UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA

DESARROLLO DE UNA HERRAMIENTA COMPUTACIONAL BASADO EN SOFTWARE LIBRE PARA EL ANÁLISIS Y MEJORAMIENTO DE IMÁGENES SATELITALES CENTRADO EN EL PROBLEMA DE LA OCLUSIÓN UTILIZANDO REPRESENTACIÓN POCO DENSA DE SEÑALES SOBRE DICCIONARIOS REDUNDANTES.

Br. Junior Alexander Marquina Marquez

Mérida, Julio, 2016

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA

DESARROLLO DE UNA HERRAMIENTA COMPUTACIONAL BASADO EN SOFTWARE LIBRE PARA EL ANÁLISIS Y MEJORAMIENTO DE IMÁGENES SATELITALES USANDO REPRESENTACIÓN POCO DENSA DE SEÑALES SOBRE DICCIONARIOS REDUNDANTES

Trabajo de Grado presentado como requisito parcial para optar al título de Ingeniero Electricista

Br. Junior Alexander Marquina Marquez

Tutor: Prof. José Luis Paredes

Mérida, Julio, 2016

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA

DESARROLLO DE UNA HERRAMIENTA COMPUTACIONAL BASADO EN SOFTWARE LIBRE PARA EL ANÁLISIS Y MEJORAMIENTO DE IMÁGENES SATELITALES CENTRADO EN EL PROBLEMA DE LA OCLUSIÓN UTILIZANDO REPRESENTACIÓN POCO DENSA DE SEÑALES SOBRE DICCIONARIOS REDUNDANTES.

Br. Junior Alexander Marquina Marquez

Trabajo de Grado presentado en cumplimiento parcial de los requisitos exigidos para optar al título de Ingeniero Electricista, aprobado en nombre de la Universidad de Los Andes por el siguiente Jurado.

Prof. José Luis Paredes Q.
Tutor

Prof. Juan Marcos Ramirez
Jurado

Prof. Nelson Perez
Jurado

Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: La voluntad.

Albert Einstein

A mis padres y hermanos.

AGRADECIMIENTOS

A Jehová, nuestro padre celestial, por darme la vida y las capacidades intelectuales necesarias para asumir la responsabilidad que implica una formación académica en la Universidad.

A mi familia por acompañarme, apoyarme y animarme en este arduo camino. Gracias a ustedes y su ejemplo de perseverancia ante las dificultades de la vida, encontraba el entusiasmo necesario para seguir adelante. Este triunfo también es suyo, ¡sin ustedes no lo hubiera logrado!

A la Sra. Gloria Ovalles quien me abrió las puertas de su hogar durante todos estos años de formación académica y me brindó el cariño y atención de una madre. ¡Muchas gracias!

A la ilustre Universidad de Los Andes, en especial a la escuela de Ingeniería Eléctrica y a su personal docente altamente calificado, por su entrega al desarrollo de nuestras habilidades profesionales. Ustedes mantienen viva la esperanza de tener un mejor país.

Al profesor José Luis Paredes, por su ejemplo de constancia, sacrificio, excelencia y sobre todo humildad. Eres un modelo digno de admirar e imitar. Gracias por los valiosos concejos y tiempo me ha aportado durante la realización de este trabajo y mi formación académica.

Al ingeniero Jerick Órdenes, quien desde el inicio de esta investigación me aportó de manera humilde sus conocimientos en el área, además de valiosos concejos que permitieron el desarrollo de este trabajo.

Al Fondo Nacional de Ciencia, Tecnología e Innovación (Fonacit) por el financiamiento parcial de este trabajo de grado bajo el proyecto Nro. *2013001663* titulado: Procesamiento de las Imágenes Satelitales usando técnicas computacionales y de modelado inteligentes, para apuntalar la soberanía nacional.

Junior Alexander Marquina Marquez.

Junior Alexander Marquina Marquez. Desarrollo de una herramienta computacional basado en software libre para el análisis y mejoramiento de imágenes satelitales centrado en el problema de la oclusión utilizando representación poco densa de señales sobre diccionarios redundantes. Universidad de Los Andes. Tutor: Prof. José Luis Paredes Q. Julio, 2016.

Resumen

Con la puesta en órbita del satélite Miranda se crea en el país una abanico de posibilidades de investigación en el campo del procesamiento digital de imágenes satelitales. Este proyecto se ha centrado en el estudio de métodos de representación poco densa aplicados a la restauración de imágenes satelitales debido a pérdida de píxeles atribuidos a oclusiones causadas por nubes y la implementación de una serie de rutinas computacionales en un lenguaje de programación de libre distribución. Para lograr este cometido se utilizó la teoría de sensado comprimido para modelar la oclusión a través de una matriz de medición, con la cual se puede encontrar la representación poco densa sobre un diccionario holográfico, que no es más que la proyección de un diccionario base sobre la matriz de medición. Finalmente, se utiliza la representación dispersa encontrada para estimar la imagen sin la oclusión al proyectar dichos coeficientes sobre el diccionario base. Se utilizan diccionarios redundantes basados en la transformada discreta del coseno, así como diccionarios aprendidos y adaptados a la estadística de las imágenes a representar. Para adaptar los diccionarios se utilizó el algoritmo K-times Singular Value Descomposition (K-SVD), mientras que para encontrar la representación dispersa se empleó el popular algoritmo de búsqueda voraz conocido como Ortogonal Matching Pursuit (OMP). Además, se diseña un mecanismo iterativo de reconstrucción para atacar oclusiones más severas.

Descriptores: Satélite Miranda, representación poco densa, desoclusión de imágenes, inpainting, K-SVD, OMP, Sensado Comprimido, Reconstrucción Iterativa.

.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El 28 de septiembre del 2012, se pone en órbita el primer satélite de percepción remota del país. El objetivo principal de esa inversión es de disponer de imágenes satelitales destinadas a apoyar la toma de decisiones, a nivel gubernamental, en áreas estratégicas como la planificación urbana y agrícola, seguridad alimentaria, gestión de recursos naturales, vigilancia de fronteras, gestión de desastres, entre otros. Este gran avance pone a Venezuela entre los países del mundo capaces de generar este tipo de imágenes, lo que abre un abanico de posibilidades de desarrollo científico en áreas distintas a las planteadas líneas arriba, que son inherentes al hecho de poseer este tipo de herramientas tecnológicas. En este sentido, se pueden profundizar en áreas como el procesamiento digital de imágenes satelitales que permitan su análisis y faciliten su aplicación en las actividades para las cuales fue pensado.

Debido al proceso de captura de las imágenes satelitales, éstas suelen estar afectadas por las condiciones atmosféricas de la región de estudio que pueden impedir el monitoreo de áreas de interés. Por causa de la posición geográfica de Venezuela, dentro de la zona de convergencia intertropical, la presencia de nubes en las escenas capturadas por el satélite siempre será común en mayor o menor medida. Este problema, denominado oclusión u obstrucción por nubes, afecta la calidad y utilidad de las imágenes tomadas desde el espacio, ya que ocasiona pérdida de información de interés. La solución de este problema obtiene gran relevancia cuando se analizan los costos de captura de las imágenes, teniendo en cuenta que el satélite representó una gran inversión para el país y su vida útil es de sólo 5 años.

Aunque en Venezuela éste es un campo nuevo de estudio, a nivel internacional se han propuesto una variedad de técnicas para recuperar la información obstruida por las condiciones atmosféricas. De todas estos enfoques, en esta investigación se explorará la aplicación de técnicas de reconstrucción de imágenes basada en la teoría de representación poco densa de señales. Esta teoría es relativamente nueva en el área de procesamiento

digital de señales, pero ha venido cogiendo auge en los últimos años. Consiste en aproximar una señal con mínimo error usando pocos elementos de un conjunto de bases denominado diccionario, conduciendo al siguiente problema de optimización:

$$\mathbf{y} = \mathbf{A}\mathbf{x} \quad \text{sujeto a} \quad \|\mathbf{x}\|_{\ell_0} \leqslant T_0. \tag{1}$$

Donde $\mathbf{y}$ es la señal objetivo, $\mathbf{A}$ el diccionario que agrupa las señales base conocidas como átomos y $\mathbf{x}$ la representación poco densa de la señal $\mathbf{y}$ en $\mathbf{A}$.

La motivación de enfocar el problema desde esta perspectiva nace debido a que en recientes investigaciones se ha aplicado esta teoría a nivel local para abordar problemas en el tratamiento digital de imágenes. Por ejemplo, en el 2012, Órdenes en [1], aplicó esta teoría a la restauración de imágenes comunes deterioradas por pérdidas de píxeles, obteniendo resultados satisfactorios. Además, más recientemente, se aplicaron estas técnicas para abordar problemas aplicados a imágenes satelitales. Por ejemplo, García en [2], enfoca el problema de detección de objetivos en imágenes satelitales resolviendo un problema basado en la representación dispersa de señales obteniendo un buen desempeño. Por otro lado, Portillo en [3] se basa en la representación poco densa de señales para aumentar la resolución espacial de una imagen satelital. Por esta razón, se espera que esto pueda extenderse al problema de restauración de imágenes ocluidas por nubes.

En otro orden de ideas, la mayoría de las técnicas de reconstrucción implementadas vía *software*, están desarrolladas en plataformas privativas como MATLAB®, lo cual puede limitar el desarrollo futuro. Como alternativa a los *software* privativos, recientemente esta creciendo el desarrollo de plataformas de cómputo en softaware libre. Esto tiene grandes ventajas en cuanto al desarrollo tecnológico se refiere, como el de disponer de código libre desarrollado por una amplia comunidad y en la que los costos de utilización son nulos. Por lo que esta opción es más atractiva para los propósitos de este proyecto.

Por todo lo expuesto, en este trabajo de grado se propone desarrollar una herramienta computacional basada en software libre para el mejoramiento de imágenes satelitales, centrado en el problema de la oclusión, con el fin de sembrar una iniciativa en la fértil y nueva industria de la observación terrestre, que impulse futuros desarrollos en este campo hechos en Venezuela, con el fin de afianzar el avance tecnológico en esta área. Además, contribuir

en el ámbito académico por todo lo que implica el desarrollo de herramientas tecnológicas a nivel local. Para lograr esta ambiciosa tarea se plantean los siguientes objetivos:

- Estudiar los fundamentos básicos del proceso de adquisición de imágenes por el satélite Miranda.
- Estudiar los enfoques generales utilizados para la reconstrucción de imágenes satelitales ocluidas por nubes.
- Estudiar la teoría de representación en diccionarios redundantes y aplicarlo a imágenes.
- Adaptar la teoría de representación poco densa al problema de la oclusión de imágenes satelitales.
- Implementar los algoritmos más adecuados usando herramientas de *software* libre.
- Integrar el módulo desarrollado en una herramienta de visualización de imágenes satelitales.

A fin de dar cumplimiento a los objetivos planteados, este trabajo se organizó en cuatro capítulos. El Capítulo 1 introduce brevemente los aspectos técnicos fundamentales del satélite Miranda, seguidamente se definirá el proceso de reconstrucción de oclusiones y se presentará una reseña de las técnicas generalmente utilizadas para eliminar las obstrucciones de las imágenes y, por último, se acotará el alcance de este trabajo. En el Capítulo 2 se hará una exposición introductoria sobre los fundamentos básicos de la teoría de representación dispersa de señales, particularizadas a las imágenes. Se describirá uno de los algoritmos más utilizados para encontrar la representación poco densa de señales y la metodología y nociones básicas del entrenamiento de diccionarios basados en el algoritmo KSVD, como método seleccionado para construir diccionarios adaptados al tipo de imágenes que se desea reconstruir. Siguiendo el orden, en el Capítulo 3 se adapta la teoría de representación poco densa al problema de la oclusión, introduciendo un modelo y las metodologías diseñadas durante el trabajo para reconstruir imágenes satelitales. También se justifica el uso del lenguaje de programación Python para la implementación de las metodologías. Posteriormente, el Capítulo 4 corresponde a la validación experimental de las

metodologías propuestas. Para lograr esto se presentan distintos experimentos controlados utilizando diferentes escenas de una imagen del Satélite Miranda, con el objetivo de evaluar su aplicabilidad en problemas prácticos. Finalmente, se presentan algunas conclusiones y recomendaciones basadas en el desempeño de cada metodología usada.

CAPÍTULO 1
MARCO TEÓRICO

El Capítulo iniciará con el estudio introductorio de algunas de las características técnicas relacionadas con las imágenes captadas por el Satélite Miranda. Seguidamente, se introducirán los conceptos básicos relacionados con el problema de la oclusión y se contextualizará al proceso de captura del Satélite Miranda. Finalmente, se presentará un análisis introductorio de los distintos enfoques utilizados para obtener imágenes limpias de oclusiones, de acuerdo a tres características inherentes a las imágenes satelitales, como son la resolución temporal, espectral y espacial. En este camino, se justificará el enfoque escogido en esta investigación para reconstruir imágenes obstruidas por fenómenos atmosféricos.

1.1 IMÁGENES SATELITALES

Desde el punto de vista de la percepción remota, una imagen satelital es una representación visual de la radiación electromagnética, reflejada por la superficie de la tierra, que captura un sensor a bordo de un satélite artificial. El proceso de recepción de la energía reflejada va a depender del tipo de tecnología con la que cuente el satélite. Para satélites pasivos, como el Satélite VRSS-1(*Venezuelan Remote Sensing Satellite-1*), las imágenes deben ser adquiridas bajo condiciones uniformes de iluminación, con el objeto de que la radiación recibida por los sensores reflejen las características de la superficie de estudio en lugar de los cambios de la fuente de iluminación u otro aspecto relacionado con la observación. Por esto, el Satélite Miranda fue diseñado para adquirir imágenes a las 10:30 a.m., lo que permite obtener un balance óptimo entre la iluminación ideal y mínima cobertura de nubes.

Tabla 1.1. Detalles de la imagen pancromática

Tipo de escaneado	*pushbroom*
Bandas espectrales	PAN: (0,45 - 0,90) μm
Resolución espacial	2,5 m/píxel
Resolución radiométrica	10 bits
Ancho de la imagen	30 km
Detector CCD	12000 píxeles de 10 µm x 10 µm
SNR	48 dB (ángulo solar respecto al *Cenit* 70°, superfice de albedo[1]0,65)

[1] Porcentaje de radiación que cualquier superficie refleja respecto a la radiación que incide sobre la misma.

1.2 IMÁGENES CAPTURADAS POR EL SATÉLITE

El satélite VRSS-1, conocido como satélite "Francisco de Miranda", cuenta con dos tipos cámaras para obtener imágenes del territorio nacional. La cámara PMC (*Panchromatic and Multispectral Camera*), con la cual se pueden captar imágenes pancromáticas y multiespectrales, con una resolución espacial de $2,5$ m/píxel para imágenes pancromáticas y una resolución espacial de 10 m/píxel para imágenes multiespectrales. Por otro lado, la cámara WMC (*Wide-swath Cameras*), con las que sólo se pueden generar fotografías multiespectrales de resolución espacial intermedia. Para mayor detalle en cuanto al proceso de adquisición y las características de las cámaras se sugiere revisar la referencia [2]. En las Tablas 1.1 y 1.2 [4] se encuentran resumidas las características técnicas de ambas cámaras que son relevantes para esta investigación. Finalmente, los productos obtenidos del proceso de adquisición son los siguientes:

- Imágenes pancromáticas captadas por la cámara PMC.
- Imágenes multiespectrales captadas por la cámara PMC.
- Imágenes multiespectrales captadas por la cámara WMC.

Tabla 1.2. Detalles de las imágenes multiespectrales.

Cámara utlizada	PMC	WMC
Tipo de escaneado	*pushbroom*	*pushbroom*
Bandas espectrales	Banda 1: Azul (0,45-0,52) μm Banda 2: Verde (0,52-0,59) μm Banda 3: Rojo (0,63-0,69) μm Banda 4: NIR[1](0,77-0,89) μm	Banda 1: Azul (0,45-0,52) μm Banda 2: Verde (0,52-0,59) μm Banda 3: Rojo (0,63-0,69) μm Banda 4: NIR (0,77-0,89) μm
Resolución espacial	10 m/píxel	16 m/píxel
Resolución radiométrica	10 bits	10 bits
Ancho de la imagen	30 km	192 km
Detector CCD	3000 píxeles de 40 µm x 40 µm	12000 píxeles de 6,5 µm x 6,5 µm
SNR	48 dB (ángulo solar respecto al *Cenit* 70°, superfice de albedo 0,65)	20 dB (ángulo solar respecto al *Cenit* 15°, superfice de albedo 0,65) 46 dB (ángulo solar respecto al *Cenit* 70°, superfice de albedo 0,65)

[1] Infrarrojo cercano (*del inglés, Near Infrared*).

1.3 RESOLUCIÓN DE UNA IMAGEN SATELITAL

La radiación captada de la superficie terrestre es un fenómeno continuo de cuatro dimensiones (espacio, tiempo, longitud de onda y radiancia), por lo que los sensores presentes en cada una de las cámaras deben muestrear cada una de estas variables con el objeto de obtener la representación de la escena. El modo en que este proceso de discretización se lleva a cabo define cuatro tipos de resolución.

1.3.1 *Resolución Espacial*

Se refiere al área espacial representada por un pixel en la imagen. Cuanto menor sea, mayor calidad de detalles se podrá percibir en una imagen obtenida. Depende de la altura

del sensor con respecto a la Tierra, el ángulo de visión, la velocidad de escaneado y las características ópticas del sensor. En el caso particular del VRSS-1, cada uno de los productos tiene asociada una resolución espacial. Por ejemplo, como puede observarse en la Tabla 1.1, para imágenes PAN el nivel de detalle espacial es de 2,5 m/píxel; esto indica que el área que representa el valor discreto de la radiancia del pixel es de 2,5 m x 2,5 m.

1.3.2 *Resolución Espectral*

Los sensores a bordo del satélite se diseñan para captar radiaciones en determinado rango de valores de longitudes de onda. Dependiendo del uso que se le dará al satélite, pueden seleccionarse sensores con bandas relativamente estrechas o anchas. Sin embargo, si las bandas son muy anchas, el sensor registrará un valor promedio de la radiancia en el rango de longitudes de onda del canal, que podría ocultar información útil. Por tanto, cuanta más bandas incluya un sensor se podrá caracterizar mejor la superficie observada. La resolución espectral se refiere al número y ancho de los canales espectrales que el sensor registra. El número de bandas y su localización en el espectro va a depender de los objetivos que se pretendan cubrir con la puesta en órbita del satélite. En el caso particular del Satélite Miranda, las cámaras capturan parte del espectro correspondiente a longitudes de ondas visibles y el infrarrojo cercano. En la Tabla 1.2 se muestra el rango de longitudes de onda captado por cada banda espectral.

1.3.3 *Resolución Radiométrica*

Se refiere a la cantidad de niveles digitales (o niveles de gris) en la que es discretizada la radiación recibida por el sensor. Esto depende del conversor análogo-digital utilizado en el proceso de muestreo. Cuanto mayor es el número de niveles de gris utilizado para representar la variación de la radiancia en el pixel, mayor detalle podrá percibirse en una imagen dada. El número de niveles de gris se expresa comúnmente en términos de dígitos binarios (bits) necesarios para almacenar el valor del nivel máximo de radiancia. Por ejemplo, el VRSS-1 cuenta con una resolución radiométrica de 10 bits para todos los productos que pueden obtenerse. Es decir, las imágenes captadas se representan 1024 niveles digitales por banda. Esto representa una excelente resolución, si se tiene en cuenta

que el ojo humano y los dispositivos de visualización comunes sólo pueden manejar 256 niveles (8 bits de resolución).

1.3.4 *Resolución Temporal*

Es una medida de la frecuencia a la cual el satélite puede obtener imágenes en una misma área. Esto está estrechamente relacionado con el tipo de órbita del satélite, así como el ancho del área cubierta por las cámaras. En el caso del satélite Miranda, la resolución temporal para el conjunto de cámaras PMC es de 57 días y para las WMC es de 12 días. Sin embargo, el satélite está equipado con un control de maniobra que permite variar el ángulo de observación del mismo, con lo que pueden disminuirse las resoluciones temporales a 4 y 3 días, respectivamente. Para mayor detalle sobre este aspecto técnico, se sugiere revisar la referencia [2].

1.4 OCLUSIÓN EN IMÁGENES SATELITALES

La oclusión en una imagen se puede definir como la pérdida de información visual, causada por algún objeto o fenómeno que obstruya la escena que se desea capturar. En el contexto de la teledetección, esto puede incluir cualquier anomalía en los sensores, que impida que éstos capten la energía reflejada por la superficie de la tierra. Sin embargo, el caso más directo que incluye esta definición es el fenómeno de la obstrucción por parte de las nubes a la captura de la energía electromagnética proveniente de las zonas ocultas por las mismas.

Como ya se mencionó, la hora de captura de imágenes del VRSS-1 fue diseñada para garantizar una mínima cobertura de nubes. No obstante, las escenas aún estarán frecuentemente contaminadas por nubosidades en mayor o menor grado. La solución de este problema obtiene gran relevancia cuando se analizan los costos de captura de las imágenes, teniendo en cuenta de que la vida útil del satélite es limitada. Por esta razón, se plantea una solución de post-procesamiento vía software, que permita editar la imagen capturada, para eliminar la obstrucción originada por las condiciones atmosféricas y estimar la información oculta por éstas. Al proceso de eliminar la obstrucción de las nubes puede ser definido como desoclusión, aunque ya ha recibido varias denominaciones en la litera-

tura científica internacional, entre ellas, remoción de nubes (*del inglés, Cloud Removal*) o liberación de nubosidades (*del inglés, cloud-free*). En este trabajo se utilizará la primera definición.

En general, la desoclusión implica dos etapas básicas:

- Detección de la Oclusión: En esta etapa se localizan las zonas contaminadas por las nubes. Este proceso puede ser manual, semiautomático o completamente automático. Por lo general, esto se realiza de manera separada e implica el uso de otras herramientas de procesamiento de imágenes. Ejemplos de técnicas de detección de nubes se desarrollaron en [5–8]. Como resultado final, se obtiene una máscara que delimitará la región a reconstruir. En la Figura 1.1, se muestra el resultado obtenido de aplicar la metodología señalada en [7].

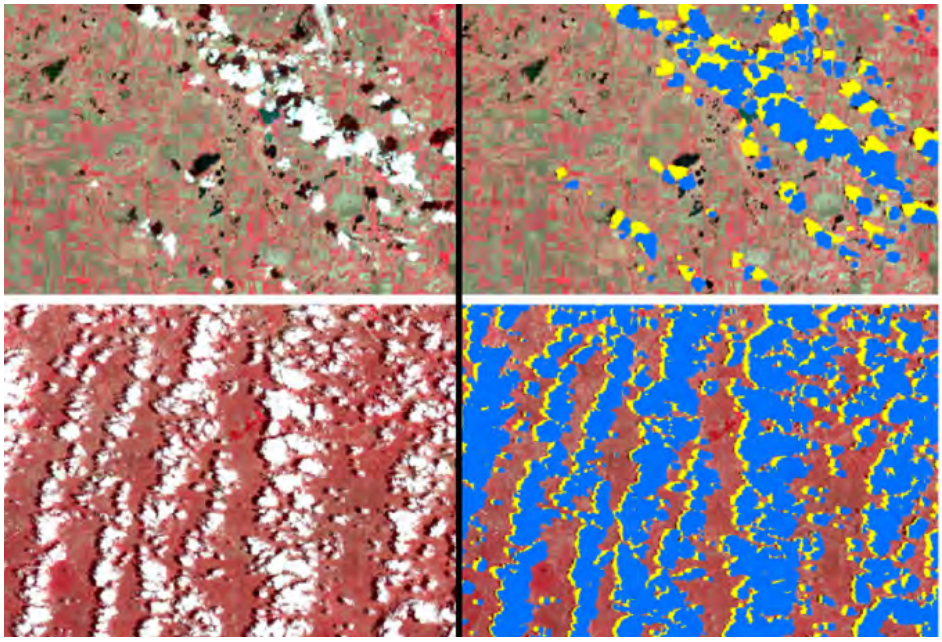


Figura 1.1. Detección automática de nubes y sus sombras. A la izquierda se encuentran las imágenes de entrada al algoritmo de detección. A la derecha se muestran las máscaras generadas por la detección de nubes (en azul) y las sombras producidas por éstas (en amarillo).

- Reconstrucción de las regiones perdidas: En esta etapa se estiman los píxeles que se utilizarán para rellenar las áreas con información perdida, producto de la oclusión. Para resolver este problema también existe un amplio abanico de posibilidades según

el enfoque de estimación escogido.

Esta investigación se centrará en la segunda etapa del proceso. A continuación se hará un breve resumen acerca de las técnicas o enfoques que se han utilizado hasta el momento para atacar el problema de pérdida de información por causa de oclusiones.

1.5 RECONSTRUCCIÓN DE IMÁGENES SATELITALES

El propósito central de esta etapa es rellenar las regiones con información perdida (delimitada por la máscara), de tal forma que sea visualmente coherente con el resto de la imagen. Hasta la fecha, no hay ningún algoritmo matemático capaz de crear información que no ha sido capturada en absoluto. Por lo tanto, las oclusiones no pueden reconstruirse sin un conocimiento previo de lo que puede existir en la zona oculta por la nube. Por esta razón, en el proceso de reconstrucción se explota alguna información presente en áreas libres de nubes para tratar de estimar lo que éstas ocultaban. Existen por lo menos, tres grandes enfoques que buscan resolver este problema, en las que el método de estimación adoptado está bien diferenciado.

1.5.1 *Reconstrucción Basada en la correlación Temporal*

La primera categoría enfoca el problema de estimación en la correlación multi-temporal de imágenes de una misma escena. En síntesis, esta metodología consiste en la selección de la mejor medición entre un conjunto de medidas adquiridas en un período de tiempo limitado para representar el píxel multi-temporal considerado durante ese período de tiempo. Este enfoque está fundamentado en la idea de que, si el período de tiempo entre las tomas es corto, la escena que estaba oculta no ha cambiado, por lo que puede utilizarse dicha información para reconstruir las regiones perdidas de la imagen. De acuerdo al modo en que se estiman las relaciones espacio-temporales del conjunto de imágenes y a cómo se reconstruye la zona ocluida, existe una gran variedad de métodos propuestos. Por ejemplo, Melgani en [9] propone un proceso de predicción contextual, basado en máquinas de soporte vectorial y predictores lineales, con el fin de determinar las relaciones temporales

entre el conjunto de imágenes y la que se desea reconstruir.

Más recientemente, Lin y col. en [10] proponen explotar la correlación entre imágenes de la misma escena, pero en lugar de efectuar la reconstrucción pixel a pixel utilizado en [9], utilizan un enfoque basado en parches (bloques de imagen) libres de nubes. Teniendo en cuenta la precisión radiométrica, los parches libres de nubes se eligen basándose en la similitud entre bloques. Para ello, utilizan dos métricas de similaridad, el MSE (del inglés, *Mean Square Error*), para medir las diferencias entre píxeles, y el SSIM (del inglés, *Structural Similarity Index*), con el que se comprueba la similaridad entre bloques. Además, añaden un umbral para descartar parches que excedan un nivel de nubosidad predeterminado. Luego, la información de los parches disponibles para la reconstrucción son analizados utilizando la ecuación de Poisson y resuelta por un proceso de optimización global. La Figura 1.2 muestra las etapas utilizadas por esta metodología para obtener la reconstrucción final.

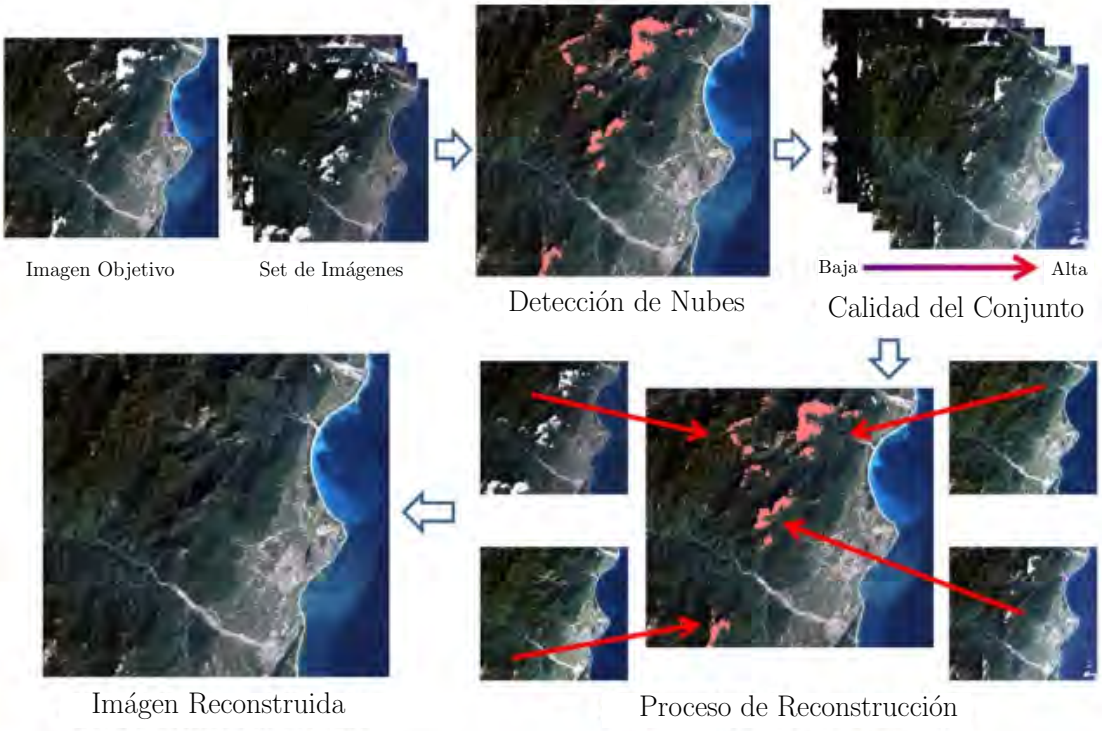


Figura 1.2. Reconstrucción basado en la correlación temporal entre imágenes de una misma área. Adaptado de [10].

1.5.2 *Reconstrucción Basada en la correlación Espectral*

Un segundo enfoque explota la relación espectral entre píxeles de una misma escena. La mayoría de las técnicas agrupadas en este enfoque, utilizan la información proveniente de una banda, en la que los efectos atmosféricos no presenten interferencia, para recuperar la banda contaminada. Esto se logra, mediante el modelado de la relación espectral entre cada una de ellas. Ejemplos de este tipo de tratamiento se desarrollaron en [11–13].

1.5.3 *Reconstrucción Basada en la correlación Espacial*

La tercera opción agrupa a las técnicas que tratan de estimar las regiones perdidas utilizando la información presente en el resto de la imagen libre de contaminación. Estos enfoques se acercan más a las técnicas tradicionales de reconstrucción de imágenes en general. El proceso de reconstrucción de una imagen digital fue inicialmente relacionado por Bertmalio y col. en [14] con el trabajo de restauración de cuadros artísticos antiguos. En dicho artículo, los autores llaman a este proceso *inpainting*, e incluyen a la desoclusión como un problema particular de esta área de la investigación.

Una imagen digital se puede modelar como una composición de dos partes [15]:

- **Estructura:** Se refiere a las regiones homogéneas de la imagen y contienen límites bien definidos. A estos se les denomina bordes y designa a las transiciones entre dos regiones de niveles de gris significativamente distintos.
- **Texturas:** Se puede definir como la repetición de un patrón espacial básico, cuya estructura puede ser periódica, o parcialmente periódica (aleatoria). Además, la textura de una superficie hace referencia a la distribución de valores de intensidad a nivel espacial. En esta investigación, relacionaremos la textura con el nivel de detalle presente en la imagen.

Como ejemplo ilustrativo, en la Figura 1.3 se resaltan las áreas homogéneas, en azul y las zonas que contienen texturas, en verde, de una imagen satelital pancromática tomada sobre el puerto de La Guaira.

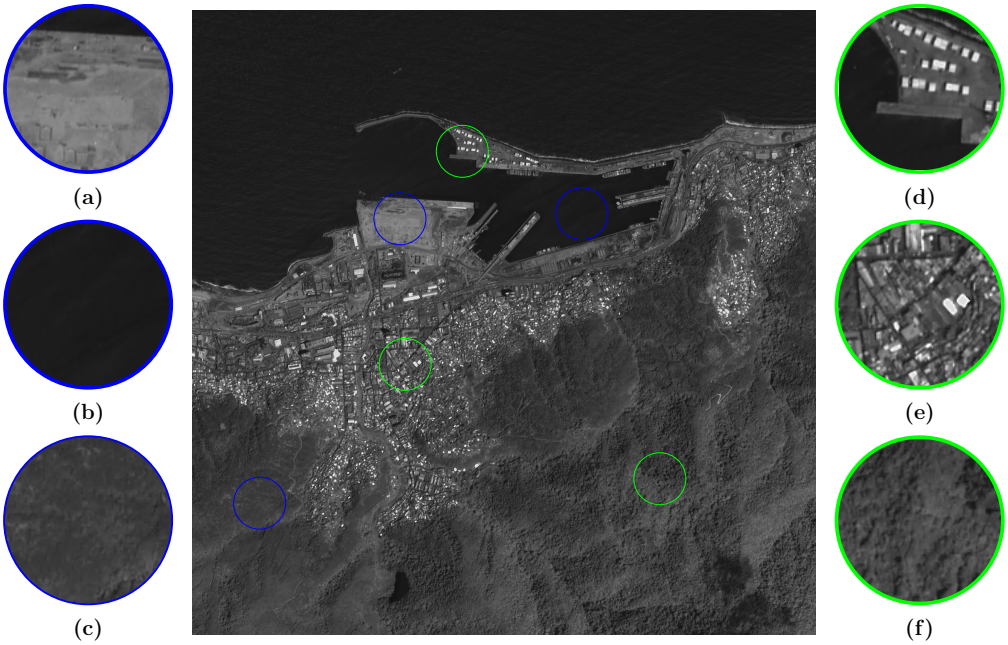


Figura 1.3. Ejemplo comparativo entre imágenes homogéneas y con texturas. (a)-(c) Regiones en la imagen con comportamiento homogéneo. (d)-(f) Zonas de la imagen con textura.

En base a estas características de la imagen, los enfoques tradicionales han tenido buenos resultado ante uno u otro componente de la imagen. Por ejemplo, los métodos basados en la difusión o propagación de información local, tienen mejor resultado ante imágenes homogéneas. En esta vertiente, Ballester y col. en [16] propagan los niveles de gris (isofotas) utilizando ecuaciones diferenciales parciales (PDE, del inglés *Partial Differential Equation*), para encontrar las direcciones de menor cambio, con el que se interpolarán o difuminarán los isofotas encontrados en las vecindades de la región perdida. Otras estrategias que siguen este enfoque de reconstrucción fueron desarrollados en [14, 17–19]. Como ya se mencionó, estos métodos tienen un buen desempeño reconstruyendo imágenes estructuradas, pero si la región ocluida es muy grande, tiende a producir un efecto de desenfoco y causar pérdidas de detalles.

Por otro lado, está el grupo de metodologías que se enfocan en reconstruir imágenes con mucha texturas o detalles, en los que una reconstrucción basada en PDE u otro método difuso, tendría resultados pobres. La idea central de esta estrategia es suponer que la información de textura que se utilizará para reconstruir el área ocluida se encuentra en otra región de la imagen sin contaminación. Entonces, los algoritmos se enfocan en

buscar el mejor parche, en zonas similares dentro de la imagen (estadística similar), que se replicará en la región ocluida. Ejemplos de éstos se encuentran en [20–23].

Por último, las imágenes naturales se componen de estructuras y texturas, por lo que otro conjunto de algoritmos combinan ambos enfoque de reconstrucción descritos. Por ejemplo, Bertalmio y col. en [24] descomponen la imagen original, en sub-imágenes de textura y estructura, para realizar la reconstrucción de cada componente por separado. Finalmente, se combinan los resultados para obtener el *inpainting* final.

Puesto que para aplicar esta metodología no hace falta información adicional de una misma escena (como otra imagen capturada en un tiempo posterior o bandas espectrales adicionales), esta opción es más atractiva para los propósitos de esta investigación debido a las características del Satélite Miranda. Algunas de estas técnicas tradicionales se han aplicado a imágenes satelitales. Por ejemplo en [25], Lorenzi y col., proponen adaptar una de las metodologías de reconstrucción tradicionales mencionadas anteriormente a imágenes de sensado remoto de alta resolución. Más recientemente, DL Sophia y col. en [26] aplican técnicas similares a la reconstrucción de imágenes satelitales ocluidas por nubes.

CAPÍTULO 2
REPRESENTACIÓN POCO DENSA DE SEÑALES EN DICCIONARIOS REDUNDANTES

Recientemente, la teoría de representación poco densa de señales, a través de conceptos relacionados con la teoría de Sensado Comprimido, se ha constituido en una opción robusta para la reconstrucción de imágenes. Por ejemplo, en [1] se expusieron técnicas basadas en esta teoría para restauración de imágenes comunes, deterioradas por un proceso de pérdidas de píxeles causado por oclusión de imágenes enmascaradas por texto. Como se mencionó en el capítulo anterior, la desoclusión de imágenes satelitales está relacionada con las técnicas de restauración de imágenes en general. En este capítulo se hará una exposición introductoria sobre los fundamentos básicos de la teoría de representación dispersa de señales, particularizadas a las imágenes. Se describirá uno de los algoritmos más utilizados para encontrar la representación poco densa de señales y la metodología y nociones básicas del entrenamiento de diccionarios basados en el algoritmo KSVD (del inglés, K-times Singular Value Decomposition), como método seleccionado para construir diccionarios adaptadas al tipo de imágenes que se desea reconstruir. Finalmente, se relacionarán estos conceptos con la representación poco densa de imágenes satelitales.

2.1 REPRESENTACIÓN POCO DENSA DE SEÑALES

Considere el sistema de ecuaciones lineales, representado matricialmente como

$$\mathbf{y} = \mathbf{A}\mathbf{x}. \tag{2.1}$$

Donde:

- $\mathbf{y}$ es una señal discreta en algún dominio, de dimensión n, es decir, $\mathbf{y} \in \mathbb{R}^n$.
- $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_K] \in \mathbb{R}^{n \times K}$, es el diccionario, cuyas columnas $\{\mathbf{a}_i\}_{i=1}^K \in \mathbb{R}^n$, normalizadas bajo la norma ℓ_2, definen las señales elementales o átomos del mismo. Dicho diccionario es de rango completo y redundante, es decir, con $K > n$.
- $\mathbf{x} = [x_1, x_2, \ldots, x_K] \in \mathbb{R}^{K \times 1}$ un vector de coeficientes en el nuevo dominio.

Otra representación equivalente de (2.1), utilizando el enfoque columnas, es

$$\mathbf{y} = \sum_{i=1}^{K} x_i \mathbf{a}_i. \tag{2.2}$$

De (2.2), se puede concluir que cualquier señal $\mathbf{y}$, en el espacio vectorial n, puede ser representada como una combinación lineal de K vectores $\mathbf{a}_i$, representados en el mismo espacio.

La teoría de representación poco densa[1] se basa en el hecho de que la mayoría de señales en aplicaciones prácticas pueden ser representadas o aproximadas por la combinación lineal de un grupo reducido de formas de onda elementales. En otras palabras, si $\mathbf{y}$ puede ser representada por la combinación lineal de unos pocos átomos en $\mathbf{A}$, ha de existir un conjunto reducido de elementos no nulos en $\mathbf{x}$ de tal forma que satisfaga (2.2). Gracias a esta novedosa técnica, ciertas características de una señal pueden ser resaltadas a través de su representación en formas de onda elementales pertenecientes al diccionario.

Dado que el sistema descrito en (2.1) es subdeterminado, existen infinitas soluciones posibles para $\mathbf{x}$. Sin embargo, J.Tropp [27], explica que en problemas de representación poco densa se busca una solución a través de un método de bajo costo computacional. Por lo que una solución aproximada es aceptada siempre que cumpla con cierto margen de desviación ε, y además garantice representaciones lo suficientemente dispersas. Esto se justifica debido a que, en ambientes reales, las señales siempre estarán contaminadas con ruido, causando que una solución exacta sea inviable. Así, el proceso se resume a encontrar el vector $\mathbf{x}$ tal que su proyección en el diccionario $\mathbf{A}$, produzca una señal cercana a $\mathbf{y}$, es decir

[1]La denominación señal poco densa es equivalente a *señal con coeficientes dispersos*, y se traduce del inglés *sparse signal*.

$$\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_{\ell_p} \leqslant \varepsilon. \tag{2.3}$$

Usualmente, las normas empleadas en métodos de aproximación son las normas ℓ_p con $p = \{0, 1, 2\,,\, \infty\}$, la cual se define como

$$\|\boldsymbol{\gamma}\|_{\ell_p} = \left(\sum_{i=1}^{N} |\boldsymbol{\gamma}_i|^p\right)^{1/p}. \tag{2.4}$$

Para el caso en que $p = 0$, la norma l_o cuenta el número de componentes en $\boldsymbol{\gamma}$ que son diferentes de cero. Por otro lado, para $p = \infty$, la norma l_∞ se reduce a determinar el valor máximo entre las componentes del vector $\boldsymbol{\gamma}$. Entonces, como el problema descrito por la ecuación 2.3 tiene infinitas soluciones, se aborda el problema desde el punto de vista de la optimización. Bajo este criterio, se busca restringir el dominio de soluciones garantizando que la representación en el diccionario sea lo suficientemente dispersa, es decir

$$\arg\min_{\mathbf{x}} \|\mathbf{x}\|_{\ell_0} \quad \text{sujeto a} \quad \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_{\ell_2} \leqslant \varepsilon. \tag{2.5}$$

Al problema de optimización anterior se le conoce como la *aproximación restringida por error*, donde ε representa el margen de desviación entre la señal y su aproximación. De forma alterna, el problema en (2.5) puede interpretarse como la minimización del error para un cierto nivel de densidad, mejor conocida como *aproximación restringida por poca densidad*, es decir

$$\arg\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_{\ell_2} \quad \text{sujeto a} \quad \|\mathbf{x}\|_{\ell_0} \leqslant T_0, \tag{2.6}$$

donde T_0 representa el número máximo de átomos utilizados para lograr la mejor representación de $\mathbf{y}$. Los problemas presentados en (2.5) y (2.6) son equivalentes, siempre y cuando se elijan adecuadamente los parámetros ε y T_0.

2.2 ALGORITMO ORTHOGONAL MATCHING PURSUIT PARA REPRESENTACIÓN POCO DENSA DE SEÑALES

Uno de los algoritmos más populares para resolver los problemas de optimización presentados en (2.5) y (2.6), es el algoritmo *Orthogonal Matching Pursuit* (OMP). Esto se debe a que presenta un balance entre errores pequeños en la reconstrucción y tiempo de ejecución bajos, siendo computacionalmente eficiente. En la Tabla 2.1 se muestra un resumen adaptado del algoritmo descrito en [28].

La estrategia básica seguida por este algoritmo es estimar el soporte[2] de la representación poco densa $\mathbf{x}$ y luego calcular un estimado de la amplitud del mismo para cada iteración, hasta que se cumpla un criterio de convergencia. Como puede observarse en el algoritmo descrito en la Tabla 2.1, el índice del átomo más correlacionado k, en la iteración t, se obtiene maximizando la proyección entre las columnas de $\mathbf{A}$ y el vector residual $\mathbf{r}$. Por otro lado, la amplitud del soporte es estimada minimizando la norma ℓ_2 del residuo. Esta restricción de mínimos cuadrados ortogonaliza el residuo, con respecto a la aproximación obtenida, evitando de esta manera que se vuelva a elegir el mismo átomo en próximas iteraciones.

En esta investigación, al utilizar este algoritmo de representación, se preferirá la solución basada en la restricción por poca densidad descrita por la ecuación (2.6). Además, al parámetro T_o se le denominará en las secciones siguientes como K_{omp}, y está relacionado con el número de átomos del diccionario que se utilizará para obtener la representación en el dominio poco denso. Es de destacar, que aunque se desea que la solución sea lo más dispersa posible, en algunas ocasiones, utilizar más átomos para encontrar la representación puede mejorar la calidad de la reconstrucción obtenida. Esto se profundizará en el siguiente capítulo.

[2]El soporte se refiere a el conjunto de indices que pertenecen a las entradas no nulas de un vector.

Tabla 2.1. Algoritmo Orthogonal Matching Pursuit.

ENTRADA	• Diccionario $\mathbf{A} \in \mathbb{R}^{n \times m}$ • Señal objetivo $\mathbf{y} \in \mathbb{R}^n$ • Para resolver por poca densidad se define el nivel T_o • Para resolver por error se define el error permitido ϵ
INICIALIZACIÓN	• Conjunto de índices $\Lambda^{(0)} = \varnothing$ • Residuo $\mathbf{r} = \mathbf{y}$ • Vector de coeficientes poco densos $\mathbf{x} = \mathbf{0}$ • Iteración t = 1
PROCEDIMIENTO	• Encontrar el átomo mas correlacionado con la señal, resolviendo $k = \arg\max \lvert\mathbf{A}^{\dagger}\mathbf{r}\rvert$[1] • Incluir el nuevo índice en el conjunto de índices ya encontrado $\Lambda^{(t)} = \Lambda^{(t-1)} \cup k$ • Actualizar el vector de coeficientes resolviendo el problema de mínimos cuadrados $x = \arg\min_x \lvert\lvert\mathbf{A}_{\Lambda^{(t)}}x - \mathbf{y}\rvert\rvert_2 = \mathbf{A}^{+}_{\Lambda^{(t)}}\mathbf{y}$[2] • Actualizar el residuo $\mathbf{r} = \mathbf{y} - \mathbf{A}_{\Lambda^{(t)}}x$ • Incrementar la iteración t = t + 1 y repetir hasta que se cumpla el criterio de parada, determinada por una de las dos posibles restricciones (poca densidad o error).
SALIDA	• El vector poco denso $\mathbf{x} \in \mathbb{R}^m$ • El conjunto de índices Λ

[1] El operador † denota la transpuesta de la matriz $\mathbf{A}$.
[2] El operador + denota la seudoinversa de la matriz $\mathbf{A}$.

2.3 DICCIONARIOS REDUNDANTES PARA REPRESENTACIÓN POCO DENSA DE SEÑALES

En el problema planteado hasta ahora, se supone conocido el diccionario que permite la representación de la señal en otro dominio conveniente. Como se mencionó anteriormente, un diccionario $\mathbf{A} \in \mathbb{R}^{n \times K}$ es una colección finita de señales elementales normalizadas, conocidas como átomos del diccionario. Analizándolo desde el punto de vista del álgebra

lineal, si el diccionario es de rango completo, es decir, expande todo el espacio columna, la representación de $\mathbf{y}$ en el mismo está garantizada. Evidentemente, la elección del diccionario que mejor descomponga la señal es determinante en la calidad de la representación dispersa y, generalmente, depende del tipo de datos a representar.

Los diccionarios sobrecompletos son utilizados en distintas áreas de investigación porque su uso puede derivar en aproximaciones más estables, más robustas y en descomposiciones más compactas, que usando diccionarios completos de bases ortogonales. Para que el diccionario sea redundante se debe cumplir estrictamente que $K > n$, lo cual implica que ha de existir dependencia lineal entre los átomos. Un diccionario puede ser pre-especificado, formado a partir de una o varias bases ortogonales, o ser diseñado de tal manera que se adapte lo mejor posible al tipo de datos que se desea representar, tratando de maximizar la representación dispersa en el mismo. En este contexto surgen los diccionarios entrenados como una alternativa de adaptar las características del diccionario a los datos que representarán.

2.3.1 *Diccionario DCT sobrecompleto*

Este diccionario se basa en la transformada discreta del coseno (DCT), que tiene la propiedad de representar una señal en términos de una sumatoria de funciones coseno muestreadas a diferentes frecuencias. Esta transformada esta caracterizada por sintetizar de manera compacta la información visualmente significativa en un conjunto reducido de coeficientes, lo que la hace ideal para el problema de representación dispersa de imágenes. Para representar de mejor manera, los cambios verticales y horizontales de la imagen es necesario definir un diccionario bidimensional, basado en la transformada ordinaria. De esta manera, se genera un diccionario 2D-DCT sobrecompleto de dimensiones $M \times N$, a partir del producto *Kronecker* de dos diccionarios 1D-DCT sobrecompleto de $\sqrt{M} \times \sqrt{N}$. Para más detalles sobre cómo se genera este diccionario, se recomienda revisar la referencia [1].

El diccionario 1D-DCT sobrecompleto se puede obtener a partir de

$$T_{ij} = \begin{cases} \frac{1}{\sqrt{M}} & si\ i = 0,\ 0 \leqslant j \leqslant M-1 \\ \cos\left(\frac{\pi ij}{N}\right) & si\ 0 < i \leqslant M-1,\ 0 \leqslant j \leqslant N-1 \end{cases} \tag{2.7}$$

En la Figura 2.1 se aprecia el conjunto de átomos de un diccionario 2D-DCT sobrecompleto y separable de 256 átomos para representar imágenes de 8×8 píxeles. La redundancia de un diccionario se define como la relación N/M. El diccionario descrito, tiene una redundancia de 4. Como se verá más adelante, este parámetro está relacionado con el nivel de compactación de la señal en el dominio del diccionario.

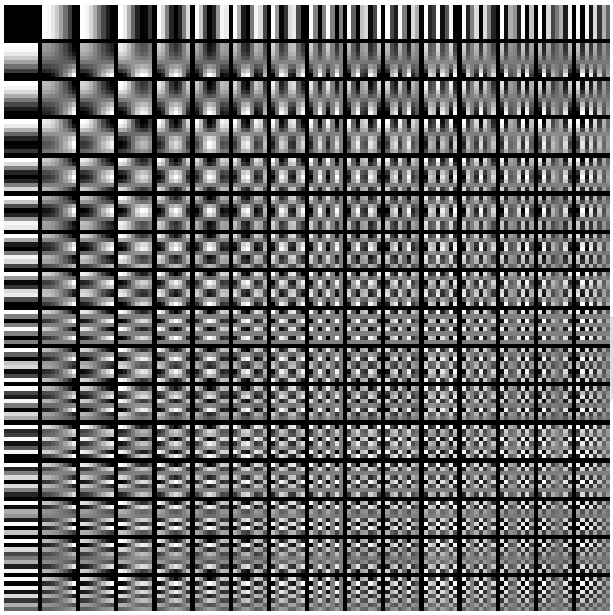


Figura 2.1. Diccionario DCT sobrecompleto de 256 átomos para representar bloques de 8×8 píxeles con una redundancia de 4.

2.3.2 *Diccionarios Entrenados utilizando K-times Singular Value Decomposition (KSVD)*

La idea fundamental que se persigue al entrenar diccionarios, es adaptar los átomos de un diccionario base a la estadística de un grupo de señales para mejorar la representación poco densa de éstas en el nuevo dominio. Existen variedad de algoritmos que se utilizan para esta tarea, pero la mayoría involucra un proceso de dos etapas básicas: 1)Representación poco densa del conjunto de imágenes de entrada y 2) actualización de los átomos del diccionario base con el objetivo de alcanzar la convergencia determinada por algún criterio de calidad.

En esta investigación se utilizará el algoritmo K-SVD como método de entrenamiento propuesto en [29], que sigue el esquema de las dos etapas mencionadas. En este sentido, se busca el mejor diccionario posible para la representación con coeficientes dispersos del grupo de señales $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N]$, tal que se minimice el siguiente problema de optimización

$$\arg\min_{\mathbf{A},\mathbf{X}} \sum_i \|\mathbf{x}_i\|_{\ell_0} \quad \text{sujeto a} \quad \|\mathbf{Y} - \mathbf{A}\mathbf{X}\|_F^2 \leqslant \epsilon, \tag{2.8}$$

para un valor establecido de ϵ. Por otro lado, se puede reescribir el problema teniendo en cuenta la estructura dispersa de cada vector, como

$$\arg\min_{\mathbf{A},\mathbf{X}} \left\{\|\mathbf{Y} - \mathbf{A}\mathbf{X}\|_F^2\right\} \quad \text{sujeto a} \quad \forall i,\ \|\mathbf{x}_i\|_{\ell_0} \leqslant T_0. \tag{2.9}$$

Donde $\|\cdot\|_F$ denota la norma matricial (norma *Frobenius*) definida, como

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |A_{i,j}|^2}. \tag{2.10}$$

Este método de entrenamiento de diccionarios se encuentra resumido en la Tabla 2.2. En la etapa de inicialización el diccionario base $\mathbf{A}^{(0)}$ puede ser utilizado cualquier diccionario pre-especificado, como el estudiado en la Sección anterior, donde los átomos son funciones parametrizadas por la función coseno, o escoger de forma aleatoria señales del conjunto $\mathbf{Y}$ como los átomos iniciales. En secciones posteriores, se utilizará la denominación k_{ksvd}, para referirse al parámetro que indica el nivel de poca densidad T_0 dentro del algoritmo descrito.

2.3.3 *Diccionario holográfico*

El diccionario holográfico es un concepto extraído de la teoría de Sensado Comprimido, diseñado para recuperar señales de cierta dimensión a partir de su representación poco densa de señales de menor dimensión. Consiste en proyectar el diccionario $\mathbf{A}$ sobre una matriz de medición $\mathbf{M}$, tal que

Tabla 2.2. Algoritmo K-SVD para entrenamiento de diccionarios.

ENTRADA	• Conjunto de señales de entrenamiento $\mathbf{Y}$. • Nivel de poca densidad T_0
INICIALIZACIÓN	• Fijar el diccionario $\mathbf{A}^{(0)} \in \mathbb{R}^{n\times K}$ con las columnas normalizadas bajo la norma ℓ_2 y el contador de iteración $J = 1$.
ITERACIÓN	Repetir hasta alcanzar convergencia • Etapa de codificación poco densa: Use cualquier algoritmo de búsqueda (por ejemplo, OMP) para calcular los vectores de representación $\mathbf{x}_i$ para cada señal $\mathbf{y}_i$, por aproximación de la solución de $\arg\min_{\mathbf{x}_i} \|\mathbf{y}_i - \mathbf{A}\mathbf{x}_i\|_{\ell_2}^2, \quad \text{sujeto a } \|\mathbf{x}_i\|_{\ell_0} \leqslant T_0, \text{ para } i = 1, 2, \ldots, N.$ • Etapa de actualización del diccionario: Para cada columna $k = 1, 2, \ldots, K$ en $\mathbf{A}^{(J-1)}$, actualizar por – Definir el grupo de señales en $\mathbf{Y}$ que usan este átomo, $\omega_k = \{i \mid 1 \leqslant i \leqslant N,\ \mathbf{x}_T^k(i) \neq 0\}$. – Calcular la matriz de error de la representación general, $\mathbf{E}_k$, por $\mathbf{E}_k = \mathbf{Y} - \sum_{j\neq k} \mathbf{a}_j \mathbf{x}_T^j$ – Restringir $\mathbf{E}_k$ escogiendo solo las columnas correspondientes a ω_k, y obtener $\mathbf{E}_k^R$. – Aplicar descomposición SVD $\mathbf{E}_k^R = \mathbf{U}\Delta\mathbf{V}^T$. Elegir la columna actualizada del diccionario $\mathbf{a}_k$ como la primera columna de $\mathbf{U}$. Actualizar el vector de coeficientes $\mathbf{x}_R^k$ como la primera columna de $\mathbf{V}$ multiplicado por $\Delta(1,1)$. • Establezca $J = J + 1$. • Verificar el criterio de parada.
SALIDA	El resultado deseado es el diccionario $\mathbf{A}^{(J)}$ obtenido después de J iteraciones.

$$\mathbf{H} = \mathbf{M}\mathbf{A} = [H_1, H_2, \ldots, H_K] \in \mathbb{R}^{n\times K}. \tag{2.11}$$

Para propósitos de esta investigación, a esta matriz se le denominará máscara y consistirá de una matriz diagonal binaria, producto de las zonas perdidas en la imagen. Esto se profundizará en secciones posteriores.

2.4 REPRESENTACIÓN POCO DENSA DE IMÁGENES

Suponga una imagen $\mathbf{I}$ dispuesta en una matriz de $M \times N$ elementos, en la que se ordena la información de cada pixel que la representa. Si se desea extender el concepto de representación poco densa estudiado en la primera Sección, es necesario ordenar lexicográficamente[3] a la matriz para obtener un vector n dimensional ($\mathbf{y} \in \mathbb{R}^n$). Recuerde que para representar esta señal de forma dispersa, es necesario un diccionario sobrecompleto $\mathbf{A} \in \mathbb{R}^{n \times K}$, con $K > n$. Esta cuestión limita el tamaño de las imágenes que podrían procesarse utilizando computadores comunes, ya que la dimensión n del vector estará dado por $n = MN$. Por esta razón, en el capítulo 15 de [28] se propone dividir la imagen en pequeños bloques y efectuar el procesamiento de cada bloque por separado. Desde esta óptica, se puede dividir la imagen solapando los bloques respectivos o evitando dicho solapamiento.

2.4.1 *Descomposición de una imagen en bloques*

Suponga la misma imagen $\mathbf{I}$ de $M \times N$, definida en la Sección anterior. Dicha imagen está formada por $\mathbf{p}$ bloques o parches de $\sqrt{n} \times \sqrt{n}$, tales que $N = a\sqrt{n}$ y $M = b\sqrt{n}$ siendo $a, b \in \mathbb{N}$. De esta manera, se podrá descomponer a la imagen original en NM/n bloques contiguos no solapados. Por otro lado, también se puede descomponer a la misma imagen en $(N-\sqrt{n}+1)(M-\sqrt{n}+1)$ bloques solapados completamente entre sí, de tal forma que un mismo píxel se encontrará en distintos bloques en los que se dividió a la imagen. Entonces, se podrá sintetizar la representación poco densa de cada uno de los bloques que componen la imagen global, realizando el ordenamiento lexicográfico y aplicándole el algoritmo OMP a cada uno de éstos. Como ya se mencionó, la ventaja de este procesamiento es que el tamaño del diccionario necesario disminuye con el tamaño del bloque escogido. Por ejemplo, si se desea procesar una imagen de 512×512 píxeles, se requiere de un diccionario, con una redundancia típica de 4, de dimensiones 262144×1048576, para realizar un procesamiento completo de la imagen. En comparación, si se desea utilizar un tamaño de bloque de 16×16, se puede utilizar un diccionario, con una redundancia de 4, de tamaño 256×1024.

Para ilustrar el concepto de procesamiento por bloques, en la Figura 2.2, se muestra

[3]El ordenamiento lexicográfico implica re-ordenar los elementos de una matriz dada, en la que cada columna se concatena con la primera, para formar un vector columna que contiene todos los elementos de la matriz.

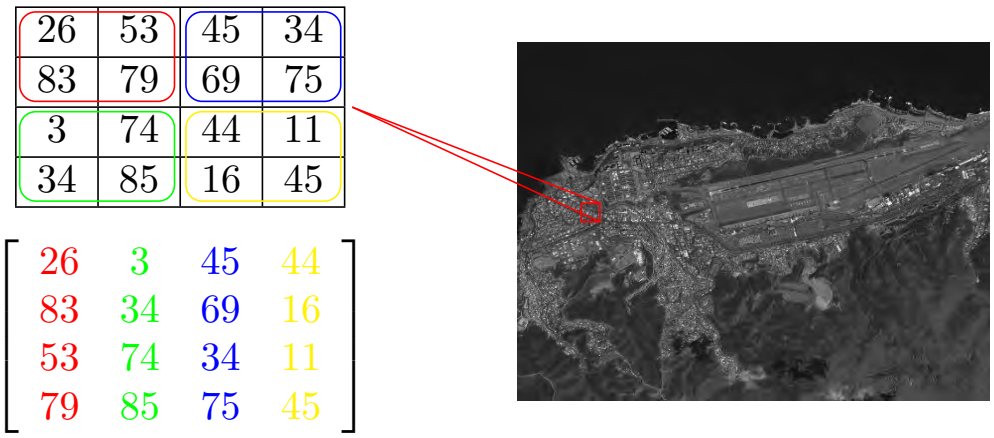
una imagen en escala de gris, a la que se le hará un procesamiento digital en regiones de la imagen. En la Figura 2.2(a) se observa una Sección de tamaño 4×4 píxeles a la que se le hará procesamiento sin solapamiento por bloques de 2×2 píxeles, los cuales se encuentran resaltados en color rojo, azul, verde y amarillo. Observe que para este caso, sólo son generados un conjunto de $NM/(\sqrt{n}\sqrt{n}) = (4)(4)/((2)(2)) = 4$ bloques no solapados. En la misma Figura, se observa la matriz generada, en el que se encuentra el ordenamiento lexicográfico del conjunto de bloques. Como puede apreciarse también, ningún píxel es repetido en otro parche de la imagen. Por otro lado, en la Figura 2.2(b) se ilustra el procesamiento por bloques solapados. En este caso, se procesará una región de 3×3 píxeles, utilizando el mismo tamaño de bloques de 2×2 píxeles. Esto genera $(N - \sqrt{n} + 1)(M - \sqrt{n} + 1) = (3 - 2 + 1)(3 - 2 + 1) = 4$ bloques solapados. Tal y como se puede apreciar en la Figura, los bloques en color rojo, azul, verde y amarillo contienen píxeles comunes. Por ejemplo, el píxel con valor 53 aparece en dos bloques, mientras que el que tiene valor 79 para el nivel de gris, aparece en los cuatro bloques. Esta repetición dependerá de la posición del píxel y el tamaño del bloque escogido para el procesamiento. Esto genera redundancia, pues cada píxel que forma parte de k bloques, debe procesarse k veces.

2.4.2 *Filtros de reconstrucción*

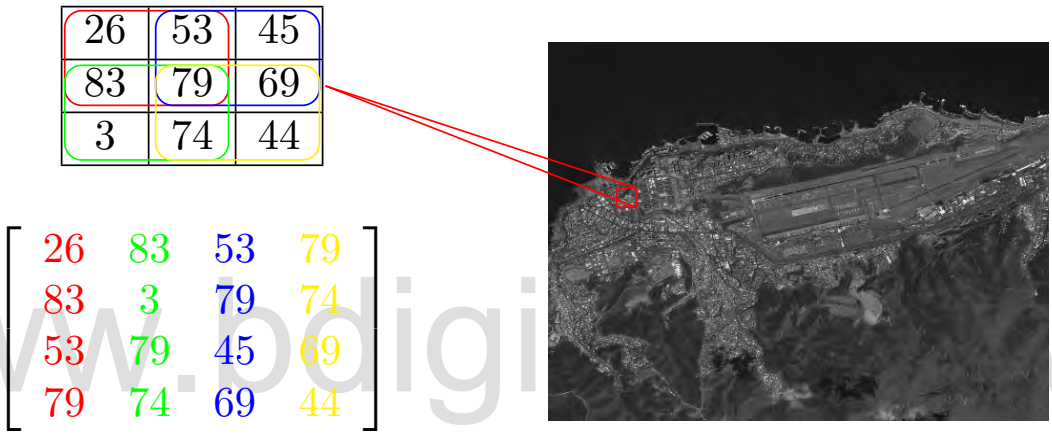
Como se explicó en la Sección anterior, al procesar con solapamiento de bloques, cada píxel de la imagen será el resultado de una estimación matemática de un conjunto de valores, producto de la redundancia del mismo. Para encontrar un valor representativo en función de los múltiples valores estimados se utilizarán los siguientes filtros de reconstrucción:

- **Filtro promediador:** Consistirá en el promedio de los píxeles del conjunto $\hat{x}_{i,j} = \sum_{z=1}^{p} \frac{x_{i,j,z}}{p}$, donde p es la cantidad usada para la estimación. Desde el punto de vista computacional, este filtro es más eficiente y sencillo de implementar.

- **Filtro de mediana:** El píxel vendrá dado por $\hat{x}_{i,j} = \text{MEDIANA}([x_{i,j,1}, \ \ldots \ , \ x_{i,j,p}])$. Como esta operación implica un reordenamiento de los datos, requiere de más costo computacional. Sin embargo, es una estimación más robusta ante valores atípicos en el conjunto estimado.



(a)



(b)

Figura 2.2. Ejemplo ilustrativo del procesamiento de una imagen por bloques de 2×2 píxeles. (a) Procesamiento sin solapamiento de bloques. (b) Procesamiento utilizando bloques solapados.

2.4.3 *Tamaño del Bloque*

Al realizar un procesamiento por bloques, el tamaño del mismo es un factor importante a elegir, ya que este determinará el tamaño del diccionario utilizado. Es común utilizar potencias de 2 para elegir los tamaños del parche, por ejemplo, en [1] se usa un tamaño de 8×8 píxeles, mientras que en [28] y [30], adicionalmente se ha registrado el uso de tamaños de 16×16 y 32×32 para el procesamiento de imágenes con píxeles perdidos. Hay que resaltar también, que entre más pequeño sea el bloque, más localizada estará la información disponible para la estimación y esto debe tomarse en cuenta al atacar el problema la pérdida de píxeles, objeto de esta investigación.

2.4.4 *Redundancia del Diccionario*

Hasta el momento se ha dicho que el diccionario utilizado en la representación dispersa de la imagen es sobrecompleto, es decir, $\mathbf{A} \in \mathbb{R}^{n \times K}$, con $K > n$. La redundancia de un diccionario sobrecompleto se define como la relación K/n, el cual se supone como un valor entero. En [31], se utilizaron redundancias de 2,4 y 8. Por otro lado, en [28], capítulo 14, se describió el uso de un diccionario sobrecompleto basado en la transformada Haar, con una redundancia de 7. Finalmente, para descomposición de imágenes utilizando un análisis de componentes morfológicas (MCA, del inglés *Morphological Component Analysis*), se ha utilizado una redundancia de 129 y 64, en el uso de diccionarios basados en curvelet de 6 niveles de resolución y DCT, respectivamente.

CAPÍTULO 3
RESTAURACIÓN DE IMÁGENES OCLUIDAS UTILIZANDO REPRESENTACIÓN POCO DENSA DE SEÑALES

Utilizando la teoría de sensado comprimido, la cual se fundamenta en los conceptos expuestos en el Capítulo anterior, se pretenden desarrollar metodologías que permitan atacar el problema de la oclusión. Primero se aborda un modelo simple para representar la pérdida de información en una imagen para luego relacionar dicho modelo con la teoría de representación dispersa con el que se pueda obtener una metodología de reconstrucción. Por otro lado, se abordan detalles relacionados con la manera en que se entrenan diccionarios adaptados a las imágenes que se desean reconstruir. En la última Sección, se argumentan razones por la que el lenguaje de programación Python es elegido para la implementación de las técnicas de reconstrucción, finalizando con una presentación de algunas de las rutinas desarrolladas y su función en la solución del problema planteado.

3.1 MODELADO DE LA OCLUSIÓN EN UNA IMAGEN

Como ya se ha mencionado, una oclusión es el proceso de pérdida de píxeles, que para el caso de imágenes satelitales este pudiera ser ocasionado por un fenómeno atmosférico. Para modelar el problema de forma matemática, basada en la representación dispersa, podemos utilizar la siguiente ecuación matricial

$$\mathbf{y}_{obv} = \mathbf{M}\mathbf{y}. \tag{3.1}$$

En este modelo, $\mathbf{y}$ representa el ordenamiento lexicográfico de un bloque de la versión original de la imagen, sin la degradación ocasionada por la pérdida de píxeles. Por otro lado, $\mathbf{y}_{obv}$ es el ordenamiento lexicográfico de un bloque de la imagen observada. Finalmente, $\mathbf{M}$ es una matriz binaria diagonal, denominada máscara, que modela el proceso de pérdida de píxeles que degrada al parche seleccionado de la imagen original. Los valores de $\mathbf{M}$ están definidos por

$$\mathbf{m}_{i,i} = \begin{cases} 0 & \text{si } \mathbf{y}_i \text{ es un pixel perdido} \\ 1 & \text{otros} \end{cases} \tag{3.2}$$

La Figura 3.1 ilustra la pérdida de píxeles de la imagen observada ($\mathbf{y}_{obv}$), que está relacionada con el valor que contenga la diagonal de M en la posición correspondiente. Observe que los píxeles perdidos son representados por el nivel de gris negro.

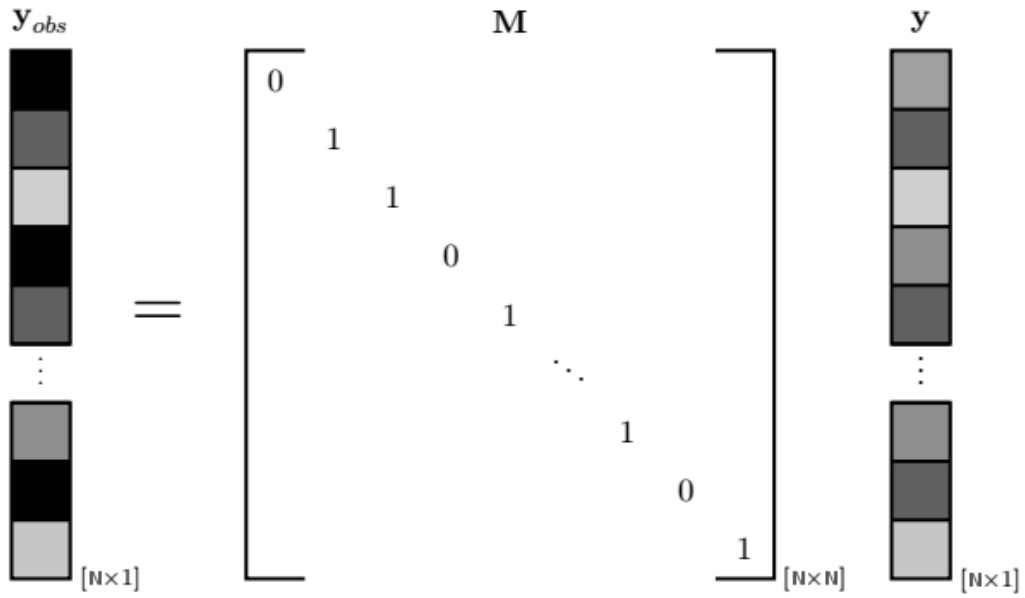


Figura 3.1. Perdida de píxeles modelada a partir de la matriz M.

Suponiendo que la imagen $\mathbf{y}$ es dispersa en algún dominio, se puede rescribir la ecuación 3.1 como

$$\mathbf{y}_{obv} = \mathbf{MAx}. \tag{3.3}$$

Si observamos detenidamente la ecuación anterior, el uso de la matriz $\mathbf{M}$ implica la

definición de un diccionario holográfico, $\mathbf{V} = \mathbf{MA}$. Entonces, la representación dispersa puede ser encontrada a partir de

$$\arg\min_{\mathbf{x}} \|\mathbf{y}_{obv} - \mathbf{V}\mathbf{x}\|_{\ell_2} \quad \text{sujeto a} \quad \|\mathbf{x}\|_{\ell_0} \leqslant T_0. \tag{3.4}$$

Al resolver este problema de optimización, se está obteniendo la representación dispersa de $\mathbf{y}_{obv}$, que es la misma señal que $\mathbf{y}$ en los píxeles conocidos. Tal como lo señala Elad en [32], si $\mathbf{x}$ es lo bastante dispersa en el dominio del diccionario holográfico $\mathbf{V}$, se puede asumir que la representación también será dispersa en el dominio del diccionario $\mathbf{A}$. Con estos supuestos, se puede decir que $\mathbf{x} = \hat{\mathbf{x}}$, obteniendo la imagen restaurada como $\hat{\mathbf{y}} = \mathbf{A}\hat{\mathbf{x}}$.

3.2 RECONSTRUCCIÓN DE LA OCLUSIÓN EN UNA IMAGEN USANDO PROCESAMIENTO POR BLOQUES

Para una imagen completa, este procesamiento se aplica a cada uno de los bloques obtenidos de la división de la imagen. Sea una imagen $\mathbf{I}$ de tamaño $N \times N$, a la que se le aplica una división en bloques de $b \times b$ píxeles, siguiendo un enfoque sin solapamiento de parches o con solapamiento. Cada bloque es ordenado lexicográficamente generando un conjunto de vectores que pueden ser dispuestos en una matriz $\mathbf{Y}_{obv} = [\mathbf{y}_{obv1}, \mathbf{y}_{obv2}, \ldots, \mathbf{y}_{obvNb}] \in \mathbb{R}^{n \times Nb}$, donde $\mathbf{y}_{obvi}$ corresponde a cada uno de los bloques, $n = b^2$ y Nb es el número de parches en que fue dividida la imagen que dependerá del tipo de solapamiento. La matriz $\mathbf{y}_{obvi}$ estará compuesta de un conjunto de bloques que están afectados por la oclusión y otro conjunto de bloques limpios. Por otro lado, suponga una máscara M, del mismo tamaño que la imagen I, que es dividida en igual número de bloques a los que se les aplica ordenamiento lexicográfico. De la misma manera, el conjunto de vectores puede ser organizado en una matriz $\mathbf{Mask} = [\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_{Nb}] \in \mathbb{R}^{n \times Nb}$, donde $\mathbf{m}_i$ corresponde al ordenamiento lexicográfico de cada uno de los bloques en que fue divida la máscara $\mathbf{M}$.

Por otro lado, al abordar el problema planteado en esta investigación, en el que sólo se desean reconstruir las áreas afectadas por la oclusión, sería conveniente manipular sólo aquellos bloques de la imagen que se encuentren ocluidos. Con esta idea en mente, sólo se procesará un grupo reducido de bloques a los que se le estimarán los píxeles que faltan.

La misma reducción es aplicada a la máscara para obtener el diccionario holográfico de cada uno de los bloques ocluidos. Por esta razón, se ejecuta un proceso de reducción de las matrices $\mathbf{Y}_{obv}$ y **Mask**, con el objeto de obtener a $\mathbf{Y}_{obvred}$ y $\mathbf{Mask}_{red}$, las cuales contendrán el ordenamiento lexicográfico de los bloques ocluidos (Nb_{ocl}).

Para cada columna se resuelve el problema de optimización descrito en la ecuación 3.4, con el fin de encontrar la representación dispersa xi de cada bloque ocluido. A partir de este conjunto de representaciones dispersas, se estima cada uno de lo bloques $\hat{\mathbf{y}}_i = \mathbf{A}\hat{\mathbf{x}}_i$ que pueden ser organizados en una matriz $\hat{\mathbf{Y}}_{red} \in \mathbb{R}^{n \times Nb_{ocl}}$. Como los bloques que no fueron afectados por la oclusión no son procesados, se obtiene la matriz $\hat{\mathbf{Y}}$ complementando el conjunto d vectores en $\hat{\mathbf{Y}}_{red}$ con el resto de la información contenida en $\mathbf{Y}_{obv}$, obteniendo la matriz $\hat{\mathbf{Y}}$. Finalmente, se reordena la información de cada columna para obtener la imagen $\hat{\mathbf{I}}$, utilizando alguno de los filtros de reconstrucción en caso de que existan píxeles redundantes. El algoritmo propuesto se resume en la Tabla 3.1.

3.3 RECONSTRUCCIÓN ITERATIVA

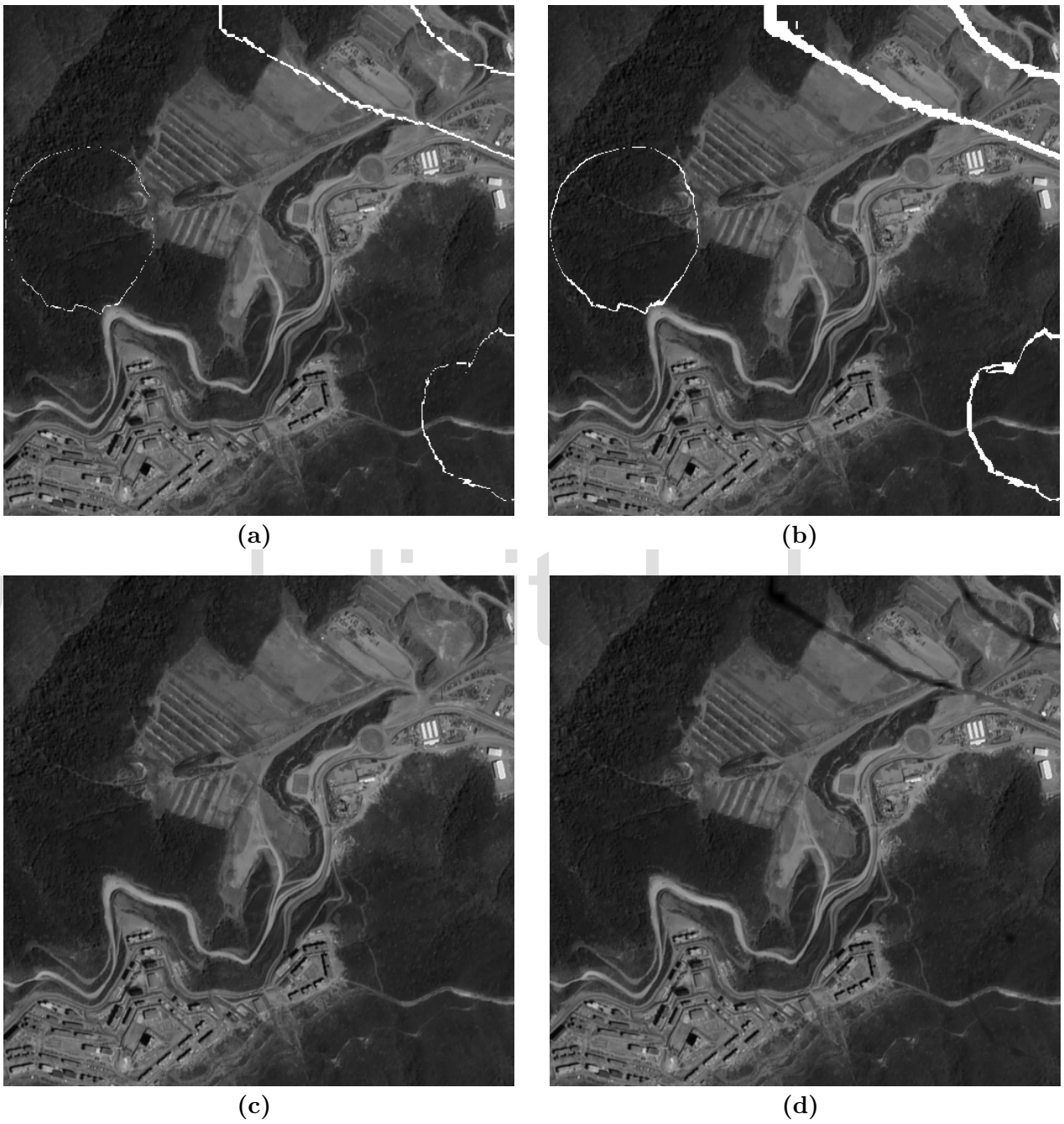
Profundizando en la cuestión planteada en el problema de optimización descrito por la ecuación 3.4, ¿qué cantidad de pérdida de información se puede permitir para obtener una reconstrucción aceptable al resolver dicho problema de optimización? Por ejemplo, suponga un procesamiento por bloques de 8×8. Está claro que para este caso se tendrá hasta un máximo de 64 muestras de nivel de gris para cada posición dentro del parche. De este número total ¿se pueden permitir 20 píxeles perdidos, por poner un número? La lógica indica que la calidad de la reconstrucción se deteriorará a medida de que el porcentaje de pérdida se incremente. Para mostrar esta idea, considere la Figuras 3.2(a) y 3.2(b), en la que se muestra a la misma escena ocluida por dos máscaras sintéticas, una más severa que la otra. Al aplicar la técnica de reconstrucción descrita en la Sección 3.1, utilizando un solapamiento completo, y un diccionario ODCT de 64×256, se obtienen las respectivas reconstrucciones en las Figuras 3.2(c) y 3.2(d). Como puede observarse, el resultado de aplicar el procesamiento a la oclusión en la Figura 3.2(a) genera un buen resultado, visualmente hablando. Por otro lado, se observa que para la oclusión más severa, hay áreas en la que el algoritmo falla, reproduciendo a la oclusión en la salida. Por lo que,

Tabla 3.1. Algoritmo de Reconstrucción de Oclusiones

Entrada	• Imagen Ocluida $\mathbf{I}_{obv}$ de tamaño $N \times N$. • Mascara $\mathbf{M}$ de tamaño $N \times N$.
Configuración de parámetros	• Definir el tamaño del bloque de procesamiento $\mathbf{b}$, que estará dado por $\mathbf{b} \times \mathbf{b}$. • Fijar el diccionario $\mathbf{A}$ de tamaño $\mathbf{n} \times \mathbf{K}$. • Seleccionar el tipo de solapamiento • Elegir el valor de $\mathbf{k}_{omp}$
Etapa de Procesamiento	• Dividir la imagen ocluida y la máscara en bloques, según el tamaño definido $\mathbf{b}$ y el tipo de solapamiento seleccionado. • Aplicar ordenamiento lexicográfico al grupo de bloques en los que se dividió la imagen ocluida, organizados en la matriz $\mathbf{Y}_{obv}$ de tamaño $\mathbf{n} \times \mathbf{Nb}$. Donde $\mathbf{n} = \mathbf{b}^2$ y $\mathbf{Nb}$ es el número de bloques. • Aplicar ordenamiento lexicográfico al grupo de bloques en los que se dividió a la imagen máscara, organizados en la matriz **Mask**, de tamaño $\mathbf{n} \times \mathbf{Nb}$. • Reducir las matrices $\mathbf{Y}_{obv}$ y **Mask** para generar las matrices $\mathbf{Y}_{obvred}$ y $\mathbf{Mask}_{red}$, que contienen sólo los bloques ocluidos. • Para cada columna $i = 1, 2, \ldots, Nb_{ocluidos}$ en $\mathbf{Y}_{obvred}$, resolver el problema de optimización $\arg\min_{\mathbf{x}} \|\mathbf{y}_{obvi} - \mathbf{m}_i \mathbf{A}\mathbf{x}_i\|_{\ell_2} \quad \text{sujeto a} \quad \|\mathbf{x}_i\|_{\ell_0} \leqslant T_0$ • Estimar cada bloque $\mathbf{y}_i = A\mathbf{x}_i$, reorganizando cada estimación en la matriz $\hat{\mathbf{Y}}$ de tamaño $\mathbf{n} \times \mathbf{Nb}$. Los bloques que no fueron procesados, se mantienen igual a los bloques en $\mathbf{Y}_{obv}$. •Realizar la reconstrucción de la imagen estimada $\hat{\mathbf{I}}$ de tamaño $\mathbf{N} \times \mathbf{N}$, a partir de $\hat{\mathbf{Y}}$, utilizando algún filtro de reconstrucción.
Salida	$\hat{\mathbf{I}}$ de tamaño $\mathbf{N} \times \mathbf{N}$

en concordancia con las ideas expuestas, la capacidad de reconstrucción del algoritmo que resuelve al problema de optimización en 3.4, comienza a fallar para un valor determinado de pérdida de píxeles. Como se mostrará en el siguiente Capítulo a través de experimentos controlados, se pudiera definir un umbral máximo de pérdida en el que se puede garantizar una reconstrucción aceptable.

En armonía con esta suposición, se plantea una metodología iterativa de reconstrucción, siguiendo las ideas de los métodos de inpainting tradicionales [16,17,20] que son iterativos y en los que se supone que una vez estimado un píxel desconocido no se vuelve a estimar.



(a) **(b)**

(c) **(d)**

Figura 3.2. Ejemplo de reconstrucción de la imagen PAN1 ocluida por dos máscaras sintéticas (a) Imagen ocluida por la máscara leve. (b) Imagen ocluida por la mascara media (c) Reconstrucción de la imagen ocluida por la máscara leve. (d) Reconstrucción de la imagen ocluida por la máscara media.

Con esto en mente, dado un umbral permitido de pérdida, se puede reconstruir iterativamente sólo los bloques que cumplan con el umbral definido, utilizando el algoritmo de reconstrucción descrito en la primera Sección de este Capítulo, hasta estimar todos los píxeles desconocidos. Evidentemente, para cada etapa de reconstrucción es necesario un proceso de identificación de la oclusión remanente, en el que se pudiera utilizar un algoritmo de detección automática. No obstante, como este proceso escapa de los propósitos de este proyecto, se implementó una metodología de detección basada en la máscara inicial, la cual es conocida. Básicamente, la idea es ir modificando a la máscara a medida que se estiman los píxeles perdidos. El algoritmo de reconstrucción iterativa se presenta en la Tabla 3.2. Para esta versión del algoritmo de reconstrucción, el umbral de pérdida permitido debe ser definido previamente. En el próximo Capítulo se presentará a través de experimentos controlados la influencia que tiene este parámetro sobre la reconstrucción obtenida.

3.4 ENTRENANDO DICCIONARIOS

En la Sección 2.3, se expuso la incidencia del diccionario en la calidad de la reconstrucción final. Ahora se analizará un detalle con respecto al proceso de entrenamiento utilizado en esta investigación. Está claro que el entrenamiento busca adaptar los átomos de un diccionario inicial a la estadística de un grupo de imágenes utilizadas en el proceso. Desde esta perspectiva, se podría utilizar un grupo aleatorio de imágenes que sean diferentes a la escena que desea restaurarse. Para ilustrar esta cuestión más claramente, supóngase que se cuenta con un conjunto de 9 imágenes de entrenamiento de tamaño 512×512 y se desea entrenar un diccionario de 64×256, tal y como usualmente se utiliza en esta investigación. Una división en bloques sin solapamiento de cada una de las imágenes generaría un grupo de 4096 bloques de 8×8, lo cual representa un grupo de parches de entrenamiento de 36864. Este número crece en proporción con la cantidad de imágenes que se deseen utilizar en el proceso de entrenamiento y al tamaño de las mismas. Esta cuestión tiene sus implicaciones computacionales, por lo que en las rutinas de entrenamiento, se limita el número de parches máximo que se pueden utilizar a 50000. Cuando este número es excedido, se elige esta cantidad de bloques de forma aleatoria del conjunto.

Tabla 3.2. Algoritmo de Reconstrucción Iterativa

Entrada	• Imagen Ocluida $\mathbf{I}_{obv}$ de tamaño $N \times N$. • Mascara $\mathbf{M}$ de tamaño $N \times N$.
Configuración de parámetros	• Definir el tamaño del bloque de procesamiento $\mathbf{b}$, que estará dado por $\mathbf{b}\times\mathbf{b}$. • Fijar el diccionario $\mathbf{A}$ de tamaño $\mathbf{n} \times \mathbf{K}$. • Seleccionar el tipo de solapamiento • Elegir el valor de $\mathbf{k}_{omp}$ • Definir el número de píxeles perdidos permitido $\mathbf{U}$ por bloque.
Etapa de Procesamiento	• Dividir la imagen ocluida y la máscara en bloques, según el tamaño definido $\mathbf{b}$ y el tipo de solapamiento seleccionado. • Aplicar ordenamiento lexicográfico al grupo de bloques en los que se dividió la imagen ocluida, organizados en la matriz $\mathbf{Y}_{obv}$ de tamaño $\mathbf{n}\times\mathbf{Nb}$. Donde $\mathbf{n} = \mathbf{b}^2$ y $\mathbf{Nb}$ es el número de bloques. • Aplicar ordenamiento lexicográfico al grupo de bloques en los que se dividió a la imagen máscara, organizados en la matriz $\mathbf{Mask}$, de tamaño $\mathbf{n} \times \mathbf{Nb}$. • Repita mientras la máscara $\mathbf{Mask}$ tenga píxeles ocluidos: – Reducción de las matrices $\mathbf{Y}_{obv}$ y $\mathbf{Mask}$ para generar las matrices $\mathbf{Y}_{obvred}$ y $\mathbf{Mask}_{red}$, que contienen sólo los bloques ocluidos que cumplen con el umbral $\mathbf{U}$. – Para cada columna $i = 1, 2, \ldots, Nb_{ocluidos}$ en $\mathbf{Y}_{obvred}$, resolver el problema de optimización $\arg\min_{\mathbf{x}} \|\mathbf{y}_{obvi} - \mathbf{m}_i\mathbf{A}\mathbf{x}_i\|_{\ell_2} \quad \text{sujeto a} \quad \|\mathbf{x}_i\|_{\ell_0} \leqslant T_0$ – Estimación de cada bloque $\mathbf{y}_i = A\mathbf{x}_i$, reorganizando cada estimación en la matriz $\hat{\mathbf{Y}}$ de tamaño $\mathbf{n} \times \mathbf{Nb}$. Los bloques que no fueron procesados, se mantienen igual a los bloques $\mathbf{Y}_{obv}$. Adicionalmente, se modifica la máscara reducida, cambiando las posiciones que fueron estimadas por 1. •Realizar la reconstrucción de la imagen estimada $\hat{\mathbf{I}}$ de tamaño $\mathbf{N}\times\mathbf{N}$, a partir de $\hat{\mathbf{Y}}$, utilizando algún filtro de reconstrucción.
Salida	$\hat{\mathbf{I}}$ de tamaño $\mathbf{N} \times \mathbf{N}$

Por otro lado, también se podría utilizar las áreas de la misma escena que no fueron ocluidas como imágenes de entrada al algoritmo de entrenamiento. En comparación, supóngase que se usan las áreas no degradadas de la misma escena, por lo que para una imagen de 512×512, en el que existan 500 bloques ocluidos, se dispondrá de una cantidad de $4096 - 500 = 3596$ parches para entrenamiento. Lo que implicará menos costo compu-

tacional y el diccionario estaría, teóricamente hablando, más adaptado a la estadística de la misma imagen. Aún así, como se mostrará en el siguiente Capítulo, se utilizan estos dos enfoques de entrenamiento.

3.5 MÁSCARA

Hasta el momento, se ha considerado a la máscara, que delimita a la oclusión, como algo conocido. En el primer Capítulo se mencionó que la generación de la misma implica un procesamiento previo de la imagen, el cuál podría ser manual o automático como el descrito en la Figura 1.1. Todos los algoritmos diseñados como parte de este trabajo, asumen que la máscara está definida por la regla descrita en la ecuación 3.2. Si algunos de los métodos mencionados no garantiza esto, se debería utilizar un mecanismo de umbralización en las que se garantice que la Máscara cumpla dichos requerimientos. Finalmente, se aplica la operación $\mathbf{y}_{obv} = \mathbf{M}y$, para obtener la imagen observada que se utiliza como entrada a los distintos algoritmos. Si bien esta operación fue descrita al comienzo de este capitulo como una multiplicación matricial, lo que esto implica es básicamente una multiplicación pixel a pixel para las mismas posiciones de la imagen. Entendiendo esto, ahora la matriz puede ser definida como una imagen del mismo tamaño que enmascara a la imagen que se desea estimar. En la Figura 3.3 se ilustra este proceso. Para efectos visuales, la oclusión en la imagen PAN1 se muestra en la Figura 3.3(c) con el nivel de gris blanco, aunque su representación real es con el nivel de gris negro, que representa el nivel 0. Observe que los píxeles ahora perdidos son obtenidos por el proceso de enmascaramiento. Aunque la imagen ocluida fue creada sintéticamente, para eliminar oclusiones ocasionadas por nubosidades en las imágenes el procedimiento será similar, luego de convertir la máscara en una imagen binaria. En resumen, para todos los algoritmos, se supone conocida la máscara y la imagen ocluida.

3.6 PAQUETES DE DESOCLUSIÓN DE IMÁGENES

Para implementar las metodologías de reconstrucción expuestas hasta el momento y realizar experimentos controlados para el análisis del problema planteado en este trabajo, se

(a) **(b)**

(c)

Figura 3.3. Ejemplo ilustrativo en el que se muestra la generación de una imagen ocluida sintéticamente. (a)Imagen sin contaminación. (b) Máscara sintética. (c) Imagen ocluida.

seleccionó el lenguaje de programación Python. Esta elección se fundamentó en que es un lenguaje de alto nivel de licencia libre, sintaxis bastante legible, uso extendido, amplio soporte y es un lenguaje multiparadigma (programación orientada objetos, imperativo y funcional) y multiplataforma (funciona en una gran cantidad de sistemas operativos), por mencionar solo algunas ventajas. Además, existen librerías científicas escritas en Phython

bastante desarrolladas y con gran capacidad, lo que lo hace ideal para los propósitos de este proyecto. Entre los módulos para cómputo científico utilizados se encuentran:

- **Numpy**: Para generación de tipos de datos científicos, como los arreglos numéricos.
- **Scipy**: En el que se definen funciones científicas de uso general.
- **Matplotlib**: Para generación de gráficos en 2D y 3D.

Durante el desarrollo de esta investigación se generaron varias funciones para el tratamiento de oclusiones en imágenes, además de diversas rutinas con las que se probaron los distintos enfoques de reconstrucción. En el CD que incluye la versión digital de este documento se encuentra una carpeta con el conjunto de rutinas generadas junto a los resultados que pueden ser fácilmente reproducidos. Cabe destacar que esta investigación forma parte de un proyecto macro, financiado por el FONACIT, en el que participaron de forma paralela otros desarrolladores. Por lo que, algunas rutinas, tales como la implementación de los algoritmos KSVD, OMP y otras de utilidad diversa, fueron desarrolladas por otros compañeros. Aún así, muchas de ellas debieron ser adaptadas a las particularidades de procesamiento del problema abordado.

CAPÍTULO 4
SIMULACIÓN Y RESULTADOS

En este Capítulo se validará de manera experimental las metodologías desarrolladas hasta el momento. Primero se modelará el fenómeno de la oclusión utilizando data sintética, con el objetivo de probar el desempeño de los algoritmos. Luego se presenta un proceso de selección de algunos de los parámetros importantes de los que depende la calidad de la reconstrucción. Para ello se utilizan medidas de desempeño comúnmente utilizadas en el área del procesamiento digital de imágenes. Distintas escenas son usadas para las pruebas experimentales, con el objetivo de dar cierta generalidad a los resultados expuestos. El Capítulo finaliza con una exposición de resultados cualitativos con el que se pretende probar el alcance de las técnicas presentadas.

4.1 DESCRIPCIÓN DE LA BASE DE DATOS

Para aplicar las técnicas hasta ahora desarrolladas, se utilizaron registros provistos por la Agencia Bolivariana para Asuntos Espaciales (ABAE). Las imágenes de prueba fueron seleccionadas de una imagen Pancromática obtenida de la región central de Venezuela, cuya coordenada geográfica del punto central de la imagen es $10°34'35.281''$ latitud Norte y $66°55'4.807''$ longitud Oeste. La misma fue capturada el día 21 de febrero del año 2.014, con un porcentaje de cobertura de nubes del 15 %. En la Figura 4.1 se muestra una versión reducida de la misma.

Se tomaron 4 secciones de 512×512 píxeles, en áreas sin presencia de nubes con el objetivo de evaluar las técnicas a partir de un conocimiento previo de las escena real. En el trabajo se le asignó la denominación Imagen PAN1, Imagen PAN2, Imagen PAN3 e Imagen PAN4, respectivamente. La primera imagen corresponde a un área cercana al complejo urbanístico Ciudad Caribia, ubicada en la zona montañosa del estado Vargas. La segunda

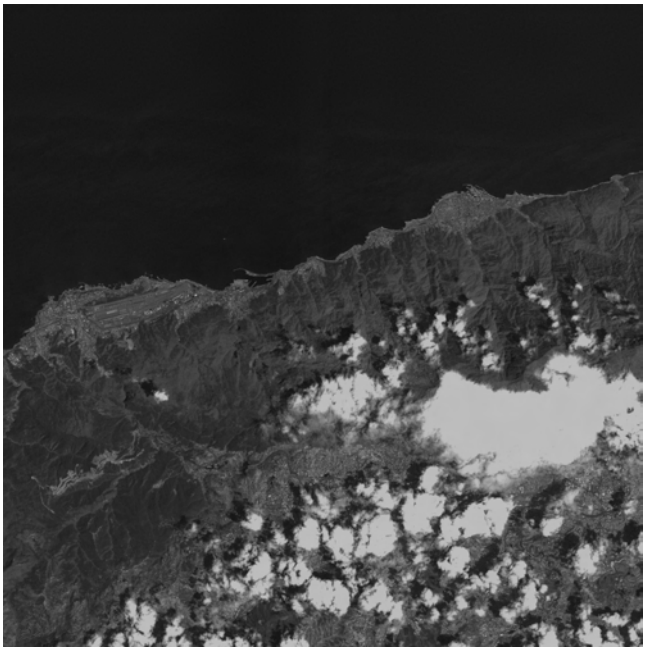
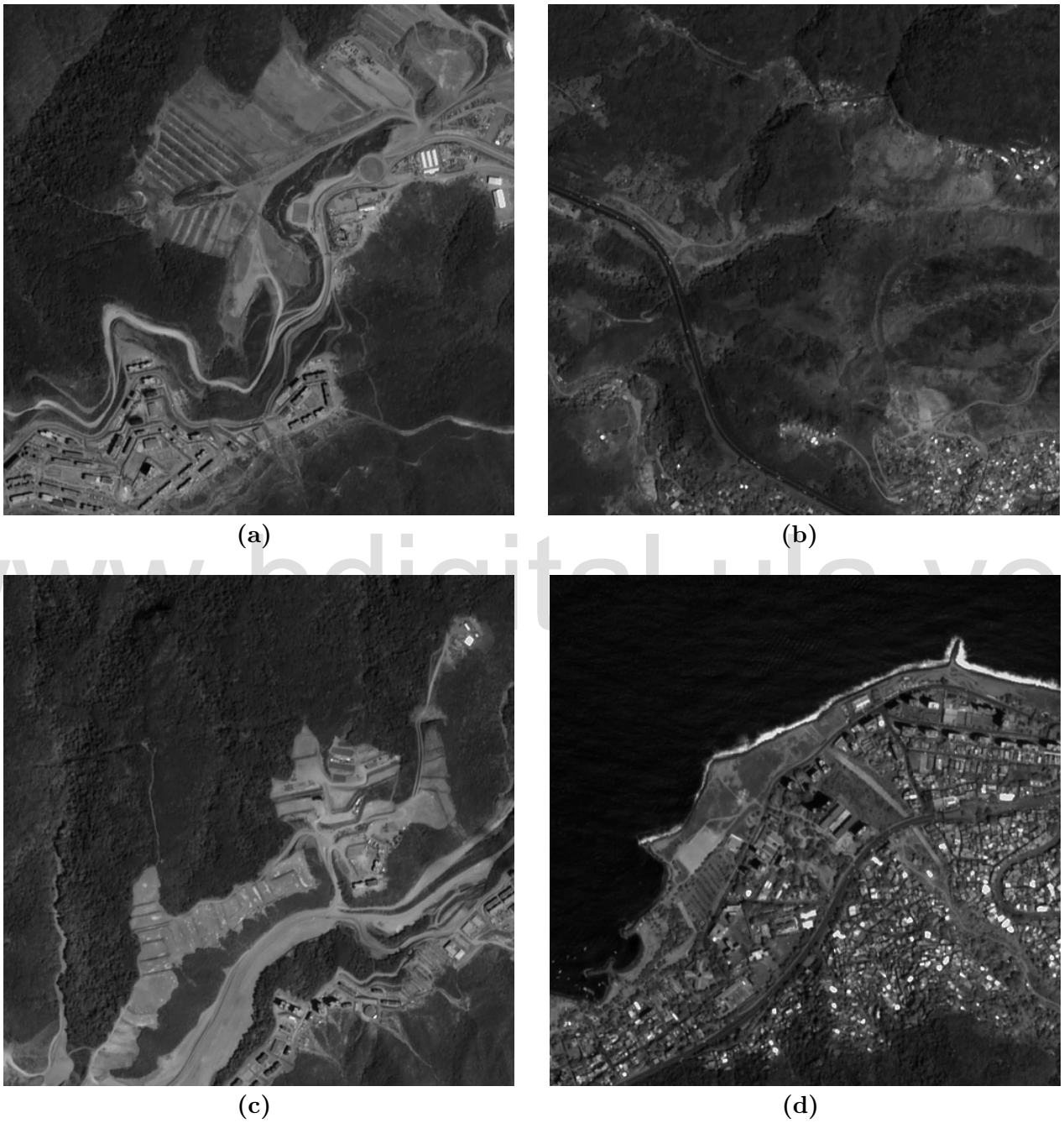


Figura 4.1. Imagen Pancromática de la región central de Venezuela captada por la cámara PAN-2.

sección, que está localizada en las afueras de Ojo de Agua, es una sección de la autopista Caracas-La Guaira. La Imagen PAN3, es una captura de una zona cercana al Plan de Manzano, vía hacia el castillo de San Joaquín, carretera los Dos Caminos. Finalmente, la cuarta imagen de prueba, está situada sobre la urbanización Punta de Brisas, en Macuto, Edo. Vargas. En la Figura 4.2 se muestra una representación visual de cada imagen. Para las técnicas de reconstrucción, no se considera la presencia de ruido aditivo o de otro tipo en la imagen además de la contaminación por pérdida de información debido a la oclusión. Por esta razón, las áreas no ocluidas de la imagen no se modificarán durante el procesamiento. Los desempeños fueron evaluados utilizando las métricas de desempeño tradicionales que se muestran en la Tabla 4.1, en las que cada píxel y_{ij} se refiere al valor real de la escena, mientras que $\hat{y}_{ij}$, se refiere al valor estimado por el proceso de reconstrucción. Estas métricas de calidad pueden ser calculadas a nivel de bloque, en los que se toman en cuenta todos los píxeles dentro de cada uno de los bloques ocluidos, o a nivel de píxel, en la que sólo se toma en consideración los píxeles estimados.



(a) **(b)**

(c) **(d)**

Figura 4.2. Secciones de prueba de 512×512 píxeles extraidas de una Imagen pancromática de la región central de Venezuela (a) Imagen PAN1 (b) Imagen PAN2 (c) Imagen PAN3 (d) Imagen PAN4.

Tabla 4.1. Medidas de desempeño

Nombre	Descripción	Ecuación
RMSE	Raíz del error medio cuadrático (*Root-Mean Square Error*)	$\sqrt{\frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\|\|y_{ij}-\hat{y}_{ij}\|\|^2}$
MAE	Error medio absoluto (*Mean Absolute Error*)	$\frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\|y_{ij}-\hat{y}_{ij}\|$
PSNR	Relación señal a ruido pico (*P*eak Signal-to-Noise Ratio)	$20\log_{10}\left\|\frac{\text{máx}(Y)}{\text{RMSE}(Y,\hat{Y})}\right\|$

4.2 RECONSTRUCCIÓN DE OCLUSIONES CAUSADAS POR NUBES

Para modelar la obstaculización de nubes sobre las imágenes de prueba seleccionadas, se generaron máscaras sintéticas con cuatro niveles de severidad definidos en cuanto a la pérdida de información (Leve, Media, Moderada y Fuerte). Como ya se ha mencionado anteriormente, la máscara es supuesta como conocida para los algoritmos de reconstrucción desarrollados. En la Figura 4.3 se pueden observar a cada una de las máscaras definidas para los experimentos.

El problema de reconstrucción planteado hasta el momento, depende de varios parámetros que se deben elegir y que tienen influencia directa en la calidad de los resultados finales. Los primeros experimentos consistieron en la búsqueda de la optimización de dos parámetros básicos, denominados en este trabajo como K_{omp} y K_{ksvd}, que están relacionados directamente con la constante de poca densidad que existe como restricción en el algoritmo OMP. Para diferenciarlos, el K_{omp} se refiere al número de átomos que se utilizarán en el proceso de representación dispersa empleados en la reconstrucción. Por otro lado, K_{ksvd} se refiere al número de átomos que el algoritmo de entrenamiento utiliza para representar a cada uno de los bloques de imagen que se utilizan para entrenar. Otros pa-

rámetros importantes son el tamaño del bloque utilizado, tipo de diccionario, redundancia del diccionario, filtros de reconstrucción utilizados, por mencionar sólo algunos. Al buscar cada optimización, se minimiza el error variando un parámetro, dejando contante el resto de los parámetros. Esta suposición es válida si se demuestra que el problema general es convexo para el conjunto de parámetros. Con los siguientes resultados no se pretende validar esto, más bien se desea encontrar una metodología práctica que pueda extenderse a los casos en general a partir de pruebas empíricas que lo fundamenten.

4.2.1 *Selección del Parámetro* K_{omp}

Como ya se aclaró, para optimizar este parámetro se definen como constante el resto de los parámetros y se varía a K_{omp}. Por esto, se seleccionó un tamaño de bloque de 8×8, diccionario preestablecido a partir de la transformada DCT con una redundancia de 4 ($A \in \mathbb{R}^{64\times256}$) y filtro de reconstrucción basado en la media aritmética de las estimaciones repetidas en el caso de que exista solapamiento de bloques. En este sentido, se utilizó la máscara leve para ocluir a la imagen PAN1, con el objetivo de obtener las reconstrucciones respectivas variando el parámetro en $K_{omp} = \{1, 2, \ldots 20\}$; y calculando el desempeño de cada reconstrucción, evaluando los errores definidos en la Tabla 4.1. Para este experimento, se hizo una comparación entre el desempeño obtenido al utilizar solapamiento de bloques, con respecto a no utilizarlo. Además, utilizando la idea de extraer el nivel DC de la imagen antes de encontrar la representación dispersa que plantea M. Elad en [28], en la Sección 2.2, del Capítulo 15; también se estudia el comportamiento de la reconstrucción obtenida al eliminar previamente el nivel continuo de la imagen antes de encontrar la representación poco densa de la imagen ocluida. En el proceso de reconstrucción este nivel extraído es nuevamente añadido, obteniendo la estimación final.

En la Figura 4.4 se observa que un valor de K_{omp} óptimo, para minimizar el error de estimación, puede tomar valores entre 5 y 15. Sin embargo, hay que tener en cuenta de que entre mayor es el número de átomos utilizados en la representación, mayor es el consumo de tiempo para el cómputo de la estimación. Por ejemplo, para obtener la reconstrucción utilizando $K_{omp} = 5$, con solapamiento y eliminando el nivel DC, se necesitaron 230 segundo, mientras que para un $K_{omp} = 10$, utilizando los mismos parámetros, se necesitaron

(a) **(b)**

(c) **(d)**

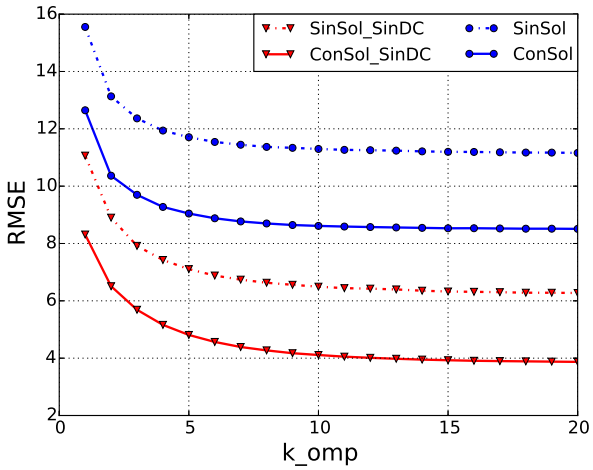
Figura 4.3. Mascaras generadas sintéticamente (a) Máscara Leve (b) Máscara Media (c) Máscara Moderada (d) Máscara Fuerte.

446 segundo. Por otro lado, observe que el desempeño mejora al utilizar un solapamiento de bloques, obteniendo una ganancia de casi 6 dB con respecto a la versión sin solapamiento de bloques. Finalmente, también hay que resaltar la mejora sustancial que existe en la reconstrucción cuando el nivel DC es extraído de la imagen antes de calcular la representación dispersa de los bloques. Observe que el desempeño de la versión tradicional, utilizando solapamiento de bloques y $K_{omp} = 5$, es mejorado por alrededor de 6 dB.

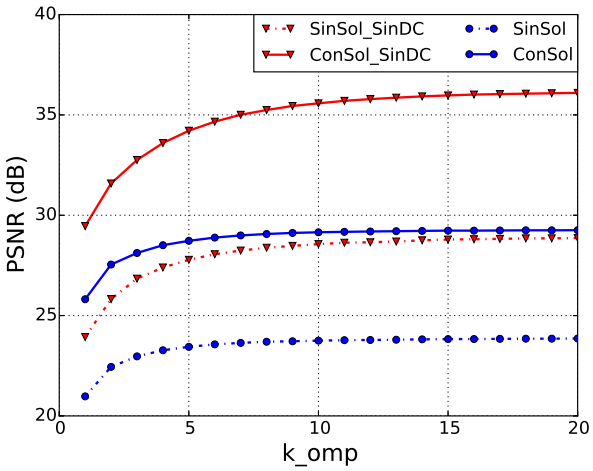
Para tener mayor soporte empírico, se repite el mismo experimento para las imágenes PAN2 y PAN3. Lo que se pretende con esto, es tratar de generalizar la tendencia en los resultados ante diferentes escenas. Los desempeños de las reconstrucciones se aprecian en la Figura 4.5, utilizando el PSNR como indicador de calidad. Observe que para ambos casos, procesar la imagen utilizando solapamiento de bloques mejora la reconstrucción, con respecto a su versión no solapada. Por otro lado, se sigue comprobando el incremento de la calidad de la reconstrucción al eliminar el nivel DC de la imagen, antes de encontrar su representación dispersa.

4.2.2 *Selección del parámetro K_{ksvd} en el entrenamiento de diccionarios*

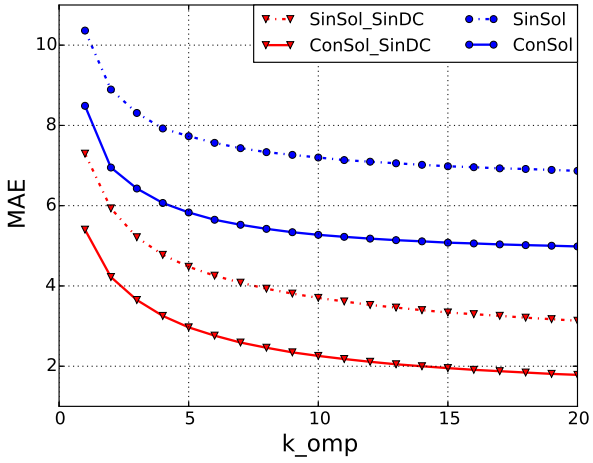
En la Sección 3.4 se comentó sobre los dos posibles enfoques de entrenamiento en cuanto al origen de las escenas que se utilizarán en el proceso. Para empezar, se evaluará el desempeño de adaptar el diccionario base a la misma imagen. Para efectuar el proceso de selección del parámetro K_{ksvd}, se diseñó un experimento basado en el anterior. Primero se dividió a la imagen PAN1 en bloques de 8×8, con el que se generaron 4096 parches no solapados de la imagen sin oclusión y 255025 bloques para el caso en que se considera solapamiento en la descomposición de la imagen. Esta situación generó dos conjunto separados de resultados, que en el proceso de análisis se denominó $SinSolapamiento_{ksvd}$ y $Solapamiento_{ksvd}$. Por otro lado, se generaron 15 diccionarios, a partir del conjunto de imágenes de entrenamiento, para cada uno de los dos tipos de descomposición. Estos diccionarios se diferencian en el número de átomos del diccionario base, empleados en el proceso de representación dispersa interna del algoritmo de entrenamiento. El diccionario base utilizado es DCT de 64×256. Con el objetivo de evaluar el comportamiento de las tendencias obtenidas en el experimento de la Sección anterior, en función del tipo de diccionario diseñado variando



(a)

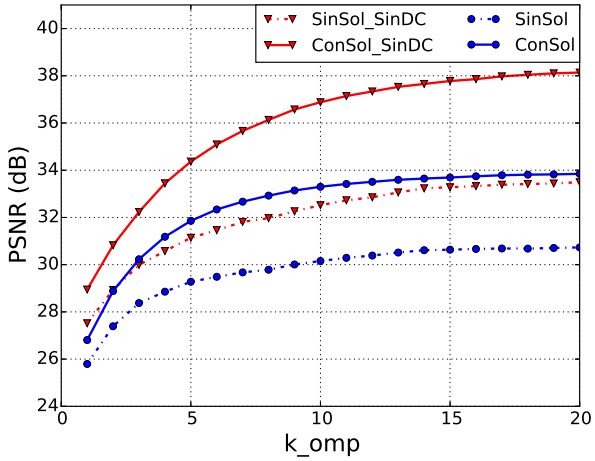


(b)

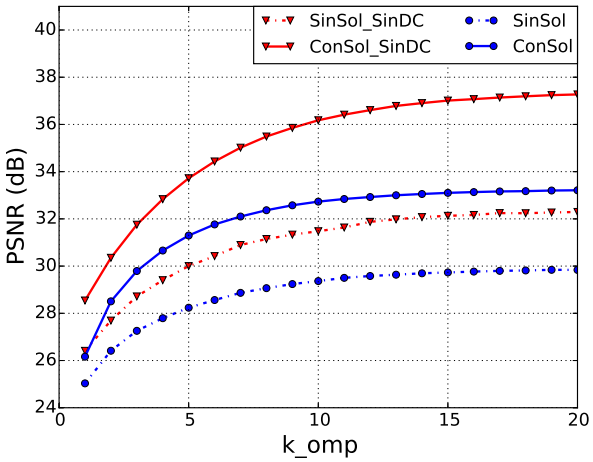


(c)

Figura 4.4. Optimización del parámetro K_{omp} para una reconstrucción de la imagen PAN1 ante una oclusión leve. (a) Error de estimación RMSE calculado a nivel de bloques. (b) Error de estimación PSNR calculado a nivel de bloques. (c) Error de estimación MAE calculado a nivel de bloques.



(a)



(b)

Figura 4.5. Optimización del parámetro K_{omp} para una reconstrucción la Imagen PAN2 y PAN3 ante una oclusión leve utilizando diccionario DCT. (a) Error de estimación PSNR calculado a nivel de bloques para la Imagen PAN2. (b) Error de estimación PSNR calculado a nivel de bloques para la Imagen PAN3.

el parámetro $K_{ksvd} = \{1, 2, \dots 15\}$, también se estudió la variación de $k_{omp} = \{1, 2, \dots 20\}$ en la calidad de las reconstrucciones. Esto generó una función de dos variables, en las que se busca optimizar ambos parámetros.

En la Figura 4.6 se muestran cuatro curvas 3D que demuestra el desempeño obtenido al reconstruir la oclusión leve en la imagen PAN1, del enfoque en el que se aplica el algoritmo de reconstrucción, eliminándole previamente el nivel DC a la imagen, utilizando el PSNR como medida de calidad. Este enfoque es el que registró mejor rendimiento, tal y como se ha demostrado en experimentos anteriores.

Los resultados mostrados indican que la forma en que se entrena el diccionario influye en la calidad de los resultados obtenidos. Estos valores de desempeño fueron calculados para toda la imagen, por lo que el rendimiento mostrado incluye a las estimaciones de los bloques que no estaban ocluidos. Para fines prácticos, el comportamiento deseado en el eje de K_{ksvd} es aquel que sea más estable, tal y como lo demuestra el comportamiento de las Figuras 4.6(a), 4.6(b)y 4.6(c). Por lo tanto, se puede inferir de este experimento controlado que para el entrenamiento es adecuado una descomposición sin solapamiento, mientras que para la estimación es óptimo utilizar solapamiento de bloques, corroborando la conclusión obtenida de la experiencia anterior. En cuanto al valor óptimo de K_{ksvd}, observe que para el comportamiento registrado en la Figura 4.6(b) la variación del parámetro no incide notablemente en el desempeño de la estimación, por lo que puede seleccionarse un valor a criterio personal. Para esta investigación, se escogió como valor óptimo $K_{ksvd} = 5$.

Por otro lado, como ya se ha mencionado, también puede utilizarse un conjunto de imágenes distintas a la de la escena, para adaptar el diccionario utilizado en la reconstrucción. En este sentido, se utilizó un conjunto de ocho imágenes de entrenamiento, que se muestran en la Figura 4.7, para adaptar la estadística de los átomos del diccionario a los bloques seleccionados aleatoriamente del conjunto. A continuación se realizó la variación del parámetro K_{omp}, tal y como se realizó en la Sección pasada. El resultado puede visualizarse en la Figura 4.8. Para el diccionario entrenado a partir del conjunto de imágenes se observa un buen desempeño al reconstruir la oclusión leve. Observe que la tendencia se ha mantenido también para este tipo de diccionario, es decir, la técnica de eliminar el nivel DC de la imagen antes de encontrar la representación dispersa, mejora la reconstrucción final. Además, la versión con solapamiento tiene un mejor desempeño para ambos enfoques

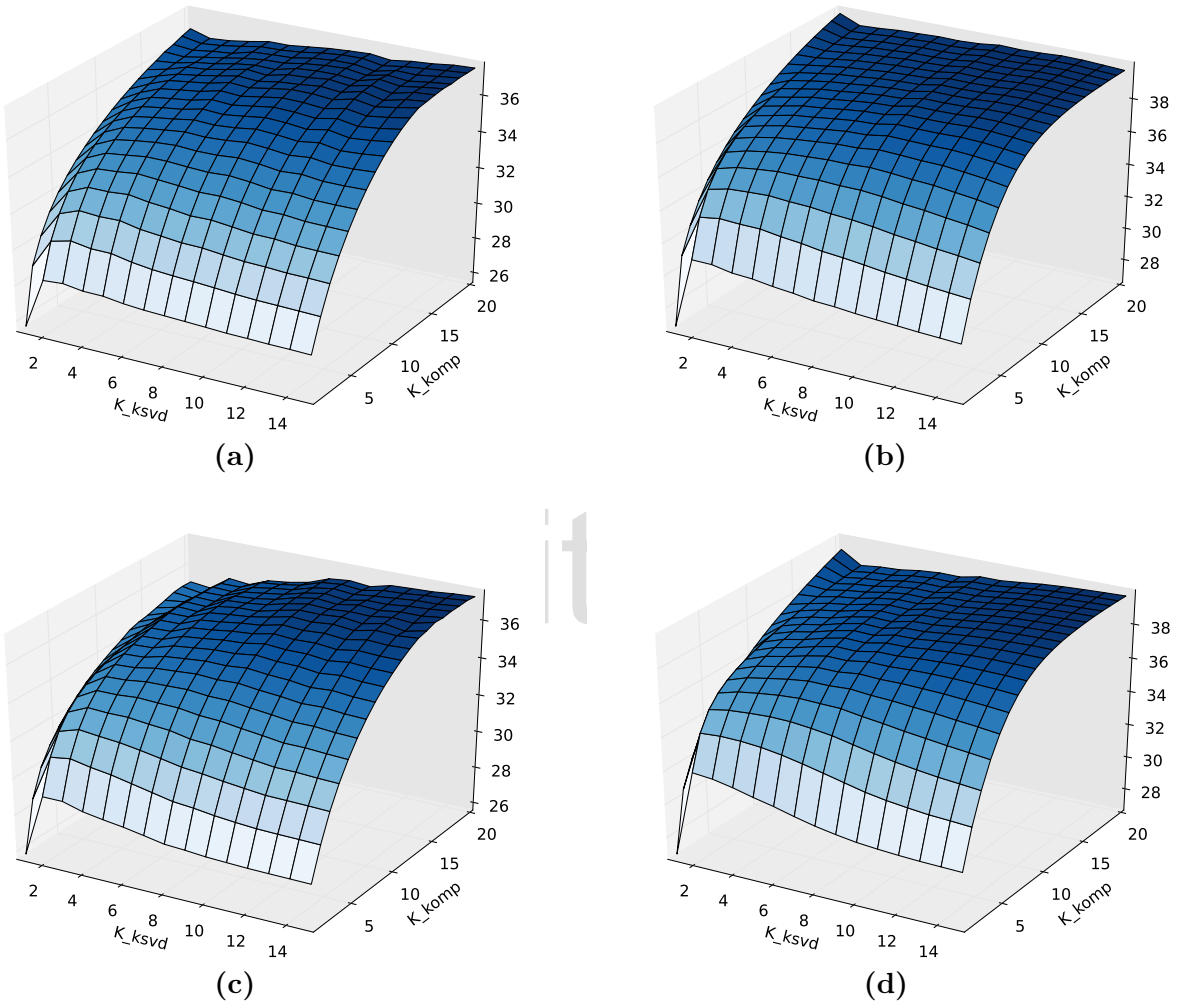


Figura 4.6. Optimización del parámetro K_{ksvd} utilizando a la misma imagen durante el entrenamiento (a) Sin solapamiento para el entrenamiento y sin solapamiento para la reconstrucción (b) Sin solapamiento para el entrenamiento y con solapamiento para la reconstrucción (c) Con solapamiento para el entrenamiento y sin solapamiento para la reconstrucción (d) Con solapamiento para el entrenamiento y con solapamiento para la reconstrucción.

de reconstrucción.

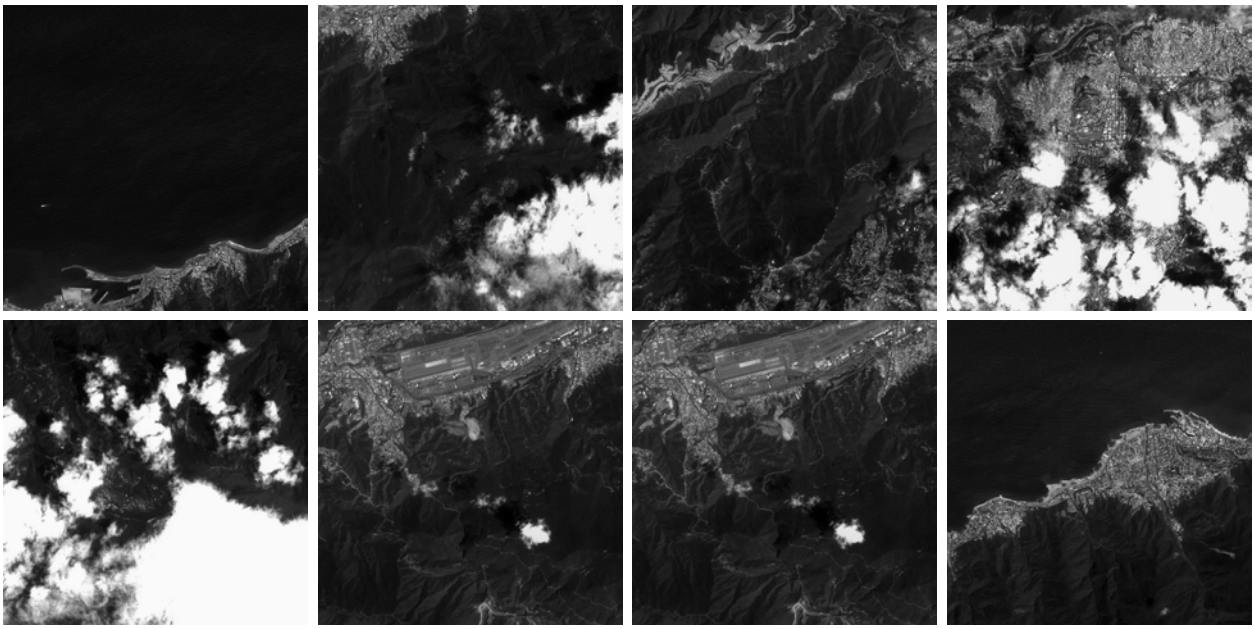
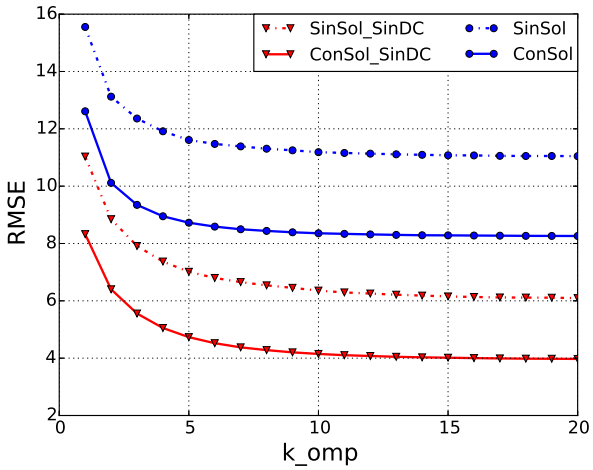
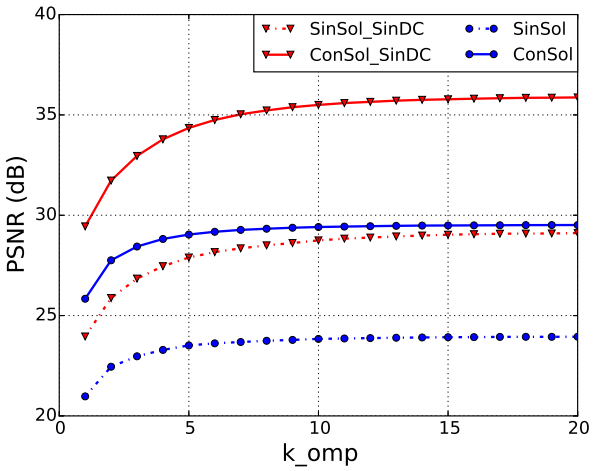


Figura 4.7. Conjunto de imágenes de entrenamiento.

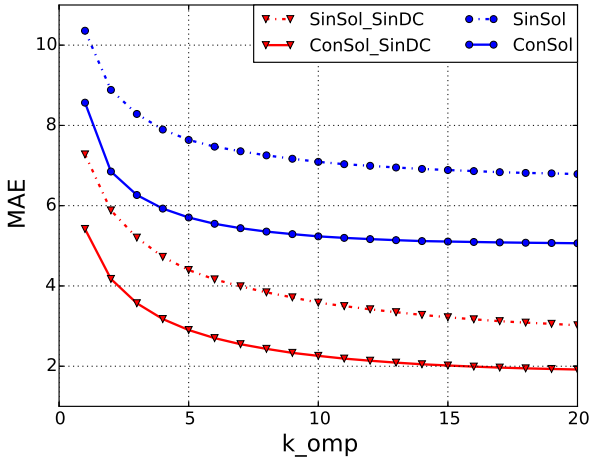
Finalmente, se puede comparar el desempeño obtenido para los mejores enfoques de las tres experiencias efectuadas. Esto se visualiza en la Figura 4.9. Para la leyenda se definió como Entrenamiento1, el logrado a partir de la misma escena a reconstruir, utilizando $k_{ksvd} = 5$ y sin solapamiento en el entrenamiento; mientras que el Entrenamiento2, se refiere al realizado de manera aleatoria a partir de un conjunto de imágenes distintas a la de la escena. De la figura puede concluirse que utilizando un diccionario entrenado mejora la reconstrucción con respecto a utilizar uno pre-establecido como el basado en DCT. Sin embargo, observe que el comportamiento de un diccionario basado en DCT tiene un desempeño bastante aceptable, teniendo en cuenta que no se requiere más computo adicional, en comparación con el proceso de entrenamiento. Por ejemplo, para $K_{omp} = 5$, se tiene un PSNR=34,35 para el Entrenamiento2, PSNR=34,60 para el Entrenamiento1 y un PSNR=34,21 para el diccionario ODCT, todos con el criterio de eliminar el nivel DC antes de encontrar la representación dispersa de la imagen. En la Figura 4.10, se ilustra este desempeño de manera visual. Por los resultados obtenidos, se justifica la utilización de un diccionario ODCT para el resto de los experimentos.



(a)

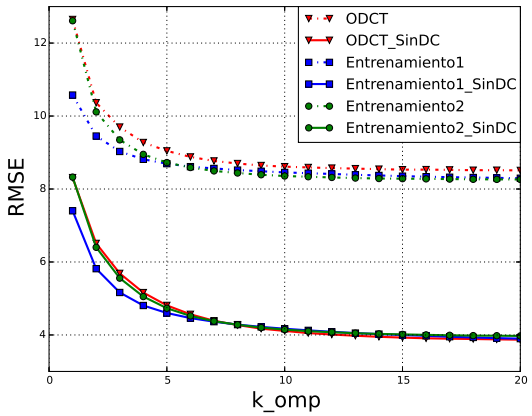


(b)

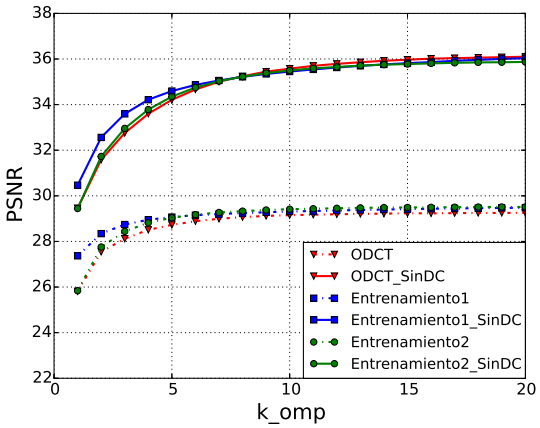


(c)

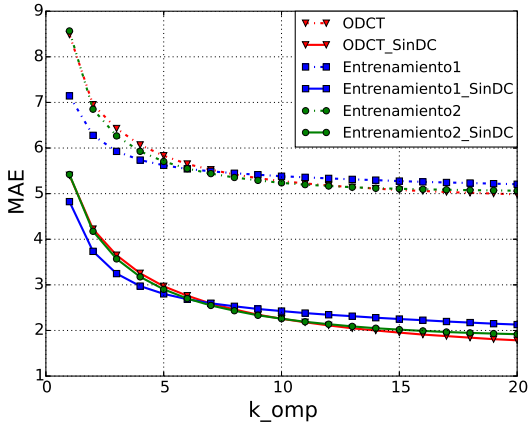
Figura 4.8. Reconstrucción de una oclusión leve en la imagen PAN1 utilizando un diccionario entrenado a partir de un conjunto aleatorio de imágenes. (a) Error de estimación RMSE calculado a nivel de bloques. (b) Error de estimación PSNR calculado a nivel de bloques. (c) Error de estimación MAE calculado a nivel de bloques.



(a)



(b)



(c)

Figura 4.9. Comparación de desempeños en la reconstrucción de la imagen PAN1 utilizando diccionario DCT, Entrenamiento con la misma imagen y entrenamiento a partir de un grupo de imágenes distintas a la de escena ante una oclusión leve . (a) Error de estimación RMSE calculado a nivel de bloques. (b) Error de estimación PSNR calculado a nivel de bloques. (c) Error de estimación MAE calculado a nivel de bloques.

(a)

(b)

(c)

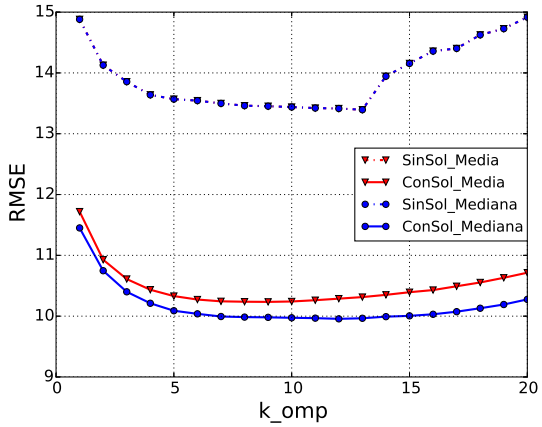
Figura 4.10. Comparación visual del desempeño de reconstrucción ante distintos diccionarios. (a) Reconstrucción utilizando Diccionario ODCT. RMSE=4,80 , PSNR=34,21 y MAE=2,97. (b) Reconstrucción utilizando Diccionario Entrenado1. RMSE= 4,60 , PSNR=34,60 y MAE=2,80. (c) Reconstrucción utilizando Diccionario Entrenado2. RMSE=4,73 , PSNR= 34,35 y MAE= 2,90.

4.2.3 *Rendimiento variando el filtro de reconstrucción*

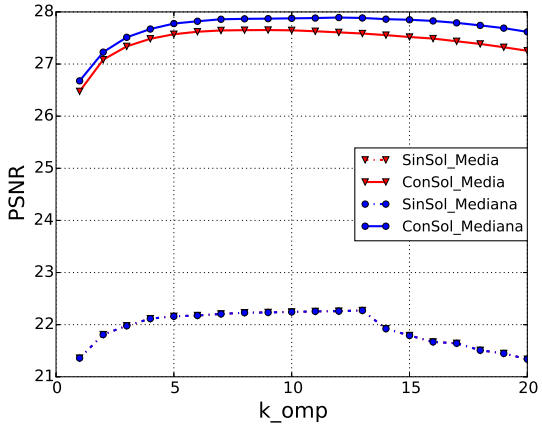
Como se fundamentó en el Capítulo 2, el filtro de reconstrucción incide en la estimación final de cada pixel ocluido. En esta Sección se diseñó un experimento, basado en el de la primera Sección, para evaluar la incidencia de este parámetro en la reconstrucción final. En la Figura 4.11, se observa una mejora de $0,3$ dB utilizando un filtro de mediana, con respecto de uno basado en la media. Sin embargo, debido a la simplicidad de cálculo que implica la estimación del promedio, se preferirá éste en la implementación final. No obstante, es bueno acotar que esto no pretende ser una demostración exhaustiva, ya que es un experimento controlado y la oclusión es pequeña. Aún así, se espera que esto pueda extenderse. Para fundamentar esto, se puede utilizar los resultados que obtuvo Portillo [3] en su investigación, en el que se observa un comportamiento similar. A pesar de que el problema abordado en dicha investigación es distinto, tiene la idea común de que a partir de múltiples estimaciones de un mismo píxel, hay que escoger un valor representativo. Para más información se recomienda referirse a los resultados presentados en dicha investigación.

4.2.4 *Evaluando la incidencia del porcentaje de pérdida de píxel en la reconstrucción*

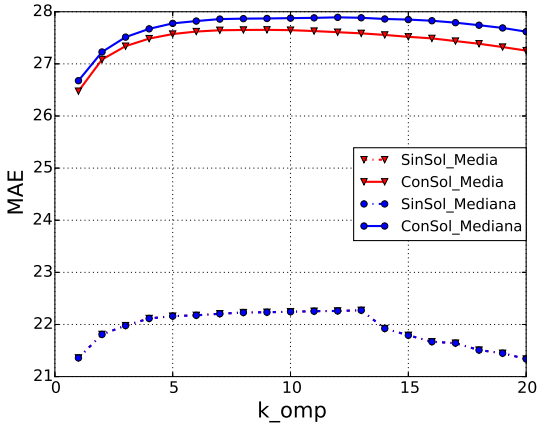
En la Sección 3.3, se introdujo la idea de que el algoritmo utilizado hasta el momento puede fallar si la pérdida de píxeles en un bloque es muy severa. En esta Sección se profundizará un poco más sobre este asunto, con el objetivo de buscar un umbral de referencia que pueda utilizarse en el algoritmo iterativo desarrollado. Un bloque reconstruido puede acercarse lo suficientemente a la imagen original, dependiendo en gran medida del tipo de imagen de la que se trate y de la intensidad con que esté degradada por la oclusión. Si una imagen es lo suficientemente "suave" como para ser representada de manera adecuada con pocos coeficientes, es posible tener una buena reconstrucción utilizando las técnicas de representación dispersa. Por otro lado, esto no será del todo cierto en imágenes con cambios más bruscos en el nivel de gris. Esto se pudo observar en la Figura 4.5, en donde el error para una imagen más suave como la PAN2 fue menor en comparación con el desempeño obtenido en las imagen PAN1 y PAN3. Para comprobar esta teoría, se elaboró un experimento que pretende ser lo suficientemente amplio como para encontrar un umbral



(a)



(b)



(c)

Figura 4.11. Comparación entre las estimaciones obtenidas al utilizar un filtro de reconstrucción basado en la media y la mediana. (a) Error de estimación RMSE calculado a nivel de bloques. (b) Error de estimación PSNR calculado a nivel de bloques. (c) Error de estimación MAE calculado a nivel de bloques.

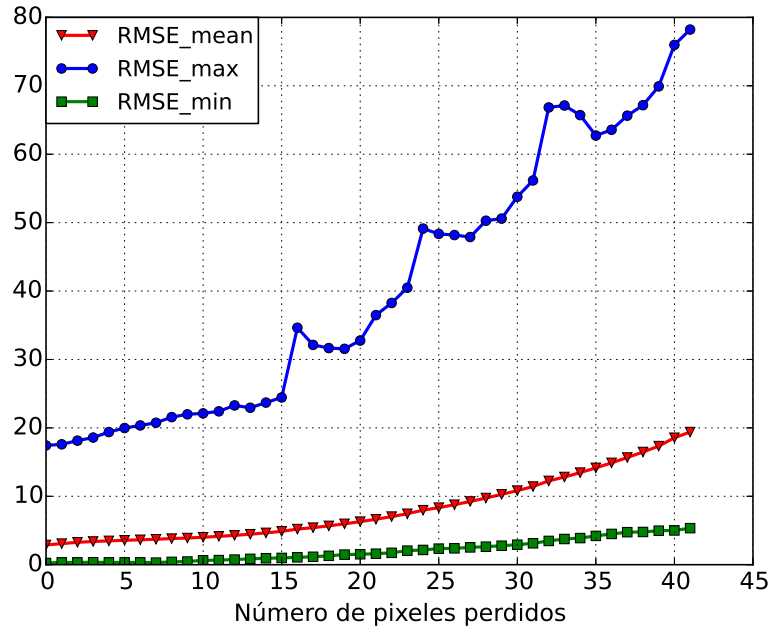
aceptable que permita estimar cualquier bloque ocluido.

El presente experimento consistió en sintetizar oclusiones automáticas en todos los bloques de una imagen por igual, para obtener un comportamiento lo suficientemente diverso como para generalizar los resultados obtenidos. La hipótesis que se desea probar es que, dado un umbral definido para la perdida de píxeles permitida en un bloque dado, la reconstrucción utilizando sensado comprimido será aceptable.

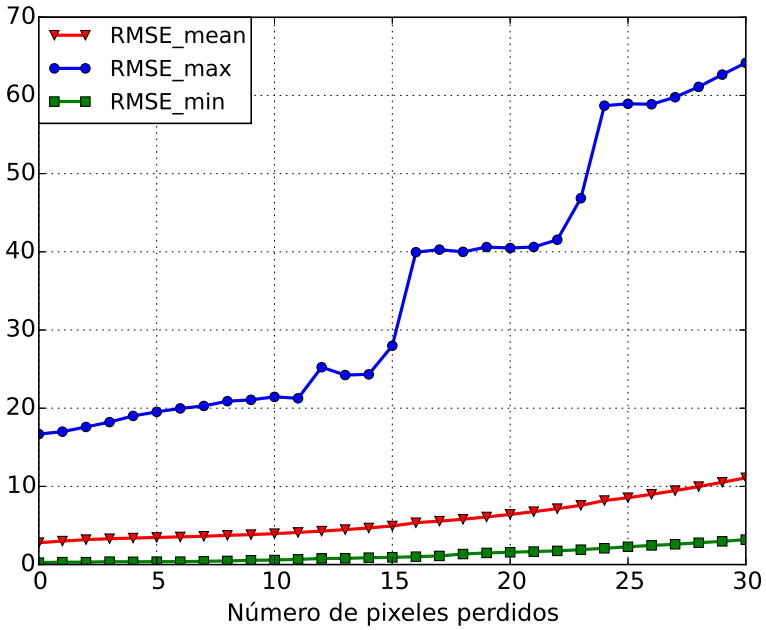
El experimento diseñado consta de dos enfoques de oclusión básica: Oclusión continua Horizontal y Oclusión continua Vertical. En concreto, la idea es ir incrementando la oclusión (píxel perdido) en una de estas direcciones e ir reconstruyendo la imagen para obtener un comportamiento Error vs Número de Píxeles perdido para cada uno de los bloques de los que se compone la imagen. Luego, dado cada uno de los resultados, calcular el promedio del error para cada pérdida de píxeles y levantar la correspondiente curva que los relacione. Para esta experiencia, se utilizó a la imagen PAN1, que fue dividida en 4096 parches de 8×8. Cada uno de los bloques fue ocluido, utilizando la misma máscara de oclusión, siguiendo alguno de los dos enfoques de obstrucción sintética descritos. Finalmente se reconstruyó cada uno utilizando solapamiento de bloques, para $K_{omp} = 5$ y filtro de reconstrucción basado en la mediana. Luego, se calcularon los errores de estimación para cada uno de los 4096 bloques de la imagen. Para cada valor de pérdida de píxeles se utilizó el promedio de las 4096 experiencias, para tener un valor representativo. También se calcularon los valores máximo y mínimo del error obtenido para cada una de las reconstrucciones con el objetivo de tener una idea del rango de variación con respecto a la media.

En la Figura 4.12 se muestran los resultados obtenidos para una oclusión sintética que se incrementa en sentido vertical y horizontal. Observe que el comportamiento es similar en ambos casos. El resultado valida la hipótesis de que entre más severa es la pérdida de píxeles, más propenso es el algoritmo a cometer un fallo. Ahora bien, ¿qué nivel de error RMSE en la variación del nivel de gris es aceptable a nivel de bloques? Este nivel será un indicio del valor del porcentaje de pérdida que se puede permitir. Por experiencias anteriores, se dedujo que un valor aceptable del RMSE es de 6 a 8. Tomando este valor de referencia, del gráfico puede inferirse que un porcentaje de pérdida aceptable ronda el 31,25 % del total de píxeles en el bloque, lo que significa que para bloques de 8×8

el umbral máximo que puede permitirse es 20 píxeles perdidos por bloque. No obstante, si se fija la atención en los otros dos valores característicos de la curva (máx. y mín.), se puede extraer algo con respecto al tipo de imagen. Existe por lo menos un bloque en el que para un nivel de pérdida de 5 píxeles, por ejemplo, el error cometido es de casi 20 niveles de gris. De manera diametralmente opuesta, existe por lo menos un bloque en el que el comportamiento es diferente, tal y como lo demuestran las curvas en color azul y verde, respectivamente. Este comportamiento se esperaba y puede atribuirse a las características propias de la imagen. Intuitivamente, los bloques que tuvieron un comportamiento cercano al descrito por la curva azul deben estar contenidos con componentes de textura, mientras que el comportamiento en verde corresponde a bloques de imágenes mas homogéneos. Aún así, el comportamiento promedio de la data generada, conlleva a pensar que es posible definir un umbral de perdida $U = 20$. No obstante, se debe tener claro que un procedimiento iterativo en el que se utilice un umbral más bajo, obteniendo un mejor desempeño en el error, puede conllevar a un mayor tiempo de cómputo. Esta experiencia se demostrará en la Sección siguiente.



(a)



(b)

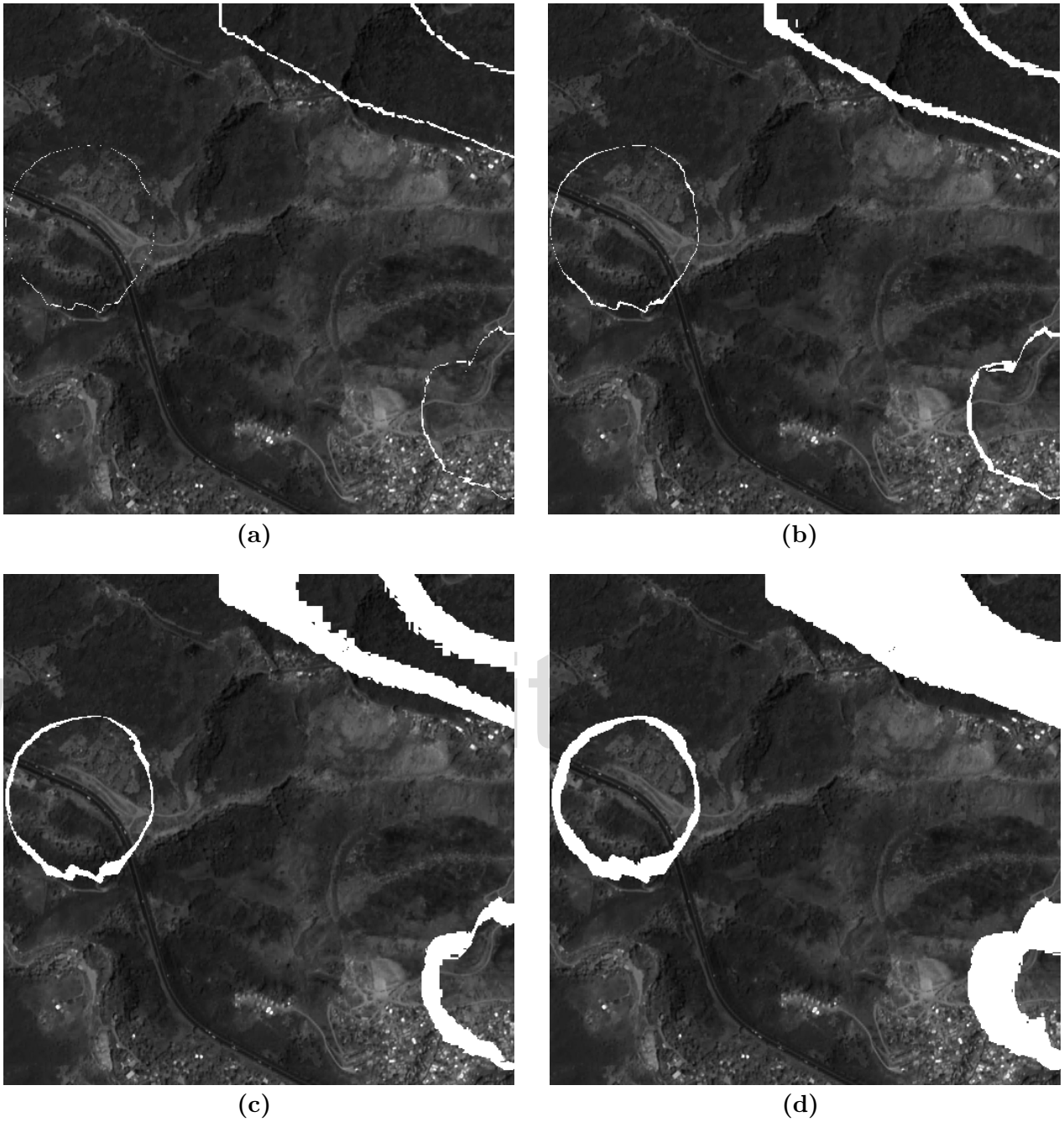
Figura 4.12. Desempeño del algoritmo de reconstrucción variando la oclusión en un bloque de 8×8 píxeles (a) Desempeño del algoritmo de reconstrucción ante una oclusión sintética horizontal variable. (b) Desempeño ante una oclusión sintética vertical variable.

4.3 ANÁLISIS CUALITATIVO

En esta Sección se presentarán los resultados visuales de las reconstrucciones de las escenas de prueba generadas ante los distintos grados de oclusión, utilizando las dos metodologías desarrolladas en el Capítulo anterior. Los parámetros escogidos son: $k_{omp} = 5$, solapamiento de bloques, filtro de reconstrucción basado en la media y diccionario ODCT con redundancia de 4. En la Figura 4.13 se observa la oclusión sintética generada por cada una de las máscaras de la Figura 4.3, sobre la imagen PAN2. Para efectos visuales, cada una de las oclusiones es mostrada en el nivel de gris blanco. Por otro lado, en la Figura 4.14 se muestra cada una de las reconstrucciones obtenidas utilizando el algoritmo de reconstrucción no iterativo. Observe que la estimación en 4.14(a) es bastante buena, con un RMSE= 3,35 niveles de gris. Por otro lado, se puede observar en las Figuras 4.14(b), 4.14(c) y 4.14(d) que el resultado empieza a empeorar a medida de que el tamaño de la obstrucción se incrementa. Por ejemplo, en la Figura 4.14(d) puede observarse que la reconstrucción falla en todas las áreas que están afectadas. Por otro lado, hay que destacar que un RMSE =10,36 por bloque implica que en la salida puede apreciarse parte de la oclusión, por lo que el valor de RMSE escogido para el umbral en la Sección anterior parece aceptable.

Por otro lado, en las Figuras 4.15 y 4.16 se puede observar el resultado de la reconstrucción iterativa en la imagen PAN3, para un umbral definido de 20 píxeles perdidos como máximo en cada bloque de la imagen en cada una de las iteraciones. Se puede observar la reconstrucción progresiva en cada iteración y la máscara resultante. Observe que la oclusión es reconstruida de manera satisfactoria en comparación con la versión que no es iterativa, tal y como puede observarse en la Figura 4.17. Sin embargo, hay que destacar que el tiempo de procesamiento se incrementa en función del número de iteraciones que se realicen para corregir la obstrucción.

A continuación, se presenta un experimento en el que es reconstruida la oclusión moderada que ha sido diseñada sintéticamente, utilizando este algoritmo de reconstrucción iterativa. Para este experimento, se varió el umbral permitido para U=[5, 10, 15, 20] para observar el desempeño del algoritmo ante variaciones del mismo. La Figura 4.18 muestra el resultado de reconstrucción de la imagen PAN1 ocluida por la máscara moderada. Observe que a medida de que el umbral es más pequeño, el algoritmo tiende a distorsionar el

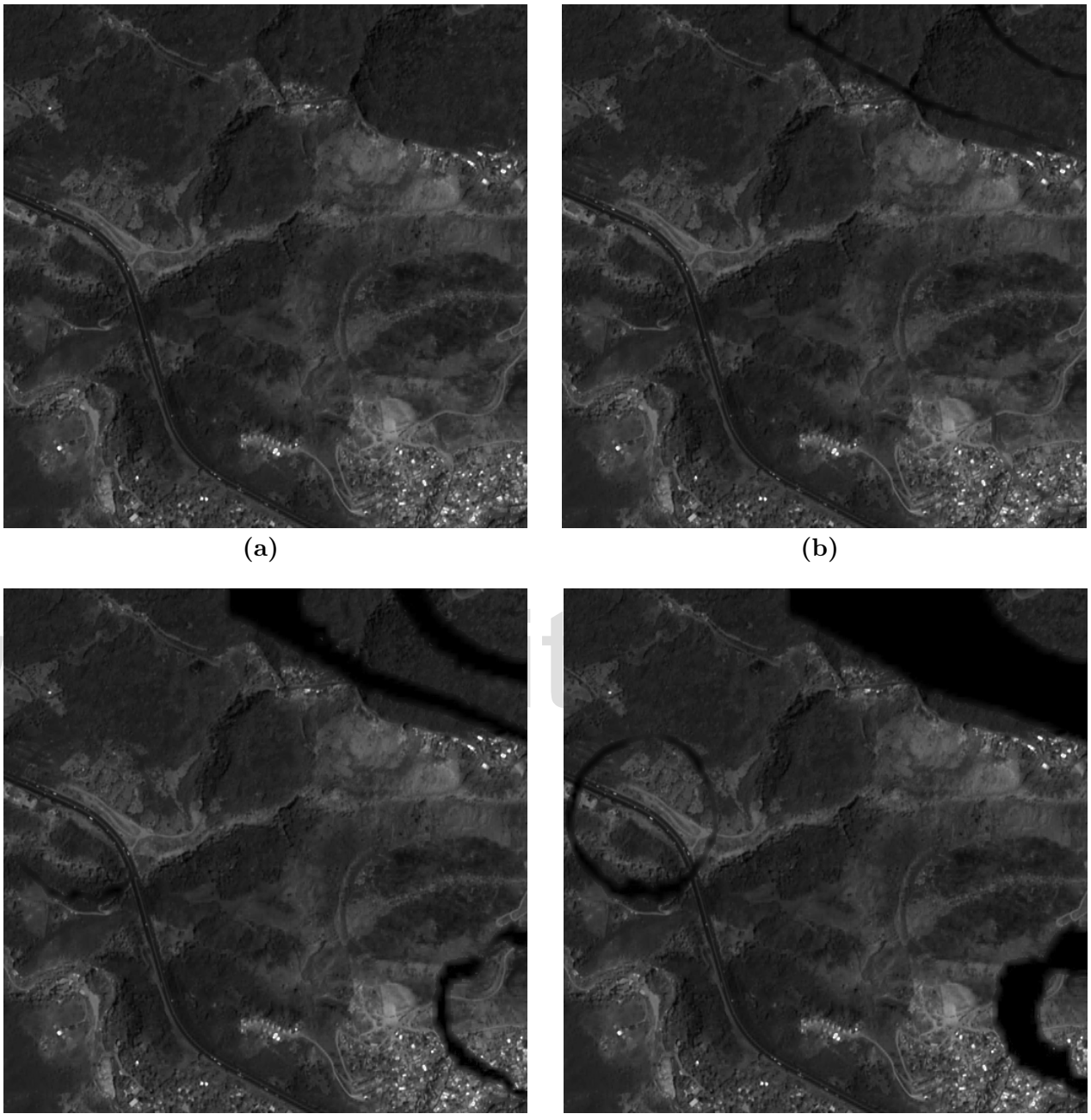


(a) **(b)**

(c) **(d)**

Figura 4.13. Oclusión generada sobre la imagen PAN2 al aplicar cada una de las máscaras sintéticas. (a) Oclusión utilizando la máscara Leve. (b) Oclusión utilizando la máscara Media. (c) Oclusión utilizando la máscara Moderada. (d) Oclusión utilizando la máscara Fuerte.

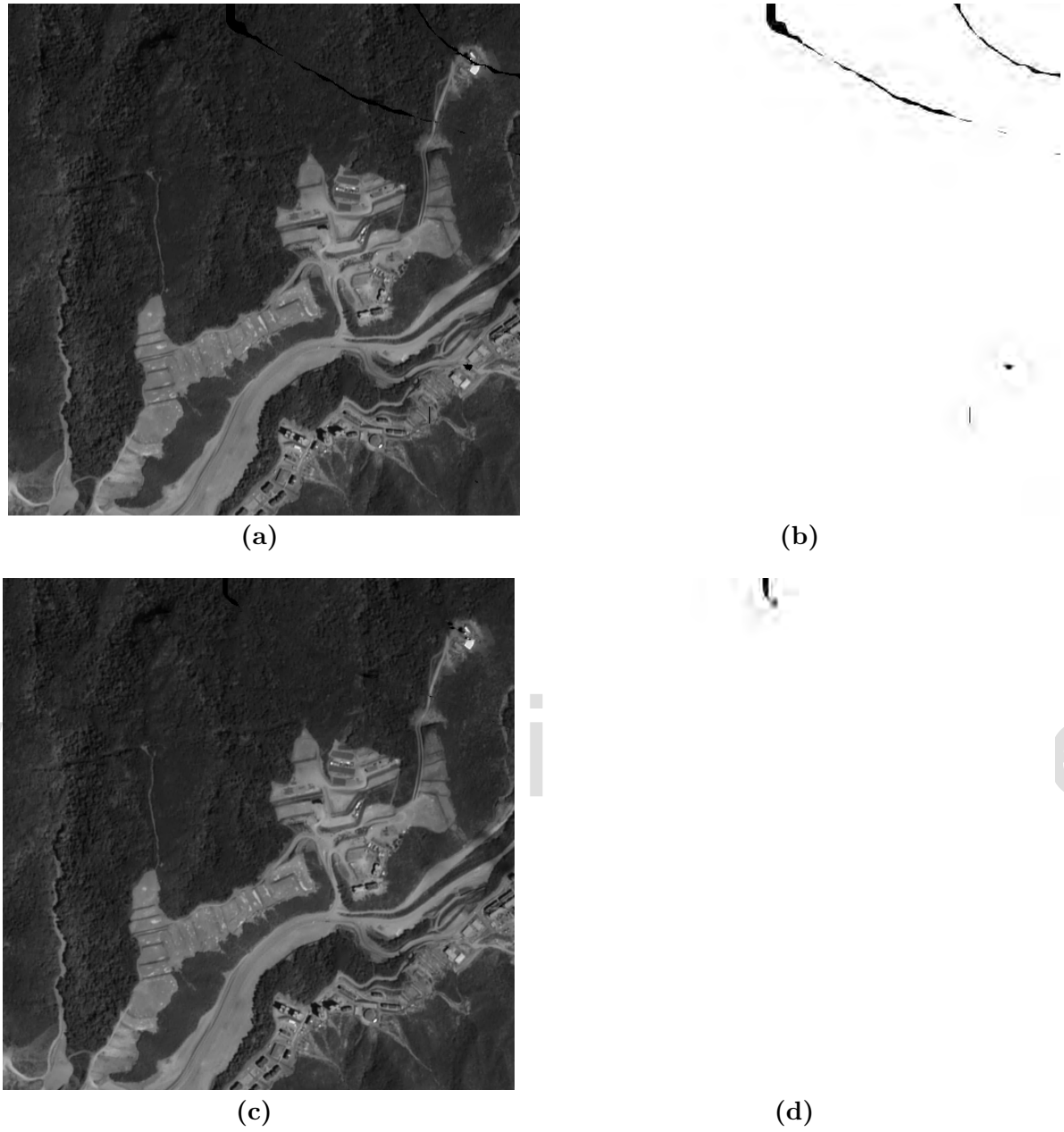
resultado, debido a que difumina la información local hacia dentro de la región ocluida, tal y como ocurre con los métodos difusos de *inpainting* tradicional. Además, al algoritmo le toma mayor número de iteraciones en terminar de estimar cada píxel perdido. Por ejem-



(a) **(b)**

(c) **(d)**

Figura 4.14. Reconstrucción de la imagen PAN2, para cada una de las máscaras sintéticas. Todos los errores fueron calculados sólo en las áreas ocluidas y a nivel de bloque. (a) Reconstrucción para la máscara leve. RMSE= 3,35, PSNR=37,62 dB y MAE=0,80 (b) Reconstrucción para la máscara media. RMSE= 10,36, PSNR= 27,82 dB y MAE=4,64 (c) Reconstrucción para la máscara moderada. RMSE= 27,46, PSNR=19,36 dB y MAE=18,05 (d) Reconstrucción para la máscara severa. RMSE= 37,03, PSNR=16,76 dB y MAE=28,95.

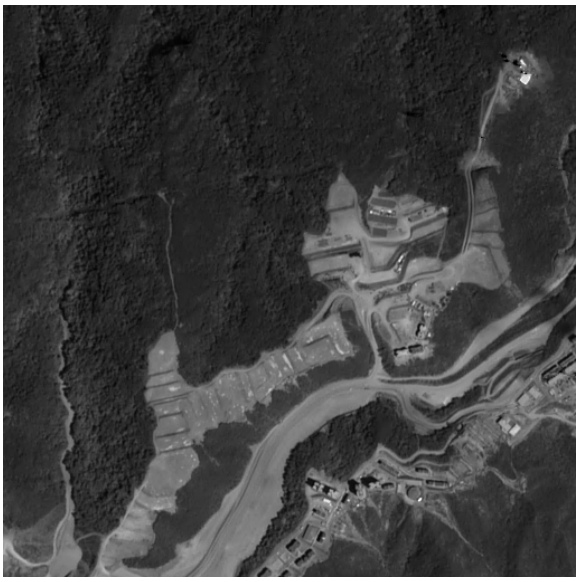


(a) **(b)**

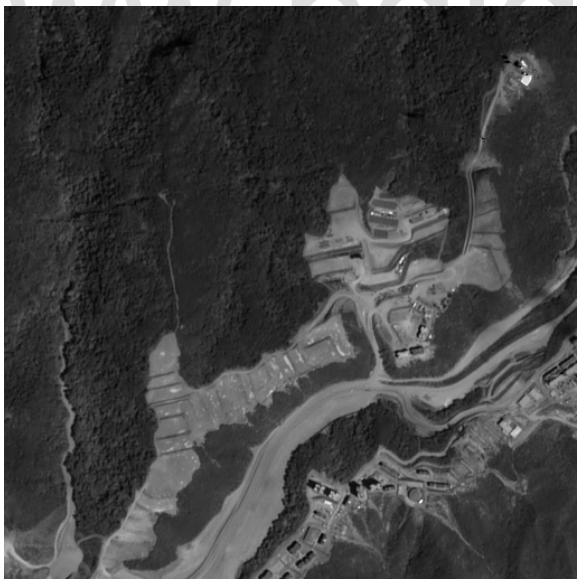
(c) **(d)**

Figura 4.15. Reconstrucción progresiva de la oclusión media en la imagen PAN3 para un umbral U=20. (a) y (b) Reconstrucción obtenida en la primera iteración y su máscara resultante, respectivamente. (c) y (d) Resultado de la segunda iteración.

plo, observe que para U=5, el tiempo de cómputo es de 2014,8 segundo, requiriendo 15 iteraciones para completar la tarea. Esto se puede contrastar con el resultado utilizando U=20, en el que el número de iteraciones requeridas es 4, utilizando un tiempo total de cómputo de 559,9 segundo. Por otro lado, se puede observar que el resultado va mejorando a medida de que el umbral se incrementa, permitiendo que menos píxeles que fueron esti-

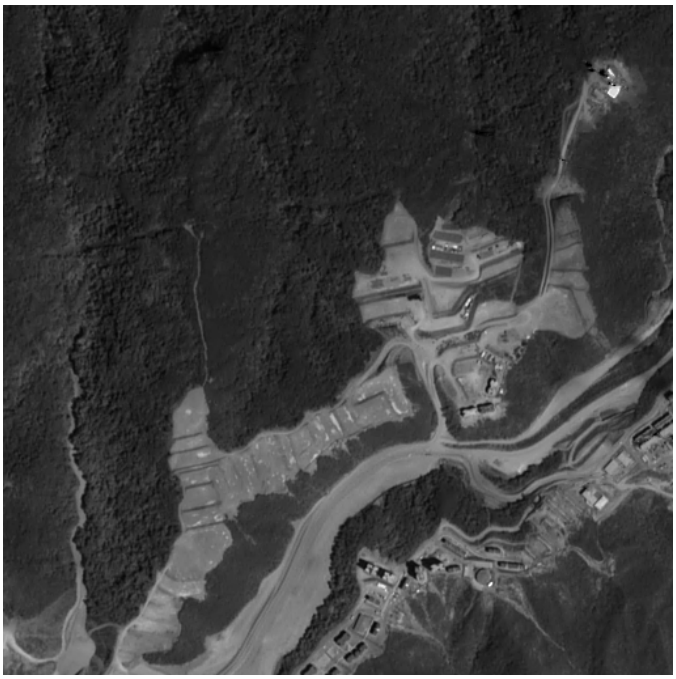


(a) (b)



(c) (d)

Figura 4.16. Reconstrucción progresiva de la oclusión media en la imagen PAN3 para un umbral U=20. (a) y (b) Reconstrucción obtenida en la tercera iteración y su máscara resultante, respectivamente. (c) y (d) Resultado de la cuarta iteración.

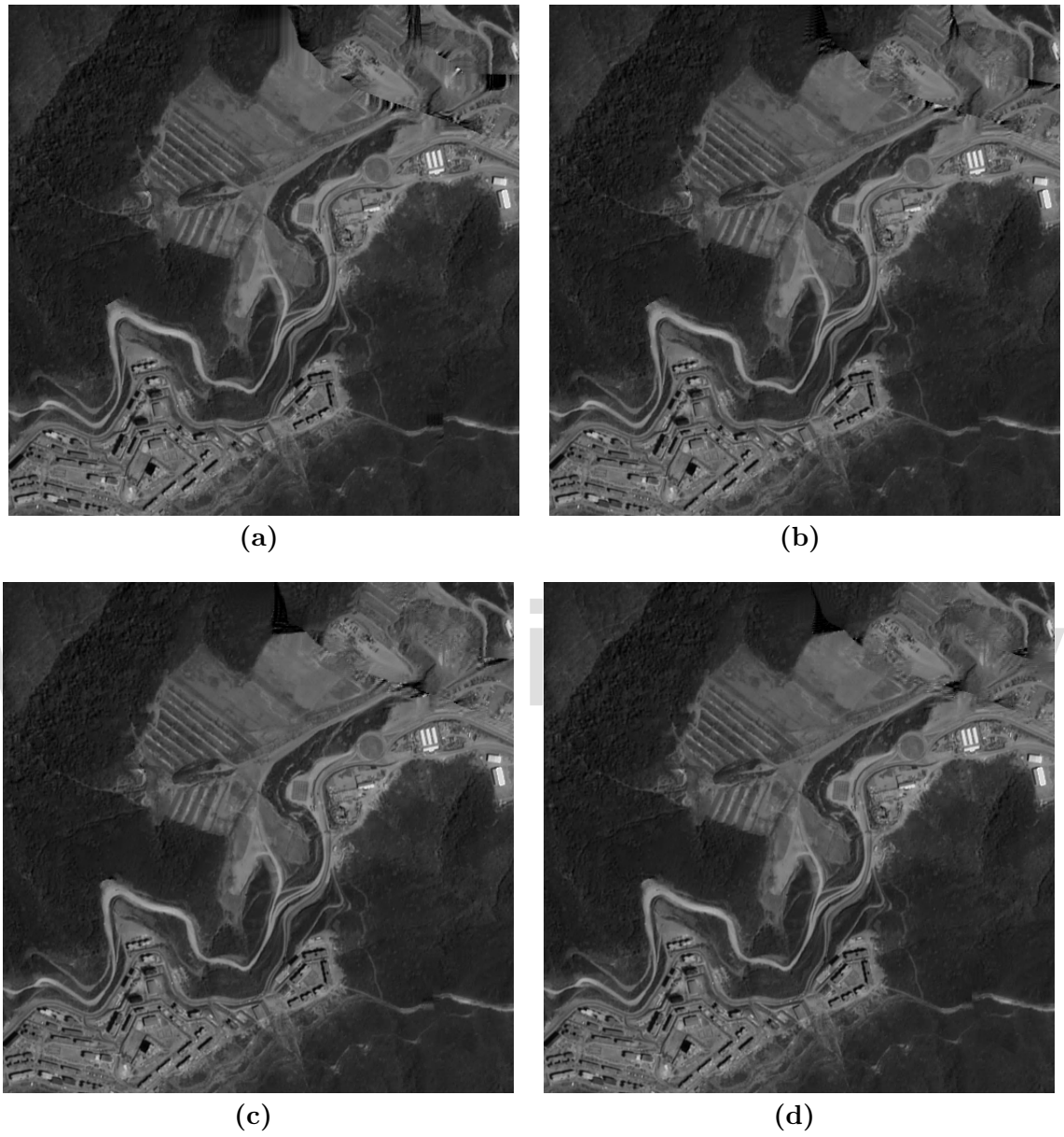


(a)

(b)

Figura 4.17. Comparación visual entre la reconstrucción iterativa y la versión no recursiva para la máscara media. (a) Reconstrucción con U=20, RMSE= 8,18, PSNR=29,87 y MAE=2,64. (b) Reconstrucción no recursiva, RMSE= 12,10, PSNR=26,47 y MAE=5,35.

mados en iteraciones anteriores incidan en la estimación de nuevos píxeles. Aún así, puede observarse que aparecen artefactos en la escena, producto de la interpolación recursiva.



(a) **(b)**

(c) **(d)**

Figura 4.18. Reconstrucción iterativa de la imagen PAN4 ante una máscara Moderada. (a) Reconstrucción utilizando un Umbral=5, 15 iteraciones y un tiempo de reconstrucción total de 2014,8 segundo. (b) Reconstrucción utilizando un Umbral=10, 7 iteraciones y un tiempo de reconstrucción total de 945,7 segundo. (c) Reconstrucción utilizando un Umbral=15, 6 iteraciones y un tiempo de reconstrucción total de 819,3 segundo. (d) Reconstrucción utilizando un Umbral=20, 4 iteraciones y un tiempo de reconstrucción total de 559,9 segundo.

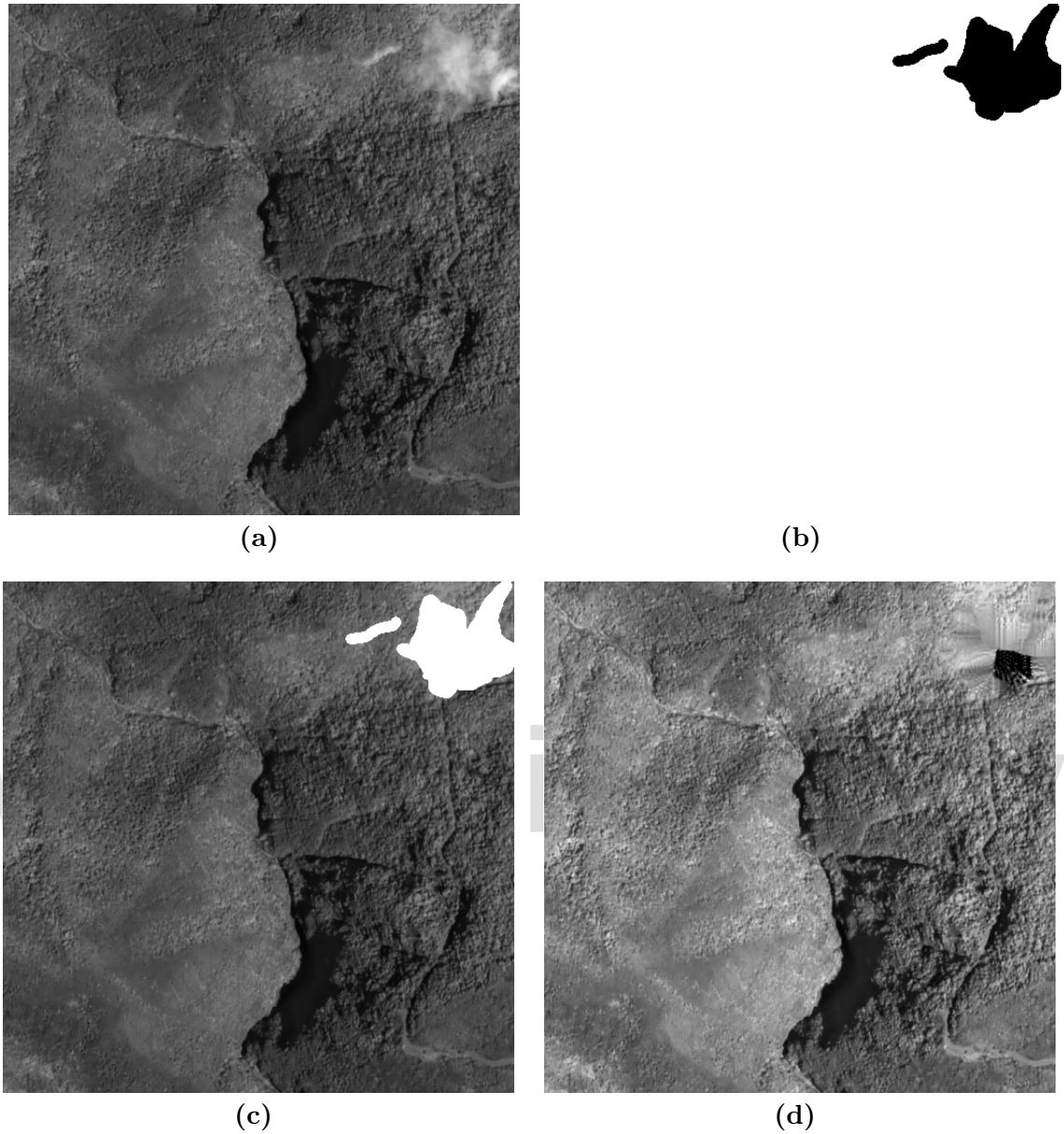
Para finalizar esta sección, se presenta un ejemplo práctico en un escenario real. La

Figura 4.19(a) muestra una zona montañosa cercana a Catia La Mar ocluida por una nube. En la Figura 4.19(b) se encuentra la máscara, resultado de un proceso de detección de nubes manual o automático. En la Figura 4.19(c) se observa la oclusión en la imagen en el nivel de gris blanco para efectos ilustrativos. Esta es la imagen de entrada al algoritmo de reconstrucción, así como la máscara que señala las áreas ocluidas y que siempre se ha supuesto conocida. En la figura 4.19(d) se muestra el resultado obtenido de aplicar el algoritmo iterativo diseñado, con un umbral U=20. Para obtener este resultado automático, se requirieron 13 iteraciones, con un tiempo total de 1659,3 segundo. Observe que, debido al tamaño de la oclusión, la reconstrucción para la nube más grande, genera artefactos visuales no deseados, así como una pérdida de detalles que es visualmente notable. Sin embargo, también se puede apreciar la calidad de reconstrucción obtenida para la nube pequeña. Esto demuestra una vez más que el desempeño de la metodología propuesta es adecuada para oclusiones pequeñas, ya que producen resultados visualmente coherentes con el resto de la imagen.

4.4 DESEMPEÑO COMPUTACIONAL

Con la idea de realizar un estudio básico de la complejidad computacional que representa la aplicación de las técnicas que se han presentado, a continuación se detallarán los tiempos de ejecución obtenidos durante la generación de resultados. Las rutinas fueron ejecutadas en un PC Linux Centos 7, procesador Intel Core i-3240 3,4 GHz con 4 GB de memoria RAM. En la Tabla 4.2 se muestran los tiempos de cómputo requerido para reconstruir una imagen de 512×512 en función del número de átomos requeridos para la reconstrucción. Se observa también el tiempo de cómputo para el caso de utilizar solapamiento en la descomposición de la imagen y para el caso en que no se utiliza. Se puede constatar que el tiempo de cómputo para el caso en el que se utiliza un solapamiento completo es considerablemente mayor en comparación con la opción sin solapamiento. Además, dichos desempeños se incrementan en función de K_{omp}, lo cual justifica la selección del valor óptimo para K_{omp} escogido en secciones anteriores.

Por otro lado, en las Tablas 4.3, 4.4, 4.5 y 4.6 se muestra el desempeño en términos del tiempo empleado por la reconstrucción iterativa para la máscara sintética Moderada



(a) **(b)**

(c) **(d)**

Figura 4.19. Ejemplo real de desoclusión de una imagen satelital obstruida por una nube. (a) Imagen ocluida por una nubes. (b) Máscara que indica la ubicación de las nubes. (c) Imagen de entrada al algoritmo, con las zonas ocluidas señaladas en color blanco. (d) Reconstrucción obtenida con el algoritmo iterativo con U=20.

en función de los umbrales definidos que fueron presentados en la Figura 4.18. Para este experimento se utilizó solapamiento de bloques y un $K_{omp} = 5$. En las mismas Tablas se puede detallar el tiempo utilizado por cada iteración, así como los bloques ocluidos remanentes que cumplen el umbral en cada una de las iteraciones. Observe que el tiempo de reconstrucción es aproximadamente constante para cada iteración.

Tabla 4.2. Tiempos de cómputo para el algoritmo de reconstrucción no iterativo en función de K_{omp}.

Komp	Sin Solp (s)	Con Solp (s)
1	1	110
2	2	137
3	3	165
4	3	199
5	4	234
6	4	273
7	5	313
8	6	358
9	6	407
10	7	456
11	8	512
12	9	572
13	10	537
14	11	707
15	12	783
16	13	862
17	15	951
18	16	1045
19	18	1147
20	19	1253

Tabla 4.4. Desempeño en tiempo de cómputo para la reconstrucción iterativa utilizando un Umbral=10.

Iteración	Tiempo (s)	Nº de Bloques (s)
1	142,0	6392
2	135,8	3303
3	134,5	2100
4	132,9	303
5	132,6	142
6	132,0	60
7	135,9	40

Tabla 4.3. Desempeño en tiempo de cómputo para la reconstrucción iterativa utilizando un Umbral=5.

Iteración	Tiempo (s)	Nº de Bloques (s)
1	139,8	3148
2	139,0	2476
3	138,5	1910
4	134,8	1467
5	134,7	770
6	136,1	414
7	134,5	214
8	131,7	125
9	134,6	77
10	132,0	71
11	130,3	62
12	133,1	43
13	130,7	32
14	133,2	23
15	131,8	8

Tabla 4.5. Desempeño en tiempo de cómputo para la reconstrucción iterativa utilizando un Umbral=15.

Iteración	Tiempo (s)	Nº de Bloques (s)
1	148,1	8283
2	135,6	3832
3	132,9	761
4	133,7	183
5	136,1	75
6	133,0	32

Tabla 4.6. Desempeño en tiempo de cómputo para la reconstrucción iterativa utilizando un Umbral=20.

Iteración	Tiempo (s)	Nº de Bloques (s)
1	148,8	9812
2	139,8	3848
3	136,8	235
4	134,4	88

CONCLUSIONES

A partir de los resultados obtenidos y de la información recavada a lo largo de este trabajo se llegó a las siguientes conclusiones:

- Las técnicas de reconstrucción basadas en sensado comprimido que se presentaron en este trabajo pueden utilizarse para restaurar imágenes satelitales, siempre que las áreas ocluidas no sean muy extensas. Esto se debe al alto contenido de detalles que contienen las imágenes imagen pancromáticas utilizadas para evaluar el desempeño.

- Un algoritmo iterativo, permite atacar de forma aceptable, regiones un poco más grandes en comparación con la versión no iterativa. Sin embargo, ante oclusiones muy severas, la reconstrucción tiende a producir artefactos indeseados en la imagen. Por lo que sigue siendo inadecuado para grandes regiones ocluidas.

- Un diccionario basado en la transformada DCT posee un rendimiento bastante aceptable con respecto a los diccionarios entrenados, ya que no se requiere cómputo adicional, como en el caso de utilizar un diccionario entrenado, en el que se debe realizar un procedimiento de entrenamiento previo.

- El tipo de imagen que es procesada influye en la calidad de reconstrucción obtenida. Una imagen con cambios suaves tendrá una mejor estimación, utilizando las técnicas expuestas, que una imagen con más detalles. Esto permite clasificar a esta metodología de reconstrucción en el campo de los métodos difusos.

- Eliminar el nivel DC de la señal a reconstruir antes de encontrar su representación dispersa tiene un mejor desempeño en la calidad de la reconstrucción obtenida utilizando representación dispersa de imágenes.

RECOMENDACIONES

- Debido a que el problema de reconstrucción presentado depende de varios parámetros que no fueron estudiados, se propone efectuar un estudio más profundo sobre la dependencia de la calidad de estimación ante la variación de los mismos. Por ejemplo, se recomienda utilizar tamaños de bloques más grandes para atacar oclusiones más severas.

- Incorporar un proceso automático de detección de zonas ocluidas en el algoritmo de reconstrucción.

- Explorar una metodología en la que se divida a la imagen en sus elementos de texturas y estructuras, utilizando un análisis de componentes morfológicas (MCA), para restaurar cada parte por separado usando los algoritmos expuestos en esta investigación.

- Combinar esta metodología con alguno de los enfoques basados en la correlación temporal de imágenes de una misma escena, para mejorar el desempeño obtenido ante grandes oclusiones.

REFERENCIAS

[1] O. Jerick, "Restauración de imágenes usando técnicas de restauración de sensado comprimido," Trabajo de Grado, Universidad de Los Andes, Noviembre 2012.

[2] G. Juan, "Desarrollo de herramientas computacionales de detección de objetos de interés en imágenes satelitales usando representación poco densa de señales en diccionarios redundantes," Trabajo de Grado, Universidad de Los Andes, Diciembre 2015.

[3] P. Luis, "Desarrollo de herramientas computacionales para superresolución imágenes satelitales usando representación poco densa de señales en diccionarios redundantes," Trabajo de Grado, Universidad de Los Andes, Mayo 2016.

[4] ABAE, "Tecnología espacial en venezuela satélite miranda (VRSS-1)," Mayo 2014.

[5] T. Statella y E. A. da Silva, "Shadows and clouds detection in high resolution images using mathematical morphology," *Pecora 17-The Future of Land Imaging*, pp. 18–20, 2008.

[6] C. Huang, N. Thomas, S. N. Goward, J. G. Masek, Z. Zhu, J. R. Townshend, y J. E. Vogelmann, "Automated masking of cloud and cloud shadow for forest change analysis using landsat images," *International Journal of Remote Sensing*, vol. 31, no. 20, pp. 5449–5464, 2010.

[7] J. D. Braaten, W. B. Cohen, y Z. Yang, "Automated cloud and cloud shadow identification in landsat mss imagery for temperate ecosystems," *Remote Sensing of Environment*, vol. 169, pp. 128–138, 2015.

[8] K.-H. Lai y C.-H. Lin, "An optimal boundary determination approach for cloud removal in satellite image," en *2012 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2012, pp. 2241–2244.

[9] F. Melgani, "Contextual reconstruction of cloud-contaminated multitemporal multispectral images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 2, pp. 442–455, 2006.

[10] C.-H. Lin, P.-H. Tsai, K.-H. Lai, y J.-Y. Chen, "Cloud removal from multitemporal satellite images using information cloning," *IEEE transactions on geoscience and remote sensing*, vol. 51, no. 1, pp. 232–241, 2013.

[11] P. Rakwatin, W. Takeuchi, y Y. Yasuoka, "Restoration of aqua modis band 6 using histogram matching and local least squares fitting," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 2, pp. 613–627, 2009.

[12] F. Chun, M. Jian-wen, D. Qin, y C. Xue, "An improved method for cloud removal in aster data change detection," en *Geoscience and Remote Sensing Symposium, 2004. IGARSS'04. Proceedings. 2004 IEEE International*, vol. 5. IEEE, 2004, pp. 3387–3389.

[13] D. P. Roy, J. Ju, P. Lewis, C. Schaaf, F. Gao, M. Hansen, y E. Lindquist, "Multitemporal modis–landsat data fusion for relative radiometric normalization, gap filling, and prediction of landsat data," *Remote Sensing of Environment*, vol. 112, no. 6, pp. 3112–3130, 2008.

[14] M. Bertalmio, G. Sapiro, V. Caselles, y C. Ballester, "Image inpainting," en *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 417–424.

[15] L. A. Vese y S. J. Osher, "Modeling textures with total variation minimization and oscillating patterns in image processing," *Journal of scientific computing*, vol. 19, no. 1-3, pp. 553–572, 2003.

[16] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, y J. Verdera, "Filling-in by joint interpolation of vector fields and gray levels," *IEEE transactions on image processing*, vol. 10, no. 8, pp. 1200–1211, 2001.

[17] M. Bertalmio, A. L. Bertozzi, y G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," en *Computer Vision and Pattern Recognition, 2001. CVPR*

2001. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 1. IEEE, 2001, pp. I–355.

[18] C. Ballester, V. Caselles, y J. Verdera, "Disocclusion by joint interpolation of vector fields and gray levels," *Multiscale Modeling & Simulation*, vol. 2, no. 1, pp. 80–123, 2003.

[19] Z. Wang, F. Zhou, y F. Qi, "Inpainting thick image regions using isophote propagation," en *2006 International Conference on Image Processing*. IEEE, 2006, pp. 689–692.

[20] A. Criminisi, P. Perez, y K. Toyama, "Object removal by exemplar-based inpainting," en *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2. IEEE, 2003, pp. II–721.

[21] A. Criminisi, P. Pérez, y K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Transactions on image processing*, vol. 13, no. 9, pp. 1200–1212, 2004.

[22] J. Wu y Q. Ruan, "Object removal by cross isophotes exemplar-based inpainting," en *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 3. IEEE, 2006, pp. 810–813.

[23] I. Drori, D. Cohen-Or, y H. Yeshurun, "Fragment-based image completion," en *ACM Transactions on graphics (TOG)*, vol. 22, no. 3. ACM, 2003, pp. 303–312.

[24] M. Bertalmio, L. Vese, G. Sapiro, y S. Osher, "Simultaneous structure and texture image inpainting," *IEEE transactions on image processing*, vol. 12, no. 8, pp. 882–889, 2003.

[25] L. Lorenzi, F. Melgani, y G. Mercier, "Inpainting strategies for reconstruction of missing data in vhr images," *IEEE Geoscience and Remote Sensing Letters*, vol. 8, no. 5, pp. 914–918, 2011.

[26] D. L. Sophia, K. Lalitha, y J. P. Chandar, "Reconstruction of cloud contaminated remote sensing images using inpainting strategy," *Int J Electron Comput Commun Technol*, vol. 3, no. 3, pp. 407–411, 2013.

[27] J. A. Tropp, "Topics in sparse approximation," Tesis de Ph.D., The University of Texas at Austin, Austin, Ago. 2004.

[28] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing.* Haifa, Israel: Springer, 2010.

[29] M. Aharon, M. Elad, y A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4311–4322, 2006.

[30] J.-L. Starck, M. Elad, y D. L. Donoho, "Image decomposition via the combination of sparse representations and a variational approach," *IEEE transactions on image processing*, vol. 14, no. 10, pp. 1570–1582, 2005.

[31] M. Elad y M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *Image Processing, IEEE Transactions on*, vol. 15, no. 12, pp. 3736–3745, 2006.

[32] M. Elad, M. Series, y M. Elad, "Five lectures on sparse and redundant representations modelling of images."

ANEXOS

Algoritmo de Reconstrucción de Oclusiones

Implementación computacional de los algoritmos propuestos en esta investigación para reconstruir áreas ocluidas de una imagen satelital.

- **Paquetes requeridos**
  - `Python 2.7`
  - `numpy`
- **Parámetros de entrada**
  - Imagen Ocluida ($\mathbf{I}_{obv}$)
  - Máscara ($\mathbf{M}$)
- Salida
  - Imagen estimada


```
import numpy as np
from matplotlib import pyplot as plt
from PIL import Image as im
from dicttools import im2col
from utilites import *
from odctdict import odct2dict
from omp_hologra import omph
from math import sqrt
from time import ctime
from scipy import misc
from skimage import img_as_float



def genargs(a):
    #Genera un diccionario de Python con los argumentos por defecto de la funcion Restauracion.


return {'Iocluida': None, 'Mascara':None,'Tbloque':8, 'Dict': None, 'Komp':5, 'Solp': 'Con','Trecons':'I', 'U'

def Train_genargs(a):
    # Funcion para generar un diccionario de python con las llaves o parametros
    # que necesitara la funcion de entrenamiento
    new_args = ksvd_genargs()
    return new_args


```

```
28  def Restauracion(arg):
29      if arg['Trecons']=='I':
30          'Reconstruccion Iterativa'
31          if arg['Solp']=='Con':
32              'Con Solapamiento'
33              i=1
34          else:
35              i=0
36          block=(arg['Tbloque'],arg['Tbloque'])
37          ' Ordenamiento Lexicografico de la imagen ocluida: '
38          ima_original_ocluida=im2col(arg['Iocluida'],block,i)
39          'Ordenamiento lexicografico de la mascara: '
40          matriz=im2col(arg['Mascara'],block,i)
41          iteracion=0
42          while matriz.min()==0:
43
44              'Reduccion de los arreglos: '
45
46              if iteracion >0:
47                  ima_original_ocluida=im2col(im,block,i)
48                  matriz=im2col(Mascara,block,i)
49
50              signal, matriz_red, index=acondicionamiento(matriz, ima_original_ocluida,arg['U'])
51
52              'Representacion Sparse de los bloques ocluidos: '
53
54              dc = np.mean(signal,0)
55              coef=omph(D,signal-dc,matriz_red,arg['Komp'])
56
57              'Reconstruccion de la imagen en la iteracion actual: '
58              im, Mascara = recons(D,coef,dc,block,(512,512),matriz,ima_original_ocluida , index, i)
59
60              'Generacion de imagen resultante en la iteracion actual: '
61              IMescalada=misc.bytescale(im, cmin=Min, cmax=Max)
62              Mescalada=misc.bytescale(Mascara, cmin=Min, cmax=1)
63              plt.imsave('ImagenReconstuida_komp'+str(arg['Komp'])+'Umbral'+str(arg['U'])\
64              +'Iteracion'+str(iteracion)+'.png', \
65              IMescalada, cmap=plt.cm.gray)
66              misc.imsave('Mascara'+'Umbral'+str(arg['U'])\
67              +'Iteracion'+str(iteracion)+'.png', \
68              Mescalada)#, cmap=plt.cm.gray)
69              iteracion+=1
70          else:
71              'Reconstruccion No Iterativa'
72              if arg['Solp']=='Con':
73                  'Con Solapamiento'
74                  i=1
75              else:
76                  i=0
77              block=(arg['Tbloque'],arg['Tbloque'])
78              ' Ordenamiento Lexicografico de la imagen ocluida: '
79              ima_original_ocluida=im2col(arg['Iocluida'],block,i)
80              'Ordenamiento lexicografico de la mascara: '
81              matriz=im2col(arg['Mascara'],block,i)
82              'Reduccion de los arreglos: '
83              im, Mascara = recons(D,coef,dc,block,(512,512),matriz,ima_original_ocluida , index, i)
84              'Representacion Sparse de los bloques ocluidos: '
85              dc = np.mean(signal,0)
86              coef=omph(D,signal-dc,matriz_red,arg['Komp'])
87              im, Mascara = recons(D,coef,dc,block,(512,512),matriz,ima_original_ocluida , index, i)
88              'Generacion de imagen resultante : '
89              IMescalada=misc.bytescale(im, cmin=Min, cmax=Max)
90              Mescalada=misc.bytescale(Mascara, cmin=Min, cmax=1)
91              plt.imsave('ImagenReconstuida_komp'+str(arg['Komp'])+'Umbral'+str(arg['U'])\
92              +'Iteracion'+str(iteracion)+'.png', \
93              IMescalada, cmap=plt.cm.gray)
94              misc.imsave('Mascara'+'Umbral'+str(arg['U'])\
95              +'Iteracion'+str(iteracion)+'.png', \
96              Mescalada)#, cmap=plt.cm.gray)
97
```

```
98  def train(new_args):
99      """
100     arguments:
101     obligatory
102         im{}
103         dicth
104         block
105
106     optional
107
108     """
109 #     checkargs(new_args)
110     b = np.array(new_args['block'])
111     # Convertimos una lista de imagenes en un arreglo de bloques para utilizarlos
112     #durante el entrenamiento
113     vim = np.array([]).reshape(np.prod(b), 0)
114     for i in range(len(new_args['im'])):
115         # Chequeando el tipo de las imagenes de entrada
116         if np.size(new_args['im'][str(i)].shape) == 3:
117             # Convierte las imagenes de RGB a YCbCr y selecciona
118             # la primera banda
119             new_args['im'][str(i)] = rgb2luv(new_args['im'][str(i)])[:, :, 0]
120
121         aux = im2col(new_args['im'][str(i)], b)
122         # Generando los bloques alta resolucion
123         vim = np.concatenate((vim, aux), 1)# Arreglo con el contenido de todos
124         #los bloques de las imagenes utilizadas en el entrenamiento
125         #vim=> vector de imagenes
126     if vim.size > 50e3 * 64:
127         npatch = round(50e3 * 64 / vim.shape[0]) + 1
128         npatch = np.random.permutation(vim.shape[1])[:npatch]
129         #Para realizar una seleccion aleatoria de subbloques de las imagenes de
130         #entrenamiento
131         vim = vim[:, npatch]
132
133     dc = np.mean(vim, 0)
134     new_args['signal'] = vim
135     KSVD(new_args)
136
137     del(new_args['im'])
138     return
```


Algoritmo *Orthogonal Matching Pursuit* con diccionario Holográfico

Modificación de la implementación computacional del algoritmo para representación poco densa OMP, resolviendo el problema restringido por poca densidad, para tomar en cuenta el diccionario holográfico.

- **Paquetes requeridos**
  - `Python 2.7`
  - `numpy`
- **Parámetros de entrada**
  - Señal objetivo (`x`)
  - Diccionario (`dic`)
  - Mascara (`M`)

  - Número de átomos del diccionario utilizado en la representación (`Komp`)

- Salida
  - Vector de representación poco densa con los coeficientes(`coef`)

```
import numpy as np


def omph(D,x,M,Komp):
    """
    omp(D,x,Komp) it's the simplest implementation of the OMP algorithm
    D: dictionary
    x: target signal
    Komp: number of non-zero coeficents
    M: Es la matrix de deformacion ordenada lexicograficamente

    """

#   Se crean los espacios para alamacenar las posiciones y los
#   valores de los coeficientes, I y coef respectivamente
    I = np.zeros(Komp,'uint64')
    coef = np.zeros((D.shape[1],x.shape[1]))
#   Bucle que recorre cada columna de la se#al objetivo y la matrix de degradacion
    for i in range(x.shape[1]):
        res = np.copy(x[:,i])
        m= np.diag(np.copy(M[:,i]))
        H=m.dot(D)
        Coef_normalizar=np.linalg.norm(H, axis=0) # Este arreglo almacena cada
        # norma del vector columna, para normalizar el diccionario
        #Estos los necesitare para entender el por que al no normalizar un diccionario
        # holografico, produce mejores resultados que normalizandolo.
        '''Para evitar un error de convergencia hay que garantizar que
        Coef_normalizar no tenga ningun valor igual a cero, si es asÃ, igualar a
        1'''
        # Con esto se evita que la SVD deje de converger
        i_zeros=Coef_normalizar==0
        Coef_normalizar[i_zeros]=1
        # Ahora continuamos con el codigo que tenÃamos
        H=H/Coef_normalizar
#   Bucle que encuentra la representacion sparse de cada se#al
        for j in range(Komp):
            k = np.argmax(np.abs(H.T.dot(res)))
            I[j] = k
            alfa = np.linalg.pinv(H[:,I[0:j+1]]).dot(x[:,i])
            res = x[:,i] - H[:,I[0:j+1]].dot(alfa)
        coef[I,i] = alfa

    return coef

```

Módulo utilities.py

Módulo con un conjunto de funciones para reconstruir la imagen estimada, así como reducir el tamaño de la data a procesar, entre otras.

```
from math import *
from numpy.random import RandomState
from scipy.signal import get_window
from skimage.transform import rescale
```

```
5  from scipy.ndimage.filters import convolve,convolve1d
6  from skimage import img_as_float
7  from dicttools import *
8
9
10
11 def reconsOLD(D,nzc,dc,b,I_size,ov = False):
12     """
13     Generate an image from the sparse coefficients
14     D: Dictionary
15     nzc: Sparse coefficients
16     dc: DC level asociate with the image
17     b: size of blocks
18     I_size: size of the image
19     ov: Must be True if the block overlap exists. It's False by default
20     """
21     if not ov:
22         v = D.dot(nzc) + dc
23         im = col2im(v,b,I_size)
24     else:
25         v = D.dot(nzc) + dc
26         im = ovr(v,1)
27     return im
28
29 def recons(D,nzc,dc,b,I_size,mascara_resultante_original, ima_original_ocluida, index,ov = False):
30     """Funcion para reconstruir una imagen ocluida, en la que solo se le estimo
31     los bloques en los que existia una oclusion de por lo menos 1 pixel. Final-
32     mente, se sustituiran solo los pixeles ocluidos. Esta puede utilizarse para
33     cualquier tipo de solapamiento.
34     Parametros:
35         D---> Diccionario utilizado
36         nzc---> Coeficientes de la representacion sparse encontrada
37         dc---> Nivel dc de la se#al original
38         b---> tupla con el tama#o del bloque
39         I_size---> tama#o de la imagen original.
40         ima_original_ocluida---> arreglo con los bloques de la imagen ocluida
41         mascara_resultante_original---> Arreglo que contiene la mascara resultante,
42         de tama#o iguala la orginal. Esta mascara se ira actualizando en la misma
43         proporcion que la imagen_orginal_ocluida.
44         index---> arreglo con los indices con las posiciones en donde existe oclusion
45         ov---> Tipo de solapamiento utilizado.
46     """
47     #Primero estimamos los bloques en donde existe oclusion, utilizando el diccionario
48     # y el nivel dc extraido de la imagen
49     v = D.dot(nzc) + dc
50     #Arreglo con los indices de la oclusion. Este paso es para obtner el numero de sub-bloques en que fue divi
51     index_shape=index.shape[0]
52     #contadores para relacionar los bloques estimados y los bloques de la imagen
53 #   original que no estaban ocluidos.
54     i_v=0
55     #Etapa de reordenamiento para obtener el arreglo con la imagen reconstruida
56     for j in range (index_shape):
57             #Bloques con oclusion
58
59             if index[j]==True:
60                 i_pix_oclui=mascara_resultante_original[:,j]==0
61                 ima_original_ocluida[:,j][i_pix_oclui]=v[:, i_v][i_pix_oclui]
62                 mascara_resultante_original[:,j][i_pix_oclui]=1
63                 i_v+=1
64             if i_v==nzc.shape[1]:
65                 break
66     if not ov:
67       im = col2im(ima_original_ocluida,b,I_size)
68       matriz=col2im(mascara_resultante_original,b,I_size)
69     else:
70        im = ovr(ima_original_ocluida,1,'mean')
71        matriz= ovr(mascara_resultante_original, 1, 'mean')
72     return im, matriz
73
74
```

```
75  def acondicionamiento(m,v, umbral):
76      """En el contexto de procesamiento de imagenes, esta funcion recibe como
77      argumento un arreglo 2-D que contiene el ordenamiento lexicografico de
78      la imagen, con solapamiento de bloques o sin solapamiento, y tiene como
79      salida una tupla T=(BloquesOcluidos,matrizOcluida,index)
80
81      v---> Arreglo con el ordenamiento lexicografico
82      umbral---> entero que define el maximo numero de pixeles perdidos permitidos
83      en un bloque.
84
85      BloquesOcluidos=Se refiere a las columnas de v, que contiene oclusiones, es
86      decir, por lo menos un pixel con valor 0
87
88      ImagenReducida= Se refiere a las colmunas de v, que no contienen oclusiones.
89      Estas columnas seran utilizadas en la reconstruccion final sin ninguna modi-
90      ficacion.
91
92      index= es arreglo boleano que contiene informacion sobre las columnas en las
93      que existe oclusion y cumple con el umbral definido.
94      """
95      #Primero creamos un indice que contiene los bloques con pixeles perdidos
96      index=m.all(axis=0)
97      #Creamos un indice con los bloques que cumplen con el umbral de pÃ©rdida
98      p=np.equal(m,0)
99      index1=np.sum(p, axis=0)<umbral
100     index3=np.logical_and(~index,index1)
101
102     matrizOcluida=m[:,index3]
103     BloquesOcluidos=v[:,index3]
104
105     #ImagenReducida=v[:,~index3]
106
107     return (BloquesOcluidos, matrizOcluida,index3)#ImagenReducida,index)
108
109
110
111
112 def ovr(matrix,ovl_step,method='mean',example=np.array([]), g=np.array([])):
113     """
114     im = ovr(arguments)
115     overlap reconstruction recover an image divided in blocks
116     using some techniques setting by the parmaeter method.
117     """
118     BlockSize = np.uint32(sqrt(matrix.shape[0]))
119     ImageSize = (np.uint32(sqrt(matrix.shape[1])) - 1) * ovl_step +  BlockSize
120     BlockSize = tuple([BlockSize,BlockSize])
121     ImageSize = tuple([ImageSize,ImageSize])
122     OriginalType = matrix.dtype
123     my_type = np.dtype([('value',OriginalType),('loc',np.int64),
124                         ('gamma',np.float64)])
125     second_type = np.dtype([('value',OriginalType),('gamma',np.float64)])
126     # Matriz de las ubicaciones de los pixel de la imagen
127     loc = np.arange(np.prod(ImageSize)).reshape(ImageSize,order='F')
128     #loc=loc.reshape(ImageSize,order='F')
129     matrix = matrix.astype(my_type)
130     # Reordenamiento en bloques
131     matrix['loc'] = im2col(loc, BlockSize, True, ovl_step)
132     # Gamma de cada bloque, segun su posicion
133     matrix['gamma'] = im2col(BlockGamma(BlockSize), BlockSize)
134     # Reordenamiento segun la ubicacion
135     matrix = np.sort(matrix.ravel(order='F'),order='loc')
136     # Maximo numero de pixeles usados para estimacion
137     Maxnpixels = (BlockSize[0] % ovl_step + BlockSize[0] // ovl_step)**2
138     # Matriz que contiene todos los pixeles usados para la estimacion en una misma fila
139     im = np.zeros((np.prod(ImageSize), Maxnpixels), second_type)
140     # El vector quantity contiene la cantidad de pixeles usados para la estimacion
141     im, quantity = ordenar(im,matrix)
142     im = im.reshape(ImageSize + tuple([-1]),order='F')
143     quantity = quantity.reshape(ImageSize,order='F')
144     if im['value'].nonzero()[0].size != matrix['value'].nonzero()[0].size:
145         print im.nonzero()[0].size
146         raise 'missed pixels'
147     if method == 'mean':
148         IM = np.sum(im['value'],axis=2) / quantity
```

```
149             return IM
150         elif method == 'median':
151             im = np.sort(im['value'],axis=2)
152             PixelEven = np.where(quantity % 2 == 0)#cantida par
153             PixelOdd = np.where(quantity % 2 == 1)#cantidad impar
154             PosOdd = quantity[PixelOdd[0],PixelOdd[1]]//2 + 1#posicion central impar
155             PosEven1 = quantity[PixelEven[0],PixelEven[1]]//2#posicion central par 1
156             #PosEven2 = quantity[PixelEven[0],PixelEven[1]]//2 + 1#posicion central par 2
157             IM = np.zeros(ImageSize,OriginalType)
158             IM[PixelOdd[0],PixelOdd[1]] = im[PixelOdd[0],PixelOdd[1], -PosOdd.astype('uint')]
159             IM[PixelEven[0],PixelEven[1]] = (im[PixelEven[0],PixelEven[1],-PosEven1.astype('uint')]
160             + im[PixelEven[0],PixelEven[1],-PosEven1.astype('uint')])/2
161             return IM
162         elif method == 'wmean':
163             #Ajuste de las dimensiones del ejemplo de acuerdo al zeros padding
164             if np.size(example) != 0:
165                 if example.shape != im.shape[:2]:
166                     aux = np.zeros(im.shape[:2],example.dtype)
167                     aux[:example.shape[0],:example.shape[1]] = example
168                     example = aux
169                 if np.size(g) == 0:
170                     w = afine(im['value'], example, np.ones(im.shape)) * (im['gamma'] != 0)
171                 else:
172                     w = afine(im['value'], example, g*im['gamma']) * (im['gamma'] != 0)
173             else:
174                 w = im['gamma']
175             return wmean(im['value'], w)
176         elif method == 'wmedian':
177             if example.shape != im.shape[:2]:#Ajuste de las dimensiones del ejemplo de acuerdo al zeros padding
178                 aux = np.zeros(im.shape[:2],example.dtype)
179                 aux[:example.shape[0],:example.shape[1]] = example
180                 example = aux
181             w = im['gamma'] * (im['gamma'] != 0)
182             return wmedian(im['value'],w)
183         else:
184             return im
185
186
187     def ordenar(im,matrix):
188         #Reordenamiento del vector de posiciones en la matriz im
189         # Donde ocurre un cambio
190         index = np.where(convolve1d(matrix['loc'],[1,-1]) !=0)
191         # Numero de piexeles usados para estimar
192         quantity = np.concatenate(([0],convolve1d(index,[1,-1])[0]))
193         # Indices en la tercera dimension del cubo de datos
194         Zindex = (im['value'] + np.arange(im.shape[1])).astype(np.uint64)
195         Zindex[0,1:] = im.shape[1]
196         for i in range(quantity.size):
197             if quantity[i]>0:
198                 Zindex[i,quantity[i]:] = im.shape[1]
199             else:
200                 Zindex[i,quantity[i]+1:] = im.shape[1]
201         Zindex = Zindex[np.where(Zindex<im.shape[1])[0],np.where(Zindex<im.shape[1])[1]]
202         im['value'][matrix['loc'],Zindex]= matrix['value']
203         im['gamma'][matrix['loc'],Zindex]= matrix['gamma']
204         quantity[np.where(quantity == 0)[0]] = 1
205         """Para corregir la rutina, agregarÃ© otra variable quantity1"""
206         quantity1=np.zeros_like(quantity)
207         for i in np.arange(np.shape(quantity)[0]):
208             quantity1[i]=np.count_nonzero(im['value'][i])
209             index=quantity1==0
210             quantity1[index]=1
211
212
213         return im,quantity1
214
215
216     def BlockGamma(BlockSize):
217         return pascal(BlockSize[0] - 1).T.dot(pascal(BlockSize[0] - 1))
218
219
220     def pascal(n):
221         line = [1]
222         for k in range(n):
223             line.append(line[k] * (n-k) / (k+1))
```

```
224     return np.array(line).reshape(1,-1) / 2.0**n
225
226
227 def wmean(signal,w):
228     return np.sum(signal * w,axis = 2) / np.sum(w,axis = 2)
229
230
231 def wmedian(signal, w):
232     sw = np.sign(w)
233     tau = 0.5 * w.sum(axis = 2)
234     my_type = np.dtype([('value', float), ('weight', float)])
235     target = np.zeros_like(signal ,dtype=my_type)
236     aux = signal * sw * -1
237     target['value'] = aux
238     target['weight'] = w
239     target = np.sort(target,order ='value',axis = 2)
240
241     w = target['weight'].cumsum(axis = 2)
242     pos = np.where(w - tau.reshape(tau.shape + tuple([1])) < 0)
243     w[pos[0],pos[1],pos[2]] = w.max()
244
245     pos_y = np.ones((signal.shape[0],1)).dot(np.array([np.arange(signal.shape[1])])).ravel().astype('uint64')
246     pos_x = np.array([np.arange(signal.shape[0])]).T.dot(np.ones((1,signal.shape[1]))).ravel().astype('uint64'
247     pos_z =  w.argmin(axis = 2).ravel()
248     return -1*target['value'][pos_x,pos_y,pos_z].reshape(signal.shape[:2])
249
250
251 def afine(x,u,y):
252     if len(x.shape) == 3:
253         y[np.where(abs(y)<1e-4)] = 1e-4
254         aux = np.zeros((u.shape) + tuple([1]),u.dtype)
255         aux[:,:,0] = u
256         u = aux
257     error = (x - u)**2
258     error = np.asarray_chkfinite(error / y)
259     return np.exp( -1.0 * error)
260
261
262 def MSE(I1,I2):
263     return np.linalg.norm(I1.ravel()-I2.ravel())**2.0/I1.size
264
265
266 def PSNR(I1,I2):
267         return 10 * log10((I1.max())**2 / MSE(I1,I2))
268
269
270 def MAE(I1,I2):
271     return np.linalg.norm(I1.ravel()-I2.ravel(),ord = 1)/I1.size
272
273
274 def imnoise(Im, d, cont='gaussian', seed=None):
275     s = RandomState(seed)
276     if cont == 'gaussian':
277         I = Im + sqrt(d) * s.randn(Im.shape[0],Im.shape[1])
278         I[np.where(I < 0)] = 0
279         I[np.where(I > 1)] = 1.0
280     elif cont == 'salt & pepper':
281         x = s.rand(Im.shape[0],Im.shape[1])
282         I = Im.copy()
283         I[np.where(x < d/2)] = 0
284         I[np.where((x >= d/2) * (x < d))] = 1
285     return I
286
287
288 def snr(Im,noise):
289     return 10 * log10(np.linalg.norm(Im)**2 / np.linalg.norm(noise)**2)
```