínez, Rocío Salas, Salvador Mena y Sandra Calixto, por todos los momentos y risas que vivimos, si no los hubiera conocido mi experiencia de la universidad hubiera sido muy distinta.

Y por último, pero no menos importante, a la gran comunidad de desarrolladores y programadores que ponen a disposición su conocimiento a través de blogs y sitios web, solucionando muchos problemas que me enfrente al desarrollar este trabajo.

Índice general

Acrónimos y abreviaturas....1
Introducción....5
1. Marco teórico....7
1.1. Objetivos....8
1.1.1 Objetivo General....8
1.1.2 Objetivos Particulares....8
1.2. Antecedentes....8
1.2.1 Incendios forestales en México....9
1.2.2 Historia de GOES....10
1.2.3 Software libre y Código Abierto....21
1.3. FIRMS....25
1.4. GOES 16....29
2. Algoritmo....47
2.1. Creación de umbrales....48
2.1.1 Creación de eventos FIRMS/VIIRS....49
2.1.2 Muestras GOES 16....52
2.1.3 Filtro para obtención de umbrales por día....53
2.2. Obtención de los puntos de calor....54
2.2.1 Preparación de imágenes GOES 16....55
2.2.2 Extracción de umbral máximo de 10 días....57
2.2.3 Capas estáticas....58
2.2.4 Pixel potencial....62
2.2.5 Prueba de contexto....64
2.2.6 Caracterización de los puntos de calor....65
3. Procesamiento de datos espaciales con Python....67
3.1. Bibliotecas y módulos utilizados....71
3.2. Procesos aplicados a datos ráster....73
3.2.1 Exploración del NetCDF....73
3.2.2 Creación de un archivo GeoTIFF....78
3.2.3 Procesos aplicados a un GeoTIFF....80
3.2.4 Prueba de contexto....81
3.3. Procesos aplicados a datos vectoriales....84
3.3.1 Descarga de los puntos FIRMS/VIIRS....84
3.3.2 Apertura y exploración de un Shapefile....85
3.3.3 Procesos al GeoDataFrame....86
3.4. Automatización del algoritmo....87
4. Resultados....88
4.1. Scripts....97

s1_umbrales_auto.py....97
s1_1_crea_puntos_firms.py....97
s1_2_netcdf_a_tif.py....98
s1_3_crea_eventos_firms.py....100
s1_4_muestras_goes_xdia.py....101
s1_5_crea_umbrales.py....103
s2_incendios_auto.py....105
s2_1_prepara_imagenes_geo.py....106
s2_2_extrae_umbral_10dias_max.py....108
s2_3_pp_y_pc.py....109
s2_4_crea_shp_csv.py....110
s3_resultados_csv_shp_xdia_auto.py....112
5. Conclusiones....114
Referencias bibliográficas....115

Acrónimos y abreviaturas

ABI - Advanced Baseline Imager
ATS - Applications Technology Satellite
CAMS - Copernicus Atmosphere Monitoring Service
CIMSS - Cooperative Institute for Meteorological Satellite Studies
CMIP - Cloud and Moisture Imagery Product
CONABIO - Comisión Nacional para el Conocimiento y Uso de la Biodiversidad
CONAFOR - Comisión Nacional Forestal
COSPAS - Cosmicheskaya Sistema Poiska Avariynyh Sudov
CSPP - Community Satellite Processing Package
DCIS - Data Collection and Interrogation Service
EOS - Earth Observing System
EPSG - European Petroleum Survey Group
EUA - Estados Unidos de America
EUMETSAT – European Organisation for the Exploitation of Meteorological Satellites
EXIS - Extreme Ultraviolet and X-ray Irradiance Sensors
GEO – Geosynchronous Earth Orbit
GLM - Geostationary Lightning Mapper
GOES - Geostationary Operational Environmental Satellite
GPS - Global Positioning System
GRB – GOES Rebroadcast
hPa - HectoPascal
IEEE - Institute of Electrical and Electronics Engineers
IG – Instituto de Geografía
IMAGER - GOES Imager
IR – Infrarred
JMA – Japan Meteorological Agency
K - Kelvin
KMA – Korea Meteorological Administration
LANCE - Land, Atmosphere Near real-time Capability for EOS
LANOT - Laboratorio Nacional de Observación de la Tierra
LEO – Low Earth Orbit
MAG – Magnetometer
MEO – Medium Earth Orbit
MIR – Middle Infrared
MODIS - Moderate Resolution Imaging Spectroradiometer
NASA - National Aeronautics and Space Administration
NDVI – Normalized Difference Vegetation Index
NetCDF - Network Common Data Form
NIR - Near InfraRed
NOAA - National Oceanic and Atmospheric Administration
NPP – National Polar-orbiting Partnership
NRT - Near Real-Time
RF – Radiofrecuencia
RGB - Red, Green, Blue
SARSAT - Search And Rescue Satellite-Aided Tracking
SEISS – Space Environment In-Situ Suite
SEM - Space Environment Monitor
SHP – Shapefile
SIG - Sistema de Información Geográfica
SMN - Servicio Meteorológico Nacional
SMS - Synchronous Meteorological Satellite
SOUNDER - GOES Sounder
SPP - Sun Pointing Platform
SUVI – Solar Ultraviolet Imager
TDRSS - Tracking and Data Relay Satellite System
TIFF - Tagged Image File Format
TIR - Thermal Infrared
TPW - Total Precipitable Water
UNAM – Universidad Nacional Autónoma de México
UTC – Universal Time Coordinated
VAS - VISSR Atmospheric Sounder
VHF - Very High Frequency
VIIRS - Visible Infrared Imaging Radiometer Suite
VISSR - Visible-Infrared Spin Scan Radiometer

Índice de figuras

Figura 1.2.1: Órbita geoestacionaria y geosíncrona, tomado de https://www.tutorialspoint.com/Geosynchronous-and-Geostationary-Satellites.....11

Figura 1.2.2: Imagen tomada desde el ATS-1 de la Tierra y la Luna juntas, tomada el 22 de Diciembre de 1966, fue la primera de su tipo. créditos: NASA https://www.nasa.gov/centers/goddard/missions/ats.html.....12

Figura 1.2.3: La primera imagen en color de la Tierra, una composición de imágenes tomadas en 1967 por el ATS-3. Creditos: NASA https://www.wikiwand.com/en/Whole_Earth_Catalog.....13

Figura 1.2.4: Ilustración de satélite ATS-6, créditos: NASA https://space.skyrocket.de/doc_sdat/ats-6.htm.....14

Figura 1.2.5: Ilustración del satélite SMS-1, créditos: NASA https://space.skyrocket.de/doc_sdat/goes-a_sms.htm.....15

Figura 1.2.6: Ilustración de GOES-16 y sus instrumentos que lleva abordo. Créditos: NASA https://www.goes-r.gov/.....17

Figura 1.3.1: Página de FIRMS de la NASA, consultado el 27/05/20 en https://firms.modaps.eosdis.nasa.gov/map.....26

Figura 1.3.2: Regiones de descarga de FIRMS, obtenido de https://firms.modaps.eosdis.nasa.gov/active_fire/#firms-shapefile.....29

Figura 1.4.1: Flota de GOES a mayo de 2020, Créditos: NASA https://www.goes-r.gov/mission/missio/n.html.....30

Figura 1.4.2: Distribución de los instrumentos de la serie GOES-R, https://www.goes-r.gov/spacesegment/instruments.html.....31

Figura 1.4.3: Escenas de escaneo de ABI, Créditos: NOAA.....35

Figura 1.4.4: Gráfico de respuesta espectral de las bandas visibles ABI (canal 1 y 2) y diagrama de transmisión atmosférica (ventanas atmosféricas). Créditos: CIMSS and ASTER spectral library and Mat Gunshor.....38

Figura 1.4.5: Gráfico de respuesta espectral de las bandas del infrarrojo cercano (NIR) ABI (canal 3, 4, 5 y 6) y diagrama de transmisión atmosférica (ventanas atmosféricas). Créditos: CIMSS and ASTER spectral library and Mat Gunshor.....39

Figura 1.4.6: Bandas espectrales visibles y NIR de ABI se muestran en áreas color azul, se encuentran debajo de *firmas espectrales* para nieve (azul), vegetación (verde), tierra (rojo) y asfalto (negro).Este diagrama explica por qué la nieve es brillante en las bandas visibles, pero oscura en la banda "Nieve / Hielo" a 1.6 µm. *Las firmas espectrales* provienen de la biblioteca espectral del radiómetro de reflexión térmica de emisión espacial avanzada (ASTER). (Schmit et al., 2018).....39

Figura 1.4.7: Espectro típico de emisión de CO_2 de un incendio, modificado de https://en.wikipedia.org/wiki/Flame_detector.....40

Figura 1.4.8: Funciones de ponderación vertical ABI IR para la atmósfera estándar de EUA que denotan la sensibilidad de cada canal en diferentes capas de la atmósfera. Unidades de presión en hectopa*scales* (hPa), 1 atmósfera (atm) = 1013.24 hPa = presión atmosférica al nivel del mar. Créditos: CMISS.....43

Figura 1.4.9: Relación entre altitud y presión atmosférica, creación propia, datos de https://www.mide.com/air-pressure-at-altitude-calculator.....44

Figura 1.4.10: Gráfico de respuesta espectral de las bandas del infrarrojo ABI (canal 7 al 16) y espectros IR de alta resolución espectral emitidos por la Tierra.....45

Figura 1.4.11: 16 bandas espectrales de ABI, 15 de enero de 2017. Las dos primeras bandas detectan lo visible, las cuatro siguientes en el NIR y las diez últimas en el IR.....46

Figura 2.1: Diagrama de flujo del script "s1_umbrales_auto.py" , especifica en que modulo se ejecuta cada proceso. Elaboración propia.....49

Figura 2.2: Diagrama de flujo del script "s2_incendios_auto.py", especifica en que modulo se ejecuta cada proceso Elaboración propia.....50

Figura 2.3: Árbol de directorios del algoritmo. Elaboración propia.....51

Figura 2.2.1: Posiciones de una matriz NumPy. Elaboración propia.....58

Figura 2.2.2: Ejemplo de kernel de 7x7 con el pixel potencial al centro.....64

Figura 3.1.1: Puntos de calor FIRMS/VIIRS originales del día 11 de mayo de 2020 y acercamiento a la zona de ejemplo (101.55°W, 101.0°W, 17.66°N, 18.06°N). Elaboración *propia con* datos de FIRMS/NASA.....74

Figura 3.1.2: FIRMS_hora_2020132 del la zona de ejemplo, los polígonos están agrupados por hora y se muestra el área en km². Creación propia.....75

Figura 3.1.3: Eventos FIRMS/VIIRS del día 11 de mayo de 2020 del recorte de ejemplo. Elaboración propia.....76

Figura 3.2.1: Banda 7 y 14 en proyección geoestacionaria, FullDisk. Elaboración propia.....80

Figura 3.2.2: Recorte de México de T3.9 y ΔT en coordenadas geográficas. Elaboración propia81

Figura 3.2.3: Umbrales óptimos del día 22 de marzo (82) al 10 de mayo (131) de 2020 a las 19:00 UTC. Creación propia.....82

Figura 3.2.4: Umbrales óptimos los 10 días anteriores al 11 de mayo, del 1ro (día 122) al 10 de mayo (día 131) de 2020 a las 19:00 UTC, la linea azul representa el umbral para el día 132. Creación propia.....82

Figura 3.2.5: Escenas descargadas del producto MCD12Q1 para generar el mosaico de la región de México. Elaboración propia con datos del USGS.....84

Figura 3.2.6: Máscaras de: A) Manchas urbanas (M_{Urb}), B) Suelo *desértico (M*Des*)* y C) Agua (M_{Agua}). Generadas a partir del producto MCD12Q1, clasificación IGBP anual. Elaboración propia.....86

Figura 3.2.7: T3.9 y ΔT reclasificadas con sus respectivos umbrales. Elaboración propia.....87

Figura 3.2.8: Acercamiento a *T*3.9 y los PP resultantes de la escena de ejemplo. Elaboración propia.....87

Figura 3.2.9: Kernel de 7x7 pixeles para un PP. Elaboración propia.....88

Figura 3.2.10: Acercamiento a los puntos de calor detectados a las 19:00 UTC el 11 de mayo de 2020. Elaboración propia.....88

Figura 4.1: Número de puntos de calor (PC) detectados del día 19 de marzo al 19 de septiembre de 2020.....92

Figura 4.2: Puntos de calor detectados por estado del 19 de marzo al 19 de septiembre de 2020 92

Figura 4.3: Puntos de calor detectados por tipo de vegetación del 19 de marzo al 19 de septiembre de 2020.....93

Figura 4.4: Promedio del umbral optimo de $T_{3.9}$ y ΔT por mes *en distintos horarios del día* durante 6 meses (marzo a septiembre de 2020). Elaboración propia.....94

Figura 4.5: Promedio del umbral optimo de $T_{3.9}$ y ΔT por hora para cada mes de estudio, (marzo a septiembre de 2020). Elaboración propia.....95

Figura 4.6: Puntos de calor detectados del día 11 de mayo de 2020 sobre 4 compuestos distintos donde se pueden observar los putos de calor.....96

Figura 4.7: Acercamiento a una serie de incendios en el estado Guerrero de los puntos de calor detectados el día 11 de mayo de 2020 sobre 4 compuestos distintos de las 19:00hrs UTC del mismo día.....97

Índice de tablas

Tabla 1.2.1: Historia y estado de todos los satélites de la familia GOES a mayo de 2020, elaboración propia, datos tomados de https://www.goes-r.gov/mission/history.html y https://www.wmo-sat.info/oscar/satellites/view/152.....19

Tabla 1.2.2: Instrumentos y sensores de cada satélite (G - GOES), creación propia, datos de https://www.wmo-sat.info/oscar/.....20

Tabla 1.2.3: Nombre completo de instrumentos y sensores de la familia GOES, creación propia, datos tomados de https://www.wmo-sat.info/oscar/.....21

Tabla 1.3.1: Atributos de un archivo shp ó txt de FIRMS convencional, datos tomados de https://earthdata.nasa.gov/earth-observation-data/near-real-time/firms.....28

Tabla 1.4.1: Instrumentos de GOES-16, datos de https://www.goes-r.gov/spacesegment/instruments.html.....32

Tabla 1.4.2: Bandas del sensor ABI, resolución espectral, espacial y radiométrica. Fuente: NASA y NOAA.....33

Tabla 1.4.3: Características y cantidades del producto CMI. Elaboración propia, datos: NASA/NOAA.....34

Tabla 1.4.4: Modos de escaneo de ABI y resolución temporal de ABI, creación propia, datos: NOAA.....35

Tabla 2.2.1: Nomenclatura NetCDF GOES-16. Nota: YYYY = Año, JJJ = Número de día del año (000 - 365), HH = Hora (00 - 23), MM = Mes (01 - 12), SS = Segundos (00 - 59), s = Mili-segundos (0-9).....56

Tabla 2.2.2: Conjuntos de datos científicos de MCD12Q1. Elaboración propia con datos del USGS.....59

Tabla 2.2.3: Leyenda y descripción de las clases del Programa Internacional de Geosfera-Biosfera, IGBP (LC_Type1). Datos: USGS.....61

Tabla 2.2.4: Atributos del producto de incendios GOES-16/ABI, datos de las ANP proporcionados por el CONANP http://sig.conanp.gob.mx/website/pagsig/info_shape.htm.....65

Tabla 3.1: Scripts y módulos ubicados en /bin.....70

Tabla 4.1: Atributos del producto de incendios GOES-16/ABI, datos de las ANP proporcionados por el CONANP.....90

Introducción

El siguiente trabajo describe la metodología para crear un algoritmo de percepción remota para detección de incendios forestales en México en tiempo casi real, utilizando imágenes satelitales GOES-16/ABI[1]. La metodología incluye la aplicación de técnicas para generar umbrales de temperatura, programación en lenguaje Python y su automatización en un servidor Debian GNU/Linux.

Las zonas forestales son imprescindibles para la vida en el planeta, entre las razones más importantes, purifican el aire que respiramos al capturar bióxido de carbono y liberar oxígeno, además de ser una fuente de materia prima en muchas actividades humanas. Sin embargo, estas zonas están siendo amenazadas por factores ajenos a las actividades forestales[2]. Una de las principales amenazas son los incendios forestales, muchos autores afirman que "El fuego es un factor ecológico natural que posibilita el rejuvenecimiento cíclico de la vegetación en muchos ecosistemas" (Guitián Rivera, 1999); no obstante, cuando se inicia por alguna actividad ó negligencia humana y no se combaten tempranamente puede llegar a tener consecuencias devastadoras para el medio ambiente, incluso para la salud y seguridad de las personas (Comisión Nacional Forestal, 2010).

Según un artículo publicado por el Centro Tecnológico Forestal de Cataluña, se entiende por *incendio forestal* a la propagación no controlada del fuego sobre la vegetación que encuentra a su paso. Aunque a menudo se usa de forma indistinta fuego e incendio, el primero es el elemento y el otro una expresión del mismo. El término de quemas se refiere al fuego controlado en zonas forestales, de pastos o agrícolas (Plana Bach et al., 2016).

1 GOES-16 es el satélite número 16 de la familia de satélites GOES (Geostationary Operational Environmental Satellite) administrados por la NASA y la NOAA, ABI (Advanced Baseline Imager) es el sensor multiespectral que porta este satélite.

2 Se entiende por actividad forestal a la actividad del sector primario que consiste en aprovechar los recursos naturales maderables y no maderables de la superficie forestal del país (INEGI, s/f)

Este trabajo se enfoca en la teledetección[3] del fuego en tiempo casi real para poder emitir una alerta temprana a las autoridades responsables de combatir los incendios forestales. Debido a ello, no se abordarán las causas que originan los incendios.

En las últimas décadas, los incendios forestales de gran extensión y duración se han convertido en un fenómeno global de gran relevancia, no solo para la conservación de las cubiertas vegetales y enclaves naturales especialmente valiosos, sino para la propia población, produciendo alarma social por afectación directa a bienes y personas, ante perdidas en ocasiones irreversibles llegando a ser fatales (Díaz Tapia, 2018)

A finales de 2016, con el lanzamiento del satélite meteorológico geoestacionario, GOES-16/ABI de la NOAA[4]/NASA, perteneciente a la serie "GOES-R"[5], se abrió una ventana de posibilidades que permitieron desarrollar nuevos algoritmos de teledetección, gracias a su mayor resolución temporal y espacial como es el caso de la detección de incendios forestales. En este contexto, es interesante explorar el potencial de las imágenes GOES-16/ABI para desarrollar e implementar un algoritmo de detección de incendios para México en tiempo casi real y mantener un monitoreo continuo del avance del fuego.

Desde el año 2017, el Laboratorio Nacional de Observación de la Tierra (LANOT) del Instituto de Geografía de la UNAM cuenta con una antena de recepción de imágenes GOES-16, Uno de los proyectos del LANOT es el desarrollo de algoritmos para la detección de puntos de calor (DPC) para México, número de convenio 52769-18297-VI-18, dirigido por la Dra. Lilia de Lourdes Manzo Delgado, en el que participan estudiantes de Ingeniería Geomática de la Facultad de Ingeniería de la UNAM.

Para el desarrollo de este algoritmo y posterior puesta en marcha en tiempo real del mismo se presentó un problema relacionado con el procesamiento de los datos, la gran cantidad de

3 Según la Agencia Espacial Europea (ESA, 2014) "La teledetección ó percepción remota es un modo de obtener información acerca de objeto o evento tomando y analizando datos sin que los instrumentos empleados para adquirir los datos estén en contacto directo con el objeto o evento de estudio".

4 National Oceanic and Atmospheric

5 La serie GOES-R es un programa de cuatro satélites (GOES-R / S / T / U) que extenderá la disponibilidad del sistema operativo de satélites GOES hasta 2036

información resultó imposible procesar con un software convencional de SIG[6] si se quiere obtener resultados en tiempo real a bajo costo. Afortunadamente en la actualidad existe el software libre y lenguajes de programación de código abierto, cuya combinación permite programar y automatizar cualquier proceso de teledetección o SIG gratis.

Esta tesis abordará el procesamiento de datos espaciales con software libre para construir el algoritmo de detección de incendios, desde la adquisición de los datos, procesamiento, visualización y automatización de datos georreferenciados en formatos ráster y vectorial. Se enfatiza y confirma que el software libre es una herramienta potente con múltiples beneficios.

Una vez que se tiene la teoría del algoritmo, se sabe qué se quiere hacer, qué hardware lo va a hacer y con qué sistema operativo; en ese momento se necesita un lenguaje para transmitir a la máquina las órdenes que se le quieren dar; esto es, el lenguaje de programación, una manera de pensar y dar órdenes a la computadora (Juganaru Mathieu, 2014).

Existe una gran variedad de lenguajes de programación y para diversos fines. Si lo que se quiere es programar utilizando datos espaciales y lenguaje científico, una de las mejores alternativas es Python. Este es un lenguaje interpretado, orientado a objetos muy poderoso, pensado y desarrollado para que su aprendizaje e implementación sea lo más sencillo posible. Por ese motivo la mayoría de software dirigido a SIG y procesamiento de datos espaciales disponible actualmente utiliza Python como lenguaje para scripting. GVSIG, QGIS o ArcGIS[7] son algunos de los ejemplos más destacados. Debido a ello, se ha convertido en una alternativa muy atractiva no solo para los desarrolladores y profesionales de programación sino también para los científicos, investigadores, artistas, y educadores (Downey et al., 2002).

6 Un sistema de información geográfica (SIG) es un sistema empleado para describir y categorizar la Tierra y otras geografías con el objetivo de mostrar y analizar la información a la que se hace referencia espacialmente.

7 GVSIG y QGIS son software para Sistemas de Información Geográfica basados en software libre y código abierto y ArcGIS es software privativo.

1. Marco teórico

1.1. Objetivos

1.1.1 Objetivo General

• Crear, implementar y automatizar un algoritmo para la detección de incendios forestales en tiempo casi real utilizando imágenes satelitales GOES-16/ABI, GNU/Linux y Python.

1.1.2 Objetivos Particulares

• Crear un manual técnico del algoritmo y los scripts de Python para el Laboratorio Nacional para la Observación de la Tierra (LANOT) del Instituto de Geografía de la UNAM, con la finalidad de que sea un elemento fundamental para adecuar y actualizar versiones posteriores del mismo.

• Crear un producto de puntos de calor en distintos formatos, que pueda ser compartido con diferentes dependencias de gobierno para que puedan tomar decisiones, análisis y emitir una alerta temprana para reducir los siniestros.

1.2. Antecedentes

Actualmente, existen productos derivados de algoritmos de percepción remota para la detección de puntos de calor asociados a incendios forestales con imágenes satelitales de resolución espacial mediana como las del sensor VIIRS[8] a bordo del satélite Suomi-NPP con una resolución espacial de 375m al nadir en las bandas que se utilizan para la detección de incendios. El satélite Suomi-NPP se encuentra en una órbita polar heliosincronica a una altitud de 824km para recorrer y observar la totalidad de la tierra cada 24hrs, pasando por un mismo punto una vez al día. La baja resolución temporal de Suomi-NPP es una limitante para mantener un seguimiento en tiempo casi real de los incendios, lo que se complica cuando el paso del satélite coincide con la presencia de condiciones atmosféricas adversas, como alta nubosidad.

8 VIIRS: Conjunto de radiómetros de imágenes infrarrojas visibles.

Es por ello que existe la necesidad de desarrollar un algoritmo de detección de puntos de calor con satélites de órbita geoestacionaria que cuentan con sensores de alta resolución temporal (cada 10 minutos) y baja resolución espacial (pixeles de 2 km), como es el caso GOES-16/ABI, que pueden ser un complemento importante de los productos existentes.

1.2.1 Incendios forestales en México

Los incendios forestales no son algo nuevo en México ni en el mundo, se calcula que hace casi 470 millones de años durante el periodo Silúrico comenzaron los incendios forestales en el planeta, esta actividad ha experimentado altas y bajas en la historia de la Tierra, principalmente relacionadas con cambios en la concentración de oxígeno atmosférico y los niveles de humedad que han caracterizado la evolución del clima (Rodríguez Trejo, 2012).

Según el Centro Nacional de Prevención de Desastres (CENAPRED), en México se tienen dos temporadas de incendios forestales, la primera corresponde a la zona centro, norte, noreste, sur y sureste del país, la cual se inicia en enero finalizando en junio, y la segunda en el noroeste del país, que se inicia en mayo y termina en septiembre, ambas coinciden con la época de mayor estiaje[9] en la República mexicana.

A nivel mundial 2019 fue un año récord de incendios forestales. Según un informe del Servicio de Monitoreo Atmosférico Copérnico (CAMS por sus siglas en inglés) en el 2019 hubo una actividad excepcional en términos de intensidad y emisiones en todo el mundo y se pronostica que esta actividad se replique en los años próximos e incluso aumente, esto es debido a que cada año la temperatura superficial en temporada de sequía es más alta, lo que aumentarán el riesgo de incendios forestales.

En México nueve de cada diez incendios forestales son causados por seres humanos, y solo el 1% corresponde a fenómenos naturales como descargas eléctricas, caídas de rayos o erupciones volcánicas. De este 90% casi la mitad se producen por actividades agropecuarias (44%); otros son ocasionados intencionalmente (19%), es decir, son quemas que se realizan en el contexto de conflictos entre personas o comunidades, tala ilegal o litigios, etcétera. De igual forma, los

9 El estiaje es el nivel de caudal mínimo que alcanzan los ríos, lagunas o el acuífero en la época de mayor calor, debido principalmente a la sequía.

incendios pueden ser provocados por fumadores que no apagan bien sus cigarros (11%) o fogatas (12%). Además, los accidentes automovilísticos, ferroviarios, aéreos y rupturas de líneas eléctricas también son algunas de las causas que pueden provocar estos fuegos no planificados (CENAPRED, 2019)

1.2.2 Historia de GOES

La era de la toma de imágenes de la Tierra desde la perspectiva de una órbita geoestacionaria comenzó el 6 de diciembre de 1966 con el lanzamiento de un sensor experimental (Cámara Spin-Scan Cloudcover) a bordo del Application Technology Satellite-1 (ATS-1) (Suomi & Parent, 1968).

Las órbitas satelitales se pueden clasificar en 3 tipos:

- LEO (Low Earth Orbit) es una órbita terrestre baja, de 500 a 900km, sus principales características son una resolución espacial mediana y una baja resolución temporal. Ejemplos: Suomi-NPP y NOAA-20
- MEO (Medium Earth Orbit) es una órbita terrestre mediana, a una altitud entre 5,000 a 12,000km, su principal uso ha sido para sistemas ó constelaciones de satélites de posicionamiento geográfico. Ejemplos: GPS, GLONASS, Galileo
- GEO (Geosynchronous Orbit) es una órbita terrestre alta, estos satélites orbitan a una altitud promedio de 35,000km y parecen permanecen fijos, se mueven acompañando a la Tierra en su movimiento. Su principal objetivo es la observación meteorológica y tiene la característica de tener una alta resolución temporal y baja resolución espacial. Ejemplos: GOES, Meteosat.

La órbita geoestacionaria es un tipo de órbita GEO sobre el plano del ecuador, a diferencia de la órbita geosíncrona que puede tener cualquier inclinación (Figura 1.2.1).

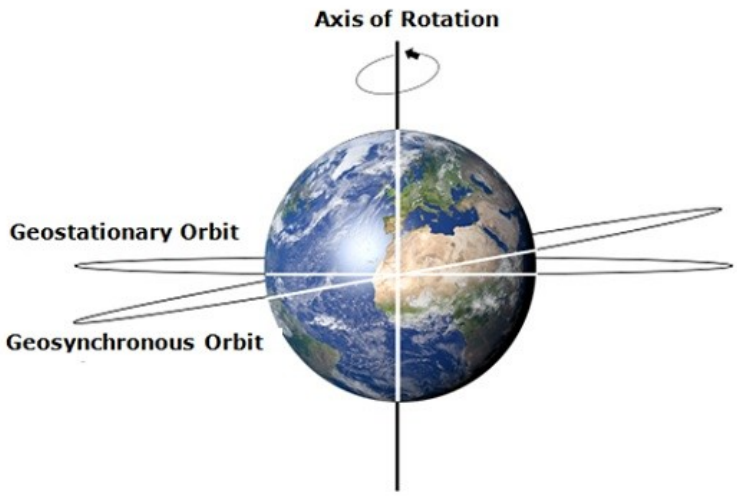

Figura 1.2.1: Órbita geoestacionaria y geosíncrona, tomado de https://www.tutorialspoint.com/Geosynchronous-and-Geostationary-Satellites

Seis satélites ATS fueron lanzados entre 1966 y 1974. Durante sus 18 años de vida el ATS-1 examinó las técnicas de estabilización de espín, investigó el entorno geoestacionario y realizó varios experimentos de comunicación. Su experimento VHF[10] probó la capacidad de actuar como un enlace entre las estaciones terrestres y los aviones, demostró la recopilación de datos meteorológicos de terminales remotas y evaluó la viabilidad del uso de señales VHF para la navegación. También transmitió programas educativos y proporcionó servicios de salud, investigación y comunitarios a los Estados Unidos y varios lugares del Pacífico. La misión también proporcionó las primeras imágenes de la capa de nubes de la Tierra completa (Neuman Ezell, 1988)

ATS-1 llevaba una cámara meteorológica en blanco y negro que transmitía las primeras imágenes de la Tierra de disco completo desde una órbita geosíncrona. El 22 de diciembre de 1966, ATS-1 capturó la primera imagen de la Tierra y la luna juntas (Figura 1.2.2), una hazaña que a menudo se atribuye erróneamente a la Voyager 1[11]. (La Voyager 1 capturó la primera

10 Frecuencia VHF en inglés Very High Frequency es la banda del espectro electromagnético para el rango de ondas electromagnéticas (ondas de radio) de radiofrecuencia de 30 a 300MHz.

11 La Voyager 1 es una sonda espacial robótica lanzada el 5 de septiembre de 1977.

imagen de un solo cuadro que mostraba toda la Tierra y la luna). El ATS-1 estaba a 35888 kilómetros de la Tierra y a más de 430000km de la luna cuando se tomó la foto.

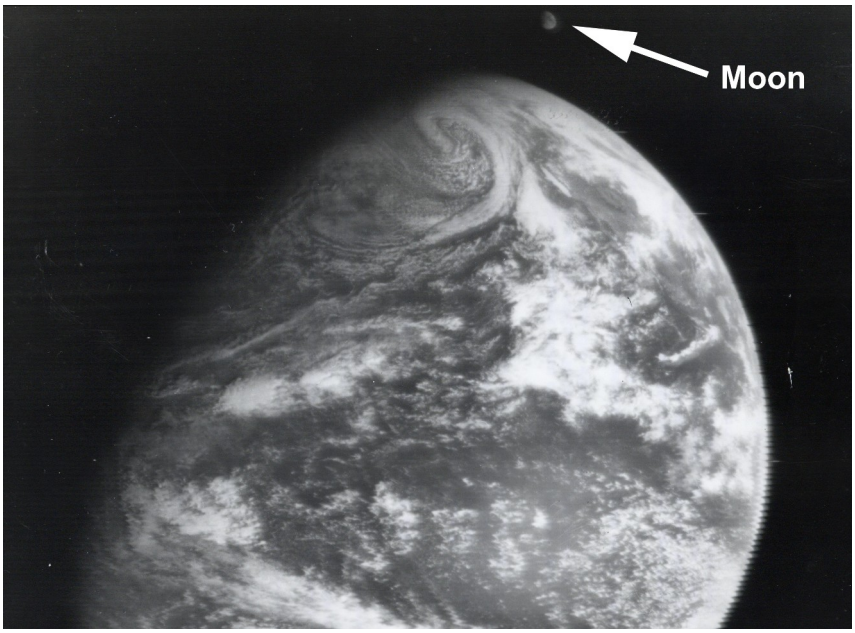

Figura 1.2.2: Imagen tomada desde el ATS-1 de la Tierra y la Luna juntas, tomada el 22 de Diciembre de 1966, fue la primera de su tipo. créditos: NASA
https://www.nasa.gov/centers/goddard/missions/ats.html

El 1967 durante su lanzamiento, el ATS-2 presentó una falla en el vehículo y puso al satélite en una órbita indeseable. Aunque el satélite permaneció funcional, el ATS-2 se desactivó después de seis meses debido a la limitada cantidad de datos útiles que podía recopilar. En ese mismo año se lanzó el ATS-3 con objetivos muy parecidos al ATS-1. Proporcionó un servicio de comunicaciones a sitios en la cuenca del Pacífico y la Antártida además de enlaces de comunicaciones de emergencia durante el terremoto en México de 1985 y el desastre de St. Helens. Apoyó el aterrizaje en la luna del Apolo con transmisiones de TV en tiempo real para su transmisión por Radio Caracas en Venezuela (Neuman Ezell, 1988). El satélite también proporcionó imágenes regulares de la cubierta de nubes para estudios meteorológicos y la primera imagen en color de toda la Tierra (Figura 1.2.3).

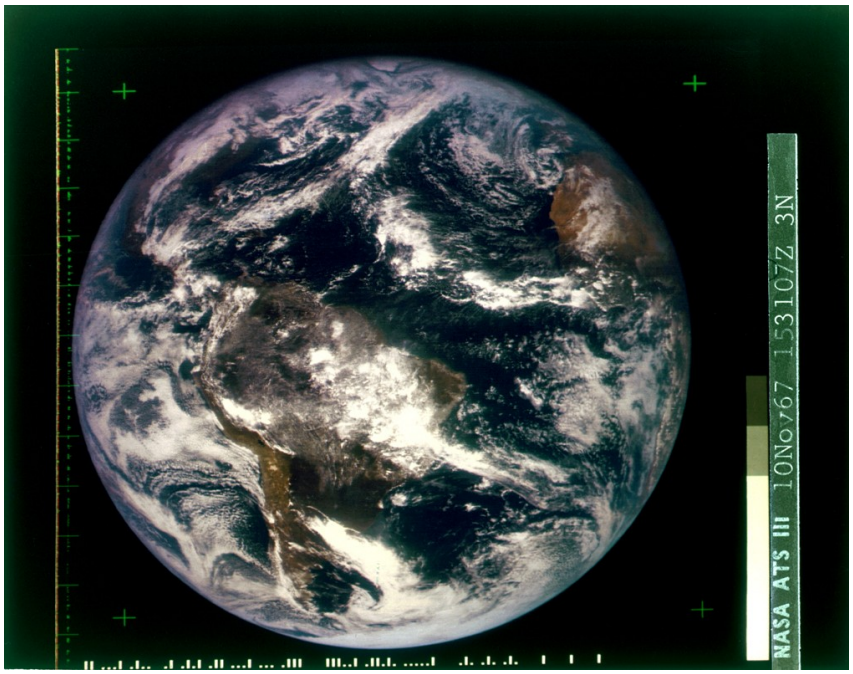

Figura 1.2.3: La primera imagen en color de la Tierra, una composición de imágenes tomadas en 1967 por el ATS-3. Creditos: NASA https://www.wikiwand.com/en/Whole_Earth_Catalog

El ATS-4 se lanzó el 10 de agosto de 1968, pero una falla en el vehículo de lanzamiento lo dejó en una órbita mucho más baja de lo planeado, haciendo que el satélite fuera casi inútil. La órbita baja y la resistencia atmosférica causaron que ATS-4 volviera a entrar en la atmósfera de la Tierra y se desintegrara el 17 de octubre de 1968. Un año después se lanzó el ATS-5; sin embargo, entró en un giro no planificado y comenzó a girar sobre el eje correcto, pero en la dirección opuesta. Como resultado, los brazos de gradiente de gravedad de la nave espacial no pudieron desplegarse y finalmente fue impulsada por encima de la órbita geoestacionaria (Neuman Ezell, 1988).

Finalmente, el ATS-6 fue lanzado el 30 de mayo de 1974. El ATS-6 era un satélite mucho más pesado y sofisticado que sus predecesores (Figura 1.2.4). Durante la misión Apollo-Soyuz[12], el satélite ATS-6 rastreó y transmitió la telemetría de la nave espacial Apollo al centro de control de Houston durante el 55 por ciento de su órbita y también proporcionó el enlace de TV desde el

12 La misión Apolo-Soyuz en 1975 fue la última del Programa Apolo

muelle Apollo-Soyuz al centro de control de Houston y a los medios de transmisión comerciales. Durante su vida de 5 años, ATS-6 fue pionero en la televisión de transmisión directa, retransmitió programación educativa a India, Estados Unidos y otros países. El vehículo también realizó pruebas de control de tráfico aéreo y practicó técnicas de búsqueda y rescate asistidas por satélite. Llevaba un radiómetro experimental que posteriormente se convirtió en un instrumento estándar a bordo de satélites meteorológicos. La nave espacial estuvo operativa hasta agosto de 1979 (Neuman Ezell, 1988).

Figura 1.2.4: Ilustración de satélite ATS-6, créditos: NASA https://space.skyrocket.de/doc_sdat/ats-6.htm

El éxito de los satélites ATS dio origen a la primera serie de satélites meteorológicos geoestacionarios SMS[13]. El SMS-1 sé lanzo el 17 de mayo de 1974 y fue el primer satélite operativo capaz de detectar condiciones meteorológicas desde una ubicación fija. Un año después sé lanzo el SMS-2. Llevaban un radiómetro de barrido por infrarrojo visible (VISSR), un monitor de entorno espacial (SEM) y un sistema de recopilación de datos (DCIS).

13 Synchronous Meteorological Satellite

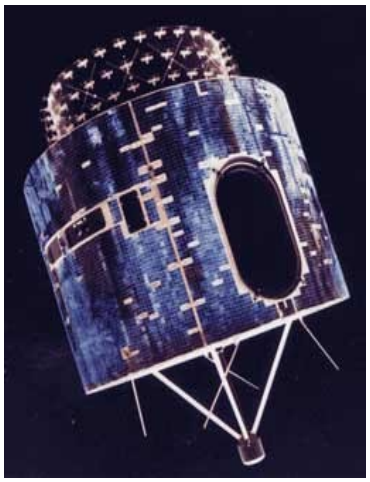

Figura 1.2.5: Ilustración del satélite SMS-1, créditos: NASA https://space.skyrocket.de/doc_sdat/goes-a_sms.htm

Después del exitoso lanzamiento de dos satélites SMS experimentales el programa "Geostationary Operational Environmental Satellite" (GOES) comenzó formalmente en 1975 como un esfuerzo conjunto de la NOAA y la NASA.

El 16 de octubre de 1975, se lanzó el primer satélite del programa GOES, GOES-A y pasó a llamarse GOES-1 una vez que llegó a la órbita, también fue llamado SMS-C por su similitud con la serie de satélites SMS (Jenner, 2017). La serie de satélites GOES se nombran con una letra conforme al alfabeto y una vez alcanzada su órbita final se les renombran con un número. GOES-B y GOES-C siguieron en 1977 y 1978. Del GOES-1 a GOES-3 fueron casi idénticos al diseño de los satélites SMS, estabilizados por rotación y transportando el VISSR, SEM y DCIS. También se les conoció como la primera generación de GOES.

GOES 4-7 fue la segunda generación y fueron diseñados de manera similar a los primeros tres satélites con algunas diferencias en sus instrumentos. El GOES-4 se lanzó el 9 de septiembre de 1980 y fue el primer satélite en llevar una Sonda Atmosférica (VAS[14]) permitiendo la medición de temperatura y humedad. Los datos derivados de este instrumento permitieron a los científicos determinar las altitudes y temperaturas de las nubes y dibujar una imagen tridimensional de su

14 Atmospheric Sounder

distribución en la atmósfera, lo que condujo a predicciones meteorológicas más precisas (Neuman Ezell, 1988). El 3 de mayo de 1986, GOES-G que habría sido GOES-7 se perdió cuando su vehículo de lanzamiento fue alcanzado por un rayo poco después del despegue. Fue así como la nomenclatura se recorrió y GOES-H se convirtió en el GOES-7.

Con el lanzamiento del GOES-8 el 13 de abril de 1994 y hasta el GOES-12 se introdujo la serie GOES-IM. Los nuevos instrumentos IMAGER y SOUNDER eran instrumentos separados y operaban de manera independiente, lo que permitía a los satélites obtener continuamente datos de imagen y sonido en lugar de alternar entre los dos modos de operación y el sistema de búsqueda y rescate de GOES comenzó a funcionar (Jenner, 2017). El 23 de julio de 2001, se lanzó GOES-M, el último satélite de la serie GOES-IM, fue el primer satélite en llevar un sofisticado telescopio de rayos X solares (SXI[15]).

La siguiente generación fue la “GOES-N Series” consistió de 3 satélites, comenzó con el lanzamiento del GOES-13 en 2006, proporcionó una vigilancia constante de los "disparadores" atmosféricos para condiciones climáticas severas como tornados, inundaciones repentinas, tormentas de granizo y huracanes (NOAA et al., s/f). En enero de 2018 el satélite se trasladó a una longitud de 60°W en donde se encuentra en modo back-up y entraría en funcionamiento en caso de que GOES-16 presentara un fallo. El GOES-14 se lanzó el 27 de junio de 2009 y el GOES-15 el 4 de marzo de 2010, ambos son prácticamente idénticos al GOES-13, las operaciones del GOES-15 finalizaron el 2 de marzo de 2020

En diciembre de 2008 la NASA y la NOAA anunciaron que estaba en marcha la construcción de la próxima generación de satélites GOES, nombrada “GOES-R Series”. El proyecto consiste de dos naves espaciales, GOES-R (16) y GOES-S (17) y dos naves espaciales adicionales (GOES-T y GOES-U). GOES-16 sé lanzo el 19 de noviembre de 2016, después de una fase de prueba en servicio en 89.5°W, el satélite se trasladó a 75.2°W sustituyendo al GOES-13. Se declaró operativo el 18 de diciembre de 2017 y está programado para estar operativo hasta 2032.

Los nuevos sensores que porta esta nueva generación (Figura 1.2.6) ve a la Tierra con tres veces más canales espectrales, cuatro veces la resolución y un escaneo cinco veces más rápido

15 Solar X-ray Imager

que el GOES de la generación anterior. Proporciona parámetros y datos clave de nubes, aerosoles, humedad, vientos, radiación solar, océanos, temperatura superficial, tormentas eléctricas, etc. (Dirk Krebs, 2020)

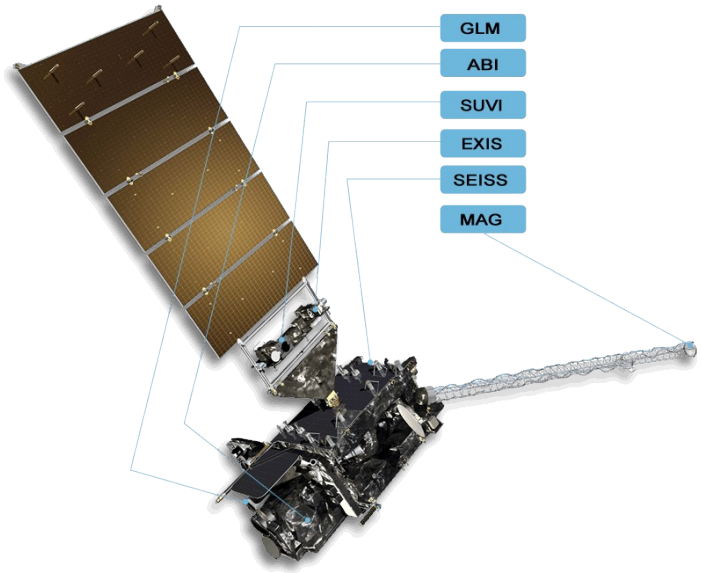

Figura 1.2.6: Ilustración de GOES-16 y sus instrumentos que lleva abordo. Créditos: NASA https://www.goes-r.gov/

El GOES-S, ahora GOES-17, se lanzó el 1 de marzo de 2018 desde el Space Launch Complex en la Estación de la Fuerza Aérea de Cabo Cañaveral, Florida, a bordo de un cohete Atlas V . El GOES-17 reemplazó al GOES-15 como el satélite operacional GOES West de NOAA el 12 de febrero de 2019, a 137,2 grados de longitud oeste sobre el Océano Pacífico.

GOES-16 y 17 son los que a mayo de 2021 están cubriendo la necesidad de los satélites meteorológicos geoestacionarios del continente americano en las longitudes 75.2°W y 137.2°W respectivamente, por lo que también se llaman GOES-East para el GOES-16 y GOES-West el 17. Para otras regiones del mundo existe otros programas de satélites meteorológicos geoestacionarios, el satélite Meteosat de la Unión Europea posicionado a 0° de longitud y Himawari de Japón que órbita a 140.7°E, entre otros.

Finalmente, el lanzamiento del GOES-T, originalmente programado para 2020, se ha retrasado debido a una anomalía del sistema de enfriamiento del sensor ABI. NOAA está

implementando cambios en el radiador ABI para GOES-T y GOES-U para reducir el riesgo de que vuelva a ocurrir una anomalía en el sistema de enfriamiento, por lo tanto, GOES-T ahora está programado para lanzarse en diciembre de 2021 y el lanzamiento de GOES-U está programado para 2024.

Nombre Lanzamiento	Nombre operacional	Nombre completo	Lanzamiento	Fin de vida (esperado)	Altitud	Longitud	Estado
ATS-A	ATS-1	Application Technology Satellite - 1	06/12/1966	01/12/1978	35,786 km	150°W	Desmantelado
ATS-B	ATS-2	Application Technology Satellite - 2	06/04/1967	N/A	N/A	N/A	No llego a orbita
ATS-C	ATS-3	Application Technology Satellite - 3	06/11/1967	01/12/1978	35786 km	45°W	Desmantelado
ATS-D	ATS-4	Application Technology Satellite - 4	10/08/1968	N/A	N/A	N/A	No llego a orbita
ATS-E	ATS-5	Application Technology Satellite - 5	12/08/1969	N/A	N/A	N/A	No llego a orbita
ATS-F	ATS-6	Application Technology Satellite - 6	30/04/1974	03/08/1979	35786 km	94°W	Desmantelado
SMS-A	SMS-1	Synchronous Meteorological Satellite - 1	17/05/1974	21/01/1981	35786 km	75°W	Desmantelado
SMS-B	SMS-2	Synchronous Meteorological Satellite - 2	06/02/1975	05/08/1982	35787 km	135°W	Desmantelado
GOES-A	GOES-1	Geostationary Operational Environmental Satellite - 1	16/10/1975	07/03/1985	35786 km	135°W	Desmantelado
GOES-B	GOES-2	Geostationary Operational Environmental Satellite - 2	16/06/1977	01/07/1993	35786 km	75°W	Desmantelado
GOES-C	GOES-3	Geostationary Operational Environmental Satellite - 3	16/06/1978	01/07/1993	35786 km	135°W	Desmantelado
GOES-D	GOES-4	Geostationary Operational Environmental Satellite - 4	09/09/1980	22/11/1988	35786 km	135°W	Desmantelado
GOES-E	GOES-5	Geostationary Operational Environmental Satellite - 5	22/05/1981	18/07/1990	35786 km	75°W	Desmantelado
GOES-F	GOES-6	Geostationary Operational Environmental Satellite - 6	28/04/1983	01/07/1989	35786 km	135°W	Desmantelado
GOES-G	N/A	Geostationary Operational Environmental Satellite	03/05/1986	N/A	N/A	N/A	No llego a orbita
GOES-H	GOES-7	Geostationary Operational Environmental Satellite - 7	26/02/1987	11/01/1996	35786 km	75°W	Desmantelado
GOES-I	GOES-8	Geostationary Operational Environmental Satellite - 8	13/04/1994	05/05/2004	35786 km	75°W	Desmantelado
GOES-J	GOES-9	Geostationary Operational Environmental Satellite - 9	23/05/1995	22/05/2003	35786 km	135°W	Desmantelado
GOES-K	GOES-10	Geostationary Operational Environmental Satellite - 10	25/04/1997	01/12/2006	35786 km	135°W	Desmantelado
GOES-L	GOES-11	Geostationary Operational Environmental Satellite - 11	03/05/2000	05/12/2011	35786 km	135°W	Desmantelado
GOES-M	GOES-12	Geostationary Operational Environmental Satellite - 12	23/07/2001	10/05/2010	35786 km	75°W	Desmantelado
GOES-N	GOES-13	Geostationary Operational Environmental Satellite - 13	24/05/2006	≥2020	35786 km	60°W	Inactivo / Back-up
GOES-O	GOES-14	Geostationary Operational Environmental Satellite - 14	27/06/2009	≥2020	35786 km	105°W	Stand-by
GOES-P	GOES-15	Geostationary Operational Environmental Satellite - 15	04/03/2010	≥2020	35786 km	128°W	Operacional
GOES-R	GOES-16	Geostationary Operational Environmental Satellite - 16	19/11/2016	≥2032	35786 km	75.2°W	Operacional
GOES-S	GOES-17	Geostationary Operational Environmental Satellite - 17	01/03/2018	≥2029	35786 km	137.2°W	Operacional

Tabla 1.2.1: Historia y estado de todos los satélites de la familia GOES a mayo de 2021, elaboración propia, datos tomados de https://www.goes-r.gov/mission/history.html y https://www.wmo-sat.info/oscar/satellites/view/152

	ATS			SMS		G-1er Gen			G-2da Gen				G-IM					G-N Series			G-R	
	ATS-1	ATS-3	ATS-6	SMS-1	SMS-2	GOES-1	GOES-2	GOES-3	GOES-4	GOES-5	GOES-6	GOES-7	GOES-8	GOES-9	GOES-10	GOES-11	GOES-12	GOES-13	GOES-14	GOES-15	GOES-16	GOES-17
SSCC																						
MSSCC																						
VHRR																						
VISSR																						
SEM/EPS																						
SEM/HEPAD																						
SEM/XRS-EUV																						
DCIS																						
SEM/MAG																						
VAS																						
IMAGER																						
SOUNDER																						
GEOS&R																						
SXI																						
ABI																						
EXIS																						
GLM																						
SUVI																						
SEISS/MPS																						
SEISS/EHIS																						
SEISS/SGPS																						

Tabla 1.2.2: Instrumentos y sensores de cada satélite (G - GOES), creación propia, datos de https://www.wmo-sat.info/oscar/

Instrumento	Nombre Completo
SSCC	Cámara Spin Scan Cloud
MSSCC	Cámara Cloud Cloud Spin Scan multicolor
VHRR	Very High Resolution Radiometer
SEM/MAG	SEM / Magnetometer
VISSR	Visible-Infrared Spin Scan Radiometer
SEM/EPS	SEM / Energetic Particles Sensor
SEM/HEPAD	SEM / High Energy Proton and Alpha Particles Detector
SEM/XRS-EUV	SEM / X-Ray Sensor - Extreme Ultra-Violet Sensor
DCIS	Data Collection and Interrogation Service
VAS	VISSR Atmospheric Sounder
GEOS&R	Geostationary Search and Rescue
IMAGER	GOES Imager
SOUNDER	GOES Sounder
SXI	Solar X-ray Imager
ABI	Advanced Baseline Imager
EXIS	Extreme Ultraviolet Sensor / X-Ray Sensor Irradiance Sensors
GLM	Geostationary Lightning Mapper
SUVI	Solar Ultraviolet Imager
SEISS/MPS	SEISS / Magnetospheric Particle Sensor
SEISS/EHIS	SEISS / Energetic Heavy Ion Sensor
SEISS/SGPS	SEISS / Solar and Galactic Proton Sensor

Tabla 1.2.3: Nombre completo de instrumentos y sensores de la familia GOES, creación propia, datos tomados de https://www.wmo-sat.info/oscar/

1.2.3 Software libre y Código Abierto

Citando una de las definiciones más formales de “software” atribuida a la IEEE (Institute of Electrical and Electronics Engineers), “Software es la suma total de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de cómputo” (IEEE, 1993). El software es la parte intangible de la computadora, es decir, programas, aplicaciones, entre otros

El movimiento del software libre nace como una necesidad al software privativo que dominaba en 1980, los ideales del software privativo establecieron la siguiente norma: «Si compartes con tu vecino, te conviertes en un pirata. Si quieres hacer algún cambio, tendrás que rogárnoslo» (Stallman, 1992). Es por lo que en 1984 Richard Stallman dio comienzo al proyecto

GNU. El nombre «GNU» es un acrónimo recursivo de «GNU No es Unix[16]». Es el programa que asigna los recursos de la máquina y se comunica con el hardware (gnu.org, s/f). El objetivo de Stallman era crear un sistema operativo completamente libre. El sistema operativo es el programa que interacciona entre el hardware, el usuario y las aplicaciones. Pero aún necesitaba un núcleo.

Fue en 1991 cuando un estudiante finlandés de 21 años llamado Linus Torvalds publicó en la red su propio núcleo de sistema operativo inspirado en Unix que más tarde se denominaría Linux. Este era precisamente el corazón que le faltaba al sistema GNU que Stallman estaba diseñando. Así, se adaptaron las herramientas de GNU al núcleo de Linux y crearon las primeras distribuciones GNU/Linux (Culebro Juárez et al., 2006). Es por eso que la forma correcta de hacer referencia a este sistema operativo es por su nombre completo GNU/Linux y no solamente Linux.

Debian es una de las distribuciones que se utilizan con mayor frecuencia en servidores cuya misión es crítica por su estabilidad y rendimiento. Ian Murdock en 1993 fue quién inició el proyecto Debian e inicialmente estaba patrocinado por la Free Software Foundation[17]. Es quizás la distribución que mejor ha sabido mantener a lo largo del tiempo la filosofía del proyecto inicial de GNU/Linux (Sánchez Gonzáles, 2009).

Un programa es software libre para el usuario siempre que, como usuario tengas las cuatro libertades (Stallman, 1992):

- Libertad 0: la libertad para ejecutar el programa sea cual sea nuestro propósito.
- Libertad 1: la libertad para estudiar el funcionamiento del programa y adaptarlo a las necesidades del usuario (el acceso al código fuente es condición indispensable para esto).
- Libertad 2: la libertad para redistribuir copias y ayudar así a tu vecino.

16 Es un sistema operativo de código cerrado desarrollado en la década de 1970 por AT&T y General Electric, en Estados Unidos.

17 La Free Software Foundation o Fundación por el Software Libre es una organización creada en octubre de 1985 por Richard Stallman y otros entusiastas del software libre con el propósito de difundir este movimiento.

- Libertad 3: la libertad para mejorar el programa y luego publicarlo para el bien de toda la comunidad —el acceso al código fuente es condición indispensable para esto.

El término *"free software"* en el mundo anglófono creaba una situación incómoda debido a la doble acepción que en ingles tiene el termino free (que puede significar gratuito o libre), aunque Stallman siempre aclaró que se refería a *"free"* en términos de libertad y no de gratuidad, las empresas no se sentían cómodas utilizando la palabra free pues temían que los usuarios creyeran que el recurso o software era totalmente gratuito al no conocer la filosofía de este movimiento.

Es por eso que en 1998, Eric S. Raymond, Bruce Perens y otros hackers involucrados en el desarrollo de software libre lanzaron la Open Software Initiative y propusieron el uso de término open source (código abierto) en contraposición al término free software (software libre) como término más atractivo al entorno empresarial y para poner énfasis en el valor diferencial de que el código fuente de su software está disponible (Culebro Juárez et al., 2006).

Para que las libertades 2 y 4 de Stallman tengan sentido se debe disponer del código fuente del programa. Por consiguiente, la accesibilidad del código fuente es una condición necesaria para el software libre (Stallman, 1992). Es por eso que todo el software libre es de código abierto, pero no a la inversa, el software de código abierto no es necesariamente software libre

Bruce Perens, de la Open Source Iniciative creó una lista de 10 condiciones que debe cumplir un programa para poder ser considerado Open Source (opensource.org, s/f)

1. Redistribución gratuita. La licencia no restringirá a ninguna de las partes a vender o regalar el software como un componente de una distribución agregada de software que contiene programas de varias fuentes diferentes. La licencia no requerirá una regalía u otra tarifa por dicha venta.
2. Código fuente. El programa debe incluir el código fuente y debe permitir la distribución tanto en el código fuente como en el formulario compilado. Cuando alguna forma de un producto no se distribuye con el código fuente, debe haber un medio bien publicitado para obtener el código fuente por un costo de reproducción razonable, preferiblemente descargando a través de Internet sin cargo. El código fuente debe ser la forma preferida

en la que un programador modificaría el programa. El código fuente deliberadamente ofuscado no está permitido.

3. Obras derivadas. La licencia debe permitir modificaciones y trabajos derivados, y debe permitir que se distribuyan bajo los mismos términos que la licencia del software original.
4. Integridad del código fuente del autor. La licencia puede restringir la distribución del código fuente en forma modificada solo si la licencia permite la distribución de "archivos de parche" con el código fuente con el fin de modificar el programa en el momento de la compilación. La licencia debe permitir explícitamente la distribución de software creado a partir de código fuente modificado. La licencia puede requerir trabajos derivados para llevar un nombre o número de versión diferente del software original.
5. No discriminación contra personas o grupos. La licencia no debe discriminar a ninguna persona o grupo de personas.
6. No discriminación contra los campos de trabajo. La licencia no debe restringir a nadie el uso del programa en un campo específico de esfuerzo. Por ejemplo, no puede restringir que el programa se use en un negocio o que se use para investigación genética.
7. Distribución de licencia. Los derechos adjuntos al programa deben aplicarse a todos aquellos a quienes el programa se redistribuye sin la necesidad de que esas partes ejecuten una licencia adicional.
8. La licencia no debe ser específica para un producto. Los derechos adjuntos al programa no deben depender de que el programa forme parte de una distribución de software en particular. Si el programa se extrae de esa distribución y se usa o distribuye dentro de los términos de la licencia del programa, todas las partes a las que se redistribuye el programa deben tener los mismos derechos que los que se otorgan junto con la distribución del software original.
9. La licencia no debe restringir otro software. La licencia no debe imponer restricciones a otro software que se distribuya junto con el software con licencia. Por ejemplo, la licencia

no debe insistir en que todos los demás programas distribuidos en el mismo medio deben ser software de código abierto.

10. La licencia debe ser neutral en tecnología. Ninguna disposición de la licencia puede basarse en ninguna tecnología o estilo de interfaz individual.

En resumen, las expresiones «software libre» y «código abierto» se refieren casi al mismo conjunto de programas. No obstante, dicen cosas muy diferentes acerca de dichos programas, basándose en valores diferentes. El movimiento del software libre defiende la libertad de los usuarios de ordenadores, en un movimiento en pro de la libertad y la justicia. Por otro lado, la idea del código abierto valora principalmente las ventajas prácticas y no defiende principios (gnu.org, s/f).

1.3. FIRMS

Información sobre incendios para el sistema de gestión de recursos (FIRMS por sus siglas en inglés) es un sistema de detección de puntos de calor desarrollado por la NASA y distribuidos por LANCE[18], un programa que distribuye datos de incendios activos en tiempo casi real (NRT[19] por sus siglas en inglés), 3 horas posteriores a la observación satelital de forma gratuita y en formato vectorial. Está disponible para dos grupos de satélites con sensores de resolución moderada, Aqua/Terra con el sensor MODIS generan el producto con bandas de 1km de resolución espacial y Suomi-NPP/NOAA-20 con el sensor VIIRS generan el producto con las bandas de 375m, la resolución espacial mejorada de los datos de 375m proporciona una mayor respuesta de incendios en áreas relativamente pequeñas y ha mejorado el rendimiento nocturno con respecto a los FIRMS de MODIS (NASA, 2020). Estos datos son publicados en la página https://firms.modaps.eosdis.nasa.gov/ (Figura 1.3.1)

18 Land, Atmosphere Near real-time Capability for EOS (LANCE) es un programa de la NASA para que los datos del Sistema de Observación de la Tierra (EOS) de algunos instrumentos, incluyendo VIIRS, estén disponibles dentro de las tres horas posteriores a la observación satelital.

19 Near Real Time

Figura 1.3.1: Página de FIRMS de la NASA, consultado el 27/05/20 en https://firms.modaps.eosdis.nasa.gov/map

Hay dos productos derivados de VIIRS, el "VIIRS (S-NPP) I Band 375m Active Fire Product NRT" y "VIIRS (NOAA-20 / JPSS-1) I Band 375m Active Fire Product NRT", el primero, al depender del satélite Suomi-NPP que fue lanzado el 28 de octubre de 2011 cuenta con datos históricos de puntos de calor desde el 20 de enero de 2012 hasta el presente. A diferencia del satélite NOAA-20 que fue lanzado el 18 de noviembre de 2017 y el producto FIRMS se comenzó a generar el primero de enero de 2020. Para el algoritmo creado se ocupó el producto derivado de VIIRS/Suomi-NPP (FIRMS/VIIRS) para la generación de los umbrales debido a la existencia de los datos históricos de este a diferencia del otro, aunque los resultados actuales son muy parecidos

Características generales de este producto:

- **Nombre corto del producto:** VNP14IMGTDL_NRT
- **Título del producto:** VIIRS (S-NPP) I Band 375m Active Fire Product NRT (Vector data)
- **Enlaces para descargar datos:** https://earthdata.nasa.gov/active-fire-data

- **Nombre largo del producto:** VIIRS (S-NPP) I Band 375m Active Fire locations NRT (Vector data) distributed by LANCE FIRMS
- **Formatos de descarga disponibles:** TXT, SHP, KML.
- **Proyección:** Coordenadas geográficas, Datum WGS84, EPSG[20]: 4326
- **Atributos:** (ver Tabla 1.3.1)

Atributo	Descripción
LATITUDE	Latitud del centro del píxel de fuego nominal de 375m
LONGITUDE	Longitud del centro del píxel de fuego nominal de 375m
BRIGHT_TI4	Temperatura de brillo del canal VIIRS I4 del píxel de fuego medido en Kelvin.
SCAN	El algoritmo se produce con un tamaño aproximado de pixel de 375m en el nadir. El valor Scan representa la resolución espacial en la dirección Este-Oeste real del pixel.
TRACK	El algoritmo se produce con un tamaño aproximado de pixel de 375m en el nadir. El valor Track representa la resolución espacial Norte-Sur real del pixel.
ACQ_DATE	Fecha de adquisición de la imagen VIIRS.
ACQ_TIME	Hora de adquisición / paso superior del satélite (en UTC).
SATELLITE	N = Suomi National Polar-orbiting Partnership (Suomi NPP) 1 = NOAA-20 (designado JPSS-1 antes del lanzamiento)
CONFIDENCE	Su objetivo es ayudar a los usuarios a medir la calidad de los píxeles de punto de fuego individuales. Los valores de confianza se establecen en bajo, nominal y alto. *Los píxeles de fuego durante el día de baja confianza generalmente están asociados con áreas de destello solar y una anomalía de temperatura relativa más baja (<15K) en el canal de infrarrojo medio I4. *Los píxeles de confianza nominales son aquellos libres de posible contaminación por destello solar durante el día y marcados por una fuerte anomalía de temperatura (> 15K) en los datos diurnos o nocturnos. *Los píxeles de alta confianza están asociados con píxeles saturados diurnos o nocturnos.
VERSION	La versión identifica la colección y la fuente del procesamiento de datos: casi en tiempo real (sufijo NRT agregado a la colección) o procesamiento estándar (solo colección). "1.0NRT" - Procesamiento NRT de la Colección 1. "1.0" - Colección 1 Procesamiento estándar
BRIGHT_TI5	Temperatura de brillo del canal VIIRS I5 del píxel de fuego medido en Kelvin.
FRP	FRP representa la potencia radiativa de fuego integrada en píxeles en MW (megavatios).
DAYNIGHT	D = fuego diurno N = fuego nocturno

Tabla 1.3.1: Atributos de un archivo shp ó txt de FIRMS/VIIRS convencional, datos tomados de https://earthdata.nasa.gov/earth-observation-data/near-real-time/firms

20 European Petroleum Survey Group (EPSG) es una organización que elaboró un repositorio de parámetros geodésicos EPSG, una base de datos que contiene información, a nivel mundial, sobre los sistemas de referencia de coordenadas y proyecciones cartográficas.

La descarga de los datos puede ser de todo el mundo ó solamente una región como se observa en la (Figura 1.3.2), como la región de “Central America” cubre todo México es la que ocuparemos.

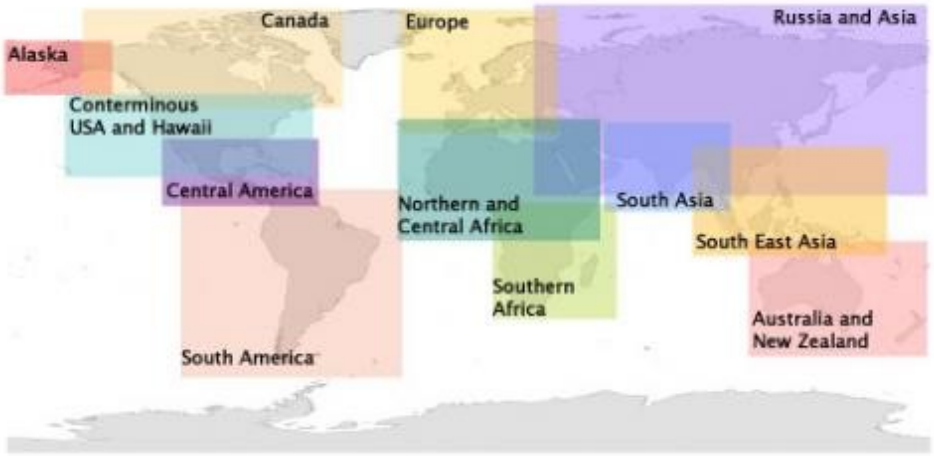

Figura 1.3.2: Regiones de descarga de FIRMS, obtenido de https://firms.modaps.eosdis.nasa.gov/active_fire/#firms-shapefile

1.4. GOES 16

GOES 16 pertenece a la serie GOES-R, que porta un conjunto de seis instrumentos científicos en la órbita geoestacionaria en cuatro naves espaciales idénticas, dos de ellas ya se encuentran operativas. El GOES 16 East y 17 West (Figura 1.4.1)

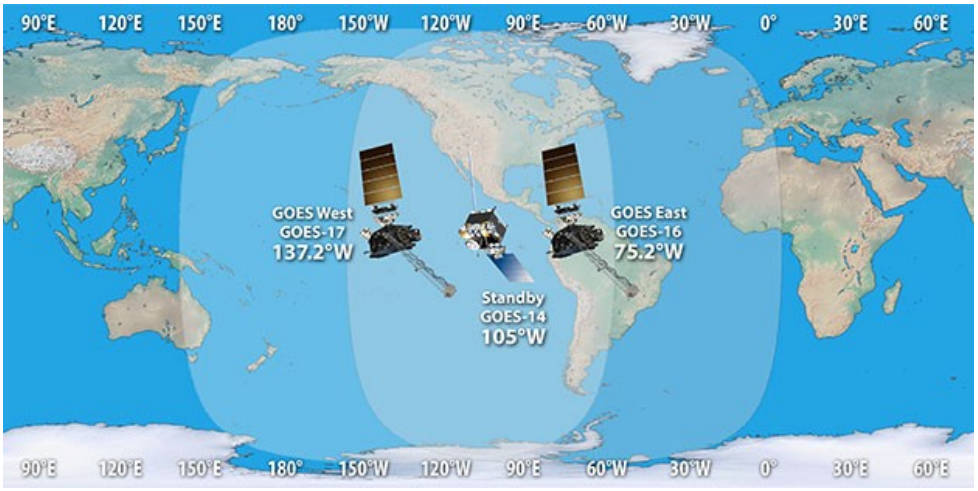

Figura 1.4.1: Flota de GOES a mayo de 2020, Créditos: NASA https://www.goes-r.gov/mission/missio/n.html

Los 6 instrumentos que porta la serie R se pueden clasificar en tres tipos, dependiendo a donde estén apuntando y cual sea su objetivo (Figura 1.4.2)

- NADIR-POINTING: Siempre están apuntando al nadir de la tierra, montados en una plataforma de precisión altamente estable y aislados dinámicamente del resto de la nave espacial. ABI y GLM
- SOLAR-POINTING: Montados en la plataforma Sun Pointing Platform (SPP) que rastrea el movimiento estacional y diario del sol en relación con la nave espacial. EXIS y SUVI.
- IN-SITU: Estudian el entorno espacial, proporcionan mediciones de partículas y campos en órbita geosíncrona. SEISS y MAG.

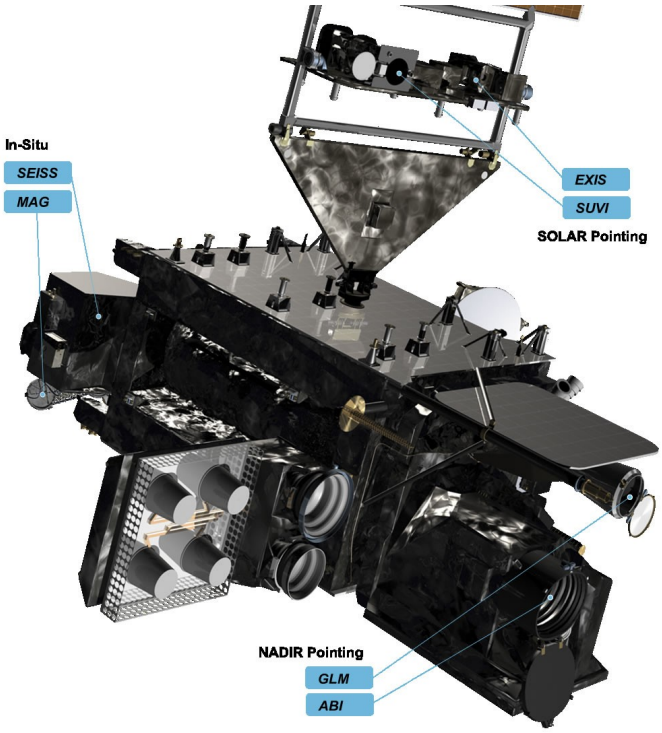

Figura 1.4.2: Distribución de los instrumentos de la serie GOES-R, https://www.goes-r.gov/spacesegment/instruments.html

En la (Tabla 1.4.1) se hace referencia a los 6 instrumentos que porta el satélite GOES-16 así como una breve descripción de los mismos, las imágenes utilizadas en este trabajo son obtenidas del sensor multiespectral ABI.

Clasificación	Instrumento	Nombre Completo	Descripción
NADIR POINTING	ABI	Advanced Baseline Imager	El sensor ABI es un radiómetro multiespectral de 16 bandas de última generación, con bandas espectrales que cubren las porciones visibles, infrarrojas cercanas e infrarrojo del espectro electromagnético.
	GLM	Geostationary Lightning Mapper	Es un detector transitorio óptico de infrarrojo cercano de un solo canal que puede detectar los cambios momentáneos en una escena óptica, lo que indica la presencia de rayos. Es el primer mapeador de rayos operacional en órbita geoestacionaria
SOLAR POINTING	SUVI	Solar Ultraviolet Imager	Es un telescopio que monitorea el sol en el rango de longitud de onda ultravioleta. Observa y caracteriza regiones activas complejas del sol, erupciones solares y las erupciones de filamentos solares que pueden dar lugar a eyecciones de masa coronal.
	EXIS	Extreme Ultraviolet and X-ray Irradiance Sensors	Es un sensor de irradiancia ultravioleta extrema y de rayos X. Es capaz de detectar erupciones solares que podrían interrumpir las comunicaciones y reducir la precisión de navegación, afectando a los satélites, las líneas aéreas de gran altitud y las redes eléctricas en la Tierra.
IN-SITU	MAG	Magnetometer	El magnetómetro proporciona mediciones del campo magnético del entorno espacial que controla la dinámica de las partículas cargadas en la región exterior de la magnetosfera. Estas partículas pueden ser peligrosas para las naves espaciales y los vuelos espaciales humanos.
	SEISS	Space Environment In-Situ Suite	Este instrumento está compuesto por cuatro sensores que monitorean los flujos de protones, electrones y iones pesados en la magnetosfera con el fin de evaluar el riesgo de descarga electrostática y el peligro de radiación para los astronautas y los satélites.

Tabla 1.4.1: Instrumentos de GOES-16, datos de https://www.goes-r.gov/spacesegment/instruments.html

La misión del sensor ABI es medir la energía solar radiante y reflectante de la Tierra en sus 16 bandas del espectro electromagnético que abarcan desde el visible hasta el infrarrojo térmico (Tabla 1.4.2) con una resolución espacial y espectral moderada y una alta resolución temporal y radiométrica (Schmit et al., 2016). Estas 16 bandas de última generación se utilizan para muchas aplicaciones relacionadas con el clima severo, los ciclones tropicales y los huracanes, la aviación, los riesgos de origen natural, las superficies terrestres y oceánicas y la criosfera.

Número de banda	Longitud de onda central (μm)	FWHM* al 50% mínimo	FWHM* al 50% máximo	Resolución espacial al nadir [km]	Tipo	Nombre	Resolución radiométrica [bits]
1	0.47	0.45	0.49	1	Visible	Azul	12
2	0.64	0.6	0.68	0.5	Visible	Roja	12
3	0.87	0.847	0.882	1	Infrarrojo cercano	Veggie	12
4	1.38	1.366	1.38	2	Infrarrojo cercano	Cirrus	12
5	1.61	1.59	1.63	1	Infrarrojo cercano	Nieve/Hielo	12
6	2.25	2.22	2.27	2	Infrarrojo cercano	Tamaño de las partículas de la nube	12
7	3.89	3.8	3.99	2	Infrarrojo térmico	Ventana de onda corta	14
8	6.17	5.79	6.59	2	Infrarrojo térmico	Vapor de agua de nivel superior	12
9	6.93	6.72	7.14	2	Infrarrojo térmico	Vapor de agua de nivel medio	12
10	7.34	7.24	7.43	2	Infrarrojo térmico	Vapor de agua de nivel inferior	12
11	8.44	8.23	8.66	2	Infrarrojo térmico	Fase superior de la nube	12
12	9.61	9.42	9.8	2	Infrarrojo térmico	Ozono	12
13	10.33	10.18	10.48	2	Infrarrojo térmico	Ventana de onda larga "limpia"	12
14	11.19	10.82	11.6	2	Infrarrojo térmico	Ventana de onda larga	12
15	12.27	11.83	12.75	2	Infrarrojo térmico	Ventana de onda larga "sucia"	12
16	13.27	12.99	13.56	2	Infrarrojo térmico	Onda larga de CO2	12

Tabla 1.4.2: Bandas del sensor ABI, resolución espectral, espacial y radiométrica. Fuente: NASA y NOAA

Como se observa en la Tabla 1.4.2 y en la Figura 1.4.4 el sensor ABI no cuenta con una banda en la región verde del visible cuya principal función es la generación de imágenes en color verdadero. Por lo tanto, es necesario simular la banda verde para generar imágenes de color real de manera efectiva (Schmit et al., 2016).

Después de una corrección radiométrica y geométrica de las bandas del sensor ABI por parte de las instituciones correspondientes (NOAA y NASA), los datos son puestos a disposición del público en un nivel de procesamiento “Level 1b (L1b)” donde la información de los pixeles ya está en unidades de radiancia, de estos datos se obtiene una serie de productos “Level 2 (L2)”, el primero de ellos es el producto “Cloud and Moisture Imagery Product (CMIP)”, donde se generan las 16 bandas en unidades de reflectancia para las bandas (1-6) y temperatura de brillo para las bandas (7-16).

Número de banda	Longitud de onda central	Valor de relleno	Rango válido (en unidades de cantidad física)	
			Mínimo	Máximo
1	0.47	65535	0	1.3
2	0.64	65535	0	1.3
3	0.87	65535	0	1.3
4	1.38	65535	0	1.3
5	1.61	65535	0	1.3
6	2.25	65535	0	1.3
7	3.89	65535	197.31	411.86
8	6.17	65535	138.05	311.06
9	6.93	65535	137.7	311.08
10	7.34	65535	126.91	331.2
11	8.44	65535	127.69	341.3
12	9.61	65535	117.49	311.06
13	10.33	65535	89.62	341.27
14	11.19	65535	96.19	341.28
15	12.27	65535	97.38	341.28
16	13.27	65535	92.7	318.26

Tabla 1.4.3: Características y cantidades del producto CMI. Elaboración propia, datos: NASA/NOAA

ABI escanea la tierra en 3 distintas escenas[21] que son: FullDisk, CONUS (Contiguous U.S.) y MESO (Mesoescala) (ver Figura 1.4.3) y los operadores también pueden definir escenas personalizadas, que se pueden subir en cualquier momento durante la misión.

21 La escena es la porción de la superficie terrestre observada por el satélite.

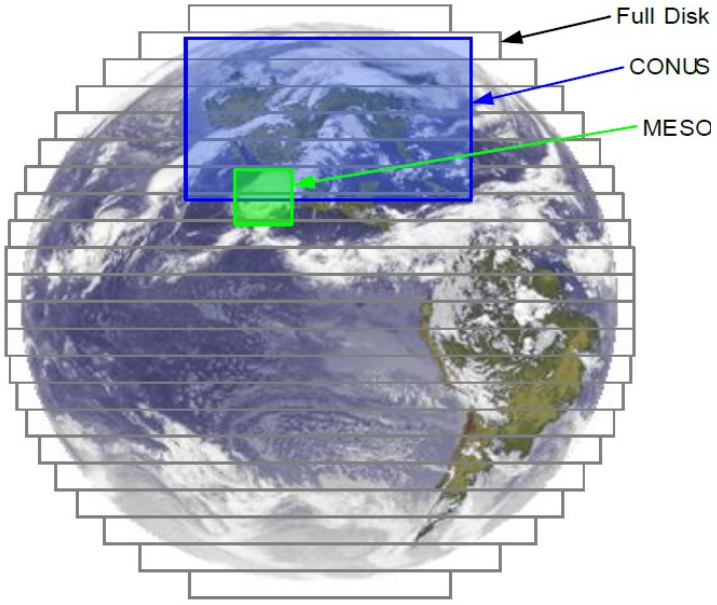

Figura 1.4.3: Escenas de escaneo de ABI, Créditos: NOAA

La resolución temporal de cada una de las tres escenas depende del modo de escaneo. ABI tiene tres modos de escaneo como se muestra en la Tabla 1.4.4

Modo de escaneo	Escena	Resolución temporal
Modo 3	FullDisk	15 min
	CONUS	5 min
	Mesoescala	30 seg
Modo 4 ó continuo	FullDisk	5 min
	CONUS	N/A
	Mesoescala	N/A
Modo 6 ó flexible	FullDisk	10 min
	CONUS	5 min
	Mesoescala	30 seg

Tabla 1.4.4: Modos de escaneo de ABI y resolución temporal de ABI, creación propia, datos: NOAA

Desde el 2 de abril de 2019, a las 16:00 UTC, el sensor ABI de GOES-16 comenzó a funcionar en el modo flexible de 10 minutos para FullDisk (Modo 6) ver Tabla 1.4.4. Hay una serie de ventajas para las imágenes de disco completo de 10 minutos. Permite que NOAA

coincida con la cadencia de exploración de disco completo los otros satélites geoestacionarios que observan al mundo desde otra latitud, la Agencia Meteorológica de Japón (JMA por sus siglas en inglés), la Administración Meteorológica de Corea (KMA por sus siglas en inglés) y la próxima generación de satélites geoestacionarios de la Organización Europea para la Explotación de Satélites Meteorológicos (EUMETSAT por sus siglas en inglés). Esto permite una cobertura avanzada casi global cada 10 minutos (NOAA & NASA, s/f)

Las 16 bandas en L1b y del producto CMIP sirven a distintos objetivos y a partir de ellas se generan múltiples productos nombrados (L2+). Cada banda tiene un uso específico según la región del espectro donde se encuentre. A continuación una descripción y usos de cada banda.

La banda visible de 0.47μm, o "Azul" sirve particularmente para monitorear aerosoles, proporciona observaciones diurnas de polvo, neblina, humo y nubes (Figura 1.4.11). Es más sensible a los aerosoles porque esa longitud de onda está en una parte del espectro electromagnético donde la dispersión atmosférica de Rayleigh[22] en cielo despejado es más frecuente. Las señales de humo y polvo en esta banda son más evidentes cuando el sol está bajo en el cielo, como el amanecer y el atardecer (Yung, 2003). Hay una transmitancia disminuida (dispersión incrementada) a longitudes de onda visibles más cortas (Figura 1.4.4). Esto es importante porque los filamentos delgados de humo, particularmente los aerosoles pequeños podrían no ser detectables a partir de bandas de longitud de onda más largas (Schmit et al., 2018). Las columnas de humo de incendios forestales se observan claramente en esta banda si el cielo está despejado.

La banda visible "Roja" en 0,64μm tiene la mejor resolución espacial (0,5 km al nadir) de todas las bandas ABI. Las nubes y el polvo espeso se ven similares a la banda 1. Más allá de los aerosoles, las bandas visibles tienen multitud de aplicaciones. Son ideales para identificar características a pequeña escala, como la niebla del río, los bordes de la niebla, o las nubes de tipo cúmulos durante el día. La banda de 64μm también se ha utilizado durante el día para

22 La dispersión atmosférica de Rayleigh ocurre cuando la radiación interactúa con moléculas y partículas en la atmósfera que tienen un diámetro menor que la longitud de onda de la radiación entrante, es la responsable del color azul del cielo.

monitorear la capa de nieve y hielo, ayudar a detectar cenizas volcánicas y analizar huracanes y tormentas de invierno (Schmit et al., 2018).

La banda 3 de 0,86μm o “Veggie” es muy sensible a la vegetación (Figura 1.4.6) y detecta nubes diurnas, niebla y algunos aerosoles. La vegetación, en general es más reflectante (es decir, más brillante) en esta banda que en las bandas visibles. Esto puede hacer que sea más difícil discernir entre nubes o aerosoles y la superficie terrestre subyacente (Figura 1.4.11). Esta banda tiene el apodo de "Veggie" porque es sensible a los cambios en la vegetación. También se utiliza para calcular el índice de vegetación de diferencia normalizada (NDVI por sus siglas en ingles). La banda de 0,86μm detecta energía en una parte del espectro electromagnético donde la vegetación es más reflectante que la tierra, como se muestra en la Figura 1.4.6, por lo tanto, las cicatrices de quemaduras también son perceptibles en la banda "Veggie" debido al contraste de reflectancia de la tierra con la vegetación. El conocimiento de dónde existen las cicatrices de quemaduras ayuda a determinar cómo se puede propagar un incendio y finalmente, esta banda es esencial para simular una banda "verde" que se necesita para una imagen en color natural (Liew, 2001)

La banda 4 que se encuentra en 1,37μm longitud de onda central, es única entre las bandas reflectantes en ABI porque como se muestra en la Figura 1.4.5 se encuentra en una región del espectro electromagnético donde la energía reflejada de la tierra es absorbida por la atmósfera (ver Figura 1.4.11), principalmente por el vapor de agua al igual que con las bandas 8,9 y 10 (ver Tabla 1.4.2). Esta banda puede detectar nubes cirrus muy delgados durante el día y estelas de vapor. Las nubes a niveles bajos también son evidentes en esta banda si la atmósfera es adecuadamente seca. Esta banda puede detectar características altamente reflectantes, como polvo o nubes, si hay poco vapor de agua sobre ellas, aunque su reflectancia probablemente sea menor que en las bandas visibles o "Veggie". La teoría sugiere que aproximadamente 0.5 pulgadas (12mm) de agua precipitable total (TPW, por sus siglas en ingles) es suficiente para absorber la mayor parte de la radiación solar a 1,37μm (Sieglaff & Schmit, 2003). Cantidades variables de humedad y su distribución vertical influyen en qué tan cerca de la superficie el

satélite puede observar características en esta longitud de onda. Por lo tanto, al ser las nubes cirrus las de mayor altitud se identifican fácilmente como se observa en la Figura 1.4.11.

El rango espectral de la banda "Nieve/Hielo" alrededor 1.61µm aprovecha la diferencia entre los componentes de refracción del agua y el hielo que controlan reflectancia de las nubes según su composición. Las nubes de agua líquida son altamente reflectantes y, por lo tanto, brillantes en esta banda, mientras que las nubes de hielo y la nieve son más oscuras porque el hielo absorbe, en lugar de reflejar la radiación en 1,61µm. En consecuencia, podría ser que los cirrus sean más oscuros, en comparación con las nubes cúmulos y estratos a base de agua más reflectantes y, en consecuencia, más claros (Schmit et al., 2018).

La banda 6 "Tamaño de partícula de la nube" también se puede utilizar para determinar el producto derivado de la fase de la nube, pero por la menor resolución espacial de la banda (ver Tabla 1.4.2) los analistas usan la banda "Nieve/Hielo". La banda de 2.2µm se usa en una gran cantidad de compuestos RGB, como los compuestos de temperatura de fuego (Schmit et al., 2018).

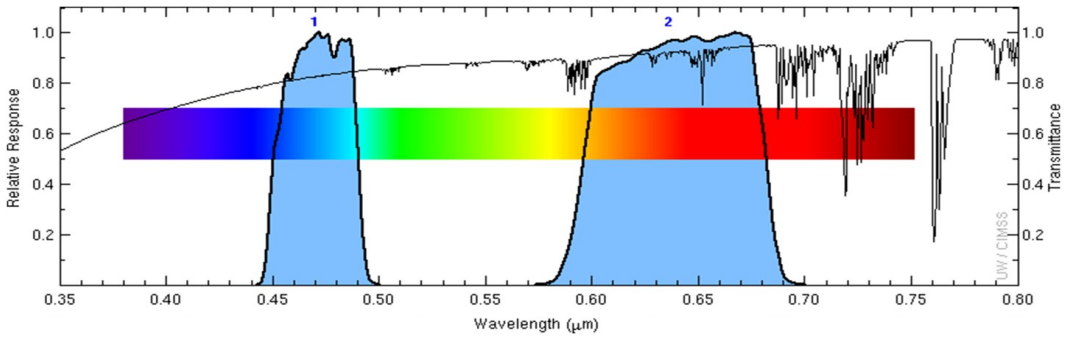

Figura 1.4.4: Gráfico de respuesta espectral de las bandas visibles ABI (canal 1 y 2) y diagrama de transmisión atmosférica (ventanas atmosféricas). Créditos: CIMSS and ASTER spectral library and Mat Gunshor

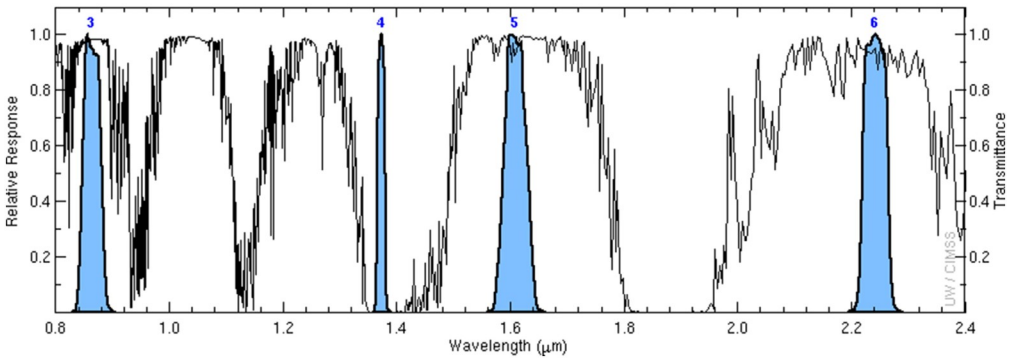

Figura 1.4.5: Gráfico de respuesta espectral de las bandas del infrarrojo cercano (NIR) ABI (canal 3, 4, 5 y 6) y diagrama de transmisión atmosférica (ventanas atmosféricas). Créditos: CIMSS and ASTER spectral library and Mat Gunshor

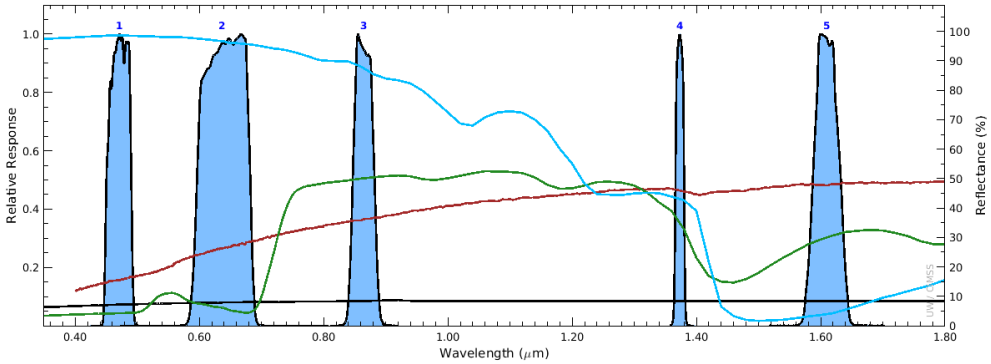

Figura 1.4.6: Bandas espectrales visibles y NIR de ABI se muestran en áreas color azul, se encuentran debajo de firmas espectrales para nieve (azul), vegetación (verde), tierra (rojo) y asfalto (negro).Este diagrama explica por qué la nieve es brillante en las bandas visibles, pero oscura en la banda "Nieve / Hielo" a 1.6 μm. Las firmas espectrales provienen de la biblioteca espectral del radiómetro de reflexión térmica de emisión espacial avanzada (ASTER). (Schmit et al., 2018)

La banda 7 de 3.9μm detecta la radiación IR terrestre emitida por la Tierra como la radiación solar reflejada durante el día, lo cual es evidente al comparar la imagen de la banda 7 (3.9μm) con las otras bandas del infrarrojo (bandas 8 a 16) en la Figura 1.4.11, su longitud de onda más corta es más sensible a la temperatura que las bandas del infrarrojo de longitud de onda más larga y en particular es más sensible a los incendios forestales, ya que como se muestra en la Figura 1.4.7 hay un pico de emisión de CO_2 alrededor de 4.3μm durante los incendios, también se utiliza

para identificar la niebla y las nubes bajas en la noche, localizar islas de calor urbano, detectar cenizas volcánicas, estimar las temperaturas de la superficie, distinguir entre masas de aire y discriminar entre tamaños de cristales de hielo durante el día. Durante la noche, la información de la temperatura es bastante confiable, pero durante el día incluye reflejos solares principalmente en suelo desnudo y en algunos tipos de nubes, las nubes de agua y las nubes altas con cristales de hielo muy pequeños reflejan la radiación solar y, por lo tanto, se representan en valores de temperatura de brillo cálidos, los cristales de hielo grandes no reflejan bien la radiación solar de 3.9µm y por lo tanto muestran temperaturas de brillo más frías. La necesidad de detectar tanto los incendios calientes y distinguir las características de las nubes frías requiere que cada pixel tenga la capacidad de capturar un rango más amplio de valores potenciales, por esta razón, las imágenes de la banda de 3.9µm tienen una resolución radiométrica de 14 bits, lo que permite 16384 valores discretos por pixel con un valor máximo de 411K, o 138°C, este es el rango más grande de todas las bandas del ABI (Schmit et al., 2018).

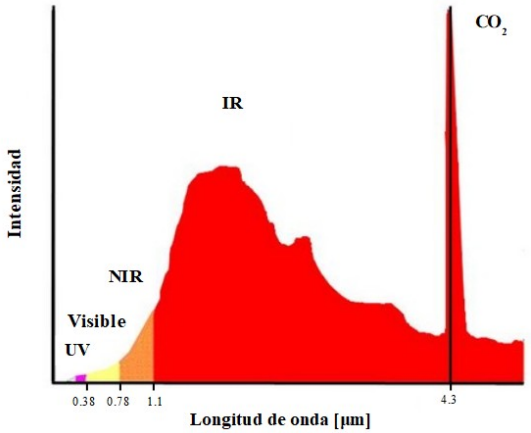

Figura 1.4.7: Espectro típico de emisión de CO_2 de un incendio, modificado de https://en.wikipedia.org/wiki/Flame_detector

El sensor ABI tiene tres bandas de "Vapor de agua" 8, 9 y 10 centradas en 6.2, 7 y 7.3µm, en general tienen usos muy parecidos con algunas diferencias, la principal es la altitud a la que se desempeñan que depende directamente de la presión atmosférica. La banda 8 "Vapor de agua de nivel superior" se usa para rastrear vientos de la troposfera superior, identificar corrientes de

chorro, pronosticar el seguimiento de huracanes y el movimiento de tormentas, estimar la humedad de nivel superior y medio e identificar regiones donde existe el potencial de turbulencia. Todo esto a presiones incluso inferiores a 300hPa (ver Figura 1.4.8) correspondiente a más de 8km de altitud (ver Figura 1.4.9), recordando que a mayor altitud menor presión atmosférica (Schmit et al., 2018).

La banda 9 de "Vapor de agua de nivel medio" de 6.9μm generalmente contiene valores de temperatura de brillo más cálidos que la banda centrada en 6.2μm, tiene muchas de las mismas aplicaciones potenciales que las otras dos bandas. En algunas masas de aire tropicales, la banda 9 puede detectar temperaturas características del medio ambiente a presiones inferiores (alturas superiores) a 500hPa (ver Figura 1.4.8 y Figura 1.4.9). En otras atmósferas secas o frías puede detectar a presiones más altas que 500hPa (Schmit et al., 2018).

La banda 10 de 7.3μm "Vapor de agua de nivel inferior" incluye algunas aplicaciones adicionales a las otras dos bandas de vapor de agua, en esta banda se destacan las columnas volcánicas que son ricas en dióxido de azufre (SO_2). Las cimas de las montañas a veces son evidentes en esta banda espectral, principalmente en atmósferas de invierno particularmente secas (Schmit et al., 2018).

En la banda 11 de 8,5μm hay poca absorción atmosférica en cielos claros, a menos que haya SO_2 de una erupción volcánica en la atmósfera donde sirve como complemento a la banda 10. El conocimiento de la emisividad es importante en la interpretación de esta banda. Las diferencias en la emisividad de la superficie a 8,5μm ocurren sobre diferentes tipos de suelo, afectando la temperatura del brillo percibido, además se usa para crear productos que determinan el tipo de nube de acuerdo a la fase de sus topes, como productos que clasifican las nubes y otros productos que determinan la microfísica de las nubes (NOAA & NASA, 2017)

En la banda 12 de longitud de onda 9.6μm gran parte de la radiación es absorbida por el ozono (O_3) presente en la atmósfera (90% en la estratosfera), de ahí se deriva el nombre dado a esta banda, además de tener una leve sensibilidad al vapor de agua. Con la banda 12 se pueden obtener algunos productos relacionados con el O_3 como la cantidad total de ozono en la columna atmosférica (NOAA & NASA, 2017).

La banda 13 de "Ventana de onda larga limpia" de 10.3µm es menos sensible que otras bandas de ventana del infrarrojo (bandas 14 y 15) a la absorción de vapor de agua, por lo tanto, mejora las correcciones de humedad atmosférica, la identificación y clasificación de las nubes y otras características atmosféricas, las estimaciones de la temperatura de la cima de la nube y el tamaño de las partículas de la nube (Schmit et al., 2018). A causa de encontrarse en una ventana atmosférica se pueden generar productos para la estimación de la temperatura de la superficie continental y la temperatura superficial del mar.

Por la sensibilidad de cada banda del infrarrojo en diferentes capas de la atmósfera (Figura 1.4.8) y los espectros del infrarrojo emitidos por la Tierra (Figura 1.4.10), podemos concluir que la cantidad de radiación en los espectros emitidos por la tierra son mayores en las bandas que son más sensibles a capas elevadas de la atmósfera porque incluyen la radiación de la Tierra y nubes altas, que reflejan o emiten energía (bandas 8,9,10,12 y 16)

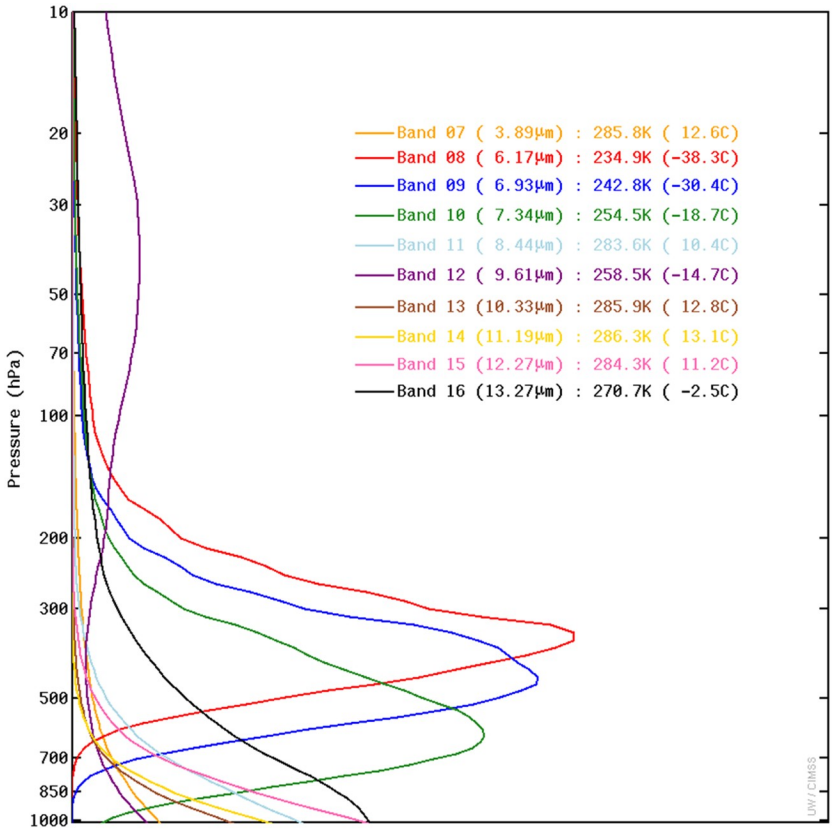

Figura 1.4.8: Funciones de ponderación vertical ABI IR para la atmósfera estándar de EUA que denotan la sensibilidad de cada canal en diferentes capas de la atmósfera. Unidades de presión en hectopascales (hPa), 1 atmósfera (atm) = 1013.24 hPa = presión atmosférica al nivel del mar. Créditos: CMISS

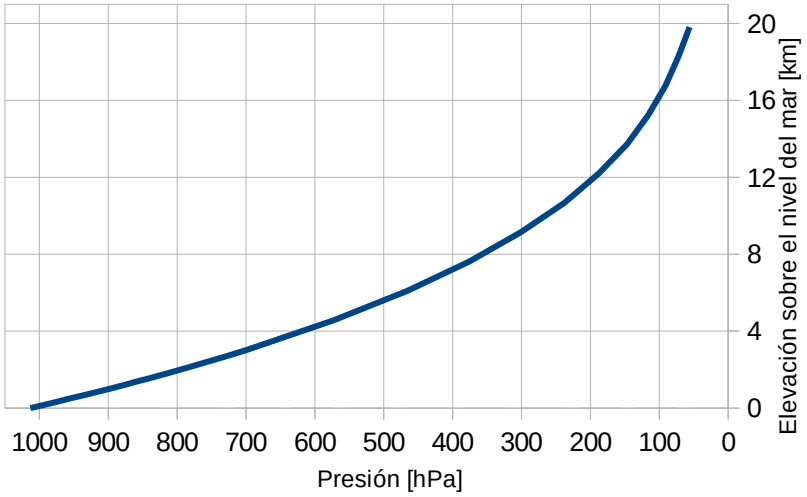

Figura 1.4.9: Relación entre altitud y presión atmosférica, creación propia, datos de https://www.mide.com/air-pressure-at-altitude-calculator

La banda 14 de 11.2µm es la tradicional "Ventana de onda larga"; sin embargo, hay más absorción de energía por el vapor de agua a esta longitud de onda en comparación con la banda 13, por ese motivo es mejor usar la banda de 10.3µm para la mayoría de las aplicaciones operativas, porque los valores de temperatura de brillo estarán más cerca de las temperaturas de las características de la superficie. Los valores en la banda de 11.2µm serán más fríos que los de la banda de 10.3µm para un mismo pixel en función de la cantidad de humedad en la atmósfera (Schmit et al., 2018).

La absorción y reemisión de vapor de agua, particularmente en la troposfera inferior, enfría ligeramente la mayoría de los valores de temperatura de brillo, incluso aunque no tengan nubes. Esto es lo que pasa con la banda 15 "Ventana de onda larga sucia" de 12,3µm, reflejará áreas de vapor de agua en áreas libres de nubes y es parte de muchos productos de línea de base, incluida la máscara de cielo despejado, las propiedades de las nubes superiores y las cenizas volcánicas. El apodo de esta banda es "Dirty Longwave Window" precisamente porque el vapor de agua enfría las temperaturas de brillo y, por lo tanto, "ensucia" la imagen (Schmit et al., 2018).

La banda 16 de 13.3µm es útil para algoritmos científicos y productos derivados que requieren delinear la tropopausa, estimar las alturas de las nubes, entre otros. A pesar de su

importancia en los productos, generalmente no se usa para la interpretación visual de los fenómenos climáticos. En esta banda, la superficie de la Tierra es evidente en cielos despejados, pero con un fuerte enfriamiento por CO_2, significa que las temperaturas de brillo en 13.3µm son más frías que en las bandas de ventana de infrarrojo tradicionales, excepto donde hay nubes en la tropopausa (Schmit et al., 2018).

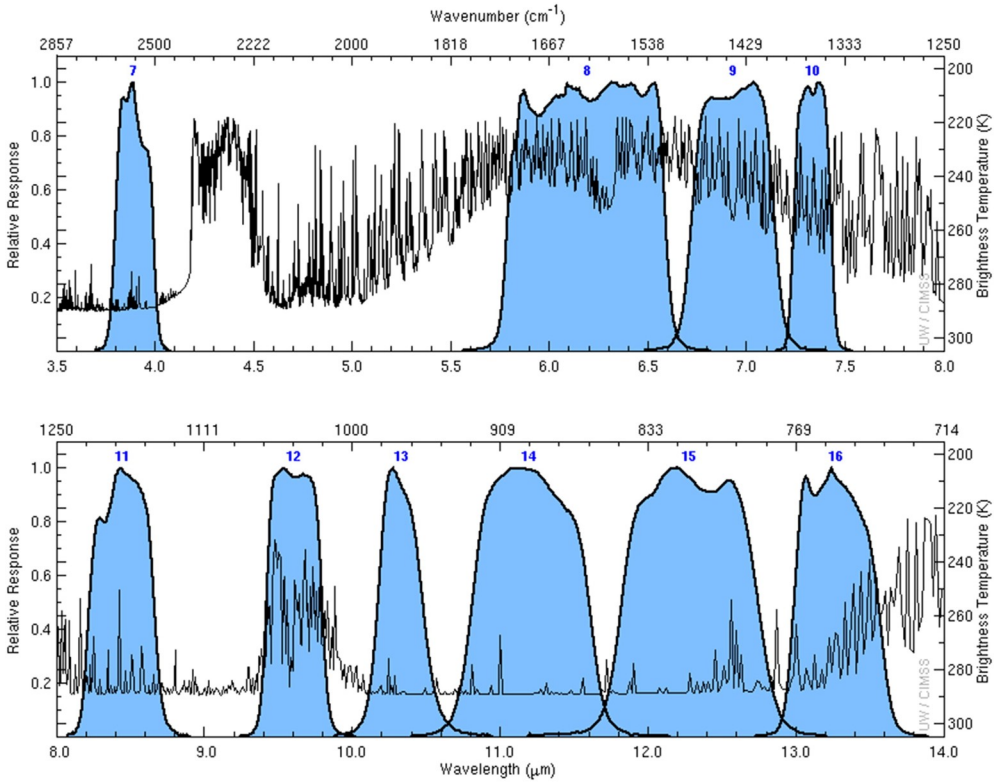

Figura 1.4.10: Gráfico de respuesta espectral de las bandas del infrarrojo ABI (canal 7 al 16) y espectros IR de alta resolución espectral emitidos por la Tierra

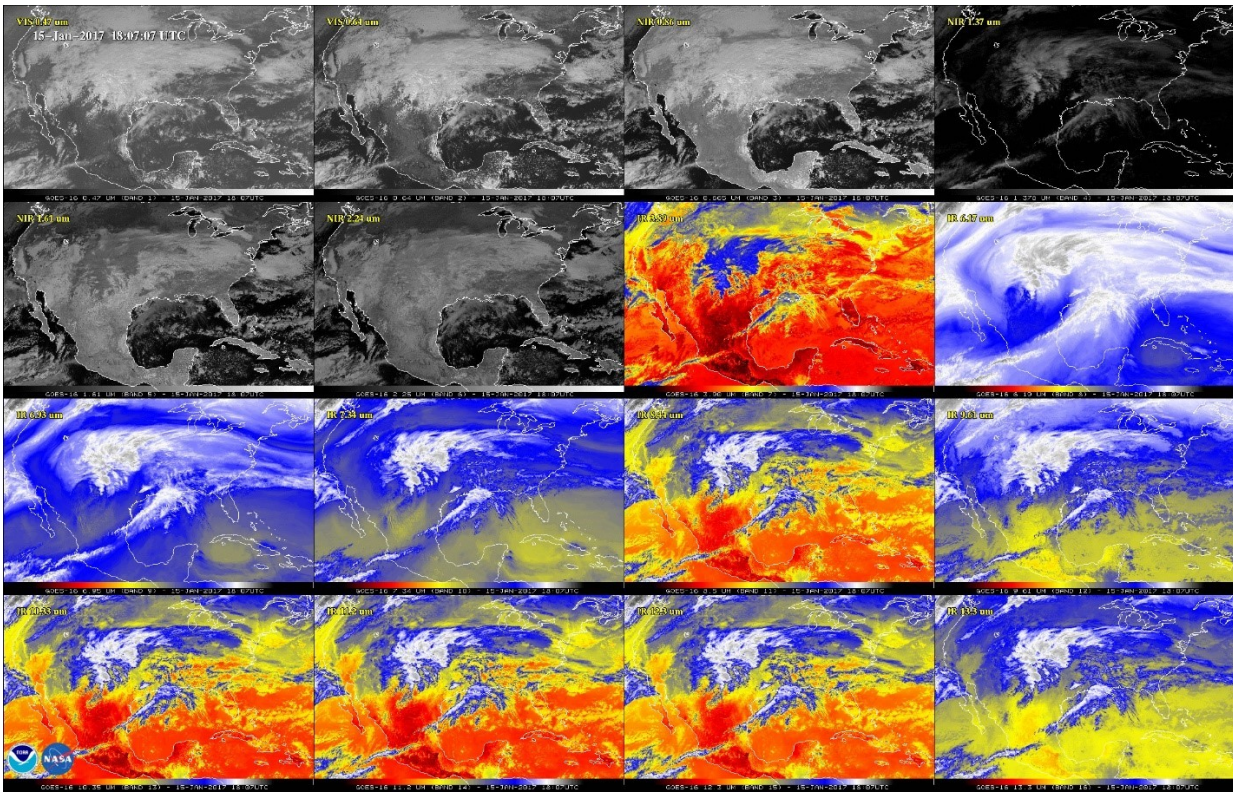

Figura 1.4.11: 16 bandas espectrales de ABI, 15 de enero de 2017. Las dos primeras bandas detectan lo visible, las cuatro siguientes en el NIR y las diez últimas en el IR.

GOES Rebroadcast (GRB) es el principal retransmisor espacial de productos y datos de GOES-R en tiempo casi real, de resolución completa y calibrada en nivel de procesamiento “Level 1b” para (ABI, MAG, SEISS, SUVI, EXIS) y datos de nivel de procesamiento “Level 2” para GLM.

Los instrumentos satelitales GOES-R transmiten datos brutos sobre la Tierra, el medio ambiente y el espacio, que son recibidos por el sistema terrestre. Las sólidas capacidades de gestión y procesamiento de datos del sistema terrestre convierten los datos sin procesar en imágenes e información que son asimiladas por los modelos informáticos. El sistema terrestre envía estos datos de nivel “Level 1b” de cada instrumento y los datos de nivel “Level 2” de la señal de GLM al satélite GOES-R para su retransmisión que puede ser recibida por terminales de lectura directa. GRB es una transmisión de lectura directa de 31 megabytes. El satélite GOES-R transmite la señal GRB a terminales de recepción en tierra compatibles en el campo de visión del

satélite. Una terminal de tierra GRB incluye una antena y un conjunto de equipos para procesar las señales de radiofrecuencia en un flujo de datos digitales e ingerir los datos GRB para producir productos meteorológicos.

En marzo de 2017, el Instituto de Geografía adquirió una terminal de tierra GRB. El Laboratorio Nacional de Observación de la Tierra es el encargado de recibir, administrar y procesar la información que es retransmitida por GOES-R en los niveles de procesamiento "Level 1" para ABI.

Para la obtención del producto CMIP a partir de las imágenes en radiancia, existen varias opciones, software privativo y de paga como TeraScan de la empresa SeaSpace que tiene la ventaja de ser muy fácil de usar y no requiere de conocimientos de programación pues incluye soporte técnico, la desventaja es que al ser software privativo necesita del pago de una costosa licencia anual y el mayor inconveniente, no se tiene acceso al código fuente, lo que impide cualquier modificación a los parámetros preestablecidos del software. La otra opción es utilizar software libre como lo es el "Paquete de procesamiento satelital comunitario para datos geoestacionarios" (CSPP Geo por sus siglas en inglés) es un proyecto desarrollado por la universidad de Wisconsin, este último es el software que se utiliza en el LANOT para generar algunos productos L2+ oficiales.

2. Algoritmo

El algoritmo de detección de incendios tiene como insumo principal dos bandas de ABI, la banda 7 en 3.9μm ($T_{3.9}$) y la banda 14 en 11.2μm ($T_{11.2}$), la primera se utiliza sola y la segunda en una diferencia de la banda 7 menos la banda 14 ($\Delta T = T_{3.9} - T_{11.2}$). Estas bandas tienen un nivel de procesamiento L2+ con valores de temperatura de brillo (BT por sus siglas en inglés) y se obtienen del producto CMIP generado por CSPP Geo.

Los algoritmos para detectar incendios se pueden agrupan en tres categorías: 1) umbral fijo, 2) multi-umbral y 3) spaciales o contextuales. Los dos primeros dependen de umbrales predeterminados empíricamente y pruebas de campo, en tanto que los algoritmos contextuales utilizan múltiples umbrales y análisis estadísticos (Manzo Delgado, 2020).

Tomando en cuenta esto último, este es un algoritmo contextual multi-umbral dinámico. Los algoritmos multi-umbral tienen la desventaja de que la temperatura de la tierra y de un incendio no es igual durante todo el día, varía dependiendo de la altura del sol, la hora de la noche y la estación del año del día en cuestión, esto hace que los algoritmos multi-umbral solo funcionen para cierta hora del día, lo cual no es un problema para satélites de órbita polar que solo pasan 1 o dos veces al día por un lugar. Pero si es un problema para los satélites en órbita geoestacionaria, donde tenemos imágenes cada 10 minutos, durante todo el día. El algoritmo considera la variación de la temperatura durante el transcurso del día, así que se consideró conveniente generar umbrales cada 10 minutos durante todo el día, todos los días del año. También se consideró el cambio de temperatura a lo largo del año, debido a las estaciones y la condiciones atmosféricas; por lo tanto, para detectar los puntos de calor asociados a incendios de una imagen, se utiliza el valor máximo de los umbrales óptimos de los 10 días anteriores a la fecha de ejecución, donde las condiciones atmosféricas son similares al día en cuestión.

Con los umbrales obtenidos se realiza una reclasificación de las bandas $T_{3.9}$ y ΔT. Los resultados son matrices booleanas, donde 1 es un posible incendio y 0 no es incendio. Estos dos productos se multiplican a su vez con tres máscaras: una máscara de agua, otra de suelo desnudo y una de manchas urbanas, que a su vez también son matrices booleanas, donde 0 es agua, suelo

desnudo o una mancha urbana. La multiplicación de estas 5 matrices booleanas genera una matriz boleana, donde los píxeles que sigan siendo 1 son nombrados pixeles potenciales (PP).

Como último paso, la matriz de pixeles potenciales es sometida a una prueba estadística de contexto, donde se crea un kernel de 7x7 pixeles para cada PP y se analizan las estadísticas por medio de una serie de condiciones que el pixel potencial debe cumplir respecto a sus vecinos y para confirmar que es un píxel que puede incluir un incendio. Esta prueba también disminuye los errores de comisión (falsos incendios), particularmente donde se agrupan numerosos puntos. Si el pixel potencial cumple con las condiciones se confirma un incendio y se procede a su caracterización.

Con el objetivo de ejemplificar todo este procedimiento, se procederá a realizar una muestra de todos los procesos que realiza el algoritmo para la detección de los incendios, tomaremos como día de ejemplo el 11 de mayo de 2020, número de día 132, donde se presentaron varios incendios en el país, en particular haremos un zoom a los datos en una serie de incendios que ocurrieron en el estado de guerrero en las coordenadas (101.55°W, 101.0°W, 17.66°N, 18.06°N) (ver Figura 2.1.1)

2.1. Creación de umbrales

La nueva técnica para determinar los umbrales de $T_{3.9}$ y ΔT, consiste en la creación de eventos FIRMS/VIIRS, donde los puntos de calor de FIRMS/VIIRS son filtrados y agrupados siguiendo ciertas reglas con la finalidad de dejar los incendios más grandes y representativos de un día, posterior a ello se realiza una extracción de los valores de BT de $T_{3.9}$ y ΔT en las coordenadas que hay un evento FIRMS/VIIRS con el objetivo de conocer la temperatura de brillo en esos pixeles, donde tenemos la certeza que a una hora de ese día (hora de paso de VIIRS) existió un incendio considerablemente grande, este proceso de extracción de valores de BT se realiza una vez, todos los días para cada hora del día y después se realiza una interpolación con funciones de spline para conocer los valores cada 10 minutos. Con esto se generan ‘x’ valores de BT en $T_{3.9}$ y ΔT para cada 10 minutos, donde ‘x’ es el número de eventos FIRMS/VIIRS del día. Para reducir el número de valores de BT de cada 10 minutos a solamente 1, se aplica una condición y se

obtiene el umbral final de ese horario. Este procedimiento se aplica a los 144 horarios de GOES-16.

En lenguaje de programación Python, este proceso se puede consultar en el subcapítulo 4.1 Scripts (s1_umbrales_auto.py) del capitulo 4 de Resultados.

2.1.1 Creación de eventos FIRMS/VIIRS

EL primer paso es descargar los puntos de FIRMS/VIIRS de la página https://firms.modaps.eosdis.nasa.gov/active_fire/#firms-shapefile, de la región "Central America" (Figura 1.3.2). El GeoDataFrame[23] descargado es del día 11 de mayo de 2020 y contiene las columnas que se mencionan en la Tabla 1.3.1, EPSG:4326[24] y el número de puntos de calor detectados ese día fue de 5850 (Figura 2.1.1). Número que se reducirá al crear los eventos FIRMS/VIIRS

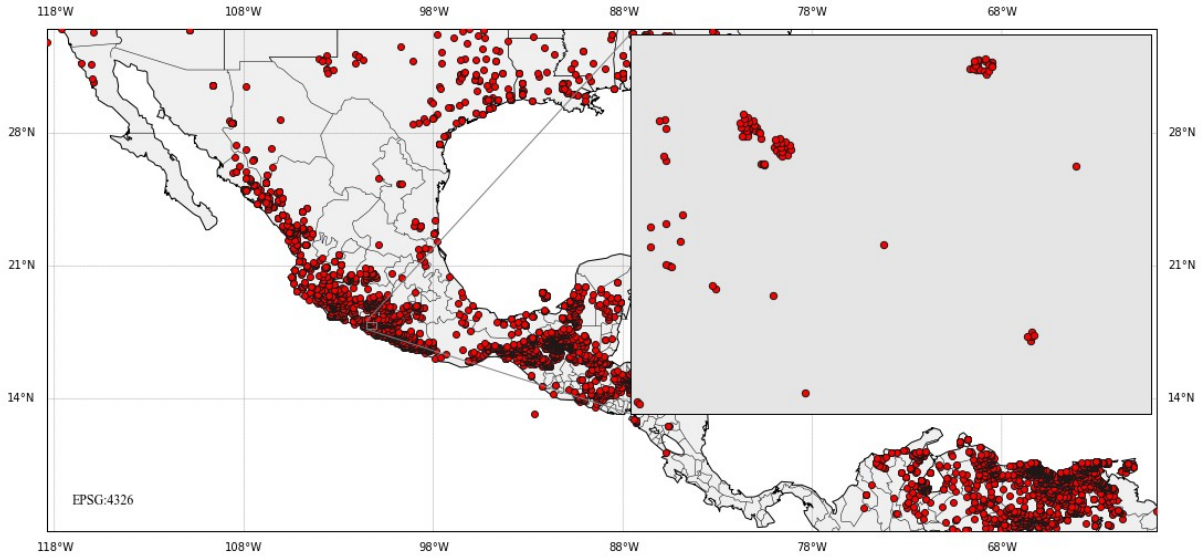

Figura 2.1.1: Puntos de calor FIRMS/VIIRS originales del día 11 de mayo de 2020 y acercamiento a la zona de ejemplo (101.55°W, 101.0°W, 17.66°N, 18.06°N). Elaboración propia con datos de FIRMS/NASA.

23 Un DataFrame contiene datos tabulares bidimensionales, de tamaño variable y potencialmente heterogéneos. Un GeoDataFrame es un DataFrame con una columna de geometría, lo que hace que pueda tener una representación espacial.

24 EPSG:4326 es el sistema de coordenadas geográficas asociadas al elipsoide y datum WGS84

El siguiente paso es hacer un buffer de 0.0025 grados decimales, equivalentes a 278m aproximadamente, tomando en cuenta que 1 grado equivale a 111.32km en el ecuador, este buffer se hizo un poco más grande que la resolución de VIIRS con el objetivo de que los polígonos se intersecten en el espacio. El siguiente paso es agrupar los polígonos por fecha y hora, el resultado de esto es pasar de 5850 puntos a 12 multipolígonos, que son los horarios distintos que registra VIIRS durante su paso por el recorte de "Central America" durante un día, estos 12 multipolígonos se deben separar en polígonos simples, por lo tanto los polígonos que no tengan una intersección espacial con otro polígono, aunque sea de la misma fecha y hora, son reparados. El resultado son 3543 polígonos simples, con tres columnas solamente, la geometría ("geometry"), la fecha ("fecha") y el área en kilómetros cuadrados ("areakm2"). El resultado se nombra FIRMS_hora_fecha, como se muestra en la Figura 2.1.2.

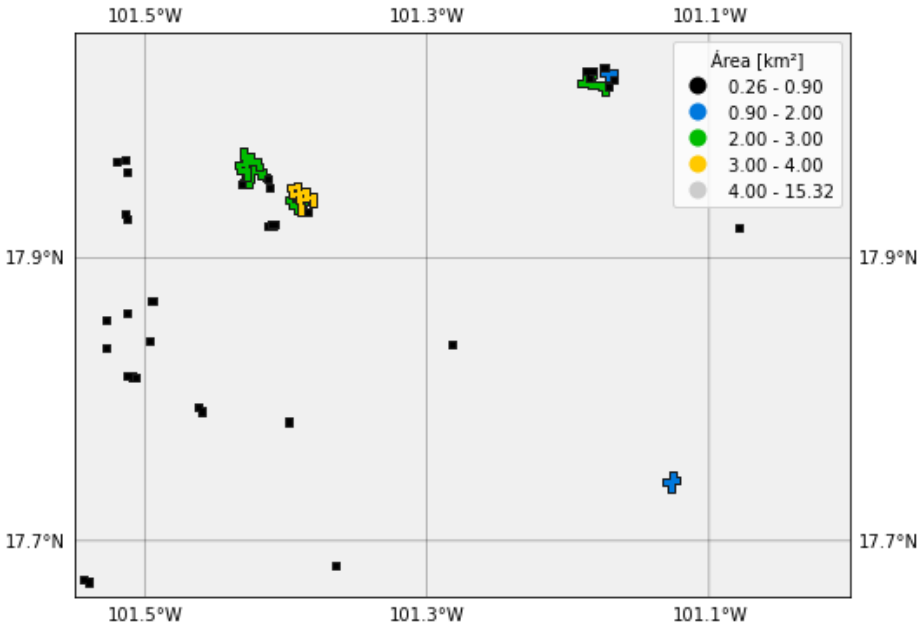

Figura 2.1.2: FIRMS_hora_2020132 del la zona de ejemplo, los polígonos están agrupados por hora y se muestra el área en km². Creación propia.

El siguiente paso consiste en agrupar los 3543 polígonos anteriores por día, como es un solo día, el resultado es 1 multipolígono que contiene a todos los 3543 polígonos del resultado anterior, ese multipolígono se separa nuevamente en polígonos simples, por lo tanto, los

polígonos que no tengan una intersección espacial con otro polígono, aunque sean del mismo día son separados, el resultado son 3249 polígonos únicos y ya no existe ningún polígono que se intersecte con otro. Los 3249 polígonos son caracterizados y se les agrega los siguientes atributos:

- npuntos: Número de puntos FIRMS/VIIRS originales que se intersectan con el polígono.
- first: Primera hora registrada de los puntos FIRMS/VIIRS que incluye el polígono.
- last: Última hora registrada de los puntos FIRMS/VIIRS que incluye el polígono.
- nhoras: Diferencia de horas (last – first), esto nos dice un aproximando de cuanto tiempo estuvo activo el incendio.

Con esta información se procede a hacer un filtro, para dejar solamente los incendios más grandes. La condición es la siguiente:

$$((nhoras > 2) \& (npuntos > 2) \& (areakm2 \geq 1))$$

Después de aplicar la condición, pasamos de 3249 a 83 eventos, estos son los eventos FIRMS/VIIRS (Figura 2.1.3), nombrados “evnt_FIRMS_ref_fecha” y servirán como referencia para extraer los valores de BT de $T_{3.9}$ y ΔT de ABI.

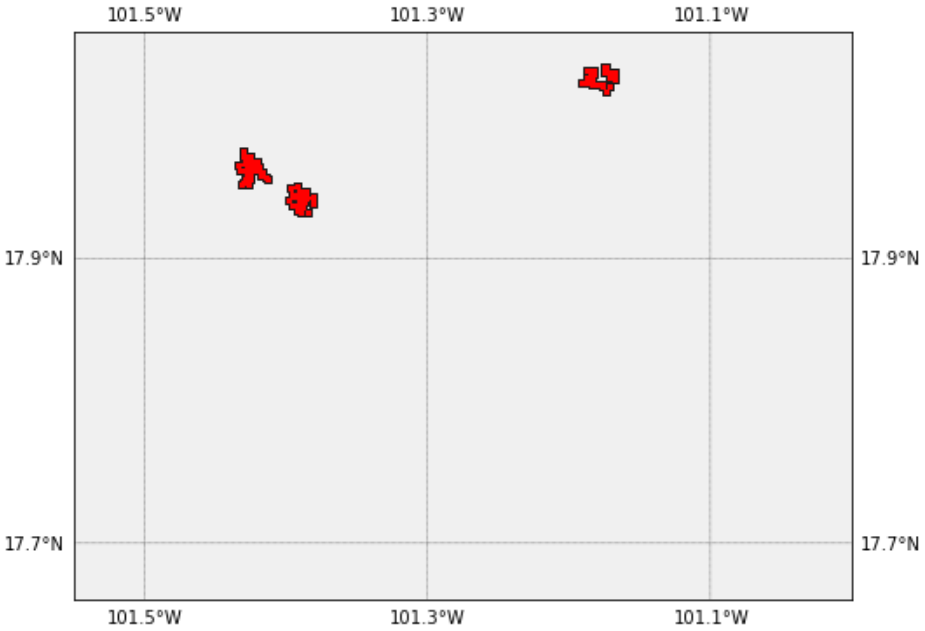

Figura 2.1.3: Eventos FIRMS/VIIRS del día 11 de mayo de 2020 del recorte de ejemplo. Elaboración propia.

En lenguaje de programación Python, este proceso se puede consultar en el subcapítulo 4.1 Scripts (s1_1_crea_puntos_firms.py y s1_3_crea_eventos_firms.py) del capitulo 4 de Resultados.

2.1.2 Muestras GOES 16

Para la obtención de las muestras de $T_{3.9}$ y ΔT en las coordenadas donde hay un evento FIRMS/VIIRS, los eventos FIRMS/VIIRS se re-proyectaron a coordenadas geoestacionarias de GOES-16 y se les aplicó un buffer de 1000m, con el objetivo de que cada evento FIRMS/VIIRS tenga una intersección espacial con dos o más pixeles de ABI, cuya resolución espacial es de 2km al nadir. Posterior a ello se extrajeron las estadísticas zonales de cada evento y se creó un nuevo DataFrame con los valores máximos registrados en cada evento, este procedimiento se aplicó a los 144 horarios del día (imágenes cada 10 minutos, 144 imágenes de cada banda al día) y el resultante es un DataFrame con 84 eventos por 144 horarios que es igual a 11,952 datos de temperatura máxima para un día de $T_{3.9}$ (valores_muestreados_ch07_fecha) y 11,952 de ΔT (valores_muestreados__dif7_14_fecha).

En lenguaje de programación Python, este proceso se puede consultar en el subcapítulo 4.1 Scripts (s1_4_muestras_goes_xdia.py) del capítulo 4 de Resultados.

2.1.3 Filtro para obtención de umbrales por día

El objetivo final es obtener un umbral para cada uno de los 144 horarios, para esto, el primer paso fue obtener el umbral por hora (24hrs) de cada uno de los 83 eventos, esto se hizo mediante la fórmula:

$$U_h = vmax - 1\,std$$

donde:

U_h = Umbral a cada hora de cada evento

vmax = Valor máximo a cada hora de cada evento

std = desviación estándar de cada hora de cada evento

Como se tienen 83 eventos para cada hora, las 24 hora del día, el resultado fueron 1992 valores de U_h, es lo mismo que 83 valores cada hora, para conseguir solamente un umbral por hora se utilizó como valor representativo el promedio de los 83 valores de U_h. Como último paso los 24 umbrales fueron usados para realizar una interpolación con funciones spline y obtener los umbrales de referencia a cada diez minutos.

Una interpolación es la obtención de nuevos puntos partiendo de un conjunto de puntos conocido. De manera específica, la interpolación segmentaría cuadrática para un conjunto de puntos está representada por ‘n’ polinomios de segundo orden (k = 2), siendo n+1 el número de puntos conocidos.

Considerando los 24 umbrales de la etapa anterior, se realizó una interpolación segmentaría cuadrática para obtener los umbrales cada diez minutos.

Entonces tendríamos que para cada día el conjunto de puntos conocidos es el siguiente:

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n); n \in \mathbb{N} / n = 23$$

Donde:

x = las 24 horas del día, a cada hora

y = umbral del canal 7

Por lo tanto, el polinomio para los intervalos de cada hora se representa en forma general como:

$$f_i(x)=a_i x^2+b_i x+c_i \quad ...(1)$$

Si consideramos a (n + 1) como el número de puntos conocidos, tendremos (n) número de ecuaciones de interpolación (1) que conforman la función de interpolación.

Entonces un spline de grado 2 es una función S(x):

$$s(x)=\begin{cases} s_0(x)=a_0(x^2)+b_0(x)+c_0; x\in[x_0,x_1] \\ s_1(x)=a_1(x^2)+b_1(x)+c_1; x\in[x_1,x_2] \\ \cdot \\ \cdot \\ \cdot \\ s_{22}(x)=a_{22}(x^2)+b_{22}(x)+c_{22}; x\in[x_{21},x_{22}] \end{cases}$$

Considerando los 24 umbrales de $T_{3.9}$ y ΔT de la etapa anterior, se realizó el mismo procedimiento descrito anteriormente para obtener los umbrales cada diez minutos. Como resultado de esta etapa, se generaron 144 pares de umbrales de referencia para $T_{3.9}$ y 144 para ΔT. Estos son los llamados umbrales óptimos por día y este procedimiento de creación de umbrales se realiza para todos los días del año.

En lenguaje de programación Python, este proceso se puede consultar en el subcapítulo 4.1 Scripts (s1_5_crea_umbrales.py) del capítulo 4 de Resultados.

2.2. Obtención de los puntos de calor

En este capítulo se describe el procedimiento para la obtención de los puntos de calor (PC), este proceso se ejecuta cada que hay una imagen nueva (cada 10 minutos) y es independiente al proceso de la creación de umbrales.

El proceso consiste en la preparación de $T_{3.9}$ y ΔT en un recorte para México, posteriormente se reclasifican basándonos en los umbrales obtenidos en el capítulo anterior, este resultado se multiplica por las capas estáticas, que sirven para filtrar errores asociados a suelo desnudo, agua

y manchas urbanas, para así obtener los pixeles potenciales a ser incendios. Los pixeles potenciales resultantes del paso anterior se someterán a una prueba estadística de contexto, para confirmar que es un punto de calor y a los puntos de calor finales se les agregara información relevante para hacer algunos análisis.

En lenguaje de programación Python, este proceso se puede consultar en el subcapítulo 4.1 Scripts (s2_incendios_auto.py) del capitulo 4 de Resultados.

2.2.1 Preparación de imágenes GOES 16

Se tomó como base las bandas 7 y 14 en formato NetCDF de la escena FullDisk generadas por CSPP Geo en nivel de procesamiento L2+, producto CMI (Figura 2.2.1). El nombre estándar para una banda con estas características es el siguiente: OR_ABI-L2-CMIPF_M6C07_G16_ sYYYYJJJHHMMSSs_eYYYYJJJHHMMSSs_cYYYYJJJHHMMSSs.nc y la nomenclatura se puede observar en la Tabla 2.2.1:

Abreviatura	Significado
OR	Operational System Real-Time Data
ABI	Advanced Baseline Imager
L2	Level 2+
CMIPF	Cloud and Moisture Image Product – FullDisk
M6	ABI Mode 6
C07	Channel Number
G16	GOES-16
sYYYYJJJHHMMSSs	Observation Start
eYYYYJJJHHMMSSs	Observation End
cYYYYJJJHHMMSSs	File Creation

Tabla 2.2.1: Nomenclatura NetCDF GOES-16. Nota: YYYY = Año, JJJ = Número de día del año (000 - 365), HH = Hora (00 - 23), MM = Mes (01 - 12), SS = Segundos (00 - 59), s = Mili-segundos (0-9)

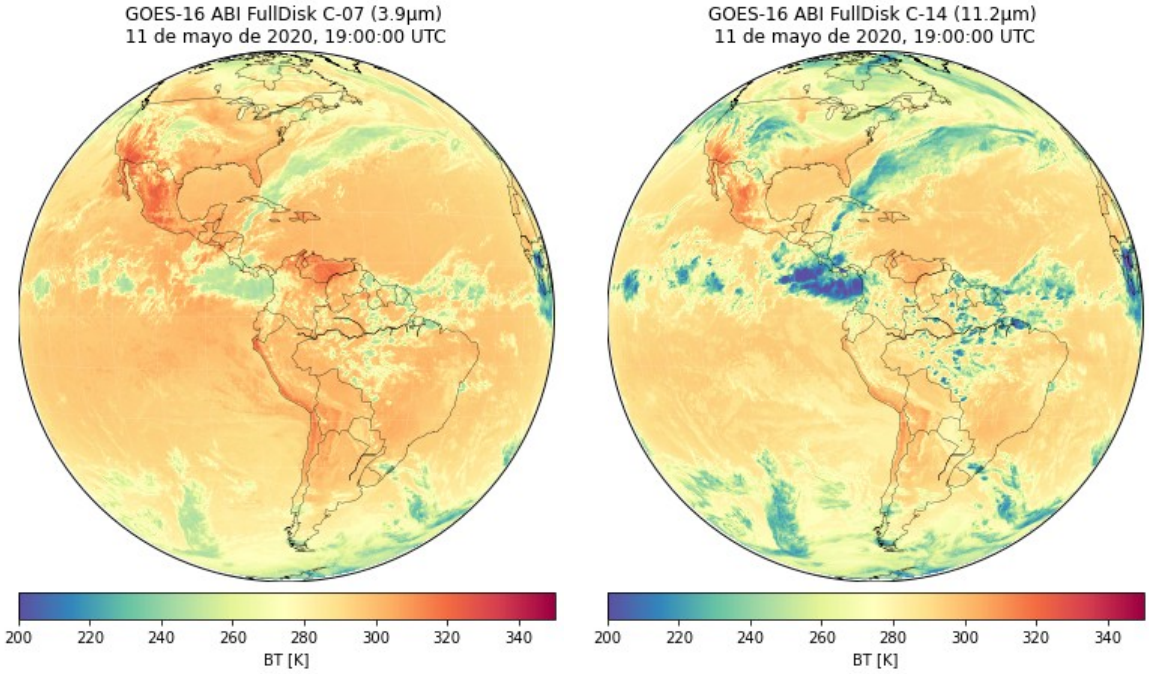

Figura 2.2.1: Banda 7 y 14 en proyección geoestacionaria, FullDisk. Elaboración propia

El siguiente paso es eliminar los valores nulos de $T_{3.9}$ y $T_{11.2}$ basándonos en el rango válido del producto CMI (Tabla 1.4.3), reproyectarlas a coordenadas geográficas (EPSG:4326), generar ΔT y hacer un recorte para México (-118.30°W, -80.07°E, 10.09°S, 33.60°N) como se muestra en la (Figura 2.2.2). El algoritmo está diseñado para el recorte de México, cambiar o hacer la escena más grande podría generar errores, falsos incendios si la región fuera más cálida o la falta de detección de incendios en regiones más frías.

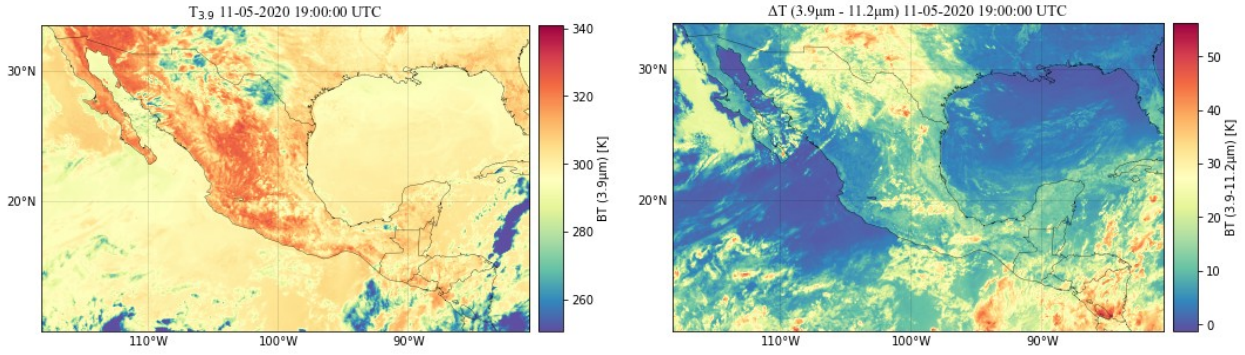

Figura 2.2.2: Recorte de México de T3.9 y ΔT en coordenadas geográficas. Elaboración propia

En lenguaje de programación Python, este proceso se puede consultar en el subcapítulo 4.1 Scripts (s2_1_prepara_imagenes_geo.py) del capítulo 4 de Resultados.

2.2.2 Extracción de umbral máximo de 10 días

Reclasificar $T_{3.9}$ y ΔT con los umbrales óptimos ($U_{3.9}$ y $U_{3.9\text{-}11.2}$) de la fecha que ese esté ejecutando es imposible, debido a que aún no se genera esta información, esta se genera al final de cada día, por lo tanto, procedimos a analizar los umbrales óptimos de los días pasados a la fecha de prueba, en ese mismo horario (19:00 UTC), como se puede observar en la Figura 2.2.3 los umbrales óptimos de un horario van variando con respecto al día, y hay cambios drásticos debido a que se pueden presentar días con muchos incendios donde los umbrales serán más estrictos y días donde no existan incendios o bien sean días nublados con presencia de nubes y los umbrales sean más bajos.

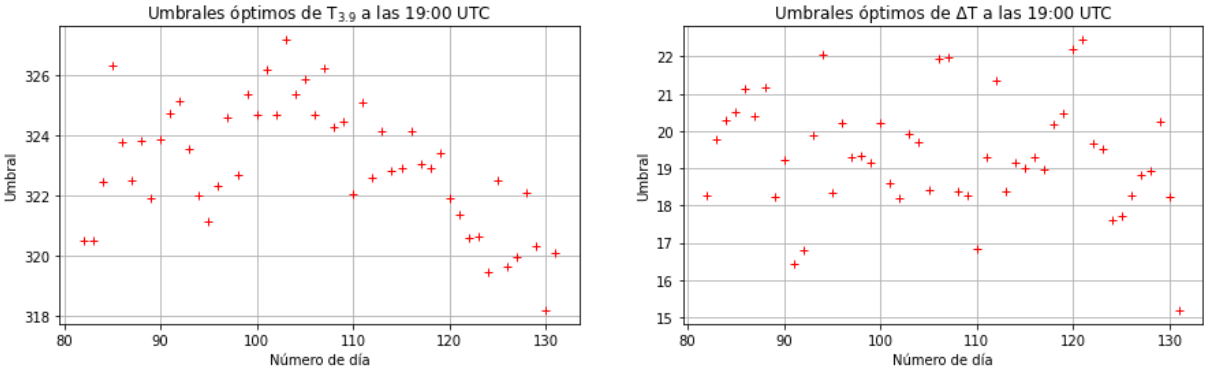

Figura 2.2.3: Umbrales óptimos del día 22 de marzo (82) al 10 de mayo (131) de 2020 a las 19:00 UTC. Creación propia

Tomando en cuenta esto último se decidió utilizar la información de los 10 días anteriores y usar como umbral el valor máximo los umbrales de ese horario, para el día 11 de mayo de 2020 (día 132) el umbral para $T_{3.9}$ ($U_{3.9}$) es: 322.52 y para ΔT ($U_{3.9\text{-}11.2}$)es: 20.27. Ver (Figura 2.2.10)

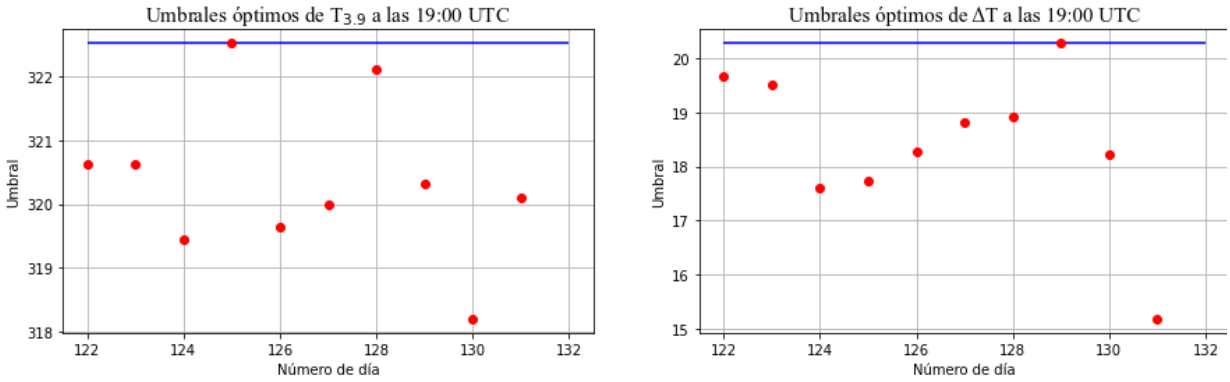

Figura 2.2.4: Umbrales óptimos los 10 días anteriores al 11 de mayo, del 1ro (día 122) al 10 de mayo (día 131) de 2020 a las 19:00 UTC, la linea azul representa el umbral para el día 132. Creación propia

En lenguaje de programación Python, este proceso se puede consultar en el subcapítulo 4.1 Scripts (Extracción de umbral máximo de 10 días) del capítulo 4 de Resultados.

2.2.3 Capas estáticas

Con el objetivo de eliminar los posibles errores asociados al agua, suelo desnudo y manchas urbanas, se crearon máscaras de cada una de estas características, estas máscaras son matrices booleanas, donde el agua, suelo desnudo y manchas urbanas son representadas por el número 0 y todo lo que no sea 0 es 1.

Las máscaras fueron generadas a partir del producto "Collection 6 MODIS Land Cover (MCD12Q1)". El producto MCD12Q1 proporciona mapas globales de la cobertura terrestre en pasos de tiempo anuales y una resolución espacial de 500m. El archivo HDF[25] contiene 13 conjuntos de datos científicos (Tabla 2.2.2), incluidos 5 esquemas de clasificación heredados: International Geosphere-Biosphere Programme (IGBP), University of Maryland (UMD), Leaf Area Index (LAI), BIOME-Biogeochemical Cycles (BGC) y Plant Functional Types (PFT) (Sulla-Menashe & A Friedl, 2018). Estos son distintos esquemas de clasificación y cambian las

25 HDF es un formato de archivo de datos utilizado por Hierarchical Data Format. Los archivos HDF se utilizan para la transferencia de datos gráficos y numéricos entre máquinas. Los archivos HDF almacenan datos generalmente relacionados con datos científicos. Esta información se almacena en bibliotecas y archivos de múltiples objetos.

clases de cada uno, estas se pueden consultar en: https://lpdaac.usgs.gov/products/mcd12q1v006/.

Nombre corto	Nombre completo	Descripción	Rango valido	Valor de relleno
LC_Type1	Land Cover Type 1	Clasificación IGBP anual	[1,17]	255
LC_Type2	Land Cover Type 2	Clasificación UMD anual	[0,15]	255
LC_Type3	Land Cover Type 3	Clasificación LAI anual	[0,10]	255
LC_Type4	Land Cover Type 4	Clasificación BGC anual	[0,8]	255
LC_Type5	Land Cover Type 5	Clasificación PFT anual	[0,11]	255
LC_Prop1	Land Cover Property 1	Capa de cobertura terrestre LCCS1	[1,43]	255
LC_Prop2	Land Cover Property 2	capa de cobertura terrestre LCCS2	[1,40]	255
LC_Prop3	Land Cover Property 3	capa de hidrología superficial LCCS3	[1,51]	255
LC_Prop1_Ass	Land Cover Property 1 Assessment	Confianza en la capa de cobertura terrestre LCCS1	[0,100]	255
LC_Prop2_Ass	Land Cover Property 2 Assessment	Confianza en la capa de uso del suelo LCCS2	[0,100]	255
LC_Prop3_Ass	Land Cover Property 3 Assessment	Confianza en la capa de hidrología superficial LCCS3	[0,100]	255
QC	Land Cover QC	Banderas de calidad del producto	[0,10]	255
LW	Land Water Mask	Máscara binaria de tierra (clase 2) / agua (clase 1) derivada de MOD44W	[1,2]	255

Tabla 2.2.2: Conjuntos de datos científicos de MCD12Q1. Elaboración propia con datos del USGS

Los archivos HDF fueron descargados de la página https://earthexplorer.usgs.gov/, se descargaron 15 escenas para completar el recorte de México, formato HDF, en coordenadas sinusoidales. Con estos datos se generó un mosaico, se proyectó a coordenadas geográficas, se remuestreó a 2km (tamaño de pixel de ABI) y se recortó a los límites de la región de trabajo para que coincidiera en número de filas y columnas con nuestros recortes de $T_{3.9}$ y ΔT y así poder hacer la multiplicación de las matrices booleanas.

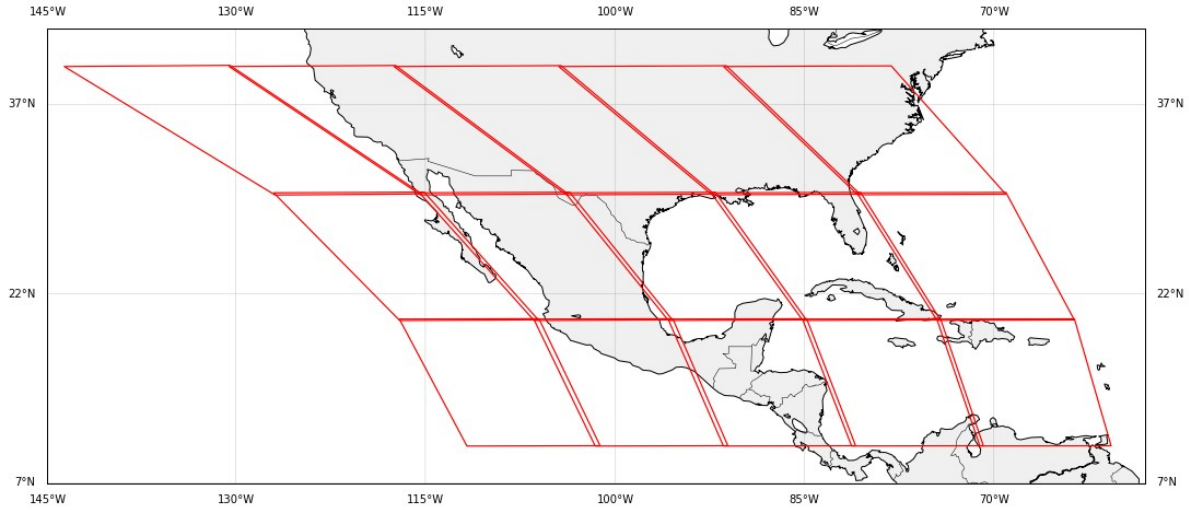

Figura 2.2.5: Escenas descargadas del producto MCD12Q1 para generar el mosaico de la región de México. Elaboración propia con datos del USGS

Para generar las máscaras utilizamos la clasificación IGBP anual (LC_Type1), esta cuenta con 17 clases, como se muestra en la Tabla 2.2.3.

Valor	Nombre en ingles (original)	Descripción en ingles (original)
1	Evergreen Needleleaf Forests	Dominated by evergreen conifer trees (canopy >2m). Tree cover >60%.
2	Evergreen Broadleaf Forests	Dominated by evergreen broadleaf and palmate trees (canopy >2m). Tree cover >60%.
3	Deciduous Needleleaf Forests	Dominated by deciduous needleleaf (larch) trees (canopy >2m). Tree cover >60%.
4	Deciduous Broadleaf Forests	Dominated by deciduous broadleaf trees (canopy >2m). Tree cover >60%.
5	Mixed Forests	Dominated by neither deciduous nor evergreen (40-60% of each) tree type (canopy >2m). Tree cover >60%.
6	Closed Shrublands	Dominated by woody perennials (1-2m height) >60% cover.
7	Open Shrublands	Dominated by woody perennials (1-2m height) 10-60% cover.
8	Woody Savannas	Tree cover 30-60% (canopy >2m).
9	Savannas	Tree cover 10-30% (canopy >2m).
10	Grasslands	Dominated by herbaceous annuals (<2m).
11	Permanent Wetlands	Permanently inundated lands with 30-60% water cover and >10% vegetated cover.
12	Croplands	At least 60% of area is cultivated cropland.
13	Urban and Built-up Lands	At least 30% impervious surface area including building materials, asphalt, and vehicles
14	Cropland/Natural Vegetation Mosaics	Mosaics of small-scale cultivation 40-60% with natural tree, shrub, or herbaceous vegetation.
15	Permanent Snow and Ice	At least 60% of area is covered by snow and ice for at least 10 months of the year.
16	Barren	At least 60% of area is non-vegetated barren (sand, rock, soil) areas with less than 10% vegetation.
17	Water Bodies	At least 60% of area is covered by permanent water bodies.
255	Unclassified	Has not received a map label because of missing inputs.

Tabla 2.2.3: Leyenda y descripción de las clases del Programa Internacional de Geosfera-Biosfera, IGBP (LC_Type1). Datos: USGS

Con la clase número 13 (Urban and Built-up Lands) se construyó la máscara de manchas urbanas (M_{Urb}), con la número 16 (Barren) la máscara de suelo desértico (M_{Des}) y con la 17 (Water Bodies) la máscara de agua (M_{Agua}) Figura 2.2.6.

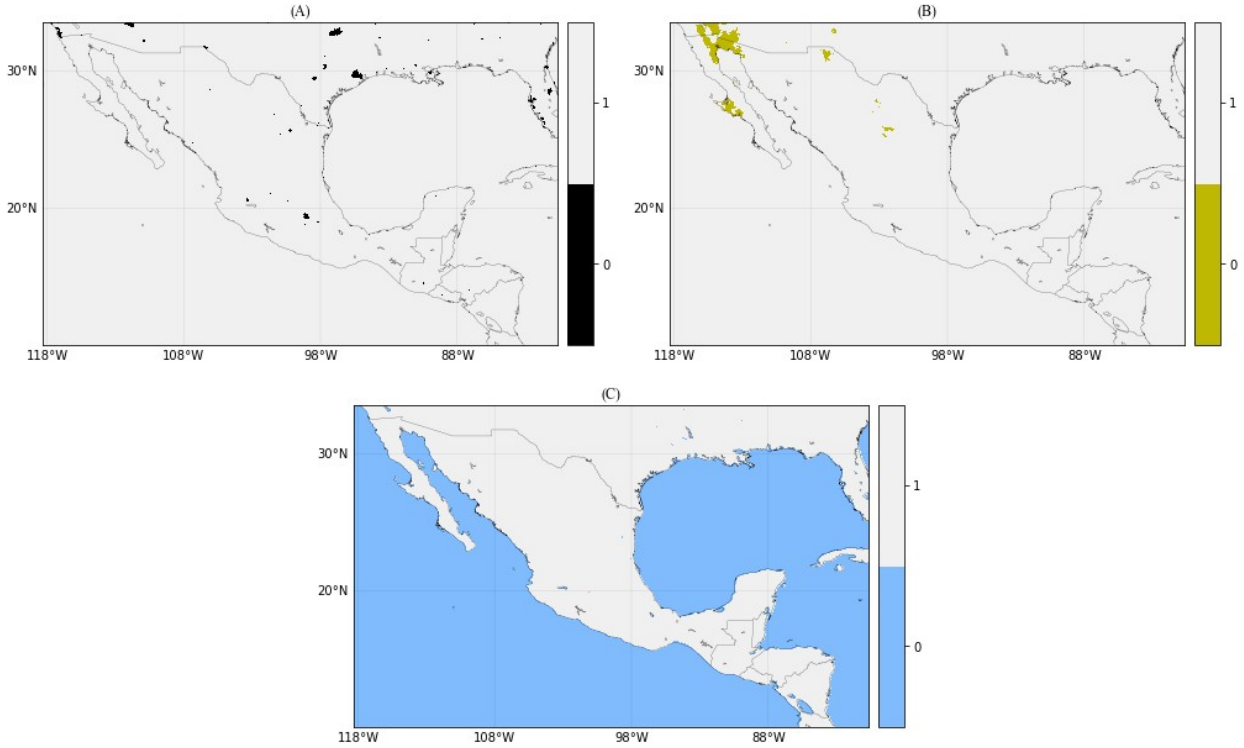

Figura 2.2.6: Máscaras de: A) Manchas urbanas (M_{Urb}), B) Suelo desértico (M_{Des}) y C) Agua (M_{Agua}). Generadas a partir del producto MCD12Q1, clasificación IGBP anual. Elaboración propia

2.2.4 Pixel potencial

Los recortes resultantes $T_{3.9}$ y ΔT de capítulo "*2.2.1 Preparación de imágenes GOES 16"* son reclasificados, $T_{3.9}$ es 1 si ($T_{3.9} > U_{3.9}$) y ΔT es 1 si ($\Delta T > U_{3.9\text{-}11.2}$), si no cumplen la condición, se les asigna el número 0. $U_{3.9}$ y $U_{3.9\text{-}11.2}$ son los umbrales obtenidos en el capítulo "*2.2.2 Extracción de umbral máximo de 10 días"*. El resultado son dos matrices booleanas donde 1 puede ser un incendio y 0 no. Para el día y horario del ejemplo el resultado se observa en la Figura 2.2.7.

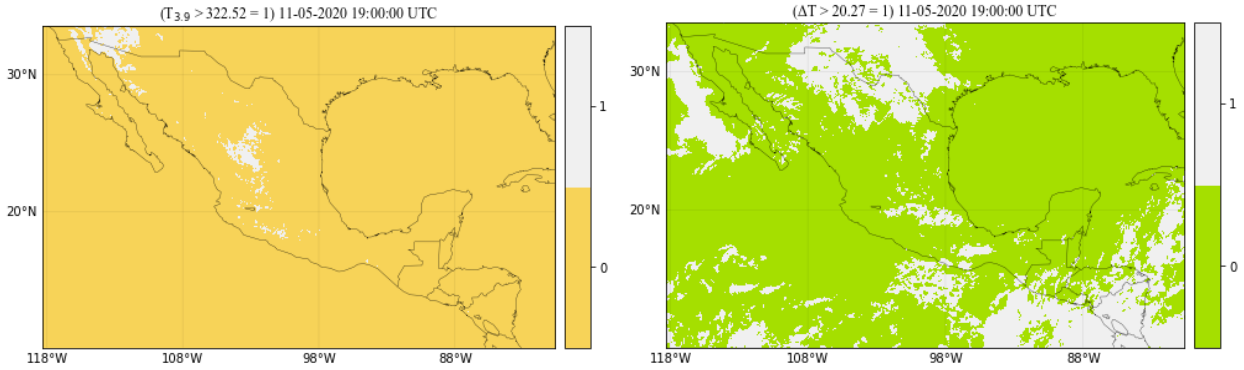

Figura 2.2.7: T3.9 y ΔT reclasificadas con sus respectivos umbrales. Elaboración propia

El último paso para generar los PP es hacer una multiplicación de las 5 matrices booleanas generadas, esto es lo mismo a aplicar la siguiente condición a cada pixel de la imagen:

$$(T_{3.9} > U_{3.9}) \& (\Delta T > U_{3.9-11.2}) \& (M_{Agua} = 1) \& (M_{Des} = 1) \& (M_{Urb} = 1)$$

Los pixeles que cumplan con la condición, son llamados pixeles potenciales. Para el la fecha del ejemplo, se detectaron 117 pixeles potenciales, con el objetivo de mostrar estos resultados, la Figura 2.2.8 es una muestra un acercamiento al área de estudio (Figura 2.1.1) de la banda 7 y los PP resultantes en ese recorte.

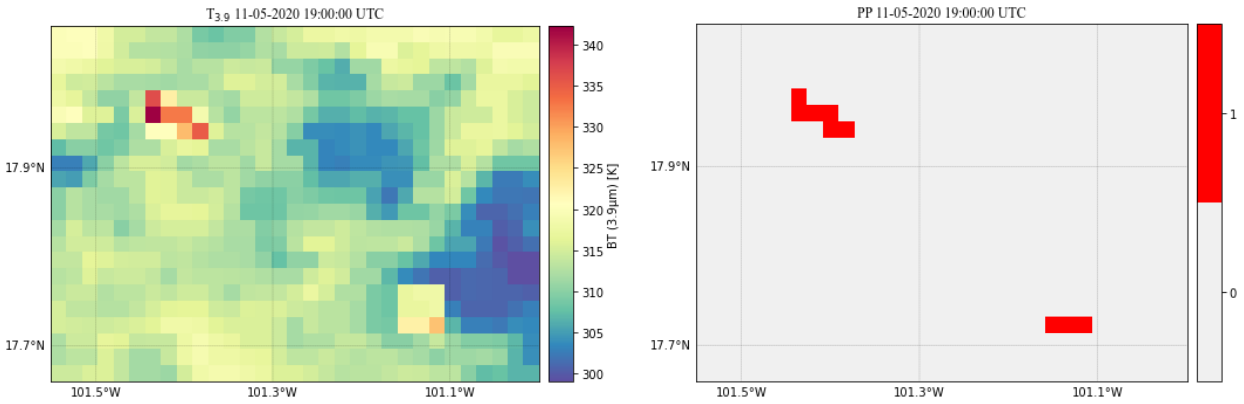

Figura 2.2.8: Acercamiento a $T_{3.9}$ y los PP resultantes de la escena de ejemplo. Elaboración propia

En lenguaje de programación Python, este proceso se puede consultar en el subcapítulo 4.1 Scripts (s2_3_pp_y_pc.py) del capítulo 4 de Resultados.

2.2.5 Prueba de contexto

La prueba de contexto cosiste en someter a una prueba estadística a cada pixel potencial con el objetivo de confirmar que el valor de temperatura de brillo del pixel sea superior al de sus vecinos, para eso se construyó un kernel de 7x7 pixeles para cada PP y este quedó en el centro de la matriz, como se observa en la Figura 2.2.9. Posteriormente, se extrajeron los valores de BT de $T_{3.9}$ y ΔT de la matriz para cada PP. Se confirma un punto de calor si cumple con la siguiente condición:

$$(PP_{3.9} > (mean(K_{3.9}) + 2.5\, std(K_{3.9}))) \& (PP_{3.9-11.2} > (mean(K_{3.9-11.2}) + 2.5\, std(K_{3.9-11.2})))$$

$\mathbf{K} =$

(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)
(0,2)	(1,2)	(2,2)	(3,2)	(4,2)	(5,2)	(6,2)
(0,3)	(1,3)	(2,3)	PP	(4,3)	(5,3)	(6,3)
(0,4)	(1,4)	(2,4)	(3,4)	(4,4)	(5,4)	(6,4)
(0,5)	(1,5)	(2,5)	(3,5)	(4,5)	(5,5)	(6,5)
(0,6)	(1,6)	(2,6)	(3,6)	(4,6)	(5,6)	(6,6)

Figura 2.2.9: Kernel de 7x7 pixeles para un PP. Elaboración propia.

Después de aplicar la prueba de contexto al 11 de mayo de 2020, pasamos de tener 117 PP a 40 puntos de calor confirmados (Figura 2.2.10).

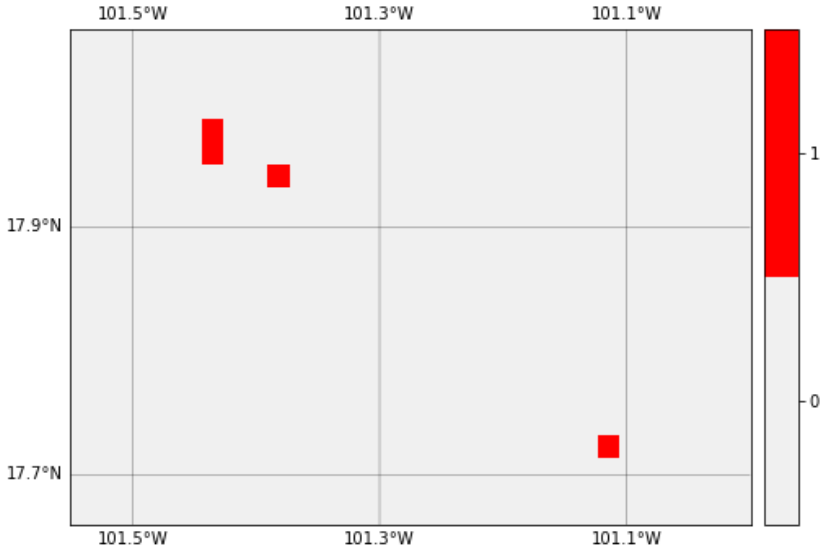

Figura 2.2.10: Acercamiento a los puntos de calor detectados a las 19:00 UTC el 11 de mayo de 2020. Elaboración propia.

En lenguaje de programación Python, este proceso se puede consultar en el subcapítulo 4.1 Scripts (s2_3_pp_y_pc.py), en el modulo "*def prueba_contexto():"* del capítulo 4 de Resultados.

2.2.6 Caracterización de los puntos de calor

Cada punto calor detectado será caracterizado con información relevante del punto. El producto final será creado en formato CSV y Shapefile con las columnas que se describen en la Tabla 2.2.4.

Nombre	Descripción	Rango de datos validos
lon	Longitud del PC	(-119°,-80°)
lat	Latitud del PC	(10°,34°)
Satelite	Satélite con el que se detecto	-
BT_c07	Temperatura de brillo en el canal 7	-
BT_c14	Temperatura de brillo en el canal 14	-
dif_c07-c14	Diferencia de temperatura de brillo (canal 7 – canal 14)	-
Fecha	Fecha de detección del PC	-
Hora	Hora de detección del PC	-
Land_Cover	Cubierta terrestre sobre la que cayó el punto de calor (datos de clasificación IGBP anual)	(1,17)
Estado	Entidad de México donde esta el PC	-
Pais	País donde esta el PC	-
ANP	Are Natural Protegida sobre la que esta el punto de calor, en caso de estarlo (datos de CONABIO).	-
freq	Frecuencia de detecciones durante el día de ese mismo PC	(1 - 144)
freq_norm	Frecuencia normalizada: Porcentaje de detecciones durante el día de ese mismo PC	(0 - 100%)

Tabla 2.2.4: Atributos del producto de incendios GOES-16/ABI, datos de las ANP proporcionados por el CONANP http://sig.conanp.gob.mx/website/pagsig/info_shape.htm.

En lenguaje de programación Python, este proceso se puede consultar en el subcapítulo 4.1 Scripts (s2_3_pp_y_pc.py) del capítulo 4 de Resultados.

3. Procesamiento de datos espaciales con Python

En este capítulo se describen los scripts, módulos y funciones utilizados y creados para la automatización, así como algunos procesos clave para la programación del algoritmo en lenguaje de programación Python en la versión 3.7.3. Los scripts se dividieron en dos grupos y cada script principal manda a llamar a una serie de módulos que contienen las funciones para hacer determinados procesos.

El primer script "s1_umbrales_auto.py" se encarga de la obtención de los umbrales diarios, esto es todos los procesos que se mencionan en el capítulo 2.1. Creación de umbrales, este script se ejecuta una vez al día, todos los días del año a las 00:00hrs CDT (Hora de verano central, UTC-5), esto es porque el producto FIRMS/VIIRS se genera dentro de las 3 horas próximas al paso del satélite y si llegara a pasar a las 23:59hrs UTC, los puntos FIRMS/VIIRS se estarían generando hasta las 02:59hrs UTC, así le damos un margen de 5hrs para evitar datos faltantes. El diagrama de flujo de la Figura 3.1 muestra los procesos que ejecuta este script y en que orden. Tiene como insumo inicial la banda 7 y 14 del producto CMIP en FullDisk de todo el día (144 imágenes por banda) y el producto FIRMS/VIIRS del día correspondiente.

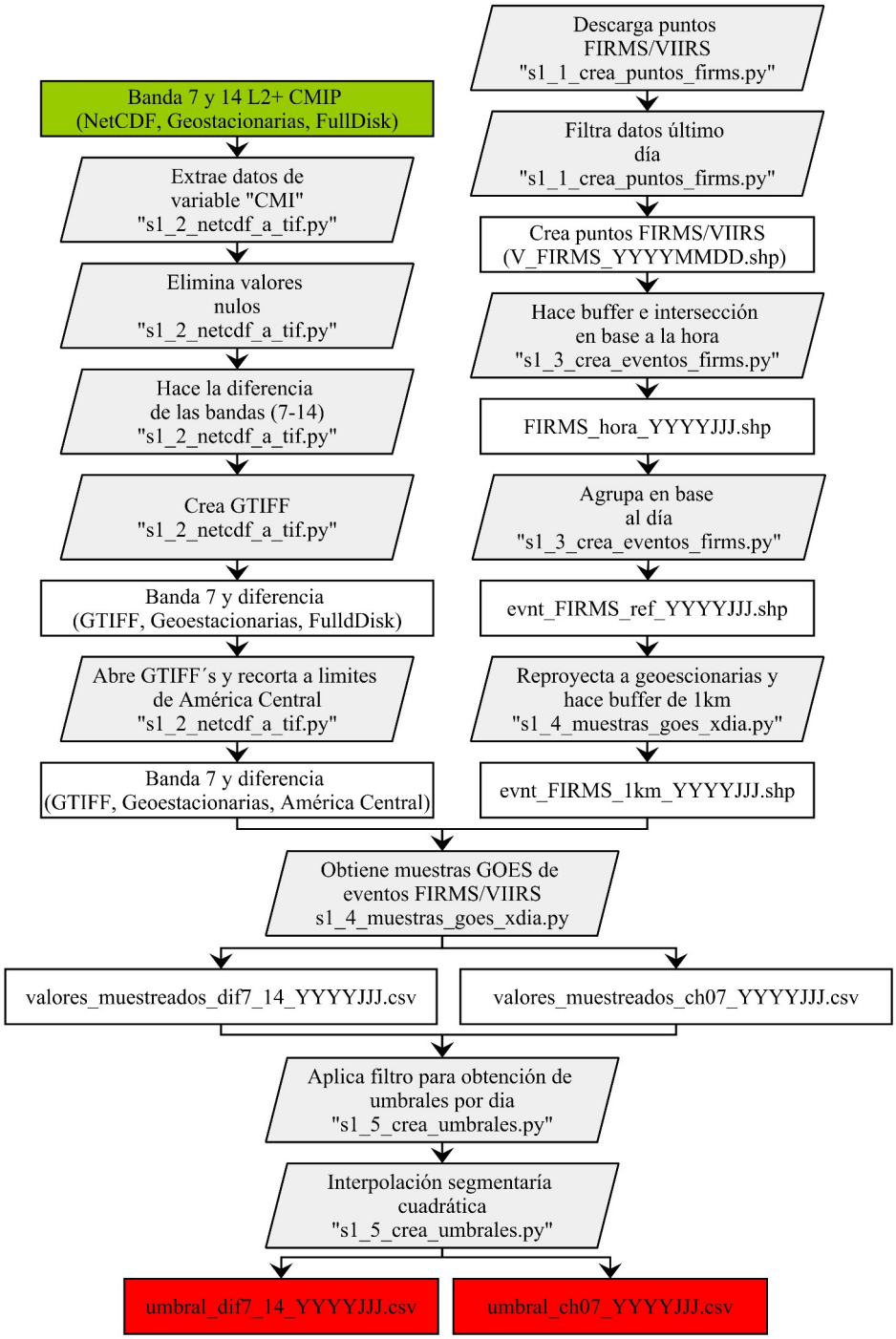

Figura 3.1: Diagrama de flujo del script "s1_umbrales_auto.py" , especifica en que modulo se ejecuta cada proceso. Elaboración propia

El segundo script “s2_incendios_auto.py” es el encargado de generar los puntos de calor cada 10 minutos, con esto nos referimos a todos los procesos que se mencionan en el capítulo 2.2.. Obtención de los puntos de calor y se ejecuta todos los días del año cada 10 minutos. La Figura 3.2 muestra los procesos que ejecuta este script cada 10 minutos

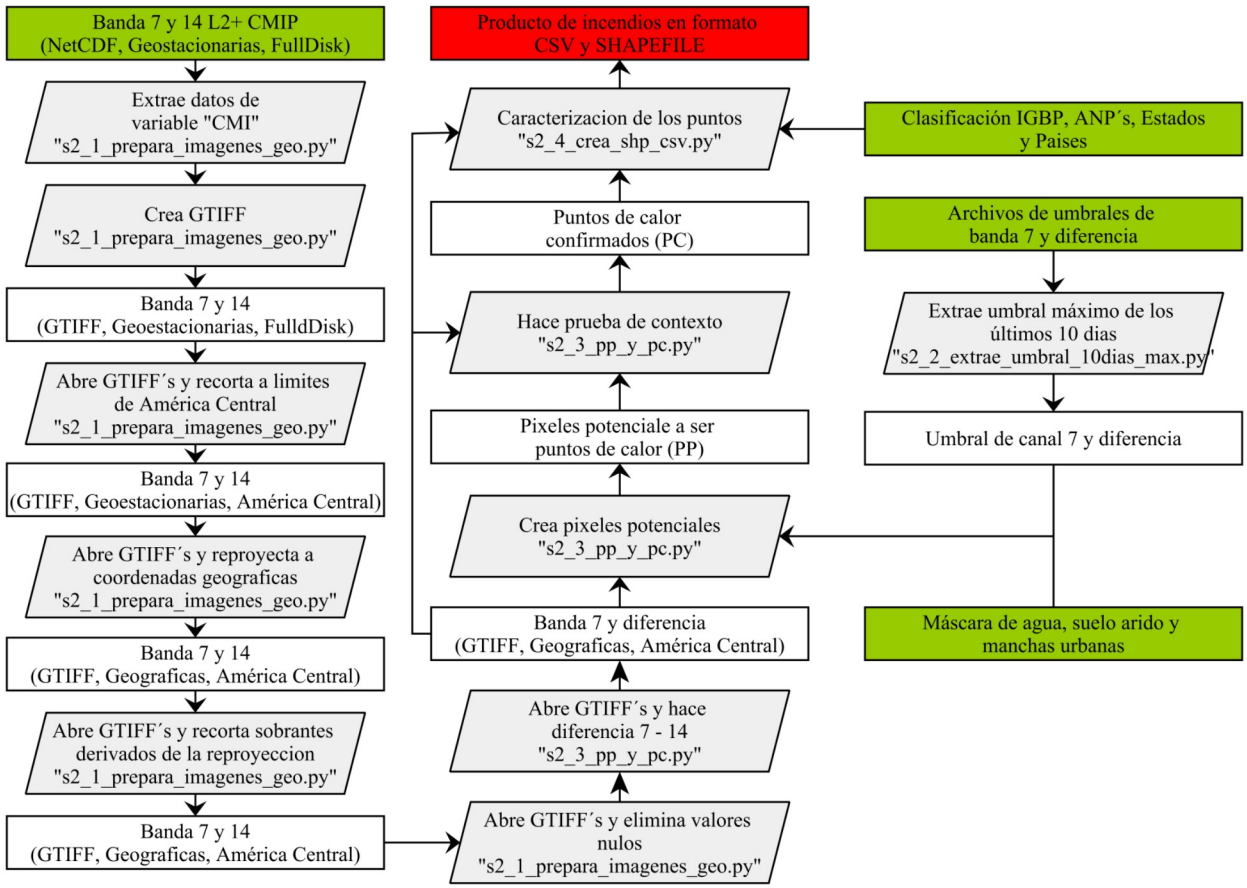

Figura 3.2: Diagrama de flujo del script "s2_incendios_auto.py", especifica en que modulo se ejecuta cada proceso Elaboración propia

Durante el proceso de ejecución del algoritmo se crean múltiples datos auxiliares que deben ser almacenados de manera temporal en algún directorio, por lo tanto, los scripts necesitan de una serie de directorios creados con anticipación para que corra de manera correcta como se muestra en la Figura 3.3. Los scripts y módulos están ubicados en el directorio /bin.

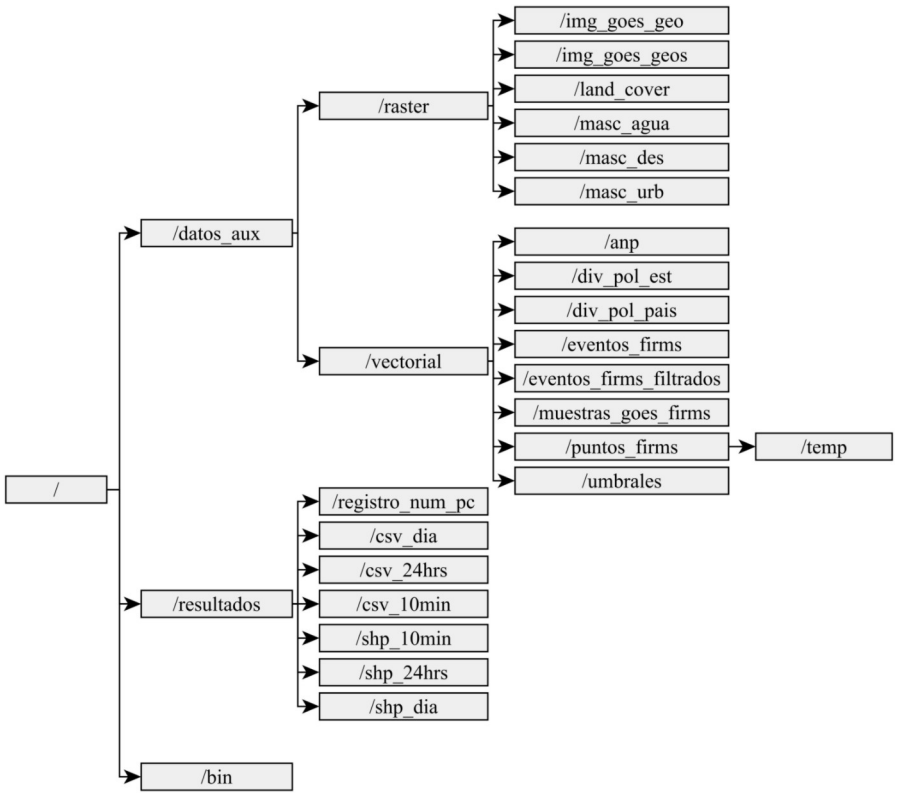

Figura 3.3: Árbol de directorios del algoritmo. Elaboración propia

/bin
s1_umbrales_auto.py
s1_1_crea_puntos_firms.py
s1_2_netcdf_a_tif.py
s1_3_crea_eventos_firms.py
s1_4_muestras_goes_xdia.py
s1_5_crea_umbrales.py
s2_incendios_auto.py
s2_1_prepara_imagenes_geo.py
s2_2_extrae_umbral_10dias_max.py
s2_3_pp_y_pc.py
s2_4_crea_shp_csv.py
s3_resultados_csv_shp_xdia_auto.py

Tabla 3.1: Scripts y módulos ubicados en /bin.

El script “s3_resultados_csv_shp_xdia_auto.py” se ejecuta todos los días a las 20:00hrs CDT (01:00hrs UTC) y su función es crear un archivo csv y shp de los resultados de todo el día, agrupando los 144 horarios del día de GOES-16/ABI en uno solo, que se guardara en “*/resultados/csv_dia* y */resultados/shp_dia”* con los nombres *GIM10_PC_YYYYJJJ.csv* y *GIM10_PC_YYYYJJJ.shp* respectivamente.

3.1. Bibliotecas y módulos utilizados

Para la programación del algoritmo se utilizaron las siguientes bibliotecas y módulos.

- Datetime: Este módulo proporciona clases para manipular fechas y horas (*docs.python.org/3/library/datetime.html*, s/f). Si bien la implementación permite operaciones aritméticas con fechas y horas, su principal uso en este algoritmo fue para transformar a diferentes formatos de fecha, principalmente de calendario gregoriano a día juliano y viceversa.
- GeoPandas: Es una biblioteca de código abierto para facilitar el trabajo con datos geoespaciales en formato vectorial en Python. GeoPandas amplía los tipos de datos utilizados por Pandas para permitir operaciones espaciales en objetos geométricos (geopandas.org, 2020).
- GDAL: “Geospatial Data Abstraction Library” es una biblioteca de traducción para formatos de datos geoespaciales vectoriales y ráster, publicada bajo una licencia de código abierto por la Fundación Geoespacial de Código Abierto (OSGeo por sus siglas en ingles). Como biblioteca, presenta un modelo de datos abstractos de un solo ráster y un modelo de datos abstractos de un solo vector a la aplicación de llamada para todos los formatos admitidos. También viene con una variedad de utilidades de línea de comandos útiles para la traducción y el procesamiento de datos (*gdal.org*, 2020). Principalmente se utilizó para crear, abrir, proyectar y recortar archivos GTiff.
- Glob: Este módulo encuentra todos los nombres de ruta que coinciden con un patrón específico de acuerdo con las reglas utilizadas por el shell de Unix, aunque los resultados se devuelven en orden arbitrario (*docs.python.org/2/library/glob.html*, 2020). Se utilizó

para encontrar y filtrar archivos o imágenes dentro de un directorio que coincidiera con ciertas características como la fecha.

- NetCDF4: “Network Common Data Form” es un módulo que puede leer y escribir archivos tanto en el nuevo formato NetCDF4 como en el antiguo NetCDF3 (*unidata.github.io/netcdf4-python*, s/f). Este módulo se utilizó para la apertura de las imágenes de GOES16/ABI.
- NumPy: “Numerical Python” es una biblioteca de Python que proporciona un objeto de matriz multidimensional, varios objetos derivados (como matrices y matrices enmascaradas) y una variedad de rutinas para operaciones rápidas en matrices, que incluyen manipulación matemática, lógica, de formas, clasificación, selección, transformadas discretas de Fourier, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria y mucho más (*numpy.org*, 2020). Todos dos datos extraídos de los archivos NetCDF se almacenaron en matrices NumPy para su procesamiento.
- Pandas: Es una biblioteca que proporciona estructuras de datos rápidas, flexibles y expresivas diseñadas para que el trabajo con datos "relacionales" o "etiquetados" sea fácil e intuitivo. Su objetivo es ser el bloque de construcción fundamental de alto nivel para realizar análisis de datos prácticos del mundo real en Python (*pandas.pydata.org*, 2020). Esta biblioteca se utilizó para almacenar los datos como si fueran tablas, con la posibilidad de hacerlos datos espaciales con la ayuda de GeoPandas.
- Rasterstats: Es un módulo de Python para resumir datasets ráster geoespaciales basados en geometrías vectoriales. Incluye funciones para estadísticas zonales y consultas de puntos interpolados (*pythonhosted.org/rasterstats*, s/f). Con este módulo se obtuvieron algunas estadísticas de los archivos de formato raster.
- Requests: Es una biblioteca HTTP de Python con el objetivo de hacer solicitudes HTTP simples y amigables (*requests.readthedocs.io/en/master*, s/f). Se utilizó para la descarga del producto FIRMS.

- SciPy: Es una biblioteca libre y de código abierto para Python. Se compone de herramientas y algoritmos matemáticos, contiene módulos para optimización, álgebra lineal, integración, interpolación, funciones especiales, procesamiento de señales y de imagen y otras tareas para la ciencia e ingeniería (*scipy.org*, 2020). Con esta biblioteca se realizó la interpolación segmentaría.
- Shapely: Es una biblioteca de Python para la manipulación y el análisis de objetos geométricos planos. Shapely no se preocupa por los formatos de datos o los sistemas de coordenadas, pero puede integrarse fácilmente con paquetes que sí lo son (*pypi.org/project/Shapely*, 2020). Fue de gran ayuda para la creación de los archivos .shp.
- Zipfile: Este módulo proporciona herramientas para crear, leer, escribir, agregar y listar un archivo ZIP (*docs.python.org/3/library/zipfile.html*, 2020). Con este módulo se descomprimieron los archivos FIRMS.

3.2. Procesos aplicados a datos ráster

3.2.1 Exploración del NetCDF

La apertura de los archivos NetCDF de GOES16/ABI en Python se realizó con el módulo netCDF4, a continuación se muestra como sería la apertura de la banda 7 del día 11 de mayo del 2020 (número de día 132) L2+ CMIP FullDisk. Este módulo no viene instalado junto con Python por defecto por lo que se deberá instalar aparte utilizando el administrador de paquetes de preferencia.

```
>>> from netCDF4 import Dataset  #Importando el módulo
>>> ds = Dataset("../OR_ABI-L2-CMIPF-
M6C07_G16_s20201321900146_e20201321909465_c20201321909539.nc")
```

Si no cuenta con una imagen NetCDF y para objetivos prácticos puede descargar una de la página http://home.chpc.utah.edu/~u0553130/Brian_Blaylock/cgi-bin/goes16_download.cgi. Esta página facilita la exploración y descarga del conjunto de datos de GOES en Amazon Web Services (AWS).

Al abrir la imagen, la variable donde se guarda el objeto es de tipo *netCDF4._netCDF4.Dataset,* en este caso es llamada *ds* y contiene atributos y métodos de la banda que podemos explorar escribiendo *ds.atributo/método()*. Podemos obtener una breve descripción de la banda, la institución responsable de su creación, su nombre y propiedades como el modo de escaneo, el nivel de procesamiento, la escena y la resolución espacial:

```
>>> ds.summary
'Single emissive band Cloud and Moisture Imagery Products are digital maps of clouds, moisture and atmospheric windows at IR bands.'
>>> ds.institution
"DOC/NOAA/NESDIS > U.S. Department of Commerce, National Oceanic and Atmospheric Administration, National Environmental Satellite, Data, and Information Services"
>>> ds.dataset_name
'OR_ABI-L2-CMIPF-M6C07_G16_s20201321900146_e20201321909465_c20201321909539.nc'
>>> ds.orbital_slot
'GOES-East'
>>> ds.timeline_id
'ABI Mode 6'
>>> ds.processing_level
'National Aeronautics and Space Administration (NASA) L2'
>>> ds.keywords
'SPECTRAL/ENGINEERING > INFRARED WAVELENGTHS > BRIGHTNESS TEMPERATURE'
>>> ds.scene_id
'Full Disk'
>>> ds.spatial_resolution
'2km at nadir'
```

Para obtener información del producto CMI que abrimos utilizaremos el método *ds.variables* que devuelve un diccionario, por lo tanto, para acceder a las llaves lo haremos por *ds.variables.keys(), así* podemos obtener también el número de bandas.

```
>>> ds.variables.keys()
['CMI', 'DQF', 't', 'y', 'x', 'time_bounds', 'goes_imager_projection', 'y_image', 'y_image_bounds', 'x_image',
'x_image_bounds', 'nominal_satellite_subpoint_lat', 'nominal_satellite_subpoint_lon', 'nominal_satellite_height',
'geospatial_lat_lon_extent', band_wavelength', 'band_id', 'total_number_of_points', 'valid_pixel_count',
'outlier_pixel_count', 'min_brightness_temperature', 'max_brightness_temperature',
'mean_brightness_temperature', 'std_dev_brightness_temperature', 'planck_fk1', 'planck_fk2', 'planck_bc1',
'planck_bc2', 'algorithm_dynamic_input_data_container', 'percent_uncorrectable_GRB_errors',
'percent_uncorrectable_L0_errors', 'earth_sun_distance_anomaly_in_AU',
'processing_parm_version_container', 'algorithm_product_version_container', 'esun', 'kappa0',
'focal_plane_temperature_threshold_exceeded_count', 'maximum_focal_plane_temperature',
'focal_plane_temperature_threshold_increasing', 'focal_plane_temperature_threshold_decreasing']
>>> str(ds.variables["band_id"][0].data) #número de banda en formato string
'7'
```

De esa misma manera podemos acceder a la variable CMI. Al imprimirla en consola nos mostrará los valores de las esquinas, como es la escena de FullDisk y el valor de relleno es 65535 nos imprimirá solamente este valor. Si guardamos esta consulta en una variable, esta será de tipo *numpy.ndarray*, que es como una matriz.

```
>>> data = ds.variables['CMI'][:].data #Guardando la variable CMI en una variable "data"
>>> data
array([[65535., 65535., 65535., ..., 65535., 65535., 65535.],
       [65535., 65535., 65535., ..., 65535., 65535., 65535.],
       [65535., 65535., 65535., ..., 65535., 65535., 65535.],
       ...,
       [65535., 65535., 65535., ..., 65535., 65535., 65535.],
       [65535., 65535., 65535., ..., 65535., 65535., 65535.],
       [65535., 65535., 65535., ..., 65535., 65535., 65535.]],
      dtype=float32)
>>> type(data) #Tipo de dato
```

```
numpy.ndarray
```

La matriz *numpy.ndarray* contendiene los valores de cada pixel y el objeto *data* ya no tiene georreferencia en ningún sistema coordenado. Las coordenadas de los pixeles pasan a ser locales y van de cero al número de pixeles, se accede a ellos como data[fila, columna] ó dicho de otra forma data[y, x].

Para conocer el tamaño de la matriz se utiliza:

```
>>> data.shape
(5424, 5424)
```

Como es una imagen de FullDisk contiene 5424 pixeles en Y y X. Para ubicar un dato en la matriz se puede utilizar el número de fila, columna. La matriz data tiene las posiciones que se muestran en la Figura 3.2.1.

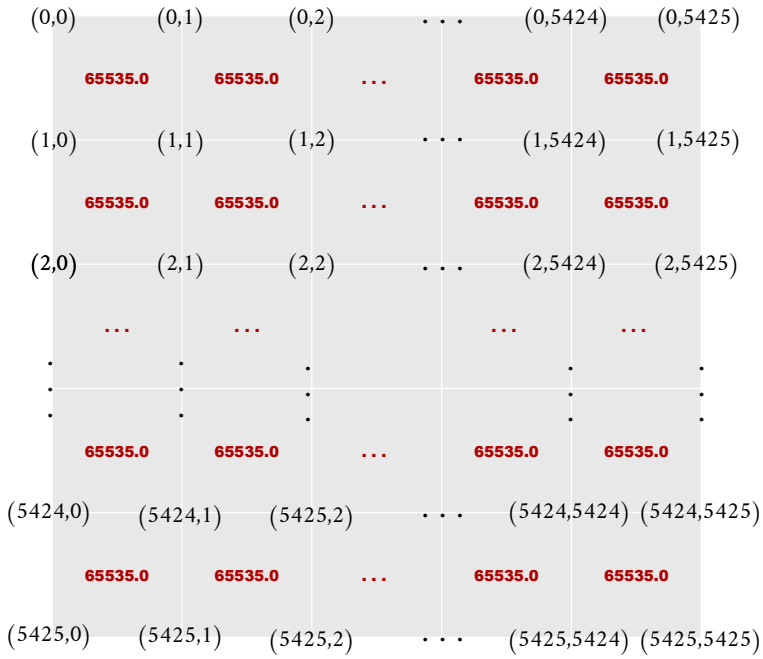

Figura 3.2.1: Posiciones de una matriz NumPy. Elaboración propia

Para eliminar los valores nulos de la matriz, como el 65535.0 y datos que son errores sobre la imagen, utilizaremos el rango válido de valores de cada banda de GOES-16 que se puede consultar en la Tabla 1.4.3 o también se pueden extraer estos valores del Dataset, de las variables *"min_brightness_temperature"* y *"max_brightness_temperature"* de la siguiente manera:

```
>>> vmin = ds["min_brightness_temperature"].valid_range[0]    #valor mínimo
>>> vmax = ds["max_brightness_temperature"].valid_range[1]    #valor máximo
>>> vmin, vmax
(197.31, 411.86)
```

Los valores fuera de ese rango se les asignó el valor *NoData* de NumPy. Para esto aplicaremos función *where* de NumPy que sirve para condicionar los datos de una matriz. Se utiliza el operador “or” (|) y el formato es el siguiente:

numpy.where((matriz > valor_máximo) | (matriz < valor_mínimo), si cumple, si no cumple)

```
>>> import numpy as np    #importando la librería como "np"
>>> data = np.where((data>vmax) | (data<vmin), np.nan, data)
>>> data
array([[nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       ...,
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan]], dtype=float32)
```

Para generar la diferencia de las bandas 7 menos 14, se puede hacer una simple resta de matrices, es la ventaja de NumPy.

```
>>> dataDif = dataC07 – dataC14
```

3.2.2 Creación de un archivo GeoTIFF

Cada píxel de la imagen se puede identificar mediante un ángulo a una determinada altura sobre la Tierra. El satélite produce ángulos en la dirección X y Y, al mismo tiempo que nos brinda información sobre la altura del instrumento. Estos parámetros nos permitirán aproximar las coordenadas de cada píxel en cualquier sistema coordenado. La extensión de la imagen y la altura del satélite se pueden obtener con las siguientes variables de Dataset.

```
>>> ds.variables['x_image_bounds'][:].data    #para x
array([0.151872, -0.151872], dtype=float32)
>>> ds.variables['y_image_bounds'][:].data    #para y
array([-0.151872, 0.151872], dtype=float32)
>>> H = ds.variables['goes_imager_projection'].perspective_point_height    #altura satelital
```

```
>>> H
35786023.0
```

La coordenada de proyección se relaciona con el ángulo de exploración mediante la siguiente relación:

angulo_escaneo(radianes) = coordenada / altura_satelital

Despejando la coordenada:

coordenada = angulo_escaneo(radianes) * altura_satelital

Estos dos datos ya los conocemos, por lo tanto podemos obtener las coordenadas de la extensión de la imagen en coordenadas geoestacionarias.

```
>>> xmin = ds.variables['x_image_bounds'][0] * H
>>> xmax = ds.variables['x_image_bounds'][1] * H
>>> ymin = ds.variables['y_image_bounds'][0] * H
>>> ymax = ds.variables['y_image_bounds'][1] * H
>>> (xmin, ymin, xmax, ymax)
(-5434894.675269887, 5434894.675269887, 5434894.675269887, -5434894.675269887)
```

Para obtener la resolución en X se resta la X máxima menos la X mínima y se divide entre el número de pixeles, así mismo con la resolución en Y.

```
>>> nx = data.shape[1]    #Número de pixeles en X
>>> ny = data.shape[0]    #Número de pixeles en Y
>>> xres = (xmax - xmin) / float(ny)    #Resolución en X
>>> yres = (ymax - ymin) / float(nx)    #Resolución en Y
>>> (xres, yres)
(2004.0172106452385, -2004.0172106452385)
```

La resolución en Y es negativa por que el origen de las matrices en NumPy es arriba a la izquierda, a diferencia del plano cartesiano que es abajo a la izquierda.

Para crear una imagen a partir de una matriz Numpy necesitamos las coordenadas de origen (esquina superior izquierda) la rotación de los pixeles y el tamaño de pixel. Se utiliza la función *SetGeoTransform* de *gdal* para establecer estos valores y luego se define la proyección con el módulo *osr*. Posteriormente se escriben los datos de la matriz sobre el archivo GeoTIFF creado.

```
>>> from osgeo import gdal, osr    #Importando las bibliotecas
>>> geotransform = (xmin, xres, 0, ymin, 0, yres)    #Datos de la geotransformación
>>> dst_ds = gdal.GetDriverByName('Gtiff').Create("…/output_name.tif", ny, nx, 1,
gdal.GDT_Float32)    #Creando el Dataset vació
>>> dst_ds.SetGeoTransform(geotransform)    #Aplicando la geotransformación al Dataset vació
>>> srs = osr.SpatialReference()    #Creando un sistema de referencia espacial vació
>>> srs.ImportFromProj4('+proj=geos +h=35786023.0 +ellps=GRS80 +lat_0=0.0 +lon_0=-75.0
+sweep=x +no_defs')    #Asignándole la proyección geoestacionaria al objeto "srs"
>>> dst_ds.SetProjection(srs.ExportToWkt())    #Definiendo el srs al Dataset del tif.
>>> dst_ds.GetRasterBand(1).WriteArray(data)    #Escribiendo los datos al Dataset del tif.
>>> dst_ds.FlushCache()    #Vacía los datos almacenados en caché de escritura en el disco.
>>> dst_ds = None    #Cierra la variable del Dataset del tif
```

3.2.3 Procesos aplicados a un GeoTIFF

Una vez creados los archivos GeoTIFF se pueden recortar, re-proyectar y manipular con más sencillez, esto es gracias a la librería Gdal. Para hacer un recorte necesitamos abrir el archivo GeoTIFF y se utiliza la función *gdal.Translate,* que sirve para convertir datos ráster en diferentes formatos, en este caso lo convertiremos nuevamente a un formato GeoTIFF pero además de eso permite algunas operaciones extras como re-muestreo, cambio de escala y recorte. Para lo último necesitamos las coordenadas de los límites del recorte deseado en la opción *-projWin:*

```
>>> ds_tif = gdal.Open("…/input_name.tif")    #Abre .tif con gdal
```

```
>>> coords = [-3700000.0, 3396435.0, 1200000.0, 750000.0]  #Coordenadas del recorte [W, N, E, S]
>>> gdal.Translate("../output_name.tif", ds_tif, options=gdal.TranslateOptions(projWin = coords,
noData = np.nan))  #Hace el recorte del Geotiff
```

Para reproyectar un GeoTIFF con Gdal se ocupa la función *gdal.Warp.* Es una utilidad de creación de mosaicos, reproyección y deformación de imágenes, puede reproyectar a cualquier proyección compatible. Las variables de entrada que necesita, es la ruta y nombre de salida del GeoTIFF, el Dataset abierto anteriormente con Gdal, el EPSG de salida, y opcionalmente el tamaño de pixel de salida. El siguiente ejemplo es para reproyectar a coordenadas geográficas (EPSG:4326) un GeoTIFF (ds_tif):

```
>>> gdal.Warp(.../output_name.tif, ds_tif, options=gdal.WarpOptions(dstSRS="EPSG:4326",
yRes=0.018, xRes=0.018, dstNodata=np.nan))
```

Para hacer la reclasificación de las bandas basándonos en los umbrales utilizaremos nuevamente la función *where* de Numpy. Una vez que tenemos los recortes reclasificados en coordenadas geográficas de la banda 7, la diferencia 7- 14 y las máscaras de agua, suelo desértico y manchas urbanas con los mismos límites y en el mismo sistema coordenado, procederemos a crear la matriz de pixeles potenciales. Esto se hace abriendo los datos mencionados y haciendo una simple multiplicación de las 5 matrices.

3.2.4 Prueba de contexto

Una vez que tenemos la matriz de pixeles potenciales, procederemos a aplicar la prueba estadística contextual a cada pixel potencial, estos tienen el valor uno. Por lo tanto, el primer paso es extraer las posiciones de los pixeles potenciales, para esto utilizaremos la función *where* nuevamente:

```
>>> pos = np.where(dataPP == 1)
>>> rows = pos[0]  #Posiciones en Y
```

```
>>> colums = pos[1]    #Posiciones en X
```

Con las posiciones de los pixeles potenciales crearemos un kernel de 7x7 pixeles a cada pixel, donde el pixel potencial quedará en el centro del kernel, supongamos el pixel potencial está en la fila 10 y columna 10, esta posición es la de arriba a la izquierda del pixel, por lo tanto, si queremos hacer un kernel de 7x7 pixeles debemos restar 3 posiciones y sumar 4 a la posición del pixel potencial, este ejemplo se muestra en la Figura 3.2.2

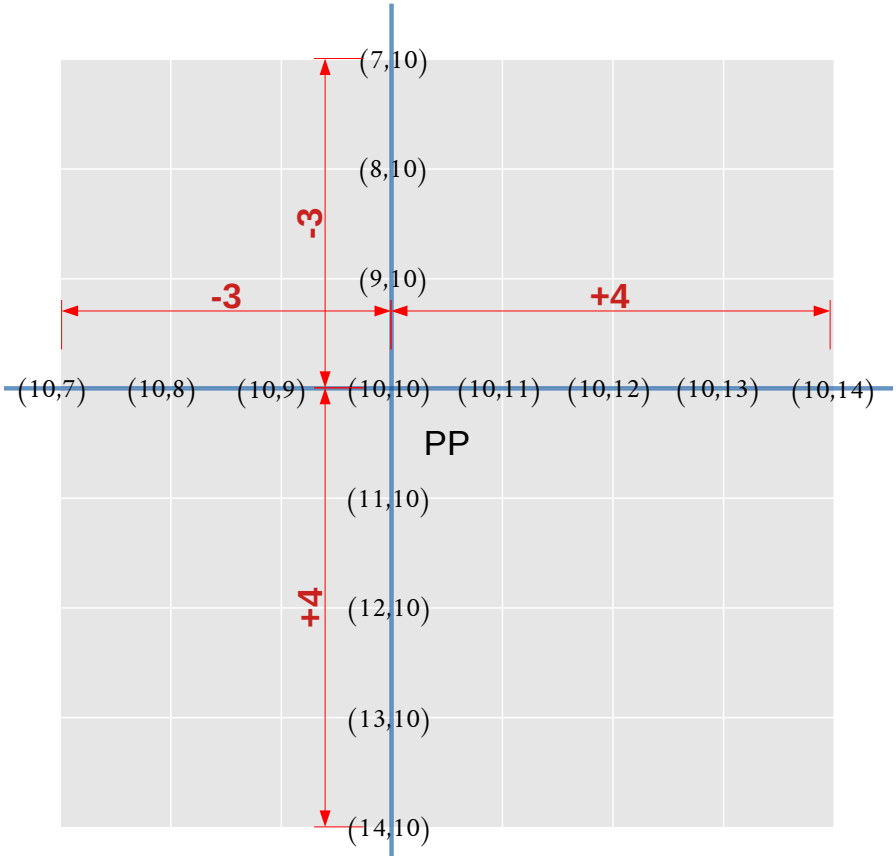

Figura 3.2.2: Ejemplo de kernel de 7x7 con el pixel potencial al centro

Para hacer esto para cada pixel potencial de la imagen, recorreremos las listas *rows* y *columns* en un ciclo for simultáneamente para tener pares ordenados, para esto se utiliza la función *zip()* de la siguiente manera:

```
>>> for p in zip(rows,columns):
...
```

Como se menciona en el capítulo 2.2.5. Prueba de contexto, se necesita un kernel para las estadísticas de la banda 7 (*dataC07*) y uno para las estadísticas de la diferencia (*dataDIF*). Estos se pueden generar de la siguiente manera y con el siguiente formato:

data[y_min : y_max , x_min : x_max]

La variable *p* tendrá las posiciones (Y,X) de cada uno de los pixeles potenciales, por lo tanto se puede acceder a la Y con *p[0]* y a la X con *p[1]*:

```
...     kernelC07 = dataC07[p[0]-3 : p[0]+4 , p[1]-3 : p[1]+4]
...     kernelDIF = dataDIF[p[0]-3 : p[0]+4 , p[1]-3 : p[1]+4]
```

El tipo de dato de los dos kernel´s es *numpy.ndarray* y una función de este objeto permite extraer estadísticas de la matriz, en este caso necesitamos la media y la desviación estándar, que obtendremos de la siguiente forma:

```
...     meanKC07 = kernelC07.mean()    # Media del kernel del canal 7
...     meanKDIF = kernelDIF.mean()    # Media del kernel de la diferencia
...     stdKC07 = kerneC07.std()    # Desviación estándar del kernel del canal 7
...     stdKDIF = kerneDIF.std()    # Desviación estándar del kernel de la diferencia
```

El valor del pixel potencial en la banda 7 y diferencia se obtiene con la posición de la siguiente manera:

```
...     ppC07 = dataC07[P[0], P[1]]
```

```
...     ppDIF = dataDIF[P[0], P[1]]
```

Con las variables anteriores ya podemos aplicar la condicional de la prueba de contexto como se muestra a continuación:

```
...     if ( ppC07 > ( meanKC07 + 2.5*stdKC07)) and ( ppDIF > (meanKDIF + 2.5*stdKDIF)):
...             dataPP[p] = 1    # Se confirman como incendios asignándole el valor 1 nuevamente
...     else:
...             dataPP[p] = 0    # Si no cumplen la condición se les asigna el valor 0 y se descartan
>>> dataPC = dataPP    # Se crea la nueva variable ó matriz de datos de puntos de calor.
```

3.3. Procesos aplicados a datos vectoriales

3.3.1 Descarga de los puntos FIRMS/VIIRS

El primer paso es la obtención de los datos de FIRMS/VIIRS de su sitio de internet, para esto se puede utilizar la librería Requests:

```
>>> url = 'https://firms.modaps.eosdis.nasa.gov/data/active_fire/suomi-npp-viirs-c2/shapes/zips/SUOMI_VIIRS_C2_Central_America_24h.zip'    # URL de los datos
>>> myfile = requests.get(url)    # Haciendo la solicitud de los datos
>>> path = "../datos_aux/vectorial/puntos_firms/temp/"    #Directorio de salida
>>> open(path+"output_name.zip", 'wb').write(myfile.content)    #Escribe el contenido de la solicitud
>>> myfile.close()    #Terminando la solicitud
```

El archivo descargado está comprimido, por lo tanto, para descomprimirlo y poder usarlo se usa el módulo zipfile:

```
>>> import zipfile
>>> zf = zipfile.ZipFile(path+"file_name.zip", "r")    #Abre el archivo .zip en modo lectura "r"
>>> for i in zf.namelist():
```

```
. . .    zf.extract(i, path = path)    #Extrae cada uno de los archivos en el directorio "path"
>>> zf.close()    #Cierra el archivo .zip
```

3.3.2 Apertura y exploración de un Shapefile

Para la apertura del archivo .shp se utiliza la librería GeoPandas, si sabemos cuáles son las columnas de la fecha se lo podemos indicar con el parámetro *parce_dates,* la variable donde se guarda el archivo es de tipo *geopandas.geodataframe.GeoDataFrame.* Como ejemplo abriremos un archivo de puntos FIRMS/VIIRS del día 11 de mayo de 2020.

```
>>> import geopandas as gpd    #Importando geopandas como gpd
>>> gdf_firms = gpd.read_file(path+"SUOMI_VIIRS_C2_Central_America_24h.shp",
parse_dates=[[5,6]])    #Apertura de un archivo .shp
>>> type(gdf_firms)
geopandas.geodataframe.GeoDataFrame
```

El GeoDataFrame contiene toda la información espacial de los puntos FIMS/VIIRS y se puede explorar sus atributos como las columnas que tiene escribiendo:

```
>>> gdf_firms.columns
Index(['LATITUDE', 'LONGITUDE', 'BRIGHT_TI4', 'SCAN', 'TRACK', 'ACQ_DATE', 'ACQ_TIME',
       'SATELLITE', 'CONFIDENCE', 'VERSION', 'BRIGHT_TI5', 'FRP', 'DAYNIGHT', 'geometry'],
       dtype='object')
```

Todos los objetos de tipo GeoDataFrame necesitan la columna "geometry", ya que es la que almacena las propiedades espaciales del objeto, ya sea un punto, línea o polígono, cada objeto tiene una geometría única.

```
>>> gdf_firms.geometry
0       POINT (-71.41564 18.46417)
1       POINT (-70.17277 11.74871)
```

```
.
1231     POINT (-99.28975 20.04627)
1232    POINT (-103.33906 20.63925)
Name: geometry, Length: 1233, dtype: geometry
```

El GeoDataFrame contiene los puntos de las últimas 24 horas, pero incluye por lo regular, puntos de dos días distintos, esto lo podemos consultar con el siguiente comando:

```
>>> gdf_firms['ACQ_DATE'].unique()
array(['2020-09-23', '2020-09-24'], dtype=object)
```

Para este caso, el producto FIRMS/VIIRS fue descargado el día 2020-09-25 y contiene puntos del final del día 23 y todos los del día 24, como solamente nos interesa el último día, aplicaremos un filtro donde solo estarán los puntos del último día registrado:

```
>>> test_date = gdf_firms['ACQ_DATE'].iloc[-1]    #Obteniendo la fecha
>>> test_date
'2020-09-24'
```

Una vez que tenemos la fecha podemos eliminar los datos que no coincidan con esta, eso se puede hacer con un ciclo iterando sobre las filas con la función *iterrows* y aplicando una condicional para que los datos que no tengan la fecha *test_date* sean eliminados:

```
>>> for index, row in gdf_firms.iterrows():    #Ciclo para iterar sobre las filas
...     if row["ACQ_DATE"] != test_date:    #Aplicando la condicional
...             gdf_firms.drop(index, inplace=True)    #Sí la condicional es correcta elimina el punto
```

3.3.3 Procesos al GeoDataFrame

Para hacer un buffer se utiliza la función *gdf.buffer* y el parámetro *cap_style* define el tipo de geómetra de salida: 1 para redondo, 2 para plano y 3 para cuadrado:

```
>>> gdf["new_geom"] = gdf.geometry.buffer(0.0025, cap_style=3)    # buffer cuadrado
```

Posteriormente, debemos definir esa nueva columna como la geometría de la siguiente forma:

```
>>> gdf.set_geometry(col="new_geom", inplace=True, drop=True)
```

Crear un archivo de Shapefile es fácil si se tiene el GeoDataFrame, lo primero que tenemos que hacer es definir el sistema coordenado de referencia (CRS por sus siglas en inglés) y se puede escribir en formato proj4:

```
>>> gdf_firms.crs = "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"    #Definiendo el CRS
>>> gdf_firms.to_file("../output_name.shp", driver='ESRI Shapefile')    #Creando un archivo Shapefile a partir de un GeoDataFrame
```

3.4. Automatización del algoritmo

Para que la ejecución de los scripts sea automática en un servidor GNU/Linux Debian se ocupó Cron y Crontab. Cron es un administrador o planificador de tareas en segundo plano (demonio ó daemon en inglés) que ejecuta procesos a intervalos regulares (cada minuto, hora, día, semana, mes, etc.). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en un fichero o archivo de texto plano llamado Crontab. (*Cron (UNIX)*, s/f)

Para listar y agregar tareas al fichero Crontab se escribe lo siguiente sobre la terminal de GNU/Linux:

```
:~$ crontab -l    #Lista las tareas existentes
:~$ crontab -e    #Edita el fichero Crontab
```

Cada línea dentro del fichero Crontab que no sea un comentario (#) será una tarea o Cron. La sintaxis del intervalo de tiempo para la ejecución es la siguiente:

m h d M D comando

Dónde:

m = Corresponde al minuto en que se va a ejecutar el script (0-59)

h = Es la hora exacta, se maneja el formato de 24 horas (0-23), siendo 0 las 12:00 de la medianoche.

d = Hace referencia al día del mes (1-31)

M = Es el mes del año (1-12)

D = Significa el día de la semana (0 a 7), donde 0 y 7 son domingo

comando = Es el comando con la ruta absoluta del script a ejecutar.

Tomando en cuenta esto último, el script s1_umbrales_auto.py que se encarga de la creación de los umbrales diarios y se ejecuta todos los días a las 00:00hrs, el comando sería el siguiente:

```
0 0 * * *       s1_umbrales_auto.py
```

El script que genera los puntos de calor s2_incendios_auto.py, se ejecuta todos los días del año cada 10 minutos, esto se lo logra con el siguiente comando.

```
*/10 * * * *    s2_incendios_auto.py
```

El script que genera los archivos csv y shp del día completo se ejecuta todos los días a las 00:20:00 hrs con el siguiente comando.

```
0 20 * * *      s3_resultados_csv_shp_xdia_auto.py
```

4. Resultados

El algoritmo de detección de incendios forestales para México utilizando imágenes satelitales GOES 16/ABI se programó y automatizó en un conjunto de 12 scripts escritos en lenguaje de programación Python (ver 4.1 Scripts). Cada script estructurado en funciones, con el objetivo de que sea mas fácil realizar cambios en futuras versiones del algoritmo en un determinado proceso.

El algoritmo se encuentra operativo en su versión más reciente desde el día 19 de marzo de 2020 y se ejecuta en el servidor llamado Kawak del Laboratorio Nacional para la Observación de la Tierra (LANOT) del Instituto de Geografía de la UNAM.

El servidor Kawak cuenta con un procesador Intel(R) Xeon(R) Gold 6128, 3.40GHz, 257GB de memoria ram y como sistema operativo GNU/Linux Debian 4.19.98-1. El proceso de obtención de los puntos de calor, se ejecuta en un tiempo promedio de 1.2 minutos en este servidor.

El resultado se crea en dos formatos vectoriales, CSV y Shapefile con la misma información en ambos formatos, el resultado de cada 10 minutos tiene el nombre GIM10_PC_YYYYMMDDHHMM, el que contiene los puntos de calor de todo el día GIM10_PC_YYYYMMDD y uno con los puntos de las últimas 24hrs que se actualiza cada 10 minutos, por lo tanto solo existe uno y tiene el nombre GIM10_PC_24hrs, la hora que se menciona en los nombres se encuentra en la zona horaria UTC (Universal Time Coordinated).

- **YYYY**: Año.
- **MM**: Mes.
- **DD**: Día.
- **HH**: Hora.
- **MM**: Minutos.

Cada uno de los archivos contiene las columnas que se muestran en la Tabla 1.2.1

Nombre	Descripción	Rango de datos validos
lon	Longitud del PC	(-119°,-80°)
lat	Latitud del PC	(10°,34°)
Satelite	Satélite con el que se detecto	-
BT_c07	Temperatura de brillo en el canal 7	-
BT_c14	Temperatura de brillo en el canal 14	-
dif_c07-c14	Diferencia de temperatura de brillo (canal 7 – canal 14)	-
Fecha	Fecha de detección del PC	-
Hora	Hora de detección del PC	-
Land_Cover	Cubierta terrestre sobre la que cayo el punto de calor (datos de clasificación IGBP anual)	(1,17)
Estado	Estado donde esta el PC	-
Pais	País donde esta el PC	-
ANP	Are Natural Protegida sobre la que esta el punto de calor, en caso de estarlo (datos de CONABIO).	-
freq	Frecuencia de detecciones durante el día de ese mismo PC	(1 - 144)
freq_norm	Frecuencia normalizada: Porcentaje de detecciones durante el día de ese mismo PC	(0 - 100%)

Tabla 4.1: Atributos del producto de incendios GOES-16/ABI, datos de las ANP proporcionados por el CONANP

Durante el periodo del 19 de marzo al 19 de septiembre de 2020, el algoritmo ha detectado 217,235 puntos de calor en México para sus distintos horarios en 33,894 ubicaciones distintas. Los meses de mayor detección fueron marzo, abril, mayo y junio. Mayo concentró el mayor número de registros; las detecciones disminuyeron notablemente a partir del mes de julio como se muestra en la Figura 4.1, coincidiendo con la temporada de incendios que menciona CENAPRED en México.

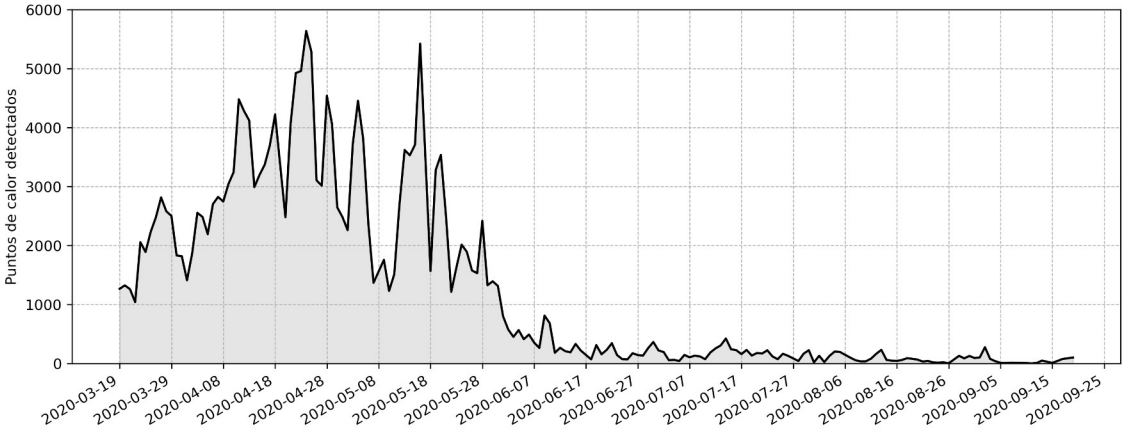

Figura 4.1: Número de puntos de calor (PC) detectados del día 19 de marzo al 19 de septiembre de 2020.

La distribución de los puntos de calor detectados por estado en el mismo periodo de tiempo se puede observar en la Figura 4.2. El estado con mayor número de detecciones fue Campeche con 53,776 puntos de calor detectados, seguido de Guerrero y Chiapas con 28,076 y 23,063 puntos de calor detectados respectivamente.

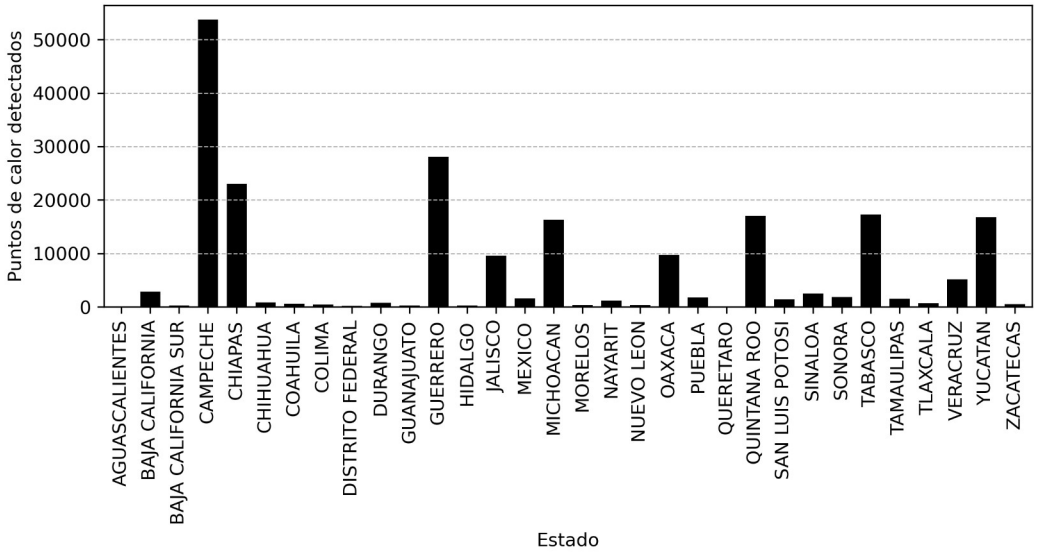

Figura 4.2: Puntos de calor detectados por estado del 19 de marzo al 19 de septiembre de 2020

El tipo de suelo o vegetación sobre el que se detectaron los puntos se puede observar en la Figura 4.3, según la clasificación anual de la cobertura terrestre de MODIS (IGBP) de 2018.

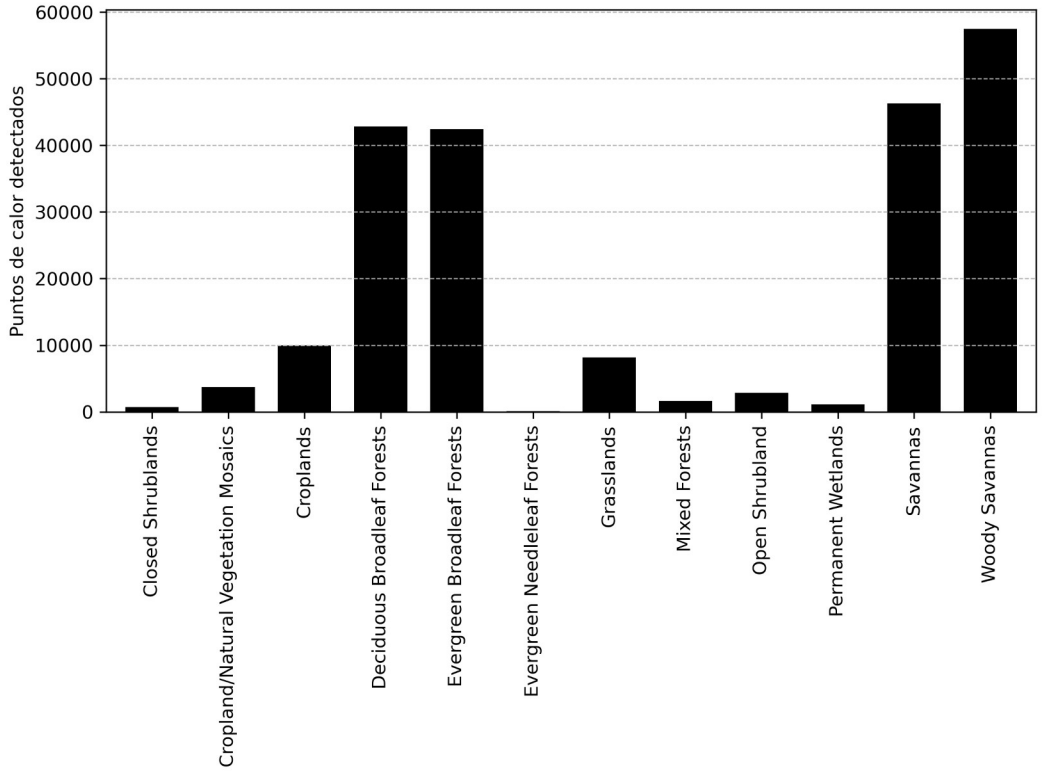

Figura 4.3: Puntos de calor detectados por tipo de vegetación del 19 de marzo al 19 de septiembre de 2020

Las Figura 4.4 nos muestran el comportamiento de los umbrales a través de los meses del año. Se observa una disminución en el umbral de $T_{3.9}$ de los horarios diurnos (15:00, 18:00 y 21:00 UTC) con respecto a los meses del año, esto es debido al cambio de temperatura superficial por las condiciones climáticas del mes en cuestión, haciendo más permisivo el algoritmo en los meses fríos y más estricto en los meses con altas temperaturas. Durante los horarios nocturnos se aprecian cambios en cada mes sin llegar a tener una tendencia clara, esto se debe a que por la noche la temperatura superficial no es tan variante durante todo el año como lo si lo es en el día.

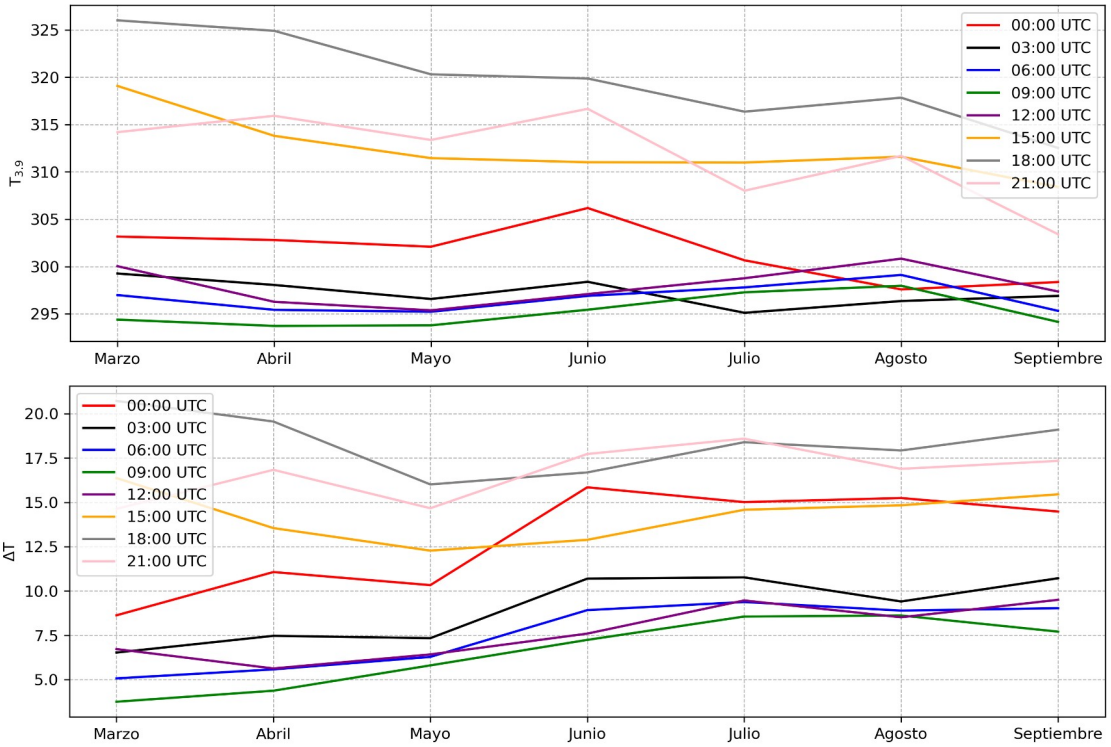

Figura 4.4: Promedio del umbral óptimo de $T_{3.9}$ y ΔT por mes en distintos horarios del día durante 6 meses (marzo a septiembre de 2020). Elaboración propia

El umbral óptimo para $T_{3.9}$ y ΔT no solo varía con respecto a los días del año, sino también respecto a la hora del día, en la Figura 4.5 se observa este comportamiento y como varía para cierta hora del día con respecto al mes, en general los umbrales más altos para $T_{3.9}$ se registran entre las 17:00 y 19:00 UTC (11:00 – 13:00 CDT) y los umbrales más bajos alrededor de las 10:00 a las 11:00 UTC (04:00 – 05:00 CDT).

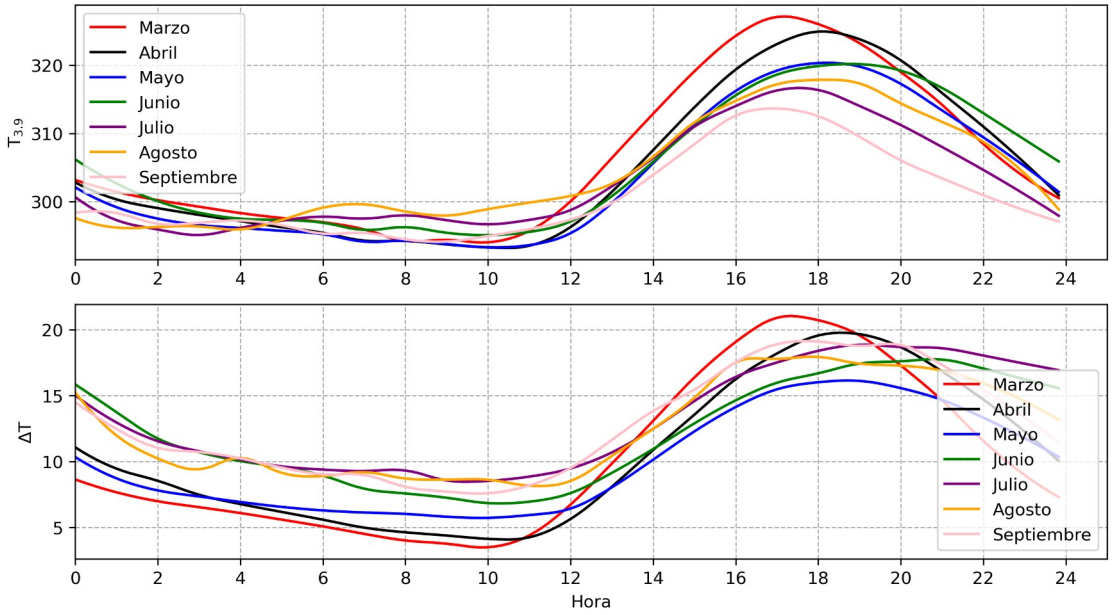

Figura 4.5: Promedio del umbral optimo de $T_{3.9}$ y ΔT por hora para cada mes de estudio, (marzo a septiembre de 2020). Elaboración propia

El producto final del algoritmo de incendios son datos vectoriales de tipo punto, pero estos se pueden plotear sobre distintos compuestos elaborados con las imágenes GOES-16/ABI como se muestra en la Figura 4.6 y en la Figura 4.7 se hace un acercamiento a una serie de incendios ocurridos en el estado de Guerrero (102.0W, 101.04°W, 17.4°N, 18.18°N). En el compuesto *true color* y *blue land* se pueden observar las columnas de humo de esa serie de incendios y en la banda 7 y el compuesto *fire temperature* se observa la diferencia de temperatura de los pixeles que presentaban incendios a la hora de la toma de esa imagen de un color más encendido.

Debido a que se plotearon los puntos de todo el día sobre la imagen de un horario en específico no todos los puntos de calor mostrados se ven claramente asociados a un pixel o columna de humo en los distintos compuestos, sin embargo, si se hiciera la representación de esos puntos sobre el compuesto del horario correspondiente se observaría la asociación

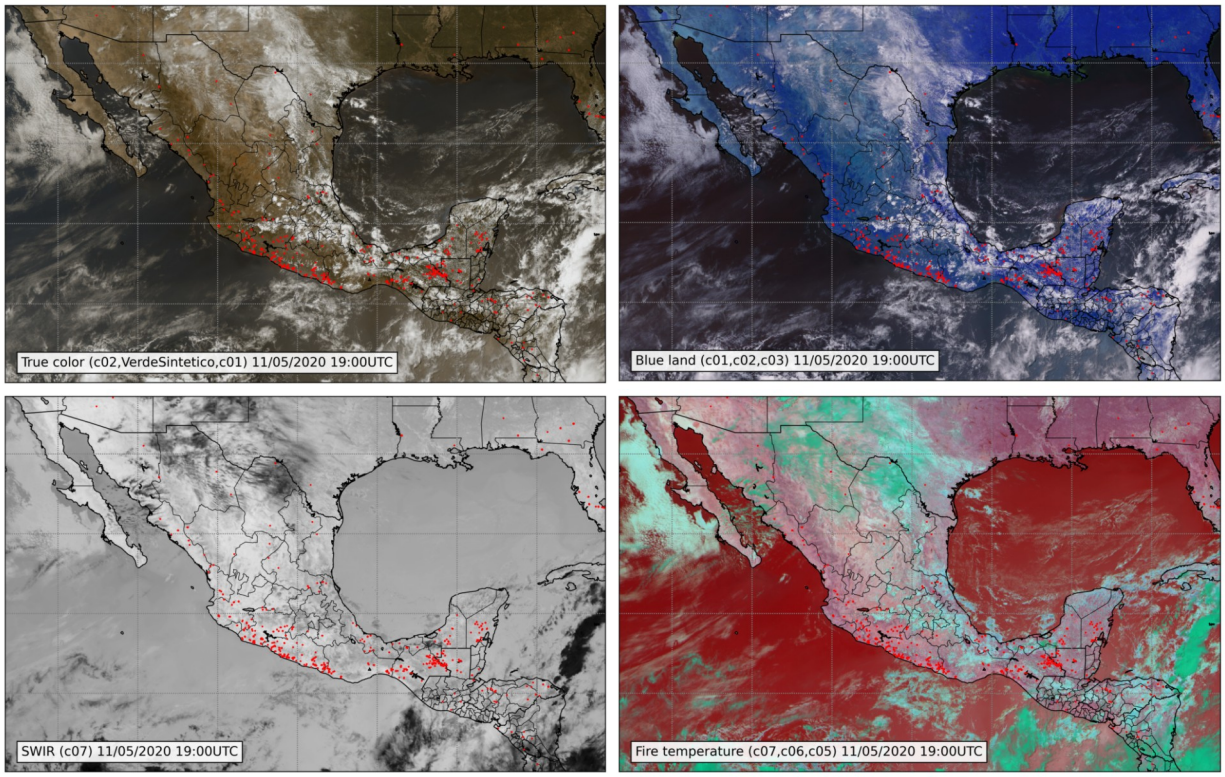

Figura 4.6: Puntos de calor detectados del día 11 de mayo de 2020 sobre 4 compuestos distintos donde se pueden observar los putos de calor

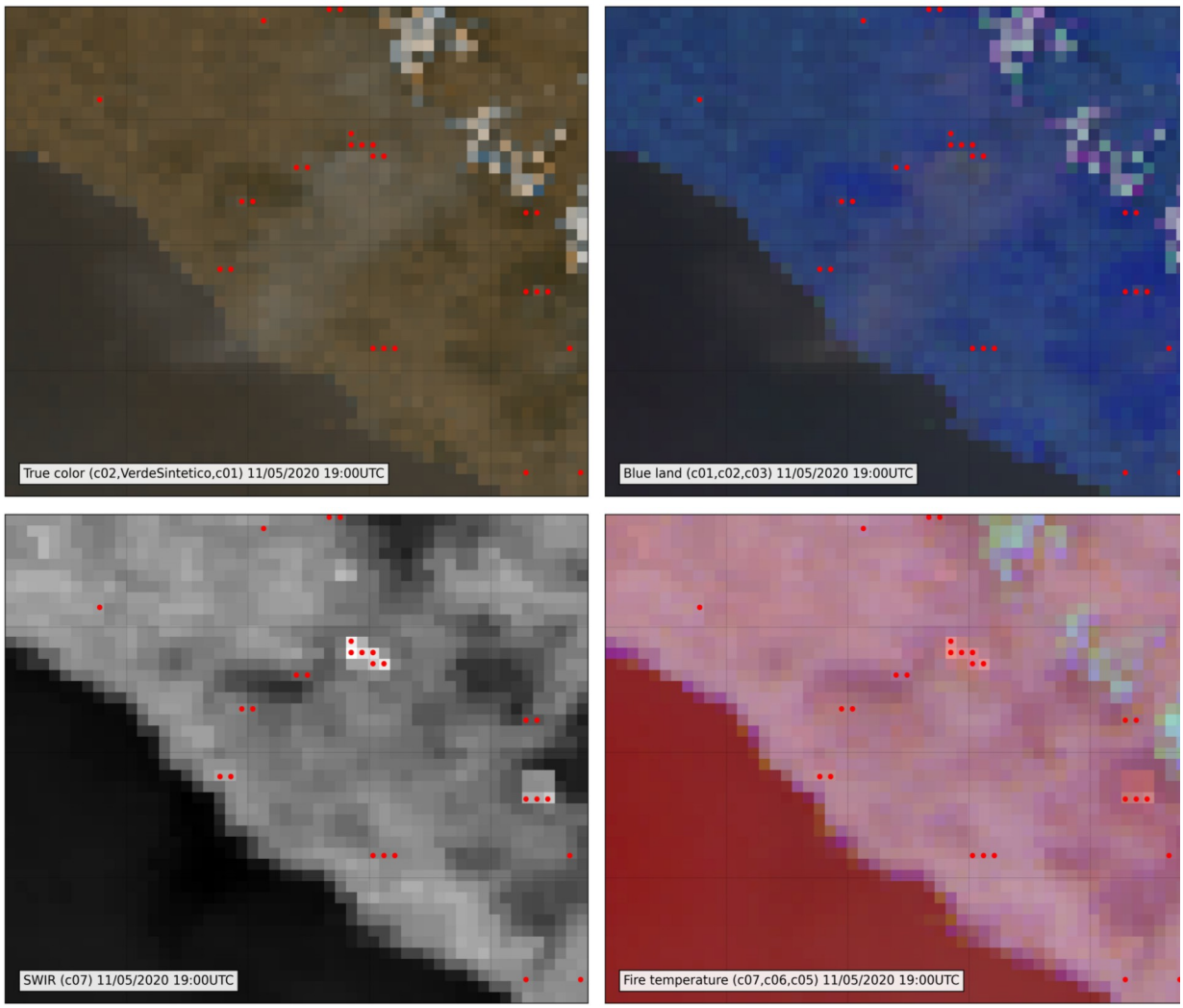

Figura 4.7: Acercamiento a una serie de incendios en el estado Guerrero de los puntos de calor detectados el día 11 de mayo de 2020 sobre 4 compuestos distintos de las 19:00hrs UTC del mismo día.

4.1. Scripts

s1_umbrales_auto.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: Colvert Gomez Rubio
"""

import shutil
import s1_1_crea_puntos_firms as s_cpf
import s1_2_netcdf_a_tif as s_n2t
import s1_3_crea_eventos_firms as s_cev
import s1_4_muestras_goes_xdia as s_mgd
import s1_5_crea_umbrales as s_cu

path_datos_aux = "/home/fires/datos_aux/"
path_netcdf_aws = path_datos_aux + "raster/temp_aws/"

path_puntos_firms = path_datos_aux + "vectorial/puntos_firms/"
path_puntos_firms_temp = path_datos_aux + "vectorial/puntos_firms/temp/"
year_nday = s_cpf.crea_puntos_firms(path_puntos_firms,path_puntos_firms_temp)

path_img_geos = path_datos_aux + "raster/img_goes_geos/"
path_img_geos_prueba = s_n2t.netcdf_a_tif(path_netcdf_aws, path_img_geos, year_nday)
#path_img_geos_prueba = "/home/fires/datos_aux/raster/img_goes_geos/2021/252/"

path_eventos_firms = path_datos_aux + "vectorial/eventos_firms/"
path_eventos_firms_filtrados = path_datos_aux + "vectorial/eventos_firms_filtrados/"
s_cev.crea_eventos_firms(path_puntos_firms, path_eventos_firms, path_eventos_firms_filtrados, year_nday)

path_muestras_goes_firms = path_datos_aux + "vectorial/muestras_goes_firms/"
s_mgd.muestras_goes_c07(path_eventos_firms_filtrados, path_img_geos_prueba, path_muestras_goes_firms, year_nday)
s_mgd.muestras_goes_dif07_14(path_eventos_firms_filtrados, path_img_geos_prueba, path_muestras_goes_firms, year_nday)
shutil.rmtree(path_img_geos_prueba, ignore_errors=True)

path_umbrales = path_datos_aux + "vectorial/umbrales/"
s_cu.crea_umbrales_c07(path_muestras_goes_firms, path_umbrales, year_nday)
s_cu.crea_umbrales_dif07_14(path_muestras_goes_firms, path_umbrales, year_nday)
```

s1_1_crea_puntos_firms.py

```
# -*- coding: utf-8 -*-
"""
@author: Colvert Gomez Rubio
"""

import os
from datetime import datetime as dt
import requests
import zipfile
import geopandas as gpd

def crea_puntos_firms(path_puntos_firms,path_puntos_firms_temp):
    """
    Descarga producto FIRMS VIIRS 375m / S-NPP, recorte de Centroamérica de las
    ultimas 24hrs, descomprime los datos y filtra los puntos del último día
    para generar el producto FIRMS de un único día, crea un shp con estos datos
    con el nombre V_FIRMS_YYYYMMDD.shp. Donde:
    YYYY: Año
    MM: Mes
    DD: Día
    """

    print("\nINICIA SCRIPT 's1_1_crea_puntos_firms.py")

    zip_file_name = "SUOMI_VIIRS_C2_Central_America_24h.zip"
    shp_file_name = "SUOMI_VIIRS_C2_Central_America_24h.shp"
```

```
    url = 'https://firms.modaps.eosdis.nasa.gov/data/active_fire/suomi-npp-viirs-c2/shapes/zips/SUOMI_VIIRS_C2_Central_America_24h.zip'
    file = requests.get(url)
    open(path_puntos_firms_temp + zip_file_name, 'wb').write(file.content)
    file.close()

    zf = zipfile.ZipFile(path_puntos_firms_temp + zip_file_name, "r")
    for i in zf.namelist():
        zf.extract(i, path = path_puntos_firms_temp)
    zf.close()

    gdf_firms = gpd.read_file(path_puntos_firms_temp + shp_file_name, parse_dates=[[5,6]])
    test_date = gdf_firms['ACQ_DATE'].iloc[-1]

    for index, row in gdf_firms.iterrows():
        if row["ACQ_DATE"] != test_date:
            gdf_firms.drop(index, inplace=True)

    dt_object = dt.strptime(test_date, "%Y-%m-%d")
    tt = dt_object.timetuple()
    n_day = str(tt.tm_yday)
    n_day = n_day.zfill(3)
    year_nday = test_date[:4] + n_day

    for n in os.listdir(path_puntos_firms_temp):
        os.remove(path_puntos_firms_temp+n)

    gdf_firms.crs= "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"
    gdf_firms.to_file(path_puntos_firms + "V_FIRMS_" + test_date[:4] + test_date[5:7] + test_date[-2:], driver='ESRI Shapefile')

    print("Puntos FIRMS creados para la fecha: ",dt_object," numero de dia: ",n_day)
    print("FINALIZA SCRIPT 's1_1_crea_puntos_firms' para ", year_nday,"\n")
    return year_nday
```

s1_2_netcdf_a_tif.py

```
# -*- coding: utf-8 -*-
"""
@author: colvert
"""

from glob import glob
from netCDF4 import Dataset
from datetime import datetime as dt
import os
from osgeo import gdal,osr
import numpy as np

def crea_carpeta(path,name):
    """
    Crea una carpeta en un determinado directorio o ruta.
    """

    os.system("mkdir " + path + name)

def extrae_datos_netcdf(ds):
    """
    Extrae los datos de la variable 'CMIP' de un dataset netCDF4._netCDF4.Dataset.
    """

    data = ds.variables['CMI'][:].data
    return data

def crea_tif_geos(data,ds,name,path):
    """
    Crea un GeoTIFF '.tif' en coordenadas geoestacionarias con los siguientes
    parámetros: h=35786023.0, ellps=GRS80, lat_0=0.0, lon_0=-75.0
    """

    H = ds.variables['goes_imager_projection'].perspective_point_height
    xmin = ds.variables['x_image_bounds'][0] * H
    xmax = ds.variables['x_image_bounds'][1] * H
```

```
    ymin = ds.variables['y_image_bounds'][0] * H
    ymax = ds.variables['y_image_bounds'][1] * H

    nx = data.shape[1]
    ny = data.shape[0]
    res_x = (xmax - xmin) / float(ny)
    res_y = (ymax - ymin) / float(nx)
    geotransform = (xmin, res_x, 0, ymin, 0, res_y)
    dst_ds = gdal.GetDriverByName('GTiff').Create(path + name, ny, nx, 1, gdal.GDT_Float32)

    dst_ds.SetGeoTransform(geotransform)
    srs = osr.SpatialReference()
    srs.ImportFromProj4('+proj=geos +h=35786023.0 +ellps=GRS80 +lat_0=0.0 +lon_0=-75.0 +sweep=x +no_defs')
    dst_ds.SetProjection(srs.ExportToWkt())
    dst_ds.GetRasterBand(1).WriteArray(data)
    dst_ds.FlushCache()
    dst_ds = None

def recorta_tif(ds,coordinates):
    """
    Recorta un GeoTIFF en las coordenadas recibidas.
    """

    name = ds.GetDescription().replace(".tif","_geos_mex.tif")
    gdal.Translate(name,ds,options = gdal.TranslateOptions(projWin=coordinates,noData=np.nan))


def elimina_nulos(data,ds):
    """
    Elimina los valores nulos de una matriz en base al rango valido de valores
    para cada banda.
    """

    min_value = ds["min_brightness_temperature"].valid_range[0]
    max_value = ds["max_brightness_temperature"].valid_range[1]
    data = np.where((data>max_value) | (data<min_value),np.nan,data)
    return data

def netcdf_a_tif(path_netcdf_aws, path_img_geos, year_nday):
    """
    Descarga y crea un recorte en formato GeoTIFF para América central de todo
    un dia en coordenadas geoestacionarias de todos los archivos NetCDF que se
    encuentren el directorio de entrada.
    """

    print("INICIA SCRIPT 's1_2_netcdf_a_tif.py")

    for n in os.listdir(path_netcdf_aws):
        if n.endswith(".nc"):
            os.remove(path_netcdf_aws + n)
        if n.endswith(".txt"):
            os.remove(path_netcdf_aws + n)
    os.chdir(path_netcdf_aws)
    os.system("sh downloadGOES16.sh " + year_nday[:4] + " " + year_nday[4:])

    nc_list = glob(path_netcdf_aws + "*CMI*")
    nc_list.sort()

    dt_object = dt.strptime(year_nday, "%Y%j")
    crea_carpeta(path_img_geos, str(dt_object.year))
    crea_carpeta(path_img_geos + str(dt_object.year)+"/", str(dt_object.timetuple().tm_yday).zfill(3))
    path_img_geos2 = path_img_geos + str(dt_object.year) + "/" + str(dt_object.timetuple().tm_yday).zfill(3) + "/"

    ch07_img_list = []
    for n in nc_list:
        if n.find("OR_ABI-L2-CMIPF-M6C07_G16_s" + year_nday) != -1:
            ch07_img_list.append(n)

    for i in ch07_img_list:
            ds_ch07 = Dataset(i)
            try:
                ds_ch14 = Dataset(glob(path_netcdf_aws+"*C14*"+i.split("/")[-1][25:38]+"*"+".nc")[0])
                data_ch07 = extrae_datos_netcdf(ds_ch07)
                data_ch14 = extrae_datos_netcdf(ds_ch14)
                data_ch07 = elimina_nulos(data_ch07,ds_ch07)
```

```
            data_ch14 = elimina_nulos(data_ch14,ds_ch14)
            data_dif = data_ch07 - data_ch14

            crea_tif_geos(data_ch07,ds_ch07,i.split("/")[-1][:-3] + ".tif",path_img_geos2)
            crea_tif_geos(data_dif,ds_ch14,(i.split("/")[-1][:-3] + ".tif").replace("C07","DIFC07-C14"),path_img_geos2)
        except:
            continue

    coords_mex_geos = [-3700000.0,3396435.0,1200000.0,750000.0]
    for n in os.listdir(path_img_geos2):
        ds_geos = gdal.Open(path_img_geos2 + n)
        recorta_tif(ds_geos,coords_mex_geos)
        os.remove(path_img_geos2 + n)

    for n in os.listdir(path_img_geos2):
        if n.endswith(".xml"):
            os.remove(path_img_geos2 + n)

    for i in os.listdir(path_netcdf_aws):
            if i.endswith(".nc"):
            os.remove(path_netcdf_aws + i)

    print("FINALIZA SCRIPT 's1_2_netcdf_a_tif.py'\n")
    return path_img_geos2
```

s1_3_crea_eventos_firms.py

```
# -*- coding: utf-8 -*-
"""
@author: Colvert Gomez Rubio
"""

import glob, os
import geopandas as gpd
import pandas as pd
from datetime import datetime as dt
from shapely.geometry.polygon import Polygon
from shapely.geometry.multipolygon import MultiPolygon

def explotar(in_df):
    """
    Recibe como entrada un DataFrame de multipolígonos y los separa  en
    polígonos simples en base a sus atributos.
    """

    out_df = gpd.GeoDataFrame(columns=in_df.columns, crs=in_df.crs)
    for idx, row in in_df.iterrows():
        if type(row.geometry) == Polygon:
            out_df = out_df.append(row,ignore_index=True)
        if type(row.geometry) == MultiPolygon:
            mult_df = gpd.GeoDataFrame(columns=in_df.columns, crs=in_df.crs)
            recs = len(row.geometry)
            mult_df = mult_df.append([row]*recs,ignore_index=True)
            for geom in range(recs):
                mult_df.loc[geom,'geometry'] = row.geometry[geom]
            out_df = out_df.append(mult_df,ignore_index=True)
    return out_df

def crea_eventos_firms(path_puntos_firms, path_eventos_firms, path_eventos_firms_filtrados, year_nday):
    """
    Busca y abre el producto FIRMS VIIRS 375m de puntos descargado en el script
    's1_1_crea_puntos_firms.py', lo depura de informacion innecesaria y hace el
    proceso de crear los eventos firms en formato shp con el nombre
    'evnt_viirs_ref_AAAAJJJ.shp' y 'evnt_viirs_1km_AAAAJJJ.shp'
    """

    print("INICIA SCRIPT 's1_3_crea_eventos_firms.py'")
    shp_date = dt.strptime(str(year_nday), "%Y%j")
    file_list = glob.glob(os.path.join(path_puntos_firms , "V_FIRMS_{}/V_FIRMS_{}.shp".format(shp_date.strftime("%Y%m
%d"),shp_date.strftime("%Y%m%d") )))
    file_list.sort()
```

```
    out_df = gpd.read_file(file_list[0], parse_dates=[[5,6]])
    out_df["fecha"] = pd.to_datetime(out_df.ACQ_DATE +" "+ out_df.ACQ_TIME)
    columns_delete = out_df.columns[0:13]

    points_df = out_df.copy()

    out_df.drop(columns_delete, axis=1, inplace=True)
    out_df["geom_pix"] = out_df.geometry.buffer(0.0025, cap_style=3)
    out_df.set_geometry(col="geom_pix", inplace=True, drop=True)
    out_df["fecha"] = out_df.fecha.astype(str)
    out_df = out_df.dissolve("fecha")
    out_df.reset_index(inplace=True)
    out_df = explotar(out_df)
    out_df["areakm2"] = out_df.geometry.to_crs(epsg=6362).area/1000000

    out_name = os.path.join(path_eventos_firms , "viirs_dsv_hora_{}.shp".format(year_nday))
    out_df.to_file(out_name)

    out_df["dia"] = pd.to_datetime(out_df.fecha).dt.day
    out_df = out_df.dissolve(out_df.dia)
    out_df.drop(["fecha", "dia"], axis=1, inplace=True)
    out_df.reset_index(inplace=True)
    out_df = explotar(out_df)
    out_df["areakm2"] = out_df.geometry.to_crs(epsg=6362).area/1000000

    points = gpd.sjoin(points_df, out_df, how='inner', op='within')
    summary_df = points[["index_right", "fecha"]].copy()
    summary_df.rename(columns={"index_right":"evnt_id"}, inplace=True)
    summary_df["hora"] = summary_df.fecha.dt.hour
    summary_df.reset_index(inplace=True, drop=True)
    summary_df2 = pd.DataFrame(summary_df.groupby(["evnt_id"])["hora"].unique())
    summary_df2.reset_index(inplace=True)
    summary_df2["npuntos"] = pd.DataFrame(summary_df.groupby(["evnt_id"])["hora"].count())
    summary_df2["first"] = pd.DataFrame(summary_df.groupby(["evnt_id"])["hora"].first())-1
    summary_df2["last"] = pd.DataFrame(summary_df.groupby(["evnt_id"])["hora"].last())+1
    summary_df2["nhoras"] = summary_df2["last"] - summary_df2["first"]

    ref_evnt_df = pd.concat([out_df, summary_df2], axis=1)
    ref_evnt_df = ref_evnt_df[(ref_evnt_df.nhoras > 2) & (ref_evnt_df.npuntos > 2) & (ref_evnt_df.areakm2 >= 1 )]
    ref_evnt_df.drop(["hora"], axis=1, inplace=True)

    out_name = os.path.join(path_eventos_firms_filtrados , "evnt_viirs_ref_{}.shp".format(year_nday))
    ref_evnt_df.to_file(out_name)

    print("FINALIZA SCRIPT 's1_3_crea_eventos_firms.py'\n")
```

s1_4_muestras_goes_xdia.py

```
# -*- coding: utf-8 -*-
"""
@author: Colvert Gomez Rubio
"""

import os
import glob
import geopandas as gpd
import pandas as pd
from rasterstats import zonal_stats

def muestreador_c07(buffer_evnt_df, gdf_evnt_buffer, file):
    date = os.path.basename(file).split("_")[3][1:12]
    data = zonal_stats(buffer_evnt_df, file, all_touched=True)
    a = gdf_evnt_buffer.join(pd.DataFrame(data, columns=['max']))
    a.rename(columns={"max":"ch7"}, inplace=True)
    a["fecha"] = date
    return a[['evnt_id', 'ch7', 'fecha']]


def muestreador_dif(buffer_evnt_df, gdf_evnt_buffer, file):
    date = os.path.basename(file).split("_")[3][1:12]
    data = zonal_stats(buffer_evnt_df, file, all_touched=True)
    a = gdf_evnt_buffer.join(pd.DataFrame(data, columns=['max']))
```

```
    a.rename(columns={"max":"dif7_14"}, inplace=True)
    a["fecha"] = date
    return a[['evnt_id', 'dif7_14', 'fecha']]

def muestras_goes_c07(path_eventos_viirs_filtrados, path_img_geos_prueba, path_muestras_goes_firms, year_nday):
    """
    Reproyecta los eventos firms creados en el script 's1_3_crea_eventos_firms.py'
    a coordenadas geoestacionarias, les hace un buffer de 1000m, calcula las
    estadísticas zonales de este producto con las imágenes GOES de la banda 7
    creadas en el script 's1_2_netcdf_a_tif.py' para los 144 horarios del dia
    y genera el archivos 'valores_muestreados_ch07_AAAAJJJ.csv'.
    """
    print("INICIA SCRIPT 's1_4_muestras_goes_xdia.py'")

    ref_evnt_df = os.path.join(path_eventos_viirs_filtrados, "evnt_viirs_ref_{}.shp".format(year_nday))
    buffer_evnt_df = os.path.join(path_eventos_viirs_filtrados,"evnt_viirs_1km_{}.shp".format(year_nday))

    gdf_evnt = gpd.read_file(ref_evnt_df)

    geos_proj = "+proj=geos +h=35786023.0 +ellps=GRS80 +lat_0=0.0 +lon_0=-75.0 +sweep=x +no_defs"
    temp_df = gdf_evnt.to_crs(crs=geos_proj)
    temp_df["geom_pix"] = temp_df.geometry.buffer(1000)
    temp_df.set_geometry(col="geom_pix", inplace=True, drop=True)
    temp_df.to_file(buffer_evnt_df)
    del(temp_df)
    gdf_evnt_buffer = gpd.read_file(buffer_evnt_df)

    search_img = os.path.join(path_img_geos_prueba , "OR_ABI-L2-CMIPF-M6C07_G16_s{}*.tif".format(year_nday))
    img_list = glob.glob(search_img)

    img_list.sort()

    samp_df = muestreador_c07(buffer_evnt_df, gdf_evnt_buffer, img_list[0])
    for i in range(len(img_list)):
        df = muestreador_c07(buffer_evnt_df, gdf_evnt_buffer, img_list[i])
        if i == 0:
            out_df = samp_df.copy()
        else:
            out_df = pd.concat([out_df, df], ignore_index=True)
        del(df)
    del(samp_df)

    out_df.to_csv(os.path.join(path_muestras_goes_firms, "valores_muestreados_ch07_{}.csv".format(year_nday)))

def muestras_goes_dif07_14(path_eventos_viirs_filtrados, path_img_geos_prueba, path_muestras_goes_firms, year_nday):
    """
    Reproyecta los eventos firms creados en el script 's1_3_crea_eventos_firms.py'
    a coordenadas geoestacionarias, les hace un buffer de 1000m, calcula las
    estadísticas zonales de este producto con las imágenes GOES de la
    diferencia de la banda 7 menos 14, creadas en el script 's1_2_netcdf_a_tif.py'
    para los 144 horarios del dia y genera el archivo
    'valores_muestreados_dif7_14_AAAAJJJ.csv'.
    """

    buffer_evnt_df = os.path.join(path_eventos_viirs_filtrados, "evnt_viirs_1km_{}.shp".format(year_nday))
    gdf_evnt_buffer = gpd.read_file(buffer_evnt_df)

    search_img = os.path.join(path_img_geos_prueba , "OR_ABI-L2-CMIPF-M6DIFC07-C14_G16_s{}*.tif".format(year_nday))
    img_list = glob.glob(search_img)
    img_list.sort()

    samp_df = muestreador_dif(buffer_evnt_df, gdf_evnt_buffer, img_list[0])
    for i in range(len(img_list)):
        df = muestreador_dif(buffer_evnt_df, gdf_evnt_buffer, img_list[i])
        if i == 0:
            out_df = samp_df.copy()
        else:
            out_df = pd.concat([out_df, df], ignore_index=True)
        del(df)
    del(samp_df)

    out_df.to_csv(os.path.join(path_muestras_goes_firms, "valores_muestreados_dif7_14_{}.csv".format(year_nday)))

    print("FINALIZA SCRIPT 's1_4_muestras_goes_xdia.py'\n")
```

s1_5_crea_umbrales.py

```
# -*- coding: utf-8 -*-
"""
@author: Colvert Gomez Rubio
"""

import pandas as pd
import os, glob
import numpy as np
from scipy.interpolate import InterpolatedUnivariateSpline


def reduccion_hora_c07(int_df, column_name, evnt_id):
    int_df = int_df[int_df.evnt_id == evnt_id].copy()

    mean_out_df = pd.DataFrame(int_df[column_name].resample('1h').mean())
    mean_out_df.rename(columns={column_name:'mean_{}'.format(column_name)}, inplace=True)

    max_out_df = pd.DataFrame(int_df[column_name].resample('1h').max())
    max_out_df.rename(columns={column_name:'max_{}'.format(column_name)}, inplace=True)

    min_out_df = pd.DataFrame(int_df[column_name].resample('1h').min())
    min_out_df.rename(columns={column_name:'min_{}'.format(column_name)}, inplace=True)

    df_salida_std = pd.DataFrame(int_df[column_name].resample('1h').std())
    df_salida_std.rename(columns={column_name:'std_{}'.format(column_name)}, inplace=True)

    out_1h_df = min_out_df.merge(mean_out_df, on=int_df.index.name).merge(max_out_df, on=int_df.index.name).merge(df_salida_std,
on=int_df.index.name)
    out_1h_df["evnt_id"] = evnt_id
    out_1h_df["umbral"] = out_1h_df.max_ch7 - out_1h_df.std_ch7

    del(mean_out_df, max_out_df, min_out_df, df_salida_std)

    return out_1h_df

def reduccion_hora_dif(int_df, column_name, evnt_id):
    int_df = int_df[int_df.evnt_id == evnt_id].copy()

    mean_out_df = pd.DataFrame(int_df[column_name].resample('1h').mean())
    mean_out_df.rename(columns={column_name:'mean_{}'.format(column_name)}, inplace=True)

    max_out_df = pd.DataFrame(int_df[column_name].resample('1h').max())
    max_out_df.rename(columns={column_name:'max_{}'.format(column_name)}, inplace=True)

    min_out_df = pd.DataFrame(int_df[column_name].resample('1h').min())
    min_out_df.rename(columns={column_name:'min_{}'.format(column_name)}, inplace=True)

    df_salida_std = pd.DataFrame(int_df[column_name].resample('1h').std())
    df_salida_std.rename(columns={column_name:'std_{}'.format(column_name)}, inplace=True)

    out_1h_df = min_out_df.merge(mean_out_df, on=int_df.index.name).merge(max_out_df, on=int_df.index.name).merge(df_salida_std,
on=int_df.index.name)
    out_1h_df["evnt_id"] = evnt_id
    out_1h_df["umbral"] = out_1h_df.mean_dif7_14

    del(mean_out_df, max_out_df, min_out_df, df_salida_std)

    return out_1h_df

def crea_umbrales_c07(path_muestras_goes_firms, path_umbrales, year_nday):
    """
    Lee el archivo csv de valores muestreados para la banda 7 del script
    's1_4_muestras_goes_xdia.py' y por medio de condiciones y una interpolación
    segmentaría cuadrática genera un archivo de umbrales óptimos, con un umbral
    para cada uno de los 144 horarios del dia, llamado 'umbral_ch07_AAAAJJJ.csv'
    """

    print("INICIA SCRIPT 's1_5_crea_umbrales.py'")

    ref_csv = os.path.join(path_muestras_goes_firms, "valores_muestreados_ch07_{}.csv".format(year_nday))
    csv_ref_list = glob.glob(ref_csv)
```

```
    if len(csv_ref_list) == 0:
        print("Sin datos")
        exit(0)
    if len(csv_ref_list) > 1:
        print("Muchos datos")
        exit(0)
    if len(csv_ref_list) == 1:
        print("Archivo de valores muestreados encontrado")

    df = pd.read_csv(csv_ref_list[0])
    df["fecha"] = pd.to_datetime(df.fecha, format='%Y%j%H%M')
    df.set_index(df.fecha, drop=True, inplace=True)
    df.drop(columns="fecha", inplace=True)

    evnt_list = list(df.evnt_id.unique())
    temp_df = reduccion_hora_c07(df, "ch7", evnt_list[0])
    for i in range(len(evnt_list)):
        temp_df2 = reduccion_hora_c07(df, "ch7", evnt_list[i])
        if i == 0:
            out_df = temp_df
        else:
            out_df = pd.concat([out_df, temp_df2])
        del(temp_df2)
    del(temp_df)

    grpby_df = out_df.groupby("fecha")
    thld_24h_df = grpby_df.umbral.describe()

    x = pd.date_range('00:00', periods=(24), freq='1h')
    x = np.round(((x.strftime("%H")).astype(int))+ ((x.strftime("%M")).astype(int)/60), 2)
    y = np.array(thld_24h_df["mean"])
    pred_interp = InterpolatedUnivariateSpline(x,y, k=2)
    xi = pd.date_range('00:00', periods=(6*24), freq='10min')
    xi = np.round(((xi.strftime("%H")).astype(int))+ ((xi.strftime("%M")).astype(int)/60.0), 2)
    yi = np.round(pred_interp(xi), 2)

    errors = 0
    for n in yi:
        if np.isnan(n) == True:
            errors += 1
    if errors == 0:
        result = pd.DataFrame(yi, xi)
        result.reset_index(inplace=True)
        result.columns = ["h", "pred"]
        result["fecha"]= ( pd.date_range('00:00', periods=(6*24), freq='10min'))
        result[["h", "pred"]].to_csv(os.path.join(path_umbrales, "umbral_ch07_{}.csv".format(year_nday)),index=False)

def crea_umbrales_dif07_14(path_muestras_goes_firms, path_umbrales, year_nday):
    """
    Lee el archivo csv de valores muestreados para de la diferencia de la
    banda 7 menos la banda 14 del script 's1_4_muestras_goes_xdia.py' y por
    medio de condiciones y una interpolación segmentaría cuadrática genera un
    archivo de umbrales óptimos, con un umbral para cada uno de los 144
    horarios del dia, llamado 'umbral_dif7_14_AAAAJJJ.csv'
    """

    ref_csv = os.path.join(path_muestras_goes_firms, "valores_muestreados_dif7_14_{}.csv".format(year_nday))
    df = pd.read_csv(ref_csv, names=["evnt_id", "dif7_14", "fecha"], skiprows=1)
    df["fecha"] = pd.to_datetime(df.fecha, format='%Y%j%H%M')
    df.set_index(df.fecha, drop=True, inplace=True)
    df.drop(columns="fecha", inplace=True)

    evnt_list = list(df.evnt_id.unique())
    temp_df = reduccion_hora_dif(df, "dif7_14", evnt_list[0])

    for i in range(len(evnt_list)):
        temp_df2 = reduccion_hora_dif(df, "dif7_14", evnt_list[i])
        if i == 0:
            out_df = temp_df
        else:
            out_df = pd.concat([out_df, temp_df2])
        del(temp_df2)
    del(temp_df)

    grpby_df = out_df.groupby("fecha")
```

```
    thld_24h_df = grpby_df.umbral.describe()

    x = pd.date_range('00:00', periods=(24), freq='1h')
    x = np.round(((x.strftime("%H")).astype(int))+ ((x.strftime("%M")).astype(int)/60), 2)
    y = np.array(thld_24h_df["mean"])
    pred_interp = InterpolatedUnivariateSpline(x,y, k=2)
    xi = pd.date_range('00:00', periods=(6*24), freq='10min')
    xi = np.round(((xi.strftime("%H")).astype(int))+ ((xi.strftime("%M")).astype(int)/60.0), 2)
    yi = np.round(pred_interp(xi), 2)

    errors = 0
    for n in yi:
        if np.isnan(n) == True:
            errors += 1
    if errors == 0:
        result = pd.DataFrame(yi, xi)
        result.reset_index(inplace=True)
        result.columns = ["h", "pred"]
        result["fecha"]= ( pd.date_range('00:00', periods=(6*24), freq='10min'))
        result[["h", "pred"]].to_csv(path_umbrales + "umbral_dif7_14_{}.csv".format(year_nday),index=False)

    print("FINALIZA SCRIPT 's1_5_crea_umbrales.py'\n")
```

s2_incendios_auto.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: Colvert Gomez Rubio
"""

import shutil
import s2_1_prepara_imagenes_geo as s_pig
import s2_2_extrae_umbral_10dias_max as s_eu10m
import s2_3_pp_y_pc as s_pppc
import s2_4_crea_shp_csv as s_csc

path_resultados = "/home/fires/resultados/"
path_datos_aux = "/home/fires/datos_aux/"

path_netcdf_cspp = "/data1/output/abi/l2/fd/"
path_img_geo = path_datos_aux + "raster/img_goes_geo/"
date_obj, path_img_geo_prueba = s_pig.prepara_img_geo(path_netcdf_cspp,path_img_geo)

path_umbrales = path_datos_aux + "vectorial/umbrales/"
thld_c07,thld_dif = s_eu10m.umbral_10dias_max(date_obj, path_umbrales)

path_masc_agua = path_datos_aux + "raster/masc_agua/"
path_masc_des = path_datos_aux + "raster/masc_des/"
path_masc_urb = path_datos_aux + "raster/masc_urb/"
path_registro_num_pc = path_resultados + "registro_num_pc/"
dataPC = s_pppc.pp_pc(path_masc_agua, path_masc_des, path_masc_urb, path_img_geo_prueba, thld_c07, thld_dif, date_obj,
path_registro_num_pc)

path_div_pol_est = path_datos_aux + "vectorial/div_pol_est/"
path_div_pol_pais = path_datos_aux + "vectorial/div_pol_pais/"
path_anp = path_datos_aux + "vectorial/anp/"
path_land_cover = path_datos_aux + "raster/land_cover/"
path_csv_10min = path_resultados + "csv_10min/"
path_shp_10min = path_resultados + "shp_10min/"
path_csv_24hrs = path_resultados + "csv_24hrs/"
path_shp_24hrs = path_resultados + "shp_24hrs/"
s_csc.crea_shp_csv(dataPC,date_obj,path_img_geo_prueba,path_div_pol_est,path_div_pol_pais,path_anp,path_land_cover,path_csv_10min,path
_shp_10min,path_csv_24hrs,path_shp_24hrs)

shutil.rmtree(path_img_geo_prueba, ignore_errors=True)
```

s2_1_prepara_imagenes_geo.py

```
# -*- coding: utf-8 -*-
"""
@author: Colvert Gomez Rubio
"""

from glob import glob
from netCDF4 import Dataset
from datetime import datetime
import os
from osgeo import gdal,osr
import numpy as np

def crea_carpeta(path,h):
    os.system("mkdir " + path + h)

def extrae_datos_netcdf(ds,type_d):
    if type_d == "banda":
        data = ds.variables[list(ds.variables.keys())[2]][:].data
    elif type_d == "producto":
        data = ds.variables[list(ds.variables.keys())[13]][:].data
    return data

def extrae_limites(ds):
    H = ds.variables['goes_imager_projection'].perspective_point_height
    xmin = ds.variables['x_image_bounds'][0] * H
    xmax = ds.variables['x_image_bounds'][1] * H
    ymin = ds.variables['y_image_bounds'][0] * H
    ymax = ds.variables['y_image_bounds'][1] * H
    return xmax,xmin,ymax,ymin

def crea_tif_geos(data,ds,path):
    xmax,xmin,ymax,ymin = extrae_limites(ds)
    nx = data.shape[1]
    ny = data.shape[0]
    xres = (xmax - xmin) / float(ny)
    yres = (ymax - ymin) / float(nx)
    geotransform = (xmin, xres, 0, ymin, 0, yres)
    name = ds.filepath().split("/")[-1]
    name = str(name)
    dst_ds = gdal.GetDriverByName('GTiff').Create(path + name.replace(".nc","_geos.tif"), ny, nx, 1, gdal.GDT_Float32)
    dst_ds.SetGeoTransform(geotransform)
    srs = osr.SpatialReference()
    srs.ImportFromProj4('+proj=geos +h=35786023.0 +ellps=GRS80 +lat_0=0.0 +lon_0=-75.0 +sweep=x +no_defs')
    dst_ds.SetProjection(srs.ExportToWkt())
    dst_ds.GetRasterBand(1).WriteArray(data)
    dst_ds.FlushCache()
    dst_ds = None

def recorta_tif_geos(ds):
    mex_win_geos = [-4079059.0, 3400000.0, 1840000, 600000.0]
    name = ds.GetDescription().replace(".tif","_mex.tif")
    gdal.Translate(name,ds,options = gdal.TranslateOptions(projWin=mex_win_geos,noData=np.nan))

def reproyecta_a_geo(ds):
    name = ds.GetDescription().replace("geos_mex.tif","geo.tif")
    gdal.Warp(name,ds,options=gdal.WarpOptions(dstSRS="EPSG:4326",yRes=0.018,xRes=0.018,dstNodata=np.nan))

def recorta_tif_geo(ds,name):
    mex_win_geo = [-117.6 ,33.6, -58.5, 6.5]
    gdal.Translate(path_img_geo_prueba + name,ds,options = gdal.TranslateOptions(projWin=mex_win_geo,noData=np.nan))

def crea_tif_geo(data ,ds ,tdato, tifNom):
    nx = data.shape[1]
    ny = data.shape[0]
    geotransform = ds.GetGeoTransform()
    if tdato == "Byte":
        dst_ds = gdal.GetDriverByName('GTiff').Create(tifNom+'.tif', nx, ny, 1, gdal.GDT_Byte)
    elif tdato == "Float32":
        dst_ds = gdal.GetDriverByName('GTiff').Create(tifNom+'.tif', nx, ny, 1, gdal.GDT_Float32)
    dst_ds.SetGeoTransform(geotransform)
    srs = osr.SpatialReference()
    srs.ImportFromProj4("+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs")
```

```
    dst_ds.SetProjection(srs.ExportToWkt())
    dst_ds.GetRasterBand(1).WriteArray(data)
    dst_ds.FlushCache()
    dst_ds = None

def elimina_nulos(ds,vmin,vmax):
    data = ds.ReadAsArray()
    data = np.where((data>vmax) | (data<vmin),np.nan,data)
    name = ds.GetDescription().split("/")[-1]
    name = name.replace(".tif","_2km")
    crea_tif_geo(data,ds,"Float32",path_img_geo_prueba + name)
    return data

def prepara_img_geo(path_netcdf_cspp,path_img_geo):
    """
    Busca las ultimas imágenes de la banda 7 y 14 generadas por CSPP Geo, las
    reproyecta a coordenadas geográficas, las recorta, elimina valores nulos o
    no validos y crea un archivo .tif de los resultados.
    """

    print("\nINICIA SCRIPT 's2_1_prepara_imagenes_geo.py'")

    netcdf_list = glob(path_netcdf_cspp + "*CMI*")
    netcdf_list.sort()
    year_day_hour = (netcdf_list[-1].split("/")[-1])[netcdf_list[-1].split("/")[-1].find("_s")+2:netcdf_list[-1].split("/")[-1].find("_s")+13]

    date_obj = datetime.strptime(year_day_hour, "%Y%j%H%M")
    crea_carpeta(path_img_geo,str(date_obj.year))
    crea_carpeta(path_img_geo+str(date_obj.year)+"/", str(date_obj.timetuple().tm_yday).zfill(3))
    crea_carpeta(path_img_geo+str(date_obj.year)+"/"+str(date_obj.timetuple().tm_yday).zfill(3)+"/",
str(date_obj.hour).zfill(2)+str(date_obj.minute).zfill(2))

    global path_img_geo_prueba
    path_img_geo_prueba = path_img_geo+str(date_obj.year)
+"/"+str(date_obj.timetuple().tm_yday).zfill(3)+"/"+str(date_obj.hour).zfill(2)+str(date_obj.minute).zfill(2)+"/"

    ds07 = Dataset(glob(path_netcdf_cspp+"*CMI*C07*s"+year_day_hour+"*"+".nc")[0])
    ds14 = Dataset(glob(path_netcdf_cspp+"*CMI*C14*s"+year_day_hour+"*"+".nc")[0])

    data07 = extrae_datos_netcdf(ds07,"banda")
    data14 = extrae_datos_netcdf(ds14,"banda")

    crea_tif_geos(data07,ds07,path_img_geo_prueba)
    crea_tif_geos(data14,ds14,path_img_geo_prueba)

    for n in os.listdir(path_img_geo_prueba):
        if n.endswith("geos.tif"):
            ds_geos = gdal.Open(path_img_geo_prueba + n)
            recorta_tif_geos(ds_geos)
            os.remove(path_img_geo_prueba + n)

    for n in os.listdir(path_img_geo_prueba):
        if n.endswith("geos_mex.tif"):
            ds_geos = gdal.Open(path_img_geo_prueba + n)
            reproyecta_a_geo(ds_geos)
            os.remove(path_img_geo_prueba + n)

    for n in os.listdir(path_img_geo_prueba):
        if n.endswith("geo.tif"):
            ds_geo = gdal.Open(path_img_geo_prueba + n)
            name = n.replace(".tif","_mex.tif")
            recorta_tif_geo(ds_geo,name)
            os.remove(path_img_geo_prueba + n)

    dsC07 = gdal.Open(glob(path_img_geo_prueba +  "*CMI*C07*geo_mex.tif")[0])
    dsC14 = gdal.Open(glob(path_img_geo_prueba +  "*CMI*C14*geo_mex.tif")[0])

    elimina_nulos(dsC07,197.31,411.86)
    elimina_nulos(dsC14,96.19,341.28)

    for n in os.listdir(path_img_geo_prueba):
        if n.endswith("_mex.tif"):
            os.remove(path_img_geo_prueba + n)
        if n.endswith(".xml"):
            os.remove(path_img_geo_prueba + n)
```

```
    print("Fecha: ",date_obj)
    print("FINALIZA SCRIPT 's2_1_prepara_imagenes_geo.py'\n")
    return date_obj, path_img_geo_prueba
```

s2_2_extrae_umbral_10dias_max.py

```
# -*- coding: utf-8 -*-
"""
@author: Colvert Gomez Rubio
"""

import csv
import os
import numpy as np

def umbral_10dias_max(date_obj, path_umbrales):
    """
    Busca en la carpeta de los archivos de umbrales los últimos 10 archivos
    mas recientes para la banda 7 y para la diferencia 7 – 14 y devuelve el
    umbral máximo de esos días para la banda 7 y diferencia 7 – 14.
    """

    print("INICIA SCRIPT 's2_2_extrae_umbral_10dias_max.py'")

    thld_c07_list = []
    thld_dif_list = []
    for n in os.listdir(path_umbrales):
        if "ch07" in n:
            thld_c07_list.append(n)
        if "dif" in n:
            thld_dif_list.append(n)
    thld_c07_list.sort()
    thld_dif_list.sort()
    thld_c07_list = thld_c07_list[-10:]
    thld_dif_list = thld_dif_list[-10:]

    thld_c07_dict = {}
    thld_dif_dict = {}

    with open(path_umbrales+n, 'r') as csvFile:
        reader = csv.reader(csvFile)
        for row in reader:
            thld_c07_dict[row[0]] = []
            thld_dif_dict[row[0]] = []

    for n in thld_c07_list:
        with open(path_umbrales+n, 'r') as csvFile:
            reader = csv.reader(csvFile)
            for row in reader:
                thld_c07_dict[row[0]].append(row[1])

    for n in thld_dif_list:
        with open(path_umbrales+n, 'r') as csvFile:
            reader = csv.reader(csvFile)
            for row in reader:
                thld_dif_dict[row[0]].append(row[1])

    hour_dec = str(round((date_obj.hour + date_obj.minute/60),2))
    thld_c07_dict = [float(i) for i in thld_c07_dict[hour_dec]]
    thld_dif_dict = [float(i) for i in thld_dif_dict[hour_dec]]
    array_u_c07 = np.asarray(thld_c07_dict)
    array_u_dif = np.asarray(thld_dif_dict)
    thld_c07 = array_u_c07.max()
    thld_dif = array_u_dif.max()
    print ("Umbral de la banda 7: ",thld_c07)
    print ("Umbral de la diferencia: ",thld_dif)

    print("FINALIZA SCRIPT 's2_2_extrae_umbral_10dias_max.py'\n")
    return thld_c07, thld_dif
```

s2_3_pp_y_pc.py

```
# -*- coding: utf-8 -*-
"""
@author: Colvert Gomez Rubio
"""

from osgeo import gdal
import numpy as np
from glob import glob

def aplanar(lst):
    return [item for sublist in lst for item in sublist]

def prueba_contexto(dataPP,dataC07,dif7_14):
    positions_pp = np.where(dataPP == 1)
    rows = positions_pp[0]
    columns = positions_pp[1]

    for t in zip(rows,columns):
        c07_pp_value = dataC07[t[0],t[1]]
        dif_pp_value = dif7_14[t[0],t[1]]
        kernel = 7
        if kernel == 3:
            subtract =1
            add=2
        elif kernel == 5:
            subtract =2
            add=3
        elif kernel ==7:
            subtract  = 3
            add = 4

        clip_c07 = dataC07[t[0]-subtract :t[0]+add, t[1]-subtract :t[1]+add]
        clip_dif = dif7_14[t[0]-subtract :t[0]+add, t[1]-subtract :t[1]+add]
        clip_c07_list = aplanar(clip_c07)
        clip_dif_list = aplanar(clip_dif)
        clip_c07_list2 = []
        clip_dif_list2 = []

        for n in clip_c07_list:
            if n != 0:
                clip_c07_list2.append(n)
        for n in clip_dif_list:
            if n != 0:
                clip_dif_list2.append(n)

        mean_c07 = np.mean(clip_c07_list2)
        mean_dif = np.mean(clip_dif_list2)
        std_c07 = np.std(clip_c07_list2)
        std_dif = np.std(clip_dif_list2)

        if (c07_pp_value > (mean_c07 + 2.5*std_c07)) and (dif_pp_value > (mean_dif + 2.5*std_dif)):
            dataPP[t] = 1
        else:
            dataPP[t] = 0
    return dataPP


def pp_pc(path_masc_agua,path_masc_des,path_masc_urb,path_img_geo_prueba,thld_c07,thld_dif,date_obj,path_registro_num_pc):
    """
    Abre las imágenes generadas por el script 's2_1_prepara_imagenes_geo.py',
    y en base a los valores de umbrales del script 's2_2_extrae_umbral_10dias_max.py'
    reclasifica las imágenes y obtiene los pixeles potenciales, después hace la
    prueba de contexto y da como resultado una matriz de puntos de calor finales.
    """

    print("INICIA SCRIPT 's2_3_pp_y_pc.py'")

    ds = gdal.Open(path_masc_agua+'masc_agua.tif')
    dataAGUA = ds.ReadAsArray()
    ds = gdal.Open(path_masc_des+'masc_des.tif')
    dataDES = ds.ReadAsArray()
    ds = gdal.Open(path_masc_urb+'masc_urb.tif')
```

```
    dataURB = ds.ReadAsArray()

    dsC07 = gdal.Open(glob(path_img_geo_prueba+"*CMI*C07*"+".tif")[0])
    dataC07 = dsC07.ReadAsArray()
    dsC14 = gdal.Open(glob(path_img_geo_prueba+"*CMI*C14*"+".tif")[0])
    dataC14 = dsC14.ReadAsArray()
    dif7_14 = dataC07 - dataC14

    dataC07_r = np.where(dataC07 > thld_c07, 1, 0)
    dif7_14_r = np.where(dif7_14 > thld_dif,1, 0)

    dataPP = dataC07_r * dif7_14_r * dataAGUA * dataURB * dataDES

    count = np.count_nonzero(dataPP == 1)
    print ('Incendios antes de la prueba de contexto: ',count)

    dataPC = prueba_contexto(dataPP,dataC07,dif7_14)

    count1 = np.count_nonzero(dataPP == 1)
    print ('Incendios despues de la prueba de contexto: ',count1)

    file = open(path_registro_num_pc+"numero_pc_"+date_obj.strftime("%Y%m%d")+".txt",'a')
    file.write('%s \t' %date_obj.strftime("%Y-%m-%d %H:%M"))
    file.write('%s \t' %thld_c07)
    file.write('%s \t' %thld_dif)
    file.write('%s \t' %count)
    file.write('%s \n' %count1)
    file.close()

    print("FINALIZA SCRIPT 's2_3_pp_y_pc.py'\n")
    return dataPC
```

s2_4_crea_shp_csv.py

```
# -*- coding: utf-8 -*-
"""
@author: Colvert Gomez Rubio
"""

import numpy as np
from osgeo import gdal
import geopandas as gpd
from glob import glob
from shapely.geometry import Point
import pandas as pd
import os

def loc_estado(x,y,shp_div_est):
    geometry = [Point(x, y)]
    df = gpd.GeoDataFrame(geometry=geometry)
    pointInPolys = gpd.tools.sjoin(df, shp_div_est, how='left',op="within")
    return pointInPolys["ENTIDAD"].values[0]

def loc_pais(x,y,shp_div_pais):
    geometry = [Point(x, y)]
    df = gpd.GeoDataFrame(geometry=geometry)
    pointInPolys = gpd.tools.sjoin(df, shp_div_pais, how='left',op="within")
    return pointInPolys["NAME_ES"].values[0]

def loc_anp(x,y,shape_anp):
    geometry = [Point(x, y)]
    df = gpd.GeoDataFrame(geometry=geometry)
    pointInPolys = gpd.tools.sjoin(df, shape_anp, how='left',op="within")
    return pointInPolys["ID_ANP"].values[0]

def nombre_lc(lc):
    if lc == 1:
        nombre_lc = "Evergreen Needleleaf Forests"
    elif lc == 2:
        nombre_lc = "Evergreen Broadleaf Forests"
    elif lc == 3:
        nombre_lc = "Deciduous Needleleaf Forests"
```

```
    elif lc == 4:
        nombre_lc = "Deciduous Broadleaf Forests"
    elif lc == 5:
        nombre_lc = "Mixed Forests"
    elif lc == 6:
        nombre_lc = "Closed Shrublands"
    elif lc == 7:
        nombre_lc = "Open Shrubland"
    elif lc == 8:
        nombre_lc = "Woody Savannas"
    elif lc == 9:
        nombre_lc = "Savannas"
    elif lc == 10:
        nombre_lc = "Grasslands"
    elif lc == 11:
        nombre_lc = "Permanent Wetlands"
    elif lc == 12:
        nombre_lc = "Croplands"
    elif lc == 13:
        nombre_lc = "Urban and Built-up Lands"
    elif lc == 14:
        nombre_lc = "Cropland/Natural Vegetation Mosaics"
    elif lc == 15:
        nombre_lc = "Permanent Snow and Ice"
    elif lc == 16:
        nombre_lc = "Barren"
    elif lc == 17:
        nombre_lc = "Water Bodies"
    return nombre_lc

def crea_csv_shp_24hrs(path_csv_10min, path_shp_10min, path_csv_24hrs, path_shp_24hrs):
    file_list=[]
    for n in os.listdir(path_shp_10min):
        if n.endswith(".shp"):
            file_list.append(n)
    file_list.sort()
    df_day = gpd.GeoDataFrame()
    for n in np.arange(1,144):
        gdf = gpd.read_file(path_shp_10min + file_list[-n])
        df_day = df_day.append(gdf)

    df_day["geom"] = df_day["lat"] * df_day["lon"]
    df_day["freq"] = df_day.groupby("geom")["geom"].transform("count")
    df_day = df_day.drop(columns=["geom"])
    df_day["freq_norm"] = (df_day["freq"] * 100) / 144
    df_day.to_file(path_shp_24hrs + "GIM10_PC_24hrs.shp", driver = 'ESRI Shapefile')
    df_day = df_day.drop(columns=["geometry"])
    df_day.to_csv(path_csv_24hrs + "GIM10_PC_24hrs"+".csv", index = False)

def crea_shp_csv(dataPC,
date_obj,path_img_geo_prueba,path_div_pol_est,path_div_pol_pais,path_anp,path_land_cover,path_csv_10min,path_shp_10min,path_csv_24hr
s,path_shp_24hrs):
    """
    Con las coordenadas de los puntos de calor resultantes del script 's2_3_pp_y_pc.py',
    extrae información de distintas fuentes que permiten su caracterización y
    crea los archivos de resultado final en formato shp y csv
    """

    print("INICIA SCRIPT 's2_4_crea_shp_csv.py'")

    num_points = np.count_nonzero(dataPC == 1)
    if num_points > 0:
        (rows,columns) = np.where(dataPC == 1)
        ds = gdal.Open(glob(path_img_geo_prueba+"*CMI*C07*"+".tif")[0])
        dataC07 = ds.ReadAsArray()
        ds = gdal.Open(glob(path_img_geo_prueba+"*CMI*C14*"+".tif")[0])
        dataC14 = ds.ReadAsArray()

        shp_div_est  = gpd.read_file(path_div_pol_est + 'destdv1gw.shp')
        shp_div_pais  = gpd.read_file(path_div_pol_pais + 'ne_10m_admin_0_countries_geo.shp')
        shp_anp  = gpd.read_file(path_anp + '182ANP_Geo_WGS84_Julio04_2019_geo.shp')

        ds_lc = gdal.Open(path_land_cover + 'Land_Cover_MCD12Q1_IGBP_geo_AC.tif')
        dataCOVER = ds_lc.ReadAsArray()
```

```
        (upper_left_x, x_size, x_rotation, upper_left_y, y_rotation, y_size) = ds.GetGeoTransform()
        df = pd.DataFrame(columns = ['lon','lat','Satelite','BT_c07','BT_c14','dif_c07-c14','Fecha','Hora','Land_Cover','Estado','Pais','ANP'])

        for r,c in zip(rows,columns):
            lat = r * y_size + upper_left_y + (y_size / 2)
            lon = c * x_size + upper_left_x + (x_size / 2)

            estado = loc_estado(lon,lat,shp_div_est)
            pais = loc_pais(lon,lat,shp_div_pais)
            anp = loc_anp(lon,lat,shp_anp)

            dif7_14 = dataC07 - dataC14
            bt_c07 = dataC07[r,c]
            bt_c14 = dataC14[r,c]
            bt_dif = dif7_14[r,c]
            lc = dataCOVER[r,c]
            lc_name = nombre_lc(lc)
            bt_c07 = float('%.2f' %bt_c07)
            bt_c14 = float('%.2f' %bt_c14)
            bt_dif = float('%.2f' %bt_dif)

            df = df.append({'lon':lon, 'lat':lat, 'Satelite':"Goes-16", 'BT_c07':bt_c07, 'BT_c14':bt_c14, 'dif_c07-c14':bt_dif,
'Fecha':date_obj.strftime("%Y-%m-%d"), 'Hora':date_obj.strftime("%H:%M"), 'Land_Cover':lc_name, 'Estado':estado, 'Pais':pais,
'ANP':anp},ignore_index=True)

        df.to_csv(path_csv_10min + "GIM10_PC_" + date_obj.strftime("%Y%m%d%H%M") + ".csv", index = False)
        df['geometry'] = df.apply(lambda row: Point(row.lon, row.lat), axis=1)
        gdf = gpd.GeoDataFrame(df, geometry='geometry')
        gdf.crs = "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"
        gdf.to_file(path_shp_10min + "GIM10_PC_" + date_obj.strftime("%Y%m%d%H%M") + ".shp", driver='ESRI Shapefile')
        crea_csv_shp_24hrs(path_csv_10min, path_shp_10min, path_csv_24hrs, path_shp_24hrs)

    print("FINALIZA SCRIPT 's2_4_crea_shp_csv.py'\n")
```

s3_resultados_csv_shp_xdia_auto.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: Colvert Gomez Rubio
"""

import geopandas as gpd
import os
from datetime import datetime

path_resultados = "/home/fires/resultados/"
path_shp_10min = path_resultados + "shp_10min/"
path_shp_dia = path_resultados + "shp_dia/"
path_csv_10min = path_resultados + "csv_10min/"
path_csv_dia = path_resultados + "csv_dia/"

def crea_archivo_dia():
    file_list=[]
    for n in os.listdir(path_shp_10min):
        if n.endswith(".shp"):
            file_list.append(n)
    file_list.sort()

    date_list = []
    for n in file_list:
        date_list.append(n[9:17])

    unique_dates = set(date_list)
    unique_dates = list(unique_dates)
    unique_dates.sort()
    date = unique_dates[-2]

    date_object = datetime.strptime(date,"%Y%m%d")
    gdf_day = gpd.GeoDataFrame()
```

```
    for i in file_list:
        if i[9:17] == date:
            gdf = gpd.read_file(path_shp_10min + i)
            gdf_day = gdf_day.append(gdf)
    gdf_day["geom"] = gdf_day["lat"] * gdf_day["lon"]
    gdf_day["freq"] = gdf_day.groupby("geom")["geom"].transform("count")
    gdf_day = gdf_day.drop(columns=["geom"])
    gdf_day["freq_norm"] = (gdf_day["freq"] * 100) / 144
    gdf_day.to_file(path_shp_dia + i[:9] + str(date_object.year) + str(date_object.month).zfill(2) + str(date_object.day).zfill(2) + ".shp", driver
= 'ESRI Shapefile')
    gdf_day = gdf_day.drop(columns=["geometry"])
    gdf_day.to_csv(path_csv_dia + "GIM10_PC_"+date+".csv", index = False)

crea_archivo_dia()
```

5. Conclusiones

Se creó, implementó y automatizó un algoritmo de detección de incendios forestales y puntos de calor para México utilizando solamente software libre. Todos los procesos a los datos espaciales fueron hechos en el lenguaje de programación Python mediante scripts que se ejecutan cada vez que se recibe una imagen (actualmente cada 10 minutos) de manera automática. Dichos scripts corren en un servidor del LANOT con sistema operativo Debian GNU/Linux y como insumo principal del algoritmo las imágenes satelitales GOES-16/ABI que se reciben en el LANOT y se procesan mediante el software CSPP Geo.

Se probó que las herramientas y posibilidades del software libre y código abierto son una solución y alternativa efectiva para hacer cualquier tipo de implementación y automatización de procesos relacionados con los SIG y percepción remota.

El algoritmo se encuentra operativo y corriendo en tiempo casi real desde el día 19 de marzo de 2020. Los resultados se crean en formato shp y csv. Hasta el día 10 de septiembre de 2020 el producto de incendios y puntos de calor se esta compartiendo con las siguientes dependencias: CENAPRED, Servicio Meteorológico Nacional, CONAFOR, CONABIO, entre otras.

La presente tesis sirve como documentación de los scripts desarrollados para el algoritmo y para se puedan realizar ajustes y cambios en los scripts y crear nuevas versiones del algoritmo.

Así mismo se pueda replicar este trabajo en cualquier proceso que se desee automatizar que contemple datos espaciales.

Referencias bibliográficas

CENAPRED. (2019). *Incendios forestales, conoce su funcionamiento para prevenir antes de que sucedan.* [Nota informativa]. CENAPRED. http://www.conacyt.gob.mx/index.php/glosario-de-terminos-sni/365-ciencia-para-la-sociedad/notas-informativas/1004-incendios-forestales-conoce-su-funcionamiento-para-prevenir-antes-de-que-sucedan

Comisión Nacional Forestal. (2010). *Incendios forestales. Guía práctica para comunicadores* (Guia Tercera edición). http://www.conafor.gob.mx:8080/documentos/docs/10/236Gu%C3%ADa%20pr%C3%A1ctica%20para%20comunicadores%20-%20Incendios%20Forestales.pdf

Cron (UNIX). (s/f). https://es.wikipedia.org/wiki/Cron_(Unix)

Culebro Juárez, M., Gómez Herrera, W. G., & Torres Sánchez, S. (2006). *Software libre vs software propietario Ventajas y desventajas*.

Díaz Tapia, M. (2018). *Incendios forestales. Experiencias y comportamiento del fuego a través de los Informes Técnicos*. Ediciones Mundi-Prensa. https://books.google.com.mx/books?id=OPyFDwAAQBAJ

Dirk Krebs, G. (2020). *GOES R, S, T, U*. Gunter's Space Page. https://space.skyrocket.de/doc_sdat/goes-r.htm

Docs.python.org/2/library/glob.html. (2020). https://docs.python.org/2/library/glob.html

Docs.python.org/3/library/datetime.html. (s/f). https://docs.python.org/3/library/datetime.html

Docs.python.org/3/library/zipfile.html. (2020). https://docs.python.org/3/library/zipfile.html

Downey, A., Elkner, J., & Meyers, C. (2002). *Aprenda a Pensar como un Programador con Python*. Wellesley, Massachusetts: Green Tea Press.

ESA. (2014). *¿Qué es la teledetección?* ESA - Eduspace. http://www.esa.int/SPECIALS/Eduspace_ES/SEMO1U3FEXF_0.html

Gdal.org. (2020). https://gdal.org/

Geopandas.org. (2020). https://geopandas.org/

gnu.org. (s/f). *GNU Operating System*. El sistema operativo GNU. https://www.gnu.org/home.es.html

Guitián Rivera, L. (1999). *Incendios históricos. Una aproximación multidisciplinar* (Araque Jimenez Eduardo, Ed.). Gráficas La Paz. https://www.unia.es/explorar-catalogo/item/incendios-historicos

IEEE. (1993). *IEEE Software Engineering Standard 729-1993: Glossary of Software Engineering Terminology*. Computer Society Press.

INEGI. (s/f). *Explotación forestal*. http://cuentame.inegi.org.mx/economia/primarias/forestal/default.aspx?tema=E

Jenner, L. (2017). *GOES Overview and History* [NASA]. GOES Satellite Network. https://www.nasa.gov/content/goes-overview/index.html

Juganaru Mathieu, M. (2014). *Introducción a la programación* (Primera Edición). Patria. https://editorialpatria.com.mx/pdffiles/9786074384154.pdf

Liew, S. (2001). *Principles of Remote Sensing, Centre for Remote Imaging, Sensing and Processing*.

Manzo Delgado, L. de L. (2020). *DETECCIÓN DE INCENDIOS DE VEGETACION PARA MÉXICO UTILIZANDO IMÁGENES GOES-16/ABI: DESCRIPCIÓN DEL ALGORITMO Y EVALUACIÓN INICIAL*.

NASA. (2020). *Fire Information for Resource Management System (FIRMS)*. https://earthdata.nasa.gov/. https://earthdata.nasa.gov/earth-observation-data/near-real-time/firms

Neuman Ezell, L. (1988). *NASA HISTORICAL DATA BOOK Volume Ill Programs and Projects 1969-1978*. https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19880016046.pdf

NOAA, & NASA. (s/f). *Goes-r*. https://www.goes-r.gov/

NOAA, & NASA. (2017). *ABI BANDS QUICK INFORMATION GUIDES*. GOES-R. https://www.goes-r.gov/mission/ABI-bands-quick-info.html

NOAA, U.S. Department of Commerce, & NASA. (s/f). *NOAA GOES-N,O,P — The Next Generation*. https://www.nasa.gov/pdf/112855main_GOESNOPWeb1.pdf

Numpy.org. (2020). https://numpy.org/

opensource.org. (s/f). *Open Source Initiative*. https://opensource.org/

Pandas.pydata.org. (2020). https://pandas.pydata.org/

Plana Bach, E., Font Bernet, M., & Serra Davos, M. (2016). *Los incendios forestales, guía para comunicadores y periodistas* (p. 32). Centro Tecnológico Forestal de Cataluña. http://efirecom.ctfc.cat/docs/efirecomperiodistes_es.pdf

Pypi.org/project/Shapely. (2020). https://pypi.org/project/Shapely/

Pythonhosted.org/rasterstats. (s/f). https://pythonhosted.org/rasterstats/

Requests.readthedocs.io/en/master. (s/f). https://requests.readthedocs.io/en/master/

Rodríguez Trejo, D. A. (2012). *Génesis de los incendios forestales*. http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S2007-40182012000300008

Sánchez Gonzáles, L. J. (2009). *Manual práctico de Linux con ejercicios*.

Schmit, T., Griffith, P., Gunshor, M., Daniels, J., Goodman, S., & Lebair, W. (2016). A closer look at the ABI on the goes-r series. *Bulletin of the American Meteorological Society*, *98*. https://doi.org/10.1175/BAMS-D-15-00230.1

Schmit, T., Lindstrom, S., Gerth, J., & Gunshor, M. (2018). Applications of the 16 spectral bands on the Advanced Baseline Imager (ABI). *Journal of Operational Meteorology*, *06*, 33–46. https://doi.org/10.15191/nwajom.2018.0604

Scipy.org. (2020). https://www.scipy.org/

Sieglaff, J., & Schmit, T. (2003). *Vegetation monitoring and thin cirrus detection on the next generation GOES imager*.

Stallman, R. (1992). *Software libre para una sociedad libre*.

Sulla-Menashe, D., & A Friedl, M. (2018). *User Guide to Collection 6 MODIS Land Cover (MCD12Q1 and MCD12C1) Product*. https://lpdaac.usgs.gov/documents/101/MCD12_User_Guide_V6.pdf

Suomi, V., & Parent, R. (1968). A color view of planet Earth. *Bulletin of the American Meteorological Society*, *49*, 74–75. https://doi.org/10.1175/1520-0477-49.2.74

Unidata.github.io/netcdf4-python. (s/f). https://github.com/Unidata/netcdf4-python

Yung, Y. (2003). An introduction to atmospheric radiation. By K. N. Liou. Academic Press. Second edition, 2002. Pp. Xiv+583. ISBN 0 12 451451 0. *Quarterly Journal of The Royal Meteorological Society - QUART J ROY METEOROL SOC*, *129*, 1741–1741. https://doi.org/10.1256/003590003102695746