PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA

CONTEO DE PERSONAS EN TIEMPO REAL USANDO VISIÓN ESTÉREO DE LÍNEA BASE AMPLIA Y GPGPU

HANS-ALBERT LÖBEL DÍAZ

Tesis presentada a la Dirección de Investigación y Postgrado
como parte de los requisitos para optar al grado de
Magister en Ciencias de la Ingeniería

Profesor Supervisor:
DOMINGO MERY

Santiago de Chile, Septiembre 2009

PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA

CONTEO DE PERSONAS EN TIEMPO REAL USANDO VISIÓN ESTÉREO DE LÍNEA BASE AMPLIA Y GPGPU

HANS-ALBERT LÖBEL DÍAZ

Miembros del Comité:
DOMINGO MERY
MIGUEL TORRES
PABLO ZEGERS
JOSÉ E. FERNÁNDEZ

Tesis presentada a la Dirección de Investigación y Postgrado
como parte de los requisitos para optar al grado de
Magister en Ciencias de la Ingeniería

Santiago de Chile, Septiembre 2009

A Carole y a mi familia

AGRADECIMIENTOS

Agradezco a todas las personas que me apoyaron y ayudaron, directa e indirectamente, en el desarrollo de esta tesis. Su ayuda, fuerza, confianza e ideas fueron las que me dieron día a día las herramientas para poder llegar a buen puerto en este proyecto. A todos ellos les agradezco sinceramente.

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

RESUMEN

El conteo automático de personas es un problema fundamental que debe ser abordado en muchos escenarios, como sistemas de control de operaciones inteligentes para el transporte público o de vigilancia automática, donde estadísticas precisas del flujo de personas son necesarias para funcionar correctamente. En la mayoría de los trabajos se han utilizado técnicas de visión monocular, logrando buen rendimiento en entornos controlados. Sin embargo, en escenarios complejos, estas pueden fallar debido a oclusión, iluminación u otros problemas relacionados. Esta situación podría solucionarse usando más fuentes de información visual, pero el aumento en el tiempo de cómputo debido al procesamiento adicional es demasiado alto para producir un sistema de conteo en tiempo real. A fin de evaluar esta alternativa, se desarrolló un sistema de visión estéreo de línea base amplia no calibrado de 2 vistas, que cuenta la gente entrando o saliendo de un sector predefinido. Además, el sistema trabaja en tiempo real gracias a una implementación paralela en GPU. Este documento detalla todos los aspectos del sistema, incluyendo las técnicas utilizadas para calcular automáticamente la geometría epipolar del sistema, solucionar oclusiones y combinar la información de las vistas para realizar el conteo; y el detector de personas, desarrollado utilizando el entorno CUDA de NVIDIA para GPU. El sistema fue probado extensivamente con cuatro secuencias de vídeo bifocales que totalizan 35 minutos y 98 personas. Los resultados confirman la validez del enfoque, logrando un aumento en la completitud promedio desde 0,80 a 0,91 con respecto al mismo sistema con una sola vista, mientras que la precisión promedio sube de 0,89 hasta 0,93. Además, el uso de GPU permite al sistema trabajar a 20 cuadros por segundo. Así, es posible inferir que el uso de más vistas generará un aumento del rendimiento global, sin un crecimiento excesivo del tiempo de ejecución.

Palabras Claves: Visión estéreo, conteo automático de personas, GPGPU.

ABSTRACT

Automatic people counting is a fundamental mid-level problem that needs to be addressed in many scenarios, like intelligent operation control systems for public transportation or automatic surveillance systems, which need accurate people flow statistics to work properly. In most works, monocular techniques have been used, achieving good performance in controlled environments. However, in complex scenarios, these techniques may fail due to occlusion, illumination or other related problems. This situation could be improved by using more sources of visual information, but the increase in computation time due to the extra processing needed is commonly too high to produce a successful real time counting system. In order to evaluate this alternative, an uncalibrated 2-view system based on wide-baseline stereo vision that counts people entering or exiting a predefined sector was developed. The system achieves real time processing by means of an efficient parallel implementation on a GPU. This work details every aspect of the system, including the techniques used to automatically estimate the epipolar geometry of the system, handle occlusions and match elements between views to perform the counting, and the people detector, developed using the Nvidia CUDA framework for GPUs. The system was extensively tested using four real case video sequences that sum around 35 minutes and 98 persons. The results confirm the validity of the approach as the overall counting recall is increased from 0.80 to 0.91 with respect to the same system using only one view, while the overall precision rises from 0.89 to 0.93. Also, the efficient use of GPGPU hardware allows the system to run at a frequency of 20 frames per second, making it perfectly suited for real time use. Thus it's possible to infer that the use of more views would generate an overall performance increase without excessively rising the running time.

Keywords: Wide-Baseline Stereo, Automatic People Counting, GPGPU.

1. INTRODUCCIÓN

1.1. Descripción del problema

Durante los últimos años, el uso de técnicas de visión por computador en aplicaciones reales se ha incrementado de gran manera. Ejemplos de esto son sistemas de reconocimiento de rostros (Zhao, Chellappa, Phillips, & Rosenfeld, 2003), detección de vehículos y peatones para ayudar a la conducción (Choi, Sung, & Yang, 2007), (G. Ma, Park, Miiller-Schneiders, Ioffe, & Kummert, 2007), conteo automático de personas (Celik, Hanjalic, & Hendriks, 2006), control de calidad (Carrasco, Pizarro, & Mery, 2008) y procesamiento de imágenes médicas (Pan, Gu, & Xu, 2008). Este auge se debe principalmente a dos motivos. El primero es la disminución de precios en los dispositivos de captura de imágenes dado por el gran avance en su tecnología, lo que ha permitido su utilización en casi cualquier lugar y situación. El segundo motivo es la madurez alcanzada por la disciplina, que ha permitido en estos últimos años la creación de novedosos algoritmos basados en los sólidos fundamentos teóricos desarrollados en sus primeros 30 años de vida.

Como se citó anteriormente, el conteo automático de personas es una aplicación donde la visión por computador ha logrado ser usada con éxito. De hecho, según Barandiaran, un sistema de conteo de personas basado en visión por computador es parte integral de un sistema de vigilancia automática (Barandiaran, Murguia, & Boto, 2008). Dentro del problema general de conteo de personas, un subconjunto que genera gran interés debido a su amplia aplicabilidad es el de conteo de personas que entran y salen de lugares pasando por una puerta. Ejemplos de sitios donde se da esta situación son las entradas de metro, supermercados o edificios. En ellos, es de gran interés conocer con exactitud la cantidad de personas que entran y salen, para por ejemplo, manejar dinámicamente la cantidad de personal necesario para atender este flujo, como los cajeros en los supermercados o vendedores en las tiendas. La necesidad de uso de la visión por computador en este problema se da principalmente por la incapacidad de los sistemas tradicionales de conteo de personas al paso, como torniquetes, barras giratorias o haces de luz, de manejar escenarios donde pasa mucha gente al mismo tiempo (T.-H. Chen, Chen, & Chen, 2006). Además, en algunos de estos lugares estos sistemas resultan demasiado invasivos y poco prácticos debido a su inherente aparataje.

Otro escenario donde el uso de un sistema de conteo automático basado en visión por computador sería muy útil es en el transporte público, más específicamente a la entrada e

interior de buses y en los paraderos. La información de flujo en estos lugares podría ser utilizada para mejorar de alguna manera el servicio entregado. Un claro ejemplo de aplicación de esto es el proyecto Conicyt Anillo ADI-32 (`http://www.ing.puc.cl/~its/`). Este proyecto, en el cual participan distintos departamentos de ingeniería de las universidades Católica y de Chile, busca generar un sistema inteligente de control del tránsito del transporte público de Santiago. Para lograr esto, se utilizarán sistemas basados en visión por computador y RFID (Oberli & Landau, 2008) para obtener, en tiempo real, estadísticas sobre el flujo y recorrido de los usuarios del sistema. A medida que los datos son obtenidos, estos son entregados junto con información de la posición de los buses a un modelo dinámico de tránsito previamente desarrollado. Finalmente, en base a los resultados entregados por el modelo se generaran estrategias de manejo vial que permitan mejorar la calidad de servicio del sistema de transporte público.

El trabajo desarrollado en esta tesis tiene como fin la creación de un sistema de conteo automático de personas usando visión por computador. La idea es que el sistema pueda ser utilizado tanto a la entrada de salas, tiendas o edificios como de buses, sin tener que hacer cambios en el algoritmo para cada uno de los escenarios.

1.2. Antecedentes

Debido a su importancia práctica, en el último tiempo se ha generado un gran interés en la investigación y el desarrollo de soluciones para este problema. Este tendencia también se repite en el área industrial, donde ya existen varios productos comerciales de alta precisión, como el PCN-1001 (Eurotech, 2007) o el AVC1 (HellaAglaia, 2007). Estos sistemas publican tasas de éxito del orden del 95% bajo condiciones de trabajo reales, gracias al uso de hardware especializado, como por ejemplo sistemas de visión infrarroja, aunque esto último redunda lamentablemente en un alto precio que imposibilita su instalación masiva.

Los métodos usados comúnmente en los sistemas de conteo de visión por computador se basan en detección de $blobs$ usando sustracción de fondo y detección de bordes (Gardel, Bravo, Jimenez, Lazaro, & Torquemada, 2007) o análisis de movimiento (H. Ma, Lu, & Zhang, 2008), (C.-H. Chen, Chang, Chen, & Wang, 2008), o la detección explicita de las personas usando clasificadores estadísticos como AdaBoost (Viola & Jones, 2001). Con el fin de reducir los efectos de la oclusión parcial, también se han propuesto sistemas basados en visión estéreo (Grammalidis & Strintzis, 2000), pero que generalmente asumen el uso de un grupo de cámaras calibradas. Una vez que la posición de las personas ha sido obtenida,

el conteo es realizado analizando la dirección del movimiento de ellas y su paso a través ciertas zonas previamente designadas como zonas de conteo.

Recientemente, las nuevas herramientas que permiten explotar el poder computacional de las tarjetas de video modernas o GPU (GPU: Graphics Processing Unit) para la computación de propósito general (Owens et al., 2007) han sido usadas con gran éxito en bastantes sistemas de visión por computador (Fung & Mann, 2005; Pan et al., 2008), permitiendo el desarrollo de soluciones antes impracticables debido a su alto costo computacional. Estas GPU son hoy en día comúnmente encontrados en computadores de escritorio y portátiles y su costo es increíblemente bajo tomando en cuenta el poder computacional que ofrecen[1]. En el caso de un sistema de conteo automático basado en visión por computador, estas herramientas podrían permitir mejorar el rendimiento de los algoritmos a un costo de procesamiento prácticamente nulo sin la necesidad de adquirir hardware especializado, permitiendo el uso de estos sistemas de manera más masiva.

1.3. Hipótesis, objetivos y descripción general del sistema

La hipótesis central de este trabajo es que las técnicas de visión estéreo de línea base amplia en conjunción con un algoritmo de detección de personas que aproveche las nuevas herramientas de programación general de GPUs, permiten realizar el conteo de personas con mayor precisión y robustez.

El objetivo principal de este trabajo es desarrollar un sistema automático de conteo de personas no calibrado basado en visión estéreo y GPUs, que requiera una mínima intervención por parte de los operadores para funcionar correctamente. Para poder cumplir con esto el sistema debe ser altamente resistente a los cambios de iluminación y al movimiento de las cámaras.

A parte de estos objetivos expuestos anteriormente, el sistema apunta a mejorar ciertos aspectos los sistemas de visión por computador para entornos no controlados. El primero es el tiempo de ejecución. Con el fin de reducirlo, se utilizó el entorno de desarrollo CUDA de Nvidia (Garland et al., 2008), que permite a los desarrolladores aprovechar la arquitectura paralela de las GPU. Esto permitió desarrollar, dentro de otras cosas, un detector de cabezas que funciona en tiempo real. El segundo aspecto es la oclusión parcial de los objetivos. Para superar este problema tan común, el sistema utiliza un modulo estimador de la geometría

[1]Una GeForce GTX260, correspondiente a la última línea de procesadores de Nvidia, cuesta en la actualidad, aproximadamente US$180.

epipolar entre vistas, basado en detectores de características locales, que son invariantes tanto a los cambios de iluminación como a los de punto de vista.

La metodología de desarrollo que se llevará a cabo es la siguiente:

(i) Se estudiarán las distintas técnicas para establecer correspondencias entre vistas mediante distintos detectores de características locales. Una vez seleccionadas, se desarrollará el módulo estimador de geometría epipolar, utilizando detectores de características locales y técnicas robustas a outliers. Se evaluará el rendimiento de este módulo en una serie de escenarios reales, calculando el error de la estimación al comparar puntos correspondientes en las imágenes.

(ii) Se investigarán las mejores alternativas para el desarrollo de aplicaciones en GPU, con el fin de utilizar el entorno más adecuado para el sistema. Luego, se estudiarán distintas técnicas de detección y seguimiento de personas robustas a cambios de iluminación, para luego ser implementadas en la GPU.

(iii) Utilizando los resultados de la estimación de la geometría epipolar y del seguimiento de personas, se evaluarán y definirán las estrategias que permitirán el manejo de las oclusiones.

(iv) Finalmente, el sistema será probado en la entrada de una sala de clases, que cuenta con ventanas que harán que la iluminación sea afectada por la luz del medio ambiente. Los conteos realizados por el sistema desarrollado serán comparados con conteos realizados por un operador humano, que serán considerados como una medición ideal.

El funcionamiento del sistema a grandes rasgos es el siguiente. Cada imagen se procesa de manera individual tratando de detectar el mayor número de personas utilizando el algoritmo desarrollado. De manera simultánea y en segundo planos, el modulo de geometría epipolar genera estimaciones de ésta, basado en correspondencias detectadas entre las vistas del sistema. Luego, usando técnicas de análisis bifocal, se tratará de establecer las relaciones entre las personas detectadas en cada una de las imágenes individuales, de forma de disminuir el problema de la oclusión. Finalmente, se realizará un seguimiento a cada persona detectada, de forma de poder obtener su dirección de movimiento y así realizar el conteo. La figura 1.1 muestra un diagrama combinado de bloques y flujo del sistema de conteo, donde cada uno de los módulos principales se encuentra representado por su nombre dentro de un rectángulo junto con una división de cada uno de sus módulos internos. La explicación detallada de estos bloques se realizará en los capítulos 3: “Detección

de cabezas basada en GPGPU", 4: "Estimación de la geometría epipolar del sistema" y 5: "Seguimiento y conteo mediante visión estéreo".

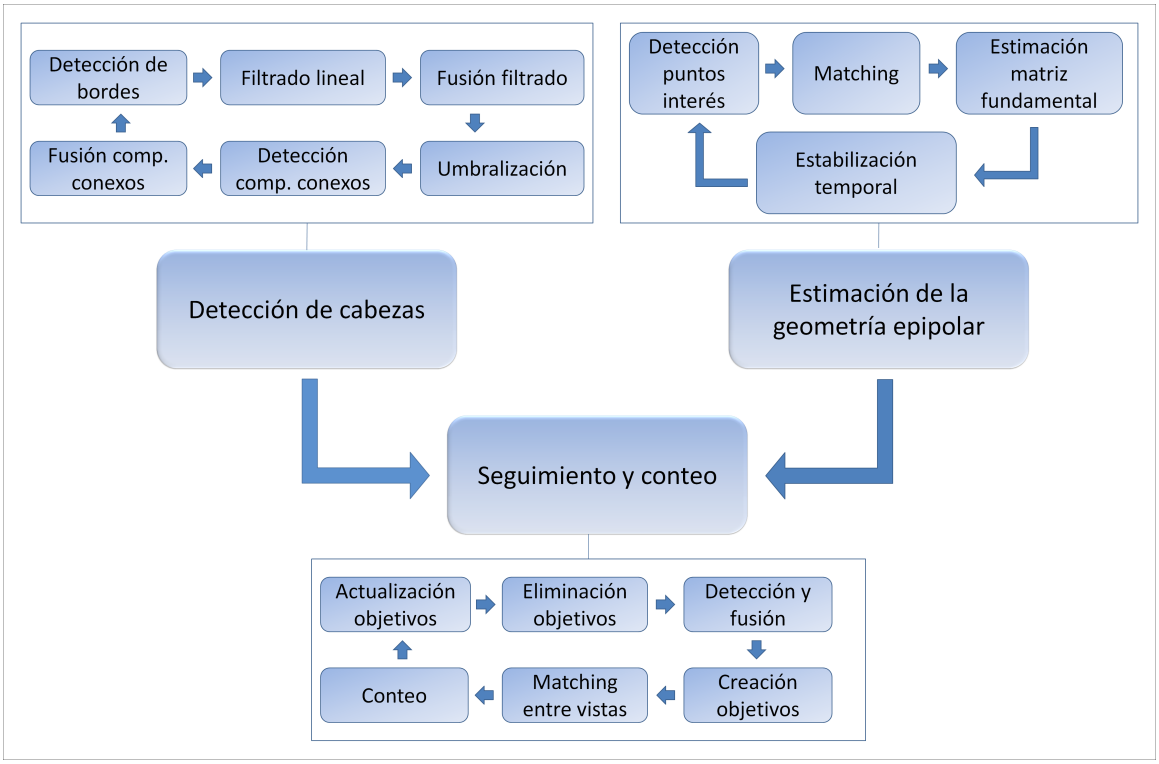


FIGURA 1.1. Diagrama de bloques del sistema de conteo.

1.4. Organización de la tesis

Para comenzar, el capítulo dos presenta las herramientas necesarias para poder comprender el trabajo realizado en esta tesis. En la primera parte se presentan los conceptos de la geometría epipolar para luego pasa al modelamiento geométrico de una cámara. Luego, se introduce la teoría del análisis bifocal seguido de su resultado más importante, la matriz fundamental. En la segunda parte y final se presenta la necesidad de introducir el paralelismo para aumentar el rendimiento y la idea de la GPU para programación de propósito general. Finalmente se presenta el entorno de desarrollo CUDA de Nvidia, que permite utilizar la potencia paralela de las GPU.

Luego, en el capítulo tres se presenta inicialmente una descripción general del sistema detector de personas. A continuación se revisa en detalle cada uno de los elementos que lo componen y las técnicas utilizadas en su implementación.

El capítulo cuatro presenta en su parte inicial los resultados que permiten la estimación de la matriz fundamental a través de puntos correspondientes. Luego se describe en detalle la obtención de estos puntos y su posterior matching. A continuación se describe la técnica de estimación utilizada para encontrar la matriz fundamental.

El quinto capítulo introduce el algoritmo de seguimiento utilizado para luego detallar las estrategias utilizadas para fusionar la información de las detecciones en las distintas vistas del sistema. Finalmente se describe la técnica de conteo utilizada.

A continuación, en el capítulo seis, se describen las pruebas realizadas a los distintos componentes. Se detallan las condiciones físicas de cada una, como por ejemplo el posicionamiento de las cámaras y la distancia entre ellas, y como se evaluará el rendimiento de cada componente. A continuación se presentan los resultados obtenidos en las pruebas descritas anteriormente.

Finalmente, en el capítulo siete se analizan y comentan los resultados obtenidos en el capítulo anterior para posteriormente cerrar con las posibles líneas de investigación futura.

2. CONCEPTOS Y HERRAMIENTAS BÁSICAS

En este capítulo se presentan y describen los conceptos y herramientas básicas necesarias para comprender el desarrollo del sistema.

2.1. Geometría proyectiva y epipolar

A pesar de que resulte extraño para muchos, los humanos estamos muy acostumbrados a las transformaciones proyectivas. Cuando miramos una fotografía vemos cuadrados que no son cuadrados, o circunferencias que no son perfectamente circulares. La transformación que traspasa estos objetos planos a la fotografía es un ejemplo de una transformación proyectiva. De esta manera, la geometría proyectiva es una de las herramientas fundamentales dentro de la visión por computador. En esta sección se describirán las unidades básicas de la geometría proyectiva y su representación. Luego se introducirán los objetos principales de estudio de la geometría proyectiva, i.e., las transformaciones proyectivas, que permiten modelar la forma en que una imagen es generada a partir de la información del espacio tridimensional. Finalmente se utilizarán todos estos conceptos para introducir la geometría epipolar, que es la herramienta fundamental para trabajar con sistemas de múltiples vistas.

2.1.1. Puntos y líneas

Los puntos y las líneas representan las unidades básicas de la geometría proyectiva. En la geometría euclidiana un punto en un plano es representado comúnmente por un par ordenado en $\Re^2$, (x, y). De esta manera, es posible considerar a $\Re^2$ como un espacio vectorial en el que (x, y) es un vector, con lo que se asocia un punto a un vector.

Una línea recta en el plano (x, y) está representada por su ecuación general:

$$ax + by + c = 0 \tag{2.1}$$

Dado que al variar los valores de a, b y c es posible generar todas las rectas posibles, otra manera de representar una recta es mediante un vector $l = [a\ b\ c]^T$. A partir de esto es posible apreciar que la correspondencia entre líneas rectas y vectores no es uno a uno, ya que $[a\ b\ c]^T$ y $[ka\ kb\ kc]^T$, a pesar de ser distintos, representan la misma línea recta para cualquier valor de k distinto de cero, por lo que son considerados equivalentes. Estos vectores son definidos como vectores homogéneos.

La ecuación (2.1) puede reescribirse utilizando el producto punto entre los vectores $l = [a\ b\ c]^T$ y $p = [x\ y\ 1]^T$ de la siguiente manera:

$$l \cdot p = l^T p = [a\ b\ c][x\ y\ 1]^T = ax + by + c = 0 \tag{2.2}$$

De manera similar a una recta, un punto $p = (x, y)$ puede ser representado por un vector $[x\ y\ 1]^T$. Sin embargo, los vectores $[kx\ ky\ k]^T$, para $k \neq 0$, pueden considerarse también como vectores equivalentes, ya que satisfacen la ecuación $[a\ b\ c]^T[kx\ ky\ k] = k(ax + by + c) = 0$. Esto permite definir la representación de los puntos en $\Re^2$ en coordenadas homogéneas como vectores de tres dimensiones cuyos dos primeros elementos son las coordenadas del punto en el plano y el tercer elemento es 1.

De manera más general, dado un vector homogéneo $v = [v_1\ v_2\ v_3]^T$ que representa un punto en $\Re^2$, las coordenadas euclidianas de este punto están dadas por $(x, y) = (\frac{v_1}{v_3}, \frac{v_2}{v_3})$.

La representación homogénea de puntos y líneas en $\Re^2$ tiene dos propiedades muy útiles en la práctica:

- Las coordenadas homogéneas del punto de intersección p entre dos rectas l y l' se pueden encontrar utilizando el producto cruz entre ellas: $p = l \times l'$.
- La representación homogénea de la recta l que une dos puntos p y p' se puede encontrar, de manera análoga al caso anterior, con el producto cruz entre p y p': $l = p \times p'$.

Es posible además extender fácilmente el concepto de coordenadas homogéneas para puntos en $\Re^n$. Un punto $p = (x_1,\ x_2,\ \ldots,\ x_n)$ cualquiera en $\Re^n$ puede ser representado por un vector $v = [v_1,\ v_2,\ \ldots,\ v_{n+1}]^T$, $v_{n+1} \neq 0$, de $n+1$ componentes, tal que se cumpla que $x_i = \frac{v_i}{v_{n+1}}$.

2.1.2. Transformaciones proyectivas

Una transformación proyectiva o proyectividad es una transformación h dada por $h : P^n \rightarrow P^n$, i.e., para vectores homogéneos de n+1 dimensiones, con la característica de que los hiperplanos son transformados en hiperplanos (rectas en rectas y planos en planos para los casos 2D y 3D respectivamente). Las proyectividades se definen entonces como $h(p) = Hp$, donde H es una matriz no singular de $n + 1 \times n + 1$. Como los resultados en la transformación no son afectados si se cambia H por kH, para $k \neq 0$, la matriz H es una matriz homogénea.

Para el sistema desarrollado en esta tesis, el interés se centra sólo en las transformaciones proyectivas 2D y 3D, ya que estas permiten en conjunción modelar de manera muy precisa la generación de imágenes a partir del mundo tridimensional. La tabla 2.1 muestra un resumen de las transformaciones proyectivas 2D junto con una breve descripción de cada una.

TABLA 2.1. Resumen de las transformaciones proyectivas 2D.

Transformación	Matriz H	Descripción
Euclídea	$\begin{bmatrix} cos(\theta) & sin(\theta) & t_x \\ -sin(\theta) & cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix}$	Produce en los elementos una rotación de θ en sentido anti horario seguido de una traslación en (t_x, t_y). Es usada principalmente para transformar ejes de coordenadas. Invariantes: i) longitud entre puntos, ii) área.
Similitud	$\begin{bmatrix} s \cdot cos(\theta) & s \cdot sin(\theta) & t_x \\ s \cdot -sin(\theta) & s \cdot cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix}$	Produce el mismo efecto que la euclídea, más escalamiento isotrópico del tamaño mediante el valor de s. Invariantes: i) ángulos entre rectas, ii) razón entre dos distancias.
Afín	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$	Distorsiona anisotrópicamente la forma de los objetos mediante los coeficientes a_{ij}, que forman una matriz no ortonormal de 2×2, seguido de una traslación en (t_x, t_y). Invariantes: i) líneas paralelas y ii) razón entre dos áreas.
General	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$	Generaliza las transformaciones anteriores al no incluir restricciones en los coeficientes. Invariantes: i) razón de cruz entre puntos alineados.

Cabe señalar que en la tabla anterior, las invariantes de una transformación en una fila inferior son heredadas por las transformaciones en filas superiores, pero no en el sentido contrario.

Dentro de las transformaciones 3D, sólo interesa la transformación 3D euclídea, ya que ella representa los cambios de coordenadas que pueden sufrir los objetos al pasar de un sistema de coordenadas a otro. Dado un sistema de coordenadas 3D $(X,\ Y,\ Z)$ que ha sufrido una rotación y una traslación, el espacio 3D en el nuevo sistema de coordenadas (X', Y', Z') queda definido en coordenadas homogéneas por una transformación 3D euclídea dada por:

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.3}$$

Donde R es una matriz ortonormal de 3×3 y t es un vector de 3×1 que definen la rotación y traslación del sistema de coordenadas respectivamente y $0^T = [0\ 0\ 0]$. La matriz R, presente en (2.3), define una rotación de los ejes de coordenadas que puede ser descompuesta en rotaciones de cada uno estos tres. Las matrices que representan cada una de estas rotaciones están dadas por R_x, R_y y R_z en la tabla 2.2.

TABLA 2.2. Matrices de rotación en los ejes X, Y y Z.

Eje rotación	Matriz
X	$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_x) & sin(\theta_x) \\ 0 & -sin(\theta_x) & cos(\theta_x) \end{bmatrix}$
Y	$R_y = \begin{bmatrix} cos(\theta_y) & 0 & -sin(\theta_y) \\ 0 & 1 & 0 \\ sin(\theta_y) & 0 & cos(\theta_y) \end{bmatrix}$
Z	$R_z = \begin{bmatrix} cos(\theta_z) & sin(\theta_z) & 0 \\ -sin(\theta_z) & cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Así, la rotación total puede ser definida como primero una rotación del eje Z, luego del eje Y y luego del eje X, lo que se puede expresar como una multiplicación de las tres matrices de rotación: $R(\theta_x, \theta_y, \theta_z) = R_x(\theta_x)R_y(\theta_y)R_z(\theta_z)$.

2.1.3. Modelamiento geométrico de una cámara

En esta sección se introducirá inicialmente el modelo de la cámara pinhole, que es el más simple de todos, para luego relajar ciertas restricciones de este modelo hasta llegar a uno para una cámara CCD, que es el que representa de manera más fiel a las cámaras utilizadas en el desarrollo.

El modelo geométrico de una cámara pinhole consiste en un centro óptico C, que coincide con el origen de un sistema de coordenadas euclidiano, en donde convergen todos los rayos de la proyección y el plano $\mathsf{Z} = f$ que es llamado el plano de imagen y en el cual esta es proyectada. Bajo este modelo, un punto en el espacio con coordenadas X=$(X,\ Y,\ Z)$, es transformado en el punto en el plano de imagen en el cual la línea que une al punto X con el centro óptico intersecta al plano de imagen. Esta situación es presentada en la figura 2.1.

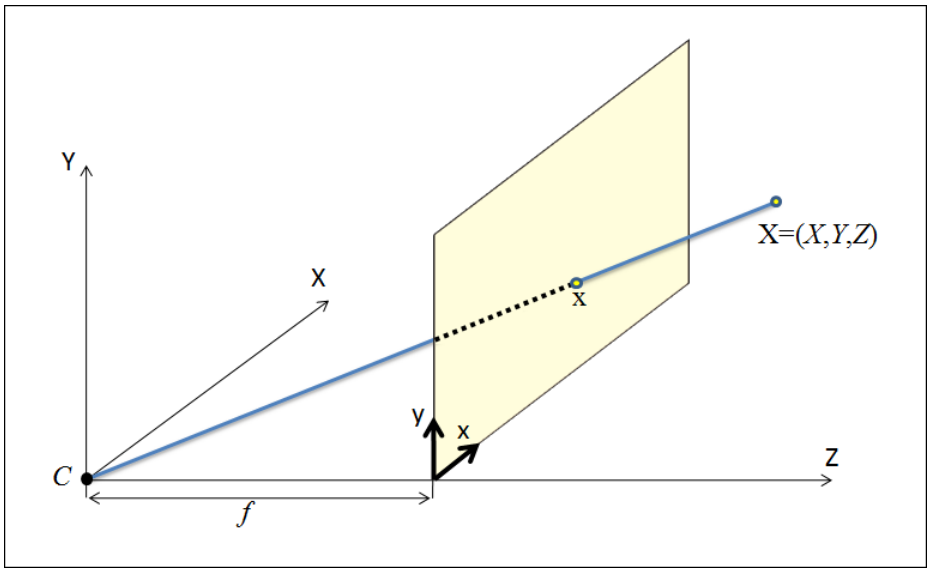


FIGURA 2.1. Modelo geométrico de una cámara pinhole.

Al observar la figura es posible determinar fácilmente, utilizando semejanza de triángulos, que las coordenadas del punto proyectado son $\left(\frac{fX}{Z}, \frac{fX}{Z}\right)$. Teniendo esto en cuenta, la transformación $3D \rightarrow 2D$ entre los puntos puede ser expresada en coordenadas homogéneas de la siguiente manera:

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.4}$$

Nombrando como $\mathbf{X}$ al vector que representa las coordenadas homogéneas del punto en el espacio tridimensional y $\mathbf{x}$ al que representa las del punto proyectado, la expresión anterior puede ser escrita en forma matricial como:

$$\mathbf{x} = P\mathbf{X} \tag{2.5}$$

donde la matriz P de 3×4 es llamada la matriz de proyección de la cámara. Esta expresión constituye el modelo básico de una cámara pinhole. A continuación se relajará la condición de que el origen del sistema de coordenadas del plano de la imagen coincide con la intersección del eje Z con este, que es conocida como el punto principal de la imagen. La relajación consiste en modificar la matriz P para incluir un desplazamiento con respecto al punto principal que represente el nuevo origen del sistema. Sean $(p_x,\ p_y)$ las coordenadas del punto principal de la imagen, luego la nueva transformación se escribe de la siguiente manera:

$$\mathbf{x} = \begin{bmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.6}$$

Una manera conceptualmente más clara de escribir la expresión anterior es definiendo la matriz K de 3×3 como:

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

La matriz K es llamada la matriz de calibración de la cámara y como se puede apreciar corresponde a una transformación de similitud 2D con factor de escala igual a f, traslación $[p_x,\ p_y]^T$ y rotación cero. Con esto, la ecuación (2.6) puede ser reescrita de la siguiente manera:

$$\mathbf{x} = K[I \mid 0]\mathbf{X} \tag{2.8}$$

Donde I es la matriz identidad de 3×3. La siguiente relajación proviene del hecho de que hasta el momento las coordenadas del punto tridimensional corresponden a un sistema en que su origen y orientación corresponden a los de la cámara. Comúnmente el mundo real tendrá un marco de referencia distinto a este, por lo que es necesario eliminar esta restricción. Los dos sistemas de coordenadas pueden ser relacionados mediante una transformación proyectiva euclídea 3D aplicada al vector $\mathbf{X}$, donde los coeficientes de la matriz trasformación reflejan la orientación y posición del sistema de coordenadas del mundo real. Con esto, la ecuación de transferencia $2D \rightarrow 3D$ queda de la siguiente manera:

$$\mathbf{x} = K[I \mid 0] \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \mathbf{X} = K \left[R \mid t\right] \mathbf{X} \tag{2.9}$$

donde $0 = [0\ 0\ 0]^T$. Finalmente, para llegar al modelo de una cámara CCD, es necesario tomar en cuenta dos factores. El primero es que en uno de estos dispositivos, los pixeles no son necesariamente cuadrados, por lo que es necesario introducir en la matriz K factores de escala distintos para cada dirección. El segundo factor es el torcimiento de los pixeles, lo que implica que los ejes x e y de la cámara no son perpendiculares. Para incluir este caso en el modelo es necesario introducir en la matriz K un coeficiente s en la posición $(1, 2)$. Aunque este parámetro tiene valor igual a cero la gran mayoría de las veces, es importante incluirlo para que el modelo sea completo. Con esto, la matriz de calibración para una cámara CCD queda de la siguiente manera:

$$K = \begin{bmatrix} \alpha_x & s & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

2.1.4. Geometría epipolar

Supongamos que se tiene un punto X en el espacio 3D que es visto en dos imágenes distintas. Sean x y $\tilde{x}$ sus proyecciones en la primera y segunda vista respectivamente. Una pregunta muy importante que puede hacerse es cuál es la relación entre los puntos correspondientes x y $\tilde{x}$. La geometría epipolar es comúnmente motivada por esta pregunta o por alguna relacionada con la búsqueda de correspondencias entre vistas distintas del mismo objeto.

Más específicamente, la geometría epipolar entre dos vistas es esencialmente la de la intersección del plano de imagen con el haz de planos que tienen a la línea base como eje. La línea base es la que une los centros ópticos de las dos cámaras. La figura 2.2 presenta una visualización general de la geometría epipolar de dos vistas.

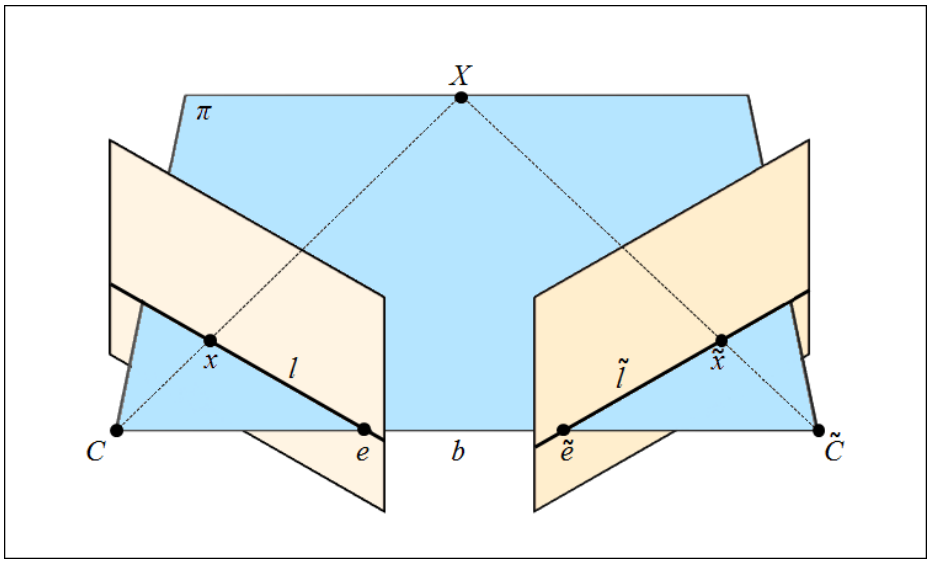


FIGURA 2.2. Geometría epipolar de dos vistas.

Volviendo al caso de los puntos correspondientes, suponiendo que se tienen las coordenadas del punto x en la primera vista, es interesante saber de qué manera están restringidas las coordenadas del punto $\tilde{x}$ en la otra vista. Para responder esta pregunta consideremos el plano π definido por la línea base b y el rayo que une C con X. De la figura anterior sabemos que el rayo entre el punto $\tilde{C}$ y X debe estar contenido en el plano π, ya que este último contiene tanto a X como a los centros ópticos de ambas cámaras. De esta manera, se puede concluir que el punto $\tilde{x}$ se encuentra en la intersección del plano Π con el plano de imagen de la segunda vista. Esta intersección está dada por la línea $\tilde{l}$, llamada línea epipolar de x, que es la proyección en la segunda vista del rayo que une C con X. A partir de esta línea es posible enunciar la llamada restricción epipolar, que señala que para que x y $\tilde{x}$ sean puntos correspondientes, el punto $\tilde{x}$ debe estar en la línea epipolar de x. Cabe destacar que todo este análisis es análogo desde el punto de vista de la segunda imagen, lo que significa que el punto x debe encontrarse sobre la línea epipolar l de $\tilde{x}$.

Desde el punto de vista de la búsqueda de correspondencias, esto representa una reducción considerable en la dimensionalidad del problema, ya que en vez de buscar en toda la imagen se busca sólo a lo largo de una línea. A manera de ejemplo, si la segunda imagen

tiene $N \times N$ pixeles, la búsqueda de correspondencia se realiza sólo en N pixeles de la imagen y no en N^2.

A parte de la línea epipolar y la línea base, existe una tercera entidad geométrica de gran importancia en la geometría epipolar, el epipolo. El epipolo es el punto de intersección de la línea base con el plano de imagen, lo que implica que es la proyección en una vista del centro óptico de la otra. Dado que todas las líneas epipolares son generadas por el haz de planos que tiene como eje a la línea base, todas las líneas epipolares de una imagen deben pasar obligatoriamente por el punto de intersección de la línea base con el plano de imagen, que es justamente el epipolo de esa vista. Esta propiedad es la característica principal del epipolo, la que como se verá más adelante en el capítulo 4, tiene una gran utilidad para restringir los resultados de la estimación de la geometría epipolar.

2.1.5. La matriz fundamental

La matriz fundamental es la representación algebraica de la geometría epipolar. En esta sección se obtendrá una expresión matemática para esta, como resultado de la búsqueda de una expresión algebraica para la línea epipolar. Supongamos que se tienen dos vistas distintas, cada una con sus respectivas matrices de proyección, P y $\tilde{P}$. Supongamos además que se tiene un punto $\mathbf{X}$ del el espacio 3D en coordenadas homogéneas y sus proyecciones en ambas vistas, también en coordenadas homogéneas, $\mathbf{x}$ y $\tilde{\mathbf{x}}$. De esta manera se tiene el siguiente par de ecuaciones:

$$\begin{cases} \mathbf{x} &= P\mathbf{X} \\ \tilde{\mathbf{x}} &= \tilde{P}\mathbf{X} \end{cases} \tag{2.11}$$

Para encontrar la expresión de la línea epipolar de x, $\tilde{l}$, se buscará primero una expresión para el rayo que va entre el centro óptico de la primera vista, C, y el punto X. Las coordenadas homogéneas de C, i.e., $\mathbf{C}$, pueden ser obtenidas a través del espacio nulo de la matriz P, ya que C es el único punto cuya proyección no está definida en la imagen. Para encontrar este rayo, es necesario tener dos puntos pertenecientes a él. Uno es C, mientras que el otro puede obtenerse a través del punto x. A pesar de que la matriz P no es invertible, es posible obtener un punto perteneciente al rayo al usar la matriz pseudo-inversa P^+ de P, definida como:

$$P^+ = P^T[P^T P]^{-1} \tag{2.12}$$

Esta matriz cumple la condición de que $PP^+ = I$, por lo que se puede definir el punto $\mathbf{X}^+$ como:

$$\mathbf{X}^+ = P^+\mathbf{x} \tag{2.13}$$

Este punto pertenece al rayo definido por C y x, ya que su proyección en la primera vista coincide con **x**, al aplicarle la matriz P. Teniendo estos dos puntos, podemos buscar la proyección de cada uno de ellos en la otra vista y luego construir con ellos una expresión para la recta que los une, que corresponde a la línea epipolar. Las proyecciones se obtienen al aplicar la matriz de transformación $\tilde{P}$ tanto a **C** como a $\mathbf{X}^+$:

$$\begin{cases} \tilde{\mathbf{e}} & = \tilde{P}\mathbf{C} \\ \tilde{\mathbf{x}}^+ & = \tilde{P}\mathbf{X}^+ \end{cases} \tag{2.14}$$

Si la línea epipolar $\tilde{l}$ contiene estos dos puntos, se puede decir entonces que su representación homogénea queda definida como:

$$\tilde{l} = \tilde{\mathbf{e}} \times \tilde{\mathbf{x}}^+ = \tilde{P}\mathbf{C} \times \tilde{P}\mathbf{X}^+ = \tilde{P}\mathbf{C} \times \tilde{P}P^+\mathbf{x} = \tilde{P}\mathbf{C} \times \tilde{P}P^T[P^TP]^{-1}\mathbf{x} \tag{2.15}$$

Si en la ecuación (2.15) se define la matriz F de 3×3 elementos como $F = \tilde{P}\mathbf{C} \times \tilde{P}P^T[P^TP]^{-1}$, llamada la matriz fundamental, es posible expresar la línea epipolar $\tilde{l}$ de **x** como $\tilde{l} = F\mathbf{x}$ y más aún, se puede definir la llamada restricción epipolar, el hecho de que $\tilde{\mathbf{x}}$ debe pertenecer a la línea epipolar $\tilde{l}$, de la siguiente manera:

$$\tilde{\mathbf{x}}^T F\mathbf{x} = 0 \tag{2.16}$$

Como se dijo antes, la matriz F es conocida como la matriz fundamental y es de gran importancia para el análisis de dos vistas, ya que F es constante para una geometría bifocal dada, i.e., no depende de **x**, $\tilde{\mathbf{x}}$ ni **X**, sólo de P y $\tilde{P}$. Algunas de las propiedades de la matriz fundamental son las siguientes:

- Las representaciones homogéneas de las líneas epipolares l y $\tilde{l}$ se definen como:

$$\begin{aligned} \tilde{l} &= F\mathbf{x} \\ l &= F^T\tilde{\mathbf{x}} \end{aligned} \tag{2.17}$$

- F es homogénea, ya que kF para $k \neq 0$ también puede ser utilizada en los cálculos anteriores.
- El determinante de F es cero.
- F tiene siete grados de libertad.

Finalmente, como que en la práctica existen errores de medición dados principalmente por la discretización del posicionamiento, dos puntos correspondientes no cumplirán la restricción epipolar, ya que uno no está exactamente sobre la línea epipolar generada por el otro, sino que muy cerca. Así, en la práctica se utiliza una restricción modificada, en la cual se cuantifica la distancia de un punto correspondiente a la línea epipolar generada por el otro. Esta distancia se calcula a partir de una línea perpendicular a la línea epipolar. De esta manera, se obtiene la llamada restricción epipolar práctica:

$$d = \frac{\left|\tilde{p}^T F p\right|}{\sqrt{\tilde{l}_1^2 + \tilde{l}_2^2}} \leq d_0 \tag{2.18}$$

donde p y $\tilde{p}$ son puntos correspondientes y $\tilde{l} = \left[\tilde{l_1}\ \tilde{l_2}\ \tilde{l_3}\right]^T = F^T p$ es la línea epipolar generada por p.

2.2. Programación de GPGPU

Como ha sido vaticinado por varios autores, el paralelismo es el futuro de la computación (Owens et al., 2008). Esta tendencia ya puede verse en la actualidad, donde los esfuerzos de desarrollo se concentran cada vez más en añadir múltiples núcleos a los procesador que a aumentar la frecuencia de operación de estos. Un claro ejemplo de esta situación puede apreciarse en las unidades de procesamiento gráfico o GPU por sus siglas en inglés, que rápidamente han ganado en el último tiempo la madurez suficiente para ser consideradas como una alternativa para realizar cálculos altamente demandantes. A pesar de que las GPU se presentan como una gran opción pensando en el futuro, su arquitectura y modelo de programación son muy diferentes a la gran mayoría de las arquitecturas secuenciales, lo que se fundamenta en que las GPU fueron diseñadas para una clase particular de problemas, relacionados con las necesidades de procesamiento de los juegos tridimensionales, que fueron los que inicialmente motivaron la creación de estas unidades. Las características de estos problemas son las siguientes:

- Los requisitos computacionales son altos: la generación de imágenes en tiempo real requiere mover millones de pixeles por segundo, y cada uno de estos requiere cientos de operaciones.
- El paralelismo es sustancial: la estructura de manejo de datos de las GPU están especialmente diseñadas para esta situación, ya que fueron diseñadas para procesar de manera independiente cada uno de los pixeles que se mostrarán posteriormente en pantalla. Cada uno de estos pixeles es procesado por una unidad de procesamiento programable independiente, lo que lo hace muy adecuado para muchos problemas con dominios distintos a este.
- La tasa de transferencia es más importante que la latencia: el sistema visual humano trabaja en escalas de milisegundos, mientras que las operaciones en procesadores modernos sólo toman un poco nanosegundos. Esta diferencia de aproximadamente 6 órdenes de magnitud permiten que ciclo de procesamiento de un sólo pixel sea largo, con cientos o miles de operaciones aplicadas antes de obtener el resultado final.

A lo largo de los años, las GPU evolucionaron de unidades muy rápidas pero poco versátiles, a sistemas altamente programables en cada una de sus etapas. Este hecho, unido a la aparición de herramientas de desarrollo maduras, fue el que permitió finalmente su aplicación en problemas de propósito general y no sólo gráficos. A modo de comparación, en la actualidad una GPU estándar, Nvidia GeForce 9800GTX - US$130, puede obtener una tasa de 648 gigaflops (GFLOPS), ó 648.000 millones operaciones de punto flotante por segundo, y un ancho de banda de más de 70 GB/s (*Wikipedia - GeForce9 Series*, 2008). Esto entrega un precio de US$0.2 por GFLOPS. Por otro lado, una CPU de alto desempeño, Intel Core i7-920 - US$280, alcanza solamente una tasa aproximada de 61 GFLOPS y un ancho de banda de 19 GB/s (*Tom's Hardware - Intel Core i7 Architecture Review*, 2008), lo que da una razón de US$4.6 por GFLOPS.

2.2.1. Programación de GPUs para propósito general

Una de las dificultades históricas del desarrollo de aplicaciones de propósito general para GPU ha sido el uso obligatorio de interfaces de programación gráfica, como OpenGL, debido a la no existencia de entornos de desarrollo adecuados. Así, las aplicaciones debían estar estructuradas de la misma manera que las aplicaciones gráficas o los juegos, lo que en la mayoría de los casos resultaba completamente contrario a la lógica del problema que se pretendía resolver. Además, era necesario utilizar estructuras y términos de computación

gráfica, como texturas y vértices, lo que hacía aún más complicado el desarrollo. Afortunadamente en los últimos años ha habido un gran avance en este tema, fundamentalmente dado por aumento del nivel de programabilidad de las GPU, lo que motivó la aparición de lenguajes y entornos más expresivos y acordes a la situación (Blythe, 2008).

Las GPU modernas utilizan un modelo de programación single-program multiple-data, o SPMD, que consiste en que el procesamiento de muchos elementos o threads independientes en paralelo utilizando el mismo programa. Este modelo entrega una flexibilidad mucho mayor al clásico modelo de paralelismo SIMD, single-instruction multiple-data, en el cual a cada elemento se le aplica la misma instrucción. El modelo SPMD permite utilizar, a diferencia del SIMD, instrucciones de control de flujo, lo que permite un mayor nivel de control sobre los elementos, redundando en la posibilidad de realizar algoritmos mucho más complejos de manera más simple.

Actualmente, una aplicación de propósito general para GPU se estructura de la siguiente manera:

(i) El desarrollador define de manera directa el dominio del problema mediante una grilla estructurada de elementos (threads). Este paso permite sacar provecho del paralelismo particular de cada aplicación, al hacer explícita la selección de la cantidad y distribución de los elementos.
(ii) Un programa SPMD realiza los cómputos para cada elemento.
(iii) El resultado de los cálculos es almacenado en la memoria global de la GPU para ser utilizado por otro programa SPMD o ser transferidos a la memoria principal del sistema.

2.2.2. Nvidia CUDA

CUDA es un entorno de desarrollo y un modelo de programación desarrollado por Nvidia para utilizar de manera óptima los recursos computacionales de sus GPU. El lenguaje de programación utilizado por CUDA es una versión extendida de C, aumentada con estructuras sintácticas que permiten describir y manejar el paralelismo de la aplicación (NVIDIA, 2008).

En CUDA, el desarrollador define funciones de C llamadas *kernels*, las cuales al ser invocadas se ejecutan N veces en paralelo, donde N es la cantidad de elementos o threads que resolverán el problema de manera paralela. Un textitkernel es definido anteponiendo la declaración "__global__" a una declaración tradicional de función en C, y es invocado con

una leve modificación de una llamada tradicional a función. Esta modificación consiste en la estructura "$<<< M, N >>>$", donde M representa la cantidad de bloques de threads a ejecutar y N la cantidad de threads por bloque. Tanto M como N son vectores, de 2 y 3 dimensiones respectivamente, que permiten asignarle estructura a la ejecución de threads, de manera de poder modelar de mejor manera el dominio del problema a resolver.

Cada uno de los threads recibe un identificador único, cuyo valor puede ser obtenido a través de las variables "blockDim", "blockIdx" y "threadIdx". Estas variables son vectores de 2, 2 y 3 dimensiones respectivamente, que permiten extraer la estructura del procesamiento definida al momento de ejecutar el textitkernel mediante los vector M y N. Esto permite el manejo de manera natural de dominios vectoriales o matriciales, que son los comúnmente usados en aplicaciones científicas o de ingeniería. El siguiente trozo de código ejemplifica esta situación, mediante un simple algoritmo para sumar los contenidos de dos matrices de $N \times N$ implementado en un textitkernel y su modo de ejecución.

```
__global__ void sumaMatriz(float A[N][N], float B[N][N],
    float C[N][N])
{
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    int j = blockIdx.y * blockDim.y + threadIdx.y;
    if (i < N && j < N)
    {
        C[i][j] = A[i][j] + B[i][j];
    }
}

int main()
{
    dim3 bloque(16, 16);
    dim3 grilla((N + bloque.x - 1) / bloque.x,
        (N + bloque.y - 1) / bloque.y);
    sumaMatriz <<< grilla, bloque >>> (A, B, C);
}
```

Cabe destacar que en el código anterior, el tipo de datos `dim3` representa un vector que puede ser de una, dos o tres dimensiones.

El modelo asume que los threads de cada bloque se ejecutan de manera paralela, y que cada bloque es independiente de los otros. Esto permite que los threads de cada bloque puedan cooperar entre ellos al compartir información mediante un espacio de memoria compartida y sincronizar su ejecución para coordinar los accesos a esta y hacer finalmente el proceso de cómputo mucho más eficiente. La cantidad máxima de threads por bloque en CUDA es de 512. La cantidad de bloques de threads es definida generalmente por el tamaño de los datos a procesar, a diferencia de la cantidad de threads por bloque, que está mayormente influida por las limitaciones de la arquitectura de hardware subyacente.

Finalmente, además del entorno de desarrollo, CUDA incluye dos librerías matemáticas de alto nivel que implementan herramientas de uso común. Estas son CUFFT, que incluye implementaciones de la transformada rápida de Fourier para una, dos y tres dimensiones, y CUBLAS, que es una librería de álgebra lineal optimizada para el manejo de matrices y vectores de gran tamaño.

3. DETECCIÓN DE CABEZAS BASADA EN GPGPU

El lograr un detector de personas rápido y robusto a cambios de iluminación y a oclusiones temporales son metas centrales que deben cumplirse para obtener un buen resultado en la etapa de conteo. Un bajo tiempo de ejecución limita las posibilidades de que un objetivo no sea detectado debido a la velocidad de su movimiento y permite que las diferencias entre imágenes consecutivas sean reducidas, lo que ayuda a facilitar el análisis al no ser necesario hacer supuestos sobre el movimiento de los objetos en el tiempo entre las imágenes. Por otro lado, la robustez a la iluminación y oclusión facilita mucho el trabajo del análisis estéreo, ya que permite que el matching entre los elementos detectados en las dos imágenes pueda realizarse con mayor certeza y precisión en la ubicación.

Para alcanzar la meta de la velocidad, las partes más demandantes del detector fueron realizadas utilizando la potencia de las tarjetas de video modernas mediante CUDA. Aunque las ventajas de esta decisión son muchas, requiere gran cantidad de esfuerzo y tiempo adaptar de forma eficiente algoritmos a este estilo de computación paralela. Además, no todos los algoritmos son paralelizables y algunos, aunque lo sean, no obtienen beneficios importantes que valga la pena el esfuerzo de adaptarlos. Intuitivamente, la idea es modelar el problema de detección de tal forma que pueda ser dividido en muchos problemas más pequeños e independientes. Al trabajar con imágenes, esto se facilita ya que es posible analizar distintas secciones de cada imagen de forma independiente. Para llevar esta intuición a un modelo más formal, podemos utilizar la ley de Amdahl (Amdahl, 1967), que describe cuánto se puede acelerar un algoritmo al paralelizar cierta fracción de él:

$$A = \frac{1}{(1-f) + \frac{f}{S}} \tag{3.1}$$

donde f representa la fracción paralelizada del algoritmo y S el nivel de aceleración de la parte paralela del algoritmo. Por ejemplo, para un algoritmo que puede ser paralelizado en un 80% y con una ganancia de velocidad de 10 veces para la fracción paralela, tendrá una aceleración total de $\frac{1}{(1-0.8)+\frac{0.8}{10}} \approx 3.6$ veces. En el caso que S tienda a infinito, es decir, existen infinitos procesadores paralelos, la ganancia se eleva a 4. Como se puede apreciar, el valor de f es mucho más preponderante que el de S a la hora de obtener una buena ganancia de velocidad. Así, para aprovechar de mejor manera las bondades que nos entregan las GPU, es necesario utilizar un algoritmo cuya componente secuencial sea muy baja,

idealmente no mayor al 10%. Ejemplos de algoritmos para este tipo de problemas hay muchos, como por ejemplo la transformada rápida Fourier, matching de vectores, algoritmos genéticos o las búsquedas por fuerza bruta en criptografía.

Pasando a la detección de objetos, el enfoque más tradicional y usado es el de ventana deslizante. Los mejores ejemplos de estas técnicas se han dado en la detección de humanos, ya sea a través del rostro (Viola & Jones, 2001) o del cuerpo entero (Dalal & Triggs, 2005), donde obtienen resultados excelentes en situaciones reales. Su funcionamiento típico consiste en deslizar una ventana del tamaño del objeto a detectar por toda la imagen y a distintas escalas y en cada posición de esta ventana aplicar una máscara o extraer ciertas características importantes del objeto y aplicarles un clasificador como AdaBoost, redes neuronales, árboles de decisión o SVM.

Los sistemas basados en extracción de características obtienen en general mejor rendimiento que los basados en la aplicación de máscaras, pero tienen el problema de que se debe realizar un entrenamiento complejo del clasificador para poder obtener buenos resultados. Para desarrollarlo se debe contar además con una gran cantidad de imágenes rotuladas (≈ 5000), tanto del objeto de detectar en todas las posiciones posibles y de los que no se quiere detectar, i.e., el fondo. Dados los resultados obtenidos, es factible construir una base de datos de ese tamaño cuando se requiere detectar un objeto en específico y bajo una cantidad de puntos de vista bastante restringida, de lo contrario, el tamaño de la base de datos aumenta considerablemente. Incluso aumentando este tamaño, los detectores de objetos bajo múltiples vistas no han tenido resultados del todo satisfactorios. Esto último descarta su uso en esta aplicación, ya que es necesario detectar personas entrando y saliendo, por lo que metodologías como la detección de rostros, que son los con mejores resultados para este tipo de enfoque, no son factibles. Aunque se han hecho intentos por disminuir la complejidad del entrenamiento (Levi & Weiss, 2004) o realizar el entrenamiento en línea (Wu & Nevatia, 2007), los resultados son bastante inferiores a los con entrenamientos complejos.

En el área de detección de personas, existen principalmente 3 distintas opciones de búsqueda para enfrentar este problema. La primera realiza la búsqueda de personas en base a la apariencia de alguna parte específica del cuerpo, como rostro, cabeza, torso, etcétera. Este método es el más utilizado debido a su simpleza, aunque presenta problemas frente a la oclusión y al ruido. El segundo enfoque enfrenta el problema buscando a las personas como un todo, i.e., detectando todo el cuerpo al mismo tiempo. El problema de esta metodología, al igual que el anterior, es su sensibilidad a la oclusión parcial. El último método busca detectar elementos individuales del cuerpo, para luego juntarlos y decidir si se trata de

una persona en base a su disposición espacial. Este enfoque es bastante robusto a las oclusiones, pero depende mucho del modelo geométrico impuesto como restricción para el posicionamiento relativo de las partes del cuerpo.

Un motivo muy fuerte para utilizar el primer enfoque es que el sector más relevante, estable y repetible para la detección de humanos bajos distintos ángulos es la cabeza (Ishii, Hongo, Yamamoto, & Niwa, 2004). Esta tiene una ventaja que facilita su detección de sobremanera al ser aproximadamente elíptica desde casi cualquier ángulo. Además, al aumentar la distancia, los ejes de la elipse que define la cabeza tienden a ser muy parecidos, lo que facilita un poco más la situación ya que puede aproximarse como un círculo.

Teniendo esto en cuenta, es posible enfrentar este problema desarrollando un detector de objetos circulares no completos y con tolerancia a la deformación elíptica. La metodología más común para esto es usar la transformada generalizada de Hough. A pesar de su gran robustez, al adaptar la transformada al caso de formas elípticas, los requerimientos de espacio se disparan, ya que es necesario utilizar acumuladores de 5 dimensiones. Además, esto complica la búsqueda de máximos lo que aumenta la sensibilidad al ruido y el tiempo de ejecución. Para solucionar estos problemas se han propuesto variadas modificaciones que buscan disminuir la cantidad de parámetros con los que modelar la elipse. En el mejor caso, se ha logrado un sólo parámetro mediante restricciones geométricas (Chia, Leung, Eng, & Rahardja, 2007) y (Xie & Ji, 2002), con resultados bastante buenos para imágenes sintéticas o ya segmentadas, pero con serios problemas en aquellas con fondos complejos y oclusión.

Otros trabajos como los de (Grammalidis & Strintzis, 2000) y (Z. Hu, Kawamura, & Uchimura, 2007) modelan tridimensionalmente la cabeza utilizando un elipsoide, pero necesitan información espacial entregada por una calibración, mapas de disparidad o por el usuario, lo que limita su uso en sistema para entornos no controlados como este.

Recientemente, (Gardel et al., 2007) propusieron un sistema de detección de cabezas bastante eficiente que utiliza una FPGA para funcionar en tiempo real. El sistema comienza con una etapa de sustracción de fondo, para luego, utilizando detección de bordes y una serie de convoluciones, detectar los elementos circulares de la imagen. A pesar de que el sistema ahí descrito es bastante simple, sus ideas base sirvieron para el desarrollo del sistema detector desarrollado, ya que puede ser fácilmente paralelizado.

El sistema de detector desarrollado consta de 6 etapas. Primero se realiza una detección de bordes, para luego procesar este resultado mediante convoluciones utilizando

máscaras circulares. Luego, se mezclan los resultados de estas convoluciones y se etiquetan los sectores donde existe alta probabilidad de que exista un elemento circular con su radio más probable. Finalmente, los sectores etiquetados son procesados y fusionados para evitar detecciones múltiples de un mismo elemento. La figura 3.1 muestra un diagrama del desarrollo del proceso de detección.

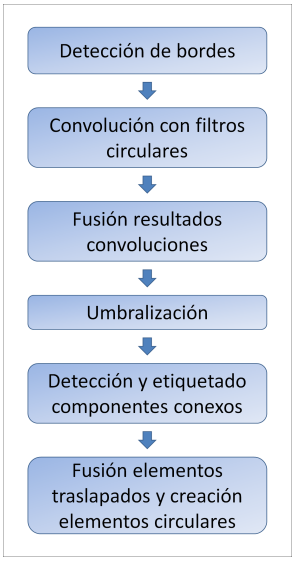


FIGURA 3.1. Diagrama de flujo del algoritmo detector de elementos circulares.

El algoritmo detector aprovecha completamente las bondades de las GPU para poder correr en tiempo real. Las etapas 1 a 4, que toman cerca del 90% del tiempo de ejecución, fueron completamente desarrolladas de forma paralela, lo que permite tener más tiempo para realizar el resto de los procesos de manera más robusta. A continuación se describen cada uno de los componentes del detector y posteriormente se presentarán sus resultados.

3.1. Detección de bordes

El detector de Canny (Canny, 1986) es reconocido como uno de los mejores detectores de bordes. Tiene una baja sensibilidad al ruido, gracias a una etapa previa de filtrado gaussiano, es bastante rápido, ya que las máscaras de convolución son separables, y entrega contornos muy bien definidos, debido a una supresión de no máximos y a un proceso de doble umbralización con histéresis. Esto último le da una ventaja clara, en esta aplicación, sobre otros detectores como el LoG (Gonzalez & Woods, 2008), ya que el siguiente paso del proceso necesita la mejor definición posible en los contornos de los objetos, con la menor cantidad de ruido y artefactos posible. Todo esto lo hace ideal para ser utilizado

en un sistema de este tipo, ya que tiene las características de robustez necesarias para un proceso inicial de bajo nivel.

La implementación utilizada es la descrita en (Luo & Duraiswami, 2008), por lo que está completamente desarrollada en CUDA. La primera etapa del algoritmo consiste en filtrar la imagen utilizando una máscara gaussiana. Como es sabido, la máscara gaussiana es separable, por lo que este proceso es extremadamente veloz. Experimentalmente el ancho de la máscara fue fijado en 15 (radio igual a 7) y $\sigma = 2$. Estos valores fueron los que mejores resultados entregaron, aunque variaciones en ellos no producen mayores efectos en el proceso. Posteriormente se realiza el proceso de cálculo de gradientes mediante 2 máscaras separables de Sobel. A continuación se calcula la magnitud y dirección de los gradientes para luego realizar la supresión de no máximos. Finalmente se aplica el proceso de doble umbralización mediante un algoritmo de crecimiento de regiones. Con cuatro iteraciones de este algoritmo se obtienen excelentes resultados, aunque es posible realizar 1 ó 2 de estas con el fin de ejecutar el algoritmo en menos tiempo, a costa de un poco menos de definición en los contornos. Los valores para los umbrales superior e inferior son bastante insensibles a variaciones, por lo que se determinó experimentalmente que los valores debían ubicarse en las cercanías del 50 para esta aplicación. Los pixeles que son detectados finalmente como bordes son marcados con un 1, mientras que el resto es marcado con un 0. Esta imagen binaria es la utilizada posteriormente en el proceso de filtrado lineal. La figura 3.2 muestra una imagen de dos personas caminando junto a su imagen de bordes.

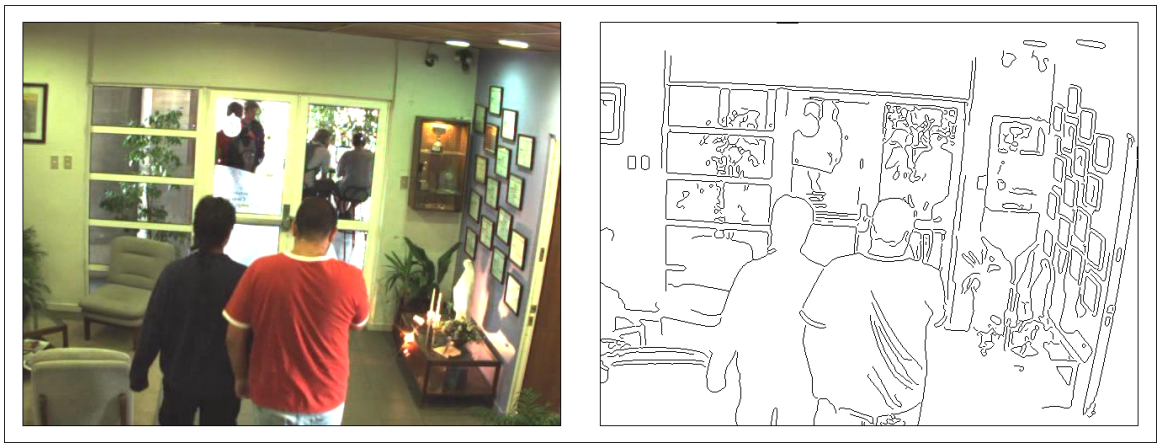


FIGURA 3.2. Resultado del proceso de detección de bordes.

3.2. Detección elementos circulares usando filtrado lineal

La etapa de filtrado lineal es la parte más importante y novedosa el algoritmo. Gracias a la potencia de las GPU, es posible aplicar reiteradamente máscaras de gran tamaño a una imagen utilizando unos pocos milisegundos por cada uno. Esto hace posible realizar procesos de filtrado más complejos y con mejores resultados.

La meta de esta etapa del algoritmo es encontrar elementos circulares en el mapa de bordes de la imagen. Además, para ser considerados como probables cabezas, estos sectores circulares deben tener la menor cantidad de artefactos en su interior, ya que los bordes de los rasgos faciales, salvo que la cámara esté muy cerca, aparecen de manera muy tenue.

Un posible enfoque es utilizar el esquema de ventana deslizante, aplicando para cada posición y escala una máscara del tamaño de la ventana que describa la forma circular buscada. Este enfoque se adapta muy bien al esquema de paralelismo que utilizan las GPU, pero es bastante complicado realizar una implementación eficiente debido a la gran cantidad de accesos a la memoria de la tarjeta de video requeridos.

Otra metodología igualmente eficiente y mucho más simple de implementar es la detección mediante convoluciones. Como se sabe, la convolución entre dos funciones representa el nivel de traslape que exhiben en cada uno de los puntos. Así, la convolución entre el mapa de bordes y una máscara que describa una región circular nos entregará, por cada pixel de la imagen, un número que representa el nivel de similitud entre la máscara aplicada y un sector de la imagen del mismo tamaño y centrado en ese punto. De esta forma, el proceso de filtrado se puede realizar aplicando una convolución por cada una de las escalas en las cuales se desea hacer la detección. Este escalamiento se puede hacer cambiando tanto el tamaño de la imagen como el tamaño de la máscara.

Así, sólo resta especificar las máscaras que describan de la mejor manera posible la forma a detectar. Estas deben favorecer figuras circulares con cierta deformación elíptica, no deben tomar en cuenta los alrededores de esta figura y por último, deben castigar la existencia de artefactos dentro de la figura. Además, desde un punto de vista práctico, estas máscaras deberían ser lo más simétricas y simples posible y deben ser fácilmente escalables, de forma de evitar cambio de las dimensiones de la imagen para obtener detecciones de distintos tamaños. La figura 3.3 muestra algunas de las familias de máscaras probadas.

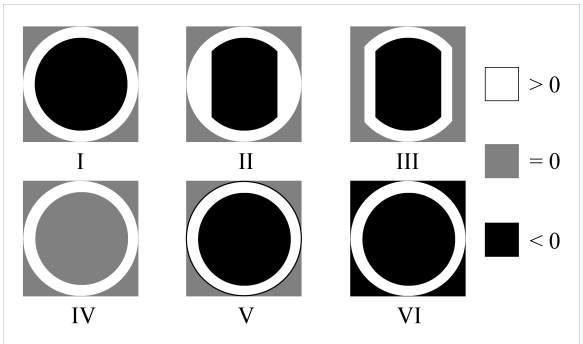


FIGURA 3.3. Distintas máscaras probadas para la detección basada en filtrado lineal.

Dentro de estas familias, la que mejores resultados entregó en la detección desde distintos puntos de vista fue la del tipo I. Estas máscaras entregan una muy buena tolerancia a la deformación elíptica gracias a sus bordes gruesos y son muy fácilmente escalables. Además, su centro con valor negativo permite un rápido rechazo de regiones no circulares o regiones con muchos artefactos interiores que en general no corresponden a cabezas. Teniendo esto en cuenta, una máscara de la familia elegida se define de acuerdo a 4 parámetros, $M(r_i, r_e, s_i, b)$, donde r_i es el radio interior, r_e el radio exterior, s_i el valor de los elementos en el sector interior, b el valor en el borde. El ancho y alto de la máscara toman su valor de acuerdo al radio exterior y está dado por la expresión $S = 2 \cdot r_e + 1$. Luego, el centro de ella se encuentra ubicado en $(r_e + 1, r_e + 1)$. Además se define el radio promedio de una máscara como $r_p = \left\lfloor \frac{r_e + r_i}{2} \right\rfloor$. Los coeficientes toman sus valores de acuerdo a las siguientes reglas basadas en su distancia d al centro de la máscara:

$$v(d) = \begin{cases} s_i, & \text{si } d < r_i \\ b, & \text{si } r_i \leq d \leq r_e \\ 0, & \text{en cualquier otro caso.} \end{cases} \tag{3.2}$$

Para el proceso de umbralización es muy importante que la aplicación de máscaras de distinta escala entregue el mismo resultado para formas circulares que sólo difieren en esa escala. Se define entonces la máscara base $M_b = M(r_{ib}, r_{eb}, s_{ib}, b_b)$. Para el escalamiento, los valores de los elementos de la máscara son modificados de acuerdo a 3 criterios. Para los elementos internos, s_i, se utiliza un escalamiento de valor inversamente proporcional al cambio de área de un círculo, de acuerdo a la formula:

$$s_i = \frac{r_{ib}^2 \cdot s_{ib}}{r_i^2} \tag{3.3}$$

Así, si $r_{ib} = 20$ y $s_{ib} = -1$, para una máscara con $r_i = 30$ este valor será igual a $\frac{20^2 \cdot -1}{30^2} = -0.\overline{4}$. Para los valores del borde, b, se utiliza un escalamiento inversamente proporcional al perímetro de un círculo. Si el área del borde de ancho $w = r_e - r_i + 1$ de una máscara corresponde a $A = 2\pi r_i + 2\pi(r_i+1) + \cdots + 2\pi(r_i+w-1) = 2\pi(wr_i + \sum_{k=1}^{w-1} k) = 2\pi(wr_i + \frac{(w)(w-1)}{2})$, entonces, el escalamiento se calcula de acuerdo a:

$$b = \frac{(wr_{ib} + \frac{w(w-1)}{2}) \cdot b_b}{wr_i + \frac{w(w-1)}{2}} \tag{3.4}$$

De esta forma, si $r_{ib} = 20$, $r_{eb} = 24$ y $b_b = 1$, para una máscara con $r_i = 30$ y $r_e = 34$ este valor será igual a $\frac{(5 \cdot 20 + \frac{5 \cdot 4}{2}) \cdot 1}{5 \cdot 30 + \frac{5 \cdot 4}{2}} = 0.643$.

Finalmente, para no perder detecciones debido a los cambios de escala, el tamaño de las máscaras se selecciona de manera que estos tengan sus bordes traslapados en un pixel. Así, si los filtros tienen un ancho de borde $w = 5$, entonces el ancho (o alto) de máscaras consecutivos difiere en 8 unidades. La figura 3.4 ejemplifica esta situación.

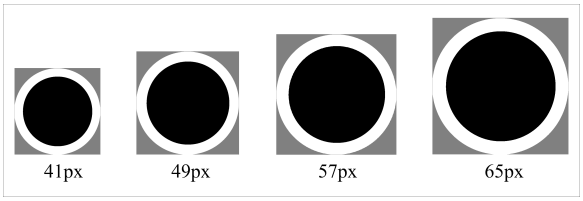


FIGURA 3.4. Máscaras consecutivas para la detección en varias escalas.

Desde el punto de vista de la implementación, todos los elementos pueden ser programados de manera eficiente utilizando CUDA. A pesar de esto, existe un punto en el que aún se puede obtener mejor rendimiento. Los filtros utilizados no son separables, i.e., no pueden ser expresados como el producto de dos máscaras unidimensionales, por lo que no es posible utilizar la técnica de convolución separable para disminuir el tiempo de ejecución. A pesar de que igualmente la convolución 2D se puede implementar de forma fácil y eficiente, su tiempo de ejecución esta directamente relacionado con el tamaño de la máscara. La complejidad computacional para una convolución 2D separable es $O(W+H)$, donde W y H son el ancho y alto de la máscara respectivamente, mientras que la de la convolución 2D tradicional es $O(W \cdot H)$. De esta forma, su tiempo de ejecución aumenta de forma cuadrática con el tamaño. Para máscaras pequeñas o separables esto no es problema,

pero cuando no ocurre esto, como en el caso de las máscaras aquí utilizadas, esto puede transformarse en un grave problema. Afortunadamente, podemos utilizar uno de los resultados más importantes de la teoría de análisis de señales, el teorema de la convolución. Sea F el operador transformada de Fourier, F^{-1} su inversa y f, g dos funciones cuya convolución existe, entonces se cumple que:

$$f * g = F^{-1}\{F\{f\} \cdot F\{g\}\} \tag{3.5}$$

Si la imagen a procesar es de $N \times N$ y la máscara de $W \times H$, la complejidad computacional de esta versión de la convolución es $O(NlogN)$, asumiendo que N es un número altamente compuesto. Luego, si se cuenta con una implementación eficiente de la transformada de Fourier, como la de la librería CUFFT disponible en CUDA, es posible realizar la convolución con máscaras grandes, i.e., $W \times H >> NlogN$, de forma más rápida utilizando sólo 2 aplicaciones de la transformada y un producto interno.

Existen algunos detalles de la implementación en CUDA que aún deben ser discutidos. El primero tiene que ver con la formulación del teorema de la convolución. En él, los vectores o matrices que representan a las funciones tienen el mismo tamaño, por lo que es necesario aumentar el tamaño tanto del mapa de bordes como de la máscara mediante un relleno con ceros hasta alcanzar el tamaño de la convolución completa. Para una imagen de $N \times N$ y una máscara de $W \times W$, el tamaño de la convolución 2D completa es $(N + W - 1) \times (N + W - 1)$.

El segundo detalle tiene que ver con el rendimiento de la transformada de Fourier con respecto al tamaño de las matrices a procesar. Como es sabido, la transformada rápida de Fourier separa el problema de muchos subproblemas de menor tamaño. De esta forma, mientras más sencilla sea la separación más bajo será el tiempo de ejecución. El caso ideal es que el tamaño de las matrices sea una potencia de 2 o un número altamente compuesto. Luego de realizar pruebas utilizando imágenes de tamaños tradicionales y máscaras de anchos entre 21 y 201 pixeles, los valores que mejor rendimiento entregaron para CUFFT fueron los múltiplos de 64, tanto para el ancho como para el alto. Por ejemplo, para una imagen de 640×480 y una máscara de 61×61, el tamaño de la convolución completa es 700×540, por lo que, aproximando a los múltiplos de 64 mayores y más próximos, el tamaño óptimo para el procesamiento queda en 704×576. De esta forma, el proceso de relleno debe hacerse, tanto para el mapa de bordes como para la máscara, hasta alcanzar el tamaño óptimo y no sólo el de la convolución completa.

Cabe destacar finalmente que todos los textitkernels de CUDA desarrollados para la convolución realizan accesos "fundidos" a memoria, con el fin de alcanzar un rendimiento óptimo. Esto se logró realizando accesos a ubicaciones de memoria contigua en threads contiguos, siendo cada uno de estos accesos de 32 bits. De esta manera, en vez de hacer un acceso a memoria por thread, se hace uno sólo cada 16 threads, reduciendo al mínimo la latencia generada por los accesos. Los detalles de las técnicas de para realizar estos accesos a memoria se puede encontrar en la guía de programación de CUDA (NVIDIA, 2008).

De esta forma, la implementación del algoritmo de filtrado puede resumirse de la siguiente manera:

(i) Preprocesamiento: Se calcula la cantidad de máscaras a ocupar en base a los tamaños de cabeza que se desea detectar. Estos valores deben ser ingresados por el usuario. Luego cada máscara es generada con el tamaño apropiado.

(ii) Copia a la GPU y relleno con ceros: Tanto el mapa de bordes como las máscaras a utilizar son copiados a la memoria de la GPU y rellenados con ceros hasta alcanzar el tamaño óptimo. Este proceso se realiza dividiendo la imagen o máscara aumentada en bloques de 16×16 pixeles. Cada uno de estos bloques es procesado por un bloque de 16×16 threads en CUDA. Cada thread verifica si sus coordenadas corresponden a las de un pixel de la matriz de origen o un pixel de relleno. En el primer caso toma el valor de la imagen o máscara correspondiente a esa coordenada y lo copia a la matriz aumentada, ubicada en la GPU, en la coordenada correspondiente. En el otro caso, escribe un cero en esa coordenada de la matriz.

(iii) Transformación inicial: Tanto el mapa de bordes aumentado como las máscaras aumentadas son trasladados al dominio de la frecuencia mediante CUFFT. Cabe notar que el resultado de estas transformaciones son matrices de números complejos. Además, la transformación es no normalizada, lo que significa que los elementos están escalados con respecto a la cantidad de elementos de la transformación. Por este motivo, antes de hacer la transformada inversa, es necesario normalizar los valores dividiendo cada uno por la cantidad de elementos de la transformación.

(iv) Modulación y normalización: Se realiza un producto elemento por elementos entre las transformaciones del mapa de bordes aumentado y cada máscara aumentada. Para esto se utilizan bloque de 256 threads. Cada thread se ocupa

de hacer el producto complejo entre un elemento del mapa de bordes transformado y un elemento de la máscara transformada. Además, divide el resultado de este producto por la cantidad de elementos de la transformación para obtener un resultado correcto al hacer la transformación inversa.

(v) Transformación Inversa: La matrices complejas resultantes de la etapa anterior son transformadas utilizando la transformada inversa de Fourier, volviéndolas a su dominio original.

(vi) Formateo de resultados: Como el resultado esperado para la convolución es, en este caso, del mismo tamaño que la imagen original, es necesario "recortar" los bordes de las matrices resultantes de la etapa anterior. La matrices buscadas son del mismo tamaño que la imagen original y comienza en el elemento $(\frac{w-1}{2}, \frac{w-1}{2})$ de las matrices completas, donde w es el ancho de la máscara aplicada en cada caso. Para realizar este proceso se utiliza una estrategia casi idéntica a la utilizada en el punto (i), sólo que en este caso cada thread verifica si su coordenada corresponde a la parte buscada de la matriz o no. En caso que corresponda, copia el contenido de esa coordenada a otra matriz, en caso contrario no hace nada.

El resultado de este proceso es una serie de mapas filtrados, cuyos máximos locales representan los centros de sectores que tienen una alta probabilidad de ser circulares. Mediante el proceso de umbralización descrito en la sección siguiente, estas ubicaciones son extraídas para su posterior procesamiento. La imagen 3.5 muestra el mapa de bordes de la figura 3.2 y el resultado de la aplicación de este proceso a este mapa utilizando máscaras en 4 escalas distintas.

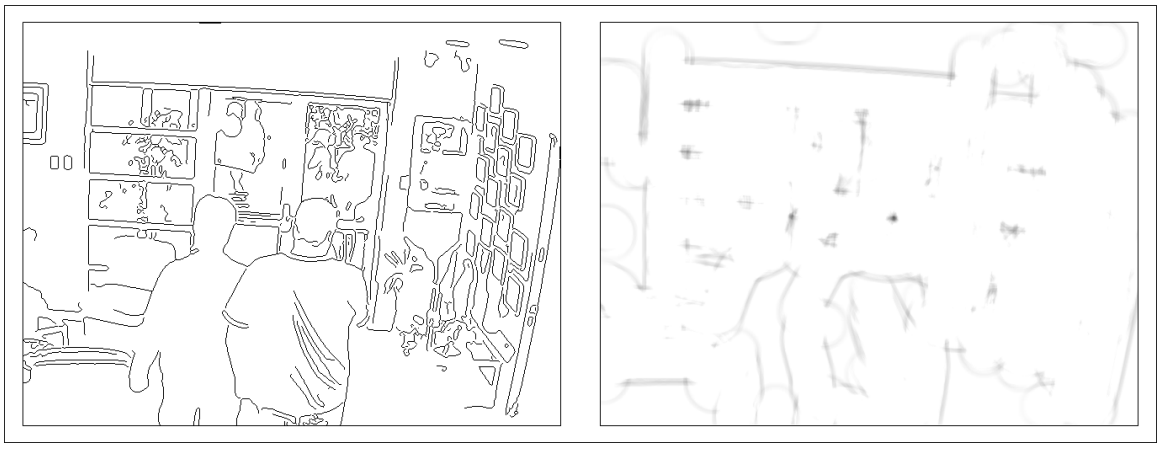


FIGURA 3.5. Resultado del proceso de filtrado lineal aplicado usando 4 máscaras de distintas escalas.

3.3. Umbralización

El proceso de umbralización se realiza progresivamente en cada uno de los mapas filtrados, para luego fusionar la información de ellos. La idea detrás de esto es realizar posteriormente sólo una detección de componentes conexos y reducir la cantidad de detecciones múltiples del mismo objeto. Si el proceso se desarrolla de forma independiente para cada mapa filtrado, se debe realizar un proceso de detección de componentes conexos por cada uno, lo que aumentaría considerablemente el tiempo de ejecución.

El proceso de umbralización parte con la definición de un umbral de aceptación. Este umbral es el que define el nivel de tolerancia del detector a figuras circulares no completas. Como en los mapas de bordes cada pixel correspondiente a un borde está marcado con un 1, una circunferencia perfecta de radio r obtendrá, en un mapa filtrado con un filtro de $r_i = r$ un valor máximo aproximado de $2\pi rb$, ubicado en el pixel correspondiente al centro de la circunferencia. Luego, sea $t \in [0 - 1]$, el nivel de tolerancia a figuras incompletas, se define el umbral de aceptación u_a como una función de t, r_{ib} y b_b:

$$u_a(t, r_{ib}, b_b) = 2\pi r_{ib} b_b t \tag{3.6}$$

De esta forma, sólo basta especificar los parámetros del filtro base F_b y el nivel de tolerancia para obtener el umbral de aceptación. Además, como los valores de los coeficientes de los filtros también se escalan con el tamaño de este, el umbral es común a todas las escalas, por lo que no es necesario modificarlo para obtener detecciones consistentes.

Una vez definido el umbral de aceptación, es posible describir el proceso de umbralización. Este comienza con el análisis del mapa que fue filtrado con la máscara de menor escala. Para cada pixel de este mapa se verifica si su valor supera o iguala al umbral de aceptación. En caso de hacerlo, el pixel es marcado con el radio promedio del filtro correspondiente a ese mapa. En caso contrario, el pixel es marcado con un 0. El resultado de este proceso es utilizado como entrada para la siguiente umbralización, junto con el mapa filtrado que corresponde a la siguiente escala de detección. Las siguiente umbralización es algo distinta a la primera, ya que después de verificar si se supera o iguala el umbral, fusiona la información la de la umbralización anterior. Al igual que en el proceso previo, si el pixel a verificar del mapa filtrado es mayor o igual al umbral, entonces se marca ese pixel con el radio promedio del filtro correspondiente a ese mapa. En caso contrario, el pixel es marcado con el valor obtenido para ese pixel en la umbralización anterior. Este proceso se repite de la misma manera en las siguientes umbralizaciones, hasta obtener el mapa final

de detecciones. Utilizando este proceso de fusión se logra suprimir una gran cantidad de detecciones múltiples del mismo objeto en escalas consecutivas, dejando sólo la que tiene el mayor tamaño. Las detecciones múltiples que no logran ser eliminadas con este proceso son analizadas en la siguiente etapa del algoritmo. La figura 3.6 muestra la imagen filtrada de la figura 3.5 junto con el resultado del proceso de umbralización para esta imagen.

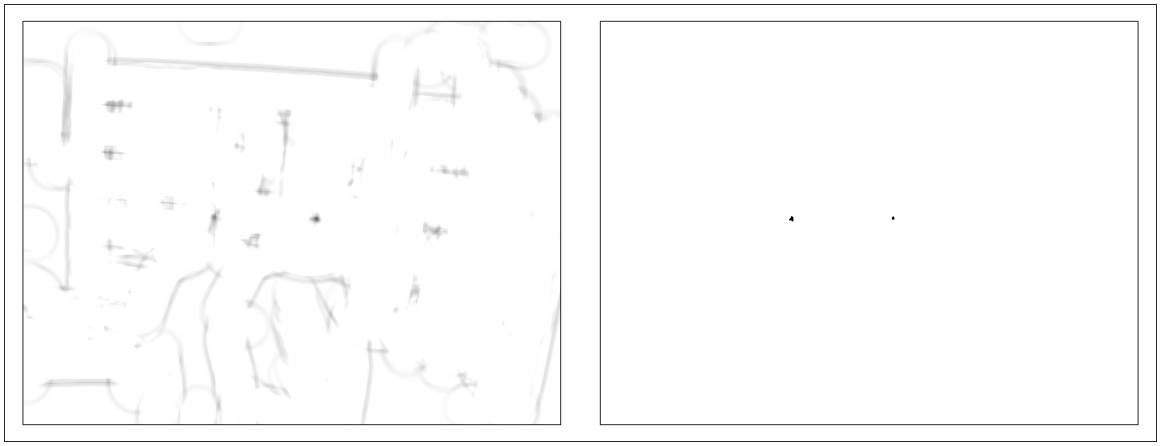


FIGURA 3.6. Resultado del proceso de umbralización.

Un mapa de detecciones final presenta grupos de pixeles vecinos que están marcados con valores iguales o muy similares. Cada uno de estos grupos representa en general una detección. En los que sean dos detecciones de objetos distintos, se discriminará de acuerdo al valor de su etiqueta. Este proceso de análisis de componentes conexos se desarrolla en la siguiente etapa del algoritmo.

La implementación del proceso de umbralización fue completamente desarrollada en CUDA. El algoritmo divide cada mapa de bordes en bloques de 16×16 pixeles y cada uno de estos es procesado por un bloque de 16×16 threads. Luego, cada uno de estos verifica si el pixel correspondiente a sus coordenadas es mayor o igual al umbral, marcando el valor que corresponde de acuerdo a los criterios descritos anteriormente. Cuando concluye el proceso para todos los mapas filtrados, el resultado es devuelto al espacio de memoria del host para la detección de componentes conexos.

3.4. Detección y fusión de componentes conexos

Para realizar cualquier seguimiento sobre las detecciones, es necesario transformarlas en estructuras bien definidas, que agrupen la mayor cantidad de información con respecto

a los pixeles que definieron la detección. Este tipo de información es por ejemplo el centroide, el área y la etiqueta de los pixeles que forman la detección, que representa finalmente el radio del elemento circular detectado.

Dadas las características del resultado de la etapa anterior del detector, para capturar esta información se utilizó la técnica de detección de componentes conexos basada en crecimiento de regiones. En vez de realizar dos pasadas por la imagen, como el algoritmo tradicional de detección (Rosenfeld & Pfaltz, 1966), el crecimiento de regiones se realiza usando sólo una, utilizando un procesamiento recursivo para hacer crecer las regiones detectadas hasta su tamaño real. Dado que las regiones a detectar son bastante pequeñas, este enfoque resulta mucho más eficiente.

El algoritmo itera por cada uno de los pixeles de la imagen, hasta que encuentra alguno que tenga valor distinto de cero. Este pixel pasa a ser la "semilla" para el algoritmo de crecimiento de regiones y es marcado como "visitado". El crecimiento consiste en buscar entre los vecinos de ese pixel los que tienen un valor igual y que no hayan sido visitados. El concepto de vecino se puede definir mediante 2 criterios, conectividad 4 y conectividad 8. La primera toma como vecinos sólo los pixeles ubicados en las cuatro direcciones principales, i.e., arriba, abajo, izquierda y derecha. Conectividad 8 toma además de estos pixeles, los ubicados en las 4 diagonales. Volviendo al algoritmo, cuando se encuentra un vecino, se agrega a una cola de pixeles por revisar. Una vez que se ha terminado con todos los vecinos del pixel actual, se extrae un pixel de la cola y se repite el proceso de búsqueda y marcado de vecinos con él. Una vez que no queden más pixeles por analizar en la cola, se calcula el centroide y el área de los pixeles analizados. Esta información se almacena en una estructura que representa el componente conexo formado por estos pixeles. Además, se define como radio del componente conexo al valor de los pixeles. Una vez creado este componente conexo, el algoritmo continúa la búsqueda de otros componentes desde el pixel que sigue inmediatamente a la última semilla utilizada, hasta terminar de recorrer la imagen.

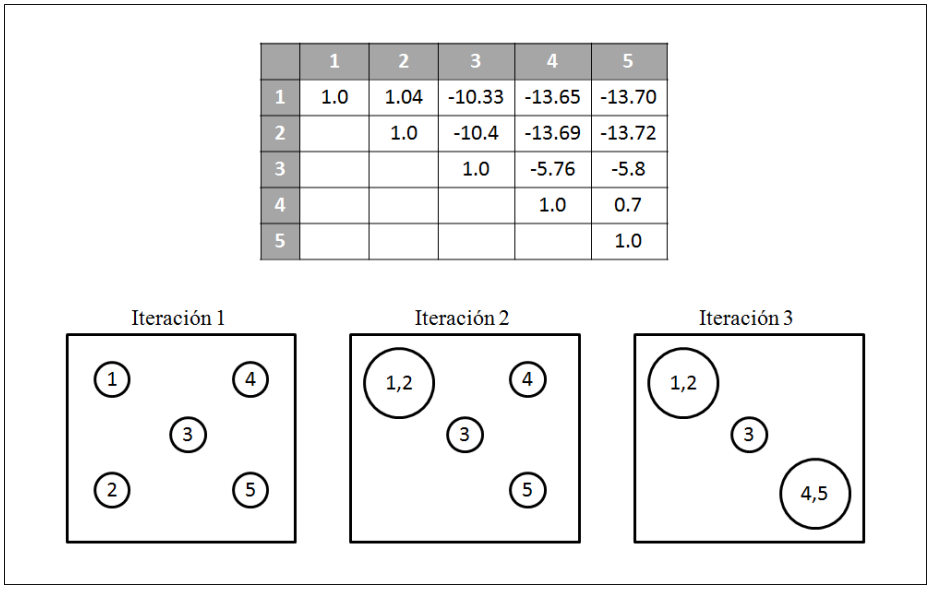
Una vez encontrados los componentes conexos y creadas las estructuras que los representan, se realiza una limpieza y fusión de estos. La idea de la limpieza es eliminar de forma temprana aquellas detecciones que tengan un área muy reducida, ya que es muy probable que hayan sido producidas por ruido o por disposiciones fortuitas de los elementos de la imagen. Empíricamente se determino que componentes conexos con áreas menores o iguales a 3 pixeles son, con gran nivel de certeza, falsos positivos.

Una vez eliminados estos componentes, se realiza el proceso de fusión para unir las detecciones múltiples de un mismo objeto. Este consiste en unir las detecciones cuyo traslape sea mayor a cierto umbral t_0. El traslape entre dos componentes conexos se define como:

$$t = \frac{r_1 + r_2 - \|\vec{c_1} - \vec{c_2}\|}{2 \times \min\{r_1, r_2\}} \tag{3.7}$$

donde $\vec{c_i}$ y r_i representan el centroide y el radio de los 2 componentes conexos comparados. Valores de t menores o iguales a cero indican que no existe traslape mientras que valores mayores o iguales a 1 indican que la circunferencia definida por uno de los componentes conexos contiene a la definida por el otro. Finalmente, los valores en el intervalo $]0, 1[$ indican que existe traslape, pero no completo, entre los componentes.

El proceso de fusión comienza formando un conjunto de un sólo elemento con cada componente conexo. Luego, de manera iterativa, se van fusionando los conjuntos que presenten elementos cuyo nivel de traslape es mayor o igual a $t_0 = 0.5$. Este proceso continua hasta que no haya más conjuntos por fusionar (Figura 3.7). A continuación, se genera una estructura representante de cada conjunto, calculando el centroide y el radio promedio de los componentes conexos que lo conforman. Estas estructuras representan las detecciones finales del algoritmo.



	1	2	3	4	5
1	1.0	1.04	-10.33	-13.65	-13.70
2		1.0	-10.4	-13.69	-13.72
3			1.0	-5.76	-5.8
4				1.0	0.7
5					1.0

FIGURA 3.7. El proceso de fusión analiza los traslapes entre elementos (table superior) y fusiona iterativamente las detecciones con traslape mayor a 0.5.

4. ESTIMACIÓN DE LA GEOMETRÍA EPIPOLAR DEL SISTEMA

La estimación de la geometría epipolar del sistema es el paso fundamental que permite relacionar de forma eficiente las detecciones en cada una de las vistas. Como se mencionó anteriormente, la matriz fundamental representa de forma compacta y elegante la geometría epipolar completa entre dos vistas del sistema.

A pesar de tener estas características tan deseables, la matriz fundamental no es una herramienta utilizada en demasía en sistemas estéreo emplazados en ambientes en los que no es posible tener un sistema de visión correctamente calibrado. La razón de esto es que para obtener una correcta estimación de ella, es necesario utilizar algoritmos bastante complejos, que si no son bien implementados, hacen que la velocidad del sistema disminuya mucho, descartando su uso para aplicaciones en las cuales el posicionamiento de las cámaras puede cambiar continuamente. Dadas las características de las aplicaciones del sistema a desarrollar, es necesario que la estimación de la geometría epipolar se realice de forma continua y con un tiempo de ejecución bajo, que permita captar correctamente los cambios en el posicionamiento de las cámaras.

Para poder cumplir con estos requisitos, se desarrolló un sistema de estimación no calibrado robusto a outliers, basado en el uso de detectores de características locales. Con el fin de comprender las bases del algoritmo estimador, es necesario analizar en más profundidad la expresión matemática de la restricción epipolar. Como ya fue descrito, dado un par de puntos correspondientes en dos vistas, $p = [x\ y\ 1]^T$ y $\tilde{p} = [\tilde{x}\ \tilde{y}\ 1]^T$ y la matriz fundamental $F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$ que los relaciona, se cumple que:

$$\tilde{p}^T F p = 0 \tag{4.1}$$

Expandiendo la expresión anterior, se llega a lo siguiente:

$$x\tilde{x}f_{11} + y\tilde{x}f_{12} + \tilde{x}f_{13} + x\tilde{y}f_{21} + y\tilde{y}f_{22} + \tilde{y}f_{23} + xf_{31} + yf_{32} + f_{33} = 0 \tag{4.2}$$

Definiendo los vectores $a = [x\tilde{x}\ \ y\tilde{x}\ \ \tilde{x}\ \ x\tilde{y}\ \ y\tilde{y}\ \ \tilde{y}\ \ x\ \ y\ \ 1]$ y $f = [f_{11}\ \ f_{12}\ \ f_{13}\ \ f_{21}\ \ f_{22}\ \ f_{23}\ \ f_{31}\ \ f_{32}\ \ f_{33}]^T$, la ecuación anterior puede reescribirse como:

$$af = 0 \tag{4.3}$$

Como se puede apreciar, cada par de puntos correspondientes genera una ecuación lineal. De esta forma, si se cuenta con suficientes correspondencias, es posible formar un sistema de ecuaciones lineales para obtener los valores de los componentes de la matriz F. Dado que F es una matriz homogénea, sólo son necesarias ocho y no nueve correspondencias para encontrar una solución (Luong, Deriche, Faugeras, & Papadopoulo, 1993), aunque en la práctica es deseable tener más puntos correspondientes para obtener una estimación de mejor calidad, debido al ruido existente en las mediciones. Así, definiendo la matriz $A = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}, n \geq 8$, el sistema de ecuaciones para encontrar los coeficientes de F queda definido por:

$$Af = 0 \tag{4.4}$$

En estricto rigor, son necesarias sólo 7 correspondencias para estimar la matriz fundamental, ya que $|F| = 0$. Al incluir esta restricción en el sistema anterior, este se transforma en uno no lineal, lo que dificulta su solución y se hace necesario el uso de técnicas más complejas que hacen más lento el proceso. Así, al utilizar el sistema lineal para realizar la estimación, es necesario asegurar la singularidad de F realizando algún proceso extra.

Volviendo a la estimación, la ecuación (4.4) es la que permite que la matriz fundamental sea calculable a partir de la escena y que no sean necesarias las matrices de calibración de cada una de las cámaras. Utilizando esto, el sistema se construyó a partir de la determinación de puntos correspondientes en cada una de las imágenes. Para lograr esto, se utilizaron detectores de características locales, que entregan la localización de puntos de interés unido a un vector descriptor del vecindario de este. Una vez determinados los puntos de interés en cada una de las imágenes, se realiza un proceso de matching entre ellos, para determinar las correspondencias. Teniendo estas correspondencias ya definidas, se utiliza RANSAC (Fischler & Bolles, 1981) unido al algoritmo lineal de 8 puntos normalizado (Hartley, 1997) para encontrar los valores de los coeficientes de F asegurando su singularidad. Posteriormente, y para aumentar la robustez del sistema, se realiza un proceso de estabilización temporal de la matriz fundamental para evitar errores puntuales producidos por oclusión o cambios extremos en la luminosidad. La figura 4.1 muestra el diagrama de flujo que describe los pasos que realiza el sistema estimador de F.

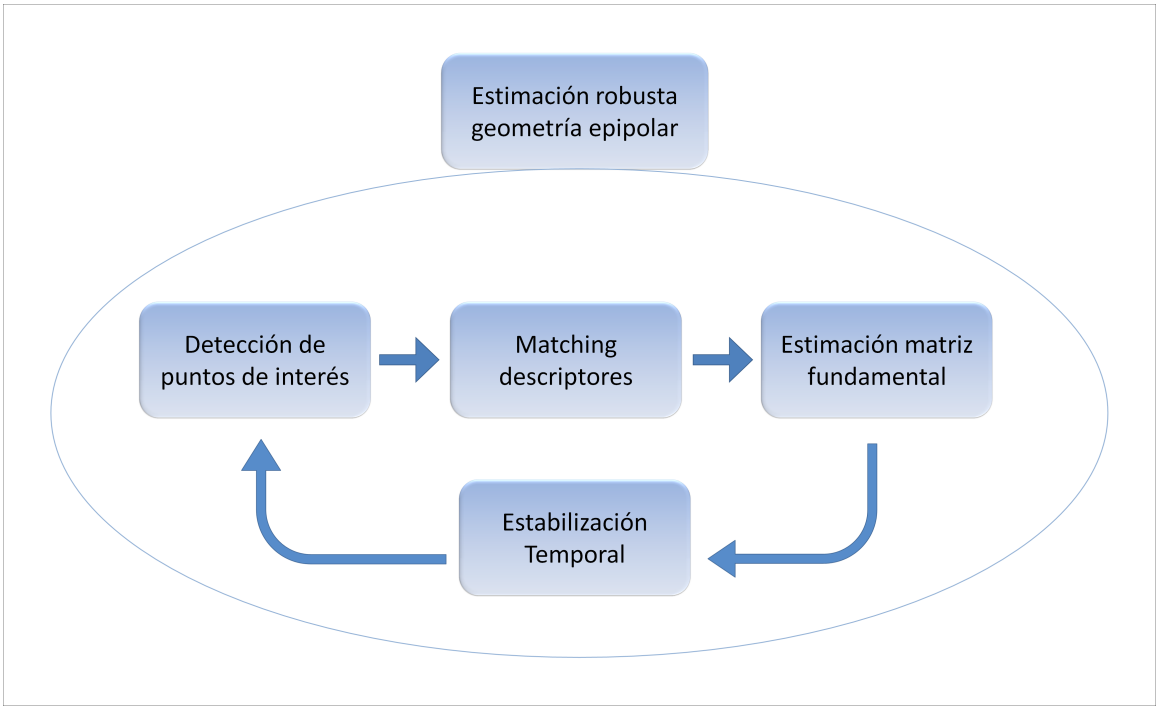


FIGURA 4.1. Diagrama de flujo del estimador de la geometría epipolar del sistema.

4.1. Establecimiento de las correspondencias entre vistas

El establecimiento de puntos correspondientes entre vistas es un problema fundamental en visión por computador (Hartley & Zisserman, 2004). Sus aplicaciones son múltiples y van desde la reconstrucción 3D hasta la realidad aumentada. Debido a esto, muchos algoritmos han sido propuestos y estudiados durante los últimos años, poniendo especial énfasis en lograr una buena descripción de cada uno de los puntos a analizar. Un algoritmo para establecer puntos correspondientes entre vistas tiene una estructura general bastante simple que consiste primero en la localización de puntos de interés en cada una de las vistas. Estas vistas pueden provenir de imágenes sincronizadas de distintas cámaras o de la misma cámara pero desde distintos puntos de vista. Luego se realiza un proceso de matching entre los puntos de interés, con el fin de encontrar las parejas que son realmente correspondientes. Las diferencias entre los distintos algoritmos radican en general en la forma que tienen de encontrar los puntos de interés y bajo qué criterio deciden si un par de estos puntos de interés son correspondientes.

El enfoque clásico para este proceso es descrito en (Zhang, Deriche, Faugeras, & Luong, 1995). En él se utiliza el detector de esquinas de Harris (Harris & Stephens, 1988)

para encontrar los puntos de interés de cada vista, para luego realizar una búsqueda exhaustiva entre ellos, comparándolos de acuerdo a la correlación existente entre sus vecindades. Los problemas de esta metodología son principalmente dos: el primero es la falta de repetibilidad del detector de Harris para ubicar los puntos de interés, debido a que no es completamente invariante a cambios de escala, de contraste y de punto de vista (Schmid, Mohr, & Bauckhage, 2000); el segundo problema es el matching realizado mediante correlación, que es muy sensible a objetos con entorno similar o a patrones reiterativos.

Para evitar estos problemas se crearon distintos algoritmos más robustos en cuanto a repetibilidad en la detección de puntos de interés y que además capturan la información del entorno de estos mediante descriptores. Estos algoritmos, llamados detectores de características locales, permiten que el antiguo problema de matching mediante correlación se transforme en un problema de comparación entre descriptores.

Para poder ser utilizados en el estimador de geometría epipolar desarrollado, era necesario que los detectores cumplieran ciertas características de invarianza mínimas, como a los cambios de iluminación (una cámara tiene una imagen más clara que la otra), de escala (diferencias en la apertura focal entre las cámaras), rotación (distinta inclinación de las cámaras) y la más importante, a cambios de punto de vista (distinta ubicación de las cámaras). Esta última es fundamental y su nivel de invarianza define de forma directa la distancia a la que se pueden poner las cámaras y por consiguiente el grado de robustez a la oclusión. Basado en los resultados de (Tuytelaars & Mikolajczyk, 2008), se evaluaron dos de los detectores más populares y con mejores resultados en cuanto a cambios de escala, iluminación y punto de vista para este tipo de aplicaciones: SIFT (Lowe, 2004) y SURF (Bay, Tuytelaars, & Gool, 2006).

El proceso de establecimiento de correspondencias comienza con la aplicación de uno de estos detectores a un par de imágenes obtenidas de dos cámaras sincronizadas. Este aspecto es fundamental para obtener buenos resultados en el sistema. Una diferencia temporal de unos pocos milisegundos entre las imágenes puede provocar errores en el posicionamiento de los puntos de interés, sobre todo en el caso de que existan objetos móviles en la escena.

Para el matching, se implementó una versión del algoritmo *nearest neighbour ratio matching* descrito en trabajo de Lowe (Lowe, 2004). Este algoritmo busca exhaustivamente para cada punto de interés en una imagen los dos puntos de interés de la otra imagen con vectores descriptores más cercanos a él, para luego calcular la razón entre las distancias de estos al descriptor del punto de la otra imagen. Para que el se registre un acierto, esta razón

debe ser menor a un umbral previamente definido. Valores muy pequeño pueden llevar a perder correspondencias correctas, mientras que valores grandes pueden llevar a introducir correspondencias falsas. El umbral fue fijado experimentalmente, siendo 0.8 el valor que mejores resultados entregó tanto para SIFT como para SURF y que además coincide con los resultados del recién citado trabajo de Lowe.

La implementación del algoritmo se hizo utilizando CUDA, lo que permitió su ejecución en tiempo real. Aunque este algoritmo utiliza una búsqueda exhaustiva, la gran cantidad de procesadores paralelos que tienen las GPU hacen que el tiempo de ejecución sea casi imperceptible. El primer paso consiste en crear en la memoria de la GPU una matriz por imagen, donde cada fila es el vector descriptor de uno de los puntos de interés de la imagen. Sean D y $\tilde{D}$ estas matrices que tendrán 64 ó 128 columnas, dependiendo si se utiliza SURF o SIFT respectivamente. Cabe destacar que los vectores descriptores tanto de SURF como de SIFT tienen norma 1, factor que resulta fundamental para el desarrollo posterior del algoritmo. El siguiente paso consiste en cuantificar la distancia entre los vectores descriptores de los distintos puntos mediante la suma de diferencias cuadráticas entre ellos, para así encontrar los dos vecinos más cercanos en la otra imagen para cada punto de interés. Sean d y $\tilde{d}$ vectores descriptores para puntos de interés cualquiera en la primera y segunda imagen respectivamente. La distancia entre estos dos vectores descriptores, $dist(d, \tilde{d})$, se obtiene de la siguiente manera

$$dist(d, \tilde{d}) = \sum_{i=0}^{n} (d_i - \tilde{d}_i)^2 = \sum_{i=0}^{n} d_i^2 + \sum_{i=0}^{n} \tilde{d}_i{}^2 - 2d \cdot \tilde{d} = 2(1 - d \cdot \tilde{d}) \qquad (4.5)$$

donde n puede ser 64 ó 128. Como se puede apreciar, el hecho que los vectores descriptores tengan norma 1 permite que el valor de las dos sumatorias de la ecuación anterior sea igual a 1, simplificando de gran manera el cálculo. Sin embargo, el efecto más grande de esta simplificación es que esta redunda en que el problema de encontrar el mínimo de la suma de diferencias cuadráticas sea análogo a encontrar los valores máximos del producto punto de los vectores descriptores d y $\tilde{d}$. Este cálculo se puede realizar de forma eficiente mediante el producto de las matrices que contienen los vectores descriptores, i.e., D y $\tilde{D}$ y buscando en la matriz resultante los valores mínimos de cada fila. Dado que en este caso las matrices son de gran tamaño, se utilizó CUBLAS, la librería de álgebra lineal incluida en CUDA, que entrega un rendimiento superior para este tipo de situaciones. Resumiendo el proceso, el primer paso consiste en calcular el producto de las matrices que contienen los vectores descriptores de los puntos de interés de cada imagen:

$$P = D\tilde{D}^T \tag{4.6}$$

A continuación, se ejecuta una búsqueda en cada fila de P para encontrar los dos valores mayores. Para realizar el proceso de manera eficiente, se utiliza un textitkernel de CUDA que recorre de forma paralela cada fila de P buscando los dos valores máximos. Sean v_{ij} y v_{ik}, $v_{ij} \geq v_{ik}$, los dos valores máximos de la fila i de la matriz P, donde v_{ij} es el valor del producto interno entre los vectores descriptores del i-ésimo punto de interés de una imagen con el el j-ésimo de la otra. Una vez encontrados, se calcula la razón:

$$R = \frac{1 - v_{ij}}{1 - v_{ik}} \tag{4.7}$$

Si esta razón es menor o igual a 0.8 existe una correspondencia entre el i-ésimo punto de interés de una imagen y el j-ésimo de la otra y se almacenan las coordenadas de ambos. En caso de ser mayor no existe correspondencia para el punto i-ésimo en la otra imagen. El resultado final de este proceso son las coordenadas homogéneas de los puntos correspondientes encontrados.

4.2. Estimación de la matriz fundamental

Una vez obtenidas las correspondencias, es posible, de acuerdo a la ecuación (4.4), obtener una estimación de la matriz fundamental a través de ellas. La metodología lógica sería entonces tomar todas correspondencias y resolver el problema utilizando el algoritmo de 8 puntos normalizado. Lamentablemente, este enfoque sólo es posible cuando todas las correspondencias, o la gran mayoría, son correctas. En la práctica este no es el caso, ya que existen muchas correspondencias que a pesar de cumplir con tener entornos muy similares, no son correctas desde el punto de vista geométrico. Casos típicos de esta situación se dan en entornos donde existen patrones repetitivos o con poca textura.

Al aumentar la distancia entre cámaras, y por consiguiente el cambio en el punto de vista, la cantidad de estas correspondencias incorrectas o outliers también crece (Bay et al., 2006). Debido a esto es necesario utilizar un algoritmo de estimación de parámetros robusto a outliers, de forma de obtener un estimación lo más correcta posible de la matriz real. RANSAC es un algoritmo ideal para este tipo de situaciones, ya que es capaz de seleccionar las muestras que cumplen el patrón real que exhiben los datos y encontrar este patrón, descartando aquellas que sólo provocan ruido. Así, el sistema estimador integra en la estructura de RANSAC al algoritmo de 8 puntos normalizado para el cálculo de F.

El algoritmo comienza su funcionamiento haciendo una estimación preliminar de la cantidad de iteraciones que debe ejecutar. La probabilidad p de elegir al menos un grupo de tamaño n libre de outliers, donde la razón de inliers de los datos es igual a w, al cabo de k intentos, se puede modelar con la siguiente expresión:

$$1 - p = (1 - w^n)^k \tag{4.8}$$

Despejando k, obtenemos una estimación para la cantidad de iteraciones necesarias:

$$k = \frac{\log(1-p)}{\log(1-w^n)} \tag{4.9}$$

Para este caso particular se utiliza $n = 8$ y para la probabilidad p un buen valor en la práctica es 0.99. Como la razón w no es conocida de antemano, el algoritmo parte asumiendo que esta toma el valor 0.5. A medida que el algoritmo se ejecuta, w se puede actualizar calculando la razón entre el tamaño del mayor conjunto de correspondencias encontrado y el tamaño total de la muestra, con lo que se puede re estimar el número de iteraciones.

Una vez obtenido el número de iteraciones, se realiza un proceso de normalización de las coordenadas de las correspondencias con el fin de hacer el cálculo de F más estable numéricamente. Esta transformación consiste en trasladar y escalar las coordenadas de los puntos de interés, por cada imagen, de tal manera que tengan posición media igual a cero y distancia promedio al origen igual a $\sqrt{2}$. Queda representada con las matrices T y $\tilde{T}$, que transforman las coordenadas de los puntos de interés de la primera y segunda imagen respectivamente.

Después de esto el algoritmo comienza a iterar. En cada iteración se selecciona aleatoriamente un conjunto de 8 correspondencias y se forma el sistema de ecuaciones (4.4). Luego se calcula una estimación preliminar de F mediante la descomposición de valor singular de la matriz A que define el sistema de ecuaciones. La solución de mínimos cuadrados para este sistema, sujeto a $\|f\| = 1$ corresponde al vector propio unitario correspondiente al valor propio más pequeño de $A^T A$ (Hartley, 1997). Este puede ser calculado realizando la descomposición de valor singular de A, $A = USV^T$, donde la última columna de V corresponde al vector solución f. Así, se logra formar $F = \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix}$, donde f_i es el

componente i-ésimo de f. Posteriormente, para forzar la singularidad de la matriz fundamental, se aplica SVD a F, $F = USV^T$. Luego, si $S = diagonal(r, s, t)$, con $r \geq s \geq t$, entonces la nueva F, ahora singular, se obtiene como $F = Udiagonal(r, s, 0)V^T$.

Una vez obtenida la estimación de F, el algoritmo recorre todas las correspondencias verificando cuales cumplen la restricción epipolar práctica, con el fin de determinar cuán bien modela la matriz estimada la relación entre las correspondencias. Para que una correspondencia definida por los puntos p y $\tilde{p}$ cumpla las restricciones del modelo estimado, se debe cumplir:

$$d = \frac{\left|\tilde{p}^T F p\right|}{\sqrt{l_1^2 + l_2^2}} + \frac{\left|p^T F^T \tilde{p}\right|}{\sqrt{\tilde{l_1}^2 + \tilde{l_2}^2}} \leq d_0 \tag{4.10}$$

Donde $l = [l_1\ l_2\ l_3]^T = Fp$ y $\tilde{l} = [\tilde{l_1}\ \tilde{l_2}\ \tilde{l_3}]^T = F^T\tilde{p}$ son las líneas epipolares generadas por p y $\tilde{p}$ respectivamente. Con esta expresión se logra forzar que la matriz fundamental calculada sean consistente en las dos imágenes. Experimentalmente se determinó que valores de d_0 entre 2 y 4 pixeles entregan buenos resultados.

En caso de que la cantidad de correspondencias fuera menor que un umbral predefinido, el algoritmo vuelve a iniciar el proceso seleccionando aleatoriamente 8 nuevas correspondencias. Experimentalmente se fijó este umbral en 16, que es un número de correspondencias que permite una estimación decente. Por otro lado, si el número de correspondencias que cumplieron la restricción es mayor que este umbral, entonces se realiza una segunda estimación de F, esta vez utilizando todas las correspondencias encontradas en el paso anterior.

A continuación se realiza nuevamente una búsqueda sobre todas las correspondencias para verificar cuantas cumplen la restricción epipolar práctica mutua (ecuación (4.10)) con esta nueva estimación más robusta de F. Si este número es el mayor que se haya registrado a lo largo de las iteraciones ya realizadas del algoritmo, entonces es almacenado junto con la matriz F calculada y se actualiza la cantidad de iteraciones restantes en base a este valor. Luego de esto verificación el algoritmo vuelve al inicio formando un nuevo grupo de 8 correspondencias.

Una vez concluidas todas las iteraciones, se denormaliza la mejor matriz fundamental calculada, utilizando los valores utilizados en la normalización de las correspondencias. Esto se realiza se la siguiente manera:

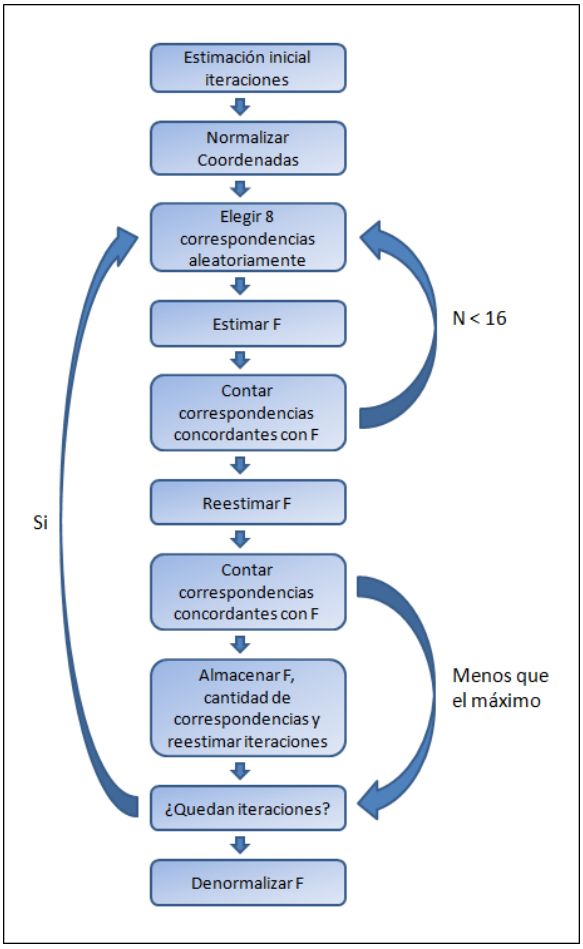


FIGURA 4.2. Diagrama de flujo del algoritmo estimador de la matriz fundamental.

$$\hat{F} = \tilde{T}^T F T \tag{4.11}$$

donde $\hat{F}$ representa la matriz denormalizada y que es finalmente entregada al estabilizador temporal. La figura 4.2 muestra un resumen de los pasos del algoritmo.

4.3. Estabilización de la estimación a través del tiempo

Para ocupar el estimador en una secuencia de video, es necesario realizar ajustes con el objetivo de que la estimación se mantenga estable a través del tiempo y refleje de forma correcta los posibles cambios en la posición de las cámaras o en los parámetros internos de estas. El comportamiento esperado es que la estimación mejore a través del tiempo,

cuando el sistema permanece estable, y que refleje los cambios lo más rápidamente posible cuando estos sean reales y no correspondan a oclusiones temporales o cambios extremos en la iluminación.

Cuando el sistema realiza la primera estimación de F, el estabilizador la almacena como la mejor. Luego, cada vez que el estimador entrega una nueva matriz fundamental, el estabilizador compara su rendimiento con el de la matriz almacenada. Esta comparación se realiza tomando las correspondencias encontradas para el par de imágenes actuales y calculando cuántas de ellas forman parte del modelo definido por cada una de las matrices. Este cálculo se realiza la misma manera que en el estimador. Luego, si una matriz es mejor que otra para el par actual de imágenes, entonces tendrá una mayor cantidad de correspondencias concordantes con el modelo que define.

Por ejemplo, en el caso de una oclusión temporal de parte importante de campo de visión de una de las cámaras, la matriz fundamental estimada a partir de estas imágenes sólo modelará de forma correcta alguno de los sectores donde existan correspondencias, por lo que será correcta sólo de manera local. En cambio, una matriz estimada sin esta oclusión modelara correctamente todo el campo de visión, con lo que con seguridad tendrá una mayor cantidad de correspondencias concordantes. Volviendo al algoritmo, en el caso que ninguna de las matrices sea notoriamente mejor que la otras, i.e., ambas tienen una cantidad similar de correspondencias, ± 5, se calcula el valor promedio de d de la expresión (4.10) para las correspondencias que formen parte de cada uno de los modelos. Aquella que tenga menor error sera almacenada como la mejor matriz.

5. SEGUIMIENTO Y CONTEO MEDIANTE VISIÓN ESTÉREO

En este capítulo se describe la etapa final del sistema, encargada de integrar los resultados del proceso de detección en cada una de la vistas mediante el uso de la estimación de la matriz fundamental y de realizar el seguimiento y conteo de las personas. Para comenzar se describe en detalle el algoritmo seguimiento utilizado y los razones para su selección. Posteriormente se detalla el proceso de fusión de información estéreo y conteo, llevados a cabo utilizando la matriz fundamental.

5.1. Seguimiento de candidatos usando CamShift

A diferencia de otras aplicaciones donde se utilizan algoritmos de seguimiento, la trayectoria de las personas cuando se dirigen o pasan a través de una entrada tiende a ser bastante simple y sin cambios bruscos, además de no presentar grandes cambios de apariencia, salvo que sea debido a oclusiones o cambios de iluminación. Como se explicará más adelante en este capítulo, las oclusiones son tratadas fusionando la información de las dos cámaras y los cambios de iluminación tienen efectos relativamente bajos debido al uso de mapas de bordes en la detección. Esto hace que los requerimientos para el algoritmo de seguimiento sean menos exigentes. Lo primordial, a parte lógicamente de tener un rendimiento aceptable en el seguimiento de personas, es que tenga un tiempo de ejecución bastante bajo, para no perder elementos que pasen muy rápido por la imagen. Dentro de la literatura existen bastantes algoritmos que cumplen con los requisitos de rendimiento, como Mean-Shift (Comaniciu & Meer, 1997), CamShift (Bradski, 1998), filtro de partículas (Isard & Blake, 1998) o el seguimiento mediante la covarianza de la región (H. Hu, Qin, Lin, & Xu, 2008). La robustez de estos enfoques radica principalmente en el modelamiento probabilístico que hacen del objeto a seguir, que les permite mantener un buen rendimiento a pesar de que el objeto cambie a través del seguimiento.

Teniendo esto en cuenta, existen dos opciones para realizar el seguimiento en tiempo real para no perder posibles objetivos. La primera es implementar una versión para GPU de algún algoritmo de seguimiento con un alto porcentaje de tareas paralelizable, como el filtro de partículas descrito en (Lozano & Otsuka, 2008). La otra opción es utilizar alguna implementación disponible públicamente de alguno de estos algoritmos, que esté optimizada para trabajar con imágenes. Convenientemente, la librería OpenCV (Bradski & Kaehler, 2008) ofrece una implementación de CamShift optimizada para procesadores multi-core e instrucciones multimedia como MMX y SSE y que utiliza el histograma de la

distribución de colores del objeto para realizar el seguimiento. Esto permite que esta implementación tome sólo un par de milisegundos por cuadro en realizar el seguimiento de un objeto. Además, esta implementación es continuamente actualizada y está optimizada para trabajar con imágenes, mientras que la integración entre la librería OpenCV y el sistema desarrollado es sumamente fácil y poco intrusiva.

A continuación se describirá brevemente el funcionamiento de CamShift y con esto las características de él que lo hacen ideal para este sistema. Luego, se describirá la integración con el detector de elementos circulares.

5.1.1. Descripción

CamShift (Bradski, 1998) es un algoritmo muy usado en visión por computador, tanto por su simpleza como por sus buenos resultados en la práctica. Este algoritmo utiliza la distribución de probabilidades del color del objeto a seguir, lo que le entrega un buen nivel de tolerancia a cambios de forma en el objeto, siempre y cuando el intervalo entre cuadros sea pequeño. Su funcionamiento es conceptualmente bastante simple y expande el funcionamiento de Mean-Shift (Comaniciu & Meer, 1997), permitiendo el seguimiento de objetos que cambian de tamaño a través del tiempo.

El primer paso del algoritmo consiste obtener un histograma de la distribución de probabilidades de la característica del objeto que dirigirá el seguimiento. Así, antes de comenzar la descripción de él, es necesario elegir la característica a seguir. Esta característica debe ser estable durante la trayectoria del objeto e idealmente robusta los cambios de iluminación. Como se mencionó antes, la trayectoria de las personas en esta aplicación es bastante estable, sin grandes cambios en la apariencia. Esto permite el uso del color como la característica directora del seguimiento, que tiene la ventaja de ser extremadamente rápida de calcular. El problema del color es buscar una representación adecuada al problema, que permita minimizar los efectos de los cambios de iluminación. Para lograr esto, se utilizó el espacio de colores HSV (Smith, 1978), que representa el color utilizando 3 valores: tono (H), saturación (S) y brillo (V). La ventaja de esta representación es que la información de la tonalidad del color esta condensada en una sola coordenada, a diferencia del espacio RGB que la distribuye entre las tres. La transformación de RGB a HSV (*OpenCV Reference Documentation*, 2006) está dada por:

$$H = \begin{cases} 0, & \text{si } MAX = MIN \\ 60 \cdot \frac{G-B}{MAX-MIN}, & \text{si } MAX = R \\ 60 \cdot \frac{B-R}{MAX-MIN} + 120, & \text{si } MAX = G \\ 60 \cdot \frac{R-G}{MAX-MIN} + 240, & \text{si } MAX = B \end{cases}$$

$$S = \begin{cases} 0, & \text{si } MAX = 0 \\ 1 - \frac{MIN}{MAX}, & \text{en otro caso} \end{cases}$$

$$V = MAX$$

donde $MAX = \max\{R, G, B\}$ y $MIN = \min\{R, G, B\}$. Tomando en cuenta la simplicidad de esta transformación, una excelente candidata para ser característica directora del seguimiento es la componente H del espacio HSV. Como se dijo antes, esta componente contiene toda la información de la tonalidad del color, lo que le permite describir de buena forma el objeto a seguir. Desde el punto de vista de esta aplicación, los elementos a seguir son bastante estables en cuanto a color a lo largo de su trayectoria ya que las personas tienden a caminar en línea recta la mayor parte del tiempo, por lo que en intervalos de tiempo y espacio pequeños, el histograma tono del sector de la cabeza que se sigue puede considerarse casi constante.

Continuando ahora con la descripción del algoritmo, una vez seleccionado el objeto a seguir, se calcula el histograma de la componente H del objetivo, H_o. El histograma a construir debe tener un tamaño adecuado para describir de manera correcta la distribución del color. Experimentalmente se determinó que entre 16 y 32 casilleros le entregan al histograma un poder descriptivo suficiente. Posteriormente, se utiliza este histograma para transformar la imagen completa en una imagen de probabilidades. Esta imagen de probabilidades se obtiene asociando el valor de cada pixel con el valor del casillero correspondiente en el histograma, para luego escalar linealmente las probabilidades del intervalo $[0, 1]$ al $[0, 255]$. De esta forma, los pixeles de esta imagen representan cuán acorde están los pixeles originales con el modelo definido por el histograma.

Una vez calculada esta imagen, se procede a realizar la búsqueda de la posición más probable del objeto modelado por H_o. La hipótesis de CamShift es que esta posición corresponde a la media de la distribución definida por esta imagen. Para encontrarla, utiliza las iteraciones de Mean-Shift, que consisten en un ascenso progresivo de la distribución en

la dirección del gradiente. Como todo ascenso del gradiente, es posible que la búsqueda se quede estancada en un máximo local. Para disminuir este efecto, CamShift utiliza ventanas de búsqueda algo más grandes que el tamaño del objeto a seguir, moviéndolas y adecuando su tamaño progresivamente a medida que se ejecutan las iteraciones de Mean-Shift. Teniendo esto en cuenta, el proceso iterativo de búsqueda del centroide de la distribución utilizado por CamShift se define de la siguiente manera:

(i) Elegir un tamaño y ubicación inicial para la ventana de búsqueda.

(ii) Calcular el centroide de la distribución en la ventana. Dada una distribución de probabilidades bidiminensional, $I_p(x, y)$, el centroide de esta puede encontrarse calculando el momento de orden cero y los momentos de orden uno para x e y, definidos como:

$$M_{00} = \sum_x \sum_y I_p(x, y),\ M_{10} = \sum_x \sum_y x I_p(x, y),\ M_{01} = \sum_x \sum_y y I_p(x, y)$$

Una vez calculados estos datos, las coordenadas x e y del centroide C de la distribución en la ventana de búsqueda queda definido por:

$$C_x = \frac{M_{10}}{M_{00}},\ C_y = \frac{M_{01}}{M_{00}}$$

(iii) Centrar la ventana de búsqueda en la posición del centroide calculado en el paso anterior.

(iv) Ajustar el tamaño de la ventana de búsqueda en función de M_{00}:

$$S = 2\sqrt{\frac{M_{00}}{256}}$$

Donde S corresponde al nuevo alto y ancho de la ventana de búsqueda.

(v) Repetir hasta alcanzar la convergencia (la posición del centroide se mueve menos que un umbral pre definido).

(vi) Modelar la forma del objeto mediante una elipse calculada a través de la distribución. Dados $M_{20} = \sum_x \sum_y x^2 I_p(x, y)$ y $M_{02} = \sum_x \sum_y y^2 I_p(x, y)$, los momentos de segundo orden para x e y respectivamente y el momento compuesto $M_{11} = \sum_x \sum_y xy I_p(x, y)$, los parámetros de la elipse se definen como:

$$a = \frac{M_{20}}{M_{00}} - C_x^2,\ b = 2\left(\frac{M_{11}}{M_{00}} - C_x C_y\right),\ c = \frac{M_{02}}{M_{00}} - C_y^2$$

Luego la orientación θ y las longitudes de los ejes mayor y menor de la elipse, e_M y e_m, se definen como:

$$\theta = \frac{\arctan\left(\frac{b}{a-c}\right)}{2}$$

$$e_M = \sqrt{\frac{a+c+\sqrt{b^2+(a-c)^2}}{2}}$$

$$e_m = \sqrt{\frac{a+c-\sqrt{b^2+(a-c)^2}}{2}}$$

5.1.2. Integración

El proceso de integración con el detector de elementos circulares es bastante directo, aunque tiene una serie de pasos necesarios para asegurar el correcto comportamiento del sistema. El primero de estos consiste en crear la estructura que representará los elementos circulares a seguir. Con este fin de se utilizó una estructura de datos llamada "objetivo de seguimiento monocular" que contiene toda la información del elemento circular junto con otros 4 datos importantes para el seguimiento: (i) la cantidad de cuadros total en que el objetivo a sido seguido, C_t, (ii) la cantidad de cuadros consecutivos en que el objetivo a estado detenido o con movimiento apenas perceptible, C_d, (iii) el histograma normalizado de la componente H, de su transformación al espacio de colores HSV, en la región dada por la circunferencia representada por el elemento circular y (iv) su trayectoria T, definida como $T = \bigcup_i T_i$, donde cada $T_i = (x_i, y_i)$ almacena la posición del objeto en el cuadro i, con $i \in [1, C_t]$.

El resto de los pasos necesarios para la integración son presentados en el proceso de seguimiento monocular a continuación:

(i) Actualización: En caso de existir objetivos monoculares a seguir, se aplica CamShift a cada uno de ellos, actualizando con esto sus posiciones, tamaños y datos de seguimiento.

(ii) Fusión de objetivos de seguimiento: Se analiza la posición de cada objetivo de seguimiento y se fusionan aquellos que tengan un nivel de traslape igual o mayor a 0.5, basado en la ecuación (3.7).

(iii) Eliminación: Se evalúa la permanencia de cada objetivo en base a su tiempo de seguimiento, su tiempo sin movimiento y su trayectoria. Son eliminados aquellos que lleven mucho tiempo en la lista, mucho tiempo sin movimiento o que

tengan trayectorias erráticas o inestables. Este último criterio elimina aquellos objetos que hayan recorrido una distancia muy grande entre cuadros consecutivos o han tenido cambios muy bruscos en su velocidad. Todos estos criterios trabajan en base a umbrales predefinidos.

(iv) Detección y fusión: Se realiza el proceso de detección de elementos circulares en la imagen y luego se comparan las posiciones de estos con las de los objetivos monoculares almacenados. Aquellos objetivos cuyo traslape con alguna detección sea mayor o igual a 0.5, nuevamente basado en la ecuación (3.7), son actualizados promediando su posición y tamaño con los del elemento circular detectado.

(v) Creación de nuevos objetivos: Aquellas detecciones que no coinciden en traslape con ningún objetivo son transformadas en nuevos objetivos de seguimiento monocular, calculando la información necesaria para el seguimiento descrita anteriormente. Luego, son agregadas a la lista de objetivos para ser procesadas a partir del próximo cuadro.

Como se puede apreciar, las "medidas de seguridad" definidas en los items (ii), (iii) y (iv) sirven para eliminar y/o corregir los falsos positivos que hacen que el seguimiento de CamShift pueda divergir o que provienen de detecciones erróneas.

5.2. Conteo de personas basado en información estéreo

Para realizar el conteo integrando la información de las dos vistas en el sistema, se expande el proceso de seguimiento monocular con una etapa de conteo para luego mezclar la información de las detecciones en las dos vistas. La mezcla se basa en el uso de la matriz fundamental, utilizando el sistema estimación descrito en el Capítulo 4. Además, es necesario definir una línea virtual de conteo en cada una de las vistas. Estas líneas deben ser definidas manualmente antes de iniciar el proceso y ser correspondientes entre sí. La ubicación de estas líneas virtuales no es un elemento que afecte el rendimiento del sistema por lo que pueden estar posicionadas en cualquier sector de las imágenes. De esta forma se requiere una mínima intervención humana, sólo para asegurar la correspondencia entre ellas. Para definir una línea de conteo se requiere especificar sus puntos de inicio y fin en coordenadas cartesianas.

Para llevar un seguimiento de los datos del proceso, el modulo de conteo estéreo utiliza tres contadores: pasadas totales por la línea de conteo, P_t, pasadas negativas, P_- y pasadas positivas, P_+. Estos últimos dos contadores representan los pasos por la línea de conteo en

una u otra dirección. En todo momento del proceso se cumple que $P_t = P_+ + P_-$, por lo que sólo es necesario actualizar los valores de P_+ y P_-. Además de esto, para poder realizar el conteo, los objetivos de seguimiento monoculares son aumentados para incluir 3 nuevos campos: (i) la cantidad de cuadros transcurridos desde la última vez que fueron contados, C_c, (ii) el sector de la imagen en que se encuentra, L, con respecto a la línea de conteo que la divide en dos, y (iii) una variable booleana que indica si el objetivo está pareado con su objetivo correspondiente en la otra vista, P. Estas valores son inicializados en -1, 0 y falso respectivamente al momento de crear el objetivo en el paso (v) del seguimiento monocular.

El proceso de conteo estéreo utiliza además el primer cuadro de cada cámara para inicializarse. Esto consiste en activar el modulo estimador de geometría epipolar con el fin de obtener la primera estimación de la matriz fundamental F. Esto es de vital importancia, ya que todo el proceso de conteo se basa en la existencia de esta estimación. En caso contrario no sería posible relacionar los resultados de cada vista. Una vez realizada esta primera estimación, el módulo es llevado a segundo plano para que realice la estimación de forma constante sin interrumpir el resto de los procesos. A continuación y para el resto de los cuadros, se realiza el proceso de conteo estéreo en sí, que se divide en tres etapas: (i) seguimiento y conteo monocular, (ii) creación de objetivos de seguimiento estéreo y (iii) conteo estéreo.

La primera etapa involucra el proceso de seguimiento monocular descrito anteriormente, con algunos elementos extra que se introducen para poder realizar el conteo. El primer paso de la etapa consiste en ejecutar el proceso de seguimiento en cada una de las vistas del sistema, para luego analizar la trayectoria de cada uno de los objetivos monoculares. Sea i el cuadro actual, $i \geq 2$ y sea O_m un objetivo monocular cualquiera en una de las vistas con $C_t >= 2$ y $C_c = -1$. Estas dos condiciones son necesarias para que un objetivo pueda ser ingresado al proceso de conteo. Dado que la trayectoria completa del objetivo se encuentra almacenada en su campo T, se verifica si en el camino recorrido entre T_{i-1} y T_i se produjo un cruce por la línea de conteo. Para realizar esto se convierte la línea de conteo correspondiente a esa vista a una ecuación de la recta en su forma general. Sean $p = (p_x, p_y)$ y $q = (q_x, q_y)$ los puntos que definen la línea de conteo, la ecuación de general de la recta que pasa por estos puntos se define en sus componente A, B y C como:

$$A = -1,\ B = \left(\frac{p_x - q_x}{p_y - q_y}\right),\ C = \left(p_x - \frac{p_x - q_x}{p_y - q_y} p_y\right) \tag{5.1}$$

Al evaluar esta ecuación en los puntos definidos en T_i y T_{i-1} es posible determinar si hubo un cruce de la recta cuando el signo de estas evaluaciones es distintos. Más específicamente, un cruce ocurre cuando:

$$sgn\left(Ax_{i-1} + By_{i-1} + C\right) \neq sgn\left(Ax_i + By_i + C\right) \tag{5.2}$$

Donde sgn es la función signo definida como:

$$sgn(x) = \begin{cases} -1, & \text{si } x < 0 \\ 0, & \text{si } x = 0 \\ 1, & \text{si } x > 0 \end{cases} \tag{5.3}$$

Luego, para aceptar este cruce como conteo, es necesario verificar si este se realizó en el sector de la recta definida por el usuario como línea de conteo. Para esto se calcula el intercepto entre la recta definida por T_{i-1} y T_i y la recta que representa la línea de conteo y si este se ubica sobre la línea definida por el usuario, entonces el cruce representa un conteo válido. Más formalmente, sea $j = (j_x, j_y)$ el intercepto, el cruce es considerado como válido si y sólo si se cumple:

$$((p_x \leq j_x \leq q_x) \vee (q_x \leq j_x \leq p_x)) \wedge ((p_y \leq j_y \leq q_y) \vee (q_y \leq j_y \leq p_y)) \tag{5.4}$$

Finalmente se fija $L = sgn\left(Ax_i + By_i + C\right)$ para todos los objetivos de cada vista y $C_c = 0$ sólo para aquellos que hayan realizado un cruce válido. Además, cualquier cruce que se realice posteriormente por el mismo sector de la línea de conteo será descartado por cierta cantidad de tiempo, con el fin de eliminar conteos múltiples del mismo objetivo.

La segunda etapa consiste en la fusión de la información estéreo, utilizando los objetivos de seguimiento monocular. Por cada uno de estos objetivos de cada vista se busca su elemento correspondiente en la otra, para así formar un "objetivo de seguimiento estéreo", el cual será posteriormente analizado por la etapa de conteo. Existen tres casos para los objetivos estéreo. El primer caso consiste en un objetivo estéreo completo, definido como $S_C = \left(M, \tilde{M}\right)$, donde S_c está formado por dos objetivos monoculares correspondientes, M y $\tilde{M}$, uno en cada vista. El segundo caso es un objetivo incompleto tipo 1, $S_{I1} = \left(\emptyset, \tilde{M}\right)$, en el cual no fue posible parear el objetivo monocular $\tilde{M}$ de la vista 2 con uno de la vista 1. El tercer caso es llamado objetivo incompleto tipo 2, $S_{I2} = (M, \emptyset)$, y

es análogo a S_{I1}, pero con la incompletitud proveniente ahora de la vista 2. La formación de estos tres tipos de objetivo es la etapa crucial del sistema, que le permite aprovechar la información entregada por la matriz fundamental para aumentar su robustez frente a las oclusiones.

El proceso se realiza de manera idéntica para cada objetivo monocular de cada vista, con la salvedad del "sentido" de la matriz fundamental. De esta forma, aquí se explicará el caso genérico nombrando a las dos vistas del sistema como V_1 y V_2, donde $F_{V_1V_2}$ es la matriz fundamental que va desde V_1 a V_2 y $F_{V_2V_1} = F_{V_1V_2}^T$ la que va en el sentido contrario. Sea O_{V_1} un objetivo de seguimiento monocular cualquiera en la vista V_1, y c_1 un vector columna que representa su centroide en coordenadas homogéneas. El proceso se inicia verificando si O_{V_1} está pareado o no, i.e., si ya forma parte de un objetivo de seguimiento estéreo completo. En caso de que sea, termina el análisis para este objetivo y se continúa con el siguiente en V_1. En caso contrario, se busca en los objetivos de seguimiento de la vista V_2 aquellos que no hayan sido pareados y cumplan la restricción epipolar práctica generada por $F_{V_1V_2}$. Sea O_{V_2} un objetivo cualquiera en V_2, c_2 un vector columna que representa su centroide en coordenadas homogéneas y r el radio del elemento circular que representa. Luego, la restricción epipolar práctica para el seguimiento estéreo queda definida como:

$$\frac{\left|c_2^T F_{V_1V_2} c_1\right|}{\sqrt{l_1^2 + l_2^2}} \leq r \tag{5.5}$$

Donde $l = [l_1\ l_2\ l_3]^T = F_{V_1V_2}c$ es la línea epipolar generada por el centroide de O_{V_1}. Como se puede apreciar, se ocupa como umbral el radio del elemento circular que representa el objetivo O_{V_2}. Esta modificación a la restricción epipolar práctica definida en (2.18) permite manejar de mejor manera los errores que puedan darse debido a un posicionamiento incorrecto del centroide del objetivo.

Dado que la restricción epipolar reduce el espacio de búsqueda a una línea y no a una ubicación en particular, es muy común que varios elementos la cumplan sin que estos sean realmente correspondientes. De esta manera es necesario elegir entre ellos aquel que presente un gran nivel de similitud con O_{V1}. Aprovechando el hecho de que la apariencia de cada objetivo ya es modelada utilizando un histograma, la similitud puede ser calculada mediante cualquier métrica de comparación de distribuciones, como correlación, Earth Mover's Distance (Rubner, Tomasi, & Guibas, 1998) o la distancia de Bhattacharyya (Djouadi, Snorrason, & Garber, 1990). Esta última es usada ampliamente y con excelentes

resultados en la detección y clasificación de objetos y fue la que mejores resultados entregó en la práctica. La distancia de Bhattacharyya entre los histogramas H_1 y H_2 se define como:

$$d_B(H_1, H_2) = \sqrt{1 - \sum_{k}^{n} \sqrt{H_1(k) \cdot H_2(k)}} \tag{5.6}$$

Donde n es la cantidad de casilleros del histograma, $H_i(k)$ es el valor del histograma i en el casillero k y $d_B \in [0, 1]$. Continuando con el proceso, para cada objetivo en V_2 que haya cumplido (5.5) se compara su histograma con el de O_{V_1} utilizando (5.6) y se selecciona aquel que tenga el valor mínimo de d_B y menor a 0.5. Sea $\tilde{O}_{V_2}$ este objetivo, en caso de existir. El siguiente paso consiste primero en formar un nuevo objetivo de seguimiento estéreo completo utilizando los objetivos monoculares y luego marcar ambos objetivos monoculares como pareados, para que no puedan ser utilizados por otro objetivo monocular para formar un objetivo estéreo. En caso de no existir $\tilde{O}_{V_2}$, O_{V_1} no es marcado como pareado y se continúa el proceso con el siguiente objetivo de la vista correspondiente. Una vez terminado este proceso para todos los objetivos de ambas vistas, se crean objetivos estéreo tipo 1 ó 2 utilizando los objetivos monoculares de ambas vistas que no hayan sido marcados como pareados.

Falta mencionar un sólo detalle con relación al paso anterior del proceso. En el seguimiento monocular existe un paso en el que se eliminan objetivos de acuerdo a ciertos criterios. En este caso, al momento de eliminar un objetivo se verifica si está pareado o no. En caso de estarlo, el valor de P del objetivo correspondiente al objetivo eliminado es fijado en falso. En caso contrario, la eliminación se realiza tal cual como antes.

La tercera etapa del proceso, conteo estéreo, se realiza verificando los cruces realizados por los objetivos de seguimiento estéreo. La idea central es realizar el conteo de manera robusta a la oclusión, aplicando distintos criterios en base a la cantidad de objetivos monoculares en el objetivo estéreo, cuántos de estos han cruzado la línea de conteo y en que momento lo hicieron.

El proceso de conteo estéreo comienza con el análisis de los objetivos estéreo creados en el paso anterior. Este análisis es dependiente del tipo del objetivo estéreo, i.e., completo o incompleto. Sea S un objeto cualquiera de los obtenidos en el paso (ii). Para el caso en que S sea completo, i.e., $S = \left(O_m, \tilde{O}_m\right)$, existen cuatro casos posibles a analizar en

base al contador de cuadros desde el último conteo de cada objetivo monocular. Sea C_c el contador de O_m y $\tilde{C}_c$ el contador de $\tilde{O}_m$. Los cuatro casos son los siguientes:

(i) $C_c > 0 \vee \tilde{C}_c > 0$: Este caso involucra al menos un objetivo de S que ya ha realizado un cruce anteriormente por la línea de conteo. Esto implica que ese objetivo no debe ser contado nuevamente. De esta forma, sin importar el valor de C_c para el otro objetivo monocular, este caso no generará un conteo, por lo que tanto P_+ como P_- se mantienen igual. Esta política permite por ejemplo eliminar los conteos múltiples del mismo objeto que podrían generarse en los casos $C_c > 0$, $\tilde{C}_c = 0$ y $C_c = 0$, $\tilde{C}_c > 0$.

(ii) $C_c = 0 \wedge \tilde{C}_c = 0$: En este caso tanto O_m como $\tilde{O}_m$ han realizado un cruce por la línea de conteo en el último proceso de actualización, por lo que se aumenta el contador de pasadas dependiendo del sentido del cruce. Para hacerlo, se utiliza el valor L almacenado en los objetivos monoculares. Si $L > 0$ entonces $P_+ = P_+ + 1$. Por otro lado, si $L < 0$ entonces $P_- = P_- + 1$. Cabe destacar que el valor de L es igual para ambos objetivos. Este es el caso ideal de conteo, en el cual el mismo objeto pudo ser seguido sin problemas a través de su paso por la línea virtual en ambas vistas.

(iii) $\Big(C_c = 0 \wedge \tilde{C}_c = -1\Big) \vee \Big(C_c = -1 \wedge \tilde{C}_c = 0\Big)$: En este caso, uno de los objetivos realizó un cruce válido como conteo en la última actualización mientras que el otro aún no ha realizado un cruce. Esta situación aparentemente anómala se da principalmente cuando las líneas de conteo de ambas imágenes no son perfectamente correspondientes, por lo que puede ocurrir que uno de los dos objetivos cruce la línea un par de cuadros antes que el otro. Para manejar este caso, se realiza el mismo proceso que en el punto anterior, es decir, se aumenta P_+ o P_- dependiendo del valor de L, sólo que en esta ocasión basándose en únicamente en el objetivo con $C_c = 0$. Este manejo permite que el cruce válido sea registrado correctamente, mientras que el próximo posible cruce del otro objetivo es manejado en el punto (i), con lo que se elimina la posibilidad de conteos múltiples del mismo objetivo.

(iv) $C_c = -1 \wedge \tilde{C}_c = -1$: Este caso involucra a dos objetivos monoculares que aún no han realizado un cruce válido por la línea de conteo. Este es el caso más común y como cabe de esperar no genera cambios tanto en P_+ como en P_-.

Como se puede apreciar, existe claramente una jerarquía en cuanto a la importancia de los valores de C_c para realizar el conteo de un objetivo estéreo. En el nivel más alto

tenemos el caso $C_c > 0$, expuesto en el punto (i), que tiene prevalencia por sobre todos los otros valores. Así, cada vez que en un objetivo estéreo exista un objetivo monocular que tenga su contador en este intervalo, sin importar el valor del otro contador el objetivo no será contado. Un nivel más abajo tenemos $C_c = 0$, puntos (ii) y (iii), que está asociado a un conteo válido y finalmente $C_c = -1$, punto (iv), que se asocia con un objetivo que no ha realizado cruce.

Finalmente, para el caso en que S sea un objetivo incompleto tipo I ó II, existen solamente dos casos, también basados en C_c:

(i) $C_c = 0$: En este caso se registra un conteo válido en el objetivo por lo que se actualiza P_+ o P_- dependiendo del valor de L.

(ii) $C_c \neq 0$: Este casi implica un objetivo que ya realizó un cruce o que aún no lo ha realizado, por lo que no conlleva una actualización en los contadores de pasadas por la línea de conteo.

Una vez concluido el proceso de conteo se itera una vez más por todos los objetivos de seguimiento de cada una de las vistas, aumentando en uno el contador C_c de cada uno de ellos, excepto en el caso que $C_c = -1$, donde el contador es dejado igual. Además, en el caso que C_c sea mayor que un umbral pre definido, se reinicializa su valor en -1 con el fin de que el objetivo pueda ser nuevamente contado en caso de que realice un nuevo cruce. El valor de este umbral no es crítico y los resultados no presentan una gran sensibilidad a sus variaciones. El valor utilizado en el sistema fue 50.

6. PRUEBAS Y ANÁLISIS DE RESULTADOS

Las pruebas se llevaron a cabo en una máquina con procesador Intel Core 2 Quad de 2.4GHz de cuatro núcleos y 4GB de memoria RAM. La GPU del sistema es una Nvidia Geforce 9800GTX+ con 768MB RAM. Esta GPU posee 128 núcleos de procesamiento independientes. El sistema operativo fue Microsoft Windows XP SP3 de 32 bits. El entorno de desarrollo utilizado fue Microsoft Visual Studio 2005 SP1 mientras que para el desarrollo del código para la GPU se utilizó Nvidia CUDA versión 2.0.

Todas las secuencias e imágenes fueron adquiridas mediante cámaras Flea2 de Point Grey Research (PointGreyResearch, 2008) conectadas mediante una interfaz IEEE 1394b. Para las imágenes y secuencias estéreo, estas cámaras fueron sincronizadas con una desviación máxima de $125\mu s$.

6.1. Detección de cabezas

Para probar el rendimiento del detector se realizó el esquema estándar evaluación, que consiste en seleccionar una serie de imágenes que contienen, en este caso, personas en escenarios reales y desde distintos puntos de vista, y luego evaluar su tasa de aciertos sobre ellas. Cada imagen contiene al menos una persona y fueron capturadas por medio propios. La base de datos total consta de 32 imágenes que contienen un total de 53 cabezas con radios entre 17 y 30 pixeles.

La figura 6.1 presenta imágenes de cabezas en cada una de las tres disposiciones posibles (frontal, posterior, lateral).

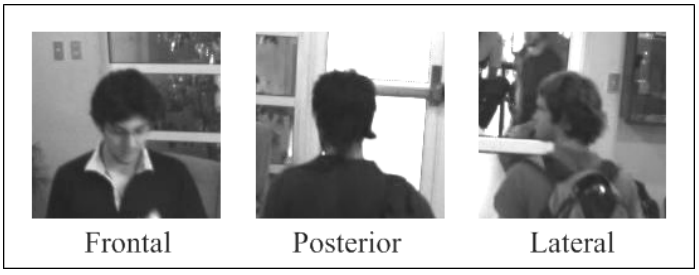


FIGURA 6.1. Imágenes de las tres orientaciones posibles para las cabezas.

La disposición lateral de una cabeza se definió como una en la cual sea visible sólo uno de los lados del rostro, sin importar si este se ve completamente. La disposición frontal ocurre cuando ambos lados de él son visibles, no necesariamente ambos de manera completa, y finalmente la disposición posterior se produce cuando ninguna parte del rostro es

visible. A modo de resumen, la tabla 6.1 presenta la cantidad de cabezas en las imágenes y su disposición en ellas.

TABLA 6.1. Resumen de la cantidad y disposición de las cabezas en las imágenes.

Frontal	Posterior	Lateral	Total Cabezas	Total Imágenes
13	26	13	53	32

Para evaluar el rendimiento se utilizan los coeficientes de la matriz de confusión de la siguiente manera. Sea M la matriz de confusión dada por:

$$M = \begin{bmatrix} TP & FP \\ TN & FN \end{bmatrix} \tag{6.1}$$

Donde TP indica la cantidad de detecciones correctas, FP la cantidad de detecciones incorrectas y FN la cantidad de objetivos no detectados. El coeficiente TN indica la cantidad de elementos correctamente no detectados, que en este escenario no tiene sentido y no es tomado en cuenta. Cabe destacar que $TP + FP$ es la cantidad total de detecciones realizadas por el sistema y $TP + FN$ representan la cantidad real de objetos que el sistema debió haber detectado. Luego, en base a TP, FP y FN se definen las medidas estadísticas precisión (P) y completitud (C), o en inglés *precision* y *recall*, de la siguiente manera:

$$P = \frac{TP}{TP + FP} \qquad C = \frac{TP}{TP + FN} \tag{6.2}$$

La precisión mide la exactitud del proceso, i.e., que tan correctas son las detecciones realizadas. Alcanza un valor máximo de 1 cuando sólo se realizaron detecciones correctas, mientras que su valor mínimo de 0 es alcanzado cuando todas las detecciones realizadas son incorrectas o cuando no se ha detectado nada. Por otro lado, la completitud mide la completitud del proceso, i.e., que cantidad de las objetivos totales fueron detectados. Al igual que la precisión, el valor de la completitud se mueve en el intervalo $[0, 1]$, donde 1 es alcanzado cuando todos los objetivos fueron detectados y 0 cuando ningún objetivo fue detectado.

Con el fin de utilizar una única medida estadística para describir el rendimiento del sistema, se define además la métrica F_β (VanRijsbergen, 1979), que pondera los valores de P y C de acuerdo al coeficiente real positivo β.

$$F_\beta = (1 + \beta^2) \cdot \frac{P \cdot C}{\beta^2 \cdot P + C} \tag{6.3}$$

Un valor de β igual a cero significa que la completitud no tiene ninguna importancia para la evaluación, mientras que para $\beta = \infty$ la precisión es la que no tiene importancia. El caso más utilizado en la práctica es $\beta = 1$, el cual le asigna la misma importancia tanto a la precisión como a la completitud. Esta métrica se mueve en el intervalo $[0,\ 1]$.

6.1.1. Resultados y análisis de sensibilidad

Para todas las pruebas se utilizaron filtros con valores $s_i = -1$ y $b = 1$, i.e., los coeficientes del sector interno son iguales a -1 y los del borde iguales a 1. En la práctica estos valores fueron los que mejores resultados entregaron, además de ser más claros desde un punto de vista conceptual.

Con el fin de obtener los mejores parámetros para el funcionamiento del detector y estudiar su nivel de sensibilidad, se realizó un análisis de sus resultados en las imágenes de la base de datos, utilizando distintos valores para el umbral de aceptación u_a, definido en la ecuación (3.6). Dado que se conoce el valor del radio del filtro base y su valor de borde, la modificación de este umbral se realiza variando el nivel de tolerancia figuras completas t. Esto entrega una base lógica para comprender la variación del rendimiento del detector en base a la modificación del umbral.

Luego, al modificar progresivamente el nivel de tolerancia, se logra que los índices TP, FP y FN varíen, con lo que se puede construir un gráfico completitud vs precisión. A partir de esta visualización es posible estimar el mejor valor la tolerancia, dadas las necesidades de detección del sistema. La figura 6.2 muestra los resultados.

Como se puede apreciar, el detector genera una gran cantidad de falsos positivos lo que redunda en un precisión bastante baja para valores de completitud sobre 0.9. En una aplicación de detección estática de cabezas o de objetos en general, esta situación sería bastante problemática, pero en este caso no genera problemas mayores ya que aproximadamente el 95% de estos falsos positivos corresponden a objetos estáticos del fondo, que en ningún momento logran cruzar la línea de conteo. Además, estas detecciones erróneas estáticas son rápidamente eliminadas por el módulo de seguimiento.

Continuando con el objetivo principal del análisis, para determinar el valor óptimo para la tolerancia, es necesario definir un criterio sobre el comportamiento deseado para el detector. En este caso, es importante que el nivel de completitud sea lo más alto posible, ya que un bajo nivel de esta implica que el detector pierde muchas cabezas, lo que redunda posteriormente en la no generación de objetivos de seguimiento correctos. Por otro lado,

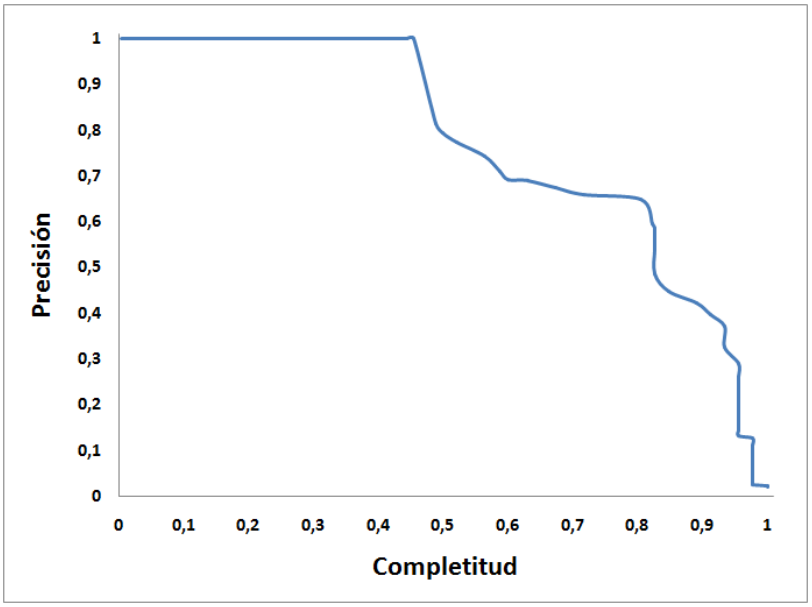


FIGURA 6.2. La curva del gráfico completitud vs. precisión muestra un comportamiento no ascendente, tradicional para este tipo de aplicaciones. Se aprecia un alto nivel de precisión hasta una completitud aproximada de de 0.45, seguida de un rápido descenso. Por otro lado, la completitud nunca disminuye de 0.45, presentado valores superiores a 0.8 para gran parte de los valores de la precisión.

una baja precisión puede ser manejada hasta cierto punto mediante el módulo de seguimiento, como fue expuesto anteriormente, por lo que es preferible darle mayor importancia en el criterio de selección de la tolerancia a la completitud. De esta manera, y basándose en la información obtenido a través del gráfico, un criterio acorde con las necesidades de la aplicación es seleccionar una tolerancia que entregue el mayor nivel de precisión dada una cierta completitud mínima deseada, idealmente superior al 90% en este caso.

Para cuantificar correctamente este criterio, se utiliza la métrica F_β, con algún valor de β que represente la mayor importancia de la completitud para el sistema. Determinar este valor es bastante complejo, ya que no hay una manera de cuantificar la mayor importancia de la completitud sobre la precisión, sólo sabemos que lo es. Así, para poder obtener un nivel de tolerancia, se probaron valores de β entre 2 y 5, obteniendo siempre la misma tolerancia para el valor máximo. La figura 6.3 presenta un gráfico t vs. F_β, con $\beta = 4$ que permite apreciar una manera clara la evolución del rendimiento del detector a medida que se modifica la tolerancia y encontrar el valor óptimo para ella. Se utilizó $\beta = 4$ porque resultó ser un valor bastante representativo para la situación de todos los valores probados.

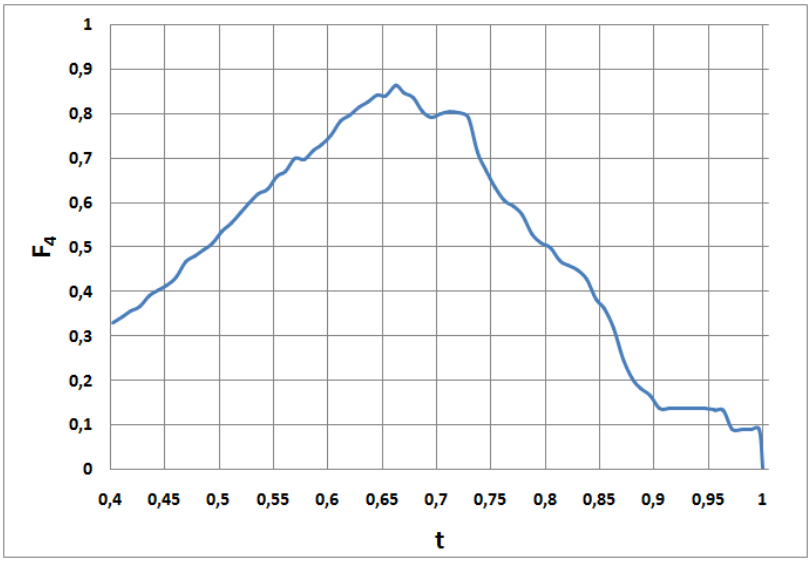


FIGURA 6.3. La curva del gráfico t vs. F_4 muestra un comportamiento bastante acorde con lo esperado, con una distribución que tiene valores más altos cuando la completitud es más alta, lo que es generado con valores de t más bajos. Esta situación se repite para valores de β pertenecientes al intervalo $[2, 5]$.

Como se puede apreciar en la figura, el gráfico tiene un máximo muy claro en $t = 0.66$, con el que se obtiene un puntaje de 0.87, precisión de 0.40 y una completitud de 0.93. Esto significa que, la mejor caracterización de una cabeza, bajo este criterio, es cuando se exige que esté al menos completa en un 66%, algo que parece bastante intuitivo y natural. Cabe destacar que el valor de F_4, para tolerancias entre 0.61 y 0.73, no varía notoriamente y se mantiene sobre el 80%, lo que significa que el detector presenta una baja sensibilidad a la modificación de este parámetro, mientras se mantenga en ese intervalo. Esto también suena bastante intuitivo, ya que no debería existir una gran diferencia entre una cabeza completa en un 61% y una en un 73%. Además, el criterio obtenido a partir de los datos parece tener bastante relación con el requisito que se le había impuesto, i.e., una completitud idealmente superior al 90%.

La obtención de estos valores es sumamente importante, ya que entregan un idea preliminar del rendimiento que podría llegar a obtener el sistema de conteo, asumiendo el correcto funcionamiento del resto de sus módulos. Para el caso de la precisión, se sabe que el sistema de conteo sólo contabilizará los cruces por la línea virtual, lo que implica que las detecciones de elementos estáticos del fondo no generarían una disminución en la precisión del conteo. Como fue anteriormente descrito, para el umbral elegido, aproximadamente el 95% de los falsos positivos corresponden a elementos del fondo, lo que significa que, al no

contar como errores para el conteo, la precisión de este podría llegar hipotéticamente hasta 0.94 aproximadamente.

Desde el punto de vista de la completitud, el valor aquí obtenido es una estimación bastante adecuada para la completitud del sistema de conteo, ya que se espera que el módulo de estimación de geometría epipolar permita asociar correctamente detecciones en las distintas vistas. Al igual que con la precisión, esta predicción está supeditada el buen funcionamiento del módulo de seguimiento.

Volviendo a la búsqueda de la tolerancia óptima para el detector, desde el punto de vista estrictamente matemático este es aquel que genera el punto que se encuentra a menor distancia del $(1,\ 1)$ en el gráfico completitud vs precisión. Luego, al graficar la función $f(t) = \sqrt{(1 - P(t))^2 + (1 - C(t))^2}$ para todas las tolerancias evaluadas, se obtiene que el mínimo es generado por una tolerancia de 0.73, con el que se obtiene una distancia al óptimo de 0.4, una precisión de 0.65 y una completitud de 0.81. Cabe destacar que 0.73 es el mayor valor para el cuál F_4 entregaba un rendimiento estable del detector. La figura 6.4 presenta los resultados.

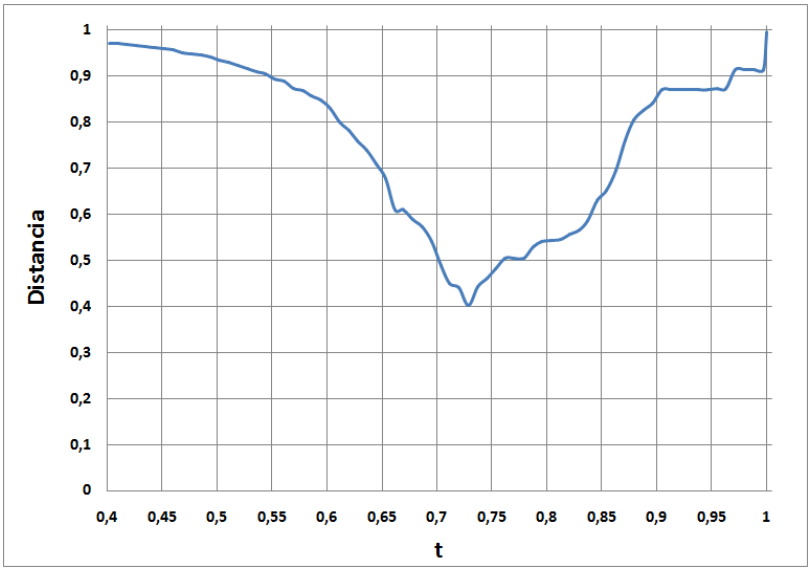


FIGURA 6.4. La distancia al óptimo muestra un valor mínimo bastante marcado en $t = 0.73$, valor que es cercano al óptimo encontrado en el gráfico t vs. F_4. A pesar de esta similitud, el comportamiento en la vecindad del máximo es menos estable que en la anterior configuración.

Para confirmar este resultado, se realizó además un tercer gráfico de evaluación, u_a vs. F_1, con el fin de asignar igual peso a tanto a la precisión como a la completitud mediante la métrica F_β. El gráfico es presentado a continuación en la figura 6.5.

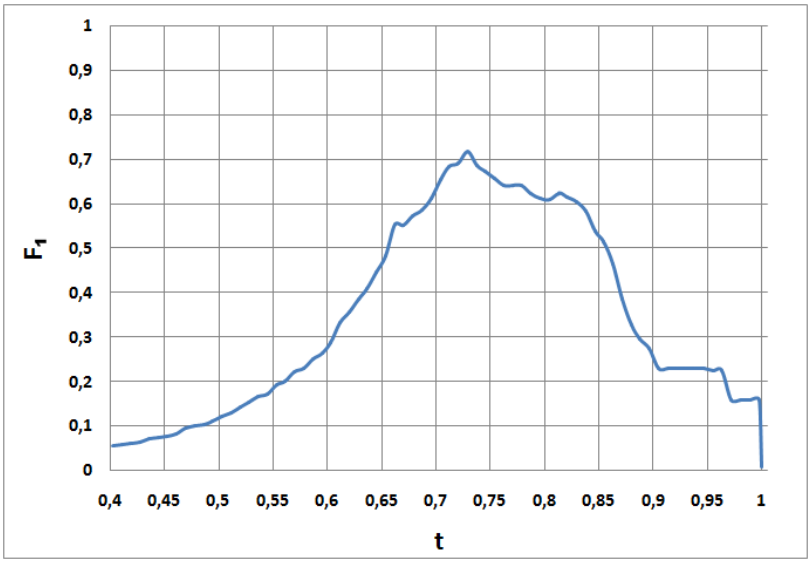


FIGURA 6.5. El gráfico muestra un comportamiento y punto máximo muy distintos al del gráfico t vs. F_4, lo que evidencia la importancia de cuantificar de manera correcta la mayor importancia de la completitud sobre la precisión en el detector.

Al analizar el gráfico se aprecia claramente que la tolerancia que obtiene el mayor puntaje es nuevamente 0.73. Esto era algo esperable, ya que ambas métricas utilizadas asignan la misma importancia tanto a la precisión como a la completitud.

Además, es posible notar la similitud en la forma entre los último dos gráficos, que difieren en su concavidad. Cabe destacar en todo caso que el gráfico de la figura 6.4 no corresponde exactamente a $1 - F_1$ debido a diferencias en sus escalas. Volviendo al gráfico, el puntaje alcanzado por la tolerancia óptima es de 0.72, mientras que el puntaje para la tolerancia seleccionada basándose en el primer criterio, 0.66, obtiene un puntaje de 0.57, que es bastante lejano al máximo. Finalmente, al analizar el gráfico con el fin de buscar un intervalo de funcionamiento estable, se puede apreciar que este se encuentra aproximadamente entre 0.71 y 0.81, que es de una extensión similar al encontrado anteriormente.

De esta manera, el proceso de pruebas arrojó 2 valores posibles para el nivel de tolerancia a figuras incompletas del módulo detector. Ambos presentan ventajas y desventajas con respecto al otro, pero dadas las características del sistema, se optó por la tolerancia obtenida por el criterio que da preferencia a la completitud, i.e., $t = 0.66$. A modo de resumen y para reafirmar las mejores características de esta tolerancia para el sistema de conteo, la tabla 6.2 muestra los resultados para la completitud obtenidos por el detector en la base de datos de imágenes utilizando los dos umbrales obtenidos mediante el análisis, segmentados de acuerdo a la disposición de las cabezas.

TABLA 6.2. Resultados de la completitud en la detección usando las dos tolerancias obtenidas.

Disposición \ Tolerancia	$t = 0.66$	$t = 0.74$
Frontal	0.80	0.80
Posterior	0.96	0.77
Lateral	1.00	0.90
Total	0.93	0.80

Como se puede apreciar, ambos umbrales entregan un buen rendimiento, aunque como era obvio, para $t = 0.66$ el rendimiento es claramente superior. Otro aspecto que merece la pena recalcar es que el detector tiene un excelente rendimiento frente a cualquier orientación de la cabeza, lo que habla muy bien de la generalidad del modelo. A pesar de eso, la orientación frontal obtiene un rendimiento algo menor, lo que se debe principalmente a los bordes generados por elementos faciales como ojos, nariz o boca.

6.1.2. Evaluación del tiempo de ejecución de la implementación

Para cuantificar la mejora en el rendimiento introducida por el uso de técnicas de programación de GPU en el algoritmo detector de cabezas, se desarrollo una implementación completa de este en una CPU tradicional. Esta implementación fue cuidadosamente ajustada para obtener el menor tiempo de ejecución posible. La etapa de detección de bordes fue desarrollada usando la versión del algoritmo de Canny disponible en la librería OpenCV (Bradski & Kaehler, 2008), mientras que la convolución en el dominio de la frecuencia fue implementada usando la librería FFTW (Frigo & Johnson, 2005). Esta librería es reconocida como la implementación de código libre más rápida de la transformada de Fourier.

Para comparar la velocidad de las implementaciones, se midió el tiempo de ejecución de cada uno de los pasos del algoritmo en el mismo conjunto de imágenes utilizados para evaluar el rendimiento de la clasificación. En la etapa de filtrado lineal se utilizaron cuatro máscaras con radios promedio de 20, 24, 28, 32. Finalmente, como la etapa de componentes conexos fue originalmente implementada en CPU, no hay diferencia para sus resultados entre las dos implementaciones. Los resultados son presentados en la tabla 6.3.

Como se aprecia, la mejora en el tiempo de ejecución es superior a siete veces, lo que permite la ejecución en tiempo real y también validar el uso de GPU para la implementación del algoritmo de detección.

TABLA 6.3. Resultados de la comparación de velocidad entre las implementaciones en CPU y GPU del algoritmo detector de cabezas.

Etapa	CPU	GPU	Mejora
Detección de bordes	12.26 ms	1.81 ms	6.8X
Filtrado lineal	144.23 ms	19.18 ms	7.5X
Umbralización	6.62 ms	0.74 ms	8.9X
Componentes conexos	0.85 ms	0.85 ms	0X
Tiempo total	163.96 ms	22.58 ms	7.3X

6.2. Estimación de la geometría epipolar

Para evaluar el rendimiento de los detectores de características locales, se utilizó una base de datos con 16 pares de imágenes estéreo de escenarios reales, todas con distinto largo de la línea base. La resolución de las imágenes va desde 640x480 a 1024x768 pixeles. Una muestra de las imágenes utilizadas puede verse en la figura 6.6. La metodología consiste en obtener inicialmente para cada imagen los puntos de interés y descriptores, tanto con SURF como con SIFT, para luego establecer las correspondencias entre cada par de imágenes estéreo. Luego, para cada detector se calculará la cantidad promedio de punto de interés y de correspondencias encontradas.

A continuación, para evaluar la relación entre cantidad de correspondencias generadas en las pruebas anteriores y la calidad de la estimación, se estimó la matriz fundamental para el subconjunto de las imágenes de la base de datos que tuvieran línea base mayor a 1.0 metro. Esta división se realizó para evaluar el rendimiento en escenarios similares a casos reales de aplicación del sistema. Se calculó la cantidad de correspondencias utilizadas en la estimación y el promedio de la distancia epipolar mutua, definida en (4.10), para las correspondencias utilizadas. Además, se evaluó el rendimiento de la estimación sobre un conjunto de 10 correspondencias elegidas a mano en cada una de las imágenes. Se midió el porcentaje de aciertos sobre este conjunto, donde se definió como un acierto en esta prueba un valor de la distancia epipolar mutua menor a 10 pixeles para un par de puntos correspondientes.

Finalmente, para verificar la calidad de la estimación y la real importancia del estabilizador temporal, se realizó un proceso de estimación continua de la geometría epipolar en una secuencia de video bifocal, registrando, en cada una de las estimaciones realizadas, el promedio de la distancia epipolar mutua para un grupo de 10 correspondencias definidas de forma manual en el escenario de la secuencia. Esta presenta el paso continuo de personas por un hall de entrada, lo que pone a prueba la habilidad del estimador para adaptarse a

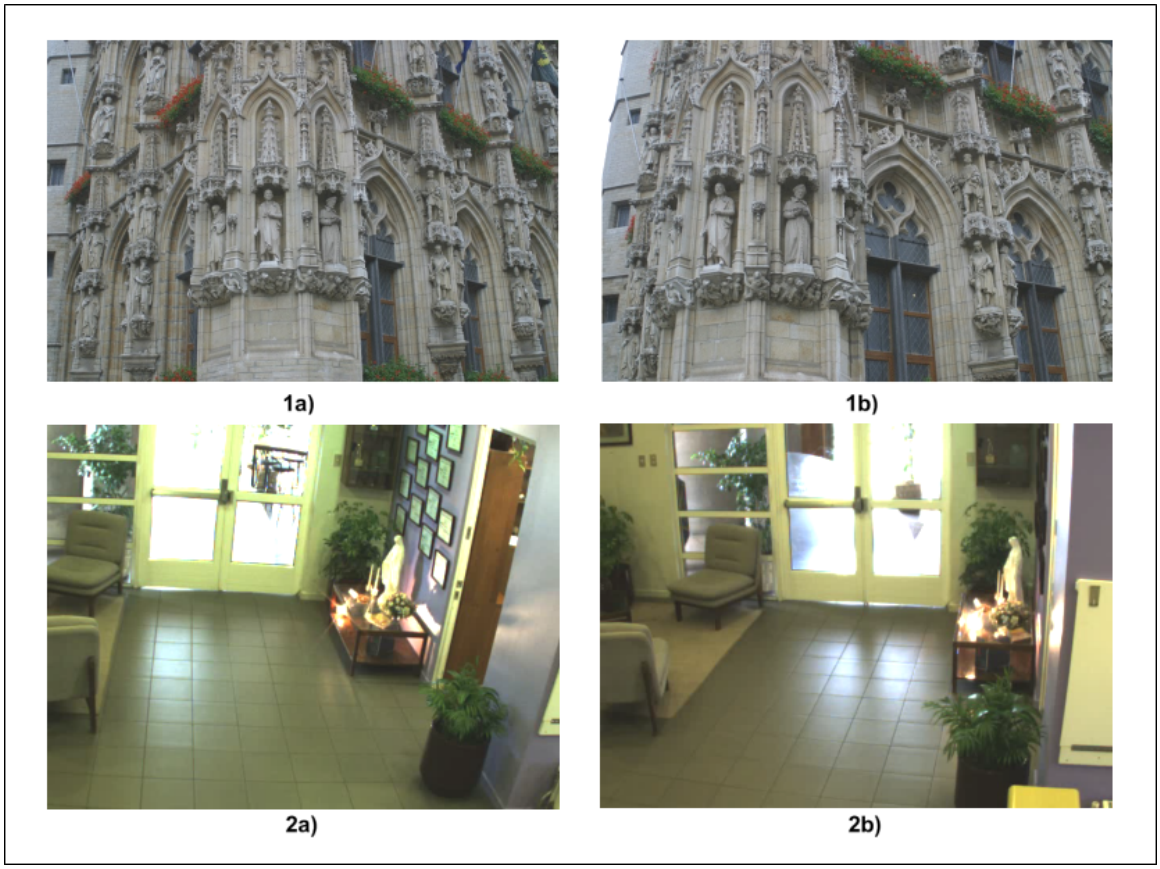


FIGURA 6.6. Las imágenes 1a) y 1b) presentan una dificultad moderada, debido a que la diferencia en los puntos de vista es compensada por la gran cantidad de puntos de interés que se pueden generar en los muros. Por otro lado, las imágenes 2a) y 2b) presentan un escenario más complejo, ya que tienen una línea base de gran longitud ($>$ 1.0m) y pocos detalles para generar puntos de interés.

un escenario cambiante. La secuencia de video fue grabada a una resolución de 640x480 pixeles y 25 cuadros por segundo.

6.2.1. Resultados generales

Para obtener buenos resultados en la estimación de F, lo más importante es que las correspondencias no sean todas coplanares, los que redunda en una matriz fundamental correcta sólo localmente. Además, es también importante que estas estén distribuidas por toda la imagen y que exista la mayor cantidad posible de éstas. Cabe destacar que para las pruebas se utilizó la mejor configuración posible para cada uno de los detectores, de manera que no existan ventajas de uno sobre el otro.

La siguiente tabla muestra la cantidad promedio de punto de interés por imagen ($\overline{PI}$) y de correspondencias por par de imágenes ($\overline{C}$) para los dos detectores utilizando la base de datos de imágenes estéreo. Además presenta el promedio de la razón entre correspondencias y puntos de interés ($\overline{C/PI}$) de cada imagen. Es importante notar que $\overline{C/PI} \neq \overline{C}/\overline{PI}$. Esta última no tiene interés para el análisis en este caso.

TABLA 6.4. Cantidad promedio de punto de interés y correspondencias para SIFT y SURF

Detector	$\overline{PI}$	$\overline{C}$	$\overline{C/PI}$
SURF	3658	1266	0.32
SIFT	2030	640	0.32

Como se puede apreciar en la tabla 6.4, SURF obtiene resultados claramente mejores en cuanto a cantidad de correspondencias e iguales en cuanto a la razón $\overline{C/PI}$. Por otro lado, la distribución de los puntos de interés por imagen de SURF es más amplia que la de SIFT, lo que debería redundar finalmente en una estimación de la geometría epipolar global y no local, y por lo tanto más robusta. Un resultado interesante puede apreciarse al segmentar las imágenes de acuerdo a su largo de línea base. La siguiente tabla muestra la información de las correspondencias de la tabla anterior, pero con los valores agrupados además de acuerdo al largo de la línea. La división es de 12 imágenes con línea base menor a 1.0 metro y 4 con línea base mayor o igual a 1.0 metro y menor a 2.0 metros.

TABLA 6.5. Resultados de análisis de las correspondencias agrupado de acuerdo al detector y al tamaño de la línea base.

Largo línea base / Detector	< 1.0 m.	≥ 1.0 m.
SURF	0.38	0.14
SIFT	0.40	0.07

Al observar esta tabla se hace bastante clara la diferencia entre SURF y SIFT, que era ocultada en parte en la tabla 6.4. Al disminuir la distancia entre cámaras, SIFT obtiene una leve ventaja, aunque dada la cantidad de correspondencias obtenidas por lo detectores esta no tiene mayor importancia. Por el contrario, cuando la distancia es mayor, SURF logra caracterizar y localizar de mejor manera cada punto de interés, lo que redunda finalmente en un porcentaje de puntos correspondientes que duplica al de SIFT. Las siguientes dos figuras (6.7 y 6.8) muestran la distribución de puntos correspondientes de SURF y SIFT para las imágenes de la figura 6.6. En ellas se puede apreciar un rendimiento similar para

SIFT y SURF en un caso de dificultad media y la ventaja de SURF sobre SIFT para un caso más complejo.

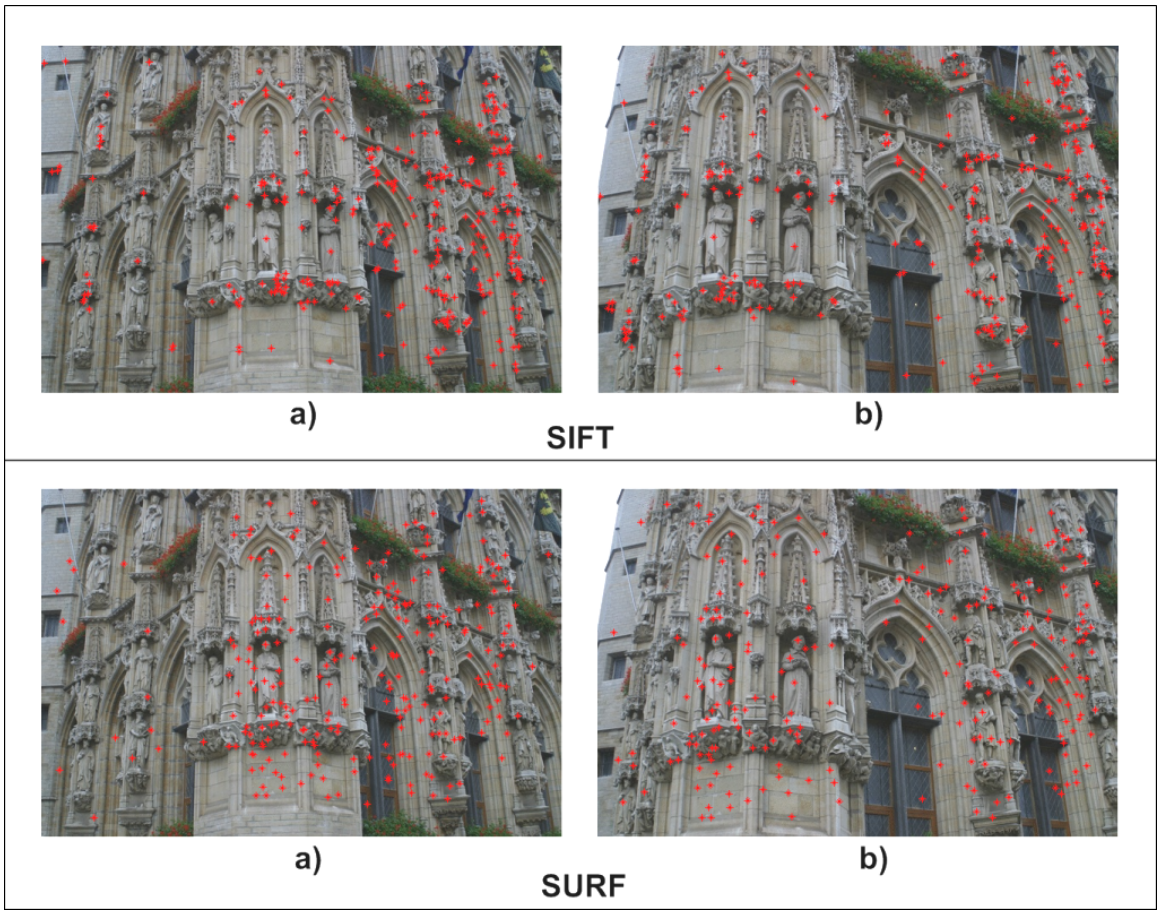


FIGURA 6.7. En la figura, los asteriscos rojos representan puntos correspondientes. Como se puede apreciar, SIFT genera una cantidad levemente mayor de correspondencias que SURF, 346 vs. 269, aunque en este caso, dada la buena distribución de las correspondencias, esta diferencia no generará un mejor rendimiento a la hora de estimar la matriz fundamental.

Esta diferencia es crucial en este caso, ya que la cantidad de correspondencias generadas con esta distancia de línea base es bastante reducida en comparación con una línea base menor. Dado que mientras mayor sea la distancia entre cámaras, mayor es la capacidad de manejar correctamente casos de oclusión en una de ellas, SURF cuenta con una clara ventaja para ser usado en el sistema de estimación. Estos resultados coinciden con lo reportado en la literatura (Bay et al., 2006), (Gil, Mozos, Ballesta, & Reinoso, 2009), con respecto a la mayor robustez de SURF para manejar casos complejos con grandes diferencias en el angulo de visión, en este caso generado por el tamaño de la línea base.

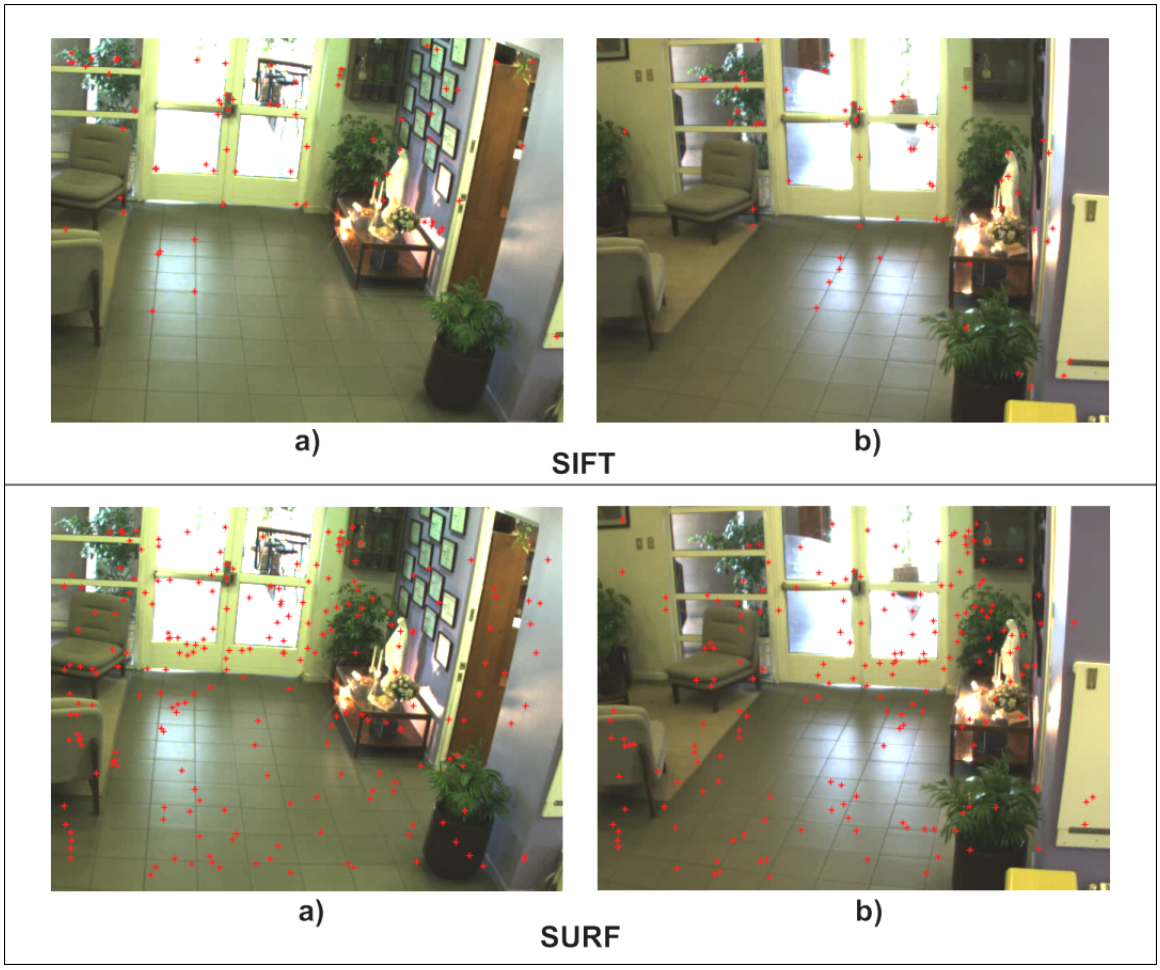


FIGURA 6.8. En esta figura la situación es la contraria a la anterior. SURF obtiene una ventaja patente, generando 3 veces más correspondencias que SIFT, 218 vs. 70. Este ejemplo muestra claramente la ventaja de SURF para casos complejos.

Aunque estos resultados son suficientes para favorecer el uso de SURF sobre SIFT, todavía queda evaluar sus rendimientos reales al momento de estimar la matriz fundamental. Se podría pensar que el hecho de tener más correspondencias implica inmediatamente una mejor calidad en la estimación, pero esto no es necesariamente cierto. Como fue descrito en la sección 4.2, las correspondencias entregadas por el proceso de matching no son 100% fiables, ya que no se les ha impuesto restricción geométrica alguna y sólo se han establecido a través de criterios de similitud de vecindarios. Con el fin de evaluar esta situación se presenta la tabla 6.6, que aparte de contener la cantidad promedio de correspondencias producidas por el proceso de matching, $\overline{C}$, muestra la cantidad de estas que fueron finalmente utilizadas en la estimación de la matriz fundamental, $\overline{C}_e$. Junto con estos datos se presenta también una estimación de la calidad de la matriz basada en la distancia epipolar

mutua medida en pixeles, d, y además la razón de aciertos sobre un conjunto de 10 correspondencias elegidas a mano por cada par de imágenes, A. Los resultados se calcularon promediando los valores obtenidos en 25 ejecuciones del algoritmo y se incluye además la desviación estándar del promedio.

TABLA 6.6. Resultados de la cantidad de correspondencias utilizadas en la estimación de la matriz fundamental y la calidad de esta.

Detector	$\overline{C}$	$\overline{C}_e$	$\overline{d}$	A
SURF	240	65 ± 8.0	1.65 px $\pm$ 0.16 px	0.88 ± 0.01
SIFT	93	33 ± 2.5	1.40 px $\pm$ 0.26 px	0.73 ± 0.02

Como se puede apreciar, la cantidad de correspondencias usadas en la estimación disminuye dramáticamente al ser comparadas con las obtenidas por el proceso de matching. A pesar de esto, la calidad de la estimación para ambos es bastante buena, aunque con una clara ventaja para SURF, dada principalmente por su mejor comportamiento para imágenes con este largo de línea base, detectado en la tabla 6.5. El valor de SIFT para A muestra claramente que la distancia epipolar promedio sobre las correspondencias no es una buena medición del rendimiento global cuando hay pocas correspondencias, ya que es muy probable que la matriz obtenida sea correctamente sólo localmente en los sectores donde se ubica la mayor cantidad de ellas. Los resultados de esta tabla permiten elegir a SURF como detector de características locales para el sistema detector en desmedro de SIFT, dada su clara ventaja en un escenario similar a los de aplicación real. Así, para las siguientes pruebas sólo se presentaran los resultados de SURF.

El alto valor en los aciertos para SURF implica que en promedio se cometerá un error aproximado muy bajo al generar la línea epipolar para buscar una correspondencia de objetivos de seguimiento. Tomando en cuenta que un objetivo de seguimiento tiene en promedio un radio del orden de los 20-25 pixeles, la calidad de la estimación nos asegura que no existirán problemas para formar los objetivos de seguimiento estéreo, dado que su proceso de formación es dirigido por la ecuación (5.5), que se basa en el radio del objetivo para la validación. Como dato extra cabe destacar que todas las estimaciones de la matriz fundamental obtuvieron determinante cero, lo que asegura su singularidad y el cumplimiento de la restricción de que todas las líneas epipolares pasen por el epipolo de cada imagen.

Finalmente, con el fin de evaluar el rendimiento de la estimación de manera dinámica y continua, se analizó el comportamiento del estimador a través del tiempo, midiendo la cantidad de correspondencias utilizadas y el promedio de la distancia epipolar mutua sobre un conjunto de correspondencias elegidas a mano, para cada una de las estimaciones hechas.

Este proceso se realizó tanto para el estimador con estabilización temporal como sin. Los resultados de la evolución de la distancia epipolar mutua son presentado en la figura 6.9.

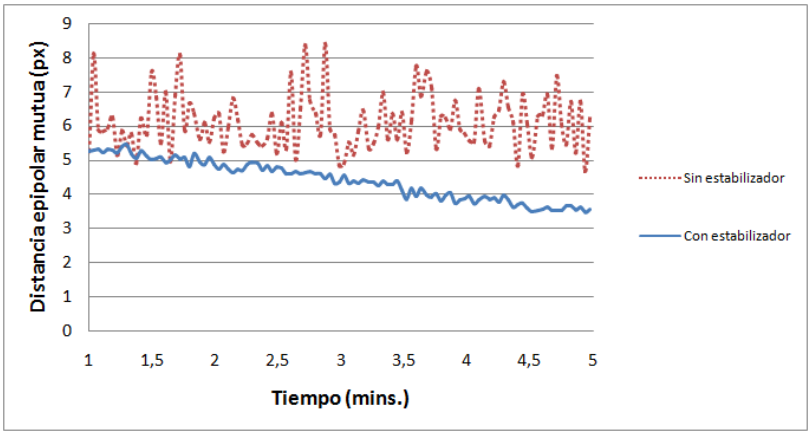


FIGURA 6.9. Evolución de la distancia epipolar mutua a través del tiempo.

En esta figura se aprecia como el estabilizador disminuye ostensiblemente las variaciones en la calidad de la estimación. Esto se debe a que el estabilizador almacena la mejor estimación que se ha realizado hasta el momento, lo que permite mantener una gran homogeneidad en los resultados, a diferencia del caso no estabilizado donde la matriz fundamental cambia en cada estimación, debido a la naturaleza estocástica del proceso de cálculo, lo que provoca cambios drásticos en la distancia, a veces altamente perjudiciales. Volviendo al gráfico, es posible apreciar como la distancia epipolar mutua disminuye a través del tiempo cuando se utiliza el estabilizador hasta aproximadamente los 4.5 minutos, donde se estabiliza en aproximadamente 3.5 pixeles. Este es nuevamente un efecto del almacenamiento de la mejor estimación realizada hasta el momento. Por otro lado, al calcular la distancia epipolar mutua promedio para el caso no estabilizado, esta obtiene un valor de 6.1 pixeles casi constante a través del tiempo, a diferencia del caso estabilizado que muestra una clara mejora de su rendimiento en el tiempo. A continuación se presentan los resultados de la evolución en el tiempo de la cantidad de correspondencias utilizadas en la estimación en la figura 6.10.

Como se puede observar en esta figura, la variación a través del tiempo en la cantidad de correspondencias del caso estabilizado es bastante grande, aunque mucho menor a la del caso no estabilizado. Esto se explica principalmente por las oclusiones generadas por el paso de personas de sectores donde hay gran cantidad de puntos de interés en las imágenes. A pesar de esto, el caso estabilizado presenta una cantidad de correspondencias promedio

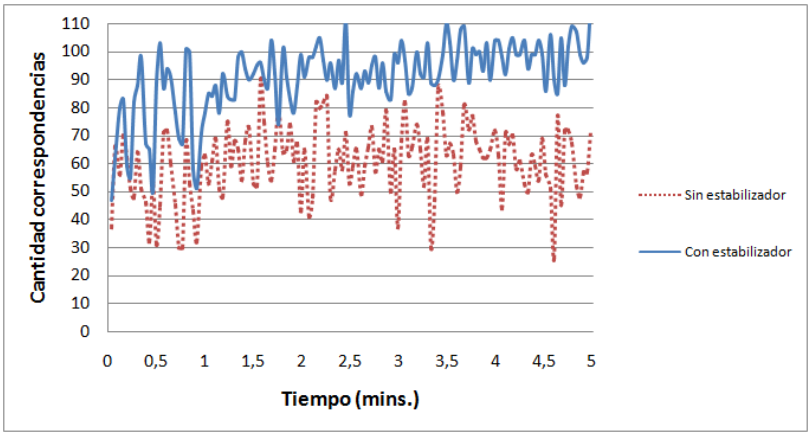


FIGURA 6.10. Evolución en el tiempo de la cantidad de correspondencias usadas en la estimación.

que aumenta en a través del tiempo, a diferencia del caso no estabilizado, en que esta se mantiene casi constante en 60.

En ambos gráficos se puede apreciar claramente como el factor dinámico de la escena puede perjudicar el rendimiento del estimador cuando no se utiliza el estabilizador. Este problema se da por oclusiones generadas por el paso de personas en sitios donde se agrupaba una cantidad importante de puntos de interés sumamente descriptivos para la estimación. Como la estimación sin estabilizador no toma en cuenta la información de estimaciones anteriores, cualquier dificultad puntual en las vistas del escenario con las que se realiza el proceso, redundará en una disminución de la calidad. Por el contrario, al incluir información temporal, el estabilizador le permite al módulo estimador adaptarse a estos cambios de mejor manera, manteniendo e incluso mejorando la calidad de la estimación a través del tiempo.

6.3. Conteo de personas

Para evaluar el sistema de conteo se utilizaron cuatro secuencias de video bifocales. Estas secuencias fueron grabadas en una resolución de 640x480 pixeles con una tasa de refresco de 25 cuadros por segundo. Las cámaras fueron ubicadas sobre la puerta de entrada de una sala de clases, apuntando a la entrada de una sala de espera, a una altura de 2.3 metros y con una separación entre ellas de 1.7 metros. La figura 6.11 presenta un diagrama del posicionamiento.

Cada secuencia presenta un escenario de flujo real, con intervalos de tráfico de gente unidos por intervalos de calma. Como fue descrito anteriormente, la línea de conteo puede

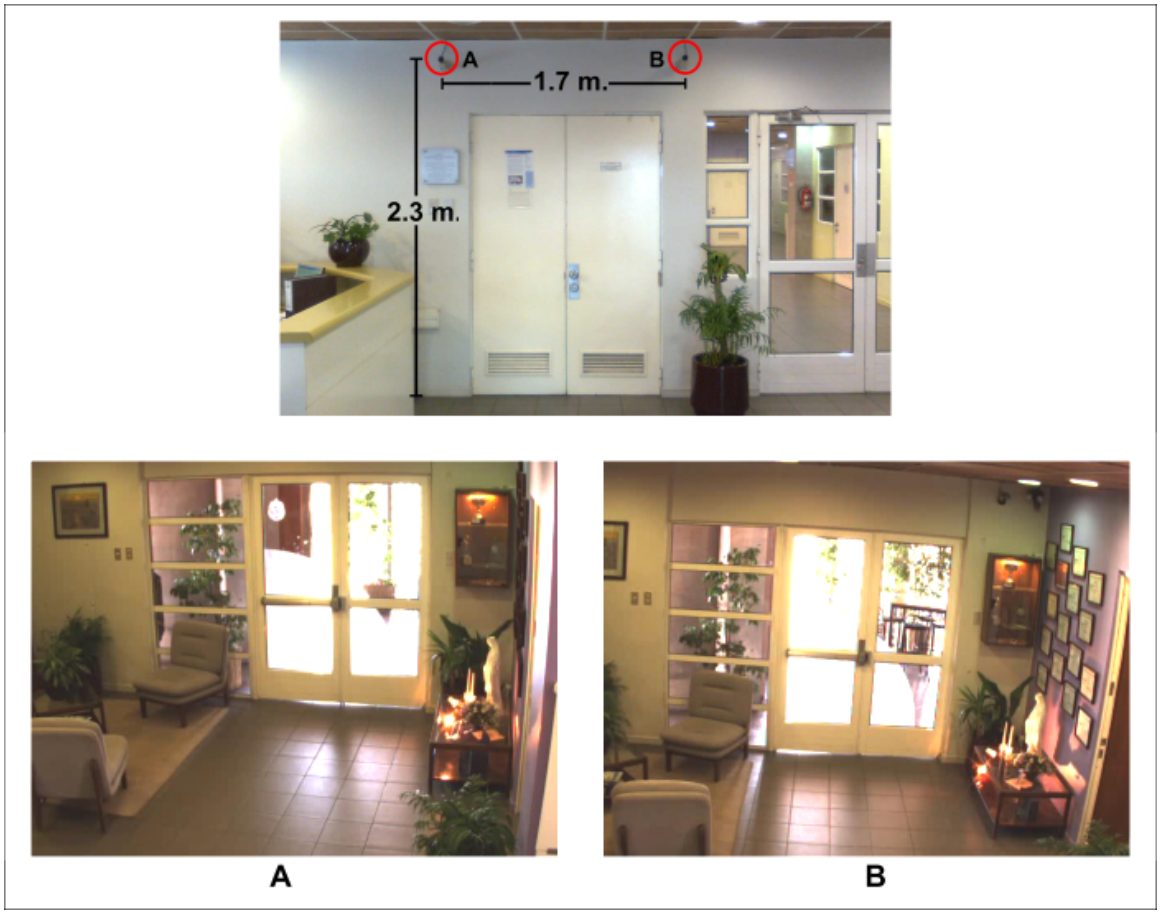


FIGURA 6.11. Diagrama de posicionamiento de las cámaras.

ser definida en cualquier sector de las imágenes y puede tener cualquier largo. Dadas las características del escenario de las secuencias, la línea de conteo se definió en cada una de ellas como una línea con inclinación casi horizontal que divide a cada imagen en 2.

Con el fin de resumir la información de cada una de las secuencias, la tabla 6.7 presenta para cada una de ellas su duración, cantidad de personas que cruzaron la línea de conteo, la cantidad de éstas que cruzaron la línea de conteo desde el sector superior hacia el inferior (Entrada) y las que la cruzaron desde el inferior al superior (Salida).

Las secuencias 3 y 4 fueron grabadas en momentos de alto tráfico de personas, en los instantes previos al inicio de clases, mientras que las secuencias 1 y 2 presentan escenarios de conteo con tráfico normal. Las secuencias procesadas y sin procesar pueden encontrarse en la dirección url `http://www.youtube.com/halobelTesis/`.

TABLA 6.7. Resumen de los datos de las 4 secuencias de conteo.

Secuencia	Duración	Cruces	Entrada	Salida
1	5:14 min.	16	9	7
2	16:59 min.	20	15	5
3	4:15 min.	26	8	18
4	8:28 min.	36	22	14

Para cuantificar el rendimiento del contador, la idea lógica y más simple es realizar una comparación entre el número de cruces detectados por el sistema y el número real de cruces. El problema de este enfoque es que es muy probable que se produzca un efecto de compensación entre las detecciones correctas y las falsas. Por ejemplo, si un cruce no es detectado y luego otro cruce es detectado dos veces, el marcador total de cruces indicará el número correcto, a pesar de que el sistema cometió dos errores.

Así, para cuantificar mejor el rendimiento del contador de personas se utilizaron, al igual que en el detector de cabezas, mediciones estadísticas basadas en los coeficientes TP, FP, FN de la matriz de confusión. En este caso, TP representa la cantidad de cruces reales por la línea de conteo correctamente detectados, FP la cantidad de cruces por la línea de conteo detectados que no corresponden a cruces reales y FN la cantidad de cruces reales que no fueron detectados. Las medidas utilizadas fueron nuevamente precisión, completitud y F_1, definidas anteriormente en (6.2) y (6.3).

6.3.1. Resultados y análisis

Cada secuencia de video fue procesada usando tanto el sistema de conteo estéreo como las 2 versiones monoculares de él, i.e., izquierda y derecha. Esta versión monocular no utiliza el módulo estimador de geometría epipolar, por lo que debería ser en teoría más sensible a falsos negativos y oclusiones en escenarios con mucho tráfico simultáneo, como las secuencias 3 y 4. Medir los resultados de este contador permite apreciar de mejor manera la ventaja entregada por el enfoque estéreo. Para cada combinación secuencia-contador, se midieron la precisión, la completitud y F_1. Además, para cada combinación secuencia-métrica se calcula la métrica "Mejora", definida como la diferencia entre el puntaje obtenido por el contador estéreo y el promedio de los puntajes obtenidos por los contadores monoculares. Los resultados son resumidos en la tabla 6.8.

Como se puede apreciar, el sistema de conteo estéreo obtiene una completitud promedio 0.11 puntos más alta que los contadores monoculares. Esta diferencia es aún más notoria en las secuencias 3 y 4, alcanzando un máximo de 0.23 puntos en la secuencia

TABLA 6.8. Resumen de resultados para los sistemas de conteo estéreo y monocular.

Secuencia	Contador	Precisión	Completitud	F_1
1	Mono Izq.	0.88	0.88	0.88
	Mono Der.	0.93	0.88	0.90
	Estéreo	0.93	0.88	0.90
	Mejora	0.02	0.00	0.01
2	Mono Izq.	0.85	0.85	0.85
	Mono Der.	0.84	0.80	0.82
	Estéreo	0.90	0.90	0.90
	Mejora	0.05	0.07	0.07
3	Mono Izq.	0.95	0.73	0.83
	Mono Der.	0.86	0.69	0.77
	Estéreo	0.96	0.92	0.94
	Mejora	0.05	0.21	0.14
4	Mono Izq.	0.87	0.72	0.79
	Mono Der.	0.88	0.78	0.83
	Estéreo	0.92	0.92	0.92
	Mejora	0.04	0.17	0.11
Total	Mono Izq.	0.89	0.80	0.84
	Mono Der.	0.88	0.79	0.83
	Estéreo	0.93	0.91	0.92
	Mejora	0.04	0.11	0.08

3 con respecto al peor de los contadores. Esta diferencia está estrechamente relacionada con el manejo de las oclusiones y los falsos negativos generados por estas por la fusión incorrecta de objetivos de seguimiento. En el caso del contador estéreo, las oclusiones son manejadas casi siempre de manera correcta, incluso con una gran cantidad de tráfico en ambas vistas. Además, los falsos negativos en una imagen son detectados generalmente de manera correcta en la otra, por lo que el efecto de estos errores disminuye de gran manera. Todo esto implica que exista una muy baja pérdida en la detección de cruces por la línea de conteo, lo que redunda finalmente en que no exista una baja en la completitud al aumentar la cantidad de tráfico de personas. Por otro lado, la gran baja de esta para los contadores monoculares en las secuencias 3 y 4, coincide justamente con el aumento en el tráfico, lo que conlleva también un aumento en las oclusiones al momento de cruzar la línea de conteo (Figura 6.12) y mayor cantidad de falsos negativos, generados por oclusiones previas de las cuales el detector no es capaz de recuperarse y por problemas de fusión de objetivos de seguimiento.

FIGURA 6.12. La gran longitud de la línea base (1.7 m.) permite una muy buena discriminación en caso de de oclusiones, aunque con un mayor tiempo de cómputo para estimar la geometría epipolar que un caso más simple (línea base menor).

También puede verse en los resultados para las secuencias 1 y 2, que la adición de información estéreo no produce un aumento en la cantidad de errores, a pesar de que la información sea en ocasiones redundante y que pueda contener falsos positivos. Esto puede apreciarse en los valores de la precisión, que no muestran una baja en las secuencias más simples.

Otro aspecto que es posible apreciar es que el nivel de precisión se mantiene bastante estable en torno a los 0.85-0.95 puntos para todos los contadores. Este hecho tiene directa relación con el detector de cabezas y coincide con los resultados obtenidos para este. Esto significa también que cualquier mejora que se realice en él tendrá un impacto inmediato en el rendimiento del contador.

Con respecto a los errores en la precisión del conteo, la mayoría de estos corresponden a detecciones múltiples del mismo objetivo, aproximadamente un 75%, generadas en sectores del cuerpo distintos a la cabeza pero con una apariencia circular bastante estable, como hombros vistos desde perfil o bolsos. El otro 25% de los errores en la precisión corresponden a detecciones falsas del fondo que se producen en lugares cercanos a la línea de conteo y que luego, por lás características inherentes del algoritmo de seguimiento por apariencia CamShift, son trasladadas levemente, lo que provoca en varios casos que el objetivo cruce la línea de conteo. En relación a los errores en la completitud, no existe un patrón regular que explique la mayoría de estos errores, salvo el de las cabezas que no considera el modelo circular utilizado, como por ejemplo gente con sombreros o gorros.

Finalmente, todos estos resultados ratifican la validez de la metodología propuesta y confirman que, si es bien utilizada, la información estéreo entrega una enorme ganancia en el rendimiento.

6.4. Resumen de Resultados

Dado que el trabajo desarrollado en esta tesis está comprendido por una gran cantidad de elementos, y cada uno de estos fue evaluado, es necesario realizar una síntesis de los resultados obtenidos en cada uno de estos módulos. Las siguientes secciones presentan cada uno de estos resultados juntos con su importancia para el rendimiento del sistema de conteo.

Detector de cabezas

- El módulo detector de cabezas muestra una alta completitud, > 0.8, para precisiones menores o iguales a 0.65, aspecto que resulta positivo para el sistema, dado que la mayoría de los falsos positivos corresponden a elementos del fondo y no presentan cruces por la línea de conteo.
- Dadas las características de los falsos positivos encontrado y el mecanismo de conteo, se utilizó en el detector de cabezas el umbral óptimo para la métrica F_4, $t = 0.66$, en detrimento de 0.74, que fue el encontrado con la métrica F_1. Este valor implica que serán aceptadas las detecciones que presenten un nivel de incompletitud máximo de 66%. La precisión para este umbral fue de 0.4 mientras que la completitud se elevó hasta 0.93.

- La implementación en GPU del detector obtiene un tiempo de ejecución 7.3 veces menor que el de el mismo detector implementado completamente en un CPU. Esto permite el uso del sistema en tiempo real.

Estimación de la geometría epipolar

- El detector de características locales SURF obtuvo un rendimiento en matching muy superior al de SIFT, en pruebas sobre imágenes estéreo con línea base superior a 1.0 metros. El número de correspondencias obtenidas por SURF fue en promedio casi el triple de las de SIFT (240 vs. 93). Este resultado coincide con el de otros trabajos disponibles en la literatura.
- Al introducir restricciones geométricas en el matching para estimar la matriz fundamental en escenarios de línea base amplia, SURF supera nuevamente de

manera amplia a SIFT, generando en promedio 65 puntos, contra sólo 33 de SIFT. Este resultado permitió elegir a SURF como el detector de características locales para el módulo de estimación.

- La estimación de la matriz fundamental para el caso estático presento una distancia epipolar mutua promedio de 1.65 ± 0.16 pixeles, sobre el conjunto de correspondencias utilizadas en la estimación. Al medir esta distancia sobre un conjunto de correspondencias elegidas a mano, el valor disminuye a 0.88 ± 0.01 pixeles.
- El uso del estabilizador temporal mejora sustancialmente el rendimiento de la estimación de la geometría epipolar en el caso dinámico. La cantidad de correspondencias utilizadas en la estimación muestra un aumento consistente en el tiempo, a diferencia del caso no estabilizado que oscila constantemente alrededor de su valor promedio. De la misma manera, la distancia epipolar mutua presenta un descenso progresivo de su valor en el caso estabilizado, hasta alcanzar un valor relativamente estable aproximadamente a los 4,5 minutos del proceso. Por otro lado, al no estabilizar la estimación, el valor de la distancia oscila nuevamente en torno a su valor promedio.

Conteo de personas

- El sistema de conteo trabaja a una tasa de 20 cuadros por segundo, lo que es posible debido al uso de CUDA para el algoritmo de detección y a CamShift para el seguimiento. Esta alta frecuencia de refresco permite un mejor seguimiento de los objetivos gracias a una menor diferencia entre imágenes consecutivas.
- La completitud promedio del contador estéreo es 0.11 puntos más alta que la de los contadores monoculares. Esta diferencia alcanza un máximo de 0.23 puntos en las secuencias más demandantes.
- El uso de información estéreo redundante no genera grandes cambios en el rendimiento, ya que la precisión de todos los contadores en todas las secuencias se mantiene bastante estable en torno a los 0.85-0.95 puntos. Esto demuestra lo expuesto anteriormente en relación al reducido efecto que tienen los falsos positivos generados por el detector de cabezas en la etapa de conteo.
- El valor promedio de la métrica F_1 para el contador estéreo es de 0.92 puntos, superando en 0.08 punto a los contadores monoculares. Este resultado ratifica la validez del enfoque propuesto.

7. CONCLUSIONES Y TRABAJO FUTURO

7.1. Conclusiones

En base a los resultados presentados en la sección anterior se puede concluir que el objetivo principal de esta tesis fue logrado. El sistema utiliza exitosamente la visión estéreo de línea base amplia para manejar el movimiento de las cámaras de manera no calibrada, mientras que tanto el algoritmo de detección de cabezas como el sistema de seguimiento y conteo presentan una buena tolerancia a los cambios de iluminación, ya que fueron desarrollado utilizando detección de bordes y componentes de color sin información de luminosidad respectivamente.

Al analizar los resultados más en detalle, se advierte que los dos módulos que alimentan al contador, detector de cabezas y estimador de la geometría epipolar, obtienen altos rendimientos de manera individual que los hacen perfectamente adecuados al sistema de conteo.

En el caso del módulo detector, su alta completitud tiene mayor importancia que su baja precisión, debido al esquema de conteo implementado. Además, la ventaja de utilizar GPUs se hace más evidente al comparar este módulo del sistema con una versión del mismo completamente desarrollado utilizando un CPU tradicional, donde el sistema trabaja a una frecuencia de refresco de tan sólo dos cuadros por segundo, lo que lo hace inutilizable en una situación real. El éxito de esta implementación abre las puertas a la adaptación de otros algoritmos al esquema de computación paralelo.

A pesar de esto es importante tomar en cuenta que una mejoría en la precisión de este módulo ayudaría de gran manera a la simplicidad del módulo de seguimiento, eliminando ciertas consideraciones relacionadas con los falsos positivos, como la eliminación de cruces múltiples y de objetivos estáticos, con el fin de mantener la precisión alta y el tiempo de ejecución bajo.

Con respecto al módulo estimador de la matriz fundamental, su bajo error en la generación de las líneas epipolares para ambas vistas permiten que el manejo de la información estéreo en la etapa de conteo sea sumamente sencilla y elegante, al sólo necesitarse el producto de matrices con vectores. Además, al no necesitar calibración permite su uso inmediato en casi cualquier entorno de operación.

Aún tomando estas bondades en cuanta, el estimador puede ser mejorado mucho, sobre todo en el plano del tiempo de ejecución y adaptación a entornos muy variables. Lo primero

puede ser logrado portando el código de manera completa a una GPU, cosa que no es trivial en este caso debido a la complejidad en los cálculos matriciales, mientras que lo segundo requiere de mecanismos inteligentes para adaptar la cantidad de puntos de interés necesarios para obtener un buena estimación. Estas mejoras permitirían el uso del sistema de conteo en entornos altamente complejos y dinámicos como paraderos de buses.

Pasando al módulo de conteo, sus resultados validan las dos grandes contribuciones de este trabajo realizadas en los dos módulos anteriores, i.e., el traspaso de procesos pesados a la GPU para disminuir el tiempo de ejecución total del sistema y el uso de restricciones de geometría epipolar para mejorar el manejo de las oclusiones parciales sin necesidad de requerir un sistema calibrado. Otro aspecto importante del uso de CUDA es que su arquitectura permite un aumento inmediato de rendimiento al añadir más GPUs al sistema, lo que entrega la posibilidad en el futuro de concentrarse en mejoras al rendimiento de los algoritmos desarrollado, en vez de la optimización del código para reducir el tiempo de ejecución

Finalmente, es posible obtener una conclusión bastante importante al analizar las diferencias entre las versiones monocular y estéreo del sistema de conteo. Como se puede apreciar, la diferencia entre ellos se ve reflejada principalmente en sus índices de completitud, que reflejan por consiguiente el manejo de las oclusiones y falsos negativos. Por otro lado, tanto sus índices de precisión como su tiempo de ejecución no presentan diferencias significativas. Así, extrapolando el comportamiento del sistema, tiene sentido pensar que al añadir una tercera cámara el rendimiento sólo se vería incrementado, sin que esto signifique una disminución en el tiempo de ejecución o un aumento de los falsos positivos (disminución de la precisión). Basado en esto cobra mucha importancia la decisión de donde centrar los esfuerzos de desarrollo de un sistema de visión de este tipo, donde las opciones son generalmente dos:

(i) La creación de un detector extremadamente robusto pero pesado en tiempo de computación que entregue excelente rendimiento en una sola vista, pero que no sea aplicable a varias debido a su tiempo de ejecución. Este es el enfoque utilizado en la mayor parte de los trabajos descritos en la literatura.

(ii) La creación de un detector simple y muy rápido unido a un sistema que fusione la información de múltiples vistas, que permitan un incrementa de rendimiento sin perjuicio de tiempo de ejecución al incluir información de una mayor cantidad de cámaras.

En este caso, el uso de CUDA y la geometría epipolar de línea base amplia permitieron la evaluación de esta última opción, demostrando claramente que es una alternativa sumamente atractiva. Basado en los precios actuales y disponibilidad de los dispositivos digitales de captura de imágenes, unido a la creciente expansión de las GPU tanto en computadores de escritorio como portátiles, puede resulta mucho más económico desarrollar un sistema como el descrito, en vez de uno altamente monolítico.

7.2. Trabajo futuro

Existe una gran cantidad de temas de investigación futura que emergieron durante el desarrollo de este trabajo. La mayoría de estos apunta a la mejora del rendimiento del sistema y a hacerlo aplicable de mejor manera en entornos no controlados. A continuación se presenta un listado de ellos junto con una breve descripción.

- Traspasar todo el procesamiento del estimador de geometría epipolar desde la CPU a la GPU para alcanzar una frecuencia de refresco que permita adecuarse de mejor manera a movimientos rápidos y constantes de las cámaras, como el caso de sistemas pan-tilt-zoom.
- Estudiar características que permitan modelar de mejor manera la apariencia de una cabeza con el fin de añadir un capa de clasificación del detector de elementos circulares. El objetivo de esto es eliminar la mayor cantidad de falsos positivos encontrados, sobre todo en escenarios con fondos complejos.
- Estudiar la aplicación de técnicas dinámicas de sustracción de fondo, que permitan mejorar la precisión de la detección y la disminución de falsos positivos y aun ser aplicable en entornos no controlados.
- Mejorar el esquema de conteo utilizando técnicas más robustas a conteos esporádicas generadas por falsos positivos. Una posible opción es añadir más líneas de conteo y analizar el orden en que fueron cruzadas.
- Realizar reconstrucción 3D de la posición de las detecciones en base a la matriz fundamental estimada, para realizar un seguimiento en base a la proyección de las detecciones al espacio 3D. La aplicación de esta mejora permitiría fusionar de mejor manera información espacial y por consiguiente perfeccionar el manejo de casos extremos de oclusión constante.
- Mejorar el sistema de seguimiento, introduciendo otras características directoras y posiblemente utilizando una implementación en GPU de un filtro de partículas.

- Probar el sistema con cámaras de menor calidad, idealmente webcams, con el fin de estudiar su aplicabilidad más masiva y también de disminuir el costo de implementación de este sistema en la práctica.
- Realizar pruebas utilizando una mayor cantidad de cámaras.
- Incluir técnicas adaptivas en el estabilizador temporal de la estimación de la geometría epipolar con el fin de ajustar la configuración del detector de características locales para que genere siempre una cantidad suficiente de puntos de interés, sin importar las características del escenario. Esto mejoraría notablemente el nivel de adaptabilidad del sistema a escenarios sumamente complejos o cuando la distancia entre cámaras varíe constantemente.
- Estudiar nuevos entornos de desarrollo para GPU distintos a CUDA, que permitan el uso de tarjetas de video de una marca distinta de NVIDIA, como OpenCL (KhronosGroup, 2009) o Stream (AMD, 2009). Al momento de desarrollo del sistema estos entornos aún no estaban disponibles o no tenían un nivel de madurez adecuado para ser utilizados.

BIBLIOGRAFIA

AMD. (2009). *Ati stream computing technical overview.* (http://developer.amd.com/gpu_assets/Stream_Computing_Overview.pdf)

Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Afips '67 (spring): Proceedings of the april 18-20, 1967, spring joint computer conference* (p. 483-485). New York, NY, USA: ACM.

Barandiaran, J., Murguia, B., & Boto, F. (2008, 7–9 May). Real-time people counting using multiple lines. In *Proc. ninth international workshop on image analysis for multimedia interactive services wiamis '08* (pp. 159–162).

Bay, H., Tuytelaars, T., & Gool, L. V. (2006). Surf: Speeded up robust features. In *9th european conference on computer vision* (p. 404-417).

Blythe, D. (2008, May). Rise of the graphics processor. *Proceedings of the IEEE*, *96*(5), 761–778.

Bradski, G. R. (1998, 19–21 Oct.). Real time face and object tracking as a component of a perceptual user interface. In *Proc. fourth ieee workshop on applications of computer vision wacv '98* (pp. 214–219).

Bradski, G. R., & Kaehler, A. (2008). *Learning opencv* (O. M. Inc., Ed.). O'Reilly Media Inc.

Canny, J. (1986, Nov.). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*(6), 679–698.

Carrasco, M., Pizarro, L., & Mery, D. (2008). Image acquisition and automated inspection of wine bottlenecks by tracking in multiple views. In *Iscgav'08: Proceedings of the 8th conference on signal processing, computational geometry and artificial vision* (pp. 84–89). Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS).

Celik, H., Hanjalic, A., & Hendriks, E. A. (2006, 8–11 Oct.). Towards a robust solution to people counting. In *Proc. ieee international conference on image processing* (pp. 2401–2404).

Chen, C.-H., Chang, Y.-C., Chen, T.-Y., & Wang, D.-J. (2008, 26–28 Nov.). People counting system for getting in/out of a bus based on video processing. In *Proc. eighth international conference on intelligent systems design and applications isda '08* (Vol. 3, pp. 565–569).

Chen, T.-H., Chen, T.-Y., & Chen, Z.-X. (2006, 1–3 June). An intelligent people-flow counting method for passing through a gate. In *Proc. ieee conference on robotics, automation and mechatronics* (pp. 1–6).

Chia, A., Leung, M., Eng, H., & Rahardja, S. (2007). Ellipse detection with hough transform in one dimensional parametric space. In *Ieee international conference on image processing, 2007. icip 2007.* (Vol. 5, p. 333-336).

Choi, J.-Y., Sung, K.-S., & Yang, Y.-K. (2007, Sept. 30 2007–Oct. 3). Multiple vehicles detection and tracking based on scale-invariant feature transform. In *Proc. ieee intelligent transportation systems conference itsc 2007* (pp. 528–533).

Comaniciu, D., & Meer, P. (1997, 17–19 June). Robust analysis of feature spaces: color image segmentation. In *Proc. ieee computer society conference on computer vision and pattern recognition* (pp. 750–755).

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Ieee computer society conference on computer vision and pattern recognition, 2005 (cvpr 2005).* (Vol. 1, p. 886-893).

Djouadi, A., Snorrason, ., & Garber, F. (1990). The quality of training sample estimates of the bhattacharyya coefficient. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(1), 92-97.

Eurotech. (2007). *Pcn-1001 passenger counter.* (http://www.eurotech.com/en/products/PCN-1001)

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381-395.

Frigo, M., & Johnson, S. G. (2005, Feb.). The design and implementation of fftw3. *Proceedings of the IEEE*, *93*(2), 216–231.

Fung, J., & Mann, S. (2005). Openvidia: Parallel gpu computer vision. In *Multimedia '05: Proceedings of the 13th annual acm international conference on multimedia* (p. 849-852). New York, NY, USA: ACM.

Gardel, A., Bravo, I., Jimenez, P., Lazaro, J. L., & Torquemada, A. (2007). Real time head detection for embedded vision modules. In *Proceedings ieee international symposium on intelligent signal processing wisp 2007* (p. 1-6).

Garland, M., Le Grand, S., Nickolls, J., Anderson, J., Hardwick, J., Morton, S., et al. (2008, July–Aug.). Parallel computing experiences with cuda. *IEEE Micro*, *28*(4), 13–27.

Gil, A., Mozos, O., Ballesta, M., & Reinoso, O. (2009). A comparative evaluation of interest point detectors and local descriptors for visual slam. *Machine Vision and Applications*.

Gonzalez, R. C., & Woods, R. E. (2008). *Digital image processing (3rd edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Grammalidis, N., & Strintzis, M. G. (2000). Head detection and tracking by 2-d and 3-d ellipsoid fitting. In *Proc. computer graphics international* (pp. 221–226).

Harris, C., & Stephens, M. (1988). A combined corner and edge detection. In *Proceedings of the fourth alvey vision conference* (pp. 147–151).

Hartley, R. I. (1997). In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(6), 580-593.

Hartley, R. I., & Zisserman, A. (2004). *Multiple view geometry in computer vision* (Second ed.; C. U. Press, Ed.). Cambridge University Press.

HellaAglaia. (2007). *Passenger counter avc1*. (http://www.hella.com/produktion/AglaiaEN/WebSite/Channels/NonAutomotiveSolutions/Automatic_Passenger_Counter_AVC1/Automatic_Passenger_Counter_AVC1.jsp)

Hu, H., Qin, J., Lin, Y., & Xu, Y. (2008, 25–27 June). Region covariance based probabilistic tracking. In *Proc. 7th world congress on intelligent control and automation wcica 2008* (pp. 575–580).

Hu, Z., Kawamura, T., & Uchimura, K. (2007). Grayscale correlation based 3d model fitting for occupant head detection and tracking. In *Proc. ieee intelligent vehicles symposium* (pp. 1252–1257).

Isard, M., & Blake, A. (1998). Condensation conditional density propagation for visual tracking. *International Journal of Computer Vision*, *29*(1), 5-29.

Ishii, Y., Hongo, H., Yamamoto, K., & Niwa, Y. (2004, 17–19 May). Real-time face and head detection using four directional features. In *Proc. sixth ieee international conference on automatic face and gesture recognition* (pp. 403–408).

KhronosGroup. (2009). *Opencl the open standard for parallel programming of heterogeneous systems overview.* (http://www.khronos.org/developers/library/overview/opencl_overview.pdf)

Levi, K., & Weiss, Y. (2004, June). Learning object detection from a small number of examples: the importance of good features. In *Proceedings of the 2004 ieee computer society conference on computer vision and pattern recognition (cvpr 2004).* (Vol. 2, p. 53-60).

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*, 91-110.

Lozano, O. M., & Otsuka, K. (2008). Real-time visual tracker by stream processing. *Journal of Signal Processing Systems*.

Luo, Y., & Duraiswami, R. (2008, 23–28 June). Canny edge detection on nvidia cuda. In *Proc. ieee computer society conference on computer vision and pattern recognition workshops cvpr workshops 2008* (pp. 1–8).

Luong, Q., Deriche, R., Faugeras, O., & Papadopoulo, T. (1993). On determining the fundamental matrix: Analysis of diferent methods and experimental results. In *Technical report, institut national de recherche en informatique et en automatique (inria).*

Ma, G., Park, S.-B., Miiller-Schneiders, S., Ioffe, A., & Kummert, A. (2007, Sept. 30 2007–Oct. 3). Vision-based pedestrian detection - reliable pedestrian candidate detection by combining ipm and a 1d profile. In *Proc. ieee intelligent transportation systems conference itsc 2007* (pp. 137–142).

Ma, H., Lu, H., & Zhang, M. (2008, 25–27 June). A real-time effective system for tracking passing people using a single camera. In *Proc. 7th world congress on intelligent control and automation wcica 2008* (pp. 6173–6177).

NVIDIA. (2008). *Nvidia cuda programming guide.* (http://developer.download.nvidia.com/compute/cuda/2_0/docs/NVIDIA_CUDA_Programming_Guide_2.0.pdf)

Oberli, C., & Landau, D. (2008, 21–24 Oct.). Performance evaluation of uhf rfid technologies for real-time passenger tracking in intelligent public transportation systems. In *Proc. wireless communication systems. 2008. iswcs '08. ieee international symposium on* (pp. 108–112).

Opencv reference documentation. (2006). (http://opencv.willowgarage.com/wiki/CvReference)

Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., & Phillips, J. C. (2008, May). Gpu computing. *Proceedings of the IEEE*, *96*(5), 879–899.

Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., KrÃ$\frac{1}{4}$ger, J., Lefohn, A. E., et al. (2007). A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, *26*(1), 80–113.

Pan, L., Gu, L., & Xu, J. (2008, 30–31 May). Implementation of medical image segmentation in cuda. In *Proc. international conference on technology and applications in biomedicine itab 2008* (pp. 82–85).

PointGreyResearch. (2008). *Point grey flea2 firewire camera.* (http://www.ptgrey.com/products/flea2/flea2.pdf)

Rosenfeld, A., & Pfaltz, J. L. (1966). Sequential operations in digital picture processing. *J. ACM*, *13*(4), 471–494.

Rubner, Y., Tomasi, C., & Guibas, L. J. (1998, 4–7 Jan.). A metric for distributions with applications to image databases. In *Proc. sixth international conference on computer vision* (pp. 59–66).

Schmid, C., Mohr, R., & Bauckhage, C. (2000). Evaluation of interest point detectors. *International Journal of Computer Vision*, *37*(2), 151-172.

Smith, A. R. (1978). Color gamut transform pairs. *SIGGRAPH Comput. Graph.*, *12*(3), 12–19.

Tom's hardware - intel core i7 architecture review. (2008). (http://www.tomshardware.com/reviews/Intel-Core-i7-Nehalem,2057.html)

Tuytelaars, T., & Mikolajczyk, K. (2008). *Local invariant feature detectors: A survey* (N. P. Inc., Ed.). Hanover, MA, USA: Now Publishers Inc.

VanRijsbergen, C. J. (1979). *Information retrieval* (2d ed. ed.) [Book]. Butterworths, London ; Boston :.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings ieee computer society conference on computer vision and pattern recognition cvpr 2001* (Vol. 1, p. 511-518).

Wikipedia - geforce9 series. (2008). (http://en.wikipedia.org/wiki/GeForce_9_Series)

Wu, B., & Nevatia, R. (2007). Improving part based object detection by unsupervised, online boosting. In *Proceedings ieee conference on computer vision and pattern recognition cvpr '07* (p. 1-8).

Xie, Y., & Ji, Q. (2002). A new efficient ellipse detection method. In *in international conference on pattern recognition 2002* (Vol. 2, p. 957-960).

Zhang, Z., Deriche, R., Faugeras, O. D., & Luong, Q. T. (1995). A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, *78*(1-2), 87-119.

Zhao, W., Chellappa, R., Phillips, P. J., & Rosenfeld, A. (2003). Face recognition: A literature survey. *ACM Comput. Surv.*, *35*(4), 399–458.