Universidad de Concepción
Dirección de Postgrado
Facultad de Ingeniería - Programa de Magíster en Ciencias de la
Ingeniería con mención en Ingeniería Eléctrica

Corrección de no uniformidad, super resolución y estabilización de escena en imágenes microscópicas infrarrojas

Tesis para optar al grado de Magíster en Ciencias de la Ingeniería con mención en Ingeniería Eléctrica

WLADIMIR ELÍAS VALENZUELA FUENTEALBA
CONCEPCIÓN-CHILE
2016

Profesor Guía: Dr. Miguel Figueroa Toro
Dpto. de Ingeniería Eléctrica, Facultad de Ingeniería
Universidad de Concepción

Universidad de Concepción
Facultad de Ingeniería
Departamento de Ingeniería Eléctrica

Profesor Patrocinante:
Dr. Miguel Figueroa Toro

Corrección de no uniformidad, super resolución y estabilización de escena en imágenes microscópicas infrarrojas

Wladimir Elías Valenzuela Fuentealba

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

Concepción, Chile
Agosto de 2016

Lisa! no puedes llegar tan lejos para luego dar marcha atrás.
Homero J. Simpson

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

Resumen

Las tecnologías de captura de imágenes microscópicas infrarrojas están basadas en un arreglo de plano focal infrarrojo (IRFPA) controlado por un circuito integrado de lectura. Debido a limitaciones en el proceso de fabricación, el IRFPA sufre de dos problemas que afectan severamente la calidad de las imágenes obtenidas: no uniformidad y baja resolución. La no uniformidad es predominada por ruido de patrón fijo, que se superpone sobre la imagen real de la escena capturada alterando la fidelidad de la información. La baja resolución limita la cantidad de detalle apreciable en los cuadros capturados y la información posible de obtener. Además, movimientos involuntarios espaciales impiden un estudio de forma correcta. Estos problemas tienen solución sobre aplicaciones en software, con un alto costo energético y económico.

Este trabajo describe la implementación un circuito digital que procesa el video de un microscopio infrarrojo long-wavelength infrared (LWIR), para corregir la no uniformidad presente, aumentar la resolución y estabilizar los movimientos espaciales. Para los dos primeros se utilizó un algoritmo de super resolución y corrección de no uniformidad simultáneo (SR-NUC), basado en métodos de descenso de gradiente para obtener una imagen de alta resolución sin ruido a partir de un grupo de imágenes de baja resolución con ruido. Este algoritmo utiliza una función de error basada en el modelo del detector y de los efectos que distorsionan la información de la escena real, que compara la proyección de la imagen de alta resolución con las imágenes de baja resolución, minimizando las diferencias iterativamente. Para el tercero se implementó un algoritmo de registro basado en suma absoluta de diferencias (SAD), siendo los desplazamientos espaciales como los principales desestabilizadores. Este algoritmo determina el desplazamiento entre dos imágenes a través de una búsqueda de la mínima diferencia absoluta entre ambas. Luego se filtran los movimientos involuntarios y se estabiliza la imagen con un proceso de reconstrucción.

El circuito digital es implementado en un sistema en chip (SoC) XC7Z020 de Xilinx, de la familia Artix-7 con un procesador ARM Cortex-A9 dóble núcleo embebido. Es de bajo consumo energético, eficiente y acelera la velocidad de cómputo comparada a una implementación en sistema de propósito general. El circuito trabaja a una velocidad de 160[MHz] de reloj y procesa 2.26 cuadros de 320x256 pixeles de 14 bits por segundo, velocidad limitada por el ancho de banda de la memoria RAM. Tiene un consumo menor a 2[W] utilizando menos del 30 % de los recursos disponibles: 8699 slices LUT, 8950 slice registers, 23 unidades BRAM y 36 DSPs.

Índice General

Índice de Figuras

Índice de Tablas

Siglas

ARS Arithmetic Right Shift

BRAM block RAM

BrCo constancia de brillo (*brightness constancy)*

FMC FPGA mezzanine card

FPA arreglo de plano focal (*Focal Plane Array)*

FPGA arreglo de compuertas programables (*Field Programmable Gate Array)*

FPN ruido de patrón fijo (*Fixed Pattern Noise)*

HR alta resolución (*High Resolution)*

HwC núcleo hardware

IBP proyección hacia atrás iterativa (*Iterative Back Projection)*

IIR respuesta al impulso infinito (*Infinite Impulse Response)*

IR infrarroja (*Infrared)*

IRFPA arreglo de plano focal infrarrojo (*Infrared Focal Plane Array)*

LMS mínimos cuadrados (*Least mean squares)*

LR baja resolución (*Low Resolution)*

LUT look up table

LVDS señal diferencial de bajo voltaje (*low voltage differential signal)*

LWIR infrarrojo de onda larga (*Long Wave Infra Red)*

MAP máximo a posteriori (*Maximum a Posteriori)*

MMCM Mixed-Mode Clock Manager

MSE error cuadrático medio (*Mean Sqared Error)*

NLM medias no locales (*Non Local Means)*

NU no uniformidad (*Nonuniformity)*

NUC corrección de no uniformidad (*Nonuniformity Correction)*

PSF función de dispersión de punto (*Point Spread Function)*

PSNR relación señal a ruido de pico (*Peak Signal-to-Noise Ratio)*

RAM memoria de acceso aleatorio (*Random Acces Memory)*

RMSE raíz del error cuadrático medio (*Root Mean Sqared Error)*

ROI región de interés (*Region of Interest)*

SAD suma absoluta de las diferencias

SoC sistema en chip (*System on a chip)*

SR super resolución (*Super Resolution)*

SR-NUC super resolución y corrección de no uniformidad simultánea (*simultaneous Super-resolution and Nonuniformity Correction)*

SwC núcleo software

Capítulo 1: Introducción

1.1. Radiación infrarroja

La luz es una forma de energía que puede ser percibida por el ojo humano, permitiendo observar objetos gracias al fenómeno denominado reflexión. Sin embargo, la luz es sólo una porción de la radiación electromagnética que comprende las longitudes de onda entre 400 y 700[nm], encontrándose entre la radiación infrarroja y la radiación ultravioleta. La radiación infrarroja (*Infrared*, IR) es denominada comúnmente luz infrarroja y posee las mismas propiedades que la luz visible, con la particularidad de que cualquier objeto con una temperatura superior a 0°[K] la produce. Esta radiación comprende el rango de longitud de onda entre 0.75 y 1000[µm], dividiéndose en 5 zonas: near-infrared (NIR) de 0.75 a 1 4[µm], short-wavelength infrared (SWIR) de 1.4 a 3[µm], mid-wavelength infrared (MWIR) de 3 a 8[µm], long-wavelength infrared (LWIR) de 8 a 15[µm] y far-infrared (FIR) de 15 a 1000[µm]. Las regiones MWIR y LWIR son referidas como "infrarrojo térmico" ya que la radiación en estas zonas son emitidas por los objetos mismos sin necesidad de luz infrarroja externa sobre estos. Como consecuencia a lo anterior, si un objeto aumenta de temperatura, entonces éste irradia más energía, y en un sistema de imágenes térmicas se puede observar más brillante. Dada las características particulares de la radiación infrarroja en los rangos MWIR y LWIR, existe una gran ventaja en términos de obtención de información en tales rangos que no puede ser obtenida en el rango visible. Esto se debe a que los sistemas de visualización convencionales necesitan una fuente de luz sobre la escena a capturar y no todos los objetos emiten luz visible. Estas ventajas hacen que la radiación infrarroja sea de gran interés en distintas áreas de desarrollo como las industrias de metal, automovilística, manutención, marina, en la agricultura, el medio ambiente, la medicina, seguridad, entre otros.

1.2. Sistemas de captura infrarroja

Existen dispositivos que capturan imágenes en la longitud de onda IR, llamadas cámaras infrarrojas. Los componentes principales son el lente que enfoca la radiación infrarroja, el detector infrarrojo y el hardware/software para procesar y desplegar lo capturado. El detector de la cámara es un arreglo de plano focal (*Focal Plane Array*, FPA) fabricado con distintos tipos

de materiales sensibles a las ondas infrarrojas, y se denomina arreglo de plano focal infrarrojo (*Infrared Focal Plane Array*, IRFPA). Los detectores se dividen en dos categorías, detectores térmicos y detectores cuánticos [2]. Dentro de los detectores térmicos comúnmente se encuentran los microbolómetros hechos de metal o de un material semiconductor. Estos detectores son de menor costo que los detectores cuánticos, pero reaccionan a la energía radiante incidente y tienen una respuesta más lenta y una sensibilidad mucho menor. Por otra parte, pese a que los detectores cuánticos son generalmente más rápidos en respuesta y tienen mayor sensibilidad que los detectores térmicos, necesitan sistemas de refrigeración para su funcionamiento, a veces incluso requieren sistemas criogenizados, elevando aún más sus costos. Los detectores cuánticos están fabricados con estructuras cristalinas tales como InSb, PtSi, InGaAs, HgCdTe, entre otros [3]. El principio de operación de los detectores cuánticos se basa en los cambios de estados de los electrones de las estructuras cristalinas del FPA, que reaccionan a los fotones incidentes de la radiación infrarroja.

1.3. Baja densidad de pixeles

Dentro de las características principales de los dispositivos de captura de imágenes está la resolución de FPA. En el caso particular de los IRFPA, los rangos de resolución van desde 160×120 pixeles hasta aproximadamente 1024×1024. La limitación de resolución máxima se debe a la complejidad del proceso de fabricación asociado a la manufactura de los IRFPA con alta densidad de detectores [4] y a la baja eficiencia de los detectores para estas longitudes de onda. Esto limita el número de elementos detectores que pueden ser integrados en el FPA, ya que reducir el tamaño de los detectores del IRFPA implica aumentar el tiempo de integración para capturar una cantidad correcta de fotones desde la escena, trayendo consigo problemas de ruidos no deseados en la imagen capturada [11]. Esto es un problema al momento de querer capturar imágenes, escenas u objetos con una calidad adecuada para análisis o escenas u objetos menores a lo que puede capturar un pixel. Para solucionar este problema existen dos alternativas: generar resolución matemáticamente con algoritmos de super resolución (*Super Resolution*, SR)[5, 6] o agregar un sistema de aumento óptico [7]. Algunos sistemas de captura de imágenes con aumento óptico en el rango IR son diseñados para capturar imágenes de un tamaño microscópico y son llamados microscopios infrarrojos.

Los microscopios infrarrojos son similares a los microscopios convencionales en luz visible, exceptuando la óptica, que es realizada con espejos en vez de lentes. La utilización de estos sistemas de captura infrarroja ha ido creciendo en los últimos años debido a que permiten

analizar escenas que con cámaras infrarrojas convencionales y microscopios convencionales no es posible. Si bien estos sistemas solucionan el problema del tamaño mínimo capturado por el IRFPA, sufren igualmente del problema de baja densidad de pixeles.

Los algoritmos de SR resuelven el problema presente en sistemas de imágenes de baja resolución (*Low Resolution*, LR), en donde se necesita obtener imágenes de alta resolución (*High Resolution*, HR). Las imágenes de HR entregan mayores detalles de la escena capturada, algo que puede ser crítico en algunas aplicaciones, como por ejemplo aplicaciones médicas [8]. Para obtener imágenes HR se requiere que la densidad de pixeles de la imagen sea alta, es decir, una mayor concentración de detectores de tamaño reducido en un área determinada [9].

Para resolver el problema de imágenes con LR sin intervenir el proceso de fabricación de los sistemas detectores, se recurren a los algoritmos de super resolución. La super resolución es el proceso de obtención de imágenes de HR respecto a la resolución original de los datos, utilizando la información de una o una secuencia de imágenes de LR [12]. Un ejemplo de los resultados obtenidos al aplicar SR se muestra en la Figura 1.1, aquí la imagen original 1.1a está en baja resolución, esto se aprecia en la región de interés (*Region of Interest*, ROI) mostrada en la imagen 1.1b, en donde luego de aplicar un algoritmo de super resolución se espera obtener algo similar a la imagen 1.1c.

Se pueden clasificar los distintos métodos de super resolución en dos categorías dependiendo de la cantidad de imágenes LR utilizadas, los métodos secuencia simple (single-frame), que utilizan una sola imagen de LR, y los de secuencia múltiple (multiple frame), que utilizan múltiples imágenes LR.

Dentro de los métodos de secuencia simple se puede encontrar la regresión Gaussiana [13], un algoritmo máximo a posteriori (*Maximum a Posteriori*, MAP) propuesto en [14], entre otros [15, 16, 17]. El principio básico de este método es añadir información desde imágenes con escenas similares a la que se quiere aplicar la super resolución, o buscar por secciones de escena similares que puedan complementarse entre sí. Por ejemplo, el algoritmo NLM busca explotar la auto similitud en imágenes naturales utilizando la información de pixeles dentro de una vecindad para asignar pesos, que esta relacionado con la similitud que existe entre ellos dada por la distancia euclidiana en una ventana determinada. Este algoritmo, según sea aplicado, permite la corrección de no uniformidad [18] o aplicar super resolución [14].

Por otra parte, los métodos de secuencia múltiple toman una secuencia de cuadros de imagen de una misma escena. La información obtenida por un cuadro simple se complementa con la información de los demás cuadros o por una parte de estos, llamada ROI, asumiendo que entre

(a) Imagen original. (b) ROI en LR. (c) ROI en HR.

Fig. 1.1: Resultado esperado al aplicar algoritmos de super resolución.

ellos existe un movimiento relativo, en donde se añade información con desplazamientos que incluyen sub-pixel [19, 20, 21, 22, 23]. Los métodos de SR de secuencia múltiple tienen dos etapas, la etapa de registro y la de reconstrucción. En la etapa de registro las imágenes LR son comparadas para estimar los cambios espaciales entre ellas (traslación, rotación y/o proyección). Estos cambios luego son utilizados en la etapa de reconstrucción de la imagen HR para combinar las imágenes de LR.

Un enfoque importante de este problema es la implementación en sistemas de tiempo real y/o de alta eficiencia. Un sistema de alta eficiencia realiza tareas de cómputo a una velocidad mayor, con menos consumo de energía y recursos, en este caso se comparan sistemas embebidos hardware con sistemas computacionales de propósito general. El trabajo presentado en [24] utiliza un algoritmo de secuencia múltiple llamado proyección hacia atrás iterativa (*Iterative Back Projection*, IBP), que consiste en el refinamiento de la escena en la dimensión HR, comenzando con una aproximación inicial de imagen HR como una interpolación del cuadro LR de referencia. Luego, de forma iterativa, esta imagen estimada en HR es llevada al dominio LR para minimizar las diferencias con el conjunto de imágenes LR generando una nueva imagen HR estimada en cada iteración. Los resultados obtenidos en el trabajo permiten corregir a una tasa de 25 cuadros por segundo una secuencia de imágenes de resolución 512×512 con 8 iteraciones, y una secuencia de imágenes de resolución 256×256 con 36 iteraciones.

1.3.1. Registro de movimiento

El registro de movimiento entre imágenes es un problema ampliamente estudiado en la literatura, siendo éste el proceso de alinear espacialmente dos imágenes de una escena con tal de que los puntos correspondientes asuman las mismas coordenadas. Este proceso permite encontrar, para cada punto de la primera imagen, su correspondiente en la segunda imagen. Se denomina a la primera imagen como imagen de referencia y a las siguientes como imagen

de interés u objetivo. La imagen de referencia se mantiene sin alterar, mientras la segunda es transformada para tener las mismas coordenadas espaciales de la imagen de referencia mediante el proceso de reconstrucción [25]. Puede darse el caso de que no todos los puntos o partes de la imagen de referencia estén presentes en la imagen de interés, imposibilitando encontrar la correspondencia de todos los puntos entre las imágenes.

La relación que existe entre la imagen de interés y la imagen de referencia se denomina transformación. Se definen cuatro clases primarias de transformaciones planas, en orden cada una agrega más grados de libertad que la anterior: euclidiana, similitud, afín y proyectiva.

Dentro de las fuentes bibliográficas, los algoritmos de registro se separan en dos tipos según el dominio utilizado [26]: registro en el dominio de la frecuencia [27, 28, 29] y registro en el dominio espacial [30, 31, 32]. Como se describe en [33], los métodos en el dominio de la frecuencia describen un desplazamiento como un desplazamiento lineal en la fase, una rotación de la imagen como una rotación en la magnitud del espectro frecuencial de la imagen. En los métodos de registro en el dominio espacial se puede analizar movimientos de mayor complejidad, como escalamiento y proyección, además de poder ser realizados en toda la imagen o una porción de ésta extendiendo los resultados a la imagen completa.

Uno de los trabajos encontrados en la literatura implementa un algoritmo de registro de imágenes directo [26]. Este trabajo estima la transformación de dos imágenes tomadas en distintos momentos de la misma región. Se asume que que existe un mapeo entre los pixeles de una imagen con la otra tal que el valor de intensidad entre las imágenes es el mismo, suposición conocida como constancia de brillo (*brightness constancy*, BrCo), y se busca estimar la transformación afín entre ellas mediante un proceso iterativo. Adicionalmente, diseña una arquitectura hardware en lógica programable equivalente sobre un arreglo de compuertas programables (*Field Programmable Gate Array*, FPGA), siendo capaz de registrar imágenes de 640×480 pixeles a una velocidad de 82 cuadros por segundo.

Adicionalmente, dentro de la literatura se han aplicado distintos métodos de registro de imágenes sobre el espectro IR [34, 35, 36] con distintas aplicaciones, respaldando el uso de estos algoritmos sobre imágenes IR. En general, los algoritmos de registro de imágenes en el espectro visible pueden ser extendidos en utilización al espectro infrarrojo, siendo posible así analizar distintas alternativas considerando cada caso particular en donde se quiera aplicar estos algoritmos.

1.4. Ruido de no uniformidad

Además del problema de baja resolución, los sistemas de cámaras infrarrojas son afectados por dos tipos de ruido que degradan la información limitando el desempeño de procesamiento de las imágenes capturadas [37]. Uno de estos es el ruido temporal, como el ruido blanco, y el otro es el ruido de patrón fijo (*Fixed Pattern Noise*, FPN). El FPN es producto de que la respuesta de los detectores del IRFPA no son idénticas debido al proceso de fabricación[10, 38]. Este FPN se superpone en la imagen de la escena degradándola y afectando su calidad.

La corrección de no uniformidad (*Nonuniformity Correction*, NUC) corresponde al proceso de compensar parcial o totalmente el problema conocido como no uniformidad (*Nonuniformity*, NU) del IRFPA. La NU es producto de por imperfecciones del IRFPA causadas por los procesos de fabricación incapaces de crear detectores IR con respuestas exactamente iguales. Como se muestra en la Figura 1.2, la falta de similitud en la respuesta de los detectores dentro del IRFPA distorsiona la información de la escena capturada (Figura 1.2a), añadiendo ruido manifestado como rejilla superpuesto sobre la escena real (Figura 1.2b). Este problema dificulta enormemente la utilización de la información obtenida de forma precisa, haciéndose necesario buscar formas de contrarrestar la NU.

Los algoritmos de NUC se dividen en dos métodos: Métodos basados en referencias y métodos basados en la escena. Los métodos basados en referencia usan información previa para calibrar la respuesta del IRFPA. Un ejemplo de esto es la calibración de múltiples puntos, en donde se utilizan distintos radiadores de cuerpo negro a varias temperaturas, permitiendo estimar los parámetros del modelo matemático del IRFPA con rendimientos de lectura radiométrica y de alta precisión. La principal desventaja que poseen estos métodos son los altos costos asociados en dinero y tiempo, ya que los cuerpos negros necesarios para la calibración son de alto costo y además se necesita mover el sistema de captura a donde se encuentran los cuerpos. Si bien este tipo de corrección presenta uno de los mejores resultados, sus desventajas hacen poco factible su implementación en línea con un sistema que lo requiera.

Por otra parte, los métodos basados en escenas requieren solamente de la información presente en las imágenes obtenidas por el sistema de captura. Este tipo de métodos utilizan distintas estrategias para compensar la NU, ya sea basándose en estadísticas [33], en redes neuronales [38, 39, 40], filtros como por ejemplo kalman [41] o métodos adaptativos complejos [42, 43]. Pese a que los resultados obtenidos con estos métodos no son tan precisos como los basados en referencias, su implementación puede ser realizada en línea sin intervenir el sistema de captura, siendo así una solución mucho menos costosa en términos de tiempo y dinero. Su principal

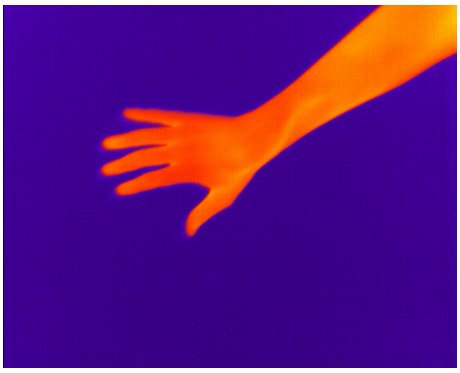
desventaja viene dada por su incapacidad de realizar una corrección radiométrica de la lectura de la cámara, además de introducir artefactos como ghosting.

Distintos trabajos han abordado la corrección de no uniformidad como un problema solucionable en línea en conjunto al sistema de captura. En [44] se implementó un algoritmo basado en redes neuronales en un FPGA capaz de realizar la corrección de no uniformidad a una tasa de 130 cuadros por segundo de vídeo proveniente de una cámara de 320×240 pixeles. El principio matemático del algoritmo se basa en imitar la retina, aprendiendo mediante estimación cuadro a cuadro los valores de offset y ganancia del modelo matemático del IRFPA, convergiendo a la solución luego de una cierta cantidad de cuadros. Esta implementación consume un total de 329[mW], siendo el consumo de potencia y la velocidad de cómputo suficientes para ser parte de un sistema en línea funcionando en tiempo real en conjunto a una cámara infrarroja sin enfriamiento modelo CEDIP Jade UC33 que trabaja a 60 cuadros por segundos con un consumo de 5[W]. Para dimensionar la eficiencia y el bajo impacto en consumo y espacio, se debe comparar esta implementación con las soluciones presentes en el mercado, que constan de sistemas refrigerados de alto consumo energético y de espacio físico. En el caso de la cámara Jade existe una alternativa refrigerada que aumenta a 35[W] el consumo de potencia, y el volumen físico pasa desde $168 \times 80 \times 80$[mm], en su versión sin refrigeración, a $270 \times 120 \times 150$[mm], en su versión refrigerada. Adicionalmente, se debe considerar que aumentar la cantidad de componentes, sobretodo mecánicos para la refrigeración, disminuye la confiabilidad y vida útil de la cámara.

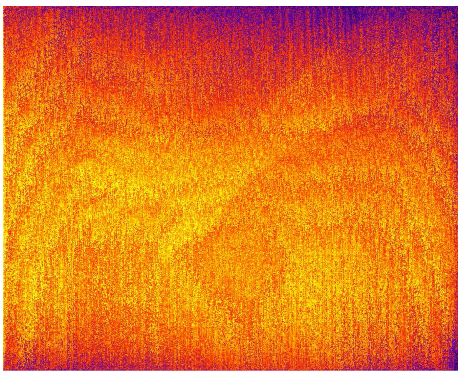
Otro trabajo, presentado en [45], implementa un algoritmo de corrección de no uniformidad basado en escenas, llamado algoritmo de rangos constantes. Este algoritmo tiene como objetivo encontrar los valores de offset y ganancia que define el modelo del detector, bajo el supuesto de que todos los detectores del arreglo capturan una escena con información aleatoria pero con distribución uniforme dentro de un rango definido por cada sensor. El análisis y trabajo sobre el algoritmo permite que pueda ser implementado en tiempo real en un sistema en línea, compensando la no uniformidad a una tasa de 250 cuadros de 720×480 pixeles por segundo, con una disipación de poder de 157[mW].

1.5. Escenas inestables espacialmente

La estabilización de imágenes tiene como objetivo eliminar o compensar los movimientos producidos por perturbaciones mecánicas, muchas veces no deseados y que impiden un correcto estudio de las escenas de interés capturadas por los sistemas de imágenes. Como se muestra en la



(a) Escena real en cámara IR



(b) Imagen con NU en cámara IR.

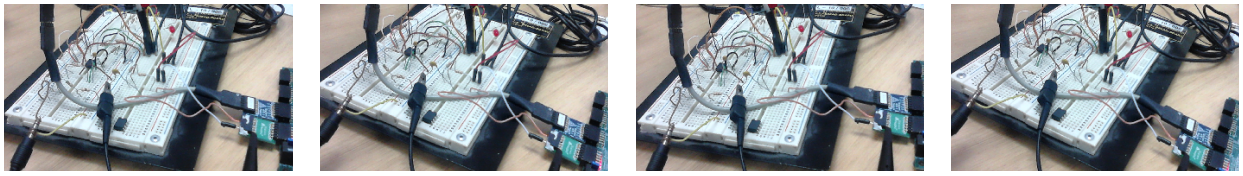
Fig. 1.2: Efecto de la no uniformidad sobre imágenes infrarrojas

Figura 1.3, pese a que una escena sea estática, si el dispositivo de captura presenta movimientos indeseados, pueden haber problemas en el análisis de la información. Este problema puede ser resuelto con sistemas ópticos o mecánicos que, usando sensores como acelerómetros o giroscopios, detectan el movimiento espacial y es compensado con sistemas físicos, como brazos mecánicos. Este tipo de soluciones suelen ser costosas o de gran tamaño, y por lo tanto inadecuadas para sistemas compactos. Existen métodos matemáticos digitales que permiten lograr la estabilización mediante el uso de la información capturada. El proceso digital de estabilización se divide en tres etapas: estimación, filtro y corrección de movimiento. La etapa de estimación de movimiento corresponde a determinar las diferencias espaciales que existen entre dos o más imágenes de una misma escena, esto se lleva a cabo con los algoritmos de registro de movimiento mencionados en la sección 1.3.1.

La etapa de filtrado de movimiento se utiliza para distinguir los movimientos voluntarios de los no deseados. Como es mencionado en [46], el filtro de movimiento siempre esta presente en las implementaciones y algoritmos de estabilización, incluso como un filtro nulo en aquellas que todos los movimientos se consideran acumulativos respecto al cuadro de referencia inicial. Dependiendo de la escena de interés capturada, los filtros varían en complejidad, teniendo desde aprendizajes simples hasta métodos basados en algoritmos complejos y elaborados como kalman [47].

El resultado de la etapa de filtrado es utilizado por la etapa de corrección de movimiento, en donde se aplica la transformación necesaria en la imagen de interés para llevarla a las coordenadas deseadas de referencia.

Existen distintos enfoques para la estabilización, uno de ellos es el presentado en [48] en donde proponen un algoritmo para estabilización de video de forma rápida, permitiendo una corrección robusta del corrimiento de rotación y traslación, con una solución de bajo costo y consumo. Se



(a) Cuadro 1. (b) Cuadro 2. (c) Cuadro 3. (d) Cuadro 4.

Fig. 1.3: Escena estática capturada sin estabilización espacial.

utiliza estimación de movimiento binaria para obtener un campo de vectores, minimizando problemas de memoria en sistemas digitales. En este trabajo se presenta un filtro que se basa en acumular el resultado de la estimación de movimiento entre cuadros de video con un factor de amortiguamiento. El factor de amortiguamiento considera tanto las diferencias anteriores como la actual de forma ponderada, permitiendo movimientos naturales del dispositivo de captura y eliminando movimientos de corta duración como vibraciones.

El trabajo hecho en [46] presenta una arquitectura hardware para estabilización digital de videos en tiempo real para sistemas embebidos de alto desempeño. Aquí, el algoritmo de estabilización analiza los cuadros de video actual y anterior para obtener un vector de estimación de movimiento, que luego es filtrado para separar los movimientos no deseados de los intencionales. Luego el vector filtrado es usado para corregir la salida del cuadro de video. La arquitectura fue implementada y validada en un FPGA de la familia Spartan-6 de Xilinx, que compensaba los movimientos involuntarios de la cámara de resolución 640×480 pixeles. La implementación logró corregir un máximo de 104.15 cuadros por segundo, disipando 24.16[mW] de potencia con un reloj de 100[MHz].

1.6. Discusión

Los problemas descritos afectan severamente la calidad de imagen en sistemas de captura con IRFPA, generando pérdida de información y detalles que es vital para las distintas aplicaciones. Consecuentemente, los microscopios infrarrojos no son ajenos a estas dificultades en la captura de imágenes. Dentro de la literatura existen variados métodos y algoritmos matemáticos para compensar y/o eliminar las perturbaciones en cámaras infrarrojas convencionales, entregando una imagen cercana a la realidad de la escena. La mayoría de las soluciones son implementadas en sistemas computacionales de propósito general en plataformas software, como en computadores de escritorio, y otros pocos en sistemas embebidos hardware para aplicaciones de tiempo real y bajo consumo enfocados a una compensación en línea.

En el estado del arte no existen implementaciones de arquitecturas digitales que aprovechen la re-utilización de la información de imágenes infrarrojas para corregir la no uniformidad y aplicar resolución simultáneamente, lo que podría permitir la disminución de utilización de recursos. Lo que se ha implementado con anterioridad utiliza sistemas en serie, es decir, aplicar corrección de no uniformidad seguida de super resolución, sin incluir etapas comunes. La simultaneidad aprovecha información existente en la escena para compensar ambos fenómenos con cálculos en común, lo que puede significar una reducción en utilización de recursos lógicos y potencia en una arquitectura en un circuito digital.

Pese a que existan múltiples trabajos enfocados a cámaras infrarrojas, no hay análisis que extiendan la aplicación de los distintos algoritmos a sistemas de captura infrarroja microscópicos. Los sistemas de captura microscópicas presentan distintos fenómenos en las escenas y configuración de captura, forzando a un enfoque dedicado a éstos.

1.7. Hipótesis de trabajo

Una arquitectura hardware especializada permite realizar una implementación más eficiente y rápida en cómputo que una aplicación software para resolver los problemas de baja densidad de pixeles, no uniformidad y desestabilización de movimiento espacial en imágenes microscópicas infrarrojas, con bajo consumo energético y con la velocidad necesaria de cómputo para ser implementada en un sistema en línea en tiempo real.

1.8. Objetivos

El objetivo de este trabajo es diseñar, implementar y evaluar un circuito digital en un FPGA algoritmos de super resolución, corrección de no uniformidad y estabilización de movimiento en imágenes microscópicas infrarrojas, para una aplicación en tiempo real, conectando el circuito digital a un microscopio infrarrojo para validar la arquitectura hardware.

Los objetivos específicos se desglosan en:

1. Adaptar el algoritmo de super resolución y corrección de no uniformidad simultánea (*simultaneous Super-resolution and Nonuniformity Correction*, SR-NUC) propuesto en [49] para posibilitar su implementación en arquitectura hardware.

2. Diseñar una arquitectura hardware modular, en lógica programable sobre un FPGA, para cada una de las etapas del algoritmo de SR-NUC adaptado en el objetivo 1.

3. Validar la arquitectura diseñada en el objetivo 2 con la implementación de aplicaciones de prueba sobre un FPGA en conjunto un microscopio infrarrojo.

4. Seleccionar un algoritmo de estabilización de imágenes adecuado para la implementación sobre un microscopio infrarrojo.

5. Diseñar una etapa que implemente el algoritmo seleccionado en el objetivo 4.

6. Validar la etapa propuesta en el objetivo 5 con la implementación de aplicaciones de prueba sobre un FPGA en conjunto al microscopio infrarrojo.

7. Enlazar etapas de SR-NUC y estabilización sobre imágenes microscópicas infrarrojas y entregar el resultado final en una pantalla.

1.9. Alcances y limitaciones

1. Las imágenes utilizadas en las pruebas e implementaciones fueron obtenidas desde el microscopio infrarrojo SOFRADIR-EC IRE 320M disponible en el laboratorio de Transmisión del DIE de la UdeC. Pese a esto, el diseño podría extenderse a otras cámaras o microscopios infrarrojos con ajustes en los parámetros de los algoritmos.

2. Dado a que la función función de dispersión de punto (*Point Spread Function*, PSF) que describe el microscopio infrarrojo SOFRADIR-EC IRE 320M está en desarrollo actualmente por otro grupo de trabajo, no se considera la etapa de desenfoque dentro del algoritmo SR-NUC.

3. Las simulaciones y pruebas se hicieron sobre el software MatLab en su versión 2014a.

4. La síntesis e implementación se hicieron sobre el software Vivado 2016.1 y Xilinx SDK 2016.1, de Xilinx.

5. Los códigos generados de descripción de hardware, con extensión **.v* quedan como código abierto a la comunidad científica.

1.10. Temario

En este trabajo presenta una implementación del algoritmo propuesto en [49], que realiza SR-NUC. Se aplica en imágenes infrarrojas microscópicas de onda larga (LWIR) y se adapta a una arquitectura de hardware en lógica programable sobre un FPGA, para acelerar su desempeño. Adicionalmente se desarrolla una etapa de estabilización de imágenes post-corrección para compensar los movimientos causados por el sistema de refrigeración de un microscopio infrarrojo y movimientos involuntarios sobre las escenas de interés.

El resto del informe se organiza como sigue:

- **Capítulo 2**: Se establece el marco teórico previo sobre los temas principales del informe, introduciendo la matemática principal de algunos algoritmos de NUC, SR y estabilización de imágenes.
- **Capítulo 3**: Se presenta el análisis del algoritmo de estabilización de imágenes utilizado y las consideraciones de diseño para la implementación. Adicionalmente se presentan resultados obtenidos mediante una implementación software en MatLab.
- **Capítulo 4**: Se presenta el análisis del algoritmo de super resolución y corrección de no uniformidad simultáneos utilizado y las consideraciones de diseño para la implementación. Adicionalmente se presentan resultados obtenidos mediante una implementación software en MatLab.
- **Capítulo 5**: Se detalla el diseño de la arquitectura diseñada para hardware embebido para la implementación de SR-NUC y estabilización de escena.
- **Capítulo 6**: Se presentan los resultados de la implementación del diseño en hardware, incluyendo detalles de utilización de recursos y consumo energético.

Capítulo 2: Marco teórico

En este capítulo se presenta una introducción teórica de los elementos matemáticos relevantes dentro del desarrollo de este trabajo. Para esto se extiende el análisis bibliográfico a los fundamentos matemáticos y físicos dentro de ésta.

2.1. Descripción del sistema

La Figura 2.1 muestra el flujo de trabajo utilizado para la plataforma de pruebas y de implementación. La radiación IR es capturada por un microscopio de modelo SOFRADIR-EC IRE 320M , que procesa y entrega la escena como una señal diferencial de bajo voltaje (*low voltage differential signal*, LVDS) sobre un protocolo camera-link. La señal LVDS es capturada por una interfaz FPGA mezzanine card (FMC) que decodifica la señal camera-link y la envía al sistema en chip (*System on a chip*, SoC), donde se procesa digitalmente la información y es devuelta mediante el mismo protocolo cameralink, codificando la señal con la misma interfaz FMC. La plataforma de trabajo esta diseñada para no interferir en el flujo original de datos, siendo así amigable con el sistema. El microscopio infrarrojo IRE 320M cuenta con las siguientes características:

- Resolución espacial de 320 × 240
- Tasa de actualización de hasta 320 cuadros por segundo, configurable.
- Rango infrarrojo de onda larga (*Long Wave Infra Red*, LWIR)
- Salidas de video de 14 bits vía Camera Link o Gigabit Ethernet.

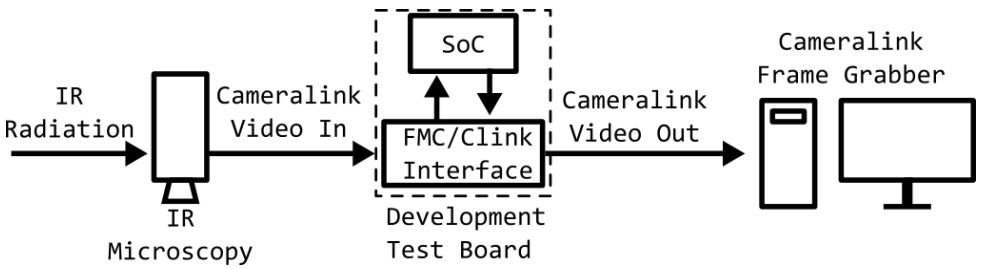


Fig. 2.1: Descripción a nivel de sistema de la plataforma de trabajo.

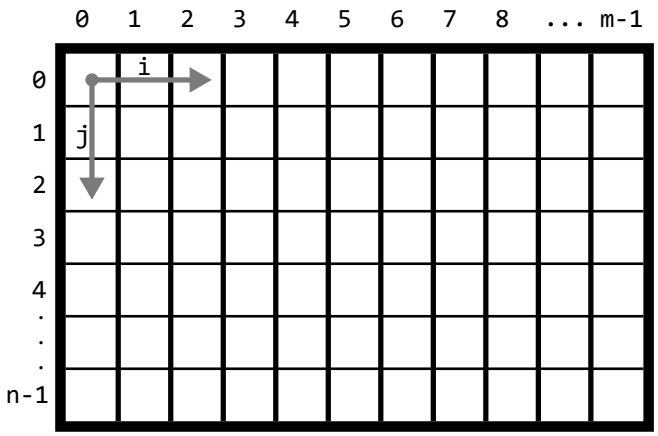
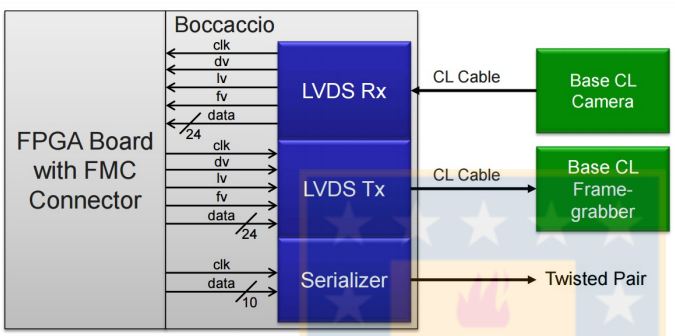
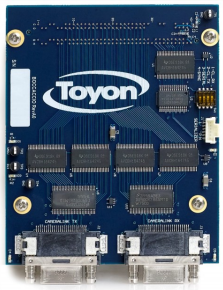


Fig. 2.2: Cuadro de video y sistema de coordenadas.



(a) Esquemático simplificado de la interfaz.



(b) Imagen física de la tarjeta.

Fig. 2.3: Tarjeta de expansión con interfaz cameralink-FMC en bidireccional.

- Corrección de no uniformidad y reemplazo de pixeles muertos sobre el microscopio, previa carga de información de cuerpos negros.

La unidad principal de datos es un cuadro de video, que se divide en líneas y éstas en unidades de pixel de 14 bits. Como se muestra en la Figura 2.2, cada cuadro tiene una dimensión de $m \times n$, con $m = 320$ y $n = 240$, con un sistema de coordenadas basado en ejes (i, j), con i para las cuentas horizontales y j para las cuentas verticales. Es decir, un cuadro de video está formado por 240 líneas de 320 pixeles.

La interfaz FMC Boccaccio de Toyon Research Corporation, mostrada en la Figura 2.3b, decodifica señales LVDS a señales paralelas de nivel CMOS y codifica en el sentido contrario. Utiliza dos circuitos integrados: receptor de 28 bits DS90CR288A LCDS para la entrada LVDS y un transmisor LVDS de 28 bits DS90CR287 para el envío. Esto permite dos cosas: no interferir en el flujo de trabajo y simplificar la adquisición y envío de datos dada la arquitectura paralela inherente dentro del SoC.

El protocolo Camera Link define una señal de reloj, tres de sincronismo y un bus de datos,

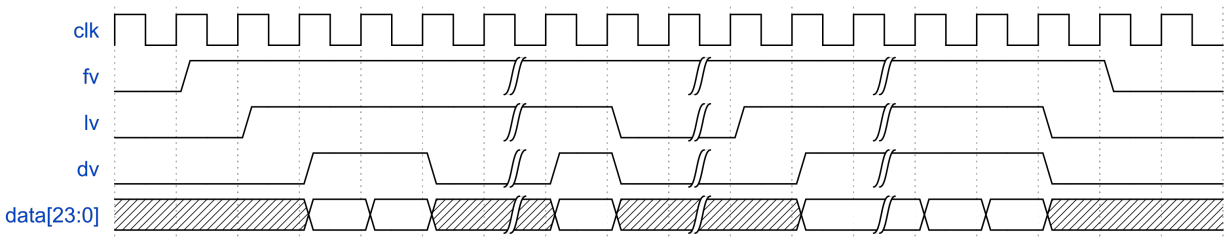


Fig. 2.4: Protocolo de comunicación de señales Camera Link decodificado.

todas sobre 4 señales en serie sobre el cable Camera Link. Como se muestra en la Figura 2.3a, en el receptor las señales son paralelizadas en un total de 28. En el gráfico de la Figura 2.4 se muestra el funcionamiento del protocolo Camera Link, la señal *clk* es el reloj de funcionamiento, la señal *fv* (frame valid) indica cuando un cuadro de imagen es válido, la señal *lv* (line valid) indica cuando una línea del cuadro es válida, la señal *dv* (data valid) indica cuando el dato es válido y la señal *data* es un bus de datos de 24 bits que contiene la información de cada pixel del cuadro. Para la recepción de datos es necesario conocer de antemano la resolución espacial de la imagen, ya que la generación de las señales de sincronismo depende de cada sistema que envía los datos.

2.2. Corrección de no uniformidad

2.2.1. Modelo del detector

El detector, denominado IRFPA, es el arreglo que permite transformar la radiación infrarroja de una escena a una imagen que se entiende computacionalmente o por el ojo humano. Existe un modelo lineal ampliamente usado en la literatura, basado en un modelo lineal estadístico que genera una independencia espacial para cada elemento del detector.

La respuesta de cada elemento (i, j) del detector de un cuadro de imagen infrarroja se representa por la siguiente aproximación lineal:

$$Y_{i,j}(k) = A_{i,j}(k) \cdot X_{i,j}(k) + B_{i,j}(k) + N_{i,j}(k), \tag{2.1}$$

donde:

- $X_{i,j}(k)$: Radiación infrarroja de entrada a cada elemento del detector (i, j).

- $A_{i,j}(k)$: Ganancia del detector para cada elemento (i, j).
- $B_{i,j}(k)$: Offset del detector para cada elemento (i, j).
- $N_{i,j}(k)$: Ruido temporal del detector para cada elemento (i, j).
- $Y_{i,j}(k)$: Salida digital del circuito de lectura.

La Ecuación (2.1) expresa matemáticamente los efectos de la no uniformidad causada por imperfecciones en el IRFPA sobre la imagen $X_{i,j}(k)$ de la escena original. La ganancia $A_{i,j}(k)$ y el offset $B_{i,j}(k)$ son parámetros que varían lentamente en el tiempo por causas externas (como la temperatura), y pueden ser determinados con algoritmos. El ruido temporal $N_{i,j}(k)$ es altamente variable en el tiempo, incluso entre cuadros, pero su efecto en factores de potencia es despreciable y normalmente se omite para análisis. La salida $Y_{i,j}(k)$ del circuito de lectura es lo que se obtiene de las cámaras infrarrojas, y es la señal que se procesa.

En general, este modelo es utilizado para la corrección de no uniformidad, estimando los parámetros de offset y/o ganancia como constantes en el periodo de tiempo de captura,en un proceso inverso de estimación de la escena real capturada:

$$X_{i,j}(k) = \frac{Y_{i,j}(k) - B_{i,j}}{A_{i,j}} \tag{2.2}$$

2.2.2. Calibración basada en referencia

La calibración basada en referencias utiliza información previa al momento de la corrección de la escena. La calibración más conocida y utilizada es la de dos puntos, que usa dos fuentes de radiación uniforme (radiadores de cuerpo negro) para formar un sistema de dos ecuaciones para cada elemento del detector:

$$Y_1 = A \cdot X_1 + B \tag{2.3a}$$

$$Y_2 = A \cdot X_2 + B, \tag{2.3b}$$

donde:

- A: Matriz de ganancia del IRFPA.
- B: Matriz de offset del IRFPA.

- X_1: Primera temperatura aplicada del radiador.
- X_2: Segunda temperatura aplicada del radiador.
- Y_1: Primera matriz de lectura del IRFPA.
- Y_2: Segunda matriz de lectura del IRFPA.

Las radiaciones de las fuentes X_1 y X_2 son conocidas, uniformes y se espera así mismo una respuesta uniforme en el plano focal infrarrojo. Las lecturas Y_1 y Y_2 del IRFPA igualmente son conocidas, y contienen la lectura uniforme alterada por la no uniformidad. Luego los parámetros A y B son estimados con un despeje simple del sistema de ecuaciones lineales:

$$A = \frac{Y_2 - Y_1}{X_2 - X_1} \tag{2.4a}$$

$$B = Y_2 - A \cdot X_2. \tag{2.4b}$$

2.2.3. Filtro espacial lineal de imágenes

Los siguientes algoritmos asumen la respuesta $X_{i,j}$ de cada detector del IRFPA está determinada por una aproximación de valores de detectores dentro de una vecindad v, dada por una ventana de dos dimensiones de Δm pixeles verticales y Δn pixeles horizontales. La ventana se construye tomando el punto central, luego se aplica el filtro con la ventana sobre el pixel central, obteniendo un nuevo valor.

Filtro de la media

El filtro de la media toma la imagen original y genera una nueva imagen dada por el promedio de los valores de intensidad de los pixeles incluidos en una vecindad predefinida. El cálculo matemático se realiza de la siguiente manera:

$$\hat{X}_{i,j}(k) = \frac{1}{4\Delta m \Delta n} \sum_{u=i-\Delta m}^{i+\Delta m} \sum_{v=j-\Delta n}^{j+\Delta n} Y_{u,v}(k), \tag{2.5}$$

Filtro de Gauss

El filtro gaussiano es similar al filtro de la media, la principal diferencia es que la media considera la misma ponderación para cada pixel incluido en la vecindad, en cambio el filtro gaussiano le asigna una ponderación según la distancia al pixel central. La ponderación está dada por una distribución gaussiana.

2.2.4. Red neuronal de Scribner

El algoritmo de Scribner busca emular matemáticamente el funcionamiento de la retina humana, a través de una adaptación de los parámetros del modelo del detector mediante mínimos cuadrados (*Least mean squares*, LMS). Para esto se toma la Ecuación (2.2) para desprender dos nuevos parámetros:

$$O_{i,j}(k) = \frac{B_{i,j}(k)}{A_{i,j}(k)}, \tag{2.6a}$$

$$G_{i,j}(k) = \frac{1}{A_{i,j}(k)}. \tag{2.6b}$$

El algoritmo de Scribner define que cada detector i, j es una neurona de una red lineal, cada una con un factor aditivo $O_{i,j}(k)$,denominado offset, y un factor multiplicativo $G_{i,j}(k)$,denominado ganancia. Para cada tiempo k la red neuronal tiene como entrada la lectura del detector y entrega en la salida una estimación de la radiación incidente real $X_{i,j}(k)$. La suposición de Scribner afirma existe una alta probabilidad de que cada detector es incidido por una radiación igual a sus vecinos cercanos. Bajo esta suposición se espera que el valor real de cada neurona sea un estimado cercano al promedio de incidencias de la vecindad cercana, siendo esta una referencia para determinar los parámetros de offset y ganancia. Utilizando el algoritmo de LMS se formula una función de error $E_{i,j}(k)$, para cada neurona, como la diferencia entre el valor esperado $T_{i,j}(k)$ y la información estimada $\hat{X}_{i,j}(k)$:

$$E_{i,j}(k) = T_{i,j}(k) - \hat{X}_{i,j}(k), \tag{2.7}$$

donde el valor esperado $T_{i,j}(k)$ se calcula como el promedio de valores de incidencia dentro de una vecindad espacial cuadrada, de radio υ, para cada detector i, j :

$$T_{i,j}(k) = \frac{1}{(2\upsilon+1)^2} \sum_{M=i-\upsilon}^{i+\upsilon} \sum_{n=j-\upsilon}^{j+\upsilon} X_{m,n}(k). \tag{2.8}$$

La aplicación de LMS se hace con gradientes asociados a cada parámetro, estableciendo la función de error cuadrático como:

$$F_{i,j}(k) = E_{i,j}(k)^2 = (X_{i,j}(k) - T_{i,j}(k))^2, \tag{2.9}$$

siendo así los gradientes:

$$\frac{\delta F_{i,j}}{\delta G_{i,j}} = \frac{8\upsilon(\upsilon+1)}{(2\upsilon+1)^2} E_{i,j}(k) Y_{i,j}(k), \tag{2.10a}$$

$$\frac{\delta F_{i,j}}{\delta O_{i,j}} = \frac{8\upsilon(\upsilon+1)}{(2\upsilon+1)^2} E_{i,j}(k). \tag{2.10b}$$

Luego se resuelve el problema de optimización usando descenso de gradiente, actualizando mediante un proceso recursivo los parámetros $O_{i,j}(k)$ y $G_{i,j}(k)$ con las siguientes expresiones:

$$G_{i,j}(k+1) = G_{i,j}(k) - \eta_g \cdot \frac{8\upsilon(\upsilon+1)}{(2\upsilon+1)^2} E_{i,j}(k) Y_{i,j}(k), \tag{2.11a}$$

$$O_{i,j}(k+1) = O_{i,j}(k) - \eta_o \cdot \frac{8\upsilon(\upsilon+1)}{(2\upsilon+1)^2} E_{i,j}(k), \tag{2.11b}$$

donde los parámetros η_o y η_g controlan la velocidad de convergencia de la optimización de la ganancia y el offset respectivamente. Estos parámetros deben ser seleccionados adecuadamente para garantizar la estabilidad del algoritmo.

2.2.5. Rangos constantes

La compensación de rangos constantes está basada en la escena y propone compensar el ruido espacial estimando los parámetros $A_{i,j}(k)$ y $B_{i,j}(k)$ utilizando la información estadística de la escena, disponible en cada cuadro de salida $Y_{i,j}(k)$. Para poder aplicar rangos constantes para la corrección de no uniformidad, se asume que bajo un movimiento permanente de la cámara en la escena se cumple que:

1. Todos los detectores tienen la misma probabilidad de ser incididos por la mínima y máxima radiancia.

2. La ganancia y offset tienen una dinámica mucho más lenta que la radiación infrarroja de la escena.

3. La distribución de probabilidad de la radiación infrarroja de la escena es uniforme en el rango $[x_{min}x_{max}]$.

Cumpliéndose estos supuestos, se le atribuye una media y una varianza a cada cuadro de la escena como:

$$\bar{X} = \frac{X_{max} + X_{min}}{2}, \tag{2.12a}$$

$$\sigma^2 = \frac{(X_{max} - X_{min})^2}{12}. \tag{2.12b}$$

Realizando una compensación adaptativa con las siguientes ecuaciones:

$$\bar{Y}(k) = (1-\alpha)Y(k) + \alpha Y(k-1), \tag{2.13a}$$

$$\sigma_Y(k) = (1-\alpha)|Y(k) - \bar{Y}(k)| + \alpha\sigma_Y(k-1), \tag{2.13b}$$

$$X(k) = \frac{(Y(k) - \bar{Y}(k))\sigma_X}{\sigma_Y(k)} + \bar{X}, \tag{2.13c}$$

donde inicialmente se estima la media y la desviación estándar de cada detector, que contienen información del offset y la ganancia asociados a la no uniformidad.

2.3. Estimación de movimiento

El proceso de estimación de movimiento, o registro de imágenes, en donde se compara la imagen de interés u objetivo I_{obj} con la imagen de referencia I_{ref}, puede ser determinada por múltiples transformaciones $\mathcal{F}$. Como se muestra en el Tabla 2.1 y se observa en la Figura 2.5, las transformaciones lineales se dividen en euclidiana, similitud, afín y proyectiva, las que agregan progresivamente mayor grados de libertad y, así mismo, más complejidad al proceso de registro. Es por eso que se debe escoger adecuadamente un tipo de transformación acorde a la aplicación para obtener la mayor cantidad de información útil para el proceso, con la menor complejidad computacional posible. La transformación euclidiana considera rotación y traslación, la transformada de similitud añade una escala isotrópica (escala igual en ambas direcciones), la transformación afín hace no isotrópica la escala y la transformación proyectiva añade una escala que cambia linealmente con la posición.

Dentro de las transformaciones para la estimación de movimiento, existen métricas de comparación que permiten determinar similitud entre imágenes. Estas se utilizan para determinar

cual es la transformación que permite minimizar la diferencia entre las imágenes objetivo y de referencia, mediante una búsqueda iterativa. Tres funciones de comparación son utilizadas típicamente para obtener una transformación rígida:

- Suma de las diferencias absolutas (SAD):

$$\sum_{(u,v)\in W} |I_{ref}(u+\Delta i, v+\Delta j) - I_{obj}(u,v)| \tag{2.14}$$

- Suma de los cuadrados de las diferencias:

$$\sum_{(u,v)\in W} (I_{ref}(u+\Delta i, v+\Delta j) - I_{obj}(u,v))^2 \tag{2.15}$$

- Correlación cruzada:

$$\sum_{(u,v)\in W} I_{ref}(u+\Delta i, v+\Delta j) \cdot I_{obj}(u,v) \tag{2.16}$$

Una vez obtenida la relación correspondiente entre las coordenadas de la imagen de interés y la de referencia, se debe aplicar la transformación geométrica $\mathcal{F}$ para pasar de un plano geométrico al otro. Existen dos alternativas:

- Mapeo directo: Transformar las coordenadas de cada pixel de la imagen de referencia a la imagen de interés.

$$I_{obj}(\mathcal{F}_x) = I_{ref}(x) \tag{2.17}$$

- Mapeo inverso: Transformar las coordenadas de cada pixel de la imagen de interés a la imagen de referencia.

$$I_{obj}(x) = I_{ref}({\mathcal{F}^{-1}}_x) \tag{2.18}$$

El mapeo que se utiliza comúnmente es el mapeo inverso, ya que el mapeo directo presenta problemas con pixeles no asignados al momento de realizar la transformación geométrica. Puede resultar que cada pixel de la imagen de interés no sea asignado a coordenadas enteras en el sistema de coordenadas de la imagen de referencia, por lo que se acude a métodos de reconstrucción de imágenes mediante un proceso de interpolación.

Tabla 2.1: Matrices simbólicas de transformaciones lineales.

Nombre	Matriz Simbólica
Original	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Euclidiana	$\begin{bmatrix} \cos\theta & \sin\theta & t_x \\ -\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$
Similitud	$\begin{bmatrix} S\cos\theta & S\sin\theta & t_x \\ -S\sin\theta & S\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$
Afín	$\begin{bmatrix} a_{00} & a_{01} & t_x \\ a_{10} & a_{11} & t_y \\ 0 & 0 & 1 \end{bmatrix}$
Proyectiva	$\begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix}$

(a) Original. (b) Euclidiana. (c) Similitud. (d) Afín. (e) Proyectiva.

Fig. 2.5: Grupo de transformaciones lineales aplicados sobre una imagen.

2.3.1. Interpolacion de imágenes

El proceso de interpolación de imágenes permite generar pixeles de una imagen a partir de la información de la misma. Existen variados tipos de interpolaciones que dependen de cómo se utiliza la información disponible. Dentro de las más conocidas y utilizadas están la interpolación del vecino más cercano, la bilineal y la bicúbica. Cada una de estas se representa por una función $f(x)$ que actúa sobre un elemento x del vector a interpolar.

Interpolación del vecino más cercano

La interpolación del vecino mas cercano, o interpolación proximal, es un método simple que selecciona el valor del punto o pixel más cercano e ignora los demás dentro de la vecindad cercana. Es la técnica más simple y fácil de implementar, pero tiene una severa pérdida en calidad, causando en algunos casos aliasing y desenfoque en la imagen resultante. La expresión general se da por:

$$f(x) = \begin{cases} 1 & -0,5 \leq |x| < 0,5 \\ 0 & \text{otro} \end{cases} \tag{2.19}$$

Interpolación bilineal

La interpolación bilineal asume que los datos de la imagen son señales lineales por tramos. Es el segundo método más simple luego de la interpolación proximal, obteniendo mejores resultados con un costo computacional mínimo. Se describe como:

$$f(x) = \begin{cases} 1 - |x| & 0 \leq |x| < 1 \\ 0 & \text{otro} \end{cases} \tag{2.20}$$

La aplicación sobre las imágenes consideran sólo la vecindad cercana para generar el nuevo elemento, mediante una ponderación dependiendo de la distancia a la que se encuentre cada uno.

Interpolación bicúbica

La interpolación bicúbica se basa en utilizar un interpolante de clase C^1 (derivada continua) *spline* cúbica por tramos. La expresión para esta técnica de interpolación es:

$$f(x) = \begin{cases} 1 - (a+3)x^2 + (a+2)|x|^3 & |x| < 1 \\ a(|x|-1)(|x|-2)^2 & 1 \leq |x| < 2 \\ 0 & \text{otro} \end{cases} \tag{2.21}$$

donde a especifica la derivada en el punto $x = 1$. El valor de a es normalmente fijado en $-0,5$, el que produce el *kernel* cúbico de tercer grado.

2.4. Super resolución

El proceso de super resolución consiste en generar una imagen en HR a partir de la información de uno o un conjunto de cuadros LR. Los mayoría de los algoritmos de super resolución se componen de dos pasos fundamentales: alineación de las muestras en un sistema de coordenadas común, luego la reconstrucción de la imagen HR. Para la alineación se pueden utilizar los métodos de estimación de movimientos mencionados en el Capítulo 2.3, estableciendo uno de los cuadros como la imagen de referencia, y el resto de cuadros dentro del conjunto como imagen objetivo.

2.5. Estabilización de escena

La estabilización de escena tiene tres etapas fundamentales: estimación de movimiento, filtro de movimiento y corrección de movimiento. El proceso comienza con la obtención del cuadro actual, que es comparado con el cuadro anterior para estimar el movimiento. Para la estimación de movimiento es necesario realizar los pasos presentados en la sección 2.3. En casos en que la transformación $\mathcal{F}$ es lineal, es posible separar el movimiento en los dos ejes (i, j) de la imagen I_{obj} objetivo:

$$i_{obj} = \mathcal{F}_i(i_{ref}, j_{ref}, \lambda_i), \tag{2.22a}$$

$$j_{obj} = \mathcal{F}_j(i_{ref}, j_{ref}, \lambda_j), \tag{2.22b}$$

en donde:

- (i_{ref}, j_{ref}): Coordenadas del pixel en la imagen de referencia I_{ref}.
- (i_{obj}, j_{obj}): Coordenadas del pixel en la imagen de interés I_{obj}.
- $\mathcal{F}_i$ y $\mathcal{F}_j$: Funciones de transformación que mapean las coordenadas desde el pixel de origen al de destino.
- λ_i y λ_j: parámetros que determinan el método que se utiliza para determinar la transformación.

Considerando las transformaciones del tipo euclidiana simplificadas (sólo traslaciones), se tiene un parámetro que determina el desplazamiento para cada eje de forma independiente. De esta forma, para calcular las coordenadas del pixel de la imagen objetivo en las coordenadas de la imagen de referencia, se utilizan las ecuaciones:

$$i_{ref} = i_{obj} + \Delta i, \tag{2.23a}$$

$$j_{ref} = j_{obj} + \Delta j. \tag{2.23b}$$

donde:

- Δi variación de movimiento en el eje horizontal i.
- Δj variación de movimiento en el eje vertical j.

Una vez determinada la transformación, se deben filtrar los parámetros para determinar y separar los movimientos voluntarios de los involuntarios, siendo estos últimos los que deben ser compensados. La etapa de filtrado depende de la agresividad de separación que se busque. Entre mejor se quiera separar los movimientos voluntarios de los involuntarios, más agresivo y complejo debe ser el filtro. Estos van desde filtros básicos, como pasabajos, hasta filtros más elaborados como filtros basados en Kalman.

2.6. Índices de calidad

Los índices de calidad son utilizados para evaluar el desempeño de los algoritmos de corrección de no uniformidad y super resolución. El índice raíz del error cuadrático medio (*Root Mean Sqared Error*, RMSE) es un método ampliamente utilizado en la literatura, y permite comparar numéricamente dos imágenes, midiendo el nivel de similitud entre ambas. Para aplicarlo en la medición de calidad de NUC o SR se debe contar con una imagen de referencia lo más cercana a lo ideal posible. Para la no uniformidad se suelen usar imágenes corregidas con calibraciones de múltiples puntos, y para la SR se suele sub-samplear una imagen en HR, aplicar SR sobre las imágenes LR resultantes y comparar el resultado con la original.

$$MSE = \frac{1}{MN} \sum_{M-1}^{i=0} \sum_{N-1}^{j=0} \|I(i,j) - R(i,j)\|^2 \tag{2.24}$$

El RMSE se basa en calcular el error cuadrático medio (*Mean Sqared Error*, MSE) entre los valores correspondientes de los pixeles de dos imágenes distintas I y R de tamaño $M \times N$. Este método calcula la raíz del cuadrado del promedio de los errores entre los pixeles correspondientes. El índice indica una mayor similitud entre imágenes con valores cercanos a cero. Este índice se suele usar en comparaciones simples de calidad.

$$RMSE = \sqrt{MSE} \tag{2.25}$$

El relación señal a ruido de pico (*Peak Signal-to-Noise Ratio*, PSNR) está basado en el RMSE, pero el índice se calcular respecto al valor MAX, máximo valor posible de un pixel dentro del sistema de captura. Esto permite una percepción en intensidad de decibeles relativos a la imagen, al contrario del RMSE que entrega valores absolutos y no relativos. Cuanto mayor sea el valor del PSNR en decibelios dB, más similitud existe entre la imagen y la referencia. Los valores típicos alcanzan entre los 30dB y los 50dB, siendo sobre 50dB una similitud alta entre las imágenes comparadas. Este índice sirve para tener una referencia en relación al mismo sistema de captura.

$$PSRN = 20log_{10}(\frac{MAX}{RMSE}) \tag{2.26}$$

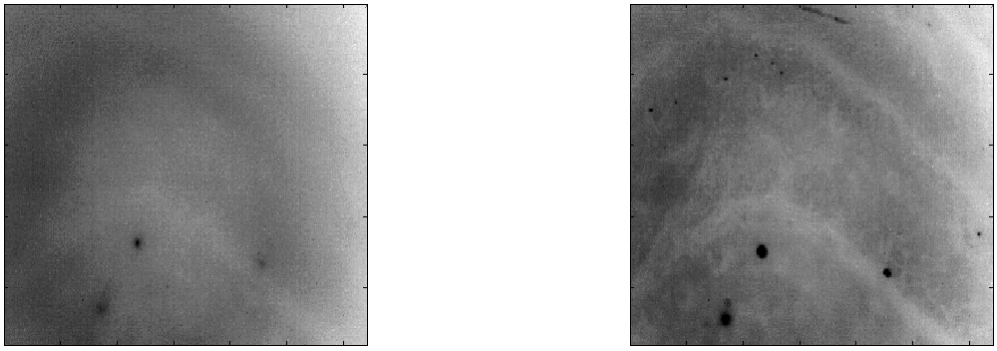
Capítulo 3: Algoritmo estabilizador de escena

En este capítulo se presenta el análisis del algoritmo de estabilización de imágenes utilizado y las consideraciones de diseño para la implementación. Se presenta el estudio realizado para determinar de forma adecuada los elementos matemáticos que involucran la estimación y el filtro de movimiento. Adicionalmente se presentan resultados obtenidos mediante una implementación software en MatLab.

3.1. Movimiento en el sistema

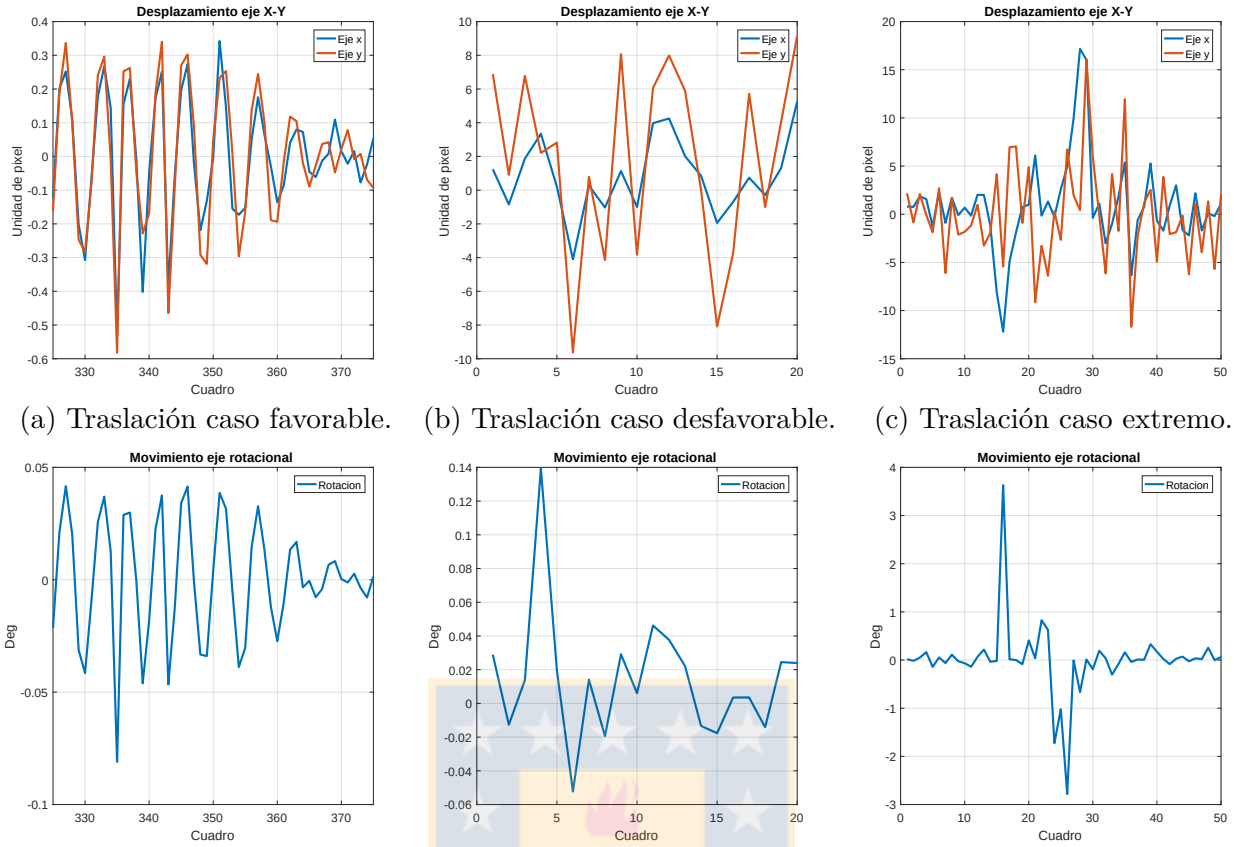
Se hizo un estudio sobre un conjunto de secuencia de videos provenientes del microscopio infrarrojo SOFRADIR-EC IRE 320M disponible en el DIE de la UdeC. El conjunto comprende más de 60 vídeos con distintas situaciones de operación, clasificadas en tres categorías:

- Extrema: Microscopio sobre una superficie elástica (una mesa común). La escena consiste en un objeto intentando ser enfocado con pulso humano. Esta condición de trabajo es influida por los movimientos del microscopio con su sistema de refrigeración, movimientos externos sobre la superficie elástica y el pulso humano, en donde para lograr el enfoque correcto se debe estar dentro de un rango de distancias del orden de los milímetros.
- Desfavorable: Microscopio anclado a una superficie rígida, con una escena en movimiento. El microscopio infrarrojo se encuentra anclado y sujeto sobre una mesa óptica, formando



(a) Escena no enfocada. (b) Escena enfocada.

Fig. 3.1: Efecto del desenfoque sobre una escena de estudio.



(a) Traslación caso favorable. (b) Traslación caso desfavorable. (c) Traslación caso extremo.

(d) Rotación caso favorable. (e) Rotación caso desfavorable. (f) Rotación caso extremo.

Fig. 3.2: Rotación y traslación entre cuadros para distintos casos.

un objeto rígido único entre ambos. La escena es enfocada a pulso humano. Acá, los efectos del enfriamiento del microscopio son absorbidos por la superficie rígida, pero existe movimiento generado por el pulso.

- Favorable: Es la forma de trabajo óptima para este proceso, tanto el microscopio como la escena están fijas en la mesa óptica. Aquí la única variación posible es la causada por la diferencia de posición dada por el sistema de enfriamiento del microscopio anclado a la mesa y la escena.

Dentro de los videos analizados, el fenómeno de desenfoque afecta severamente el análisis de registro, llevando a la matriz de transformación a valores sin lógica respecto a lo que realmente ocurre en la escena. Como se muestra en la Figura 3.1, el desenfoque (Figura 3.1a) impide casi en totalidad la visualización del objetivo real (Figura 3.1b). Este tipo de casos ocurre en situaciones de operación extrema en gran medida, desfavorable en menor medida y favorables no ocurre. Dado a que el objetivo de la implementación es mejorar la calidad de imágenes en escenas reales, las escenas desenfocadas no están dentro de los casos de estudio, ya que no es posible obtener información para análisis típicos de este tipo de imágenes. Pese a que el algoritmo original

propuesto en [49] considera un operador que resuelve el desenfoque, la complejidad de estudio y obtención de la PSF necesaria para tal objetivo imposibilita considerar el operador.

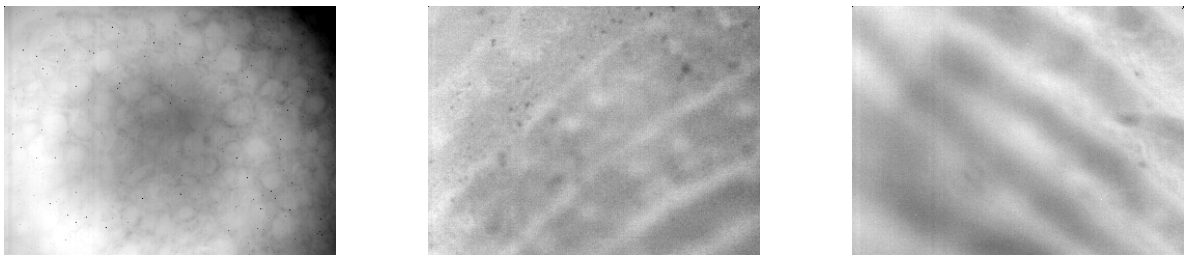
Las pruebas realizadas sobre los videos buscaron la transformación euclidiana entre cada cuadro, es decir, traslación y rotación. A continuación se detallarán los resultados generales de cada situación de operación, con ejemplos particulares dentro de éstas. Se determinan 3 parámetros que definen la matriz de transformación entre la imagen de referencia y la de interés, un par para traslación y uno para rotación.

Las gráficas de las Figura 3.2 muestran los valores de los parámetros obtenidos mediante MatLab al aplicar registro euclidiano entre cuadros, es decir, comparar cada cuadro con el anterior. Las imágenes de la Figura 3.3 muestran los cuadros de interés para análisis cualitativo de las gráficas correspondientes a los casos favorables, desfavorables y extremos, respectivamente.

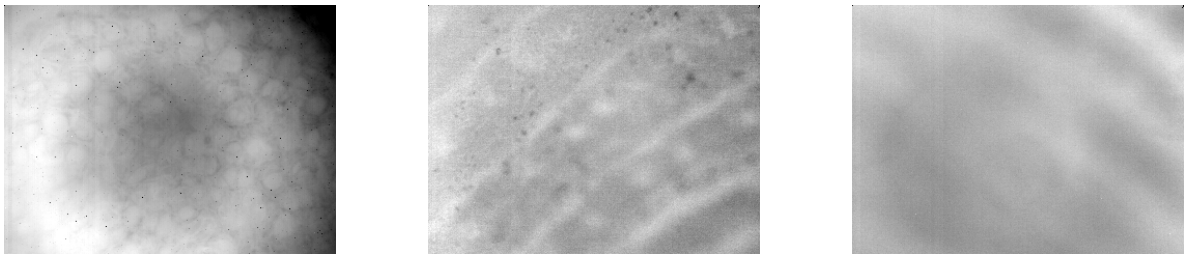
La escena del caso de estudio favorable corresponde a un trozo de poliestireno expandido posicionado en la zona de enfoque del microscopio. Se puede apreciar en las sub-figuras 3.2d y 3.2a que la variación temporal de los parámetros tiene una forma cíclica en torno al valor cero. Lo anterior se debe a que el movimiento que se percibe es producto del sistema de refrigeración del microscopio, principalmente el extractor de aire que funciona con un motor posiblemente desbalanceado en masa. Esta conclusión se basa en que el microscopio está anclado a una mesa óptica y además la escena de interés está posada sobre la misma, descartando cualquier otro movimiento externo posible. La rotación observada en este caso no supera los 0.08 grados sexagesimales y la traslación máxima es cercana a los 0.6 unidades de pixel. Los cuadros de interés de las Figuras 3.3a y 3.3a se escogieron cuando la rotación entre dos cuadros es la máxima estimada, en donde casi no se observan diferencias.

El caso de estudio desfavorable corresponde a una mano enfocada a pulso para ser vista en el microscopio. Como se muestra en la gráfica de la Figura 3.2e en este tipo de casos la rotación máxima aumenta a no más de 0.2 grados sexagesimales, pero al contrario del caso favorable, como se muestra en la Figura 3.2e el desplazamiento entre cuadros supera fácilmente la unidad, producido principalmente por el pulso de la mano. Si bien los grados de rotación aumentan, siguen siendo despreciables, no ocurre así con los desplazamientos de traslación. Los cuadros de interés de las Figuras 3.3b y 3.3e se escogieron cuando la traslación entre dos cuadros es máxima, en donde se puede distinguir la diferencia entre ambos.

El caso de estudio extremo corresponde a un brazo humano enfocado a pulso, con el microscopio sobre una mesa común. Como se puede observar de la gráfica 3.2f, la rotación en grados sexagesimales supera ampliamente la unidad en algunos cuadros, particularmente en aquellos



(a) Cuadro 1 favorable. (b) Cuadro 1 desfavorable. (c) Cuadro 1 extremo.



(d) Cuadro 2 favorable. (e) Cuadro 2 desfavorable. (f) Cuadro 2 extremo.

Fig. 3.3: Cuadros de interés para distintos casos.

Tabla 3.1: Características de parámetros por caso.

Caso de estudio	Rotación Máxima	Orden de desplazamiento
Favorable	Menor a 0.1º	Menor a 1
Desfavorable	Nunca mayor que 0.5º	Generalmente mayor a 1
Extrema	-Nunca mayor a 1º enfocado -Mayor que 1º desenfocado	Mucho mayor 1

en que ocurre un desenfoque en la escena. Lo anterior se presenta en las Figuras 3.3c y 3.3c, en donde los cuadros de interés corresponden al frame 16 de la gráfica 3.2f. Tal como se muestra en la gráfica 3.2c, el desplazamiento aumenta en relación al caso desfavorable, principalmente debido a que el caso extremo toma en cuenta que el microscopio no está fijo y además la escena es enfocada a pulso.

El resto de los videos dentro del set fueron estudiados con la misma metodología, en donde los resultados obtenidos se replican entre casos de estudio, resumidos el la Tabla 3.1. Para los casos favorables se obtiene una rotación muy pequeña y movimientos de desplazamiento sub-pixel. Para los casos desfavorables sigue siendo una rotación pequeña y movimientos de pixel mayores a uno. Para los casos extremos, al momento de desenfocarse la escena se obtienen valores de rotación sobre 1º. Como el principal interés es mejorar la calidad de imagen en escenas reales, los casos con desenfoque no estarán dentro de los casos abordados por este trabajo. Del estudio se concluye que es necesario considerar sólo movimientos de desplazamiento y no de rotación para la etapa de registro.

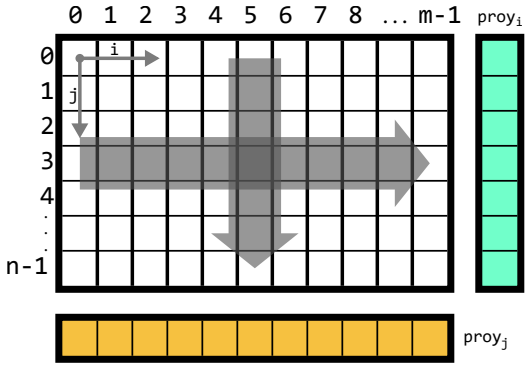


Fig. 3.4: Proyección integral de imagen sobre sus ejes.

3.2. Proyección integral con SAD en imágenes

La proyección sobre ejes permite reducir la dimensión del problema y reducir la complejidad computacional necesaria para obtener resultados sobre un conjunto de datos. Es un algoritmo de procesamiento previo que transforma imágenes de dos dimensiones a dos vectores de una dimensión. Las ecuaciones

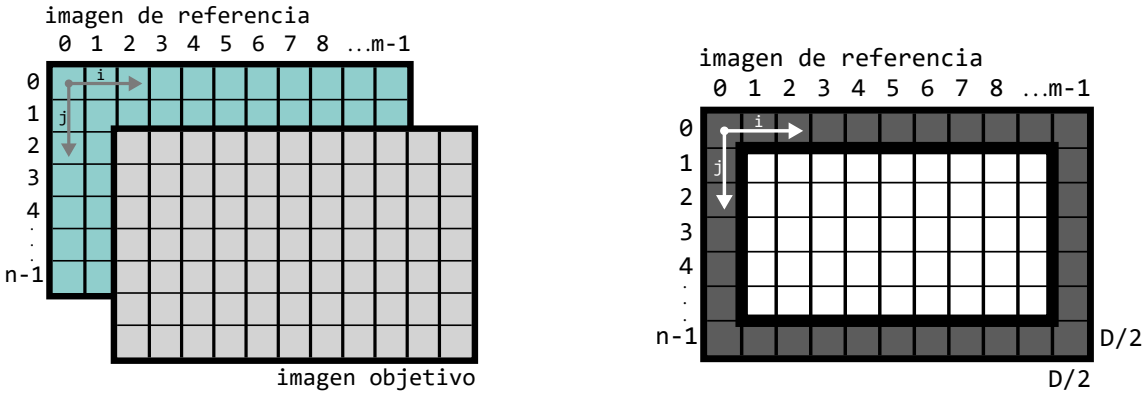
$$proy_i(v) = \sum_{v=0}^{n-1} I(u,v), \tag{3.1a}$$

$$proy_j(u) = \sum_{u=0}^{m-1} I(u,v), \tag{3.1b}$$

definen matemáticamente la proyección integral en los ejes i ey j. Como se muestra en la Figura 3.4, cada elemento del vector $proy_i$ se calcula como la suma vertical de cada fila de la imagen $I_{i,j}$, teniendo igual número de elementos como filas de la imagen. Por otra parte, cada elemento del vector $proy_j$ se calcula como la suma horizontal de cada columna de la imagen $I_{i,j}$, teniendo igual número de elementos como columnas de la imagen. La utilización de las proyecciones se hace en el paso de estimación de movimiento, donde las ecuaciones se modifican para ser aplicadas en vectores de una dimensión. Por lo anterior, se debe calcular la proyección en cada cuadro de imagen.

Aplicando las proyecciones integrales para la obtención de la suma absoluta de las diferencias (SAD), sobre la Ecuación (2.14), se desprenden dos ecuaciones, una para cada eje:

$$SAD_i(\Delta i) = \sum_{v=0}^{n_2-1} |proy_{i_{obj}}(v+\Delta i) - proy_{i_{obj}}(v)|, \tag{3.2a}$$



(a) Problema de mapeo a coordenadas de referencia.

(b) Restricción de resolución de la imagen resultante.

Fig. 3.5: Restricción de coordenadas.

$$SAD_j(\Delta j) = \sum_{v=0}^{m_2-1} |proy_{j_{obj}}(u+\Delta j) - proy_{j_{obj}}(u)|, \tag{3.2b}$$

en donde para obtener el desplazamiento en cada eje, Δi y Δj, se minimizan estas ecuaciones. Para la minimización se realiza una búsqueda iterativa de valores Δi y Δj, variando entre un mínimo Δi_{min} y un máximo Δi_{max}, determinados de forma equidistantes al desplazamiento nulo (Δi_{max}=-Δi_{min}). El valor mínimo y máximo del rango depende de la escena de estudio, siendo para el microscopio un rango de $[-12, 12]$ unidades de pixeles en cada eje para los casos estudiados (favorables y desfavorables) en LR, lo que se traduce en $[-24, 24]$ unidades de pixeles luego de obtener la estimación HR. Finalmente, para encontrar los parámetros de desplazamiento que definen la transformación entre dos imágenes se resume como:

$$\Delta i = \underset{\Delta u \in [-\Delta i_{max} \Delta i_{max}]}{\operatorname{argmin}} SAD_i(\Delta u), \tag{3.3a}$$

$$\Delta j = \underset{\Delta v \in [-\Delta j_{max} \Delta j_{max}]}{\operatorname{argmin}} SAD_j(\Delta v), \tag{3.3b}$$

Como se muestra en la Figura 3.5a, al aplicar la corrección de movimiento existen pixeles que pueden quedar fuera de las coordenadas de la imagen objetivo, que deben ser mapeados a las coordenadas origen. Para solucionar este problema se establece una restricción D sobre las coordenadas de destino, reduciendo el intervalo de cada eje en un factor de dos veces el desplazamiento máximo, calculada como $D = (\Delta i_{max} - \Delta i_{min})$. Con esta restricción, la resolución original de $m \times n$ pasa a $(m - D) \times (n - D)$, y como se grafica en la Figura 3.5b se genera un marco sin datos en la imagen resultante.

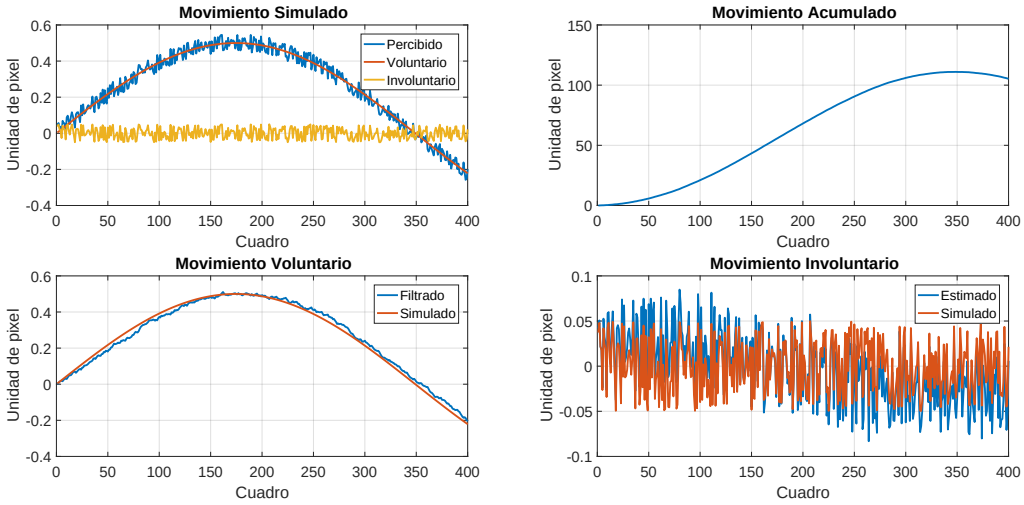


Fig. 3.6: Resultados de aplicación de filtro sobre movimiento.

3.3. Estabilización y filtrado de movimiento

El objetivo de la estabilización es ajustar todos los cuadros a una referencia determinada. Como el sistema de captura está compuesto de una secuencia de cuadros, entonces lo intuitivo es tomar la referencia como el primer cuadro de la secuencia, y el resto de cuadros alinearlos a ésta. Para poder determinar la transformación de cada cuadro con el primero se tienen dos opciones: comparación directa y acumulación. La comparación directa es aplicar la estimación de movimiento entre el cuadro actual y el primero. La acumulación aprovecha la linealidad de la transformación rígida para establecer que la diferencia entre el cuadro actual y el primero es la suma de todos los desplazamientos hasta ese punto. Suponiendo una escena estática y sólo vibraciones involuntarias, la suma acumulada tendrá un valor siempre en torno a un desplazamiento nulo, ya que sólo presentaría pequeñas variaciones. En una escena en movimiento voluntario con vibraciones involuntarias, la acumulación tendería a crecer en alguna de las direcciones para luego estabilizarse una vez que la escena se deja de mover de forma voluntaria.

En las gráficas de la Figura 3.6 se muestra un ejemplo de filtro de movimiento. Se simuló un movimiento percibido por el sistema como la suma de un movimiento voluntario y uno involuntario. Si se ajustan todos los cuadros a la primera referencia se tienen movimientos por sobre las 100 unidades de pixel, recortando gran parte de la imagen y perdiendo información. Si el movimiento se filtra, entonces se puede separar nuevamente en una estimación de movimiento voluntario e involuntario, estabilizando sólo el movimiento involuntario. Se utilizó un filtro pasabajos respuesta al impulso infinito (*Infinite Impulse Response*, IIR) autorregresivo de primer orden, que utiliza un factor de olvido para filtrar las altas frecuencias del movimiento acumulado:

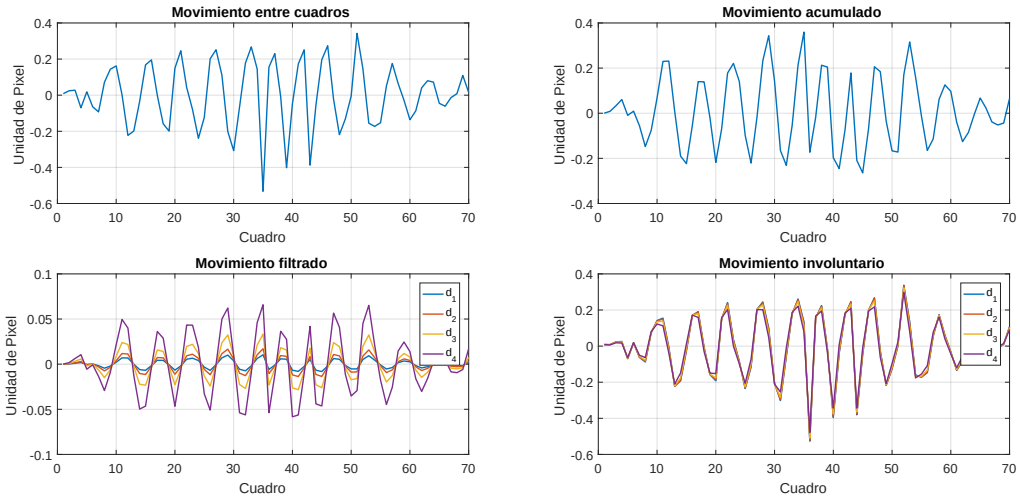


Fig. 3.7: Aplicación de filtro sobre eje i en caso favorable.

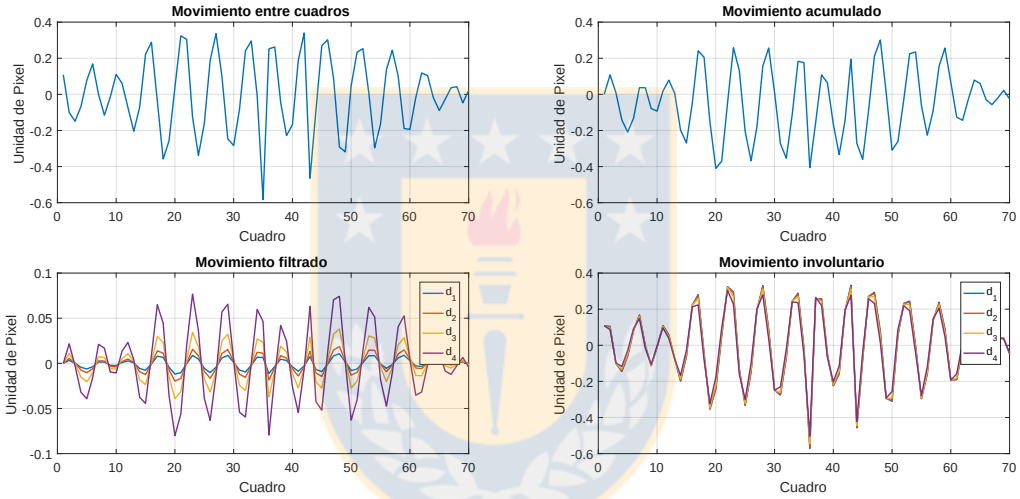


Fig. 3.8: Aplicación de filtro sobre eje j en caso favorable.

$$MV(k) = \delta MV(k-1) + (1-\delta) \cdot MA(k), \tag{3.4}$$

donde:

- $MV(k)$: Movimiento voluntario en el cuadro k.
- $Mp(k)$: Movimiento percibido en el cuadro k.
- δ: Factor de olvido del filtro.

Se aplicó el filtro sobre distintos casos reales del microscopio, principalmente sobre casos favorables desfavorables. Se muestra un caso de cada uno para evidenciar los resultados de la aplicación del filtro sobre el movimiento de los videos. Se utilizaron valores de factor de olvido

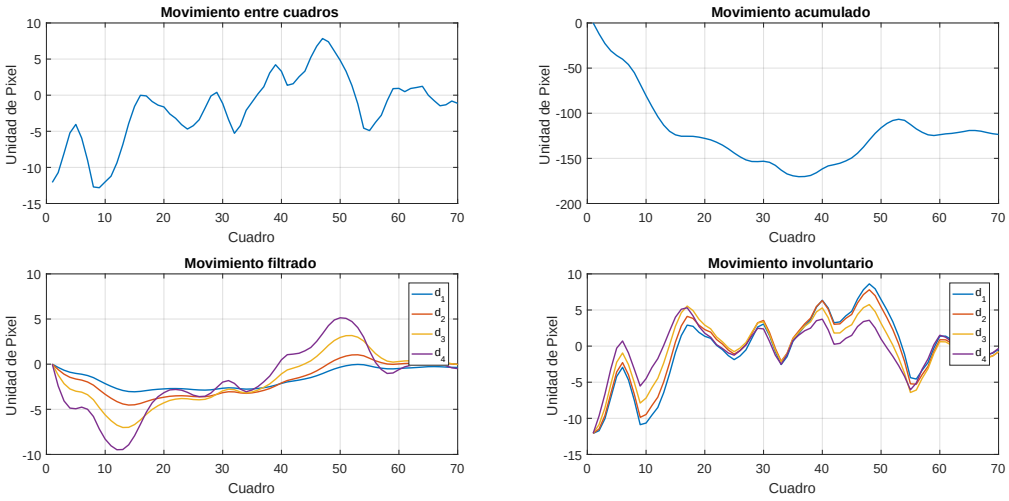


Fig. 3.9: Aplicación de filtro sobre eje i en caso desfavorable.

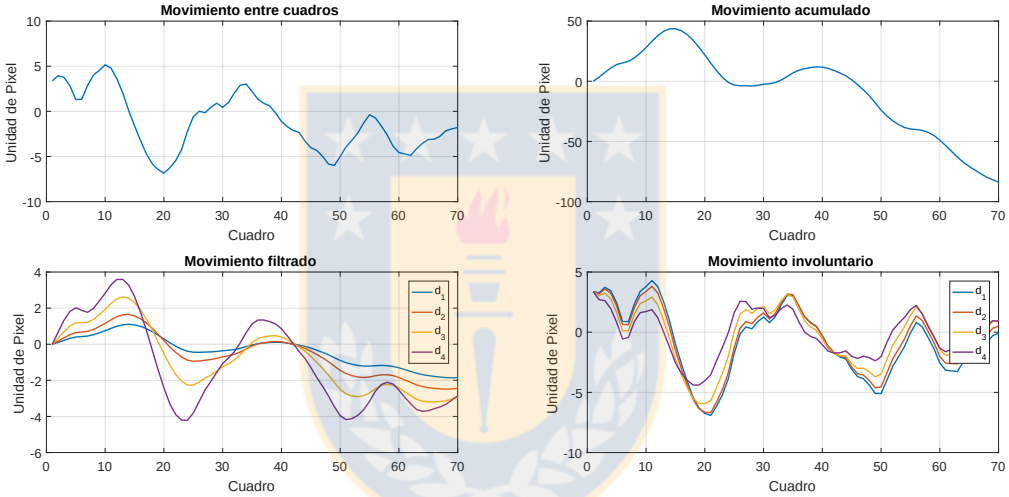


Fig. 3.10: Aplicación de filtro sobre eje j en caso desfavorable.

iguales a: $\delta_1 = 0{,}97$,$\delta_1 = 0{,}95$,$\delta_1 = 0{,}9$ y $\delta_1 = 0{,}8$. El primer caso es favorable y se muestra en la figuras 3.7 y 3.8, para los ejes i y j respectivamente. Este tipo de casos se asume una escena estática, y como se muestra en las gráficas de movimiento acumulado, éste en ambos ejes oscila siempre en torno al movimiento nulo. Filtrando el movimiento se espera una curva muy cercana a una línea recta en valor cero, que es lo que efectivamente ocurre. Así, el movimiento involuntario estimado es casi idéntico al mismo movimiento entre cuadros de video. Para una mejor aproximación al movimiento involuntario se utilizan valores de δ cercanos a 1, pero esto puede traer problemas en movimientos voluntarios de alta frecuencia.

Para la aplicación sobre el caso desfavorable se tienen movimientos voluntarios, en este caso particular causados por el movimiento de una mano humana. Como se muestra en las figuras 3.9 y 3.10, al separar en ambos ejes el movimiento voluntario del involuntario existen diferencias

según el valor de δ. La curva de movimiento voluntario depende del factor de olvido del filtro, cuanto más cercano a uno menos movimiento en alta frecuencia se puede ver en las curvas de δ_1 y δ_2, lo que deja más movimiento involuntario. Como la secuencia de video captura una mano en movimiento, no es pensado que una persona ejecute un movimiento voluntario de menos de 10 cuadros a una taza de 60 cuadros por segundo, que es lo que ocurre para δ_3 y δ_4. Como resultado de ambos análisis de casos, además de los no incluidos acá, se define el valor como $\delta = 0{,}95$, que permite filtrar adecuadamente los movimientos involuntarios de alta frecuencia, pero mantener la mayoría de los voluntarios intactos.

3.4. Consideraciones

Dado que la etapa de estimación de movimiento se necesita en el algoritmo SR-NUC, se extiende este análisis para tales motivos. Adicionalmente, el algoritmo de SR-NUC utiliza desplazamientos enteros en la etapa de alineación de imágenes, por lo que para este caso no es necesario considerar desplazamientos decimales en la búsqueda iterativa. Así mismo, en la etapa de estabilización, es suficiente con estabilizar movimientos enteros, ya que compensar y estabilizar movimientos sub-pixeles no trae ventajas mayores en comparación al costo numérico de la reconstrucción de imágenes. Por lo anterior, esta etapa es completamente calculada con números enteros y es posible de implementar directamente en un sistema digital sin necesidad de análisis de representación de números decimales.

Para el filtro de movimiento se utiliza aritmética decimal, pero como se implementó en lenguaje software C, no es necesario hacer análisis de representación en punto fijo, ya que la unidad de procesamiento utilizada tiene incluida aritmética de punto flotante.

Capítulo 4: Algoritmo de super resolución y no uniformidad simultáneos

En este capítulo se presenta el análisis del algoritmo de super resolución y corrección de no uniformidad en imágenes infrarrojas. Se presenta el estudio realizado para determinar de forma adecuada los elementos matemáticos que involucran la aplicación de SR-NUC, incluyendo las consideraciones de diseño para una aplicación sobre arquitectura hardware. Adicionalmente se presentan los resultados obtenidos mediante una implementación software en MatLab, haciendo pruebas de simulación de condiciones y sintonización.

4.1. Descripción del algoritmo

El algoritmo de SR-NUC propuesto en [49] está diseñado para aplicar super resolución y corrección de uniformidad simultáneamente en una secuencia de imágenes infrarrojas. Se usan términos de regularización espacial para explotar tanto las NLM y la ausencia de correlación espacial entre la escena y las fuentes de ruido de no uniformidad. Se presenta un algoritmo iterativo basado en la estrategia de minimización de gradiente.

Como se detalla en [49], primero se define un modelo de observación del sistema de captura infrarrojo:

$$\underline{y}_k = \mathcal{D}_k \mathcal{C}_k \mathcal{F}_k \underline{x} + \underline{b} + \underline{\eta}_k. \tag{4.1}$$

En la Ecuación (4.1), $\underline{y}_k$ representa una de las k imágenes de salida dentro de la secuencia, siendo de LR, con ruido y movimiento a nivel sub-pixel. Esta imagen de salida es una modificación de la escena real capturada $\underline{x}$ con HR y libre de ruido. La traslación y rotación de las muestras respecto a la escena son abstraídas por el operador $\mathcal{F}_k$. El operador $\mathcal{C}_k$ representa el PSF, que combina los efectos de difuminado del lente de la cámara, las turbulencias atmosféricas y la respuesta del sensor. El proceso de disminución de la resolución, debido a la limitación del IRFPA, se representa en el operador $\mathcal{D}_k$. El ruido de no uniformidad se modela asumiendo solamente el factor aditivo, dado que es el predominante en este tipo de sistemas, y se expresa

en el vector $\underline{b}$. El vector $\underline{b}$ no depende de k, ya que se asume que todas las imágenes dentro de la secuencia son capturadas en un corto periodo de tiempo con las mismas condiciones de operación. El término $\underline{\eta}_k$ es el vector de ruido de lectura aditivo, puede ser despreciado ya que su efecto sobre la imagen es mínimo.

Dado lo anterior, se formula una función de costo para estimar la imagen HR desde una secuencia de k imágenes LR con ruido:

$$\Phi(\underline{x}, \underline{b}) = \frac{1}{2} \sum_{k=1}^{P} \|\mathcal{D}_k \mathcal{C}_k \mathcal{F}_k \underline{x} + \underline{b} - \underline{y}_k\|_2^2 + \alpha \|\underline{x} - \mathbf{W}\underline{x}\|_2^2 + \beta[(\mathcal{D}_k \underline{x} - \mu_{\mathcal{D}_k \underline{x}})^T \underline{b}]. \tag{4.2}$$

Primero se considera un término de fidelidad que mide el MSE de la imagen HR estimada cuando es proyectada al espacio de las imágenes LR con ruido. Este término se minimiza cuando la proyección es similar a las imágenes LR. En segundo lugar se utiliza un proceso de regularización mediante NLM, que filtra el ruido de la imagen compensándolo y manteniendo los detalles de bordes y texturas de la imagen. El tercer término corresponde a la correlación cruzada entre la estructura espacial de la escena y el ruido de no uniformidad aditivo. Los parámetros α y β controlan el intercambio de peso entre los tres términos.

La estimación de los valores de $\underline{x}$ y $\underline{b}$ se realiza mediante optimización de descenso de gradiente, usando la función de costo de la Ecuación (4.2), obteniendo las siguientes ecuaciones derivando respecto a cada vector:

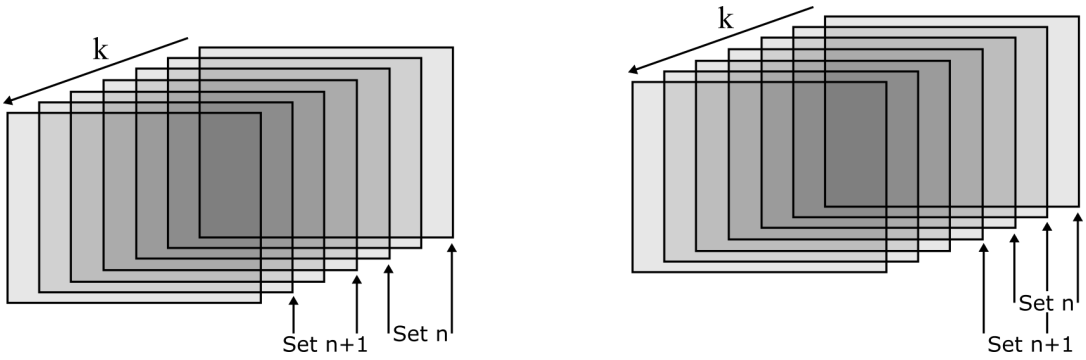
$$\nabla_{\underline{x}} \Phi(\underline{x}, \underline{b}) = \sum_{k=1}^{P} \mathcal{F}_k^T \mathcal{C}_k^T \mathcal{D}_k^T (\mathcal{D}_k \mathcal{C}_k \mathcal{F}_k \underline{x} + \underline{b} - \underline{y}_k) + \alpha(\underline{x} - \mathbf{W}\underline{x}) + \beta(\mathcal{D}_k^T \underline{b}), \tag{4.3}$$

$$\nabla_{\underline{b}} \Phi(\underline{x}, \underline{b}) = \sum_{k=1}^{P} (\mathcal{D}_k \mathcal{C}_k \mathcal{F}_k \underline{x} + \underline{b} - \underline{y}_k) + \beta(\mathcal{D}_k \underline{x} - \mu_{\mathcal{D}_k \underline{x}}). \tag{4.4}$$

El procedimiento para estimar el ruido de no uniformidad y la imagen en HR sin ruido, basado en descenso gradiente iterativo se escribe como:

$$\hat{\underline{x}}^{m+1} = \hat{\underline{x}}^{m} - \lambda \nabla_{\underline{x}} \Phi(\underline{x}, \underline{b}), \tag{4.5}$$

$$\hat{\underline{b}}^{m+1} = \hat{\underline{b}}^{m} - \lambda \nabla_{\underline{b}} \Phi(\underline{x}, \underline{b}), \tag{4.6}$$



(a) Sets independientes. (b) Ventana deslizante.

Fig. 4.1: Enfoques del algoritmo para aplicación en tiempo real.

en donde los valores estimados se actualizan en cada iteración con una tasa de aprendizaje λ que controla la velocidad de convergencia del algoritmo y debe ser escogido de tal forma de garantizar estabilidad en el algoritmo. Un valor muy pequeño de λ se traduce en un aprendizaje lento, requiriendo una cantidad elevada de iteraciones para llegar a un valor de mínimo estimado cercano al real. Un valor elevado de λ generaría sobre-aprendizaje e incluso, de ser excesivamente elevado, causar divergencia.

Por lo descrito anteriormente, la elección de la tasa de aprendizaje se debe realizar mediante una búsqueda de valores posibles para el cumplimiento de algún criterio determinado.

4.2. Aplicación en tiempo real

Lo primero a considerar es la aplicación final del algoritmo. Como se mencionó en la hipótesis, se espera implementar el algoritmo SR-NUC en lógica programable sobre un FPGA para lograr la corrección de no uniformidad y super resolución en imágenes infrarrojas en línea. Una aplicación en línea cambia el enfoque que se le da al algoritmo, el que está diseñado para un set de imágenes determinado disponibles en todo tiempo de ejecución. Este nuevo enfoque debe permitir utilizar la información de entrada que proviene del sistema de captura. Cada nueva imagen aporta información para el algoritmo y debe ser considerada.

Como se muestra en la Figura 4.1 se pueden definir dos enfoques: sets independientes de imágenes y ventana deslizante. La imagen 4.1a muestra el enfoque de sets independientes, en donde se considera al conjunto de imágenes de forma separada, sin incluir información de las imágenes ya utilizada en los sets anteriores. Este enfoque entrega una holgura mayor tiempo de cálculo y procesamiento para lograr la convergencia del algoritmo SR-NUC y calcular con mayor precisión los valores estimados $\underline{\hat{x}}$ y $\hat{\underline{b}}$ de las ecuaciones 4.5 y 4.6 respectivamente, ya que

es posible efectuar un mayor número de iteraciones en el proceso de descenso de gradiente antes de tener un nuevo set. La desventaja de este enfoque es la reducción del número de cuadros por segundo desde la fuente de imágenes, ya que por cada set de P imágenes LR con ruido sólo se obtiene una imagen de alta resolución sin ruido, siendo un factor de P veces menos cuadros por segundo.

Por otra parte, el enfoque de ventana deslizante mostrado en la Figura 4.1b, toma cada nuevo cuadro de imagen para incluirlo en el set y descarta el cuadro más antiguo. Al contrario de utilizar sets independientes, la ventana deslizante no reduce la cantidad de cuadros por segundo, pero ya no se cuenta con la holgura de tiempo para calcular los valores estimados $\underline{\hat{x}}$ y $\underline{\hat{b}}$, ya que para cada cuadro nuevo se debe comenzar a iterar inmediatamente para obtener la imagen HR sin ruido.

Dado el objetivo de diseño de un acelerador de cómputo del algoritmo SR-NUC, se decide utilizar un enfoque de ventana deslizante, puesto que se espera poder procesar una secuencia de video sin perder la tasa de actualización de cuadros por segundo.

4.3. Inicialización de vectores

En el algoritmo original de SR-NUC las condiciones iniciales de los valores estimados $\underline{\hat{x}}$ y $\underline{\hat{b}}$ se calculan para sólo un set de imágenes. El vector $\underline{\hat{x}}$ se aproxima inicialmente como una interpolación bicúbica de la primera imagen LR $\underline{y}_k$, luego sobre ésta se aplica el proceso de refinado mediante descenso de gradiente. El vector de no uniformidad $\underline{\hat{b}}$ se considera inicialmente como una matriz de ceros, que cambia su valor en cada iteración aproximándose a un estimado cercano al real esperado.

Dado que el caso de estudio del presente trabajo tiene un enfoque de aplicación en línea, se dispone de información previa entregada por el cálculo de SR-NUC en el set anterior. Para el ruido de no uniformidad, una vez que este valor sea estable dentro de un rango, se espera que su valor se mantenga dentro de tales márgenes, ya que la no uniformidad tiene una variación lenta en el tiempo respecto a las tasas de captura típica de los dispositivos. Esta suposición es respaldadas en distintos trabajos dentro de la literatura, por lo que es posible considerar el valor final de un set como el inicial del set siguiente. Se puede suponer entonces que el número de iteraciones necesarias para la convergencia del ruido de no uniformidad debe disminuir, ya que el enfoque propuesto tiene una ventaja por sobre el original.

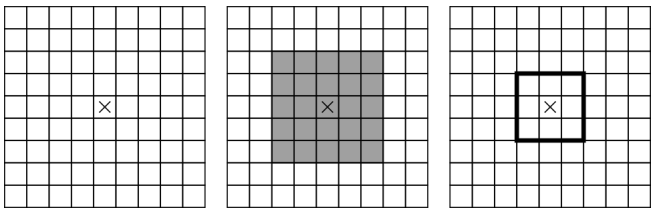


Fig. 4.2: Pixel objetivo, vecindad de búsqueda y ventana de comparación para NLM.

Para el vector de imagen estimada en alta resolución, no se puede afirmar lo mismo ya que pueden existir escenas con dinámica temporal que impidan suponer que una imagen será similar entre un cuadro de video y el siguiente. Las alternativas que existen son disntintas aproximaciones iniciales como interpolación bilineal y del vecino más cercano. La interpolación bilineal es mucho menos costosa en términos de cálculos y accesos a memoria que la bicúbica, ya que la bicúbica utiliza un radio de búsqueda de tamaño 16 para cada pixel y la bilineal los 4 pixeles contiguos al central, con un costo muy menor en calidad de reconstrucción de imagen. Por otra parte, el vecino más cercano entrega una reconstrucción muy pobre pero a un costo computacional muy bajo, ya que sólo replica el pixel cercano. Dado a que se trabajará sobre una plataforma hardware y no se quiere perder calidad, se propone utilizar como primera aproximación la interpolación bilineal.

4.4. Medias no locales

El algoritmo de NLM se describe como una búsqueda de autosimilitud de una imagen. Como se muestra en la Figura 4.2, primero se definen una vecindad de búsqueda, que se utilizará en cada pixel, y una ventana de comparación. El pixel objetivo se encuentra en el centro de la vecindad y se le asigna su ventana de comparación. Así mismo, para todos los pixeles dentro de la vecindad se tiene su propia ventana de comparación. Luego, como se visualiza en la Figura 4.3, la ventana central, correspondiente a la del pixel objetivo, es comparada mediante distancia euclidiana con el resto de las ventanas que se forman al tomar cada pixel dentro de la vecindad. Las distancias euclidianas son utilizadas para asignarle un peso a cada pixel dentro de la vecindad, que es inversamente proporcional a ésta, es decir, entre mayor sea el valor de la distancia euclidiana menor es el peso asignado. Como resultado entre más parecido sean las ventanas objetiva y la de comparación, mayor es el valor del peso asignado.

Matemáticamente, el algoritmo NLM se describe como sigue. Dada una imagen discreta $v = \{v(i)|i \in \mathbf{I}\}$, el valor valor estimado $NL(v)(i)$ se calcula como la media ponderada de todos los pixeles en la imagen,

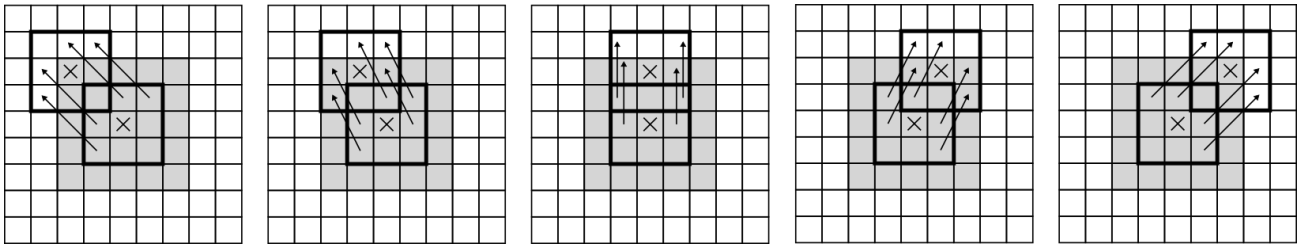


Fig. 4.3: Concepto inicial de algoritmo NLM.

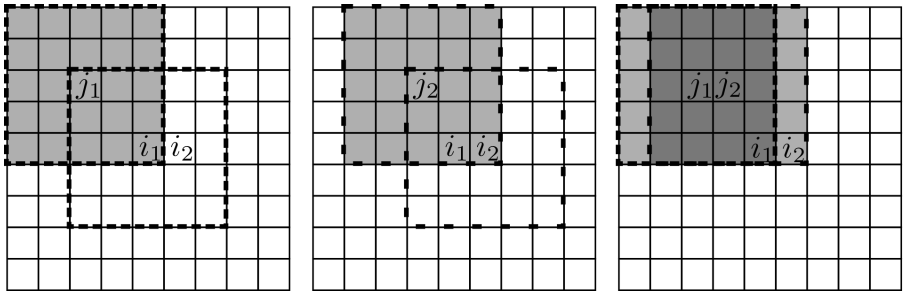


Fig. 4.4: Recurrencia de datos calculados NLM.

$$NL(v)(i) = \sum_{j \in \mathbf{I}} W(i,j)v(j), \tag{4.7}$$

en donde los pesos $\{W(i,j)\}_j$ dependen de la similitud que existe entre los pixeles i y j, satisfaciendo la condición $0 \leq W(i,j) \leq 1$ y $\sum_j W(i,j) = 1$.

Para determinar la similitud entre los pixeles de la imagen, se define un sistema de vecindades en $\mathbf{I}$. Un sistema de vecindades en $\mathbf{I}$ es una familia $\{\mathcal{N}_i\}_{i \in \mathbf{I}}$ de subconjuntos de $\mathbf{I}$ tal que para todos los $i \in \mathbf{I}$ se cumple $\{i \in \mathcal{N}_i\}$ y $\{j \in \mathcal{N}_i \Rightarrow i \in \mathcal{N}_j\}$. La restricción de v a la vecindad $\mathcal{N}_i$ se escribirá como $v(\mathcal{N}_i)$

$$v(\mathcal{N}_i) = (v(j), j \in \mathcal{N}_i). \tag{4.8}$$

Estas vecindades pueden tener distintas formas dependiendo de la aplicación, pero se utiliza como estándar una vecindad cuadrada de tamaño $n \times n$. La similitud entre dos pixeles i y j dependerán de la similitud de los vectores $v(\mathcal{N}_i)$ y $v(\mathcal{N}_j)$. Los pixeles con un vecindades similares a $v(\mathcal{N}_i)$ tendrán ponderaciones mayores en la media. Para determinar la similitud entre los vectores $v(\mathcal{N}_i)$ y $v(\mathcal{N}_i)$ se puede calcular una distancia Euclidiana Gaussiana ponderada, $\|v(\mathcal{N}_i) - v(\mathcal{N}_j)\|^2_{2,g}$. Luego, los pesos asociados con la distancia cuadrática se definen por

$$W(i,j) = \begin{cases} \omega(i,j) & : \text{si } j \in \mathcal{N}_i \\ 0 & : \text{otro caso} \end{cases}, \tag{4.9}$$

con

$$\omega(i,j) = \frac{1}{Z(i)} e^{\frac{-\|v(\mathcal{N}_i) - v(\mathcal{N}_j)\|^2_{2,g}}{h^2}} \tag{4.10}$$

en donde h controla el decaimiento de la función exponencial y $Z(i)$ es el factor normalizador

$$Z(i) = \sum_j e^{\frac{-\|v(\mathcal{N}_i) - v(\mathcal{N}_j)\|^2_{2,g}}{h^2}} \tag{4.11}$$

A simple vista, pensar en implementar este algoritmo de forma directa en una arquitectura digital en línea trae consigo múltiples desafíos asociados principalmente al uso de memorias. Esto se infiere de la Ecuación (4.10), en donde para cada pixel se debe calcular la distancia euclidiana de cada vecindad $v(\mathcal{N}_j)$ dentro de $v(\mathcal{N}_i)$, siendo un total de $4 \times \text{size}(v(\mathcal{N}_i))^2$ accesos por pixel. Adicionalmente se debe considerar el tamaño de la imagen $\mathbf{I}$, aumentando el número de accesos de forma lineal respecto al tamaño de la imagen y de forma cuadrática al tamaño de la vecindad. Dado lo anterior, el objetivo principal es buscar una forma de minimizar los accesos a memoria ya que es costoso tanto en recursos hardware como en tiempos de espera.

Analizando cualitativamente el algoritmo NLM se puede notar que habrá recurrencia de cálculos aritméticos al realizar el cómputo de la distancia euclidiana. A modo de ejemplo, como se muestra en la Figura 4.4, primero se desea calcular la distancia euclidiana entre las vecindades del pixel i_1 y j_1 marcada en color gris tenue. Luego se desea calcular la distancia euclidiana entre las vecindades a del pixel i_2 y j_2 marcada en color gris tenue. Se debe notar que la posición de j_1 respecto a i_1 es la misma de j_2 respecto a i_2, que corresponde a la esquina superior izquierda. Como los pixeles i_1 e i_2 son vecinos, y j_1 y j_2 están en la misma posición relativa, se produce una intersección de cálculos hechos al momento de obtener la distancia euclidiana, intersección marcada en gris oscuro. Si extendemos este análisis para la misma posición relativa de vecindades, entonces para obtener el valor de distancia euclidiana basta solamente con hacer un cálculo usando el nuevo pixel, ya que todas las demás distancias de la misma son inferidas de las intersecciones de las vecindades anteriores. Visto de otro modo, cada una de las distancias individuales servirá para múltiples distancias euclidianas entre vecindades de la misma posición relativa, esto se muestra en la Figura 4.5, en donde se reúnen 9 ventanas de 3×3 y el cálculo individual de la vecindad j_5 se utiliza en las que la rodean. Se le llamará celda

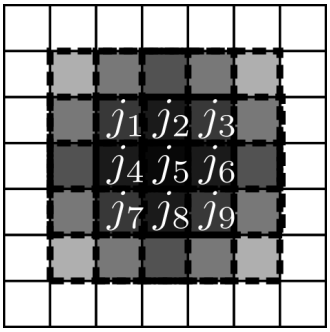


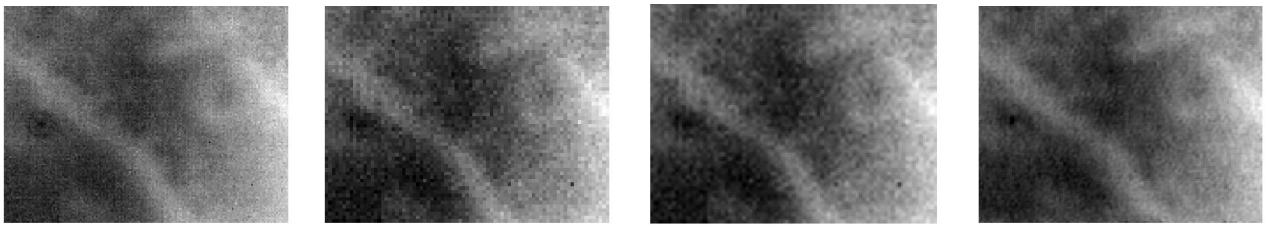
Fig. 4.5: Recurrencia de datos calculados NLM.

al conjunto de estás recurrencias relativas, siendo un vector de distancias individuales particular para esta posición relativa entre vecindades. Extendiendo nuevamente el análisis hecho para la posición relativa mencionada, podemos concluir que en todas las posiciones relativas se repite la misma particularidad. Lo anterior permite pensar en un diseño en lógica hardware secuencial, aprovechando la modalidad de datos de entrada que es pixel a pixel, generando cada una de las celdas que correspondan. Finalmente, para obtener el valor de los pesos $\omega(i, j)$ es necesario tomar solamente el grupo de pixeles correspondientes de cada una de las celdas generadas y utilizarlos dentro de la Ecuación (4.10). Dado a que la finalidad del NLM es asignar un peso inversamente proporcional a la diferencia entre ventanas, cuando la distancia es muy elevada el peso asignado tiende a cero, siendo despreciable dentro de los cálculos. En caso de que el pixel central esté rodeado de una vecindad muy distante, en una escena natural significará que no es válido, y será reemplazado automáticamente por el promedio de la vecindad.

4.5. Prueba de SRNUC

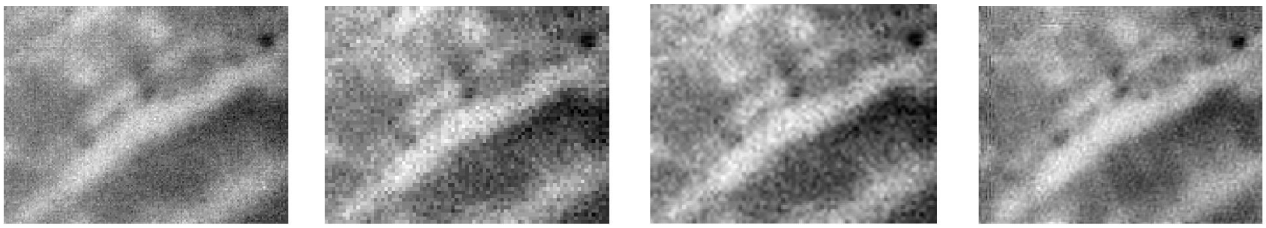
Se probó la efectividad del algoritmo SR-NUC sobre distintas secuencias de imágenes obtenidas por el microscopio infrarrojo SOFRADIR-EC IRE 320M . Se utilizó como base el script para Matlab puesto a disposición por los autores de [49], con la sintonización correspondiente a los parámetros de aprendizaje de las ecuaciones 4.6 y 4.5, utilizando 4 cuadros consecutivos, una vecindad de 3×3 para NLM. Sobre el script base se hicieron los cambios propuestos en las secciones anteriores.

Para la sintonización de parámetros se hizo una búsqueda empírica sobre un set de valores de λ, β y μ, utilizando PSNR como métrica de comparación. El PSNR se obtuvo generando artificialmente imágenes de baja resolución y con ruido añadido a partir de una imagen conocida, proveniente desde el microscopio IR de trabajo. La búsqueda empírica analizó los tres parámetros de forma separada, dejando los otros dos fijos en el momento del análisis. La combinación de



(a) Imagen original. (b) Imagen decimada. (c) Interpolación bicúbica (d) Resultado SR-NUC.

Fig. 4.6: Resultados de SR-NUC sobre una imagen microscópica de la yema de un dedo.



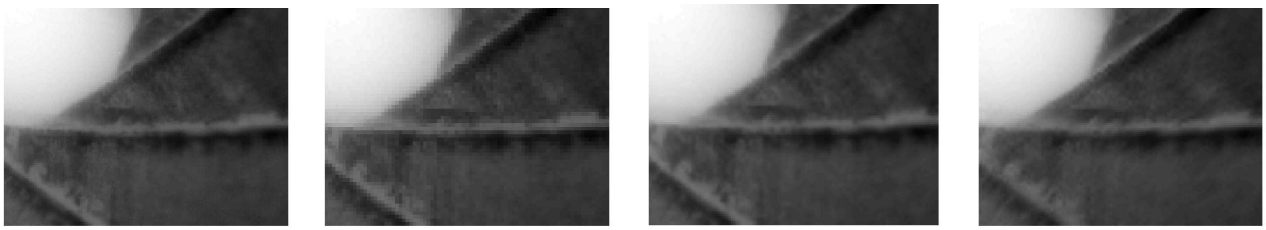
(a) Imagen original. (b) Imagen decimada. (c) Interpolación bicúbica (d) Resultado SR-NUC.

Fig. 4.7: Resultados de SR-NUC sobre una imagen microscópica del dorso de una mano.

los tres mejores resultados de PSNR, es decir el mayor valor alcanzado, se escogió como la sintonización utilizada. Esta corresponde a: $\lambda = 0{,}6$,$\alpha = 0{,}05$ y $\mu = 0{,}01$.

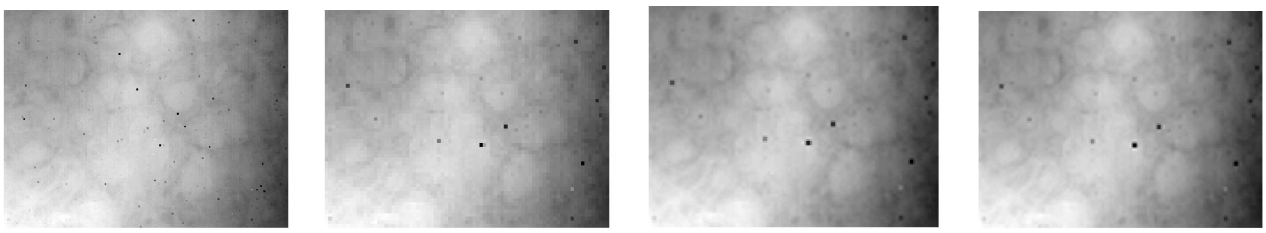
Las imágenes fueron reducidas a la mitad de su resolución original mediante decimación. El objetivo es, a partir de un set de imágenes decimadas, aproximarse a la imagen de resolución original. Para motivos de comparación, en cada set de prueba se utiliza la imagen obtenida con interpolación bicúbica, par determinar de forma visual si hay mejoras o no. Los set de pruebas a mostrar son cuatro, dos situaciones favorables y dos no favorables.

Los casos desfavorables son mostrados en las figuras 4.6 y 4.7, y corresponden a la yema de un dedo y a un dorso de una mano, respectivamente. Las imágenes originales (figuras 4.6a y 4.7a) fueron reducidas a la mitad de resolución (4.6b y 4.7b). Estos casos presentan mayor ruido de no uniformidad apreciable dado el movimiento que existe en la escena. Las imágenes obtenidas con el algoritmo SR-NUC, mostradas en las figuras 4.6d y 4.7d, tienen mayor parecido



(a) Imagen original. (b) Imagen decimada. (c) Interpolación bicúbica (d) Resultado SR-NUC.

Fig. 4.8: Resultados de SR-NUC sobre una imagen microscópica de una placa metálica.



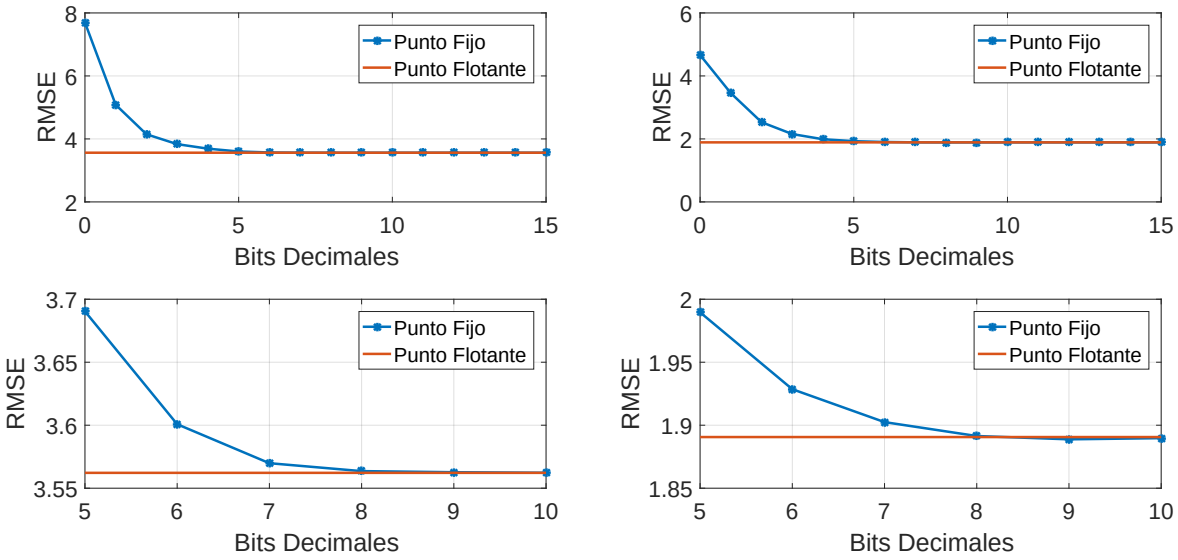
(a) Imagen original. (b) Imagen decimada. (c) Interpolación bicúbica (d) Resultado SR-NUC.

Fig. 4.9: Resultados de SR-NUC sobre una imagen microscópica de un trozo de poliestireno expandido.

visual a la imagen original que la aproximación mediante interpolación bicúbica presentada en las imágenes 4.6c y 4.7c. Las imágenes obtenidas con interpolación bicúbica presentan granularidad y definición de formas pobre. Los resultados obtenidos con SR-NUC presentan menos granularidad y mayor definición de formas, y además un mayor parecido a la imagen original.

Los casos favorables son mostrados en las figuras 4.8 y 4.9, y corresponden a una pantalla de metal y a un trozo de poliestireno expandido, respectivamente. Las imágenes originales (figuras 4.8a y 4.9a) fueron reducidas a la mitad de resolución (4.8b y 4.9b). Estos casos presentan poco ruido de no uniformidad apreciable dado el bajo movimiento que existe en la escena. Las imágenes obtenidas con el algoritmo SR-NUC, mostradas en las figuras 4.8d y 4.9d, tienen un parecido visual a la imagen original similar a la aproximación mediante interpolación bicúbica presentada en las imágenes 4.8c y 4.9c. La principal diferencia se aprecia en los bordes presentes en las imágenes, un efecto denominado en inglés como *aliasing* que altera la percepción de bordes y líneas que no van en las direcciones base (ejes x e y). La imagen obtenida con SR-NUC tiene una menor componente de aliasing en comparación a la imagen obtenida con interpolación bicúbica.

En ambos casos, favorables y desfavorables, se obtienen mejoras de resolución y definición de imagen respecto a la interpolación bicúbica. Según el trabajo presentado en [49], su propuesta es comparable con métodos actuales en la literatura, como por ejemplo algoritmos complejos basados en MAP. Se concluye que el algoritmo resuelve satisfactoriamente los problemas de baja resolución y no uniformidad de imágenes provenientes del microscopio infrarrojo SOFRADIR-EC IRE 320M ,incluyendo los cambios propuestos sobre cada etapa presente en éste.



(a) RMSE en poliestireno expandido.

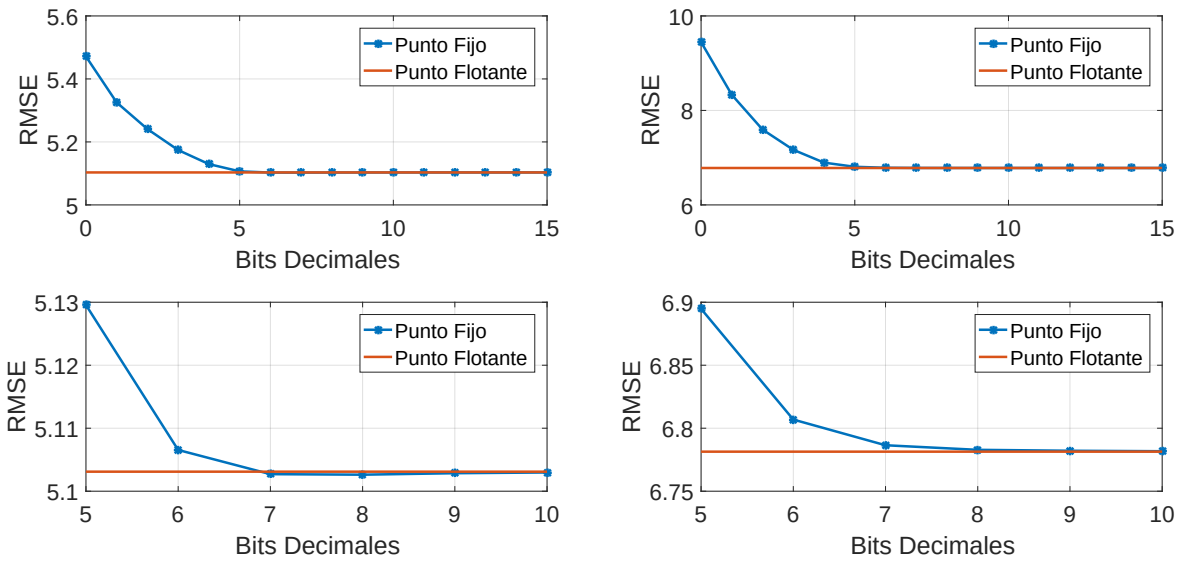
(b) RMSE en placa de metal.

Fig. 4.10: Evolución de RMSE para distintas representaciones de punto fijo en casos favorables (arriba rango completo, abajo rango de interés en acercamiento).

4.6. Representación de punto fijo

Para la selección del número de bits se modificó el script escrito en Matlab para aplicar el algoritmo SR-NUC reiteradas veces cambiando la representación de punto flotante a punto fijo con distinto número de bits de representación decimal. Se utilizó como métrica el valor RMSE, que es un indicador de similitud entre dos imágenes. En este caso se comparó el RMSE entre la imagen de alta resolución original con las generadas con decimación y aumentadas en resolución y reducción de ruido con SR-NUC, para las distintas representaciones de bits decimales, además se utilizó el valor RMSE de punto flotante. El objetivo es buscar la mínima representación posible que entregue un resultado cercano al punto flotante, ya que aumentar la cantidad de bits de representación presenta una mayor carga computacional, pero llega un punto en que aumentarla más ya no genera diferencias en resultados.

Los resultados obtenidos se presentan en las gráficas de las figuras 4.10 y 4.11. Cada gráfica indica el valor RMSE de referencia obtenido con punto flotante en una línea continua. La línea con puntos de cada gráfica indica el valor RMSE obtenido para distintas cantidades de representación de bits decimales. La búsqueda realizada fue desde cero hasta quince bits de representación decimal, y cada gráfica tiene una igual pero con zoom a la zona de interés. Las cuatro gráficas son de los mismos cuatro casos de prueba hechos en la sección 4.5. Analizando las gráficas se puede observar que a medida que la cantidad de bits para representación decimal



(a) RMSE en dorso de una mano.

(b) RMSE en yema de un dedo.

Fig. 4.11: Evolución de RMSE para distintas representaciones de punto fijo en casos desfavorables (arriba rango completo, abajo rango de interés en acercamiento).

aumenta, disminuye el valor RMSE, sólo en casos particulares aumenta como en las figuras 4.10b y 4.11a en los bits 9 y 8 respectivamente. En todas se puede apreciar el mismo patrón de descenso de RMSE con una aproximación muy cercana entre la representación de 8 bits decimales y la punto flotante, luego de esa cantidad de bits de representación no hay mayor cambio. Se concluye que no es estrictamente necesario tener mayor cantidad que 8, ya que más allá de esa representación no hay una ganancia significativa en información.

Capítulo 5: Arquitectura digital en SoC

En este capítulo se presenta una descripción de la arquitectura hardware que implementa los algoritmos de corrección de no uniformidad y super resolución simultáneo y estabilización de imagen sobre un SoC. El capítulo se divide en tres secciones. La primera sección describe la arquitectura general del sistema diseñada sobre el SoC. La segunda sección describe la arquitectura sobre la lógica programable en harwdare, incluyendo los elementos de los algoritmos abordados en los capítulos 3 y 4 que se implementaron sobre el núcleo hardware (HwC). En la tercera sección se describe la configuración del procesador ARM Cortex A-9 incluido en el núcleo software (SwC), incluyendo pseudocódigo y funciones utilizadas dentro de la programación, para la implementación de algunos elementos de los algoritmos revisados en los capítulos 3 y 4.

5.1. Arquitectura general

Se diseñó una arquitectura híbrida que combina la utilización de lógica programable para tareas de alta intensidad computacional, y la utilización del procesador integrado para tareas de baja intensidad numérica. En la Figura 5.1 se muestra la estructura general de la plataforma de trabajo, la lógica programable se denomina HwC, el núcleo programable en C se denomina SwC. La comunicación y transferencia de datos entre HwC y SwC se realiza a través de la memoria memoria de acceso aleatorio (*Random Acces Memory*, RAM), que se divide en un área

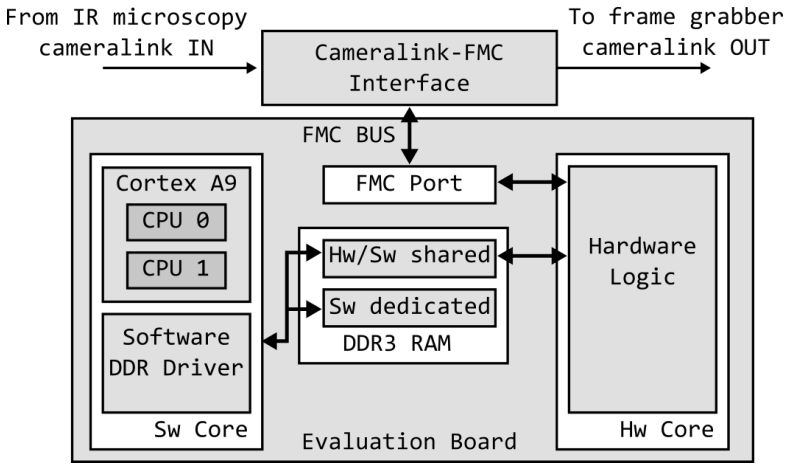


Fig. 5.1: Arquitectura general sobre el SoC.

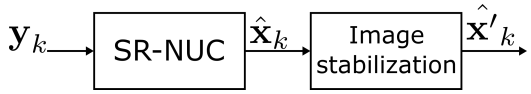


Fig. 5.2: Diagrama general de flujo de datos.

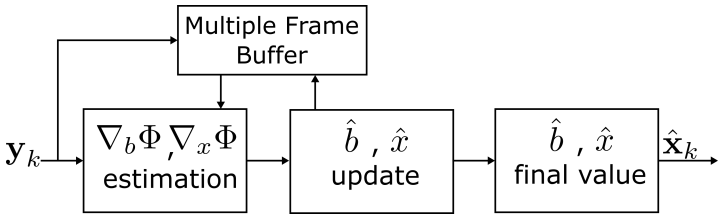


Fig. 5.3: Diagrama general para algoritmo SR-NUC.

dedicada para el SwC y un área mixta compartida entre ambos. Para la recepción y el envío de datos se utilizó una tarjeta de expansión con una interfaz camera-link conectada al puerto FMC, accesible desde la lógica programable en hardware.

Como se muestra en la Figura 5.2 el diagrama general divide la arquitectura digital en dos etapas: primero super resolución-corrección de no uniformidad y segundo estabilización de imagen. La imagen de entrada al sistema proveniente desde el microscopio infrarrojo se representa por $\mathbf{y}_k$, que es procesada en la etapa SR-NUC, obteniéndose una imagen estimada $\hat{\mathbf{x}}_k$ limpia de no uniformidad y con super resolución, compensadas de forma simultánea. La imagen $\hat{\mathbf{x}}_k$ es procesada luego por la etapa de estabilización de imagen, en donde se filtran los movimientos voluntarios e involuntarios y se compensan digitalmente estos últimos, obteniendo una imagen $\hat{\mathbf{x}'_k}$ estable espacialmente.

Como se muestra en la Figura 5.3 la arquitectura general del algoritmo SR-NUC se divide en cuatro bloques principales. El bloque *Multiple Frame Buffer* almacena los cuadros de imágenes y_k de las ecuaciones 4.3 y 4.4. El bloque *estimation* aplica todos los operadores para la estimación del error de las ecuaciones 4.3 y 4.4. El bloque *update* actualiza iterativamente los parámetros estimados $\hat{b}$ y $\hat{x}$ mediante el proceso de descenso de gradiente de las ecuaciones 4.5 y 4.6, utilizando las imágenes almacenadas en el buffer y las estimaciones. Los valores temporales de $\hat{b}$ y $\hat{x}$ son almacenados en el bloque *Multiple Frame Buffer* para el proceso de actualización. Una vez ocurridas las P iteraciones se almacenan los valores finales estimados $\hat{b}$ y $\hat{x}$ en el bloque *final value*.

La arquitectura general del algoritmo de estabilización mostrada en la Figura 5.4 se divide en seis bloques. El bloque *projected* genera las dos proyecciones de la imagen en las dos direcciones espaciales i e j para la reducción de dimensionalidad, para cada cuadro de video $\hat{x}$. Se almacenan ambas proyecciones en el bloque *projections buffers* para conservar siempre la proyección anterior

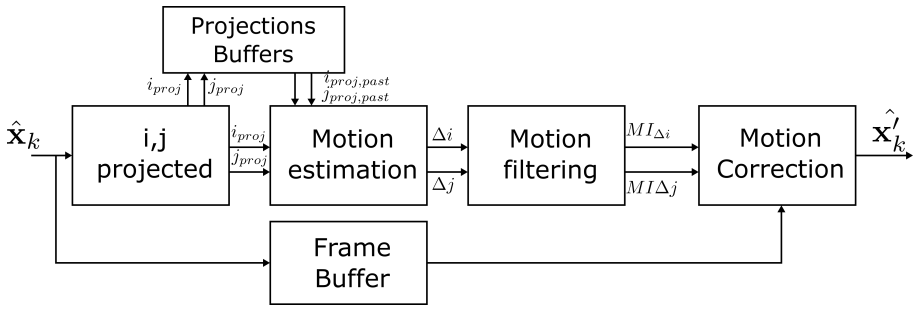


Fig. 5.4: Diagrama general de estabilización digital de movimiento.

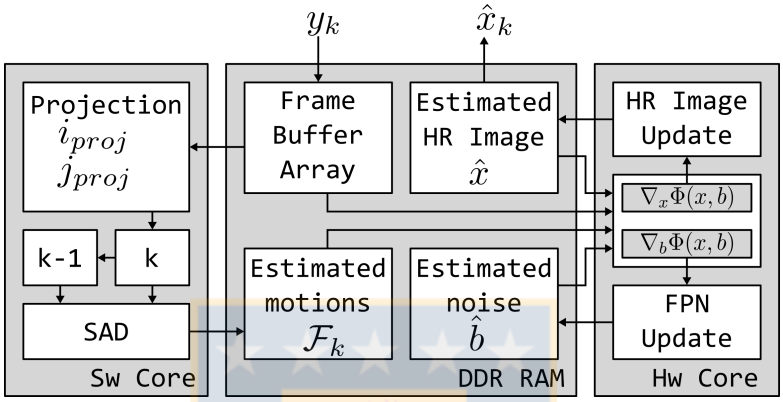


Fig. 5.5: Arquitectura general SR-NUC sobre SoC.

a la actual. La proyección actual y la anterior son comparadas en el bloque *motion estimation* para estimar el movimiento que existe entre el cuadro actual y el anterior, Δi y Δj para las direcciones i e j respectivamente. Los movimientos en ambas direcciones son filtrados en el bloque *motion filtering*, que separa los movimientos voluntarios de los involuntarios. Finalmente el bloque *motion correction* corrige el movimiento involuntario en la imagen $\hat{x}$ almacenada temporalmente en el bloque *frame buffer*.

La arquitectura general del algoritmo SR-NUC sobre el SoC es mostrada en la Figura 5.5. En esta se separan las tareas y distribución de datos a través de las dos jerarquías principales: SwC y HwC. Las tareas se realizará en el HwC son las de estimación y actualización de los parámetros $\hat{b}$ y $\hat{x}$, pero en el caso de la estimación de movimiento entre cuadros, utilizada en el operador $\mathcal{F}$ en el proceso de alineación de los cuadros, se calculará en el SwC ya que sólo requiere ser calculada una vez por cuadro y la intensidad numérica de cálculo se ve reducida al disminuir la dimensionalidad con las proyecciones en los ejes.

La estabilización de movimiento se trabajó completa sobre el SwC, ya que se aplica una vez por cada cuadro y la carga computacional es lo suficientemente baja para hacerlo entre cuadros de imagen. Se requiere de almacenar las proyecciones del cuadro anterior $k-1$ para actualizar el vector de movimiento de la Ecuación (3.4) de ambos ejes. Una vez actualizado el vector de

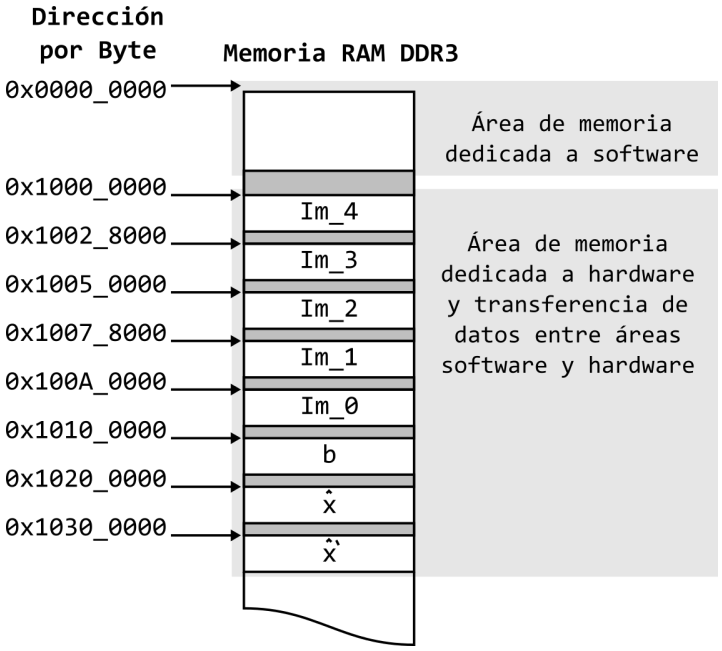


Fig. 5.6: Distribución de datos en memoria RAM.

movimiento se aplica la corrección sobre la imagen k actual y se almacena en un nuevo búfer de imagen para su posterior despliegue/envío.

Dado a que el algoritmo SR-NUC trabaja en dos tamaños de resolución espacial (HR y LR) los datos hacia hardware son transmitidos desde la RAM en líneas de la imagen de HR, y para la LR se transmite una línea cada línea por medio de la HR. La distribución de datos se muestra en la Figura 5.6, donde se separan las dos áreas principales: área de memoria dedicada al SwC y el área dedicada a la transferencia de datos entre SwC y HwC. La sección de RAM de software es completamente controlada por el procesador, por lo que no es posible conocer la distribución de los datos. En el caso HwC la distribución de memoria se asigna manualmente en el IDE Vivado de Xilinx al momento del diseño hardware, la que comienza en la dirección 0x1000_000, correspondiente a los 256 MB superiores de la memoria. La configuración sitúa los 5 cuadros de video infrarrojo necesarios de forma contigua, luego el parámetro de no uniformidad b, la imagen $\hat{x}$ de HR estimada y la imagen $\hat{x}'$ HR estabilizada espacialmente.

El flujo general de datos y trabajo se resume en el algoritmo 1. En el paso 1 se escribe el nuevo cuadro de imagen proveniente desde cameralink en memoria RAM, en la posición que contiene el cuadro más antiguo. En el segundo paso se procesan los cuatro cuadros de imagen más nuevos, ya que el más antiguo será sobre-escrito durante este proceso, y se estima el movimiento de los tres cuadros anteriores al nuevo con respecto a éste. Desde el paso 3 al 4 se estiman los parámetros x y b, para obtener una imagen $\hat{x}$ de alta resolución estimada. En el paso 5 se calculan las proyecciones de $\hat{x}$ sobre sus ejes y se estima, mediante el algoritmo de

Algoritmo 1 Flujo de trabajo

1: Escritura del nuevo cuadro k en la posición más antigua.
2: Estimar movimiento entre cuadros respecto a la referencia.
3: **for** j=1 to P **do**
   Estimar $\Delta_x\Phi(x,b)$ y $\Delta_b\Phi(x,b)$
4: **end for** Finaliza estimación $\hat{x}$
5: Calcular proyecciones de $\hat{x}$ y SAD.
6: Filtrar movimiento y compensar movimiento.

SAD, la diferencia de desplazamiento entre el último cuadro $\hat{x}$ y el anterior. Finalmente, en el paso 6, se filtra el movimiento para la separación de los involuntarios de los voluntarios, y se compensa.

5.2. Núcleo hardware

En esta sección se detalla la parte del diseño sobre HwC del algoritmo SR-NUC, puesto que la estabilización de imágenes sólo se aplica en SwC . Las ecuaciones 4.3 y 4.4, para la actualización de parámetros de SR-NUC, se basan en operadores matemáticos que se aplican reiteradas veces sobre el set de y_k imágenes. Se diseñó una arquitectura modular que permitió reutilizar los módulos y simplificar la organización del diseño. Como se ejemplifica con el operador de decimación en la Figura 5.7, se estableció una unidad modular como una caja operadora que recibe una señal de imagen de entrada y la entrega procesada en un mismo formato. Las unidades modulares que requieren más complejidad se comunican con la siguiente mediante cuatro señales digitales de entrada/salida. En la Figura 5.8 se muestra un ejemplo de las señales del protocolo diseñado: la señal *clk* es el reloj al que se transmiten los datos de cada pixel de imagen, la señal *pixel* es un bus de datos que contiene cada pixel de la imagen, la señal *first data* indica cuando se envía un stream de H pixeles horizontales de la imagen, y la señal *no data* indica cuando se esta procesando un streaming de datos aún no listos en una nueva linea de la imagen. Las señales *first data* y *no data* se mantienen un ciclo de reloj en alto para cada nueva línea de la imagen de entrada, por lo que cada uno de los módulos dentro del diseño debe contener la información previa de la resolución de la imagen a procesar, de esta forma utilizar los datos correctamente.

El diagrama mostrado en la Figura 5.9 corresponde al diseño modular para el cálculo y actualización de los parámetros de SR-NUC, desarrollado en las ecuaciones 4.3, 4.4, 4.5 y 4.6.

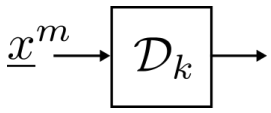


Fig. 5.7: Módulo operador de decimación de imagen.

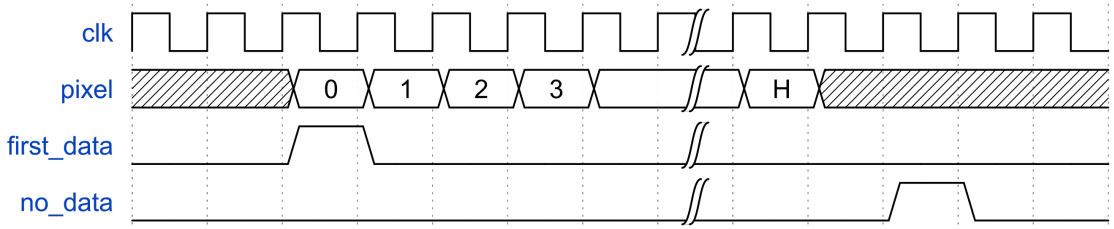


Fig. 5.8: Protocolo de comunicación entre módulos.

A continuación se describe el diseño de cada módulo diseñado sobre el HwC. Para términos de simplificación de los diagramas se ha definido *fd* para hacer referencia a *first data* y *nd* para hacer referencia a *no data*.

5.2.1. Decimación

El operador de decimación, representado por $\mathcal{D}$, omite datos de pixel por medio y línea por medio de imágenes en HR para llevarlas a LR. La arquitectura es mostrada en la Figura 5.10, formada por una máquina de estados y un buffer de línea. La máquina de estados controla las señales de lectura y escritura del buffer y las señales de comunicación. El buffer de línea tiene un tamaño de la mitad del ancho horizontal de la imagen en alta resolución.

Como se grafica en la Figura 5.11, la máquina de estados comienza con la señal *fd_in* que indica que se reciben datos desde *data_in*, separando dos mitades de operación. La primera es sólo la escritura en el buffer de los datos impares con la señal *we_en* en alto para la escritura. Los datos se guardan en orden desde la dirección 0 hasta la dirección h (mitad menos uno del tamaño horizontal de la imagen). La segunda mitad operación comienza a la mitad de la escritura en el buffer y, al mismo tiempo que se escribe, se leen los datos hacia la salida indicando con con *fd_out* en comienzo. Durante el envío la señal *sl_out* es 1 para escoger con el multiplexor los datos desde el buffer. A la siguiente línea, y todas las pares, la señal de salida es solamente *nd_out*, indicando que no hay datos que procesar, es decir, se salta la línea.

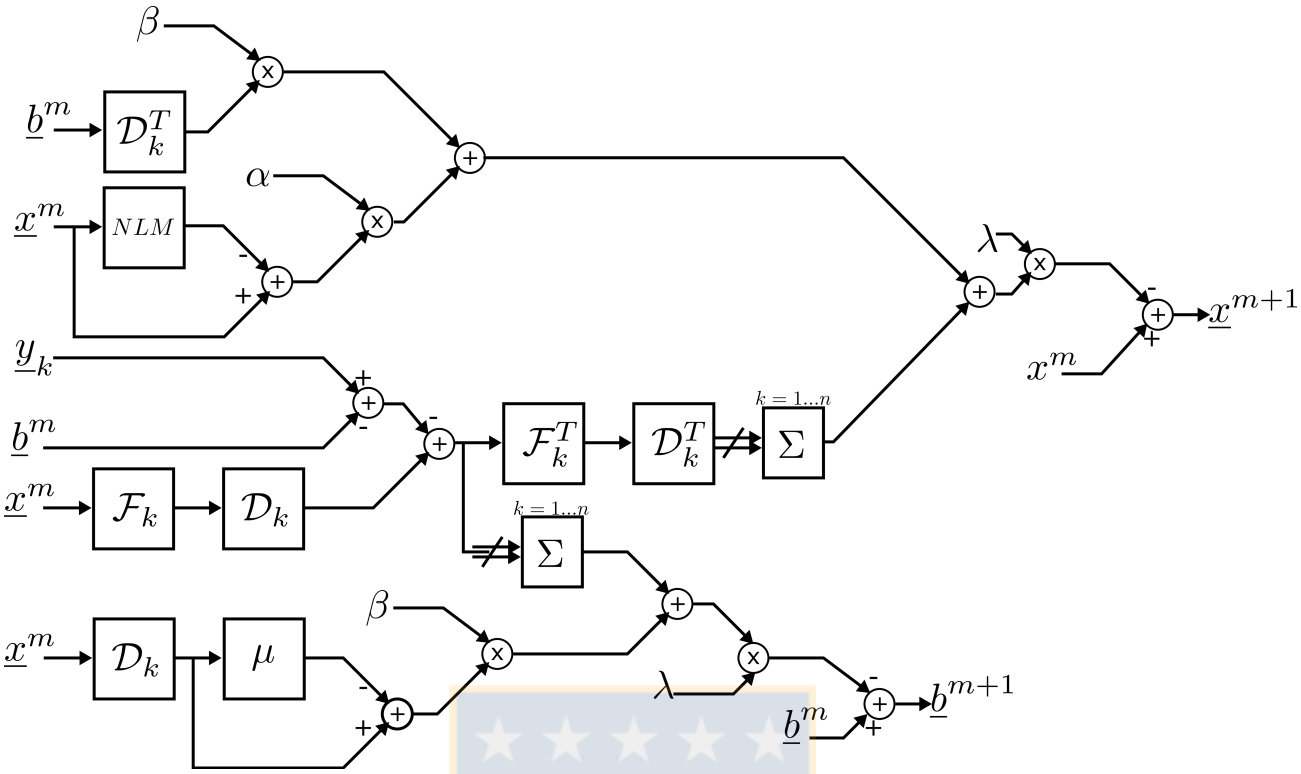


Fig. 5.9: Diagrama modular de arquitectura de SR-NUC sobre el HwC.

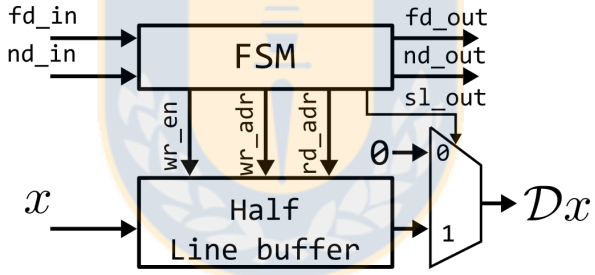


Fig. 5.10: Arquitectura hardware para módulo decimación $\mathcal{D}$.

5.2.2. Interpolación

El operador de interpolación, representado por por $\mathcal{D}^{\mathcal{T}}$, debe generar imágenes en HR a partir de los datos de una imagen en LR. La arquitectura es mostrada en la Figura 5.12, formada por una máquina de estados, dos buffer de línea del tamaño de una línea LR y registros para almacenamiento temporal y sincronización. Ya que la interpolación utilizada es la bilineal, las líneas impares de las imágenes en HR utilizarán sólo una línea de la imagen en LR, y para las líneas pares se utiliza el promedio dos líneas de la imagen en LR. El multiplexor manejado por la señal *line_sl* selecciona ya sea la línea correspondiente en LR para una línea impar en HR con los valores 0 y 1, o el promedio con el valor 2, el que es calculado con una suma y un corrimiento aritmético hacia la derecha (Arithmetic Right Shift (ARS)) equivalente a dividir

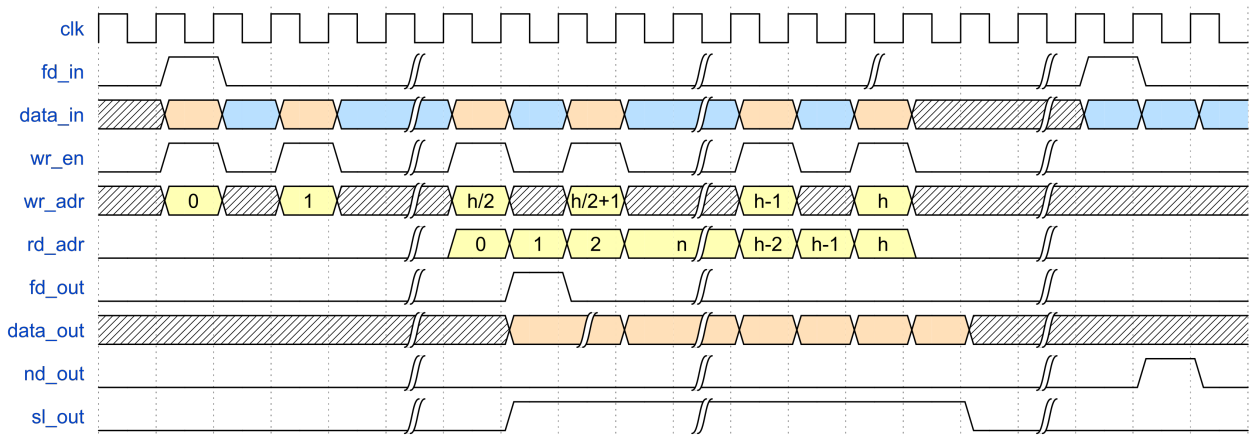


Fig. 5.11: Señales de control para módulo decimación $\mathcal{D}$.

por dos. En cualquiera de los tres casos debe deben generar los pixeles intermedios, calculado como la media entre el pixel actual y el anterior utilizando nuevamente una operación suma con un ARS y seleccionando la salida con la señal *px_sl* según corresponda.

Las señales en el tiempo de la máquina de estados son graficadas en la Figura 5.13, que se divide en dos partes: cuando se recibe *fd_in* y cuando se recibe *nd_in*. Cuando se recibe *fd_in* existe una línea de entrada válida de la imagen en la señal *data_in*, la que es almacenada completamente en el buffer de linea con los datos más antiguos, mientras el otro buffer se mantiene intacto. Esto se observa en las señales de habilitación de escritura *wr_en1* y *wr_en2*, las que nunca están en estado lógico 1 en un mismo ciclo de reloj. Adicionalmente la línea utilizada para el cálculo de los datos de salida se extrae del otro buffer que contiene los datos de la línea anterior. Para la siguiente línea se intercambia los buffer para lectura y escritura, esto se refleja en la señal *line_sl* que alterna su valor entre 1 y 0, que es la que controla la salida del multiplexor de selección de línea. En la parte en que se recibe la señal *nd_in* se debe generar una línea intermedia en HR a partir de dos líneas LR. Esto se efectúa utilizando los dos buffers de línea que contienen los datos de las últimas dos líneas en LR, siendo lo principal de esta parte la señal *line_sl* que se mantiene en valor 2 para seleccionar el promedio de ambas líneas. Para ambas partes las señales *fd_out* y *px_sl* se comportan de la misma formas, la señal *fd_out* se mantiene en 1 durante un ciclo de reloj para el primer dato de salida y la señal *px_sl* alterna entre 0 y 1 para seleccionar una salida en *data_out* entre un pixel original o uno promediado.

5.2.3. Alineación espacial de imagen

Como se presentó anteriormente, la alineación espacial de la imagen se trabajó como desplazamientos horizontales y verticales enteros. Bajo este método se define una forma de trabajo

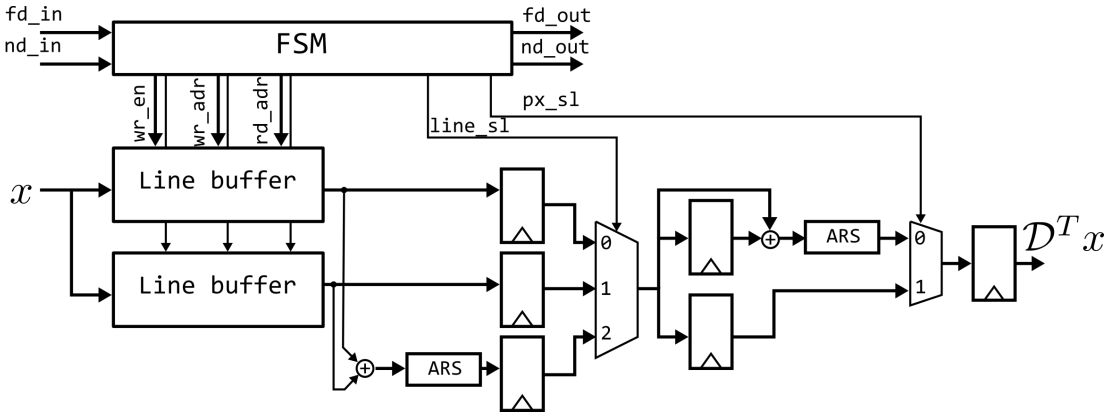


Fig. 5.12: Arquitectura hardware para módulo interpolación $\mathcal{D}^{\mathcal{T}}$.

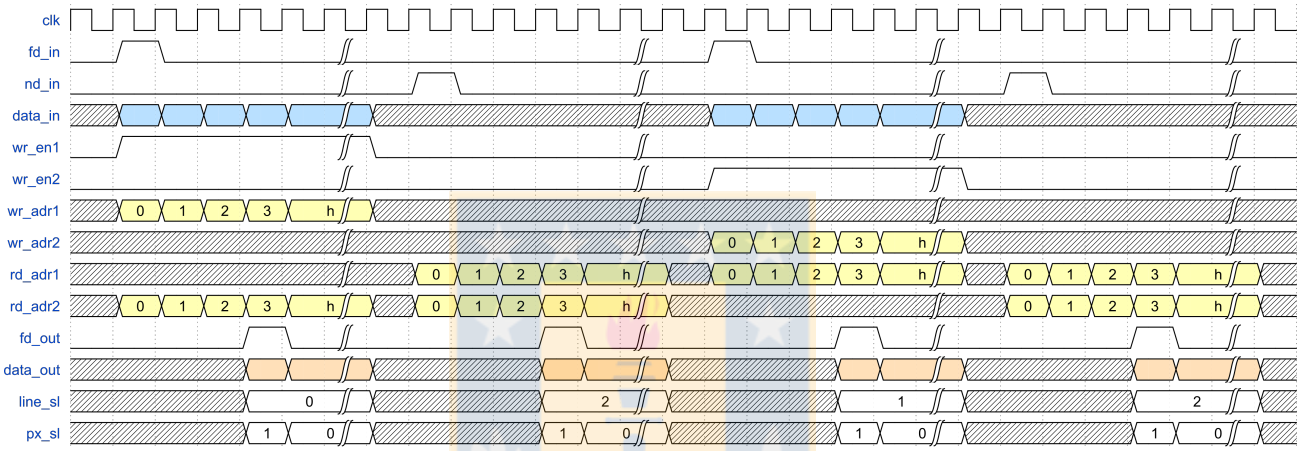


Fig. 5.13: Señales de control para módulo interpolación $\mathcal{D}^{\mathcal{T}}$.

simplificada en hardware, ya que no se requiere una reconstrucción de la imagen mediante procesos de interpolación complejos. En efecto, se requiere de una lectura adecuada de las imágenes almacenadas en los buffers de cuadros y_k, alojados en memoria RAM. Se trabajó sobre una máquina de estados que definía un offset de lectura desde RAM hacia el HwC. El offset de lectura está determinado por un registro temporal escrito desde el SwC, que indica la cantidad de pixeles horizontales y líneas verticales que deben ser omitidos para la lectura. Este registro es actualizado cuadro a cuadro, con el objetivo de que los cuadros estén alineados siempre al cuadro actual de análisis del algoritmo SR-NUC. El offset de lectura se debe enviar como señal a la interfaz entre HwC y RAM explicada en la sección 5.2.6. Este offset de lectura permite escoger el punto de partida para leer las líneas horizontales de la imagen.

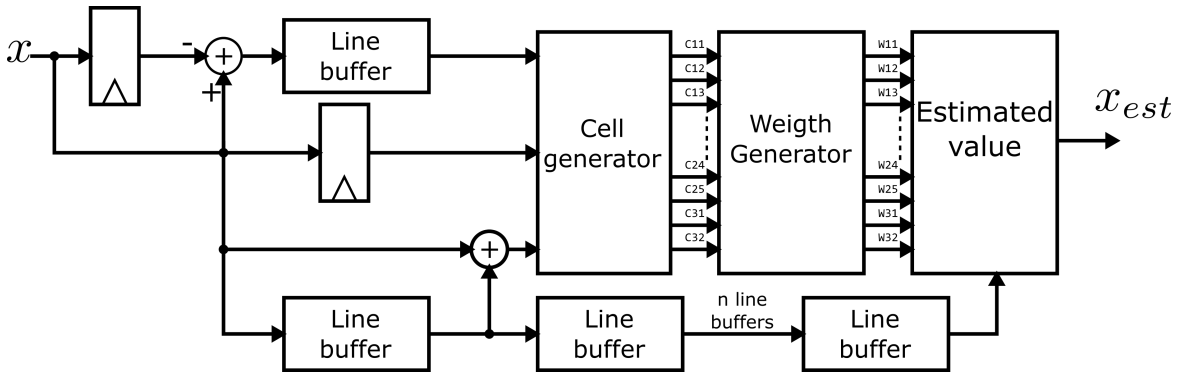


Fig. 5.14: Arquitectura hardware para módulo NLM.

5.2.4. Medias no locales

Para el diseño de la arquitectura en HwC del algoritmo de NLM se utilizó el análisis presentado en la sección 4.4. Como se muestra en la Figura 5.14, se divide el cálculo en 3 etapas: generador de celdas, generador de pesos y cálculo de valor estimado. Para el cálculo de los valores de la celda se necesita el valor de la diferencia entre el pixel actual y el anterior de la misma línea, el valor actual del pixel y los valores de la línea anterior. Esto se logra utilizando sumadores, registros temporales de almacenamiento y buffers de línea basados en block RAM (BRAM).

El módulo *Cell generator* calcula las distancias euclidianas entre las ventanas de comparación y la central de la Ecuación (4.9) (argumento de exponencial). Las distancias euclidianas se calculan aprovechando la característica de redundancia del cálculo de diferencias y la linealidad de éstas. Para lo anterior se almacenan tanto las diferencias verticales como las horizontales en múltiples buffers de línea para cada una de las distancias dentro de la vecindad ν, y se construye el mapa mediante la selección adecuada. Una vez que se calculan las distancias, éstas pasan a el generador de pesos *Weight Generator*, que estima la matemática restante de la Ecuación (4.9). Para la función exponencial se trabajó con una LUT de 2048 elementos de 16 bits, aproximación suficiente para el rango de datos en los que se encuentran las distancias euclidianas, bastando con inicializar correctamente una BRAM a datos predefinidos de exponencial. Luego de calcular la exponencial se utilizó un estimador de división basado en Newton Rhapson de 4 etapas en pipeline para normalizar los pesos. Una vez obtenidos los pesos, en la sección *Estimated value* se aplica la ventana de filtro NLM para obtener el valor estimado final.

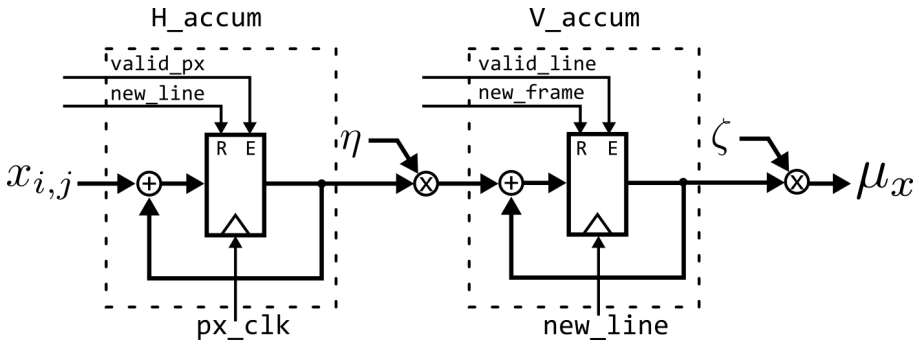


Fig. 5.15: Arquitectura hardware para cálculo de la media.

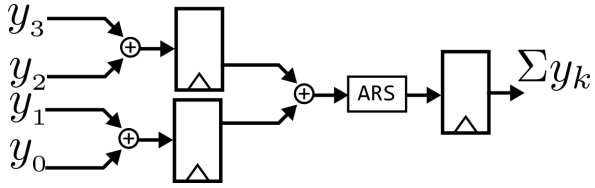
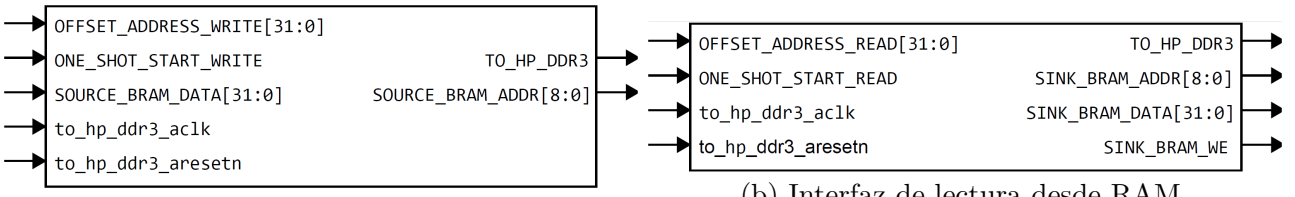


Fig. 5.16: Arquitectura hardware para promedio de cuatro señales.

5.2.5. Módulos aritméticos

Dos módulos aritméticos fueron diseñados para incluirlos en el diseño hardware del algoritmo SR-NUC mostrado en la Figura 5.9. Uno de ellos es la estimación de la media de un cuadro de imagen, representado por μ_x. El diseño hardware se muestra en la Figura 5.15, la que se basa principalmente en dos acumuladores, uno horizontal (*H_accum*) y otro vertical (*V_accum*). El primero acumula la suma de los pixeles por cada línea, para esto se realimenta un registro que acumula mientras los pixeles sean válidos (*valid_px*) a un reloj de pixel (*pc_clk*), con un reset que lleva a cero el registro cada vez que comienza una nueva línea (*new_line*). El valor de salida del acumulador horizontal es multiplicado por el factor η, que equivale al inverso del tamaño de la línea, es decir, se envía el promedio de cada línea a la siguiente etapa de acumulación vertical. La acumulación vertical *(H_accum*) es el mismo principio que la acumulación horizontal, pero en este caso la señal de habilitación es *valid_line*, la de reset es *new_frame* y la de reloj de escritura es *new_line*. Finalmente la acumulación vertical se multiplica por el factor ζ, equivalente al inverso del tamaño vertical de la imagen, que equivale al promedio de la imagen.

El promedio de señales presentado en la Figura 5.16 se basa en dos etapas básicas de suma. Se utilizan registros intermedios para sumar las señales en pares, ya que sumar más señales aumenta la carga en un mismo ciclo de reloj. Luego se suma la suma y se ejecuta un ARS, que es equivalente a dividir por cuatro, para calcular el promedio resultante.



(a) Interfaz de escritura a RAM.

(b) Interfaz de lectura desde RAM.

Fig. 5.17: Diagrama de módulo interfaz entre BRAM y RAM.

5.2.6. Interfaz BRAM-DDR

Para la comunicación de datos entre la memoria RAM que utiliza el SwC y las memorias BRAM a la que tiene acceso el HwC se utilizó un módulo diseñado por alumnos del laboratorio VLSI del DIE de la UdeC. Cada una de las memorias BRAM que requiera leer o escribir desde o hacia la memoria RAM debe utilizar este módulo. Éste usa señales de buses de inter-conexión de alto performance denominados HP AXI Interconnect, las señales tienen el nombre de "aclk" y "aresetn", estas conexiones se crean automáticamente con la herramienta de diseño que se utiliza para programar la lógica en el HwC. Para este tipo de módulos se necesita establecer la dirección base en memoria RAM y la cantidad de datos a transferir como parámetro interno de configuración. Dado a que existen dos tipos de operaciones con las memorias RAM, escritura y lectura, se separa el módulo en 2 sub-módulos de transferencia de datos (Figura 5.17).

Para la escritura (Figura 5.17a) es necesario establecer un offset de lectura respecto a la dirección base de memoria RAM en la entrada "OFFSET_ADDRESS_WRITE" que puede ser de hasta 32 bits de ancho. Los datos que provienen desde la memoria BRAM para escribir a memoria RAM deben ser enviados por la entrada "SOURCE_BRAM_DATA" de tamaño 32 bits, ya que es el tamaño de palabra de la memoria RAM. Para iniciar el proceso de escritura hacia memoria RAM se le envía una señal de un ciclo de duración en "ONE_SHOT_START_WRITE", con esta señal el módulo escribirá en memoria RAM tantos datos como el parámetro de cantidad de datos a transferir lo indique a la vez que gestiona la dirección de lectura de salida de la BRAM conectada a la interfaz.

Para la lectura (Figura 5.17b) es necesario establecer un offset de lectura respecto a la dirección base de memoria RAM en la entrada "OFFSET_ADDRESS_READ" que puede ser de hasta 32 bits de ancho. Los datos que provienen desde la memoria RAM para escribir a memoria BRAM deben ser capturados desde la salida "SINK_BRAM_DATA" de tamaño 32 bits, ya que es el tamaño de palabra de la memoria RAM. Para iniciar el proceso de lectura desde la memoriaRAM se le envía una señal de un ciclo de duración en "ONE_SHOT_START_READ", con esta señal el módulo escribirá en memoria BRAM tantos datos como el parámetro de

```
// Definición de direcciones
#define IM_FROM_CL        0x10000000
#define IM_HR_FROM_HW     0x10200000
#define IM_HR_TO_CL_OUT   0x10300000
//Punteros a variables en RAM DDR dedicada a hardware.
uint16_t        *Frame_Cl_16b       = (int*)IM_FROM_CL;
uint32_t        *Frame_From_HW      = (int*)IM_HR_FROM_HW;
uint16_t        *Frame_HR_To_HW_NM  = (int*)IM_HR_TO_CL_OUT;
```

Fig. 5.18: Localización de datos en memoria RAM para procesamiento en SwC.

cantidad de datos a transferir lo indique a la vez que gestiona la dirección de escritura de entrada y la habilitación de escritura con "SINK_BRAM_WE", de la BRAM conectada a la interfaz.

5.3. Núcleo software

En el núcleo software se llevan a cabo tareas de los dos algoritmos: SR-NUC y SAD. Para el caso de SR-NUC se calcula la diferencia de movimiento entre los cuadros de entrada desde el streaming de video del microscopio infrarrojo. En el caso de la estabilización se calcula SAD, las proyecciones, se filtran y compensan los movimientos. Como ambas etapas de los dos algoritmos tienen en común las proyecciones y el SAD, se dividen las secciones por operación y no por algoritmo.

5.3.1. Consideraciones

Al ser SwC que se trabaja en lenguaje de programación C, se presentarán extractos de código. Adicionalmente se considera de antemano que todos los datos se trabajan sobre memoria RAM, y que aquellos provenientes desde la zona de intercambio mostrada en la Figura 5.1 son válidos al momento de la lectura.

Para el inicio del análisis de estimación de movimiento de SR-NUC se espera una interrupción desde el HwC que indica que el cuadro actual ha sido escrito completo en memoria RAM, esta interrupción es enviada por el controlador en el HwC que recibe y decodifica los datos desde cameralink hacia memoria RAM. En el caso de la estabilización, se esperará una interrupción desde el HwC que indica que la imagen HR ha sido estimada y escrita por completo en memoria RAM, esta interrupción es enviada por el modulo *final value* de la arquitectura del algoritmo

```
1. for( j=0 ; j<j_high ; j++ ){
2.   for( xi=0 ; i<i_width ; i++ ){
3.      if( i == 0 )
4.         i_projection[ j ] = image_data[ j*i_width ];
5.      else
6.         i_projection[ j ] = image_data[ i+j*i_width ]+i_projection[ j ];
7.      if( j == 0 )
8.         j_projection[ i ] = image_data[ i ];
9.      else
10.        j_projection[ i ] = image_data[ i+j*i_width ] + j_projection[ i ];}}
```

Fig. 5.19: Cálculo de proyecciones en ejes i e j de una imagen.

SR-NUC.

5.3.2. Proyecciones integrales

Las proyecciones integrales de las imágenes se utilizaron para reducir la dimensionalidad y alivianar la carga computacional del cálculo de SAD. Como las imágenes completas están disponibles en RAM se puede trabajar directamente sobre el vector de imagen desde el SwC. En la Figura 5.19 se presenta un extracto de código de la función que calcula la proyección. La función recorre la imagen con dos ciclos **for**, que recorren las lineas horizontales comenzando desde la primera en orden vertical (ver líneas 1 y 2 del extracto de código). El tamaño de la imagen es representado por los parámetros *y_high* y *x_width*.

La proyección en i acumula los datos de dirección horizontal, en un vector de tamaño vertical. En cada uno de los pixeles iniciales *i=0* de las líneas tal valor es asignado como el inicial de la acumulación de esa linea (ver líneas 3 y 4 del extracto de código). Luego se acumula cada valor siguiente de la misma línea (ver línea 6).

La proyección en j acumula los datos de dirección vertical, en un vector de tamaño horizontal. En la línea inicial *j=0* los valores del vector se asignan equivalentes a esa línea (ver líneas 7 y 8 del extracto de código). Luego se acumulan los valores correspondientes en cada coordenada horizontal en el vector de acumulación (ver línea 10).

Una vez finalizada las iteraciones con los dos ciclos **for**, se tiene el valor de las proyecciones en ambos ejes de la imagen actual. Para el siguiente paso, suma absoluta de diferencias, se debe contar con las proyecciones del cuadro actual y del anterior. Se hace un respaldo temporal de los vectores del cuadro actual, que serán considerados como los pasados.

```
1. for ( i=0 ; i<V_RES ; i++)
2.   i_pas_proj[i] = i_new_proj[i];
3. for ( j=0 ; j<H_RES ; j++)
4.   j_pas_proj[j] = j_new_proj[j];
```

Fig. 5.20: Respaldo de proyecciones en ejes i e j de cuadro anterior.

```
1.  for ( i=0 ; i<max_delta*2+1 ; i++ ){
2.     SAD = 0;
3.     for( j=0 ; j<len-(max_delta*2+1) ; j++ ){
4.        if( pas_projection[ j+i ] > new_projection[ max_delta+j ] )
5.           abs_tmp = pas_projection[ j+i ] - new_projection[ max_delta+j ];
6.        else
7.           abs_tmp = new_projection[ max_delta+j ] - pas_projection[ j+i ];
8.        SAD = SAD + abs_tmp;}
9.     if(SAD<min_SAD){
10.       min_SAD = SAD;
11.       motion  = i-max_delta;}}
```

Fig. 5.21: Cálculo de estimación de movimiento en ejes i e j de una imagen.

5.3.3. Suma absoluta de diferencias

Para la suma absoluta de diferencias se utilizó la proyección integral. La proyección integral es un vector de una dimensión, pero se desplaza el vector de referencia en relación al objetivo, para comparar el desplazamiento que existe en cada eje. En la Figura 5.21 el vector de referencia es la proyección del cuadro pasado *pas_projection*, y se desplaza desde 0 a *max_delta·2+1* (línea 3), mientras que el vector objetivo es el cuadro nuevo o actual, que se desplaza y mantiene fijo en un valor de *max_delta* (líneas 5 u 7). De esta forma ambos vectores son comparados a través de la suma absoluta (línea 4 y 6) de la diferencia que exista entre los elementos correspondientes al añadir el desplazamiento al vector de referencia en la búsqueda.

En cada desplazamiento de vector, el valor *SAD* obtenido es comparado con el valor mínimo *SAD* hasta el momento (línea 9), de ser menor al mínimo se reemplaza y se define tal desplazamiento como el que define la transformación entre ambos cuadros (líneas 10 y 11). Se utilizan dos ciclos **for** anidados, en donde el padre recorre los distintos desplazamientos (línea 1) y se setea la *SAD* igual a cero, ya que es el acumulador. El **for** hijo recorre los vectores, considerando un recorte igual al desplazamiento máximo posible, es decir, la longitud del vector menos dos veces el desplazamiento máximo a considerar (línea 3).

```
1.  MV_i = MV_i·alpha + motion_i·(1-alpha);
2.  MV_j = MV_j·alpha + motion_j·(1-alpha);
3.  UM_i = motion_i-MV_i;
4.  UM_j = motion_j-MV_j;
```

Fig. 5.22: Filtro de movimiento y estimación de movimiento involuntario.

5.3.4. Filtrado y compensación de movimiento

El filtro de movimiento se aplica directamente sobre el SwC, con operaciones independientes para cada eje. El pseudo-código se muestra en la Figura 5.22, y se ejecuta en cada cuadro nuevo de video que proviene desde el microscopio. En las líneas 1 y 2 se filtra el movimiento voluntario MV desde el movimiento calculado en cada eje, con el factor de olvido *alpha*. Una vez que se tiene el movimiento voluntario estimado, se le resta al real, para obtener así el movimiento involuntario en las líneas 3 y 4.

Para la compensación de movimiento se utiliza acceso a memoria de forma desplazada y una segunda zona de memoria RAM para almacenar la imagen desplazada. Dado a que son traslaciones enteras, se transforma la imagen mediante offsets del índice de los elementos de la imagen original. Para aplicar de forma entera se transforma a entero el vector de movimiento involuntario obtenido en la etapa de filtro y estimación.

Capítulo 6: Resultados de la implementación

En este capítulo se presentan los resultados de la implementación de la arquitectura hardware diseñada en el Capítulo 5 sobre la tarjeta de desarrollo ZedBoard de Xilinx. La utilización de recursos lógicos, consumo potencia, velocidad máxima y caminos críticos fueron obtenidos mediante la herramienta de síntesis Vivado 2016.1 de Xilinx, utilizando la descripción Verilog del sistema para la implementación. Debido a que la herramienta de síntesis y diseño posee modos optimizados para la implementación, los resultados de utilización de recursos lógicos pueden no ser del todo exactos respecto a lo diseñado. Para la validación de resultados se utilizaron dos sistemas de prueba. El primero consta de pre-cargar imágenes del microscopio infrarrojo al SoC mediante archivos de extensión .h generados por MatLab, que incluyen las matrices de distintos cuadros de video para distintos casos de aplicación probados en software. Luego se enviaron los resultados de las matrices obtenidas en el SoC por el puerto serial hacia un computador de escritorio, corroborando que los resultados obtenidos fueron los esperados. El segundo modo de prueba fue mediante la conexión de una núcleo de cámara infrarroja de modelo FLIR TAU 2 de 640x512 pixeles de resolución espacial y 14 bits de datos, con protocolo camera-link, con el objetivo de probar que la implementación no interfiriese en el flujo original de trabajo, siendo sólo un puente invisible al protocolo camera-link. Para esto se configuró dentro del HwC la adquisición de datos para capturar a la misma resolución espacial del microscopio, reduciéndola desde los 640x512 a 320x240.

La tarjeta de desarrollo ZedBoard de Xilinx, mostrada en la Figura 6.1, que incluye un SoC XC7Z020-1CLG484. Las características principales de la plataforma son:

- Procesador ARM Cortex-A9 de dóble núcleo de hasta 667[MHz] de operación.
- 512[MB] de memoria DDR3 de 1050[Mbps] de ancho de banda.
- Puerto de expansión FMC.
- Puertos de salida de video HDMI y VGA.
- 8 interruptores, 8 luces led y 5 pulsadores.

La lógica programable incluida dentro del SoC es equivalente a un FPGA Artix-7 de la serie 7 con las siguientes especificaciones:

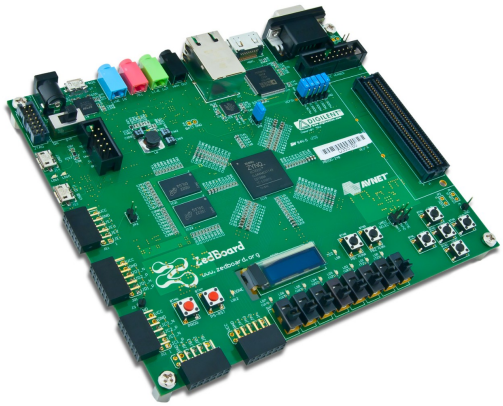


Fig. 6.1: Tarjeta de desarrollo ZedBoard de Xilinx (fuente [1]).

- 53.200 look up table (LUT).
- 106.400 flip-flops.
- 140 unidades de BRAM equivalentes a 560 KB.
- 220 slices DSP de 48 bits.

El SoC incluye además el procesador ARM Cortex-A9, comunicable con la lógica programable por medio de la memoria RAM y sistema de entrada/salida utilizando los buses AXI disponibles. La lógica programable se diseña y sintetiza con el IDE Vivado, software de desarrollo de Xilinx, describiendo la arquitectura en lenguaje verilog y generando un archivo descriptivo para programar el FPGA. Se exporta la descripción al IDE SDK, complementario de Vivado, en donde se configura y programa el procesador ARM en lenguaje C en conjunto con la lógica programable. El costo comercial de la tarjeta de desarrollo bordea los 475 USD, y el costo del SoC por unidad es cercano a los 125 USD.

6.1. Utilización de recursos

En la Tabla 6.1 se muestra el resultado de utilización de recursos por jerarquía del sistema, que es separada en:

- Axis COM: Interconexión del sistema que trabaja como puente de comunicación entre la memoria RAM DDR3 y la lógica programable. Incluye además todas las señales de interrupción y datos simples que se envían desde el SwC al HwC. Se usó un total de ocho puertos AXI: cinco para el almacenamiento de las imágenes LR, uno para la imagen

Tabla 6.1: Utilización de recursos en jerarquías principales.

	Slice LUTs	Slice Registers	Slices	Block RAM Tile	DSPs
Axis COM	7460	9118	3200	12,5	0
DDR-IFace	517	754	271	0	0
DRAW	1986	2628	966	4	24
SR-NUC	8699	8950	1305	23	36
Interfaz	588	711	313	6	4
Total	19553	22332	6151	46,5	64
Disponible	53200	106400	13300	140	220
Porcentaje	36.18 %	20.83 %	45.53 %	32.5 %	29.09 %

HR estimada, uno para la estimación del ruido b y uno para la imagen HR estimada con movimiento espacial estabilizado.

- DDR-IFace: Interfaz de comunicación que realiza la transferencia de datos entre HwC y RAM.
- DRAW: Jerarquía diseñada para pruebas de implementación. Incluye interfaces BRAM a DDR, módulos de dibujo en pantalla, mapas de colores y controladores de video. Principalmente se usó en despliegue de video por el puerto VGA de la plataforma.
- SR-NUC: Núcleo principal que incluye toda la lógica de la implementación en el HwC del algoritmo de SR-NUC.
- Interfaz: Control de flujo de datos de las imágenes LR y HR, que permite realizar el análisis de SR-NUC. Incluye las máquinas de estado que cuentan las líneas horizontales, número de cuadros, pixeles por línea, entre otros. Incluye además la captura y la generación de señales en el protocolo de camera-link.

La jerarquía del núcleo SR-NUC es la que utiliza la mayor cantidad de recursos lógicos, llegando a un 13.9 % de uso recursos en promedio del total disponible en la plataforma de trabajo. De forma particular utiliza el 16.35 % de slice LUTs, 8.41 % de slice registers, 9.81 % de slices, 16.43 % de unidades BRAM y un 16.36 % de unidades matemáticas DSP. El desglose por jerarquía modular del núcleo SR-NUC se describe en la Tabla 6.2, donde se juntaron los módulos por tipo. El módulo que más recursos utiliza es el de NLM, con una alta utilización de slice LUTs, unidades BRAM y DSP. Lo anterior se debe a la alta intensidad computacional requerida en multiplicaciones y memoria por el algoritmo de cálculo de NLM. Se agregaron líneas de delay de pixeles y buffers de línea para sincronizar las señales y realizar correctamente los cálculos.

Tabla 6.2: Utilización de recursos módulos dentro del núcleo SR-NUC.

Name	Slice LUTs	Slice Registers	Slice	Block RAM Tile	DSPs
Decimacion	47	79	26	1	0
Signal Difference	320	327	85	0	0
Gain	112	68	33	0	0
Interpolacion	1564	834	480	5	0
Line Buffer	83	133	51	2	0
Signal Mean	192	194	52	0	0
NLM	4502	24	7	15	36
Pixel Delay	1830	7229	547	0	0
Signal Adder	32	62	18	0	0
Otros	17	0	6	0	0
Total	8699	8950	1305	23	36
Disponible	53200	106400	13300	140	220
Porcentaje	16.35 %	8.41 %	9.81 %	16.43 %	16.36 %

Las lineas de delay de pixeles fueron implementadas sobre registros de desplazamiento, por eso su utilización es mayormente en Slice Registers. Los buffers de línea fueron implementados sobre BRAM, incluyendo además el control de direcciones, estos módulos utilizan 2 unidades de BRAM, divididas en 4 buffers de línea, donde cada uno ocupa menos de la mitad del bloque. Los retardos a pixeles y buffers de línea fueron necesarios ya que la información de las imágenes es llevada al HwC línea a línea desde la RAM. En las etapas de interpolación se usaron 5 unidades BRAM, que concuerda con la arquitectura diseñada, en donde cada unidad de interpolación utiliza dos buffers de línea menores a la mitad de una unidad BRAM, lo que traduce a una BRAM por interpolación y cinco interpolaciones en la arquitectura de SR-NUC presentada en la sección 5.2. El conjunto de módulos Gain son los multiplicadores de ganancia de los parámetros dentro de SR-NUC, correspondientes a una multiplicación por una fracción, cuyo denominador es una potencia de dos para ejecutar un desplazamiento y el numerador el factor correspondiente para que la fracción sea un valor cercano al parámetro.

La segunda jerarquía en cantidad recursos utilizados es la de Axis COM, que es indispensable dentro de la arquitectura, ya que se encarga del intercambio de datos entre SwC y HwC. La principal razón de la utilización de recursos es que se cuenta sólo con 4 buses de alto performance (HP), de los cuales uno se utiliza en DRAW, los otro tres se dividen entre los ocho puertos AXI. Al tener ocho puertos y 3 buses, se deben distribuir de tal manera que la transferencia sea eficiente. Por lo anterior, se configuraron los buses en el perfil de alto performance y no de baja

Tabla 6.3: Consumo de potencia dinámica de implementación en SoC por recurso.

	Power [mW]	Power [%]
SwC	1.546	77.42
I/O	42	2.1
MMCM	122	6.11
DSP	27	1.35
BRAM	66	3.3
Logic	60	3
Signals	73	3.66
Clocks	61	3.05
Total	1997	

utilización.

La jerarquía DRAW no es indispensable dentro del sistema, puesto que sólo fue agregada como parte de las pruebas de implementación. Pese a esto forma parte de cerca del 10 % de los recursos utilizados, lo que no es menor. Si es que fuese necesario disminuir la cantidad de recursos utilizados, esta jerarquía puede eliminarse sin verse afectada la funcionalidad de la implementación.

En el caso de la estimación, filtro y estabilización movimiento en la escena, no es posible con las herramientas disponibles estimar el impacto de potencia sobre el SoC, ya que se encuentra incluido dentro del SwC y la herramienta de análisis no incluye desglose sobre éste.

6.2. Sistema implementado

Los resultados de consumo de potencia fueron estimados con la herramienta XPower Analyzer de Xilinx, implementando la arquitectura a una velocidad de reloj de 160[MHz] en el HwC, 600[MHz] en el SwC, el despliegue de video a 65[MHz] en VGA a una resolución de 1024x768 y 20[MHz] de reloj de pixel por camera-link.

La Tabla 6.3 muestra el consumo de potencia del SoC por recurso y la Tabla 6.4 muestra el consumo de potencia por módulo jerárquico de la arquitectura implementada. Se observa que el mayor consumo de potencia se debe al SwC, que corresponde al procesador ARM embebido dentro del SoC, que es el consumo típico de esta unidad, disipando cerca del 78 % del total del sistema. El segundo recurso que más utiliza potencia es el Mixed-Mode Clock Manager

Tabla 6.4: Consumo de potencia de implementación en SoC por módulo jerárquico.

	Power [mW]	Power [%]
Axis COM	81.1	4.15
DDR-IFace	6	0.31
DRAW	204	10.44
SR-NUC	93	4.76
Interfaz	22	1.13
SwC	1546	79.22
Total	1952.1	
Total HwC	406.1	

(MMCM), que corresponde a los módulos generadores de relojes dentro del SoC de Xilinx, necesarios para los distintos módulos que requieren relojes de alto performance dentro de la arquitectura. En el desglose por jerarquía se puede observar la cantidad de consumo del núcleo dedicado a SR-NUC, que alcanza 93[mW], el de la interfaz de comunicación que es de 22[mW], las interfaces DDR-BRAM de 6[mW] y la comunicacion AXI de 81.1 [mW]. El consumo de el HwC por si solo es de 406.1 [mW] y del SwC de 1546 [mW].

Dado a que el mayor consumo de potencia está dado por el SwC, si fuese necesario disminuir el consumo de potencia es posible trasladar la parte de arquitectura diseñada sobre este núcleo al núcleo hardware de diseño lógico. Esta posibilidad no incurriría en una complejidad elevada, ya que el algoritmo de estimación y filtro de movimiento ejecutado en el SwC se basa en: proyecciones de vectores mediante acumulación de valores en un bloque de memoria, comparación iterativa de variables escalares, multiplicaciones por parámetros definidos; todos elementos propios del HwC. Adicionalmente, el módulo jerárquico DRAW, al no ser indispensable, puede igualmente ser eliminado para disminuir de 406.1 [mW] a 202.1[mW] de consumo de potencia en el HwC. Dejando de lado el módulo DRAW, se observa que el mayor consumo energético se divide entre el módulo de Axis COM y el SR-NUC, que son los que ejecutan toda la matemática principal y la transferencia de datos.

La velocidad máxima a la que puede funcionar el HwC es de 160 [MHz], limitado principalmente por la velocidad de transmisión de datos entre la memoria DDR y las BRAM en el HwC. El circuito digital es capaz de corregir 2.26 cuadros de 320x240 pixeles por segundo, con 30 iteraciones SR-NUC y estabilización de escena. En contraste a una implementación en MatLab capaz de corregir 1 cuadro cada 5 minutos aproximadamente bajo las mismas condiciones de funcionamiento, se consigue una aceleración de cercana a 600 veces. Las pruebas en softawre

Tabla 6.5: Resultados comparativos de PSRN sobre tres casos

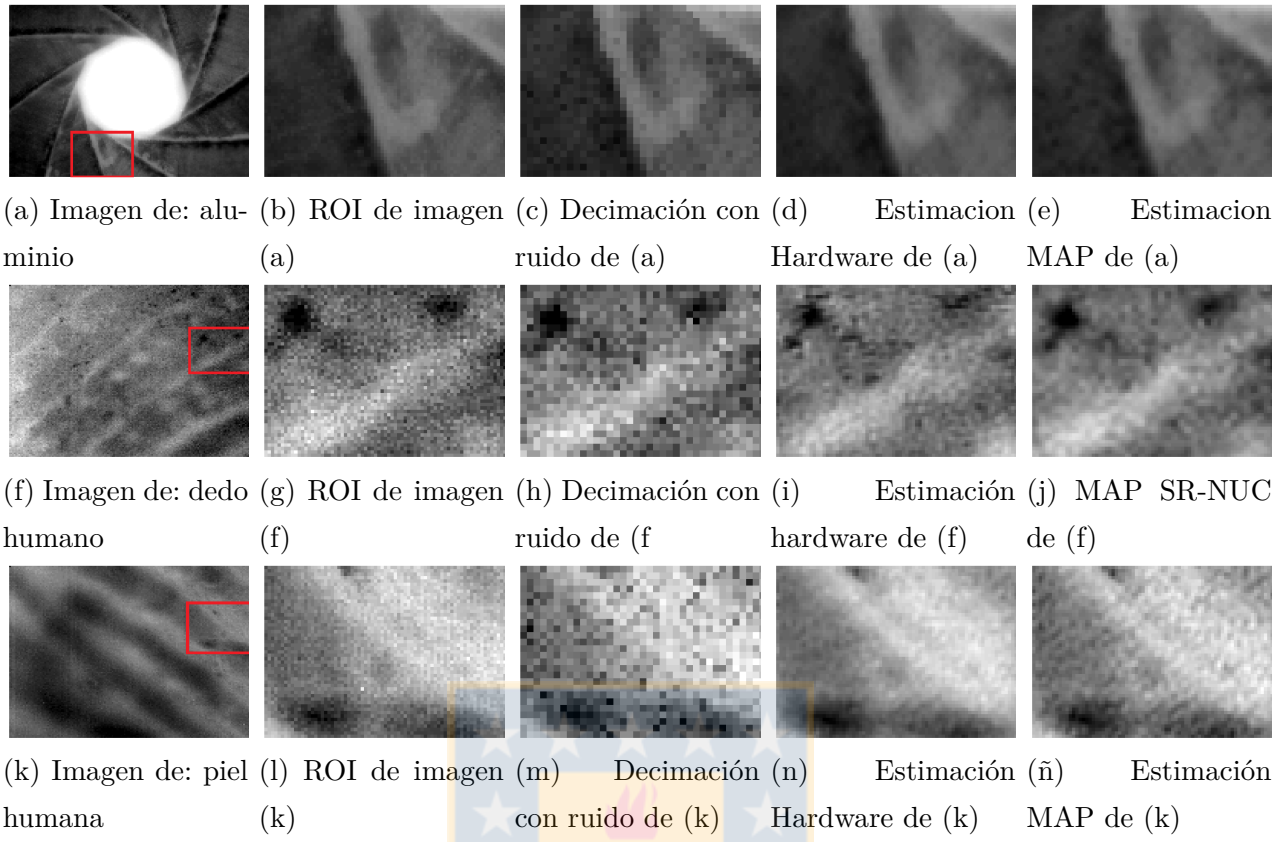
Caso	HW SR-NUC	SW SR-NUC	MAP SR-NUC
1	65.56dB	65.59dB	67.98dB
2	57.83dB	57.95dB	57.60dB
3	65.51dB	65.52dB	64.90dB

MatLab fueron realizadas sobre un computador de escritorio con un procesador Intel Core I5 de 4 núcleos de 3,4[GHz] y 16GB de RAM DDR3 en un sistema operativo windows10 de 64bits, que requiere una fuente de poder sobre los 200[W] para funcionar, es decir, casi 100 veces en comparación a los requerimientos de potencia en hardware.

Pese a lo anterior, la velocidad de implementación no es suficiente para corregir en tiempo real video de 320x240 a una tasa de 60 cuadros por segundo, quedando a una velocidad 30 veces más lenta. El límite viene dado principalmente por el ancho de banda de la memoria DDR, que es de 1050[Mbps], que se traducen en 134,400 [KBs]. Si se considera que cada cuadro es de 320x240 datos de 14 bits, entonces para traer un solo cuadro son necesarias 160 transacciones de 1 [KB]. Cada iteración de SR-NUC lee 4 cuadros, las cuatro muestras, y escribe 5 por iteración, una imagen HR de 4 veces el tamaño y el ruido de no uniformidad b, es decir 1350 [KB] por iteración. Si se consideran 30 iteraciones, entonces para cada imagen HR estimada es necesario transferir 40,500 [KB], es decir, en un caso ideal en donde no exista latencia ni retardos asociados, y la memoria DDR sólo sea usada por el HwC, es posible alcanzar un máximo de $134,400/40,500$ =3.31 cuadros de alta resolución por segundo mediante SR-NUC. En la práctica, este número se ve reducido considerablemente de 3.31 a 2.26 cuadros por segundo, ya que adicionalmente el SwC utiliza igualmente la memoria DDR y el bus tiene retardos asociado a las múltiples peticiones de datos.

En la Figura 6.2 se muestra el resultado de aplicar SR-NUC en implementación hardware sobre tres casos: placa de aluminio (caso 1), dedo humano (caso 2) y piel humana (caso 3). Los resultados de PSNR mostrados en la Tabla 6.5 son comparaciones entre el diseño hardware SR-NUC, implementación software de SR-NUC equivalente en punto flotante con una implementación software del algoritmo MAP SR-NUC visto en el estado del arte. Los valores numéricos demuestran un buen desempeño numérico de la implementación hardware, tanto con su análoga en software como con el algoritmo MAP SR-NUC.

Los resultados finales mostrados en la Figura 6.3 fueron desplegados a través de una aplicación de National Instruments incluidos en la interfaz cameralink framme grabber a PCI-express mostrado en la Figura 6.3d, utilizada para capturar imágenes en tiempo real desde una conexión



(a) Imagen de: aluminio | (b) ROI de imagen (a) | (c) Decimación con ruido de (a) | (d) Estimacion Hardware de (a) | (e) Estimacion MAP de (a)

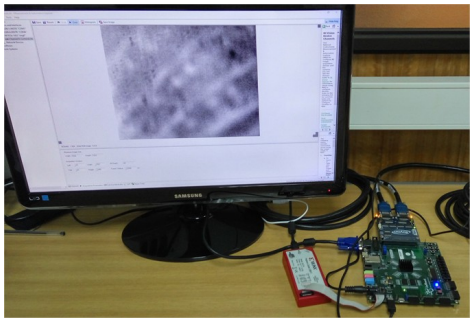
(f) Imagen de: dedo humano | (g) ROI de imagen (f) | (h) Decimación con ruido de (f | (i) Estimación hardware de (f) | (j) MAP SR-NUC de (f)

(k) Imagen de: piel humana | (l) ROI de imagen (k) | (m) Decimación con ruido de (k) | (n) Estimación Hardware de (k) | (ñ) Estimación MAP de (k)

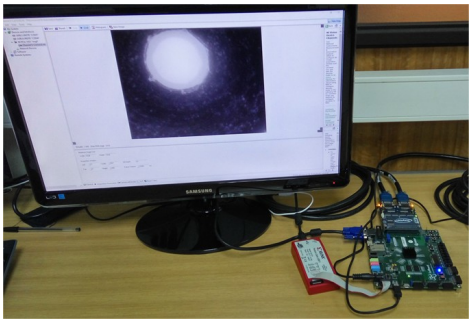
Fig. 6.2: Resultados y comparación sobre distintas imágenes en implementación hardware

camera link. Los resultados mostrados son obtenidos mediante el primer sistema de pruebas, en donde se pro-cargaron la información de video del microscopio infrarrojo. En la Figura 6.3a se muestra el resultado sobre el dorso de una mano, en la Figura 6.3b se muestra el resultado sobre una configuración con una fibra óptica y en la Figura 6.3c se muestra el resultado sobre poliestireno expandido. En las tres figuras se puede observar además el uso de la interfaz de comunicación cameralink-FMC para el envío de imágenes desde el SoC hacia el framme grabber.

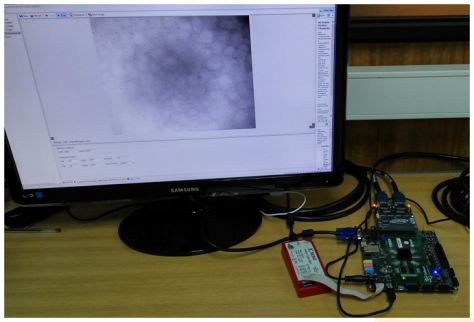
En la Figura 6.4a se muestra el circuito funcionando como puente entre la cámara infrarroja TAU 2 y el framme grabber. Esta plataforma de pruebas captura las imágenes y trabaja sobre los cuadros guardados en los buffers temporales que se actualizan cuadro a cuadro. Dado a que sólo se pueden corregir cerca de 2 cuadros por segundo y la cámara trabaja a 30 cuadros por segundo, el algoritmo itera sobre los cuadros actualizados en tiempo real, es decir, utiliza la información de 15 cuadros distintos para la corrección. En la Figura 6.4b se muestra el despliegue de video adicional para poder comparar visualmente la imagen estimada en HR con estabilización espacial y el cuadro original de imagen.



(a) Resultado sobre dorso de una mano.



(b) Resultado sobre fibra óptica.



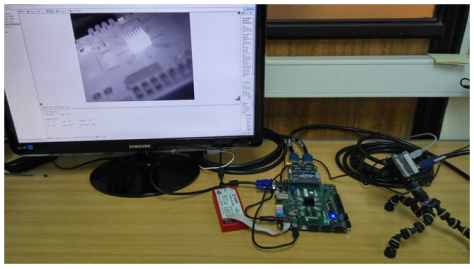
(c) Resultado sobre trozo de poliestireno expandido.

(d) Framme grabber PCI-express en computador de trabajo.

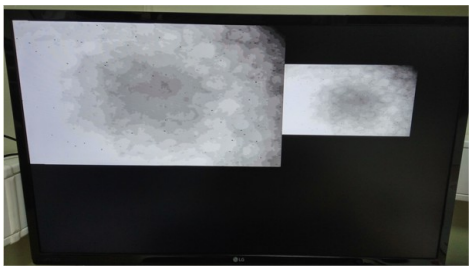
Fig. 6.3: Resultados de la implementación y plataforma de pruebas.

6.3. Discusión

Para permitir una solución que resuelva los problemas de no uniformidad, baja resolución y estabilización de escena, en un sistema el línea y tiempo real se deben cumplir con la capacidad de memoria y el ancho de banda suficiente dentro del circuito digital, directamente relacionadas entre sí. La capacidad de memoria debe ser suficiente para abarcar al menos todos los cuadros de imágenes necesarios para la aplicación de SR-NUC por cada cuadro de video. Se debe almacenar



(a) Prueba de conexión camera-link en cámara infrarroja TAU 2 en línea.



(b) Prueba de despliegue de videos en VGA, a la izquierda imagen HR y a la derecha imagen original LR.

Fig. 6.4: Pruebas adicionales para comprobar la implementación sobre la tarjeta de desarrollo.

cinco cuadros de video en LR y un cuadro del ruido de patrón fijo b de 320x240 pixeles de 14 bits, es decir, 750 KB de memoria considerando cada pixel como una palabra de 16 bits y una memoria de ancho de 32 bits. Adicionalmente se necesita almacenar la información de la estimación de imagen en HR, de tamaño 640x480 pixeles de 14 bits, que significan 600 KB de memoria con las mismas consideraciones. Es decir, un total de al menos 1350 KB de memoria dentro del SoC son necesarias.

El ancho de banda debe ser capaz de procesar cuatro de esos cuadros de video LR, el cuadro del FPN y el cuadro HR, por cada iteración del algoritmo SR-NUC. El número de iteraciones necesarias son 30 por cuadro y la tasa de refresco típica es de 60 cuadros por segundo, significando así 1800 lecturas de todos los cuadros por segundo. Con una implementación de memoria en bloques BRAM para cada cuadro, la lectura puede ser realizada en paralelo entre los distintos cuadros almacenados en BRAM. La cantidad de datos leídos por segundo es determinada por la imagen estimada en HR, la de mayor tamaño, que comprendería 600[KB/cuadro]×1800[cuadro/segundo]=1055[MB/s]. Las plataformas actuales permiten leer 32 bits (o 4 bytes) por ciclo de reloj, es decir, para obtener 1055[MB/s] se necesita una frecuencia de reloj igual a 1055[MB/s]/4[B]=264[MHz].

En resumen es necesaria una plataforma hardware con memoria sobre los 1350 [KB] en BRAM con velocidad de 264[MHz] de reloj de lectura mínima para lograr procesar 60 cuadros por segundo con 30 iteraciones por cuadro. En caso de querer aumentar la capacidad de procesamiento de cuadros se requiere aumentar la velocidad de reloj de lectura, por ejemplo si se deseara procesar 120 cuadros por segundo se debe aumentar a 528 [MHz] la velocidad de reloj de lectura de las BRAM.

Una plataforma actual que cumple con los requisitos requeridos es la tarjeta de desarrollo Nexys Video de Xilinx, que incluye un FPGA XC7A200. Este FPGA incluye 1664[KB] de memoria BRAM con frecuencias de trabajo por sobre los 500[MHz]. Teóricamente se pueden lograr una velocidad de procesamiento de hasta 113 cuadros por segundo, con 30 iteraciones por cuadro, capacidad muy superior a lo alcanzado en la plataforma utilizada en este trabajo. En la Figura 6.5 se muestra una propuesta de arquitectura digital sobre un FPGA en lógica programable. A diferencia de la arquitectura de la Figura 5.5 para el SoC, la arquitectura propuesta presenta:

- Datos de imágenes se localizan completamente sobre BRAM.
- El cálculo de movimiento entre cuadros se ejecuta sobre un módulo lógico en hawrdare.

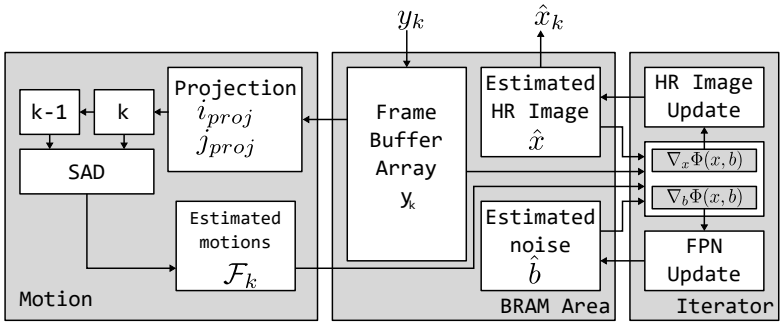


Fig. 6.5: Arquitectura general propuesta para SR-NUC sobre FPGA Nexys Video.

- Módulo iterador calcula el descenso de gradiente.
- El movimiento estimado pasa directamente al iterador.
- El resto de la arquitectura se mantiene sin alteraciones.

Capítulo 7: Conclusiones

La implementación del circuito digital sobre un SoC presentado en este trabajo permite corregir no uniformidad y aplicar super resolución simultáneos, y adicionalmente estabilizar los movimientos involuntarios de cuadros de video provenientes de un microscopio infrarrojo, aumentando la velocidad de cómputo presentada en el trabajo original desde donde se basa la arquitectura diseñada. El circuito es capaz de compensar los fenómenos de no uniformidad, baja resolución y inestabilidad espacial en un cómputo en línea, considerando los protocolos de comunicación de entrada y salida del sistema para ser transparente a una aplicación final. Los resultados son aplicables a imágenes naturales capturadas por el microscopio infrarrojo de estudio, en particular para piel humana o escenas con dinámica lenta en relación al tiempo de captura. La arquitectura diseñada es general y permite cambios en parámetros, y para otro tipo de aplicaciones se deben replicar los análisis de los algoritmos para su configuración adecuada.

La separación de los distintos videos en tres casos determinados permitió que el análisis cualitativo se simplificara y los resultados fueran extendidos dentro de los límites establecidos, definidos por los casos favorables y desfavorables. El desenfoque de la escena impide un análisis correcto de las imágenes para los algoritmos implementados, ya que se decidió no incluir el operador PSF que compensa el desenfoque. Pese a esto, el algoritmo SR-NUC tuvo un buen desempeño en los casos favorables y desfavorables, que son los casos ideales y estándar de trabajo con el microscopio infrarrojo.

La estabilización de escena permitió satisfactoriamente los movimientos indeseados en las escenas, sobretodo en los casos desfavorables, en donde el pulso humano introduce vibraciones indeseadas. El filtro otorga la posibilidad de que existan movimientos voluntarios en la escena, algo natural en sistemas de captura, en caso contrario el movimiento acumulado llegaría a valores inalcanzables para la etapa de reconstrucción.

Pese a lo anterior, los resultados obtenidos en la plataforma de trabajo utilizada, no permiten una corrección de estos problemas en tiempo real, cuya limitación se impone principalmente por el ancho de banda de la memoria de acceso aleatorio DDR3 utilizada para el almacenamiento temporal de los cuadros de video. Sin embargo, la arquitectura ha sido validada en la implementación y es posible utilizarla como base para una aplicación en tiempo real, cumpliendo con condiciones suficientes de almacenamiento de datos en bloques de memoria hardware dedicada

y velocidades de reloj superiores a los 200[MHz].

La utilización de recursos lógicos dentro del SoC da posibilidad a integrar más sistemas de análisis de imágenes u otras tareas complementarias dentro del mismo. Los recursos energéticos se vieron afectados principalmente por el SwC, cuya potencia de trabajo mínima está determinada por el procesador ARM Cortex-A9 embebido. Aún así el circuito cumple con eficiencia las tareas en comparación a una implementación en software sobre un computador de arquitectura general, en una implementación compacta, con una potencia menor a 2 [W] y con mayor desempeño numérico.

El uso del SwC para tareas de bajo costo computacional tiene la ventaja de reducir la dificultad de diseño en lógica hardware y aprovechar unidades estándares. Una desventaja enfrentadas por la utilización del SwC fue la dificultad de sincronización de señales y la transmisión de datos entre HwC y SwC, que además sufren de tiempos de latencia no determinísticos. Otra desventaja es el gran consumo de energía por parte del SwC, cercano al 77 % del total, que es elevado en comparación al consumo del núcleo SR-NUC, más aún considerando que las tareas implementadas en éste son de bajo costo computacional.

Como trabajo futuro, para una implementación en línea y en tiempo real, se propone utilizar una plataforma de trabajo XC7A200T de Xilinx, incluida en la tarjeta de desarrollo Nexys Video, que cuenta con un puerto FMC. Este FPGA cuenta con 1664 [KB] de memoria BRAM de alta velocidad, capaz de funcionar desde 250[MHz] hasta 500[MHz], y un reloj interno de funcionamiento por sobre los 450[MHz]. Estas características se estiman suficientes, y la arquitectura digital diseñada en este trabajo puede ser utilizada como base para una aplicación en línea y tiempo real.

Bibliografía

[1] XilinxInc, "Xup zedboard," http://www.xilinx.com/support/university/boards-portfolio/xup-boards/XUPZedBoard.html, 2016.

[2] A. Rogalski, "Infrared detectors: status and trends," *Progress in Quantum Electronics*, vol. 27, pp. 59–210, 2003.

[3] Hamamatsu, "Technical information: Characteristics and use of infrared detectors," Tech.Rep. SD - 12, HAMATSU PHOTNICKS K.K., Solid State Division (March 2011).

[4] R. C. Hardie, K. J. Barnard, J. G. Bognar, E. E. Armstrong, and E. A. Watson, "High-resolution image reconstruction from a sequence of rotated and translated frames and its application to an infrared imaging system," *Optical Engineering*, vol. 37, no. 1, pp. 247–260, 1998. [Online]. Available: http://dx.doi.org/10.1117/1.601623

[5] X. Yang, W. Wu, K. Liu, K. Zhou, and B. Yan, "Fast multisensor infrared image super-resolution scheme with multiple regression models," *Journal of Systems Architecture*, pp.–, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1383762115001411

[6] W. Kong, P. Cao, X. Zhang, L. Cheng, T. Wang, L. Yang, and Q. Meng, "Near-infrared super resolution imaging with metallic nanoshell particle chain array," *Plasmonics*, vol. 8, no. 2, pp. 835–842, 2013. [Online]. Available: http://dx.doi.org/10.1007/s11468-013-9480-7

[7] L. M. Miller, "Infrared microsperctroscopy and imaging," Oct 2002.

[8] J. Paz, M. Perez, and I. Miranda, "Diagnostic quality of high resolution jpeg 2000 compressed ct and mr brain images," *Information and Communication in Medicine, Telemedicine and e-Health*, vol. 25/5, pp. 334–337, 2009.

[9] S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: a technical overview," *Signal Processing Magazine, IEEE*, vol. 20, no. 3, pp. 21–36, May 2003.

[10] D. A. Scribner, M. Kruer, and J. M. Killiany, "Infrared focal plane array technology," *Proceedings of the IEEE*, vol. 79, no. 1, pp. 66–85, Jan 1991.

[11] J. Farrell, F. Xiao, and S. Kavusi, "Resolution and light sensitivity tradeoff with pixel size," *Proc. SPIE*, vol. 6069, pp. 60 690N–60 690N–8, 2006. [Online]. Available: http://dx.doi.org/10.1117/12.646805

[12] S. Borman and R. Stevenson, "Super-resolution from image sequences-a review," in *Circuits and Systems, 1998. Proceedings. 1998 Midwest Symposium on*, Aug 1998, pp. 374–378.

[13] H. He and W.-C. Siu, "Single image super-resolution using gaussian process regression," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, June 2011, pp. 449–456.

[14] K. Zhang, X. Gao, D. Tao, and X. Li, "Single image super-resolution with non-local means and steering kernel regression," *Image Processing, IEEE Transactions on*, vol. 21, no. 11, pp. 4544–4556, Nov 2012.

[15] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *Computer Vision, 2009 IEEE 12th International Conference on*, Sept 2009, pp. 349–356.

[16] S. Ram and J. Rodriguez, "Single image super-resolution using dictionary-based local regression," in *Image Analysis and Interpretation (SSIAI), 2014 IEEE Southwest Symposium on*, April 2014, pp. 121–124.

[17] T. Yuan, W. Yang, F. Zhou, and Q. Liao, "Single image super-resolution via sparse kpca and regression," in *Image Processing (ICIP), 2014 IEEE International Conference on*, Oct 2014, pp. 2130–2134.

[18] H. Yu, Z.-j. Zhang, F.-s. Chen, and C.-s. Wang, "Non-local means-based nonuniformity correction for infrared focal-plane array detectors," vol. 9301, 2014, pp. 93 013K–93 013K–9. [Online]. Available: http://dx.doi.org/10.1117/12.2074379

[19] M. Alam, J. Bognar, R. Hardie, and B. Yasuda, "Infrared image registration and high-resolution reconstruction using multiple translationally shifted aliased video frames," *Instrumentation and Measurement, IEEE Transactions on*, vol. 49, no. 5, pp. 915–923, Oct 2000.

[20] D. Sorrentino and A. Antoniou, "Multiframe image super-resolution using quasi-newton algorithms," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, May 2008, pp. 264–267.

[21] Y. Chen, W. Jin, L. Wang, C. Liu, and W. Chen, "Robust multiframe super-resolution reconstruction based on regularization," in *Computer Symposium (ICS), 2010 International*, Dec 2010, pp. 408–413.

[22] S. Farsiu, M. Robinson, M. Elad, and P. Milanfar, "Fast and robust multiframe super resolution," *Image Processing, IEEE Transactions on*, vol. 13, no. 10, pp. 1327–1344, Oct 2004.

[23] S. Peleg, D. Keren, and L. Schweitzer, "Improving image resolution using subpixel motion," *Pattern Recognition Letters*, vol. 5, no. 3, pp. 223 – 226, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0167865587900675

[24] M. E. Angelopoulou, C.-S. Bouganis, P. Y. K. Cheung, and G. A. Constantinides, *Reconfigurable Computing: Architectures, Tools and Applications: 4th International Workshop, ARC 2008, London, UK, March 26-28, 2008. Proceedings.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, ch. FPGA-Based Real-Time Super-Resolution on an Adaptive Image Sensor, pp. 125–136. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-78610-8_14

[25] A. Goshtasby, *Image Registration: Principles, Tools and Methods*, ser. Advances in Computer Vision and Pattern Recognition. Springer London, 2012. [Online]. Available: https://books.google.cl/books?id=VthiwQOT6t4C

[26] B. A. White, "Using fpgas to perform embedded image registration," Master's thesis, College of Engineering and Computer Science and in The Burnett Honors College at the University of Central Florida, 2009.

[27] P. Vandewalle, S. Süsstrunk, and M. Vetterli, "A frequency domain approach to registration of aliased images with application to super-resolution," *EURASIP J. Appl. Signal Process.*, vol. 2006, pp. 233–233, Jan. 2006. [Online]. Available: http://dx.doi.org/10.1155/ASP/2006/71459

[28] T. Jerbi, V. Burdin, J. Leboucher, E. Stindel, and C. Roux, "2-d x2013;3-d frequency registration using a low-dose radiographic system for knee motion estimation," *Biomedical Engineering, IEEE Transactions on*, vol. 60, no. 3, pp. 813–820, March 2013.

[29] I. Dinov, M. Mega, P. Thompson, R. Woods, D. Sumners, E. Sowell, and A. Toga, "Quantitative comparison and analysis of brain image registration using frequency-adaptive wavelet shrinkage," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 6, no. 1, pp. 73–85, March 2002.

[30] D. S. Kim, K. Lee, K. E. Lee, and S. S. Han, "Joint optimization of spatial registration and histogram compensation for microscopic images," in *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE*, Aug 2006, pp. 3779–3782.

[31] Z. Xin-hua, G. Hui-dong, and X. Zhi-jun, "An asynchronous multisensor spatial registration algorithm," in *Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on*, vol. 4, Aug 2007, pp. 16–20.

[32] P. Anuta, "Spatial registration of multispectral and multitemporal digital imagery using fast fourier transform techniques," *Geoscience Electronics, IEEE Transactions on*, vol. 8, no. 4, pp. 353–368, Oct 1970.

[33] R. A. Redlich, "Desarrollo de hardware dedicado para procesamiento de imágenes en tiempo real," Master's thesis, Departamento de Ingeniería Eléctrica, Universidad de Concepción, Septiembre 2014.

[34] X. Zhang and D. Wu, "Substation remote infrared image registration based on multi-scale harris corner and hierarchical guiding match strategy," in *Image and Signal Processing (CISP), 2010 3rd International Congress on*, vol. 6, Oct 2010, pp. 2681–2685.

[35] N. Li, L. Huang, F. Jie, and G. Zhang, "Registration of infrared image and digital graphic based feature," in *Pattern Recognition (CCPR), 2010 Chinese Conference on*, Oct 2010, pp. 1–5.

[36] V. Agostini, S. Delsanto, F. Molinari, and M. Knaflitz, "Evaluation of feature-based registration in dynamic infrared imaging for breast cancer diagnosis," in *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE*, Aug 2006, pp. 953–956.

[37] J. M. Mooney, F. D. Sheppard, W. S. Ewing, J. E. Ewing, and J. Silverman, "Responsivity nonuniformity limited performance of infrared staring cameras," *Optical Engineering*, vol. 28, no. 11, pp. 281 151–281 151–, 1989. [Online]. Available: http://dx.doi.org/10.1117/12.7977112

[38] D. Scribner, K. Sarkady, M. Kruer, J. Caulfield, J. Hunt, M. Colbert, and M. Descour, "Adaptive retina-like preprocessing for imaging detector arrays," in *Neural Networks, 1993., IEEE International Conference on*, 1993, pp. 1955–1960 vol.3.

[39] S. Jian, Z. Enhui, C. Le, H. Yanyan, and F. Yaqiong, "Neural network based correction of infrared thermal imager for short distance measurement," 2009.

[40] S. N. Torres, C. S. Martin, D. G. Sbarbaro, and J. E. Pezoa, *Image Analysis and Recognition: Second International Conference, ICIAR 2005, Toronto, Canada, September 28-30, 2005. Proceedings.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, ch. A Neural Network for Nonuniformity and Ghosting Correction of Infrared Image Sequences, pp. 1208–1216. [Online]. Available: http://dx.doi.org/10.1007/11559573_146

[41] P. Goyal, "Nuc algorithm based on kalman filter for improvement in detector offset and gain drift," in *Emerging Trends in Electronic and Photonic Devices Systems, 2009. ELECTRO '09. International Conference on*, Dec 2009, pp. 136–139.

[42] R. W. Means, B. Von Tersch, and D. A. Scribner, "Adaptive infrared non-uniformity correction," DTIC Document, Tech. Rep., 1999.

[43] S. N. Torres, E. M. Vera, R. A. Reeves, and S. K. Sobarzo, "Adaptive scene-based nonuniformity correction method for infrared-focal plane arrays," in *AeroSense 2003*. International Society for Optics and Photonics, 2003, pp. 130–139.

[44] N. Celedon, R. Redlich, and M. Figueroa, "Fpga-based neural network for nonuniformity correction on infrared focal plane arrays," in *Digital System Design (DSD), 2012 15th Euromicro Conference on*, Sept 2012, pp. 193–200.

[45] R. Redlich, G. Carvajal, and M. Figueroa, "An fpga-based real-time nonuniformity correction system for infrared focal plane arrays," in *Application-Specific Systems, Architectures and Processors (ASAP), 2011 IEEE International Conference on*, Sept 2011, pp. 202–208.

[46] L. Araneda and M. Figueroa, "Real-time digital video stabilization on an fpga," in *Digital System Design (DSD), 2014 17th Euromicro Conference on*, Aug 2014, pp. 90–97.

[47] E. Yaman and S. Ertürk, "Image stabilization by kalman filtering using a constant velocity camera model with adaptive process noise," *Proc. of ELECO*, pp. 152–155, 2001.

[48] S. Auberger and C. Miro, "Digital video stabilization architecture for low cost devices," in *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, Sept 2005, pp. 474–479.

[49] P. Meza, G. Machuca, S. Torres, C. S. Martin, and E. Vera, "Simultaneous digital super-resolution and nonuniformity correction for infrared imaging systems," *Appl. Opt.*, vol. 54, no. 21, pp. 6508–6515, Jul 2015.