UNIVERSIDAD DE PIURA

FACULTAD DE INGENIERÍA

Hacia una cadena de procesamiento de datos de un radar meteorológico. Perspectivas para el radar PIUXX

Tesis para optar el Título de
Ingeniero Mecánico - Eléctrico

Elmer Jeanpierre Lopez Ramirez

Asesor:
Dr. Ing. Rodolfo Rodríguez Arisméndiz

Piura, noviembre de 2021

UNIVERSITAS STUDIORUM PIURENSIS

Resumen

La ocurrencia del Fenómeno El Niño Costero del 2017 motivó a investigadores de la Universidad de Piura a implementar un radar de lluvias en su campus. Este radar proporciona datos de reflectividad que requieren de diversas correcciones hasta su conversión a valores de precipitación. Estas correcciones, que reciben el nombre de cadena de procesamiento, se encuentran configuradas de manera predeterminada por el software adquirido conjuntamente con este radar. Al tratarse del primer radar de lluvias permanente del país, no existe un panorama claro de cuales son estas correcciones y los problemas que mitigan.

El estudio se realizó a través de una serie de etapas secuenciales. Se elaboró una base conceptual de los radares meteorológicos y las correcciones que existen en la literatura científica. Posteriormente, se evaluó la cadena de procesamiento predeterminada y se buscaron opciones para mejorar su desempeño. Por último, se proporcionan lineamientos hacia una cadena de procesamiento alternativa que comprende librerías existentes y herramientas adicionales programadas en Python.

La cadena de procesamiento predeterminada, debido a su carácter de implementación en tiempo real, se centra en procedimientos sencillos. Por otra parte, se encontraron diversas librerías que implementan partes de la cadena de procesamiento.

Es posible mitigar problemas adicionales en el radar a través del uso de cadenas de procesamiento alternativas. La cadena propuesta representa un avance importante en la visualización de los datos, la corrección del clutter y la predicción a corto plazo.

Tabla de contenido

Lista de tablas

Lista de figuras

Introducción

El radar meteorológico es una excelente herramienta para el monitoreo y predicción a corto plazo de la precipitación. Este genera valores de reflectividad que permiten estimar valores de precipitación con una resolución espacial mayor que la red pluviométrica local. Estos datos representan una gran oportunidad para aplicaciones como predicción, gestión de riesgos, manejo de cuencas, actividad agrícola y otros.

Como con cualquier otra fuente, los datos no están exentos de errores e incertidumbres. En el caso del radar estos se presentan por la resolución variable, el efecto de los lóbulos laterales, la atenuación y los retornos de ecos no meteorológicos (clutter), por nombrar los más importantes.

Estos errores y problemas se mitigan a través de una serie de correcciones conocidas como cadena de procesamiento. Sin embargo, los métodos de corrección no son perfectos y si son mal utilizados pueden inducir aún más errores en lugar de eliminarlos. El software del radar presenta procedimientos generales para corregir algunos de los errores. Dependiendo de las características del radar y del lugar donde se encuentre ubicado, estas correcciones pueden no ser suficientes. En consecuencia, es necesario evaluar el desempeño de las correcciones en los datos y comprobar si estas son suficientes bajo las condiciones locales. De no ser el caso, se requiere buscar o generar alternativas para el procesamiento de los datos.

La presente tesis se enfoca en abordar dicha problemática a través de una revisión de los posibles errores presentes en los datos del radar y la evaluación de la cadena de procesamiento predeterminada y sus opciones.

Para una mejor comprensión del contenido de la tesis, se detalla el contenido de cada capítulo.

En el primer capítulo, se realiza una breve introducción al contexto en el cual se desarrolla esta tesis; se explica la metodología y los casos que se utilizarán para mostrar las correcciones de los errores.

En el segundo capítulo, se explica el funcionamiento de los radares, su desarrollo, así como el software predeterminado para el procesamiento de los datos.

En el tercer capítulo, se explican las fuentes de error, los métodos para su corrección y su impacto en la calidad de los datos.

En el cuarto capítulo, se evalúa el desempeño de la cadena de procesamiento predeterminada, sus virtudes y limitaciones.

En el quinto capítulo, se evalúan los desarrollos de software disponibles para el procesamiento alternativo comparándose entre sí y con la cadena predeterminada.

En el sexto capítulo, se propone una cadena de procesamiento alternativa que aborda los errores presentes en los datos.

Por último, se presentan las conclusiones, recomendaciones y líneas de acción para trabajos futuros.

Capítulo 1
Aspectos generales

Con el adelanto tecnológico en la electrónica se desarrollaron sensores y modelos de predicción meteorológica que produjeron predicciones razonables de la precipitación. Estos forman parte importante en la planificación de actividades a pequeña escala como reuniones y eventos (fiestas y viajes familiares) y a gran escala como la agricultura o la gestión de riesgos y desastres.

Se solía pensar que los modelos serían suficientes para predecir las condiciones meteorológicas a largo plazo con suficiente exactitud en tanto más y más variables fuesen ingresadas. Con los descubrimientos de Edward Lorentz, iniciador de la teoría del caos, se demostró que las variables meteorológicas conformaban un sistema caótico. Esto implica que la resolución de la predicción se degradará de forma inexorable a medida que el horizonte de predicción sea cada vez más grande.

Las predicciones se dividieron de acuerdo con el horizonte temporal de predicción. Así la predicción de corto plazo se ciñe al orden de horas mientras que las predicciones a medio y largo plazo presentan ordenes mayores (días, semanas o meses). Estas últimas han sido cubiertas por los modelos de predicción numérica. En cambio, el radar meteorológico y sus productos se utilizan para la predicción a corto plazo y muy corto plazo con resultados superiores a estos modelos.

Pese a la gran oportunidad que representa el radar, garantizar la calidad de sus productos requiere de diversas correcciones, que se implementan bajo el nombre de cadena de procesamiento.

En esta tesis se analiza la cadena de procesamiento actual y se presentan las alternativas disponibles hacia una cadena de procesamiento que mitigue los errores presentes en las mediciones del radar meteorológico. Este capítulo tratará de los objetivos, justificación, estructura y metodología de este documento.

1.1 Objetivos

- Analizar las necesidades de la cadena de procesamiento local
- Evaluar la cadena de procesamiento actual del radar

- Proponer alternativas para la implementación de una cadena de procesamiento alternativa

1.2 Marco contextual

El clima normal de la costa norte del Perú puede verse afectado por tres posibles condiciones océano-atmosféricas: el fenómeno El Niño, el fenómeno La Niña y el fenómeno de El Niño Costero. El fenómeno El Niño (FEN) es un evento climático cíclico caracterizado por el aumento generalizado de la temperatura superficial del mar en el sector Oriental y Central del Pacifico Ecuatorial por más de doce meses. El fenómeno La Niña es la contraparte del FEN. Se caracteriza por el descenso de la temperatura superficial del sector Oriental Central del Océano Pacifico. Como consecuencia, se presentan fuertes sequías en América del Sur. El fenómeno El Niño Costero, en cambio, es de naturaleza local. Esta caracterizado por el aumento anómalo de las aguas del mar cercanas a las costas de Sudamérica (norte del Perú y sur del Ecuador). Sin embargo, a pesar de tener menor duración, sus consecuencias son similares a las del FEN, por lo que muchas veces son confundidos.

En presencia de cualquiera de los fenómenos El Niño se producen fuertes precipitaciones en los meses de verano que afectan la costa norte del Perú, que superan considerablemente los acumulados mensuales promedio. En consecuencia, se produce un aumento del caudal de ríos y quebradas en esta zona del país. Debido a la alteración climática, usualmente aumentan los casos de enfermedades como el cólera y dengue. Las actividades agrícolas y pesqueras resultan severamente afectadas. Todo ello origina cuantiosas pérdidas para la región.

Durante el verano del 2017, Piura sufrió los estragos del fenómeno El Niño Costero. El 27 de marzo, luego de una fuerte lluvia de casi 12 horas, el caudal del río Piura aumento hasta alcanzar un máximo de 3468 m/s. El río Piura se desbordó en la parte más baja de su cuenca inundando los distritos de Piura, Castilla, Catacaos, Cura Mori, La Arena y El Tallán. Este evento dejó 89.709 damnificados; 18 personas fallecidas, 375.265 personas afectadas y 83.957 viviendas afectadas en la región (INDECI, 2017).

Posterior al desastre, diversas instituciones tomaron acciones para ayudar a prevenir futuros eventos similares. La Universidad de Piura propuso la adquisición de un radar escaneador de lluvias. Bajo la dirección del Dr. Rodolfo Rodríguez se formuló el proyecto “Implementación de un radar escáner de lluvias a tiempo real como sistema de alerta temprana para previsión de inundaciones, investigaciones eco-climáticas y ayuda a la agricultura en la región Piura, la más impactada por El Niño”.

El equipo científico de ese proyecto estuvo integrado por los doctores Rodolfo Rodríguez, Raúl La Madrid, César Chinguel, Antonio Mabres, Gastón Cruz; la magíster Roxana Fernández y el bachiller Eddie Valdiviezo (UDEP); así como por los doctores Danny Scipión, del Instituto Geofísico del Perú (IGP) y Ruetger Rollenbeck de la Universidad de Marburg (Alemania).

El proyecto fue cofinanciado por Innóvate Perú y la Universidad de Piura bajo la convocatoria de Equipamiento Científico. Además, el proyecto tuvo la colaboración de Petroperú, la Asociación Peruana de Productores y Exportadores de Mango (APEM) y la Universidad de Marburg.

El radar, denominado PIUXX, se terminó de implementar en el campus de la Universidad de Piura a finales del mes de abril del 2019 y fue inaugurado el 10 de mayo. Servirá para monitorear las precipitaciones, así como para realizar investigaciones de los mecanismos de formación y dinámica de la precipitación a nivel regional.

1.3 Metodología

Los procedimientos en la cadena de procesamiento están asociados a cierto grado de incertidumbre. Sin otras fuentes de información, o a veces incluso con ellas, no es posible cuantificar totalmente los efectos de las correcciones. Debido a ello, durante el desarrollo de esta tesis se consideran tres casos para una evaluación por lo menos cualitativa. Estos consisten en datos de reflectividad con las mínimas correcciones (dBuZ), proporcionados por el radar PIUXX con un alcance de 100 km en una grilla polar de 360 x 1000 valores. Estos datos se representan en coordenadas polares para formar imágenes donde los valores de reflectividad están representados por una escala de colores (Figura 01 y Figura 02). En ellos, se muestra la aparición de ecos no meteorológicos, en adelante clutter, y ecos meteorológicos correspondientes a precipitación. Los casos seleccionados representan tres situaciones significativas en función de la presencia de clutter y ecos meteorológicos.

El caso 1, denominado “Solo Clutter”, consiste en una situación típica de condiciones sin precipitación. Los valores de reflectividad en la zona central (Figura 02 izquierda) corresponden a ecos no meteorológicos provenientes de objetos ubicados en las cercanías del radar como árboles, edificios, torres de agua, etc. Dado que estos objetos son estáticos, los valores de reflectividad en esta zona deberían aparecer siempre en los datos.

El caso 2, denominado “Precipitación tipo 1”, consiste en datos de clutter bajo condiciones de propagación anómala. Los datos escogidos corresponden a una ligera lluvia ocurrida el 14 de mayo del 2019. Este caso fue seleccionado porque la precipitación se encuentra fuera del área de acción usual del clutter (Figura 01 central), con lo cual la remoción del clutter resulta relativamente sencilla.

El caso 3, denominado “Precipitación tipo 2”, consiste en datos de precipitación durante la lluvia del 14 de mayo del 2019. Una de las celdas de precipitación se encuentra superpuesta sobre la ubicación del clutter (Figura 01 y 02 derecha). Debido a ello, la distinción entre el clutter y la precipitación es difusa; complicando la eliminación del clutter.

Fig. 1 Casos de estudio. Imágenes polares de la reflectividad en el alcance del radar (100 km)

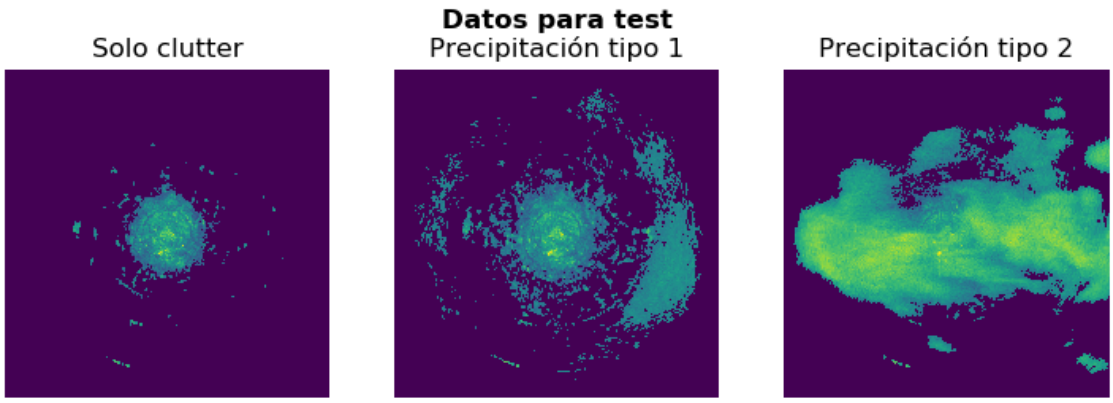


Fig. 2 Casos de estudio en detalle. Acercamiento al área central, representada es un cuadrado de 20 km de lado.

Capítulo 2
Radar meteorológico

El radar meteorológico es un radar diseñado específicamente para la detección de hidrometeoros, ya sea en forma de lluvia, granizo o nieve.

Los radares surgieron previamente a la Segunda Guerra Mundial. Inicialmente su aplicación fue netamente militar: la detección de aviones enemigos. Sin embargo, los radares a veces detectaban ecos en el cielo que no correspondían a la presencia de naves enemigas. A estos ecos sin razón aparente se les denomino “ángeles”. Posteriormente se descubrió que estos ecos eran causados por nubes de precipitación y grupos de aves (Rosengaus, 1995).

Tras la guerra, los radares pasaron de las aplicaciones militares hacia las civiles. Una de las primeras aplicaciones fue la meteorología, donde los radares se adaptaron para la detección de los hidrometeoros. Es así como nació propiamente el radar meteorológico.

Desde entonces los radares meteorológicos han evolucionado hasta convertirse en herramientas indispensables para los meteorólogos. A partir del desarrollo de la tecnología digital se incorporaron los registros de los datos, antes solo era posible almacenarlos mediante fotografías tomadas a las pantallas. La tecnología Doppler permitió estimar la velocidad radial de la atmosfera mientras que la doble polarización sirve para la clasificación de los hidrometeoros, así como para la eliminación de ruidos.

Las principales agencias de meteorología formaron extensas redes de radares para monitorear su territorio (Figura 03). Sin embargo, el alto costo de los radares hizo que solo los países plenamente desarrollados pudiesen permitirse la adquisición de estos. No es sino hasta años recientes, cuando la adaptación de radares marinos ofreció alternativas económicas para países en vías de desarrollo como el nuestro.

En Latinoamérica la implementación de los radares meteorológicos es reciente y dispar entre países. Países como Brasil, Venezuela y Argentina poseen actualmente una red de radares que cubre buena parte de su territorio. En cambio, Paraguay y Bolivia tienen radares antiguos y que requieren renovación. Los países restantes planean adquirir o fabricar sus propios radares. Algunos países poseen radares manejados por entidades privadas como universidades y centros de investigación como Ecuador.

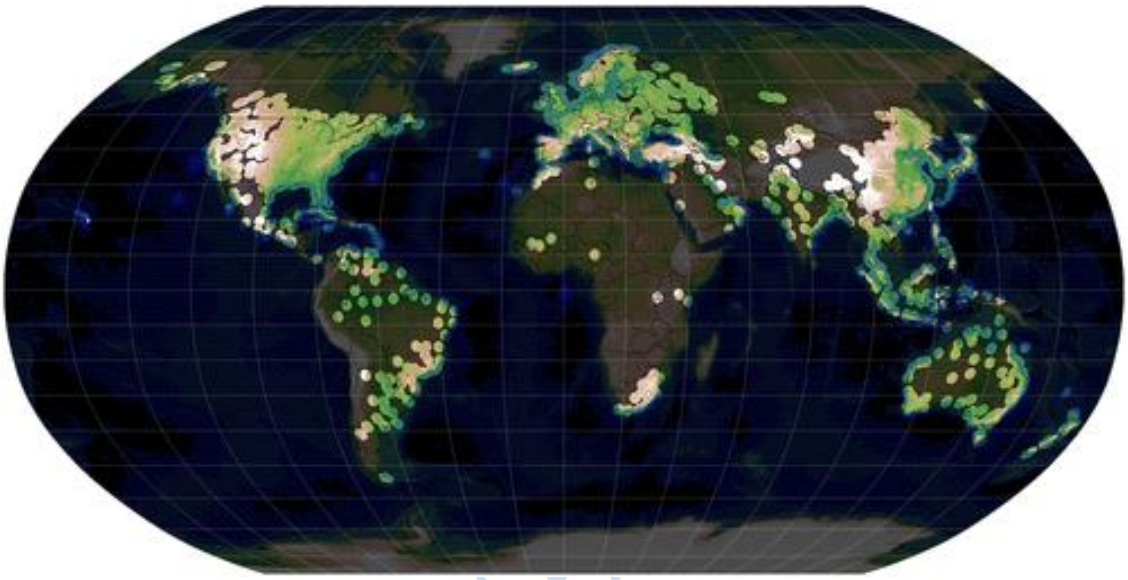


Fig. 3 Estimación de la cobertura de radares meteorológicos en el mundo
Fuente: Saltikoff et al. (2019)

En el Perú, las iniciativas de adquirir radares meteorológicos son muy recientes. En el 2018, el Instituto Geofísico del Perú trajo un radar bajo el marco de cooperación con el Advanced Radar Research Center de la Universidad de Oklahoma. Este radar fue instalado en la sierra de Lima y funciono solo por espacio de unos meses. De similar forma, la Fuerza Aérea del Perú trajo un radar durante inicios del 2019 en calidad de préstamo de una empresa norteamericana. En ese sentido, el radar instalado en la Universidad de Piura representa el primer radar meteorológico permanente del país.

2.1 Principio de funcionamiento

El radar meteorológico está basado en la tecnología homónima, acrónimo de "RAdio Detecting And Ranging" (Detección y localización por radio). El radar emite ondas electromagnéticas mediante su antena transmisora en una dirección determinada. A través del giro de la antena, se realiza un barrido que escanea la atmosfera cercana en busca de blancos meteorológicos. Cuando uno de estos blancos es alcanzado, la onda se refleja en múltiples direcciones dependiendo de la naturaleza del blanco y de la onda. Una pequeña parte de la energía dispersada es reflejada en la misma dirección de dónde provino y captada por la antena receptora del radar. Luego dicha señal es procesada para determinar la posición del blanco y su intensidad.

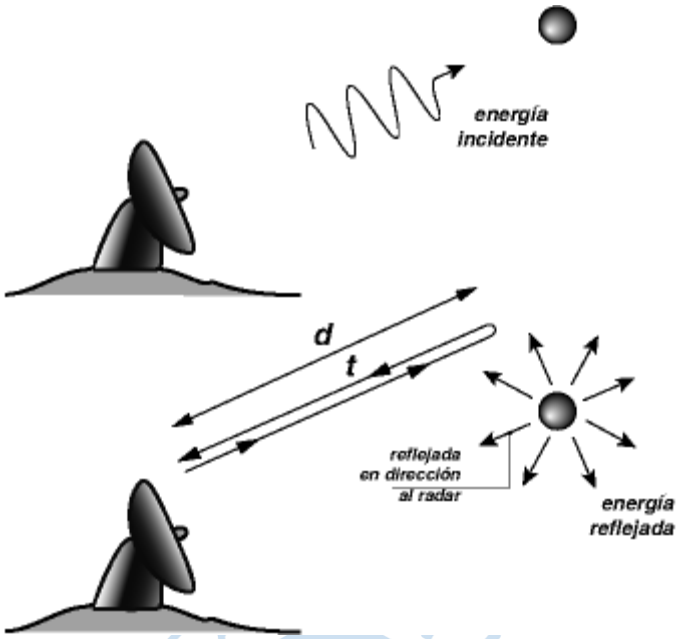


Fig. 4 Esquema de la energía captada y reflejada por un blanco.
Fuente: Sánchez – Diezma & Corral (2000)

La señal pulsada viaja a una velocidad muy cercana a la velocidad de la luz. A partir del tiempo transcurrido entre la emisión y recepción de la señal se calcula la distancia desde el radar aplicando la siguiente fórmula.

$$d = \frac{1}{2} c \times t \qquad \textbf{Ec. 01}$$

Donde d representa la distancia desde el radar al blanco atmosférico, c representa la velocidad de la luz en el aire y t el tiempo transcurrido entre la emisión y recepción de la señal. El factor de dos es consecuencia de que la señal recorre la distancia dos veces: ida y vuelta desde el radar al blanco atmosférico (Figura 04).

Los radares comerciales trabajan a distintas frecuencias. Las frecuencias de trabajo típicas son del orden de 10^6 Hz o Mega Hertz (MHz) o incluso valores superiores. Los radares militares, por el tipo de blancos a detectar y la distancia que abarcan, utilizan las frecuencias más bajas y de mayor longitud de onda. Estos radares usualmente trabajan con las bandas HF, VHF, UHF, L, S, C y X. Los valores relativos de las frecuencias y longitudes de onda de estas bandas se ilustran en el Figura 05. Los radares meteorológicos, en cambio, usan bandas de frecuencia con longitudes de onda menores. Estos trabajan con las bandas de frecuencia S, C, X y en ocasiones con la banda K.

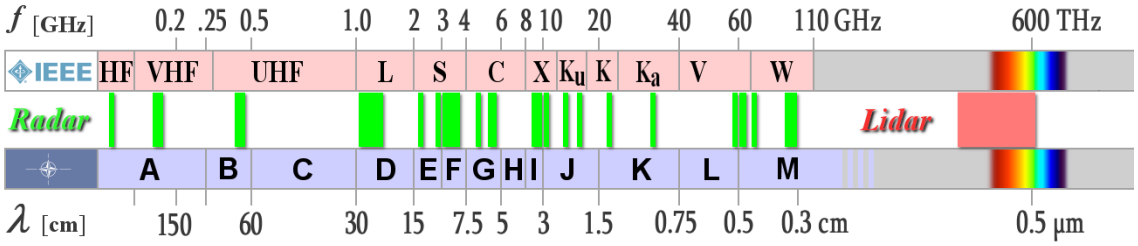


Fig. 5 Bandas de frecuencias usuales de los radares comerciales y sus correspondientes longitudes de onda (λ)

Fuente: Burns (2015)

La elección de la banda de frecuencia del radar es siempre una solución de compromiso. Las bandas con longitudes de onda más pequeñas son capaces de detectar hidrometeoros con un menor tamaño en comparación a radares de longitudes de onda más grandes. Sin embargo, sufren problemas de atenuación de la señal con la distancia, lo que limita su alcance. Los radares de longitudes de onda más grandes en cambio son menos sensibles, pero son menos afectados por la atenuación. Por ello tienen alcances nominales más grandes. La elección de la banda de frecuencia es una solución intermedia entre la sensibilidad y el alcance pretendido.

Los radares banda S son prácticamente insensibles a la atenuación. Se utilizan en lugares donde la precipitación es fuertemente convectiva y las gotas de agua son grandes. Sin embargo, ello obliga a usar antenas de mayor tamaño y el coste del equipo se eleva. El alcance nominal es superior a los 250 km.

Los radares banda C son usados en países ubicados en latitudes altas, en los que los eventos meteorológicos no son tan intensos. Por ello, se tiende más hacia la sensibilidad del radar para capturar precipitaciones ligeras. Estos radares ya sufren problemas de atenuación, por lo que su alcance oscila entre los 130 y 250 km.

Los radares banda X son muy sensibles a la atenuación. Por ello pese a su gran sensibilidad, al principio no generaban interés práctico para las aplicaciones meteorológicas. Posteriormente fueron utilizados para monitorear precipitación con alcances reducidos de entre 30 y 60 km. Solo en años recientes los radares banda X han logrado alcances de hasta 120 km. Debido a su reducido tamaño en comparación a los otros, se puede utilizar en aplicaciones móviles como complemento de otros radares.

Los radares de banda K se usan para la detección de nubes y precipitaciones. Debido a la atenuación tienen alcances cortos de unos pocos kilómetros.

Como consecuencia del amplio alcance y la poca atenuación, los radares de bandas S y X son los más utilizados en el mundo (Figura 06). La elección entre uno y otro está fuertemente vinculada a la latitud en la que se encuentra ubicado cada radar. De esta forma, los radares banda C abundan en latitudes altas, mientras los radares banda S se encuentran

en mayor proporción en latitudes medias. Por otro lado, los radares banda X solo representan un pequeño porcentaje del total de radares. Note también la poca cantidad de radares meteorológicos presentes en latitudes ecuatoriales.

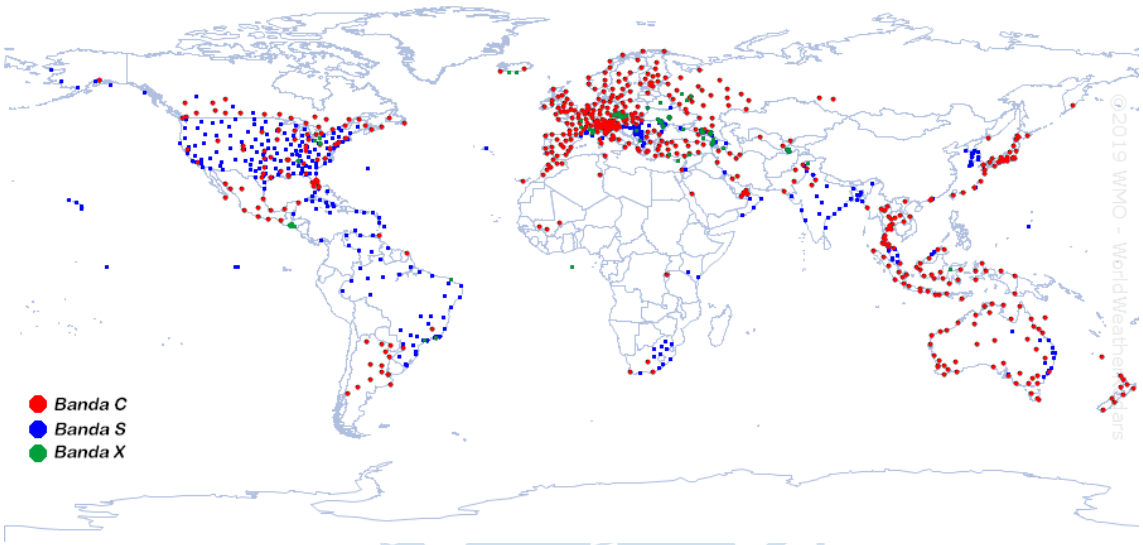


Fig. 6 Distribución de radares.
Fuente: WMO Radar Database

2.2 Ecuación del radar

La ecuación del radar estima la potencia recibida por la antena del radar P_r [W]. Si se considera la emisión isotrópica de la potencia transmitida P_t [W] por parte de la antena transmisora, la densidad de potencia P_S [W/m²] sobre la superficie esférica a una distancia r de la antena puede ser determinada como:

$$P_S = \frac{P_t}{4\pi r^2} \qquad \textbf{Ec. 02}$$

Para generalizar la ecuación si la antena no es isotrópica, se multiplica esta expresión por la ganancia de la antena G. La potencia incidente P_g [W] sobre una porción del área A_t de dicha superficie esférica será:

$$P_\sigma = \frac{P_t \times G \times A_t}{4\pi r^2} \qquad \textbf{Ec. 03}$$

La potencia efectivamente recibida por la antena estará en función de su sección transversal o área efectiva A_e. Para determinar esta densidad de potencia, consideraremos el blanco como un emisor isotrópico de la potencia P_σ y a la antena como blanco de área A_e.

$$P_r = \frac{P_\sigma \times A_e}{4\pi r^2} = \frac{P_t \times G \times A_t \times A_e}{\left(4\pi r^2\right)^2} \qquad \textbf{Ec. 04}$$

El área efectiva A_e es equivalente a:

$$A_e = \frac{G\lambda^2}{4\pi} \qquad \textbf{Ec. 05}$$

Donde λ representa la longitud de onda del radar. Reemplazando la ecuación, se tiene:

$$P_r = \frac{P_t \times G^2 \times \lambda^2 \times A_t}{(4\pi)^3 \times r^4} \qquad \textbf{Ec. 06}$$

La ecuación anterior resume el comportamiento de la mayoría de los radares no meteorológicos. A diferencia de los objetivos tradicionales de los radares que detectan un solo blanco de tamaño considerable, la precipitación corresponde a la unión de muchos blancos pequeños dispersos en el volumen de muestreo. En consecuencia, el área A_t es reemplazada por una sumatoria de secciones transversales σ :

$$P_r = \frac{P_t \times G^2 \times \lambda^2}{(4\pi)^3 \times r^4} \sum_{i=1}^{n} \sigma_i \qquad \textbf{Ec. 07}$$

Asumiendo que las gotas de agua se encuentran homogéneamente distribuidos en el volumen de muestro V_m (Figura 7) correspondiente a un tronco de cono circular es equivalente a:

$$V_m = \pi \left(r \frac{\Delta\theta}{2} \right) \left(r \frac{\Delta\phi}{2} \right) \frac{c\tau}{2} \qquad \textbf{Ec. 08}$$

Donde τ representa la duración del pulso emitido, mientras que $\Delta\theta$ y $\Delta\phi$ representan el ancho del haz en las dos direcciones perpendiculares. Esta ecuación representa una aproximación del volumen y solo es válida para valores pequeños de ϕ y θ. Dado que los radares utilizan resoluciones de 1° - 2° la ecuación es perfectamente aplicable.

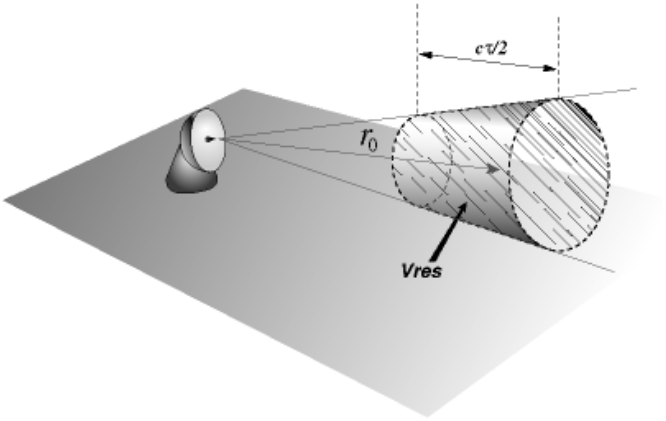


Fig. 7 Volumen de muestreo.
Fuente: Sánchez – Diezma & Corral (2000)

La sumatoria de las secciones transversales se normaliza en el volumen de muestreo:

$$\bar{P}_r = \frac{P_t G^2 \lambda^2 \Delta\theta \Delta\phi c\tau}{512\pi^2 r^2} \sum_{vol} \sigma_i \qquad \textbf{Ec. 09}$$

Se agrega un factor de corrección de $2\ln(2)$ debido a la forma Gaussiana de la emisión de las ondas por parte de la antena transmisora.

$$\bar{P}_r = \frac{P_t G^2 \lambda^2 \Delta\theta \Delta\phi c\tau}{1024\ln(2)\pi^2 r^2} \sum_{vol} \sigma_i \qquad \textbf{Ec. 10}$$

Si los objetivos son mucho más pequeños que la longitud de onda del radar se puede utilizar la teoría de dispersión de Rayleigh para determinar la sección transversal. Bajo esa suposición, la sección transversal está directamente relacionada con el diámetro de las partículas.

$$\sigma = \frac{\pi^5}{\lambda^4} |K|^2 D^6 \qquad \textbf{Ec. 11}$$

Donde $|K|^2$ representa el índice complejo de refracción y D el diámetro de cada partícula. Reemplazando:

$$\bar{P}_r = \frac{\pi^3 P_t G^2 \Delta\theta \Delta\phi c\tau}{1024\ln(2)\lambda^2} \times \frac{|K|^2}{r^2} \times \sum_{vol} D^6 \qquad \textbf{Ec. 12}$$

En ocasiones se agrega un factor dependiente de la distancia para tomar en cuenta el efecto de la atenuación a lo largo de un haz.

La Ecuación 12 consta de tres partes. La primera de ellas está relacionada con factores geométricos y de diseño del radar. Debido a ello, se suele agrupar este factor bajo la constante C del radar. La segunda parte está relacionada con la distancia y el tipo de objetivos a detectar ($|K|^2 \cong 0.93$ para el agua y $|K|^2 \cong 0.197$ para el hielo). La última parte se relaciona con el número de partículas y su tamaño. Esta parte se denomina reflectividad (Z) y es uno de los parámetros básicos proporcionados por el radar. Reescribiendo:

$$\bar{P}_r = C \frac{|K|^2}{r^2} Z \qquad \textbf{Ec. 13}$$

La ecuación anterior pone en evidencia que la potencia recibida varía con el inverso del cuadrado de la distancia. Se diferencia por tanto de los radares tradicionales, en los cuales la potencia recibida varía inversamente a la cuarta potencia de la distancia.

Las ecuaciones presentadas muestran porque a mayores longitudes de onda los equipos resultan mucho más caros. Además, la última ecuación pone en evidencia la pérdida de resolución del radar conforme la distancia aumenta.

2.3 Cadena de procesamiento

La cadena de procesamiento hace referencia a todos los procesos comprendidos desde la obtención de los datos crudos hasta la elaboración productos aprovechables por modelos o para presentación al público. Sin embargo, “datos crudos” y “productos aprovechables” presentan límites muy permeables entre las personas que trabajan con los radares meteorológicos. Así pues, para los fabricantes puede hacer referencia a las correcciones que se hacen desde los voltajes del transmisor hasta su conversión digital en valores de reflectividad. Esto es cierto incluso para los operadores del radar meteorológico. Para algunos puede significar todas las correcciones realizadas a la data tales como la eliminación de clutter, corrección de ecos espurios y atenuación; mientras que para otros tiene que ver con la cuantificación de la calidad o con la generación de productos o con una combinación de estos.

El procesamiento de la data del radar involucra dos pasos sucesivos (Figura 8). El primero de ellos, denominado **procesamiento de la señal** es la extracción de los parámetros crudos del radar como la reflectividad o la velocidad Doppler a partir de las señales del receptor. El segundo paso, llamado **procesamiento de los datos** o **generación de productos** es el procesamiento posterior de los parámetros crudos del radar para obtener información valiosa para propósitos meteorológicos o hidrológicos. En general, estos dos pasos son realizados en computadoras diferentes. El procesamiento de la señal se lleva a cabo localmente, mientras que el procesamiento de los datos se realiza donde los datos son enviados (Büyükbas et al., 2006).

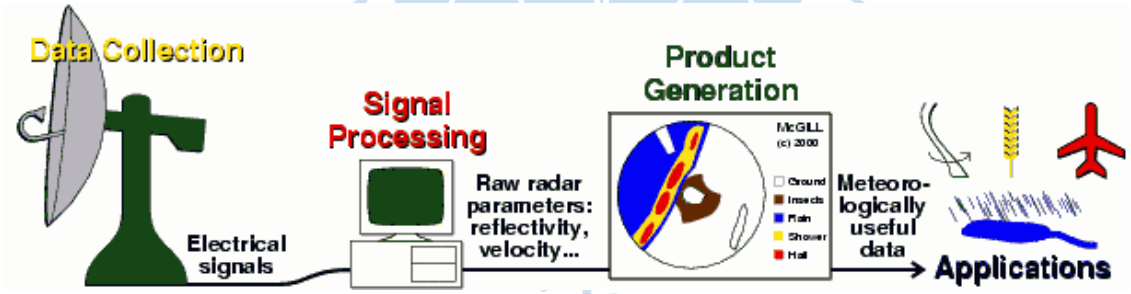


Fig. 8 Obtención, procesamiento de señal y generación de productos.
Fuente: Büyükbas et al. (2006)

Esta descripción equipara los términos procesamiento de los datos con generación de productos, lo cual no es estrictamente cierto. La generación de productos estrictamente no involucra el preprocesamiento de los datos, ni tampoco considera la generación de índices de calidad de los datos, que propiamente corresponden a metadatos y no a productos.

Por ello, en este documento se prefiere el uso del segundo término **procesamiento de los datos** en lugar de **generación de productos**. En ese contexto, “cadena de procesamiento” hará referencia tanto a las correcciones como a la generación de productos y metadatos asociados desde su adquisición hasta productos útiles.

Durante la primera mitad del año 2019, la Organización Meteorológica Mundial (OMM) definió formalmente Niveles de los datos del radar meteorológico. La clasificación propuesta utiliza cinco niveles, que abarcan desde los datos sin procesar en el receptor hasta datos compuestos por múltiples instrumentos (Tabla 1).

Tabla 1 Niveles de los datos del radar meteorológico

Nivel	**Definición**
Nivel 0	Datos de resolución completa de la antena receptora a su frecuencia de muestreo. Generalmente solo disponibles internamente en el sistema. Puede necesitarse equipo especial para medir y guardar este tipo de datos.
Nivel 1	Data en unidades del sensor, también conocida como series de tiempo o datos I/Q (en fase y cuadratura). Producida por los instrumentos de la unidad procesadora de señales(Signal Processor). Generalmente no registrados, excepto por limitadas duraciones en radares operativos. Frecuentemente registrado en radares de investigación.
Nivel 2	Variables derivadas o momentos(reflectividad, velocidad radial, reflectividad diferencial, etc.) a resolución completa después de la agregación y el filtrado. Organizada en coordenadas polares por rayos y compartimientos de rango. También conocida como data de barrido o escaneo volumétrico.
Nivel 3	Productos del radar derivados primariamente de la data Nivel 2. Puede encontrarse en coordenadas polares(Clasificación de hidrometeoros, métricas de calidad, etc.) o en otros sistemas de coordenadas como perfiles verticales o cuadriculas cartesianas (CAPPI, estimaciones de tasas de precipitación, etc.) .
Nivel 4	Productos de orden superior, los cuales pueden incluir datos de múltiples fuentes. Estos incluyen productos compuestos de múltiples radares como mezclas con otras fuentes(satélites, pluviómetros, NWP, etc.).

Fuente: Traducción de (WMO, 2019)

Muchos de los problemas que se presentan en el radar son problemas no completamente resueltos. Estos dependen del tipo de radar escogido, su ubicación y las condiciones atmosféricas. Por ello, cada lugar selecciona las correcciones que se ajustan a sus necesidades. En ese sentido, cada cadena de procesamiento es diferente; no obstante, generalmente todas incluyen procedimientos para la eliminación del clutter y las correcciones del apantallamiento.

Los pasos o etapas en la cadena de procesamiento dependen del tipo de productos que se quieren generar. De acuerdo a las aplicaciones que tendrán los productos, cada operador generará los que necesite o considere relevantes. A su vez, estos condicionan las correcciones realizadas. Por ejemplo, un operador interesado en predicción a corto plazo hace

énfasis en eliminar todos los ecos fijos que disminuyen el rendimiento del algoritmo de flujo óptico que utiliza. En cambio, aquellos que muestran los resultados al público optan por un filtrado que disminuya la tasa de falsos positivos pues de lo contrario pierden credibilidad. Esto hace que la cadena de procesamiento sea un proceso que siempre puede ser mejorado y sobre el cual se trabaja constantemente.

2.4 Descripción técnica del radar PIUXX

En esta tesis se utilizará los datos del radar meteorológico PIUXX de la Universidad de Piura (Figura 09). Este radar se encuentra ubicado dentro del campus de la universidad, específicamente en la Estación Científica Ramón Múgica (05° 10´ 14.0" LS; 80° 38' 18.6" LO, 67 m.s.n.m). El radar utiliza una frecuencia de 9.41 GHz dentro de la banda X de única polarización. No dispone de tecnología Doppler. En la Tabla 2 se indican algunos de sus parámetros.

Fig. 9 Torre metálica y radar PIUXX.
Fuente: Udep

Tabla 2 Parámetros técnicos del radar PIUXX

Parámetro	Valor
Modelo	RS – 120
Frecuencia de operación	9410 ($\pm$30 MHz)
Banda de frecuencias	Banda X
Longitud de onda λ	3 cm
Potencia de pico	25 Kw
Ancho del haz	2°
Alcance máximo	100 km

Intervalo de rango (range bin)	100 m		
Modos de operación	3		
Frecuencia de repetición de pulso (PRF)	1500 Hz	1000 Hz	833 Hz
Ancho del pulso τ	500 ns	700 ns	1200 ns
Longitud de pulso λ_p	75 m	105 m	180 m
Numero de ángulos de elevación	1 (configurable manualmente)		

El radar PIUXX tiene un alcance de 100km y en la región Piura cubre las provincias de Talara, Sullana, Paita, Piura, Sechura y Morropón así como una parte de Ayabaca. Con ello, abarca cerca del 70% de la región (Figura 10).

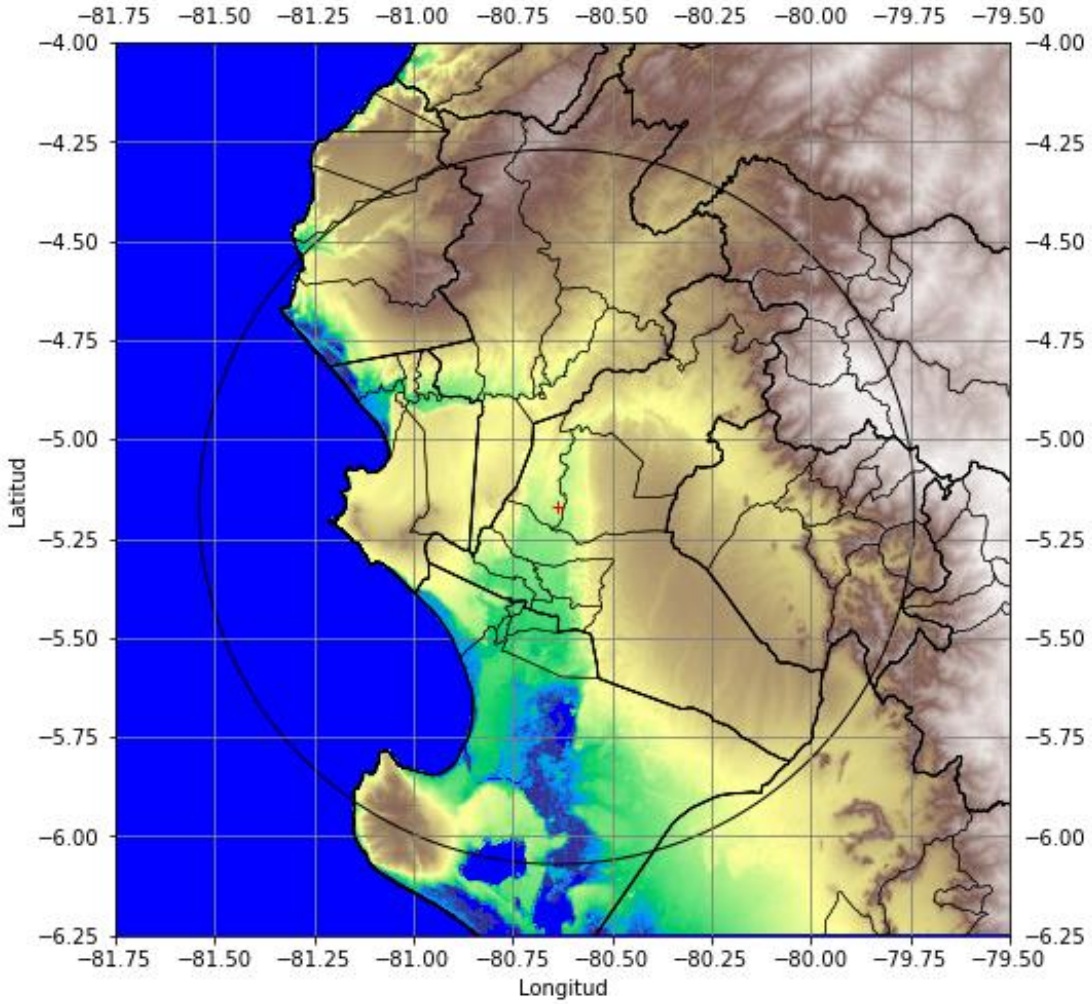


Fig. 10 Alcance del radar PIUXX (Proyección WSG84)

Actualmente, el radar PIUXX trabaja en su tercer modo de operación (Tabla 2). La frecuencia de repetición de pulso (PRF) utilizada corresponde a 833 Hz. La duración y longitud del pulso son 1200 ηs y 180 metros respectivamente. La antena del radar presenta un ángulo

de elevación de +2°. La adquisición de los datos se lleva a cabo en intervalos regulares de 5 minutos.

2.5 Software

La operación del radar PIUXX se hace desde una PC que utiliza el software Rainbow 5 como sistema de gestión de los datos. Este sistema fue adquirido conjuntamente con el radar. Cuenta con tres programas principales y una serie de módulos complementarios. Mediante el programa Rainview se gobierna el funcionamiento al radar y se puede visualizar los datos crudos. El programa Rainview Analyzer se utiliza para generar productos operacionales derivados y realizar conversiones a otros formatos. Por último, a través de RainScoutMet se puede visualizar tanto los datos crudos como los productos generados.

2.5.1 Rainbow 5. El programa Rainbow permite definir la estrategia de escaneo y la visualización tanto de los principales parámetros como de la data sin procesar. A través de su sencilla interfaz permite gestionar el funcionamiento de más de un solo radar.

Entre los parámetros configurables (Figura 11) se encuentran: el intervalo de rango, la corrección del rango, el alcance máximo registrado, el modo de operación y algunas configuraciones básicas para los filtros.

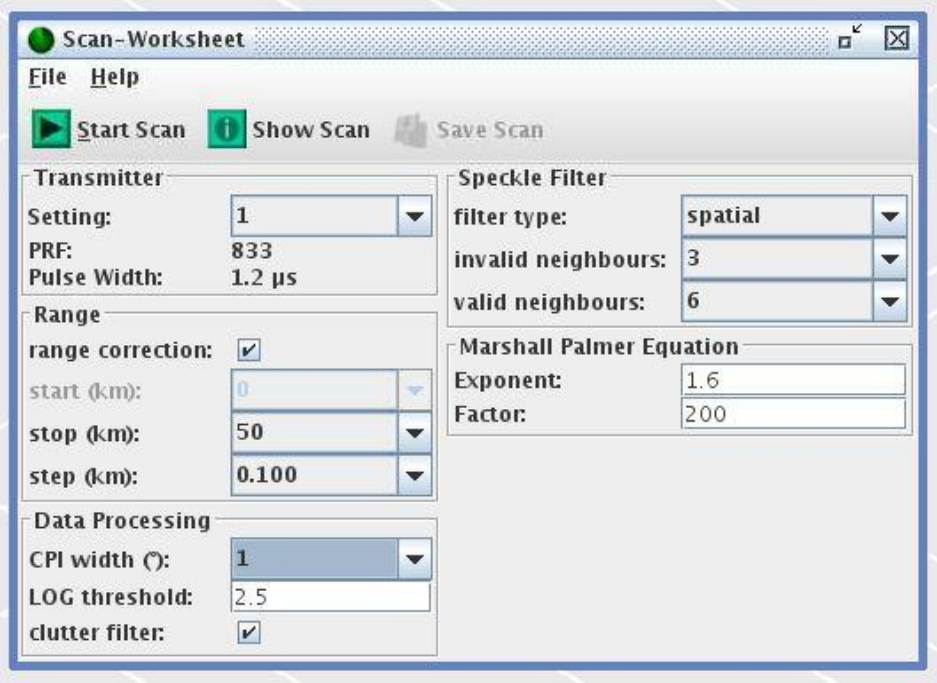


Fig. 11 Configuraciones disponibles.

A través de su pestaña BITE, acrónimo de Built in Test Equipment, se pueden leer los principales parámetros de operación del radar durante la operación. Esto permite monitorear el funcionamiento del radar y detectar posibles alertas y fallas. Sin embargo, dicha información no está disponible al usuario más que para visualización, por lo que no se puede

usar para analizar la incidencia y naturaleza de las fallas. También es posible visualizar alguno de los parámetros de diseño del radar, sus configuraciones así como su ubicación geográfica mediante las otras pestañas (Figura 12).

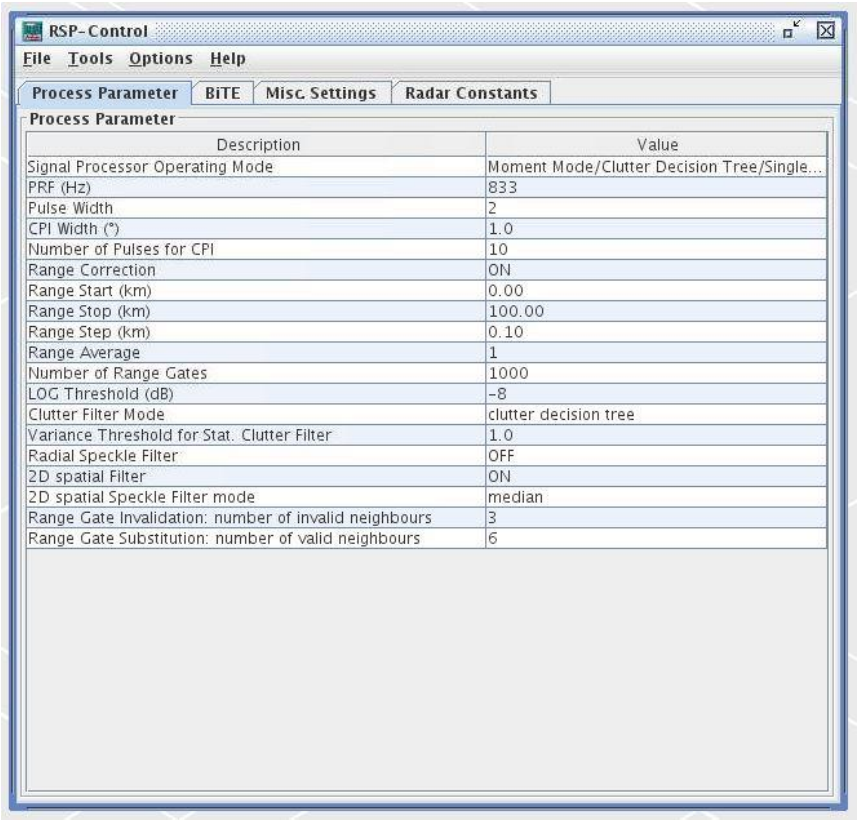


Fig. 12 Parámetros durante la operación.

La visualización de los datos se puede realizar de dos maneras distintas. La primera de ellas consiste en el clásico PPI (Plano Indicador de Posición). Esta visualización permite opciones de zoom y niveles de resolución (Figura 13). Esta representación está disponible para los datos de reflectividad con o sin las correcciones mínimas, así como para la tasa de precipitación basándose en la ecuación de Marshall – Palmer ($Z = 200\ R^{1.6}$). Los parámetros *a* y *b* de la relación Z-R ($Z = Ar^b$) pueden ser modificados en la configuración de acuerdo con la calibración que el usuario realice.

La principal diferencia entre los datos crudos de reflectividad no corregida (dBuZ) y los corregidos (Dbz) radica en la aplicación del filtrado y la eliminación de los datos dentro de la zona anterior al rango mínimo detectable. Los datos en valores de intensidad de precipitación se obtienen a partir de los datos de reflectividad corregida.

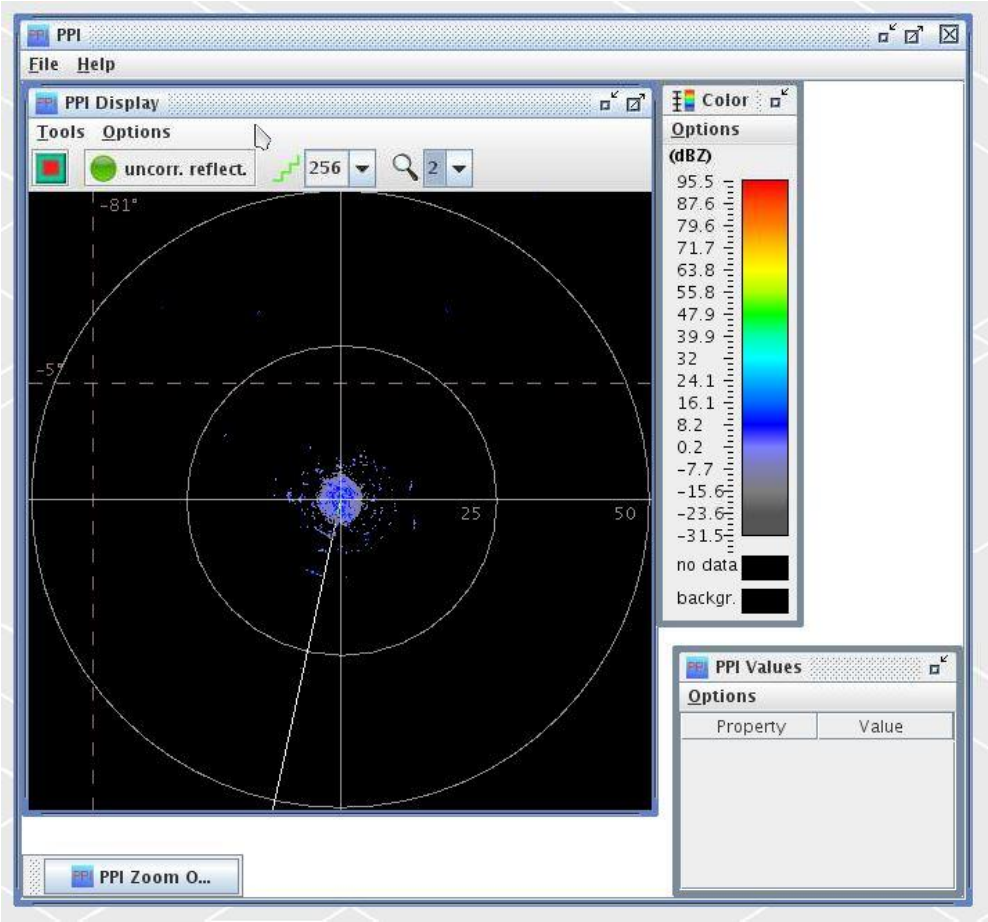


Fig. 13 Visualización PPI.

La otra visualización disponible muestra la reflectividad a lo largo de cada haz (Figura 14). Esto permite reconocer de manera sencilla el alcance máximo en el cual se recibe la señal de un blanco meteorológico.

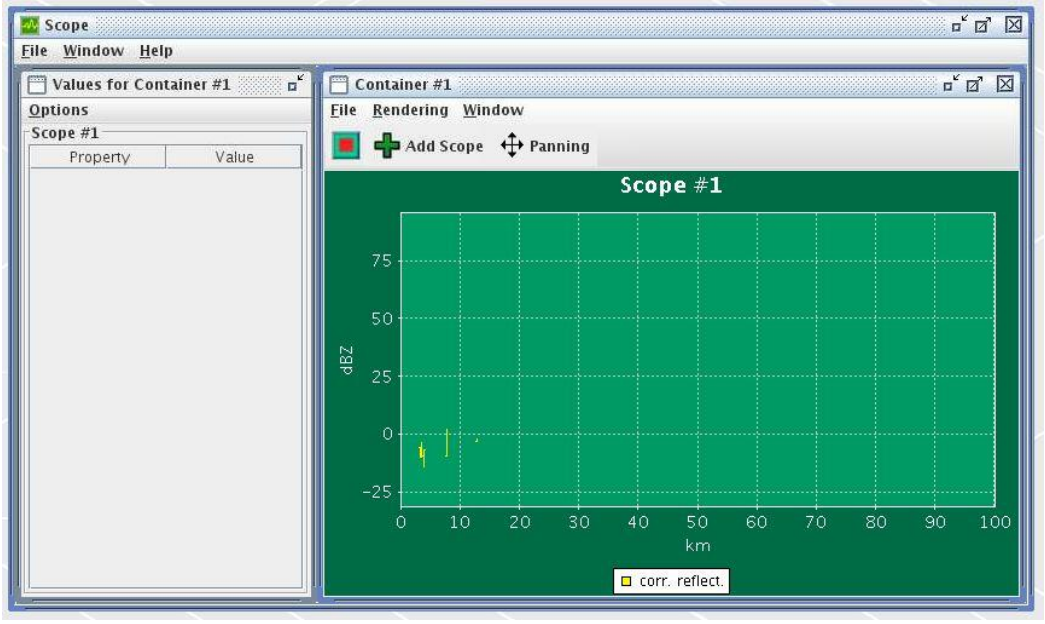


Fig. 14 Visualización del haz de reflectividad recibida.

2.5.2 Rainview Analyzer El programa Rainview Analyzer está destinado a procesar la data cruda y generar productos derivados a partir de esta. El programa puede funcionar tanto en modo online con los datos que se están generando, o en modo offline a partir de una selección de datos por parte del usuario.

El programa funciona a partir de un sistema de tareas "Tasks" apilables para generar productos. Cada tarea suele presentar unos pocos parámetros de configuración. Las tareas son divididas en 4 tipos: pre – procesamiento, productos de primer nivel, productos de segundo nivel y post – procesamiento.

En su catálogo, Rainbow 5 detalla los productos de forma diferente, acorde con sus aplicaciones y correcciones. Estos comprender los productos estándar, productos hidrológicos y de predicción (Tabla 3).

Tabla 3 Tareas disponibles en Rainview Analyzer

RADAR VOLUME CORRECTIONS
OCC – Occultation Correction
3DCDP – 3D Polar Clutter Map Processing
STANDARD METEOROLOGICAL PRODUCTS
PPI – Plan Position Indicator
HYDROLOGICAL PRODUCTS
RR – Rainfall Rate
PAC – Precipitation Accumulation
RSA – River Sub Catchment Accumulation
RGRT – Rain Gauge Radar Total
FORECASTING AND WARNING PRODUCTS
RTR – Rain Tracking
WRN – Feature Detection and Warning

2.5.3 RainScoutMet Este programa permite la visualización de los datos en bruto y los productos generados por RainView Analyzer. La visualización es implementada de manera online como offline. En la primera forma, se selecciona el tipo de producto a mostrar para ver los datos más recientes. En cambio, de manera offline, es necesario seleccionar los archivos que se desea leer.

El programa trabaja bajo una visualización por capas, de forma similar a cualquier SIG. Esto permite la adición de detalles como divisiones políticas, orografía, ríos y ciudades. De esa manera, la gráfica resultante proporciona mucha más información, y, sobre todo, información más sencilla de interpretar (Figura 15).

Además de las capas adicionales, otras herramientas están disponibles. Opciones de zoom y arrastre, típicas de cualquier visualizador, están disponibles. Por otra parte, es posible realizar animaciones a partir de la secuencia de datos. La opción de guardado está disponible, pero solo para formato .png como salida y constituye una captura de pantalla del programa.

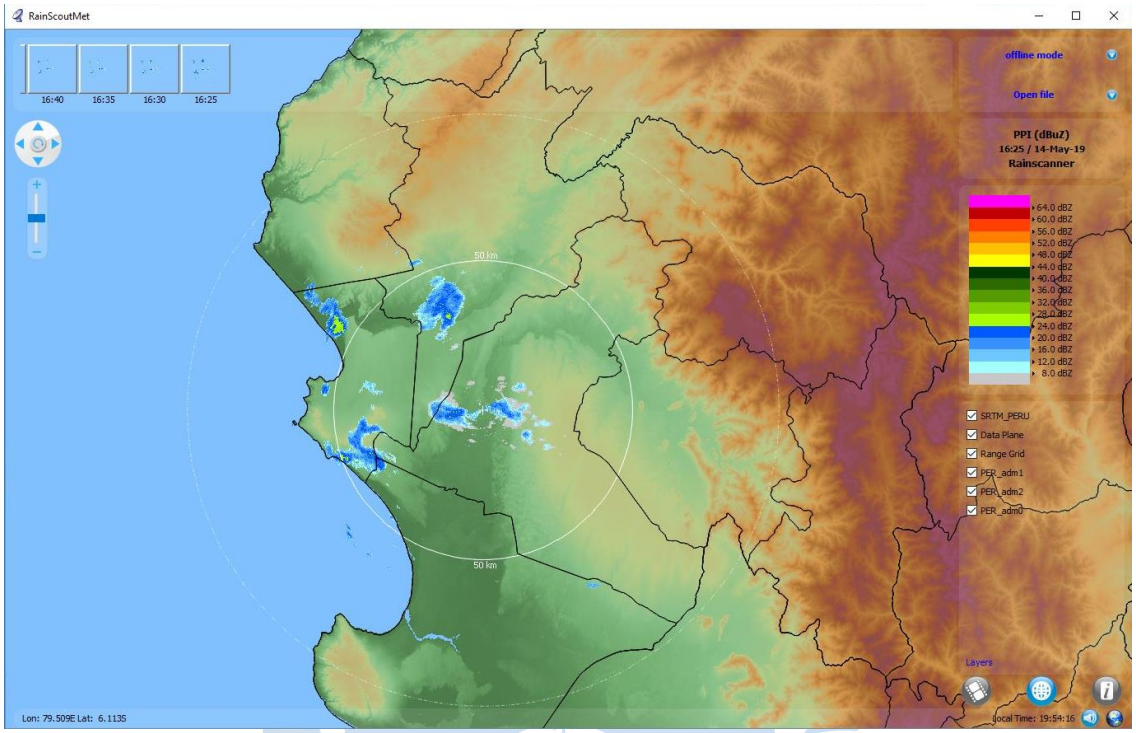


Fig. 15 Interfaz del software RainScoutMet.

Capítulo 3
Errores y correcciones

Las mediciones realizadas por el radar meteorológico presentan varias limitaciones y errores que se deben corregir. Sin un adecuado conocimiento de estos, los datos generados pueden conducir a información errónea o a valores de precipitación sub o sobre estimados. Por ello, este capítulo se centra en describir aquellas consideraciones importantes que se deben tener en cuenta al analizar los resultados del radar, los errores y sus soluciones.

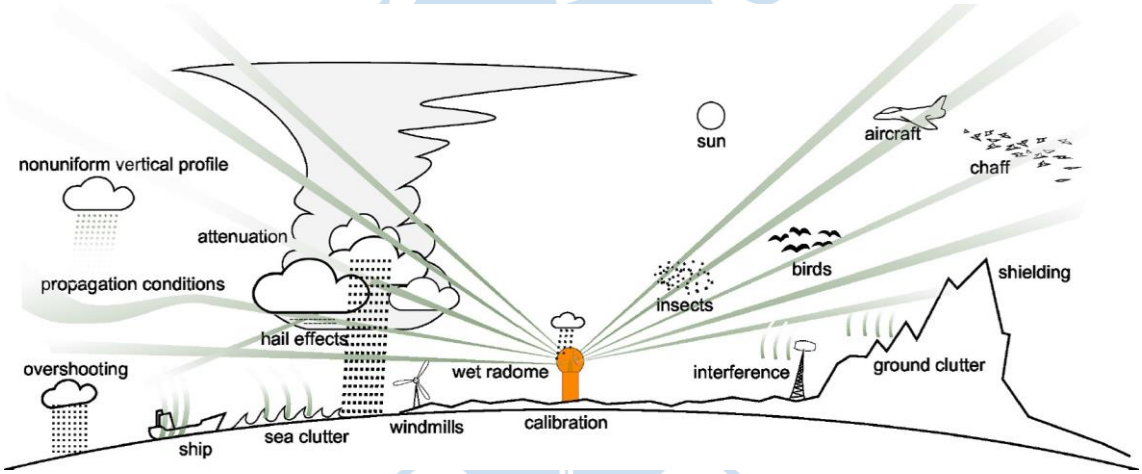


Fig. 16 Fenómenos que afectan la calidad de los datos del radar.
Fuente: Holleman, Michelson, Galli, & Germann (2006)

La Figura 16 ilustran las fuentes de los principales problemas que se pueden presentar en las detecciones con un radar meteorológico. Estos tienen tres grandes efectos sobre los datos. El primero de ellos, está relacionado con la resolución del radar y su estrategia de escaneo. Debido al sistema de coordenadas polares que utiliza, la resolución disminuye con la distancia al radar, mientras que la altura de cada compartimiento aumenta. La segunda consecuencia está relacionada con la aparición de falsas alarmas debido a los ecos no meteorológicos que se presentan como aves, aeronaves e insectos, así como el apantallamiento de ciertas zonas debido al relieve topográfico. Por último, el paso de los datos de reflectividad hacia la estimación de la precipitación tampoco está exenta de errores. La incertidumbre en la relación Z-R utilizada y los efectos de atenuación condicionan el uso de los datos para propósitos cuantitativos. Sin una adecuada calibración y comprensión de los

errores y limitaciones, la data producida por el radar puede ser malinterpretada o infravalorada.

Los errores y limitaciones no están sujetos a generar solo una de las consecuencias descritas, sino al contrario, estos errores suelen tener varias consecuencias y se combinan de tal forma que su efecto a menudo solo puede estimarse de manera conjunta.

La Figura 17 ilustra el impacto de los fenómenos sobre los datos del radar.

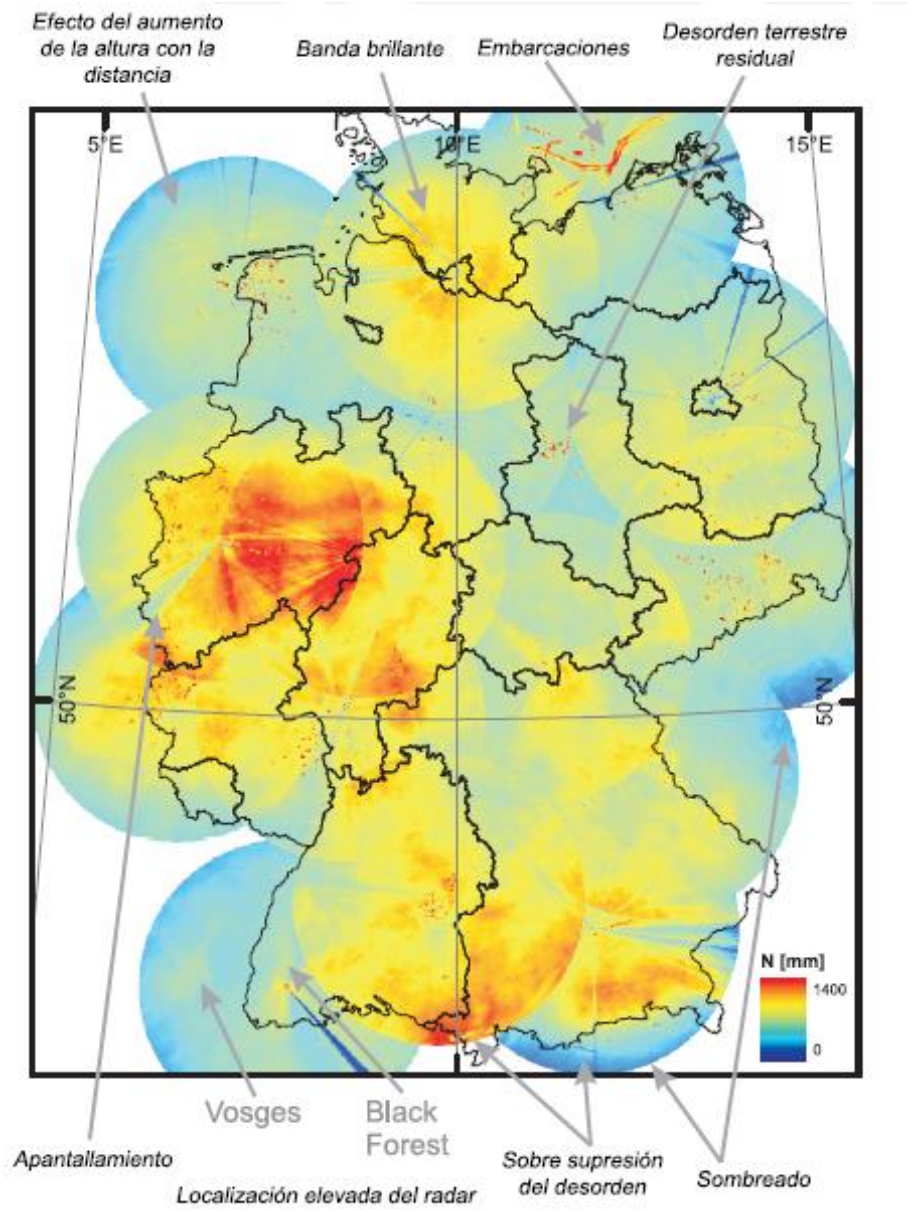


Fig. 17 Precipitación acumulada “corregida” a partir de los datos del radar (DWD RY-product, German composite) durante el 2009.

Fuente: Pfaff (2013)

3.1 Problemas vinculados a la resolución

Debido al giro de la antena y la propagación del rayo en la atmosfera, los datos del radar tienen naturaleza polar. Usualmente, la estrategia de escaneo es el "barrido volumétrico" que consta de barridos sucesivos a ángulos de elevación constantes. En el caso del radar PIUXX solo se puede realizar barridos a un solo ángulo de elevación. Los datos recolectados son de naturaleza polar y se encuentran definidos por la distancia desde el radar r y el ángulo azimut θ para determinado ángulo de elevación ϕ.

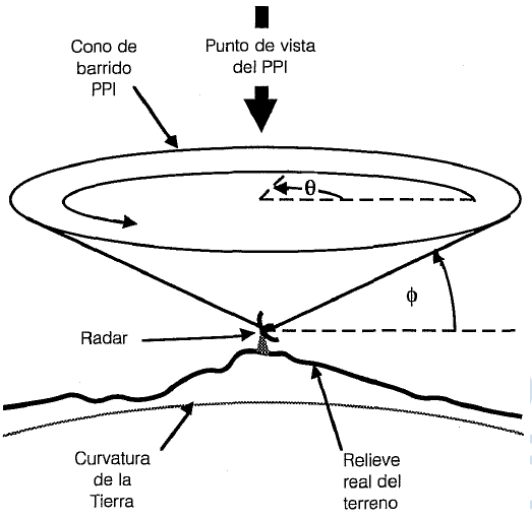


Fig. 18 Barrido para un ángulo de elevación constante.
Fuente: Rosengaus (1995)

La naturaleza del escaneo obliga a que los datos pierdan resolución conforme aumenta la distancia ® desde el blanco hasta el radar, pues se escanea un volumen más grande de la atmosfera. En base a la ecuación del volumen de muestreo V_m se observa que este aumenta en forma proporcional al cuadrado de la distancia. Situación similar ocurre con los datos de la potencia recibida.

A medida que el tamaño del volumen de muestreo V_m aumenta, resulta cada vez más difícil mantener la hipótesis de la distribución homogénea de las gotas en dicho volumen. Debido a ello, el radar puede sesgar los datos, especialmente en los límites de las celdas de precipitación (Rosengaus, 1999). De esta forma, una misma tormenta no será reportada de la misma forma si se ubicase a diferentes distancias respecto del radar (Figura 19).

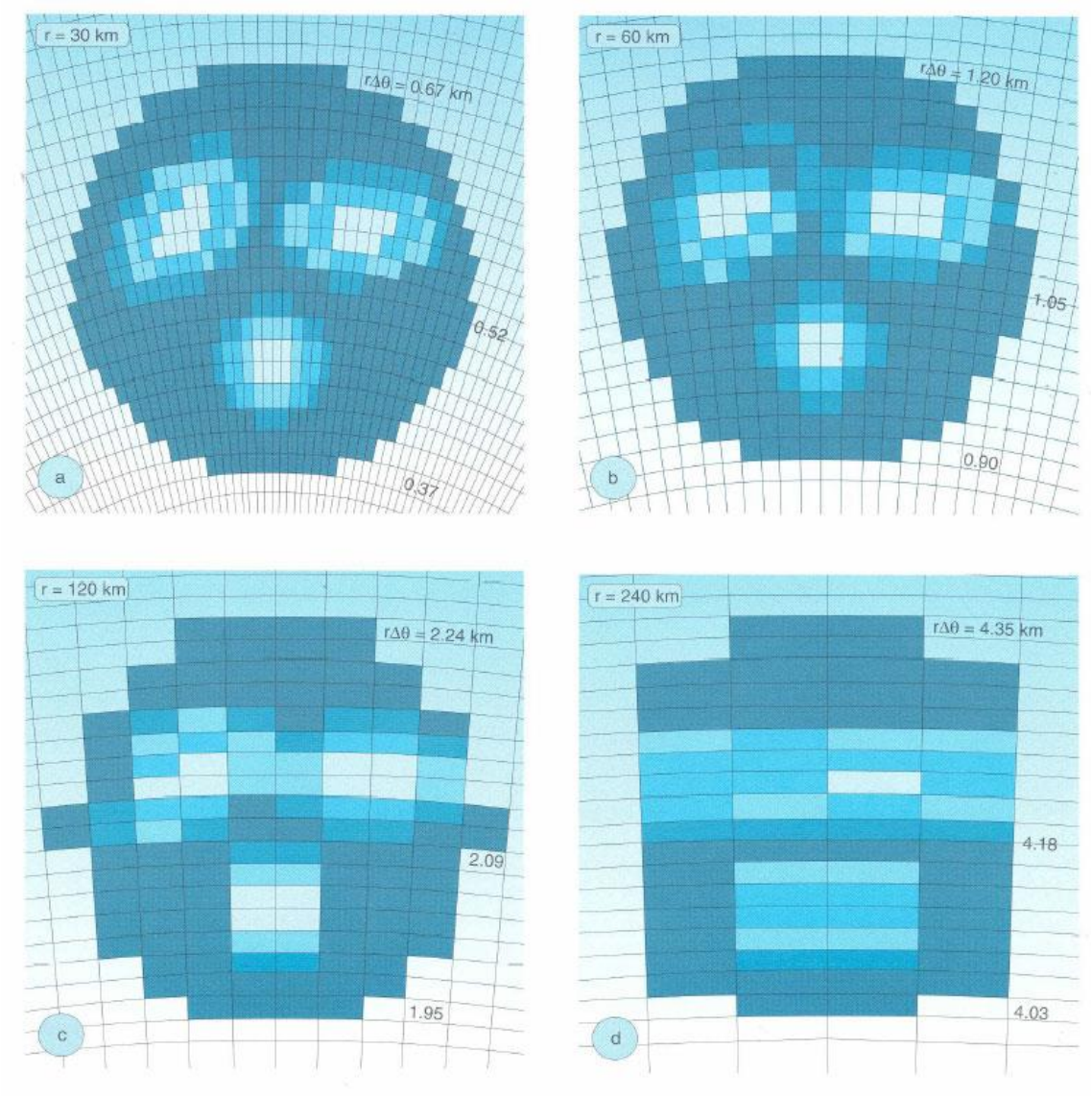


Fig. 19 Una misma tormenta percibida a diferentes distancias del radar
Fuente: Rosengaus (1999)

Por las mismas razones, tampoco es posible diferenciar dos objetos ubicados en el mismo volumen de muestreo. Así, en los datos del radar, se sobre estima el tamaño de objetos pequeños de alta reflectividad como aviones, bandadas de aves o enjambres de insectos que se pudiesen presentar.

La distancia radial entre dos datos sucesivos o intervalo de rango está limitada por la frecuencia de repetición del pulso (PRF). Para discretizar el muestreo espacial, se envía un tren de pulsos y se registra cada cierto intervalo de tiempo Δt, adecuado para medir volúmenes consecutivos, correspondiente a $\Delta t = t/2$ donde t representa la duración del pulso emitido. El factor de dos se corresponde a lo expuesto anteriormente, en el cual la señal tiene que viajar dos veces: desde el transmisor hasta el blanco y luego desde este hacia el receptor (Figura 20).

El intervalo de rango (range bin) es configurable dentro de cierto rango definido por el ancho de pulso y la longitud de repetición del pulso. Sin embargo, el valor se ubica siempre más cerca del límite superior.

$$\frac{c\tau}{2} < range\ bin < \frac{cPRT}{2} \qquad \textbf{Ec. 14}$$

Donde PRT representa la duración del pulso y es el inverso de la frecuencia de repetición del pulso (PRF).

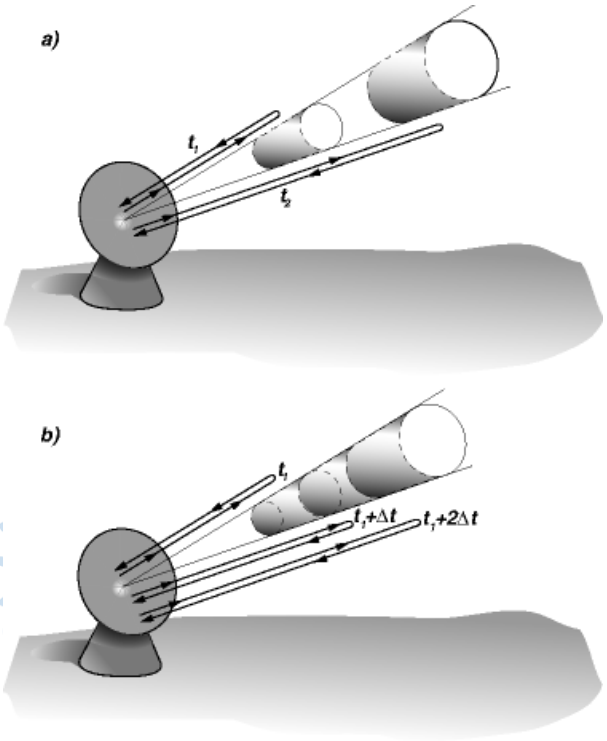


Fig. 20 (a) Potencia registrada en los instantes t_1 y t_2, (b) si ésta se registra en instantes de tiempo separados dt = t / 2, se corresponderá con volúmenes V_m consecutivos.

Fuente: Sánchez-Diezma & Corral (2000)

Esto define la resolución radial del radar, que define el valor mínimo con el cual se puede muestrear todo el alcance efectivo. A diferencia de la resolución ligada al volumen de los datos, está resolución es fija y se mantiene con la distancia.

La resolución azimutal está ligada al espectro de potencia de la señal generada. La potencia de la señal no se distribuye de manera uniforme, sino que se distribuye de manera lobular dentro de un volumen de tronco de cono (Figura 21). Usualmente se asume que sigue un patrón Gaussiano o mediante una serie de lóbulos secundarios. Se define como la mitad del ángulo para el cual existe un ancho de potencia en la antena de 3 dB. Está definida en las dos direcciones perpendiculares a la línea de propagación de la señal (planos E y H), y a veces

se les denomina horizontal y vertical. Esta resolución está ligada a parámetros constructivos de la antena y el transmisor más que al sistema polar de los datos.

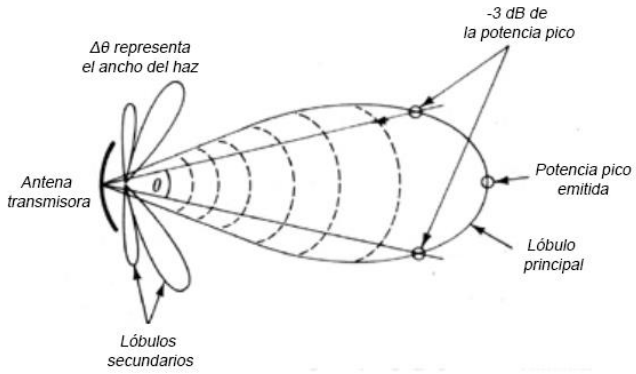


Fig. 21 Ancho de haz (beamwidth).

Fuente: *https://electronics.stackexchange.com/questions/349433/a-basic-question-about-the-meaning-of-db-of-an-antenna-radiation-pattern*

Los ángulos de elevación hacen que el volumen de muestreo se encuentre cada vez a mayor altura de la superficie terrestre. Debido al gran alcance de los radares, del orden de decenas de kilómetros, no es posible obviar la curvatura terrestre. Por eso ni siquiera para un ángulo de elevación $\phi = 0°$ el escaneo será horizontal. La altura se puede determinar a partir de las pautas de (Doviak & Zrnic, 1993) cómo:

$$H = \sqrt{r^2 + \left(k_e r_e\right)^2 + 2rk_e r_e \sin\left(\phi\right)} - k_e r_e + h_a \qquad \textbf{Ec. 15}$$

Donde r representa la distancia hasta el radar, k_e representa el factor de ajuste debido al gradiente de reflectividad que afecta la propagación del haz en la atmósfera. La constante r_e representa el radio terrestre, ϕ el ángulo de elevación mientras que h_a representa la altitud que corresponde a la ubicación del radar.

El aumento de la altura con la distancia tiene importantes consecuencias hidrológicas. No se está midiendo la precipitación efectiva sobre el suelo y ni siquiera a una altitud constante. Esto origina que la ubicación de la precipitación registrada por el radar difiera de la precipitación que alcanza la tierra y es registrada por los pluviómetros. Por tanto, se requieren correcciones que consideren estos efectos para aplicaciones hidrológicas.

La altura no constante puede ocasionar diversos tipos de errores respecto a la distribución de la precipitación y su intensidad (Figura 22). Lluvias ligeras a niveles superiores pueden evaporarse durante su caída a la tierra, ocasionando que el radar mida precipitación sin que está ocurra. También es posible que suceda el proceso inverso, y las pequeñas gotas de agua durante su caída se fusionen a través de un fenómeno denominado coalescencia. Esto hace que la distribución espacial real sea menor mientras que su intensidad sea mayor que la

registrada por el radar. El llenado parcial del volumen de muestreo también es posible, sin embargo, su efecto ya ha sido mencionado. Por último, es posible que la precipitación suceda a ángulos de elevación menores que el de escaneo, por lo que la precipitación podría no ser registrada.

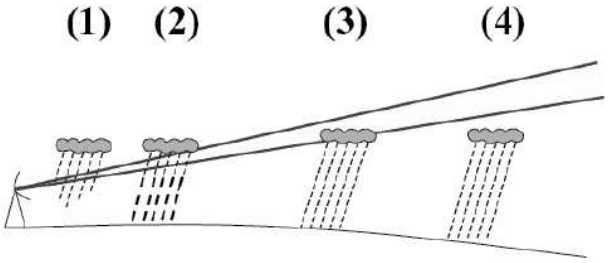


Fig. 22 Posibles errores causados por el aumento de altura: (1) evaporación (2) condensación o coalescencia (3) llenado parcial (4) muestreo sobre la precipitación (overshooting).
Fuente: Delobbe (2006)

La mayoría de estos problemas son abordados mediante el barrido volumétrico. Incluyendo la información de diferentes ángulos de elevación se obtiene la distribución tridimensional de la precipitación. Lamentablemente ese no es el caso del radar PIUXX. Dado que no es posible mitigar estos problemas a través de dicho método, se deberá estar alerta ante posibles discrepancias entre los datos y la realidad.

Los radares meteorológicos actuales son mono estáticos. Es decir, la antena transmisora y receptora son la misma. Durante el tiempo en que el radar se encuentra en transmisión no es posible recibir señales, lo que se logra apagando el receptor mediante un duplexor. Por ello, el radar meteorológico no puede detectar o medir la reflectividad en la porción de la atmosfera más cercana a este.

Si se define t_{sw} como el tiempo de intercambio entre la función de transmisión y recepción es posible determinar el rango de medición mínimo como:

$$R_{\min} = \frac{c\left(\tau + t_{sw}\right)}{2} \qquad \textbf{Ec. 16}$$

Dado que el ancho de pulso es del orden de 0.5 -1.2 us, el rango de medición mínimo será a partir de 75 a 180 metros. El valor real, sin embargo, es frecuentemente mayor. Esto se debe a que se puede usar pulsos más largos o al aumento de tiempo por el intercambio mecánico entre la recepción y transmisión. Aun así, el rango mínimo a menudo se ve aún más disminuido debido al efecto de los lóbulos secundarios y el clutter. Sin embargo, a menudo este efecto es olvidado, debido a que la mayoría de los procesadores de señal completan dichos datos.

Para mostrar este efecto en el radar PIUXX, se realizó una comparación entre los datos con las correcciones más básicas, dentro de los cuales se incluye la distancia mínima. Se recuperan las zonas de datos vacíos en la cercanía del radar. A partir de estos, parece que el rango mínimo está cerca de los 900 metros a la redonda. Este resultado empírico es una mejor estimación que la realizada a partir de la ecuación anterior. Es importante recalcar que el resultado es consistente con lo expuesto por (Rosengaus, 1995). Debido a la presencia de clutter y otros métodos de corrección, los resultados deben tomarse como una estimación y solo para poner en evidencia la influencia de este fenómeno en los datos. Cualquier otra interpretación va más allá del propósito original de esta explicación.

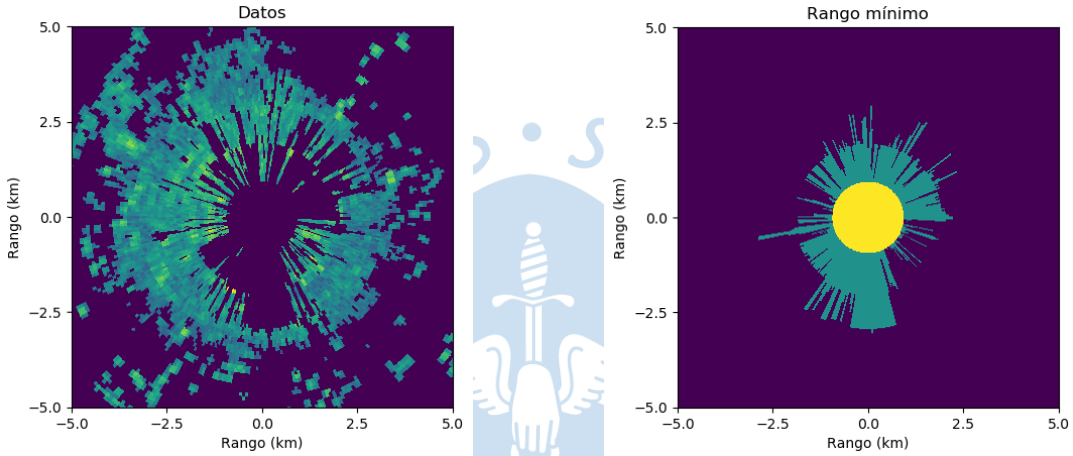


Fig. 23 Rango mínimo (a) PPI típico de datos corregidos en condiciones de no precipitación (b) Estimación del rango mínimo (amarillo: rango mínimo, verde: correcciones adicionales usuales por clutter)

Por último, cada escaneo para un solo ángulo de elevación se completa durante un giro de la antena. La velocidad del motor es de 12 RPM, por lo que técnicamente es posible registrar un dato cada 5 segundos. Sin embargo, la adquisición de los datos en tiempos cortos propicia la aparición de errores aleatorios que se deben principalmente a defectos en los instrumentos electrónicos y otros. Por otra parte, si se utilizan tiempos de adquisición de datos demasiado grandes, los eventos meteorológicos de dinámicas rápidas pueden no ser captados apropiadamente. Por ello, RainView Analyzer limita la generación de la data cruda a intervalos de 30 segundos como mínimo. Actualmente el radar PIUXX trabaja con un tiempo de generación de 5 minutos.

La configuración de este tiempo abre mayores posibilidades al uso de los datos. Aviones y objetos similares, que aparecen durante la visualización de los datos a tiempo real, no aparecen en los datos crudos generados en intervalos de 5 minutos. Sin embargo, es posible que aparezcan en los datos si el intervalo es menor. Esto requiere de mayores etapas de filtrado para el uso de los datos, pero al mismo tiempo permite su uso para el rastreo de aviones cercanos y alarmas relacionadas con estos para proporcionar información al aeropuerto.

Hasta ahora se han tratado aquellos problemas y limitaciones que afectan a la resolución del radar. La siguiente sección se ocupa del segundo problema mencionado en la introducción de este capítulo: la aparición y/o desaparición de los ecos no meteorológicos.

3.2 Problemas vinculados a ecos no meteorológicos

Lugares que presentan un fuerte relieve orográfico son frecuentemente problemáticos para el radar. La intersección del haz con montañas y similares puede originar fuertes retornos que bloquean la propagación del haz más allá de estas (Figura 24). Esto disminuye el alcance efectivo del radar para determinadas direcciones azimutales a la vez que origina discontinuidades en los datos.

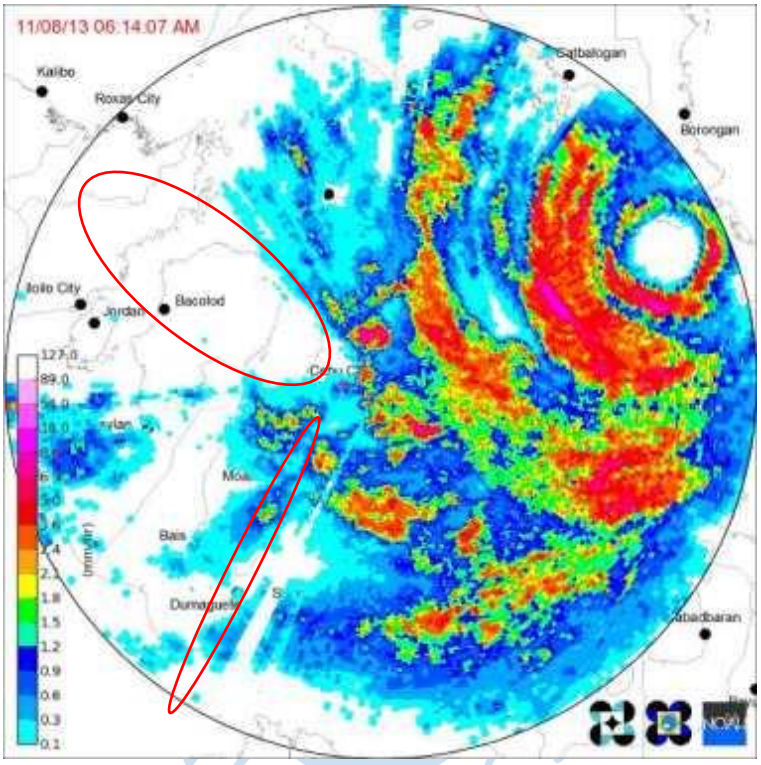


Fig. 24 Tifón Haiyan(Yolanda) capturado por el radar Cebu. Las elipses en rojo corresponden al efecto de apantallamiento.
Fuente: Crisologo (2016)

Este problema denominado apantallamiento (beam blockage) ha sido cuantificado a través de simulaciones basadas en modelos de elevación digital (DEM) para encontrar las zonas donde el efecto es considerable (Bech, Codina, Lorente, & Bebbington, 2003; Bech, Gjertsen, & Haase, 2007).

La mala comprensión de este efecto ha llevado a malas interpretaciones a usuarios inexpertos o a personas ajenas a la meteorología. Notable es el caso presentado en la Figura 25, en el cual algunas páginas web locales argumentaban la existencia de un escudo protector de oración en las zonas sin precipitación aparente durante un tifón en Filipinas (Crisologo, 2016).

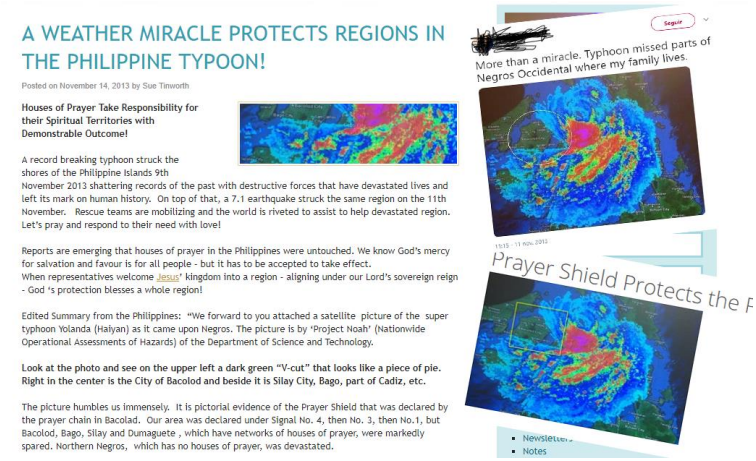


Fig. 25 Recortes del "escudo protector" durante el paso del tifón Yolanda.

Hechos como estos refuerzan la importancia de esta sección para la correcta interpretación de los resultados.

Una vez se ha determinado las zonas afectadas por el apantallamiento, es posible solucionar el problema de tres formas principales: la primera de ellas involucra colocar esas secciones como “No data”. Este procedimiento limita la aplicación de los datos solo a las regiones libres del apantallamiento. La segunda opción consiste en reemplazar los valores por interpolación de los valores cercanos que sufran el efecto de apantallamiento. Esto es solo posible si las regiones apantalladas no son muy grandes, y aun así, los valores no tienen la misma calidad que el resto de los datos. La tercera opción consiste en extrapolar los datos de la misma ubicación, a partir de un ángulo de elevación superior. Esto se realiza usando el Perfil Vertical de Reflectividad (VPR). Por ello, solo es posible en radares cuya estrategia de escaneo sea el barrido volumétrico. De igual forma, es posible utilizar procedimientos mixtos.

Otro de los problemas presentes en los datos del radar son las motas que consisten en pequeños ruidos en los datos del radar (Figura 26). También se presentan motas inversas, que corresponden a celdas de reflectividad mínima rodeadas por celdas donde se registra alta reflectividad producto de precipitación (Osrodka, Szturc, & Jurczyk, 2014). Estos son ruidos similares al clásico “ruido sal y pimienta” solo que más espaciados. Sin embargo, la aplicación de filtros de mediana, usual ante este tipo de ruidos, distorsiona los detalles. (Peura, 2002)

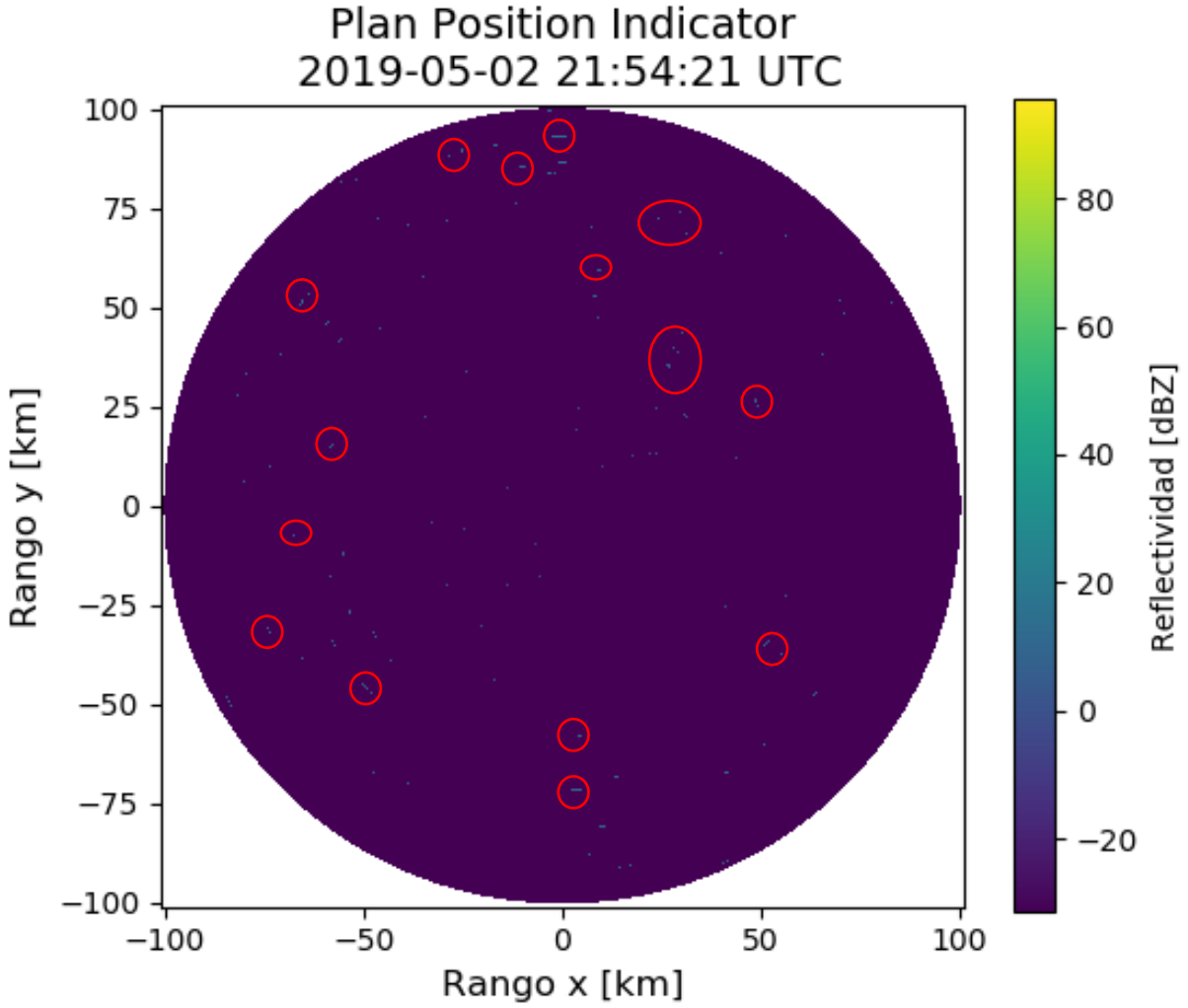


Fig. 26 Presencia de motas (rojo) en los datos crudos al iniciar el radar.

Estos a menudo son eliminados por la unidad Signal Processor durante la generación y adquisición de los datos. Sin embargo, algunos de estos pueden sobrevivir al proceso y aparecer en los datos. Para correcciones posteriores se usa su pequeño tamaño como condición de filtrado mediante algoritmos basados en Visión Artificial o Procesamiento de Imágenes.

Usando la información de su pequeño tamaño, las motas pueden ser eliminadas a través de un proceso de tres pasos. Primero se realiza una binarización que distingue los datos de los valores de reflectividad mínima. Luego, se aplica una ventana cuadrada de tamaño impar (generalmente 5x5 o 3x3) donde se realiza una suma de los valores de la ventana. Si estos superan cierto umbral, son considerados ecos meteorológicos, de lo contrario son eliminados de los datos originales. Una descripción de este procedimiento desde un punto de vista más matemático puede encontrarse en (Osrodka et al., 2014). Las motas inversas son eliminadas por un proceso similar.

El filtrado de las motas también puede llevarse a cabo a través de un proceso de segmentación y etiquetado. A través del algoritmo de "Segmentación de cuenca" (Watershed segmentation) o a través del etiquetado por elementos conectados, se generan etiquetas para todos los objetos detectados. Luego, mediante un umbral de número de pixeles se filtran las motas. Este procedimiento recibe el nombre de SPECK y es usado por el Finnish Meteorological Institute (Peura, 2002).

El alcance del radar está determinado por la atenuación de la señal emitida. No obstante, las señales son ondas electromagnéticas y se propagan mucho más allá. Algunos ecos meteorológicos más allá del alcance seleccionado pueden ser lo suficientemente fuertes como para ser capturados. Estos tienen una duración mayor, y por tanto no serán registrados sino hasta la siguiente recepción. El radar supone que todos los ecos recibidos se encuentran dentro del alcance dado y no podrá diferenciar que este eco no pertenece a este rango. Estos ecos son denominados ecos de retorno o viaje múltiple y aparecen en los datos de manera deformada (Figura 27). Esto se debe a que la distancia a la cual aparecen en los datos no se corresponde con su auténtica posición respecto del radar. Dado que la señal aumenta con la distancia, su tamaño y valor de reflectividad son subestimados.

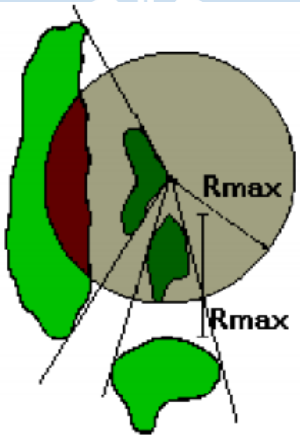


Fig. 27 Ecos de viaje múltiple.
Fuente: Büyükbas et al. (2006)

Los ecos de retorno múltiple son un problema a considerar principalmente en los radares Doppler. Estos presentan dos modos de escaneo de la atmósfera: modo "reflectividad" donde miden la reflectividad y "modo Doppler" donde miden también la velocidad radial. La gran diferencia entre estos modos es su alcance, el radar en "modo Doppler" tiene un alcance cercano a la mitad del alcance en modo "reflectividad". Debido a ello, son especialmente propensos a este problema, y en algunos casos pueden aparecer hasta ecos de tercer o cuarto viaje.

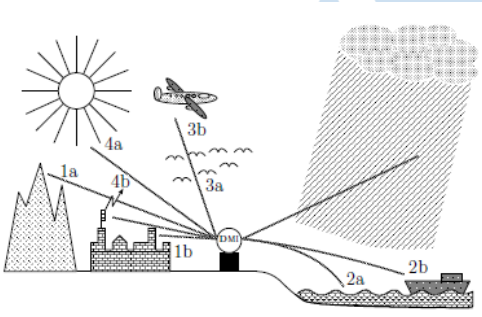
Los ecos de retorno múltiple pueden ser identificados a través de comparación con otras fuentes de información como satélites o a través de la inspección visual a la atmósfera. Otra manera consiste en aprovechar sus características para la identificación. Debido al

retorno múltiple, los ecos de este tipo presentan un perfil diferente al de una tormenta. Usualmente son más alargados, con valores de reflectividad pequeños y de baja altitud. Sin embargo, pueden ser confundidos con pequeños núcleos de tormentas convectivas (Büyükbas et al., 2006).

La manera más eficaz de identificar y eliminar estos ecos es aprovechando la dependencia del rango con la frecuencia de repetición de pulso (PRF). De esta forma, si se cambia la frecuencia y los ecos no han cambiado su ubicación, estos corresponden a ecos adquiridos sin ambigüedad. Caso contrario, se deben a ecos de retorno múltiple. Los radares modernos usan la información de dos PRF para eliminar los ecos de retorno múltiple o incluso técnicas más avanzadas basadas en el mismo principio (Cao, Zhang, Palmer, & Lei, 2012; Pirttilä, Lehtinen, Huuskonen, & Markkanen, 2005).

A menudo, se agrupan la mayoría de los errores relacionados a la aparición de ecos no deseados bajo la categoría de clutter. Es importante mencionar que clutter designa genéricamente cualquier clase de eco no buscado. Por ejemplo, en un radar de uso militar, la precipitación será un eco no deseado; mientras que en el radar meteorológico representa el objeto de estudio. En ese sentido, se utilizará la palabra clutter para designar a cualquier eco no meteorológico.

La clasificación del clutter es diversa. La mayoría de los autores lo clasifican en función de su fuente. Otros en cambio, debido al gran uso de la velocidad Doppler para identificarlo, lo clasifican como estacionario y no estacionario. Dado que el radar PIUXX no presenta características Doppler, se prefiere la primera clasificación. Dentro de esta existen cuatro categorías: clutter terrestre, clutter marino, clutter aéreo e interferencias (Figura 28).



1) Clutter terrestre causado por (a) montañas, (b) edificios/turbinas eólicas.
2) Clutter marino causado por (a) superficie del mar, (b) embarcaciones.
3) Clutter aéreo causado por (a) aves/insectos, (b) aeronaves.
4) Interferencia causada por (a) el sol, (b) antenas transmisoras.

Fig. 28 Tipos de clutter.
Fuente: Bøvith (2008)

El clutter terrestre es causado por la intersección de los lóbulos secundarios del radar; y en algunos casos del lóbulo principal, con elementos ubicados sobre la superficie terrestre. Debido al volumen de escaneo y al aumento de la altura ya explicados, estos se encuentran con frecuencia en las cercanías del radar y solo para pequeños ángulos de elevación. Suelen ser causados por objetos artificiales de gran reflectividad como estructuras metálicas.

Otros ecos que causan clutter terrestre son las montañas y el relieve topográfico. Estos, sin embargo, ya han sido abordados antes mediante el efecto de apantallamiento. Debido a la poca influencia de este efecto para el actual ángulo de elevación, a menudo al mencionar clutter terrestre solo haremos referencia a los edificios y similares.

De especial interés resulta el clutter terrestre producido por las turbinas eólicas. A diferencia del resto de clutter, estas presentan velocidad Doppler. Con el aumento de la energía renovable, las turbinas eólicas se han convertido en un problemática actual en el radar meteorológico.

El clutter marino se produce cuando el escaneo intercepta la superficie del mar. Para que esto suceda son necesarias dos condiciones. La primera de ellas es que el radar este ubicado cerca al mar. La Figura 28 muestra de manera clara la otra condición: el radar deberá ubicarse en un lugar lo suficientemente elevado como para realizar escaneos con ángulos de elevación negativos ($\phi < 0°$). Además, el radar podría reconocer las embarcaciones sobre la superficie del mar dificultando aún más su eliminación.

El clutter marino es especialmente problemático. A diferencia del clutter terrestre de naturaleza fija, el clutter marino exhibe patrones de velocidad debido a las mareas y a los barcos. Dado que la mayoría de los métodos de eliminación del clutter utilizan la velocidad Doppler, estos métodos proporcionan pobres resultados en este tipo de clutter. Otros procedimientos utilizan parámetros polarimétricos o lógica difusa para proporcionar mejores resultados.

Afortunadamente, debido al ángulo de elevación usado ($\phi = 2°$), el radar PIUXX no tendría problemas de clutter marino. Además, el radar PIUXX no podría ser usado con ángulos de elevación negativos por el problema de apantallamiento que se generaría.

El denominado clutter aéreo es de naturaleza no estacionaria. A diferencia de los tipos anteriores, puede encontrarse a ángulos de elevación mayores. Sin embargo, suele solo aparecer en las cercanías del radar debido al aumento del volumen de muestreo. Dado que es en su mayoría causado por objetos de pequeño tamaño, puede ser fácilmente eliminado basado en esta característica. De ese modo, su eliminación es parecida a la de las motas.

Este tipo de clutter no representa un problema mayor para el radar. En el caso del radar PIUXX, debido a la aplicación de un speckle filter en la unidad Signal Processor la mayoría de este clutter es eliminado.

La interferencia es causada por la emisión de señales de frecuencias similares a la emitida por el radar. El sol, es una fuente de radiación electromagnética que abarca la frecuencia de los radares. De esta forma, si se apunta hacia él, el radar puede percibir ecos. Sin embargo, esto solo sucede para los ángulos de elevación más grandes.

La interferencia del sol no necesariamente debe verse como algo completamente negativo. El sol puede ser usado para una calibración de reflectividad mediante medidores de

la radiación del flujo solar o para monitorear el funcionamiento operacional del radar (Andersson, 2000; Iwan Holleman, Huuskonen, Kurri, & Beekhuis, 2010).

La interferencia debido a otras fuentes como las antenas receptoras está en función del tipo de radar usado. La interferencia no será la misma para radares de distintas bandas. Sin embargo, el aumento del número de equipos inalámbricos está ocasionando un aumento de la interferencia, principalmente en la banda C y algunos casos en la banda S. De no modificarse las regulaciones, la aplicación operativa de estos radares podría estar en peligro (Saltikoff et al., 2016).

La eliminación del clutter, es tal vez, el problema más estudiado en el ámbito del radar. El problema es tan importante que uno de los criterios de selección del radar y su ubicación involucra el efecto esperado sobre los datos del radar. Este problema puede ser disminuido posicionando el radar en lugares elevados, de tal forma que la mayoría de clutter no aparezca en los escaneos más bajos. Sin embargo, este es solo uno de los factores a considerar. Otros aspectos, como el área de interés y el tipo de precipitación pueden definir las características del radar. Usualmente, la solución comprende un balance entre los factores. Debido a ello, es usual que parte de la eliminación del clutter tenga que realizarse durante el procesamiento de los datos.

La eliminación del clutter terrestre durante el procesamiento se realiza de manera diferente según el tipo de radar. Los radares más básicos, que no poseen más que un ángulo de elevación y carecen de cualquier otra tecnología usan métodos de detección basados en mapas de clutter registrados durante condiciones de no precipitación. Los radares con escaneo volumétrico aprovechan que el clutter se presenta solo en los ángulos de elevación más bajos. Por ello, los valores de reflectividad del clutter pierden rápidamente la correlación entre ángulos de escaneo sucesivos. A través de este patrón, se pueden incluir algoritmos sencillos para su eliminación. Los radares más avanzados utilizan otros parámetros a parte de la reflectividad para eliminar el clutter.

Con la tecnología Doppler, los radares son capaces de estimar la velocidad radial de los objetos. El clutter terrestre, de naturaleza fija, puede ser eliminado de manera sencilla en estos radares. La eliminación se realiza a través de un filtro que elimina aquellos datos con velocidades nulas o cercanas a cero. Sin embargo, algunos ecos meteorológicos pueden ser filtrados también. Esto se debe a que el radar no mide la velocidad sino solo la componente radial. Si la precipitación se mueve de manera tangencial o en una dirección cercana a esta, el filtro basado en la velocidad podría eliminarla. Además, no todos los tipos de clutter son de naturaleza fija, por lo que algunos no serían filtrados de esta forma.

Los procesos descritos se llevan a cabo durante el procesamiento de la señal. Sin embargo, los datos son usados posteriormente, por lo que a menudo se aplican capas de procesamiento adicionales dependiendo de los objetivos y aplicación de los datos. La mayoría de los algoritmos que proporciona la literatura pertenecen a esta categoría.

Pese a la gran cantidad de algoritmos disponibles, los más recientes están abocados a la investigación de las características de doble polarización. Además, casi todos involucran la velocidad Doppler en ellos. Aquellos que involucran radares sin estas dos tecnologías, usualmente utilizan las características del clutter presentes en los escaneos volumétricos. Estos involucran el perfil vertical de reflectividad en los ecos, su altura máxima o la correlación entre ángulos de elevación sucesivos. Lamentablemente, el radar PIUXX carece de estas tecnologías. Por ello, los algoritmos realmente aplicables son más bien pocos (Tabla 4).

Los algoritmos restantes pueden ser clasificados en tres categorías: aquellos que usan el análisis de textura, análisis estadístico o fuentes alternativas. El primer grupo aprovecha que el clutter presenta una textura heterogénea, es decir que los valores de reflectividad sucesivos cambian drásticamente. El segundo aprovecha la persistencia del clutter, y a través de los datos acumulados durante un periodo de tiempo relativamente largo, se determina la ubicación del clutter. El último grupo contrasta los datos con fuentes de información alternativas como pueden ser imágenes satelitales, disdrómetros u otros radares.

Los algoritmos propiamente no realizan la eliminación del clutter, sino que en su lugar identifican la presencia del clutter. Por ello, la salida de los algoritmos no corresponde a los datos de reflectividad corregidos, sino a una máscara de los datos. Esta máscara, de naturaleza binaria, identifica con 0 y 1 la ausencia/presencia de clutter. El proceso de eliminación a partir de los datos identificados como clutter queda a elección del usuario.

Tabla 4 Métodos aplicables para la detección de clutter

Autor	Tipo de algoritmo	Datos de entrada	Disponible
(Haddad, Adane, Sauvageot, Saudoki, & Nailli, 2004)	Análisis textural	Reflectividad 2D	No
(Saudoki & Haddad, 2013)	Lógica difusa Análisis textural	Reflectividad 2D	No
(Lakshmanan, Zhang, Hondl, & Langston, 2012)	Análisis estadístico	Reflectividad 2D Set de datos previos para calibración	Si
Gabella filter (Gabella & Notarpietro, 2002)	Análisis textural Discriminación por tamaño	Reflectividad 2D	Si
Filter_cloudtype from Wradlib	Comparación con fuentes alternativas	Reflectividad 2D Datos satelitales	Si

(Heistermann, Jacobi, & Pfaff, 2013)			
Histocut from Wradlib (Heistermann et al., 2013)	Análisis estadístico	Reflectividad 2D Set de datos previos	Si
Cluttermap (SELEX, 2017b)	Análisis estadístico	Reflectividad 2D Set de datos previos	Si

Por último, existe un tipo especial de clutter, denominado Propagación Anómala (AP). Este se produce cuando las condiciones atmosféricas son tales que el haz del radar no se propaga de manera usual. Fenómenos como la super refracción o la sub-refracción del haz son posibles. Estos suelen producirse luego de la presencia de tormentas convectivas en las cuales el frente frío origina condiciones favorables para la aparición de AP (Hubbert, Dixon, & Ellis, 2009). Como consecuencia, se detecta clutter en ubicaciones diferentes a las usuales.

La propagación anómala es especialmente difícil de eliminar de manera operacional. Dado que no corresponde con precipitación, la manera más sencilla de eliminarla es a través de la confrontación con otras fuentes. Para ello, la mayoría de las agencias meteorológicas dependía de personal entrenado para su reconocimiento y eliminación.

La inspección visual de los datos de reflectividad de escaneo del volumen del radar revela que los ecos resultantes de la propagación de señales anómalas tienen poca extensión vertical. Además, se presentan solo en los barridos de elevación más bajos (generalmente $\phi \leq 2°$), dependiendo del gradiente vertical del índice de refracción. Los ecos propagados de manera anómala, similares al clutter terrestre cerca del radar, pierden relación entre sí rápidamente en el espacio y son espacialmente heterogéneos en su mayor grado. Por lo tanto, los ecos AP pueden reconocerse en los datos de reflectividad por su mayor variabilidad espacial que los ecos de precipitación. En contraste con el clutter terrestre regular (estacionario), que exhibe una correlación de tiempo más larga que los ecos meteorológicos, los ecos AP pueden parecerse mucho a la precipitación, exhibiendo crecimiento, decadencia y movimiento similar. a la de las tormentas (Steiner & Smith, 2002).

Los algoritmos para la detección de la propagación anómala, al igual que los de clutter, utilizan más que el simple campo de reflectividad 2D. Por ello, tampoco hay muchos aplicables. Sin embargo, este problema debe ser detectado en los datos. Sin indicios, no hay razón para buscar algoritmos e implementarlos. Por desgracia, se tienen pocos datos como para que el conjunto sea lo suficientemente representativo para obtener conclusiones a partir de ellos.

3.3 Problemas vinculados a la relación Z-R

La última gran clasificación de los datos no involucra errores propiamente en los datos, sino que limita su aplicabilidad. Los datos en bruto que se obtienen del radar se encuentran en dBZ, que representa una unidad logarítmica de la reflectividad.

$$dBZ = 10\log\left(\frac{Z}{1\,mm^6/m^3}\right) \qquad \textbf{Ec. 17}$$

Donde Z representa la reflectividad mientras que el factor de 1 mm^6/m^3 es escogido para hacer que la reflectividad en Dbz sea adimensional.

Además del uso del radar como herramienta de monitoreo de las precipitaciones; este resulta prometedor para áreas como la hidrología. Conocer la distribución espacial de las tormentas a tiempo real, representa un gran avance respecto a las redes pluviométricas tradicionales. Sin embargo, los datos son inútiles sino se puede relacionar la reflectividad con la tasa de precipitación.

A partir de lo expuesto en el capítulo pasado, la reflectividad puede calcularse como:

$$Z = \int N(D) \times D^6 dD \approx \sum D^6 \qquad \textbf{Ec. 18}$$

Donde $N(D)$ representa la distribución de las gotas y D el diámetro.

La segunda parte de la integral refleja el tratamiento en el volumen de muestreo. La integral puede aproximarse a la suma de la sexta potencia de los diámetros de todas las gotas en el volumen de muestreo.

La tasa de precipitación, calculada como el flujo sobre una superficie puede ser determinada como:

$$R = \int N(D)\left(\frac{\pi D^3}{6}\right) v(D)\, dD \qquad \textbf{Ec. 19}$$

Donde $v(D)$ representa la velocidad terminal de caída de una gota de diámetro D.

Como se deduce de las ecuaciones anteriores, no es posible establecer una relación lineal entre la tasa de precipitación y la reflectividad. Adicionalmente, la distribución de las gotas es característica del tipo de precipitación por lo que las relaciones exactas no son aplicables sin más información. Téngase en cuenta que la distribución de las gotas cambia no solo de tormenta en tormenta, sino que puede cambiar incluso dentro de la misma tormenta.

Diversos autores trabajaron para resolver la relación entre reflectividad Z y tasa de precipitación R. Se implementaron ajustes exponenciales, funciones Gamma, ajustes Log

normal y distribuciones Weibull. Posteriormente, se descubrió que la relación $Z-R$ podía expresarse bajo una expresión general de carácter exponencial (Guijarro, 2001).

$$Z = aR^b \qquad \textbf{Ec. 20}$$

A partir de la relación anterior, se desarrollaron marcos de estimación empíricos para determinar los coeficientes a y b. Estos comparaban la distribución de tasas de precipitación con los valores de reflectividad para realizar un ajuste. En consecuencia, muchas relaciones empíricas nacieron (Figura 29). La más conocida entre ellas es la Ecuación de Marshall – Palmer ($Z = 200\ R^{1.6}$), pero otras como la Ecuación utilizada por la red NEXRAD también se encuentran disponibles.

Sin embargo, dada la distribución de radares en el mundo (Figura 3), no hay muchas relaciones calculadas para en latitudes ecuatoriales. Por ello, deben usarse las relaciones Z-R con precaución. Una vez se hallan recolectado suficientes datos, es necesario validar la relación escogida o calibrar una versión propia sobre los datos.

RELACIONES Z-R (Marshal - Palmer) $Z = aR^b$				
a	**b**	**Aplicación**	**Autor**	**Fuente**
200.0	1.60	Lluvias generales estratiformes	Mashall – Palmer	Marshall and Palme, 1948
300.0	1.35	Lluvias convectivas	Sekhon – Srivastava	Sekhon y Srivastava, 1971
176.5	1.29	General-Trópico-Maceió/(Brasil)	Molion y Moraes	Da Silva, 2003 y De Assis, 2004
215.9	1.35	Convectivas-trópico-Maceió/ (Brasil)	Marcia Da Silva	Da Silva, 2003
171.7	1.19	Estratiformes-Trópico-Maceió/ (Brasil)	Marcia Da Silva	Da Silva, 2004
172.8	1.33	General-Trópico-Maceió/(Brasil)	Marcia Da Silva	Da Silva, 2005
371.0	1.24	Conectiva-Trópico-oeste de África		*Nzeukou et aL, 2002*
162.0	1.48	Estratiforme-Trópico oeste de África		*Nzeukou et aL, 2003*
167.8	1.26	Estratiformes-Trópico-(Brasil)	Univ.Federal de Alagoas	*Da Silva, 2003*
65.5	1.69	Conectivas-Trópico-(Brasil)	Univ.Federal de Alagoas	*Da Silva, 2003*
300.0	1.40	Conectiva-Miami (USA)-Cuba	WSR-88D-Woodley	*Woodley, 1975-Linsley, 1998 – Rodríguez,2000*
2000.0	2.00	Precipitación sólida	Carlson-Marshall	*Carlson-Marshall,1972*
21.0	1.71	Lluvia orográfica	Battan	*Battna, 1973/Crozier, 1975*
486.0	1.37	Tormentas Eléctricas	Battan	*Battna, 1973/Crozier, 1976*
178.0	2.21	Nieve (UK)	Gunn and Marshall	*Harrold, 1965/Crozier, 1977*
300.0	1.50	General	McCromick	*McCormick, 1970/Rinehart, 2007*
250.0	1.20	Sistemas conectivos tropicales	Resenfeld Tropical	*Zeitler,2006 NOAA*
130.0	2.00	Lluvias estratiformes de invierno	EastCool Stratiform	*Zeitler,2006 NOAA*
75.0	2.00	Lluvias estratiformes de invierno	WestCool Stratiform	*Zeitler,2006 NOAA*
140.0	1.50	Lloviznas		*Insmet Cuba – Doppler meteorological Obs Part B*
250.0	1.20	Huracán Tormenta tropical		*Insmet Cuba – Doppler meteorological Obs Part B*
436.0	1.43	Trailing portion		*Insmet Cuba – Doppler meteorological Obs Part B*
124.0	1.64	Central core		*Insmet Cuba – Doppler meteorological Obs Part B*
667.0	1.33	Leanding portion		*Insmet Cuba – Doppler meteorological Obs Part B*
500.0	1.50	Tormentas eléctricas		*Insmet Cuba – Doppler meteorological Obs Part B*
450.0	1.46	Convectivas	Fujiwara, 1965	*Fujiwara, M. , 1965/Gerstner-Heinemann, 2005*
200.0	1.60	Estratiforme – Banda C – Italia	Krajewski et al, 1995	*Picciotti et al., (2008)/(Krajewski et al., 1995*
800.0	1.60	Conectivo – Banda C – Italia	Krajewski et al, 1996	*Picciotti et al., (2008)/(Krajewski et al., 1996*
348.0	1.81	Estratiforme – Banda X – Japon – Italia	Maki et al., 2005	*Picciotti et al.,(2008)/MAki et al., 2005*
134.0	1.55	Convectivo – Banda X- Japon-Italia	Maki et al., 2006	*Picciotti et al.,(2008)/MAki et al., 2006*
162.0	1.48	Convectivo Dakar Senegal	Nzeukoe, Sauvageot (2002)	Nzeukoe ,Sauvageot (2002)/ da Silva Moreas (2003)
371.0	1.24	Estratiforme Dakar Senegal	Nzeukoe, Sauvageot (2002)	Nzeukoe, Sauvageot (2002)/ da Silva Moreas (2003)

Fig. 29 Relaciones Z – R.

Fuente: Gómez Vargas (2015)

La atenuación, ya mencionada antes, es uno de los principales problemas de los radares banda X. Debido a ella, el alcance de estos radares es pequeño a comparación de los radares que usan otras bandas de frecuencia. No obstante, esa no es la única de sus consecuencias. La atenuación de la señal es más intensa a medida que los rayos atraviesan las tormentas que se encuentran en la atmósfera. De manera similar a la luz al atravesar un cuerpo translucido, el radar disminuye la precisión con la que se reportan otras tormentas que se encuentran más allá de la primera.

La atenuación así descrita, presenta un grave problema: limita la "visión" del radar justo cuando es más necesario. Un caso particularmente importante sucede cuando se producen tormentas intensas sobre el radar. Esta situación se denomina "radomo húmedo" en referencia a la cubierta del radar. Una precipitación intensa sobre el radar puede reducir dramáticamente la calidad de los datos. Dado que se presentan lluvias intensas en Piura durante el FEN, la corrección de la atenuación debe ser una de las actividades prioritarias para un uso efectivo del radar.

Muchos de los algoritmos modernos utilizan las características polarimétricas para corregir la atenuación. Aquellos algoritmos que no utilizan ese enfoque realizan una corrección puerta – puerta (Jacobi & Heistermann, 2016). Este enfoque corrige la atenuación sobre los datos de un haz del radar, a través de un parámetro denominado PIA (Path Integration Attenuation). Sin embargo, este enfoque puede ser inestable, y disminuir la calidad de los datos si no se utiliza adecuadamente. Mal utilizado, "el remedio podría ser peor que la enfermedad".

Pese a la gran lista de problemas mencionados, estos no son los únicos presentes en el radar, sino que corresponden a algunos de los más importantes y aquellos que tienen o pueden tener influencia en los datos del radar PIUXX. La banda brillante, cambios de fase entre otros son problemas adicionales que no son tratados en este documento. Sin embargo, la literatura científica ofrece mucha información al respecto de estos problemas y los algoritmos para su corrección. Como ya debería ser usual a este punto, muchos de los algoritmos están basados en características polarimétricas que no se encuentran disponibles.

Los datos del radar, por tanto, deben sufrir diversas correcciones para garantizar su aplicabilidad. Sin una corrección de los errores y su conocimiento, los datos se prestarán a malas interpretaciones y no cumplirán su propósito. No obstante, debe tenerse presente que las correcciones no son deterministas ni perfectas. Por ello, estas correcciones deben ser validadas para garantizar su rendimiento, y, sobre todo; que no añadan nuevos errores a los datos. Las correcciones no deben ser vistas como procesos concluidos que solucionan todos los problemas. En su lugar deben verse como procesos perfectibles y sujetos a cambios y mejoras en el tiempo.

Capítulo 4
Cadena de procesamiento predeterminada

En esta sección se describe brevemente la cadena de procesamiento actual de los datos del radar PIUXX, que es proporcionada por el proveedor del radar (Selex, Alemania). En primer lugar, se proporcionará una visión general de la cadena de procesamiento. Posteriormente, el resto del capítulo se dedicará a explicar las correcciones disponibles y la generación de productos.

4.1 Información general

La cadena de procesamiento actual parte de la obtención de los datos crudos de reflectividad (Z) y se centra más en la generación de productos de carácter hidrológico que en la corrección de los datos. Esto se debe a las limitaciones que presenta este especifico modelo de radar meteorológico, a partir del cual el software solo garantiza la generación de algunos productos básicos como el plano indicador de posición o la precipitación acumulada. Pero para radares de mayores capacidades (polarimétricos y/o Doppler) la gama de productos que se pueden generar es mayor.

Actualmente, el procesamiento corresponde a una configuración inicial realizada en el software Rainview Analyzer. Dado que el radar PIUXX ha sido instalado recientemente, está configuración es provisional y puede estar sujeta a cambios.

Los datos crudos son generados a partir de las configuraciones realizadas en el software Rainbow 5. Estos son presentados en dos versiones: corregida y no corregida. La diferencia radica en una serie de filtros utilizados por el software e implementados mediante un árbol de decisión. Estas dos versiones de los datos crudos se guardan por separado en archivos con la terminación dbZ y dbuZ respectivamente.

Los datos pasan a Rainview Analyzer donde a través de una serie de tareas son pre – procesados y se generan productos operacionales. Posteriormente, estos pueden ser proyectados y cambiados a formatos comunes como .png, .jpg, .netcdf. La elección de las tareas a utilizar es decisión del usuario.

Actualmente, la única corrección utilizada corresponde a un mapa de clutter estático. Los productos generados comprenden el Indicador Plano de Posición (PPI), Tasa de Precipitación y Precipitación Acumulada en el lapso de una hora.

Debe tenerse en cuenta que la principal limitación de esta cadena de procesamiento es el tiempo. Las correcciones y generación de productos se realizan en periodos cortos. Por ello, los algoritmos implementados se caracterizan por su poca complejidad computacional. No debe confundirse, sin embargo, la poca complejidad computacional con resultados pobres. Los algoritmos tratan de resolver los problemas de acuerdo con sus limitaciones. Esta idea estará presente en el resto del capítulo de forma implícita.

4.2 Correcciones a bajo nivel

Dado que los datos se recolectan por medio de un software comercial, se encuentran ciertas restricciones al nivel de datos que se encuentran disponibles para el usuario. Usualmente, los datos en los niveles más bajos de la cadena de procesamiento del radar constituyen datos de voltaje o corriente que son captados por medio de la antena receptora al recibir radiación electromagnética (Figura 30). Estos datos son procesados mediante un sistema A/D y amplificados para poder determinar la potencia recibida, y a través de la ecuación del radar se determina el parámetro de reflectividad (Z). El radar PIUXX proporciona como datos crudos estos últimos, por lo que la cadena de procesamiento de datos se encuentra ya en un nivel de procesamiento más alto.

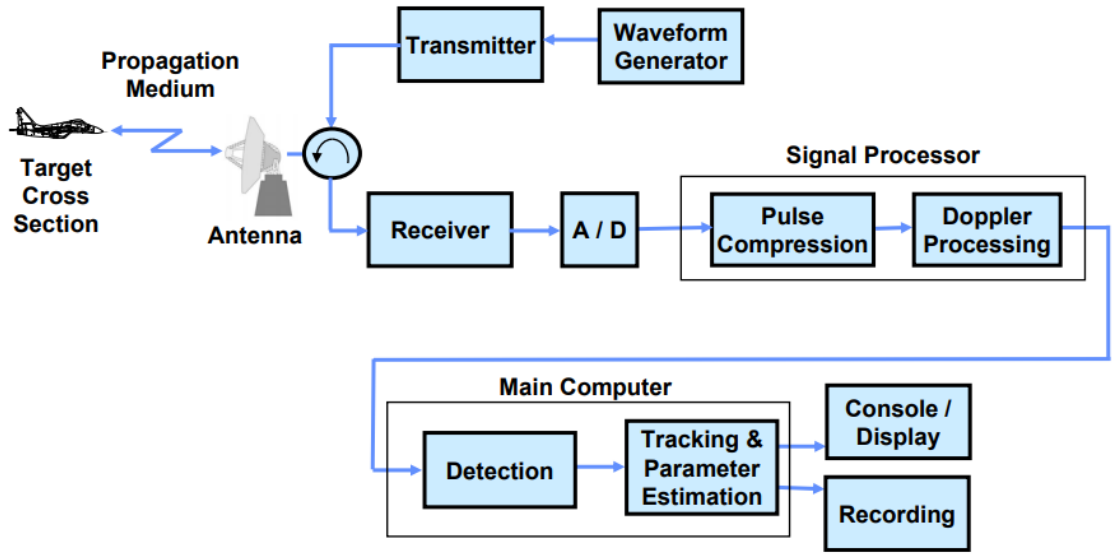


Fig. 30 Diagrama de bloques típico del procesamiento inicial de los datos del radar
Fuente: O'Donnel (2007)

La mayor parte de las correcciones se llevan a cabo de manera automática en la unidad Signal Processor. Por ello, las correcciones que se detallan en esta sección corresponden solo a aquellas que son conocidas.

Los datos son sometidos a un proceso de árbol de decisión donde se filtran los valores espurios. Este proceso comprende pruebas de coherencia entre los datos, pruebas estadísticas y la aplicación de un umbral de detección. Para más detalles del algoritmo véase la documentación del software o los apéndices presentados por (Joss & Lee, 1995). El filtrado

no es configurable y distingue los valores crudos de reflectividad en datos de reflectividad corregida (dBZ) y no corregida (dBuZ). En base a la inspección realizada, el filtrado involucra la eliminación de los datos en la zona ciega del radar. Además del árbol de decisión se pueden realizar otras correcciones. La primera de ellas involucra la corrección de los datos debido a la atenuación de la amplitud con la distancia. La segunda consiste en un filtro speckle espacial o radial con parámetros configurables (Figura 10).

Otras configuraciones como el tiempo de generación y la aplicación de un filtro de mediana están también disponibles (Figura 31). Como ya se mencionó, a medida que se utilizan intervalos de generación más pequeños es más probable que más ruido aparezca en los datos.

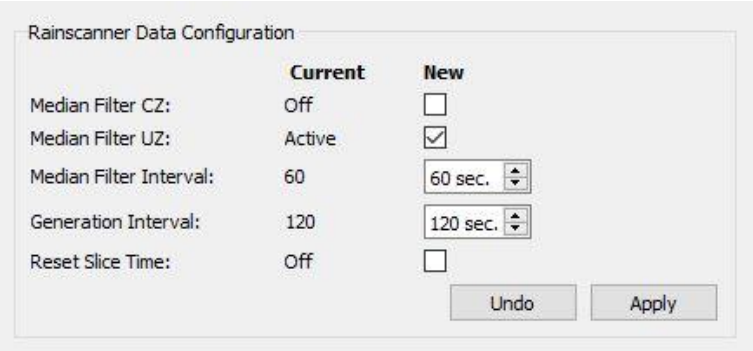


Fig. 31 Intervalo de generación y filtros de mediana.
Fuente: SELEX (2017b)

El filtro de mediana es un filtro de suavizado y puede perder los detalles más pequeños. Dado que el algoritmo no se encuentra disponible solo se puede suponer que lo más probable es que este filtro sea unidimensional y se implemente en el eje temporal. En ese caso, el tamaño de la ventana será el correspondiente al número de escaneos durante el intervalo de generación.

Los filtros speckle y mediana están destinados a disminuir el ruido en los datos. Sin embargo, es precisamente por su acción que los objetos más pequeños como los aviones detectados desaparecen de los datos.

4.2 Pre – procesamiento

Una vez generados los datos crudos, estos pasan a RainView Analyzer, donde reciben las primeras correcciones y se generan los productos. Durante el pre – procesamiento es posible aplicar dos algoritmos.

El primero de ellos es Corrección de ocultación (OCC). Ha sido diseñado para reemplazar zonas de apantallamiento en los datos. Estas se ingresan de manera manual a través de la definición de áreas bloqueadas con forma de trapecio circular. Estas se definen a partir de los puntos extremos, proporcionando los limites azimutales de la zona y la distancia a partir de la cual inicia el apantallamiento. El reemplazo de los datos se realiza por

interpolación lineal. Debido a que no existen problemas significativos de apantallamiento detectados, esta corrección no es utilizada por el radar PIUXX.

El otro procedimiento disponible es Mapa de Clutter 3D y Procesamiento de datos (3DCDP). Este consiste en la aplicación de la aplicación de un cluttermap y la selección de zonas para interpolación o descarte. A través de estas correcciones se puede eliminar la presencia del clutter terrestre y las zonas apantalladas.

La elaboración del cluttermap se basa en la composición estadística de datos conseguidos bajo condiciones libres de precipitación. El usuario selecciona los datos pasados que no corresponden a precipitación y a partir de estos se genera un cluttermap representativo. La selección puede parecer un tanto arbitraria, pero está basada en la naturaleza fija del clutter y con datos correctamente escogidos debería proporcionar buenos resultados. Además de la selección de los datos y el tipo de composición (mediana, media, cuartiles, etc.), es posible configurar una opción denominada “clutter spread” que considera el valor más alto como representativo dentro de una ventana de tamaño real fijo. Algunos parámetros extras están disponibles para su configuración.

A partir de la elaboración del cluttermap existen dos posibilidades para eliminar el clutter. La primera posibilidad, denominada Marcado y corrección (Flagging and Correction) se centra en la detección del clutter e interpolación. La segunda, denominada Sustracción, se centra en su eliminación.

El algoritmo de Marcado y corrección corresponde a la interpolación de los datos marcados por el cluttermap. Esta interpolación puede ser lineal (azimutal o radial) o bilineal (azimutal y radial) dependiendo de la disponibilidad de los datos adyacentes. La ventaja de este algoritmo se aprecia cuando la precipitación se superpone en la zona de clutter. Bajo esas circunstancias, la interpolación proporciona una recuperación del patrón de precipitación en la zona (Figura 32). No obstante, el algoritmo puede presentar problemas de “estabilidad”. Imagine que el cluttermap no ocupase totalmente la zona del clutter terrestre. ¡Los puntos restantes de clutter constituirían los datos adyacentes a partir de los cuales se interpolaría! En ese caso, la interpolación podría dar resultados aún peores si no se selecciona adecuadamente el cluttermap.

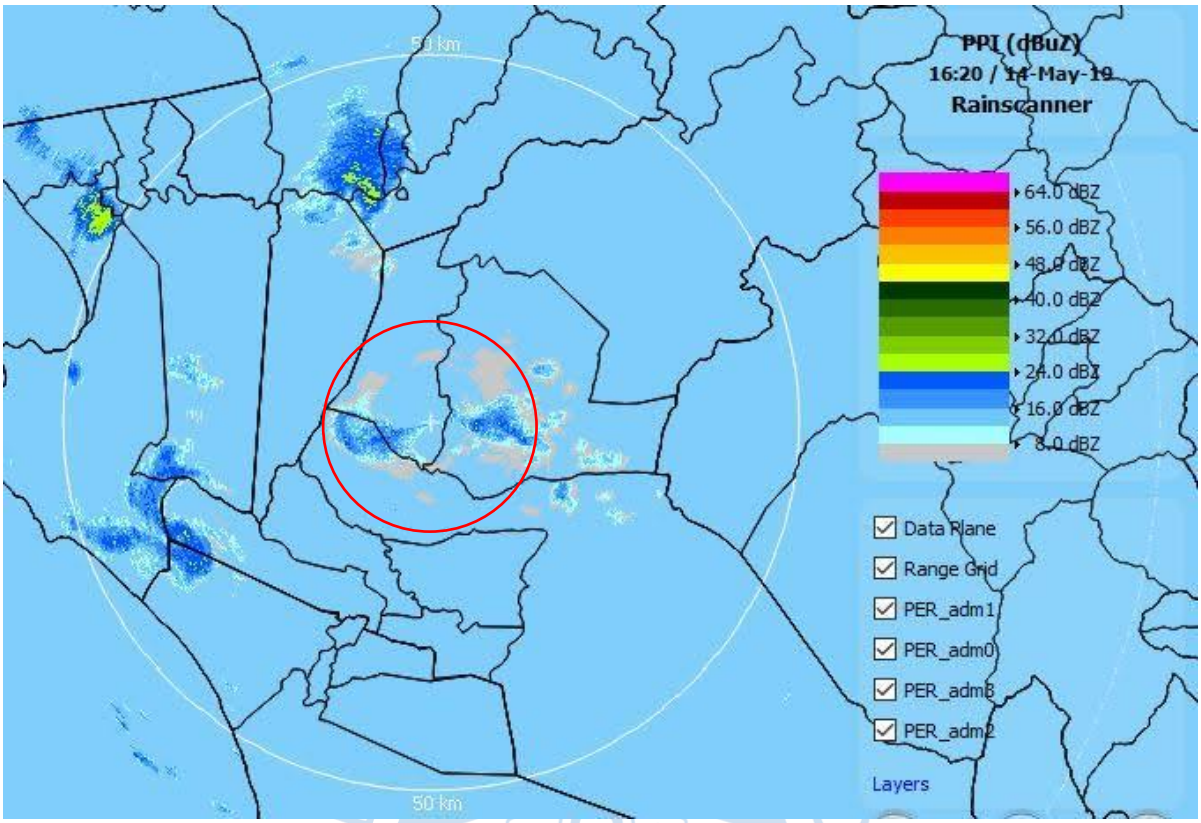


Fig. 32 Caso "Precipitación tipo 2" corregido mediante Marcado y corrección

El algoritmo Subtraction consiste en restar el mapa de clutter a los datos crudos tal y como se muestra en la Figura 33. A diferencia del anterior, este método no presenta problemas frente a la presencia de precipitación ni tampoco problemas de estabilidad. Sin embargo, dependiendo del método de composición es posible que queden algunos datos de clutter sin filtrar.

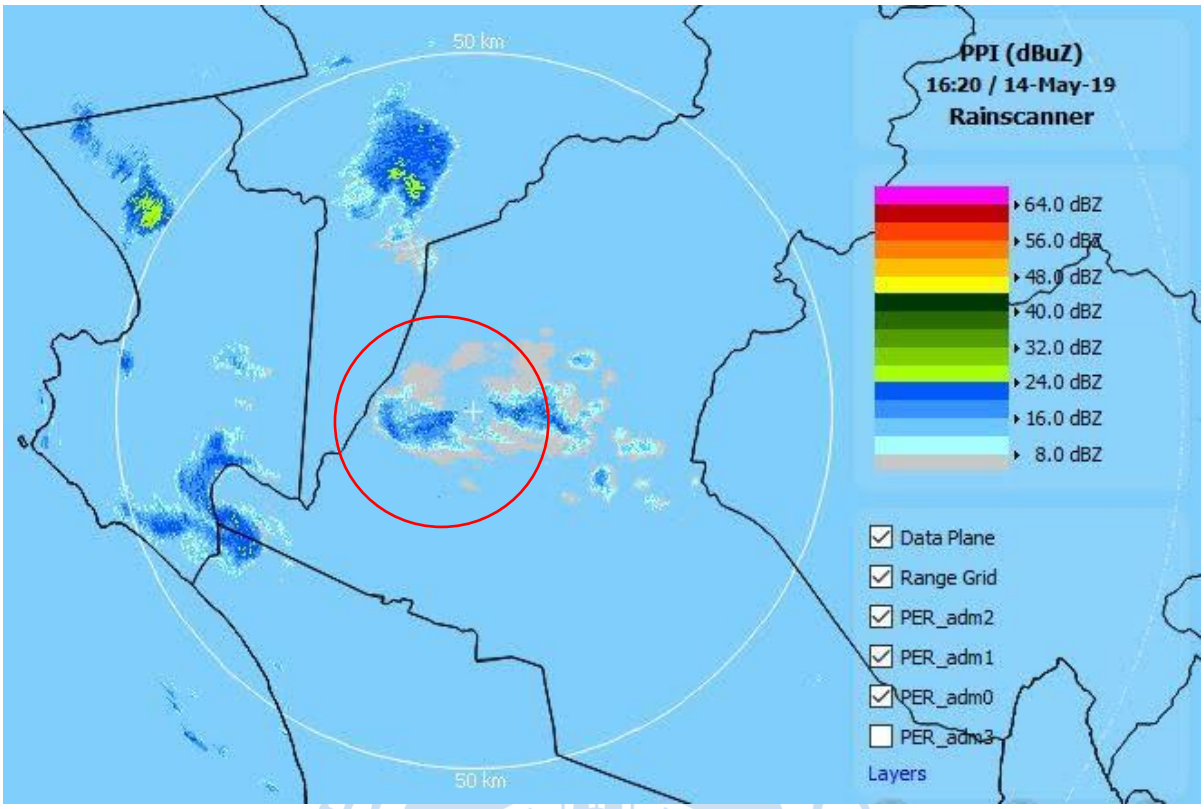


Fig. 33 Caso "Precipitación tipo 2" corregido mediante Sustracción

Existe un último método, que no fue mencionado antes porque no es aplicable a los datos del radar PIUXX. Este consiste en la extrapolación de los datos de clutter a partir de ángulos de escaneos superiores.

Además del cluttermap, es posible seleccionar zonas adicionales de manera gráfica para su interpolación o extrapolación, así como colocarlos en “No data”. Todos estos procedimientos dependen enteramente de configuraciones que realice el usuario.

4.3 Productos

Luego de las correcciones del pre – procesamiento los datos están disponibles para la generación de productos operacionales. Estos productos están divididos en dos niveles. El primer nivel comprende aquellos productos básicos que se pueden generar a partir de los datos crudos. El segundo nivel, en cambio, utiliza como entradas los productos del primer nivel u otros del segundo nivel, tal y como se muestra en la Figura 34. Algunos productos del segundo nivel pueden requerir información adicional de otros instrumentos.

Los productos se basan en la definición de tareas apilables. Cada producto presenta dos o tres parámetros configurables. Sin embargo, dada la naturaleza apilable, un producto final puede tener muchos parámetros asociados. El apilamiento simplifica la definición de las tareas y las hace más accesibles al usuario.

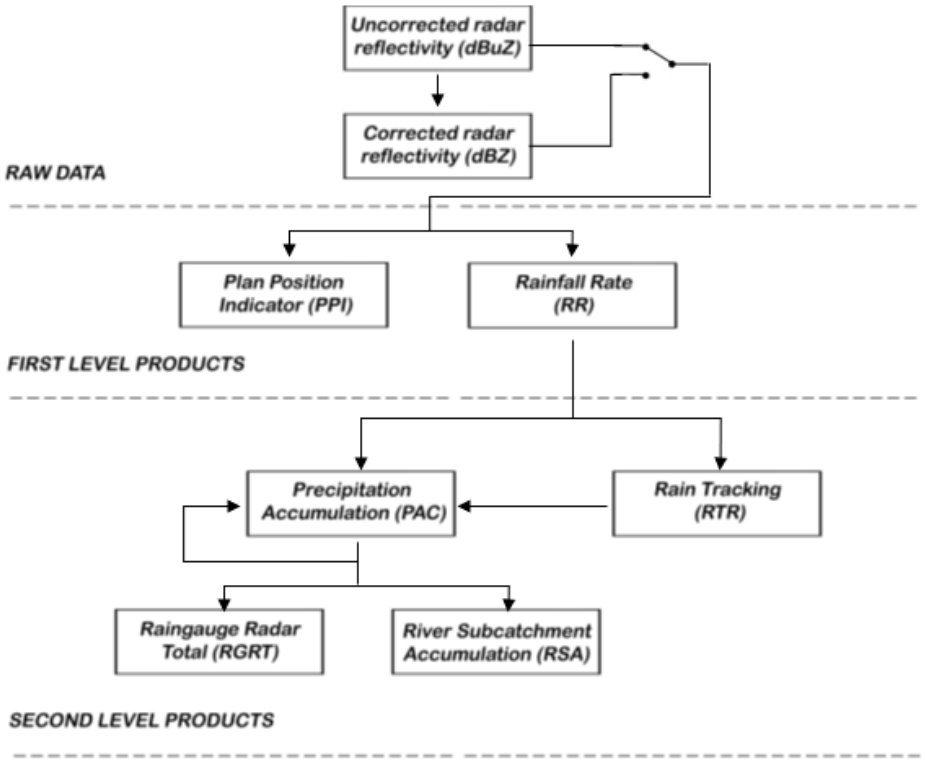


Fig. 34 Estructura de los productos.

El primer nivel abarca dos productos: Plano Indicador de Posición (PPI) y Tasa de precipitación (RR). El primero de ellos, consiste en la clásica visualización de los datos de cualquier radar. Este muestra los datos de reflectividad o precipitación utilizando las coordenadas polares (r,θ) correspondientes al escaneo durante un ángulo de elevación $\phi = cte$ Esta representación es la más usual de los datos, y propiamente corresponde a un sistema polar. Se usa para realizar una rápida identificación de las tormentas. Sin embargo, al menos en su presentación el PPI proporcionado consiste en una proyección de los datos y no la representación en un sistema polar. Para ello, requiere del alcance máximo a representar y la resolución en pixeles para la proyección de los datos.

Debido al aumento de la altitud con la distancia, el PPI no corresponde a la precipitación a una altura ni volumen constantes. Cuando se dispone de datos adquiridos en un barrido volumétrico, se utiliza en su lugar un producto denominado CAPPI, que si corresponde a los datos a altura constante. Este producto tiene aplicaciones más útiles que el PPI.

La tasa de precipitación (RR) se estima aplicando una relación $Z-R$ ($Z=aR^b$) a los datos crudos. La relación usada por defecto corresponde a la relación de Marshall – Palmer ($a=200$ y $b=1.6$). La representación de los datos es exactamente igual a la descrita para el PPI. Por tanto, requiere de los mismos parámetros de configuración (alcance y resolución). A diferencia del PPI que proporciona información de carácter cualitativo, la información de la tasa de

precipitación es de carácter cuantitativo. Sin embargo, los valores solo serán válidos si la relación $Z-R$ ha sido correctamente calibrada.

Los productos del segundo nivel están centrados principalmente en aplicaciones hidrológicas. Sin embargo, uno de ellos ha sido diseñado para realizar predicción a muy corto plazo. Los productos de este nivel son cuatro: Precipitación Acumulada (PAC), Rastreo de precipitación (RTR), Acumulación en cuenca (RSA) y Radar – Pluviómetro Total (RGRT).

El producto PAC consiste en los valores de precipitación acumulada (mm) durante cierto intervalo de tiempo (Figura 35). Este producto es aquel que posee el mayor tipo de entradas posibles, puesto que se puede realizar a partir de un RTR, RR o incluso a partir de otro PAC. Si la entrada corresponde a los productos RR o RTR, los datos son calculados a partir de un promedio acumulado entre los datos dentro del intervalo. Si la entrada consiste en otros PAC, la salida corresponde a una simple acumulación de los valores. Este producto sirve de base para la mayoría de las aplicaciones hidrológicas.

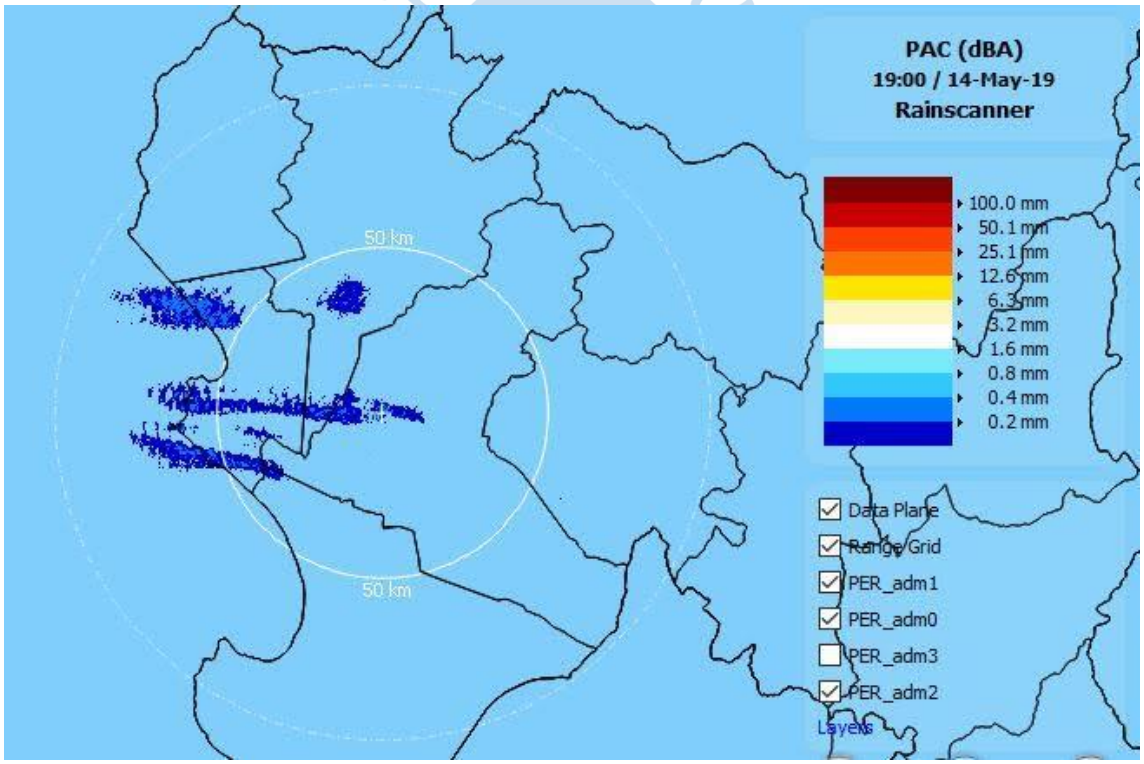


Fig. 35 Precipitación acumulada estimada de la ligera llovizna del 14 de mayo del 2019

El producto RTR permite realizar una predicción a corto plazo (nowcasting) con un horizonte de tiempo de hasta una hora. Este producto proporciona una secuencia de predicciones en horizonte de tiempo seleccionado que pueden ser configurados hasta un valor de veinte. El algoritmo se basa en dividir los datos en bloques y buscar coincidencias entre cada elemento con los elementos más parecidos en los datos previos. A partir de la identificación se estima el movimiento de cada bloque y se interpola el campo de velocidad. Este es el parámetro que finalmente se utiliza para realizar la predicción. A diferencia de otros

productos proporciona varias salidas entre las cuales se encuentran los datos iniciales, la secuencia de datos predichos y el producto PAC asociado a la predicción. Este último puede utilizarse para incrementar la resolución temporal de los datos y mejorar la acumulación de precipitación.

El producto RSA evalúa el caudal que ingresa a una cuenca a partir de sus coordenadas y la precipitación acumulada en dicha zona. Para ello, requiere como entradas el producto PAC y la información de las coordenadas de la cuenca definidas en una tabla. Proporciona como resultado un archivo en formato ASCII que contiene el volumen de agua que ingresa a la cuenca (Q_{total}), así como el caudal promedio (Q_{ave}), la precipitación acumulada (R_{total}) y la tasa de precipitación promedio (R_{ave}). Esta información está destinada a ser utilizada por los modelos hidrológicos tradicionales en tiempo real.

El producto RGRT integra y compara la información de la precipitación acumulada con la información de pluviómetros. De esta forma, resulta sencillo contrastar los datos del radar con la información de los pluviómetros que actúan como referencia.

Debido a que esta parte podría ser un poco densa de entender se presenta una tabla de resumen (Tabla 5).

Tabla 5 Productos de Rainview Analyzer

Siglas	**Producto**	**Definición**
Productos de primer nivel		
PPI	Plan Position Indicator	Representación cartesiana de la reflectividad para un ángulo de elevación.
RR	Rainfall Rate	Tasa de precipitación (mm/h) calculada en base a una relación $Z-R$
Productos de segundo nivel		
PAC	Precipitation Accumulation	Precipitación acumulada durante un intervalo de tiempo
RTR	Rain Tracking	Predicción de tasa de precipitación en un horizonte de tiempo determinado a intervalos regulares
RSA	River Subcatchment Accumulation	Proporciona el volumen de ingreso, el caudal promedio, la precipitación acumulada y la tasa de precipitación promedio de la cuenca proporcionada.
RGRT	Raingauge Radar Total	Precipitación acumulada e información de pluviómetros durante el mismo intervalo de tiempo.

4.4 Post – procesamiento

Por defecto, solo los datos crudos pueden ser leídos y procesados por programas de software libre. Los productos de ambos niveles, en cambio, solo pueden visualizarse en los programas del propio software como RainScoutMet o RainDart. Debido a ello, se ofrecen técnicas de post – procesamiento orientadas a la transformación de la data en formatos comunes para poder utilizar los productos procesados.

1) Controlador de formatos externos (EFH)

Permite transformar a formatos estándares de radares como BUFR y HDF5. A diferencia del resto de productos, se realiza a través de comandos en línea.

2) Proyección de imagen

Permite referenciar los datos en longitud y latitud usando un sistema de proyección estándar (proj4). Este procesamiento no cambia el tipo de producto generado, por ello es posible seguir generando otros productos partir de este.

3) Conversión a ráster

Permite convertir los productos en formatos tipo ráster como GeoTiff , BitMap, netCDF, JPG, entre otros para su incorporación en un Sistema de Información Geográfica (SIG).

4) Conversión a formato vectorial

Permite convertir los productos de información múltiple (RSA y RGRT) en formatos shapedata para su incorporación en un Sistema de Información Geográfica (SIG). Los formatos a los cuales se puede convertir son Atlas BNA, ESRI Shapefile, GeoRSS, GML, GMT y KML.

5) Conversión ASCII

Convierte los productos en formato ASCII para poder leerlos en diversos programas.

4.5 Evaluación

La cadena de procesamiento que se puede elaborar a través del software resulta sencilla y potente para la eliminación del clutter. Sin embargo, no se proporciona ningún algoritmo para la corrección de los problemas de atenuación ni de eliminación de la propagación anómala. Dado que el radar PIUXX es un radar de banda X, la atenuación es un problema particularmente importante. Esta carencia podría limitar el uso de los datos, especialmente durante precipitaciones intensas como las que se presentan durante el Fenómeno El Niño.

Los productos que se pueden generar son principalmente útiles para aplicaciones hidrológicas. Los productos de segundo nivel, especialmente aquellos que utilizan información externa resultan un tanto tediosos de configurar. El hecho de tener que adaptar dicha información a un formato particular es tal vez el inconveniente más grande para que

productos como RSA y RGRT sean utilizados. Por otra parte, la precipitación acumulada que proporciona el producto PAC muestra un patrón antinatural debido a que se realiza en intervalos de 5 minutos (Figura 35). No obstante, ese patrón puede ser fácilmente subsanado aprovechando un aumento de resolución temporal utilizando el producto RTR.

La exportación de los datos a formatos estándar resulta critica para el aprovechamiento posterior de los datos. La gran cantidad de formatos que proporciona el software hace que este aspecto se encuentre plenamente cubierto.

Salvo condiciones especiales, la cadena de procesamiento predeterminada se presenta como una buena opción para el procesamiento de los datos. Paradójicamente, el gran problema que posee esta precisamente ligado a su sencillez. Debido a ella, los procesos presentan pocos parámetros y son estandarizados. En tanto los datos representan una oportunidad para probar algoritmos, la estandarización así como la imposibilidad de adicionar correcciones limitan su uso efectivo. En ese sentido, es la idea de que la cadena puede mejorarse y la búsqueda de nuevas aplicaciones para los datos la principal motivación para explorar otras alternativas de procesamiento.

UNIVERSITAS STUDIORUM PIURENSIS

Capítulo 5
Cadenas de procesamiento alternativas

En este capítulo se exploran los distintos softwares donde se puede llevar a cabo el procesamiento de los datos, sus ventajas y limitaciones.

A diferencia del resto de capítulos, es necesario dar razones para que una cadena de procesamiento alternativa tenga relevancia por sí misma. Si ya tenemos una cadena de procesamiento sencilla y potente … ¿Por qué molestarnos en elaborar otra?

Como ya se ha visto, la cadena de procesamiento abarca solo algunas correcciones. Debido a su requerimiento temporal la mayoría de los procesos son sencillos. No todos los problemas mencionados antes se solucionan. Por ejemplo, se dejan de lado las correcciones de propagación anómala y la atenuación. Por tanto, la cadena de procesamiento predeterminada resulta más adecuada para la visualización e información cualitativa. Sin embargo, son necesarias correcciones posteriores para poder utilizar los datos con carácter cuantitativo.

La elección de una cadena de procesamiento es una decisión de dos pasos. Por una parte, se seleccionará los algoritmos y demás correcciones que deben realizarse sobre los datos. Por otro lado, la programación se lleva a cabo en un software o un lenguaje de programación. Su elección es tan importante como los algoritmos, pues determinará la complejidad de la implementación.

En este capítulo se discutirá precisamente la elección del software y las opciones disponibles. Estas se han separado en dos grandes grupos: software propietario y software libre.

5.1 Software propietario

El termino software propietario hace referencia a todo software donde no es posible para el usuario acceder al código fuente y/o realizar modificaciones sobre este[1]. Las restricciones limitan la distribución y modificación con carácter legal.

No poder acceder al código fuente no necesariamente debe ser visto como algo malo. La mayoría de los programas informáticos que usamos pertenece a esta categoría, y dudo que

[1] https://es.wikipedia.org/wiki/Software_propietario

el lector encuentre problemas con ello. Como usuarios no necesitamos conocer cómo funcionan los programas por dentro, sino que los vemos como simples herramientas.

En el caso del radar meteorológico, los softwares disponibles son ofrecidos por los fabricantes de radares. Por ello no es de extrañarse la gran proporción de propietarios de radares que usan este tipo de software. Además, el vínculo con los fabricantes resulta particularmente importante, pues la mayor parte del entrenamiento para el uso de los radares proviene de estos (cerca del 60%). (Sireci & WMO, 2015)

Para analizar este tipo de software y sus características, se seleccionaron los tres principales softwares (exceptuando el ya visto Rainbow) y se compararon a través del folleto informativo que proporciona cada fabricante. Los programas elegidos fueron: IRIS[2], EDGE[3] y FROG-MURAN[4].

El programa IRIS, propiedad de la empresa SIGMET, es propiamente un sistema de información de radar. Comprende el monitoreo, análisis, generación de productos y visualización.

El programa EDGE es distribuido por la empresa norteamericana Enterprise Electronics Corporation. Corresponde al acrónimo de Enterprise Doppler Radar Graphics Environment, por lo que ha sido diseñado para radares con capacidades Doppler. Comprende el control del radar, así como el análisis, visualización y transferencia de los datos.

FROG-MURAN es propiedad de la empresa alemana GAMIC. Está diseñado para soportar cualquier radar con capacidades Doppler. Corresponde propiamente a un sistema con 11 aplicaciones mediante las cuales se procesa y visualiza los datos.

De acuerdo con la comparación entre folletos, los programas anteriores presentan características muy similares. Se encuentran divididos en varios programas y presentan una interfaz al usuario. Todos ellos están basados en la generación de productos a través de tareas (Task). Al mismo tiempo, los productos que ofrecen son prácticamente los mismos.

Las diferencias son pocas, pero existen. A partir de los folletos, IRIS y FROG-MURAN ofrecen más opciones para incluir otras fuentes de información como información satelital o generar composites usando la información de más de un radar. Por su parte, EDGE ofrece más algoritmos para la generación de predicciones. Además, aunque no sea posible inferirlo de los folletos, existirán diferencias entre los algoritmos que utilizan para generar cada producto. Estas diferencias no serán demasiadas, pero son las que determinan la performance de un software frente a otro.

---

[2] https://www.vaisala.com/sites/default/files/documents/IRIS-Focus-Brochure-B211502EN.pdf

[3] http://www.radarmeteo.com/downloads/EDGE.pdf

[4] https://www.gamic.com/signal-processing/frog-muran-weather-radar-software/

El software propietario también ha sido evaluado por otros interesados como el Real Instituto de los Países Bajos (KNMI). Este comparó Rainbow 5, IRIS 8 y MURAN 3 (en ese entonces FROG y MURAN eran software en competencia). Al igual que el pequeño análisis presentado, el software como conjunto es muy parecido. Las principales diferencias radican en el control y configuración del radar, el método de proyección geográfica de los datos y la configuración de los productos. El informe publicado en el 2005 dio como ganador al software Rainbow 5 (Iwan Holleman & Beekhuis, 2005).

Debido al parecido entre el software propietario, cambiar de software no parece añadir opciones de corrección y post procesamiento para los datos. Adicionalmente, el alto costo de estos los hace opciones inviables para el radar PIUXX. Como consecuencia, podría parecer que cualquier procesamiento deseado debe ser programado como si fuese por primera vez.

5.2 Software libre

El software libre es aquel software cuyo código fuente puede estudiarse, modificarse y redistribuirse y mejorarse de manera libre[5]. Como ejemplos tenemos el sistema operativo LINUX, el lenguaje de programación Python o la biblioteca libre OPENCV.

Poder acceder al código fuente tiene innumerables ventajas. Por una parte, el código puede estudiarse, entenderse desde otras perspectivas y mejorarse. La distribución del software libre facilita los procesos fácilmente reproducibles y el desarrollo de estándares entre los involucrados. De esta manera se facilita la transferencia de conocimientos y se evita perder tiempo “reinventando la rueda”.

El código abierto no es ajeno a los problemas. Involucrarse en un software de este tipo requiere de conocimientos básicos de programación. En el software propietario se seleccionaban valores y opciones a través de una interfaz sencilla. En cambio, en el software libre un mínimo de conocimiento de programación resulta esencial. Adicionalmente, dado que al inicio los proyectos de este tipo son llevados por pocas personas, resulta vital conocer su visión acerca del tipo de software que desarrollan.

El software libre tiene una gran presencia en la comunidad de usuarios de radares meteorológicos. Su desarrollo comenzó desde los años 90 cuando se lanzó el software RSL como librería proporcionada por la NASA. En el año 2002, se lanzó LROSE/TITAN una biblioteca de código abierto para el procesamiento y lectura de datos de radares meteorológicos. Posteriormente, el software BALTRAD+ fue lanzado como librería de código abierto en el 2012 (Heistermann et al., 2015). Con el desarrollo de más herramientas de este tipo, el software libre se presenta como un complemento excelente para el post procesamiento de los datos del radar.

[5] https://es.wikipedia.org/wiki/Software_libre

El desarrollo del software libre se fortaleció a partir de la publicación conjunta de los principales desarrolladores en el 2013. Desde entonces, se ha desarrollado una comunidad de software de radares meteorológicos (https://openradarscience.org/). Como parte de la colaboración conjunta, se ofrecen cursos de cada software durante cada ERAD, la principal conferencia de radares meteorológicos.

Actualmente, existe un gran número de desarrollos de software que se ocupan del procesamiento de los datos de radares meteorológicos. Propiamente no todos corresponden a librerías o a aplicaciones. Sin embargo, son mencionados pues muchos de ellos podrían crecer hasta convertirse en una librería o al menos inspirar al desarrollo de algoritmos u aplicaciones de los datos.

A continuación, se proporciona una lista de los paquetes encontrados con la información técnica correspondiente (Tabla 6 y Tabla 7).

Tabla 6 Características técnicas de los desarrollos de software

Nombre	Versión	Distribución	Lenguaje en el cual fue programado	API	Plataforma
Ama	0.1	Github	Python	Python	Linux, Mac, Windows
ARM Py – ART	1.10.2	Conda, Github, Pypi	Python, Cython, C, FORTRAN	Python	Linux, Mac, Windows
ART – View	1.2.3	Conda, Github	Python	Python	Linux, Mac, Windows
Baltrad	3.0	Git	C, Java, Python	C, Java, Python	Linux, Mac
BioRAD	0.4.0	CRAN Repository	R	R	Linux, Mac, Windows
HKO-7	-	Github	C, Python	Python	Linux, Windows
INTA-Radar	3.0.0	Github	Python, Shell	Python	Linux, Windows
MMM-Py	1.6	Github	Python	Python	Linux, Mac
MultiDop	0.3	Github	C, Python, Roff, FORTRAN	Python	Linux, Mac, Windows
Precipitation-Nowcasting	-	Github	Python	Python	Linux
Py-RadarMet	-	Github	Python	Python	Linux, Mac, Windows
Pysteps	1.1.0	Conda, Github, Pypi	Python	Python	Linux, Mac, Windows
PyTDA	1.1.1	Github	Python	Python	Linux, Mac
Radar.IRIS	1.0.0	CRAN Repository	R	R	Linux, Mac, Windows
RadX (TITAN/LROSE)	2019/01/29	Github	C, C++, Perl, Python	Python	Linux, Mac
Rainymotion	0.1	Github	Python	Python	Linux, Mac, Windows
RSL	1.50	Servidor propio	C	C	Linux
SwirlsPy	2.0.7	Conda, Gitlab	Python, …	Python	Linux

TINT	0.1.0	Github	Python	Python	Linux, Mac, Windows
Weather radar GUI	-	Github	Python	Python	Linux, Mac, Windows
Wradlib	1.5.0	Conda, Github, Pypi	Python	Python	Linux, Mac, Windows
Wradvis	-	Github	Python	Python	Linux, Mac, Windows

Tabla 7 Comentarios acerca de los desarrollos de software

Nombre	**Principal aspecto**	**¿Ligado a algún otro desarrollo?**	**Comentarios**
Ama	Aplicación móvil Visualización Procesamiento básico	Wradlib en back-end	Desarrollado como proyecto de tesis de grado. Lee archivos en formato binario HDF5. No presenta formato de librería
ARM Py – ART	Procesamiento Proyección Visualización	Wradlib para lectura de varios formatos	Biblioteca ampliamente usada Apoyada por el Instituto Argonne Utiliza la clase Radar para gestionar los datos
ART – View	GUI Visualización	ARM Py-ART en back-end	Complemento visualizador de Py-ART Presenta muchas dependencias, por lo que la importación es lenta Visualizador potente
Baltrad	Lectura de datos Procesamiento Exportación a otros formatos	-	Desarrollo modular Presenta numerosas herramientas
BioRAD	Visualización Detección de aves e insectos	-	Opciones diversas para visualización Muy bien documentada
HKO-7	Nowcasting	-	Complemento del Paper “Deep Learning for Precipitation Nowcasting: A Benchmark and A New Model”. Apoyada por el Observatorio de Honk Kong Implementa Persistencia Euleriana.

			Presenta 2 modelos de flujo óptico: ROVER, variante no lineal. Presenta 4 modelos de redes neuronales: TrajGRU, ConvGRU, 2D CNN y 3D CNN.
INTA-Radar	Lectura Composición Proyección Detección de granizo	Wradlib y Py-ART como dependencias	Desarrollado como proyecto para los radares INTA. Diseñado para trabajar con los formatos Rainbow (.vol o .azi) No presenta formato de librería
MMM-Py	Lectura Visualización	-	Diseñado para leer los formatos de mosaicos multi radar multi sensor (MRMS) distribuidos por la NOAA. Desarrollado por la NASA
MultiDop	Análisis tridimensional de la velocidad	Py-ART para la lectura de datos	Requiere información de dos o más radares sobre la misma zona. Desarrollado por la NASA y el instituto Argonne
Precipitation-Nowcasting	Nowcasting	-	Implementa un modelo ConvLSTM No escrita como librería, pero es lo suficientemente sencillo como para adaptarlo o usarlo
Py-RadarMet	Cálculos fundamentales de radar	-	Principalmente útil para desarrolladores Muchas de las funciones han migrado hacia Wradlib u otros paquetes
Pysteps	Nowcasting	-	Implementa 4 modelos de nowcasting: Lucas-Kanade, DARTS, VET y Difusión anisotrópica. Implementa nowcasting probabilístico y opciones de descomposición en cascada.
PyTDA	Análisis de turbulencia	Py-ART para la lectura de datos	Trabaja con información de un escaneo PPI o volumétrico.

			Desarrollado por la NASA y el instituto Argonne
Radar.IRIS	Lectura Procesamiento básico	-	Desarrollado para leer el formato IRIS de VAISALA
RadX (TITAN/LROSE)	Lectura Procesamiento	-	Presenta adaptaciones para la lectura en programas como Matlab y Java
Rainymotion	Nowcasting	Wradlib como dependencia menor	Implementa 5 modelos de nowcasting: Persistencia Euleriana, Sparse, SparseSD, Dense y DenseRotation
RSL	Lectura Procesamiento	-	-
SwirlsPy	Lectura Procesamiento Esquemas de acumulación	-	Apoyada por el Observatorio de Honk Kong Distribución solo a las agencias meteorológicas nacionales. Muy bien documentada
TINT	Identificación Seguimiento	Py-ART para la lectura de datos	Implementación en Python del algoritmo TITAN
Weather radar GUI	GUI Visualización	-	Lee archivos binarios
Wradlib	Lectura Procesamiento Proyección	-	Una de las librerías más populares Abarca prácticamente la totalidad de la cadena de procesamiento
Wradvis	GUI Visualización	Wradlib en back-end	Visualizador de Wradlib en desarrollo

En la Tabla 6 se muestran las características técnicas de los desarrollos. Se proporciona la versión, los canales de distribución, el lenguaje en que fue programado y en el que se puede utilizar así como el sistema operativo en el cual funciona.

La distribución indica la vía por la cual se proporcionan los paquetes. También proporciona una idea de la dificultad de la instalación. Conda representa una distribución a través de Anaconda Cloud. Mediante el uso de entornos, representa la manera más sencilla de instalación. Por su parte, Github representa repositorios públicos alojados en Github. La dificultad de instalación es variable y depende de cada repositorio. Mientras que algunos se encuentran empaquetados y muestran claramente las dependencias, otros solo proporcionan el código fuente sin ninguna otra indicación. Git y Gitlab resultan parecidos a Github, con el añadido que se deben conocer los comandos git básicos. Pypi representa una distribución a través del Python Package Index. La instalación se realiza a través de un comando. Sin embargo, la solución de conflictos entre dependencias con otros paquetes instalados previamente la hacen una de las opciones más tediosas. Por último, el servidor propio se refiere a todos aquellos paquetes no indexados de maneras usuales. La configuración y dificultad dependerán de cada paquete.

La plataforma menciona el sistema operativo necesario para ejecutar los paquetes. Resulta particularmente importante en el entorno local pues la mayoría de los piuranos usan un sistema operativo Windows, y cambiar de sistema operativo puede parecerles un cambio muy grande.

API proporciona el lenguaje de programación en el cual es posible utilizar cada paquete. Como se observa en la Tabla 6, Python es el lenguaje de programación con más paquetes disponibles.

El lenguaje en el cual fue programado proporciona la referencia del código fuente. Si la documentación fuese incompleta o se requiera estudiar o modificar el código fuente, probablemente requiera conocer los lenguajes de programación mencionados.

Adicionalmente, la comunidad de software de radares meteorológicos ofrece una máquina virtual que comprende los principales paquetes.

Wradlib y Py-ART representan las dos librerías más completas para el procesamiento de los datos. La gran cantidad de algoritmos, su comunidad y las herramientas construidas en torno suyo las respaldan. Ambas son buenas opciones como base de casi cualquier cadena de procesamiento.

Debido a su importancia, ambas se presentan con mayor detalle.

5.2.1. Wradlib

Wradlib (Heistermann et al., 2013; Pfaff, Heistermann, & Jacobi, 2012) es una de las primeras librerías de código abierto para el procesamiento de datos de radares meteorológicos. Fue desarrollada en el 2012 por Thomas Pfaff, Maik Heistermann y Stephan Jacobi. El proyecto fue financiado por el Ministerio Federal Alemán de Investigación y Educación (Heistermann et al., 2015). Desde entonces, Wradlib ha evolucionado hasta convertirse en una solución profesional para el análisis de datos de radar meteorológico. Su naturaleza interactiva y el hecho que abarca prácticamente la totalidad de la cadena de procesamiento la convierten en una solución completa. Actualmente, es utilizado tanto por agencias gubernamentales como en el mundo académico. Su comunidad de usuarios **wradlib-users** cuenta con 200 miembros activos que proporcionan una comunidad ágil y con amplio soporte.

Wradlib representa una de las plataformas más versátiles para el procesamiento de datos de radares meteorológicos debido las siguientes características :

1) Multiplataforma : Debido a su implementación en Python, puede usarse tanto en Windows como en Linux
2) Gran cantidad de formatos de lectura
3) Implementación de alto nivel

Wradlib basa su implementación en Python. La versión actual, 1.5.0 cuenta con 17 módulos (Tabla 8).

Tabla 8 Módulos de Wradlib

Módulo	Propósito
wradlib.adjust	Calibración pluviométrica
wradlib.atten	Corrección de atenuación
wradlib.classify	Clasificación de los hidrometeoros
wradlib.clutter	Identificación de clutter
wradlib.comp	Composición de datos
wradlib.dp	Radares polarimétricos
wradlib.georef	Georreferenciación
wradlib.io	Entrada y salida de datos
wradlib.ipol	Interpolación
wradlib.qual	Métricas de calidad
wradlib.trafo	Transformación de los datos
wradlib.util	Herramientas diversas
wradlib.verify	Comparación pluviométrica
wradlib.vis	Visualización
wradlib.vpr	Perfil vertical de reflectividad(VPR)

wradlib.zonalstats	Estadísticas locales
wradlib.zr	Conversiones Z – R

5.2.2 Py – ART

Python ARM Radar Toolkit (Helmus & Collis, 2016) es un paquete para leer, visualizar, corregir y analizar datos de radares meteorológicos. Fue desarrollado como un proyecto del Laboratorio Nacional Argonne y el Centro de Investigación Climática de Medición de Radiación Atmosférica. Pese a su reciente lanzamiento, ha sido acogido por una gran proporción de usuarios de radares meteorológicos como complemento de otras librerías de código abierto.

Al igual que Wradlib, su implementación está basada en Python. La versión actual 1.10.1 está organizada en once módulos (Tabla 9).

Tabla 9 Módulos de Py -ART

Módulo	Propósito
pyart.core	Clases básicas y cambio a coordenadas cartesianas
pyart.io	Entrada y salida de formatos estándar
pyart.aux_io	Entrada y salida de formatos especiales
pyart.correct	Correcciones para radares Doppler y polarimétricos
pyart.retrieve	Recuperación de datos típicos
pyart.graph	Visualización
pyart.map	Visualización en coordenadas cartesianas
pyart.filters	Filtrado de datos
pyart.util	Herramientas diversas
pyart.bridge	Enlace a herramientas externas
pyart.testing	Herramientas beta (en desarrollo)

A diferencia de Wradlib se presentan menos herramientas para la calibración e integración de datos de estaciones pluviométricas terrestres, pero se compensa con la gran cantidad de herramientas que se han construido alrededor de él como TINT, ART-View, MultiDop y otras.

Capítulo 6
Cadena de procesamiento propuesta

En este capítulo se verá a detalle los pasos que comprende la cadena de procesamiento alternativa. Los procedimientos que se explicarán están basados en los algoritmos disponibles en las librerías discutidas en el capítulo anterior. Adicionalmente, se diseñan y programan algunos otros algoritmos en base a descripciones disponibles en la bibliografía.

6.1 Lectura de datos

Los datos crudos del radar se distribuyen usando la extensión ".azi". El nombre que reciben se encuentra determinado por la fecha y hora en la cual fue generado el archivo. Se utiliza la siguiente designación:

yyyyMMddhhmm00000xxx.azi

Donde las letras "y" representa el año, "M" el mes, "d" el día, "h" la hora y "m" el minuto. Las letras "x" corresponden al tipo de archivo, dbuZ para datos no corregidos y dbZ para datos corregidos.

La extensión ".azi" representa al escaneo PPI o azimutal y se encuentra en formato XML. El archivo consta del encabezado típico y meta data en dicho formato y la sección correspondiente a los datos en formato binario (BLOB). Adicionalmente la sección BLOB generalmente se encuentra comprimida. Debido a ello, la decodificación de los datos por parte del usuario muchas veces no es posible. Por ello, se limita el uso de los datos al software propietario. El uso del formato binario es una de las mayores barreras para el uso efectivo de los datos (Belmonte & Saibene, 2017; Heistermann et al., 2015).

Los formatos crudos han sido asimilados para lectura por la librería Wradlib. Este formato fue provisto por la misma empresa Gematronik, por lo que el código está exento de errores o de naturaleza empírica. El resto de las librerías capaces de leer el formato utilizan un envoltorio de la función de Wradlib para leer este tipo de archivos.

La función de lectura en Wradlib está basada en el uso de la clase OrderedDict. La clase hereda de los diccionarios (clase Dict). A diferencia de ellos, cuyos par clave-valor se pueden mostrar en cualquier orden, los pares clave-valor de OrderedDict se mostrarán siempre en el

orden en el cual han sido agregados. Esta clase se distribuye en la librería predeterminada Collections.

Al usar la función se crea una instancia de la clase OrderedDict que sigue la estructura mostrada en el Apéndice 1. Esta se organiza usando otras instancias OrderedDict imbuidas. Los metadatos corresponden casi en su totalidad a cadenas de caracteres (str) que necesitan transformación. Los datos, en cambio. Se encuentran codificados en formatos uint. La descripción de los parámetros y la estructura del objeto se encuentran descritas en el Manual de Software de Formato de archivos (SELEX, 2017ª), donde se detallan las estructuras utilizadas para almacenar y distribuir los datos.

Debido al formato uint que presentan los datos, es necesario escalarlos para convertirlos a valores en reflectividad (dBZ). La escala utilizada es de naturaleza lineal. Esta presenta como valor mínimo -31.5 dBZ y como valor máximo 95.5 dBZ con una digitalización de 16 bits. Todos estos valores se encuentran almacenados como meta data en el objeto.

$$v_{dBZ} = v_{\min} + v_{u\,\mathrm{int}} \times \frac{v_{\max} - v_{\min}}{2^{\#bits}} \qquad \textbf{Ec. 21}$$

Los datos se presentan en una matriz de dos dimensiones. Las dimensiones están ligadas a la cuadricula polar, es decir, al número de divisiones del ángulo azimutal y del rango. Después de los valores de reflectividad, estos constituyen los valores más importantes.

Los valores del rango se definen por el rango máximo y el número de divisiones. A partir de estos valores, fácilmente se puede crear el vector correspondiente.

Los valores azimutales se distribuyen usando un esquema similar al de los datos de reflectividad. Sin embargo, se diferencian en que el valor mínimo corresponde a 0° y los datos presentan una resolución de 1° por lo que la fórmula utilizada varía ligeramente. Al igual que los anteriores, estos valores se incluyen en la meta data.

$$v_{azi} = v_{u\,\mathrm{int}} \times \frac{v_{\max / range}}{2^{\#bits}} \times res \qquad \textbf{Ec. 22}$$

El resto de meta data comprende la localización geográfica, datos del radar, estrategia de escaneo y configuraciones de los filtros y umbrales utilizados. Dado que su estructura corresponde por defecto a una cadena de caracteres, se requiere conocer su significado a detalle antes de realizar cualquier transformación. En los scripts Metadata_radar y Nuevo_entender_datos del Apéndice 5 se proporcionan pautas para su uso. Como guía básica de esta sección véase el cuaderno *Load and inspect data from a Rainbow file* de la documentación de Wradlib.

La otra posibilidad de lectura de datos consiste en utilizar la librería Py-ART. Sin embargo, esta solo utiliza una función envoltorio alrededor de la función de Wradlib. La razón por la cual se le dedica atención es a causa de la propia librería. Esta utiliza la clase propia

Radar para sus procedimientos. Debido a ello, el uso de la librería para lectura es un tanto diferente.

La lectura de datos en Py-ART es mucho más sencilla. A diferencia de Wradlib, no hay necesidad de realizar conversiones ni fórmulas para transformar los datos, pues estas se incluyen en la función envoltorio. Sin embargo, existen diferencias importantes. La principal es la clase Radar, cuya arquitectura está basada en el formato CF/ Radial Data. Este formato es de carácter autodescriptivo, por lo cual, los datos y los metadatos presentarán meta data descriptiva asociada. El uso de una clase *propia* hace que el usuario tenga que familiarizarse con los atributos y métodos de esta. Debido a que no existe ningún ejemplo que utiliza data de formato Rainbow, puede ser difícil para un usuario novato poder utilizar la clase sin complicaciones. Además, los formatos Rainbow correspondientes a valores de reflectividad no corregidos generan un campo llamado "unfilltered_reflectivity". Este campo no es utilizado directamente por las funciones de Py-ART y tiene que ser renombrado a "reflectivity" para una poder utilizar las funciones de manera sencilla.

Los datos en Py-ART son diferentes. Mientras que Wradib usa como estructura básica la clase **numpy.ndarray**, Py-ART utiliza matriz enmascaradas de la clase **numpy.ma.array**. Debido a ello, algunas operaciones son un tanto diferentes. Adicionalmente, antes se mencionó que los valores azimutales y de rango definen una cuadricula matricial que corresponden a cada dato. Mientras que los datos de Wradlib proporcionan el inicio de cada intervalo, los datos de Py-ART se encuentran centrados en cada compartimiento de la cuadricula (Figura 36).

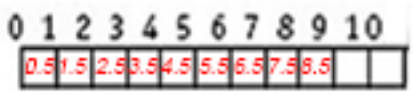


Fig. 36 Esquema de cuadriculas Wradlib (negro) y Py-ART(rojo).

Los datos (leídos en cualquiera de las dos librerías) no se encuentran ordenados de manera azimutal. Por ello, no se pueden realizar operaciones que involucren más de un dato sin un ordenamiento adicional. Dada la naturaleza polar de los datos, esta transformación consiste en un desplazamiento de los datos desde el primer ángulo encontrado en los valores azimutales hacia el valor de cero.

Debido a lo expuesto, la lectura de los datos en Wradlib no es tan simple como llamar a una función. Por eso, se decidió crear una clase envoltorio basada en diccionarios que recogiese los datos transformados y la meta data más importante. Esta clase se denominó RadarData y constituye la base de lectura de los procesos siguientes. Sin embargo, esta solo es una clase de paso. Considero que usar una clase reviste de complejidad el problema para el usuario y desarrollador. Si bien ayuda a la gestión de la meta data, aumenta la complejidad de las funciones y agranda la curva de aprendizaje. De esa manera no se hace más que propiciar el uso de la clase como una caja negra o hacerla poco atractiva.

6.2 Visualización

Dado que no siempre llueve, es necesario seleccionar los datos de precipitación. El resto de los datos no tiene utilidad en sí mismos. Para el usuario que trabaja con los datos, la representación de estos le resulta fundamental. Ya sea para un análisis exploratorio o para ejemplificar un suceso particular, la visualización es una pieza necesaria para el aprovechamiento de los datos.

Los datos crudos y los productos pueden ser visualizados a través de Rainview Analyzer. Sin embargo, los datos solo podrán observarse si se posee la licencia del software. Ello limita las opciones para que múltiples personas trabajen con la misma información simultáneamente.

Existen múltiples opciones para solucionar el problema. La opción más sencilla es convertir los datos a un formato estándar. De esta manera se pueden leer y visualizar en cualquier programa. Esta opción que parece ideal contiene ciertos obstáculos. Los datos se encuentran en una cuadricula polar. Por tanto, su representación directa origina una presentación bastante inusual (Figura 37).

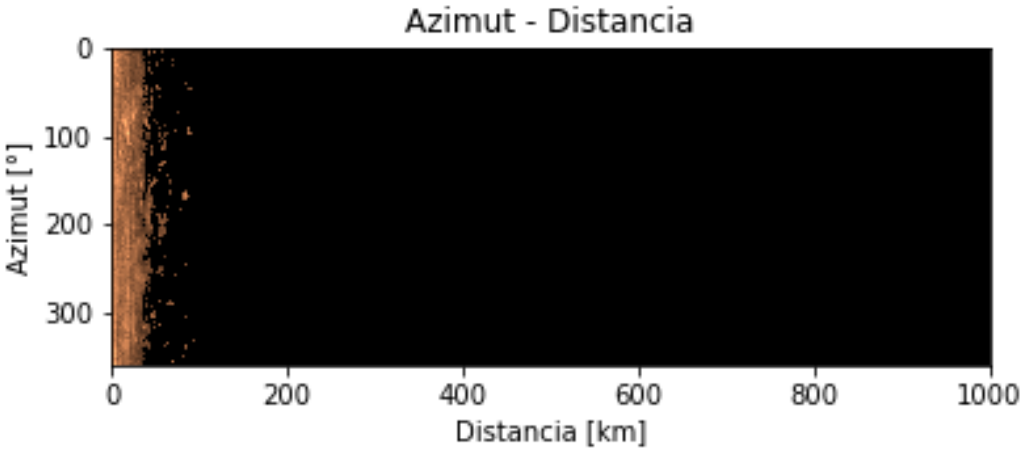


Fig. 37 Representación de los datos en la cuadricula polar (distancia-azimut).

Para representar los datos sobre una cuadricula cartesiana existen dos opciones. La primera de ellas es usar un sistema polar que represente los datos. Sin embargo, los visualizadores trabajan con sistemas cartesianos, por lo que una herramienta así probablemente no exista. La otra opción consiste en proyectar la data al sistema cartesiano o a un sistema de proyección geográfica. Este método incluye transformaciones en los datos como interpolaciones y georreferenciación. Si bien esto no afecta demasiado la visualización, los datos que se proporcionarían ya no serían datos crudos.

La proyección de datos crudos es soportada por Wradlib y Py-ART por lo que es posible generar representaciones sencillas. Sin embargo, para analizar un conjunto de datos, los gráficos generados en línea o a través de la GUI de Matplotlib pueden no ser la mejor opción por la cantidad de gráficos o ventanas que se generan.

Opciones como ART-View y Wradvis representan soluciones encaminadas a resolver ese problema. ART-View es una herramienta poderosa. No obstante, la cantidad de dependencias la hacen una herramienta de carga pesada. Además, al aumentar el número de dependencias se dificulta la integración en un entorno determinado. Como muestra, ART-View actualmente es incompatible con la versión más reciente de Matplotlib. Wradvis, en cambio, parece encaminado a ser una alternativa más liviana y con las mismas funcionalidades. Por desgracia, es un proyecto secundario de Wradlib y aún se encuentra en desarrollo.

Debido a ello, se decidió crear un visualizador propio, que permitiese la lectura del formato Rainbow, ligero y con pocas dependencias. Debido a sus características se decidió nombrarlo Wradchibi (**W**eather **Rad**ar **Chibi**). El programa lee los archivos Rainbow y está basado en la GUI de Matplotlib. Además de la visualización, proporciona opciones de guardado, zoom, y muestra parte de la meta data (Figura 38). El código se muestra en el Apéndice 3.

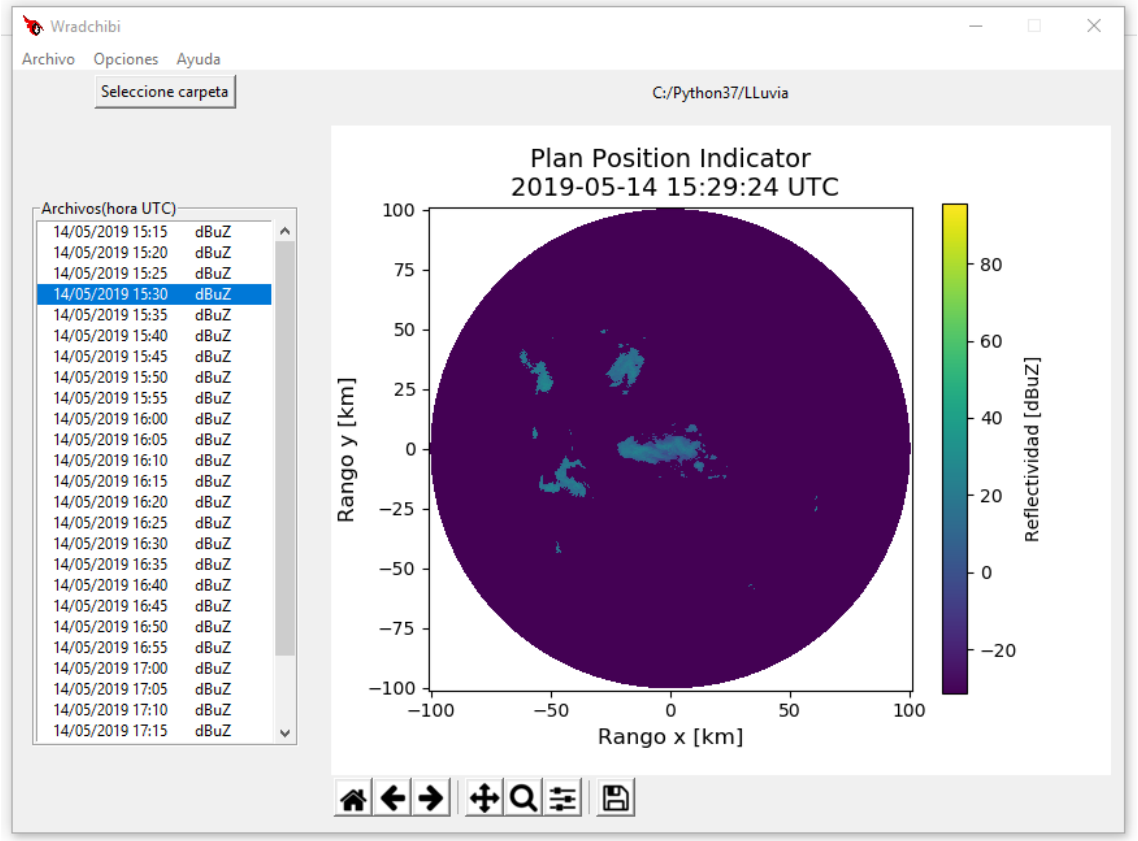


Fig. 38 Interfaz de Wradchibi.

La interfaz creada es sencilla e intuitiva con el usuario. El desarrollo puede utilizarse en cualquier computadora pues está enteramente desarrollado con software libre. Sin embargo, no ha sido empaquetado. La razón es que las herramientas involucran la programación. Dado que este no es un ámbito ampliamente conocido aquí, prefiero que traten con la programación desde el inicio en lugar de ocultarla. Por ello, la programación que utiliza es

funcional en lugar de la clásica programación orientada a objetos usada en este tipo de desarrollos.

6.3 Corrección por apantallamiento

Dependiendo de la topografía del terreno cercano al radar se puede obtener el efecto de apantallamiento mencionado. A fin de determinar su existencia se utilizó la librería Wradlib para calcular la fracción de bloqueo del haz a lo largo de su trayectoria. Para ello, se utilizaron las funciones **wradlib.qual.beam_block_fraction** y **wrl.qual.cum_beam_block_frac** que se basan en el esquema propuesto por (Bech et al., 2003). Se pueden encontrar más detalles en el cuaderno respectivo en el Apéndice o en la documentación de la librería.

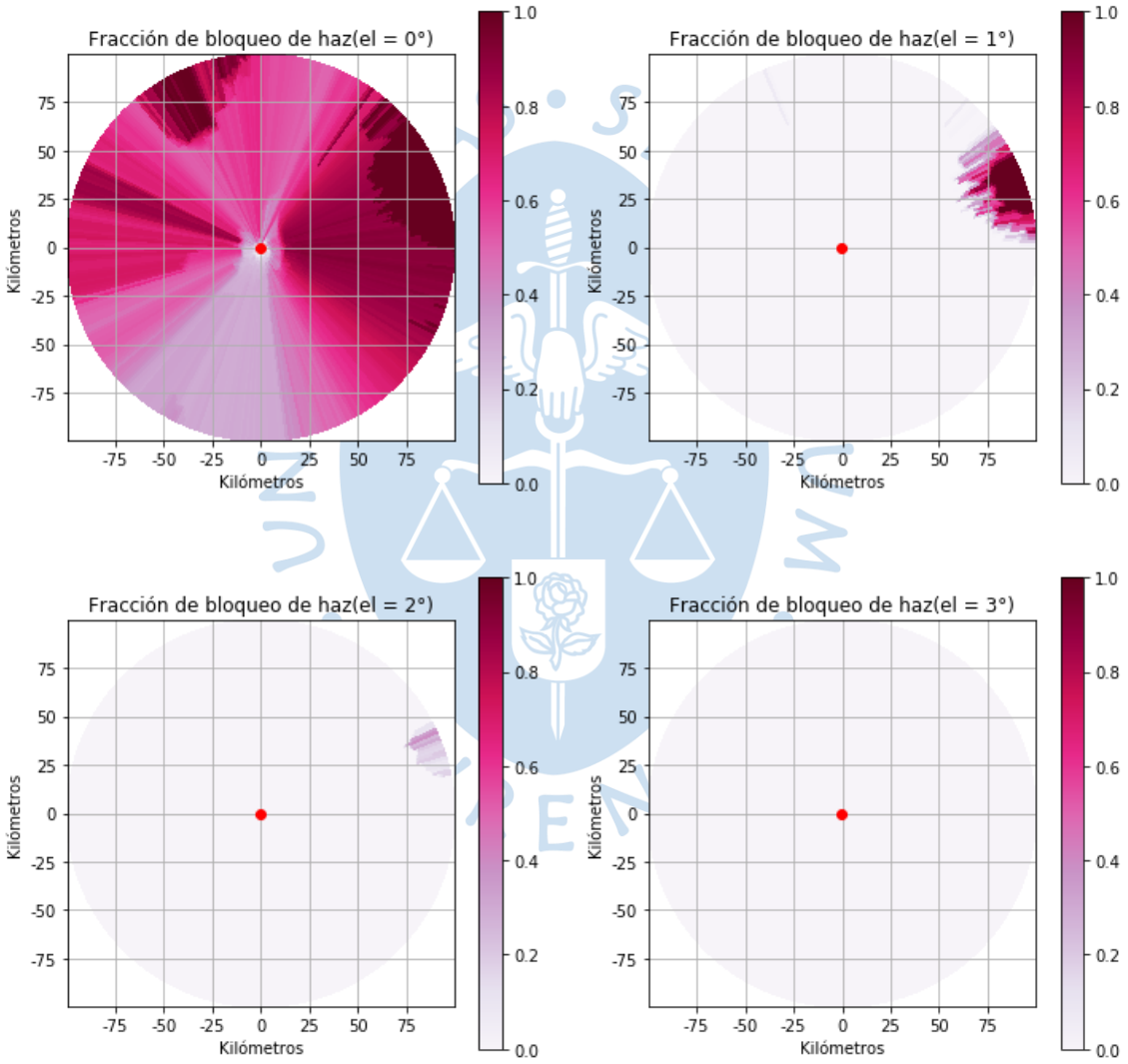


Fig. 39 Fracción de bloqueo del haz para diferentes ángulos de elevación para el radar PIUXX.

Los resultados (Figura 39) muestran que no es posible utilizar un ángulo de elevación $\phi = 0°$. Existirán demasiadas zonas donde los datos no serán confiables, lo que disminuiría considerablemente las prestaciones del radar. Por otra parte, para el ángulo de elevación

$\phi = 3°$ y superiores el radar evita cualquier bloqueo. Sin embargo, ángulos de elevación superiores presentan problemas relacionados con la altura y no son convenientes para aprovechar todo el alcance del radar.

Casos particularmente interesantes son los dos ángulos centrales ($\phi = 1°$ y $\phi = 2°$). En el primero de ellos, se puede observar un posible bloqueo en dirección noreste debido a la cordillera de los Andes en Ayabaca (Figura 09). De aparecer este en el radar, puede aprovecharse que solo representa una pequeña fracción para ser usado como objetivo de referencia para la calibración y estimación de la atenuación (Delrieu, Caoudal, & Creutin, 1997). Para el ángulo de elevación $\phi = 2°$ el bloqueo es prácticamente inapreciable y no aparece en radar como clutter. Al momento de la elaboración de este documento, no se han presentado precipitaciones en esa zona como para verificar el efecto del bloqueo de haz.

En consecuencia, el apantallamiento no debería ser un gran problema. De acuerdo con el análisis, solo una pequeña sección puede ser afectada por este fenómeno. Sin embargo, a partir de los datos de precipitación acumulados hasta el momento no se puede afirmar ni descartar la presencia del fenómeno.

Durante el ERAD 2018, otro procedimiento similar ha sido codificado. Este utiliza DEM de alta resolución y mapea los obstáculos presentes en lugar de asignarles un valor promedio. El procedimiento no se basa en el ancho del haz como el procedimiento utilizado por defecto en Wradlib sino que utiliza la distribución de potencia Gaussiana para determinar el efecto. Lamentablemente, no se dispone de un DEM con la suficiente resolución como para realizar las pruebas.

La determinación experimental puede realizarse a través de un simple acumulado de precipitación. En este, las zonas con una precipitación acumulada muy baja muy probablemente corresponderán a zonas apantalladas. Para ello, habrá que esperar la estación lluviosa. Si se desea, puede utilizarse la función histo_cut de la librería Wradlib para su identificación. Esta función detecta las anomalías en el histograma de los valores de precipitación acumulada (Figura 40). Las anomalías superiores corresponderán a clutter, mientras que las anomalías de más bajo valor corresponderán a las zonas apantalladas. Sin embargo, al tratarse de un método estadístico, es necesario acumular muchos datos para que la información sea confiable. La documentación recomienda un plazo de meses a años para su uso.

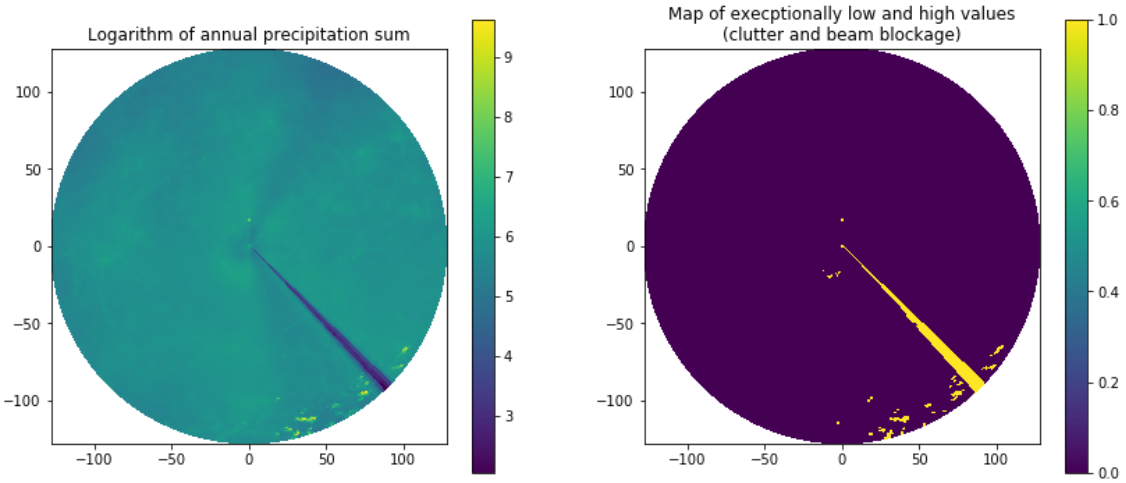


Fig. 40 Ejemplo de la función histo_cut.
Fuente: Documentación de Wradlib

A partir de la identificación de las zonas apantalladas, se pueden realizar correcciones por interpolación. Sin embargo, esto solo deberá hacerse si la zona es relativamente pequeña. De lo contrario, la estimación en base a cualquier interpolación puede llevar a conclusiones erróneas.

6.4 Detección de clutter

Los algoritmos de detección de clutter aplicables a radares de polarización simple son muy pocos. Como se vio en un capítulo anterior, los autores se han concentrado en el desarrollo de algoritmos que involucran la velocidad Doppler y parámetros polarimétricos. Ninguno de ellos es aplicable aquí. De los procedimientos que quedan, la mayoría utiliza fuentes de información alternativa para determinar la presencia de clutter. Aun así, quedan por analizar algoritmos como la función histo_cut, el filtro Gabella y los filtros de máscara o cluttermap (véase Tabla 04).

El filtro histo_cut se basa en la detección de anomalías estadísticas en los datos de precipitación acumulada. Para ello, requiere de acumulados de varios meses o de años. Obviamente el procedimiento es aplicable con menos datos, pero su calidad será mucho menor. Dado que no se tiene más que un evento de precipitación en los datos, no será posible por ahora determinar la presencia de apantallamiento. Sin embargo, la presencia de clutter si se puede determinar. Basado en la idea del filtro histo_cut, se realizó un acumulado no de precipitación, sino de valores diferentes a reflectividad mínima. Estos permitirán realizar estimaciones como si se tratasen del filtro histo_cut. Los resultados se presentan en la Figura 41.

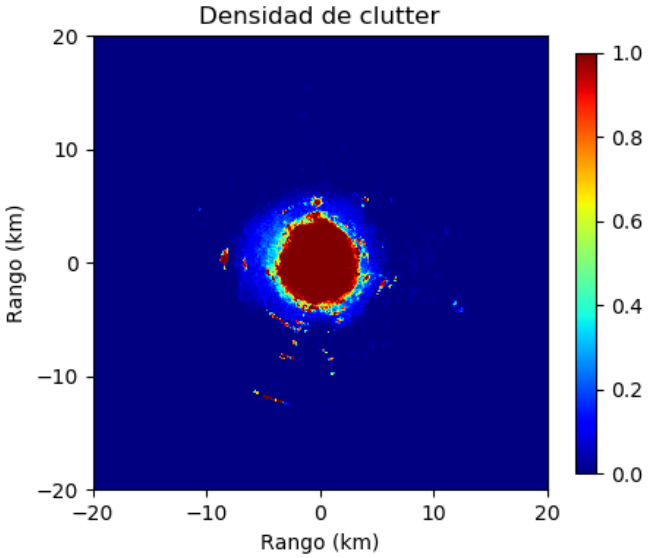


Fig. 41 Probabilidad de presencia de clutter.

Los datos representan la probabilidad de presencia de clutter en determinada ubicación. Los resultados se distinguen del clutter teórico estudiado en secciones pasadas. El clutter no es tan fijo como se podría suponer al inicio, sino que puede aparecer o no en algunos lugares. Esto se debe a que si bien los objetos que provocan clutter son fijos, las condiciones atmosféricas que determinan la propagación del haz no lo son. Por ello, se encontrarán variaciones no solo de valor sino también de presencia.

En la Figura 41 se observa una zona de clutter persistente alrededor del radar, mientras que existen pequeños objetos con las mismas características a medida que nos alejamos de este. Téngase en cuenta que esta zona abarca mucho más que la zona ciega producida por el efecto del uso de una única antena para transmisión y recepción. A su vez, existen zonas donde el clutter puede o no aparecer. Estas plantean inconvenientes para la detección del clutter en el caso que se presente precipitación en las cercanías del radar.

Al igual que el filtro histo_cut (Wradlib), es posible plantearse la detección del clutter a partir de los datos, simplemente escogiendo un umbral para el cual se reconoce como clutter/no clutter.

El otro filtro aplicable consiste en el filtro Gabella (Gabella & Notarpietro, 2002). Este se basa en evaluar la textura de los datos. Las zonas de clutter presentan texturas de valores altos que pierden correlación rápidamente. El algoritmo también aprovecha el tamaño de los elementos para eliminarlos. Si los elementos ocupan muy pocas celdas en la grilla, es muy poco probable que correspondan a precipitación. El filtro proporciona buenos resultados para eliminar los valores de los objetos pequeños, pero no para eliminar la concentración de clutter cercana al radar (Figura 42). Esto se debe a que el filtro no fue diseñado para funcionar por si solo y debería ser usado posteriormente a la aplicación de un mapa de clutter (Gabella & Notarpietro, 2002).

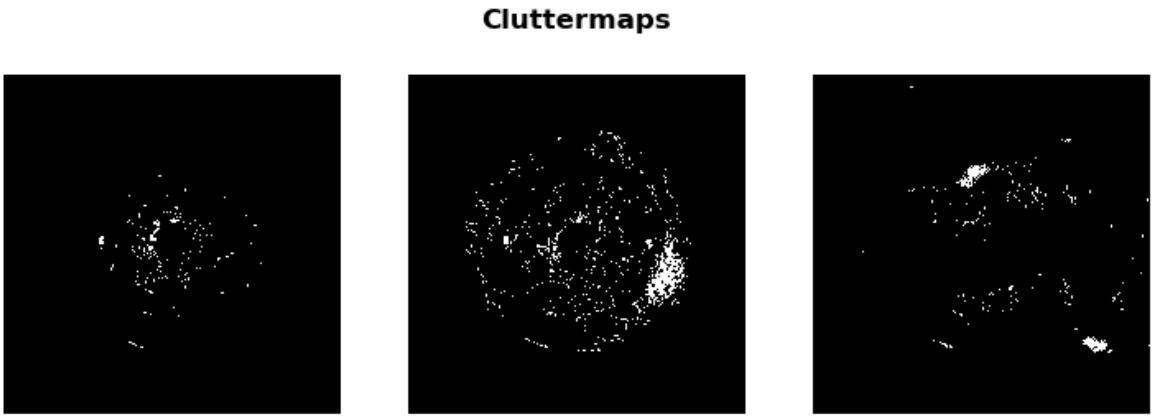


Fig. 42 Detección de clutter usando el filtro Gabella en los casos de estudio (Solo Clutter, Precipitación tipo 1 y Precipitación tipo 2).

El ultimo filtro para detección consiste en la elaboración de un mapa de clutter. Este se basa en la detección de clutter a través de un análisis estadístico. Para ello, se recolectan datos de radar libres de precipitación. A través del estudio de las zonas recurrentes en los datos, se marcan las zonas como clutter. En ese sentido, el cluttermap corresponde más a una idea que a un procedimiento estándar y debido a la selección de los datos puede ser considerado algo arbitrario.

La primera forma de elaboración de un cluttermap consiste en una selección manual de las zonas por parte de una persona adiestrada. Este procedimiento, que puede parecer banal, fue uno de los primeros procedimientos para mitigar la presencia de clutter. Sin embargo, este método no solo es tedioso, sino que está sujeto a diferencias dependiendo de las personas que lo llevan a cabo.

Desde entonces, la forma más sencilla de elaboración de un cluttermap consiste en los cluttermap estáticos. A través de un proceso de agregación estadístico (mínimo, mediana, moda, cuartiles, conteo, etc.) se crea una distribución de clutter representativa. Luego, se aplica un umbral que distingue la presencia de clutter. Por defecto, el umbral corresponderá al valor de reflectividad mínimo detectado. Sin embargo, este umbral puede ser mayor y utilizarse el valor mínimo que corresponde a precipitación detectable.

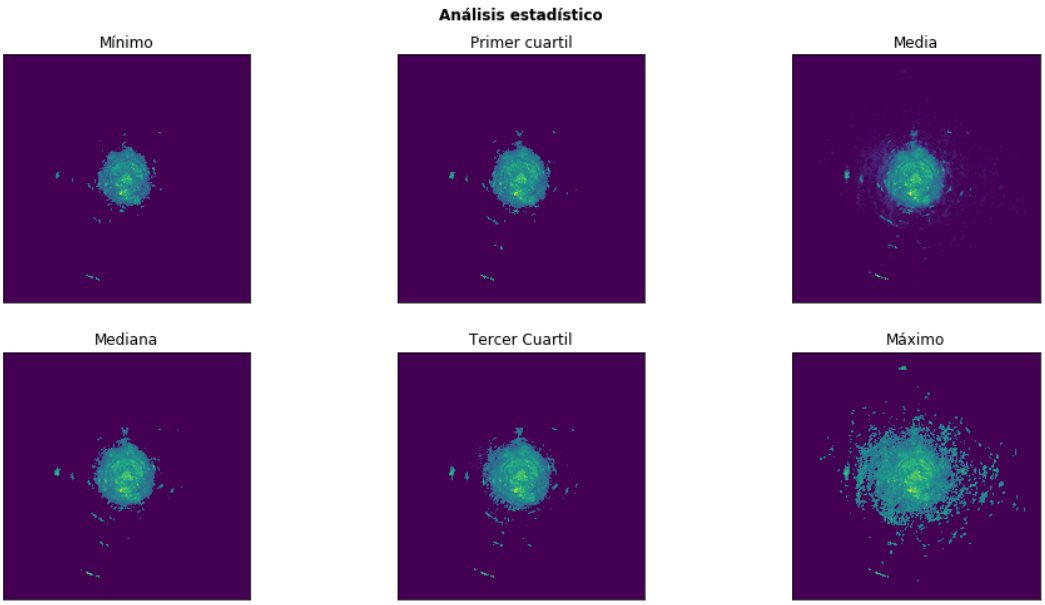


Fig. 43 Análisis estadístico a partir de las muestras recolectadas.

Existe una gran acumulación de clutter en las cercanías del radar (Figura 43). El primer planteamiento probablemente sea eliminar esa zona de los datos que reporta el radar. Sin embargo, los datos del radar PIUXX responden a las necesidades de la ciudad de Piura. En ese sentido, la ciudad de Piura es la primera interesada en esos datos. Es ahí donde radica la importancia de recuperar esa zona.

Los resultados muestran que a veces aparece clutter en mayor proporción. Este fenómeno es raro, pues la diferencia entre los datos mínimos y los correspondientes al tercer cuartil son prácticamente idénticos. Ello indica, que aunque no comúnmente, el clutter puede extenderse a otras zonas cercanas.

La Figura 44 muestra los cluttermap correspondientes luego de una binarización. Usar los datos de los cuartiles y el minimo no hace gran diferencia. Sin embargo, debido a la presencia no persistente de clutter los gráficos de media y maximo registran muchos lugares adicionales como clutter. Por una parte, ser conservadores y utilizar el cluttermap minimo no eliminará todo el clutter y debería ser usado en conjunto a otros métodos.

Los cluttermap estaticos son una primera alternativa automatica para la detección de clutter. Sin embargo, existe un fuerte inconveniente para su utilización. Al ser de naturaleza estatica, no están preparados para recibir variaciones en la distribución del clutter. Particularmente, presentan un pésimo desempeño cuando la precipitación se encuentra en las mismas zonas donde se suele presentar el clutter.

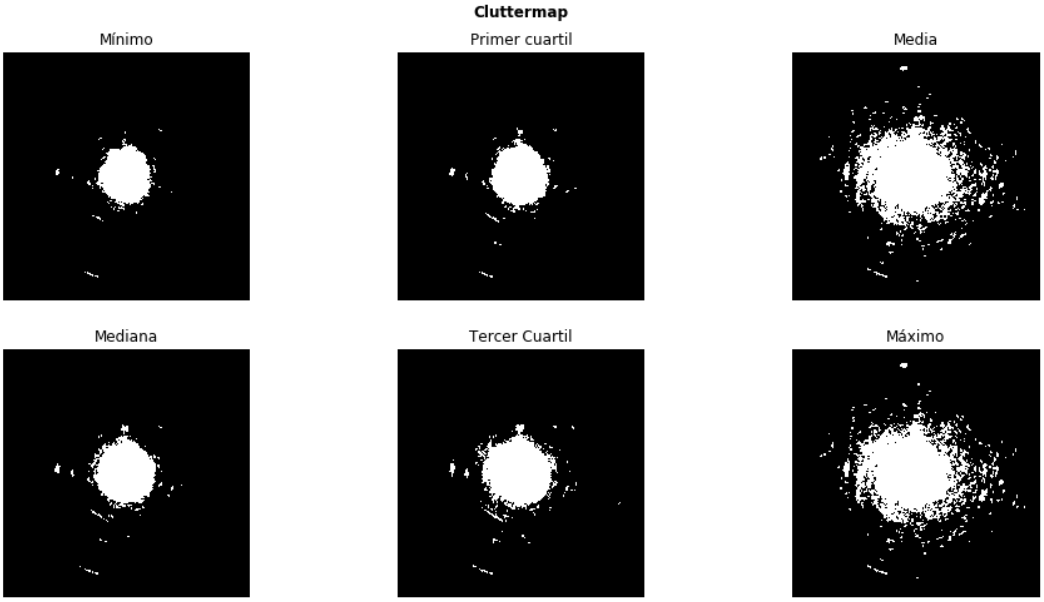


Fig. 44 Cluttermap estáticos generados.

En respuesta a la superposición de precipitación en las zonas de clutter se decidió elaborar cluttermap dinámicos. A diferencia de los cluttermap estáticos, estos dependen de la data de entrada. Por ello, es posible que se obtengan mejores resultados para esta clase de detección.

Se programó un primer algoritmo sencillo al que se denominó **limits**. Este algoritmo toma como idea base la misma premisa que los cluttermap estáticos: la distribución del clutter. A diferencia de estos, que consideran el clutter fijo, el algoritmo incluye la distribución del clutter. Para ello detecta los limites usuales de los valores de reflectividad del clutter. Si los datos se encuentran dentro del intervalo, se consideran como clutter. De lo contrario, se considera que pertenecen a precipitación.

El procedimiento **limits** es de hecho muy simple. No obstante, no debe asociarse su simplicidad con su eficacia para la remoción del clutter. La Figura 45 muestra que este simple algoritmo proporciona resultados incluso mejores que el filtro Gabella con los casos de estudio. A diferencia de los cluttermap estáticos, presenta un comportamiento acorde a los datos de entrada. Por ello, facilita la recuperación de los datos de precipitación en la región donde se ubica el clutter normalmente. Sin embargo, no presenta un comportamiento continuo en el espacio. Para solucionar ese problema, el algoritmo debe ser complementado con otros algoritmos que eliminen las discontinuidades.

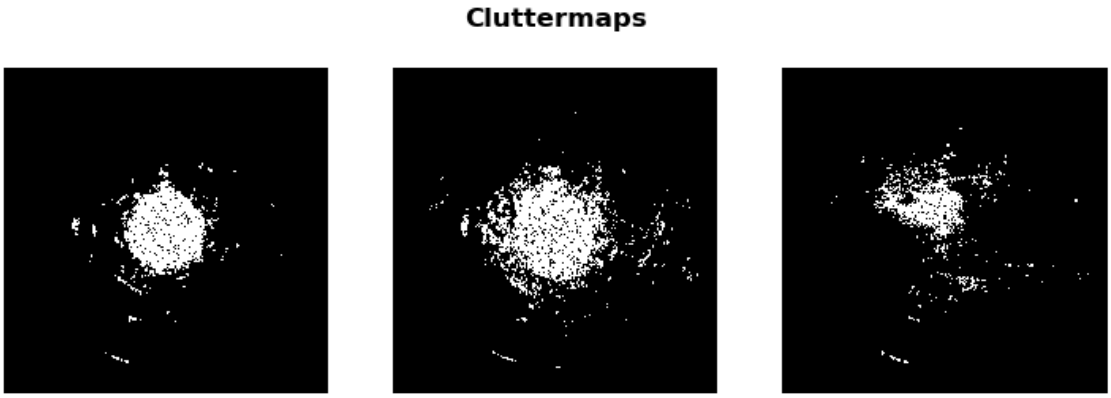


Fig. 45 Cluttermap elaborado con la función **limits** para cada caso de estudio (Solo Clutter, Precipitación tipo 1 y Precipitación tipo 2).

Dada la persistencia del clutter, otra alternativa viable consiste en aprovechar la secuencia temporal. En lugar de condensar la información usando estadística, es posible condensarla en dominio de la frecuencia. Se desarrolló el algoritmo **fourier** basado en esa idea. El nombre proviene del uso de la transformada de Fourier para el cambio de dominio. El algoritmo mide los cambios entre el conjunto de datos fuera de condiciones de precipitación y el mismo al agregar el dato a filtrar. Los cambios en el dominio de la frecuencia son claros indicativos de clutter.

La Figura 46 muestra los resultados al realizar la detección de clutter sobre los casos de estudio. En estos se aprecia un comportamiento intermedio entre los cluttermap estáticos antes vistos y el cluttermap dinámico. Si bien este procedimiento por su descripción podría parecer muy prometedor, las configuración del umbral para la discriminación entre clutter y precipitación coloca trabas a su implementación efectiva.

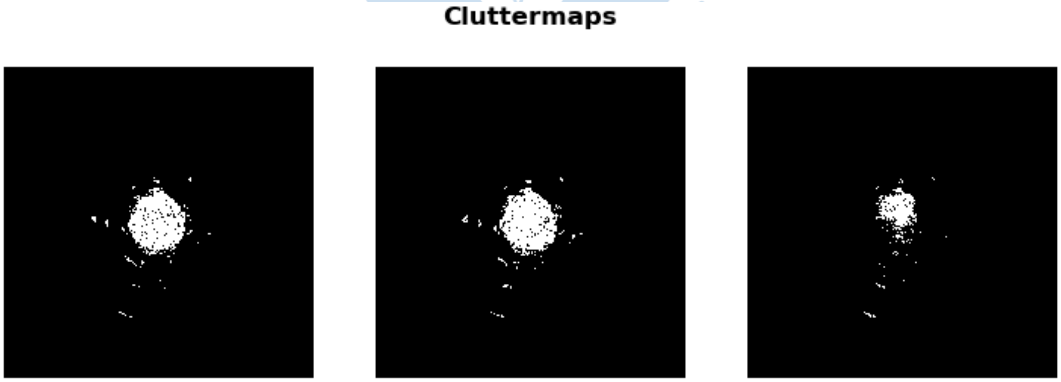


Fig. 46 Cluttermap elaborado en base a la función **fourier** para cada caso de estudio (Solo Clutter, Precipitación tipo 1 y Precipitación tipo 2).

Otro de los procedimientos desarrollados corresponde a la función **fourier2**. Como su nombre indica, la función sigue un procedimiento similar a la función **fourier**. La diferencia radica en la configuración del umbral, que en la función **fourier2** se lleva a cabo en el dominio temporal. Por ello, la selección del umbral resulta más sencilla al usuario.

Se evaluaron otras opciones como la elaboración de un filtro basado en clasificación. En este se determinaba si los datos correspondían solo a clutter o contenían precipitación. De clasificarse como solo clutter, el algoritmo proporciona una matriz compuesta por valores de reflectividad mínima. En caso contrario, no se realiza ninguna acción, con lo cual se cede la detección a otro algoritmo.

Los algoritmos mencionados detectan el clutter. Por si solo esto no es suficiente. Una vez detectado el clutter, existen varias opciones para su corrección. La primera de ellas consiste en interpolar los datos marcados a partir de sus vecinos en el mismo escaneo PPI. Para que la interpolación sea efectiva se requieren dos condiciones: que la zona a interpolar no sea muy grande. Si eso sucede, el patrón de los datos a interpolar puede no corresponder con el de la zona a interpolar originando inconsistencias. La otra condición requerida es una eliminación agresiva del clutter. Si en los datos quedase clutter residual, es posible que la interpolación expanda el rango de acción de ese clutter ocasionando el efecto contrario al que se busca.

El algoritmo de interpolación más sencillo que proporciona Wradlib es la interpolación del Vecino Más Cercano. Tal y como su nombre dice, la interpolación asigna el valor del dato o vecino más cercano. Este método de interpolación es uno de las más sencillos y los más rápidos. Sin embargo, es también el que más se ve afectado en caso de que las condiciones mencionadas para interpolación se incumplen. La Figura 47 muestra la aplicación de este método de interpolación usando un cluttermap estático basado en la mediana de los datos. Se observa que la interpolación no ha corregido de manera efectiva el clutter, especialmente entre más precipitación permanece. Para la precipitación, este método no muestra un patrón suave sobre los datos interpolados. Por ello, se desaconseja su uso.

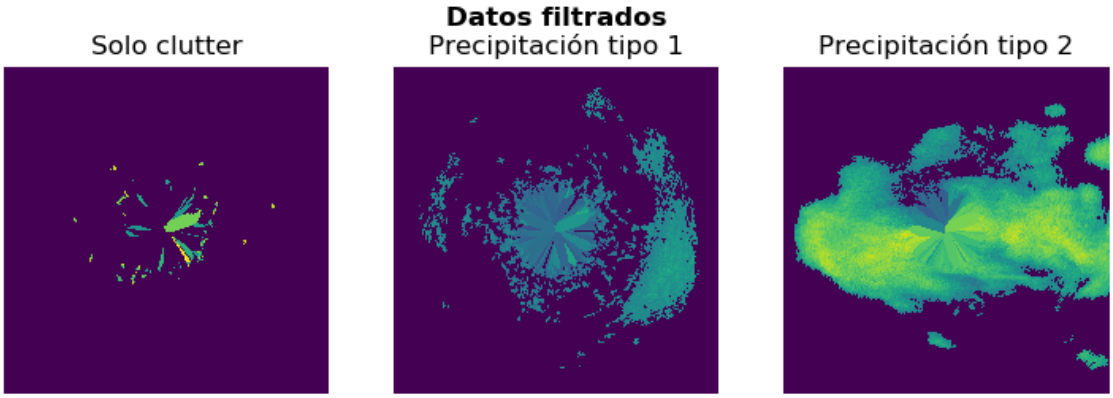


Fig. 47 Interpolación del Vecino Más Cercano usando el cluttermap de mediana.

Otros métodos de interpolación están disponibles como la interpolación Distancia Inversa Ponderada (Idw) o el método de interpolación por Krigging. Ambos representan mejores estimaciones al ideal de patrón a obtener. No obstante, estos métodos utilizan una carga computacional más elevada y están sujetos a las mismas dos condiciones mencionadas

antes. La Figura 48 muestra que si bien los patrones son más suaves usando un mecanismo de interpolación más avanzado, es necesario capturar la mayor cantidad de clutter para que los resultados sean aceptables.

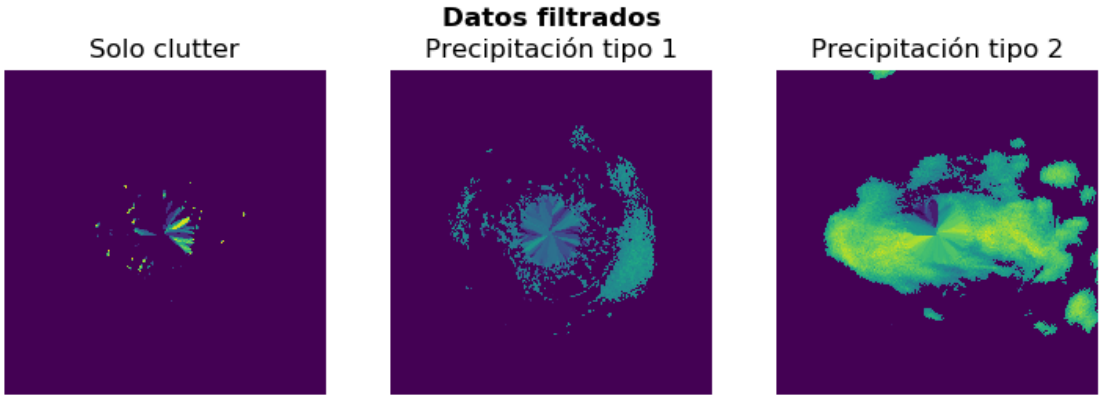


Fig. 48 Interpolación Idw usando el cluttermap de mediana.

Note que tanto la Figura 47 como la Figura 48 han sido elaboradas a partir de un filtro estático un tanto conservador para mostrar el efecto. Para evitar cualquier tipo de suspicacias, se muestra el resultado usando el cluttermap correspondiente al máximo usando la interpolación del Vecino Más cercano en la Figura 49. Debido a que corresponde a la máxima distribución de clutter registrada, los resultados para la condición de Solo Clutter son los mejores. No es ese el caso de los otros dos ejemplos, pues el efecto de la interpolación sigue apareciendo. De esta forma se demuestra la importancia de las condiciones o de limitar el alcance de la interpolación a zonas pequeñas.

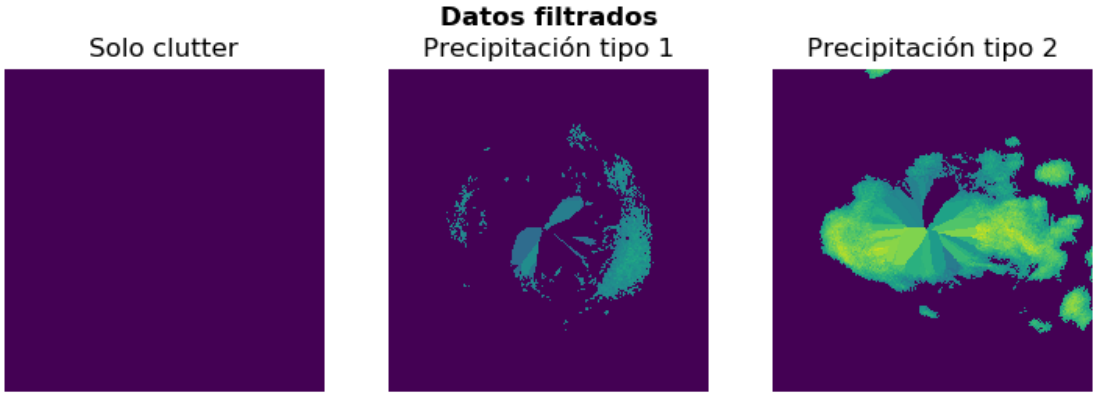


Fig. 49 Interpolación del Vecino Más Cercano usando el cluttermap máximo.

La otra posibilidad consiste en extrapolar los datos a partir de escaneos con ángulos de elevación superiores. Sin embargo, el radar PIUXX solo realiza escaneos con un solo ángulo de elevación, por lo cual, este procedimiento no es aplicable.

Las ultimas posibilidades consisten en la sustitución de los valores marcados por No Data u otro valor. Este procedimiento es útil cuando el clutter y la precipitación no se encuentran en la misma zona. En ese caso puede reemplazarse el clutter por el valor de

reflectividad mínimo. Sin embargo, si se encuentran mezclados, los resultados rompen con el patrón de precipitación y no representan una buena estimación (Figura 50).

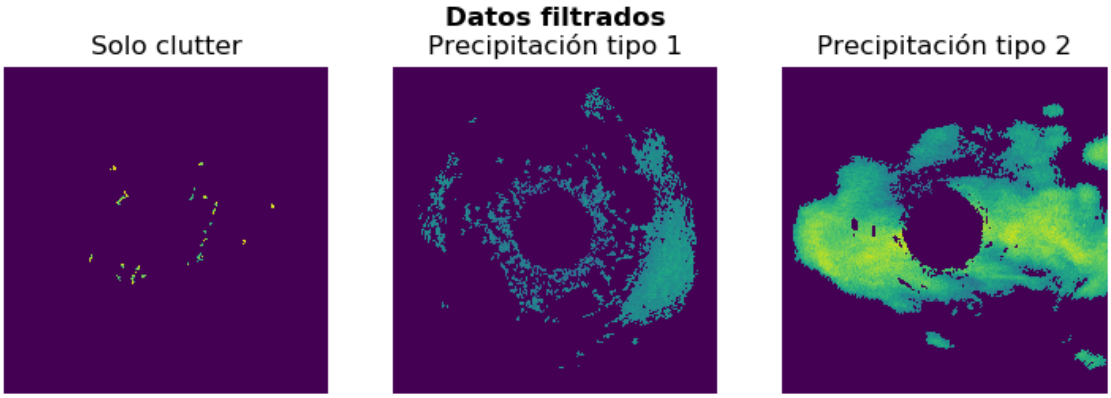


Fig. 50 Sustitución a partir del cluttermap estático de mediana.

Para complementar estos procedimientos se programaron 3 filtros basados en la continuidad espacial. En cierta forma, resultan parecidos al filtro speckle mencionado en capítulos anteriores. Sin embargo, todos ellos están basados en el Procesamiento Digital de Imágenes y en los conceptos de vecindad y conectividad.

El primero de ellos es **filter_mask**. Este trabaja sobre la imagen binaria de los datos. Utiliza un Kernel para realizar un filtrado convolucional y generar una máscara. Al aplicar un umbral a los datos se generan los cluttermap correspondientes. Sin embargo, este filtro requiere conocimientos de Procesamiento Digital de Imágenes para escoger adecuadamente los Kernel y los umbrales.

El segundo de estos filtros se llama **filter_conectivity**. Como su nombre indica se basa en la conectividad. Al igual que el anterior trabaja sobre la imagen binaria. A través de un proceso de Segmentación y Etiquetado se agrupan todos los datos conectados. Aquellos con un tamaño en pixeles menor al umbral escogido son marcados como speckles para el filtrado.

El tercer filtro se denomina **filter_area**. Es muy similar al anterior. Como los datos corresponden a coordenadas polares, los datos más cercanos al radar corresponden a menor área que aquellos más alejados. Es por ello por lo que este filtro se basa en la detección por área en lugar de tamaño relativo como el filtro anterior.

Por último, también fueron programados los filtros correspondientes a la cadena de procesamiento original. Detalles y comparaciones de todos estos procedimientos se pueden encontrar en los cuadernos correspondientes (Apéndice 5).

6.5 Corrección de atenuación

Las correcciones por atenuación aplicables corresponden a enfoques puerta-puerta basados en calcular la atenuación a lo largo de un haz (PIA). Como se mencionó antes, estos enfoques son inestables y aplicar la corrección puede ser peor que no hacerlo. Por ello, los

algoritmos de corrección puerta-puerta actuales incluyen restricciones en el valor máximo del PIA.

Dado el carácter exponencial de las relaciones Z-R, debe tenerse en cuenta que el aumento en dBZ producto de la atenuación debe ser pequeño. Como ejemplo, un aumento de 5 dBZ corresponde en promedio a factor de 2 en la tasa de precipitación. Sin embargo, las restricciones no toman en cuenta el valor base. De esta forma, el aumento de 5 dBZ mencionado podría corresponder a una corrección de 2 mm/h a 4 mm/h como también a una corrección de 20 mm/h a 40 mm/h.

En resumen, la corrección de atenuación a lo largo de un haz es procedimiento que debe usarse con cuidado. Dado que el cálculo del PIA depende de los datos, es necesario colocar restricciones para que ninguno de los problemas mencionados se presente.

Los principales métodos basados en el cálculo del PIA se encuentran programados en la librería Wradlib. Py-ART, en cambio, solo ofrece métodos de corrección de atenuación basados en parámetros polarimétricos.

6.6 Conversión a tasa de precipitación

La reflectividad (Z) es solo una medida indirecta de la precipitación (R). Ya con todas las correcciones, los datos se deben convertir a tasa de precipitación. Las relaciones Z-R vistas nos permiten realizar este paso con facilidad. Sin embargo, la relación a utilizar debe ser calibrada a partir del ajuste de los datos comparándolos con mediciones pluviométricas. Dado que la cantidad de datos no es suficiente como para efectuar una comparación, se sugiere usar por defecto la ecuación de Marshall-Palmer ($Z = 200R^{1.6}$).

El radar proporciona valores de reflectividad entre -31.5 y 95.5 dBZ. Dada la transformación exponencial de las relaciones Z-R, las relaciones proporcionaran sobrestimaciones para valores altos de reflectividad. La Figura 51 muestra que las relaciones Z-R proporcionan valores de tasas de precipitación absurdos en límite de la escala. Esto se debe a dos factores distintos. Por una parte, es muy difícil que se registre valores de reflectividad tan elevados. Por otro lado, la relación Z-R calibrada es confiable hasta valores de reflectividad cercanos a 55dBZ. Valores superiores suelen asociarse a granizo. Para este tipo de precipitación, la reflectividad no está asociada directamente a la intensidad. Por ello, en caso de presentarse valores muy altos de reflectividad, se debe tener cuidado al analizarlos y comunicarlos.

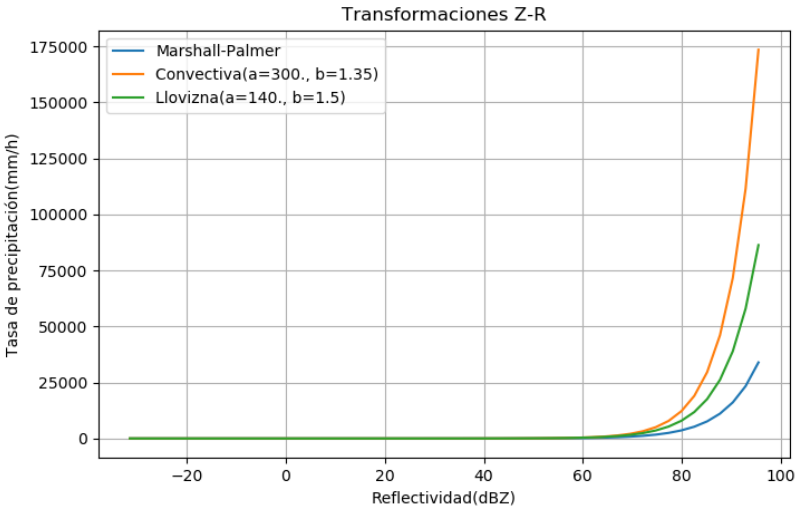


Fig. 51 Relaciones Z-R en el rango de reflectividad del radar PIUXX

La Figura 52 muestra la escala adaptada para el régimen usual de reflectividad. Note que, dado el carácter exponencial de la relación Z-R, puede resultar engañoso un pequeño aumento de la reflectividad. Por ello, no se recomienda mostrar al público los valores de reflectividad directamente, a no ser con carácter cualitativo.

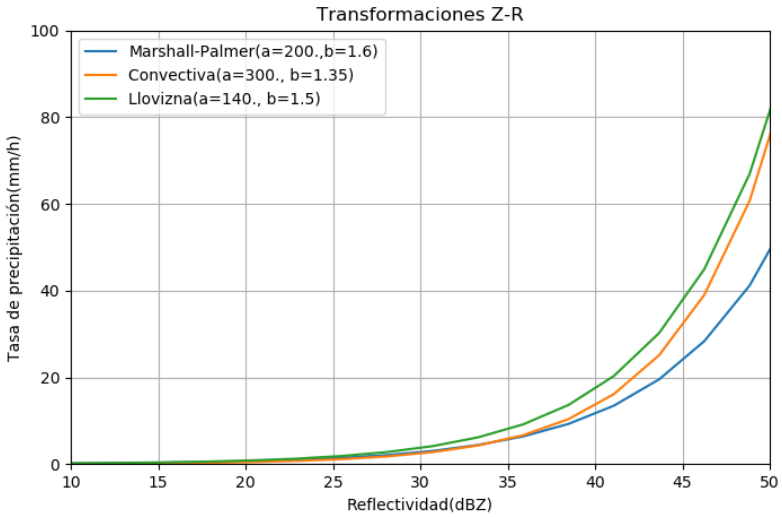


Fig. 52 Detalles de la transformación Z-R

La relación Z-R tiene carácter exponencial. Por ello, nunca proporcionará un valor de cero en la intensidad o tasa de precipitación. Por tanto, se recomienda utilizar un umbral mínimo a partir del cual se considera precipitación efectiva. Dado que la red pluviométrica detecta como mínimo 0.1 mm/h y valores menores se clasifican como Trazas, se recomienda utilizar este valor como umbral. La Figura 53 muestra que se puede despreciar todos aquellos valores menores a 5 dBZ.

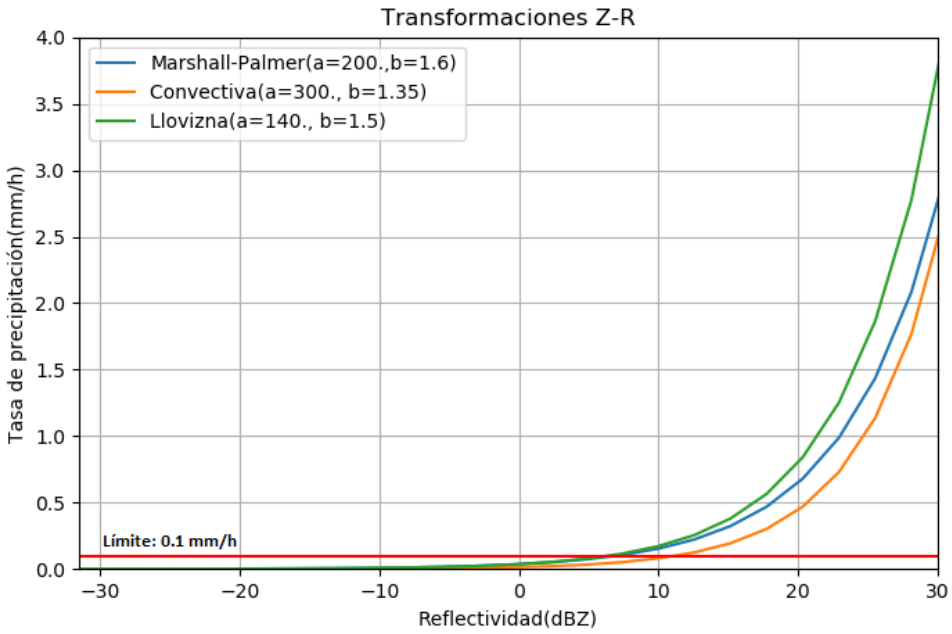


Fig. 53 Umbral de detección propuesto

La calibración de la relación Z-R consiste en un ajuste de los datos de reflectividad para asemejarse a las mediciones obtenidas por los pluviómetros. En ese sentido, no debe ser vista más que como un ajuste importante. Por ello, al mismo tiempo que se calibra o se determina está relación, debe cuantificarse el grado de incertidumbre asociada a la estimación cuantitativa de precipitación.

6.7 Acumulación

Ya sea de manera informativa o por aplicaciones hidrológicas, es necesario pasar de los datos de tasa de precipitación hacia la precipitación acumulada durante un periodo de tiempo. Junto con la tasa máxima de precipitación, corresponden a los valores que la ciudadanía espera conocer de un evento de precipitación.

Dada el formato digital de los datos, estos corresponden a capturas sucesivas de la atmosfera cercana. La forma más básica de obtener la precipitación acumulada consiste en considerar los datos como constantes durante el intervalo de tiempo entre ellos (resolución temporal).

$$A = t \times \sum R_i \qquad \textbf{Ec. 23}$$

Donde A representa la precipitación acumulada, t la resolución temporal en segundos y R corresponde a los datos de tasa de precipitación.

La precipitación es un fenómeno de naturaleza dinámica. Por ello, cualquier consideración estática tendrá errores inherentes. Los errores serán más frecuentes entre más cambie el campo de precipitación durante un intervalo. Esto sucede especialmente durante la formación de precipitación convectiva, que puede desplazarse de manera rápida y presenta

poco tiempo de vida. Cuando eso sucede, los datos acumulados presentan un aspecto extraño, que no corresponde al patrón real de precipitación. Esto no solo afecta los datos de manera visual, sino que proporciona zonas vacías entre las cuales no se muestra que exista precipitación.

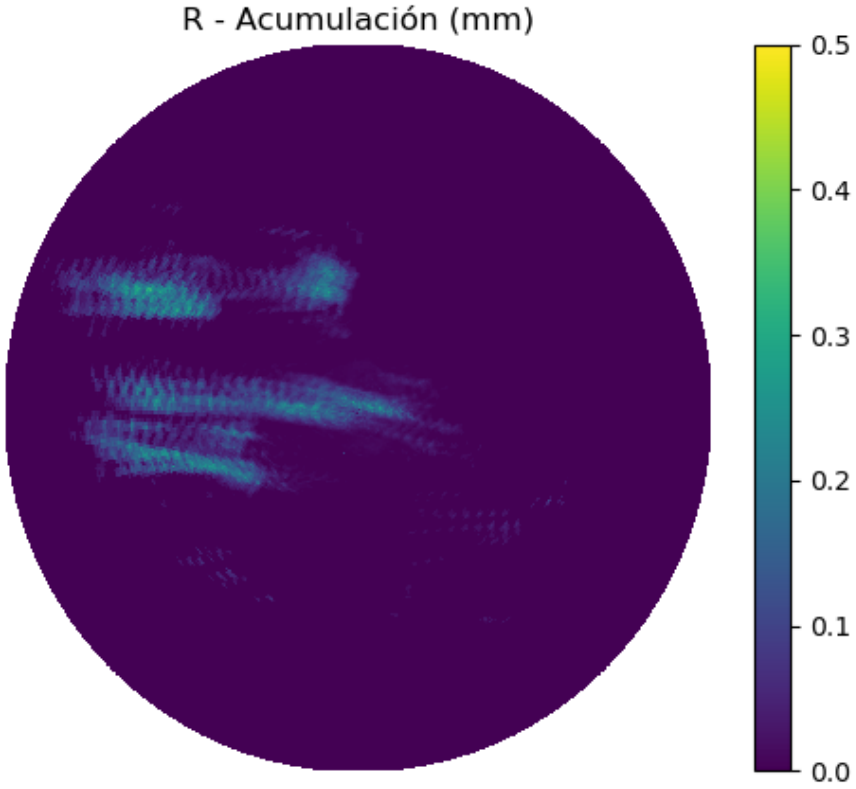


Fig. 54 Acumulación simple.

La Figura 54 muestra la acumulación usando la ecuación anterior sobre los datos de una hora de una pequeña llovizna sobre la ciudad de Piura. Note el extraño patrón que presentan como consecuencia de la resolución temporal.

Una mejor estimación de la precipitación acumulada consiste en considerar el promedio de los datos obtenidos en el intervalo. Este enfoque, al tomar ambos extremos, representa una mejor aproximación. Sin embargo, el resultado final dependerá de la resolución temporal de los datos y la velocidad que presenten la precipitación. Para los datos mencionados, no se obtuvo una mejora significativa de los resultados.

$$A = t \times \sum 0.5\left(R_i + R_{i+1}\right) \qquad \textbf{Ec. 24}$$

El ultimo enfoque consiste en considerar el proceso de advección de la precipitación. En este enfoque se determina el campo de velocidad que presentan los datos. A partir del campo de velocidad se crean datos interpolados con un tiempo de muestreo mucho menor. La precipitación acumulada se calcula a partir de los datos interpolados. Este enfoque

proporciona resultados que se aproximan mucho más a la verdadera distribución de la precipitación.

La acumulación por advección no es un enfoque que se encuentre disponible en las librerías más usuales. De acuerdo con lo investigado, SwirlsPy es la única librería que proporciona métodos para realizar este procedimiento. Sin embargo, esta librería solo está disponible para Agencias Meteorológicas Nacionales a pedido. Debe considerarse también, que como ha sido definido representa más una idea que un algoritmo programable. Por ello, existirán diferentes versiones y adaptaciones de esta.

La acumulación por advección como procedimiento, ha sido presentado al menos de tres maneras distintas. El algoritmo mencionado en (Pfaff, 2013) adapta el procedimiento para interpolar los datos de manera ponderada hacia atrás y hacia adelante. El algoritmo propuesto por (Racoma, Crisologo, & David, 2015) en cambio se basa en la estimación de la velocidad a través de la detección de las celdas de precipitación. También, se dispone del algoritmo que permite implementar Rainview Analyzer a partir de la combinación de los productos RTR y PAC.

El algoritmo propuesto por (Racoma et al., 2015) se basa en la detección de celdas de precipitación acumulada de dos horas. La estimación de la velocidad se produce al comparar las posiciones de los centroides de la tormenta en los datos sucesivos. A partir de la estimación de la velocidad se interpolan los datos con una resolución temporal menor. Dada la larga acumulación necesaria, no es aplicable para tormentas convectivas y de corta vida.

El algoritmo aplicado por RainView Analyzer se basa en la estimación del campo de velocidad a través de la coincidencia de bloques (Block Matching) usando correlación cruzada. Este algoritmo utiliza solo la predicción hacia adelante para aumentar la resolución temporal. Debido a ello, puede producir pobres estimaciones ante cambios en el patrón de velocidad. Sin embargo, representa un procedimiento superior al de la simple acumulación que utiliza el producto PAC.

El algoritmo de Acumulación por Advección (Pfaff, 2013) representa el más completo de los algoritmos, pues incluye los datos siguientes utilizando la interpolación hacia atrás. Por ello, se decidió programar este algoritmo. Los detalles de la implementación se comentan abajo.

Primero se estima el campo de velocidad a partir de un algoritmo de flujo óptico. A partir del campo, se utiliza un esquema de advección semi-Lagrangiana backward y forward para producir datos interpolados con una resolución temporal superior. Los datos se acumulan usando un enfoque ponderado para combinar los efectos de la advección hacia adelante y hacia atrás (Figura 55).

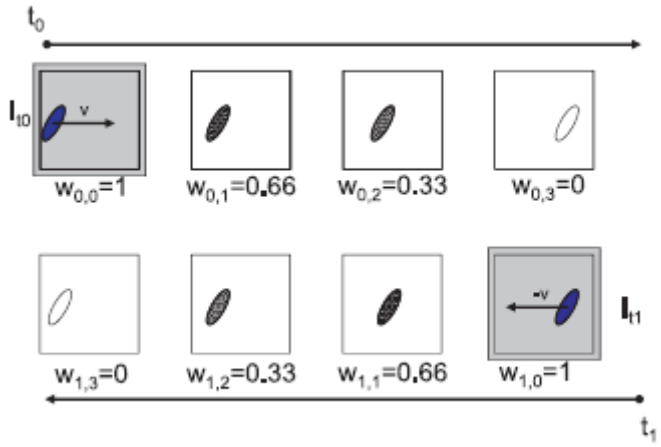


Fig. 55 Esquema de advección
Fuente: Pfaff (2013)

Para la programación de este enfoque se utilizó la librería Pysteps. A partir de su algoritmo de flujo óptico Lucas-Kanade se determinó el campo de velocidad. Posteriormente este se utilizó para generar datos interpolados con resolución de 30 segundos utilizando el método semi-lagrangiano. Esta interpolación fue realizada tanto hacia adelante con el dato que inicia cada intervalo, como hacia atrás con el dato que corresponde al inicio del próximo intervalo. Posteriormente se realizó la acumulación utilizando una suma ponderada de valores linealmente espaciados entre uno y cero.

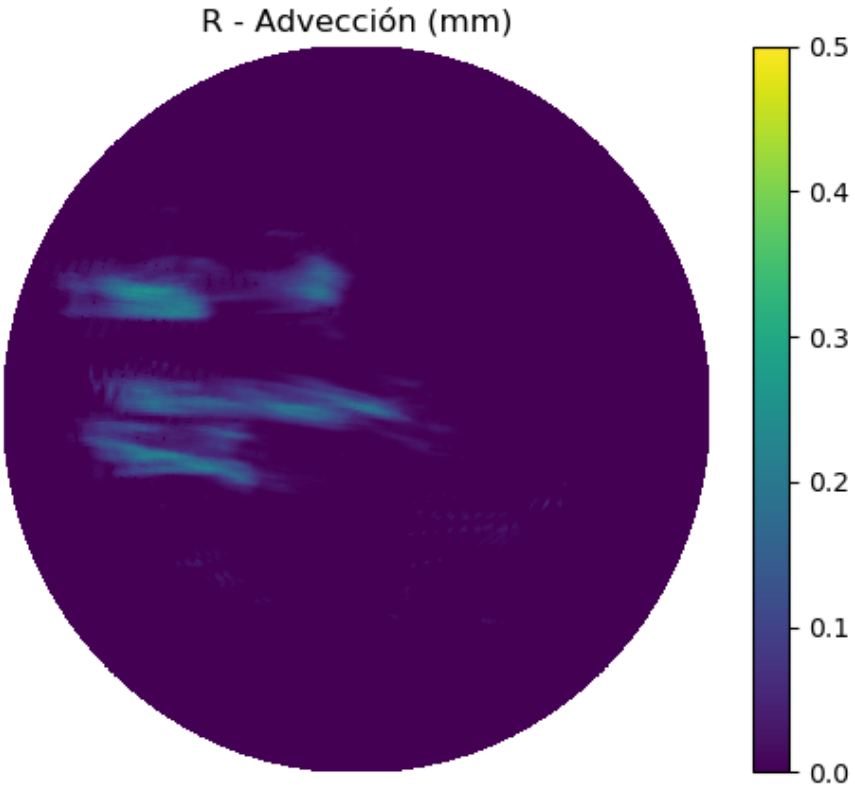


Fig. 56 Acumulación usando el método de advección descrito.

6.8 Proyección

Los datos aún se encuentran en coordenadas polares. Por ello, es necesario convertirlos a coordenadas cartesianas y/o proyectarlos. Este paso es fundamental para la distribución de los datos y para garantizar su aplicabilidad. Usualmente comprende uno de los últimos pasos para evitar la propagación de los errores que se producen al interpolar los datos en el proceso. El proceso de proyección y georreferenciación están dominados por tres parámetros: el sistema de proyección geográfica, la grilla de proyección y el método de interpolación.

Los sistemas de proyección geográfica recrean un modelo geográfico de la tierra diseñado para su representación en un mapa. Cada sistema esta optimizado para conservar mejor una propiedad (área, ángulo o distancia). Por tanto, la elección de uno u otro dependerá de la aplicación que se les dará a los datos. Py-ART utiliza una proyección basada en la librería PROJ. Wradlib, en cambio, se basa en la creación de objetos OSR de la librería GDAL. Wradlib también acepta la definición del sistema de proyección a partir del número EPSG que corresponden a una codificación estándar de los sistemas de proyección geográfica.

La grilla de proyección corresponde a la matriz de coordenadas sobre las cuales se proyectarán los datos. Usualmente es una matriz uniformemente distribuida. Sin embargo, a veces se utilizan grillas irregulares definidas según la forma de una cuenca de interés. Sea cual sea el caso, la grilla define la resolución espacial de los datos proyectados. A mayor resolución el patrón representado será más suave y ocupará más espacio. Por el contrario, para una menor resolución los datos ocuparán menos espacio a costa de perderse los detalles. Note que la resolución es artificial, y en principio, puede ser configurada a cualquier número. Eso no quiere decir que los datos puedan aumentar indefinidamente de resolución, pues la resolución autentica de los datos está ligada a la cuadricula polar original.

Por último, dado que los datos se están proyectando será necesario definir un método de interpolación. Como se vio en la sección referente al clutter, la elección de la interpolación puede conducir a diferentes resultados. Wradlib incluye la interpolación Vecino Más Cercano, Idw y Krigging. Py-ART en cambio, no dispone de algún parámetro configurable en sus funciones, por lo que la interpolación probablemente se lleve a cabo en alguna función de la librería PROJ.

Dado que estos procedimientos ya se encuentran implementados en las librerías, se decidió complementarlos con ejemplos usando los datos del radar PIUXX.

6.9 Predicción a corto plazo (nowcasting)

La predicción a corto plazo representa otra de las aplicaciones de los radares meteorológicos. Frente a los eventos de precipitación, es necesario estimar el desarrollo de las tormentas, especialmente en caso de lluvias intensas o en zonas fácilmente inundables.

Existen diversas estrategias para la predicción a corto plazo. La predicción a corto plazo tradicional se basa en la determinación del campo de velocidad y la extrapolación de los datos a partir de este (Figura 57). Las estrategias más básicas dividen los datos en una cuadricula y los correlacionan con los datos tomados al siguiente intervalo de tiempo para estimar el campo de velocidad. Otros procedimientos estiman el campo de velocidad a través de técnicas de Flujo Óptico. Sin embargo, estas estrategias solo funcionan bien para tiempos cortos pues no tienen en cuenta la evolución de la tormenta. Por ello, la investigación actual en esta área trata no solo de proporcionar una predicción, sino de cuantificar la incertidumbre asociada a través del nowcasting probabilístico.

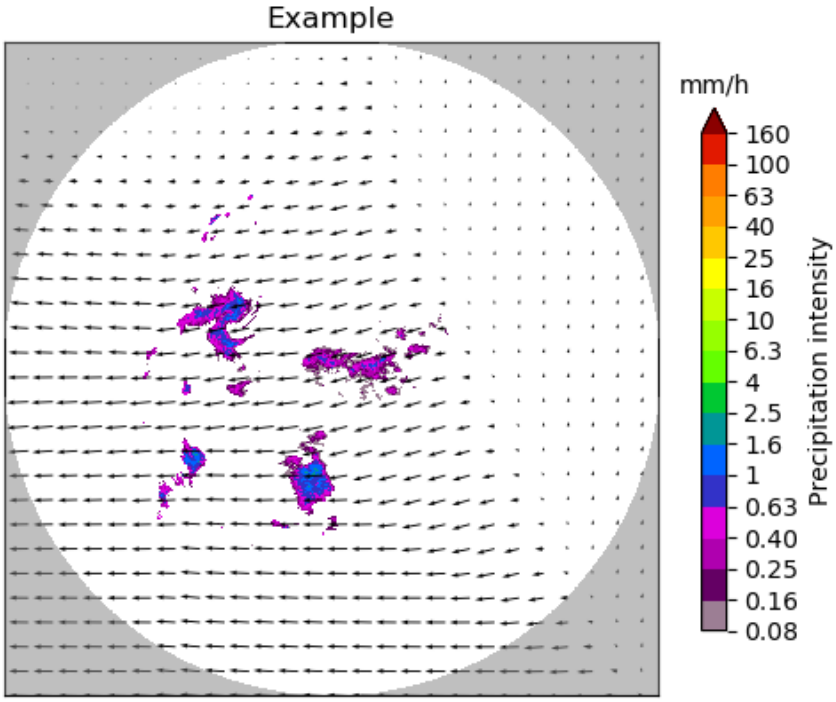


Fig. 57 Campo de velocidades

Aparte de los métodos clásicos y debido al auge de la Inteligencia Artificial, han surgido predicciones a corto plazo basadas en redes neuronales. La mayoría de estas predicciones basan su arquitectura en capas convolucionales y capas LSTM para aprovechar las características espaciales y temporales de los datos, respectivamente. Las capas convolucionales han sido usadas con muy buenos resultados para imágenes. Dado el formato de grilla de los datos, la analogía con las imágenes justifica el uso de este tipo de capas. Por otra parte, las capas LSTM (Long – Short Term Memory) han sido utilizadas para la predicción de series temporales y aplicadas para la predicción de movimientos en videos. Debido a ello, estas capas resultan la base de casi cualquier arquitectura encontrada en la literatura.

Rainymotion (Ayzel, Heistermann, & Winterrath, 2018) y Pysteps (Pulkkinen et al., 2019) son las dos librerías que abordan la predicción a corto plazo. Ambas han sido desarrolladas durante los dos últimos años. Por su parte, Rainymotion proporciona modelos

de referencia para comparación con modelos en desarrollo. En ese sentido corresponde más a un módulo de referencia que a una librería propiamente. Pysteps, en cambio, ha sido desarrollada para el nowcasting probabilístico. Su desarrollo incluye una gran variedad de algoritmos. A diferencia de Rainymotion, Pysteps está organizada propiamente como una librería. De cualquier forma, ambas proporcionan un excelente punto de inicio para la predicción a corto plazo.

Rainymotion cuenta con 5 modelos agrupados en tres familias. El primero de ellos consiste en la Persistencia Euleriana, que mantiene el último dato en el siguiente intervalo de tiempo. Este modelo que puede parecer trivial proporciona muy buenos resultados para tiempos muy cortos y tormentas de dinámicas lentas. Los otros dos modelos pertenecen a la familia “Sparse”. Estos se basan en la detección de características entre datos sucesivos para crear una matriz de extrapolación. Los últimos dos modelos pertenecen a la clase “Dense”. Estos se basan en la estimación del campo de velocidad a través del Flujo Óptico.

Pysteps está centrada en proporcionar diversas opciones para el pronóstico a corto plazo y especialmente el nowcasting probabilístico. Pysteps está dividido en 11 módulos (Tabla 10).

Tabla 10 Módulos de Pysteps

Módulo	Propósito
pysteps.cascade	Descomposición en diferentes escalas espaciales
pysteps.extrapolation	Extrapolación del campo de velocidades
pysteps.io	Entrada y salida de formatos estándar
pysteps.motion	Estimación del campo de velocidades
pysteps.noise	Generación de ruido
pysteps.nowcasts	Implementación de diferentes métodos de nowcasting
pysteps.postprocessing	Estadísticas y ajustes de distribuciones de probabilidad
pysteps.timeseries	Autocorrelación temporal
pysteps.utils	Herramientas de preprocesamiento
pysteps.verification	Métricas de verificación
pysteps.visualization	Visualización

Dado que ya existían librerías que abordaban la predicción a corto plazo, se decidió por crear ejemplos sobre los datos del radar PIUXX. Esto es especialmente significativo para el caso de la librería Rainymotion que no posee ningún tutorial o ejemplo propio.

6.10 Comparativo entre la cadena de procesamiento original y propuesta

Tal y como se ha mostrado a lo largo de este capítulo, la principal finalidad de la cadena de procesamiento alternativo es poder proporcionar las mismas opciones de procesamiento

que la cadena original con la ventaja añadida que el procesamiento se puede realizar en un software libre; eliminando por tanto la dependencia del software incorporado en el radar.

Sin embargo, la cadena propuesta añade algoritmos propios y otros recogidos de la literatura a fin de eliminar los problemas y errores presentados en los datos y que ya se han detallado en el Capítulo 3.

Para facilitar una comparación entre los algoritmos que conforman ambas cadenas de procesamiento se presenta la siguiente tabla.

Tabla 11 Comparativo entre la cadena original y la cadena propuesta

Proceso	Cadena original	Cadena propuesta	Comentario
Visualización	RainScoutMet	Wradchibi	Versión en software libre
Corrección de apantallamiento	OCC	Algoritmo basado en librería Wradlib	El análisis de los datos arrojo que no era necesario
Eliminación de clutter	**Detección:** Cluttermap basado en agregación estadistica **Eliminación:** Flagging Subtraction	**Detección:** Filtro histo_cut Filtro Gabella Cluttermap basado en agregación estadística Limits Fourier Fourier2 **Eliminación:** Flagging Subtraction Interpolation	Se añadieron métodos de detección de clutter basados en correlaciones temporales de los datos. Se añadió un método de eliminación de clutter por interpolación.
Eliminación speckles	-	Filter_mask Filter_conectivity Filter_area	Se añadieron filtros para la eliminación de motas basados en el Procesamiento Digital de Imágenes.
Corrección de atenuación	-	Algoritmos PIA	Se investigaron los algoritmos aplicables. Faltan datos para una evaluación completa.

Conversión a tasa de precipitación	RR	Relaciones ZR	Versión en software libre
Acumulación	Acumulación simple	Acumulación simple Acumulación + advección	Se implementó un algoritmo de acumulación superior
Nowcasting	RTR (Algoritmo basado en Block Matching)	Algoritmo basado en librería Py-Steps y Rainymotion	

UNIVERSITAS STUDIORUM PIURENSIS

Conclusiones

Se ha estudiado y analizado la cadena de procesamiento del radar de lluvias PIUXX, instalado en la Estación Científica Ramón Múgica de la Universidad de Piura. Esta cadena de procesamiento es la que provee el fabricante del radar y en la cual el usuario tiene pocas opciones para configuración.

Los datos crudos del radar requieren correcciones. A través de los análisis realizados, se determinó que los datos presentan problemas de clutter, atenuación y falta de calibración. El clutter presente en los datos se ubica en las cercanías del radar, específicamente dentro de los primeros 20 km a la redonda. El clutter es de tipo terrestre y está causado en su mayoría por objetos artificiales como edificios, torres y grandes tanques de agua. Por otro lado, la atenuación es un problema a tener en cuenta debido al tipo de banda en la cual trabaja el radar. De todas las bandas de frecuencia utilizadas por los radares meteorológicos la banda X corresponde a la banda de frecuencias con mayores problemas de atenuación. Adicionalmente, la falta de una relación Z-R calibrada limita la utilidad de los datos para la estimación cuantitativa de la precipitación. Por último, no se encontraron problemas de apantallamiento para el ángulo de elevación utilizado.

Además de las correcciones que los datos requieren por la ubicación del radar y las condiciones atmosféricas subyacentes, otros aspectos como el sistema polar y la visualización obligan a incluir pasos adicionales para el aprovechamiento efectivo de los datos.

La cadena de procesamiento predeterminada resulta fácil de configurar y potente en resolver los problemas que aborda pese a la sencillez de sus algoritmos. Desafortunadamente, el preprocesamiento disponible solo cubre los problemas de clutter y apantallamiento. La atenuación no es cubierta por la cadena de procesamiento predeterminada. El resto de las correcciones, como la proyección y la conversión a formatos estándar, se encuentran plenamente cubiertas.

Se delinearon los procedimientos para implementar una cadena de procesamiento alternativa utilizando Python. Esta comprende herramientas basadas en librerías existentes y ampliamente usadas así como algoritmos y herramientas diseñadas para afrontar problemas específicos de la cadena de procesamiento predeterminada. Al estar basada en software libre, se pueden incluir correcciones adicionales a medida que se detecten problemas en los datos.

La cadena propuesta representa una mejora en aspectos como visualización de los datos crudos, detección de clutter y predicción a corto plazo. Esta cadena de procesamiento alternativa se describe en el desarrollo de la tesis, mientras que en los apéndices se incluye los códigos que permiten su implementación así como una lista de ejemplos.

Recomendaciones

Los desarrollos de software mencionados no poseen carácter definitivo. Al tratarse de software libre, muchos de ellos evolucionarán e incluirán nuevos algoritmos y herramientas. En ese sentido, se recomienda tomar la información mencionada sobre los mismos solo como referencia. Además, el hecho que puedan ser estudiados y modificados se presenta como una magnífica oportunidad para formar vínculos de cooperación con otras universidades e institutos que trabajan con radares meteorológicos.

El trabajo realizado puede ser visto como completo o incompleto en dos sentidos distintos. Por una parte, se han abordado la mayor parte de las correcciones disponibles y los productos que se pueden generar. Por otra, debido a la reciente adquisición y la falta de lluvias, en muchos de los aspectos cubiertos no se han realizado análisis comparando los resultados con información pluviométrica. A partir de los datos que se recolecten en el periodo lluvioso, se recomienda complementar los lineamientos con los análisis correspondientes, especialmente la calibración de la relación Z-R.

Los aspectos comentados en la cadena de procesamiento son en su mayoría de carácter hidrológico. Es necesario complementarlos con aquellos necesarios para el monitoreo. Hace falta automatizar la gestión de la información. En ese sentido, se requiere discriminar los datos que corresponden a precipitación y determinar los lugares (provincias o pueblos) en los cuales se presenta la lluvia. Esta información permitirá realizar alertas o proporcionar datos útiles en tiempo real a la población.

Buena parte del trabajo ha consistido en transformar los datos crudos en productos útiles. No es posible adjudicarles el término “útiles” a los datos si no hay personas que los usen. Se recomienda entonces distribuir los datos sin mayores restricciones. Por sí solo, esto tal vez no sea suficiente. Sin una adecuada promoción e instrucción, los datos no serán

utilizados. Todos estos aspectos deberán ser cubiertos para garantizar que realmente el radar adquirido beneficie realmente a la ciudad de Piura y sus habitantes.

Referencias bibliográficas

Andersson, T. (2000). *Using the sun to check some weather radar parameters*. Norrköping.

Ayzel, G., Heistermann, M., & Winterrath, T. (2018). Optical flow models as an open benchmark for radar-based. *Geoscientific Model Development Discussions*, *12*, 1387–1402. https://doi.org/10.5194 / gmd-12-1387-2019

Bech, J., Codina, B., Lorente, J., & Bebbington, D. (2003). The sensitivity of single polarization weather radar beam blockage correction to variability in the vertical refractivity gradient. *Journal of Atmospheric and Oceanic Technology*, *20*(6), 845–855. https://doi.org/10.1175/1520-0426(2003)020<0845:TSOSPW>2.0.CO;2

Bech, J., Gjertsen, U., & Haase, G. (2007). Modelling weather radar beam propagation and topographical blockage at northern high latitudes. *Quarterly Journal of the Royal Meteorological Society*, *1991*(June), 1191–1204. https://doi.org/10.1002/qj.98

Belmonte, M., & Saibene, Y. (2017). Radar Meteorológico de la EEA Anguil Contribuciones para su uso en el sector agropecuario. In *Ediciones INTA*. Retrieved from https://inta.gob.ar/sites/default/files/inta_radar_meteorologico_de_la_eea_anguil.pdf

Bøvith, T. (2008). *Detection of Weather Radar Clutter* (Technical University of Denmark). Retrieved from https://orbit.dtu.dk/files/4995451/phd201_thb-net.pdf

Burns, T. (2015). Weather Radar (The Basics). Retrieved from Head in the Clouds Feet on the Ground website: http://inthecloudhead.blogspot.com/2015/12/101-weather-radar-basics.html

Büyükbas, E., Sireci, O., Hazer, A., Temir, I., Macit, A., & Gecer, C. (2006). Training Material on Weather Radar Systems. *Instruments and Observing Methods*, (88), 354.

Cao, Q., Zhang, G., Palmer, R. D., & Lei, L. (2012). Detection and Mitigation of Second-Trip Echo in Polarimetric Weather Radar Employing Random Phase Coding. *IEEE Transactions on Geoscience and Remote Sensing*, *50*(4), 1240–1253. https://doi.org/10.1109/TGRS.2011.2164927

Crisologo, I. (2016). What causes blind spots in radar images? Retrieved August 25, 2019, from Philippine Radar Network website: https://philippineradarnetwork.wordpress.com/2016/05/18/radar-beam-blockage/

Delobbe, L. (2006). *Estimation des précipitations à l ' aide d ' un radar météorologique Table des Matières*. Bruselas.

Delrieu, G., Caoudal, S., & Creutin, J. D. (1997). Feasibility of using mountain return for the

correction of ground-based X-band weather radar data. *Journal of Atmospheric and Oceanic Technology*, *14*(3), 368–385. https://doi.org/10.1175/1520-0426(1997)014<0368:FOUMRF>2.0.CO;2

Doviak, R. J., & Zrnic, D. S. (1993). *Doppler radar and weather observations* (2nd ed.; Accademy Press, Ed.). San Diego.

Gabella, M., & Notarpietro, R. (2002). Ground clutter characterization and elimination in mountainous terrain. *ERAD*, 305–311. Retrieved from https://www.copernicus.org/erad/online/erad-305.pdf

Gómez Vargas, E. (2015). *Modelo para la estimación cuantitativa de precipitación a partir de datos de radares polarimétricos*. Pontificia Universidad Javeriana.

Guijarro, R. S.-D. (2001). *Optimización de la medida de lluvia por radar meteorológico para su aplicación hidrológica Tesis Doctoral*. Universitat Politècnica de Catalunya.

Haddad, B., Adane, A., Sauvageot, H., Saudoki, L., & Nailli, R. (2004). Identification and filtering of rainfall and ground radar echoes using textural features. *International Journal of Remote Sensing*, *25*(21), 4641–4656. https://doi.org/https://doi.org/10.1080/01431160310001654455

Heistermann, M., Collis, S., Dixon, M. J., Giangrande, S., Helmus, J. J., Kelley, B., … Wolff, D. B. (2015). The emergence of open-source software for the weather radar community. *Bulletin of the American Meteorological Society*, *96*(1), 117–128. https://doi.org/10.1175/BAMS-D-13-00240.1

Heistermann, M., Jacobi, S., & Pfaff, T. (2013). Technical Note: An open source library for processing weather radar data (wradlib). *Hydrology and Earth System Sciences*, *17*(2), 863–871. https://doi.org/10.5194/hess-17-863-2013

Helmus, J. J., & Collis, S. M. (2016). The Python ARM Radar Toolkit (Py-ART), a Library for Working with Weather Radar Data in the Python Programming Language. *Journal of Open Research Software*, *4*. https://doi.org/10.5334/jors.119

Holleman, I, Michelson, D., Galli, G., & Germann, U. (2006). Quality information for radars and radar data. In *EUMETNET OPERA*.

Holleman, Iwan, & Beekhuis, H. (2005). *Evaluation of three Radar Product Processors*.

Holleman, Iwan, Huuskonen, A., Kurri, M., & Beekhuis, H. (2010). Operational monitoring of weather radar receiving chain using the sun. *Journal of Atmospheric and Oceanic Technology*, *27*(1), 159–166. https://doi.org/10.1175/2009JTECHA1213.1

Hubbert, J. C., Dixon, M., & Ellis, S. M. (2009). Weather radar ground clutter. Part II: Real-time identification and filtering. *Journal of Atmospheric and Oceanic Technology*, *26*(7), 1181–1197. https://doi.org/10.1175/2009JTECHA1160.1

INDECI. (2017). Séptimo Boletín Estadístico Virtual del INDECI de la Gestión Reactiva. *Boletín Estadístico Virtual de La Gestión Reactiva*, *7*, 112.

Jacobi, S., & Heistermann, M. (2016). Benchmarking attenuation correction procedures for six years of single-polarized C-band weather radar observations in South-West Germany. *Geomatics, Natural Hazards and Risk*, *7*(6), 1785–1799. https://doi.org/10.1080/19475705.2016.1155080

Joss, J., & Lee, R. (1995). The application of radar-gauge comparisons to operational precipitation profile corrections. *Journal of Applied Meteorology*, *34*(12), 2612–2630. https://doi.org/10.1175/1520-0450(1995)034<2612:TAORCT>2.0.CO;2

Lakshmanan, V., Zhang, J., Hondl, K., & Langston, C. (2012). A statistical approach to mitigating persistent clutter in radar reflectivity data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *5*(2), 652–662. https://doi.org/10.1109/JSTARS.2011.2181828

O'Donnel, R. (2007). RES.LL-001 Introduction to Radar Systems. Retrieved May 25, 2019, from MIT OpenCourseWare website: https://ocw.mit.edu/resources/res-ll-001-introduction-to-radar-systems-spring-2007/#

Osrodka, K., Szturc, J., & Jurczyk, A. (2014). Chain of data quality algorithms for 3-D single-polarization radar reflectivity (RADVOL-QC system). *Meteorological Applications*, *21*, 256–270. https://doi.org/10.1002/met.1323

Peura, M. (2002). Computer vision methods for anomaly removal. *Second European Conference on Radar Meteorology (ERAD02)*, 312–317.

Pfaff, T. (2013). *Processing and Analysis of Weather Radar Data for Use in Hydrology*.

Pfaff, T., Heistermann, M., & Jacobi, S. (2012). wradlib - an Open Source Library for Weather Radar Data Processing. *ERAD*.

Pirttilä, J., Lehtinen, M. S., Huuskonen, A., & Markkanen, M. (2005). A proposed solution to the range-Doppler dilemma of weather radar measurements by using the SMPRF codes, practical results, and a comparison with operational measurements. *Journal of Applied Meteorology*, *44*(9), 1375–1390. https://doi.org/10.1175/JAM2288.1

Pulkkinen, S., Nerini, D., Pérez Hortal, A. A., Velasco-Forero, C., Seed, A., Germann, U., & Foresti, L. (2019). Pysteps: an open-source Python library for probabilistic precipitation nowcasting (v1.0). *Geoscientific Model Development Discussions*, *12*, 4185–4219. https://doi.org/10.5194/gmd-2019-94

Racoma, B. A., Crisologo, I. A., & David, C. P. (2015). Accumulation-based Advection Field for Rainfall Nowcasting. *Journal of the Philippine Geoscience and Remote Sensing Society*, *1*(1). Retrieved from https://www.researchgate.net/publication/303289455_Accumulation-based_Advection_Field_for_Rainfall_Nowcasting

Rosengaus, M. (1995). Fundamentos de radares meteorologicos: aspectos clasicos (primera de dos partes). *Ingenieria Hidraulica En Mexico*, *10*(1), 55–74.

Rosengaus, M. (1999). *Utilizando Radar Meteorológico*. *6*(junio), 185–198.

Saltikoff, E., Cho, J. Y. N., Tristant, P., Huuskonen, A., Allmon, L., Cook, R., … Joe, P. (2016). The threat to weather radars by wireless technology. *Bulletin of the American Meteorological Society*, *97*(7), 1159–1167. https://doi.org/10.1175/BAMS-D-15-00048.1

Saltikoff, E., Friedrich, K., Soderholm, J., Lengfeld, K., Nelson, B., Becker, A., … Tassone, C. (2019). An overview of using weather radar for climatological studies: Successes, challenges and potential. *Bulletin of the American Meteorological Society*. https://doi.org/10.1175/bams-d-18-0166.1

Sánchez-Diezma, R., & Corral, C. (2000). Curso de Postgrado de Climatología Aplicada: El radar

meteorológico y sus aplicaciones hidrológicas. Retrieved from http://crahi.upc.edu/index.php?option=com_content&view=article&id=59%3Acurs-radar&catid=39&Itemid=109&lang=es

Saudoki, L., & Haddad, B. (2013). Classification of radar echoes with a textural–fuzzy approach: an application for the removal of ground clutter observed in Sétif (Algeria) and Bordeaux (France) sites. *International Journal of Remote Sensing*, *34*(21), 7447–7463. https://doi.org/10.1080/01431161.2013.823522

SELEX. (2017a). *Rainbow 5 File Format*. Neuss-Rosellen.

SELEX. (2017b). *RainView Analyzer® User Guide*. Neuss-Rosellen.

Sireci, O., & WMO. (2015). *Evaluation of CIMO Weather Radars Survey and Web-based Weather Radar Database*.

Steiner, M., & Smith, J. A. (2002). Use of three-dimensional reflectivity structure for automated detection and removal of nonprecipitating echoes in radar data. *Journal of Atmospheric and Oceanic Technology*, *19*(5), 673–686. https://doi.org/10.1175/1520-0426(2002)019<0673:UOTDRS>2.0.CO;2

WMO. (2019). *Commission Weather Radar Data Requirements for Climate Monitoring*. Geneva.

Apéndices

UNIVERSITAS STUDIORUM PIURENSIS

Apéndice 1. Estructura de datos de entrada

rbdict (**OrderedDict**)
- volume(**OrderedDict**)
  - @datetime(**str**)
  - @owner(**str**)
  - @type(**str**)
  - @version(**str**)
  - history(**OrderedDict**)
    - @pdfname(**str**)
    - @ppdfname(**str**)
    - @sdfname(**str**)
    - rawdatafilesr(**OrderedDict**)
      - file(**str**)
  - scan(**OrderedDict**)
    - @date(**str**)
    - @name(**str**)
    - @time(**str**)
    - advancedchanged(**str**)
    - detailedchanged(**str**)
    - pargroup(**OrderedDict**)
      - @refid(**str**)
      - antspeed(**str**)
      - csr(**str**)
      - datatypes(**str**)
      - datatypes_format(**NoneType**)
      - filtermode(**str**)
      - filterwidth(**str**)
      - firstele(**str**)
      - freqagil(**str**)
      - highprf(**str**)
      - lastele(**str**)
      - log(**str**)
      - lowprf(**str**)
      - masterdatatypes(**NoneType**)
      - multitrip(**str**)
      - multitripprfmode(**str**)
      - numele(**str**)
      - numtrip(**str**)
      - posazi(**str**)
      - posele(**str**)
      - pulsewidth(**str**)
      - rangesamp(**str**)
      - rangestep(**str**)
      - scanstrategy(**str**)
      - sectorscan(**str**)

sectrip(**str**)
speckle(**str**)
sqi(**str**)
stagger(**str**)
startazi(**str**)
startele(**str**)
stopazi(**str**)
stopele(**str**)
stoprange(**str**)
timesamp(**str**)
scantime(**str**)
slice(**OrderedDict**)
@refid(**str**)
anglestep(**str**)
antspeed(**str**)
dynz(**OrderedDict**)
@max(**str**)
@min(**str**)
filtermode(**str**)
gdrx_clutter_flag_filter(**str**)
highprf(**str**)
log(**str**)
noise_power_dbz(**str**)
noise_power_dbz_dpv(**str**)
posangle(**str**)
rangesamp(**str**)
rangestep(**str**)
rspradconst(**str**)
slicedata(**OrderedDict**)
@date(**str**)
@time(**str**)
rawdata(**OrderedDict**)
@bins(**str**)
@blobid(**str**)
@depth(**str**)
@max(**str**)
@min(**str**)
@rays(**str**)
@type(**str**)
data(**uint8**)
rayinfo(**OrderedDict**)
@blobid(**str**)
@depth(**str**)
@rays(**str**)
@refid(**str**)
data(**uint16**)
startangle(**str**)

stopangle(**str**)
stoprange(**str**)
timesamp(**str**)
unitid(**str**)
sensorinfo(**OrderedDict**)
@id(**str**)
@name(**str**)
@type(**str**)
alt(**str**)
beamwidth(**str**)
lat(**str**)
lon(**str**)
wavelen(**str**)

Apéndice 2. Bases radar.py

```
# -*- coding: utf-8 -*-
"""
Created on Fri Jun 28 12:53:18 2019

@author: Elmer Lopez Ramirez
"""

#---------- FUNCIONES DE CADENA DE PROCESAMIENTO DEL RADAR ----------#

#Librerías
#import wradlib as wrl
import matplotlib.pyplot as plt
import numpy as np
import datetime
import warnings
from math import ceil

from wradlib.io import read_rainbow
from wradlib.vis import plot_ppi, plot_ppi_crosshair

warnings.filterwarnings('ignore')


class RadarData:

    def __init__(self):
        self.d_type = 'Reflectividad'
        self.data = dict()
        self.metadata = dict()
        self.process = dict ()

    def set_metadata(self,key,value):
        self.metadata[key] = value

    def set_data(self,key,value):
        self.data[key] = value

    def set_fecha(self,horalocal = -5):
        date = self.metadata['date']
        year,month,day = int(date[0:4]),int(date[5:7]),int(date[8:10])
        time = self.metadata['time']
        hour = int(time[0:2])
        minute = 5*ceil(int(time[3:5])/5)
        self.UTC = datetime.datetime(year,month,day,hour)
```

```
        self.UTC = self.UTC + datetime.timedelta(minutes = minute)
        delta = datetime.timedelta(hours = horalocal)
        self.local_time = self.UTC + delta

    def set_name(self,value):
        self.name = value


#Función lectura de datos

def read_data(abs_file):

    #Cargamos archivo Rainbow
    rbdict = read_rainbow(abs_file)

    #Obtenemos datos azimutales
    azi = rbdict['volume']['scan']['slice']['slicedata']['rayinfo']['data']
    azidepth = float(rbdict['volume']['scan']['slice']
              ['slicedata']['rayinfo']['@depth'])
    azirange = float(rbdict['volume']['scan']['slice']
              ['slicedata']['rayinfo']['@rays'])
    azires = float(rbdict['volume']['scan']['slice']['anglestep'])
    azi = (azi * azirange / 2**azidepth) * azires

    #Obtenemos los valores relacionados al alcance del radar
    stoprange = float(rbdict['volume']['scan']['slice']['stoprange'])
    rangestep = float(rbdict['volume']['scan']['slice']['rangestep'])
    r = np.arange(0, stoprange, rangestep)

    #Obtenemos los datos de reflectividad
    data = rbdict['volume']['scan']['slice']['slicedata']['rawdata']['data']
    datadepth = float(rbdict['volume']['scan']['slice']
               ['slicedata']['rawdata']['@depth'])
    datamin = float(rbdict['volume']['scan']['slice']
             ['slicedata']['rawdata']['@min'])
    datamax = float(rbdict['volume']['scan']['slice']
             ['slicedata']['rawdata']['@max'])
    data = datamin + data * (datamax - datamin) / 2 ** datadepth

    #Obtenemos metadata importante
    unit = rbdict['volume']['scan']['slice']['slicedata']['rawdata']['@type']
    time = rbdict['volume']['scan']['slice']['slicedata']['@time']
    date = rbdict['volume']['scan']['slice']['slicedata']['@date']
    lon = float(rbdict['volume']['sensorinfo']['lon'])
    lat = float(rbdict['volume']['sensorinfo']['lat'])

    #Reordenamos la información de cada azi
```

```
    azi_ord, data_ord = ordenar(azi,data)

    #Organizamos la información en la clase creada
    clase = RadarData()

    clase.set_data('azi',azi_ord)
    clase.set_data('range',r)
    clase.set_data('data',data_ord)

    clase.set_metadata('unit',unit)
    clase.set_metadata('time',time)
    clase.set_metadata('date',date)
    clase.set_metadata('lon',lon)
    clase.set_metadata('lat',lat)
    clase.set_name(abs_file)


    return clase

def ordenar(azi,data):
    """Permite ordenar los datos en el eje azimuth a partir de cero"""
    azi = np.array([round(x) for x in azi])
    azi_ord = sorted(azi)
    tmp = azi_ord.index(azi[0])
    azi_ord = np.array(azi_ord)

    data_ord = np.zeros_like(data)
    data_ord[0:tmp,:]=data[360-tmp:360,:]
    data_ord[tmp:360,:]=data[0:360-tmp,:]
    return(azi_ord,data_ord)


def simple_plot(fig,radardata, proj = '',crosshair = False):

    r = radardata.data['range']
    azi = radardata.data['azi']
    data = radardata.data['data']
    unit = radardata.metadata['unit']
    date = radardata.metadata['date']
    time = radardata.metadata['time']
    d_type = radardata.d_type

    #Configuraciones de diseño
    kwargs = dict()
    kwargs['cmap'] = plt.cm.viridis
    kwargs['vmin'] = -31.5
    kwargs['vmax'] = 95.5
```

```
fig.clf()

if proj == '':
   cgax, pm = plot_ppi(data,r=r, az=azi,fig=fig,**kwargs)
   cgax.set_xlabel('Rango x [km]',fontsize = 12)
   cgax.set_ylabel('Rango y [km]',fontsize = 12)
elif proj == 'cg':
   cgax, pm = plot_ppi(data, r=r, az=azi,fig=fig,proj = 'cg',**kwargs)
   caax = cgax.parasites[0]
   caax.set_xlabel('Rango x [km]',fontsize = 12)
   caax.set_ylabel('Rango y [km]',fontsize = 12)
   plt.text(1.0, 1.05, 'Azimuth', transform=caax.transAxes, va='bottom',
   ha='right')

if crosshair :
   plot_ppi_crosshair(site=(0,0), ranges=[25, 50, 75,100],
                angles=[0, 90, 180, 270],
                line=dict(color='white'),
                circle={'edgecolor': 'white'})
plt.xlim([-101,101])
plt.ylim([-101,101])

title = 'Plan Position Indicator\n {} {} UTC'.format(date,time)
t = plt.title(title, fontsize=14)
t.set_y(1.00)

cbar = plt.gcf().colorbar(pm, pad=0.05)
cbar.set_label(d_type +' ['+  unit + ']')
```

Apéndice 3. Interfaz.py

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 15 08:30:10 2019

@author: Elmer Jeanpierre Lopez Ramirez
@email: jeanpierre.lopez.ramirez@gmail.com
@cellphone: 990 641 345
@version: 3.3.0
@license: MIT License
"""

# Librerías
import tkinter as Tk
from tkinter import filedialog
from tkinter import messagebox
from PIL import Image, ImageTk
from pathlib import Path
import Bases_GUI as radar

import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.backends.backend_tkagg import NavigationToolbar2Tk
from matplotlib.backend_bases import key_press_handler
from os import getcwd

# Variables globales
global folder_path
global group
global scrollbar

# Historial
global historyname, history
historyname, history = [], []

# Zoom in/out
global size
size = 100
size_pred = [100, 75, 50, 25, 15, 10]


# FUNCIONES

def browse_button():
    """Función comando del button "Seleccione carpeta". Sirve para realizar
    la selección de la carpeta así como para presentar en el Listbox la lista
```

```
    de archivos válidos para la selección. En caso de no encontrarse archivos
    válidos se presentará el mensaje "No hay archivos válidos" en el Listbox"""

    global lfechas
    global lext
    global lrelf
    global filename
    global size

    filename = filedialog.askdirectory()
    if filename == '':
        filename = getcwd()

    folder_path.set(filename)
    print(filename)

    lista_archivos = names(filename)
    lfechas, lext, lrelf = label_generator(lista_archivos)
    list_files.delete(0, Tk.END)
    if len(lfechas) != 0:
        for index in range(len(lfechas)):
            element = lfechas[index]
            list_files.insert(Tk.END, element)
    else:
        list_files.insert(0, 'No hay archivos válidos')


def names(ruta=Path.cwd()):
    """Devuelve una lista con los nombres de los archivos contenidos en la ruta
    especificada. Dicha lista no  filtrada.
    Argumentos :
        ruta (pathlib.WindowsPath/str) Directorio o carpeta del cual se extraen
                                los archivos
    Salidas:
        list_name (list)   Lista con los nombres relativos de los archivos.
                        La extensión está incluida. """
    list_name = [arch.name for arch in Path(ruta).iterdir() if arch.is_file()]
    return list_name


def show_view(event):
    global size
    """Función de evento (event function) para refrescar el Canvas con la gráfica
    del archivo seleccionado"""
    index = list_files.curselection()

    if (len(index) == 1
```

```
        and list_files.get(index[0]) != 'No hay archivos válidos'):
        index = index[0]
        element = lfechas[index]
        print(element)
        archivo = lrelf[index]
        radardata = radar.read_data(archivo)
        radar.simple_plot(fig=fig, radardata=radardata)
        canvas.draw()
        canvas.get_tk_widget().pack(side=Tk.TOP, fill=Tk.BOTH, expand=1)
        optionsmenu.entryconfig(index='Zoom in', state=Tk.NORMAL)
        optionsmenu.entryconfig(index='Zoom out', state=Tk.DISABLED)
        size = 100
        if len(history) == 20:
            historyname.pop()
            history.pop()
        historyname.append(element)
        history.append(archivo)

def label_generator(lista_nombres):
    """Devuelve tres listas con los formatos adaptados de los nombres de archivos
    para su visualizacion Se recogen sólo los archivos con extensión .azi .
    Argumentos:
        lista_nombres (list)  Lista de nombres relativos de los archivos.
    Salidas :
        list_time (list)  Lista con elementos en formato str (dd/mm/yyyy hh:mm)
        list_type (list)  Lista con elementos en formato str(dBZ o dBuZ)
        list_filt (list)  Lista original filtrada """

    list_time, list_type, list_filt = [], [], []
    for nombre in lista_nombres:
        if nombre.endswith('.azi'):
            list_filt.append(filename + "/" + nombre)
            tmp = (nombre[6:8] + "/" + nombre[4:6] + "/"
                   + nombre[0:4] + " " + nombre[8:10] + ":" + nombre[10:12])
            if nombre.find('dBZ') != -1:
                ext = 'dBZ'
                list_type.append('dBZ')
            elif nombre.find('dBuZ') != -1:
                list_type.append('dBuZ')
                ext = 'dBuZ'
            else:
                list_type.append(' ')
                ext = ''
            tmp = '    ' + tmp  + '      ' + ext
            list_time.append(tmp)
    return list_time, list_type, list_filt
```

```
def process_text(text1, text2):
    """Concatenación de los elementos de dos listas con salto de linea.
      Las listas deben tener el mismo número de elementos.
    Argumentos:
        text1 (list) Lista 1
        text2 (list) Lista 2
    Salidas:
        tmp   (str) Cadena con saltos de linea y espacios intermedios
                entre los elementos de las listas. """
    tmp = ""
    for element in range(len(text1)):
        tmp = tmp + text1[element] + "    " + text2[element] + "\n"
    return tmp


def abrir_archivo():
    """Abre un archivo tipo .azi y lo muestra en el visualizador"""
    files = filedialog.askopenfile(filetypes=(("Azi files", "* .azi"),
                                   ("All files", "* . * ")))
    if files is not None and files.name.endswith('.azi'):
        print(files.name)
        folder_path.set(files.name)
        radardata = radar.read_data(files.name)
        radar.simple_plot(fig=fig, radardata=radardata)
        canvas.draw()
        canvas.get_tk_widget().pack(side=Tk.TOP, fill=Tk.BOTH, expand=1)
    else:
        messagebox.showerror(message='Archivo inválido', title='Error')


def root_quit():
    """Destruye la ventana actual."""
    root.withdraw()
    plt.close()
    root.destroy()


def zoom_in():
    """Función asociada a realizar acercamientos progresivos a la gráfica,
    estos están ligados a la lista global size_pred """
    global size
    index = size_pred.index(size)
    index = index + 1
    size = size_pred[index]
    if index < len(size_pred):
        plt.xlim([-size, size])
```

```
        plt.ylim([-size, size])
        canvas.draw()
    else:
        optionsmenu.entryconfig(index='Zoom in', state=Tk.DISABLED)
    if size == min(size_pred):
        optionsmenu.entryconfig(index='Zoom in', state=Tk.DISABLED)
    else:
        optionsmenu.entryconfig(index='Zoom out', state=Tk.NORMAL)


def zoom_out():
    """Función asociada a realizar alejamientos progresivos a la gráfica,
    estos están ligados a la lista global size_pred """
    global size
    index = size_pred.index(size)
    if index > 0:
        size = size_pred[index - 1]
        plt.xlim([-size, size])
        plt.ylim([-size, size])
        canvas.draw()
    else:
        optionsmenu.entryconfig(index='Zoom out', state=Tk.DISABLED)
    if size == max(size_pred):
        optionsmenu.entryconfig(index='Zoom out', state=Tk.DISABLED)
    else:
        optionsmenu.entryconfig(index='Zoom in', state=Tk.NORMAL)


def info():
        ventana = Tk.Toplevel(root)
        ventana.withdraw()
        ventana.title('Contacto')
        ventana.iconbitmap('Simbolo3D.ico')
        ventana.resizable(width=False, height=False)
        imagen = Image.open("Message2.png")
        img = ImageTk.PhotoImage(imagen, master=ventana)
        label = Tk.Label(ventana, image=img)
        label.image = img
        label.pack(fill="both", expand="yes")
        ventana.grab_set()
        ventana.deiconify()
        ventana.geometry('+400+200')
        ventana.wait_window(root)


def copiar():
    root.clipboard_clear()
```

```
    index = list_files.curselection()[0]
    root.clipboard_append(lrelf[index])


def historial():
    global lfechas, lrelf
    print('Historial')
    list_files.delete(0, Tk.END)
    if len(history) != 0:
        for element in historyname:
            list_files.insert(Tk.END, element)
            lfechas = historyname.copy()
            lrelf = history.copy()
    else:
        list_files.insert(0, 'No hay archivos válidos')


# Funciones relacionadas a eventos


def on_key_press(event):
    print("you pressed {}".format(event.key))
    key_press_handler(event, canvas, toolbar)


def desplegar_menu(*event):
    evento = event[0]
    list_files.select_clear(0, last=Tk.END)
    list_files.select_set(list_files.nearest(evento.y))
    try:
        menu_portapapeles.tk_popup(evento.x_root, evento.y_root, 0)
    finally:
            menu_portapapeles.grab_release()


# Cuerpo principal

# Ventana madre
root = Tk.Tk()
root.withdraw()
root.title('Wradchibi')
root.resizable(width=False, height=False)
root.iconbitmap('Simbolo3D.ico')

# Menú de selección
menubar = Tk.Menu(root)
filemenu = Tk.Menu(menubar, tearoff=0)
```

```
filemenu.add_command(label="Selecione carpeta", command=browse_button)
filemenu.add_command(label="Seleccione archivo", command=abrir_archivo)
filemenu.add_separator()
filemenu.add_command(label="Salir", command=root_quit)
menubar.add_cascade(label="Archivo", menu=filemenu)

optionsmenu = Tk.Menu(menubar, tearoff=0)
optionsmenu.add_command(label='Zoom in', command=zoom_in)
optionsmenu.add_command(label='Zoom out', command=zoom_out)
optionsmenu.add_separator()
optionsmenu.add_command(label="Historial", command=historial)
menubar.add_cascade(label="Opciones", menu=optionsmenu)

helpmenu = Tk.Menu(menubar, tearoff=0)
helpmenu.add_command(label='Contacto', command=info)
menubar.add_cascade(label='Ayuda', menu=helpmenu)

root.config(menu=menubar)

# Label para el directorio seleccionado
folder_path = Tk.StringVar()
folder_path.set('Carpeta no seleccionada')
lbl1 = Tk.Label(master=root, textvariable=folder_path)
lbl1.grid(row=0, column=3, padx=15, pady=5)

# Button de selección
select_button = Tk.Button(root, text="Seleccione carpeta",
                    command=browse_button)
select_button.grid(row=0, column=1)

# Cuadro de empaquetamiento
group = Tk.LabelFrame(root, text="Archivos(hora UTC)")
group.grid(row=1, column=1, padx=15, pady=10)

# Listbox
list_files = Tk.Listbox(group, activestyle=Tk.DOTBOX)
list_files.pack(side=Tk.LEFT)
list_files.config(width=30, height=25)
list_files.bind('<Double-Button-1>', show_view)
list_files.bind('<Return>', show_view)
list_files.bind("<Button-3>", desplegar_menu)

# Barra de desplazamiento
scrollbar = Tk.Scrollbar(group)
scrollbar.pack(side=Tk.RIGHT, fill=Tk.BOTH)
list_files.config(yscrollcommand=scrollbar.set)
scrollbar.pack(side=Tk.RIGHT, fill=Tk.BOTH)
```

```
# Lienzo y figura asociada
frame = Tk.Frame(root)
frame.grid(row=1, column=3, padx=10, pady=10)

fig = plt.figure(figsize=(6, 5), dpi=100)
canvas = FigureCanvasTkAgg(fig, master=frame)
canvas.draw()
canvas.get_tk_widget().pack(side=Tk.TOP, fill=Tk.BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, frame)
toolbar.update()
canvas.get_tk_widget().pack(side=Tk.TOP, fill=Tk.BOTH, expand=1)
canvas.mpl_connect("key_press_event", on_key_press)

# Popup Menú
menu_portapapeles = Tk.Menu(root, tearoff=0)
menu_portapapeles.add_command(label="Copiar", command=copiar)

# Protocolos
root.protocol("WM_DELETE_WINDOW", root_quit)
root.deiconify()

# Bucle
root.mainloop()
```

Apéndice 4. Utils.py

```
# -*- coding: utf-8 -*-
"""
Created on Sat Jul  6 12:17:38 2019

@author: Elmer Jeanpierre Lopez Ramirez
@email: jeanpierre.lopez.ramirez@gmail.com
@cellphone: 990 641 345
@version: 1.0.0
@license: MIT License
"""
# Analisis de clutter

# Librerías
from pathlib import Path
import matplotlib.pyplot as plt
import numpy as np
import wradlib as wrl
import warnings

from copy import copy
from random import shuffle
from scipy.fftpack import fft, ifft
from scipy.stats import mode
from sklearn.mixture import BayesianGaussianMixture
from scipy.stats import norm
from scipy import ndimage
import skimage

import Bases_GUI as radar

warnings.filterwarnings('ignore')

__version__ = "1.0.0"

#############################################################################
###
################################## FUNCIONES
#####################################


def zona(fig, r_values, azi_values=[0, 360]):
    """Función para la selección manual de zonas para cluttermap. Grafica las
    zonas(trapecio circular).
    Argumentos:
        fig (plt.Figure) Figura en la cual se mostrará la gráfica
```

```
        r_values (list)  Lista de los dos valores(minimo y maximo) en distancia
                         correspondientes a la zona a evaluar
        azi_values (list) Lista de los dos valores(minimo y maximo) en ángulo
                         correspondientes a la zona a evaluar """
    r_min = r_values[0]
    r_max = r_values[1]
    azi_min = azi_values[0]
    azi_max = azi_values[1]
    range_r = int(r_max - r_min)
    range_azi = azi_max - azi_min
    mask = np.ones(shape=(range_azi, range_r))
    r = np.arange(r_min, r_max) / 10
    azi = np.arange(azi_min, azi_max)
    ax, pm = wrl.vis.plot_ppi(mask, r=r, cmap=plt.cm.gray, alpha=0.2, az=azi, fig=fig)
    return fig


def complementos(fig, rmax=100, title='Datos de ejemplo',
                 xlabel='Rango (km)', ylabel='Rango (km)'):
    """Función decorativa"""
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.xlim(-rmax, rmax)
    plt.ylim(-rmax, rmax)
    plt.xticks(np.linspace(-rmax, rmax, num=5))
    plt.yticks(np.linspace(-rmax, rmax, num=5))


def complement_ax(ax, rmax=15):
    """Función decorativa"""
    ax.set_xlim([-rmax, rmax])
    ax.set_ylim([-rmax, rmax])
    ax.set_xticks([])
    ax.set_yticks([])
    return ax


def complement_scan(fig, limits=[-31.5, 40]):
    """Función decorativa"""
    plt.title('Líneas de escaneo')
    plt.ylim(limits)
    plt.xlabel('Distancia(km)')
    plt.ylabel('Reflectividad(dBZ)')
    plt.grid()
```

```
def replace(matrix, mat_bool):
    """Función de indexación booleana. Reemplaza los valores de una matriz por
    los minimos valores de reflectividad
    Argumentos:
        matrix (numpy.array) Matriz de reflectividad
        mat_bool (numpy.array) Matriz booleana que actuará de máscara para el
                    reemplazo
    Ambas matrices deben ser del mismo tamaño.
    Salidas:
        matrix (numpy.array) Matriz con valores reemplazados en base a
        la máscara"""
    tmp = copy(matrix)
    tmp[mat_bool] = -31.5
    return tmp


def files(ruta=Path.cwd()):
    """Devuelve una lista con los nombres de los archivos contenidos en la ruta
    especificada. Dicha lista no esta filtrada.
    Argumentos :
        ruta (pathlib.WindowsPath/str) Directorio o carpeta del cual se extraen
                        los archivos
    Salidas:
        list_name (list)   Lista con los nombres relativos de los archivos.
                    La extensión está incluida. """
    list_name = [arch.name for arch in Path(ruta).iterdir() if arch.is_file()]
    return list_name


def valids(lista):
    """Función de comprobación de los archivos de entrada. Devuelve una lista
    con los elementos que presentan el formato .azi
    Argumentos:
        lista (list) Lista con las cadenas de caracteres a evaluar
    Salidas:
        list_valid  (list) Lista de las cadenas que terminan en .azi """
    list_valid = []
    for element in lista:
        if element.endswith('.azi'):
            list_valid.append(element)
    return list_valid


def dataclass(ruta, list_relfile='Easter egg'):
    """Función para transformar una lista de nombres de archivos en objetos
    de la clase Radardata
    Argumentos:
```

```
        ruta (str) Cadena correspondiente al directorio
        list_relfile (list) Lista de cadenas de caracteres que corresponden
                    a los nombres relativos de los archivos del
                    directorio
    Salidas:
        clutterdata (list) Lista con los objetos tipo Radardata de la ruta dada
    En caso no se proporcione la lista, está se obtendra a partir de la ruta
    proporcionada"""
    if list_relfile == 'Easter egg':
        list_relfile = valids(files(ruta))
    clutterdata = []
    for rel_name in list_relfile:
        abs_name = ruta + '/' + rel_name
        clutterdata.append(radar.read_data(abs_name))
    return clutterdata


def matriz3d(clutterdata):
    """Función que proporciona una matriz de tres ejes en base a una lista
    de objetos Radardata
    Argumentos:
        clutterdata (list) Lista con los objetos Radardata
    Salidas:
        matriz (np.array) Matriz de tamaño (360,1000,len(clutterdata))
                    que contiene todos los datos de reflectividad de los
                    objetos. """
    matriz = np.zeros((360, 1000, len(clutterdata)))
    for index, clutter in enumerate(clutterdata):
        data_dbz = clutter.data['data']
        matriz[:, :, index] = data_dbz
    return matriz


def ejes():
    """Función de utilidad. Proporciona los valores más comunes de r y azi.
    Salidas:
        azi (numpy.array) Valores del ángulo azimutal entre 0 y 360° espaciados
                    cada 1°
        r (numpy.array) Valores de distancia entre 0 y 100 km espaciados 0.1 km
        """
    r = np.arange(0, 1000) / 10
    azi = np.arange(0, 360)
    return azi, r


def testeo(explore=False, rmax=15):
    """Función que carga los datos para probar los diferentes cluttermap.
```

```
    Consta de 3 tipos de datos:
    1) No hay precipitación, solo clutter
    2) Precipitación no superpuesta en la zona de clutter(Precipitación tipo 1)
    3) Precipitación superpuesta en la zona de clutter(Precipitación tipo 2)
    Argumentos:
        explore (bool) Permite escoger si se grafican los datos o no. Por
                defecto no lo hace.
    Salidas:
        testdata (list) Datos numpy.array embutidos en una lista
        title_test (list) Nombres clave de los datos """

    file_test1 = 'C:/Python37/Clutter/2019050222000000dBuZ.azi'
    file_test2 = 'C:/Python37/Lluvia/2019051416350000dBuZ.azi'
    file_test3 = 'C:/Python37/Lluvia/2019051415200000dBuZ.azi'
    title_test = ['Solo clutter', 'Precipitación tipo 1',
                  'Precipitación tipo 2']

    testclass, testdata = [], []
    testclass.append(radar.read_data(file_test1))
    testclass.append(radar.read_data(file_test2))
    testclass.append(radar.read_data(file_test3))

    testdata.append(testclass[0].data['data'])
    testdata.append(testclass[1].data['data'])
    testdata.append(testclass[2].data['data'])

    azi, r = ejes()

    if explore:
        # Visualizamos los datos
        fig = plt.figure(figsize=(9, 4), dpi=100)
        tmp = plt.title('Datos para test', fontweight='bold')
        tmp.set_y(0.9)
        plt.axis('off')
        for index, test in enumerate(testdata):
            ax = fig.add_subplot(131 + index)
            ax, pm = wrl.vis.plot_ppi(testdata[index], r=r, ax=ax)
            ax.set_title(title_test[index])
            ax = complement_ax(ax, rmax=rmax)
            ax.axis('off')

    return testdata, title_test


def testeo_results(cluttermaps, results, rmax=20, title1='Cluttermaps',
                   title2='Datos filtrados',*args, **kwargs):
    """Función de evaluación gráfica de la aplicación de los cluttermap.
```

```
    Argumentos:
        cluttermaps (list) Lista con las matrices de los cluttermap aplicados
        results  (list) Lista con las matrices de los resultados obtenidos
        r_max (int) Límites de la gráfica a mostrar.
        title (str) Título de la gráfica de cluttermap
    Salidas:
        Presenta dos gráficas en pantalla, la primera de ellas corresponde a
        los cluttermap y la segunda al resultado de aplicarlos. Las gráficas
        ya presentan un formato de salida por defecto. """
    azi, r = ejes()
    testdata, title_test = testeo()

    if cluttermaps is not None:
        fig = plt.figure(figsize=(9, 4), dpi=100)
        tmp = plt.title(title1, fontweight='bold')
        tmp.set_y(0.9)
        plt.axis('off')

        for index, cluttermap in enumerate(cluttermaps):
            ax = fig.add_subplot(131 + index)
            ax, pm = wrl.vis.plot_ppi(cluttermap, r=r, cmap=plt.cm.gray, ax=ax,
                                      *args, **kwargs)
            ax = complement_ax(ax, rmax=rmax)
            ax.axis('off')

    fig = plt.figure(figsize=(9, 4), dpi=100)
    tmp = plt.title(title2, fontweight='bold')
    tmp.set_y(0.9)
    plt.axis('off')

    for index, result in enumerate(results):
        ax = fig.add_subplot(131 + index)
        ax, pm = wrl.vis.plot_ppi(result, r=r, ax=ax,*args, **kwargs)
        ax.set_title(title_test[index])
        ax = complement_ax(ax, rmax=rmax)
        ax.axis('off')

    # Puede añadirse como salidas fig1, fig2 para que el usuario pueda
    # exportar las gráficas sin complicaciones.


def ploteo(data, rmax=100, circle=False, title='Datos', colorbar=False,
           *args,**kwargs):
    """Función de utilidad. Grafica los datos de manera sencilla a la vez
    que lo decora con parámetros por defecto. """
    azi, r = ejes()
    if circle:
```

```
        r = r[0:round(rmax * 10)]
        data = data[:, 0:round(rmax * 10)]
    fig = plt.figure(figsize=(5, 5), dpi=100)
    cgax, pm = wrl.vis.plot_ppi(data, r=r, az=azi, fig=fig,*args,**kwargs)
    complementos(fig, rmax=rmax, title=title)
    if colorbar:
        plt.colorbar(pm, shrink=0.75)
    return fig


def test_battery(explore=False):

    ruta = 'C:\Python37\Batería de pruebas'
    list_name = files(ruta)
    list_valid = valids(list_name)

    data_test = dataclass(ruta)

    if explore:
        print('\nBatería de pruebas')
        print('Número de archivos:', len(list_valid))
        print("\n{0:7} {1:30} {2} ".format("N)".ljust(7), "Nombre del archivo",
              "Fecha y hora local"))
        for index, data in enumerate(data_test):
            data.set_fecha()
            tmp = data.name.find('/')
            tmp = data.name[tmp + 1:]
            i = str(index) + ")"
            i.ljust(7)

            print("{0:7} {1:30} {2} ".format(i, tmp, str(data.local_time)))

    return data_test


def testeo_battery(cluttermap, arg=dict(), function=None,
                   n=len(test_battery()), rmax=20, heatmap=False,
                   *args, **kwargs):

    data_test = test_battery(explore=False)
    azi, r = ejes()

    title = ["Cluttermap", "Dato original", "Dato filtrado", "Heatmap"]

    try:
        print("\nBatería de datos")
        print("Método de filtrado escogido: {}".format(function.__name__))
```

```
except AttributeError:
    print("No se ha pasado como argumento ninguna función")
    return

if n > len(data_test):
    print("El número de muestras pedidas es más grande que el original")
    n = len(data_test)
    number = range(len(data_test))
elif n == len(data_test):
    number = range(len(data_test))
elif n < len(data_test):
    data_test = [[x, data_test[x]] for x in range(len(data_test))]
    shuffle(data_test)
    data_test = data_test[0:n]
    number = [x[0] for x in data_test[0:n]]
    data_test = [x[1] for x in data_test[0:n]]

testdata = []
for test in data_test:
    testdata.append(test.data['data'])

clutterdata, data_filter = [], []

arg2 = copy(arg)
gabella = arg2.get('gabella')
if gabella is None:
    gabella = True
else:
    arg2.pop('gabella')

if type(cluttermap) == np.ndarray:
    print('Método generación cluttermap: Estático')
    for test in testdata:
        data_no_clutter = function(data=test, cluttermap=cluttermap,
                                   gabella=gabella)
        data_filter.append(data_no_clutter)
        clutterdata.append(cluttermap)
else:
    print("Método generación cluttermap: {}".format(cluttermap.__name__))
    for test in testdata:
        arg2['test'] = test
        cltmp = cluttermap(**arg2)
        data_no_clutter = function(data=test, cluttermap=cltmp,
                                   gabella=gabella)
        data_filter.append(data_no_clutter)
        clutterdata.append(cltmp)
```

```
    print("Número de muestras: {}".format(n))
    print("Distancia desde el centro: {} km".format(rmax))

    for index in range(len(data_filter)):
        fig = plt.figure(figsize=(9, 4), dpi=100)
        tmp = plt.title('Test N° {}'.format(number[index]), fontweight='bold')
        tmp.set_y(0.9)
        plt.axis('off')

        ax = fig.add_subplot(131)
        ax, pm = wrl.vis.plot_ppi(clutterdata[index], cmap=plt.cm.gray,
                                  r=r, ax=ax)
        ax.set_title(title[0])
        ax = complement_ax(ax, rmax=rmax)
        ax.axis('off')

        ax = fig.add_subplot(132)
        ax, pm = wrl.vis.plot_ppi(testdata[index], r=r, ax=ax, vmin=-31.5,
                                  vmax=95.5)
        ax.set_title(title[1])
        ax = complement_ax(ax, rmax=rmax)
        ax.axis('off')

        ax = fig.add_subplot(133)
        ax, pm = wrl.vis.plot_ppi(data_filter[index], r=r, ax=ax, vmin=-31.5,
                                  vmax=95.5)
        ax.set_title(title[2])
        ax = complement_ax(ax, rmax=rmax)
        ax.axis('off')

    return data_filter


def area(units='m2'):
    azi, r = ejes()
    area = np.pi * ((r + 0.1)**2 - r ** 2) / (2 * 180)
    if units == 'm2':
        area = area * 1000 ** 2
    return area


def area_matriz():
    area_per_bin = area()
    azi, r = ejes()
    Area = np.zeros(shape=(len(azi), len(r)))
    for i in range(len(azi)):
        Area[i, :] = area_per_bin
```

```
    return Area

##############################################################################
###
#########################   MÉTODOS CLUTTERMAP
##############################


def different(clutter, test=None, value=-31.5):
    return clutter > value


def classification(test, clutter=None, value=0.94):

    clutter_pos = np.zeros(shape=(360, 1000))
    clutter_pos = clutter_pos == 0
    clutter_pos[0, 0] = False
    clutter_neg = np.ones(shape=(360, 1000))
    clutter_neg = clutter_neg == 0
    clutter_pos[0, 0] = True

    unique, counts = np.unique(test, return_counts=True)
    stats_test = dict(zip(unique, counts))[-31.5]
    stats_test = stats_test / (360 * 1000)

    if stats_test > value:
        return clutter_pos
    else:
        return clutter_neg


def limits(test, clutter_min, clutter_max):
    return (test < clutter_max) * (test > clutter_min)


def fourier(test, Matriz, threshold=0.1):
    shape = list(Matriz.shape)
    shape[2] = shape[2] + 1
    Fourier = fft(Matriz, n=shape[2], axis=2) / (shape[2] - 1)
    Mat_expand = np.zeros(shape=shape)
    Mat_expand[:, :, 0:shape[2]-1] = Matriz
    Mat_expand[:, :, shape[2] - 1] = test
    Fourier_expand = fft(Mat_expand, n=shape[2], axis=2) / shape[2]
    tim = Fourier - Fourier_expand
    tim = abs(tim)
    underc = tim[:, :, 0]
    underc = (underc > threshold)
```

```
    return ~underc*(test > -31.5)


def fourier2(test, Matriz, n_armonic=20, threshold=7):
    shape = list(Matriz.shape)
    shape[2] = shape[2] + 1
    Fourier = fft(Matriz, n=shape[2], axis=2)
    Fourier[:, :, n_armonic:shape[2]] = 0

    Mat_expand = np.zeros(shape=shape)
    Mat_expand[:, :, 0:shape[2] - 1] = Matriz
    Mat_expand[:, :, shape[2] - 1] = test

    Fourier_expand = fft(Mat_expand, n=shape[2], axis=2)
    tim = Fourier - Fourier_expand
    transf = ifft(tim)
    transf = abs(transf)

    return transf[:, :, -1] > threshold


##############################################################################
###
##########################  MÉTODOS FILTRADO  ################################

def interpolation(data, cluttermap, gabella=True,*args, **kwargs):
    """Método de interpolación polar del clutter """
    data_no_clutter = wrl.ipol.interpolate_polar(data, cluttermap, *args, **kwargs)
    if gabella:
        clutter_gabella = wrl.clutter.filter_gabella(data_no_clutter, tr1=12,
                                     n_p=6, tr2=1.1)
        data_no_clutter = wrl.ipol.interpolate_polar(data_no_clutter,
                                     clutter_gabella, *args, **kwargs)
    return data_no_clutter


def sustitution(data, cluttermap, gabella=True,*args, **kwargs):
    """Método de reemplazo polar del clutter  por el valor de -31.5 dBZ. """
    data_no_clutter = replace(data, cluttermap)
    if gabella:
        clutter_gabella = wrl.clutter.filter_gabella(data_no_clutter, tr1=12,
                                     n_p=6, tr2=1.1)
        data_no_clutter = wrl.ipol.interpolate_polar(data_no_clutter,
                                     clutter_gabella,*args, **kwargs)
    return data_no_clutter
```

```
def classifilter(data, cluttermap, gabella=True):
    """Método de filtrado por clasificación. Solo se clasifica el resultado
    si se considera clutter"""
    if cluttermap[1, 1] is True:
        data_no_clutter = np.zeros(shape=(360, 1000)) - 31.5
    else:
        if gabella:
            clutter_gabella = wrl.clutter.filter_gabella(data, tr1=12, n_p=6,
                                                         tr2=1.1)
            data_no_clutter = wrl.ipol.interpolate_polar(data, clutter_gabella)
        else:
            data_no_clutter = data
    return data_no_clutter


##############################################################################
###
############################  MÉTODOS HEATMAPS
################################


def dist(test, ref):
    stats = abs(test - ref)
    return stats


def distribution(test, Matriz):

    def prob(listprob, ref, value):
        idx = (np.abs(ref - value)).argmin()
        return listprob[idx]

    muestras = np.arange(-34, 50, 2)
    shape = Matriz.shape
    stats = np.zeros(shape=(shape[0], shape[1]))

    for r_value in range(shape[1]):
        for azi_value in range(shape[0]):

            data = Matriz[azi_value, r_value, :]
            data = data.reshape(len(data), -1)
            Model = BayesianGaussianMixture(n_components=2,
                                            covariance_type='spherical')
            Model.fit(data)

            X = norm(loc=Model.means_[0], scale=np.sqrt(Model.covariances_[0]))
            Y = norm(loc=Model.means_[1], scale=np.sqrt(Model.covariances_[1]))
```

```
        aportex = X.cdf(muestras)*Model.weights_[0]
        aportey = Y.cdf(muestras)*Model.weights_[1]

        acum = aportex + aportey

        listprob = [acum[x+1]-acum[x] for x in range(len(acum) - 1)]

        if test[azi_value, r_value] < 50:
            stats[azi_value, r_value] = 1 - prob(listprob, muestras,
                                    test[azi_value, r_value])
        else:
            stats[azi_value, r_value] = 0

    return stats

############################## FILTRADO ESPACIAL
################################


def filter_mask(mask, test, threshold=8):
    binary = test > -31.5
    data_mask = ndimage.correlate(binary, mask)
    data_mask = abs(data_mask)
    return data_mask >= threshold


def mode_binary(test,mode):
    if mode == 'fill':
        binary = test == -31.5
    elif mode == 'filter':
        binary = test > -31.5
    else:
        binary = test >-31.5
    return binary


def filter_conectivity(test, connect=2, mode='filter'):
    binary = mode_binary(test, mode)
    label_im, nb_labels = ndimage.label(binary)
    sizes = ndimage.sum(binary, label_im, range(nb_labels + 1))
    mask_size = sizes < connect
    remove_pixel = mask_size[label_im]
    binary_two = copy(binary)
    binary_two[remove_pixel] = 0
    cluttermap = ~binary_two * binary
    return cluttermap
```

```
def filter_area(test, threshold=5000, mode='filter'):
    binary = mode_binary(test, mode)
    label_im, nb_labels = ndimage.label(binary)
    mask = area_matriz()
    sizes = ndimage.sum(mask, label_im, range(nb_labels + 1))
    mask_size = sizes < threshold
    remove_pixel = mask_size[label_im]
    binary_two = copy(binary)
    binary_two[remove_pixel] = 0
    cluttermap = ~binary_two * binary
    return cluttermap


def erosion_and_propagation(test, mask=None, mode='filter'):
    binary = mode_binary(test, mode)
    eroded_square = ndimage.binary_erosion(binary)
    reconstruction = ndimage.binary_propagation(eroded_square, mask=binary)
    return ~reconstruction * binary


def opening_and_closing(test, mask=None, mode='filter'):
    binary = mode_binary(test, mode)
    open_img = ndimage.binary_opening(binary)
    close_img = ndimage.binary_closing(open_img)

    return close_img * ~binary



############################## ESPECIALES ###################################

def texture(matriz,half_wsize):
    wsize = 2*half_wsize + 1
    element = matriz.copy()
    shift = range(-half_wsize,half_wsize + 1)
    count = np.zeros_like(element)
    for i in shift:
        tmpa = np.roll(element, i, axis=0)
        for j in shift:
            tmpr = np.roll(tmpa, j, axis=1)
            count += abs(tmpr - element)
    count = count / wsize **2
    return count

def pseudo_gabella(matriz,half_wsize,thr):
```

```
    element = matriz.copy()
    shift = range(-half_wsize,half_wsize + 1)
    count = - np.ones_like(element)
    for i in shift:
        tmpa = np.roll(element, i, axis=0)
        for j in shift:
            tmpr = np.roll(tmpa, j, axis=1)
            count += (tmpr - element) > thr
    return count

def texture_eje(matriz,half_wsize, axis=0):
    wsize = 2*half_wsize + 1
    element = matriz.copy()
    shift = range(-half_wsize,half_wsize + 1)
    count = np.zeros_like(element)
    for i in shift:
        tmpa = np.roll(element, i, axis=axis)
        count += abs(tmpa - element)
    count = count / wsize
    return count

def pseudogabella_eje(matriz, half_wsize, axis=0, thr=6):
    element = matriz.copy()
    shift = range(-half_wsize,half_wsize + 1)
    count = -np.ones_like(element)
    for i in shift:
        tmpa = np.roll(element, i, axis=axis)
        count += (tmpa - element) < thr
    return count


def maximum(Matriz, selem):
    # Escalamos los datos originales hacia 0. y 1.
    tmp = Matriz.copy()
    cvalue = -31.5
    range = (31.5 + tmp.max())
    tmp = (tmp - cvalue)/ range
    # Máximo local
    result = skimage.filters.rank.maximum(tmp, selem=selem)
    # Regresamos al espacio original
    result = skimage.img_as_float(result)
    result = (result * range) + cvalue
    return result


def mitigation(data, Cluttermap, mask=False):
    data_z = wrl.trafo.idecibel(data)
```

```
    clutter_z = wrl.trafo.idecibel(Cluttermap)
    data_corr = data_z - clutter_z
    data_corr_dbz = wrl.trafo.decibel(data_corr)
    if mask:
        binimg = data_corr_dbz > Cluttermap.min()
        data_corr_dbz[~binimg] = -31.5
    return data_corr_dbz

def proyectar(data, r, azi, res, radius=100000., el=2, radar_location=(-80.638333,-
5.170278,56),
              ipol = wrl.ipol.Nearest, mask=False):
    "Función de proyección desde coordenadas rectangulares en (m y °) hacia una grilla polar
de resolución res "
    polargrid = np.meshgrid(r, azi)
    coords, rad = wrl.georef.spherical_to_xyz(polargrid[0], polargrid[1],el, radar_location)
    x = coords[..., 0]
    y = coords[..., 1]
    spacing = 2*radius/res
    xgrid = np.linspace(x.min(), x.max(), spacing)
    ygrid = np.linspace(y.min(), y.max(), spacing)
    grid_xy = np.meshgrid(xgrid, ygrid)
    grid_xy = np.vstack((grid_xy[0].ravel(), grid_xy[1].ravel())).transpose()
    xy=np.concatenate([x.ravel()[:,None],y.ravel()[:,None]], axis=1)
    gridded = wrl.comp.togrid(src=xy, trg=grid_xy, radius=100000.,
center=np.array([x.mean(), y.mean()]),
                      data=data.ravel(), interpol=ipol)
    gridded = np.ma.masked_invalid(gridded).reshape((len(xgrid), len(ygrid)))
    if mask :
        gridded = np.ma.masked_values(gridded, gridded.min())
    return xgrid, ygrid, gridded


def gridplot(figure, gridded, xgrid, ygrid, number=111, n = 0.1,
             label='Reflectividad [dBZ]', *args, **kwargs ):
    ax = figure.add_subplot(number, aspect="equal")
    pm = plt.pcolormesh(xgrid, ygrid, gridded,*args, **kwargs)
    cbar = plt.colorbar(pm, shrink=0.75)
    cbar.set_label(label)
    plt.title('Datos proyectados en la grilla')
    plt.xlabel("Easting (m)")
    plt.ylabel("Northing (m)")
    plt.xlim(n * min(xgrid), n * max(xgrid))
    plt.ylim(n * min(ygrid), n * max(ygrid))
    plt.grid(color="grey")
    return figure
```

Apéndice 5. Cuadernos de ejemplo

Lectura de datos	https://github.com/ElmerJeanpierreLopez/PIUXX/blob/master/Examples/00_Explorando%20datos%20de%20clutter.ipynb
Detección de clutter	https://github.com/ElmerJeanpierreLopez/PIUXX/blob/master/Examples/01_An%C3%A1lisis%20data%20clutter.ipynb
Detección de clutter	https://github.com/ElmerJeanpierreLopez/PIUXX/blob/master/Examples/03_Evaluaci%C3%B3n%20subtraction%20RainviewAnalyzer.ipynb
Detección de clutter	https://github.com/ElmerJeanpierreLopez/PIUXX/blob/master/Examples/04_Evaluaci%C3%B3n%20Flagging%20and%20Correction%20RainviewAnalyzer.ipynb
Interpolación	https://github.com/ElmerJeanpierreLopez/PIUXX/blob/master/Examples/05_M%C3%A9todos%20de%20interpolaci%C3%B3n.ipynb
Filtros espaciales	https://github.com/ElmerJeanpierreLopez/PIUXX/blob/master/Examples/06_Flagging%20espacial.ipynb
Clasificación de datos	https://github.com/ElmerJeanpierreLopez/PIUXX/blob/master/Examples/07_Clasificaci%C3%B3n%20de%20datos.ipynb
Proyección	https://github.com/ElmerJeanpierreLopez/PIUXX/blob/master/Examples/08_Resoluci%C3%B3n%20al%20proyectar.ipynb
Cadena de procesamiento	https://github.com/ElmerJeanpierreLopez/PIUXX/blob/master/Examples/10_Cadena%20de%20procesamiento.ipynb
Apantallamiento	https://github.com/ElmerJeanpierreLopez/PIUXX/blob/master/Examples/11_Bloqueo%20de%20haz.ipynb
Apantallamiento	https://github.com/ElmerJeanpierreLopez/PIUXX/blob/master/Examples/12_Beam%20blockage%20analysis.ipynb
Seguimiento	https://github.com/ElmerJeanpierreLopez/PIUXX/blob/master/Examples/13_Extracci%C3%B3n%20de%20caracter%C3%ADsticas.ipynb
Detección de clutter	https://github.com/ElmerJeanpierreLopez/PIUXX/blob/master/Examples/An%C3%A1lisis%20data%20precipitaci%C3%B3n.ipynb
Visualización	https://github.com/ElmerJeanpierreLopez/PIUXX/blob/master/Examples/An%C3%A1lisis%20data%20precipitaci%C3%B3n.ipynb