Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestría en Ciencias en Inteligencia Artificial

Implementación en Hardware de un Algoritmo de Dehazing

Tesis

Que como parte de los requisitos para obtener el Título de
Maestro en Ciencias en Inteligencia Artificial

Presenta:
Miguel Ángel Moncada Malagón
Dirigido por:
Dr. Juan Manuel Ramos Arreguín

Dr. Juan Manuel Ramos Arreguín
Presidente

Dr. Jesús Carlos Pedraza Ortega
Secretario

Dr. Marco Antonio Aceves Fernández
Vocal

Dr. José Emilio Vargas Soto
Suplente

Dr. Efrén Gorrostieta Hurtado
Suplente

Centro Universitario Querétaro, QRO
México.
Enero 2020

A mi familia ya que sin ellos no sería lo que soy, y a cada una de las personas que se cruzaron en mi camino durante la realización de este trabajo ya que para bien o para mal y de alguna manera influyeron en mí.

Agradecimientos

A Dios por cuidarme y protegerme y haberme dado todo lo que necesitó para llegar hasta el día de hoy.

A mis padres por haberme dado todo su apoyo y amor incondicional y porque sin ellos no sería lo que soy.

A mi novia Brenda por haberme apoyado y motivado a esforzarme cada día para cumplir una meta más, por estar a mi lado y ayudarme en todo momento a encontrar una solución a cada problema que se me presento.

A mis profesores de la maestría en especial a mi director de tesis el Dr. Juan Manuel Ramos Arreguín por haberme guiado de la mejor manera en la realización de esta tesis, y por sus valiosos aportes que sin duda alguna fueron parte vital para culminar este trabajo, al Dr. Jesús Carlos Pedraza Ortega por sus valiosas enseñanzas dentro y fuera de las materias, a mi tutor el Dr. Marco Antonio Aceves Fernández por su apoyo y consejos brindados durante la maestría, al Dr. José Emilio Vargas Soto por su experiencia compartida y motivación a ser personas emprendedoras, al Dr. Saúl Tovar Arriaga por su labor de coordinador de la maestría y haberme apoyado en las cuestiones académicas.

A mis compañeros y amigos de la maestría por haberme acompañado durante esta etapa, en especial a mi amigo Mario que siempre fue una buena influencia y por su motivación a siempre buscar más herramientas y conocimiento en el área de la Inteligencia Artificial.

Al Conacyt por la beca otorgada ya que si ella no hubiera podido dirigir todo mi esfuerzo y dedicación para la culminación de esta maestría.

A la UAQ por proveerme la infraestructura y el material necesario para desempeñar mis actividades de investigación durante la maestría, y por la beca semestral otorgada.

Resumen

En este trabajo de investigación se presenta una arquitectura de hardware para realizar *dehazing* en imágenes únicas.

En ocasiones, las imágenes capturadas por un sensor, una cámara convencional, muestran defectos en las imágenes ocasionados por cuestiones atmosféricas como pueden ser neblina, contaminación, smog, lluvia entre otros. Por medio de los algoritmos de *dehazing* se pueden eliminar estos defectos, en particular en este proyecto se utilizó un algoritmo de *dehazing* basado en el *DCP*.

Los dispositivos programables *FPGA* son capaces de ejecutar operaciones en paralelo debido a su arquitectura interna y con esto se logra un mayor rendimiento en el procesamiento de imágenes.

La metodología se dividió en 3 etapas. En la primera etapa, se generaron imágenes con *haze* sintético a partir de una base de datos con imágenes relacionadas a vehículos autónomos. En la segunda etapa se creó una arquitectura para realizar procesamiento de imágenes en general. Y por último se crearon los bloques lógicos encargados de procesar y aplicar las funciones del algoritmo de *dehazing*.

El proyecto se encuentra orientado para tareas de visión por computadora relacionadas con vehículos autónomos y sistemas de conducción avanzada (*ADAS*).

(**Palabras clave:** Visión por Computadora, Procesamiento de imágenes, *FPGAs*, *Hardware* Reconfigurable, *Dehazing*, Canal Oscuro, *Dark Channel Prior, single-images*)

Abstract

In this research work, a hardware architecture is presented to perform dehazing on single-images.

Sometimes, the images captured by a sensor, a conventional camera, show defects in the images caused by atmospheric issues such as fog, pollution, smog, rain among others. By means of the dehazing algorithms these defects can be eliminated, in particular in this project a dehazing algorithm based on DCP was used.

FPGA programmable devices are able to execute operations in parallel due to their internal architecture and this achieves a higher performance in image processing.

The methodology was divided into 3 stages. In the first stage, images with synthetic haze were generated from a database with images related to autonomous vehicles. In the second stage, an architecture was created to perform image processing in general. And finally the logical blocks were created to process and apply the functions of the dehazing algorithm.

The project is oriented to computer vision tasks related to autonomous vehicles and advanced driving systems (ADAS).

(**Keywords**: Computer Vision, Image Processing, FPGAs, Reconfigurable Hardware, Dehazing, Dark Channel, Dark Channel Prior, *single-images*)

Índice general

Contenido

Índice de Figuras

Índice de Tablas

Lista de Acrónimos

ADAS: (*Advanced Driver Assistance Systems*, Sistemas Avanzados de Asistencia al Conductor)

AHE: (*Adaptive Histogram Equalization*, Ecualización de Histrograma Adaptivo)

CLAHE: (*Contrast Limited Adaptive Histogram Equalization*, Ecualización de Histograma Adaptativo Limitada por Contraste)

CPU: (*Central Processing Unit*, Unidad Central de Procesamiento)

DB: (*Data Base*, Base de Datos)

DCP: (*Dark Channel Prior*, Principio del Canal Oscuro)

DDP: (*Dense Depth Posterior,* Profundidad Densa Posterior)

FPGA: (*Field-Programmable Gate Array*, Matriz de Compuertas Lógicas Programable en Campo)

FSM: (*Finite State Machine,* Máquinas de Estados Finitos)

HDL: (*Hardware Description Language*, Lenguaje de Descripción de Hardware)

IA: Inteligencia Artificial

IEEE: (*Institute of Electrical and Electronics Engineers*, Instituto de Ingenieros Eléctricos y Electrónicos)

KIT: (*Karlsruhe* Institute of Technology, Instituto de Tecnología *Karlsruhe*)

LIDAR: (*Laser Imaging Detection and Ranging, Detección y Alcance de Imágenes Láser*)

LUT: (*lookup table,* tabla de consulta)

MSE*:* (*Mean Squared Error,* Error Cuadrático Medio)

NIR: (Near-Infrared Reflectance, Reflectancia infrarroja cercana)

PC: (*Personal Computer*, Computadora Personal)

PDI: Procesamiento Digital de Imágenes

PLL: (Phase-Locked Loop, Bucle de Desplazamiento de Fase)

PSNR: (*Peak Signal to Noise Ratio*, Proporción Máxima de Señal a Ruido)

RAM: (*Random Access Memory*, Memoria de acceso aleatorio)

RGB: (*Red, Green, Blue*, Rojo, Verde, Azul)

RTC: (*Real-Time Computing*, Procesamiento en Tiempo Real)

SSIM: (*Structural Similiarity Index*, Índice de Similitud Estructural)

VGA: (*Video Graphics Array*, Matriz de gráficos de vídeo)

VHDL: (*Very High Speed Integrated Circuit Hardware Description Language*, Lenguaje de Descripción de Hardware para Circuitos Integrados de Muy Alta Velocidad)

VHSIC: (*Very High Speed Integrated Circuit*, Circuitos Integrados de muy Alta Velocidad)

CAPÍTULO 1

Introducción

1.1. Descripción del problema

En ocasiones, la acumulación de grandes cantidades de partículas en la atmósfera puede producir efectos negativos en la captura digital de una imagen. Normalmente esto suele ser ocasionado por condiciones climáticas adversas como puede ser la niebla, la neblina o por la contaminación en el medio. Estos defectos en las imágenes ocurren debido a que los cúmulos de estas partículas suspendidas en la atmosfera dispersan la luz que los objetos reflejan, lo que provoca una alteración en la señal capturada por la cámara. Como resultado de esto, las imágenes suelen presentar:

- Baja visibilidad
- Bajo contraste
- Distorsión de los colores
- Pérdida de la fidelidad de los colores

Esto deriva en una reducción de la calidad en la imagen. Para efectos prácticos, en la literatura, a los fenómenos físicos que producen este tipo de defectos en las imágenes se les ha englobado con el termino *haze.*

Podemos encontrar diversos escenarios en los cuales se puede presentar *haze* (Khoury, 2016). En la Figura 1.1 se muestran algunos de los más comunes.

Figura 1.1: Ejemplos de escenarios con *haze*. (a) escenarios subacuáticos; (b) carreteras cubiertas por niebla; (c) contaminación del aire en las ciudades; (d) imágenes satelitales; (e) niebla nocturna; (f) interior de una fábrica con presencia de gases (imagen propia con información de (Khoury, 2016)).

1.2. Justificación

La presencia de *haze* en las imágenes dificulta las aplicaciones de los sistemas de visión por computadora, en donde uno de los objetivos es la extracción de información o características de las imágenes (Snyder y Qi, 2017), afectando la precisión en sus técnicas como la detección y el seguimiento de objetos, así como también dificulta las tareas relacionadas con sistemas de vigilancia, navegación autónoma y en *ADAS*. Es por esto que surge la necesidad de la existencia de métodos para resolver este problema, los cuales han sido nombrados en la literatura como métodos de *dehazing*.

En el área de visión por computadora se utiliza ampliamente el modelo de dispersión atmosférica propuesto por *Narasimhan* en 2001 (Schechner, Narasimhan y Nayar, 2001). El modelo describe la formación de una imagen captada por una cámara

y requiere básicamente tres parámetros: la imagen de entrada, la luz atmosférica global y el mapa de transmisión (En la Figura 1.2 se muestra gráficamente este modelo). En la mayoría de los casos se desconocen los dos últimos parámetros, teniendo únicamente la imagen capturada por el sensor (los cuales se pueden encontrar comúnmente en la literatura como *single-image*). Es por ello que eliminar el *haze* o al menos atenuarlo teniendo una imagen única es un problema desafiante.

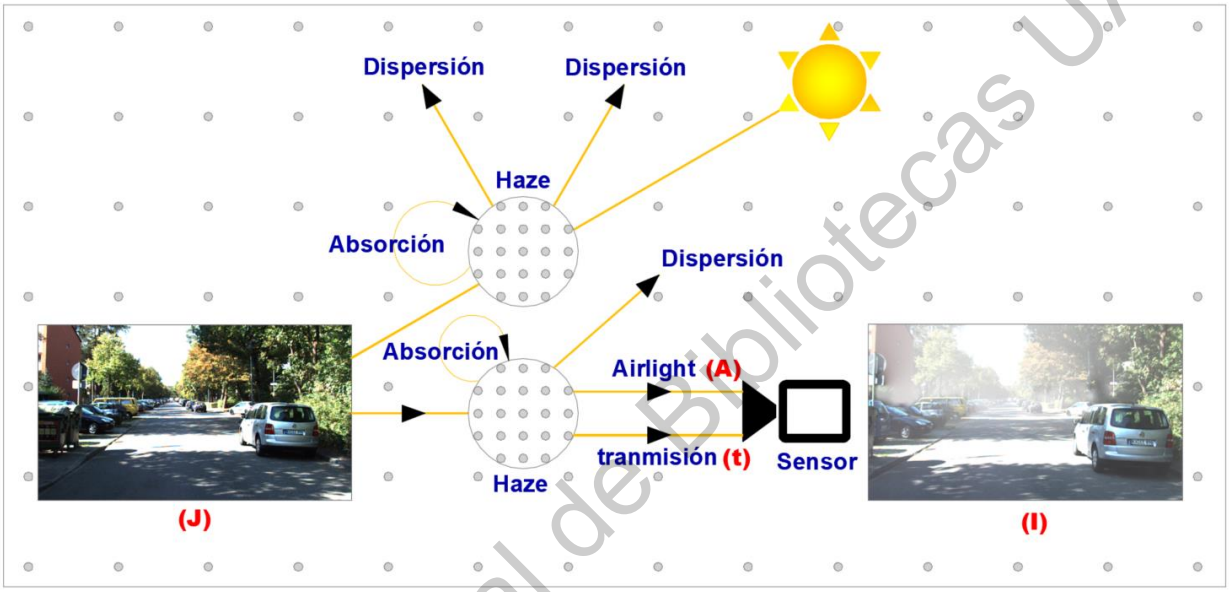


Figura 1.2: Modelo de dispersión atmosférica. luminosidad real de la escena (J), luz atmosférica (A), mapa de transmisión (t), intensidad de la imagen capturada por el sensor (I) (imagen propia).

Los algoritmos de *dehazing* actuales comparados con los primeros, se han ido mejorando en cuanto a la necesidad de disminuir su tiempo de cómputo, aumentar la calidad de restauración y versatilidad ante distintos tipos de imágenes (Salazar, 2019), sin embargo, su implementación en sistemas embebidos para aplicaciones de visión por computadora, requieren mejorar aún más estas capacidades. Es por ello que, dadas las características que posee un *FPGA* (principalmente el procesamiento en paralelo), en este trabajo se llevó a cabo la implementación de un algoritmo, generalizado, para realizar *dehazing* en imágenes únicas.

1.3. Hipótesis

El diseño de una arquitectura implementada en un *FPGA* para remover *haze* en imágenes únicas utilizando un algoritmo de *dehazing* basado en el principio del canal oscuro, procesa las imágenes a una mayor velocidad que implementando el algoritmo en otros sistemas embebidos, y puede contribuir en trabajos futuros para el procesamiento en sistemas de tiempo real o *real-time computing* (*RTC*, por sus siglas en inglés) para realizar tareas de visión por computadora.

1.4. Objetivos

1.4.1. Objetivo general

Diseñar una arquitectura en hardware para procesar imágenes únicas con *haze* mediante un algoritmo de *dehazing* basado en el principio del canal oscuro.

1.4.2. Objetivos específicos

- Diseñar e implementar una arquitectura para el procesamiento de imágenes *RGB* en un *FPGA*.
- Establecer una comunicación entre la PC y el *FPGA* para la transmisión de las imágenes *PC-FPGA* y viceversa.
- Diseñar un controlador para el almacenamiento de las imágenes en el *FPGA*.
- Diseñar un controlador en el *FPGA* para la visualización de las imágenes en un monitor *VGA*.
- Obtener el canal oscuro de imágenes con *haze* en el *FPGA*.

- Implementar un algoritmo de *dehazing* (basado en el principio del canal oscuro) en el *FPGA* para procesar imágenes relacionadas con sistemas de conducción autónoma.

1.5. Estado del arte

Un área de Inteligencia Artificial (IA) que ha tenido gran relevancia en los últimos años, es la visión por computadora, la cual es una disciplina que estudia cómo procesar, analizar e interpretar imágenes de forma automática (Forsyth y Ponce, 2011). Gracias a los grandes avances en el área de la electrónica, a su crecimiento exponencial que ha tenido en los últimos años y con ello el aumento de poder de procesamiento de los *CPU*, a finales de la década pasada se hizo factible el estudio y desarrollo de métodos para afrontar el problema del *haze* digitalmente (Salazar, 2019). A continuación, se describen algunas de las investigaciones más sobresalientes que presentan algoritmos para resolver el problema del *haze* con una sola imagen.

En 2008 *Tan* observa que una imagen libre de *haze* tiene un contraste más alto en diferencia a una imagen con *haze* y lo remueve mediante la maximización del contraste local en la imagen. Visualmente los resultados son agradables, pero por lo común el algoritmo produce imágenes sobresaturadas y que no parecen naturales por lo que suelen no ser validos físicamente (Tan, 2008).

Años más tarde en la literatura aparecieron otros métodos para la remoción de *haze* en los cuales se utilizaron filtros de media, mediana, bilaterales, trilaterales, filtrado gaussiano, filtrado guiado, así como también ecualización de histogramas o algunas de sus variantes como ecualización de histograma adaptativo o *Adaptive Histogram Equalization* (*AHE*, por sus siglas en inglés) y ecualización de histograma adaptativo limitada por contraste o *Contrast Limited Adaptive Histogram Equalization* (*CLAHE*, por sus siglas en inglés).

En 2009 en *Tarel* y Hautiere (Tarel y Hautiere, 2009) se agrega un filtro de mediana con el objetivo de eliminar los halos de luz en las imágenes. El algoritmo propuesto tiene bajo costo computacional debido a que su complejidad es una función lineal del número de pixeles de la imagen.

En 2010 *Kaiming He* introduce el principio del canal oscuro o *Dark Channel Prior* (*DCP*, por sus siglas en inglés), el cual está basado en un hecho estadístico que muestra que la mayoría de las ventanas locales en imágenes sin *haze* contienen pixeles con baja intensidad. Se obtuvieron resultados satisfactorios pero su principal desventaja es que el procesamiento es lento y por ende no puede ser usado en *RTC*, además sus resultados sufren de halos alrededor de los bordes (He, Sun y Tang, 2010). A pesar de lo anterior el *DCP* ha sido usado ampliamente y ha servido como base en otros algoritmos reportados en la literatura para realizar el *dehazing* donde se intenta disminuir la complejidad del algoritmo.

Dos años más tarde en 2012 *Kaiming He* realizó una variación a su algoritmo del *DCP* donde toma en cuenta el hecho de que los dispositivos de captura de la escena realizan ajustes automáticos, como el balance de blancos. Por lo cual propuso solucionar el problema de los halos de luz aplicando una corrección de balance de blancos y descomponiendo la imagen en dos componentes, la luz reflejada en los objetos y la luz ambiental (He, Wang, Xiong y Feng, 2012).

En 2013 en *Zhang* y cols. (Zhang, Liu, Yang y Wu, 2013) se propone un algoritmo para realizar *dehazing* a imágenes únicas el cual combina métodos que están basados en modelos físicos y no físicos. Primeramente, utiliza el *DCP* para estimar la luz atmosférica, en segundo lugar, utiliza el algoritmo *Retinex* (Land y McCann, 1971) basado en dos filtros bilaterales, el cual se aplica al brillo de la imagen de entrada con *haze* para mejorarlo, después se obtiene una aproximación de la transmisión a través de un filtro de mediana adaptativo para finalmente recuperar la escena libre de *haze* a través del modelo de dispersión atmosférica.

En 2015 *Saggu y Singh* (Saggu y Singh, 2015) presentaron una revisión sobre varios algoritmos y técnicas de eliminación de neblina, tales como: "*Dark Channel Prior*", "*CLAHE*", Filtrado bilateral, *"Mix CLAHE*" y filtrado trilateral, en el cual se demuestra que la técnica basada en canales oscuros previos "*Dark Channel prior*" ha proporcionado mejores resultados sobre otras técnicas disponibles.

En 2015 *Bai* y cols. (Bai, Wu, Xie y Wen, 2015) propuso un método para la eliminación de neblina basado en el método *DCP* que utiliza un filtro guiado para optimizar la transmisión del medio. El método utiliza un submuestreo y el método de interpolación para transformar una imagen de alta resolución en una de baja resolución para reducir la cantidad de cálculo. El algoritmo se implementó en un *DSP multi-core* de la compañía TI. Los resultados experimentales mostraron que el método procesa una imagen con una resolución de 600*400 en menos de 40 ms y puede satisfacer la demanda del proceso de imagen en tiempo real.

En 2016 *Zhang & Zao* (Zhang y Zhao, 2016) propusieron un algoritmo de *dehazing* implementado en una *FPGA Stratix* con una velocidad de procesamiento de 116 MHz para realizar *RTC*, basado en el canal oscuro, en el cual, se empleó un método aproximado para estimar los valores de la luz atmosférica y el mapa de transmisión. La arquitectura en hardware está orientada hacia aplicaciones de sistemas embebidos y los resultados en simulación indicaron que el hardware es altamente eficiente, obteniendo buenos resultados en la restauración de las imágenes y satisfaciendo los requisitos en tiempo real inclusive para imágenes de gran tamaño.

En 2018 *Kim y Jeon.* (Kim y Jeon, 2018) demostró que la implementación de algoritmos de *DCP* en *FPGA* arroja resultados con una velocidad de procesamiento mayor a otros sistemas embebidos. La implementación de estas técnicas de eliminación de neblina en *FPGA* facilita el desarrollo de algoritmos avanzados de visión por computadora reduciendo tiempos de procesamiento al hacer uso de la programación en paralelo.

En 2018 en *Salazar-Colores et al.* (Salazar-Colores, Ramos-Arreguin, Echeverri,

Cabal-Yepez, Pedraza-Ortega, y Rodriguez-Resendiz, 2018) se propuso un nuevo algoritmo de *dehazing* basado en el *DCP* el cual además se combina con morfología matemática y un filtro gaussiano, los cuales son algoritmos de baja complejidad computacional. El rendimiento del algoritmo se comparó con la literatura previa y los resultados reportaron un menor tiempo de procesamiento con lo cual se abre la posibilidad de ser aprovechado en sistemas de tiempo real.

Los trabajos anteriormente descritos y su aportación se resumen en la Tabla 1.1

Tabla 1.1: Resumen del estado del arte (creación propia).

Año	Autor	Aportación
2008	*Robby Tan* (Tan, 2008)	Maximización del contraste de las imágenes para remover el *haze.* Desventaja: Los resultados son visualmente agradables pero físicamente pueden no ser válidos y generalmente se producen imágenes sobresaturadas.
2009	*Tarel et al.* (Tare y Hautiere, 2009)	Agregó un filtro de mediana para eliminar los halos de luz en las imágenes. Ventaja: El algoritmo tiene un bajo costo computacional.
2010	*Kaiming He* (He, Sun y Tang, 2010)	Introdujo el principio del canal oscuro (*DCP*, por sus siglas en inglés). Desventaja: El procesamiento es lento y no puede ser usado en *RTC*.
2012	*Kaiming He* (He, Wang, Xiong y Feng, 2012)	Aplicó una corrección de balance de blancos y descompuso la imagen en dos componentes, la luz reflejada en los objetos y la luz ambiental para solucionar el problema de los halos de luz.
2013	*Zhang et al.* (Zhang, Liu, Yang y Wu, 2013)	Propuso un algoritmo para realizar *dehazing* a imágenes únicas el cual combina métodos que están basados en modelos físicos y no físicos.

2016	*Zhang & Zhao* (Zhang y Zhao, 2016)	Propuso un algoritmo de dehazing implementado en una *FPGA Stratix* con una velocidad de procesamiento de 116 *MHz* para realizar *RTC*, basado en el canal oscuro, en el cual, se empleó un método aproximado para estimar los valores de la luz atmosférica y el mapa de transmisión.
2018	*Salazar-Colores* (Salazar-Colores, Ramos-Arreguin, Echeverri, Cabal-Yepez, Pedraza-Ortega, y Rodriguez-Resendiz, 2018)	Propuso un algoritmo basado en el *DCP* el cual además se combina con morfología matemática y un filtro gaussiano. Con el cual se obtuvo un menor tiempo de procesamiento comparado con los trabajos de la literatura previa.

1.6. Alcances y limitaciones

El presente trabajo tiene como alcances el uso de una arquitectura en el *FPGA* para procesar imágenes con *haze* y eliminar este efecto causado por condiciones climatológicas adversas. Pretende contar con una arquitectura robusta para procesar este tipo de imágenes. Para esto, se generó una biblioteca de bloques lógicos los cuales

Las limitaciones del presente trabajo radicaron principalmente en los recursos de memoria RAM para el almacenamiento de las imágenes, y en la dificultad que tiene el manejo del tipo de memorias RAM tipo DDR2 en un FPGA.

Tomando como base el algoritmo propuesto por *K. He* se llevó a cabo la implementación en el *FPGA*, sin embargo, se eliminó, uno de los 5 pasos, quedando solo 4, el que se refiere al *soft matting* se omitió por cuestiones de simplificar el algoritmo, sin embargo, este paso puede ser implementado en trabajos futuros.

1.7. Organización de la tesis

El documento se encuentra organizado en 5 capítulos, los cuales se describen a continuación:

Capítulo 1: Se introduce acerca del tema de investigación, exponiendo una descripción del objeto de estudio, los elementos que fundamentan el problema planteado, el planteamiento de la hipótesis y los objetivos; general y específicos, así como los alcances y las limitaciones.

Capítulo 2: Se sientan las bases que conforman el modelo teórico de una imagen con *haze*, se exponen detalladamente los conceptos y definiciones referentes a la investigación tales como: visión por computadora, imágenes digitales, procesamiento digital de imágenes, imágenes en color *RGB*, la atmósfera, la composición del aire, la dispersión atmosférica, la luz atmosférica, el mapa de transmisión de una imagen, el principio del canal oscuro, *dehazing* en imágenes únicas, un algoritmo de *dehazing* basado en el *DCP* y por último sistemas embebidos: *FPGAs*.

Capítulo 3: Se describen de forma detallada cada uno de los pasos que se siguieron para el desarrollo del presente trabajo, así como también se muestran y se describen las técnicas aplicadas para el desarrollo de la arquitectura para el PDI y la descripción de los bloques lógicos involucrados en la implementación del algoritmo de *dehazing* en el *FPGA.*

Básicamente la metodología que se aplicó se divide en tres etapas. En la primera, se explica la metodología para la generación de las imágenes usando la *DB: Kitti*, en la segunda etapa, se describe la arquitectura general para PDI en el *FPGA*, mientras que en la tercera etapa se describe finalmente la arquitectura para realizar *dehazing* en el

FPGA.

Capítulo 4: En este capítulo se muestran los resultados obtenidos, primeramente, se muestran los resultados de cada una de las 3 etapas descritas en la metodología, para la etapa 1 se muestran los resultados de la generación de las imágenes producidas en el *software Matlab* versión *R2017b,* para la etapa 2 se muestran algunos filtros aplicados a las imágenes para mostrar el desempeño de la arquitectura general para PDI en el *FPGA*. Finalmente se plasman las conclusiones y se discuten el trabajo futuro que le compete al presente trabajo de investigación.

CAPÍTULO 2

Marco Teórico

2.1. Visión por Computadora

La Visión por Computadora es un área dentro del campo de la IA, que se encarga de estudiar los procesos mediante el cual una máquina, generalmente una computadora (o en ocasiones cuando la situación lo requiere un sistema embebido) sea capaz de reconocer el contenido de una imagen automáticamente.

Dentro de un sistema de visión por computadora (ver Figura 2.1) existen unos pasos ordenados a seguir para obtener el resultado final. En primer lugar, el sistema recibe una entrada, la cual es una imagen. En segundo lugar, se realiza el procesamiento de la imagen y con esto se extraen ciertas características (la selección de los procesos de procesamiento como pueden ser filtros, dependen de la tarea a realizar como podría ser detección de objetos, seguimiento de objetos entre otros). En tercer lugar, estas características, se proporcionan a un sistema de reconocimiento de patrones, el cual finalmente debe tomar una decisión.

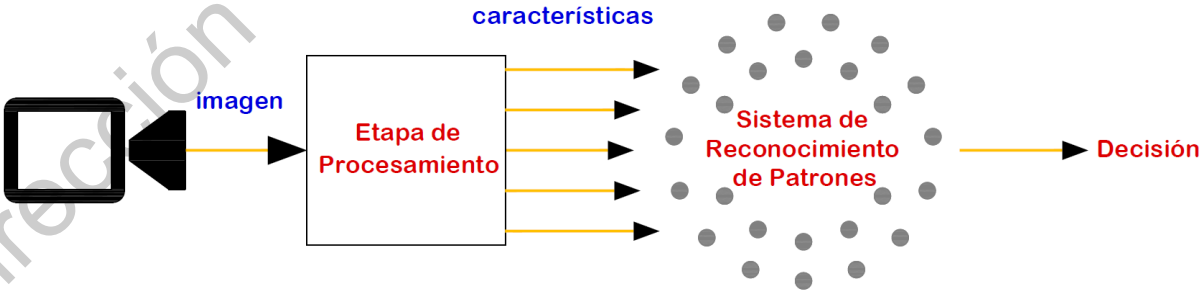


Figura 2.1: Proceso de un Sistema de Visión por Computadora (imagen propia con base en (Schechner, Y. Y., Narasimhan, S. G. y Nayar, 2001)).

2.2. Procesamiento de imágenes en color *RGB*

2.2.1. Imagen Digital

- **Píxel**: Es la mínima unidad que compone a una imagen digital.

Una imagen puede definirse como una función bidimensional, *f (x, y)*, donde *x* e *y* son las coordenadas espaciales, y la amplitud de *f* en cualquier par de coordenadas (*x, y*) es denominada la intensidad de la imagen en ese punto (González y Wintz,1996). Cuando el tamaño de la imagen es finito y los 3 valores (*x, y* e intensidad) son cantidades discretas se le denomina que es una imagen digital.

- **Imagen Digital**: Es un conjunto de elementos finitos denominados píxeles, los cuales poseen 3 valores, un par de coordenadas (*x, y*) y un valor de intensidad para cada par. Normalmente se representan por medio de una matriz.

2.2.2. Procesamiento Digital de Imágenes

El Procesamiento Digital de Imágenes (PDI), es el conjunto de métodos y técnicas que se aplican a las imágenes digitales con el objetivo de mejorar la calidad o facilitar la búsqueda de información (González y Wintz,1996). Estas técnicas se dividen en 2 categorías principales: imagen-imagen (es el conjunto de técnicas que a la salida entregan otra imagen) e imagen-atributos (es el conjunto de técnicas que a la salida entregan atributos de la imagen).

En la Figura 2.2 podemos observar estas categorías y sus correspondientes tipos de tratamiento a las imágenes. En el caso de las operaciones morfológicas pueden

pertenecer tanto a la categoría imagen–imagen como a la categoría imagen–atributos dependiendo de la técnica.

Figura 2.2: Tipos de procesos dentro del PDI (imagen propia con base en (González y Wintz,1996)).

Como se observa en la Figura 2.2 el primer paso es la adquisición de las imágenes, sin embargo, después de este paso no necesariamente se tienen que aplicar todos los procesos a la imagen, todo depende del propósito y los objetivos. El filtrado y la mejora de imágenes tiene como propósito realzar detalles o características en la imagen. La restauración de las imágenes también tiene como propósito mejorar la apariencia de la imagen, pero las técnicas se basan en modelos matemáticos de degradación de la imagen. El procesamiento de imágenes en color cubre técnicas especiales dedicadas al color de la imagen. El estudio de las ondas es la base para representar las imágenes en varios grados de resolución y se utiliza principalmente en comprensión. La compresión se encarga de reducir el almacenamiento requerido para guardar una imagen. Las operaciones morfológicas comprenden las técnicas para extraer

componentes de la imagen que son útiles en la representación y descripción de formas en la imagen. Las técnicas de la segmentación dividen la imagen en sus partes o elementos que la constituyen. En la representación y la descripción de las imágenes se identifican las regiones de una imagen y sus fronteras. Las fronteras son el conjunto de píxeles que separan una región de la imagen de otra. Por último, en el reconocimiento de objetos se asignan etiquetas a los objetos en función de sus características (González y Wintz,1996).

2.2.3. Imagen en Color *RGB*

En el modelo *RGB* (*Red, Green, Blue*) (ver Figura 2.3) el color de cada pixel es representado mediante 3 valores llamados bandas o canales, los cuales (como sus siglas lo describen) pertenecen a los componentes espectrales primarios, Rojo, Verde y Azul. Cada canal de color puede tomar un valor en un rango de 0 a 1.

En la Figura 2.3 a la derecha se puede apreciar una representación tridimensional de este modelo el cual es llamado: cubo unitario *RGB*. En un monitor estos valores se combinan para producir el color del pixel.

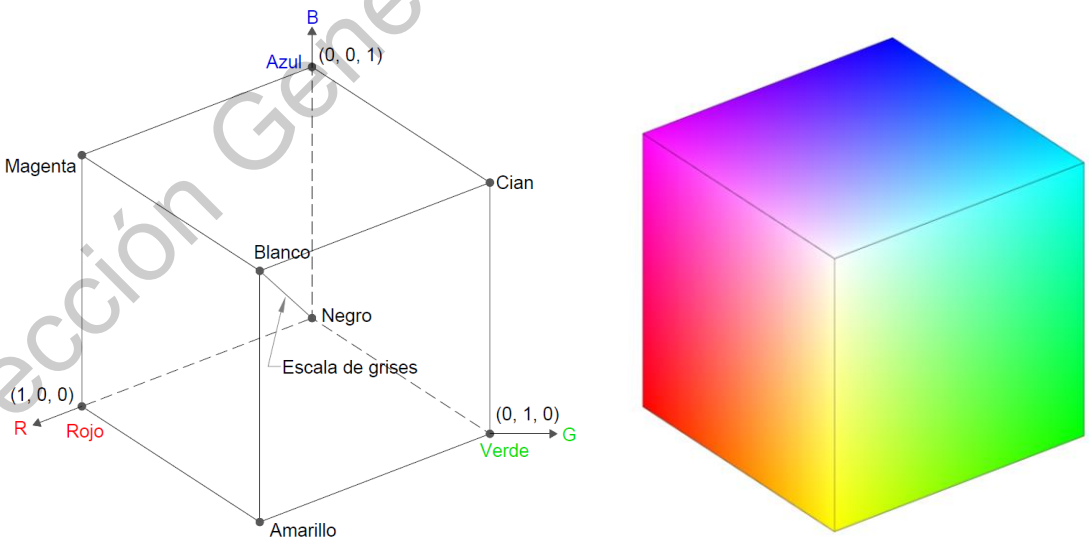


Figura 2.3: Modelo *RGB* de una imagen, cubo unitario *RGB* (imagen propia con base en (González y Wintz,1996)).

2.3. Atmósfera y su relevancia en el *haze*

La presencia de la atmósfera es fundamental para el desarrollo de la vida en el planeta, así mismo es una parte esencial en los modelos de *dehazing*, ya que los elementos que la conforman; aire, vapor de agua, polvo y partículas provocan la aparición de los fenómenos que afectan la visibilidad y por ende ocasionan la degradación en las escenas capturadas por una cámara.

- **Atmósfera**: Se denomina atmósfera a la esfera gaseosa que cubre la Tierra y se presenta como una mezcla de aire, H2O (en estado gaseoso), polvo y partículas radioactivas (La Atmósfera, (s.f.)).

2.3.1. Composición del aire en la atmósfera

- **Aire**: Mezcla de gases que forman la atmósfera de la tierra (Gases del Aire Composición y Propiedades, (s.f.)).

El aire puro y seco se compone de gases, principalmente de nitrógeno y oxígeno, una pequeña cantidad de argón, dióxido de carbono y otros gases (En la Figura 2.4 se pueden observar las cantidades de los componentes del aire puro y seco dadas en porcentajes).

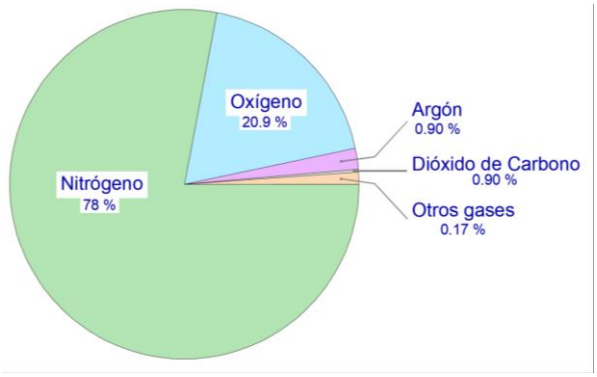


Figura 2.4: Modelo *RGB* de una imagen, cubo unitario *RGB* (imagen propia con base en (González y Wintz,1996)).

Las moléculas de aire no dispersan la luz dado que su tamaño es menor que las longitudes de onda de la radiación visible (Khoury, 2016). Sin embargo, además de los gases que componen el aire, la atmósfera contiene partículas líquidas y sólidas suspendidas en el aire, las cuales provienen de diversas fuentes, entre las principales se encuentran; la actividad volcánica, polvos generados por elementos de construcción y arrastrados por el aire como arena y grava, productos de combustión y evaporación.

Estas partículas causan la dispersión de la luz y absorben su radiación. Lo anterior conlleva a diversos efectos visuales y su gravedad varía de acuerdo al tamaño de las partículas y sus concentraciones, en la Tabla 2.1 se pueden observar el tamaño y la concentración de las partículas presentes en las condiciones climáticas más comunes; niebla, neblina, nubes y lluvia.

Tabla 2.1: Condiciones climáticas y tipos de partículas asociadas, tamaños y concentraciones (creación propia con base en (Khoury, 2016)).

Condición climática	**Tipo de partícula**	**Radio *(μm)***	**Concentración (cm^{-3})**
Aire	Molécula	10^{-4}	10^{19}
Niebla	Aerosol	10^{-2} - 1	10^{3} - 10
Neblina	Gota de agua	1 - 10	100 - 10
Nube	Gota de agua	1 - 10	300 - 10
Lluvia	Gota de agua	10^{2} - 10^{4}	10^{-2} - 10^{-5}

2.4. Dispersión Atmosférica

En visión por computadora se utiliza ampliamente el modelo de dispersión atmosférica propuesto en 2001 por *Narasimhan* (Schechner, Narasimhan, y Nayar, 2001) para describir una imagen con *haze* (ver ecuación 2.1).

$$\boldsymbol{I}(\chi) = \boldsymbol{J}(\chi)t(\chi) + \boldsymbol{A}(1 - t(\chi)) \tag{2.1}$$

donde:

- I es la intensidad observada.
- χ es un píxel con coordenadas *(x, y)*.
- ***J****(x)* es la intensidad real de la escena.
- ***A*** es la luz atmosférica global del aire (*Airlight*).
- *t(x)* es el mapa de transmisión (*Tranmission Map*).

La variable ***I*** es la intensidad de luz observada o capturada por el sensor de la cámara y dentro de los algoritmos de *dehazing* representa la imagen con *haze.* ***J*** es la intensidad real de la escena, es decir, es la intensidad de luz que reflejan los objetos realmente y que en condiciones ideales (sin ningún tipo de *haze)* sería capturada sin efecto alguno sobre la imagen por el sensor de la cámara, por lo cual, dentro de los algoritmos de *dehazing* representa la imagen sin *haze.* ***A*** es la luz atmosférica global del aire, (se describe a detalle en el apartado 2.5). El mapa de transmisión *t* describe la porción de luz que llega al sensor de la cámara sin dispersarse o ser absorbida por las partículas en el aire (su modelo se describe a detalle en el apartado 2.6).

2.4.1. Luz atmosférica (*Airlight*)

Es un fenómeno el cual se presenta visiblemente para el observador a grandes distancias e influye en la iluminación de la escena donde el tono más cercano al horizonte se torna con una mayor luminosidad. La fuente causante de este incremento de luminosidad a la distancia es la luz solar que se encuentra entre el observador y el objeto observado, la cual, es dispersada por el aire hasta los ojos del observador (Salazar, 2019).

2.4.2. Mapa de transmisión de una imagen (*Transmission Map*)

El mapa de transmisión *t* de una imagen representa la porción de luz que llega al sensor de la cámara. En el caso ideal, con un valor de luz atmosférica ***A*** constante, es

decir, siendo la luminosidad en la atmosfera homogénea, el mapa de transmisión *t* se puede calcular mediante la siguiente ecuación:

$$t = e^{-\beta d} \tag{2.2}$$

donde:

- β es el coeficiente de dispersión atmosférica.
- d es la profundidad de la imagen.

Como puede observarse en la ecuación 2.2, el valor de transmisión para cada píxel en la imagen depende de la distancia entre el sensor y el objeto y del coeficiente de dispersión atmosférica β, el cual es una constante asociada a las partículas en la atmósfera de la escena.

2.5. Principio del Canal Oscuro (*DCP*)

El canal oscuro, propuesto por *He* en el año 2010 es una característica que se presenta en las imágenes adquiridas en escenarios exteriores y que tienen la particularidad de no presentar el efecto de *haze*. Está basado en un hecho estadístico el cual parte de que, la mayoría de las ventanas locales aplicadas a imágenes sin afectaciones, contienen pixeles con una intensidad muy baja, cercana al 0, en al menos uno de los 3 canales (*R, G,* o *B).*

El *DCP* se define como: el conjunto de valores (dentro del espacio de color *RGB)* más pequeño dentro de un parche de tamaño *n*, esto se expresa formalmente en la ecuación 2. 3.

$$I^{oscuro}(x,y) = \min[\min I^{c}(z)] \tag{2.3}$$
$$C \in (R,G,B),\ \ z \in \Omega(x,y)$$

donde:

- I^c es un canal de color de ***I*** (*R, G* o *B*).
- Ω *(x, y)* es un parche local de tamaño n centrado en *(x, y)*.
- *z* es el índice del pixel dentro del parche.

En la literatura se han propuesto diversos algoritmos de *dehazing* basados en este principio, al trabajar con imágenes únicas se desconocen los valores de la luz atmosférica *A* y el mapa de transmisión *t,* y el *DCP* ha resultado ser un método muy efectivo para la estimación de estas variables. En la sección 2.9 se describe un algoritmo de *dehazing* basado en el *DCP* y se muestran las ecuaciones que permiten las estimaciones de *A* y *t.*

2.6. *Dehazing*

Dentro del procesamiento digital de imágenes existen varias técnicas para llevar a cabo la eliminación de los defectos en las imágenes generados por cuestiones atmosféricas, que como se han mencionado pueden ser por la niebla, neblina, o la contaminación en el medio entre otros factores. *Dehazing* (también conocido como *Haze Removal* o eliminación de neblina), es el término asociado en el PDI que hace referencia a la aplicación de estas técnicas.

2.6.1. *Dehazing* en imágenes únicas

En la actualidad se tiene una amplia variedad de métodos de *dehazing*, los cuales se pueden clasificar en 2 grupos principales: métodos con una única imagen de entrada y métodos con múltiples entradas (ver Figura 2.5).

Los métodos con múltiples entradas son costosos dado que se requiere información adicional a la imagen por lo cual al tener que procesar más datos su costo computacional es mayor y son menos inviables para implementar en aplicaciones de

tiempo real. Además, generalmente se requieren de dispositivos adicionales a la cámara como pueden ser polarizadores, sensores de reflectancia infrarroja cercana o *Near-Infrared Reflectance* (*NIR*, por sus siglas en inglés) y sensores de distancia, por lo cual se eleva el costo monetario en la obtención de los datos y para su implementación en sistemas embebidos.

Los métodos que utilizan una sola imagen como entrada son más prácticos y accesibles dado que no requieren información adicional de la escena por lo cual son más viables para *RTC*. Los algoritmos que muestran mejores resultados se basan en invertir el modelo de dispersión atmosférica y en la estimación de la luz atmosférica y del mapa de transmisión. Es aquí donde se ubican los algoritmos de *dehazing* basados en el *DCP*.

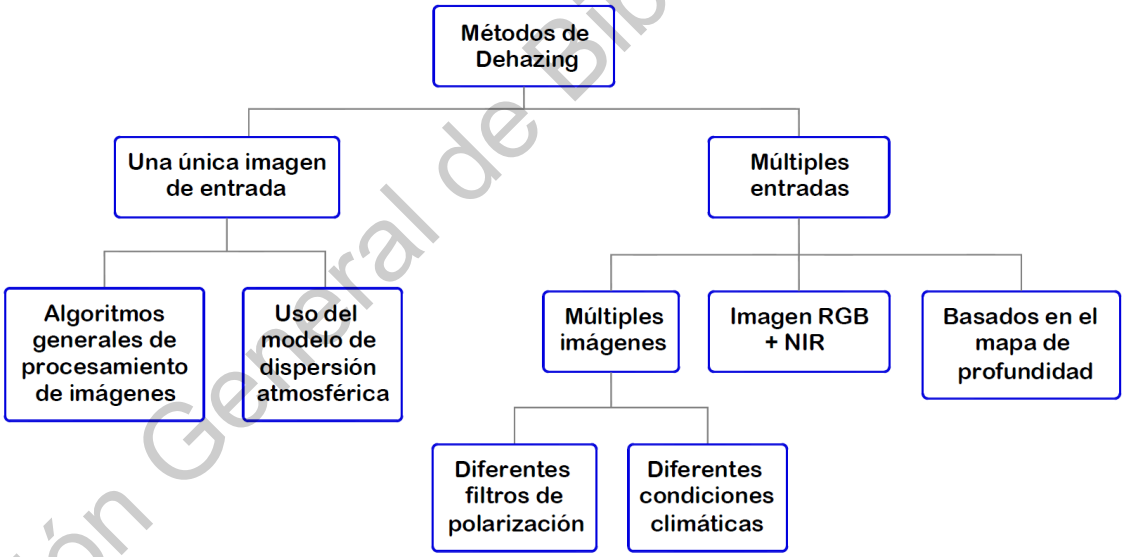


Figura 2.5: Tipos de métodos de *dehazing* (imagen propia con base en (Salazar, 2019) y con información de (Khoury, 2016)).

A pesar de contar con una amplia variedad de métodos para distintos enfoques, cada uno tiene sus limitaciones y no existe un método reportado en la literatura que sea eficiente para cualquier condición de *haze.* Además, debido a la amplia variedad de

entornos y condiciones en las cuales se puede presentar el fenómeno del *haze*, la medición del desempeño de los algoritmos de *dehazing* se torna una tarea complicada.

2.6.2. Algoritmo de *Dehazing* basado en el *DCP*

Se debe especificar que en este trabajo de investigación se utilizó un algoritmo de *dehazing* basado en el *DCP* para su posterior implementación en el *FPGA*, basándonos en el trabajo de *Kaiming He* (He, Sun y Tang, 2010) debido a su simplicidad y además dado que fue el primer algoritmo de *dehazing* propuesto en la literatura que utiliza el *DCP.*

El desempeño del algoritmo ha reportado mejorar dos aspectos fundamentales en la eliminación de neblina (Salazar, 2019):

- Tiempo de procesamiento
- Calidad de restauración de la imagen

La estructura general del algoritmo de *dehazing* propuesto por *K. He* se ilustra en la Figura 2.6. Utilizando el *DCP* la eliminación de neblina se lleva a cabo mediante cinco pasos, a la izquierda de la Figura podemos observar los pasos ordenados de arriba hacia abajo:

1. Obtención del canal oscuro (I^{oscuro}): Se calcula usando parches con un tamaño pequeño y fijo para reducir el tiempo de procesamiento (3x3, 9x9, 11x11).
2. Estimación de la luz atmosférica ($\hat{A}$): es el valor más alto encontrado en el subconjunto del 0.1% de los píxeles con mayor intensidad del canal oscuro.
3. Estimación del mapa de transmisión ($\hat{t}$): se calcula mediante el complemento del canal oscuro de la imagen con *haze* normalizada.

4. Refinamiento del mapa de transmisión ($t_{refinada}$): se realiza mediante una técnica llamada *soft matting*.
5. Recuperación de la luminosidad real de la escena (***J***): se obtiene con la imagen de entrada ***I*** (imagen con *haze*) y las estimaciones de *A* y *t*.

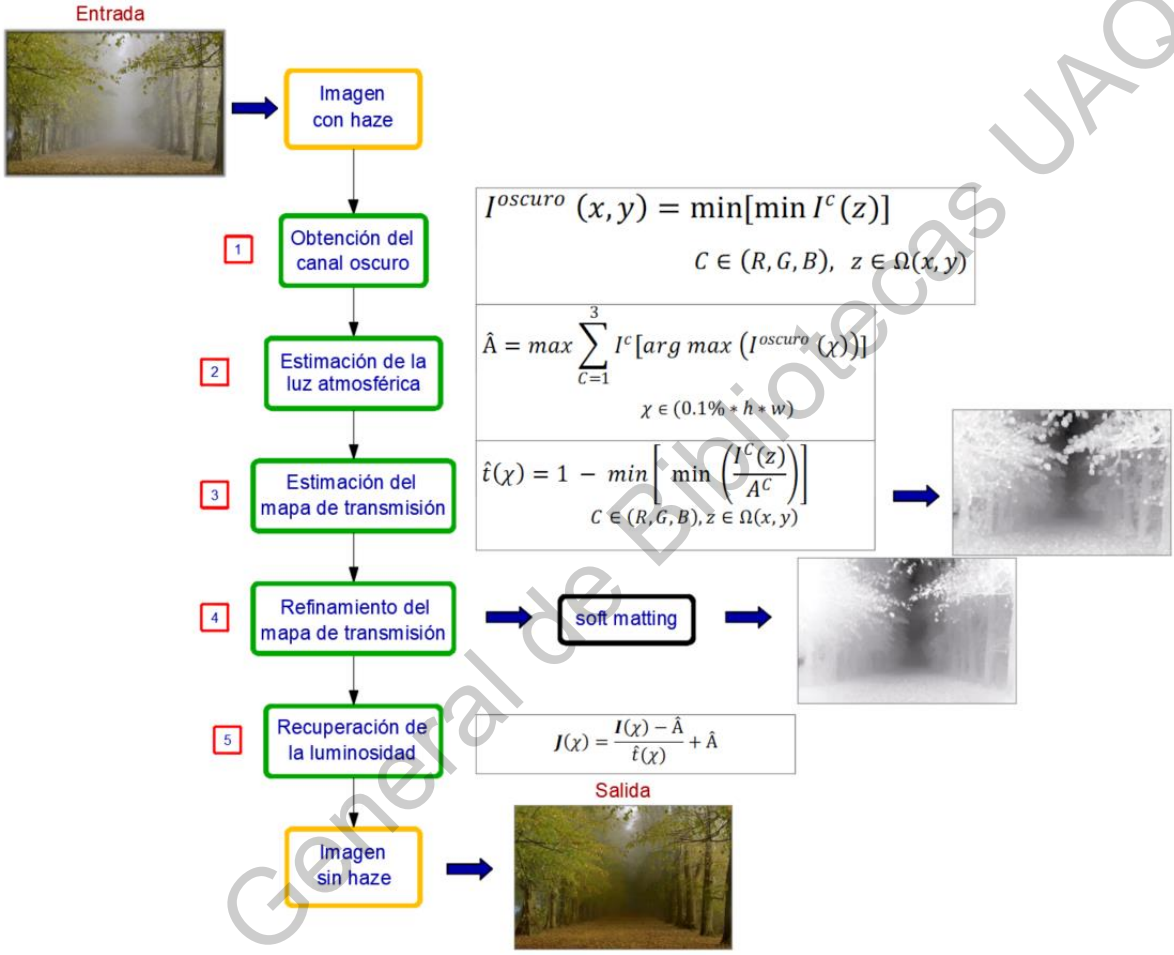


Figura 2.6: Estructura del algoritmo de *dehazing* propuesto por K. He (imagen propia con información de (He, Sun y Tang, 2010)).

En la parte central de la Figura anterior se muestran las ecuaciones que determinan cada uno de los pasos del algoritmo. Los pasos están perfectamente ordenados y los cálculos posteriores requieren de los cálculos realizados previamente. En las ecuaciones podemos observar que el método de *K. He* hace uso en gran medida del canal oscuro para calcular las diferentes etapas, es por ello que como primer paso

se realiza el cálculo del canal oscuro para posteriormente realizar las estimaciones de *A* y *t*, las cuales están representadas con $\hat{A}$ y $\hat{t}$ respectivamente.

La estimación de la luz atmosférica se representa mediante la siguiente ecuación:

$$\hat{\mathrm{A}} = max \sum_{C=1}^{3} I^{c}[arg\ max\ (I^{oscuro}(\chi))] \qquad (2.4)$$
$$\chi \in (0.1\% * h * w)$$

En la ecuación 2.4 se observa claramente que el cálculo de $\hat{A}$ depende directamente del canal oscuro. Asimismo, la estimación del mapa de transmisión se representa mediante la siguiente ecuación:

$$\hat{t}(\chi) = 1 - min\left[\ min\left(\frac{I^{C}(z)}{A^{C}}\right)\right] \qquad (2.5)$$
$$C \in (R, G, B), z \in \Omega(x, y)$$

Si se analiza la ecuación 2.5, se puede observar que depende implícitamente del canal oscuro, dado que se calcula el valor mínimo de los canales de color de un pixel normalizados con la luz ambiental en ese canal. Esta ecuación se establece mediante la suposición de que la transmisión en la ventana local Ω *(x, y)* es constante, sin embargo, la realidad es que la transmisión no siempre es constante en una ventana y el cálculo del canal oscuro basado en parches conduce a un mapa de transmisión visualmente borroso, por lo cual, la estimación del mapa de transmisión suele ser inexacta.

Para resolver lo anterior suele aplicarse un paso adicional en el cual se refina el mapa de transmisión mediante diversas técnicas. La técnica que se aplica en el método original del *DCP* de *K. He* se conoce como *soft matting.* Entre las técnicas que se han utilizado en trabajos posteriores y que han mostrado ser eficientes al realizar esta tarea se encuentran: el filtrado bilateral, el filtrado gaussiano y el filtrado guiado.

El refinamiento del mapa de transmisión corrige algunos defectos como los halos de luz que se pueden llegar a presentar en las imágenes recuperadas, o que las escenas no parezcan naturales, debido a que los valores de $\hat{A}$ y $\hat{t}$, son meramente estimaciones. El refinamiento mejora visualmente las imágenes recuperadas, sin embargo, esta etapa aumenta considerablemente el tiempo de procesamiento computacional, por lo cual, en este trabajo se ha excluido esta etapa.

Una vez obtenidas las estimaciones de la luz atmosférica y el mapa de transmisión, se puede aplicar el paso final y mediante la ecuación 2.6 recuperar la luminosidad de la escena para cada valor de píxel χ.

$$\boldsymbol{J}(\chi) = \frac{\boldsymbol{I}(\chi) - \hat{\mathrm{A}}}{\hat{t}(\chi)} + \hat{\mathrm{A}} \tag{2.6}$$

2.7. Sistemas Embebidos

Un sistema embebido se puede definir como un sistema electrónico para procesamiento de datos, el cual está programado para realizar tareas o funciones específicas. Además, adicionalmente el sistema puede incluir componentes electrónicos, eléctricos y mecánicos (Aceves y Arreguin, 2011).

2.8. *FPGAs*

Entre los dispositivos que realizan el procesamiento de datos en un sistema embebido se encuentran los *FPGAs*.

Una Matriz de Compuertas Lógicas Programable en Campo o *Field-Programmable Gate Array* (*FPGA*, por sus siglas en inglés) es un dispositivo lógico programable, el cual puede ser programado por el usuario para llevar a cabo funciones lógicas específicas.

2.8.1. Arquitectura interna de los *FPGAs*

Lo que se realiza al programar un *FPGA* es la implementación de un circuito previamente diseñado, para ello, los *FPGA* tienen una estructura interna (a lo que llamamos la arquitectura interna del *FPGA*) compuesta por 3 elementos básicos:

1. Bloques de entrada/salida (E/S)
2. Bloques lógicos (BL)
3. Interconexiones programables

Esta es la disposición (general) de los bloques que componen un *FPGA* (ver Figura 2.7).

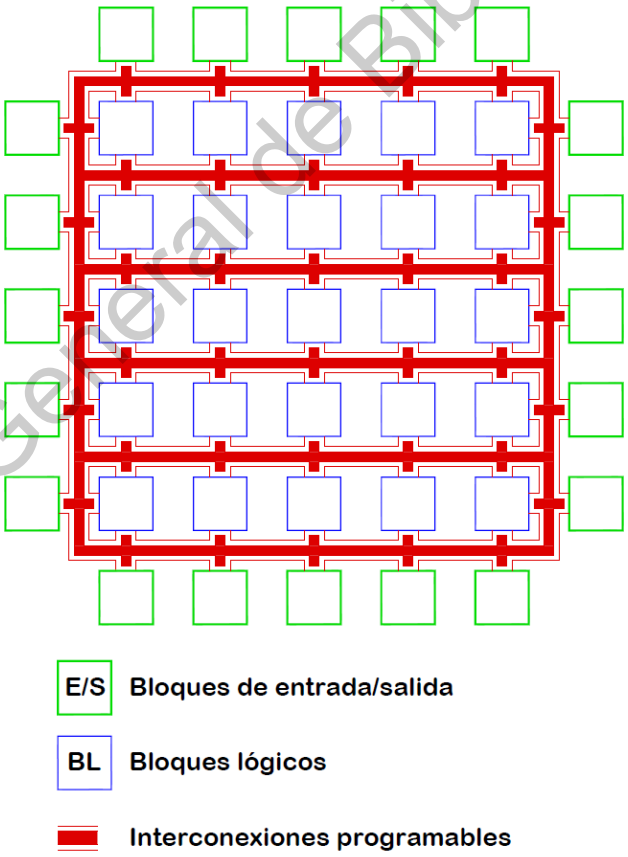


Figura 2.7: Arquitectura interna de un *FPGA* (imagen propia con información de (Floyd, 2006)).

2.8.2. El lenguaje *VHDL*

Existen diversas formas de introducir el diseño del circuito lógico en el *FPGA*, entre las más utilizadas se encuentran las basadas en texto y gráficamente. En este proyecto se utilizó la manera textual, mediante el lenguaje de descripción de hardware *VHDL*.

Un Lenguaje de Descripción de Hardware o *Hardware Description Language* (*HDL*, por sus siglas en inglés) es un lenguaje que se utiliza para describir la estructura de circuitos electrónicos.

El lenguaje descriptivo que ha tenido mayor difusión en los últimos años y que es el mayormente utilizado en la descripción de hardware es el *VHDL*. Esto se debe a que ha sido estandarizado por el *IEEE* y de esta forma son portables a cualquier plataforma (Troncoso, 2007).

VHDL es un acrónimo que proviene de la combinación de dos acrónimos: *VHSIC (Very High Speed Integrated Circuit)* y *HDL (Hardware Description Language.)* Este lenguaje es utilizado para programar Dispositivos Lógicos Programables o *Programmable Logic Device* (*PLDs*, por sus siglas en inglés), Circuitos Integrados para Aplicaciones Específicas o Application-Specific Integrated Circuit (*ASICs*, por sus siglas en inglés) y dispositivos similares, así como Matrices de Compuertas Lógicas Programables en Campo o *Field-Programmable Gate Array* (*FPGAs*, por sus siglas en inglés), siendo esta la plataforma para el procesamiento digital de las imágenes utilizada en el presente trabajo.

CAPÍTULO 3

Metodología

3.1. Descripción general

Para llevar a cabo la implementación del algoritmo de *dehazing* en el *FPGA*, se siguió una metodología que consistió de tres etapas principales, las cuales son:

1. Generación de las imágenes con *haze* sintético
2. Arquitectura general para PDI en el *FPGA*
    a. Diseño
    b. Implementación
3. Arquitectura para realizar *dehazing* en el *FPGA*
    a. Diseño
    b. Implementación

En la Figura 3.1 se puede observar de manera general las etapas de la metodología aplicada, así como también los pasos (por orden de prioridad) que conforman a cada una de ellas. A continuación, se describen de manera breve cada una de las etapas que conforman la metodología:

- **Etapa 1:** Es el primer paso que se llevó a cabo y comprende la generación de las imágenes con las cuales se trabajó en este proyecto. Estas imágenes fueron creadas a partir de la Base de Datos o *Data Base* (*DB*, por sus siglas en inglés): *Kitti* (Karlsruhe Institute of Technology, 2020), la cual es una *DB* que pertenece a un *Benchkmark* que contiene imágenes de escenas en exteriores y que fue creado para realizar tareas de visión por computadora relacionadas al desarrollo de la conducción autónoma. La *DB Kitti* nos proporcionó el mapa de profundidad de cada escena, con

los cuales se generaron diversos efectos de *haze* sintético sobre las imágenes. En la sección 3.2 de este capítulo se describe más a detalle este proceso.

- **Etapa 2:** Esta etapa se dividió en 3 sub-etapas, en la primera se desarrolló el modelo de comunicación entre el *PC* y el *FPGA* vía serial, lo cual permitió realizar la adquisición de las imágenes en el *FPGA*. Una vez adquiridas las imágenes, en la segunda etapa se desarrolló la arquitectura para el almacenamiento de las imágenes en las memorias del *FPGA* (*Flash* y *RAM*). Por último, se desarrolló el controlador de vídeo, lo cual permitió la exposición de las imágenes mediante el puerto *VGA* del *FPGA*. En la sección 3.3 de este capítulo se describe más a detalle este proceso.

- **Etapa 3:** Finalmente, en esta etapa se diseñó la arquitectura para realizar *dehazing*, para lo cual se crearon los bloques funcionales referentes a los procesos involucrados en el algoritmo de *dehazing,* el cual consta de 4 pasos: obtención del canal oscuro, estimación de la luz atmosférica, estimación del mapa de transmisión y la recuperación de la luminosidad en la escena. En la Figura 3.2 podemos observar estos procesos etiquetados como 3.1, 3.2, 3.3 y 3.4, y se encuentran numerados por orden de prioridad, siendo la estimación de la luz atmosférica ($\hat{A}$) dependiente de la obtención del canal oscuro (I^{oscuro}), la estimación del mapa de transmisión ($\hat{t}$) dependiente de $\hat{A}$ y I^{oscuro}, y finalmente la recuperación de la luminosidad (J) es dependiente de todas las variables anteriores. En la sección 3.4 de este capítulo se describe más a detalle este proceso.

A lo largo de este capítulo se describe más a detalle cada una de las etapas.

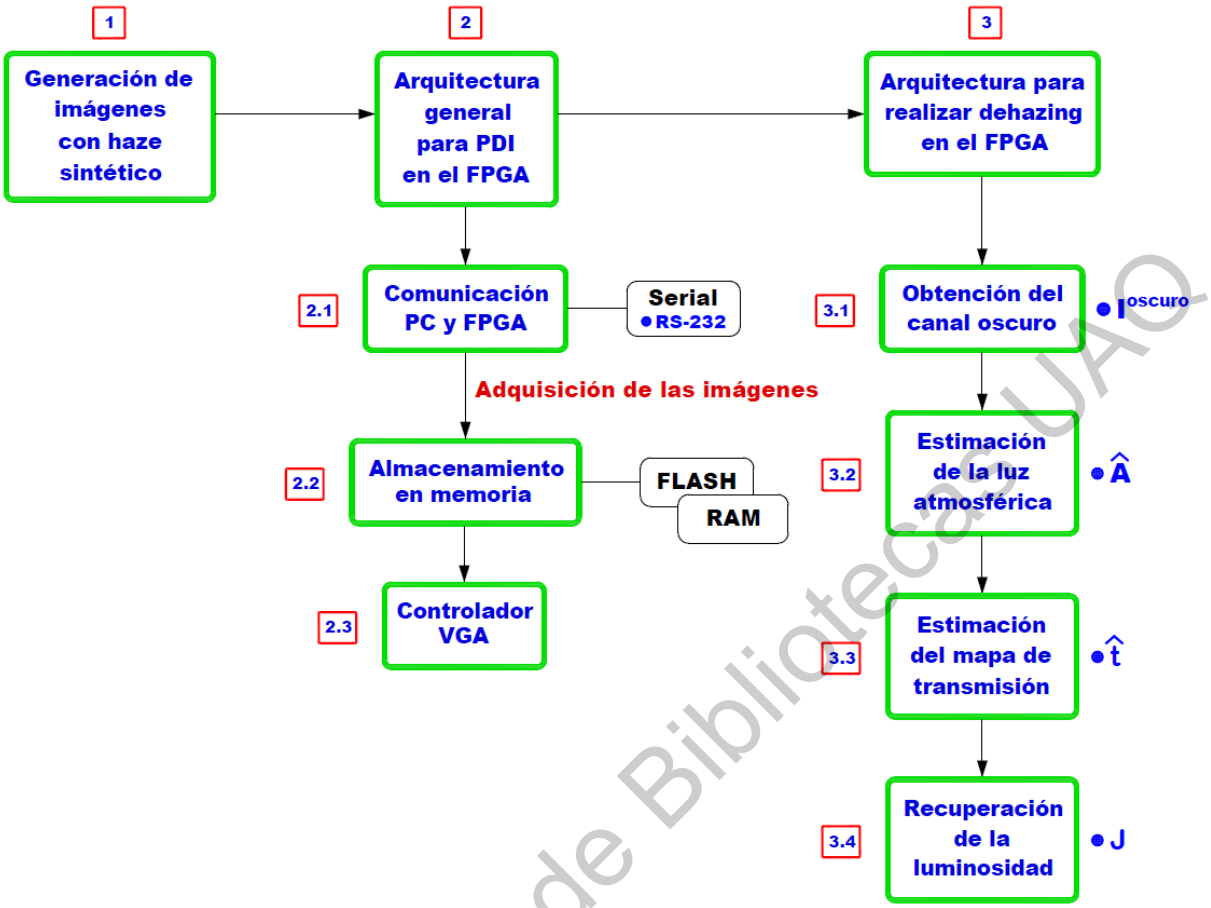


Figura 3.1: Metodología general para la implementación del algoritmo de *dehazing* en el *FPGA* (imagen propia).

Las etapas se encuentran numeradas por orden de prioridad, siendo las posteriores dependientes de las anteriores. La arquitectura se desarrolló empleando el modelo *top-down*, el cual consiste en partir de lo general a lo particular, dividiendo el sistema complejo en varios niveles de jerarquía, hasta alcanzar los niveles más básicos donde se requieren de bloques muy simples y elementales para la descripción del hardware como pueden ser: conmutadores, comparadores, sumadores, multiplicadores, tablas de consulta o *lookup tables (LUTs)*, registros, contadores, secuenciadores y Máquinas de Estados Finitos o *Finite State Machines* (*FSMs,* por sus siglas en inglés).

Las arquitecturas implicaron dos aspectos relevantes a tomarse en cuenta: su diseño y su implementación. El diseño, como ya se ha mencionado es de tipo jerárquico, el cual se desarrolló mediante una distribución en bloques funcionales comprendida por niveles (ver Figura 3.2). Mientras que la implementación requiere tomar en cuenta los recursos propios del *FPGA*, así como las características y periféricos disponibles en la tarjeta de desarrollo, los cuales se muestran en la sección: Especificaciones del Sistema, contenida en este capítulo.

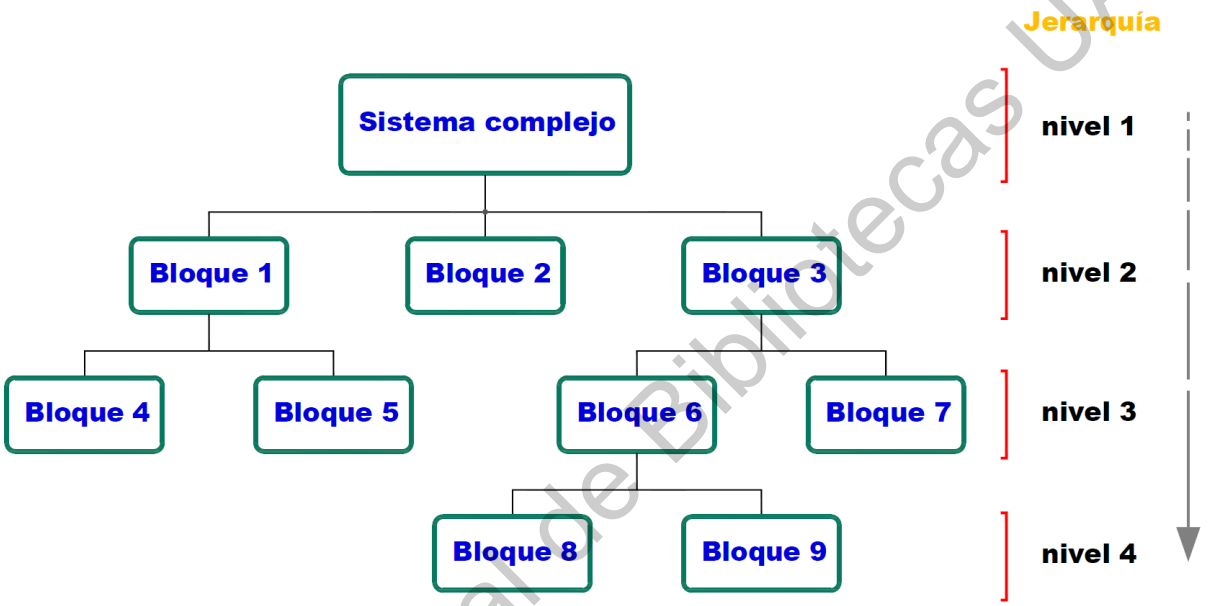


Figura 3.2: Diseño jerárquico *top-down* (imagen propia con base en (Troncoso, 2007)).

3.2. Generación de las imágenes usando la *DB: Kitti*

El punto de partida dentro de la metodología desarrollada, se encuentra previo al diseño de la arquitectura en el *FPGA* (ver Figura 3.1). Siendo un proyecto sobre PDI, las imágenes que fueron utilizadas son elementos esenciales, la materia prima para el trabajo.

Antes de llevar a cabo la adquisición de las imágenes en el sistema embebido, es recomendable contar con una base de datos bien definida y conocer sus características,

para poder resolver el problema de la manera más adecuada. Dentro de la arquitectura en *hardware* se realizó una interfaz RS-232 para realizar la adquisición de las imágenes, pero, previo a esto, se llevó a cabo una metodología en torno a la obtención de las imágenes de interés para este proyecto. Dicho esto, es necesario recalcar que el proyecto está enfocado al mejoramiento de la calidad en imágenes relacionadas con sistemas de conducción autónoma para su posterior uso en tareas de visión por computadora, ya que, de ello dependió la correcta selección de las imágenes, y el pre procesamiento que se les aplicó.

Por ello es importante darle un peso adecuado a este tema y dedicar esta sección para detallar el origen y la obtención de las imágenes, así como también, para describir su composición. Las imágenes originales se obtuvieron de un *Benchkmark,* proveniente del Instituto de Tecnología *Karlsruhe* o *Karlsruhe* Institute of Technology (*KIT*, por sus siglas en inglés) (Karlsruhe Institute of Technology, 2020). *Kitti* es un *Benchkmark* dedicado a la investigación y desarrollo de técnicas y algoritmos para la mejora de las tareas de visión por computadora de mayor interés dentro del área de los vehículos autónomos de uso particular, como lo son: visión estéreo, flujo óptico, profundidad en la escena, odometría, detección y seguimiento de objetos y detección de la carretera.

L*a DB: Kitti,* contiene 1000 imágenes de escenarios en carreteras, las imágenes vienen además acompañadas además de su mapa de profundidad o *sparse depth map* correspondiente. Los datos de profundidad de las escenas fueron tomados por un sensor de Detección y Alcance de Imágenes Láser o *Laser Imaging Detection and Ranging* (*LIDAR,* por sus siglas en inglés). Los *LIDAR* son sensores creados por la empresa *Velodyne,* los cuales, detectan y miden la distancia hacia los objetos mediante la emisión de pulsos laser infrarrojos.

En base a la teoría del modelo de dispersión atmosférica el cual describe una imagen con *haze* (ver ecuación 2.1), podemos generar una imagen con *haze* mediante 3 variables*:*

1. la intensidad real de la escena (*J*).

2. el mapa de transmisión (*t*).
3. la luz atmosférica global (*A*).

Partiendo de lo anterior, se diseñó una metodología para la generación de imágenes con *haze* (ver Figura 3.3) a partir de la *DB: Kitti.* Dado que la escena, resultado de estas operaciones no es una escena captada con una cámara en un entorno real, la imagen producida tendrá un *haze* sintético.

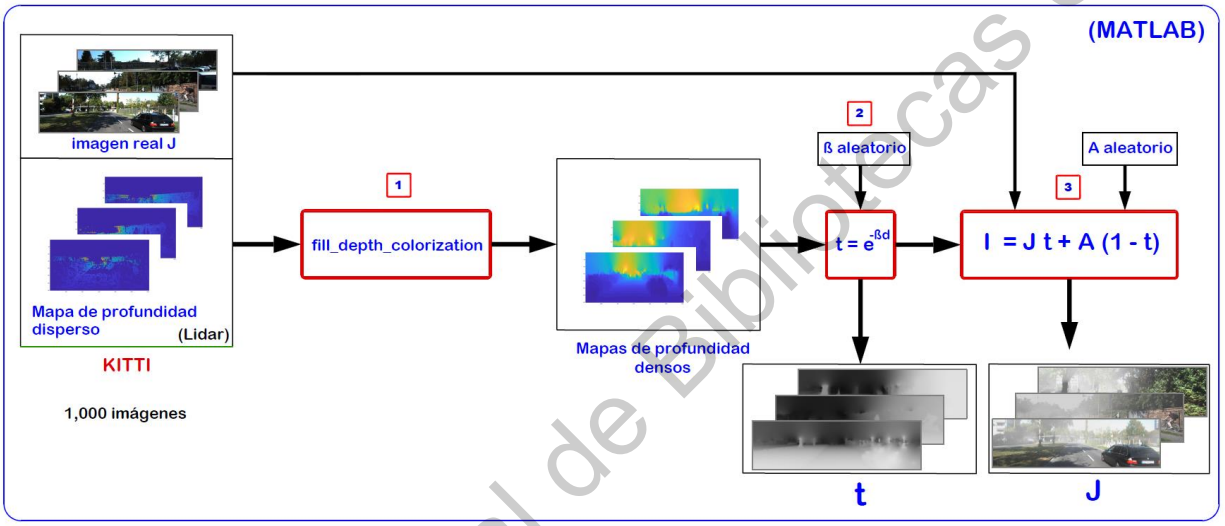


Figura 3.3: Metodología para la generación de imágenes con *haze* sintético usando la base de datos *Kitti* (imagen propia).

Para aplicar la metodología de la Figura 3.3. se utilizó el *software Matlab,* versión *R2017b,* esta metodología se describe de la siguiente manera:

El primer paso fue procesar los mapas de profundidad dispersos mediante una función creada por *Nathan Silberman* (Nathan Silberman, (s.f.)) la cuál es una adaptación del código de colorización creado por *Anat Levin* (Levin, (s.f.)). Esta función transforma los mapas dispersos en mapas densos, esto se realizó debido a que para la obtención de los mapas de transmisión es necesario introducir en el modelo de dispersión atmosférica (representado por la ecuación 2.2), los mapas de profundidad densos.

Una vez que se obtuvo de cada escena la profundidad densa posterior o *Dense Depth Posterior* (*DDP,* por sus siglas en inglés) se obtuvieron los mapas de transmisión, para ello, fue necesario además agregar el valor del coeficiente de dispersión atmosférica (β), esta es una variable de la cual dependen los niveles de *haze* en la escena, sabiendo esto, se agregaron valores aleatorios para crear diferentes efectos de *haze* sintético.

Finalmente aplicando el modelo de dispersión atmosférica (ver ecuación 2.1) con la transmisión las imágenes reales y valores aleatorios para el *Airlight* (*A)*, se generaron las imágenes con *haze* sintético. Se generaron 3 imágenes con *haze* sintético para cada una de las imágenes reales, es decir, con las 1,000 imágenes de la *DB: Kitti* se obtuvieron 3,000 imágenes sintéticas, las cuales fueron utilizadas posteriormente para la aplicación de la arquitectura creada para realizar *dehazing* en el *FPGA*.

3.3. Arquitectura general para PDI en el *FPGA*

Para realizar el procesamiento de las imágenes en el *FPGA* se requirió de un modelo generalizado que interactuará entre la *PC* y el *FPGA* para la adquisición de las imágenes con *haze* generadas a partir de la *DB: Kitti,* para posteriormente almacenarlas en la memoria interna del *FPGA.* Seguido de esto fue necesario crear una interfaz *VGA* en el *FPGA* para que, de esta manera, se pudieran visualizar las imágenes procesadas por el *FPGA* (ver Figura 3.4).

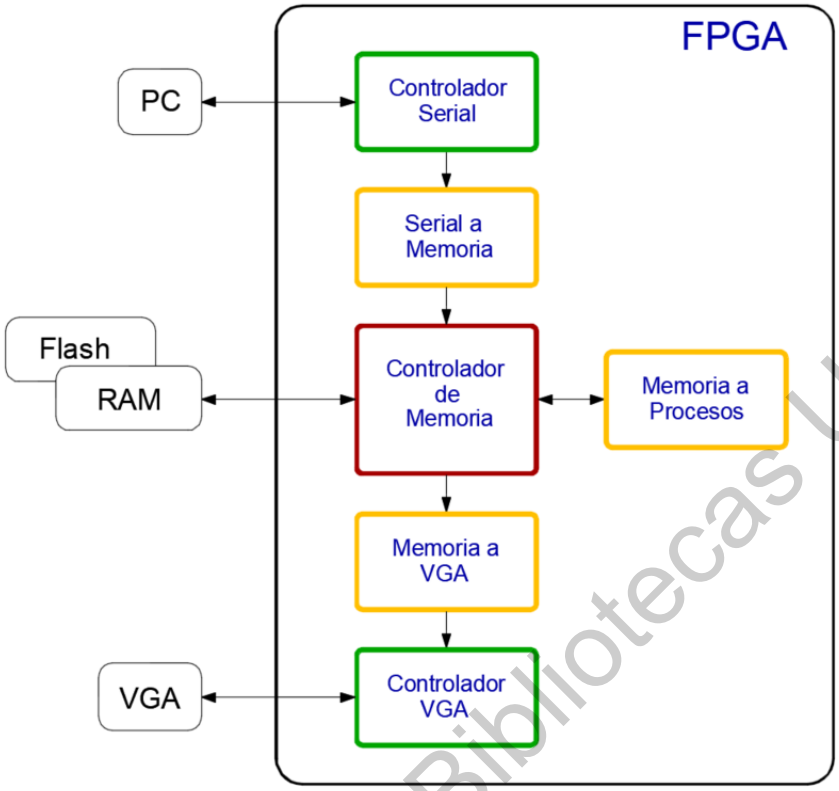


Figura 3.4: Diagrama a bloques generalizado de la arquitectura para procesamiento de imágenes en *FPGA* (imagen propia con base en (Blanco, 2015)).

Como parte de la segunda etapa de la etapa metodología, comenzamos preparando el entorno. Para ello lo primero que hay que tomar en cuenta para realizar e implementar un diseño en un *FPGA*, son las especificaciones de la tarjeta de desarrollo que se va a utilizar. En este proyecto el sistema se diseñó enfocando a la tarjeta de desarrollo *Nexys 4 DDR* de la marca *Digilent* (Digilentinc, (s.f.)).

3.3.1. Especificaciones del Sistema

Este apartado en particular corresponde a los materiales utilizados en este proyecto. La *Nexys 4 DDR* es una tarjeta de desarrollo en la cual se pueden realizar

una gran variedad de diseños y con la facilidad de poder probarlos con su gran variedad de dispositivos ya instalados.

Los principales puertos y periféricos de la tarjeta son los siguientes:

- 16 interruptores
- puente de selección *UART – USB*
- Salida *VGA* de 12bits
- Acelerómetro de 3 ejes
- Memoria *RAM* tipo *DDR2 (128 MiB)*
- *PMOD* de señales analógicas
- 16 *LEDs*
- 2 *LEDs RGB*
- Salida de audio por *PWM*
- Sensor de temperatura
- Serial *FLASH*
- Puerto de programación y comunicación *USB-JTAG*
- 2 *Displays* de 7 segmentos de 4 dígitos, en total 8 dígitos multiplexados
- Conector de *micro SD*
- Micrófono *PDM*
- 10/100 *Ethernet PHY*
- 4 conectores *Pmod*
- *USB HID HOST* para: mouse, teclado y memorias *USB*

También es importante tomar en cuenta las características del *FPGA,* el cual es un *Artix – 7 100T*, las cuales se mencionan a continuación:

- 15,850 sectores lógicos con 4 LUT de 6 entradas y 8 *Flip-Flops*
- 4,860 *Kbits* de bloques rápidos de memoria *RAM*
- 6 administradores de reloj, cada uno con bucle de desplazamiento de fase o *Phase-Locked Loop* (*PLL*, por sus siglas en inglés)

- 240 sectores *DSP*
- Reloj interno con frecuencia superiores a 450 *MHz*
- Conversor analógico-digital interno (*XADC*).

En la Figura 3.5 se muestran enumeradas las ubicaciones de los principales componentes de la tarjeta de desarrollo *Nexys 4 DDR*. En la Tabla 3.1 se muestra la descripción de los componentes de la Figura 3.5.

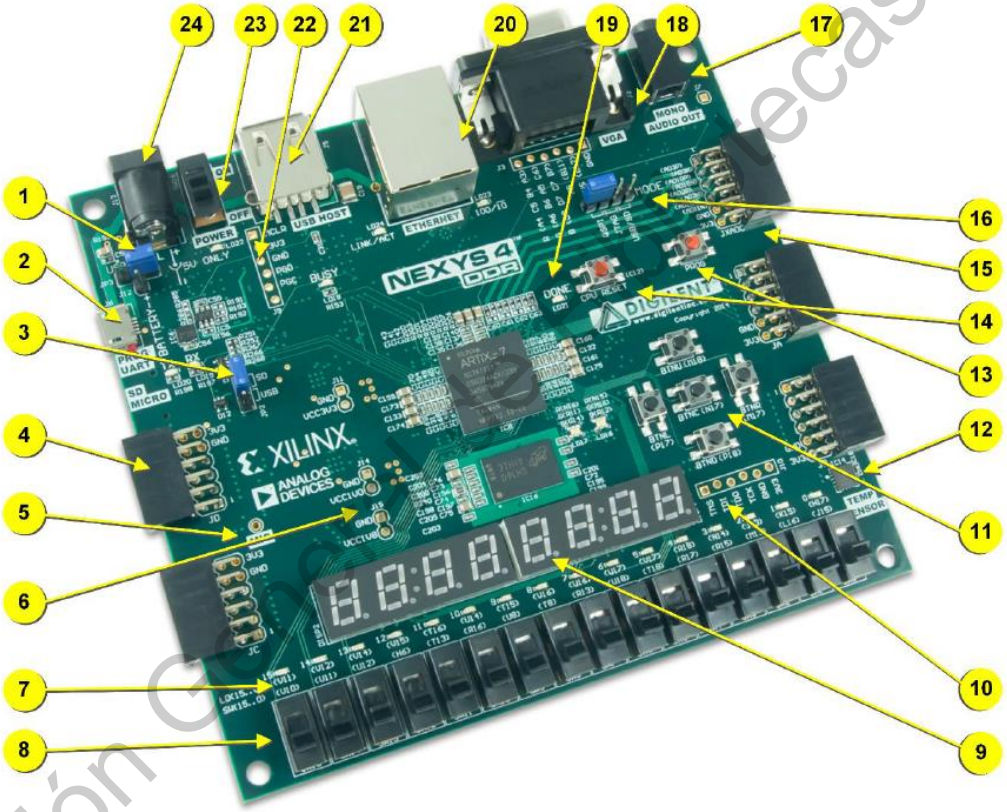


Figura 3.5: Principales componente de la tarjeta *Nexys 4 DDR* (Digilentinc, (s.f.)).

Tabla 3.1: Principales componentes de la tarjeta Nexys 4 DDR y su descripción (creación propia con base en (Digilentinc, (s.f.))).

N°	Descripción del Componente
1	Jumper para seleccionar la fuente de alimentación
2	Puerto micro *USB* para la programación y para el *UART*
3	Jumper para seleccionar el modo de configuración
4	Puertos *Pmod*
5	Micrófono
6	Punto de prueba del voltaje de alimentación
7	*LEDs* (16)
8	Interruptores (16)
9	Pantalla de 7 segmentos
10	Conector para *JTAG* tradicional
11	Interruptores Pulsadores
12	Sensor de Temperatura
13	Botón de *reset* de configuración del *FPGA*
14	Botón de *reset* de *CPU*
15	*Pmod* de conector *XADC*
16	Jumper de selección para modo de programación
17	Conector de audio
18	Conector de monitor *VGA*
19	*Led* indicador de una configuración exitosa
20	Conector de *ethernet*
21	Conector *USB host*
22	*Jumper* para la programación de los microcontroladores *PIC24*
23	Interruptor *on/off*
24	Conector de entrada de alimentación

3.3.2. Interfaz de Comunicación entre el *PC* y el *FPGA*

Una vez diseñada la arquitectura general anteriormente mostrada, se prosiguió a realizar el diseño de la comunicación entre la *PC* y el *FPGA*, con el objetivo de llevar a cabo la adquisición de las imágenes en el *FPGA*.

La comunicación establecida para estos fines, fue el protocolo de comunicación serial mediante realizó mediante el puerto serie del *FPGA*, para lo cual, se creó una interfaz en *Visual Studio 2017* (ver Figura 3.6).

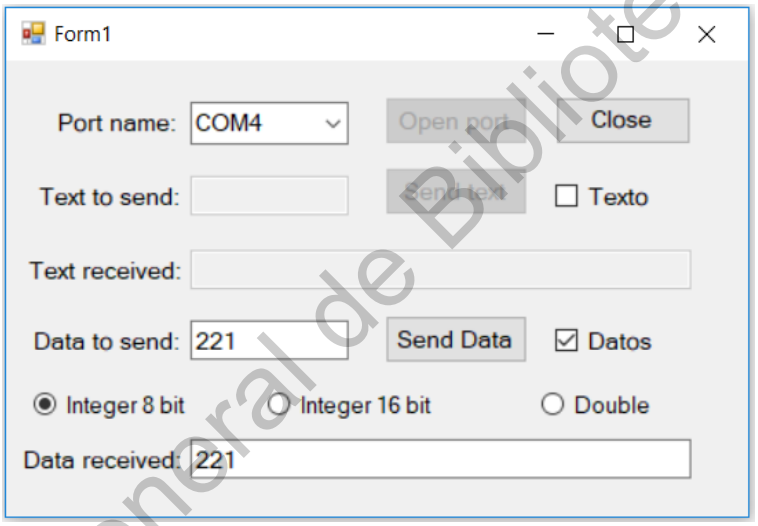


Figura 3.6: Interfaz de comunicación serial en *Visual Studio 2017* (imagen propia).

Esta interfaz permite transferir los datos de los canales *RGB* de cada uno de los pixeles a una velocidad de 115,200 baudios. El puerto *VGA* de la *Nexys2* es de 8 bit de color por lo que cada pixel de la imagen se envía como un arreglo de la forma: [RRRGGGBB].

3.3.3. Almacenamiento en el *FPGA*

Como ya se ha descrito, el modelo permite almacenar las imágenes tanto en la memoria *FLASH* como en la memoria externa del *FPGA*. Esto se definió así con el objetivo de utilizar la memoria *FLASH* para almacenar las imágenes con *haze* que se reciben de la *PC* y los resultados finales, de esta forma no es necesario cargar las imágenes para cada nuevo experimento, o cada que se realicen modificaciones en el programa del *FPGA*. Por otro lado, la memoria *RAM* se utiliza para almacenar los cálculos y operaciones puntuales que se realicen mediante los procesos que se aplican a las imágenes con *haze*.

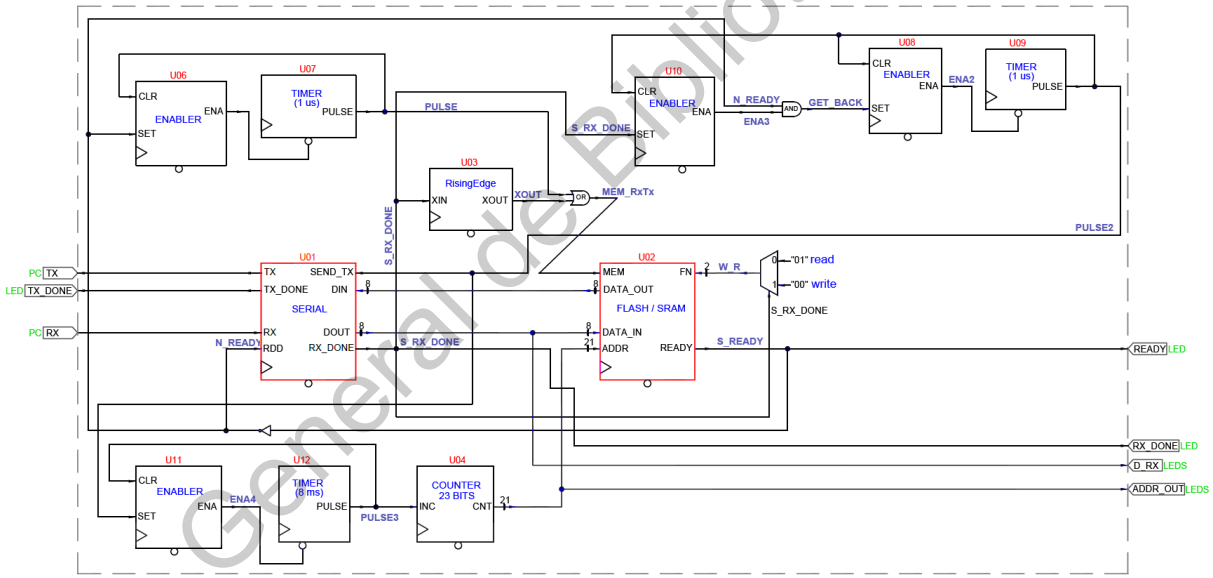


Figura 3.7: Arquitectura para el almacenamiento en memoria *RAM/FLASH* vía serial (imagen propia).

Una vez diseñado el controlador de comunicación serial se añadió la arquitectura para el almacenamiento mostrada en la Figura 3.7, la cual funciona de la siguiente manera:

- La imagen se comienza a recibir por el puerto serial, una vez que llego el byte completo se activa una bandera
- La cual ordena al controlador de memoria que se escriba el dato
- Por último, se incrementa la dirección de memoria y se pone en espera para recibir el siguiente byte.

Inicialmente se estaba trabajando con la tarjeta Nexys2 y el *software Xilinx* esto porque se contaba con el trabajo de Foper de Elías ya mencionado, pero dado a que es una tarjeta algo obsoleta, se estaban teniendo un poco de dificultades con el manejo de la memoria *RAM*. Con la Nexys4 se tuvo la ventaja del software vivado que tiene más herramientas, se pueden crear bloques de memoria *RAM* y además se pueden cargar archivos iniciales desde la *PC*, con lo cual se puede cargar una imagen inicial.

Para poder hacer uso de la memoria *RAM* de tipo *DDR2* la página web de *Xilinx* proporciona un bloque para pasar de una memoria *RAM* a una de tipo *DDR2*. Con esto se simplifica bastante la comunicación con la memoria *DDR2* en donde básicamente se tienen unos bits de control que conectan los 2 bloques lógicos y el controlador pasa a funcionar como un bloque lógico de una memoria estática.

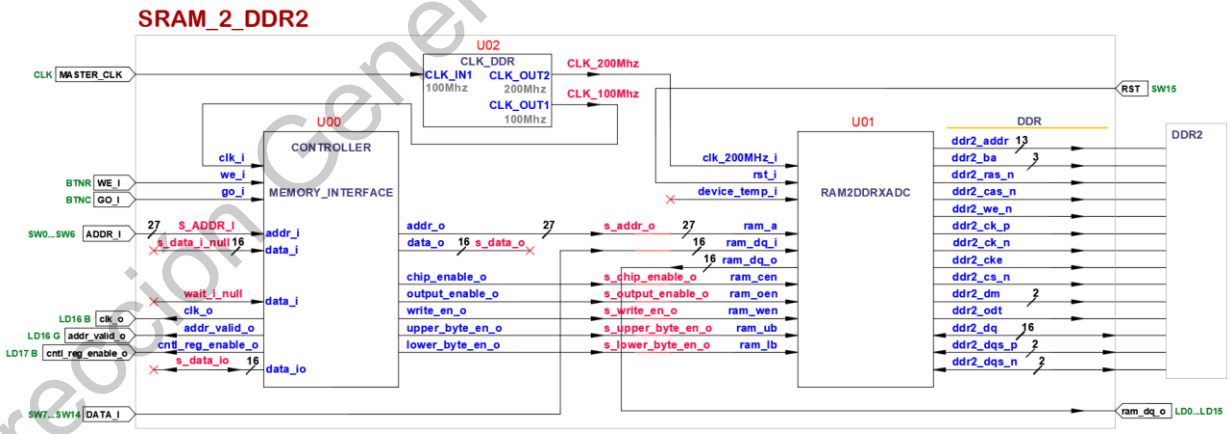


Figura 3.8: Adaptación del almacenamiento en memorias (imagen propia).

3.3.4. Interfaz *VGA*

La conexión entre el *FPGA* y el monitor *VGA* se realizó para poder visualizar las imágenes, el *FPGA* tiene un puerto *VGA* de 12 bits de color.

Como ya se ha mencionado, en un principio se utilizó el propio puerto de la *Nexys 4DDR*, sin embargo, se desea tener una mayor gama de colores para mostrar las imágenes con una mayor fidelidad, para lo cual se ha diseñado una tarjeta con un conector *VGA* de 24 bits el diagrama electrónico se muestra en la Figura 6. Los diagramas de conexión entre el puerto *VGA* y el *FPGA* propios de la tarjeta *Nexys2*, los cuales sirvieron de guía para realizar el prototipo de la interfaz *VGA* de 24 bits de color, se pueden consultar en el documento *Nexys2 Reference Manual* en el apartado *VGA Port* (página 10) (Nathan Silberman, (s.f.)).

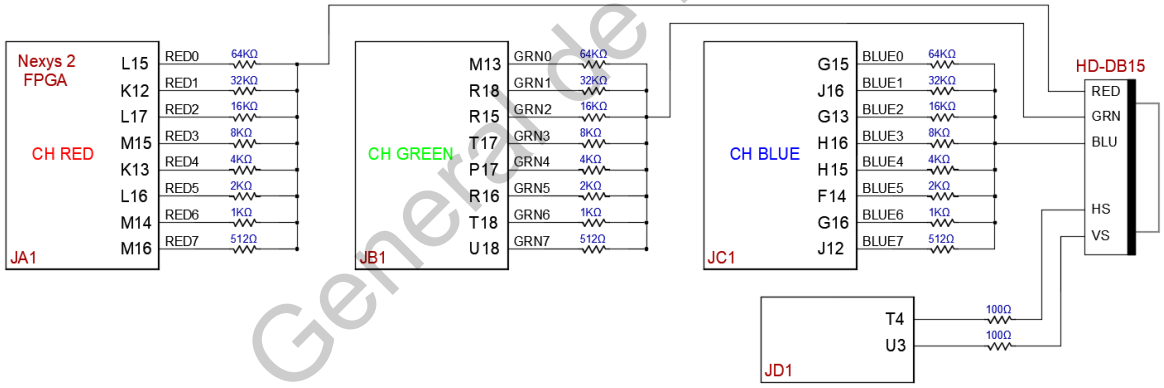


Figura 3.9: Diagrama de conexiones entre el *FPGA* y el Puerto *VGA* (imagen propia).

La cantidad de colores que se puede tener con n cantidad de bits se describe en la ecuación 3.1.

$$n° \; de \; colores = \; 2^{n \; bits} \tag{3.1}$$

Por lo que con 24 *bits* tenemos una gama de 16.7 millones de colores, lo cual

nos permite mostrar las imágenes con una mayor calidad. El prototipo de esta tarjeta puede observarse en la Figura 3.10.

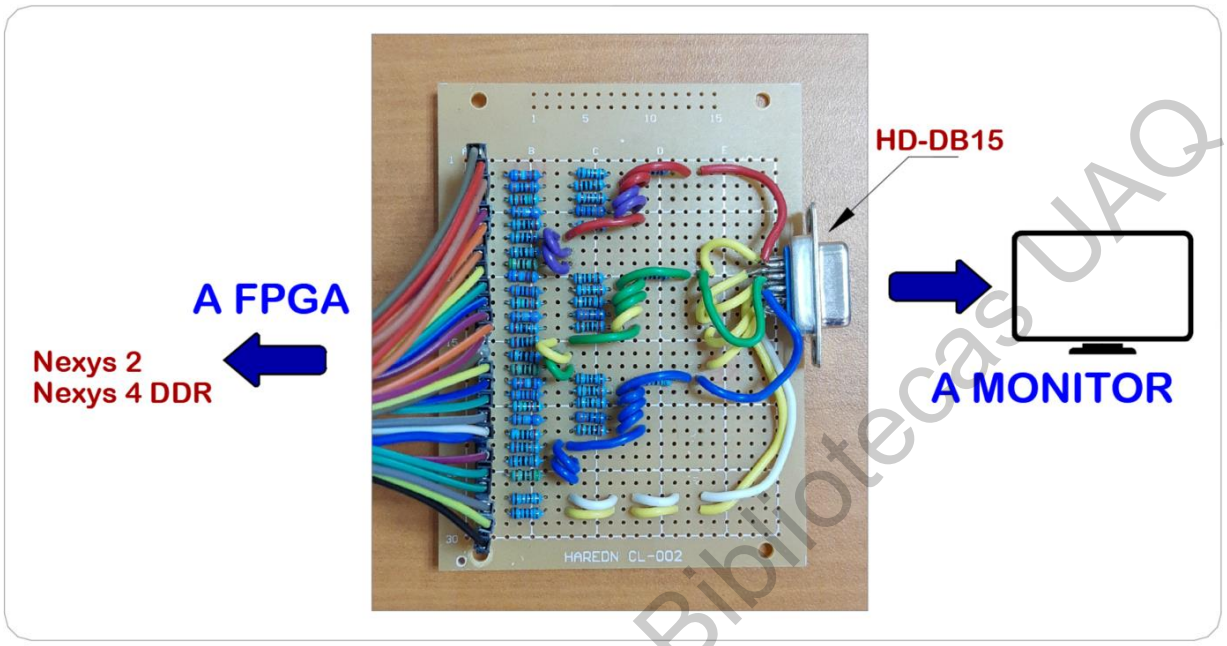


Figura 3.10: Prototipo de una interface *VGA* de 24 bits de color para conexión entre *FPGA* y monitor (imagen propia).

3.4. Arquitectura para realizar *Dehazing* en el *FPGA*

Una vez tenemos el entorno general para realizar procesamiento de imágenes en el *FPGA* procedimos a implementar el algoritmo de *dehazing* para lo cual, se creó una arquitectura con las características y elementos mostrados en la Figura 3.1. La arquitectura se describió y se sintetizó utilizando el software *Vivado* versión 2018.3 de la compañía *Xilinx*. Básicamente se crearon los bloques lógicos funcionales referentes a los procesos involucrados en el algoritmo de *dehazing*, el cual consta de 4 pasos. Estos bloques lógicos se encuentran en el segundo nivel según el modelo de jerarquías utilizado (ver Figura 3.2).

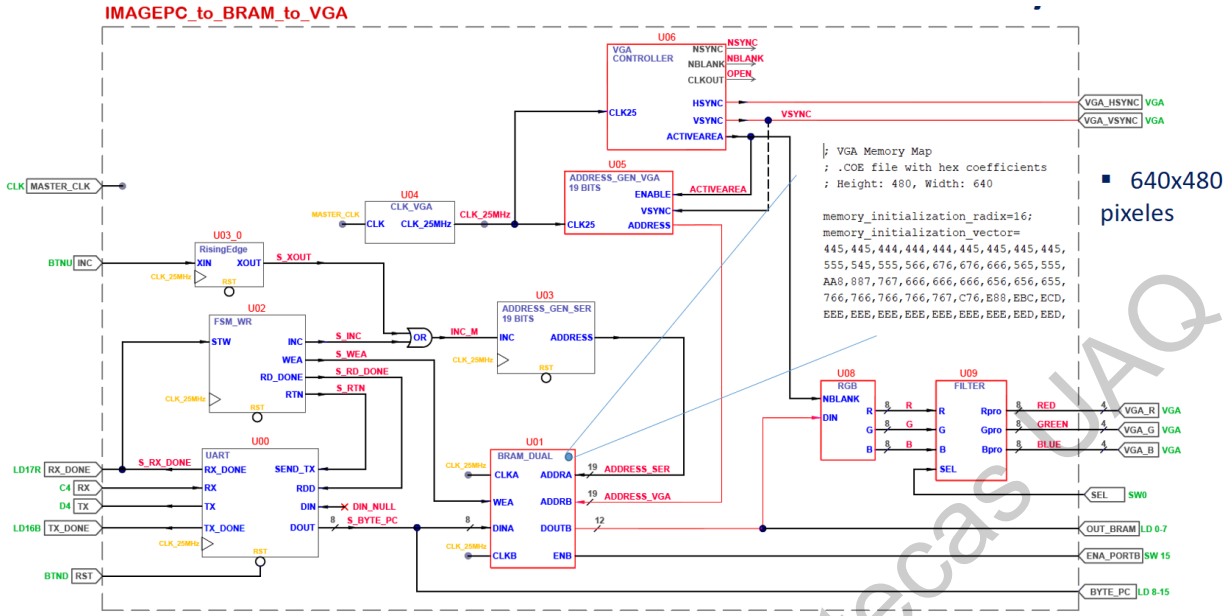


Figura 3.11: Arquitectura para la obtención del canal oscuro (imagen propia).

La arquitectura de la Figura 3.11 se describe de la siguiente manera: En la primera sección tenemos el controlador de comunicación serial (Bloque U00), el cual se comunica con una máquina de estados (Bloque U02) la cual se encarga de controlar la escritura y el puerto de direcciones, a su vez estos 2 bloque se encuentran en sincronía con un generador de direcciones (Bloque U03), esto con el objetivo de llenar el bloque de memoria *RAM* (Bloque U01).

El *software Vivado* cuenta con un catálogo de bloques funcionales predeterminados, mediante esta herramienta creó y se configuró un bloque de memoria *RAM* tipo Dual el cual tiene la particularidad de tener dos puertos de direcciones, en uno de ellos, podemos escribir la imagen, mientras que por el otro podemos leerla, esto de forma independiente y asíncronamente. Esta sección describe la comunicación entre la *PC* y el *FPGA* vía serial para poder trasmitir la imagen al *FPGA*, sin embargo, a su vez se hizo uso de una funcionalidad dentro de la configuración de los bloques de memoria *RAM* dentro del software, con la cual, es posible cargar una imagen con un archivo inicial con extensión tipo (.coe). Podemos observar un ejemplo de estos archivos creados en *Matlab* en la Figura 3.12. Este archivo fue creado desde el software *Matlab,*

en el cual se convirtió una imagen de tamaño de 640 x 480 pixeles. La tarjeta maneja 12 bits de color, esto quiere decir que tenemos 4 bits por cada canal.


```
; VGA Memory Map
; .COE file with hex coefficients
; Height: 480, Width: 640

memory_initialization_radix=16;
memory_initialization_vector=
445,445,444,444,444,445,445,445,445,444,445,445,445,445,445,455,455,455,555,555,555,555,555,555,555,555,555,555,655,655,555,555,
555,545,555,566,676,676,666,565,555,555,555,555,655,865,866,866,766,767,778,788,788,888,898,897,776,666,667,967,B89,BAB,BBC,BBA,
AA8,887,767,666,666,666,656,656,655,655,655,655,666,666,677,877,C87,C98,CA8,DA8,DA9,DA9,DA9,DA9,CA9,C98,888,787,766,766,766,766,
766,766,766,766,767,C76,E88,EBC,ECD,EDE,EDE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,
EEE,EEE,EEE,EEE,EEE,EEE,EEE,EED,EED,DDD,DCB,CBA,BBA,BBB,ABB,AAA,AA9,888,877,877,877,877,877,877,978,A89,B89,CAA,ECC,EEE,EEE,EEE,
EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,DEE,ACC,9A9,999,998,888,888,888,988,888,988,A99,BAB,DBB,
ECC,EDE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EED,EDC,ECB,EDA,DCA,DBA,DBB,EBC,ECD,EDD,EED,EED,EEE,EEE,EEE,EEE,
EEE,EEE,EEE,EED,CDC,BCB,9AA,A99,BAB,BBB,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBB,BBB,BBB,BBB,BBB,BBB,
BBB,BBB,BBB,BBB,BBB,BBB,BBB,BBB,BBB,BBB,BBB,BBB,BBB,BBB,CBB,CBB,BBC,BBB,9AA,89A,898,888,888,888,988,888,888,888,888,888,888,888,
888,989,99A,9AA,AAA,AAA,9AA,9AA,AAB,AAB,AAB,ABC,CBC,DCD,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,CDE,ABC,AAA,BAA,BBC,ACC,
9BB,AAA,AAB,AAB,BBC,BBC,BBC,CCC,CCE,CDE,DDE,DEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,
EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,
EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,
EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EDE,DDE,EEE,EEE,
EEE,EEE,EEE,EDD,DDD,DCC,CBB,BBB,BAA,BAA,BAA,BBA,CCB,DCC,EDC,EDD,DEE,CEE,CDD,BCB,AAA,AAA,AAA,AAA,BAA,BAA,BAA,AAA,AAA,AAA,AAA,AAA,
BAA,BAA,BAA,BAA,BAA,AAA,AAA,BAA,BAA,BAA,AAA,BAA,BA9,BA9,AAA,AAA,AAA,AAA,AAA,AA9,AA9,AA9,AA9,AA9,AA9,AA9,AA9,AA9,AAA,AAB,ABC,ABB,
BBA,BBA,AAB,AAA,AAA,AA9,AA9,AA9,AA9,A99,AA9,AA9,AA9,AA9,AA9,AA9,AA9,BA9,BA9,BAA,AA9,AA9,B99,BA9,BA9,BA9,CA9,BA9,AAA,9AA,AAA,AA9,
BA9,BA9,BAA,BA9,BA9,AA9,AA9,AA9,AA9,A99,A99,A99,A99,A99,A99,AA9,AA9,AA9,AA9,A99,AA9,A99,AA9,AA9,AA9,AA9,AA9,AA9,A99,A99,A99,A99,
999,999,A99,AA9,AA9,AA9,AA9,AA9,BA9,BA9,AA9,AA9,AA9,AA9,ABA,ACA,ABB,BAB,BAA,BAA,BBB,AAB,AAA,BBB,CCC,DCC,DBB,AAA,A99,A99,A99,A99,
A99,A99,A99,999,999,A99,A99,BA9,BA9,CAA,CA9,BA9,BAA,A9A,999,AA9,BA9,CAA,CAA,BA9,AA9,AA9,BAA,BAA,BAA,BAA,CAA,CAA,BAA,AA9,BA9,BB9,
545,544,544,544,544,445,445,445,445,444,444,445,445,455,455,455,445,445,445,545,555,555,555,555,555,555,555,555,555,555,555,555,
555,555,655,656,666,676,666,665,555,555,555,555,655,865,866,876,767,767,778,788,787,887,887,886,776,666,667,967,A89,BAB,DBC,ECB,
EB9,CA8,988,877,767,666,666,656,656,655,655,655,665,665,666,766,866,978,A98,BA8,BA9,DA9,DA9,DA9,CA9,CA9,9A9,798,787,767,667,666,
766,766,766,766,767,B76,E88,EBB,ECD,EDE,EDE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,
EEE,EEE,EEE,EEE,EEE,EEE,EEE,EED,EED,DED,DCB,CBA,BBA,BBB,ABB,AAA,A99,888,877,877,877,877,877,877,978,A88,B89,CAB,ECC,EEE,EEE,EEE,
EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,DED,ACC,9A9,999,899,888,888,888,888,888,988,A89,BAB,DBB,
ECC,EDE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EED,BCC,BBB,BAA,CBA,DBA,DBB,EBC,ECD,EDD,EDC,EEC,EEE,EEE,EEE,EEE,
EEE,EEE,EEE,EED,DDC,BCB,8AA,A99,BAB,BBB,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBC,BBB,BBB,BBB,BBB,BBB,BBB,
BBB,BBB,BBB,BBB,BBB,BBB,BBB,BBB,BBB,BBB,BBB,BBB,BBB,BBB,BBB,BBB,BBC,BBC,ABC,AAB,999,888,888,888,888,888,888,888,888,888,888,888,
888,989,999,99A,AAA,AAB,AAB,9AA,AAA,AAA,AAB,AAB,CBC,DCD,EDE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,DDE,BCC,BBB,AAA,ABC,ABC,
ABB,AAA,AAB,BAB,BBC,BBC,CCC,CCD,CCE,CDE,DDE,DEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,
EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,
EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,
EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,DDE,DDE,DEE,EEE,
EEE,EEE,EDE,EDD,DDC,CCB,BBB,BAB,BAB,BAB,BAA,BAA,CBA,CBA,DCB,DCD,DDE,CEE,CDD,BCB,BAA,BA9,AA9,AA9,BAA,BAA,BAA,AAA,AAA,AAA,AAA,BAA,
BAA,BAA,BA9,BAA,BA9,AAA,AAA,BAA,BAA,AAA,AAA,BAA,BAA,BAA,AA9,AA9,AA9,AA9,AAA,AAA,AAA,AAA,AAA,AAA,AA9,AA9,AA9,AA9,AA9,AAA,AAA,AAA,
BA9,BA9,AAA,AAA,AA9,A99,A99,AA9,A99,AA9,AA9,AA9,AA9,A99,A99,A99,AA9,BA9,BA9,BA9,AA9,AA9,BA9,BA9,BAA,BAA,CAA,BAA,AAA,9A9,AA9,AA9,
BA9,BA9,BA9,BA9,BA9,AA9,AA9,A99,A99,A99,A99,A99,A99,A99,A99,AA9,AA9,AA9,AA9,AA9,A99,A99,AA9,AA9,AA9,AA9,AA9,AA9,A99,A99,AA9,A99,
999,999,A99,AA9,AA9,AA9,AA9,BAA,BAA,BA9,BA9,AA9,AA9,AA9,AA9,AAA,BAA,BAA,BAA,CBB,BBC,BBC,BAA,BCB,CDD,DDE,CCD,AAC,A9A,A99,A99,A99,
A99,AAA,A99,A99,A99,A99,A99,BA9,BA9,BA9,BBA,BAA,AAA,AAA,AAA,BA9,BA9,CAA,CBA,CBA,BAA,BAA,CBA,CBA,DBA,DBA,DBA,CBA,AAA,AA9,BA9,CA9,
545,544,544,544,544,545,545,545,555,544,544,545,545,555,555,455,445,445,544,545,445,555,555,555,555,555,555,555,555,555,555,555,
```


Figura 3.12: Imagen de 640x480 convertida a una extensión tipo (.coe) mediante el *software Matlab* (imagen propia).

Seguido de lo anterior, en la sección final, se tiene el controlador *VGA*, el cual ya fue descrito en la sección 3.3.4. Mediante este controlador sincronizamos los datos de la memoria *RAM* con la imagen ya cargada y le ordena la obtención de la trama de datos de cada una de las direcciones en memoria, esto a una frecuencia de 25 MHz. Esta trama entra en un bloque llamado *RGB* (Bloque U08), el cual separa los canales y se manda al bloque para obtener el canal oscuro.

3.5. Selección del tamaño del parche

Un parámetro clave en el algoritmo de *K. He* es el tamaño del parche (He, Sun y Tang, 2010). El *DCP* mejora para un tamaño de parche más grande porque aumenta la probabilidad de que un parche contenga un píxel oscuro.

La arquitectura anterior obtiene solamente el mínimo de los 3 canales *RGB* por lo cual aquí se agrega un bloque para aplicar un parche de 3x3, tomar los 9 valores de los pixeles y nuevamente de estos obtener el valor mínimo.

En la Figura 3.13 se puede apreciar gráficamente el proceso de aplicación del parche: se tienes la matriz de pixeles de una imagen de tamaño 640x480, tenemos los valores de los 3 canales en cada pixel, 1 *byte* por color por lo que se tiene un rango de 0-255. Primero se selecciona el valor mínimo de entre los canales *RGB* en cada uno de los píxeles como se observa en la imagen central de la Figura 3.13 y después, se selecciona el menor de estos 9 valores y se asigna como valor central del parche.

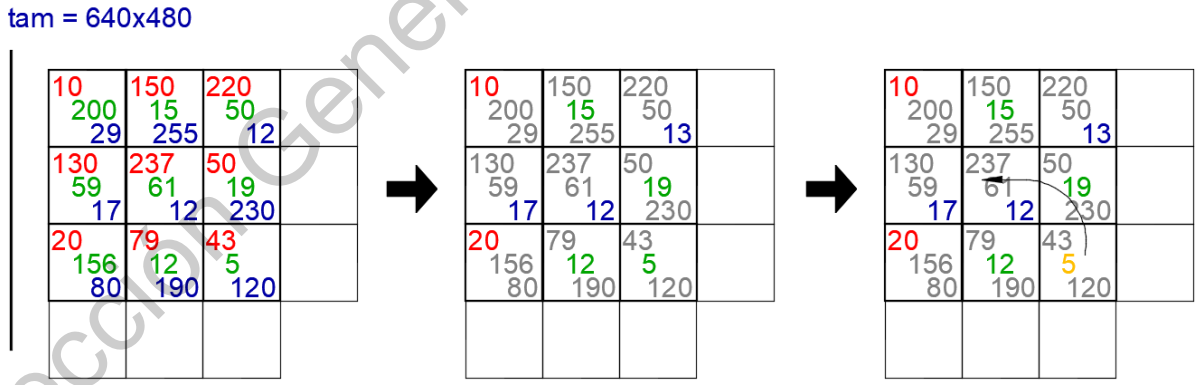


Figura 3.13: Aplicación del parche de tamaño 3x3 (imagen propia).

3.6. Arquitectura implementada Final

Tomando como base el algoritmo propuesto por *K. He* se llevó a cabo la implementación en el *FPGA*, sin embargo, se eliminó, uno de los 5 pasos, quedando solo 4, el que se refiere al *soft matting* se omitió por cuestiones de simplificar el algoritmo, sin embargo, este paso puede ser implementado en trabajos futuros.

Finalmente, la arquitectura de hardware para la obtención del canal oscuro con la aplicación del parche de tamaño 3x3 se muestra en la Figura 3.14. A grandes rasgos, es una arquitectura estructural con un controlador de *VGA*, un bloque de memoria *RAM* tipo dual el cual se creó con el catalogo *IP* de *Vivado*, un módulo que se encarga de procesar los 3 canales *RGB*, un bloque que recibe los valores *RGB* y los procesa para obtener el canal oscuro, un generador de direcciones se encarga de sincronizar el controlador de *VGA* y el acceso al bloque de *RAM*. Una vez que se obtiene el mínimo de los 3 canales, para aplicar un parche de 3x3 y obtener el *DCP* se llena otro bloque de memoria tipo dual (Bloque U08), mediante un comparador de 9 valores, que obtiene el mínimo de estos (Bloque U11), lo anterior, se controla mediante una máquina de estas (Bloque U13), la cual permite sincronizar el almacenamiento del bloque de memoria *RAM* (Bloque U08), a su vez se cuentan con generadores de direcciones los cuales también son controlados por la máquina de estados y son indispensables para la sincronización del llenado de parche de 3x3.

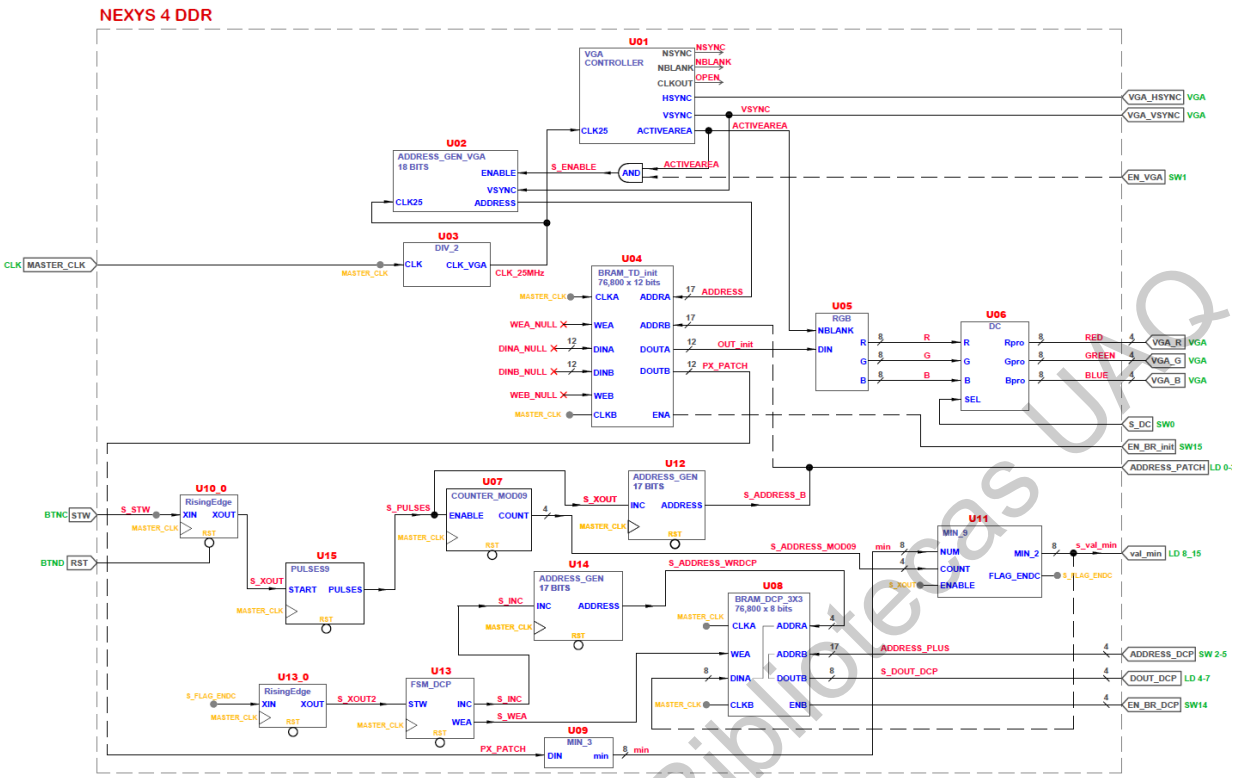
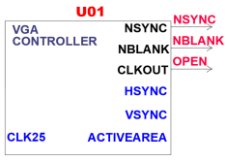
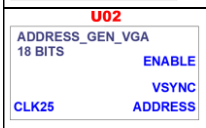
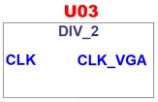
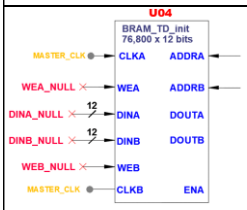
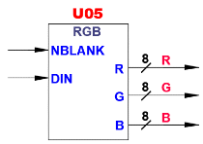
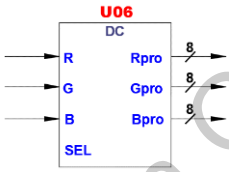
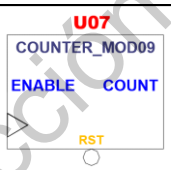
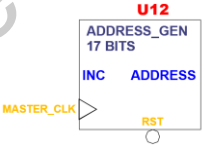
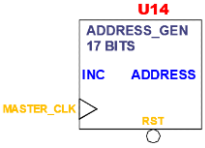
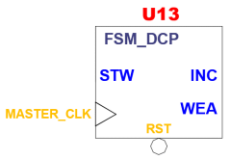
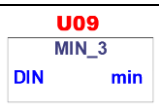
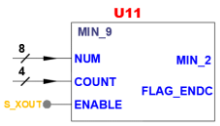
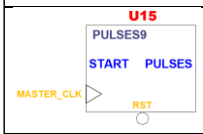
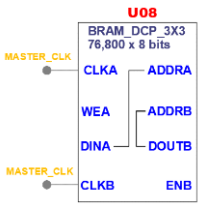


Figura 3.14: Obtención del Canal Oscuro con la aplicación del parche de tamaño 3x3 (imagen propia).

A continuación, en la Tabla 3.2 se muestra una biblioteca de cada uno de los componentes o bloques lógicos creados e implementados en *VHDL* para la obtención del canal oscuro con un parche de tamaño 3x3 y se describe la funcionalidad.

Tabla 3.2: Biblioteca de bloques funcionales creados e implementados en la arquitectura del algoritmo de *dehazing* y su descripción (creación propia).

Bloques Lógicos	
Diagrama	**Descripción**
	El controlador VGA funciona a una frecuencia de 25MHz la cual es proporcionada por el bloque U03. El bit ActiveArea sincroniza las direcciones de memoria del bloque U04 y los bits Hsync y Vsync sincronizan la señal en el mapa de pixeles del monitor VGA.
	Es un generador de direcciones el cual permite genera las direcciones para accesar al bloque de memoria RAM U04. Las direcciones de memoria tienen un tamaño de 18 bits correspondiente a la representación del número máximo de pixeles en una imagen de 320x240, 76,800 bits en total.
	Es un divisor de frecuencia que utiliza el principio de negar la señal de reloj (en este caso se utilizó el reloj maestro del FPGA de 100 MHz) en uno de sus niveles para dividir la frecuencia por 2, esto se realizó una vez más para obtener la frecuencia de 25 MHz.
	Es un bloque de memoria RAM tipo Dual. Es de un tamaño en específico 76,800 por 12 bits de profundidad, pero cuenta con 2 puertos para accesar a sus direcciones al mismo tiempo, un puerto se utiliza para escribir, mientras que el otro se utiliza para leerlo y al mismo tiempo mediante la señal de salida DOUTA se conecta al bloque RGB (U05) para posteriormente ser enviado a los canales RGB del VGA.
	Es un bloque que recibe en la entrada DIN la trama de 12 bits de cada una de las direcciones de la memoria del bloque U04 y se encarga de separar la trama en sus 3 canales R, G y B para posteriormente enviarles al bloque U06.
	Este bloque recibe los 3 canales RGB individuales y los procesa para obtener de ellos el canal con el valor mínimo lo que corresponde a la función de obtención del canal oscuro.
	Es un contador de modulo 9 el cual una vez se habilita mediante el bit ENABLE comienza la cuenta de 0 a 9. Una vez alcanzada la cuenta se reinicia en 0.
	Es un generador de direcciones el cual permite genera las direcciones para extraer los datos de la imagen precargada del bloque de memoria RAM U04, las cuales son enviadas al Bloque U09 para obtener el canal mínimo.
	Es un generador de direcciones el cual permite genera las direcciones para accesar al bloque de memoria RAM U08. Las direcciones de memoria tienen un tamaño de 18 bits correspondiente a la representación del número máximo de pixeles en una imagen de 320x240, 76,800 bits en total.

Bloques Lógicos	
Diagrama	**Descripción**
	Es la máquina de estados encargada del control de almacenamiento del nuevo mapa de bits generado para el DCP con parche de tamaño de 3x3 (Bloque U08). Sincroniza las direcciones de memoria de U08 y ordena los valores de escritura provenientes de U11.
	Este bloque recibe el píxel original y lo procesa para obtener el canal RGB mínimo, recibe una trama de datos de tipo [RRRRGGGGBBBB] y despues de evaluar entrega una trama de datos [MMMM0000] en donde M pertenece a (R,G,B)
	Este bloque recibe uno por uno, cada uno de los mínimos de los canales RGB, y con ello, aplica el parche de 3x3, procesando de estos 9 valores el valor mínimo. Al final del proceso envía este valor mínimo a la memoria del bloque U08 que es donde se almacenan los valores del DCP.
	Envía 9 pulsos para controlar el contador del bloque U07.
	Es un bloque de memoria RAM tipo Dual. Es de un tamaño en específico 76,800 por 8 bits de profundidad. En este bloque se va almacenando el mapa de bits de la imagen del DCP con parche de 3x3, mediante el control de la maquina de estados FSM_DCP del bloque U13.

En la Figura 3.15, se puede observar la arquitectura final para la recuperación de la radiación de la imagen, la cual puede ser implementada en un *FPGA,* el cual posea una memoria RAM estática para su fácil acceso. La diferencia entre la arquitectura de la Figura 3.14 y está, se puede observar en la parte inferior separada por la línea punteada, se agregaron los bloques lógicos del U15 al U24.

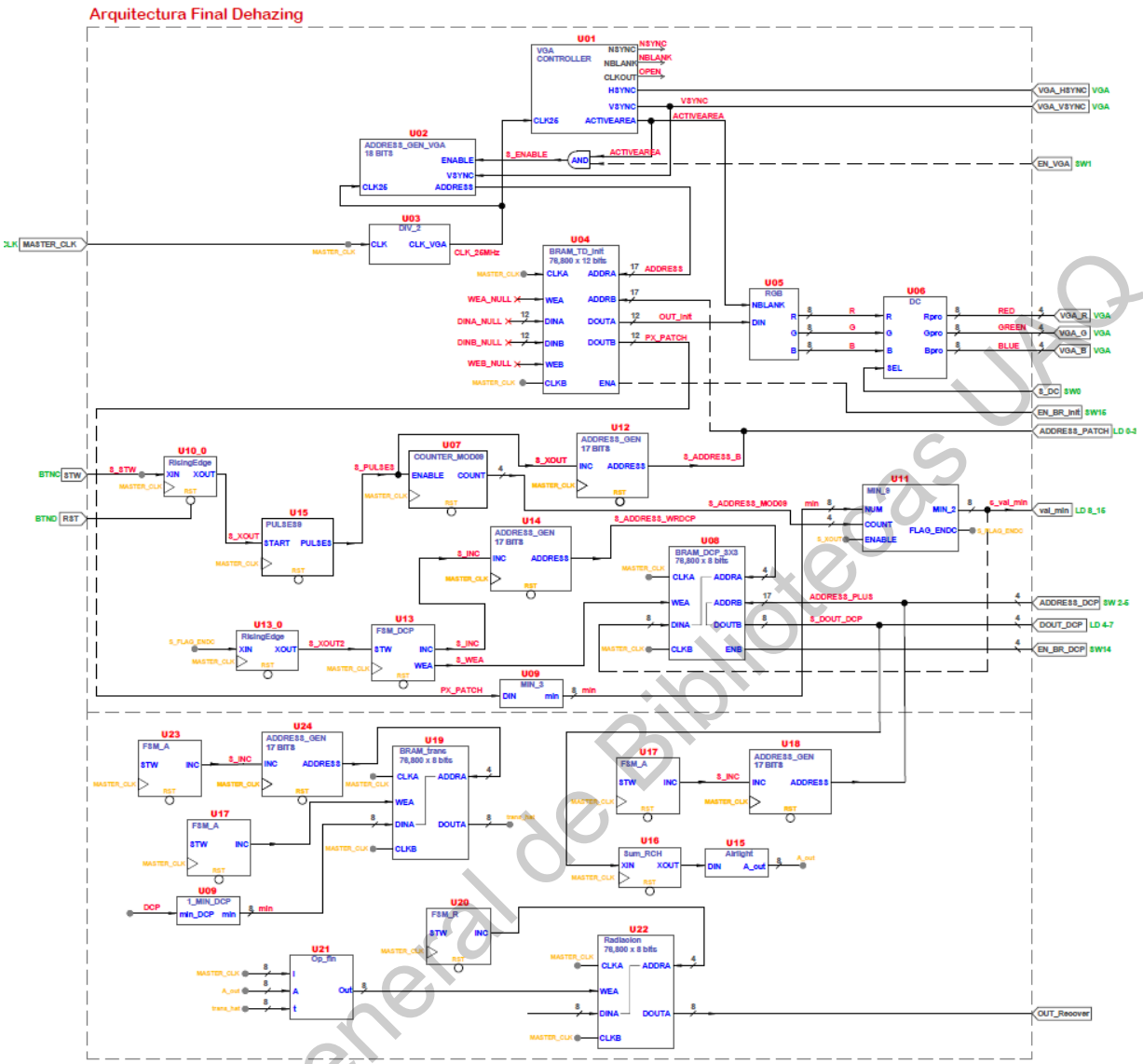


Figura 3.15: Arquitectura Final del Algoritmo de *Dehazing* para implementar en *FPGA* (imagen propia).

Resultados y conclusiones

4.1 Resultados Etapa 1: Pre procesamiento de las imágenes

En la Figura 4.1 se muestran algunos ejemplos de los resultados obtenidos al densificar los mapas dispersos de las imágenes. Como entradas del proceso en la primera columna de izquierda a derecha tenemos las imágenes *RGB*, en la segunda columna sus respectivos mapas de profundidad dispersos, y en la tercera columna se muestran los resultados generados por la función: *fill_depth_colorization*.

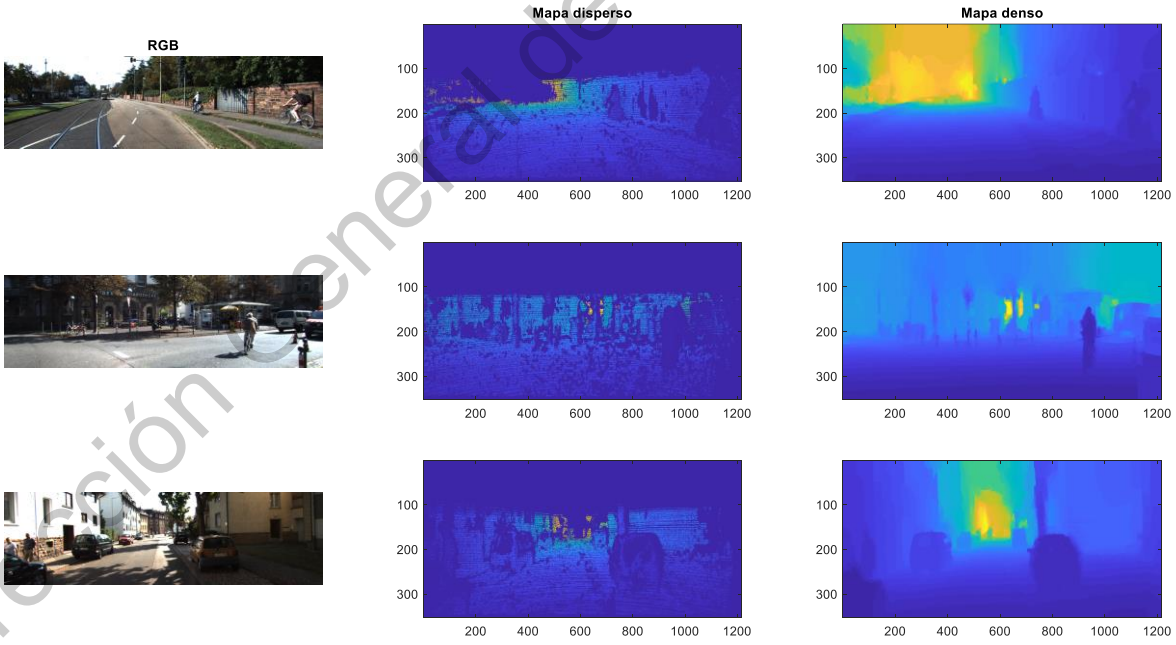


Figura 4.1: Mapas de profundidad densos creados en *Matlab* (creación propia).

Este pre procesamiento de las imágenes se llevó a cabo en el *software Matlab*, versión *R2017b* en un sistema con las siguientes características:

- Procesador: *Intel Core i5- 8250U a 1.6GHz*
- Memoria *RAM*: 8.0 GB

Una vez obtenidos los mapas de profundidad densos se pudo proceder a la obtención de los mapas de transmisión mediante el *software Matlab* (ver Figura 4.2).

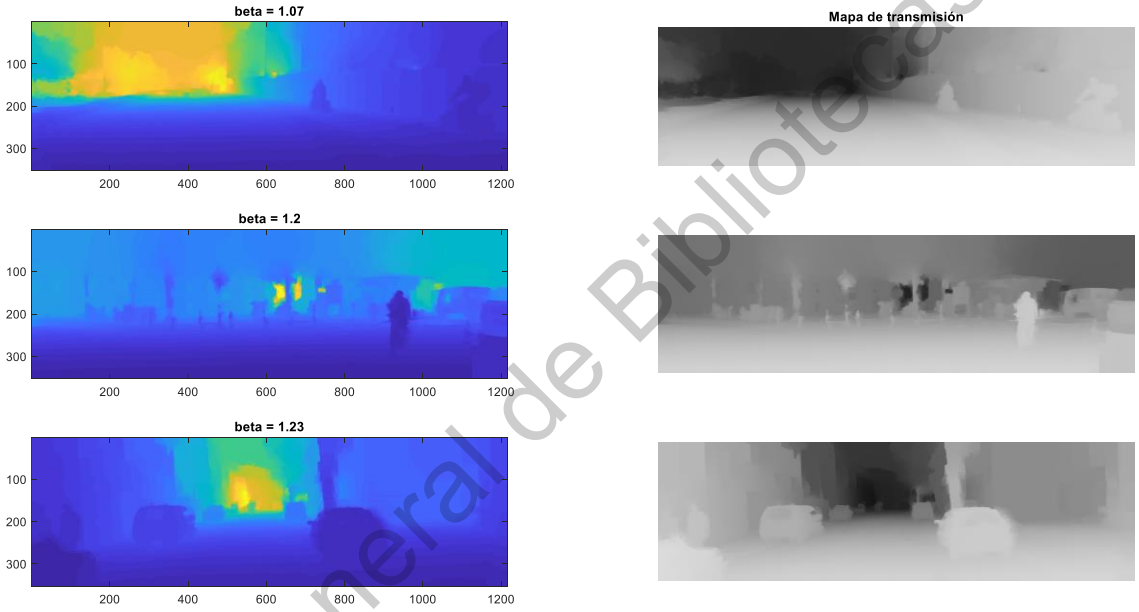


Figura 4.2: Mapas de transmisión creados en el *software Matlab* (creación propia).

En la Figura 4.2, se pueden apreciar en la columna de la izquierda los datos de entrada, los cuales fueron: los coeficientes de dispersión atmosférica (β), los cuales fueron valores aleatorios, y los mapas de profundidad densos obtenidos previamente en el paso anterior. En la columna derecha se observan los mapas de transmisión obtenidos.

Finalmente, en la Figura 4.3 se muestran a la izquierda, las imágenes de entrada (imágenes sin *haze)* y a la derecha, los resultados que generó la metodología descrita

en el apartado 3.2 (imágenes con *haze* sintético), las cuales se utilizaron posteriormente a esto para la evaluación del desempeño de la arquitectura para realizar *dehazing* en el *FPGA*.

Figura 4.3: Imágenes con *haze* sintético generadas con el *software Matlab* (creación propia).

En la tabla 4.1 se muestran los parámetros utilizados en la generación de las imágenes con *haze* sintético, la luz ambiental o *Airligth* (*A*) y el coeficiente de dispersión atmosférica (β).

Tabla 4.1: Parámetros para la creación de las imágenes con *haze* sintético (creación propia).

Imagen	**A**	β
1	1	1.07
2	0.76	1.2
3	0.99	1.23

4.2 Resultados Etapa 2: Arquitectura general para PDI

Para mostrar el desempeño de la arquitectura general (implementada en la *Nexys 4 DDR*) para procesamiento de imágenes se aplicaron 2 filtros; binarización y escala de grises.

Para obtener la binarización de la imagen se creó un bloque el cual realiza su función mediante segmentación por umbral. La segmentación por umbral es un método en la cual los pixeles toman solo uno de dos valores posibles, 0 si el valor del pixel es menor al umbral establecido y 1 si el valor del pixel es mayor o igual al umbral.

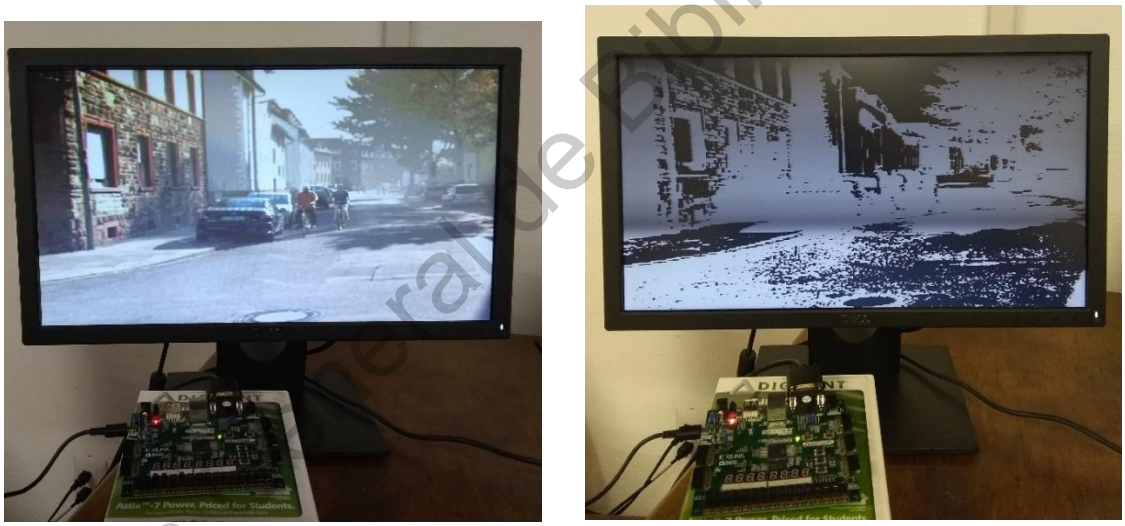


Figura 4.4: Binarización (creación propia).

Para obtener la escala de grises, se obtuvieron los valores de cada canal *RGB* para cada uno de los pixeles, luego se realiza una sumatoria entre los canales para sacar un promedio de estos, y por último a la salida se asigna el valor promediado en cada canal *RGB*, es decir, se repite para cada uno de los canales *RGB*.

Figura 4.5: Escala de grises (creación propia).

4.3 Resultados Etapa 3: Arquitectura para realizar *Dehazing* en el *FPGA*

Si se aplica un parche de tamaño n = 1, la ecuación 2.3 se reduce a encontrar solamente el valor mínimo sobre los 3 canales *RGB*, con lo cual se obtiene el canal oscuro. El canal oscuro es la representación más básica del *DCP*.

Los primeros experimentos dentro de en este trabajo se centraron en la obtención del Canal Oscuro, posteriormente, se pretende aplicar parches de diferentes tamaños, n=3,5,7,9 en futuros trabajos.

Figura 4.6: Canal Oscuro obtenido del *FPGA*, tamaño 1 (creación propia).

En la Figura 4.6 a la izquierda se muestran las imágenes con *haze* y a la derecha se muestra en canal oscuro de las imágenes con un parche de tamaño 1.

Figura 4.7: Canal Oscuro obtenido del *FPGA*, tamaño 3x3 (creación propia).

En la Figura 4.7 se muestran el *DCP*, el canal oscuro de una imagen con un parche de tamaño 3x3.

4.4 Conclusiones

Una de las grandes ventajas que se tienen en este proyecto, es el haber creado (a partir de la *DB: Kitti* (Karlsruhe Institute of Technology, 2020)) la base de datos de las imágenes con *haze* sintético. Ya que con esto se tiene un *ground truth* y esto dará la pauta para la evaluación final del desempeño del sistema embebido.

En muchos de los trabajos de *dehazing* las métricas comúnmente utilizadas son: la del Error Cuadrático Medio o *Mean Squared Error* (*MSE*, por sus siglas en inglés), la Proporción Máxima de Señal a Ruido o *Peak Signal to Noise Ratio* (*PSNR*, por sus siglas en inglés) y en algunos casos con el Índice de Similitud Estructural o *Structural Similiarity Index* (*SSIM*, por sus siglas en inglés), sin embargo, muchas veces las evaluaciones y comparaciones entre los algoritmos se realizan con la descripción de una Figura en la cual se muestran los resultados de unas cuantas imágenes de cada uno de los algoritmos comparados, con ambigüedades tales como: se obtuvieron resultados visualmente mejores en comparación a los otros algoritmos, se recuperaron colores más fieles a la realidad, entre otros.

Además, por si fuera poco, las imágenes suelen ser las mismas, obvio esta, que es un compendió de imágenes que se ha transmitido de artículo en artículo para poder demostrar la superioridad de un algoritmo con respecto al anterior. Esto hace que, no se están evaluando los algoritmos para una variedad más amplia de situaciones y escenarios, los cuales claro, está son una infinidad en la realidad.

Esto es comprensible, dado a que en la gran mayoría de trabajos se trabaja con las imágenes afectadas por el efecto del *haze* real, se desconoce la imagen libre de *haze* y siempre se desconocerá, sin embargo, si se trabaja con imágenes sin *haze,* como es el caso de este trabajo, y se le aplica después el *haze* sintético, teniendo como *ground* truth las imágenes originales sin *haze*, al final podremos comparar con exactitud los resultados producidos por el algoritmo.

A pesar de que se trabajó con una interfaz de memoria para controlar un tipo de

memoria *RAM* dinámica, el control de esta fue muy inestable. Para evitar esto es recomendable trabajar con memorias *RAM* estáticas.

4.5 Trabajo Futuro

Dadas las limitaciones de memoria ya mencionadas, el algoritmo se completó hasta la obtención del *DCP*, los pasos posteriores requieren más memoria para almacenar las matrices de las operaciones.

Sin embargo, la arquitectura es totalmente migrable dado que los bloques lógicos creados en *VHDL* en este proyecto así lo permiten, a excepción del bloque de memoria *RAM* tipo Dual que fue tomado de la propiedad intelectual de *Xilinx* y su *software Vivado*, por lo que, para un trabajo futuro en el cual se complemente el algoritmo con los pasos restantes, se propone utilizar una tarjeta con memoria *RAM* estática, dadas sus características de fácil accesibilidad y manejo, una de estas tarjetas puede ser la *DE2-115* de la marca *Altera*.

Cabe mencionar que dentro de los experimentos se utilizó una cámara, se obtuvieron y se procesaron las imágenes adquiridas mediante esta. A estas imágenes se les aplicó los filtros de binarización y escala de grises creados para la evaluación del desempeño de la arquitectura general para PDI. Los procesamientos mostraron buen desempeño en Tiempo Real en este tipo de filtros, por lo cual se propone probar el algoritmo de *dehazing* en tiempo real con la arquitectura creada para futuros trabajos.

Bibliografía

Aceves, M. A. y Arreguin, J. M. R. (2011), *Fundamentos de Sistemas embebidos - Mediante lenguajes descriptivos de hardware*, Querétaro, Qro., México: Asociación Mexicana de Mecatrónica A.C.

Anat Levin (s.f.). https://www.cse.huji.ac.il/~yweiss/Colorization/ Accedido: 21-01-2020.

Bai, L., Wu, Y., Xie, J., & Wen, P. (2015). Real time image haze removal on multi-core dsp. Procedia Engineering, 99, 244-252.

Digilent, A National Instruments Company (2019). Nexys 2. Digilent Documentation. https://reference.digilentinc.com/reference/programmable-logic/nexys-2/start. Accedido: 14-01-2020.

Floyd, T. L., & Caño, J. G. (1997). *Fundamentos de sistemas digitales*. Prentice Hall.

Forsyth D., and Ponce J. (2011), *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference. 2nd edition.

Gases del Aire Composición y Propiedades. (s.f.). Recuperado de https://www.areaciencias.com/quimica/componentes-del-aire.html Accedido: 10-01-2020.

González, R. C., & Woods, R. E. (1996). *Tratamiento digital de imágenes* (Vol. 3). New York: Addison-Wesley.

He, K., Sun, J., & Tang, X. (2010). Single image haze removal using dark channel

prior. *IEEE transactions on pattern analysis and machine intelligence, 33*(12), 2341-2353.

He, R., Wang, Z., Xiong, H., & Feng, D. D. (2012, December). Single image dehazing with white balance correction and image decomposition. In 2012 *International Conference on Digital Image Computing Techniques and Applications (DICTA)* (pp. 1-7). IEEE.

Karlsruhe Institute of Technology, The KITTI Vision Benchmark Suite. (2020). Recuperado de: http://www.cvlibs.net/datasets/kitti/ Accedido: 17-01-2020.

Khoury, J. (2016). *Model and quality assessment of single image dehazing* (Doctoral dissertation).

Kim, J. Y., & Jeon, J. W. (2018, January). Implementation of a Single-Image Haze Removal Using the FPGA. In Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication (pp. 1-7).

La Atmósfera. (s.f.). Recuperado de https://lageografia.com/geografia-fisica/atmosfera Accedido: 10-01-2020.

Land, E. H., & McCann, J. J. (1971). Lightness and retinex theory. *Josa, 61*(1), 1-11.

Nathan Silberman (s.f.). https://cs.nyu.edu/~silberman/about.html Accedido: 21-02-2020

Nexys 4 DDR Reference Manual [Reference.Digilentinc]. (s.f.). Recuperado de: https://reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr/reference-manual Accedido: 03-02-2020.

Saggu, M. K., & Singh, S. (2015). A review on various haze removal techniques

for image processing. International journal of current engineering and technology, 5(3), 1500-1505.

Salazar-Colores, S. (2019). *Un Algoritmo para Corregir el Problema del Hazing en Procesamiento de Imágenes* (tesis doctoral).

Salazar-Colores, S., Ramos-Arreguin, J. M., Echeverri, C. J. O., Cabal-Yepez, E., Pedraza-Ortega, J. C., & Rodriguez-Resendiz, J. (2018). Image dehazing using morphological opening, dilation and Gaussian filtering. *Signal, Image and Video Processing, 12*(7), 1329-1335.

Schechner, Y. Y., Narasimhan, S. G., & Nayar, S. K. (2001, December). Instant dehazing of images using polarization. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001 (Vol. 1, pp. I-I). IEEE.

Silva, G. E. B., Resendiz, J. R., Hurtado, E. G., Ortega, J. C. P., & Arreguin, J. M. R. (2015). Didactic platform for image processing experiments based on digital design. *IEEE Latin America Transactions*, 13(10), 3398-3404.

Snyder, W. E., & Qi, H. (2017). *Fundamentals of Computer Vision*. Cambridge University Press.

Tan, R. T. (2008, June). Visibility in bad weather from a single image. In 2008 *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1-8). IEEE.

Tarel, J. P., & Hautiere, N. (2009, September). Fast visibility restoration from a single color or gray level image. In 2009 *IEEE 12th International Conference on Computer Vision* (pp. 2201-2208). IEEE.

Troncoso, R. D. F. R. (2007). *Electrónica digital y lógica programable*.

Universidad de Guanajuato.

Zhang, B., & Zhao, J. (2016). Hardware implementation for real-time haze removal. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 25*(3), 1188-1192.

Zhang, H., Liu, Q., Yang, F., & Wu, Y. (2013). Single image dehazing combining physics model based and non-physics model based methods. *Journal of Computational Information Systems*, 9(4), 16231631.

Anexos

UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE INGENIERÍA
DIPFI POSGRADO INGENIERÍA
13° COLOQUIO

Se otorga la presente

CONSTANCIA a:

MIGUEL ÁNGEL MONCADA MALAGÓN

Por su participación como **Asistente** y **Presentador Oral** en el evento:

13° Coloquio de Posgrado de la Facultad de Ingenería

20, 21 y 22 de Noviembre de 2019
Facultad de Ingeniería

Dr. Manuel Toledano Ayala
DIRECTOR
FACULTAD DE INGENIERÍA

Dr. Juan Carlos Jáuregui Correa
Jefe de la División de Investigación y Posgrado
FACULTAD DE INGENIERÍA

Santiago de Querétaro, Querétaro, 8 de mayo de 2020

C. Miguel Ángel Moncada Malagón

PRESENTE

Me permito informarle que el comité evaluador de la revista "Perspectivas de la Ciencia y la Ingeniería" ha decidido **aceptar** el artículo detallado a continuación:

Título:
Diseño de una Arquitectura en Hardware para la obtención del Canal Oscuro en Imágenes con Haze

Código de registro:
97

Autores:
Miguel Ángel Moncada Malagón, José Eduardo Gaspar Badillo, Juan Manuel Ramos Arreguín, Jesús Carlos Pedraza Ortega y Marco Antonio Aceves Fernández

Para su **publicación** en la 5° Edición de esta revista, misma que estará disponible en el sitio: http://perspectivasciencia.uaq.mx/index.php/ojs, después del proceso de corrección de estilo. Le reiteramos nuestro agradecimiento por su aporte y esperamos seguir contando con su participación en ediciones posteriores.

Atentamente

M.C. Alepth Hain Pacheco Estrada
Co-editor de la revista Perspectivas de la Ciencia y la Ingeniería, 5° Edición