UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

Profesor Patrocinante:
Dr. Miguel Figueroa Toro

Informe de tesis para optar al grado de:
Magíster en Ciencias de la Ingeniería con mención en Ingeniería Eléctrica

REGISTRO MULTIMODAL DE IMÁGENES VIS, SWIR Y LWIR EN HARDWARE DEDICADO

Concepción, Agosto de 2018

Javier Alfredo Cárdenas Drews

Universidad de Concepción
Facultad de Ingeniería
Departamento de Ingeniería Eléctrica

Profesor Patrocinante:
Dr. Miguel Figueroa Toro

REGISTRO MULTIMODAL DE IMÁGENES VIS, SWIR Y LWIR EN HARDWARE DEDICADO

Javier Alfredo Cárdenas Drews

Informe de tesis para optar al grado de:
"Magíster en Ciencias de la Ingeniería con mención en Ingeniería Eléctrica"

Comisión:
Dr. Jorge Pezoa Nuñez
Dr. Gonzalo Carvajal Barrera

Concepción, Chile, Agosto de 2018

Resumen

El registro de imágenes se define como el proceso de alinear imágenes de una misma escena capturadas bajo diferentes condiciones. Cuando estas imágenes corresponden a distintos espectros se denomina registro multimodal. El proceso de registro se basa en encontrar una transformación geométrica que permita proyectar la imagen objetivo sobre el espacio de coordenadas de la imagen de referencia, obteniendo una correspondencia punto a punto entre ellas. El registro consta de cuatro componentes: espacio de características, medida de similitud, espacio de búsqueda, y estrategia de búsqueda. El registro multimodal permite apreciar información complementaria sobre una escena, habilitando la utilización de técnicas de análisis de imágenes más potentes y robustas, pero debido a la complejidad de los algoritmos este alineamiento se realiza típicamente en computadores de propósito general, lo que conlleva un alto uso de energía, tiempo, y espacio en comparación a los dispositivos usados para la adquisición de imágenes.

En este trabajo se diseñó un algoritmo de registro multimodal entre imágenes *visible* (Vis), *infrarrojo de onda corta* (SWIR) e *infrarrojo de onda larga* (LWIR) compuesto de una fase de calibración inicial y un proceso de registro cuadro a cuadro. El algoritmo utiliza una versión modificada de *histograma de gradientes orientados* (HOG) como extractor de características, la medida de similitud Chi-cuadrado para emparejarlas, y transformación proyectiva en conjunto con interpolación bilineal para registrar las imágenes. Se diseñó una arquitectura heterogénea software/hardware del algoritmo propuesto, ejecutando la fase de calibración en el núcleo software y el proceso de registro cuadro a cuadro en el núcleo hardware, con un montaje experimental propuesto que integra una cámara Vis, SWIR y LWIR, cada una con su propio núcleo de procesamiento, registrando las imágenes Vis-SWIR y SWIR-LWIR a medida que son obtenidas por los distintos dispositivos de adquisición.

El sistema fue implementado en un *sistema-en-un-chip* (SoC) Zynq XC7Z020 de Xilinx. El prototipo ejecuta de forma separada la fase de calibración inicial y el proceso de registro de imágenes cuadro a cuadro en el procesador embebido y lógica programable respectivamente. La fase de calibración incluye la extracción y emparejamiento de características en conjunto con el cálculo de los parámetros de la transformación. El proceso de registro cuadro a cuadro aplica la transformación proyectiva e interpolación a cada cuadro de la imagen objetivo. El núcleo hardware puede operar a más de 60fps en imágenes de 640×512 pixeles a una frecuencia de reloj de 66.6MHz, utilizando el 20 % de los recursos lógicos y cerca del 55 % de los recursos de memoria del SoC. El sistema consume 1.888W de potencia, de los cuales 1.541W corresponden a la potencia consumida por el procesador y sólo 0.180W al núcleo de registro implementado en hardware.

Durante las diferentes etapas de este trabajo he recibido el apoyo financiero del Programa de Becas para Estudios de Magíster en Chile de la agencia CONICYT del gobierno de Chile.

Tabla de contenidos

Índice de figuras

Índice de tablas

Acrónimos

ASIC	*Circuito integrado de aplicación específica. Del inglés "application-specific integrated circuit"*
CC	*Coeficiente de correlación. Del inglés "correlation coefficient"*
CCD	*Dispositivo de carga acoplada. Del inglés "charge-coupled device"*
CMOS	*Semiconductor complementario de óxido metálico. Del inglés "complementary metal oxide semiconductor"*
CT	*Tomografía computarizada. Del inglés "computed tomography"*
DoG	*Diferencia de Gaussiana. Del inglés "difference of Gaussian"*
DSP	*Procesador digital de señales. Del inglés "digital signal processor"*
FFD	*Deformaciones de forma libre. Del inglés "free-form deformations"*
FPA	*Arreglo de plano focal. Del inglés "focal plane array"*
FPGA	*Arreglo de compuertas programables. Del inglés "field programmable gate array"*
HOG	*Histograma de gradientes orientados. Del inglés "histogram of oriented gradients"*
InGaAs	*Arseniuro de indio y galio. Del inglés "indium-gallium-arsenide"*
IR	*Infrarrojo. Del inglés "infrared"*
IRFPA	*Arreglo de plano focal infrarrojo. Del inglés "infrared focal plane array"*
L-BFGS	*BFGS de memoria limitada. Del inglés "limited memory BFGS"*
LVDS	*Señal diferencial de bajo voltaje. Del inglés "low-voltage differential signaling"*
LWIR	*Infrarrojo de onda larga. Del inglés "long wave infrared"*
MI	*Información mutua. Del inglés "mutual information"*
MR	*Resonancia magnética. Del inglés "magnetic resonance"*
MRF	*Campos aleatorios de Markov. Del inglés "markov random field"*
MWIR	*Infrarrojo de onda media. Del inglés "mid wave infrared"*
NEDT	*Temperatura diferencial equivalente al ruido. Del inglés "noise equivalent differential temperature"*
NGF	*Campo de gradientes normalizado. Del inglés "normalized gradient fields"*
NIR	*Infrarrojo cercano. Del inglés "near infrared"*

NMI	*Información mutua normalizada. Del inglés "normalized mutual information"*
NU	*No uniformidad. Del inglés "non-uniformity"*
OF	*Flujo óptico. Del inglés "optical flow"*
PET	*Tomografía de emisión de positrones. Del inglés "positron emission tomography"*
PF	*Filtrado de partículas. Del inglés "particle filtering"*
POI	*Puntos de interés. Del inglés "points of interest"*
RANSAC	*Consenso de muestra aleatoria. Del inglés "random sample consensus"*
RGB	*Rojo-verde-azul. Del inglés "red-green-blue"*
R-IR	*Infrarrojo reflectivo. Del inglés "reflective infrared"*
ROI	*Región de interés. Del inglés "region of interest"*
SIFT	*Transformación de característica invariante a la escala. Del inglés "scale-invariant feature transform"*
SoC	*Sistema-en-un-chip. Del inglés "system-on-a-chip"*
SR	*Superresolución. Del inglés "super-resolution"*
SSD	*Suma de diferencias al cuadrado. Del inglés "sum of squared differences"*
SURF	*Características robustas aceleradas. Del inglés "speeded-up robust features"*
SWIR	*Infrarrojo de onda corta. Del inglés "short wave infrared"*
T-IR	*Infrarrojo térmico. Del inglés "thermal infrared"*
VOx	*Óxido de vanadio. Del inglés "vanadium oxide"*
Vis	*Visible. Del inglés "visible"*

Capítulo 1: Introducción

1.1. Introducción general

El registro de imágenes puede ser definido como el proceso de establecer una relación punto a punto entre dos imágenes, con lo cual una imagen objetivo es mapeada al sistema de coordenadas de una imagen de referencia. Este proceso se realiza para generar una imagen con información coherente entre los varios dispositivos de captura, espacios de tiempo, perspectivas o detectores, siendo utilizada por ejemplo para procesos de visión por computadora, imagenología médica [1, 2, 3, 4, 5, 6], formación de imágenes panorámicas, y alineamiento de imágenes satelitales [7]. El registro de imágenes usualmente tiene cuatro componentes [8]: espacio de características, medida de similitud, espacio de búsqueda, y estrategia de búsqueda. Un algoritmo de registro de imágenes busca una transformación geométrica para alinear ambas imágenes al emparejar *puntos de interés* (POI) usando una medida de similitud. La selección de estos componentes está fuertemente ligada a la aplicación en específico, ya que normalmente no hay conjunto que produzca resultados óptimos en todos los casos [9].

Los algoritmos de registro de imágenes son típicamente clasificados de acuerdo a los tipos de imágenes con las cuales operan y la información utilizada para el alineamiento, dividiéndose en procesos basados en la intensidad de los pixeles y procesos basados en características extraídas de las imágenes. El proceso de registro con imágenes que se encuentran en el mismo rango del espectro electromagnético se denomina registro monomodal, mientras que alinear imágenes provenientes de distintas longitudes de onda se denomina registro multimodal. Las imágenes multimodales son usualmente adquiridas por detectores que operan en base a principios físicos diferentes, lo que en la mayoría de los casos implica que las imágenes son adquiridas por diferentes cámaras simultáneamente, y consecuentemente el registro de imágenes es necesario para producir una imagen coherente entre las diferentes perspectivas de las cámaras.

Registrar imágenes que provienen de diferentes longitudes de onda permite obtener más información sobre la escena que usando una imagen monomodal, obteniendo características complementarias de los objetos observados en ella. Esta información adicional puede permitir técnicas más poderosas y robustas en el análisis de imágenes. Aplicaciones de análisis de imágenes multimodales incluyen reconocimiento de rostros [10, 11], emparejamiento espectral de rostros [12, 13], identificación de grietas estructurales [14], seguimiento de peatones [15, 16, 17], y detección de cáncer de piel [18, 19].

La porción del espectro electromagnético que puede ser visualizado por el ojo humano se denomina espectro *visible* (Vis) y se encuentra entre los 0.4µm y 0.7µm de longitud de onda. En este espectro se pueden percibir características tales como la forma, textura y color de los objetos observados. Aplicaciones en este espectro incluyen fotografía, medición espectral atmosférica [20], odontología [21] y reconocimiento de rostros [22, 23, 24, 25]. Asimismo, existe información que es apreciable de forma parcial o exclusiva en otros espectros, como en el *infrarrojo* (IR) que se encuentra entre los 0.7µm y 1,000µm.

Particularmente interesante son las imágenes adquiridas en los espectros *infrarrojo de onda corta* (SWIR) e *infrarrojo de onda larga* (LWIR), usualmente denominados *infrarrojo reflectivo* (R-IR) e *infrarrojo térmico* (T-IR) respectivamente. El espectro R-IR revela muchas características que son invisibles en el espectro Vis, tal como el agua entre los 1,450nm y 1,900nm de longitud de onda, o el vidrio en SWIR [26]. R-IR también nos permite ver a través de algunos materiales que son opacos en Vis, como el silicio sobre los 1,100nm [27] o petróleo [28], y es más robusto ante interferencia de niebla o humo. SWIR es también más sensible a la iluminación, lo cual nos permite trabajar en condiciones de escasa iluminación [29], pero el detector puede saturarse fácilmente con fuentes de iluminación potentes como el sol. Aplicaciones para imágenes SWIR incluyen detección de piel humana para reconocimiento de rostros [30] o misiones de rescate [31], identificación de materiales y análisis de su distribución espacial en pinturas de bellas artes [32, 33]. Muchos de los métodos usados en procesamiento de imágenes en el rango visible también pueden ser utilizados en R-IR [26].

T-IR permite estimar la temperatura de objetos y su distribución espacial en la escena. Imágenes de rostros en este espectro se ven menos afectadas por la postura y expresiones faciales que imágenes Vis, permitiendo incluso la extracción de características biométricas como patrones de vasos sanguíneos [34]. La habilidad de percibir la distribución de temperatura en una escena en T-IR hace a los algoritmos más robustos ante cambios de iluminación, dificultando la facilidad de engañar al sistema por ejemplo con una foto de un objeto [26]. Aplicaciones en este espectro incluyen reconocimiento de rostros [35], identificación de lesiones de piel cancerígenas [36], termografía IR para evaluación no destructiva de materiales, imagenología del cerebro, detección de cáncer de mamas, desórdenes vasculares periféricos, y otras aplicaciones médicas [34]. Sin embargo, oclusiones por elementos opacos como anteojos o cambios en la compensación de temperatura y estado emocional del sujeto [26] son algunos de los problemas más grandes al trabajar con este tipo de imágenes.

El registro de imágenes multimodales tiene una amplia variedad de soluciones de acuerdo a la aplicación, no obstante, la mayoría de ellas está orientada a software y requieren de largos

tiempos de ejecución debido a la complejidad de los algoritmos, ya que en la mayoría de los casos las imágenes multimodales tienen muy pocas características comunes. Como consecuencia, es frecuentemente imposible realizar el registro en línea con la adquisición de las imágenes, por lo cual este proceso es usualmente ejecutado fuera de línea, ralentizando el análisis de las imágenes y comprometiendo la habilidad de un sistema inteligente para reaccionar a la información de las imágenes adquirida en tiempo real. En consecuencia, usando aceleración por hardware para realizar el registro multimodal de las imágenes en línea puede permitir un nuevo rango de aplicaciones en cámaras inteligentes y sistemas embebidos de alto desempeño, al producir la información coherente desde múltiples dispositivos de adquisición a la tasa de adquisición de las imágenes. Dentro de los sistemas que requieren la habilidad de reaccionar rápidamente ante la información entregada se encuentran los lazos de control con dispositivos de adquisición de imágenes para instrumentación, como por ejemplo, para monitorear procesos de combustión [37]. El registro de imágenes en este caso se podría utilizar, por ejemplo, para realizar una reconstrucción espectral con los espectros Vis, SWIR y LWIR que permita estimar la energía de la llama en cualquier banda dentro de este rango, ampliando de esta forma las posibilidades de control.

En este trabajo se presenta un algoritmo y una arquitectura heterogénea para realizar el registro multimodal distribuido de imágenes Vis-SWIR-LWIR en tiempo real. El algoritmo realiza dos procesos de registro simultáneo entre las imágenes Vis-SWIR y SWIR-LWIR con el mapa de coordenadas común SWIR. El algoritmo extrae las características desde ambas imágenes, empareja las características correspondientes, y usa esta información para determinar los parámetros de una transformación proyectiva que mapea la imagen objetivo en el sistema de coordenadas de la imagen de referencia, obteniendo los valores de los pixeles en la imagen transformada mediante interpolación bilineal. Implementado en un *sistema-en-un-chip* (SoC) Zynq XC7Z020 de Xilinx, el prototipo puede registrar imágenes de 640×512 pixeles a más de 60fps, de esta forma habilitando la operación del sistema a la tasa de adquisición de imágenes.

1.2. Hipótesis

Una arquitectura hardware especializada permite ejecutar un algoritmo de registro multimodal distribuido, sobre cámaras inteligentes Vis, SWIR y LWIR, con un mejor desempeño y eficiencia que una implementación en un computador de propósito general, mediante el diseño y utilización de un algoritmo de registro personalizado.

1.3. Objetivos

El objetivo de este trabajo es plantear e implementar un algoritmo que permita realizar el registro multimodal simultáneo entre imágenes Vis, SWIR y LWIR, dividiéndolo en dos subprocesos de registro independientes, diseñando y construyendo un circuito digital en un *arreglo de compuertas programables* (FPGA).

Los objetivos específicos de este trabajo fueron:

1. Definir algoritmo de registro a implementar.
2. Realizar pruebas experimentales en software para asegurar convergencia de algoritmo.
3. Adaptar algoritmo para su implementación en una arquitectura hardware personalizada.
4. Validar arquitectura diseñada en el objetivo anterior con pruebas experimentales sobre un FPGA.
5. Enlazar y adaptar las arquitecturas diseñadas para cada nodo para implementar el sistema multicámara.
6. Validar la arquitectura diseñada en el objetivo anterior con pruebas experimentales sobre un FPGA en conjunto con las cámaras.

1.4. Alcances y limitaciones

- Los instrumentos utilizados serán las cámaras LWIR Tau 2, SWIR Snake SW OEM y Vis comercial, tarjetas de desarrollo ZedBoard Z-7020 de Xilinx, tarjetas de interfaz FMC CameraLink y cables CameraLink disponibles en el *laboratorio de VLSI*.
- Las simulaciones y pruebas en software se harán con el software Matlab, mientras que la síntesis e implementación en hardware se hará con el software Vivado y Xilinx SDK, de Xilinx.
- Las cámaras estarán sobre un montaje fijo, donde no habrá cambios bruscos de las posiciones entre ellas salvo por perturbaciones de los mismos dispositivos.

- Las cámaras capturan áreas similares dentro de una misma escena, con leves diferencias de perspectiva, posición y distancia hacia la escena.
- Las imágenes a utilizar están dentro de un mismo rango de resolución, o en caso contrario se asume se conoce la razón de resolución de pixeles entre las cámaras considerando el área enfocada, el tamaño del lente y la cantidad de pixeles.

1.5. Temario

En este trabajo se presentó un algoritmo y arquitectura heterogénea para realizar el registro multimodal de imágenes Vis-SWIR-LWIR a la tasa de adquisición de imágenes. El algoritmo extrae las características desde ambas imágenes, empareja las características correspondientes, y usa esta información para determinar los parámetros de una transformación proyectiva que mapea la imagen objetivo en el sistema de coordenadas de la imagen de referencia, determinando los valores de los pixeles en la imagen transformada usando interpolación bilineal.

El resto del informe se organiza de la siguiente forma:

- **Capítulo 2:** Muestra el estado del arte sobre algoritmos de registro multimodal de imágenes e implementaciones en hardware de estos, junto con temas relevantes a la implementación en sistemas embebidos.
- **Capítulo 3:** Establece el marco teórico sobre los temas principales del informe, introduciendo la matemática utilizada por las diferentes partes del algoritmo.
- **Capítulo 4:** Presenta el algoritmo de registro diseñado para alinear las imágenes, describiendo por partes cada una de sus fases.
- **Capítulo 5:** Muestra las pruebas en software realizadas para evaluar el algoritmo propuesto.
- **Capítulo 6:** Describe las características del hardware utilizado, un análisis de punto fijo para las operaciones a implementar en hardware, y un diseño de la arquitectura en detalle.
- **Capítulo 7:** Muestra los resultados de la implementación en hardware de la arquitectura diseñada en el capítulo anterior.
- **Capítulo 8:** Presenta un sumario del informe, enumera las conclusiones de este trabajo, y propone trabajo futuro.

Capítulo 2: Estado del arte

2.1. Introducción

En este capítulo se contextualizó el tema de este trabajo a través de una revisión bibliográfica. Esta contextualización se divide en tres secciones: los métodos de registro de imágenes y sus subdivisiones; procesamiento de video en sistema embebidos y comerciales; y una discusión. Las primeras dos secciones recopilan y resumen la información disponible en la literatura con sus principales características. En la discusión se analiza la información recopilada en las secciones anteriores que establecieron el curso a seguir en este trabajo.

2.2. Métodos de registro de imágenes

El registro de imágenes es el proceso en el cual se proyecta una imagen objetivo, a través de una transformación geométrica, al espacio de coordenadas de la imagen de referencia, de forma de alinear ambas imágenes espacialmente. Este proceso comprende cuatro pasos [9]: (1) detección de características, (2) emparejamiento de puntos, (3) estimación de transformación geométrica, y (4) transformar la imagen objetivo en conjunto con alguna técnica de interpolación para adecuarla al sistema de coordenadas del mapa de referencia. Primero se deben extraer los puntos de interés en ambas imágenes, consiguientemente emparejarlos y con ellos determinar los parámetros de una transformación geométrica para proyectar la imagen objetivo a la imagen de referencia, estableciendo una correspondencia punto a punto entre ellas.

Los algoritmos de registro de imágenes usualmente se dividen según la forma en cómo se detectan los puntos de interés: basados en la intensidad de los pixeles, en la extracción de características, o un híbrido de ambos. También se pueden clasificar según el tipo de transformación geométrica utilizada, la cual puede ser lineal o no lineal. Las transformaciones lineales modelan diferencias globales de geometría entre las imágenes, dentro de las cuales se encuentra la rotación, el escalamiento y la traslación. Las no lineales permiten transformaciones elásticas o no rígidas, siendo capaces de deformar localmente la imagen objetivo para alinearla con la de referencia, dentro de ellas se incluyen funciones de base radial, fluidos viscosos y difeomorfismos. Otra forma de clasificarlos es según las modalidades consideradas, pudiendo ser monomodal o multimodal. En este trabajo se utilizan imágenes de los espectros Vis, SWIR y LWIR, por lo

cual se le dará mayor énfasis a los trabajos que contemplan el registro multimodal de imágenes.

2.2.1. Métodos basados en intensidad

Los métodos basados en la intensidad de los pixeles comparan patrones de intensidad en ambas imágenes directamente usando medidas de similitud. Este proceso utiliza secciones de la imagen o ventanas móviles sobre esta, donde los centros de estas ventanas son tratados como POI, con lo cual posteriormente se establece una relación entre los POI de ambas imágenes mediante una medida de similitud.

Es usual que los métodos de registro multimodal actuales utilicen una variante de *información mutua* (MI) [38] como medida de similitud, o un método basado en ella como *información mutua normalizada* (NMI). Un ejemplo de esto lo encontramos en el trabajo de *Babak et al.* [39] donde usan la entropía condicional para reducir el efecto de la interpolación en el cálculo de MI, estimando para esto la varianza en regiones de una imagen que están superpuestas con componentes conectadas específicas de la otra imagen. El origen de este método viene de la premisa que MI no sólo es sensible a la disimilitud entre imágenes, sino que también a operaciones de interpolación aplicadas en el registro. Otra forma de aplicar este tipo de métodos para el registro es utilizar una zona delimitada dentro de una o ambas imágenes, como en el trabajo de *Chen et al.* [40] se registra una *región de interés* (ROI) de secuencias de video IR y Vis mediante una transformación rígida, maximizando la MI usando el método Simplex e interpolación por el vecino más cercano. Al trabajar con secuencias de imágenes asumen que la transformación necesaria para el registro entre ellas es muy similar cuadro a cuadro.

Además de MI y sus variantes, existe una amplia gama de medidas de similitud que utilizan de forma directa el valor de los pixeles, como *coeficiente de correlación* (CC) o *suma de diferencias al cuadrado* (SSD), sin embargo este tipo de métodos presenta típicamente mejores resultados en imágenes monomodales en comparación con su aplicación en imágenes de distinta modalidad. Aun así, existen casos satisfactorios de aplicaciones multimodales, como en el trabajo de *Wanmei et al.* [1] se presenta un algoritmo de registro multimodal basado en SSD que incorpora un modelo polinomial para hacer corrección de intensidad y así lograr el registro de imágenes T1-T2, siendo capaz de corregir rotaciones de 15°. Este método presenta resultados mejores que CC pero similares a MI, sin embargo el método tiene varios mínimos locales y asume una relación funcional entre las intensidades de ambas imágenes.

2.2.2. Métodos basados en características

Los métodos basados en características buscan correspondencia entre rasgos (tales como puntos, líneas o contornos) de la imagen objetivo o una región de esta con respecto a la imagen de referencia, estableciendo de esta forma una correspondencia entre un número de puntos que son significativamente distinguibles en ambas imágenes. Es usual utilizar el centroide de las características detectadas como POI al momento de calcular los parámetros óptimos de la transformación geométrica que alinea las imágenes a registrar.

Dentro de los métodos de detección automática de características se encuentran aquellos basados en la detección de esquinas, como el detector de Harris [41] o aquellos basados en la detección usando el espacio de escala como los algoritmos *transformación de característica invariante a la escala* (SIFT) [42] y *características robustas aceleradas* (SURF) [43]. Al igual que en el caso de MI, hay muchas modificaciones y variantes de estos algoritmos utilizados para realizar el registro de imágenes. Un ejemplo de esto es el trabajo presentado por *Hossain et al.* [2], donde se modifica SIFT simétrico usando *diferencia de Gaussiana* (DoG) en el espacio de escala para mejorar el registro rígido multimodal de imágenes de *resonancia magnética* (MR), T1, T2 y Vis-*infrarrojo cercano* (NIR). Para esto, los autores usan la mediana de la diferencia de las orientaciones dominantes entre ambas imágenes como alineamiento inicial, logrando de esta forma resultados hasta un 30 % mejores que el algoritmo SIFT simétrico original, con un error promedio menor a 4 pixeles. El método propuesto funciona mejor que SIFT simétrico debido a que la normalización en base al ángulo derivado es menos susceptible a errores en comparación con orientaciones dominantes, y al no haber fusión del descriptor la perdida de información que ocurre en SIFT simétrico puede ser totalmente evadida. Otro ejemplo lo muestran *Firmenich et al.* [44], donde además de utilizar una modificación de SIFT usan una generalización multiespectral del detector de esquinas de Harris y DoG para registrar imágenes NIR y Vis, logrando de esta forma invariancia a cambios de 180° en la dirección del gradiente, con errores menores a 3 pixeles en comparación con *consenso de muestra aleatoria* (RANSAC) como registro verdadero.

Un método que ha llamado la atención últimamente es *campo de gradientes normalizado* (NGF) debido a su fácil paralelización de operaciones. El trabajo de *Rühaak et al.* [3] es un buen ejemplo de esto, donde lo utilizan para el registro multimodal de imágenes de *tomografía de emisión de positrones* (PET) y *tomografía computarizada* (CT) optimizando por Gauss-Newton, corrigiendo así errores de 10cm de traslación y 15° de rotación mediante transformación afín o rígida. La idea principal de este algoritmo es medir el ángulo entre dos gradientes de imágenes y alinearlos de forma paralela o antiparalela. Además de su paralelización, resalta el bajo consumo de memoria que requiere esta implementación, logrando un error promedio

de 2.12mm en el proceso de registro, y requiriendo de 1.69s en un computador i7-2600 de 3.4GHz. Existen también algoritmos menos comunes para realizar el registro que de igual manera presentan resultados satisfactorios, como en el trabajo de *Hajebi et al.* [45] utilizan modelos de congruencia de fase para discernir profundidad en imágenes estereoscópicas IR comparando las características mediante coeficientes de Log-Gabor a diferentes frecuencias y orientaciones. En este trabajo se evaluó el método de forma manual usando ventanas de 5×5, donde un 48 % de las características encontradas fueron emparejadas y el error final fue de un 4.5 %.

Una alternativa a realizar el proceso de registro de forma continua para un conjunto de cámaras es hacer este registro como una calibración inicial de escena para un montaje fijo, asumiendo para esto que las cámaras mantienen su posición en este montaje. Un ejemplo de esto se muestra en el trabajo de *Hess et al.* [14], donde se hace un registro inicial mediante la búsqueda de características en las imágenes para calibrar un sistema estático. Utilizando esta pieza de montaje seguro, los autores generaron un mosaico de imágenes térmicas y visibles para el análisis no destructivo de estructuras. Este acercamiento se basa en la geometría fija del montaje y la posición relativa de las cámaras, donde la comparación visual de los mosaicos registrados demuestra que el registro inicial es casi indistinguible al de realizar la fase de registro para cada par de imágenes, comparando para esto 44 pares de imágenes térmico-visible usando la transformación proyectiva.

2.2.3. Métodos híbridos

Los métodos de registro híbridos utilizan medidas de similitud basadas tanto en la intensidad de los pixeles como en características de las imágenes, con lo cual son típicamente más robustos tanto en la detección como en la asociación de puntos de interés.

Es usual que los métodos híbridos utilicen MI junto con otro algoritmo basado en características para el proceso de registro. Un ejemplo de esto se encuentra en el trabajo de *Pluim et al.* [4], donde además de MI utilizan la magnitud y orientación del gradiente para el registro de imágenes MR, CT y PET mediante transformación afín. Los autores asumen que las imágenes representan la misma estructura anatómica pero en diferente modalidad, con lo cual se espera que ambas estructuras deben tener la misma orientación, y misma u opuesta dirección. De esta forma integran información espacial a una medida de probabilidad de información, haciendo más robusto el proceso de registro. De manera similar, el trabajo presentado por *Barrera et al.* [46] usa la información del gradiente junto con MI, pero en este caso se utiliza el espacio de escala para alinear las imágenes IR térmicas con las Vis. Al propagar los valores por los

distintos niveles y escalas de la pirámide acentúan las diferencias entre los máximos locales en comparación con utilizar sólo la métrica de MI, facilitando de esta forma el proceso de asociación de puntos de interés. Otros autores, como *Woo et al.* [47], incorporan al cálculo de MI la información espacial y geométrica del voxel a través del operador de Harris 3D. Esta variante del algoritmo se basa en que una de las imágenes tiene mayor resolución que la otra, con lo cual el operador de Harris 3D determina la contribución de la intensidad de cada voxel en el histograma, registrando de esta forma regiones espacialmente significativas.

Además de MI también se utilizan métricas derivadas de esta como NMI para métodos híbridos. Un ejemplo de esto lo presentan *Yu et al.* [5], los cuales registran imágenes médicas usando NMI e imágenes de gradiente con peso en el mismo espacio modal. La idea tras estas imágenes de gradiente en el mismo espacio modal es guiar el proceso de registro con los bordes de las estructuras, donde la transformación se realiza mediante una grilla B-Spline jerárquica y el modelo de registro *deformaciones de forma libre* (FFD). Este método presentó mejores resultados que NMI para el registro no rígido de imágenes CT y MR con un rendimiento de 96 % en 50 casos. Otro trabajo donde también se utiliza NMI, pero con otro tipo de características de la imagen, es el presentado por *Reducindo et al.* [6], los cuales registran imágenes MR en varios pasos. Primero aproximan la deformación global usando una transformación rígida, basada en *filtrado de partículas* (PF) y NMI. Posteriormente se transforman las imágenes a un nuevo espacio de color usando la entropía de Shannon, se obtiene el *flujo óptico* (OF) entre ellas usando el algoritmo de Horn y Shuck en espacio de escala, y se hace registro no rígido. Los resultados muestran un error menor a dos pixeles en la estimación del campo vectorial de deformación.

De manera similar al trabajo presentado por *Reducindo et al.* [6], otros autores utilizan la transformación de imágenes a un nuevo espacio de color en el proceso de registro multimodal de imágenes, como lo presentado por *Shi et al.* [48], donde se registran imágenes MR y CT aplicando una transformación de intensidad a estas y luego usando ambos pares de imágenes (original y transformada), junto con una función de energía en base a *campos aleatorios de Markov* (MRF) y optimizado por *BFGS de memoria limitada* (L-BFGS), para encontrar el mínimo de la función para el alineamiento. La transformación de intensidad reduce el problema a un registro quasi-monomodal, donde además se utiliza un esquema multinivel para trabajar en espacio de escala, corrigiendo de esta forma deformaciones globales en los niveles más gruesos y afinando el registro en los niveles más finos. Una alternativa un poco más compleja, pero que igualmente muestra buenos resultados, es propuesta por *Lu et al.* [49], los cuales integran el reconocimiento de características al algoritmo multimodal difeomórfico demons usando funciones Gaussianas de base radial. La transformación utilizada es no rígida, donde la función de energía está compuesta por MI, información de estas características y un termino de regularización. Al comparar los

resultados con el registro manual de las imágenes el error promedio es menor a 1.5 pixeles.

2.3. Procesamiento de video en sistemas embebidos

Los sistemas embebidos son unidades de procesamiento con una tarea en específico dentro de un sistema más grande, como por ejemplo la corrección de *no uniformidad* (NU) [50] que comúnmente necesitan los *arreglo de plano focal infrarrojo* (IRFPA) de las cámaras IR producto de las imperfecciones en su fabricación, o aumentar la resolución de una cámara de baja resolución mediante algoritmos de *superresolución* (SR) [51]. Existen varios tipos de sistemas embebidos que pueden realizar este tipo de tareas. Dentro de estos encontramos los FPGA, compuestos por un arreglo de bloques con lógica programable y una jerarquía de interconexiones reconfigurable que permite implementar lógica digital con un bajo costo y tiempo asociado, siendo ideales para prototipos. También se encuentran los *circuito integrado de aplicación específica* (ASIC) que, aun cuando no permiten reconfiguración como los FPGA y poseen un costo inicial de diseño muy superior a estos, poseen una mayor capacidad de personalización y para grandes volúmenes de producción su costo por unidad es ínfimo. Sistemas más avanzados como SoC incluyen FPGA y ASIC como parte de su infraestructura además de típicamente núcleos de procesamiento, *procesador digital de señales* (DSP), memoria e interfaces de comunicación en un mismo dispositivo. Los tipos de procesamiento anteriormente mencionados, y varios otros relacionados con el procesamiento de imágenes, se presentan en dispositivos denominados cámaras inteligentes, donde es usual que además contengan un procesador, memoria y puertos de comunicación.

Volviendo al problema de registro de imágenes, existe una amplia gama de trabajos que buscan su solución en sistemas embebidos. Un ejemplo de esto es el trabajo de *Inostroza et al.* [19], donde se realiza el registro multimodal de imágenes Vis y video LWIR en un SoC Zybo Z-7010 de Xilinx. El registro se hace en base a la detección de cuatro esquinas de un marcador contenido en una ROI mediante el detector de Harris, descenso de gradiente, y utilizando transformación afín. La parte de hardware realiza la detección y extracción de características, y en software se estiman los parámetros de la transformación y se mapea cada cuadro de video LWIR en la imagen Vis. De manera similar al caso anterior, *White* [52] implementa un núcleo de registro de imágenes en una tarjeta de desarrollo ML506 de Xilinx, donde el OF modela la transformación afín global entre las imágenes y mediante su estimación se obtienen los parámetros de la transformación. Otra alternativa es utilizar tarjetas DSP para aumentar el flujo de cómputo, como en el trabajo de *Berg et al.* [53] se implementa un algoritmo de registro en

una arquitectura multinúcleo distribuida usando mínimos cuadrados, transformación rígida y optimización multinivel con Gauss-Newton en un sistema integrado por cuatro tarjetas DSP TI C6678 con 8 núcleos cada una.

Al trabajar con un conjunto de dispositivos de adquisición es usual que la transmisión de información proporcionada por cada uno hacia un sistema de procesamiento sea uno de los mayores problemas, ya que el flujo de la información obtenida es en muchos casos mayor al que puede ser procesado por el sistema. Una solución a este tipo de problemas es la distribución del cómputo entre los dispositivos de adquisición, descentralizando de esta manera el procesamiento y reduciendo de forma significativa tanto la carga computacional como el ancho de banda necesario para la transmisión de información. Un ejemplo de este tipo de soluciones se puede apreciar en el trabajo de *Bourrasset et al.* [54], donde el procesamiento en una red de cámaras inteligentes se realiza de forma distribuida a nivel de cada cámara mediante comunicación Ethernet. El montaje consiste en dos nodos, un switch Ethernet y un computador para validación, donde cada nodo de esta red está compuesta por una cámara *semiconductor complementario de óxido metálico* (CMOS), un FPGA y seis memorias externas.

En el ámbito comercial existen varios sistemas de captura que permiten apreciar imágenes desde distintos puntos de vista ya alineadas, ya sea en la misma sección del espectro electromagnético o imágenes multimodales. Para el primer caso tenemos ejemplos como MCAW de Tetracam [55] que permite focalizar 6 bandas del espectro Vis-NIR en un sistema integrado de imágenes registradas utilizando 6 dispositivos de captura con una resolución de $1,280 \times 1,024$ pixeles a 1Hz por imagen utilizando 9.6W, donde las imágenes se registran mediante escalado, rotación y traslación. En el caso multimodal existen productos como FLIR Duo de FLIR [56], compuesta por un sistema de dos cámaras integradas en el mismo dispositivo, que permite obtener imágenes LWIR de 160×120 pixeles registrada con una imagen Vis de $1,920 \times 1,080$ pixeles a 75-83Hz consumiendo 3.3W. Este sistema permite añadir la información de bordes obtenida en el espectro Vis al LWIR, lo cual es registrado mediante la calibración de las cámaras y corrección usando distancia focal, permitiendo de forma adicional mejorar el alineamiento horizontal y vertical manualmente por el usuario.

Existen además dispositivos de captura que permiten obtener imágenes de distintas partes del espectro desde un mismo punto de vista, ya sea en una misma imagen ampliando el rango detectado por el sensor, o mediante un conjunto de sensores que comparten el mismo lente con un filtro que separa el haz de luz en los distintos rangos espectrales para cada sensor. En el primer caso tenemos cámaras como la Tigris-640 de Xenics [57] que trabaja en el espectro *infrarrojo de onda media* (MWIR) pero permite aumentar el rango espectral incluyendo parte

de SWIR, con una resolución de 640 × 512 pixeles por imagen a 357Hz utilizando 25W. Sin embargo, sólo se obtiene una imagen pues en esencia se cambia el rango de lo capturado por el detector y no la cantidad de imágenes obtenidas por el dispositivo. Para el caso de cámaras con filtros internos existen productos como FD-3SWIR de FluxData [58] que permiten especificar los detectores al momento de fabricación, aumentando de esta forma la personalización de la cámara para determinadas aplicaciones. Para esta cámara en particular se puede elegir entre *dispositivo de carga acoplada* (CCD) o *arseniuro de indio y galio* (InGaAs) para capturar combinaciones de imágenes como Vis-NIR-SWIR con 2CCD-InGaAs, pudiendo además focalizar longitudes de onda específicas en estos espectros con una resolución de 640 × 512 pixeles a 30Hz utilizando 4.5W. Cabe mencionar que como las imágenes tienen el mismo lente no es necesario el proceso de registro, pues son fabricadas para enfocar la misma escena desde el mismo punto de vista. Otro exponente de este tipo de cámaras lo constituye PixelCam OEM de PIXELTEQ [59], que de forma similar al ejemplo anterior permite seleccionar 4 bandas a focalizar mediante filtros de luz en Vis-NIR con un detector CCD, o en el rango NIR-SWIR con un detector InGaAs. Esta tiene una resolución que varía entre los 640 × 512 pixeles a 8 megapixeles con una tasa de 21-44Hz para los distintos modelos.

2.4. Discusión

El registro de imágenes es esencial al momento de analizar información proveniente de múltiples dispositivos de adquisición. En la literatura se encontró una gran gama de soluciones, sin embargo la mayoría de ellas es diseñada para un problema en específico y no funciona correctamente para todas las posibles combinaciones de imágenes. Además, estas soluciones son típicamente orientadas a software debido a la complejidad de los algoritmos utilizados, significando en la gran mayoría de los casos que este procesamiento se realiza fuera de línea. Si bien los algoritmos con medidas de similitud híbridas presentan típicamente mejores resultados y mayor robustez, usualmente requieren cómputos más complejos y tiempos más extensos para alinear las imágenes, haciéndolos menos atractivos para implementaciones en sistemas embebidos donde los recursos disponibles son limitados. Sin embargo, de ellos se debe destacar la noción de que típicamente agregando información espacial como bordes y formas al cálculo de medidas de similitud basadas en intensidad mejora la calidad de los resultados.

Variantes de MI son usualmente utilizadas para métodos basados en la intensidad debido al alto éxito en el proceso de registro entre diversas modalidades. No obstante, esta medida es altamente no convexa y difícil de implementar [60], además de estar fuertemente ligada a

características como el tamaño de la ventana de búsqueda e información contenida en dicha ventana. Si bien SIFT y SURF son fuertemente utilizados debido a los buenos resultados en la detección de características, uno de los mayores problemas es la alta cantidad de operaciones necesarias al trabajar con todo el espacio de escala simultáneamente. Otro método basado en características es NGF, el cual permite una alta paralelización de operaciones y por consiguiente un flujo de procesamiento muy superior a algoritmos como MI, SIFT o SURF. Transformaciones lineales son típicamente preferidas al momento de implementar algoritmos de registro en sistemas embebidos, ya que el costo computacional que ellas requieren es mucho menor a sus contrapartes no lineales, haciendo más factible su implementación en hardware. A esto se agregan los buenos resultados de alineamiento global que típicamente obtienen estos métodos, aún cuando los no lineales presentan un mejor alineamiento local a mayor complejidad y costo computacional.

Los distintos productos comerciales analizados muestran la existencia de un mercado para el registro de imágenes, con exponentes como FLIR Duo o MCAW. La alternativa a estas cámaras son dispositivos de captura que comparten un lente y separan la escena hacia los detectores mediante filtros ópticos. No obstante, ambos tipos de dispositivos tienen la desventaja que tanto la resolución como la sección del espectro capturado quedan fijos al momento de fabricación, lo cual en el caso de imágenes registradas por distintos dispositivos de captura permite una mayor personalización. Tal como se muestra en el trabajo de *Bourrasset et al.* [54] dividir el cómputo del algoritmo a nivel de los dispositivos de adquisición permite descentralizar el proceso de registro, disminuyendo de esta forma la cantidad de información a ser procesada y aumentando la fluidez del procesamiento.

Considerando todo lo anterior, se propone en este trabajo descentralizar el algoritmo de registro de imágenes y distribuirlo en las distintas tarjetas de desarrollo que se incorporan como parte del sistema de adquisición de imágenes. Debido a la factibilidad de implementación en hardware se escogieron las transformaciones lineales como transformación geométrica, factores comunes en las imágenes como las formas de los objetos para realizar el alineamiento, y métricas que puedan utilizar esta información para extraer características que puedan ser emparejadas en las imágenes a registrar.

Capítulo 3: Marco teórico

3.1. Introducción

En este capítulo se describen las herramientas matemáticas y algoritmos básicos usados en el algoritmo de registro propuesto, el cual se describe en el siguiente capítulo haciendo referencia a las herramientas aquí descritas. Se parte con una introducción general al proceso de registro, seguido de los algoritmos utilizados para la extracción de características y métrica de similitud. Posteriormente se describen las transformaciones geométricas lineales y se da especial énfasis en la transformación proyectiva, seguido de procesos de interpolación para obtener el valor de los pixeles una vez sus coordenadas fueron transformadas. Finalmente se menciona brevemente el proceso de estimación con el cual se obtienen los parámetros de la transformación geométrica.

3.2. Registro de imágenes

En la Figura 3.1 se muestra un diagrama del proceso de registro, el cual se describe a continuación. El proceso comienza considerando una matriz de transformación inicial para modificar geométricamente la imagen objetivo, en este caso la matriz identidad. Luego se interpola la imagen transformada para ajustarla a la grilla de coordenadas de referencia mediante alguna técnica de interpolación. El siguiente paso es extraer en ambas imágenes características o POI para asociarlos mediante la métrica de similitud. Luego de tener estos puntos emparejados se procede a la fase de optimización donde se evalúa si un criterio de detención se ha alcanzado, en cuyo caso el proceso de registro termina, o en caso contrario se calculan nuevos parámetros para la transformación geométrica y se vuelve a iniciar el proceso con estos parámetros.

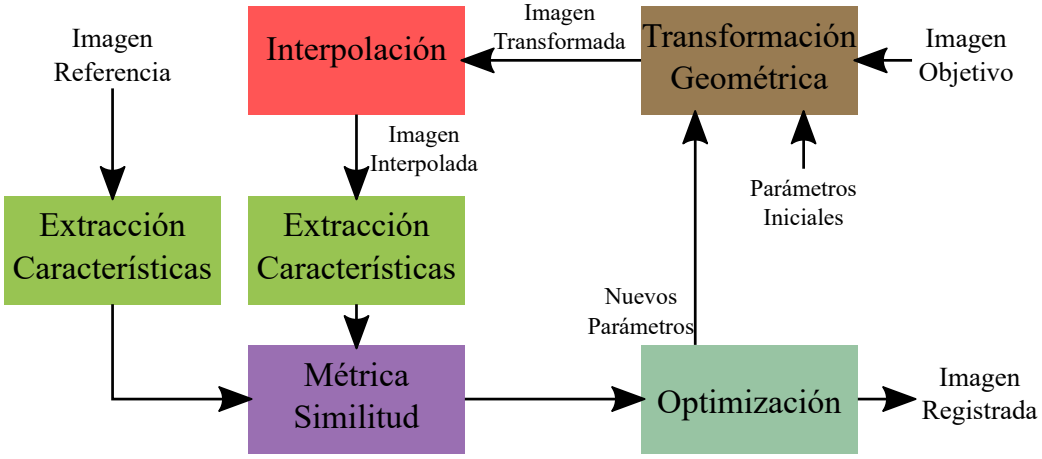


Figura 3.1: Diagrama del proceso de registro. Fuente: Elaboración propia.

3.3. Extracción de características

En esta sección se describen los distintos algoritmos utilizados en la extracción de características. El algoritmo propuesto en este informe utiliza una variante de *histograma de gradientes orientados* (HOG) para extraer características de POI del contorno de objetos que pueden ser visualizados tanto en la imagen de referencia como en la imagen objetivo. Esta variante utiliza NGF en vez de la magnitud del gradiente para construir el histograma de cada celda.

3.3.1. Histograma de gradientes orientados

HOG fue originalmente propuesto por Dalal y Triggs [61], usado como un descriptor de características en procesamiento de imágenes para reconocimiento de objetos y detección de peatones. Este algoritmo se basa en el gradiente de los pixeles y analiza ventanas dentro una imagen centradas en POI, los cuales son divididos en secciones más pequeñas denominadas celdas. Para cada celda se crea un histograma de frecuencia que representa las direcciones de los gradientes ponderados por su magnitud. Estos histogramas obtenidos por celda son concatenados para todas las celdas de la ventana, formando un descriptor espacialmente orientado del POI. Los bins del histograma de cada celda son rangos igualmente espaciados de orientaciones de gradiente, y los valores son las magnitudes del gradiente o una función de ellos.

Como se mencionó anteriormente, HOG utiliza la magnitud y ángulo de gradiente de los pixeles pertenecientes a las imágenes a analizar. Debido a esto, primero se necesita calcular las imágenes de gradiente en las direcciones x e y para cada imagen a analizar, con las cuales se

determinan tanto la magnitud como el ángulo de gradiente de cada pixel. Esto se puede realizar de diversas formas, como por ejemplo realizando la convolución entre los operadores Sobel y la imagen $\mathcal{R}$ sobre la cual se desea determinar su gradiente:

$$\mathbf{g}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathcal{R}, \qquad \mathbf{g}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathcal{R}, \tag{3.1}$$

donde $\mathbf{g}_x$ y $\mathbf{g}_y$ son las imágenes de gradientes en el eje x e y respectivamente de la imagen $\mathcal{R}$. De esta forma, la magnitud y ángulo para cada pixel esta dado por:

$$g_M^{(x,y)} = \sqrt{\left(g_x^{(x,y)}\right)^2 + \left(g_y^{(x,y)}\right)^2}, \tag{3.2}$$

$$g_A^{(x,y)} = \operatorname{atan}\left(\frac{g_y^{(x,y)}}{g_x^{(x,y)}}\right), \tag{3.3}$$

donde $g_x^{(x,y)}$ y $g_y^{(x,y)}$ son los gradientes en las direcciones x e y en la posición (x, y), $g_M^{(x,y)}$ es la magnitud del gradiente y $g_A^{(x,y)}$ el ángulo de este. Generalmente se llega al consenso [61] de que aplicando HOG usando las orientaciones de gradiente sin signo con 9 bins sobre celdas rectangulares logra buenos resultados.

Se determinó empíricamente al experimentar con imágenes Vis, SWIR y LWIR que utilizando bloques rectangulares de histogramas con 9 bins (ángulos sin signo) produce un descriptor capaz de emparejar POI del contorno de objetos entre imágenes Vis, SWIR y LWIR. Cada bloque esta compuesto por 2 conjuntos de 4 celdas no superpuestas (los 2 conjunto si están superpuestos entre ellos), formando un histograma de 72 elementos para cada POI, lo cual se ilustra en la Figura 3.2. Se utiliza NGF en vez de la magnitud del gradiente del pixel para formar el histograma de cada celda. El primer conjunto de 4 celdas tiene un tamaño de celda de 16×16, y el segundo un tamaño de celda de 32×32. El primer conjunto es utilizado como el valor formal del descriptor, y el segundo conjunto para guiarlo y aumentar la precisión del emparejamiento. De esta forma, cada celda del primer conjunto esta escalada entre 0 y 1 y el segundo conjunto entre 0 y 0.25, para que dentro de cada conjunto cada celda tenga el mismo valor para la posterior comparación de los histogramas. La normalización del gradiente ayuda a disminuir las diferencias entre los valores de gradientes obtenidos debido a las diferentes físicas correspondientes a las imágenes de diferentes módulos, y concentrarse solamente en donde se producen estos gradientes y con que forma.

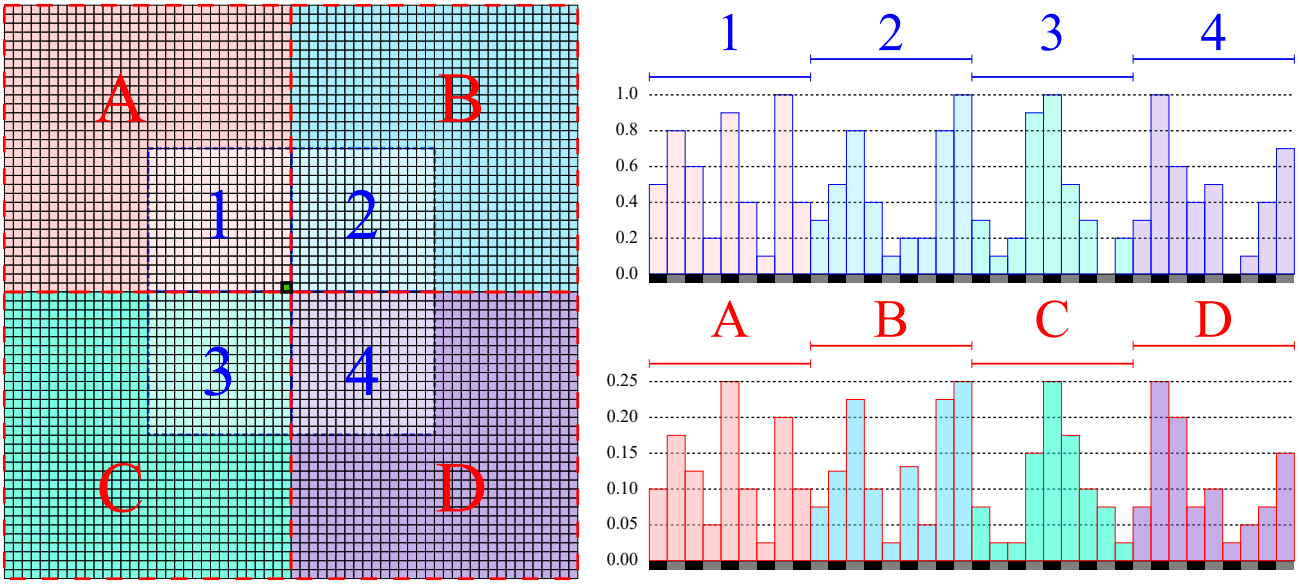


Figura 3.2: Descriptor HOG utilizado. Fuente: Elaboración propia.

3.3.2. Campo de gradientes normalizado

Se basa en una interpretación de la similitud entre dos imágenes [62]: "dos imágenes son consideradas similares si los cambios en intensidad ocurren en las mismas ubicaciones". Estos cambios en intensidad pueden ser identificados por su gradiente. Sin embargo el gradiente también mide la magnitud del cambio, el cual puede estar relacionado con el dispositivo de adquisición de imágenes, y no con diferencias en la imagen, por lo cual no es aconsejable usar el gradiente directamente. Debido a esto, el gradiente es reemplazado por el gradiente normalizado, lo cual remueve información no deseada y se centra en el lugar de los cambios en vez de su magnitud. De esta forma, la magnitud de gradiente normalizada de un pixel de posición x, y usando NGF es:

$$g_{M_{NGF}}^{(x,y)} = \frac{\sqrt{\left(g_x^{(x,y)}\right)^2 + \left(g_y^{(x,y)}\right)^2}}{\sqrt{\left(g_x^{(x,y)}\right)^2 + \left(g_y^{(x,y)}\right)^2 + \varepsilon^2}}, \tag{3.4}$$

donde ε es un parámetro de regularización que determina qué es considerado como ruido [63]. Sin este parámetro, el ruido en la imagen exaltado por la normalización corrompería la información importante dada por el campo de gradientes normalizado. Esto es, en regiones donde ε tiene un valor mucho mayor que el gradiente, el valor de $g_{M_{NGF}}^{(x,y)}$ tiende a 0 y por lo tanto no tiene un valor significativo. En regiones donde ε es mucho menor que el gradiente, el valor de $g_{M_{NGF}}^{(x,y)}$ se acerca mucho al caso no regularizado, por lo cual estas regiones marcan una diferencia sustancial.

3.4. Medidas de similitud

3.4.1. Distancia Chi-cuadrado

La distancia Chi-cuadrado es una medida de distancia Euclidiana ponderada que compara la similaridad de dos muestras donde sus diferencias están enfatizadas. Esto es, las muestras son más similares si el valor de su distancia Chi-cuadrado disminuye. Esta métrica de distancia es útil al comparar datos entre dos histogramas [64] y se calcula de la siguiente manera:

$$D_{chi}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{x_i + y_i}, \tag{3.5}$$

donde $\mathbf{x}$ e $\mathbf{y}$ son las muestras a ser comparadas, x_i e y_i los elementos de $\mathbf{x}$ e $\mathbf{y}$ respectivamente, n la cantidad de elementos de cada muestra, y $D_{chi}(\mathbf{x}, \mathbf{y})$ es la distancia Chi-cuadrado entre $\mathbf{x}$ e $\mathbf{y}$. Se debe tener en cuenta que cuando $x_i + y_i$ es 0, entonces $\frac{(x_i - y_i)^2}{x_i + y_i}$ se considera como 0.

3.5. Transformaciones geométricas lineales

En esta sección se describen las principales características de las distintas transformaciones geométricas lineales con las cuales se pueden alinear las imágenes globalmente. Una descripción más detallada se puede apreciar en *Hartley et al.* [65]. Para facilitar el uso de las transformaciones geométricas sobre las imágenes, las coordenadas de los pixeles serán representadas en el espacio de coordenadas homogéneas, permitiendo aplicar dichas transformaciones mediante un producto matricial sobre las coordenadas.

Las coordenadas homogéneas describen la posición de los pixeles en las imágenes, representados por puntos en $\mathbb{R}^2$, en el plano proyectivo $\mathbb{P}^2$, representado por puntos en $\mathbb{R}^3$. Dado un punto $\mathbf{p}$, con coordenadas (p_x, p_y) en $\mathbb{R}^2$, para ser representado con coordenadas homogéneas (p_u, p_v, p_w) en $\mathbb{R}^3$ se tiene que:

$$\begin{bmatrix} p_x & p_y \end{bmatrix}^{\mathrm{T}} \equiv \begin{bmatrix} p_u & p_v & p_w \end{bmatrix}^{\mathrm{T}}, \tag{3.6}$$

donde para cada número real p_w distinto de cero el valor de p_x y p_y esta dado por:

$$p_x = \frac{p_u}{p_w}, \qquad p_y = \frac{p_v}{p_w}. \tag{3.7}$$

En caso de que $p_w = 0$ se dice que dicho punto en coordenadas homogéneas se encuentra en el infinito. En particular, considerando que $p_w = 1$ se tiene que $p_x = p_u$ y $p_y = p_v$, con lo cual estos puntos pueden ser finalmente expresados como:

$$\begin{bmatrix} p_x & p_y \end{bmatrix}^{\mathrm{T}} \equiv \begin{bmatrix} p_x & p_y & 1 \end{bmatrix}^{\mathrm{T}}, \tag{3.8}$$

en $\mathbb{R}^2$ y $\mathbb{R}^3$ respectivamente.

Una transformación geométrica, también denominada homografía, es un mapa invertible $\mathcal{M}$ en coordenadas homogéneas hacia coordenadas homogéneas, donde tres puntos permanecen en la misma línea si y sólo si la transformación de los mismos también lo hace. La inversa de una homografía también es una homografía, así como también lo es la composición de dos homografías. Transformar el punto $\mathbf{p}$ usando el mapa $\mathcal{M}$ en coordenadas homogéneas en el punto $\mathbf{q}$ consiste simplemente en multiplicar la matriz de transformación $\mathbf{M}$ por las coordenadas del punto $\mathbf{p}$, es decir:

$$\mathbf{q} = \mathcal{M}(\mathbf{p}) = \mathbf{M}\mathbf{p}. \tag{3.9}$$

Se define de esta forma la matriz de transformación $\mathbf{M}$ como:

$$\mathbf{M}(a,b,c,d,e,f,g,h,i) = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}, \tag{3.10}$$

donde $\{a, b, c, d, e, f, g, h, i\}$ son los parámetros de la transformación y definen que tipo de transformación geométrica se aplica sobre las coordenadas. Considerando por ejemplo la transformación identidad, la cual no produce cambios en las coordenadas de la imagen, y con $\mathbf{p} = \begin{bmatrix} p_x & p_y & 1 \end{bmatrix}^{\mathrm{T}}$ y $\mathbf{q} = \begin{bmatrix} q_x & q_y & 1 \end{bmatrix}^{\mathrm{T}}$, se tiene que para el ejemplo anterior:

$$\begin{bmatrix} q_x \\ q_y \\ 1 \end{bmatrix} = \mathbf{q} = \mathcal{M}(\mathbf{p}) = \mathbf{M}\mathbf{p} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} \Longrightarrow \begin{bmatrix} q_x \\ q_y \\ 1 \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix}. \tag{3.11}$$

A continuación se describe la transformación geométrica lineal utilizada para alinear las imágenes en base a las consideraciones mencionadas anteriormente.

3.5.1. Transformación proyectiva

La transformación proyectiva [65] es una generalización de la transformación afín, combinando traslación, escalamiento, rotación y perspectiva. Tiene ocho grados de libertad y sólo se conservan las líneas rectas. A diferencia del caso afín, no es posible distinguir entre la preservación de la orientación y su reversión. La matriz de transformación $\mathbf{M}$ puede ser representada como:

$$\mathbf{M}(\mathbf{h}) = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix}, \tag{3.12}$$

con $\mathbf{h} = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 \end{bmatrix}^{\mathrm{T}}$ los parámetros de la transformación. Una transformación proyectiva entre dos planos puede ser calculada de la correspondencia de cuatro pares de puntos, donde no hayan tres puntos colineales en cualquier plano. Al igual que los casos anteriores, la última columna es responsable de los efectos de traslación, y en particular la última fila es responsable de los efectos no lineales en la proyección, donde un punto en el infinito puede ser mapeado a un punto finito debido a la no linealidad que ingresan estos parámetros.

Para determinar los parámetros de la transformación proyectiva consideremos primero otra representación de la misma:

$$\mathbf{M}(\mathbf{h}) = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1^{\mathrm{T}} \\ \mathbf{m}_2^{\mathrm{T}} \\ \mathbf{m}_3^{\mathrm{T}} \end{bmatrix}, \tag{3.13}$$

con $\mathbf{m}_1^{\mathrm{T}}$, $\mathbf{m}_2^{\mathrm{T}}$ y $\mathbf{m}_3^{\mathrm{T}}$ las filas de la matriz de transformación $\mathbf{M}(\mathbf{h})$. De esta forma se puede representar la transformación del punto $\mathbf{p}$ utilizando la matriz de transformación proyectiva $\mathbf{M}$ para obtener el punto $\mathbf{q}$ como:

$$\begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} = \mathbf{q} = \mathbf{M}(\mathbf{h})\mathbf{p} = \begin{bmatrix} \mathbf{m}_1^{\mathrm{T}}\mathbf{p} \\ \mathbf{m}_2^{\mathrm{T}}\mathbf{p} \\ \mathbf{m}_3^{\mathrm{T}}\mathbf{p} \end{bmatrix}. \tag{3.14}$$

Para determinar los parámetros de esta transformación podemos recurrir al producto cruz

entre $\mathbf{q}$ y $\mathbf{M}(\mathbf{h})\mathbf{p}$:

$$\mathbf{q} \times \mathbf{M}(\mathbf{h})\mathbf{p} = \begin{bmatrix} q_y \mathbf{m}_3^{\mathrm{T}}\mathbf{p} - q_z \mathbf{m}_2^{\mathrm{T}}\mathbf{p} \\ q_z \mathbf{m}_1^{\mathrm{T}}\mathbf{p} - q_x \mathbf{m}_3^{\mathrm{T}}\mathbf{p} \\ q_x \mathbf{m}_2^{\mathrm{T}}\mathbf{p} - q_y \mathbf{m}_1^{\mathrm{T}}\mathbf{p} \end{bmatrix} = \begin{bmatrix} -q_z \mathbf{p}^{\mathrm{T}}\mathbf{m}_2 + q_y \mathbf{p}^{\mathrm{T}}\mathbf{m}_3 \\ q_z \mathbf{p}^{\mathrm{T}}\mathbf{m}_1 - q_x \mathbf{p}^{\mathrm{T}}\mathbf{m}_3 \\ -q_y \mathbf{p}^{\mathrm{T}}\mathbf{m}_1 + q_x \mathbf{p}^{\mathrm{T}}\mathbf{m}_2 \end{bmatrix} = \mathbf{0}, \tag{3.15}$$

que es 0, pues es el producto cruz entre el mismo vector. Ordenando con respecto a $\mathbf{m}_i$ y considerando que:

$$\begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \mathbf{m}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{h} \\ 1 \end{bmatrix}, \tag{3.16}$$

se tiene que:

$$\begin{bmatrix} \mathbf{0} & -q_z \mathbf{p}^{\mathrm{T}} & q_y \mathbf{p}^{\mathrm{T}} \\ q_z \mathbf{p}^{\mathrm{T}} & \mathbf{0} & -q_x \mathbf{p}^{\mathrm{T}} \\ -q_y \mathbf{p}^{\mathrm{T}} & q_x \mathbf{p}^{\mathrm{T}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{h} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3.17}$$

de lo cual se puede observar que el sistema de ecuaciones es lineal con respecto a $\mathbf{h}$. Aun cuando hay tres ecuaciones para cada par de puntos, sólo dos de ellas son linealmente independientes, por lo que la tercera es usualmente omitida [65]. De esta forma, para cada par de puntos $\mathbf{p}$ y $\mathbf{q}$ se tiene el sistema de ecuaciones:

$$\begin{bmatrix} 0 & 0 & 0 & -q_z p_x & -q_z p_y & -q_z p_z & q_y p_x & q_y p_y & q_y p_z \\ q_z p_x & q_z p_y & q_z p_z & 0 & 0 & 0 & -q_x p_x & -q_x p_y & -q_x p_z \end{bmatrix} \begin{bmatrix} \mathbf{h} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{3.18}$$

donde considerando que estamos trabajando en coordenadas homogéneas en la forma $\mathbf{p} = \begin{bmatrix} p_x & p_y & 1 \end{bmatrix}^{\mathrm{T}}$, el sistema de ecuaciones para cada par de puntos $\mathbf{p}$ y $\mathbf{q}$ queda como:

$$\begin{bmatrix} p_x & p_y & 1 & 0 & 0 & 0 & -q_x p_x & -q_x p_y \\ 0 & 0 & 0 & p_x & p_y & 1 & -q_y p_x & -q_y p_y \end{bmatrix} \mathbf{h} = \begin{bmatrix} q_x \\ q_y \end{bmatrix}. \tag{3.19}$$

De la Ecuación 3.19, necesitamos por lo menos 4 pares de puntos para resolver el sistema de ecuaciones, ya que tenemos 8 variables desconocidas. El sistema de ecuaciones para 4 pares de

puntos queda como:

$$\begin{bmatrix} q_x^1 \\ q_y^1 \\ q_x^2 \\ q_y^2 \\ q_x^3 \\ q_y^3 \\ q_x^4 \\ q_y^4 \end{bmatrix} = \begin{bmatrix} p_x^1 & p_y^1 & 1 & 0 & 0 & 0 & -q_x^1 p_x^1 & -q_x^1 p_y^1 \\ 0 & 0 & 0 & p_x^1 & p_y^1 & 1 & -q_y^1 p_x^1 & -q_y^1 p_y^1 \\ p_x^2 & p_y^2 & 1 & 0 & 0 & 0 & -q_x^2 p_x^2 & -q_x^2 p_y^2 \\ 0 & 0 & 0 & p_x^2 & p_y^2 & 1 & -q_y^2 p_x^2 & -q_y^2 p_y^2 \\ p_x^3 & p_y^3 & 1 & 0 & 0 & 0 & -q_x^3 p_x^3 & -q_x^3 p_y^3 \\ 0 & 0 & 0 & p_x^3 & p_y^3 & 1 & -q_y^3 p_x^3 & -q_y^3 p_y^3 \\ p_x^4 & p_y^4 & 1 & 0 & 0 & 0 & -q_x^4 p_x^4 & -q_x^4 p_y^4 \\ 0 & 0 & 0 & p_x^4 & p_y^4 & 1 & -q_y^4 p_x^4 & -q_y^4 p_y^4 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \end{bmatrix}, \tag{3.20}$$

con $\mathbf{p}^i$ y $\mathbf{q}^i$ el i-ésimo par de puntos. Con los valores de $\mathbf{h}$ ya calculados, se proyecta el punto $\mathbf{p}$ resolviendo:

$$\begin{aligned} p_x &= \frac{h_1 q_x + h_2 q_y + h_3}{h_7 q_x + h_8 q_y + 1}, \\ p_y &= \frac{h_4 q_x + h_5 q_y + h_6}{h_7 q_x + h_8 q_y + 1}. \end{aligned} \tag{3.21}$$

3.6. División por Newton-Rhapson

Como se muestra en la Ecuación 3.21, para aplicar la transformación proyectiva a las coordenadas se debe realizar una división por un término dependiente de las coordenadas, es decir, por un término no constante. Ya que los FPGA normalmente no incluyen divisores hardware, se necesitan algoritmos para calcular esta división. Uno de estos es el algoritmo de Newton-Rhapson [66], el cual está basado en multiplicaciones y restas sucesivas, converge en pocas iteraciones, y puede ser fácilmente implementado como un pipeline en FPGAs.

El problema de la división se puede describir como el producto del dividendo por el inverso del divisor:

$$Q = \frac{n}{d} = n \times \frac{1}{d} = n \times D, \tag{3.22}$$

donde D es el inverso del divisor d, n es el dividendo, y Q el cociente.

Para encontrar el inverso D utilizando Newton-Rhapson se necesita una función $f(D)$ que

sea cero cuando $D = \frac{1}{d}$, sin embargo esto no se puede calcular sin saber de antemano el valor del inverso de d. Una función que si funciona es $f(D) = \frac{1}{D} - d$, ya que se parte de una aproximación inicial D_0 del inverso de d, para lo cual una iteración del algoritmo de Newton-Rhapson da:

$$D_{i+1} = D_i - \frac{f(D_i)}{f'(D_i)} = D_i - \frac{\frac{1}{D_i} - d}{\frac{-1}{D_i^2}} = D_i + D_i \times (1 - d \times D_i) = D_i \times (2 - d \times D_i)\,, \quad (3.23)$$

lo cual puede ser calculado a partir de una aproximación inicial D_0 del inverso del divisor utilizando sólo multiplicaciones y restas.

Newton-Rhapson busca iterativamente D y lo multiplica por n para calcular Q. Este algoritmo trabaja en tres pasos: Primero, estima una primera aproximación D_0; luego, iterativamente calcula una aproximación más precisa D_i, hasta llegar a la aproximación D_S con la precisión deseada; finalmente, multiplica n por D_S para obtener Q. La iteración D_i se puede calcular como:

$$D_{i+1} = D_i \times (2 - d \times D_i)\,, \quad (3.24)$$

con D_i la aproximación actual del inverso de d, y D_{i+1} la aproximación a ser calculada.

3.7. Métodos de interpolación

Al realizar el registro de imágenes es típicamente necesario muestrear las coordenadas de la imagen objetivo ya transformada para adecuarla al mapa de coordenadas o grilla de la imagen de referencia, ya que usualmente las coordenadas resultantes en el proceso de registro no coinciden perfectamente con las de la grilla original. La interpolación de imágenes consiste en aproximar el valor de un pixel para valores no dados de este teniendo valores de otros pixeles que se encuentran en coordenadas próximas.

En la Figura 3.3 se muestran las grillas de dos imágenes sobre un mismo eje de coordenadas que serán utilizadas en la explicación de los algoritmos de interpolación, donde sus pixeles se denotan como $p_{x,y}$ para la imagen objetivo transformada y $g_{x,y}$ para la imagen de referencia, donde x, y denota la posición en los ejes x e y respectivamente.

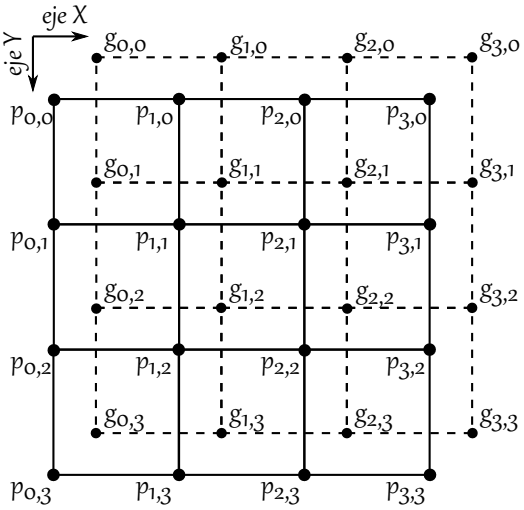


Figura 3.3: Grilla para interpolación. Fuente: Elaboración propia.

3.7.1. Interpolación bilineal

La interpolación bilineal es la composición de dos interpolaciones lineales sucesivas en dos direcciones diferentes. Primero se interpola en una dirección y después nuevamente en la otra dirección. Si escogemos un sistema de coordenadas donde los valores de los cuatro puntos $p_{x,y}$ que rodean a $g_{x,y}$ son conocidos y los denotamos con coordenadas $p^{(0,0)}$, $p^{(0,1)}$, $p^{(1,0)}$ y $p^{(1,1)}$ respectivamente, y considerando las coordenadas del pixel $g_{x,y}$ en este sistema, que se denotan como $\hat{x}$ e $\hat{y}$ para los ejes x e y respectivamente (con valores entre 0 y 1 por el sistema de coordenadas elegido), entonces la fórmula de interpolación se simplifica a:

$$g_{x,y} = p^{(0,0)}(1-\hat{x})(1-\hat{y}) + p^{(0,1)}(1-\hat{x})\hat{y} + p^{(1,0)}\hat{x}(1-\hat{y}) + p^{(1,1)}\hat{x}\hat{y}. \tag{3.25}$$

En el caso mostrado en la Figura 3.4 el valor de $g_{1,2}$ esta dado por los cuatro puntos $p_{1,1}$, $p_{1,2}$, $p_{2,1}$ y $p_{2,2}$ que son $p^{(0,0)}$, $p^{(0,1)}$, $p^{(1,0)}$ y $p^{(1,1)}$ respectivamente.

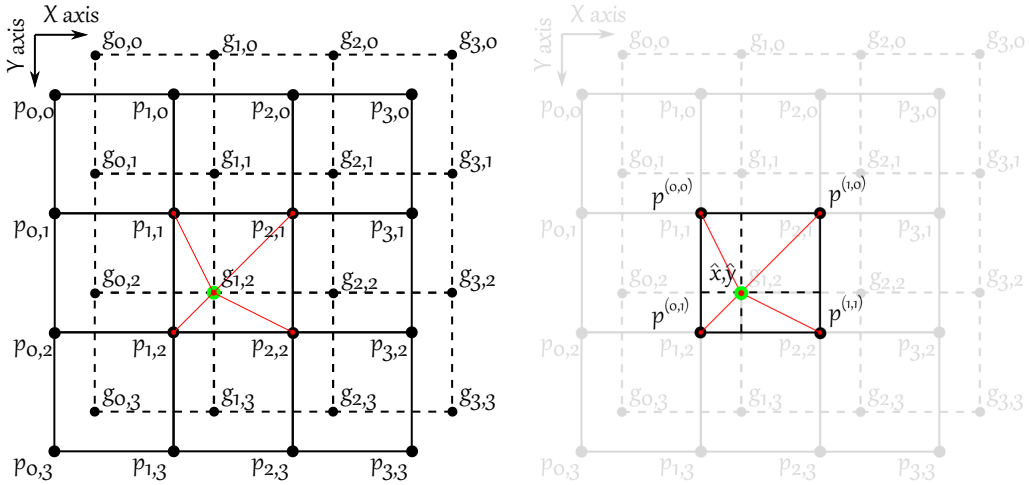


Figura 3.4: Interpolación bilineal. Fuente: Elaboración propia.

3.8. Métodos de estimación

El objetivo de la estimación es ajustar los parámetros de la transformación geométrica dado un conjunto de pares de puntos ya asociados $\mathbf{p}^i$ y $\mathbf{q}^i$ mediante un criterio de optimización. A continuación se describen los métodos de estimación utilizados en este trabajo.

3.8.1. Mínimos cuadrados

El objetivo de mínimos cuadrados es encontrar el valor de los parámetros que mejor se ajuste a un modelo dado un conjunto de datos, donde el óptimo se encuentra cuando la función de costo I tiene su valor mínimo:

$$I = \frac{1}{2}\sum_{i=1}^{n} e_i^2, \tag{3.26}$$

con e_i el error de aproximación del i-ésimo par de datos, n el número de pares de datos, y m el número de parámetros a determinar. Consideremos el caso de encontrar los parámetros de la

transformación proyectiva:

$$\underbrace{\begin{bmatrix} q_x^1 \\ q_y^1 \\ q_x^2 \\ q_y^2 \\ \vdots \\ q_x^n \\ q_y^n \end{bmatrix}}_{\mathbf{b}} = \underbrace{\begin{bmatrix} p_x^1 & p_y^1 & 1 & 0 & 0 & 0 & -q_x^1 p_x^1 & -q_x^1 p_y^1 \\ 0 & 0 & 0 & p_x^1 & p_y^1 & 1 & -q_y^1 p_x^1 & -q_y^1 p_y^1 \\ p_x^2 & p_y^2 & 1 & 0 & 0 & 0 & -q_x^2 p_x^2 & -q_x^2 p_y^2 \\ 0 & 0 & 0 & p_x^2 & p_y^2 & 1 & -q_y^2 p_x^2 & -q_y^2 p_y^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ p_x^n & p_y^n & 1 & 0 & 0 & 0 & -q_x^n p_x^n & -q_x^n p_y^n \\ 0 & 0 & 0 & p_x^n & p_y^n & 1 & -q_y^n p_x^n & -q_y^n p_y^n \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \end{bmatrix}}_{\mathbf{h}}, \tag{3.27}$$

donde $\mathbf{p}^i$ y $\mathbf{q}^i$ son el i-ésimo par de puntos ya asociados, $\mathbf{h}$ el vector de parámetros a encontrar de la transformación proyectiva, m la cantidad de parámetros a encontrar que en este caso es 8, y n el número de pares de puntos. De esta forma el error para el i-ésimo par de datos es:

$$e_i = b_i - \mathbf{A}_i^{\mathrm{T}} \mathbf{h}, \tag{3.28}$$

con b_i el i-ésimo elemento de $\mathbf{b}$, y $\mathbf{A}_i^{\mathrm{T}}$ la i-ésima fila de $\mathbf{A}$. De esta forma la función de costo se puede expresar como:

$$I = \frac{1}{2} \sum_{i=1}^{n} e_i^2 = \frac{1}{2} \sum_{i=1}^{n} \left(b_i - \mathbf{A}_i^{\mathrm{T}} \mathbf{h} \right)^2, \tag{3.29}$$

lo que se puede representar de forma matricial como:

$$I = \frac{1}{2} \left\| \mathbf{e} \right\|^2 = \frac{1}{2} \left\| \mathbf{b} - \mathbf{A}\mathbf{h} \right\|^2. \tag{3.30}$$

El mínimo de la función de costo se puede encontrar con los valores de los parámetros que anulan al gradiente, lo cual para el caso lineal se puede estimar de la forma:

$$\hat{\mathbf{h}} = \left(\mathbf{A}^{\mathrm{T}} \mathbf{A} \right)^{-1} \mathbf{A}^{\mathrm{T}} \mathbf{b}, \tag{3.31}$$

donde $\hat{\mathbf{h}}$ es la estimación del vector de parámetros $\mathbf{h}$.

3.8.2. Descenso de gradiente

La estimación de los parámetros de la transformación proyectiva se obtiene iterativamente utilizando descenso de gradiente, donde el indice para la iteración k es:

$$I(k) = \frac{1}{2}\sum_{i=1}^{n} e_i(k)^2 = \frac{1}{2}\sum_{i=1}^{n}\left(b_i(k) - \mathbf{A}_i(k)^{\mathrm{T}}\mathbf{h}(k)\right)^2, \tag{3.32}$$

con $I(k)$ el valor de la función de costo en la iteración k, $\mathbf{e}(k)$ el error en la estimación de los parámetros en la iteración k, $\mathbf{h}(k)$ la estimación de los parámetros en la iteración k, y tanto $\mathbf{b}(k)$ como $\mathbf{A}(k)$ contienen las coordenadas de los pares de puntos utilizados para la estimación de los parámetros en la iteración k. De esta forma, el cálculo iterativo de los parámetros por descenso de gradiente es:

$$\mathbf{h}(k) = \mathbf{h}(k-1) - \eta\nabla I(k-1) = \mathbf{h}(k-1) - \eta\sum_{i=1}^{n} e_i(k-1)\mathbf{A}_i(k-1), \tag{3.33}$$

donde k indica la iteración y η el tamaño del paso que determina la convergencia de este método.

Capítulo 4: Algoritmo de registro

4.1. Descripción general

En este capítulo se describe el Algoritmo 1 de registro diseñado, el cual está dividido en dos fases: una calibración inicial para calcular la transformación geométrica que registra la imagen objetivo $\mathcal{O}$ al sistema de coordenadas de la imagen de referencia $\mathcal{R}$ (líneas 1-15); el registro propiamente tal entre cada cuadro de video de $\mathcal{O}$ y $\mathcal{R}$ utilizando la matriz de transformación calculada en la fase de calibración (líneas 16-19). La imagen SWIR es utilizada como referencia $\mathcal{R}$, y las imágenes Vis y LWIR como imagen objetivo $\mathcal{O}$, donde el algoritmo se utiliza entre una imagen $\mathcal{R}$ y una imagen $\mathcal{O}$ para registrar los pares de imágenes Vis-SWIR y SWIR-LWIR, utilizando el mapa de referencia común $\mathcal{R}$ para tener las tres imágenes alineadas en el mismo sistema de coordenadas. Para la fase de calibración se necesita que el contorno de los objetos sea claramente distinguible en ambas imágenes para poder obtener una matriz de transformación de forma correcta, además de que el factor de escala, rotación y enfoque entre ambas imágenes sea similar.

Algorithm 1: Algoritmo de registro.

1 Definir parámetros iniciales para la matriz de transformación $\mathbf{M}^0$.
2 Capturar la escena actual con ambas cámaras.
3 Detectar el contorno en las imágenes objetivo $\mathcal{O}$ y referencia $\mathcal{R}$.
4 Formar una mascara de POI para $\mathcal{O}$ y $\mathcal{R}$.
5 **for** *cada iteración* k **do**
6 Aplicar el descriptor HOG a los POI.
7 Detectar el centroide de los POI en $\mathcal{O}$ y $\mathcal{R}$.
8 Determinar el ángulo y distancia de los POI y centroides.
9 Calcular D^k_{chi} para emparejar POI de $\mathcal{O}$ hacia $\mathcal{R}$.
10 Ordenar las similaridades y descartar falsos positivos.
11 **for** *cada iteración* i **do**
12 Estimar $\mathbf{M}^k_i$ con los POI emparejados.
13 Transformar los POI de $\mathcal{O}$ usando $\mathbf{M}^k_i$.
14 Determinar el error medio entre los puntos emparejados.
15 Terminar si se llega a un umbral de error o límite de iteraciones.
16 **for** *cada cuadro de video a registrar* **do**
17 Calcular las coordenadas transformadas $\Omega = \mathcal{M}(\mathcal{O})$.
18 Interpolar los valores para obtener el cuadro registrado $\mathcal{T}$.
19 Combinar $\mathcal{T}$ con $\mathcal{R}$ y desplegar las imágenes.

4.2. Fase de calibración inicial

En esta sección se describe en detalle la fase de calibración inicial del Algoritmo 1, la cual comprende las líneas 1-15 del mismo. Esta calibración consta de 5 procesos: detectar el contorno de los objetos de las imágenes objetivo y referencia (línea 3 del Algoritmo 1); extraer los POI que se encuentran en el contorno detectado en ambas imágenes para formar una máscara de puntos en ambas imágenes (línea 4); aplicar el descriptor HOG para extraer las características de estos POI en ambas imágenes (línea 6); emparejar las características de los POI de la imagen objetivo con los de la imagen de referencia utilizando la distancia Chi-cuadrado (línea 9); determinar los parámetros de la matriz de transformación proyectiva, utilizando estos POI emparejados, mediante descenso de gradiente (líneas 11-15).

La fase de calibración comienza estableciendo la matriz identidad como valor inicial para la matriz de transformación $\mathbf{M}^0$ entre la imagen objetivo $\mathcal{O}$ (ya sea la imagen Vis o LWIR) y la imagen de referencia $\mathcal{R}$ (línea 1). La escena actual es capturada por ambas cámaras obteniendo una imagen $\mathcal{O}$ y una imagen $\mathcal{R}$ estática, con la cual se determinará la matriz de transformación que registra ambas escenas (línea 2).

El contorno de los objetos presentes en la escena se detecta mediante un filtro Gaussiano y la magnitud del gradiente utilizando los operadores sobel en las imágenes $\mathcal{O}$ y $\mathcal{R}$ (línea 3). El filtro Gaussiano se utiliza para disminuir la cantidad de texturas visibles en las imágenes y que pueden ser detectados como bordes, específicamente en los espectros Vis y SWIR. Posteriormente se aplican los operadores sobel de la Ecuación 3.1 para calcular la magnitud del gradiente en ambas imágenes, luego de lo cual mediante una umbralización se determina que es considerado como borde. Los puntos son seleccionados en estas imágenes de contornos como POI si su magnitud de gradiente es la mayor dentro de la vecindad observada por una ventana de 3×3 con este punto candidato en el centro de la ventana, de esta forma los POI corresponden a puntos donde se presentan cambios fuertes de gradiente dentro del contorno de los objetos (línea 4).

Una vez detectados los POI en $\mathcal{O}$ y $\mathcal{R}$, se procede a la extracción de características para cada punto utilizando HOG como se muestra en la Figura 3.2, generando un descriptor de 72 elementos para cada POI (línea 6). Adicionalmente se determina el centroide de los POI en $\mathcal{O}$ y $\mathcal{R}$, la distancia y ángulo entre cada POI de $\mathcal{O}$ con cada POI de $\mathcal{R}$, y la distancia y ángulo entre sus centroides (líneas 7 y 8). Posteriormente se calcula la métrica de similitud entre los puntos de cada imagen utilizando distancia Chi-cuadrado presentada en la Ecuación 3.5 entre cada POI de $\mathcal{O}$ con cada POI de $\mathcal{R}$ (línea 9). Para cada punto en $\mathcal{R}$, el algoritmo ordena las distancias hacia cada punto en $\mathcal{O}$, creando un emparejamiento preliminar al seleccionar la

mínima distancia Chi-cuadrado (línea 10). En este emparejamiento preliminar, cada POI de $\mathcal{R}$ es emparejado con un único POI de $\mathcal{O}$, pero POIs de $\mathcal{O}$ pueden ser emparejados a 0 o más POIs en $\mathcal{R}$. El emparejamiento preliminar es iterativamente refinado eliminando parejas basándose en la distancia y ángulo entre los POI, utilizando también la distancia y ángulo del centroide de los POIs en ambas imágenes (calculados en la línea 8), hasta que sólo quedan POI emparejados 1 a 1 entre ambas imágenes (línea 10).

Luego de ordenar los puntos emparejados y descartar emparejamientos falsos positivos, se utiliza un algoritmo de descenso de gradiente para estimar iterativamente la matriz de transformación $\mathbf{M}_i^k$ (resolviendo la Ecuación 3.20) y se proyectan las coordenadas de cada POI en $\mathcal{O}$ usando la Ecuación 3.21 hacia un conjunto de coordenadas proyectadas Ω (líneas 12-15). El algoritmo de descenso de gradiente itera hasta llegar a un umbral satisfactorio en el error medio entre los POI emparejados, o hasta alcanzar un límite de iteraciones (en los experimentos, este límite se estableció como $i = 20,000$ iteraciones). El proceso de estimación completo (líneas 6-15) se repite k veces (en los experimentos se determinó que $k = 6$), usando la matriz de transformación estimada en la iteración k anterior para refinar el emparejamiento preliminar calculado en la línea 9.

4.3. Fase de registro cuadro a cuadro

En esta sección se describe brevemente la fase de registro de cada cuadro del Algoritmo 1, la cual comprende las líneas 16-19 del mismo. Esta fase se aplica para cada par de imágenes $\mathcal{O}$ y $\mathcal{R}$, a forma de obtener registros Vis-SWIR o SWIR-LWIR según sea el caso, utilizando la imagen SWIR como referencia y la Vis o LWIR (según sea el caso) como imagen objetivo. Luego de obtener los parámetros de la transformación proyectiva que mapean de forma óptima los puntos de la imagen $\mathcal{O}$ al sistema de coordenadas de la imagen $\mathcal{R}$ (determinados en la fase de calibración, obteniendo como resultado la matriz de transformación $\mathbf{M}_{i=20,000}^{k=6}$ que por simplicidad se denotará como $\mathbf{M}$), se aplica la matriz de transformación proyectiva $\mathbf{M}$ a la imagen objetivo $\mathcal{O}$ para obtener las coordenadas transformadas al sistema de referencia $\Omega = \mathcal{M}(\mathcal{O})$ (línea 17). Una vez obtenidas las coordenadas transformadas se utiliza interpolación bilineal para determinar el valor en cuentas digitales del pixel que corresponde a esas coordenadas, obteniendo la imagen proyectada $\mathcal{T}$ al sistema de referencia que contiene el valor de los pixeles de $\mathcal{O}$ interpolados (línea 18). Finalmente, se combinan $\mathcal{T}$ y $\mathcal{R}$ para desplegar las imágenes registradas (línea 19).

Capítulo 5: Evaluación del algoritmo

5.1. Descripción del sistema

En el montaje, las cámaras Vis, SWIR y LWIR están en una posición fija para capturar y registrar una escena, como se muestra en la Figura 5.1. Para realizar el registro entre dos imágenes se asume que los objetos permanecen en aproximadamente las mismas posiciones relativas a las cámaras, y que el enfoque se mantiene. Para las pruebas en software, se capturaron imágenes de una misma escena fuera de línea donde los contornos de los objetos para la calibración podían ser visualizados por todas las cámaras. Debido a la alta sensibilidad de la cámara SWIR ante la luz sólo se capturaron imágenes en interiores, para evitar saturar el detector de esta cámara. En el caso de la cámara LWIR se consideraron objetos dentro de la escena que tengan gradientes de temperatura con respecto al entorno.

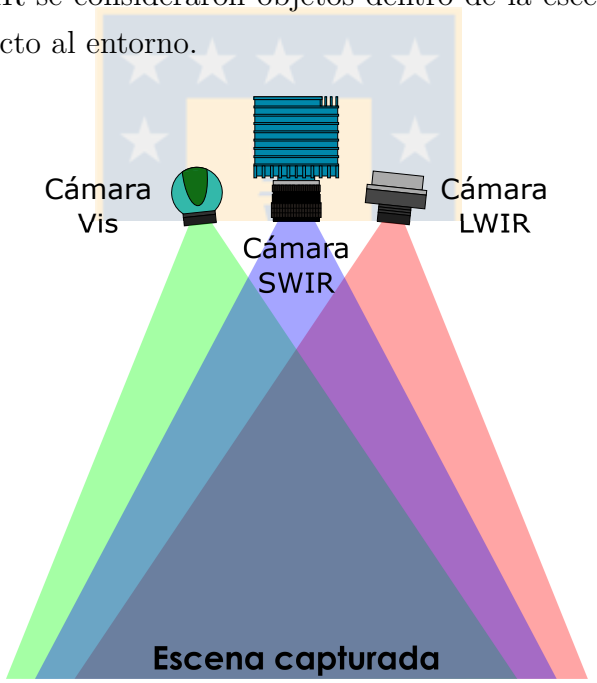


Figura 5.1: Diagrama montaje de cámaras. Fuente: Elaboración propia.

5.2. Instrumentos utilizados

En esta sección se describen los instrumentos utilizados para adquirir las imágenes con las cuales se realizaron los experimentos. Estos consisten en una cámara Vis comercial que captura imágenes en el espectro visible, una cámara LWIR Tau 2 de FLIR que captura imágenes en el

espectro IR térmico, y una cámara SWIR Snake SW OEM de Sofradir que captura imágenes del espectro IR reflectivo. A continuación se describen en detalle dichos dispositivos de adquisición.

5.2.1. Cámara Vis

La cámara Vis utilizada es una cámara web comercial, capaz de capturar imágenes y video en el espectro visible en el espacio de color *rojo-verde-azul* (RGB). Tiene un sensor CCD, operando (teóricamente) en el rango 0.4µm a 0.7µm del espectro electromagnético. Permite el despliegue de video digital a 640×480 pixeles a través de USB.

5.2.2. Cámara SWIR

La cámara SWIR utilizada es la Snake SW OEM de Sofradir, la cual es capaz de capturar imágenes y video en el espectro IR reflectivo. Esta cámara cuenta con una interfaz óptica del tipo montura C, y un lente intercambiable de 25mm de distancia focal, con iris y foco ajustable de forma manual. Tiene un sensor InGaAs con un tamaño de pixel de 15µm, operando en el rango 0.9µm a 1.7µm del espectro electromagnético. Posee una eficiencia cuántica mayor al 70 % en el espectro comprendido entre 1µm y 1.6µm de longitud de onda. Permite el despliegue de video digital a 640×512 pixeles a 60 cuadros por segundo, con señales del tipo *señal diferencial de bajo voltaje* (LVDS) y una resolución de 14bits por pixel, esto a través de la interfaz CameraLink.

5.2.3. Cámara LWIR

La cámara LWIR utilizada es la Tau 2 de FLIR, la cual es capaz de capturar imágenes y video en el espectro IR térmico. Esta cámara cuenta con un lente de 13mm de distancia focal, con foco ajustable de forma manual. El *arreglo de plano focal* (FPA) de esta cámara está basado en microbolómetros no refrigerados de *óxido de vanadio* (VOx) con un tamaño de pixel de 17µm, operando en el rango 7.5µm a 13.5µm del espectro electromagnético. Tiene una *temperatura diferencial equivalente al ruido* (NEDT) menor a 50mK a $f/1.0$. Puede capturar un rango de temperatura entre -40°C y 160°C operando en alta ganancia, y entre -40°C a 550°C en baja ganancia. Permite el despliegue de video digital a 640×512 pixeles a 30 cuadros por segundo, con señales de tipo LVDS o CMOS en 14bits u 8bits por pixel, a través de un conector Hirose de 50 pines o de la interfaz CameraLink.

5.3. Registro de imágenes Vis y SWIR

En esta sección se muestran las pruebas realizadas para el registro Vis-SWIR. Tres pares de imágenes Vis-SWIR utilizadas en estas pruebas se muestran en la Figura 5.2. Primero se muestran los resultados del registro de estas imágenes utilizando el Toolbox de Matlab, luego utilizando el algoritmo propuesto, y finalmente una comparación entre los resultados obtenidos.

(a) Imagen 1 SWIR. (b) Imagen 2 SWIR. (c) Imagen 3 SWIR.

(d) Imagen 1 Vis. (e) Imagen 2 Vis. (f) Imagen 3 Vis.

Figura 5.2: Imágenes utilizadas en los experimentos. Fuente: Elaboración propia.

5.3.1. Resultados usando información mutua

Se utilizó el Toolbox de Matlab para hacer el registro entre imágenes Vis y SWIR. Este Toolbox utiliza la intensidad de los pixeles, MI como medida de similitud, un esquema piramidal, y transformación afín para registrar las imágenes. De forma empírica se determinaron los parámetros de este Toolbox, sin embargo para poder registrar correctamente las imágenes estas debieron ser recortadas y redimensionadas a imágenes de la misma resolución (en este caso 640×512 pixeles), como se muestra en las imágenes Vis y SWIR de las Figuras 5.3, 5.4 y 5.5.

En las Figuras 5.3 a 5.5 se muestran los resultados de registro Vis-SWIR utilizando el Toolbox

de Matlab. En cada una de estas figuras se muestra el desalineamiento inicial entre las imágenes Vis y SWIR, y el resultado del registro entre ambas imágenes seleccionando distinta cantidad de iteraciones como parámetro de este Toolbox.

(a) Imagen Vis. (b) Imagen SWIR. (c) Sin registrar.

(d) Iteración 10. (e) Iteración 50. (f) Iteración 200.

(g) Iteración 500. (h) Iteración $1,000$. (i) Iteración $2,000$.

Figura 5.3: Registro Vis-SWIR utilizando información mutua, imagen 1. Fuente: Elaboración propia.

En el caso del registro Vis-SWIR utilizando este Toolbox, se puede apreciar de las Figuras 5.3, 5.4 y 5.5 que a mayor cantidad de iteraciones el algoritmo del Toolbox converge a una solución de mayor calidad, y ya con $2,000$ iteraciones o más las imágenes están visualmente registradas para los tres pares de imágenes.

(a) Imagen Vis. (b) Imagen SWIR. (c) Sin registrar.

(d) Iteración 10. (e) Iteración 50. (f) Iteración 200.

(g) Iteración 500. (h) Iteración $1,000$. (i) Iteración $2,000$.

Figura 5.4: Registro Vis-SWIR utilizando información mutua, imagen 2. Fuente: Elaboración propia.

(a) Imagen Vis. (b) Imagen SWIR. (c) Sin registrar.

(d) Iteración 10. (e) Iteración 50. (f) Iteración 200.

(g) Iteración 500. (h) Iteración $1,000$. (i) Iteración $2,000$.

Figura 5.5: Registro Vis-SWIR utilizando información mutua, imagen 3. Fuente: Elaboración propia.

5.3.2. Resultados usando algoritmo propuesto

Se programó el Algoritmo 1 en Matlab para hacer el registro entre imágenes Vis y SWIR. Este algoritmo utiliza HOG como extractor de características, la distancia Chi-cuadrado como medida de similitud, y transformación proyectiva para registrar las imágenes.

En las Figuras 5.6 a 5.8 se muestran los resultados de registro Vis-SWIR utilizando el algoritmo propuesto. En cada una de estas figuras se muestra el desalineamiento inicial entre las imágenes Vis y SWIR (Para la imagen 1, Figura 5.6a), iteraciones intermedias de registro del algoritmo (Figuras 5.6b a 5.6h), y el resultado del registro entre ambas imágenes (Figura 5.6i).

En cada imagen, k corresponde a la iteración actual del proceso de estimación de los parámetros de la transformación proyectiva, e i corresponde a la cantidad de iteraciones del algoritmo de descenso de gradiente para estimar los parámetros de la transformación proyectiva dentro de una iteración k. Las Figuras 5.6b y 5.6c muestran iteraciones intermedias del algoritmo de descenso de gradiente para encontrar los parámetros óptimos dentro de la iteración $k = 1$ de la estimación de los parámetros. Las Figuras 5.6d a 5.6i muestran el resultado del alineamiento con los parámetros determinados en las iteraciones $k = 1$ a $k = 6$, donde la iteración $k = 6$ corresponde al resultado del registro entre ambas imágenes.

De las Figuras 5.6, 5.7 y 5.8 se puede apreciar que las imágenes son registradas globalmente de forma satisfactoria, sin embargo se logra apreciar un contorno levemente deformado entre ambas imágenes, el cual se debe principalmente a la baja calidad de la imagen Vis y en segundo lugar a las diferentes texturas que se pueden apreciar entre las imágenes Vis y SWIR. Esta baja calidad y diferentes texturas repercute en la formación de la máscara de POIs de la imagen Vis que se utiliza para el alineamiento entre ambas imágenes (detectando bordes en lugares donde la textura es altamente no uniforme, detectando bordes considerablemente más gruesos que en la imagen SWIR, o no detectando contornos en lugares donde la imagen SWIR si detecta), lo cual a su vez afecta los parámetros de la transformación calculada con estos puntos emparejados entre ambas imágenes. Aun así, el algoritmo logra registrar satisfactoriamente ambas imágenes.

(a) Sin registrar. (b) Iteración $k = 1$, $i = 100$. (c) Iteración $k = 1$, $i = 1,000$.

(d) Iteración $k = 1$. (e) Iteración $k = 2$. (f) Iteración $k = 3$.

(g) Iteración $k = 4$. (h) Iteración $k = 5$. (i) Iteración $k = 6$.

Figura 5.6: Registro Vis-SWIR utilizando el algoritmo propuesto, imagen 1. Fuente: Elaboración propia.

(a) Sin registrar. (b) Iteración $k = 1$, $i = 100$. (c) Iteración $k = 1$, $i = 1,000$.

(d) Iteración $k = 1$. (e) Iteración $k = 2$. (f) Iteración $k = 3$.

(g) Iteración $k = 4$. (h) $k = 5$. (i) Iteración $k = 6$.

Figura 5.7: Registro Vis-SWIR utilizando el algoritmo propuesto, imagen 2. Fuente: Elaboración propia.

(a) Sin registrar. (b) Iteración $k = 1$, $i = 100$. (c) Iteración $k = 1$, $i = 1,000$.

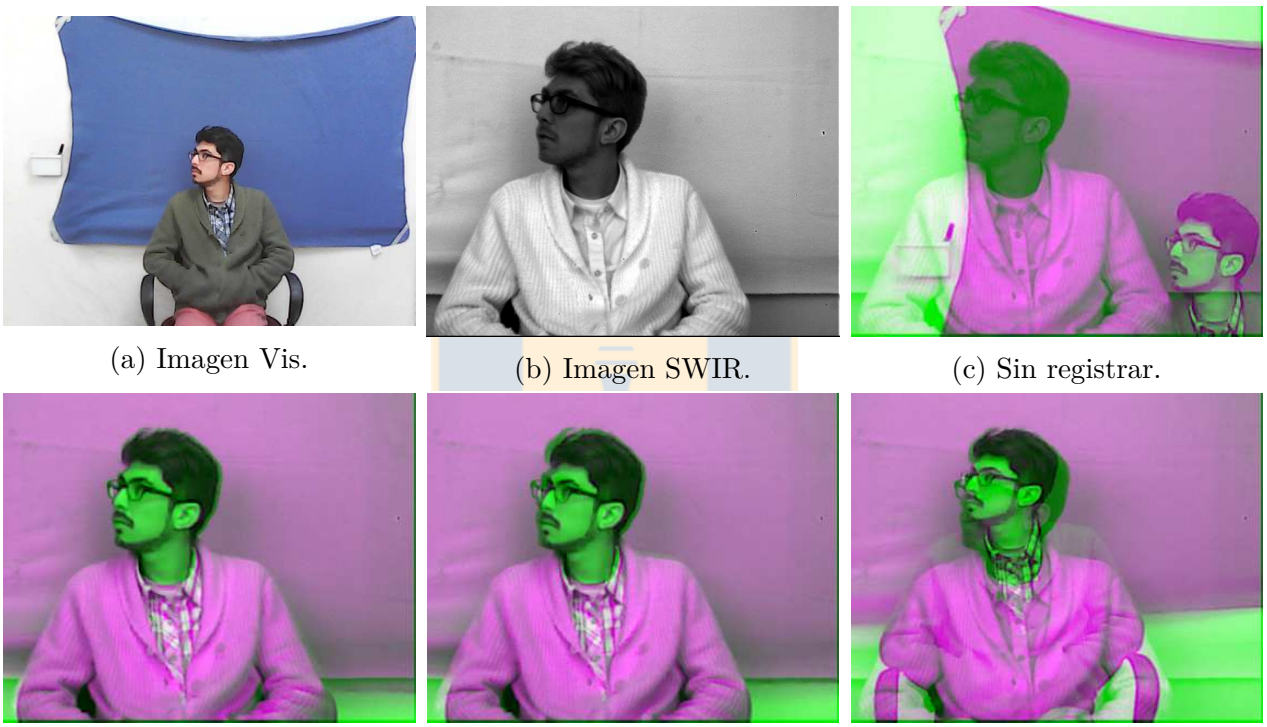
(d) Iteración $k = 1$. (e) Iteración $k = 2$. (f) Iteración $k = 3$.

(g) Iteración $k = 4$. (h) Iteración $k = 5$. (i) Iteración $k = 6$.

Figura 5.8: Registro Vis-SWIR utilizando el algoritmo propuesto, imagen 3. Fuente: Elaboración propia.

5.3.3. Resultados comparación algoritmo

En esta sección se comparan los resultados del registro utilizando el Toolbox de Matlab, el algoritmo propuesto, y el registro de forma manual entre las imágenes anteriormente mostradas sin recortar ni redimensionar dichas imágenes. En las Figuras 5.9, 5.10 y 5.11 se pueden apreciar las diferencias entre el algoritmo de registro y el Toolbox de Matlab para las imágenes anteriormente mencionadas. El Toolbox de Matlab falla en registrar de forma global las imágenes para los tres casos mostrados cuando estas no están recortadas ni redimensionadas, sin embargo logra alinear dónde se encuentra la información aunque con varias deformaciones.



(a) Imagen Vis. (b) Imagen SWIR. (c) Sin registrar.

(d) Registro manual. (e) Registro algoritmo. (f) Registro Toolbox.

Figura 5.9: Comparación registro Vis-SWIR, imagen 1. Fuente: Elaboración propia.

Se calculó el error promedio de transformar el conjunto de 640×512 pixeles entre el algoritmo propuesto, el Toolbox de Matlab y el registro manual para las tres imágenes. Para esto se utilizó la Ecuación 3.21 para determinar las coordenadas del pixel para las matrices de transformación encontradas con los distintos métodos, y se calculó la distancia Euclidiana promedio entre estas coordenadas. De esta forma, el algoritmo propuesto tiene un error promedio de 9 pixeles en comparación con el registro manual de las imágenes, de 42 pixeles con el Toolbox de Matlab, y el error promedio entre el Toolbox de Matlab y el alineamiento manual es de 43 pixeles.

(a) Imagen Vis. (b) Imagen SWIR. (c) Sin registrar.

(d) Registro manual. (e) Registro algoritmo. (f) Registro Toolbox.

Figura 5.10: Comparación registro Vis-SWIR, imagen 2. Fuente: Elaboración propia.

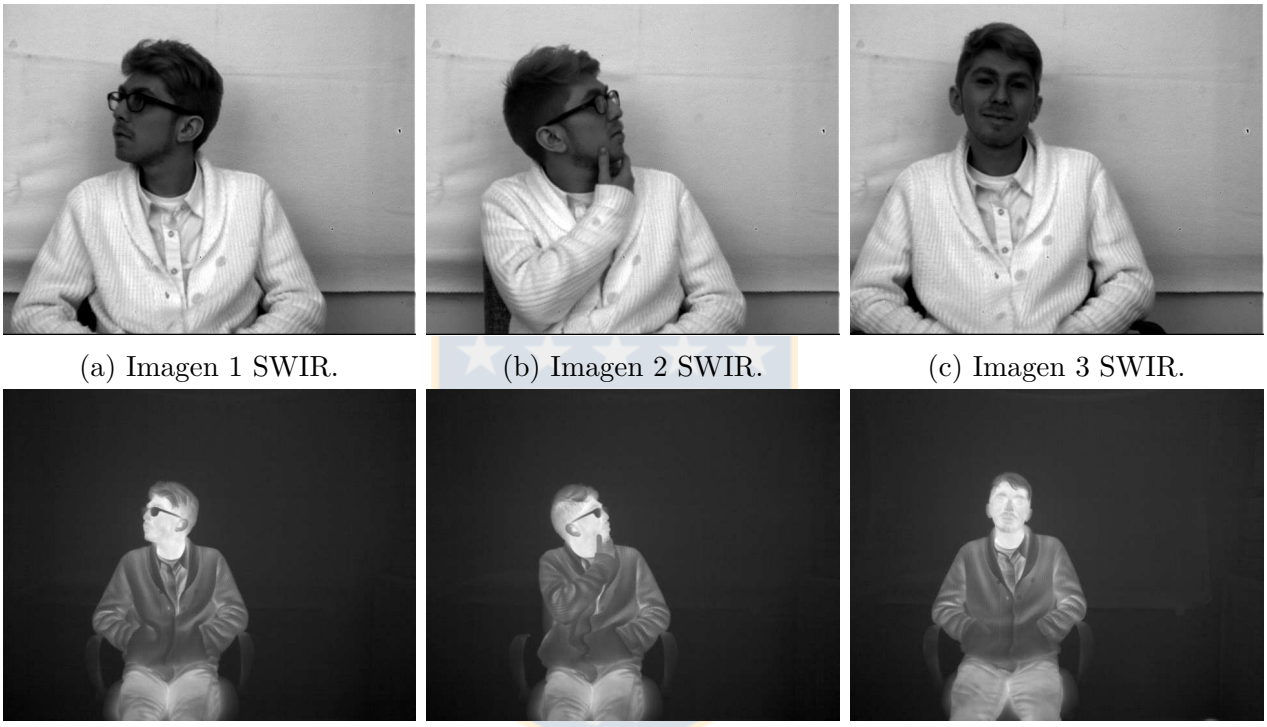
(a) Imagen Vis. (b) Imagen SWIR. (c) Sin registrar.

(d) Registro manual. (e) Registro algoritmo. (f) Registro Toolbox.

Figura 5.11: Comparación registro Vis-SWIR, imagen 3. Fuente: Elaboración propia.

5.4. Registro de imágenes SWIR y LWIR

En esta sección se muestran las pruebas realizadas para el registro SWIR-LWIR. Tres pares de imágenes SWIR-LWIR utilizadas en estas pruebas se muestran en la Figura 5.12. Primero se muestran los resultados del registro de estas imágenes utilizando el Toolbox de Matlab, luego utilizando el algoritmo propuesto, y finalmente una comparación entre los resultados obtenidos.



(a) Imagen 1 SWIR. (b) Imagen 2 SWIR. (c) Imagen 3 SWIR.

(d) Imagen 1 LWIR. (e) Imagen 2 LWIR. (f) Imagen 3 LWIR.

Figura 5.12: Imágenes utilizadas en los experimentos. Fuente: Elaboración propia.

5.4.1. Resultados usando información mutua

Se utilizó el Toolbox de Matlab para hacer el registro entre imágenes SWIR y LWIR. Este Toolbox utiliza la intensidad de los pixeles, MI como medida de similitud, un esquema piramidal, y transformación afín para registrar las imágenes. De forma empírica se determinaron los parámetros de este Toolbox, sin embargo para poder registrar correctamente las imágenes estas debieron ser recortadas y redimensionadas a imágenes de la misma resolución (en este caso 640×512 pixeles), como se muestra en las imágenes SWIR y LWIR de las Figuras 5.13, 5.14 y 5.15.

En las Figuras 5.13 a 5.15 se muestran los resultados de registro SWIR-LWIR utilizando el Toolbox de Matlab. En cada una de estas figuras se muestra el desalineamiento inicial entre las imágenes SWIR y LWIR, y el resultado del registro entre ambas imágenes seleccionando distinta cantidad de iteraciones como parámetro de este Toolbox.

(a) Imagen SWIR. (b) Imagen LWIR. (c) Sin registrar.

(d) Iteración 10. (e) Iteración 50. (f) Iteración 200.

(g) Iteración 500. (h) Iteración $1,000$. (i) Iteración $2,000$.

Figura 5.13: Registro SWIR-LWIR utilizando información mutua, imagen 1. Fuente: Elaboración propia.

En el caso del registro SWIR-LWIR utilizando este Toolbox, se puede apreciar de la Figuras 5.14 que a mayor cantidad de iteraciones el algoritmo del Toolbox converge a una solución de mayor calidad, y ya con $2,000$ iteraciones o más las imágenes están visualmente registradas. Para las imágenes de las Figuras 5.13 y 5.15, el Toolbox de Matlab converge a una solución aceptable con 50 iteraciones del algoritmo, pero con más iteraciones diverge, pero manteniendo

dónde se encuentra la información aunque con deformaciones de escala y rotación.

(a) Imagen SWIR. (b) Imagen LWIR. (c) Sin registrar.

(d) Iteración 10. (e) Iteración 50. (f) Iteración 200.

(g) Iteración 500. (h) Iteración $1,000$. (i) Iteración $2,000$.

Figura 5.14: Registro SWIR-LWIR utilizando información mutua, imagen 2. Fuente: Elaboración propia.

(a) Imagen SWIR. (b) Imagen LWIR. (c) Sin registrar.

(d) Iteración 10. (e) Iteración 50. (f) Iteración 200.

(g) Iteración 500. (h) Iteración $1,000$. (i) Iteración $2,000$.

Figura 5.15: Registro SWIR-LWIR utilizando información mutua, imagen 3. Fuente: Elaboración propia.

5.4.2. Resultados usando algoritmo propuesto

Se programó el Algoritmo 1 en Matlab para hacer el registro entre imágenes SWIR y LWIR. Este algoritmo utiliza HOG como extractor de características, la distancia Chi-cuadrado como medida de similitud, y transformación proyectiva para registrar las imágenes. En las Figuras 5.16 a 5.18 se muestran los resultados de registro SWIR-LWIR utilizando el algoritmo propuesto. En cada una de estas figuras se muestra el desalineamiento inicial entre las imágenes SWIR y LWIR (Para la imagen 1, Figura 5.16a), iteraciones intermedias de registro del algoritmo

(Figuras 5.16b a 5.16h), y el resultado del registro entre ambas imágenes (Figura 5.16i).

(a) Sin registrar. (b) Iteración $k = 1$, $i = 100$. (c) Iteración $k = 1$, $i = 1,000$.

(d) Iteración $k = 1$. (e) Iteración $k = 2$. (f) Iteración $k = 3$.

(g) Iteración $k = 4$. (h) Iteración $k = 5$. (i) Iteración $k = 6$.

Figura 5.16: Registro SWIR-LWIR utilizando el algoritmo propuesto, imagen 1. Fuente: Elaboración propia.

En cada imagen, k corresponde a la iteración actual del proceso de estimación de los parámetros de la transformación proyectiva, e i corresponde a la cantidad de iteraciones del algoritmo de descenso de gradiente para estimar los parámetros de la transformación proyectiva dentro de una iteración k. Las Figuras 5.16b y 5.16c muestran iteraciones intermedias del algoritmo de descenso de gradiente para encontrar los parámetros óptimos dentro de la iteración $k = 1$ de la estimación de los parámetros. Las Figuras 5.16d a 5.16i muestran el resultado del alineamiento con los parámetros determinados en las iteraciones $k = 1$ a $k = 6$, donde la iteración $k = 6$ corresponde al resultado del registro entre ambas imágenes.

(a) Sin registrar. (b) Iteración $k = 1$, $i = 100$. (c) Iteración $k = 1$, $i = 1,000$.

(d) Iteración $k = 1$. (e) Iteración $k = 2$. (f) Iteración $k = 3$.

(g) Iteración $k = 4$. (h) Iteración $k = 5$. (i) Iteración $k = 6$.

Figura 5.17: Registro SWIR-LWIR utilizando el algoritmo propuesto, imagen 2. Fuente: Elaboración propia.

De las Figuras 5.16 y 5.17 se puede apreciar que las imágenes son registradas globalmente de forma satisfactoria, sin embargo en el caso de la Figura 5.18 se puede apreciar una deformación en la parte superior de la imagen registrada, la cual se debe principalmente por la falta de POI en la parte superior de la imagen SWIR. La ausencia de POI en la sección superior de la imagen 3 se debe a que el descriptor necesita de una ventana de 64×64 centrada en dicho punto, es decir existe un borde de 32 pixeles al rededor de los limites de la imagen donde no se pueden tener descriptores, y el contorno de la parte superior de la imagen SWIR para el caso de la Figura 5.18 se encuentra dentro de este borde de 32 pixeles cercano a los limites de la imagen. Debido a esto, la mayor cantidad de puntos de interés en las demás secciones

de la imagen tienen mayor presencia al momento del cálculo de los parámetros óptimos de la transformación, y por consecuencia no puede correguir correctamente el alineamiento de pares de puntos mal emparejados en la periferia superior de la imagen. Aun así, el algoritmo logra registrar satisfactoriamente ambas imágenes de forma global para los tres casos, con una deformación en la parte superior de la imagen registrada en el caso de la imagen 3.

(a) Sin registrar. (b) Iteración $k = 1$, $i = 100$. (c) Iteración $k = 1$, $i = 1,000$.

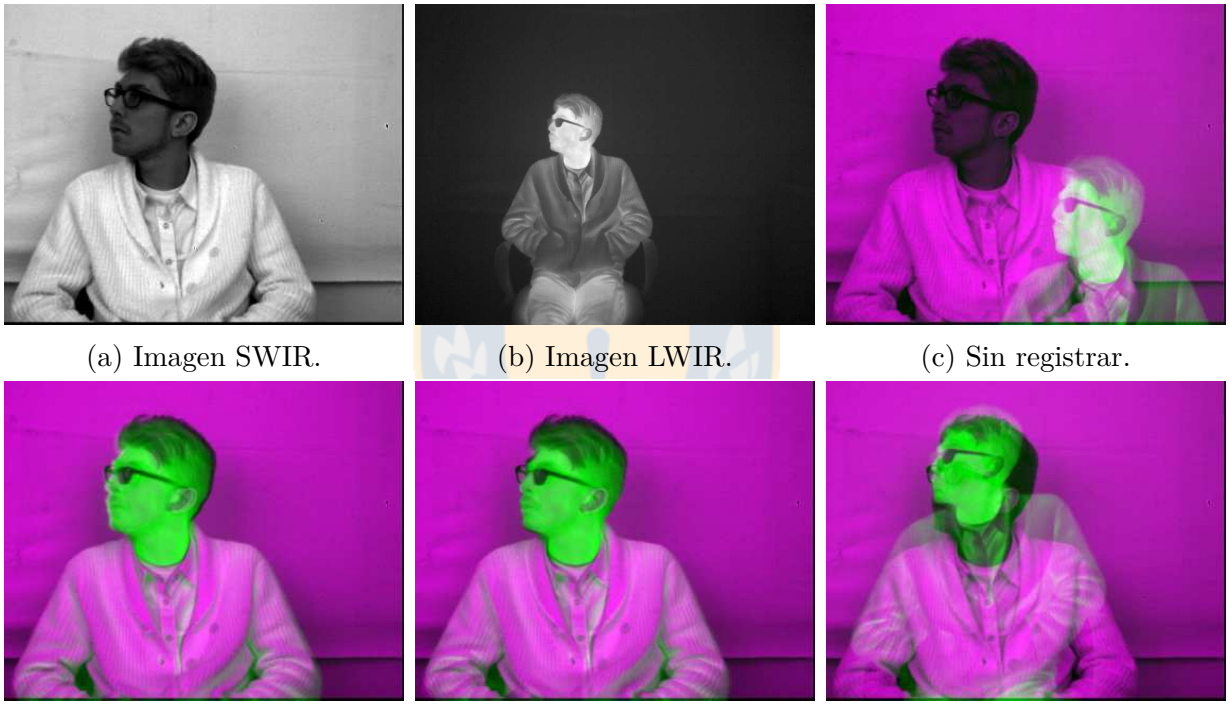
(d) Iteración $k = 1$. (e) Iteración $k = 2$. (f) Iteración $k = 3$.

(g) Iteración $k = 4$. (h) Iteración $k = 5$. (i) Iteración $k = 6$.

Figura 5.18: Registro SWIR-LWIR utilizando el algoritmo propuesto, imagen 3. Fuente: Elaboración propia.

5.4.3. Resultados comparación algoritmo

En esta sección se comparan los resultados del registro utilizando el Toolbox de Matlab, el algoritmo propuesto, y el registro de forma manual entre las imágenes anteriormente mostradas sin recortar ni redimensionar dichas imágenes. En las Figuras 5.19, 5.20 y 5.21 se pueden apreciar las diferencias entre el algoritmo de registro y el Toolbox de Matlab para las imágenes anteriormente mencionadas. El Toolbox de Matlab falla en registrar de forma global las imágenes para los tres casos mostrados cuando estas no están recortadas ni redimensionadas, sin embargo logra alinear dónde se encuentra la información aunque con varias deformaciones. El Toolbox presenta un registro aun más deforme y la información se observa menos centrada en comparación con el registro de imágenes Vis-SWIR utilizando el mismo Toolbox.

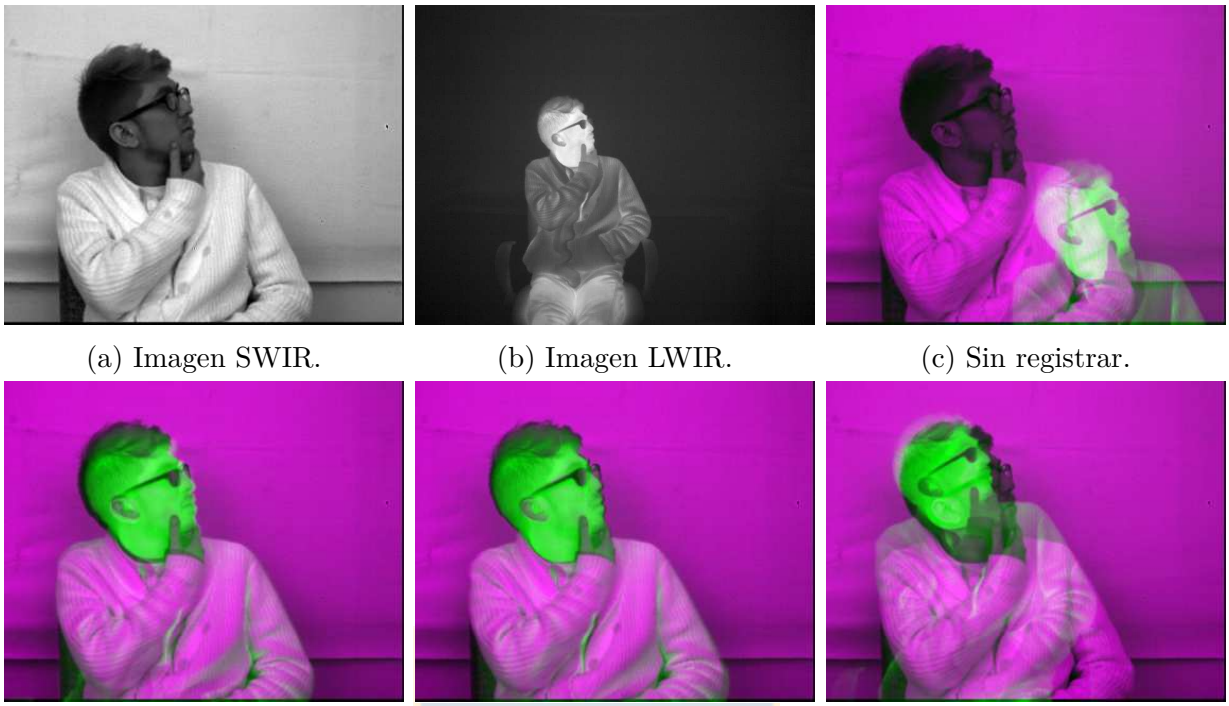


(a) Imagen SWIR. (b) Imagen LWIR. (c) Sin registrar.

(d) Registro manual. (e) Registro algoritmo. (f) Registro toolbox.

Figura 5.19: Comparación registro SWIR-LWIR, imagen 1. Fuente: Elaboración propia.

Se calculó el error promedio de transformar el conjunto de 640×512 pixeles entre el algoritmo propuesto, el Toolbox de Matlab y el registro manual para las tres imágenes. Para esto se utilizó la Ecuación 3.21 para determinar las coordenadas del pixel para las matrices de transformación encontradas con los distintos métodos, y se calculó la distancia Euclidiana promedio entre estas coordenadas. De esta forma, el algoritmo propuesto tiene un error promedio de 16 pixeles en comparación con el registro manual de las imágenes, de 75 pixeles con el Toolbox de Matlab, y el error promedio entre el Toolbox de Matlab y el alineamiento manual es de 70 pixeles.



(a) Imagen SWIR. (b) Imagen LWIR. (c) Sin registrar.

(d) Registro manual. (e) Registro algoritmo. (f) Registro toolbox.

Figura 5.20: Comparación registro SWIR-LWIR, imagen 2. Fuente: Elaboración propia.

(a) Imagen SWIR. (b) Imagen LWIR. (c) Sin registrar.

(d) Registro manual. (e) Registro algoritmo. (f) Registro toolbox.

Figura 5.21: Comparación registro SWIR-LWIR, imagen 3. Fuente: Elaboración propia.

Capítulo 6: Diseño arquitectura hardware

6.1. Descripción general

La Figura 6.1 muestra un diagrama general del montaje propuesto, integrado por las distintas cámaras y sus correspondientes sistemas embebidos, separado por nodos. Cada nodo de este diagrama está compuesto por un dispositivo de adquisición de imágenes (ya sea una cámara Vis, SWIR o LWIR) y un SoC que combina un procesador embebido y lógica reconfigurable para procesar las imágenes capturadas. Los SoCs se comunican entre ellos mediante comunicación Ethernet, y hacia sus respectivas cámaras usando una interfaz FMC CameraLink o USB. En el prototipo, los SoCs Vis y LWIR registran los pares de imágenes Vis-SWIR y SWIR-LWIR respectivamente, y el SoC SWIR combina ambos pares de imágenes en el mapa de referencia común SWIR para desplegar estas imágenes alineadas en un monitor. De esta forma, el algoritmo de registro de imágenes se ejecuta en los nodos Vis y LWIR usando la imagen SWIR, transferida a los respectivos nodos mediante Ethernet, como referencia.

Cada SoC implementa una arquitectura heterogénea, mapeando las operaciones con restricciones de tiempo real en la lógica reconfigurable, y ejecutando las operaciones más irregulares y menos críticas en tiempo de ejecución en el procesador. Se implementó la fase de calibración en software utilizando el procesador embebido, la cual comprende la extracción de características, emparejamiento de puntos y cálculo de parámetros de transformación, y corresponde a las líneas 1-15 del Algoritmo 1. Las demás fases del algoritmo se implementaron en la parte de lógica reconfigurable del SoC, comprendiendo la transformación proyectiva y el cálculo del valor de los pixeles mediante interpolación bilineal, las cuales fueron diseñadas e implementadas como

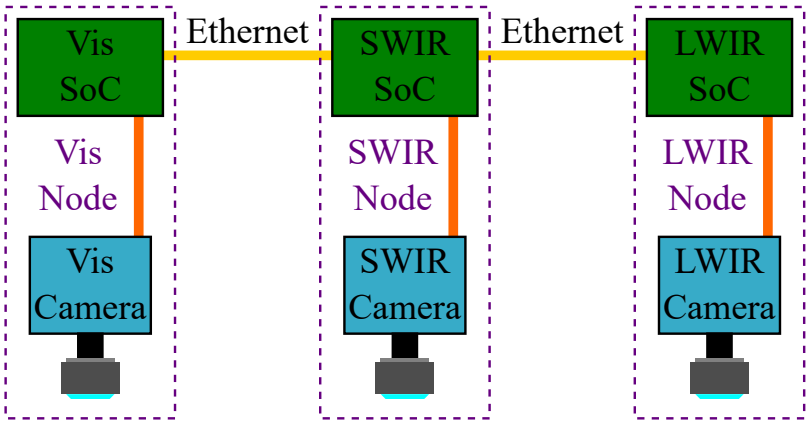


Figura 6.1: Diagrama general del sistema. Fuente: Elaboración propia.

arquitecturas altamente personalizadas y segmentadas (en forma pipeline), correspondiente a los pasos 16-19 del Algoritmo 1.

Las operaciones a ser ejecutadas en la lógica reconfigurable del SoC no poseen un cambio sustancial en el exponente de su representación, por lo cual pueden ser representadas en aritmética de punto fijo. Para esto, primero se realiza un análisis para determinar el largo de palabra necesario para representar los números de las distintas operaciones a ejecutarse dentro del circuito. Posteriormente se describe la arquitectura hardware del sistema y se discuten sus módulos clave.

6.2. Análisis de punto fijo

Para implementar las operaciones en la unidad personalizada de hardware se realizó un análisis del impacto que tienen el tamaño de palabra y la cantidad de bits de representación fraccionaria en la aritmética de punto fijo. Este análisis se realizó para las variables que no poseen un cambio sustancial en el exponente de su representación y pueden ser representadas en punto fijo, lo cual en este caso incluye todas las operaciones a ser implementadas en hardware, es decir, desde que se calculan las coordenadas transformadas de la imagen objetivo hasta que se obtiene la imagen registrada. Para esto se consideraron de forma separada los análisis para las operaciones de la transformación proyectiva, división por Newton-Rhapson, e interpolación bilineal.

6.2.1. Transformación proyectiva

Los parámetros de la transformación proyectiva son calculados en el procesador embebido, luego de lo cual una aproximación hacia aritmética de punto fijo es requerida para cada parámetro debido a su posterior utilización en la unidad hardware para transformar cada coordenada de la imagen objetivo al mapa de referencia. Para determinar la cantidad de bits fraccionarios de estos parámetros se evaluó la precisión en punto fijo para cada uno de los parámetros de forma separada con diferentes niveles de exactitud en cuanto a su representación, comparando el error promedio de aplicar la matriz de transformación $\mathbf{M}$ al conjunto de coordenadas usado por las cámaras del sistema, esto es, coordenadas en el rango 640×512 pixeles. Estas coordenadas pueden ser representadas con 10 bits en el caso del eje x, y con 9 bits para el caso del eje y, pero por simplicidad se representarán ambos en 10 bits.

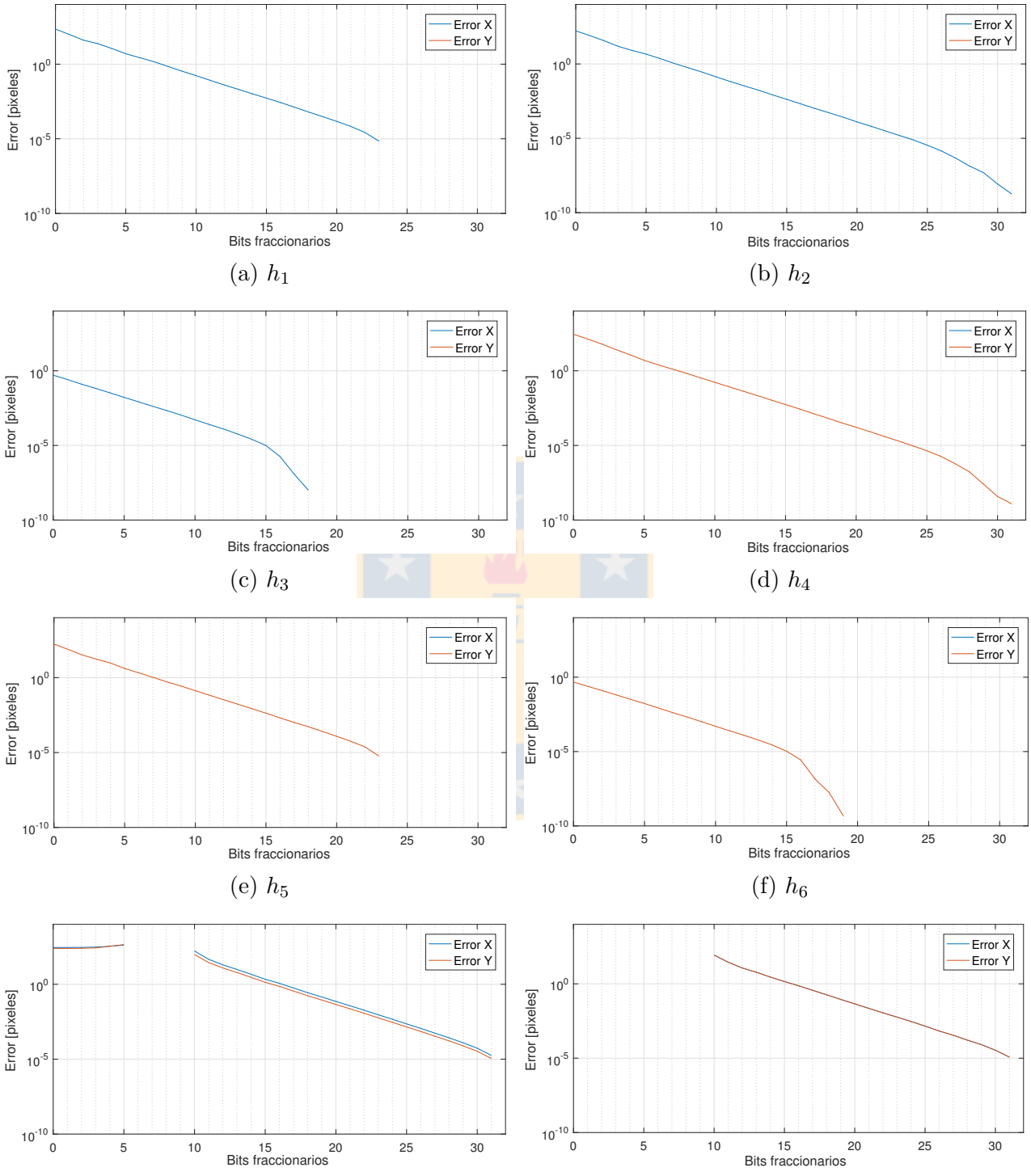
Tabla 6.1: Bits fraccionarios, parámetros transformación proyectiva. Fuente: Elaboración propia.

Parámetro	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8
Bits	21	21	13	21	21	13	30	30

Como conjunto de pruebas se tomaron 101 muestras representativas de los resultados del registro entre 9 pares Vis-SWIR y SWIR-LWIR para los parámetros de la transformación. Incluyendo además la inversa de dichas transformaciones finalmente se obtiene un conjunto de $3,636$ matrices de transformación. Ya que el valor de los pixeles está representado en el peor caso con 16 bits, estos tienen un valor máximo de $65,535$ cuentas digitales. Debido a que en imágenes reales no se espera cambios bruscos entre los valores de pixeles aledaños se considera un cambio máximo del $10\,\%$ de este valor máximo, es decir $6,554$. Con todo lo anterior, se estima que con un error promedio menor a 10^{-4} pixeles en la coordenada calculada, resultante de reducir la cantidad de bits fraccionarios de un parámetro, la precisión restante influye de forma despreciable en el valor de la interpolación del pixel transformado.

En las Figuras 6.2a a 6.2h se muestra el error promedio en pixeles en el valor de las coordenadas calculadas al aplicar la transformación geométrica en el conjunto de coordenadas, con distintos niveles de precisión de bits fraccionarios para cada parámetro. Las leyendas “Error X” y “Error Y” corresponden al error promedio en el cálculo de las coordenadas x e y al aplicar la matriz de transformación.

En la Tabla 6.1 se muestran los bits fraccionarios necesarios de los parámetros de la transformación proyectiva en la Ecuación 3.12 para obtener un error promedio en coordenadas menor a 10^{-4} pixeles con respecto a la representación de todos los parámetros en formato punto flotante con precisión simple. Finalmente, considerando los parámetros con la precisión indicada en la Tabla 6.1 se obtiene un error promedio en coordenadas de 1.3427×10^{-4} pixeles en el eje x y 1.4288×10^{-4} pixeles en el eje y.



(a) h_1 (b) h_2

(c) h_3 (d) h_4

(e) h_5 (f) h_6

(g) h_7 (h) h_8

Figura 6.2: Análisis de punto fijo, transformación proyectiva, parámetros **h**. Fuente: Elaboración propia.

6.2.2. División por Newton-Rhapson

El divisor en el algoritmo de Newton-Rhapson puede ser positivo o negativo dependiendo de los parámetros de la transformación proyectiva obtenidos, donde su valor se encuentra usualmente próximo a 1 de acuerdo a los experimentos realizados, ya que los valores de los parámetros h_7 y h_8 son usualmente muy pequeños para imágenes reales, pues estos controlan la deformación y perspectiva de la imagen transformada. Para facilitar el cálculo del inverso del divisor se considera que el divisor se encuentra d sin signo en el rango $0.5 \leq d < 1$, y por lo tanto su inverso D se encuentra en el rango $2 \geq D > 1$.

Ya que el divisor d multiplicado por su inverso D debe ser 1, denotamos el porcentaje de error del divisor d con respecto a la i-ésima aproximación de su inverso D_i como:

$$e_{d,D_i} = |1 - d \times D_i| \times 100. \tag{6.1}$$

Para apreciar el error en cuentas digitales de la i-ésima aproximación del inverso D_i con respecto al valor real del mismo, expresado en punto flotante y denotado como D_{FULL}, tenemos la ecuación:

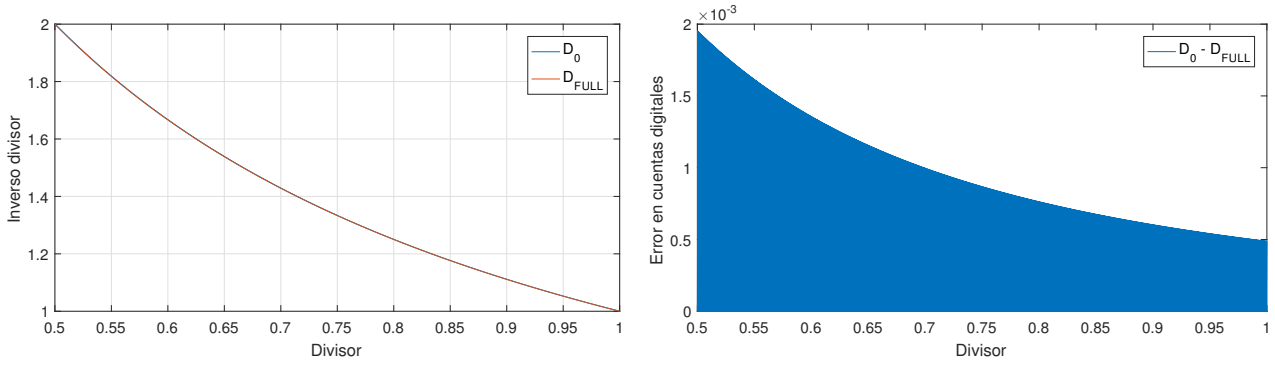
$$e_{D_{FULL},D_i} = |D_i - D_{FULL}| \,. \tag{6.2}$$

Se deben analizar la cantidad de bits que se utiliza en las multiplicaciones de la Ecuación 3.24, la cual puede ser dividida en dos partes:

$$\begin{aligned} D_{i_{PIVOT}} &= 2 - d \times D_i \,, \\ D_{i+1} &= D_i \times D_{i_{PIVOT}} \,, \end{aligned} \tag{6.3}$$

con $D_{i_{PIVOT}}$ un valor auxiliar para implementar con mayor facilidad la Ecuación 3.24 en hardware.

La Figura 6.3a muestra los valores del divisor d en el rango $0.5 \leq d < 1$, su inverso en representación punto flotante D_{FULL}, y una aproximación inicial en punto fijo D_0. D_{FULL} está en el rango $2 \geq D_{FULL} > 1$ con una precisión acotada a 24 bits de parte fraccionaria. Para poder realizar un análisis con estos valores se separó cada valor por 2^{-24} cuentas digitales, obteniendo un conjunto de 2^{23} números para el rango $0.5 \leq d < 1$ con valores entre $2 \geq (D_{FULL}) > 1$. La aproximación inicial D_0 está compuesta por 2^{10} valores en representación punto fijo para este mismo rango, también representada con 24 bits de parte fraccionaria pero separados por 2^{-11} cuentas digitales entre cada valor.



(a) (b)

Figura 6.3: Análisis de punto fijo, división, D_0 y D_{FULL}. Fuente: Elaboración propia.

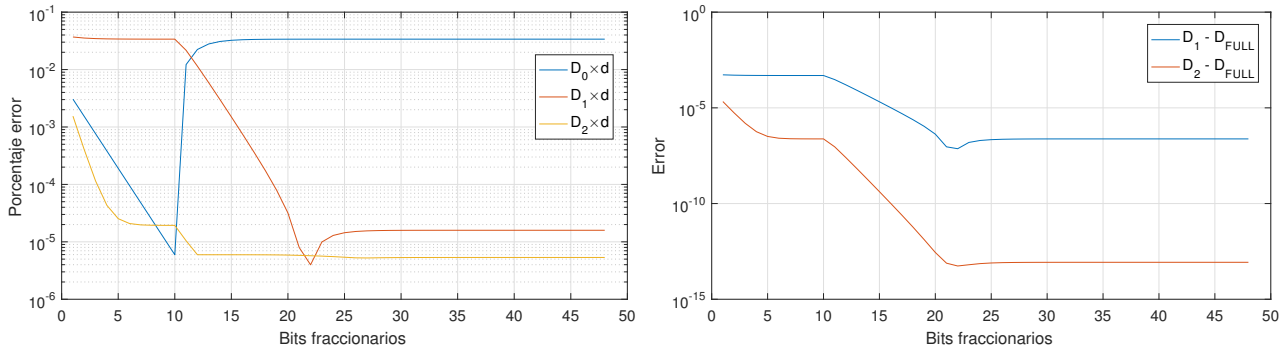
Para poder hacer una comparación 1 a 1 entre D_{FULL} y D_0, es decir con la misma cantidad de valores (2^{23}), cada valor de D_0 se repite 2^{13} veces a forma de obtener una aproximación escalonada de D_{FULL} por secciones. D_0 será utilizado como un banco de valores iniciales para obtener una mejor aproximación de D en la siguiente iteración utilizando el método de Newton-Rhapson y denotado como D_1, de igual forma utilizando D_1 se obtendrá una mejor aproximación D_2, y así sucesivamente hasta llegar a la precisión deseada D_S. Este método tiene la ventaja de que es un algoritmo iterativo y requiere una baja cantidad de valores almacenados como aproximación inicial D_0, pudiendo implementarlo de forma segmentada. En la Figura 6.3b se muestra el error entre D_{FULL} y D_0, el cual es menor a 2×10^{-3} cuentas digitales.

Empíricamente se determinó que con dos iteraciones del algoritmo de Newton-Rhapson se obtiene un error promedio entre D_2 y d menor a 1.2×10^{-14} % (utilizando la aproximación inicial D_0), de esta forma influyendo de forma despreciable en el posterior cálculo de la interpolación. Ya que para cada iteración de Newton-Rhapson se requieren dos multiplicaciones (Ecuación 6.3) se deben realizar 4 análisis de punto fijo, uno por cada multiplicación, con lo cual se tiene el sistema de ecuaciones:

$$\begin{aligned} D_{0_{PIVOT}} &= 2 - d \times D_0 \ , \\ D_1 &= D_0 \times D_{0_{PIVOT}} \ , \\ D_{1_{PIVOT}} &= 2 - d \times D_1 \ , \\ D_2 &= D_1 \times D_{1_{PIVOT}} \ . \end{aligned} \tag{6.4}$$

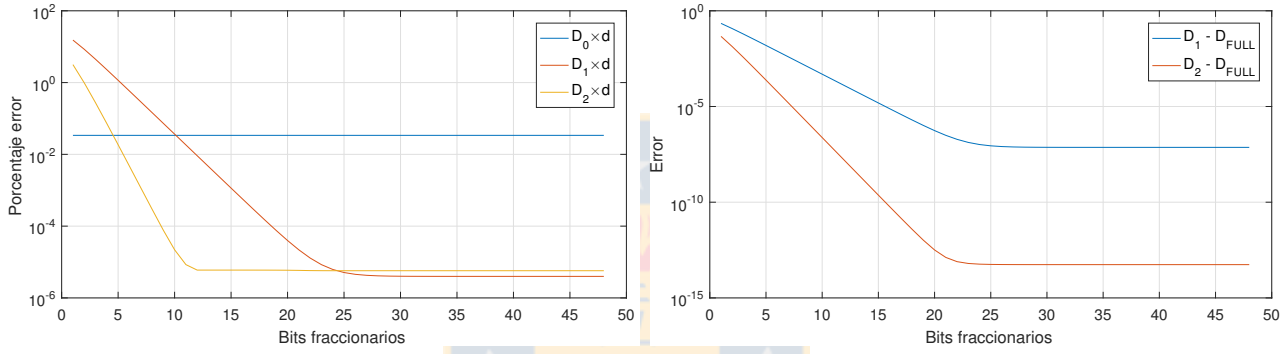
En las Figura 6.4a se observa que con 22 bits fraccionarios para $D_0 \times d$ se obtiene un valor estable en dicha multiplicación, y en la Figura 6.4b se observa un error promedio de 7.3×10^{-8} y 5.4×10^{-14} cuentas digitales para D_1 y D_2 al compararlos con D_{FULL} respectivamente.

La variable intermedia $D_{0_{PIVOT}} = 2 - d \times D_0$ sólo necesita una resta extra por el resultado



(a) (b)

Figura 6.4: Análisis de punto fijo, división, $D_0 \times d$. Fuente: Elaboración propia.



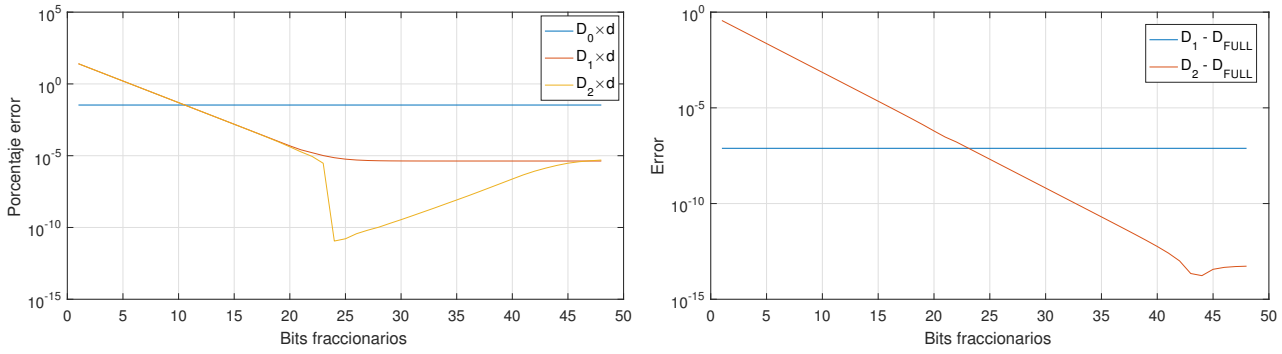
(a) (b)

Figura 6.5: Análisis de punto fijo, división, $D_1 = D_0 \times D_{0_{PIVOT}}$. Fuente: Elaboración propia.

de la multiplicación anterior, $d \times D_0$, y por lo tanto no requiere de análisis de bits fraccionarios. El análisis de la variable $D_1 = D_0 \times D_{0_{PIVOT}}$ se puede desprender de la Figura 6.5a, donde se observa que con 27 bits fraccionarios se obtiene un valor estable en dicha multiplicación. De la Figura 6.5b se observa un error promedio de 7.7×10^{-8} y 5.5×10^{-14} cuentas digitales para D_1 y D_2 al compararlos con D_{FULL} respectivamente, donde al disminuir la precisión en más operaciones el error aumenta levemente.

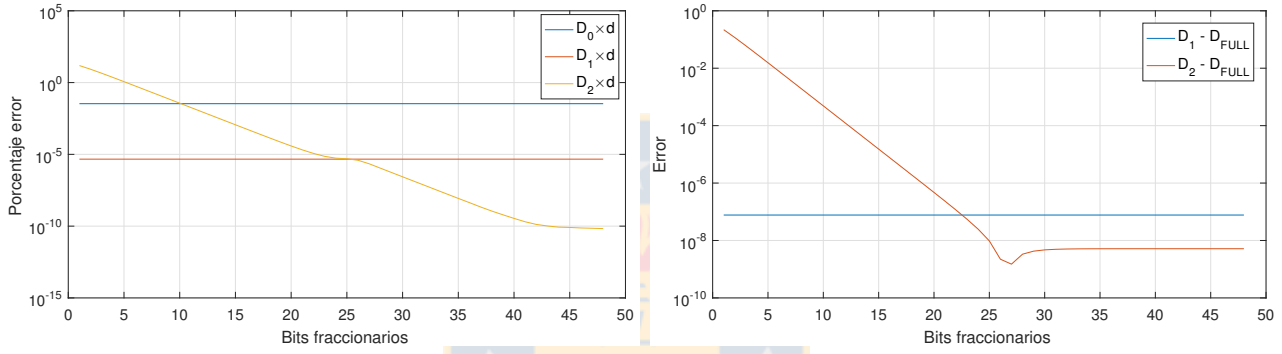
En la Figura 6.6a se observa que con 27 bits fraccionarios para $D_1 \times d$ se obtiene un valor estable en dicha multiplicación, y en la Figura 6.6b se observa un error promedio de 5.1×10^{-9} cuentas digitales para D_2 con respecto a D_{FULL}, y el error de D_1 no cambia pues estas variables analizadas no influyen en su valor. Aún cuando el error de D_2 aumentó considerablemente, aún se encuentra dentro de un margen aceptable para no afectar en gran medida las siguientes operaciones.

De manera similar, después de calcular la variable intermedia $D_{1_{PIVOT}} = 2 - d \times D_1$ se realiza



(a) (b)

Figura 6.6: Análisis de punto fijo, división, $D_1 \times d$. Fuente: Elaboración propia.



(a) (b)

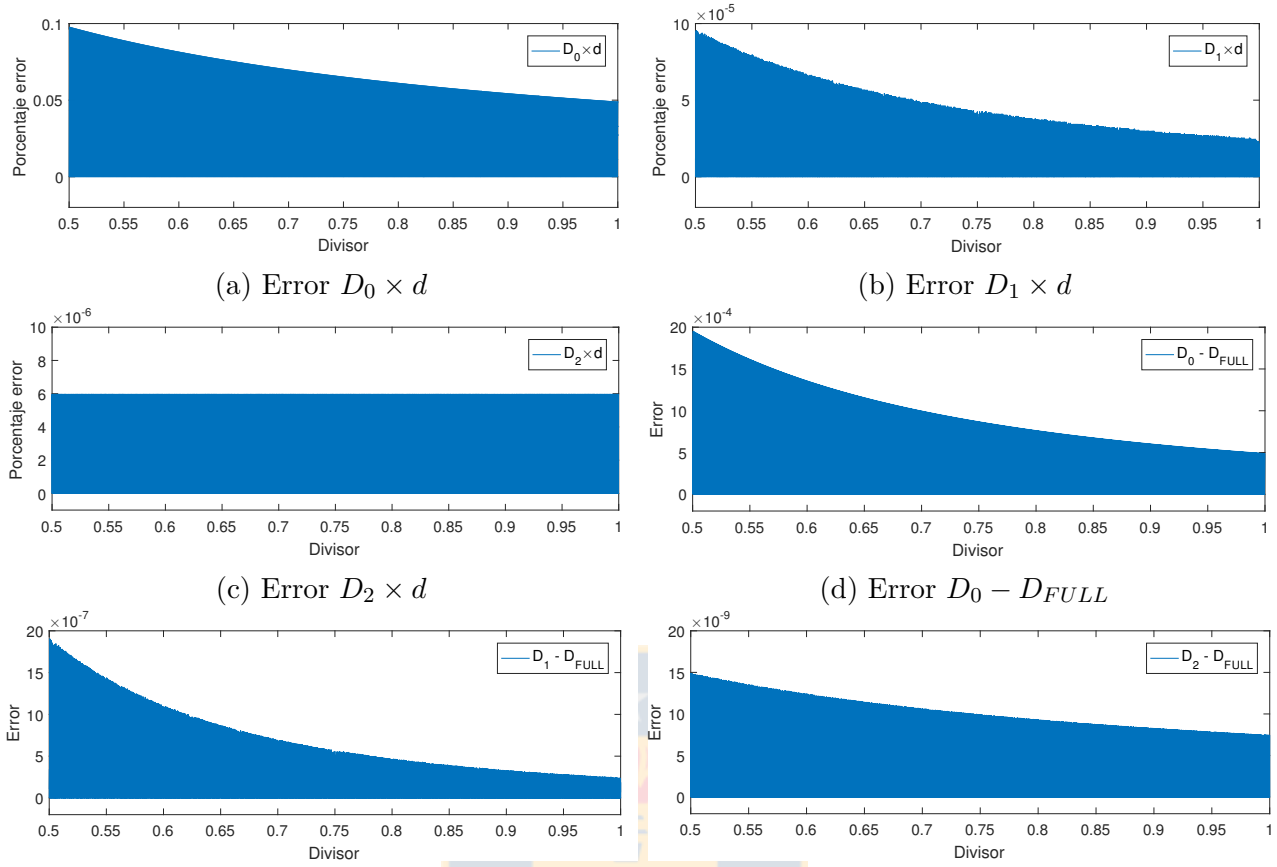
Figura 6.7: Análisis de punto fijo, división, $D_2 = D_1 \times D_{1_{PIVOT}}$. Fuente: Elaboración propia.

el análisis de punto fijo para $D_2 = D_1 \times D_{1_{PIVOT}}$, el cual se puede apreciar en las Figuras 6.7a y 6.7b. De estos gráficos se puede observar que con 27 bits fraccionarios D_2 tiene un error promedio de 1.4×10^{-9} cuentas digitales al compararlo con D_{FULL}. Los resultados de los distintos análisis de bits de punto fijo para la parte fraccionaria necesarios para las diferentes partes de la división por Newton-Rhapson se pueden apreciar en la Tabla 6.2.

Tabla 6.2: Bits fraccionarios, división Newton-Rhapson. Fuente: Elaboración propia.

Operación	$d \times D_0$	$D_0 \times D_{0_{PIVOT}}$	$d \times D_1$	$D_1 \times D_{1_{PIVOT}}$
Bits	22	27	27	27

Finalmente, de las Figuras 6.8a, 6.8b y 6.8c se puede observar que el porcentaje de error para todo el conjunto de datos, calculado con la Ecuación 6.1 en cada caso, es menor a 1×10^{-1} %, 1×10^{-4} % y 6×10^{-6} % para las aproximaciones D_0, D_1 y D_2 al ser multiplicados por el divisor d, respectivamente. De las Figuras 6.8d, 6.8e y 6.8f se puede observar que los errores D_0, D_1



(a) Error $D_0 \times d$

(b) Error $D_1 \times d$

(c) Error $D_2 \times d$

(d) Error $D_0 - D_{FULL}$

(e) Error $D_1 - D_{FULL}$

(f) Error $D_2 - D_{FULL}$

Figura 6.8: Análisis de punto fijo, división, resultado final. Fuente: Elaboración propia.

y D_2 con respecto a D_{FULL} son menores a 2×10^{-3}, 2×10^{-6} y 1.5×10^{-8} cuentas digitales respectivamente.

6.2.3. Interpolación bilineal

Las entradas a las operaciones de interpolación bilineal son valores enteros de pixeles de hasta 16 bits de representación, con lo cual tienen un valor mínimo de 0 cuentas digitales y máximo de $65,535$. Las coordenadas transformadas de los pixeles pueden llevar a un error en la interpolación de los valores dependiendo de la precisión de las coordenadas calculadas, específicamente la parte fraccionaria. De esta forma se usaron las coordenadas fraccionarias de los pixeles con valores entre 0 y 1, con diferentes niveles de precisión variando entre 1 a 25 bits de representación fraccionaria, para obtener el error promedio de los valores de los pixeles interpolados. Se realizó el análisis de punto fijo con un conjunto representativo de valores de pixeles en 16 bits. De la Figura 6.9 se observa que con 20 bits de precisión fraccionaria para

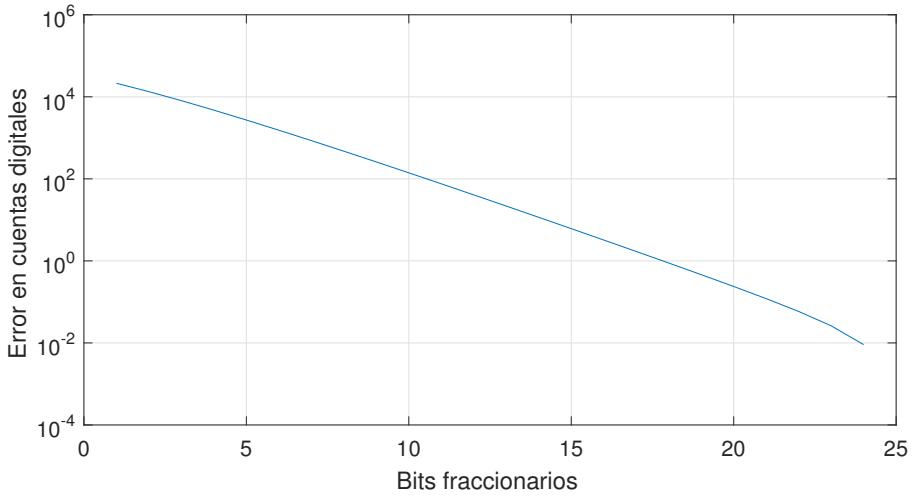


Figura 6.9: Análisis de punto fijo, interpolación bilineal. Fuente: Elaboración propia.

las coordenadas transformadas de los pixeles se obtiene un error promedio menor a una cuenta digital, correspondiente a una unidad en la magnitud del pixel, para el valor del pixel interpolado. Con esto la cantidad de bits para representar las coordenadas de un pixel son $10 + 20 = 30$ bits, donde los 10 bits corresponden a la cantidad de bits para representar una coordenada en el mapa de 640×512 pixeles.

6.3. Arquitectura del sistema

La Figura 6.10 muestra una vista de alto nivel de la arquitectura diseñada para el sistema, separando las fases del algoritmo implementadas en software de las que se implementaron en hardware. Las flechas azules indican el movimiento de datos entre las distintas partes del sistema, mientras que las flechas rojas indican señales de control para la comunicación entre los distintos módulos. El procesador embebido ejecuta el algoritmo de HOG para extraer las características, emparejar los POI mediante la distancia Chi-cuadrado, y calcular los parámetros de la transformación proyectiva. Ya que estas operaciones se realizan en la fase de calibración, no son críticas en el tiempo, son complejas computacionalmente, y requieren altas cantidades de memoria, es más apropiado el implementarlas en software, lo cual significa que se realizarán fuera de línea.

El bloque principal de la unidad de hardware es el núcleo “Image Registration”, el cual está compuesto por un módulo que mapea las coordenadas de la imagen objetivo al sistema de referencia usando los parámetros de la transformación proyectiva calculados por el procesador

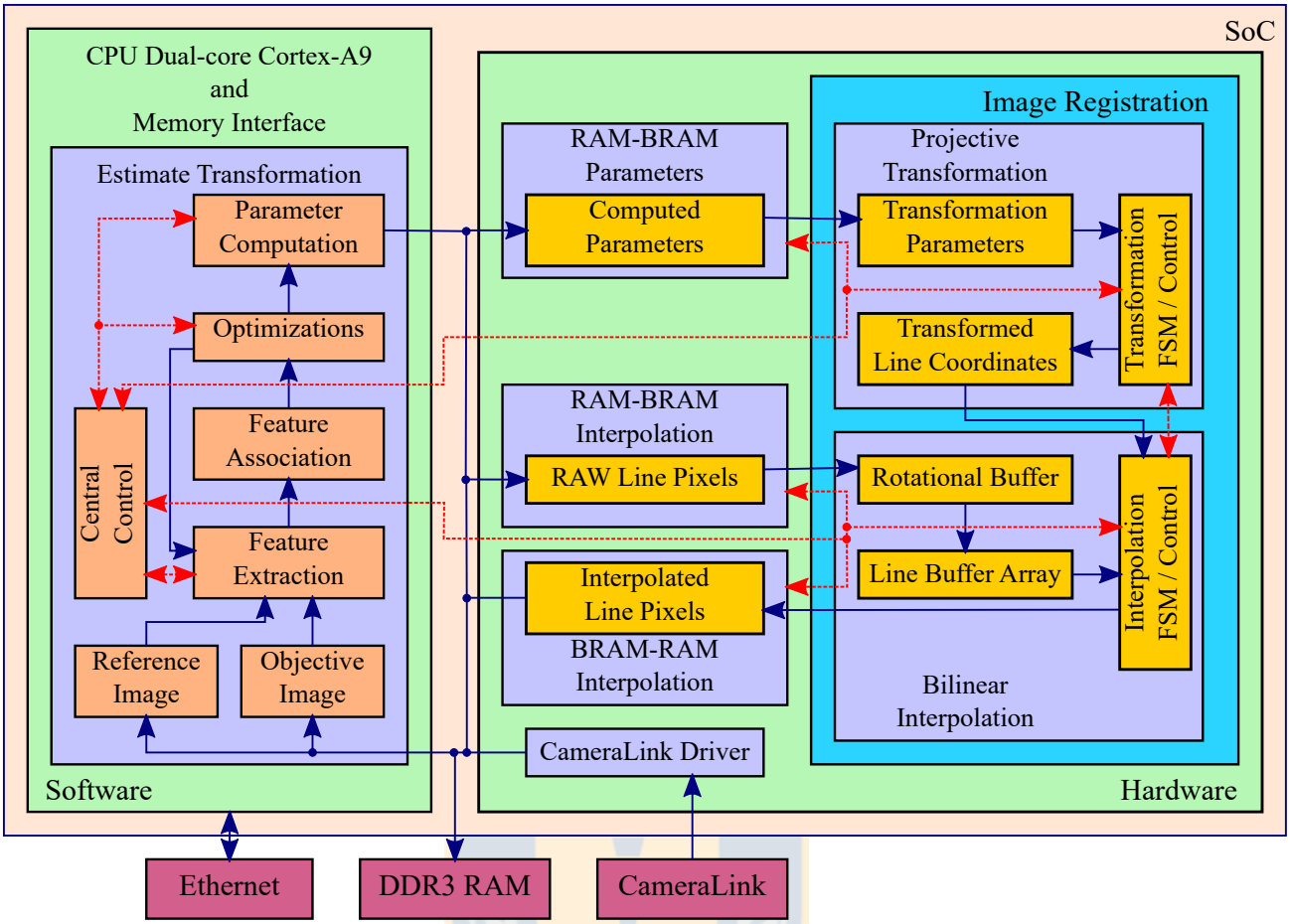


Figura 6.10: Arquitectura general del sistema. Fuente: Elaboración propia.

embebido (“Projective Transformation”), y un módulo que interpola los valores de los pixeles mediante interpolación bilineal (“Bilinar Interpolation”). También hay módulos para transferir datos entre RAM externa y BRAM interna para: leer los valores de los pixeles del cuadro a registrar (“RAM-BRAM Interpolation”); leer los parámetros de la transformación proyectiva calculados por el procesador embebido (“RAM-BRAM Parameters”); escribir los valores de los pixeles ya registrados al nodo SWIR para su visualización (“BRAM-RAM Interpolation”). También hay un driver hardware de comunicación CameraLink (“CameraLink Driver”) que provee los datos de las imágenes obtenidas desde las cámaras hacia el procesador.

Se reservaron 256MB de RAM, desde la dirección 0x10000000 hasta 0x1FFFFFFF, para uso de datos compartidos entre el procesador y la lógica programable, así como para el proceso de calibración. En este espacio de memoria se almacenan: el cuadro SWIR obtenido mediante Ethernet para el proceso de calibración; cuadro actual de la cámara correspondiente al nodo de registro (Vis o LWIR según sea el nodo) tanto para la calibración como para la interpolación y el cuadro ya registrado; parámetros de la transformación proyectiva calculados por software y

las distintas variables necesarias para este cómputo. De esta forma, la comunicación de datos entre software y hardware se realiza a través de RAM. Las señales de control desde la lógica programable al procesador fueron implementadas como interrupciones, y desde el procesador hacia la lógica programable mediante escritura de registros.

6.3.1. Arquitectura transformación proyectiva

El módulo "Projective Transformation" transforma las coordenadas de los pixeles de la imagen objetivo al sistema de coordenadas de la imagen de referencia usando la Ecuación 3.16. El procesador embebido calcula los parámetros de la transformación y almacena estos valores en la RAM externa, usando una aproximación en punto fijo de los valores calculados. El algoritmo de transformación proyectiva implementado en este módulo calcula qué coordenadas de la imagen objetivo p_x y p_y corresponden a las coordenadas de la imagen de referencia g_x y g_y. Esto con motivo de solicitar los 4 valores de los pixeles aledaños a esa coordenada en el mapa objetivo para obtener su valor en el mapa de referencia al realizar la interpolación por el módulo "Bilinear Interpolation".

En la Figura 6.11 se muestra un esquema de las entradas y salidas de este módulo. Estas entradas y salidas están divididas principalmente en tres grupos: "IOA_clk" que es el reloj global utilizado en los distintos módulos; el bus "IOB_Parameters" encargado de comunicarse con el módulo "RAM-BRAM Parameters" para cambiar los parámetros de la transformación; y el bus "IOC_Transformation" que está a la espera de señales de control del módulo "Bilinear Interpolation" para iniciar el proceso de transformación de coordenadas y enviar las coordenadas transformadas de cada línea cuando se soliciten mientras se calculan las coordenadas transformadas de la siguiente linea. Para esto cuenta con dos Buffers de línea dobles, donde en cada Buffer doble se almacenan las coordenadas transformadas correspondientes al eje x e y. Debido a las diferencias entre los lentes de las cámaras, lo cual resulta en distintos campos de visión para ambas cámaras, la imagen de referencia fue escalada para obtener una densidad de información por pixel similar entre ambas cámaras.

En la Figura 6.12 se muestra un esquema de las entradas y salidas del módulo "RAM-BRAM Parameters", compuesto principalmente por cuatro grupos de señales: "IOA_clk"; el bus "IOB_Parameters" para comunicarse con el módulo "Projective Transformation"; el bus "IOD_System_Parameters" compuesto por señales de control para comunicarse con el procesador; y el bus "IOE_RAM" para solicitar los datos desde RAM. Cuando un nuevo set de parámetros se encuentra disponible, el procesador envía una señal "Parameters_ready" que pro-

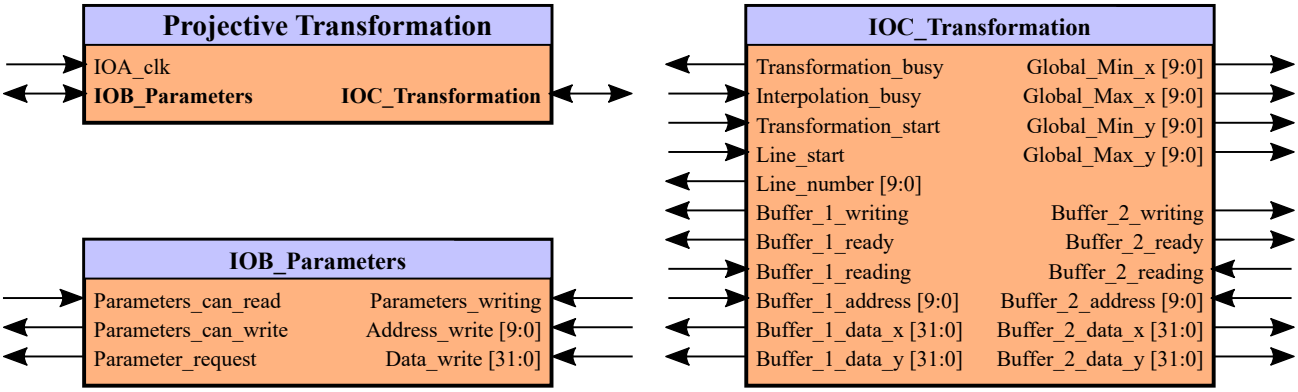


Figura 6.11: Entradas y salidas módulo "Projective Transformation". Fuente: Elaboración propia.

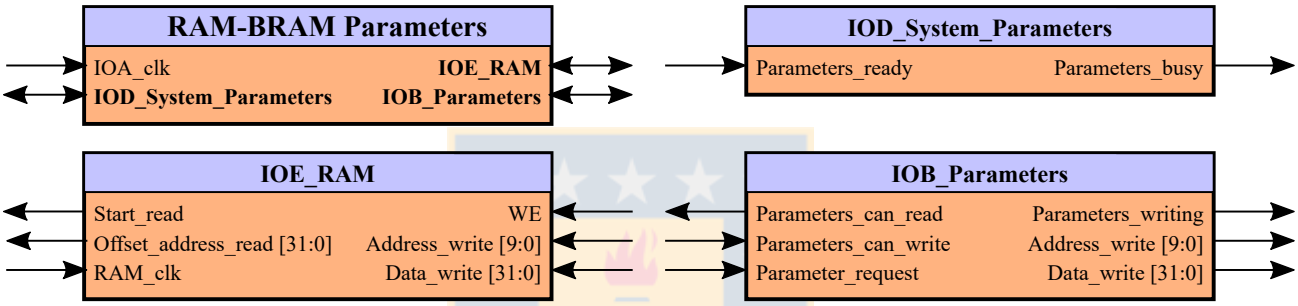


Figura 6.12: Entradas y salidas módulo "RAM-BRAM Parameters". Fuente: Elaboración propia.

voca estos parámetros se lean desde memoria RAM, respondiendo al procesador con una señal "Parameters_busy" para indicar que se están leyendo los parámetros y que el procesador no escriba en esa dirección de memoria. Un proceso de handshaking entre los módulos "Projective Transformation" y "RAM-BRAM Parameters" garantiza que los parámetros sólo serán actualizados cuando estos no están siendo utilizados por el módulo "Projective Transformation", para evadir problemas de consistencia.

La Figura 6.13 muestra las etapas principales del pipeline del módulo "Projective Transformation", donde los registros entre cada fase fueron omitidos para facilitar la comprensión del flujo de datos. En la primera etapa el circuito calcula las multiplicaciones de la Ecuación 3.16, entre los parámetros de la transformación y las coordenadas de la imagen objetivo. En la segunda etapa se calcula el valor del dividendo para p_x y p_y, denotado como n_x y n_y respectivamente, y el valor del divisor común d. Un divisor pipeline de 9 etapas calcula el inverso D_2 de d. En la etapa final, los valores de p_x y p_y se obtienen al multiplicar el inverso del divisor común por los respectivos dividendos n_x y n_y.

El módulo "Projective Transformation" fue diseñado para procesar un flujo continuo de

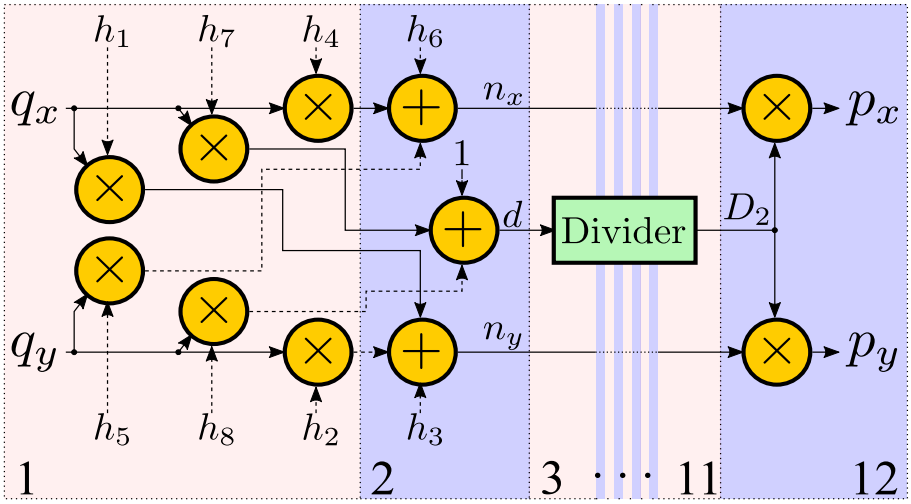


Figura 6.13: Pipeline simplificado, transformación proyectiva. Fuente: Elaboración propia.

coordenadas de pixeles provenientes de la misma línea, con un retardo inicial dado por la latencia del mismo pipeline. También calcula cuantas líneas y columnas en el sistema de coordenadas de la imagen objetivo se necesitan para interpolar cada línea transformada al sistema de referencia para acelerar el proceso de interpolación, esto al sólo solicitar datos que estén dentro de estos limites.

6.3.2. Arquitectura divisor

El divisor de Newton-Rhapson fue diseñado como un circuito segmentado, donde se calcula el inverso del divisor de acuerdo a la regla de iteración de la Ecuación 3.24. Este módulo almacena tanto el signo como los desplazamientos para escalar el valor del divisor al rango sin signo $[0.5, 1[$, reduciendo de esta forma la cantidad de memoria necesaria para la tabla de búsqueda y aumentando la precisión de su aproximación inicial. Cuando el inverso del divisor está calculado, tanto el signo como los desplazamientos son aplicados nuevamente a este inverso para restaurar el número a su rango correcto.

La Figura 6.14 muestra las etapas del divisor segmentado. En la segunda etapa del pipeline del módulo “Projective Transformation”, mostrado en la Figura 6.13, el circuito obtiene el valor del divisor común d, guardando su signo en la etapa 3. En la cuarta etapa se escala el valor de d produciendo un valor normalizado $\hat{d}$, también almacenando la cantidad de desplazamientos para restaurar el número a su rango original. En la siguiente etapa, una tabla de búsqueda entrega la primera aproximación del inverso del divisor $\hat{D}_0$, la cual también está normalizada y sin signo. Las etapas 6 a 9 ejecutan dos iteraciones de la Ecuación 3.24 para obtener aproximaciones

sucesivas del inverso del divisor. Las etapas 10 y 11 restauran el signo y rango original del divisor usando los desplazamientos almacenados, obteniendo finalmente D_2.

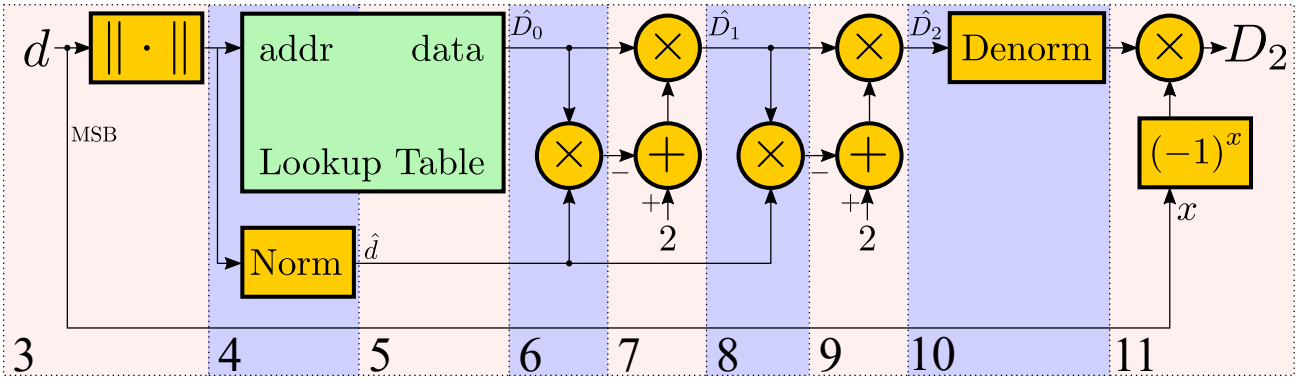


Figura 6.14: Pipeline simplificado, divisor. Fuente: Elaboración propia.

6.3.3. Arquitectura interpolación bilineal

El módulo "Bilinear Interpolation" determina los valores de los pixeles registrados para cada línea de la imagen. El cuadro actual de la imagen, almacenado en la RAM externa, es adquirido usando el módulo de comunicación "RAM-BRAM Interpolation". Estos valores de pixeles son temporalmente almacenados en un "Line Buffer Array" usando las BRAM disponibles en el chip, con lo cual el circuito puede interpolar los pixeles en cuanto se envían las coordenadas transformadas de la línea actual por parte del módulo de transformación. En la Figura 6.15 se muestra un esquema de las entradas y salidas de este módulo, compuesto principalmente por cinco grupos de señales: "IOA_clk"; el bus "IOF_System_Frame" compuesto por señales de control para comunicarse con el procesador, las cuales indican cuando hay un cuadro de imagen a registrar disponible en RAM y avisan al procesador cuando la unidad de hardware está utilizando esta sección de memoria para que no se sobrescriban datos, para lo cual se escribe de forma alternada por parte del procesador en dos direcciones de memoria A y B que indican cuando hay un cuadro nuevo de la imagen objetivo mediante las señales de control "Frame_A_new" y "Frame_B_new" respectivamente; el bus "IOG_BRAM" encargado de comunicarse con el módulo "RAM-BRAM Interpolation" para solicitar nuevas líneas del cuadro actual a RAM; el bus "IOH_Interpolation" encargado de comunicarse con el módulo "BRAM-RAM Interpolation" para escribir en RAM las líneas ya registradas a una dirección fija de memoria y luego enviarlas al nodo SWIR mediante comunicación Ethernet para posteriormente visualizar las imágenes registradas; y el bus "IOC_Transformation" encargado de indicarle al módulo "Projective Transformation" cuando iniciar la transformación de coordenadas y cuando enviar líneas de coordenadas transformadas.

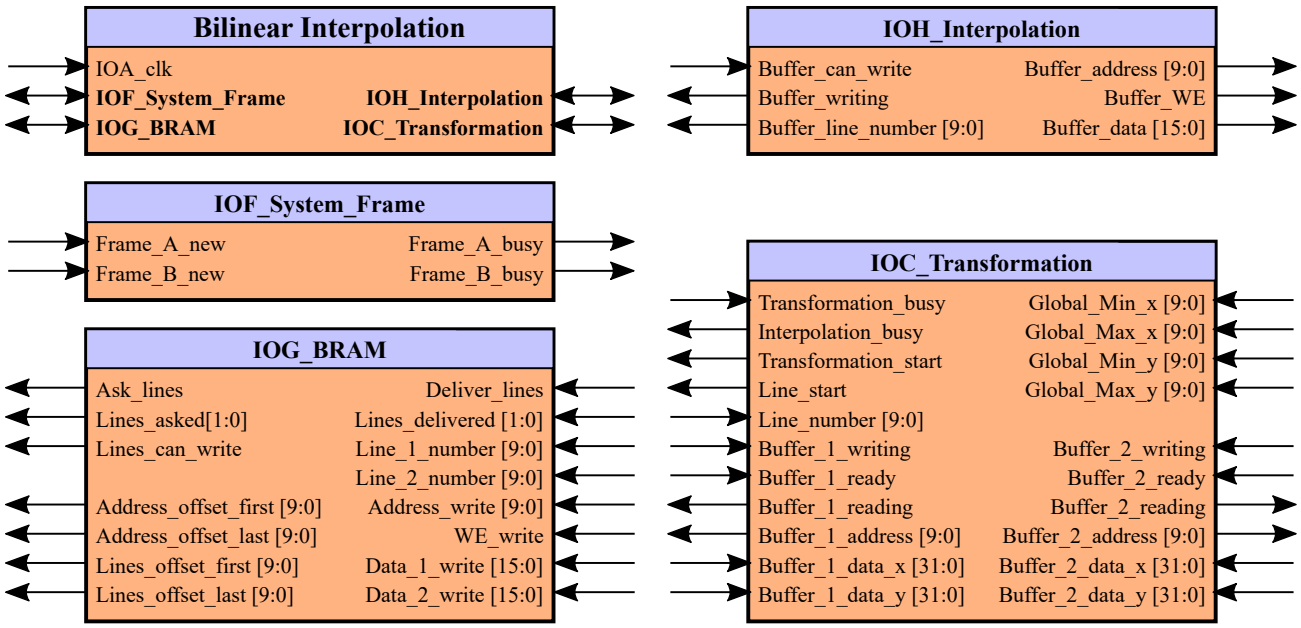


Figura 6.15: Entradas y salidas módulo “Bilinear Interpolation”. Fuente: Elaboración propia.

En la Figura 6.16 se muestra un esquema de las entradas y salidas del módulo “RAM-BRAM Interpolation”. Este módulo está encargado de solicitar líneas de un cuadro de la imagen objetivo, en especifico solicita de a dos líneas para acelerar el proceso de llenado del “Line Buffer Array” del módulo “Bilinear Interpolation”. Este módulo está compuesto principalmente por cinco grupos de señales: “IOA_clk”; el bus “IOI_RAM” encargado de solicitar la primera de dos líneas de la imagen objetivo a RAM; el bus “IOJ_RAM” encargado de solicitar la segunda de dos líneas de la imagen objetivo a RAM; el bus “IOG_BRAM” encargado de comunicarse con el módulo “Bilinear Interpolation” para enviar las líneas solicitadas por este; y el bus “IOF_System_Frame” para recibir las señales de control “Frame_A_busy” y “Frame_B_busy” y sólo solicitar líneas a RAM cuando se está trabajando en alguno de los cuadros de forma valida.

En la Figura 6.17 se muestra un esquema de las entradas y salidas del módulo “BRAM-RAM Interpolation”, compuesto principalmente por cuatro grupos de señales: “IOA_clk”; el bus “IOC_Transformation” para sólo escribir líneas en RAM cuando el proceso de transformación de coordenadas se inició con éxito; el bus “IOH_Interpolation” encargado de comunicarse con el módulo “Bilinear Interpolation” y esperar a señales y datos que indiquen se deben escribir líneas a RAM; y el bus “IOK_RAM” encargado de enviar las líneas registradas del cuadro actual de la imagen objetivo a RAM.

La Figura 6.18 muestra las etapas principales del pipeline del módulo “Bilinear Interpolation”. En la primera etapa se solicitan las coordenadas transformadas de los pixeles p_x y p_y del

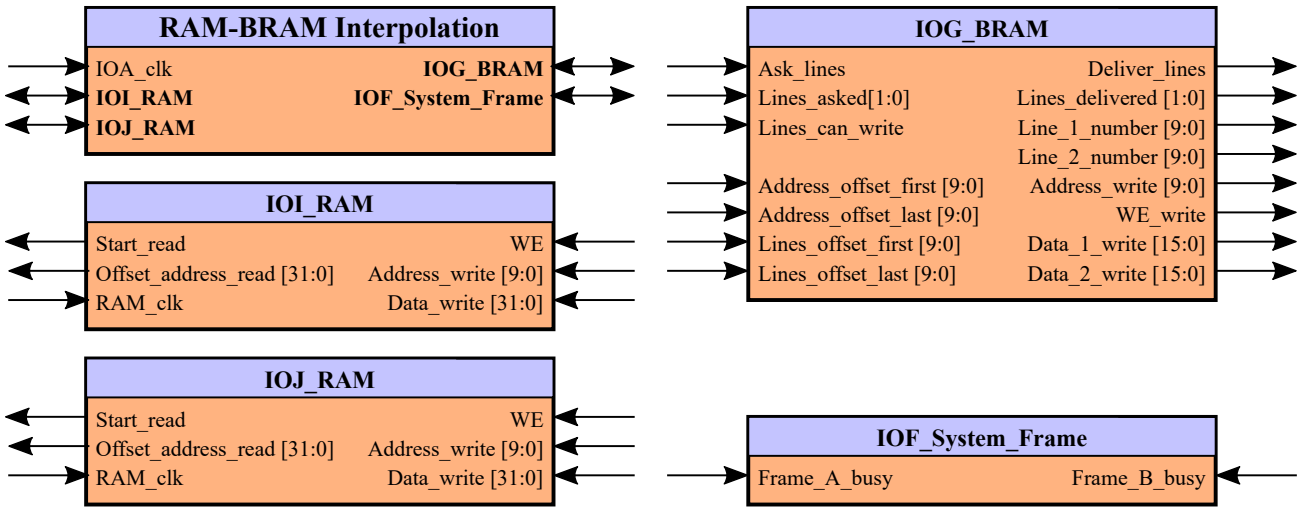


Figura 6.16: Entradas y salidas módulo "RAM-BRAM Interpolation". Fuente: Elaboración propia.

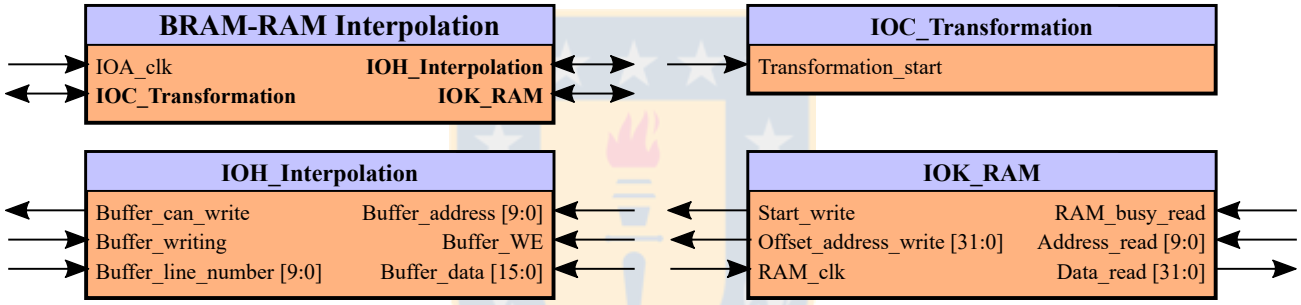


Figura 6.17: Entradas y salidas módulo "BRAM-RAM Interpolation". Fuente: Elaboración propia.

mapa objetivo correspondientes a las coordenadas del sistema de referencia g_x y g_y al módulo "Projective Transformation", las cuales son enviadas en la segunda etapa. Siguiendo la Ecuación 3.25, los valores de los cuatro pixeles más cercanos a las coordenadas transformadas p_x y p_y son solicitados en la etapa 3 desde las líneas de la imagen objetivo almacenadas en el "Line Buffer Array", los cuales son adquiridos en la etapa 4. Las respectivas contribuciones de estos pixeles solicitados al valor interpolado son determinadas en las etapas 3 y 4. Las etapas 5 a 7 calculan las operaciones restantes de la Ecuación 3.25 para obtener el valor interpolado del pixel.

La rotación entre las imágenes a registrar determina cuántas líneas son necesarias para interpolar una línea de la imagen objetivo al sistema de coordenadas de referencia. El número de líneas en el "Line Buffer Array" se limitó a 60 en el diseño debido a limitaciones en los recursos BRAM disponibles. Esta cantidad de líneas corresponde aproximadamente a 5° de rotación para imágenes de 640×512 pixeles. Ya que el montaje fue diseñado para alinear horizontalmente las

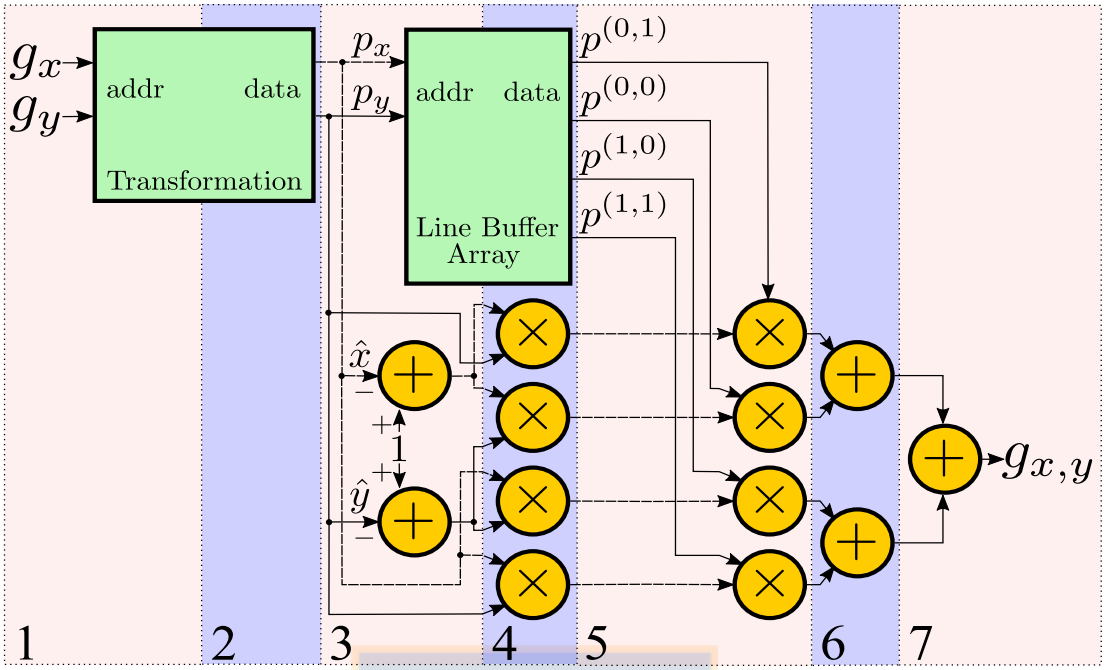


Figura 6.18: Pipeline simplificado, interpolación bilineal. Fuente: Elaboración propia.

cámaras, esta tolerancia en la rotación es suficiente para nuestra aplicación.

Cada línea en el "Line Buffer Array" se separó en direcciones de memoria pares e impares, de esta forma facilitando la segmentación de las operaciones al permitir la lectura de dos datos por línea de forma simultanea. Al principio del proceso de interpolación de un cuadro, si el módulo "Projective Transformation" no está modificando los parámetros de la transformación, se realiza un proceso de handshaking mediante las señales de control del bus "IOC_Transformation" para bloquear modificaciones en estos parámetros hasta que la interpolación del cuadro actual esté completada. Luego, se llena el "Line Buffer Array" con las primeras 60 líneas a ser usadas en la interpolación, de esta forma estos pixeles pueden ser reemplazados cuando no se necesiten a medida que las líneas del cuadro actual son procesadas. El módulo de interpolación fue diseñado para realizar cálculos en un flujo continuo de pixeles para la línea actual, con un retardo inicial dado por la propia latencia del pipeline.

Capítulo 7: Resultados implementación

7.1. Descripción de la tarjeta de desarrollo

Se utilizó una tarjeta de desarrollo ZedBoard, que contiene un SoC Zynq-7000 XC7Z020 de Xilinx. Esta tarjeta contiene lo necesario para implementar el prototipo, tal como: comunicación Ethernet para comunicar las tarjetas; interfaz FMC para conectar las cámaras SWIR y LWIR a través de comunicación CameraLink; memoria RAM externa; y salida VGA para el despliegue de las imágenes registradas. Las principales características de esta tarjeta son:

- 512 MB de memoria RAM DDR3.
- Interfaz Gigabit Ethernet.
- Interfaz tarjeta SD.
- Interfaz FMC.
- Salida HDMI (1080p) y VGA (8-bit).
- 8 switch, 8 LEDs, 5 botones push.

El SoC Zynq-7000 XC7Z020 contiene un procesador Dual-core ARM Cortex-A9 de 667MHz y lógica programable, utilizando buses de comunicación AXI para comunicación entre el procesador y la lógica programable. La lógica programable es equivalente a la familia Artix-7, la cual cuenta con las siguientes características:

- 53,200 LUTs.
- 220 DSP.
- 140 Block RAMs de 36Kb (4.9 Mb total).
- 106,400 Flip-Flops.

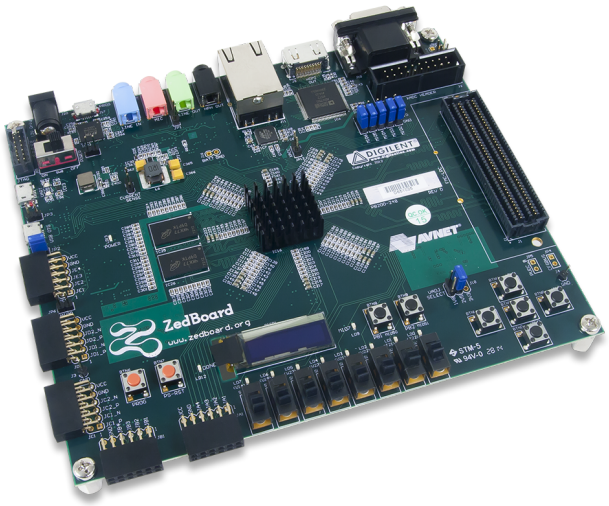


Figura 7.1: Tarjeta de desarrollo ZedBoard. Fuente: https://www.amazon.com/dp/B0797P9F1Z.

7.2. Descripción del sistema

Se diseñó el montaje experimental de la Figura 7.2. Los nodos Vis, SWIR y LWIR contienen una tarjeta de desarrollo ZedBoard Z-7020 con un SoC XC7Z020, el cual lee cuadros de imágenes desde las cámaras usando una interfaz FMC CameraLink. También intercambian datos de imágenes a través de Ethernet. Se implementó de forma independiente la fase de calibración y el proceso de registro cuadro a cuadro que se realiza en los nodos Vis y LWIR.

El nodo Vis está conectado a una Webcam con una resolución de 640×480 pixeles. El nodo SWIR está conectado a una cámara Snake SW OEM de Sofradir, la cual entrega una imagen monomodal en el rango 0.9µm a 1.7µm, con una resolución de 640×512 pixeles. El nodo LWIR usa una cámara térmica Tau 2 de FLIR, operando entre 7.5µm y 13.5µm con una resolución de 640×512 pixeles. Debido a diferencias en los sistemas ópticos de ambas cámaras, el nodo SWIR escala la imagen a 377×302 pixeles antes de transferir dicha imagen a los nodos LWIR.

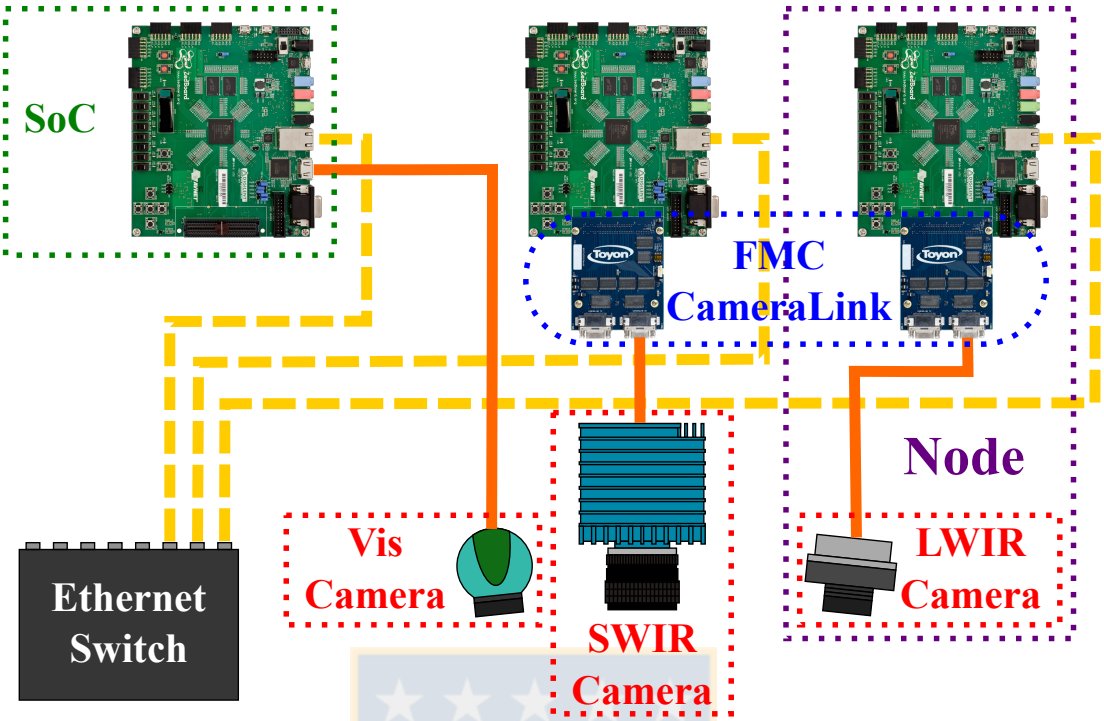


Figura 7.2: Montaje del sistema. Fuente: Elaboración propia.

7.3. Resultados

El núcleo hardware registra las imágenes, correctamente compensando rotaciones, escalamiento, traslación y diferencias en perspectiva. El tiempo de la fase de calibración depende de la cantidad de POI detectados en cada imagen. Por motivos de memoria se limitó la cantidad de POI a un número máximo de 1,024 para la imagen objetivo y de referencia, con lo cual el procesador embebido toma aproximadamente 5 minutos para determinar los parámetros de la matriz de transformación proyectiva que registra la imagen objetivo al mapa de referencia.

El camino crítico de la arquitectura hardware se puede encontrar en el pipeline del divisor, en la última multiplicación que calcula el inverso del divisor D_2 y almacena su valor en un registro. El mínimo periodo de reloj alcanzado fue de 12ns, lo cual corresponde a una frecuencia máxima de reloj de 66.6MHz. El tiempo requerido para registrar cada cuadro depende de la transformación respectiva calculada entre ambas cámaras, ya que esto controla la cantidad de datos usados de la imagen objetivo, y cuán seguido se debe acceder la RAM externa. Sin embargo, en todos los experimentos el tiempo de procesamiento del núcleo hardware para un cuadro fue menor a 15ms y generalmente rondando los 3ms a una resolución de 640×512. Esto significa que a esta resolución las imágenes pueden ser registradas a más de 60 cuadros por

Tabla 7.1: Potencia dinámica del sistema. Fuente: Elaboración propia.

Sección	Potencia [mW]
Procesador	1,541
DSP	18
BRAM	85
Lógica	29
Señales	43
Relojes	15
Total	1,730

Tabla 7.2: Potencia dinámica según módulo. Fuente: Elaboración propia.

Módulo	Potencia [mW]
Registration Core	180
Bilinear Interpolation	134
Projective Transformation	40
RAM-BRAM Interpolation	4
BRAM-RAM Interpolation	2
RAM-BRAM Parameters	<1

segundo, permitiendo operación a la tasa de adquisición de imágenes de la cámara inteligente, posterior a la fase de calibración de ambas cámaras, la cual toma menos de 5 minutos.

El prototipo consume 1.888W de potencia, de los cuales 1.730W son potencia dinámica y 0.159W estática. El núcleo de registro consume 180mW, de los cuales 134mW se deben al módulo de interpolación bilineal y 40mW al módulo de transformación proyectiva. El resto de la potencia es utilizada en las varias interfaces de comunicación y en los recursos de generación y distribución de reloj. En la Tabla 7.1 se muestra el consumo de potencia dinámica del sistema separada por recursos utilizados, y en la Tabla 7.2 se puede apreciar la potencia consumida por el módulo “Registration Core” y los módulos que lo componen. De estas tablas se puede apreciar que la mayor cantidad de potencia consumida se debe al módulo “Bilinear Interpolation” por la cantidad de BRAM usadas en el “Line Buffer Array” donde se almacenan las líneas de la imagen objetivo que son utilizadas para la interpolación.

Tabla 7.3: Utilización de recursos. Fuente: Elaboración propia.

Módulo	LUTs	Registers	BRAM	DSP
Bilinear Interpolation	5,928	3,537	68	9
Projective Transformation	1,382	1,747	3	36
RAM-BRAM Interpolation	171	176	2	1
BRAM-RAM Interpolation	153	83	2	1
RAM-BRAM Parameters	61	58	0	0
Disponibles Sistema	53,200	106,400	140	220
Utilizados Registration Core	7,675	5,601	75	47
Utilizados Sistema	9,714	7,788	75	47
Porcentaje Registration Core	14.43 %	5.26 %	53.57 %	21.36 %
Porcentaje Sistema	18.26 %	7.32 %	53.57 %	21.36 %

En la Tabla 7.3 se muestran los recursos utilizados por el sistema y el núcleo de registro, separado por los módulos que lo componen. El sistema utiliza alrededor del 20 % de los recursos lógicos disponibles y cerca del 55 % de las BRAMs disponibles, estas últimas mayormente utilizadas en el “Line Buffer Array”.

Capítulo 8: Conclusiones

8.1. Sumario

En este trabajo se presentó un algoritmo de registro multimodal entre imágenes Vis, SWIR y LWIR, realizando el registro entre los pares Vis-SWIR y SWIR-LWIR. El algoritmo utiliza HOG como extractor de características de POI que se encuentran en el borde de los objetos, la distancia Chi-cuadrado como medida de similitud para emparejar estos puntos entre los pares de imágenes Vis-SWIR y SWIR-LWIR, y transformación proyectiva en conjunto con interpolación bilineal para registrar los pares de imágenes.

Se diseñó una arquitectura heterogénea que ejecuta el algoritmo de registro compuesta en dos fases: una calibración inicial y un proceso de registro cuadro a cuadro. La fase de calibración es ejecutada en el procesador embebido de la tarjeta de desarrollo e incluye la extracción y emparejamiento de características, utilizando esta información para calcular los parámetros de la transformación proyectiva que registra cada par de imágenes respectivamente. El proceso de registro cuadro a cuadro fue mapeado a una unidad de hardware segmentada en la lógica programable de la arquitectura, la cual aplica la transformación proyectiva, utilizando los parámetros calculados en software, a cada cuadro de la imagen objetivo para determinar sus coordenadas en el mapa de referencia, y utiliza interpolación bilineal para determinar el valor en cuentas digitales de los pixeles registrados.

El montaje experimental propuesto está compuesto por tres nodos, donde cada nodo a su vez está compuesto por una cámara (Vis, SWIR o LWIR) y una unidad de procesamiento, ejecutando el algoritmo de registro multimodal de imágenes Vis-SWIR y SWIR-LWIR de forma distribuida en los nodos Vis y LWIR respectivamente. Se implementó de forma separada la fase de calibración y registro cuadro a cuadro de un nodo en el SoC Zynq XC7Z020 de Xilinx. La fase de calibración implementada en el procesador embebido requiere de aproximadamente 5 minutos para determinar los parámetros de la transformación proyectiva. La implementación del núcleo hardware requiere menos de 15ms para procesar imágenes de 640×512 pixeles a una frecuencia de reloj de 66.6MHz, usando el 20 % de los recursos lógicos y cerca del 55 % de los recursos de memoria del SoC, permitiendo el procesamiento a la tasa de adquisición de imágenes. El sistema consume 1.888W de poder, de los cuales 1.541W corresponden a la potencia consumida por el procesador y sólo 0.180W al núcleo de registro implementado en hardware.

8.2. Conclusiones

Los experimentos realizados muestran que el algoritmo propuesto es capaz de registrar los pares de imágenes Vis-SWIR y SWIR-LWIR utilizando el proceso de calibración inicial de forma satisfactoria. Para alinear correctamente los POI con el algoritmo diseñado, los objetos utilizados en la calibración deben poder ser visualizados tanto en la imagen objetivo como en la imagen de referencia, debe existir una mínima o nula rotación entre ellas, y el contorno de los objetos utilizados debe encontrarse fuera del borde de 32 pixeles aledaños a los bordes de las imágenes.

Los resultados de la implementación hardware del algoritmo de registro muestran que el sistema puede operar sobre los 60 cuadros por segundo, posterior a la calibración inicial en el procesador embebido. La parte del algoritmo implementada en el procesador embebido es de baja complejidad, sin embargo requiere una cantidad de memoria que no está disponible en la lógica programable del SoC. El sistema permite integrar más unidades de procesamiento de imágenes al SoC, ya que el núcleo hardware sólo utiliza el 20 % de los recursos lógicos del sistema embebido, siempre que estas unidades no requieran altas cantidades de memoria de la unidad hardware, pues el sistema también utiliza cerca del 55 % de la memoria interna disponible en este SoC. En términos de consumo de potencia, el núcleo hardware consume cerca del 10 % de la potencia disipada por el sistema completo, donde la mayor parte es utilizada por el procesador embebido, lo cual da un amplio margen de consumo energético para estas unidades que pueden agregarse para procesamiento de imágenes.

8.3. Trabajo futuro

Ya que en este trabajo sólo se pudieron realizar las pruebas de la fase de calibración usando el procesador embebido y del núcleo hardware como entes independientes, sin comunicación entre ellas, y de la misma forma, sin comunicación entre los distintos nodos, queda como trabajo futuro el realizar la comunicación entre ellos y completar el prototipo. Las arquitecturas de los distintos nodos diseñadas en este trabajo son lo suficientemente similares como para que la implementación del sistema distribuido completo utilizando la arquitectura diseñada sea posible con leves modificaciones.

Como trabajo futuro adicional se propone ahondar más en el algoritmo en búsqueda de optimizaciones de la fase de calibración que permitan su implementación de forma parcial o total en la lógica programable, principalmente en el uso de memoria requerido por esta fase, como

por ejemplo disminuir la cantidad de POI que estén muy cerca entre si y no entreguen mayor información para determinar los parámetros de la transformación. Otra opción es buscar una alternativa a la fase de calibración con el mismo objetivo de implementarla junto con la fase de registro cuadro a cuadro en la lógica programable. En ambos casos, la arquitectura del núcleo hardware puede ser integrada sin problemas a estos futuros proyectos con modificaciones mínimas en torno a la comunicación con el procesador embebido. También queda pendiente utilizar ambos procesadores en el caso de este SoC, lo cual posibilitaría el funcionamiento continuo de la fase de cálculo de parámetros y recepción/envío de cuadros de video desde/hacia el nodo SWIR mediante comunicación Ethernet.

Bibliografía

[1] W. Ou and C. Chefd'Hotel, "Polynomial intensity correction for multimodal image registration," in *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, June 2009, pp. 939–942.

[2] M. T. Hossain, G. Lv, S. W. Teng, G. Lu, and M. Lackmann, "Improved symmetric-sift for multi-modal image registration," in *2011 International Conference on Digital Image Computing: Techniques and Applications*, Dec 2011, pp. 197–202.

[3] J. Rühaak, L. König, M. Hallmann, N. Papenberg, S. Heldmann, H. Schumacher, and B. Fischer, "A fully parallel algorithm for multimodal image registration using normalized gradient fields," in *2013 IEEE 10th International Symposium on Biomedical Imaging*, April 2013, pp. 572–575.

[4] J. P. W. Pluim, J. B. A. Maintz, and M. A. Viergever, "Image registration by maximization of combined mutual information and gradient information," *IEEE Transactions on Medical Imaging*, vol. 19, no. 8, pp. 809–814, Aug 2000.

[5] S. Yu, X. Liu, and W. Chen, "Ct and mr multimodal registration guided by weighted gradient images," in *2011 4th International Congress on Image and Signal Processing*, vol. 2, Oct 2011, pp. 1099–1103.

[6] I. Reducindo, E. Arce-Santana, D. U. Campos-Delgado, and F. Vigueras-Gomez, "Non-rigid multimodal image registration based on local variability measures and optical flow," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Aug 2012, pp. 1133–1136.

[7] K. Yang, L. Karlstrom, L. C. Smith, and M. Li, "Automated high-resolution satellite image registration using supraglacial rivers on the greenland ice sheet," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 3, pp. 845–856, March 2017.

[8] L. G. Brown, "A survey of image registration techniques," *ACM Comput. Surv.*, vol. 24, no. 4, pp. 325–376, Dec. 1992.

[9] B. Zitová and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, no. 11, pp. 977–1000, 2003.

[10] Y. Yang, "Non-rigid image registration for visible color and thermal ir face," in *2016 International Conference on Audio, Language and Image Processing (ICALIP)*, July 2016, pp. 279–284.

[11] D. A. Socolinsky, A. Selinger, and J. D. Neuheisel, "Face recognition with visible and thermal infrared imagery," *Computer Vision and Image Understanding*, vol. 91, no. 1–2, pp. 72 – 114, 2003, special Issue on Face Recognition.

[12] F. Nicolo and N. A. Schmid, "Long range cross-spectral face recognition: Matching swir against visible light images," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1717–1726, Dec 2012.

[13] T. Bourlai and B. Cukic, "Multi-spectral face recognition: Identification of people in difficult environments," in *2012 IEEE International Conference on Intelligence and Security Informatics*, June 2012, pp. 196–201.

[14] M. Hess, F. Kuester, and M. Trivedi, "Multimodal registration of high-resolution thermal image mosaics for the non-destructive evaluation of structures," in *2014 IEEE International Conference on Imaging Systems and Techniques (IST) Proceedings*, Oct 2014, pp. 216–221.

[15] Y. Xia, S. R. Qu, and L. Cai, "Multi-object tracking based on thermal-visible video sequence fusion," in *Proceedings of the 32nd Chinese Control Conference*, July 2013, pp. 3685–3690.

[16] A. Torabi, G. Massé, and G. A. Bilodeau, "Feedback scheme for thermal-visible video registration, sensor fusion, and people tracking," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, June 2010, pp. 15–22.

[17] A. Leykin, Y. Ran, and R. Hammoud, "Thermal-visible video fusion for moving target tracking and pedestrian classification," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, June 2007, pp. 1–8.

[18] S. E. Godoy, M. M. Hayat, D. A. Ramirez, S. A. Myers, R. S. Padilla, and S. Krishna, "Detection theory for accurate and non-invasive skin cancer diagnosis using dynamic thermal imaging," *Biomed. Opt. Express*, vol. 8, no. 4, pp. 2301–2323, Apr 2017.

[19] F. Inostroza, J. Cárdenas, S. E. Godoy, and M. Figueroa, "Embedded multimodal registration of visible images on long-wave infrared video in real time," in *2016 Euromicro Conference on Digital System Design (DSD)*, Aug 2016, pp. 176–183.

[20] *Colorimetric and spectroradiometric characteristics of narrow-field-of-view clear skylight in Granada, Spain*, vol. 18, no. 2, Febrero 2001.

[21] *Labial teeth and gingiva color image segmentation for gingival health-state assessment*, Amsterdam (Netherlands), 2012.

[22] B. S. Manjunath, R. Chellappa, and C. von der Malsburg, "A feature based approach to face recognition," in *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun 1992, pp. 373–378.

[23] Y. Adini, Y. Moses, and S. Ullman, "Face recognition: the problem of compensating for changes in illumination direction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 721–732, Jul 1997.

[24] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, Julio 1997.

[25] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, Feb 2009.

[26] R. S. Ghiass, O. Arandjelović, A. Bendada, and X. Maldague, "Infrared face recognition: A comprehensive review of methodologies and databases," *Pattern Recognition*, vol. 47, no. 9, pp. 2807 – 2824, 2014.

[27] J. HASHAGEN, "Swir applications and challenges: A primer," *EuroPhotonics*, pp. 26–29, Septiembre 2014.

[28] . Ervik, S. M. Hellesø, S. T. Munkejord, and B. Müller, "Experimental and computational studies of water drops falling through model oil with surfactant and subjected to an electric field," in *2014 IEEE 18th International Conference on Dielectric Liquids (ICDL)*, June 2014, pp. 1–6.

[29] T. Bourlai, N. Narang, B. Cukic, and L. Hornak, "On designing a swir multi-wavelength facial-based acquisition system," *Proc. SPIE*, vol. 8353, pp. 83 530R–83 530R–14, 2012.

[30] J. Dowdall, I. Pavlidis, and G. Bebis, "Face detection in the near-ir spectrum," *Image and Vision Computing*, vol. 21, no. 7, pp. 565 – 578, 2003, computer Vision beyond the visible spectrum.

[31] A. S. Nunez and M. J. Mendenhall, "Detection of human skin in near infrared hyperspectral imagery," in *IGARSS 2008 - 2008 IEEE International Geoscience and Remote Sensing Symposium*, vol. 2, July 2008, pp. II–621–II–624.

[32] J. K. Delaney, E. Walmsley, B. H. Berrie, and C. F. Fletcher, "Multispectral imaging of paintings in the infrared to detect and map blue pigments," in *(Sackler NAS Colloquium) Scientific Examination of Art: Modern Techniques in Conservation and Analysis*. Washington, DC: The National Academies Press, 2005, pp. 120–136.

[33] P. Ricciardi, J. K. Delaney, M. Facini, and L. Glinsman, "Use of imaging spectroscopy and in situ analytical methods for the characterization of the materials and techniques of 15th century illuminated manuscripts," *Journal of the American Institute for Conservation*, vol. 52, no. 1, pp. 13–29, 2013.

[34] B. Lahiri, S. Bagavathiappan, T. Jayakumar, and J. Philip, "Medical applications of infrared thermography: A review," *Infrared Physics and Technology*, vol. 55, no. 4, pp. 221 – 235, 2012.

[35] J. E. Soto and M. Figueroa, "An embedded face-classification system for infrared images on an fpga," *Proc. SPIE*, vol. 9249, 2014.

[36] S. E. Godoy, D. A. Ramirez, S. A. Myers, G. von Winckel, S. Krishna, M. Berwick, R. S. Padilla, P. Sen, and S. Krishna, "Dynamic infrared imaging for skin cancer screening," *Infrared Physics and Technology*, vol. 70, pp. 147 – 152, 2015, proceedings of International Conference on Quantum Structures Infrared Photodetectors, 2014.

[37] L. Arias, D. Sbarbaro, and S. Torres, "Removing baseline flame’s spectrum by using advanced recovering spectrum techniques," *Appl. Opt.*, vol. 51, no. 25, pp. 6111–6116, Sep 2012.

[38] P. Viola and W. M. Wells, "Alignment by maximization of mutual information," in *Proceedings of IEEE International Conference on Computer Vision*, Jun 1995, pp. 16–23.

[39] B. A. Ardekani and A. Bachman, "A new image similarity measure with reduced sensitivity to interpolation and generalizability to multispectral image registration," in *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, Aug 2006, pp. 3053–3056.

[40] H. mei Chen, P. K. Varshney, and M. A. Slamani, "On registration of regions of interest (roi) in video sequences," in *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance, 2003.*, July 2003, pp. 313–318.

[41] C. Harris and M. Stephens, "A combined corner and edge detector," in *In Proc. of Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[42] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ser. ICCV '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 1150–1157.

[43] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*, ser. ECCV'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 404–417.

[44] D. Firmenich, M. Brown, and S. Süsstrunk, "Multispectral interest points for rgb-nir image registration," in *2011 18th IEEE International Conference on Image Processing*, Sept 2011, pp. 181–184.

[45] K. Hajebi and J. S. Zelek, "Dense surface from infrared stereo," in *Applications of Computer Vision, 2007. WACV '07. IEEE Workshop on*, Feb 2007, pp. 21–21.

[46] F. Barrera, F. Lumbreras, and A. D. Sappa, "Multimodal template matching based on gradient and mutual information using scale-space," in *2010 IEEE International Conference on Image Processing*, Sept 2010, pp. 2749–2752.

[47] J. Woo, M. Stone, and J. L. Prince, "Multimodal registration via mutual information incorporating geometric and spatial context," *IEEE Transactions on Image Processing*, vol. 24, no. 2, pp. 757–769, Feb 2015.

[48] Y. Shi, Y. Yuan, X. Zhang, and Z. Liu, "Markov random field model based multimodal medical image registration," in *2012 IEEE 11th International Conference on Signal Processing*, vol. 1, Oct 2012, pp. 697–702.

[49] H. Lu, P. C. Cattin, L. P. Nolte, and M. Reyes, "Diffusion weighted imaging distortion correction using hybrid multimodal image registration," in *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, March 2011, pp. 594–597.

[50] A. Wolf, J. E. Pezoa, and M. Figueroa, "Modeling and compensating temperature-dependent non-uniformity noise in ir microbolometer cameras," *Sensors*, vol. 16, no. 7, 2016.

[51] R. Redlich, L. Araneda, A. Saavedra, and M. Figueroa, "An embedded hardware architecture for real-time super-resolution in infrared cameras," in *2016 Euromicro Conference on Digital System Design (DSD)*, Aug 2016, pp. 184–191.

[52] B. A. White, "Using fpgas to perform embedded image registration," Major thesis, University of Central Florida, Orlando, Florida, 2009.

[53] R. Berg, L. König, J. Rühaak, R. Lausen, and B. Fischer, "Highly efficient image registration for embedded systems using a distributed multicore dsp architecture," *Journal of Real-Time Image Processing*, pp. 1–21, 2014.

[54] C. Bourrasset, L. Maggiani, J. Sérot, F. Berry, and P. Pagano, "Dreamcam: A fpga-based platform for smart camera networks," in *2013 Seventh International Conference on Distributed Smart Cameras (ICDSC)*, Oct 2013, pp. 1–2.

[55] Tetracam Inc., "Macaw (MCAW) tetracam's multiple camera array wireless system," accedido en Abril de 2017. [Online]. Available: http://www.tetracam.com/Products-Macaw.htm

[56] FLIR Systems, "FLIR Duo compact dual-sensor thermal imager for drones," accedido en Abril de 2017. [Online]. Available: http://www.flir.com/suas/duo/

[57] Xenics Infrared Solutions, "Tigris-640 cooled midwave infrared camera for scientific and industrial applications," accedido en Abril de 2017. [Online]. Available: http://www.xenics.com/es/camera/tigris-640

[58] FluxData Inc., "FD-3SWIR," accedido en Abril de 2017. [Online]. Available: http://www.fluxdata.com/products/fd-3swir

[59] PIXELTEQ Inc., "PixelCam OEM multispectral cameras," accedido en Abril de 2017. [Online]. Available: https://pixelteq.com/pixelcam/

[60] E. Haber and J. Modersitzki, *Beyond Mutual Information: A Simple and Robust Alternative*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 350–354.

[61] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, ser. CVPR '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 886–893.

[62] E. Haber and J. Modersitzki, *Intensity Gradient Based Registration and Fusion of Multimodal Images*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 726–733.

[63] J. Modersitzki, *FAIR: Flexible Algorithms for Image Registration*. 3600 Market Street, Floor 6, Philadelphia, PA 19104: Society for Industrial and Applied Mathematics (SIAM), 2009.

[64] O. Pele and M. Werman, "The quadratic-chi histogram distance family," in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 749–762.

[65] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, NY, USA: Cambridge University Press, 2003.

[66] N. M. Nenadic and S. B. Mladenovic, "Fast division on fixed-point dsp processors using newton-raphson method," in *EUROCON 2005 - The International Conference on Çomputer as a Tool"*, vol. 1, Nov 2005, pp. 705–708.