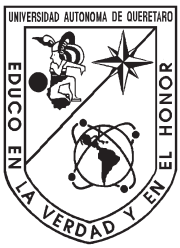


UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

Facultad de Ingeniería

Maestría en Instrumentación y Control Automático

Implementación del algoritmo SAD para el empate de imágenes estéreo en FPGA

TESIS

Que como parte de los requisitos para obtener el grado de

MAESTRÍA EN INSTRUMENTACIÓN Y CONTROL AUTOMÁTICO

presenta

Gerardo Ornelas Vargas

dirigido por

Dr. Rodrigo Castañeda Miranda

M. en C. Juan José García Escalante

Centro Universitario, Querétaro, Qro. Junio de 2008

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestría en Ciencias Línea Terminal en
Instrumentación y Control Automático

IMPLEMENTACIÓN DEL ALGORITMO SAD PARA EL EMPATE DE IMÁGENES ESTÉREO EN FPGA

TESIS

Que como parte de los requisitos para obtener el grado de

Maestro en Ciencias

Presenta:
Gerardo Ornelas Vargas

Dirigido por:
Dr. Rodrigo Castañeda Miranda

SINODALES

Dr. Rodrigo Castañeda Miranda
Presidente — Firma

M. en C. Juan José García Escalante
Secretaria — Firma

Dr. Roque Alfredo Osornio Rios
Vocal — Firma

Dr. Ivan Terol Villalobos
Suplente — Firma

Dr. René de Jesús Romero Troncoso
Suplente — Firma

Nombre y Firma
Dr. Gilberto Herrera Ruiz
Director de la Facultad

Nombre y Firma
Dr. Luis Gerardo Hernández Sandoval
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Diciembre de 2008
México

Resumen

El presente trabajo es la primer parte de la implementación de un sistema genérico de visión estereoscópica en FPGA. Abarca desde la interfaz con los sensores de imagen CMOS, hasta la implementación de un algoritmo de búsqueda por área para la generación de mapas de disparidades. Las etapas de calibración y rectificación fueron relizadas con herramientas de Matlab. El aporte del trabajo es un enfoque novedoso en la implementación del algoritmo de búsqueda por área logrando incrementar la capacidad de procesamiento de 1.03 fps hasta 12.79 fps con la posibilidad de duplicarlo agregando una serie de registros. Respecto a otros trabajos se mejora el desempeño del algoritmo quedando a un nivel competitivo con algoritmos de busqueda por características. Como parte de los resultados, se muestran los recursos utilizados en el FPGA por cada una de las etapas implementadas y los resultados de cada etapa del procesamiento con las imágenes de prueba sawtooth, venus, cones y Tsukuba.

Abstract

This work is the generic stereo vision system FPGA implementation first part. It embraces from CMOS image sensor interface, to disparity map generator by an area based search algorithm implementation. Calibration and rectification steps where carried out with Matlab tools. This work contribution is a novel approach in area based algorithm implementation. This approach boost its processing capabilities from 1.03 fps to 12.79 fps, with possibilities of double this performance by mean of a series of registers. Making a comparision between this work and others, it turns out the improvement in performance, having processing capabilities that can compete with characteristics based search algorithms. As part of results, FPGA usage resources are shown for each implemented step. Also every processing step with test images sawtooth, venus, cones and Tsukuba.

Dedicado a todos aquellos que confían en lo que les digo.

Agradecimientos

Agradezco todos aquellos que me han apoyado hasta hoy, y que seguro lo seguirán haciendo el día de mañana. Gracias a todos, sin su participación en mi paso por el mundo este trabajo no hubiera sido realizado. Finalmente, al CONACYT por la beca que me permite estar en el posgrado.

Índice general

Índice de cuadros

Índice de figuras

Introducción

Actualmente existe un avance considerable en el desarrollo de algoritmos para el manejo de imágenes, sin embargo la mayoría de las implementaciones de estos algoritmos están hechas en software lo que reduce la eficiencia de los algoritmos en términos de tiempo. Es decir, no hay manera de procesar las imágenes de video al momento en que se están generando.

Hoy en día existen en el mercado circuitos integrados como los Procesadores Digitales de Señales (DSP, por sus siglas en inglés Digital Signal Processor) que al ser diseñados para el procesamiento de señales, permiten que los algoritmos que se implementan en estos cuenten con una mayor velocidad de realización. Sin embargo, hacer un sistema de multiprocesamiento o procesamiento en paralelo resulta en una labor muy complicada ya que se necesitan varios de estos chips y ello implica hacer un diseño PCB más costoso y complicado.

Los FPGA representan la mejor opción para poder implementar algoritmos de procesamiento de imágenes aprovechando el procesamiento en paralelo. Aunado a que da la flexibilidad de poder reconfigurar el chip por completo cuantas veces sea necesario. Esto significa que un diseño PCB enfocado a aprovechar las capacidades de reconfiguración del FPGA podría servir para hacer las pruebas de virtualmente todos los algoritmos que puedan ser implementados sin tener que utilizar mas circuitos FPGA u otras tarjetas.

¿Pero porque implementar el procesamiento en Hardware? Existen tres razones principales, la velocidad, el espacio y el costo de producción.

La velocidad se refiere a la cantidad de operaciones sobre la imagen que se pueden hacer por segundo, los FPGA tienen capacidad de hacer un procesamiento en tiempo real, es decir, 30 cuadros por segundo o más.

El espacio se refiere a la portabilidad, un sistema completo de visión estéreo puede ocupar, considerando las cámaras, un espacio de no más de $0{,}2dm^3$, podría ser capaz de funcionar de manera autónoma (utilizando baterías y/o paneles solares) y sería bastante sencillo de instalar. Un sistema en PC ocupa un mayor espacio y requiere una instalación eléctrica, lo que no le da mucha portabilidad.

Por último, el costo de producción. A nivel de prototipos el costo de desarrollo puede ser similar por el costo de las licencias para las distintas plataformas de FPGA que existen respecto de las soluciones en software. Sin embargo, una vez que el prototipo ha sido probado, depurado, corregido y validado, el costo de producción se reduce drásticamente, aproximadamente a una octava parte del costo de la solución con PC.

En este trabajo, se implementarán una serie de algoritmos para el procesamiento de imágenes estereoscópicas. Inicialmente, la etapa de configuración de los sensores de imágenes CMOS, así como la adquisición de las imágenes captadas por estos. La calibración y la rectificación de las imágenes quedaron fuera de línea por limitaciones de recursos de la tarjeta de pruebas y por la facilidad que presenta el matlab para estas operaciones. La transformación RANK también fue implementada en el FPGA. Aprovechando la capacidad de procesamiento y los recursos internos de los FPGA se replanteó la implementación del algoritmo más básico para el empate de imágenes, la suma de diferencias absoutas (SAD, en inglés Sum of Absolute Differences), para fines de comparación en uso de recursos y tiempo de procesamiento también se implementó el SAD en una versión similar a la que se emplea en software. Los resultados muestran una reducción en el tiempo de procesamiento de 9.5 veces, así mismo un incremento en el uso de recursos de 16.46 veces.

Las posibles aplicaciones de un instrumento de visión estéreo son, por mencionar algunas, el control de calidad, herramienta de soporte para el CAD/CAM, fitomonitoréo, construcción de mapas aéreos, en el área forense (reconstrucción de la escena de un crimen), monitoreo de estructuras, mediciones sin contacto, ingeniería inversa, análisis de formas, entre otras.

Objetivos e hipótesis del trabajo

Objetivo general

- Implementar un sistema de visión estereo en hardware, que pueda procesar 15 pares de cuadros por segundo, de 640x480 píxeles y un tamaño de ventana menor o igual a $\lambda = 3$, usando el algoritmo SAD.

Objetivos específicos

- Se implementará la interfaz con los sensores de imagen CMOS para obtener las imágenes de entrada.
- Se implementará y probará la transformación RANK como etapa de preprocesamiento para el empate de imágenes.
- Se implementará la forma tradicional del algoritmo para empate de imágenes SAD.
- Se implementará una forma alternativa del SAD que aproveche los recursos del FPGA para incrementar a cuando menos 15 pares de cuadros por segundo su capacidad de procesamiento.
- Verificación del algoritmo y cálculo de los recursos.

Hipótesis general

- Es posible implementar un sistema de visión estéreo en un FPGA de bajo costo que logre procesar 15 pares de cuadros por segundo bajo las condiciones de 640x480 píxeles de resolución y un tamaño de ventana de $\lambda = 3$, usando el algoritmo SAD.

Hipótesis específicas

- La naturaleza de la interfaz de los sensores CMOS exige un nivel de paralelismo alcanzable con un FPGA.
- La arquitectura de la transformación RANK tiene una implementación directa y económica en FPGA.
- La arquitectura del algoritmo de empate de imágenes SAD tiene también una implementación directa en FPGA.
- Es posible cambiar la forma de implementar el algoritmo SAD para aprovechar las capacidades del FPGA y mejorar el desempeño en términos del tiempo de procesamiento.

Capítulo 1

Antecedentes y justificación

1.1. Antecedentes

Un ejemplo del empleo de FPGA lo constituye el hecho de que los fabricantes de cámaras industriales han empezado a utilizarlos en la construcción de sus cámaras, para el preprocesamiento y la interconectividad de las cámaras por ejemplo el CAML-MOD3 de VMETRO para conectar los Camera Link, XRI-1200 DALSA un cooprocesador auxiliar para el procesamiento de imágenes, el sistema de cámaras CCA-BC200 de Sanyo. También existen trabajos como el de Hajimowlana et al. (1999) donde hacen un sistema de detección de defectos combinando un FPGA como interfaz con la cámara y bloque de preprocesamiento, un microprocesador como bloque de postprocesamiento y una PC para interfaz con el usuario. El trabajo de Nelson (2000) está enfocado a la implementación de las operaciones morfológicas básicas para el procesamiento de imágenes con una extensa explicación de la algoritmia a través de Matlab. Woodfill and Herzen (1997) propone un sistema de reconstrucción tridimensional basado en 16 FPGA con 16 MB de RAM con una topología toroidal para PCI. Hariyama et al. (2005) propone el diseño de un microprocesador para hacer el cálculo de la suma absoluta de diferencias logrando resultados de 80 veces mejor velocidad respecto al mismo algoritmo ejecutándose en un Pentium IV a 2 GHz. Por último, la empresa Videre Desing comercializa un dispositivo basado en un FPGA de Xilinx que obtiene las imágenes de dos CMOS ubicados de manera estereoscópica

y envía la disparidad entre las dos imágenes junto con el video a través de una interfaz Firewire, el costo del dispositivo es de $1,400 dólares.

Szappanos et al. (2008) desarrollaron un cooprocesador para activar el envío de imagenes tras la ocurrencia de algún evento programado para así evitar consumo del ancho de banda y procesamiento innecesario de imágenes. Ren et al. (2006) utilizan un FPGA Virtex II Pro con un CPU embebido para el sistema de rastreo por visión de un robot para incrementar la velocidad de procesamiento de las imágenes. Dorrington et al. (2007), diseñaron un sistema de medición de rango e intensidad para cada píxel de una escena con una presición submilimétrica usando un FPGA. Kumara et al. (2007) mejoran el desempeño de una cámara que capta rayos infrarrojos para dar una imágen térmica, la ventaja que muestran es que tiene una calidad estándar bajo diferentes condiciones ambientales. Garrido et al. (2005) hacen un prototipo para la codificación y decodificación del protocolo de video H.263 bajo un esquema de procesadores especializados en transformación, cuantización, estimación de movimiento y compensación de movimiento, controlados por un procesador central, todo el sistema está implementado en un FPGA. Karabernou and Terranti (2005) implementan la transformación Hough basados en algoritmos CORDIC (COordinate Rotation DIgital Computer) y el gradiente para detectar líneas, esta implementación cabe en un FPGA y logra un desempeño de tiempo real. Hirota et al. (2006) desarrollaron un sistema de detección de neutrones y su sistema de adquisición de datos usando FPGA como un elemento clave para lograr el desempeño requerido. Hussmann and Ho (2003) presentan una implementación en FPGA de un algoritmo de búsqueda de contornos a nivel de subpíxeles con una capacidad para 2000 cuadros por segundo pudiendo mejorarlo aun más dependiendo del sensor.

Por el lado de la reconstrucción tridimensional Oram (2001) hace una recopilación y propuesta del uso de las geometrías euclidianas y no euclidianas para hacer la reconstrucción tridimensional a partir de dos, tres y cuatro puntos focales con métodos lineales y no lineales. Igualmente explora algoritmos de autocalibración y tiene un alcance hasta algoritmos para video. Woo (1998), propone esquemas para la codificación de las imágenes estereoscópicas para evitar en lo posible el problema inherente a este tipo de imágenes

que es el aumento del ancho de banda, que requiere ser del doble, pues en vez de ser una imagen son dos. Las codificaciones que propone Woo son de suma utilidad para este trabajo debido a que las imágenes que propone enviar son imágenes ya procesadas en vez de las imágenes completas para su procesamiento anterior. Díaz et al. (2007a) aprovechan los recursos para el paralelismo que ofrecen los FPGA y obtienen un procesador de imágenes estéreo con un desempeño de 52 cuadros por segundo en imágenes de 1280x960 píxeles. Díaz et al. (2007b) proponen un algoritmo para el empate de imágenes basado en sistemas naturales con muy buenos resultados en términos de relación desempeño/recurso, la implementación la realizan en un FPGA.

En aplicaciones prácticas de sistemas de visión estéreo tenemos trabajos como el de Huh et al. (2008) donde aplican la visión estereo para la detección de obstaculos en vehiculos aplicando varios métodos de empate para mejorar la robustez del sistema. Muñoz-Salinas et al. (2007) usan la visión estéreo y el color para la detección y seguimiento de personas con miras a la aplicación en interacción humano-máquina. Sun et al. (2007) aplican la visión estéreo para medir el ancho y detección de pliegues en granos de avena. Bertozzi et al. (2007) presentan un sistema de visión estereo para la detección de peatones usando cámaras de lejano infrarrojo, combinando tres formas de detección para diferentes condiciones ambientales. Tian et al. (2007) desarrollaron un sistema que permite generar un model 3D de las piezas considerando incluso los soportes para estas en las maquinas-herramienta, de manera que se pueden generar los programas de control numérico libres de colisiones. Rovira-Más et al. (2008) proponen un método para generar mapas 3D de los campos basado en imágenes estéreo e información de GPS, estos mapar sirven de apoyo para la agricultura de precisión. Sun et al. (2006) utilizan la visión estéreo para medir la distancia entre la vegetación y las líneas de transmisión de energía eléctrica y con esto tomar medidas de prevención y seguridad al respecto. Andersen et al. (2005) muestran el potencial de la visión estéreo para la medición de características como altura y área foliar de plantas en cultivos. Estas son solo una pequeña muestra de las aplicaciones para visión estéreo que están siendo investigadas. En la siguiente sección se explicarán los conceptos básicos de la visión estereocópica.

1.2. Fundamentación teórica

En la reconstrucción tridimensional existen varias etapas que se deben de considerar:

Los principales parámetros asociados a la adquisición de imágenes son:

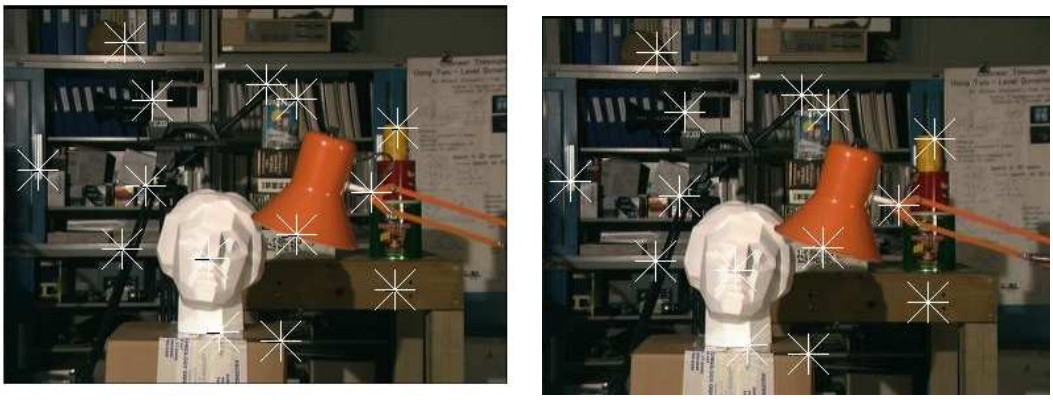
- Escenario
- Diferencia de tiempo entre imágenes
- Hora del día
- Fotometría (Medición de las propiedades de la luz)
- Resolución
- Posición relativa de las cámaras

Adquisición de imagen

La adquisición se puede realizar en varias formas, por ejemplo, con imágenes del mismo instante, con una ligera diferencia en el tiempo o incluso de días distintos. También pueden ser tomadas de posiciones ligeramente distintas o con una amplia diferencia. La hora y las condiciones atmosféricas resultan ser importantes en las imágenes que son de días distintos.

Modelado de la cámara

Se refiere a encontrar los puntos de correspondencia en las imágenes estéreo. Los puntos de correspondencia son proyecciones de un solo punto en un escenario tridimensional (figura 1.1). La diferencia en las posiciones de dos puntos de correspondencia en su respectiva imagen es llamada disparidad. La disparidad es una función tanto de la posición del punto en la escena y la posición, orientación y características físicas de las cámaras estereoscópicas. En la figura 1.2 podemos ver un punto P_L con coordenadas en el espacio



(a) Perspectiva izquierda (b) Perspectiva derecha

Figura 1.1: En a) se muestran puntos de la escena en la imagen de la cámara izquierda, en b) se muestran estos mismos puntos pero en la imagen de la cámara derecha.

(X, Y, Z) cuya proyección en la imagen izquierda está dada por la coordenada (u_L, v_L) y en la imagen derecha por (u_R, v_R). O_R y O_L son la ubicación de las cámaras derecha e izquierda respectivamente. En el caso del diagrama, hay un desplazamiento solo en el eje x por lo que $v_R = v_L$ así que la disparidad (d) entre ese punto en una y otra imagen está dada por $d = u_L + u_R$.

Cuando se conocen los atributos de estas cámaras, los puntos de correspondencia pueden ser mapeados en un escenario en tres dimensiones. Un modelo de cámara es una representación de los atributos físicos y geométricos de las cámaras estereoscópicas. Debe tener un componente relativo, el cual relaciona el sistema de coordenadas de una cámara con la otra y es independiente de la escena, y también debe tener un componente absoluto que relaciona el sistema de coordenadas de una de las cámaras con el sistema de coordenadas del escenario (Trucco and Verri, 1988). Además de proveer la función que mapea pares de puntos de correspondencia en puntos del escenario, un modelo de cámara puede ser usado para obligar a la búsqueda de puntos de emparejamiento de los puntos de correspondencia de la imagen en una sola dimensión. Cualquier punto en el mundo tridimensional, junto con los centros de proyección de un sistema de dos cámaras, definen un

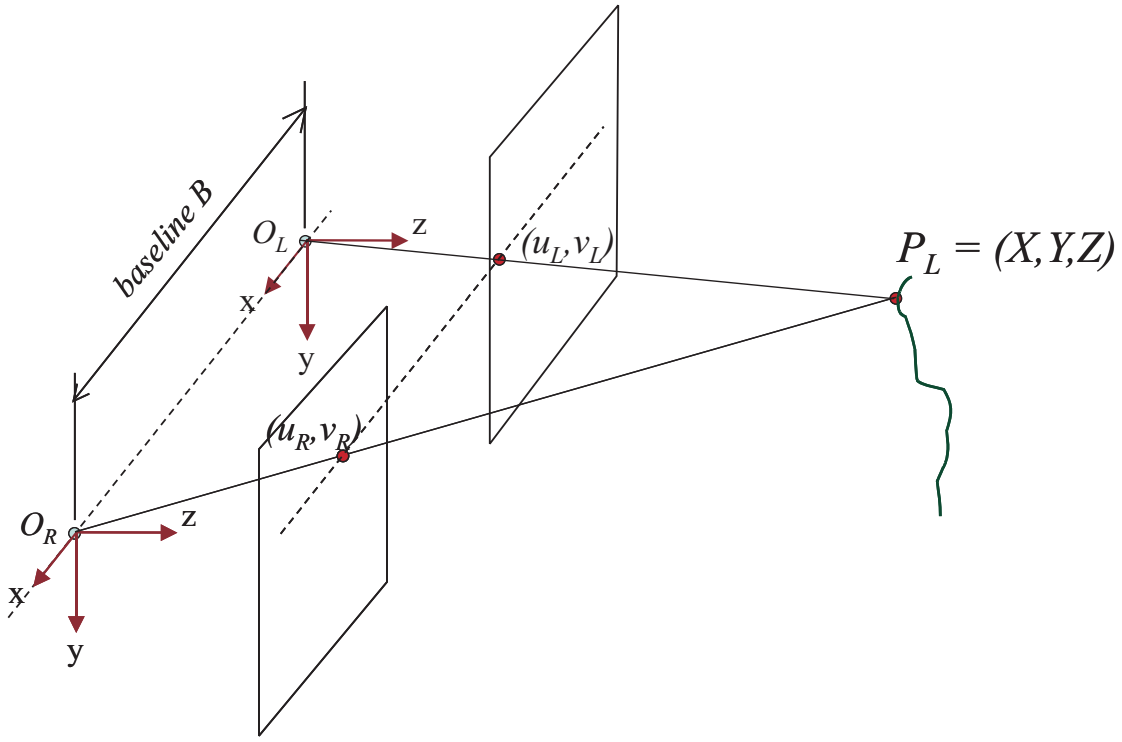


Figura 1.2: Diagrama de la localización de un punto en el espacio real y sus proyecciones en dos imágenes con distintas perspectivas.

plano llamado epipolar. La intersección de un plano epipolar con el plano de una imagen es llamada línea epipolar (figura 1.3). Todo punto en una línea epipolar dada debe corresponder a un punto en la línea epipolar en la otra imagen. La búsqueda del emparejamiento de un punto en la primer imagen debe por lo tanto ser limitado a una vecindad unidimensional en el plano de la segunda imagen, con lo que se logra una reducción considerable en la complejidad computacional.

Cuando las cámaras son orientadas de forma tal que solo exista un desplazamiento horizontal entre ellas, entonces la disparidad puede ocurrir solamente en dirección horizontal, y las imágenes estéreo se dice que están en correspondencia. Cuando un par de imágenes estéreo están en correspondencia, las líneas epipolares coinciden con las líneas de escaneo

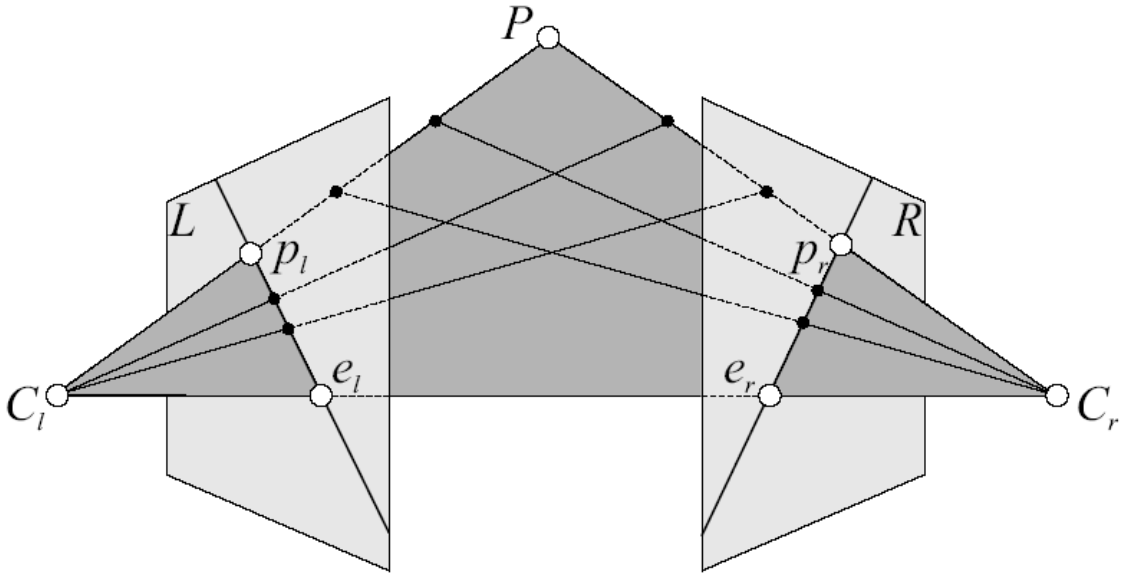


Figura 1.3: Ejemplo de geometría epipolar.

horizontal de las fotografías digitalizadas lo que hablita el emparejamiento en una manera relativamente simple y eficiente. En caso de que la distorsión radial o bien, la disposición mecánica de las cámaras no permitan la alineación para que solo existe al desplazamiento horizontal, es necesario hacer una rectificación de las imágenes.

La diferencia en posición y orientación de dos cámaras estereoscópicas es llamado el modelo relativo de la cámara. Estos modelos se requieren para determinación de la profundidad y para aprovechar la posibilidad de forzar el plano epipolar. Gennery (1979) desarrollo un método para encontrar el modelo relativo de la cámara utilizando unos cuantos puntos de emparejamiento. Su método busca diferencias en ángulo, elevación, desplazamiento, inclinación y distancia focal. Fischler and Bolles (1981) han dado varios resultados con respecto al número mínimo de puntos que se necesitan para obtener una solución del problema de modelado de cámara, dando una sola imagen y una seria de correspondencias entre los puntos en la imagen y los ubicaciones espaciales de los puntos; también proveen una técnica para encontrar el modelo de cámara completo, a pesar de que las correspon-

dencias tengan un alto porcentaje de error. Aunque este trabajo estaba dirigido al problema de establecer un mapeo entre una imagen y una base de datos de locaciones geográficas, es posible aplicar los resultados al problema estéreo.

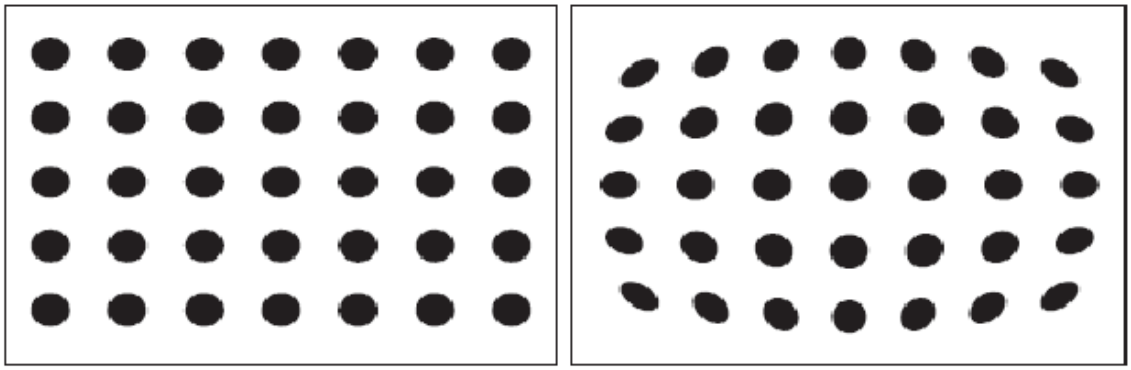


Figura 1.4: Ejemplo de distorsión radial.

Hay dos errores que se deben corregir en la cámara, estos son los errores por distorsión tangencial y los errores por distorsión radial. La distorsión tangencial es cuando no existe un centrado en los lentes, el efecto es que hay un desplazamiento perpendicular con respecto a las líneas radiales. La distorsión radiales es cuando existe una imperfección de curvatura en los lentes, esto provoca un desplazamiento radial alrededor del eje óptico de la cámara (figura 1.4).

En resumen lo importante en el modelo de la cámara es:

- Conocimiento de la posición de las cámaras y de sus parámetros.
- Soluciones con escasos puntos de empate.
- Conocimiento de la ubicación en tres dimensiones de lo que observamos.
- Habilidad para solventar errores.
- Compensación por la distorsión de las imágenes.

1.2.1. Adquisición de las características

(a) Perspectiva izquierda (b) Perspectiva derecha

Figura 1.5: En este par de imágenes, las característica lineales son los contornos de las ventanas y la banqueta, mientras que las características puntuales pueden ser todos los vértices de las mismas

Es la manera en la que uno selecciona un punto o un área con la cual va a hacer el emparejamiento de las imágenes. Esta selección tiene que ser acorde en medida de lo posible con el método de empate.

Existe una preferencia sobre las características con aspecto de punto (figura 1.1), sobre todo cuando el modelo de la cámara no es conocido y los emparejamientos no están en las líneas epipolares, esto se debe a que a diferencia de las características lineales (figura 1.5) los puntos no son ambiguos y se pueden emparejar en cualquier dirección . Por otro lado, las características lineales deben de estar orientadas a lo largo de las líneas epipolares si es que se quieren emparejar de manera correcta. Otra ventaja de las que tienen forma de punto respecto a las de línea es que pueden ser emparejadas sin importar la distorsión de la perspectiva. En los enfoques de correlación de área, los emparejamientos de puntos se hacen para obtener el modelo de cámara antes de seguir con emparejamientos más extensos.

Si el modelo de cámara es conocido o derivado en un paso previo, entonces los bordes pueden ser utilizados como características de emparejamiento. Muchos modelos de contorno distintos han sido propuestos como la base para algoritmos de detección de contorno. Típicamente un algoritmo de este tipo no produce una línea perfectamente bien definida sino que produce además una magnitud que se relaciona con el contraste alrededor del eje y algunas veces incluso una "dirección" para el eje. En el caso de ejes con "magnitud" grande, la mayoría de los algoritmos resultantes producen resultados similares para operadores de tamaños similares. Aunque el tamaño, dirección y magnitud han sido utilizados como características para tomar decisiones de emparejamiento, su mérito relativo no ha sido establecido.

En resumen las propiedades de las características locales que son importantes para el problema de la visión estereoscópica son:

- Forma
- Tamaño
- Contraste
- Contenido semántico
- Densidad de ocurrencia
- Atributos fácilmente medibles
- Únicas / Distinguibles

Empate de las imágenes

Aquí hay que hacer una diferenciación entre el simple empate de imágenes y el empate o emparejamiento estereoscópico de imágenes, para eso nos basaremos en los siguientes puntos:

- Las diferencias importantes en las imágenes estéreo son resultado de un cambio en el punto de vista y no por cambios, por ejemplo, de escenario.

- La mayoría de los cambios significativos van a ocurrir en la apariencia de objetos cercanos y obstrucciones. Otros cambios en geometría y fotometría pueden darse por un mal control de las condiciones un la toma de la imagen, pero son evitables.

- El modelado estereoscópico generalmente requiere que una alta cantidad de puntos continuos sean emparejados.

Idealmente quisiéramos encontrar correspondencias de cada píxel en ambas imágenes de un par estéreo. Sin embargo, es obvio que la información contenida en un solo píxel es insuficiente para un emparejamiento sin ambigüedad. En la práctica se requiere que una cantidad coherente de píxeles sea la que se va a emparejar. Estos píxeles son determinados y emparejados de dos maneras distintas:

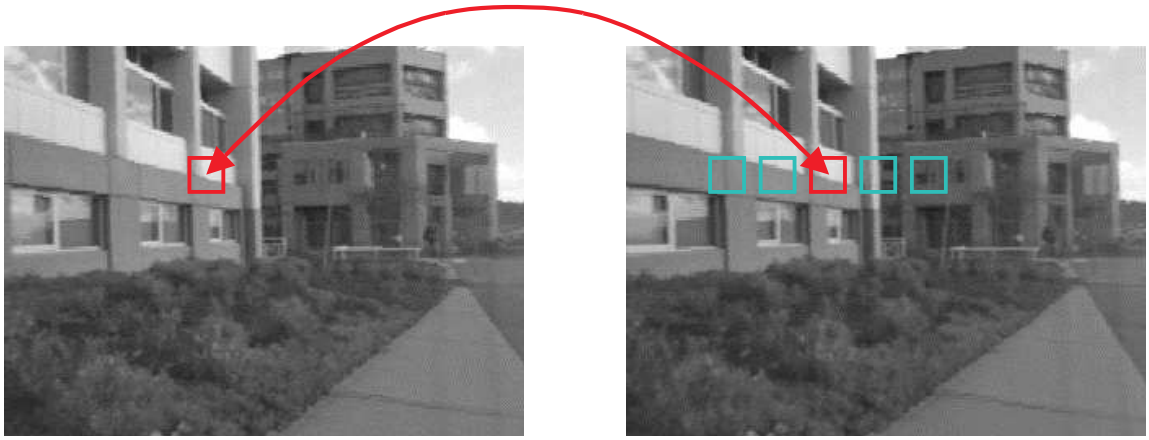


Figura 1.6: Ejemplo de una búsqueda por área.

- Por Área. Vecindades de píxeles de tamaño regular son las unidades básicas que son empatadas. Este enfoque se basa en el “supuesto de continuidad” que afirma que a cierta resolución la imagen se compone de porciones de superficies continuas, por lo tanto, píxeles adyacentes en la imagen representan puntos contiguos en el espacio. Este enfoque es acompañado de manera casi invariable por las técnicas de emparejamiento basadas en correlación para establecer las correspondencias. La figura 1.6 muestra un ejemplo de la búsqueda por área, se toma una ventana de una

de las imágenes y se compara con otras ventanas en la otra imagen hasta encontrar una correpondencia.

- Por Característica. “Características semánticas” (Cuando se conocen características físicas y/o la geometría espacial) o “características de intensidad anómala” (patrones aislados de intensidad anormal que no necesariamente tienen significado físico) son las unidades básicas que se emparejan. Características semánticas del tipo genérico incluyen bordes de obstrucciones (figura 1.7), vértices de estructuras lineales, y marcadores de superficies prominentes(figura 1.8); las de dominio específico pueden incluir la esquina o el pico de un edificio o la superficie de una calle; las características de intensidad anómala incluyen cruces por cero. Los métodos para los emparejamientos por características incluyen técnicas de clasificación simbólica, así como correlación.

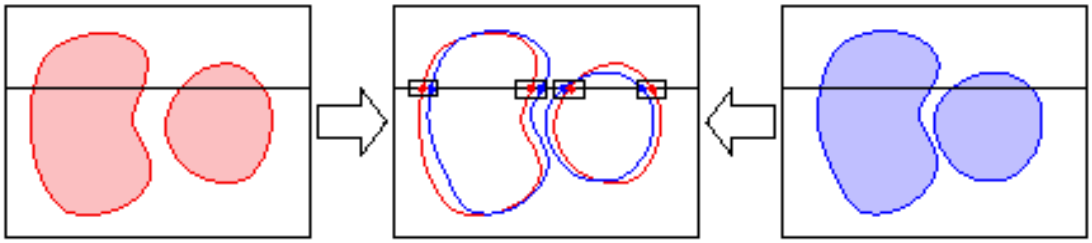


Figura 1.7: Ejemplo de una búsqueda por área.

Obviamente el emparejamiento por características no puede dar un mapa de profundidad útil por si mismo. Esto hace que tenga que ser aumentado con un paso de interpretación basada en modelo o por un emparejamiento por área. Cuando se usa en conjunto con el emparejamiento por área, el emparejamiento por características se considera que es más confiable que el emparejamiento por área solo y puede forzar la búsqueda de coincidencias de correlación.

El enfoque de emparejamiento por correlación intenta resolver los problemas de ambigüedad usando toda la información local que sea posible para tomar decisiones acerca

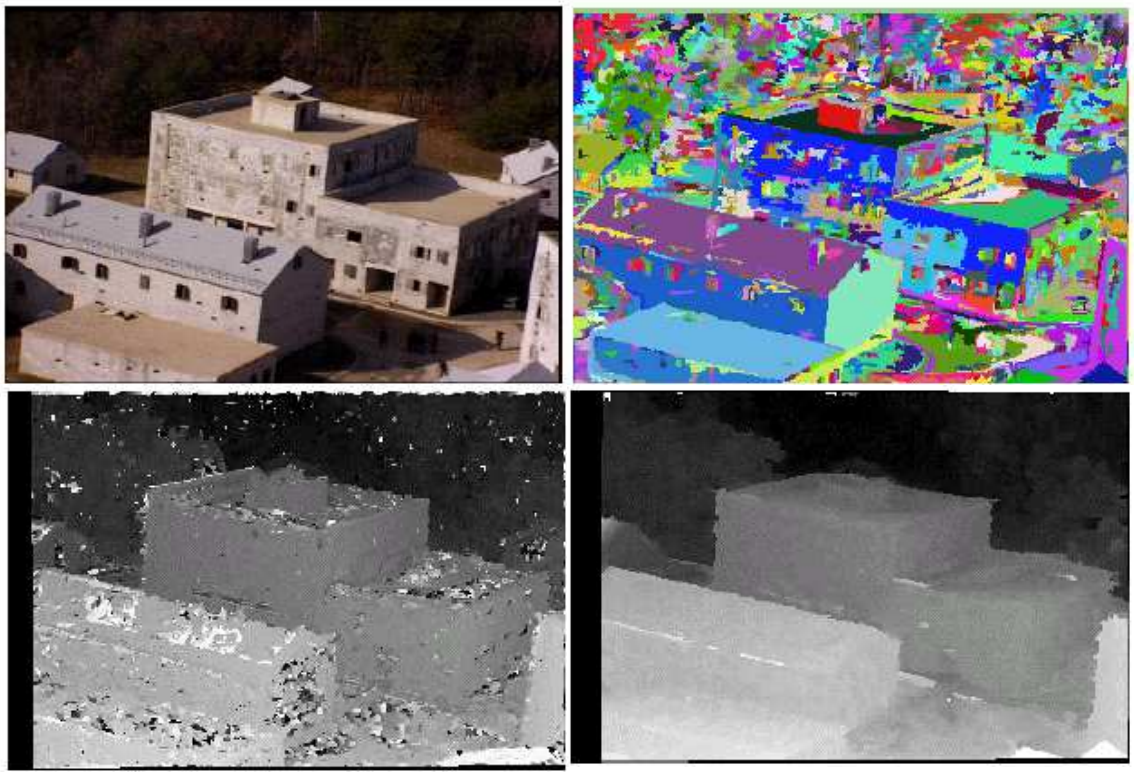


Figura 1.8: Ejemplo de una búsqueda por área.

de las coincidencias potenciales, pero toma cada decisión de manera independiente a las demás. El enfoque de etiquetado relajado no utiliza toda la información disponible, sin embargo las decisiones las toma en base al total de las posibles coincidencias.

El emparejamiento es complicado por distintos factores relacionados con la geometría de las imágenes estéreo. Algunas áreas visibles en una imagen pueden estar obstruidas en la otra, esto puede llevar a emparejamientos erróneos. También las estructuras periódicas pueden generar confusión en el algoritmo por confundir un punto con otro especialmente si las características de la imagen generadas por estas estructuras están demasiado juntas respecto de la disparidad de las características.

En resumen, los atributos que diferencian las técnicas de emparejamiento incluyen:

- Resolución de ambigüedad global versus local
- Emparejamiento por área versus por características

Para obtener un mapa de disparidad de alta densidad de píxeles sería necesario comparar una característica con toda la imagen contraria, esto resta velocidad a los algoritmos por lo que es necesario reducir en lo posible los accesos a memoria y/o la cantidad de operaciones que hay que realizar y una vez encontrado un punto candidato verificar que este sea por la posible ambiguedad. Las condicionantes usadas para reducir tanto complejidad computacional como ambigüedad incluyen:

- Epipolar
- Continuidad
- Jerarquía
- Secuenciación

El criterio que puede ser utilizado para evaluar distintas técnicas de emparejamiento incluyen:

- Exactitud (precisión del empate a un nivel de subpíxeles)
- Confiabilidad (robustez frente a los errores de clasificación)
- Generalidad (aplicabilidad a diferentes tipos de escenarios)
- Previsibilidad (disponibilidad del desempeño de los modelos)
- Complejidad (costo de implementación, requerimientos computacionales)

1.2.2. Determinación de la profundidad.

La determinación de profundidad no ha sido un trabajo principal en la reconstrucción tridimensional debido en parte a que una vez que las imágenes han sido emparejadas el cálculo de la distancia es una cuestión de triangulación. Aún así, este paso tiene ciertas dificultades sobre todo si los emparejamientos son un poco imprecisos o poco confiables.

Para empezar, el error en la medición de la distancia es directamente proporcional al error posicional de los emparejamientos e inversamente proporcional a la longitud de la línea estereoscópica básica. Alargar la línea estereoscópica complica el problema del emparejamiento al incrementar tanto el rango de disparidad como la diferencia en apariencia de las características a emparejar. Varias estrategias han sido utilizadas para superar este problema.

En muchos casos, los emparejamientos se hacen con una precisión de tan solo un píxel. Sin embargo, tanto la correlación de área como el emparejamiento por características proveen una mejor precisión. La precisión de subpixel utilizando la correlación de área requiere de una interpolación sobre la correlación de superficie. A pesar de algunos métodos de detección de características pueden localizar características mas grandes que un píxel, se depende mucho del tipo de operador que se utilice, y para esto no hay técnicas generales que se puedan aplicar.

Otro enfoque es establecer una precisión de un píxel pero utilizando múltiples vistas. Un empate de un par de vistas en particular representa un estimado en la profundidad cuya incertidumbre depende tanto de la precisión del emparejamiento y la longitud de la línea estereoscópica. Empates de muchos pares de vistas pueden ser promediados de manera estadística para encontrar un estimado más preciso. La contribución de un empate al estimado de la profundidad final puede ser medido de acuerdo con los factores que intervienen en la confianza o precisión del emparejamiento.

En resumen, las medidas de profundidad se pueden mejorar de varias maneras, con un

costo computacional adicional:

- Estimación del subpixel
- Incrementar la línea estereoscópica
- Promedio estadístico sobre múltiples vistas.

1.3. Tecnologías para la implementación de los algoritmos

Aquí hago una breve descripción de las posibilidades de implementación, sus ventajas y desventajas. Adelanto que la opción a utilizar es el diseño en hardware a través de FPGA, utilizando como lenguaje descriptivo de hardware el VHDL por las razones que se numeran más adelante.

1. Programas para el Procesamiento Digital de Señales para PC.

   Los programas de procesamiento digital de señales en PC permiten un desarrollo rápido de algoritmos, así como correcciones y pruebas de los mismos. Es común que los diseñadores de hardware prueben los algoritmos de manera previa en alguno de estos programas previo al inicio del diseño en hardware.

   Un ejemplo de este tipo de programas es el Matlab. Matlab esta diseñado para manipular matrices en general, para varias aplicaciones de procesamiento de imágenes resulta ideal debido a que maneja las imágenes como matrices. Sin embargo cuando la meta es una implementación en hardware usualmente los algoritmos se describen de una manera similar a la que se piensa utilizar en hardware, esto deriva en algoritmos más lentos.

Sistemas como IDL(por sus siglas en inglés, Interactive Data Languaje) y su componente gráfico ENVI (acrónimo de ENvironment for Visualizing Images) son enfocados a aplicaciones de procesamiento de imágenes e incluyen varios algoritmos preprogramados de uso común en el procesamiento de imágenes, un ejemplo de aplicación lo dan Xie et al. (2003). Pero aún los programas especializados corriendo en PC no son capaces de procesar bloques de imágenes de alta resolución debido a que los procesadores de las PC son de propósito general. De aquí que la optimización tenga que ser a nivel de hardware.

2. Circuitos Integrados de Aplicación Específica.

   Circuitos Integrados de Aplicación Específica (ASIC, por sus siglas en inglés) representan una tecnología donde los ingenieros crean un hardware destinado a una aplicación específica, cuyo diseño utiliza distintos tipos de herramientas. Una vez que el diseño ha sido implementado dentro del ASIC, no puede ser modificado. La capacidad de integración que ofrecen estos dispositivos es tal, que el permite el paralelismo de los algoritmos. Sin embargo, exceptuando aplicaciones comerciales de alto volumen, ASIC son considerados como demasiado costosos para la mayoría de los diseños. Además, si el diseño del hardware tiene presenta algún error que no haya sido descubierto antes de la producción no puede ser corregido sin un costo considerablemente alto.

3. Procesadores Digitales de Señales Dedicados.

   Procesadores Digitales de Señales (DSP, en ingles) como los que produce Texas Instruments son un tipo de dispositivos de hardware que se encuentran entre los ASIC y la PC en términos de desempeño y complejidad de diseño. Pueden ser programados tanto con ensamblador como con código C, lo que representa una de las ventajas de esta plataforma. Aún es necesario tener conocimientos de diseño de hardware,

pero la curva de aprendizaje es menor que en algunas otras opciones de diseño, debido a que la mayoría de los ingenieros conocen el lenguaje C antes de trabajar con los sistemas DSP. Sin embargo el paralelismo solo puede ser explotado utilizando múltiples DSP. El desempeño de los algoritmos ciertamente es mejor que en una PC, pero en algunos casos, los sistemas ASIC o FPGA son la única opción para un diseño. Aún así, DSP son un método común y eficiente para procesar los datos en tiempo real.

Un área donde los DSP son particularmente útiles es en el diseño de sistemas con punto flotante. En ASIC como en FPGA, las operaciones de punto flotante ofrecen retos difíciles para su implementación aunque en el procesamiento de imágenes no es realmente importante debido a que las imágenes consisten exclusivamente de números enteros.

Los avances tecnológicos han desembocado en la aparición de DSP y FPGA de muy alto desempeño, lo que impacta directamente en las capacidades de los algoritmos. Se espera que cada una de estas arquitecturas mejore su desempeño y capacidad de manera similar, generación de chips tras generación.

4. FPGA

Representan la tecnología de la computación reconfigurable, la cual de alguna manera resulta ideal para el procesamiento de video. El cómputo reconfigurable se refiere a procesadores que pueden ser programados con un diseño, y después reprogramados (o reconfigurados) una cantidad virtualmente ilimitada de acuerdo con los cambios de las necesidades del diseñador. Los FPGA generalmente consisten de bloques combinacionales o Look Up Tables (LUT), flip flops y algo de memoria de acceso aleatorio que es “cableada” utilizando una vasta red de interconexiones. Toda la lógica en el FPGA puede ser “recableada” o reconfigurada, con diferentes diseños, tantas veces como el diseñador requiera.

Los FPGA tienen capacidad suficiente para hacer diseños en paralelo dentro del mismo chip, cosa que no es posible con los diseños para DSP. Los métodos de diseño de ASIC pueden ser aplicados en los diseños para FPGA lo que le permite al diseñador implementar diseños a nivel de compuertas. Sin embargo, los ingenieros usualmente utilizan algún lenguaje descriptivo de hardware como VHDL o Verilog, que permiten una metodología de diseño similar al diseño de software. Este enfoque permite reducir el costo de soporte y de abstracción del diseño.

1.4. Opciones para el diseño en FPGA

Para hacer un diseño en FPGA, el diseñador cuenta con varias opciones para la implementación del algoritmo. Aunque el diseño a nivel de compuertas resulte en diseños altamente optimizados, la curva de aprendizaje se considera prohibitiva para la mayoría de los ingenieros, y el conocimiento adquirido puede no ser portable a través de las arquitecturas de los FPGA. A continuación se presentan algunos Lenguajes Descriptivos de Hardware (HDL en inglés) de alto nivelen los que se pueden diseñar los algoritmos para FPGA.

Verilog HDL

Originalmente fue creado como lenguaje de simulación. Actualmente puede ser utilizado para síntesis de diseños de hardware y muchas herramientas de software ofrecen soporte para este lenguaje. Es similar a otros HDLs, pero su nivel de adopción ha ido decreciendo a favor del VHDL que es un estándar mas abierto. Aún hay bastantes diseñadores que prefieren Verilog sobre de VHDL para diseño de hardware, y algunos departamentos de diseño utilizan solamente Verilog. Como diseñador de hardware es importante, cuando menos conocerlo.

Altera HDL

Altera HDL (AHDL) es de uso exclusivo y solo tiene soporte con las herramientas de desarrollo específicas de Altera. Esto puede considerarse como una deficiencia, sin embargo, al ser exclusivo, su uso puede resultar en un diseño de hardware más eficiente, siempre y cuando la portabilidad no sea requisito. Típicamente en el ambiente de diseño, distintas arquitecturas de FPGA son utilizadas para distintos diseños, lo que significa que el tiempo que se invierta aprendiendo AHDL sea desperdiciado si mas tarde se elige un FPGA de Xilinx por ejemplo.

Circuitos Integrados de muy Alta Velocidad HDL

En años recientes, el Circuitos Integrados de muy Alta Velocidad HDL (VHDL) se ha convertido en una especie de estándar industrial para el diseño de hardware de alto nivel. Debido a que es un estándar abierto de IEEE, tiene soporte por una gran variedad de herramientas de diseño y es intercambiable (siempre que se apegue al estándar) entre las herramientas de los fabricantes. También soporta la inclusión de módulos de tecnología específica para una síntesis más eficiente en los FPGA.

La primera versión de VHDL, IEEE 1076-87, apareció en 1987 tuvo una actualización en 1993 titulada IEEE 1076-93. Este es un lenguaje de alto nivel similar al lenguaje de programación Ada, que pretende apoyar el diseño, verificación, síntesis y prueba de diseños de hardware.

Capítulo 2

Materiales y métodos

2.1. Parámetros y calibración de la cámara

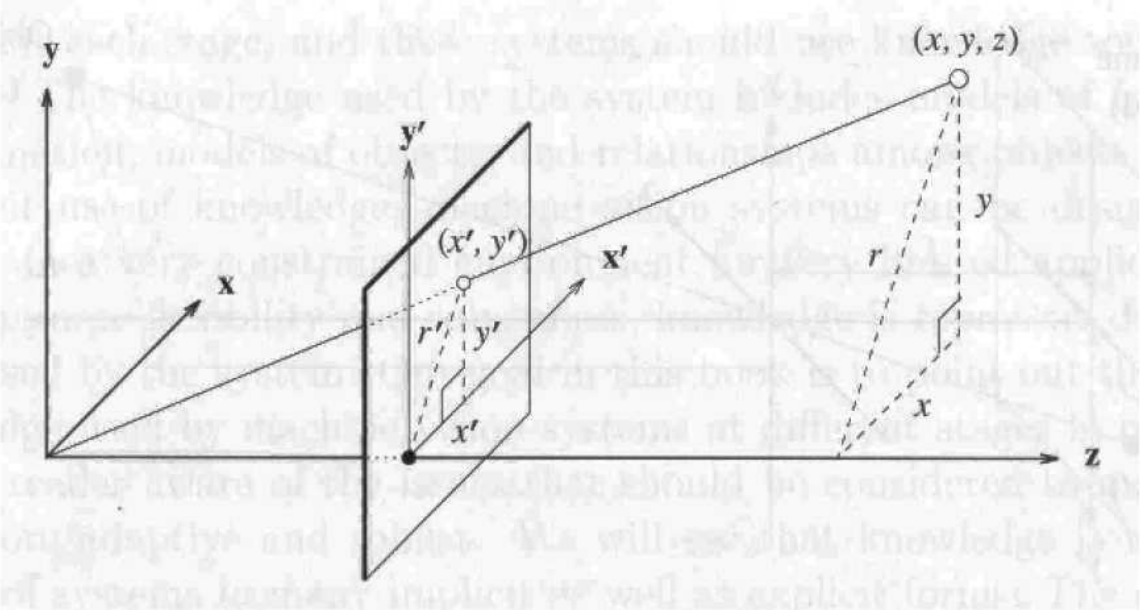


Figura 2.1: Proyección del punto del espacio real en el plano de la imagen.

La figura 2.1 nos permite apreciar la necesidad de conocer la información acerca de

la ubicación, orientación, distancia focal, distorsiòn de los lentes entre otros. Esto para poder hacer una estimación real de la imagen que obtenemos. Es decir, conociendo los parámetros del sistema de cámaras, podemos obtener imágenes rectificadas que tengan una relación directa entre las distancias entre píxeles con las distancias reales entre objetos. El orden en el que se da esta transformación se ilustra en la figura 2.2

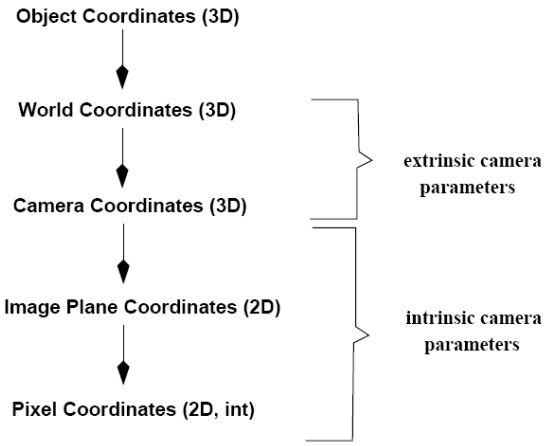


Figura 2.2: Flujo de la transformación de las coordenadas del mundo real a la imagen a partir de los parámetros de calibración.

2.1.1. Parámetros extrínsecos

Los parámetros extrínsecos son los que permiten definir la ubicación y orientación de la cámara respecto a una referencia conocida. En general son definidos como cualquier conjunto de parámetros geométricos que identifican únicamente la relación entre las coordenadas reales de un punto y las de la cámara (figura 2.3).

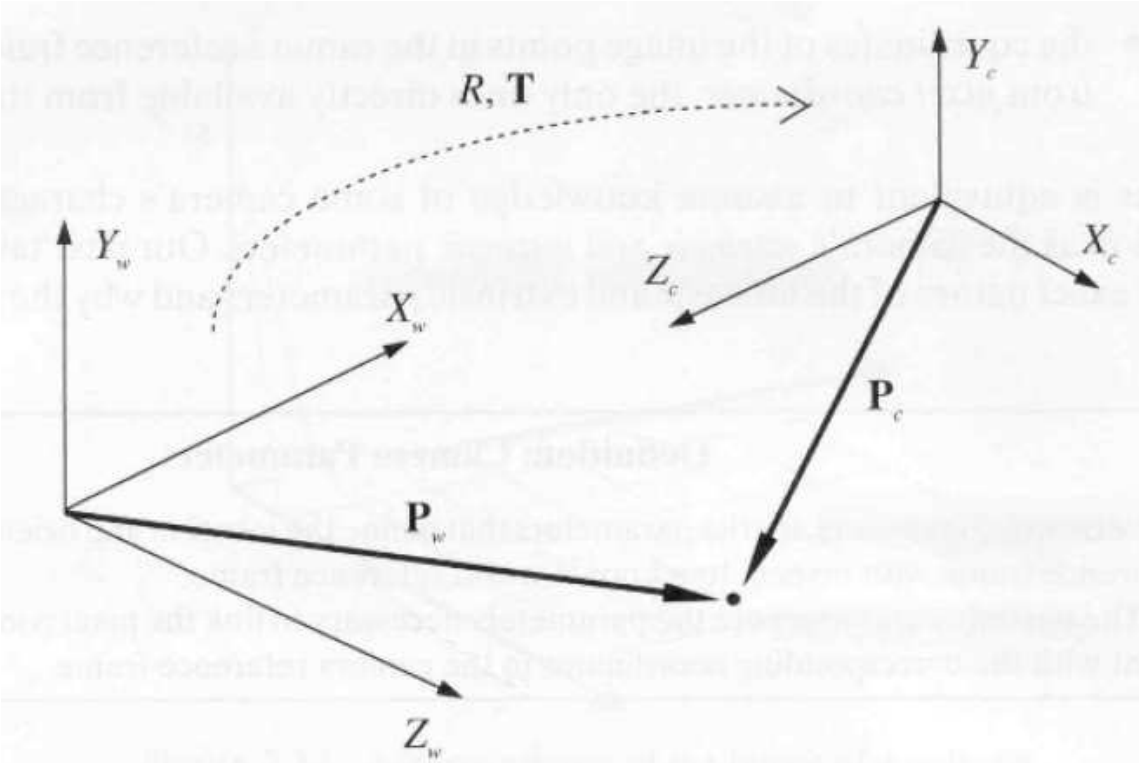


Figura 2.3: Proyección de las coordenadas del punto reales (P_w) y las coordenadas de la cámara (P_c).

$$P_c = R(P_w - T) \text{ donde } R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad (2.1)$$

$$\text{Si } P_c = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \text{ y } P_w = \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix}$$

entonces

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_w - T_x \\ Y_w - T_y \\ Z_w - T_z \end{bmatrix}$$

$$o$$

$$\begin{aligned} X_c &= R_1^T(P_w - T) \\ Y_c &= R_2^T(P_w - T) \\ Z_c &= R_3^T(P_w - T) \end{aligned} \tag{2.2}$$

Donde ${R^T}_i$ corresponde al iésimo renglón de la matriz de rotación

2.1.2. Parámetros intrínsecos

Los parámetros intrínsecos de una cámara son aquellos datos necesarios para relacionar las coordenadas de un píxel con sus correspondientes coordenadas en el cuadro de referencia de la cámara. Los parámetros intrínsecos de la cámara son datos necesarios para carecterizar los aspectos ópticos, geométricos y digitales de la cámara. Estos parámetros están definidos como:

- Distancia focal f.
- Coordenadas del píxel central (o_x, o_y).
- Tamaño efectivo del píxel en dirección vertical y horizontal en milímetros (s_x, s_y).
- Coeficiente de distorsión radial (k_1, k_2).

Aplicando la proyección de la perspectiva tenemos la ecuación 2.3, de manera visual tenemos la figura 2.4.

$$x = f\frac{X_c}{Y_c} = f\frac{R_1^T(P_w - T)}{R_3^T(P_w - T)}, y = f\frac{Y_c}{Y_c} = f\frac{R_2^T(P_w - T)}{R_3^T(P_w - T)} \tag{2.3}$$

$$\begin{aligned} x &= -(x_{im} - o_x)s_x \quad \text{o bien} \quad x_{im} = -x/s_x + o_x \\ y &= -(y_{im} - o_y)s_y \quad \text{o bien} \quad y_{im} = -y/s_y + o_y \end{aligned} \tag{2.4}$$

En forma de matriz tenemos

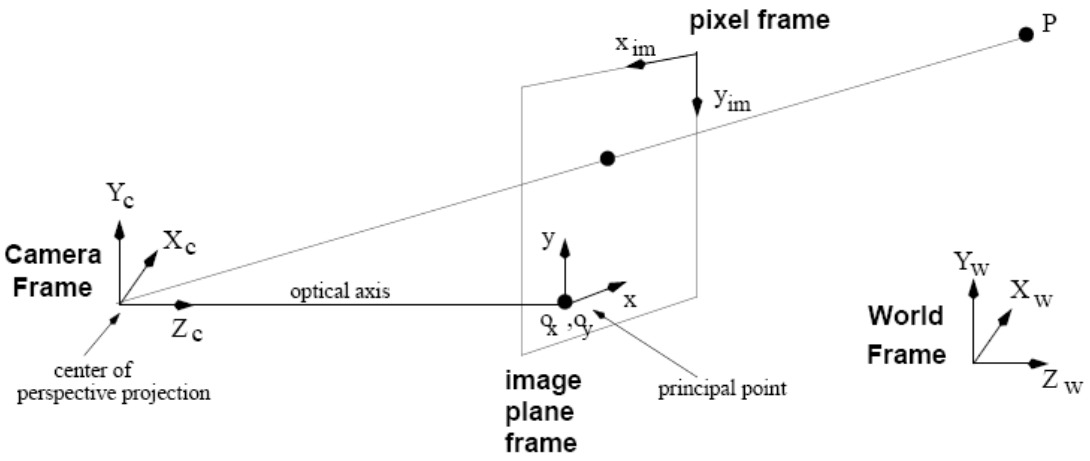


Figura 2.4: De coordenadas de la imagen a coordenadas del píxel (P_c).

$$\begin{bmatrix} x_{im} \\ y_{im} \\ 1 \end{bmatrix} = \begin{bmatrix} -1/s_x & 0 & o_x \\ 0 & -1/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.5}$$

Relacionando las coordenadas de los píxeles con las coordenadas reales:

$$-(x_{im} - o_x)s_x = f\frac{R_1^T(P_w - T)}{R_3^T(P_w - T)}, \quad -(y_{im} - o_y)s_y = f\frac{R_2^T(P_w - T)}{R_3^T(P_w - T)}$$

o también (2.6)

$$x_{im} = -f\frac{R_1^T(P_w - T)}{s_x R_3^T(P_w - T)} + o_x, \qquad y_{im} = -f\frac{R_2^T(P_w - T)}{s_y R_3^T(P_w - T)} + o_y$$

Considerando la distorisión radial de los lentes tenemos:

$$\begin{aligned} x &= x_d(1 + k_1 r^2 + k_2 r^4) \\ y &= y_d(1 + k_1 r^2 + k_2 r^4) \end{aligned} \tag{2.7}$$

Donde (x_d, y_d) son las coordenadas de los puntos distorsionados ($r^2 = x_d^2 + y_d^2$) y k_1 y k_2 son la distorsión radial.

Para combinar los parámetros intrínsecos con los extrínsecos tenemos primero la matriz de parámetros intrínsecos

$$M_{in} = \begin{bmatrix} -1/s_x & 0 & o_x \\ 0 & -1/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

y la de parámetros extrínsecos

$$M_{ex} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & -R_1^T T \\ r_{21} & r_{22} & r_{23} & -R_2^T T \\ r_{31} & r_{32} & r_{33} & -R_3^T T \end{bmatrix} \tag{2.9}$$

usando coordenadas homogeneas tenemos

$$\begin{bmatrix} x_h \\ y_h \\ w \end{bmatrix} = M_{in} M_{ex} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = M \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.10}$$

La homogenización es necesaria para poder obtener las coordenadas de los píxeles

$$\begin{aligned} x_{im} &= \frac{x_h}{w} = \frac{m_{11}X_w + m_{12}Y_w + m_{13}Z_w + m_{14}}{m_{31}X_w + m_{32}Y_w + m_{33}Z_w + m_{34}} \\ y_{im} &= \frac{y_h}{w} = \frac{m_{21}X_w + m_{22}Y_w + m_{23}Z_w + m_{24}}{m_{31}X_w + m_{32}Y_w + m_{33}Z_w + m_{34}} \end{aligned} \tag{2.11}$$

M es también llamada matriz de proyección. Con esto la relación de los puntos en 3D y sus proyecciones en 2D pueden verse como una transformación lineal del espacio proyectivo $(X_w, Y_w, Z_w, 1)^T$ al plano proyectivo $(x_h, y_h, w)^T$

2.1.3. Calibración

Considerando la relación de los parámetros intrínsecos y extrínsecos con la proyección de los puntos del espacio real en la imagen, resulta necesario conocer estos valores.

Existen varios métodos de calibración de parámetros, por ejemplo Zhang (1999) propone un algoritmo de calibración flexible para obtener parámetros distorsión radial del lente, a partir de cuando menos dos imágenes de perspectivas distintas de un patrón planar. Sin embargo esta calibración solo es para una cámara individual. Fusiello et al. (2000) proponen un algorimo lineal y general para sistemas de cámaras estéreo que en base a la proyección de un plano en la imagen de ambas cámaras puede estimar todos los parámetros del sistema de cámaras y además rectificar las imágenes para obtener una línea epipolar colineal para ambas imágenes. He and Li (2008) proponen una calibración que no requiere conocimiento del sistema de cámaras, este algoritmo utiliza los contornos de los objetos para estimar la matriz de rotación y los puntos que desaparecen en las imágenes para obtener la matriz de parámetros intrínsecos. Menudet et al. (2008) proponen un método basado en homografías de un escenario plano desconocido desde diferentes perspectivas, con una simplificación de la geometria a través de una serie de restricciones derivadas de la métrica de rectificación en términos de parámetros intrínsecos y de orientación respecto al plano.

2.2. Rectificación de las imágenes

Dado un par de imágenes estéreo, la rectificación determina una trasnformación de cada imagen tal que pares de líneas epipolares se vuelvan colineales y paralelas a uno de los ejes de la imagen (figura 1.3 y figura 2.5), usualmente el eje horizontal. Si conocemos los parámetros intrínsecos y extrínsecos. Asumimos que el origen del cuadro de la imagen de referencia es el punto principal y que la distancia focal es igual a f. Con esto, hay que seguir los siguientes cuatro pasos:

- Rotar la cámara izquierda hasta que el epipolo se vaya al infinito junto con el eje horizontal.
- Aplicar la misma rotación a la cámara derecha para recuperar la geometría original.
- Rotar la imagen derecha en R.
- Ajustar la escala en ambos cuadros de referencia de las cámaras.

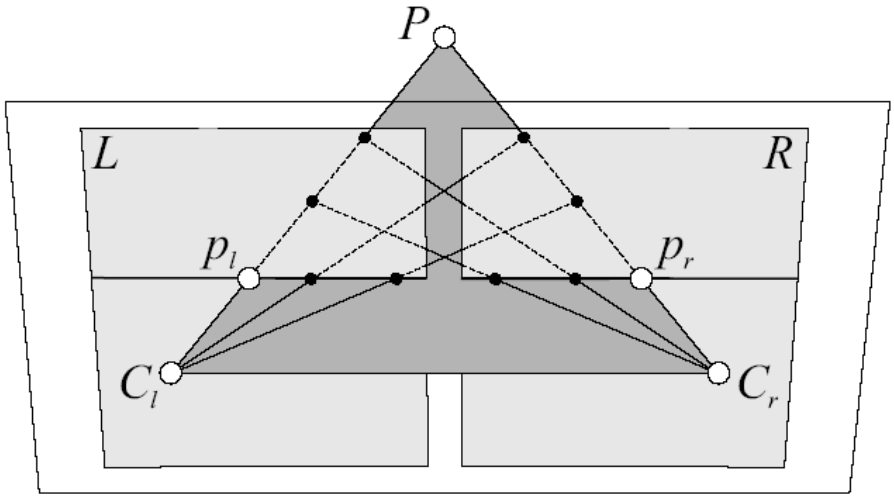


Figura 2.5: Geometría epipolar de dos imágenes estéreo rectificadas.

Se puede ver que son transformaciones lineales por lo que para construir la matriz de rectificación a partir de tres vectores mutuamente ortogonales. El primer vector, e_1, está dado por el epipolo, y ya que el centro de la imagen es el origen, entonces e_1 coincide con la dirección de la traslación, o bien

$$e_1 = \frac{T}{\|T\|} \tag{2.12}$$

Tenemos la condición de que e_2 tiene que ser ortogonal a e_1. Para este propósito se calcula y normaliza el producto cruz de e_1 con el vector de dirección del eje óptico para obtener

$$e_2 = \frac{1}{\sqrt{T_x^2 + T_y^2}} [-T_y, T_x, 0]^\tau \tag{2.13}$$

El tercer vector, e_3 se determina con

$$e_3 = e_1 x e_2 \tag{2.14}$$

Así la matríz ortogonal definida como

$$R_{rect} = \begin{bmatrix} e_1^\tau \\ e_2^\tau \\ e_3^\tau \end{bmatrix} \tag{2.15}$$

hace la rotación de la cámara izquierda alrededor del centro de proyección de tal manera que la línea epipolar se vuelve paralela al eje horizontal, lo que es igual al primer paso del algoritmo descrito arriba.

El algoritmo se puede replantear del siguiente modo:

1. Construir la matriz R_{rect} como en 2.15;
2. Establecer $R_l = R_{rect}$ y $R_r = RR_{rect}$;
3. para cada punto de la imagen izquierda $p_l = [x, y, f]^\tau$ calcular

$$R_l p_l = [x', y', z'] \tag{2.16}$$

   y las coordenadas del punto rectificado correspondiente p'_l como

$$p'_l = \frac{f}{z'} [x', y', z'] \tag{2.17}$$

4. Repetir el paso 3 usando R_r y p_r

Esto nos da los píxeles de las imágenes ya rectificados.

2.3. La transformación RANK

Banks et al. (1999) presentan un trabajo para mejorar la robustez de los algoritmos de empate basados en área. Esta mejora la hacen con dos diferentes transformaciones, la transformación RANK y la trasnformación CENSUS. Para el trabajo presente, solo se consideró la implentación del algoritmo RANK debido a que el número de bits después de

la transformación es menor con RANK (ecuación 2.18) que con CENSUS (ecuación 2.19) para el mismo tamaño de ventana.

$$\begin{aligned} &\text{Número de bits } = \lceil log_2(2 * \lambda + 1)^2 \rceil \\ &\text{donde } \lambda \text{ es el tamaño de la ventana} \end{aligned} \tag{2.18}$$

$$\begin{aligned} &\text{Número de bits } = (2 * \lambda + 1)^2 \\ &\text{donde } \lambda \text{ es el tamaño de la ventana} \end{aligned} \tag{2.19}$$

La definición de la transformación rank es

$$\begin{aligned} R_{i,j} &= \sum_{(u,v) \in W} f(u,v) \\ \text{donde } f(u,v) &= \begin{cases} 1, & I(i+u, j+v) < I(i,j) \\ 0, & \text{otro caso} \end{cases} \end{aligned} \tag{2.20}$$

Por lo tanto, la nueva representación del píxel en un número entre 0 y $N-1$ donde N es el número de elementos en la ventana. Esta representación puede requerir menos bits que el original (dependiendo del tamaño de la ventana). Esto puede se útil para salvar memoria al almacenar el resultado de la transformación. En 2.21 hay un ejemplo para una ventana $\lambda = 1$.

$$\begin{pmatrix} 37 & 142 & 12 \\ 95 & 132 & 46 \\ 176 & 69 & 157 \end{pmatrix} = 5 \tag{2.21}$$

2.4. La suma de diferencias absolutas

El SAD es un algoritmo para elejir la ventana con la mayor similitud en las búsquedas por área, de entre todas las ecuaciones esta es la que demanda menos cómputo. A cambio, es la que tiene resultados más pobres. El SAD está definido por la ecuación 2.22. La ecuación 2.23 es otro método de seleción llamado Suma de Diferencias Cuadradas o SSD y la ecuación 2.24 es la definición de la Correlación Cruzada Normalizada NCC. Para

cualquiera de los algoritmos, por cada par de ventanas es necesario aplicar el algoritmo, la ventana con el menor resultado es la seleccionada, siempre y cuando cumpla con estar por debajo de un umbral mínimo para su selección.

$$SAD = \sum_{(u,v)\in W} |I_1(u,v) - I_2(x+u, y+v)| \tag{2.22}$$

donde (x, y) son las coordenadas del centro de la ventana que se desplaza

$$SSD = \sum_{(u,v)\in W} (I_1(u,v) - I_2(x+u, y+v))^2 \tag{2.23}$$

donde (x, y) son las coordenadas del centro de la ventana que se desplaza

$$NCC = \frac{\sum_{(u,v)\in W} I_1(u,v) * I_2(x+u, y+v)}{\sqrt{\sum_{(u,v)\in W} I_1^2(u,v) * \sum_{(u,v)\in W} I_2^2(x+u, y+v)}} \tag{2.24}$$

donde (x, y) son las coordenadas del centro de la ventana que se desplaza

Se puede apreciar que el costo computacional del algoritmo NCC es muy superior a los otros dos por la operación $(x^{1/2})$ y la división que ha que realizar. La diferencia entre el SAD y el SSD solo es una operación x^2, sin embargo, al momento de plantear varias ventanas operando al mismo tiempo ese bloque puede representar un retardo muy superior a la simple $|x - y|$ que tiene un costo muy bajo en comparación.

2.5. Configuración y adquisición del sensor de imágenes CMOS

La interfaz con los sensores de imágenes CMOS (figura 2.6) fue desarrollada en VHDL para el FPGA. Esta interfaz involucra una sección de configuración y una de adquisición

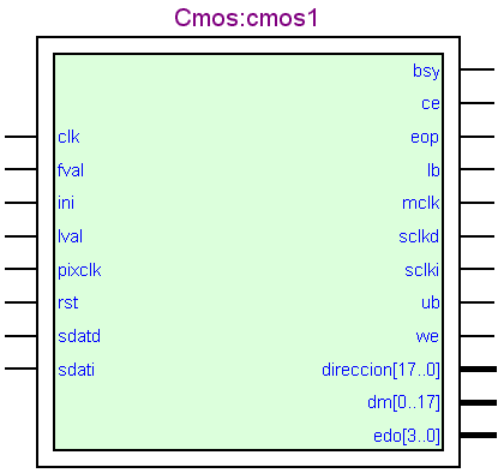


Figura 2.6: Entidad del bloque de configuración y adquisición de los sensores CMOS

(figura 2.7). La sección de configuración establece ganancias de los colores, resolución de la imagen, si el sensor usará uno o dos ADC para la conversión de valores, entre otras cosas. La parte de adquisición consiste en generar las señales de sincronización para el funcionamiento del sensor.

2.5.1. Bloque configuración

Como podemos apreciar en la figura 2.7 hay dos bloques de configuración, esto es para la buena configuración de ambos sensores. La configuración de los sensores CMOS se realiza a través de un protocolo serial (figura 2.8). La señal de datos es bidireccional, este es el motivo por el que se requieren dos bloques separados. Actualmente el bloque de configuración está hecho para enviar el código que establece que la imagen adquirida por el sensor será de 640 x 480. La figura 2.9 muestra la simulación del bloque de configuración estableciendo el tamaño de la imágen.

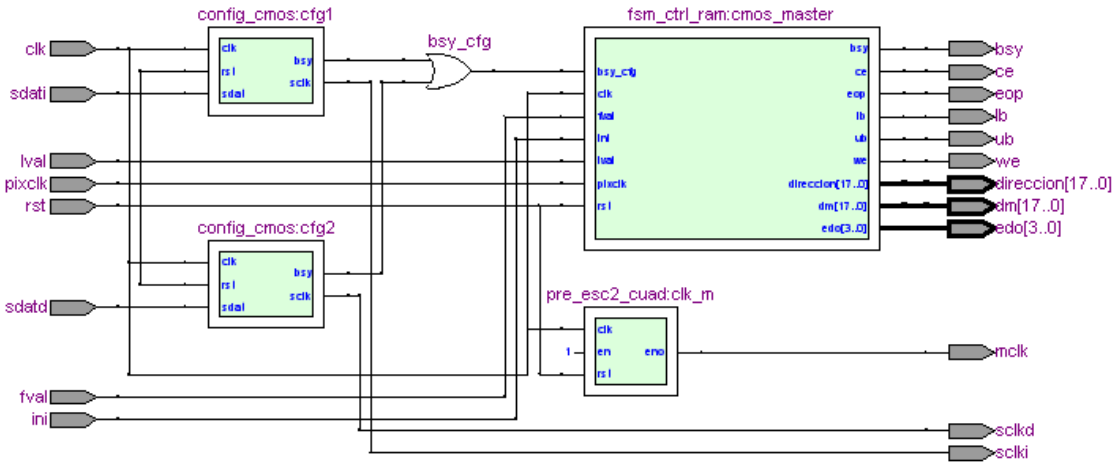


Figura 2.7: Arquitectura del bloque de configuración y adquisición de los sensores CMOS

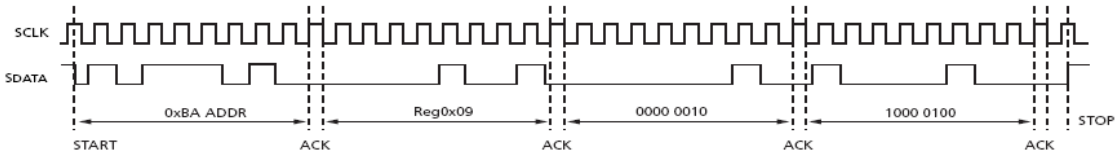


Figura 2.8: Diagrama de tiempos para la configuración del sensor CMOS

2.5.2. Bloque adquisición

El bloque de adquisición se compone del generador de señal de reloj y del bloque de control de ram ambos se aprecian en la figura 2.7. La señal de reloj máxima soportable por el sensor CMOS es de 25 MHz, ya que la frecuencia de trabajo del sistema es 50 MHz, solo requerimos un divisor de frecuencia. El bloque de control de RAM se compone de dos bloques funcionales internos (figura 2.10). Estos bloques, a partir de que la configuración de los sensores ha terminado y se ha solicitado la adquisición de una imagen, se encargan de recibir las señales de control que regresa el sensor CMOS así como el manejo de los datos provenientes de este mismo. Estas señales nos dan la información de cuándo el vector de datos tiene un píxel válido (figura 2.11 y 2.12). Cuando hay un píxel válido

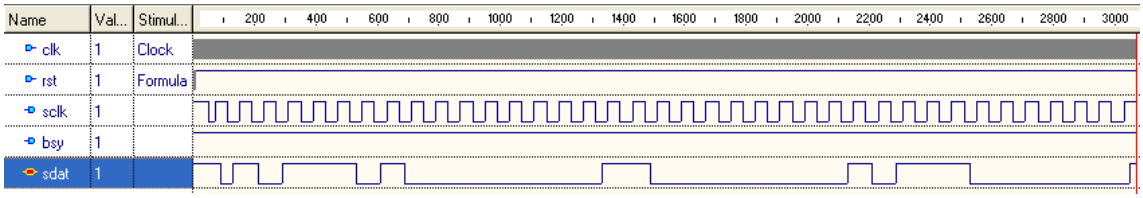


Figura 2.9: Simulación del bloque de adquisición del sensor CMOS

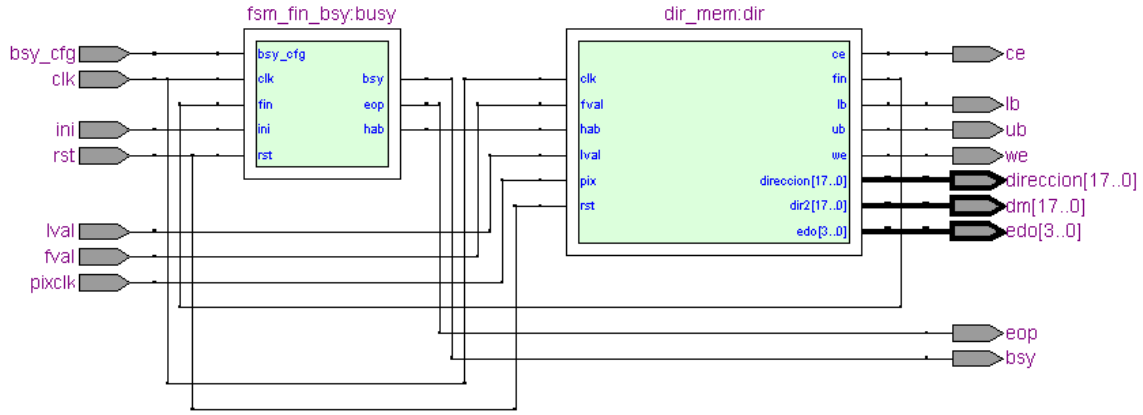


Figura 2.10: Funciones internas del bloque de configuración y adquisición.

este bloque guarda en la memoria ram externa el píxel e incrementa la dirección de la ram. Asi mismo, los bloques se encargan de generar una señal de busy mientras se encuentran adquiriendo una nueva imagen y una señal de fin de imagen cuando han terminado de adquirir.

El sistema para adquirir los pares de imágenes obtenidos por los sensores utiliza los bloques descritos en la sección 2.5 además de una comunicación serial RS-232. El sistema se maneja a través de una PC por medio del seríal. El código *0x23* significa obtener imágen izquierda, el código *0x24* obtener imagen derecha y el *0x25* enviar la imagen a la PC.

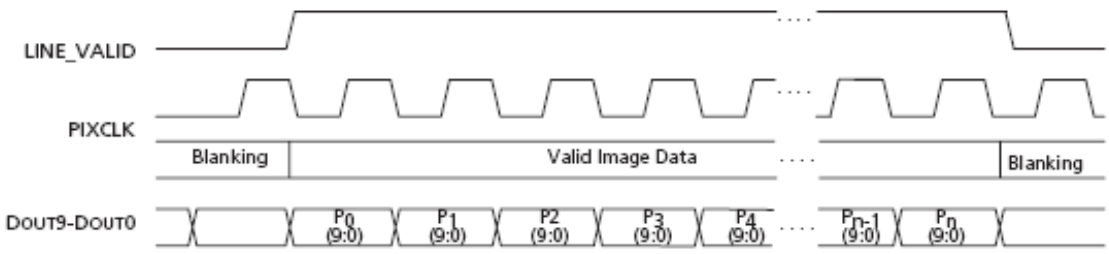


Figura 2.11: Diagrama de tiempos para píxel válido dentro de una línea válida

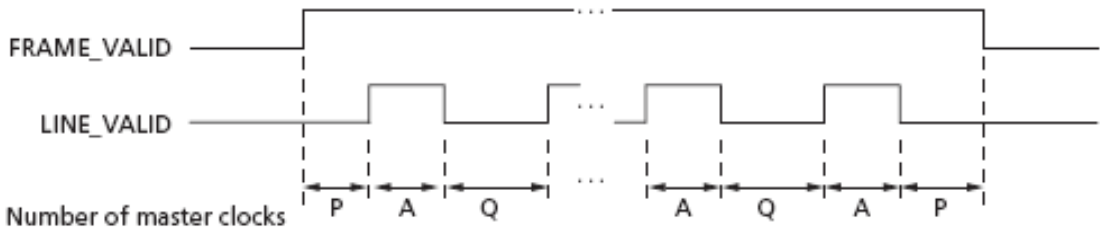


Figura 2.12: Diagrama de tiempos para una línea válida dentro de un frame válido

2.6. Calibración y rectificacion con Matlab

La calibración del sistema de cámaras fue realizada con ayuda del Camera Calibration Toolbox para matlab, desarrollado por Klaus Strobl, Wolfgang Sepp, Stefan Fuchs, Cristian Paredes y Klaus Arbter del Institute of Robotics and Mechatronics. Este toolbox tiene una calibración basada en el modelo de Browns (1966) que incluye la distorsión radial y tangencial de los lentes. Así mismo, considera la distancia focal y la cuadratura del píxel.

2.7. Implementación de la transformación RANK

El bloque para la transformación rank (figura 2.13)trabaja de la siguiente forma. El bloque lector de ventana adquiere una ventana de $\lambda = 2$ de la imagen almacenada en la memoria de entrada y la guarda en el arreglo de registros. Al finalizar la lectura de la

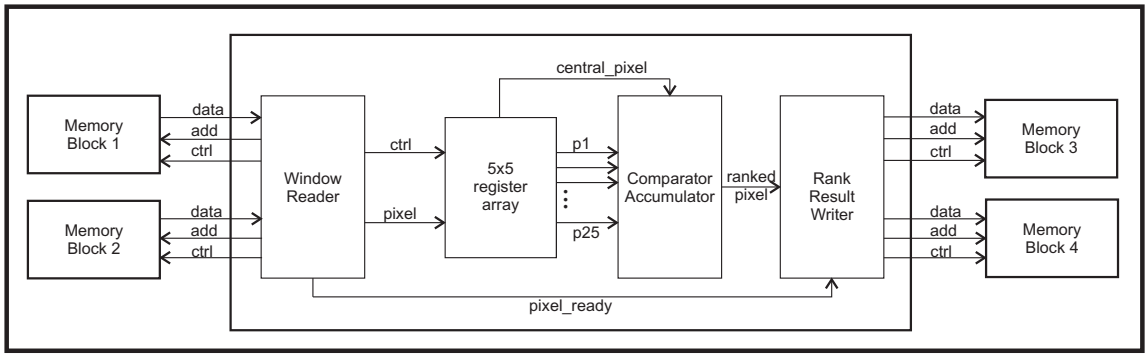


Figura 2.13: Diagrama a bloques de la transformada RANK

ventana también envía una señal para guardar el píxel resultante. El arreglo de registros selecciona el píxel central y pasa éste junto con todos los demás píxeles, de forma paralela al siguiente bloque. El bloque comparador acumulador hace el cálculo del valor rank de esa ventana y envía dicho valor al bloque de escritura de rank.

2.8. Implementación de la suma de diferencias absolutas

En el algoritmo SAD, recordando la ecuación 2.22 tiene solo tres operaciones, una resta, una función de valor absoluto y una suma acumulativa. Las primeras dos operaciones tienen cuando menos dos formas de solución. La primer forma es extendiendo en un bit el tamaño del pixel, esto con el fin de permitir números negativos y después verificar el signo del resultado y obtener el complemento a2. La segunda forma es primero verificar cual de los dós píxeles es mayor y restar al mayor el menor. De esta manera siempre se obtiene un número positivo. La figura 2.14 muestra en bloques la operación. Es esta forma la que se implementó como operación báse para el SAD en ambos casos.

2.8.1. SAD normal

Le llamo SAD normal a la forma natural de la implementación del algoritmo de búsqueda por área con el SAD como medida de similitud. La forma natural es tomar una ventana de alguna de las imágenes y empezar a buscar en la otra imagen una ventana

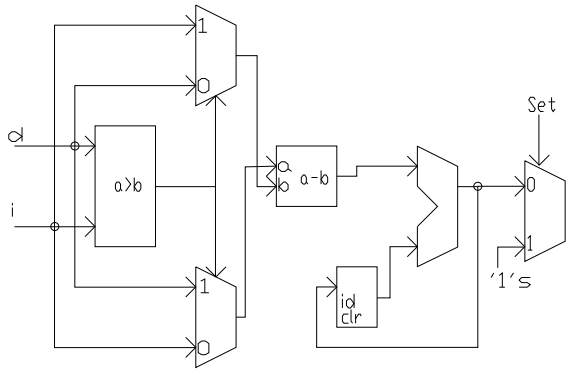


Figura 2.14: Diagrama a bloques de la diferencia absoluta

que sea lo más similar posible. Una vez que se ha terminado esa búsqueda, tomamos una nueva ventana y volvemos a realizar la búsqueda de su equivalente en la otra imagen. Gracias a la geometría epipolar, la calibración y la rectificación esta búsqueda solo implica desplazamientos en el eje horizontal. En hardware esto implica un simple corrimiento de registros, es decir, una vez que hemos leido la primer ventana completa solo requerimos leer una nueva columna en dirección al desplazamiento y tenemos la ventana nueva con la cual comparar. Sin embargo, cuando cambiamos la ventana de referencia, es necesario volver a leer la primer ventana con la que vamos a comparar de manera completa.

En la figura 2.15 tenemos dentro del FPGA un bloque que controla la lectura de la imagen izquierda y uno para la imagen derecha, la forma en que lee las imágenes solo depende de si la ventana de referencia la toma de la imagen derecha o la izquierda. Consideraremos que la ventana de referencia es la de la derecha. Cada pulso de reloj entra un píxel nuevo por cada imagen, estos píxeles son acomodados en el arreglo de registros de $(2 * \lambda + 1)^2$ registros que es donde se va guardando la ventana de compraración y referencia. Cada uno de los registros del arreglo tiene una salida hacia el bloque de la sumatoria de diferencias absolutas. Las columnas de la ventana izquierda son comparadas con las de la derecha

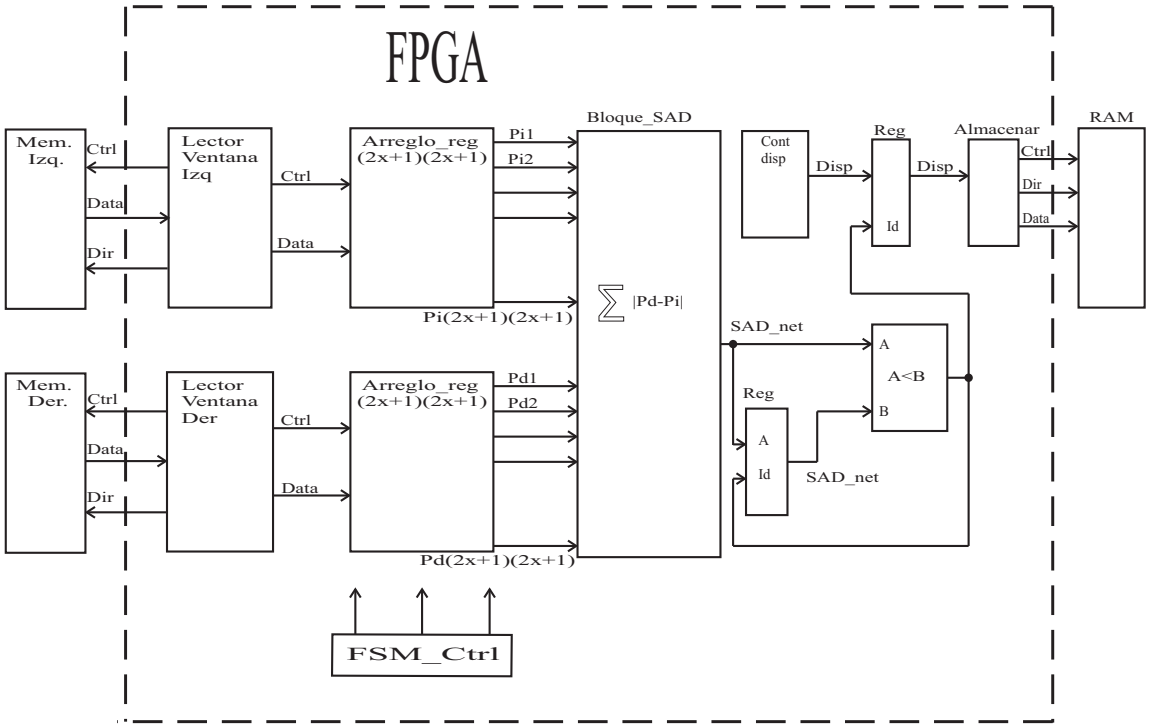


Figura 2.15: Diagrama a bloques del algoritmo SAD

mediante el bloque de diferencias absolutas y los resultados son sumados por un arbol de sumadores. Aunque parecen muchos sumadores que pudieran generar un retardo grande el máximo retardo solo depende de la columna más reciente y de la suma del resultado de esta columna con el resultado de las demás ya que estas columnas tienen como tiempo máximo para su retardo el tiempo que tarda la última columna en llenarse con los nuevos datos.

Una vez que está listo el resultado de la sumatoria se compara este resultado con el valor de la comparación anterior, en caso de ser el nuevo resultado menor, se almacena en el registro para servir de comparación y se almacena en otro registro la disparidad de la ventana que se analizó en ese momento. En caso de ser el resultado mayor simplemente se mantienen los datos que ya estaban almacenados. Al llegar al límite de la disparidad que uno está dispuesto a buscar se almacena el contenido del registro de disparidad en la memoria y el lector de la imagen derecha lee una nueva columna en tanto que el lector de

la imagen izquierda lee toda la primer ventana de comparación y el ciclo se repite.

2.8.2. SAD de implementación mejorada

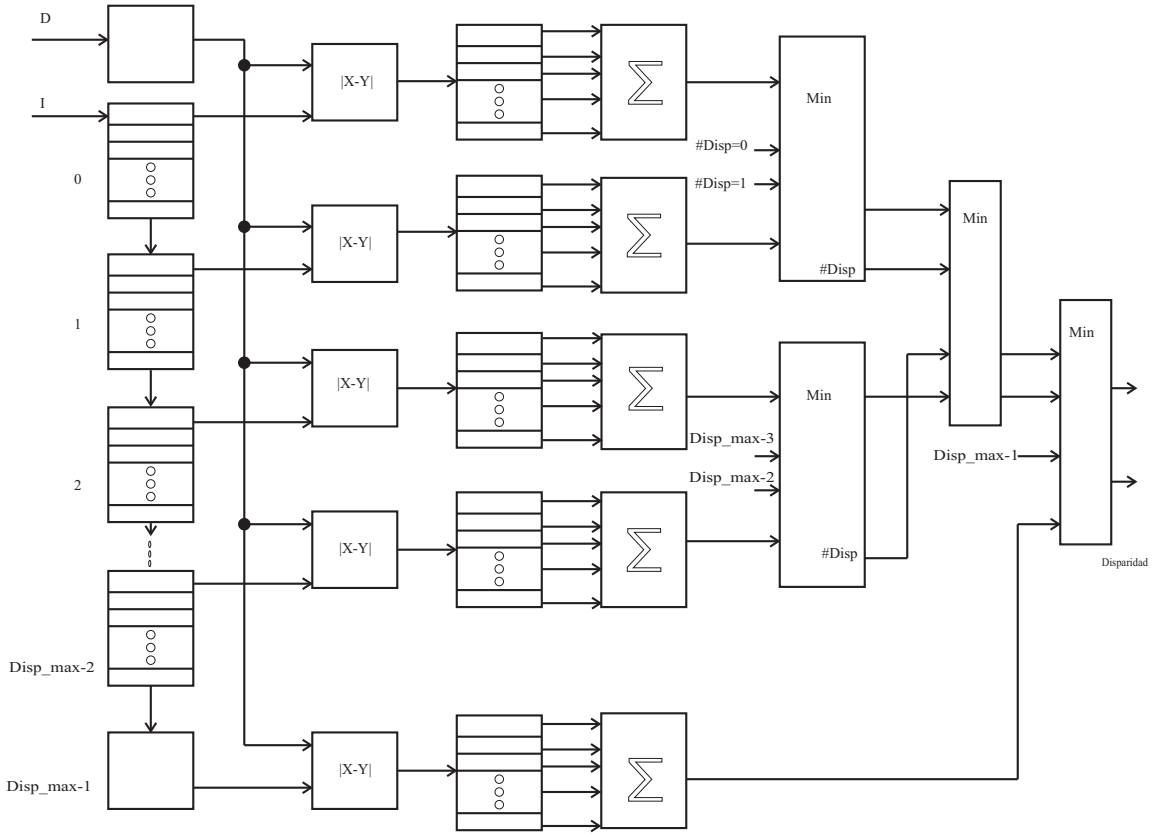


Figura 2.16: Diagrama a bloques del algoritmo SAD optimizado a hardware

Considerando el aumento de recursos como memoria interna, estructuras para operaciones de sumas y multiplicación, y elementos lógicos, uno se puede preguntar si es posible comparar todas las posibles ventanas con la imagen de referencia al mismo tiempo y además no tener que releer todos los píxeles que ya han sido leidos con anterioridad. Si analizamos un poco, en la forma natural del algoritmo es posible que lleguemos a leer una ventana de comparación un número de veces igual a la disparidad máxima que buscamos.

Si pudieramos ahorrar todas esas lecturas a cambio de sacrificar recursos del FPGA y solo tener que leer una vez dichas ventanas tendríamos un ahorro en pulsos de reloj de hasta $disp_{max} * (2 * \lambda + 1)$. Esto es precisamente lo que hace el SAD optimizado a hardware.

2.9. Esquema de pruebas para los bloques

Figura 2.17: Tarjeta de desarrollo DE2 de terASIC

Las pruebas de cada bloque descrito anteriormente se hicieron en una tarjeta DE2 de la empresa terASIC (figura 2.17) con un FPGA Ciclone II 2C35 de 672 pines, 512 KB de RAM, 4MB de memoria flash y un oscilador de 50MHz. Un accesorio para esta tarjeta que consta de dos sensores de imagen CMOS MT9M011 de 1.3 Megapíxeles (figura 2.18). Para las pruebas también fue necesaria una PC con Matlab, Builder C y puerto serial.

Aunque la DE2 ofrece la posibilidad de trabajar con memoria RAM dinámica no se utilizó porque para el tamaño de imágen de 640x480 píxeles solo se requieren 300 KB de espacio. Sin embargo, esto limita un poco la posibilidad de hacer toda la prueba en un solo ciclo. Es decir, hubo la necesidad de implementar cada bloque por separado y no en un

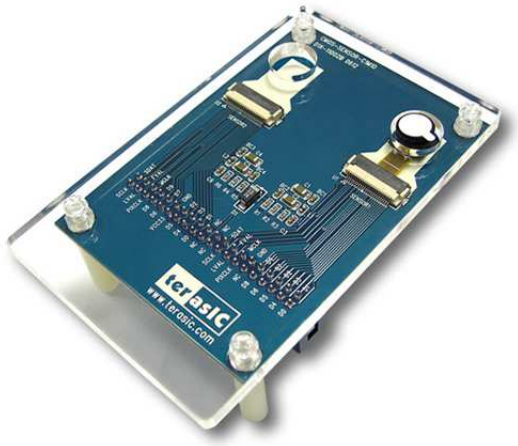


Figura 2.18: Tarjeta periférica con 2 sensores de imagen CMOS (en la imagen solo aparece colocado 1)

solo sistema como hubiera sido lo ideal.

A continuación describo en que consistieron las pruebas de los distintos bloques

2.9.1. Bloque auxiliar

Primero, se diseñó un bloque que almacena en la memoria flash de la DE2 los datos que se envían desde la PC a través del puerto serial (figura 2.19). La PC enviá un comando que la máquina decodificadora interpreta y decide si debe almacenar los siguientes datos o no. El proceso de escritura de la memoria flash es lento, puede llegar a tener retardos de hasta 10 ms, por lo que este bloque tiene que responder que el dato o datos han sido guardados en la memoria flash para que el programa en la PC pueda continuar. Aunque la memoria flash tiene un retardo en el ciclo de escritura de hasta 10 ms, el tiempo de acceso para lectura es de 10 ns, lo que la hace eficiente para lectura.

Este bloque se implementó para almacenar la imagen de entrada del bloque RANK y de los dos bloques SAD. Para el caso particular del bloque SAD, se almacenaban las imágenes izquierda y derecha en la misma memoria flash en la dirección 0x00400 y 0x00800 respec-

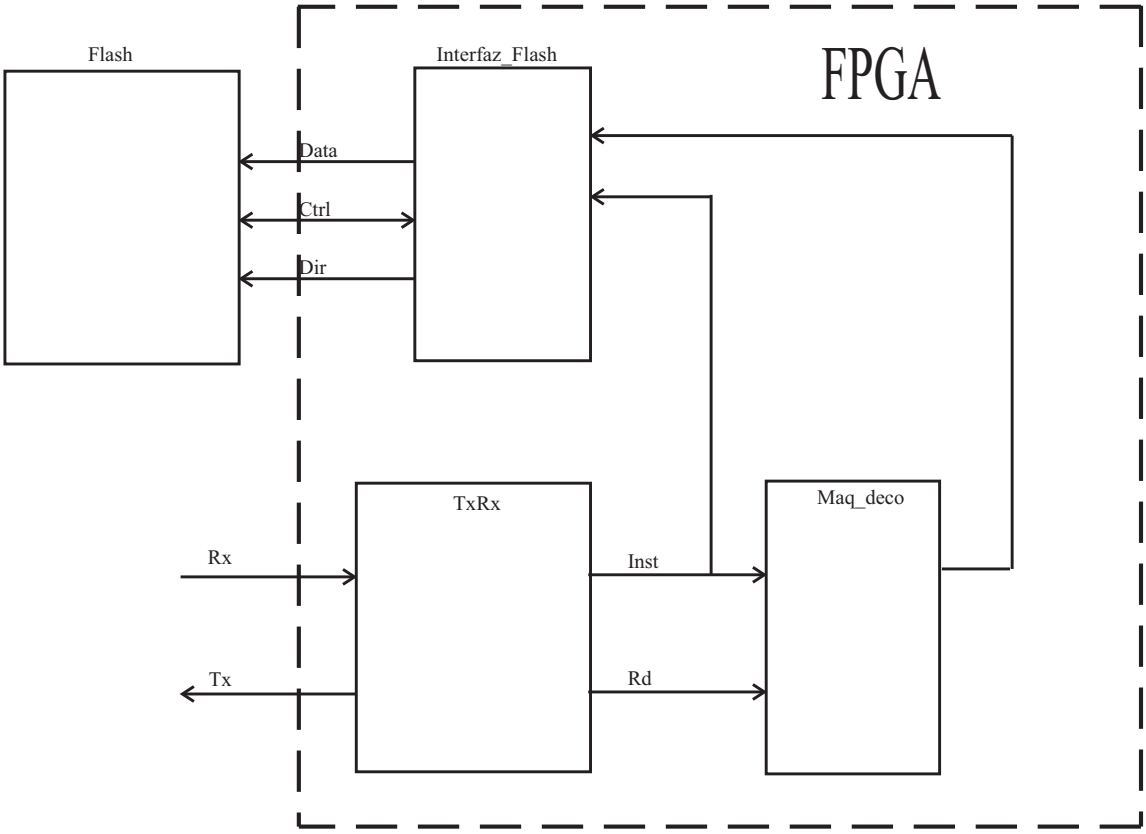


Figura 2.19: Módulo auxiliar para almacenar una imagen en la memoria flash

tivamente.

2.9.2. Prueba configuración y adquisición

El bloque de configuración y adquisición requirió la implementación de la comunicación serial y de una máquina decodificadora para poder recibir instrucciones (figura 2.20). Estas instrucciones consistían en la captura de la imagen captada por alguno de los sensores CMOS, ya que solo se podía tener una imagen en memoria a la vez. En este caso no se consideró la memoria flash como una opción viable debido a los tiempos de retardo y a que el sensor entrega un píxel cada 40 ns, considerando los posibles 10 ms de retardo

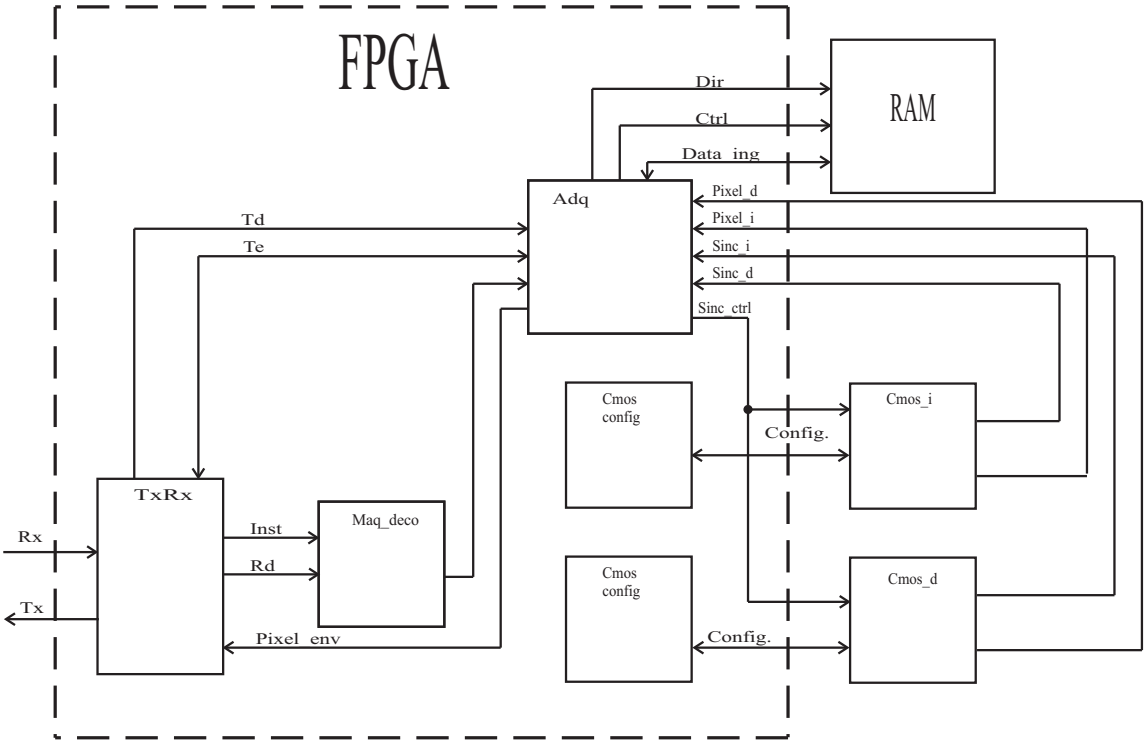


Figura 2.20: Sistema de prueba de la etapa del bloque de configuración y adquisición

de la memoria flash esta queda fuera de las posibilidades.

Cuando el bloque de adquisición ha terminado de almacenar la imagen captada en la memoria indica que ha finalizado la operacion. Con esta señal, la máquina decodificadora inicia el envío de la imagen por el serial a la PC.

2.9.3. Prueba RANK y SAD

Los módulos de pruebas para el bloque RANK y los Bloques SAD son prácticamente idénticos. Una vez que la imagen es almacenada por el bloque auxiliar, el FPGA se reconfigura con la configuración que se ve en las figuras 2.21 y 2.22 que sirve de interfaz con la memoria flash y con el bloque RANK, SAD normal o SAD mejorado. El módulo de

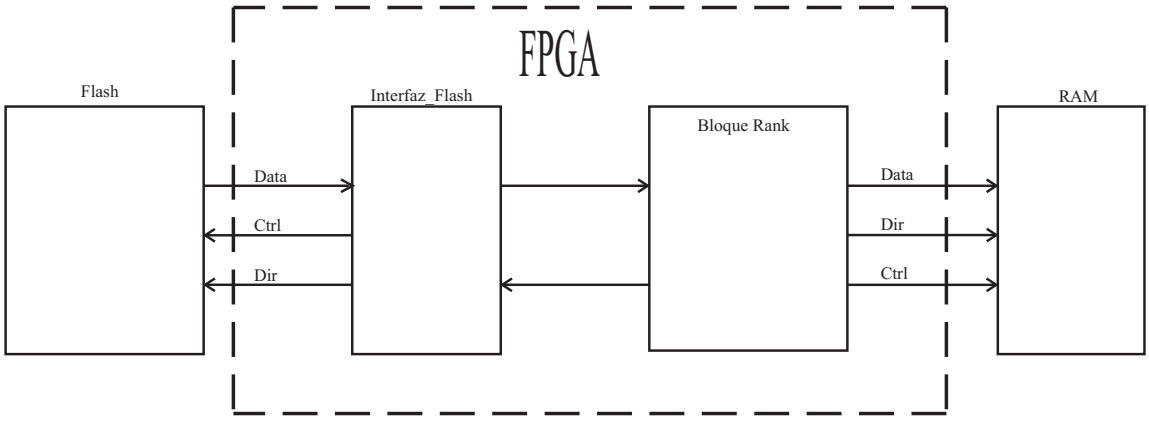


Figura 2.21: Sistema de prueba de la transformación RANK

interfaz con la memoria flash entrega un pixel por pulso de reloj o bien dos píxeles cada dos pulsos de reloj. Esto es porqué el bloque RANK solo requiere una entrada de un pixel, en tanto que los bloques SAD requieren 2 y que entren de manera concurrente.

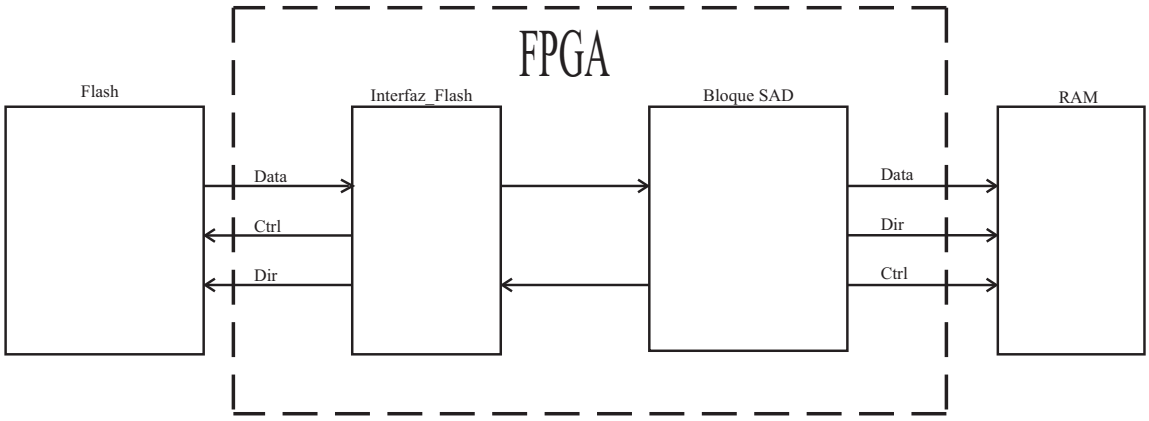


Figura 2.22: Sistema de prueba del algoritmo SAD (normal y optimizado)

Capítulo 3

Resultados y Discusión

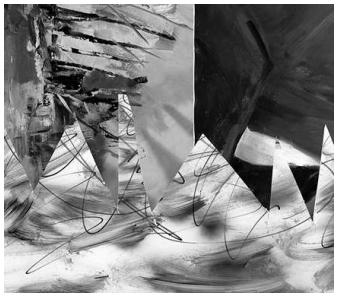
3.1. Calidad del mapa de disparidad

En la figura 3.1 se muestra la imagen sawtooth 3.1(a) junto con las imágenes resultantes después de las distintas etapas del procesamiento (3.1(b) es la imagen sawtooth después de aplicarle la transformación RANK, 3.1(c) es el mapa de disparidad resultante aplicando el algoritmo SAD a la imagen directamente y finalmente 3.1(d) es el mapa de disparidad aplicando el algoritmo SAD a la imagen tratada eno la transformasión RANK). Las imágenes 3.1(b), 3.1(c) y 3.1(d) han sido multiplicadas por un factor de 10 para poder hacer su visualización más sencilla.

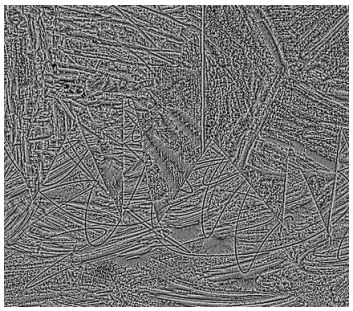
3.2. Tiempo de procesamiento

Para el caso de la adquisición de la imágen la velocidad de adquisición depende de la configuración del sensor de imágenes. Un valor máximo que sirve de referencia es el de 60 cuadros por segundo a una resolución de 640x480 píxeles. El sensor, no soporta señales de sincronización mayores a 25MHz. Sin embargo, esto no resulta ser una limitante debido al resto de los procesamientos que son significativamente más lentos.

Para poder estimar los cuadros por segundo que puede ejecutar la implementación de



(a) Imagen original



(b) RANK

(c) SAD

(d) RANK-SAD

Figura 3.1: Se muestra el resultado de la aplicación del RANK y el SAD, también del SAD sin pasar por la trasnformación RANK.

los algoritmos es necesario conocer los siguientes detalles:

- Retardo combinacional máximo
- Numero total de accesos a memoria (lectura y escritura en caso de usar solo una memoria).
- Tiempo de acceso a memoria.

El primero y último dato nos dan información acerca de la frecuencia de trabajo máxima que podemos usar, si consideramos que por cada pulso de reloj podemos leer uno o

dos datos (de acuerdo a la cantidad de datos de entrada de nuestro diseño) entonces solo necesitamos dividir el la frecuencia de trabajo entre el número total de accesos a memoria.

Las herramientas de síntesis nos dan información respecto al retardo combinacional máximo, y las hojas de especificaciones nos dicen el tiempo de acceso a la memoria en particular que estemos utilizando. En cuanto al número total de accesos a memoria, es necesario realizar un cálculo.

Para calcular el número total de accesos a memoria es necesario conocer el tamaño de la imagen, el tamaño de la ventana (λ) que va a usar el algoritmo y, para el SAD normal, la disparidad máxima que vamos a buscar. Estos valores los establecimos al inicio del trabajo y son, 640x480 para el tamaño de la imagen, tamaño de ventana $\lambda = 2$ para la transformación RANK o $\lambda = 3$ para los SAD y disparidad máxima de 30.

La memoria flash tiene un tiempo de acceso de 10 ns y la memoria RAM tiene un tiempo de escritura de 10 ns igualmente. Estos factores aplican igual para el bloque RANK, SAD y SAD de implementación mejorada. Esto limita la frecuencia de trabajo a una no mayor a los 100 MHz.

El bloque RANK con una ventana de $\lambda = 2$ tiene un retardo combinacional máximo de 13.566 ns. Esto establece una frecuencia máxima de trabajo de 73.7 MHZ. El número total de accesos es igual a $((640 - 2\lambda)x(480 - 2\lambda)x(2\lambda + 1)) = 1513680$. Haciendo la división tenemos 48.68 cuadros por segundo. Este valor es teórico ya que la frecuencia de trabajo que se usó en las pruebas fue de 50 MHz, a esta frecuencia podemos procesar 33.03 cuadros por segundo.

El bloque SAD en su versión normal con una ventana de $\lambda = 3$, tiene un retardo máximo de 15.512 ns. Esto establece una frecuencia máxima de trabajo de 64.46 MHZ. El número total de accesos es igual a $((640 - 2\lambda)x(480 - 2\lambda)x(2\lambda + 1)xdisp_{max}) = 63108360$ para imágenes que no han sido procesadas con al transformación RANK y $((636 - 2\lambda)x(476 - 2\lambda)x(2\lambda + 1)xdisp_{max}) = 62181000$. Haciendo la división tenemos

1.02 y 1.03 cuadros por segundo respectivamente. Igual que para el bloque RANK la frecuencia de trabajo que se usó en las pruebas fue de 50 MHz siendo reducida a 25 MHz para poder emular las dos memorias, a esta frecuencia (50 MHz, en el caso ideal de haber tenido dos memorias separadas) podemos procesar 0.8 cuadros por segundo.

El bloque SAD de implementación mejorada con una ventana de $\lambda = 3$, tiene un retardo máximo de 26.52 ns. Esto establece una frecuencia máxima de trabajo de 64.46 MHZ. El número total de accesos es igual a $((640 - 2\lambda)x(480 - 2\lambda)x(2\lambda + 1)) = 2103612$ para imágenes que no han sido procesadas con al transformación RANK y $((636 - 2\lambda)x(476 - 2\lambda)x(2\lambda + 1)) = 2072700$. Haciendo la división tenemos 11.88 y 12.79 cuadros por segundo respectivamente. Igual que para el bloque RANK la frecuencia de trabajo que se usó en las pruebas fue de 25 MHz, a esta frecuencia podemos procesar 12.06 cuadros por segundo.

3.3. Recursos del FPGA

El FPGA tiene una cantidad limitada de lógica programable que es dónde el diseño va implementado. Es por esto que se vuelve necesario conocer cuánto de esta lógica progamable está siendo utilizada por los diseños, en caso de que querramos agregar alguna etapa interna nueva. También para poder establecer una comparación entre los algoritmos como en el caso del SAD normal y el SAD de implementación mejorada distinta de la velocidad de procesamiento.

Las herramientas de síntesis actuales nos ofrecen información en cuanto a los bloques funcionales que componen el diseño, así como el porcentaje de uso del FPGA y hasta el número de interconexiones que el diseño requiere. Los resultados de uso de recursos se dan por una tabla con el contenido de bloques funcionales y otra tabla de porcentaje de uso para los bloques de configuración y adquisición, RANK, SAD normal y SAD de implementación mejorada.

3.3.1. Bloque de configuración y adquisición

La tabla 3.1 muestra los bloques lógicos que componen el bloque de configuración y adquisición. Podemos apreciar que los bloques funcionales más complejos son los multiplicadores de 8x8-bit y 9x10-bit. La mayor parte de los registros son parte del bloque de configuración. La tabla 3.2 indica que usa el 4 % de los recursos totales del FPGA EP2C35 y solo 383 Flip Flops.

3.3.2. Bloque RANK

La tabla 3.3 muestra los bloques lógicos que componen el bloque RANK. Podemos apreciar que los bloques funcionales más complejos son los sumadores. Los 5 comparadores son porque las comparaciones se hacen de columna en columna. La tabla 3.4 indica que usa el 1 % de los recursos totales del FPGA EP2C35 y solo 269 Flip Flops.

3.3.3. SAD normal

La tabla 3.5 muestra los recursos en términos de estructuras lógicas del bloque SAD normal. Podemos apreciar que los bloques funcionales más complejos son los sumadores. Los 5 comparadores son porque las comparaciones se hacen de columna en columna. La tabla 3.6 indica que usa el 2 % de los recursos totales del FPGA EP2C35 y solo 161 Flip Flops.

3.3.4. SAD optimizado a hardware

La tabla 3.7 muestra los recursos en términos de estructuras lógicas del bloque SAD normal. Podemos apreciar que los bloques funcionales más complejos son los sumadores. Los 5 comparadores son porque las comparaciones se hacen de columna en columna. La tabla 3.8 indica que usa el 2 % de los recursos totales del FPGA EP2C35 y solo 161 Flip Flops.

Cuadro 3.1: Estructuras lógicas necesarias para el bloque de configuración y adquisición

Estructura	cantidad
ROM 8x8-bit	1
Multiplicador 8x8-bit	3
Multiplicador 9x10-bit	2
Sumador 10-bit adder	6
Sumador 11-bit adder	1
Sumador 16-bit adder	2
Sumador 19-bit adder	5
Sumador 22-bit adder	2
Restador 8-bit subtractor	1
Sumador 9-bit adder	7
Sumador Restador 9-bit addsub	2
Contador ascendente 2-bit	1
Contador ascendente 9-bit	1
Registro 1-bit	7
Registro 10-bit	3
Registro 11-bit	1
Registro 19-bit	3
Registro 22-bit	2
Registro 8-bit	19
Registro 9-bit	3
Comparador mayor que 8-bit	8
Comparador menor que 8-bit	8
Multiplexor 1-bit 8 a 1	1
Multiplexor 8-bit 4 a 1	7
Multiplexor 9-bit 4 a 1	2
Buffer triestado 1-bit	1
Buffer triestado 16-bit	1
Buffer triestado 8-bit	1

Cuadro 3.2: Elementos lógicos necesarios para el bloque de configuración y adquisición

Elementos lógicos	cantidad	porcentaje
Totales	1334	4 %
Combinacional	1318	4 %
Registros	383	1 %

Cuadro 3.3: Estructuras lógicas necesarias para el bloque RANK

Estructura	cantidad
sumador 10-bit	1
sumador 19-bit	1
Sumador 3-bit	1
sumador con carry de entrada 3-bit	1
sumador 8-bit	1
sumador/restador 9-bit	1
Flip-Flops	269
comparador menor que 8-bit	5
Xor 1-bit	2

Cuadro 3.4: Elementos lógicos necesarios para el bloque RANK

Elementos lógicos	cantidad	porcentaje
Totales	424	1 %
Combinacional	339	1 %
Registros	269	1 %

Cuadro 3.5: Estructuras lógicas necesarias para el bloque SAD normal

Estructura	cantidad
Sumador 10-bit	2
Sumador 19-bit	3
Sumador 4-bit	1
Restador 4-bit	9
Sumador 8-bit	8
Restador 8-bit	2
Sumador Restador 9-bit	2
Registro 1-bit	8
Registro 10-bit	2
Registro 19-bit	1
Registro 4-bit	19
Registro 8-bit	2
Registro 9-bit	2
Comparador mayor que 4-bit	9
Comparador menor que 8-bit	1
Multiplexor 8-bit 4 a 1	1

Cuadro 3.6: Elementos lógicos necesarios para el bloque SAD normal

Elementos lógicos	cantidad	porcentaje
Totales	516	2 %
Combinacional	516	2 %
Registros	161	1 %

Cuadro 3.7: Estructuras lógicas necesarias para el bloque SAD optimizado

Estructura	cantidad
ROM 32x30-bit	1
Sumador 12-bit	30
Sumador 16-bit	180
Sumador 6-bit	1
Restador 6-bit	1
Restador 8-bit	30
Flip-Flops	4528
Comparador mayor que 16-bit	29
Multiplexor 12-bit 4-to-1	240
Multiplexor 8-bit 4-to-1	205

Cuadro 3.8: Elementos lógicos necesarios para el bloque SAD optimizado

Elementos lógicos	cantidad	porcentaje
Totales	8494	26 %
Combinacional	8493	26 %
Registros	4528	14 %

Capítulo 4

Conclusiones

Los FPGA ofrecen la posibilidad de crear un sistema de adquisición de imágenes provenientes de uno, dos o más sensores de imágenes de acuerdo a las necesidades del proyecto, no están condicionados por una estructura rígida sino por el diseño de la tarjeta de circuito impreso donde se encuentren montados.

La transformación RANK no representa un gasto de recursos y su implementación es sencilla, sin embargo la mejora en el mapa de disparidad que puede aportar esta transformación depende del contenido de la imagen. Mientras mayor textura con un alto contraste tenga la imagen, la mejora es más notoria

La capacidad que tiene el algoritmo SAD para definir si una ventana es mas similar que otra, depende del tamaño de ventana que se use. Sin embargo, el aumentar el tamaño de ventana implica que los objetos o puntos pequeños pueden perderse totalmente y su pisparidad no ser encontrada.

El algoritmo SAD es de una implementación también sencilla y tiene muchas variantes de implementación posible. Dependiendo de los recursos que estemos dispuestos a usar o bien, de la velocidad que estemos dispuestos a tolerar. Aun la forma natural del algoritmo, en hardware resulta ser considerablemente más rápida que su implementación en software.

El empate de imágenes mediante la búsqueda por área con el algoritmo SAD puede alcanzar un desempeño de más de 15 cuadros por segundo bajo las características de una ventana de $\lambda = 3$ y una resolución de 640x480 píxeles. Aunque el estado actual del diseño tiene un desempeño de 11.29 mapas de disparidad por segundo, hay una forma de mejorar ese desempeño a un bajo costo. Considerando que el máximo retardo está dado por la etapa de comparación de la salida de los bloques SAD. Una forma de reducir ese retardo es agregando una o dos etapas de registros para guardar las comparaciones intermedias y poder mientras tanto empezar una nueva ventana. El costo en este caso es que se incrementa en uno o dos pulsos el tiempo de latencia, lo que impactaría en el manejo de los resultados para almacenarlos en la memoria.

Como trabajo futuro hay varias cosas en el corto y mediano plazo. Al corto plazo está la inclusión de memoria dinámica ya sea con un diseño de driver propio o mediante algún IP core comercial y el diseño de una estrategia de manejo de memoria para poder acceder al siguiente nivel en el procesamiento de imágenes. En el mediano plazo esta el trabajo en un método basado en sistemas naturales para el empate de imágenes y los algoritmos de empate por segmentación. Igualmente en el mediano plazo implementar algoritmos de autocalibración. A un largo plazo está el diseñar un sistema autónomo para aplicaciones generales y aplicaciones específicas.

Bibliografía

Andersen, H., Reng, L., and Kirk, K. (2005). Geometric plant properties by relaxed stereo vision using simulated annealing. *Computers and Electronics in Agriculture*.

Banks, J., Bennamoun, M., and Corke, P. (1999). Fast and robust stereo matching algorithms for mining automation. *Digital Signal Processing*.

Bertozzi, M., Broggi, A., Caraffi, C., Rose, M. D., Felisa, M., and Vezzonia, G. (2007). Pedestrian detection by means of far-infrared stereo vision. *Computer Vision and Image Understanding*.

Browns, D. (1966). Decentering distortion of lenses. *Photogrammetric Engineering*.

Díaz, J., Ros, E., Carrillo, R., and Prieto, A. (2007a). Real-time system for high-image resolution disparity estimation. *IEEE Transactions of Image Processing*.

Díaz, J., Ros, E., Sabatini, S., Solari, F., and Mota, S. (2007b). A phase-based stereo vision system-on-a-chip. *Biosystems*.

Dorrington, A. A., Cree, M., Payne, A., Conroy, R., and Carnegie, D. (2007). Achieving sub-millimetre precision with a solid-state full-field heterodyning range imaging camera. *Measurement Science and Technology*.

Fischler, M. and Bolles, R. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *ACM*.

Fusiello, A., Trucco, E., and Verri, A. (2000). A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*.

Garrido, M., Sanza, C., Jiménez, M., and Meneses, J. (2005). The rapid prototyping experience of an h.263 video coder onto fpga. *Microprocessors and Microsystems*.

Gennery, D. (1979). Stereo-camera calibration. *Proc. Image Understanding Workshop*.

Hajimowlana, S., Muscedere, R., Jullien, G., and Roberts, J. (1999). An in-camera data stream processing system for defect detection in web inspection tasks. *Real-Time Imaging*.

Hariyama, M., Kobayashi, Y., Sasaki, H., and Kameyama, M. (2005). Fpga implementation of a stereo matching processor based on window-parallel-and-pixel-parallel architecture. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*.

He, B. and Li, Y. (2008). Camera calibration from vanishing points in a vision system. *Optics & Laser Technology*.

Hirota, K., Satoh, S., Sakai, K., Shinohara, T., Ikeda, K., Mishima, K., Yamada, S., Oku, T., ichi Suzuki, J., Furusaka, M., and Shimizu, H. (2006). Development of neutron anger-camera detector based on flatpanel pmt. *Physica B: Condensed Matter*.

Huh, K., Park, J., Hwang, J., and Hong, D. (2008). A stereo vision-based obstacle detection system in vehicles. *Optics and Lasers in Engineering*.

Hussmann, S. and Ho, T. (2003). A high-speed subpixel edge detector implementation inside a fpga. *Real-Time Imaging*.

Karabernou, S. and Terranti, F. (2005). Real-time fpga implementation of hough transform using gradient and cordic algorithm. *Image and Vision Computing*.

Kumara, A., Sarkarb, S., and Agarwal, R. (2007). A novel algorithm and fpga based adaptable architecture for correcting sensor non-uniformities in infrared system. *Microprocessors and Microsystems*.

Menudet, J., Becker, J., Fournel, T., and Mennessier, C. (2008). Plane-based camera self-calibration by metric rectification of images. *Image and Vision Computing*.

Muñoz-Salinas, R., Aguirre, E., and García-Silvente, M. (2007). People detection and tracking using stereo vision and color. *Image and Vision Computing*.

Nelson, A. (2000). Implementation of image processing algorithms on fpga hardware. Master's thesis, Arizona State University.

Oram, D. (2001). Projective reconstruction and metric models from uncalibrated video sequences. *unknown*.

Ren, Y., Zhu, J., Yang, X., and Ye, S. (2006). The application of virtex-ii pro fpga in high-speed image processing technology of robot vision sensor. *Journal of Physics: Conference Series*.

Rovira-Más, F., Zhang, Q., and Reid, J. F. (2008). Stereo vision three-dimensional terrain maps for precision agriculture. *Computers and Electronics in Agriculture*.

Sun, C., Berman, M., Coward, D., and Osborne, B. (2007). Thickness measurement and crease detection of wheat grains using stereo vision. *Pattern Recognition Letters*.

Sun, C., Jones, R., Talbot, H., Wu, X., Cheong, K., Beare, R., Buckley, M., and Berman, M. (2006). Measuring the distance of vegetation from powerlines using stereo vision. *ISPRS Journal of Photogrammetry and Remote Sensing*.

Szappanos, A., Kocsisa, G., Molnár, A., Sárkozia, J., and Zoletnik, S. (2008). Event detection intelligent camera development. *Fusion Engineering and Design*.

Tian, X., Deng, H., Fujishima, M., and Yamazaki, K. (2007). Quick 3d modeling of machining environment by means of on-machine stereo vision with digital decomposition. *CIRP Annals - Manufacturing Technology*.

Trucco, E. and Verri, A. (1988). *Introductory Techniques for 3-D Computer Vision*. Prentice Hall.

Woo, W. (1998). Rate-distortion based dependent coding for stereo images and video: Disparity estimation and dependent bit allocation.

Woodfill, J. and Herzen, B. V. (1997). Real-time stereo vision on the parts reconfigurable computer. *Proceedings IEEE Symposium on Field-Programmable Custom Computing Machines*.

Xie, H., Hicks, N., Keller, G., Huang, H., and Kreinovich, V. (2003). An idl/envi implementation of the fft-based algorithm for automatic image registration. *Computers & Geosciences*.

Zhang, Z. (1999). A flexible new technique for camera calibration. Technical report, Microsoft Research, Microsoft Corporation.