Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Programa de Posgrado en Ciencias en Ciencias de la Computación

---

Seguimiento confiable de objetos basado en métodos de correspondencia

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de

Doctor en Ciencias

Presenta:

Daniel Miramontes Jaramillo

Ensenada, Baja California, México

2016

Tesis defendida por

Daniel Miramontes Jaramillo

y aprobada por el siguiente Comité

Dr. Vitaly Kober

Director del Comité

Dr. Hugo Homero Hidalgo Silva

Dr. Josué Álvarez Borrego

Dr. Víctor Hugo Díaz Ramírez

Dr. Jesús Favela Vara

Coordinador del Programa de Posgrado en Ciencias de la Computación

---

Dr. Rufina Hernández Martínez

Director de Estudios de Posgrado

Resumen de la tesis que presenta Daniel Miramontes Jaramillo como requisito parcial para la obtención del grado de Doctor en Ciencias en Ciencias de la Computación.

Seguimiento confiable de objetos basado en métodos de correspondencia

Resumen aprobado por:

Dr. Vitaly Kober

El sentido de la vista es una de las características más preciadas por el ser humano, pues es mediante ésta que podemos percibir el mundo a nuestro alrededor. Es por ello que la ciencia ha investigado desde hace varios siglos los mecanismos de visión. Recientemente, con el desarrollo de nuevas tecnologías se ha logrado avanzar en la ciencia al dotar a robots de visión y aumentar nuestra propia visión con nuevas capacidades.

El problema de seguimiento se define como la localización persistente de un objeto a lo largo de una secuencia de video, a cada imagen en la secuencia se le llama cuadro, y se dice que la secuencia es en tiempo real si se presentan 30 cuadros por segundo (FPS). Un algoritmo de seguimiento devuelve la trayectoria del objeto a través de la secuencia.

En el presente trabajo de investigación se estudiaron diversos algoritmos de seguimiento, y se realizó un estudio comparativo de éstos contra un algoritmo propuesto para medir la viabilidad de este último. Todo algoritmo de seguimiento cuenta con tres elementos principales: 1) el modelo del objeto, es una interpretación matemática del objeto, cuantitativa y estándar, que pueda ser reproducida consistentemente en cualquier tipo de escena donde se encuentre el objeto, en nuestro caso utilizamos histogramas de gradiente orientado como descriptor del objeto; 2) el detector, o módulo de correspondencia, el cual se encarga de buscar al modelo del objeto en el área del cuadro a procesar, en este trabajo se utiliza un procesamiento iterativo de actualización de histogramas, extrayendo descriptores de todo el cuadro para finalmente realizar una comparación del modelo del objeto con los descriptores de la escena para determinar la posición del objeto en cada cuadro; 3) el modelo de movimiento, el cual trata de predecir la posición del objeto en el siguiente cuadro a procesar, en este trabajo se usa un modelo de series de tiempo para ajustar el movimiento del objeto a una parábola mediante regresión cuadrática de mínimos cuadrados.

El algoritmo propuesto se implementó en una tarjeta gráfica para el cómputo concurrente de datos en cada uno de sus módulos, procesando las secuencias de video en tiempo real. Las secuencias de video se adquieren mediante la cámara Kinect de Microsoft, la cual proporciona imágenes de profundidad utilizadas para la segmentación del objeto y manejo de oclusión.

Palabras Clave: **Seguimiento, ventanas circulares, histogramas de gradiente orientado, mapa de profundidad, procesamiento paralelo de imágenes**

Abstract of the thesis presented by Daniel Miramontes Jaramillo as a partial requirement to obtain the Doctor of Science degree in Computer Sciences.

Reliable object tracking based on matching methods

Abstract approved by:

Dr. Vitaly Kober

Visual perception is one of the most appreciated features of the human, because it helps us to perceive and interpret the surrounding environment by processing information contained in visible light. Few centuries ago, the science has understood the mechanisms of our vision. Recently, with the development of new technologies, the science has achieved significant advances in robot vision and other vision-based technical systems.

The tracking problem is to persistently locate an object of interest in a video sequence. Each image in the sequence is known as frame, the processing carries out in real time if the rate is 30 frames per second (FPS). A tracking algorithm returns the object trajectory by means of the video sequence.

In this work, we propose a new tracking algorithm and perform a comparison study of the proposed algorithm with the State-Of-The-Art tracking algorithms to measure the viability of our proposal. Basically, tracking algorithms have three elements: 1) the mathematical model of object and scene signals; the model can be consistently reproduced in any scene where the object is present, for this purpose, in this work we use the histogram of oriented gradients as the object descriptor; 2) the detector (matching module) is the search mechanism of the object in any frame to be processed; in this work we use an iterative histogram update process, computing descriptors from frames to compare against the object descriptors in order to locate the object position in each frame; 3) the motion model, this module predicts the object position in the next frame to be processed; in this work we use a time series model to fit the object movement to a parabola by the least squares quadratic regression.

The proposed algorithm is implemented in a graphics card for the concurrent processing of data at each module of the algorithm to achieve real time processing. The video sequences are acquired with the Microsoft Kinect camera; this camera provides depth images used for image segmentation and occlusion management.

Keywords: **Tracking, circular windows, histogram of oriented gradients, depth map, parallel image processing**

Dedicatoria

A Pamela, mi soporte, guía

y fuente de inspiración.

Agradecimientos

Quisiera expresar mi sincero agradecimiento:

A Pamela por su amor, soporte, paciencia y tolerancia aún en los momentos más difíciles, no lo hubiera logrado sin ti.

A mi familia por su apoyo incondicional y paciencia, aún y cuando me vieran estresado y me dijeran: "...te ves estresado, ya deja de estudiar mijo...".

Al Dr. Vitaly Kober por todas sus enseñanzas, consejos y paciencia cuando batallaba en entender lo que me quería explicar.

A los miembros de mi comité de tesis: Dr. Hugo Homero Hidalgo Silva, Dr. Josué Álvarez Borrego y Dr. Víctor Hugo Díaz Ramírez, por sus comentarios y aportes a mi trabajo.

A todos los doctores, maestros, y demás personal del Departamento de Ciencias de la Computación por todas las lecciones que me enseñaron dentro y fuera del aula.

A mis compañeros, por hacer amena mi estadía en estos años de estudio, por compartir sus experiencias y complementar el conocimiento. Gracias a todos, Valeria, Julia, Adriana, América, Mario (el Krus), Fermín, Vanesa, Hugo, Armando, Franceli y a los demás que estuvieron a la hora de la comida, gracias sodomitas.

Al Centro de Investigación Científica y de Educación Superior de Ensenada por darme la oportunidad de realizar mis estudios de posgrado.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme el apoyo económico para realizar mis estudios de doctorado mediante la beca No. 242880.

Tabla de contenido

Página

Lista de figuras

Figura Página

Figura Página

Figura Página

Figura Página

Figura Página

Figura Página

Lista de tablas

Capítulo 1. Introducción

Los seres vivos tenemos la capacidad de percibir al mundo que nos rodea por medio de diversos estímulos; cada estímulo es captado por un sensor el cual lo convierte a impulsos eléctricos que son procesados por el sistema nervioso central del ser vivo. Según el tipo de estímulo hemos clasificado los sensores en cinco categorías o sentidos: tacto, gusto, olfato, audición y vista. Cada sensor nos permite describir un aspecto del mundo en particular, haciendo que la falta de uno o varios sentidos dificulte nuestra capacidad de enfrentar los diversos problemas que se nos presentan día a día.

Debido a la enorme importancia que tienen nuestros sentidos para desarrollarnos en el ambiente, constantemente estamos estudiando y mejorando nuestro entendimiento sobre ellos con diversos objetivos; desde mejorar nuestras capacidades naturales con dispositivos como telescopios, microscopios, antenas e incluso técnicas para mejorar el sabor y olor de los alimentos; hasta dispositivos que reemplacen algún sentido perdido o deteriorado, como dispositivos de audio, lentes y cirugías láser para los ojos, prótesis, entre otras.

Uno de los sentidos que consideramos entre los más importantes es la vista, ya que nos presenta gráficamente el mundo a nuestro alrededor. Mediante la vista podemos describir cualquier objeto con una cantidad enorme de características y clasificarla según el aprendizaje que hayamos tenido sobre ellas; podemos mantener la atención en un objeto específico tanto tiempo como lo deseemos aún cuando sufra transformaciones durante el tiempo de exposición. A estos procesos les denominamos correspondencia y seguimiento.

Con el objetivo de entender estos procesos, múltiples disciplinas estudian los sistemas visuales, entre éstas estan las ciencias computacionales; en las cuales se trata de proveer al sistema computacional con los medios, procesos, técnicas y algoritmos para procesar el mundo visual. Estos procesos que son naturales para los seres vivos y nuestro cerebro los maneja subconscientemente, un sistema computacional requiere la especificación detallada de lo que debe hacer para el objetivo que requiere hacerlo. Así pues, mientras un ser vivo realiza operaciones de ajuste de iluminación, reconocimiento, cambios de perspectiva y escala, entre muchas otras cosas en tiempo real y, a nuestro

parecer, sin mucho esfuerzo, el sistema computacional requiere un conjunto de complicados algoritmos y una gran cantidad de operaciones matemáticas para realizar solo una de la tareas descritas anteriormente.

La investigación y tecnología que ha evolucionado hasta el día de hoy nos permiten tener conocimientos avanzados de nuestra visión y dispositivos de captura de imágenes con mayores capacidades que las del ojo humano, como percibir luz infrarroja para generar imágenes invariantes a iluminación. Con este conocimiento y tecnología previos podemos seguir avanzando la ciencia para construir aplicaciones más rápidas y confiables que nos permitan dotar a sistemas computacionales y robots con sistemas de visión cada vez más parecidos al nuestro, donde se tenga una visión general que resuelva todos los problemas en tiempo real de manera constante y no solo un conjunto de problemas para alguna aplicación específica por un tiempo limitado.

Para ello, debemos trabajar en mejorar los sistemas actuales, hacer que se ejecuten en tiempo real y de manera concurrente. Luego, debemos aplicar el conocimiento que tenemos, resolviendo los problemas de visión. El problema tratado en este trabajo es el seguimiento, el cual tiene variadas aplicaciones no solo en el área computacional y robótica, sino también en varias ramas de la ciencia y actividades humanas; por ejemplo, las ciencias naturales, seguridad, medicina, manufactura, entre otras. Y en muchas de estas actividades está implícita la necesidad de conocer la posición y trayectoria de los objetos en un campo tridimensional.

1.1. Definición del problema

Dada una secuencia de imágenes o una secuencia de video, ubicar y dar seguimiento a un objeto o a un grupo de objetos específicos. Garantizar un grado de confiabilidad respecto al desempeño del seguimiento en diversas situaciones que se pueden presentar como oclusión del objeto, salida del objeto de la escena, entrada del objeto en la escena, variaciones en la iluminación, ruido en la escena o ligeras transformaciones geométricas implícitas del objeto o la escena como el cambio de apariencia de éstos. El problema de localización del objeto consiste en poder diferenciarlo de otros objetos y de la escena por la cual transita, donde los mapas de profundidad pueden ayudar a resaltar los objetos y eliminar el fondo, facilitando la localización.

1.2. Objetivos de la investigación

1.2.1. Objetivo general

Desarrollar métodos de seguimiento de objetos tridimensionales basados en métodos de correspondencia en secuencias de video en tiempo real, en las que exista un grado de confiabilidad respecto a distintas métricas de desempeño, tolerancia a ligeras distorsiones geométricas y ruido.

1.2.2. Objetivos específicos

1. Obtener el conocimiento teórico de los métodos de seguimiento actuales.
2. Desarrollar métodos de seguimiento de objetos basados en métodos de correspondencia capaces de detectar objetos en imágenes y videos en escala de grises en tiempo real.
3. Desarrollar métodos de seguimiento de objetos basados en métodos de correspondencia mediante el uso de mapas de profundidad capaces de detectar objetos en imágenes y videos en escala de grises en tiempo real.
4. Desarrollar métodos de seguimiento de objetos basados en métodos de correspondencia capaces de detectar objetos en imágenes y videos en escala de grises degradados por ruido en tiempo real.
5. Analizar los métodos propuestos respecto a tolerancia a ruido, tolerancia a distorsiones geométricas y tiempo de ejecución.
6. Comparar los métodos propuestos con los métodos de seguimiento existentes.
7. Desarrollar el sustento matemático formal de los métodos propuestos.
8. Determinar la complejidad computacional teórica y determinar la viabilidad práctica de los métodos desarrollados.

1.3. Limitaciones y suposiciones

- Se supone que las secuencias de video de entrada están degradadas por algún tipo de ruido que es posible modelar.

- Se supone que los objetos a seguir en los videos o secuencias de imágenes deben estar presentes en algún momento en el transcurso del tiempo de exposición de éstos.
- Se supone que los objetos pasan por una serie de ligeras transformaciones geométricas como cambios de escala y rotaciones que modifican la apariencia de éstos.
- El trabajo se centra en objetos tridimensionales rígidos con pocas o ninguna articulación.
- Se van a utilizar diversas fuentes de adquisición de imágenes y videos como cámaras CCD y cámaras Kinect para crear los mapas de profundidad.
- Las secuencias de video tienen un tamaño de 640×480 pixeles con una frecuencia de 30 FPS, que es lo máximo que proporciona la cámara Kinect al utilizar imágenes de profundidad.
- Las tarjetas gráficas (GPU) tienen limitaciones de memoria y unidades de cómputo para el proceso de datos concurrentes.
- El tipo de tarjeta gráfica limita el lenguaje y las características del código paralelo.

1.4. Trabajo previo

El seguimiento puede ser abordado desde dos perspectivas diferentes: seguimiento por movimiento o seguimiento por detección. Los algoritmos de seguimiento por movimiento detectan el movimiento del objeto, estimando su posición, velocidad y dirección de acuerdo a las transformaciones de los pixeles en los cuadros anteriores. Estos algoritmos requieren ser inicializados para producir trayectorias finas; sin embargo, tienden a acumular errores y suelen fallar en caso de perder de vista al objeto. Los algoritmos de seguimiento por detección estiman la posición del objeto en cada cuadro de video de manera independiente, y a menudo requieren entrenamiento fuera de línea (Kalal *et al.*, 2010).

El algoritmo presentado en este trabajo es un algoritmo híbrido, pues detecta al objeto en cada cuadro de video en un fragmento del cuadro extraído mediante un modelo de

movimiento de la información del objeto en cuadros anteriores. El algoritmo propuesto utiliza una serie de características como los Histogramas de Gradiente Orientado (HOG, por sus siglas en inglés) (Dalal y Triggs, 2005) en una estructura geométrica de ventanas circulares para la correspondencia y un modelo de movimiento junto al uso de mapas de profundidad para el seguimiento. Se considera un algoritmo de seguimiento por movimiento y detección.

El primer paso para el seguimiento por detección es contar con un algoritmo de correspondencia confiable, existen diversos métodos para realizar la correspondencia como filtraje por correlación y los métodos basados en características.

En los métodos basados en filtraje (Manzur *et al.*, 2012; Aguilar-González y Kober, 2012), se crean un conjunto de filtros desde diversos puntos de vista del objeto, cada filtro contiene alguna degradación como rotación en y fuera de plano, escala, cambios de iluminación, ruido, entre otras. La medida de similitud se realiza mediante la correlación del conjunto de filtros y la imagen de entrada. La desventaja de estos métodos es que su calidad y velocidad depende del número de filtros utilizados, donde los filtros comunmente fallan en la localización del objeto deformado por rotaciones fuera de plano.

Los métodos basados en características calculan un descriptor en el objeto. Este descriptor puede ser dado como puntos de interés en una región específica de la imagen como el caso de la Transformada de Características Invariante a Escala (SIFT, por siglas en inglés) (Lowe, 2004), las Características Robustas Aceleradas (SURF, por sus siglas en inglés) (Bay *et al.*, 2008), el FAST orientado y BRIEF Rotado (ORB, por sus siglas en inglés) (Rublee *et al.*, 2011) y sus variantes (Calonder *et al.*, 2010; Ortiz, 2012; Liao *et al.*, 2011), los cuales calculan vectores de características como histogramas de gradiente orientado o cadenas binarias tanto en el objeto como en la imagen para después compararlos entre sí.

Los descriptores de plantilla, toman una región de la imagen obteniendo histogramas de intensidad, color y/o gradiente, así como otras características para localizar al objeto en la escena (Tate y Northern, 2008; Sarvaiya *et al.*, 2009; Zalesky y Lukashevich, 2011; Miramontes-Jaramillo *et al.*, 2014). Estos descriptores son sencillos de implementar en sistemas paralelos como sistemas optodigitales (Manzur *et al.*, 2012; Díaz-Ramírez y Ko-

ber, 2007) o tarjetas gráficas (GPU) (Ouerhani *et al.*, 2013; Sanders y Kandrot, 2010).

Una vez seleccionado un descriptor, el algoritmo de correspondencia robusto procede a utilizar técnicas de seguimiento o modelos de movimiento. Dependiendo de la forma que tenga el descriptor, es posible clasificar al seguidor de diferentes formas; por ejemplo, seguidores de puntos (Buchanan y Fitzgibbon, 2007; Intille *et al.*, 1997; Sbalzarini y Koumoutsakos, 2005; Sethi y Jain, 1987), que consiste en obtener características alrededor de diversos puntos del objeto y compararlos con puntos extraídos en la escena; seguidores de silueta (Haritaoglu *et al.*, 2000; Sato y Aggarwal, 2004; Talu *et al.*, 2011), que consisten en seguir un conjunto de puntos que conforman la silueta del objeto concurrentemente; seguidores de plantilla (Schweitzer *et al.*, 2002; Nejhum *et al.*, 2010; Fieguth y Terzopoulos, 1997), que consisten en el proceso de extracción de características dentro de una ventana deslizante sobre un cuadro de video, las cuales se compararán con las características del objeto.

Los algoritmos de seguimiento del Estado-Del-Arte suelen ser algoritmos de plantilla que utilizan un conjunto de bolsas de características para entrenamiento como el algoritmo de Aprendizaje de Múltiples Instancias en Línea (OMIL, por sus siglas en inglés) (Babenko *et al.*, 2009), para clasificar características dentro y fuera del área de soporte del objeto; Seguimiento–Aprendizaje–Detección (TLD, por sus siglas en inglés) (Kalal *et al.*, 2010), que consiste en un conjunto de *expertos* que determinan el error de localización del objeto y lo corrigen de ser necesario. Este par de algoritmos actualizan la vista del objeto en cada iteración. El Seguimiento por RGBD, Oclusión y Flujo Óptico (RGBDOcc+OF, por sus siglas en inglés) (Song y Xiao, 2013), consiste en la extracción de diversos descriptores de gradiente en un espacio tridimensional con información de profundidad; y el Seguimiento por Flujo Óptico Conciente de Oclusión (OAPF, por sus siglas en inglés) (Meshgi *et al.*, 2014), se basa en el algoritmo de flujo de partículas sobre información de profundidad para determinar el movimiento del objeto. Estos últimos dos algoritmos utilizan información de profundidad para manejar la oclusión del objeto.

1.5. Organización de la tesis

Este trabajo se encuentra organizado de la siguiente manera: en el Capítulo 2 se presentan los fundamentos matemáticos, de teoría de algoritmos y tecnológicos nece-

sarios para el desarrollo del algoritmo propuesto y de los algoritmos presentados. En el Capítulo 3 se presentan los algoritmos de correspondencia del Estado-Del-Arte que se utilizaron para comparar el desempeño del módulo de detección del algoritmo propuesto. El Capítulo 4 presenta los algoritmos de seguimiento del Estado-Del-Arte que se utilizaron para comparar el desempeño del algoritmo propuesto. En el Capítulo 5 se presentan las bases de datos y puntos de referencia que se utilizaron para medir el desempeño de los algoritmos; también se describen los criterios de evaluación de los algoritmos. El Capítulo 6 presenta detalladamente el algoritmo propuesto, en cada sección se describen los diferentes aspectos del algoritmo así como su complejidad teórica, finalizando con los resultados del desempeño de los algoritmos evaluados. En el Capítulo 7 se presentan las conclusiones del trabajo realizado, se propone trabajo futuro para seguir desarrollando el algoritmo y se presentan las publicaciones en revistas y congresos que se realizaron a lo largo del trabajo de tesis. Finalmente se presenta la lista de referencias bibliográficas así como dos Apéndices, el primero con el código de inicialización del GPU con OpenCL y el segundo con la derivación de la regresión cuadrática de mínimos cuadrados.

Capítulo 2. Fundamentos

En este capítulo se describen los fundamentos y conceptos necesarios para una mejor comprensión del trabajo de tesis. Se definen tres secciones: primero, los fundamentos matemáticos, donde se definen los conceptos que sientan las bases del algoritmo propuesto y los algoritmos del Estado-Del-Arte presentados; segundo, los fundamentos de algoritmos, donde se explican conceptos de teoría de algoritmos y cálculo de tiempo de ejecución, tanto para un procesador como para multiprocesadores; finalmente, fundamentos tecnológicos, donde se discuten las características y funcionamiento de las cámaras de profundidad y los GPU.

2.1. Fundamentos matemáticos

2.1.1. Imagen

Una imagen es una distribución espacial de radiación en un plano finito. Matemáticamente, esta distribución espacial puede ser representada como una función bidimensional continua $I(x, y)$. Sin embargo, dado que las computadoras no pueden manejar imágenes de funciones continuas, sino solo arreglos de números digitales, es necesario representar las imágenes como un arreglo de puntos bidimensional.

Cada punto en la rejilla bidimensional se denomina *pixel* o *pel*, que significa *elemento de imagen*. El pixel representa la radiación o intensidad en cada posición de la rejilla (x, y), por lo general se encuentran en un patrón rectangular, aunque existen otras formas de rejillas bidimensionales (ver Figura 1).

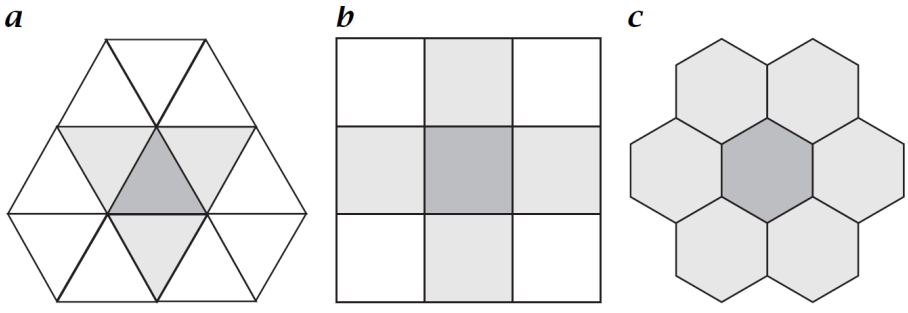

Figura 1: Rejillas bidimensionales comunmente utilizadas: a) rejilla triangular, b)rejilla rectangular, c) rejilla hexagonal. Imagen recuperada de (Jähne, 2002), pág. 32.

Para que la función de una imagen continua $I(x, y)$ sea útil para procesamiento digital, debe ser digitalizada. Para ello, el dispositivo de captura de imágenes realiza dos operaciones: muestreo y cuantización. El muestreo determina la resolución espacial de la imagen digital; es decir, la cantidad de muestras tomadas de la señal análoga de tal forma que la señal digital conserve la forma general de la señal original. Como se puede ver en la Figura 2, el conjunto de muestras describen la señal original; sin embargo existen detalles que se pierden mientras la cantidad de muestras disminuyen, por lo que sería difícil reconstruir la señal original perfectamente.

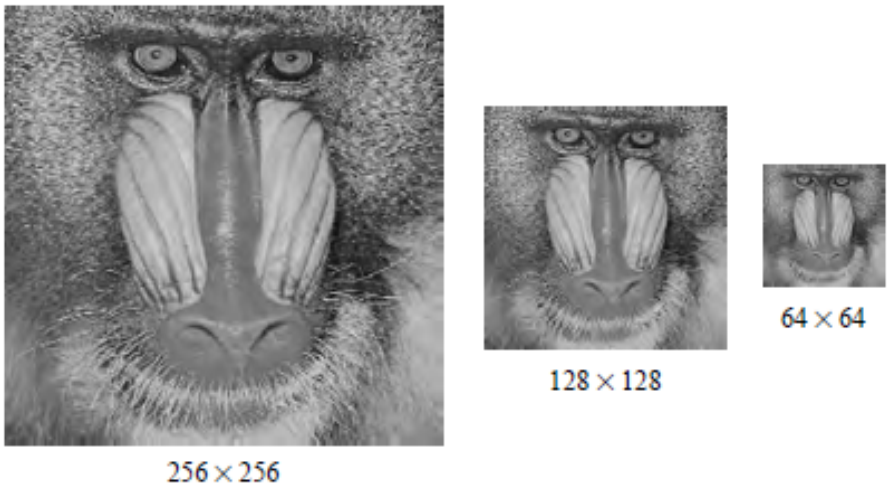

Figura 2: Imagen con tres diferentes resoluciones de muestreo. Los detalles disminuyen proporcionalmente con el número de muestras tomadas. Imagen recuperada de (Bovik, 2009), pág. 10.

La cuantización define el rango discreto de los valores que puede tomar cada elemento de imagen, normalmente se le llama intensidad, ya que es un registro de la intensidad de la radiación que incide en el sensor. La cantidad de radiación debe ser mapeada a un número limitado de valores discretos. Decimos que una imagen es blanco y negro cuando tenemos solo dos niveles; cuando tenemos un rango de valores entre blanco y negro, decimos que la imagen está en escala de grises; si la imagen tiene varios canales, se dice que la imagen es a color. Generalmente las imágenes digitales se cuantizan a 256 valores por canal, los cuales contienen 8 bits por cada canal. Otros niveles de cuantización comunes son de 12, 16 y 24 bits (Jähne, 2002; Bovik, 2009). La Figura 3 muestra un ejemplo de cuantización.

Una secuencia de imágenes a través del tiempo se denomina secuencia de video; luego la función de imagen agrega una tercera componente $I(x, y, k)$, donde k es el tiem-

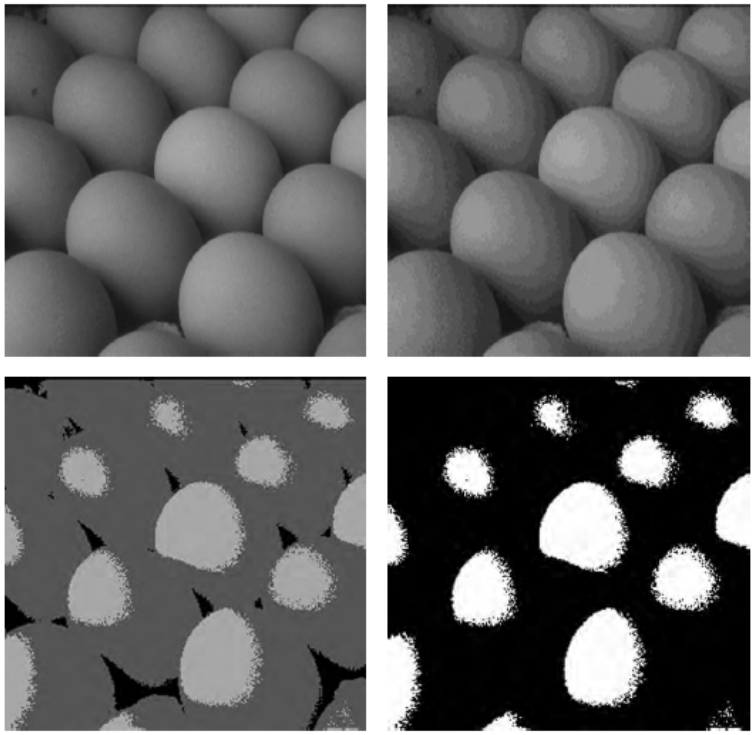

Figura 3: Imagen mostrada con diferentes niveles de cuantización: arriba a la izquierda 8 bits, arriba a la derecha 4 bits, abajo a la izquierda 2 bits y abajo a la derecha 1 bit. Imagen recuperada de (Bovik, 2009), pág. 18.

po en que la imagen (o cuadro) $I(x, y)$ fue capturada. La velocidad de una secuencia de video se mide en Cuadros Por Segundo (FPS, por sus siglas en inglés), lo cual determina la fluidez de movimiento en la escena al ser percibida por el ojo humano. El estándar de video se encuentra en los 30 FPS; sin embargo, en los últimos años, con el avance de la tenología, las nuevas películas y animaciones se capturan en 60 FPS, presentando una mejor calidad de video. Existen también cámaras de propósito específico que capturan video a 120 FPS, esta es la llamada *super velocidad* y es utilizada en el ámbito científico para capturar a detalle algunos fenómenos naturales.

2.1.2. Transformada de Fourier

El análisis de frecuencia es una metodología poderosa y ampliamente usada, la cual se centra en una serie de herramientas como las transformadas de Fourier, Laplace y Z, entre otras. La transformada de Fourier es un tipo particular de transformada integral que nos permite ver a las imágenes y al procesamiento de imágenes desde una perspectiva alterna al transformar el problema a un espacio diferente.

El procesamiento de imágenes se basa en distribuciones, o funciones espaciales bidimensionales de intensidad o color que existen, comunmente, en un espacio real. La transformada de Fourier actúa sobre estas funciones para producir una forma equivalente en un espacio abstracto llamado *espacio de frequencia*.

La idea fundamental del análisis de Fourier es que cualquier señal, ya sea en función de tiempo, espacio u otra variable, puede ser expresada como una combinación lineal ponderada de funciones armónicas, como el seno y el coseno, teniendo diferentes periodos o frecuencias. Estas son llamadas componentes de frecuencia espacial de la señal.

La transformada de Fourier es una extensión de las series de Fourier que resulta cuando un periodo de la función representada se extiende y se aproxima al infinito. La transformada de una función es expresada como la descomposición ponderada de un conjunto de funciones armónicas.

La transformada de Fourier $\mathcal{F}$ transforma una función $f(t)$ a una representación en el dominio de la frecuencia, $\mathcal{F}\{f(t)\} = \mathrm{F}(\omega)$. La función compleja F se denomina como espectro de frecuencia complejo, en el cual es sencillo visualizar proporciones relativas de diferentes frecuencias. Luego, la transformada de Fourier $\mathcal{F}$ de una función continua está dada por

$$\mathcal{F}\{f(t)\} = \mathrm{F}(\omega) = \int_{-\infty}^{\infty} f(t)\mathrm{e}^{-2\pi i\omega t}dt\,. \tag{1}$$

La transformada inversa de Fourier $\mathcal{F}^{-1}$ está dada por

$$\mathcal{F}^{-1}\{\mathrm{F}(\omega)\} = f(t) = \int_{-\infty}^{\infty} \mathrm{F}(\omega)\mathrm{e}^{2\pi i\omega t}d\omega\,. \tag{2}$$

En el caso de señales discretas, una función continua f es muestreada en intervalos regulares de la forma $f(n)$, donde $n = 0, ..., N-1$; y la transformada discreta de Fourier (DFT) se define como

$$\mathrm{F}(k) = \frac{1}{N}\sum_{n=0}^{N-1} f(n)\mathrm{e}^{-2\pi i\frac{nk}{N}}\,, \tag{3}$$

y su inversa como

$$f(n) = \sum_{k=0}^{N-1} \mathrm{F}(k)\mathrm{e}^{2\pi i\frac{nk}{N}}\,. \tag{4}$$

El espectro $\mathrm{F}(k)$ es periódico con periodo N.

La transformada de Fourier siempre existe para señales digitales, incluyendo imágenes, pues éstas presentan límites y tienen un número finito de discontinuidades. La transformada de fourier unidimensional puede ser fácilmente generalizada a dos dimensiones, en el cual la transformada de Fourier bidimensional también utiliza funciones armónicas para su descomposición espectral. Sea f una función de dos coordenadas (x, y) en un plano que representa a una imagen. La transformada de Fourier 2D para la imagen continua f esta definida por la integral

$$\mathrm{F}(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \mathrm{e}^{-2\pi i(xu+yv)} dxdy \,, \tag{5}$$

y su transformada inversa esta dada por

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathrm{F}(u, v) \mathrm{e}^{2\pi i(xu+yv)} dudv \,. \tag{6}$$

Por simplicidad podemos definir $\mathcal{F}\{f(x, y)\} = \mathrm{F}(u, v)$; luego podemos derivar las siguientes propiedades correspondientes a la transformada de Fourier unidimensional desde el punto de vista del procesamiento de imágenes

- Linealidad

$$\mathcal{F}\{af_1(x, y) + bf_2(x, y)\} = a\mathrm{F}_1(u, v) + b\mathrm{F}_2(u, v) \,. \tag{7}$$

- Desplazamiento del origen en el dominio de la imagen

$$\mathcal{F}\{f(x - a, y - b)\} = \mathrm{F}(u, v) \mathrm{e}^{-2\pi i(au+bv)} \,. \tag{8}$$

- Desplazamiento del origen en el dominio de la frecuencia

$$\mathcal{F}\{f(x, y) \mathrm{e}^{2\pi i(u_0 x+v_0 y)}\} = \mathrm{F}(u - u_0, v - v_0) \,. \tag{9}$$

- Si $f(x, y)$ es real, entonces

$$\mathrm{F}(-u, -v) = \mathrm{F}^*(u, v) \,. \tag{10}$$

donde $(*)$ denota complejo conjugado.

- Convolución y correlación, los cuales serán tratados a detalle en las siguientes subsecciones.

De manera similar a la transformada de Fourier en una dimensión, la transformada de Fourier bidimensional puede ser usada para imágenes discretas reemplazando las integrales por sumas de la siguiente manera

$$\mathrm{F}(u,v) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\mathrm{e}^{-2\pi i\left(\frac{mu}{M}+\frac{nv}{N}\right)}\,, \tag{11}$$

donde $u = 0, 1, ..., M-1$ y $v = 0, 1, ..., N-1$; y su inversa como

$$f(m,n) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} \mathrm{F}(u,v)\mathrm{e}^{2\pi i\left(\frac{mu}{M}+\frac{nv}{N}\right)}\,, \tag{12}$$

donde $m = 0, 1, ..., M-1$ y $n = 0, 1, ..., N-1$.

Las aplicaciones de la transformada de Fourier en el procesamiento digital de señales e imágenes son muy variadas: asistencia en el filtraje de ruido, detección de bordes y esquinas en la función de imagen, restauración de imágenes corruptas, correspondencia basada en convolución, caracterización de siluetas, compresión de imágenes, entre muchas otras (Solomon y Breckon, 2011; Taneja, 2008; Sonka *et al.*, 2008).

2.1.3. Convolución

Una convolución es un operador matemático que transforma dos funciones f y g en una tercera función que representa la magnitud en la que se superponen f y una versión trasladada e invertida de g. Una convolución es un tipo muy general de media móvil.

Un impulso ideal es una señal de entrada importante denominado usando la distribución de Dirac $\delta(x,y)$, para la cual

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \delta(x,y)dx\,dy = 1\,, \tag{13}$$

donde $\delta(x,y) = 0$ para todo $(x,y) \neq 0$.

La *propiedad de cernimiento* de la distribución de Dirac provee el valor de la función $f(x, y)$ en el punto (λ, μ)

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y)\delta(x - \lambda, y - \mu)dx\, dy = f(\lambda, \mu)\,. \tag{14}$$

La ecuación de cernimiento es usada para describir el proceso de muestreo de una función de imagen continua $f(x, y)$. Al definir una función de imagen como una combinación lineal de pulsos de Dirac en los puntos (a, b) cubriendo todo el plano de la imagen, luego el muestreo es ponderado por la función de imagen $f(x, y)$

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(a, b)\delta(a - x, b - y)da\, db = f(x, y)\,, \tag{15}$$

lo cual se conoce como *convolución*.

La convolución ($\otimes$) es una operación importante en el análisis lineal del procesamiento de imágenes, pues es una integral que expresa la cantidad de empalme de una función $f(t)$ mientras se desplaza sobre otra función $h(t)$, la cual ha sido invertida. La convolución define el proceso de filtraje lineal usando el filtro h. La convolución unidimensional $f \otimes h$ de las funciones f y h en un rango finito $[0, t]$ está dado por

$$(f \otimes h)(t) = \int_0^t f(t)h(t - \tau)d\tau\,. \tag{16}$$

La integral de convolución tiene límites $-\infty$, ∞; sin embargo, podemos restringirla al intervalo $[0, t]$, pues se supone que las coordenadas negativas contienen valores iguales a cero.

La convolución satisface un conjunto de propiedades importantes. Sean f, g y h funciones, y a una constante escalar, luego

$$f \otimes h = h \otimes f\,, \tag{17}$$

$$f \otimes (g \otimes h) = (f \otimes g) \otimes h\,, \tag{18}$$

$$f \otimes (g + h) = (f \otimes g) + (f \otimes h)\,, \tag{19}$$

$$a(f \otimes g) = (af) \otimes g = f \otimes (ag)\,. \tag{20}$$

La convolución puede ser generalizada a mayores dimensiones; en el alcance de este trabajo, la convolución bidimensional de las funciones f y h se define como $f \otimes h$ y está dada por la integral

$$\begin{aligned}(f \otimes h)(x,y) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(a,b)h(x-a,y-b)da\,db \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-a,y-b)h(a,b)da\,db \\ &= (h \otimes f)(x,y)\,.\end{aligned} \tag{21}$$

En procesamiento de imágenes, la convolución discreta es expresada por medio de sumas en lugar de integrales, y con límites en el plano de la imagen

$$(f \otimes h)(x,y) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(x-m,y-n)h(m,n)\,; \tag{22}$$

sin embargo, los límites no previenen el uso de la convolución, ya que se asume que fuera del dominio de la imagen existe un infinito de ceros. Los filtros lineales son comunmente utilizados para preprocesamiento local de imágenes y para restauración de imágenes.

Las operaciones lineales calculan el valor resultante de un pixel $g(i,j)$ en la imagen de salida como una combinación lineal de las intensidades locales del vecindario $\mathcal{V}$ del pixel $f(i,j)$ de la imagen de entrada. La contribución de los pixeles en el vecindario son ponderados por coeficientes h

$$f(i,j) = \sum_{(m,n)\in\mathcal{V}} h(i-m,j-n)g(m,n)\,. \tag{23}$$

Esto es el equivalente a la operación de convolución con una plantilla h, y se le llama máscara u operador de convolución. Los vecindarios $\mathcal{V}$ son elegidos de dimensiones impares tanto en columnas como en filas, con la finalidad de tener un pixel central definido; aunque existen algunas excepciones, como el operador de gradiente de Roberts que se verá más adelante (Sonka *et al.*, 2008).

2.1.3.1. Teorema de convolución

Teorema 2.1 *El teorema de convolución establece que la relación existente entre dos funciones y sus transformadas de Fourier está dada por*

$$f(x,y) \otimes h(x,y) \Leftrightarrow \mathrm{F}(u,v)\mathrm{H}(u,v)\,, \tag{24}$$

y

$$f(x,y)h(x,y) \Leftrightarrow \mathrm{F}(u,v) \otimes \mathrm{H}(u,v)\,, \tag{25}$$

donde F *y* H *son las transformadas de fourier de las funciones* f *y* h *respectivamente (Gonzalez y Woods, 2002).*

2.1.4. Correlación

En estadística, la correlación entre dos variables aleatorias X y Y es una medida numérica de *similitud* de la relación lineal entre ellas. Existen varios coeficientes para calcular la correlación entre dos variables. Si la distribución de datos es normal se usa el *coeficiente de correlación de Pearson*, que es conocido simplemente como *coeficiente de correlación*, y está dado por

$$corr(\mathrm{X},\mathrm{Y}) = \frac{cov(\mathrm{X},\mathrm{Y})}{\sigma_{\mathrm{X}}\sigma_{\mathrm{Y}}} = \frac{\mathrm{E}[(\mathrm{X}-\mu_{\mathrm{X}})(\mathrm{Y}-\mu_{\mathrm{Y}})]}{\sigma_{\mathrm{X}}\sigma_{\mathrm{Y}}}\,, \tag{26}$$

donde cov es la covarianza entre las variables, σ es la desviación estándar de las variables aleatorias, μ es el valor esperado o media de una variable aleatoria y E es el operador de valor esperado. Si la distribución de datos no es normal, el coeficiente de correlación puede ser de Spearman o de Kendall.

La correlación de Pearson está definida sí y solo sí las desviaciones estandar son finitas y diferentes de cero; y tiene un rango de $[-1,1]$. La correlación es 1 cuando existe una relación lineal perfecta entre las variables y 0 cuando las variables no tienen ninguna relación entre ellas. Los valores entre -1 y 1 indican el grado de dependencia lineal entre las variables.

La correlación entre dos funciones bivariadas $f(x,y)$ y $h(x,y)$ con media 0 está definida por

$$f(x,y) \circ h(x,y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f^*(m,n) h(x+m, y+n)\,, \tag{27}$$

donde $\circ$ denota la operación de correlación, f^* es el complejo conjugado de f, y M y N son los límites de la función. Las imágenes son, por lo general, funciones reales y pares, en cuyo caso $f^* = f$. La función de correlación tiene la misma forma que la función de convolución con la excepción de complejo conjugado y el cambio de signos en uno de los términos de la operación.

La principal aplicación de la correlación es como una medida de similitud de correspondencia. Sea $f(x,y)$ una imagen que contenga un objeto de interés $h(x,y)$; luego, si existe una correspondencia entre estas dos funciones, su correlación tendrá un valor máximo en el punto en que h corresponda con f. Por lo general, se elige un umbral de correlación (Th) para determinar si una función corresponde con otra, éste puede encontrarse en un rango entre $(0,1]$ (Rosenkrantz, 2008; Gonzalez y Woods, 2002).

2.1.4.1. Teorema de correlación

Teorema 2.2 *Sean* $\mathrm{F}(u,v)$ *y* $\mathrm{H}(u,v)$ *las transformadas de Fourier de* $f(x,y)$ *y* $h(x,y)$ *respectivamente. El teorema establece la relación entre la correlación espacial* $f(x,y) \circ h(x,y)$*, y el producto en el dominio de la frecuencia* $\mathrm{F}^*(u,v)\mathrm{H}(u,v)$

$$f(x,y) \circ h(x,y) \Leftrightarrow \mathrm{F}^*(u,v)\mathrm{H}(u,v)\,, \tag{28}$$

formalmente, la correlación en el dominio espacial puede ser obtenido mediante la transformada inversa de Fourier del producto $\mathrm{F}^*(u,v)\mathrm{H}(u,v)$*. De la misma manera, la correlación en el dominio de la frecuencia puede obtenerse mediante una multiplicación en el dominio espacial (Gonzalez y Woods, 2002),*

$$f^*(x,y)h(x,y) \Leftrightarrow \mathrm{F}(u,v) \circ \mathrm{H}(u,v)\,. \tag{29}$$

2.1.4.2. Teorema de autocorrelación

Teorema 2.3 *Se define como autocorrelación cuando ambas funciones son idénticas, luego el teorema de autocorrelación establece*

$$f(x,y) \circ f(x,y) \Leftrightarrow |\mathrm{F}(u,v)|^2 , \tag{30}$$

y

$$|f(x,y)|^2 \Leftrightarrow \mathrm{F}(u,v) \circ \mathrm{F}(u,v) , \tag{31}$$

luego, obtenemos la energía total de la función f. Al realizar esta operación en el espacio de frecuencia obtenemos lo que se llama espectro de varianza que al volver al espacio de tiempo mediante la transformada inversa de Fourier obtenemos la secuencia de autocorrelación de la señal o el mapa de autocorrelación de la función (Gonzalez y Woods, 2002).

2.1.5. Histograma

Un histograma es una representación gráfica de la distribución de datos de una variable aleatoria. El rango de valores se divide en una serie de intervalos discretos, donde la altura de cada intervalo es determinada por la frecuencia de los datos que caen en dicho intervalo. Dependiendo de la aplicación o las características del histograma, los intervalos pueden ser equidistantes o tener longitudes variables (Pearson, 1895). Un ejemplo sencillo de un histograma puede verse en la Figura 4.

El histograma puede ser normalizado, representando la distribución de probabilidad de la variable aleatoria, mostrando frecuencias relativas donde la suma total de frecuencias es igual a 1. La normalización ayuda a realizar comparaciones entre histogramas de una manera más sencilla, pues aunque los datos contengan constantes aditivas o multiplicativas, el valor normalizado es el mismo (Freedman *et al.*, 1998).

El histograma H_I más sencillo de una imagen digital I grafica la frecuencia de los niveles de gris en la imagen. Luego, H_I es una función discreta con un rango de valores $[0, L-1]$,

$$H_I(l) = J , \tag{32}$$

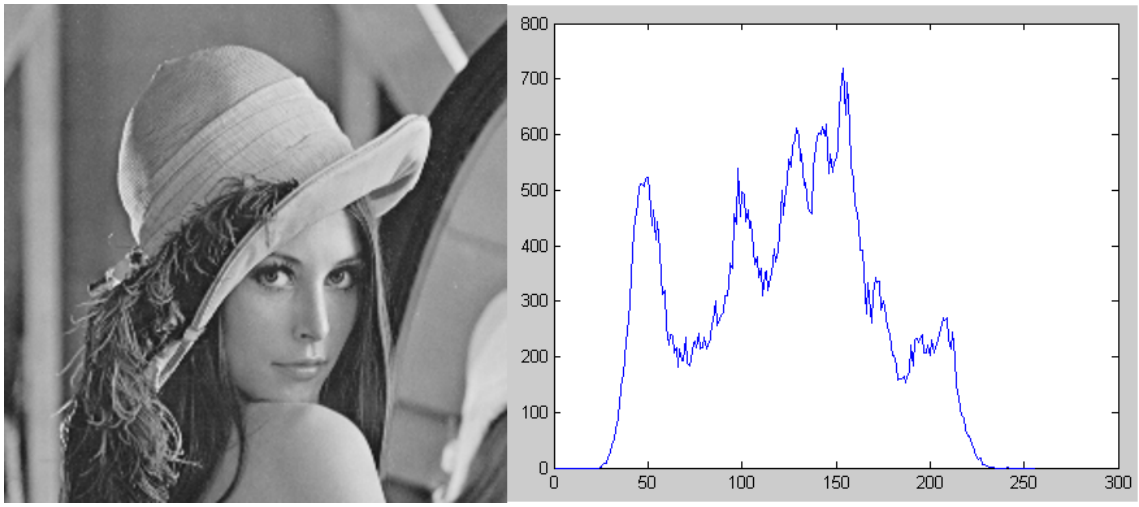

Figura 4: Histograma de la distribución de intensidades de la imagen de Lena en escala de grises.

donde L es el grado de cuantización de la imagen, J es la cantidad de ocurrencias del nivel de gris l, para cada $l = 0, ..., L-1$. Entonces, el algoritmo para calcular el histograma involucra solo el conteo de niveles de gris.

En procesamiento de imágenes, el histograma reduce la dimensionalidad de la imagen original I, por lo que existe pérdida de información; luego, la imagen I no puede ser reconstruida a partir del histograma H_I (Bovik, 2000). Otros tipos de histogramas y su construcción serán descritos en los algoritmos del Estado-Del-Arte y el algoritmo propuesto en los capítulos posteriores.

2.1.6. Gradiente

El gradiente $\vec{\nabla} f$ de un campo escalar f es un campo vectorial que indica en cada punto del campo escalar la dirección de máximo incremento del mismo. El gradiente se define como el campo vectorial cuyas funciones coordenadas son las derivadas parciales del campo escalar, esto es

$$\vec{\nabla} f(r) = \left(\frac{\partial f(r)}{\partial x_1}, ..., \frac{\partial f(r)}{\partial x_n} \right) . \tag{33}$$

Geométricamente el gradiente es un vector que se encuentra normal a una superficie o curva en el espacio en el cual se le está estudiando en un punto cualquiera, permitiendo de esta manera calcular fácilmente las derivadas direccionales.

En procesamiento de imágenes, el gradiente se usa para la detección de bordes. Los detectores de bordes son métodos muy importantes, pues son utilizados para localizar los cambios en las funciones de intensidad, donde los bordes son aquellos pixeles donde la función de intensidad cambia abruptamente. Los bordes son, hasta cierto grado, invariantes a iluminación y a la perspectiva, lo que resalta su importancia en la percepción de la imagen.

Para comprender la forma de una imagen es suficiente con considerar aquellos elementos cuyos bordes tengan las magnitudes más prominentes; lo cual nos permite describir la imagen con un conjunto reducido de datos sin afectar en gran medida el contenido de ésta. La detección de bordes proporciona una adecuada generalización de los datos de la imagen.

Los bordes son propiedades inherentes a cada pixel y son calculadas mediante el comportamiento de la función de la imagen en el vecindario de dicho pixel. Las derivadas de primer orden de una imagen digital se basan en varias aproximaciones del gradiente de dos dimensiones. El gradiente de una imagen (x, y) en la posición del plano (x, y) se define como el vector

$$\overrightarrow{\nabla} I(x,y) = G = [g_x,\ g_y] = \left[\frac{\partial I(x,y)}{\partial x},\ \frac{\partial I(x,y)}{\partial y}\right] . \tag{34}$$

Los bordes pueden ser representados mediante un vector variable de dos dimensiones, magnitud y orientación. La magnitud del borde es la misma magnitud del gradiente, mientras que la orientación del borde está rotada -90° respecto a la dirección de gradiente. La dirección del gradiente muestra la dirección de máximo cambio de la función, como se ilustra en la Figura 5.

La magnitud del gradiente $|\overrightarrow{\nabla} f(x,y)|$ y su orientación θ son funciones continuas de la imagen, y están dadas por

$$|\overrightarrow{\nabla} f(x,y)| = \sqrt{\left(\frac{\partial G}{\partial x}\right)^2 + \left(\frac{\partial G}{\partial y}\right)^2}, \tag{35}$$

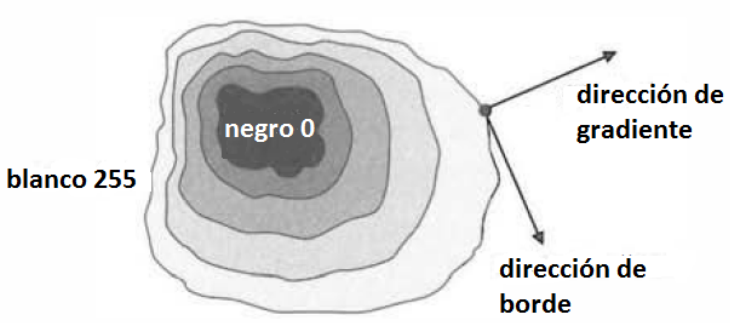

Figura 5: Representación del cambio de intensidades en una imagen, con negro en el centro $f(x,y) = 0$ hasta blanco en la sección exterior $f(x,y) = 255$ y las direcciones del borde y el gradiente ortogonales entre sí. Imagen recuperada de (Sonka *et al.*, 2008), pág. 133.

y

$$\theta = \arctan\left(\frac{\frac{\partial G}{\partial x}}{\frac{\partial G}{\partial y}}\right) . \tag{36}$$

Si una imagen digital es discreta, entonces, las Ecuaciones (35) y (36) deben aproximarse mediante diferencia de pixeles; siendo estas diferencias en las direcciones horizontal y vertical. Generalmente se toman en cuenta los pixeles en el vecindario cercano del pixel al cual se le calculará el gradiente, generalmente con distancia $n = 1$, de la siguiente manera

$$\begin{aligned} g_x &= f(x+n, y) - f(x-n, y)\,, \\ g_y &= f(x, y+n) - f(x, y-n)\,, \end{aligned} \tag{37}$$

donde g_x y g_y son los valores de gradiente discreto en las direcciones horizontal y vertical respectivamente.

A lo largo de los años, diversos operadores de gradiente han sido desarrollados. Estos operadores examinan un vecindario local del pixel, siendo luego, operadores de convolución, y pueden ser expresados mediante máscaras de convolución. Estos operadores son capaces de detectar la dirección de los bordes y son representados por una dupla de máscaras, cada una para una dirección particular. A continuación se presentan los operadores de gradiente más importantes (Sonka *et al.*, 2008; Solomon y Breckon, 2011; Gonzalez y Woods, 2002).

Operador de Roberts El operador de Roberts fue una de las primeras máscaras de convolución para calcular el gradiente. Utiliza un par de máscaras de 2×2 alrededor del

pixel definidas por

$$h_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad h_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{38}$$

Su principal desventaja es su alta sensibilidad al ruido, pues cuenta con pocos pixeles para aproximar el gradiente.

Operador de Laplace El operador de Laplace ∇^2 aproxima la segunda derivada del gradiente y proporciona solamente magnitud, pues es invariante a rotación. Utiliza una máscara de 3×3, las máscaras para un vecindario-4 y vecindario-8 están dadas por

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{39}$$

Operador de Prewitt Este operador calcula la primera derivada con máscaras de 3×3, aunque pueden utilizarse máscaras más grandes. Existen 8 máscaras para calcular el gradiente con variadas direcciones, aunque usualmente se utilizan solo dos, vertical y horizontal, obteniendo buenos resultados. Las dos principales máscaras son

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, \quad h_1 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}. \tag{40}$$

Operador de Sobel El operador de Sobel es de los más utilizados en el procesamiento de imágenes, pues los elementos 2 y -2 actúan como elementos de suavizado, obteniendo así cierto nivel de robustez al ruido. Los operadores vertical y horizontal están dados por

$$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, \quad h_1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}. \tag{41}$$

Operador de Robinson Este operador, a diferencia de los demás operadores que calculan la primera derivada, utiliza información del pixel central; este operador también

tiene un efecto de suavizado.

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}, \quad h_1 = \begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}. \tag{42}$$

Operador de Kirsch Este operador utiliza valores mucho mayores que los demás operadores, balanceando la ecuación para dar un mayor valor a una dirección del operador y balanceando con los otros 5 vecinos alrededor del pixel central.

$$h_1 = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix}, \quad h_1 = \begin{bmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{bmatrix}. \tag{43}$$

2.1.7. Transformaciones geométricas

Las transformaciones geométricas sobre las imágenes modifican la posición y relación espacial de los pixeles; pero no modifican los valores de intensidad de los pixeles, ya sea nivel de gris o color.

Al estudiar las transformaciones geométricas, se debe tener presente el concepto de *forma*. Una forma implica la existencia de bordes, una forma es toda información geométrica que se mantiene después de que los efectos de localización, escalamiento y rotación han sido filtrados del objeto; de modo que, la descripción básica de una forma es un conjunto de pares ordenados que sean suficientes para describir al objeto. Matemáticamente, se describe una forma al ubicar un número finito de N puntos en el borde del objeto y concatenarlos en un *vector de forma*, que, en el caso de un objeto en dos dimensiones se denota como

$$\mathbf{x} = [(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)], \tag{44}$$

o en forma matricial como

$$\mathbf{S} = \begin{bmatrix} x_1 & x_2 & x_3 & ... & x_N \\ y_1 & y_2 & y_3 & ... & y_N \end{bmatrix}, \tag{45}$$

Luego, cuando un objeto es capturado, ya sea en imágenes estáticas o en video, en realidad se capturan instancias del objeto, las cuales nos muestran una forma limitada del objeto real en dos dimensiones al momento de la captura. Al capturar diversas instancias, se puede apreciar al objeto con diferentes características: ya sea un aumento o disminución en su forma, escala; distinta orientación del objeto, rotación; distinta posición respecto a otra instancia, traslación; todas estas transformaciones solo modifican la forma del objeto, no su apariencia, como se muestra en la Figura 6. Sin embargo, cuando un objeto modifica su forma, se considera que ha cambiado su perspectiva, o que ha rotado en un espacio tridimensional.

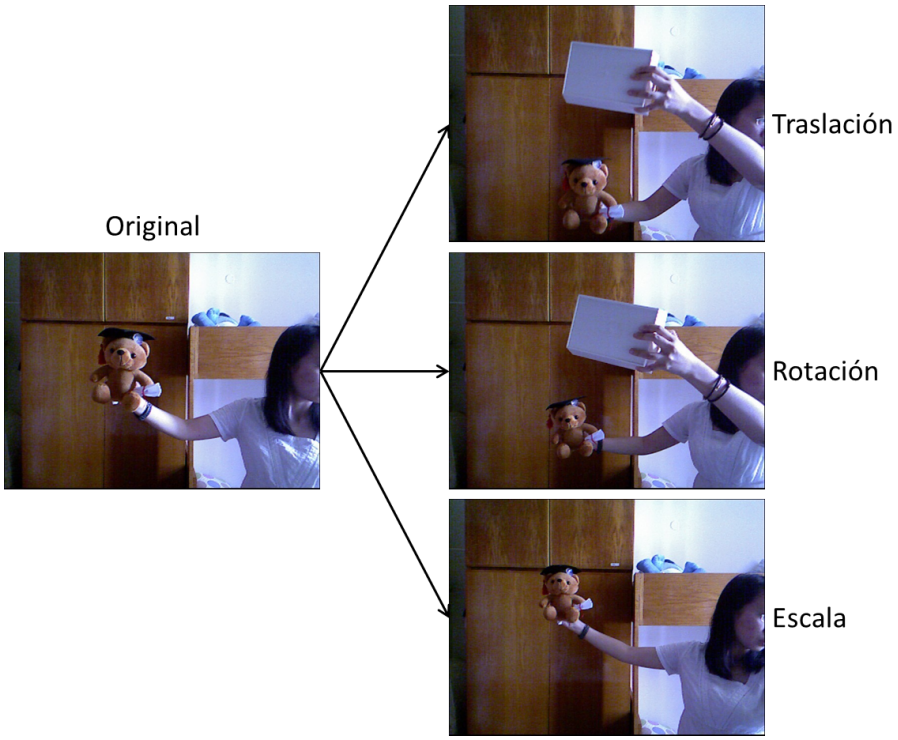

Figura 6: Deformaciones geométricas que no cambian la apariencia del objeto: traslación, rotación y escala.

Las transformaciones geométricas se llevan a cabo mediante matrices de transformación **T**, generalmente mediante matrices de 2×2 que producen un nuevo conjunto de coordenadas **S'** dada por

$$\mathbf{S'} = \mathbf{T}\,\mathbf{S}\,, \tag{46}$$

con cada par ordenado como

$$\mathbf{x'} = \mathbf{T}\mathbf{x} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{47}$$

La transformación más básica es la traslación de una imagen, sean (α_1, α_2) valores escalares constantes; luego, la operación de traslación se lleva a cabo mediante el desplazamiento de las coordenadas de cada par ordenado por estas constantes, de la siguiente manera

$$f'(x, y) = f(x - \alpha_1, y - \alpha_2)\,. \tag{48}$$

Esta operación es usada en sistemas de despliegue de imágenes cuando se requiere mover la imagen en sentido vertical, horizontal o ambos.

La rotación de una imagen f por un ángulo θ relativo a la horizontal se realiza mediante la siguiente matriz de transformación

$$\mathbf{T}_\theta = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}\,. \tag{49}$$

Como el punto de rotación no se encuentra definido en el centro de la imagen sino en cada pixel, el resultado puede caer fuera del dominio de la imagen. De la misma manera, el resultado de la rotación difícilmente va a coincidir con la posición exacta del pixel, a excepción de rotaciones cada $90°$, es necesario realizar una interpolación para establecer el valor de los pixeles de acuerdo a su vecindario.

La transformación de escalamiento puede reducir o aumentar el tamaño del objeto en cuestión, la matriz de escalamiento está dada por

$$\mathbf{T}_S = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \tag{50}$$

donde $s \in \mathbb{R}$ .En caso de reducir el tamaño del objeto, varios pixeles caerán en la misma posición reultante, por lo cual se aplica una operación estadística para definir el valor del pixel. Por el contrario, si se aumenta el tamaño de la imagen, el valor del pixel se expandirá

en su vecindario para rellenar espacios vacíos (Solomon y Breckon, 2011; Bovik, 2009).

2.1.8. Ruido en imágenes

Es común que las imágenes sean degradadas por errores aleatorios, estas degradaciones son conocidas como *ruido*. El ruido ocurre bajo diversas situaciones como: captura de la imagen, transmisión, procesamiento; y, puede ser dependiente o independiente del contenido de la imagen.

El ruido se describe mediante sus características de probabilidad. El *ruido blanco*, también conocido como *ruido ideal* tiene un espectro de potencia constante; luego, todas las frecuencias del ruido se encuentran presentes y tienen la misma magnitud. El ruido blanco es comunmente usado para modelar las peores aproximaciones de degradación; sin embargo, el ruido blanco simplifica los cálculos.

Un caso especial de ruido blanco es el *ruido Gaussiano*, el cual se define mediante una variable aleatoria que recibe su densidad de probabilidad (ndf) mediante una distribución Gaussiana. Su función de densidad unidimensional está dada por

$$ndf(x) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{\frac{-(x-\mu)^2}{2\sigma^2}} , \tag{51}$$

donde μ es el valor esperado y σ la desviación estándar de la variable aleatoria. El ruido Gaussiano es una buena aproximación al ruido que ocurre en muchos casos, como en la naturaleza.

Los tipos de ruido más comunes que existen son el *ruido aditivo* y el *ruido multiplicativo*. Sea f' una imagen observada, ésta puede descomponerse en dos componentes, una imagen ideal f y una componente de ruido n. Luego, la imagen degradada por ruido aditivo puede modelarse como (ver Figura 7)

$$f'(x, y) = f(x, y) + n(x, y) , \tag{52}$$

el ruido Gaussiano es comunmente considerado como ruido aditivo.

Figura 7: Modelo de ruido aditivo. La imagen observada f' resulta de la adición de ruido n a la imagen ideal f.

El segundo modelo de ruido más común es el *ruido multiplicativo*, la imagen degradada por ruido multiplicativo puede modelarse como (ver Figura 8)

$$f'(x, y) = f(x, y) \cdot n(x, y)\,, \tag{53}$$

un ejemplo de ruido multiplicativo es el ruido de salpicadura.

Figura 8: Modelo de ruido multiplicativo. La imagen observada f' resulta de la multiplicación de ruido n con la imagen ideal f.

El modelo aditivo describe apropiadamente al sistema cuando el ruido en el modelo es independiente de la imagen; mientras que el modelo multiplicativo se describe mejor cuando el ruido es dependiente de los valores de la imagen.

Existen otros tipos o modelos de ruido: como el ruido de cuantización que ocurre cuando no se utilizan suficientes niveles de gris para desplegar la imagen y aparecen contornos falsos, el ruido impulsivo que ocurre con pixeles corruptos cuyos valores difieren significativamente con su vecindario, y el ruido sal y pimienta que se presenta cuando una imagen contiene una saturación masiva de ruido impulsivo (Sonka *et al.*, 2008; Bovik, 2009).

2.1.9. Imagen integral

La imagen integral es una representación intermedia de la imagen con la cual es posible procesar de forma rápida características rectangulares. La imagen integral en la posición (x, y) contiene la suma de los pixeles a la izquierda y sobre el pixel (x, y) incluyéndolo, como se muestra a continuación

$$I_{\sum}(x, y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j)\,. \tag{54}$$

Usando la imagen integral, cualquier suma rectangular puede procesarse mediante la suma de cuatro puntos en la imagen integral como se muestra en la Figura 9. De tal forma que es una manera más efectiva de extraer sumas en áreas específicas de la imagen de una manera eficiente (Viola y Jones, 2001).

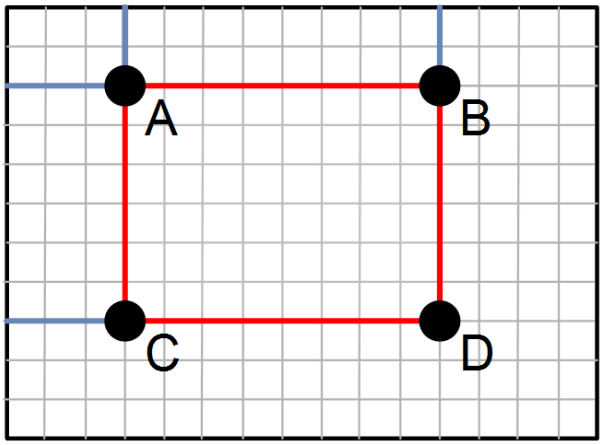

Figura 9: La suma de pixeles dentro del rectángulo marcado puede procesarse mediante la suma de cuatro puntos en la imagen integral $A + D - (B + C)$.

2.2. Fundamentos de algoritmos

Un algoritmo es un procedimiento computacional bien definido, el cual toma un valor o un conjunto de valores (*entrada*) y produce un valor o un conjunto de valores (*salida*). Un algoritmo es una secuencia de pasos computacionales que transforman una entrada en una salida. Un algoritmo es correcto si para cada instancia del problema la salida es la correcta; entonces, el algoritmo resuelve el problema computacional. Un algoritmo debe proveer una descripción precisa del procedimiento computacional a seguir. En esta

sección se discuten algunos conceptos del análisis de algoritmos tanto en ejecución serial como en ejecución paralela.

2.2.1. Análisis de algoritmos

Analizar un algoritmo significa predecir los recursos que el algoritmo utilizará; estos recursos pueden ser memoria, ancho de banda, hardware requerido; sin embargo, el principal recurso a medir es el tiempo de procesamiento. Aunque existen diversos modelos para el análisis de algoritmos, lo más sencillo es utilizar el modelo de una *máquina de acceso aleatoria (RAM, por sus siglas en inglés)*, que utiliza un solo procesador, pues es el que refleja de manera más cercana al modelo computacional genérico de hoy en día. El modelo RAM es un modelo *serial*, es decir, las instrucciones se ejecutan una detrás de la otra y no de forma concurrente (Cormen *et al.*, 2001).

El modelo RAM puede ejecutar algunas operaciones en tiempo constante, como: suma, resta, multiplicación, división, residuo, piso, y techo, entre otras operaciones básicas. Sin embargo, no se debe abusar del modelo y asumir que operaciones de ordenamiento o búsqueda también se ejecutan en tiempo constante. Los tipos de datos se consideran enteros o de tiempo flotante, tomando siempre el mismo tiempo para ejecutar las operaciones sobre ellos.

El tiempo de procesamiento de un algoritmo es comunmente descrito en función del tamaño de su entrada; por lo tanto, primero debemos definir los términos *tamaño de entrada* n y *tiempo de ejecución* $\mathbb{T}^*[\cdot]$. El *tamaño de entrada* es por lo general el número de elementos que contiene la entrada, por ejemplo, la cantidad de números a ordenar en un vector; en otros casos puede ser la cantidad de bits que se utilizarán, entre otros. El *tiempo de ejecución* de un algoritmo para una entrada particular es el número de operaciones primitivas o pasos ejecutados; se considera entonces que cada línea del programa se ejecuta en tiempo constante, y cada vez que se ejecute diche línea le tomará el mismo tiempo. Luego, el tiempo de ejecución del algoritmo $\mathbb{T}^*(\mathrm{n})$ es la suma del tiempo de ejecución de cada línea del algoritmo.

A continuación se presentan algunos conceptos importantes en el análisis de algoritmos.

El *orden de crecimiento* de un algoritmo es la cantidad de operaciones que le toma dependiendo de su tamaño de entrada, y solamente tomando el valor principal de la fórmula (n); luego, este crecimiento ignora valores constantes, pues en la mayoría de los casos es insignificante comparado con el crecimiento del elemento principal, ignorando incluso cualquier constante de éste término. Entonces, el orden de crecimiento presenta una sencilla caracterización de la eficiencia de un algoritmo y permite realizar comparaciones con versiones alternas del algoritmo.

La notación utilizada para describir el tiempo de procesamiento de los algoritmos se define como *notación asintótica*; ésta es un lenguaje que nos permite analizar el tiempo de procesamiento de los algoritmos al identificar su comportamiento mientras el tamaño de entrada del algoritmo se incrementa. A continuación se presentan, de forma general y sin extender mucho en detalles, las notaciones asintóticas más utilizadas en computación:

Notación-Θ Sea una función $g(\mathrm{n})$, se denota por $\Theta(g(\mathrm{n}))$ el conjunto de funciones

$$\begin{aligned}\Theta(g(\mathrm{n})) \;=\; \{f(\mathrm{n}) \,:\, &existen\, constantes\, positivas\, c_1, c_2\, y\, \mathrm{n}_0 \\ &tales\, que\, 0 \leq c_1 g(\mathrm{n}) \leq f(\mathrm{n}) \leq c_2 g(\mathrm{n})\, \forall\, \mathrm{n} \geq \mathrm{n}_0\}\end{aligned} \quad . \tag{55}$$

Una función $f(\mathrm{n})$ pertenece al conjunto $\Theta(g(\mathrm{n}))$ si existen constantes positivas c_1 y c_2 tales que quede entre $c_1 g(\mathrm{n})$ y $c_2 g(\mathrm{n})$, para una n suficientemente grande.

Esta definición requiere que cada miembro de $f(\mathrm{n}) \in \Theta(g(\mathrm{n}))$ sea asintóticamente no negativa, es decir, $f(\mathrm{n})$ es no negativa siempre que n sea lo suficientemente grande; luego, $g(\mathrm{n})$ debe ser asintóticamente no negativa o el conjunto $\Theta(g(\mathrm{n}))$ será un conjunto vacío.

Notación-O La notación-Θ es una función con límites asintóticos superior e inferior, cuando solo consideramos el límite asintótico superior usamos la notación-O. Esta notación es la mayormente usada en la literatura y representa el peor caso de ejecución; es la notación que se utilizará para analizar el algoritmo propuesto. Sea una función $g(\mathrm{n})$, se denota por $\mathrm{O}(g(\mathrm{n}))$ el conjunto de funciones

$$\begin{aligned}\mathrm{O}(g(\mathrm{n})) \;=\; \{f(\mathrm{n}) \,:\, &existen\, constantes\, positivas\, c\, y\, \mathrm{n}_0 \\ &tales\, que\, 0 \leq f(\mathrm{n}) \leq c g(\mathrm{n})\, \forall\, \mathrm{n} \geq \mathrm{n}_0\}\end{aligned} \quad . \tag{56}$$

La notación-O es usada frecuentemente para describir el tiempo de ejecución de un algoritmo al inspeccionar la estructura general de éste.

Notación-Ω Así como la notación-O provee el límite asintótico superior de una función, la notación-Ω provee el límite asintótico inferior. Sea una función $g(\mathrm{n})$, se denota por $\Omega(g(\mathrm{n}))$ el conjunto de funciones

$$\begin{aligned}\Omega(g(\mathrm{n})) \quad = \quad & \{f(\mathrm{n}) \ : \ existen\, constantes\, positivas\, c\, y\, \mathrm{n}_0 \\ & tales\, que\, 0 \leq cg(\mathrm{n}) \leq f(\mathrm{n})\, \forall \mathrm{n} \geq \mathrm{n}_0\}\end{aligned} \quad . \tag{57}$$

Mediante las tres definiciones anteriores, puede probarse el siguiente teorema

Teorema 2.4 *Para cada dos funciones $f(\mathrm{n})$ y $g(\mathrm{n})$, se tiene que $f(\mathrm{n}) = \Theta(g(\mathrm{n}))$ sí y solo sí $f(\mathrm{n}) = \mathrm{O}(g(\mathrm{n}))$ y $f(\mathrm{n}) = \Omega(g(\mathrm{n}))$.*

La Figura 10 muestra gráficamente los conceptos de las notaciones-$\{\Theta, \mathrm{O}, \Omega\}$.

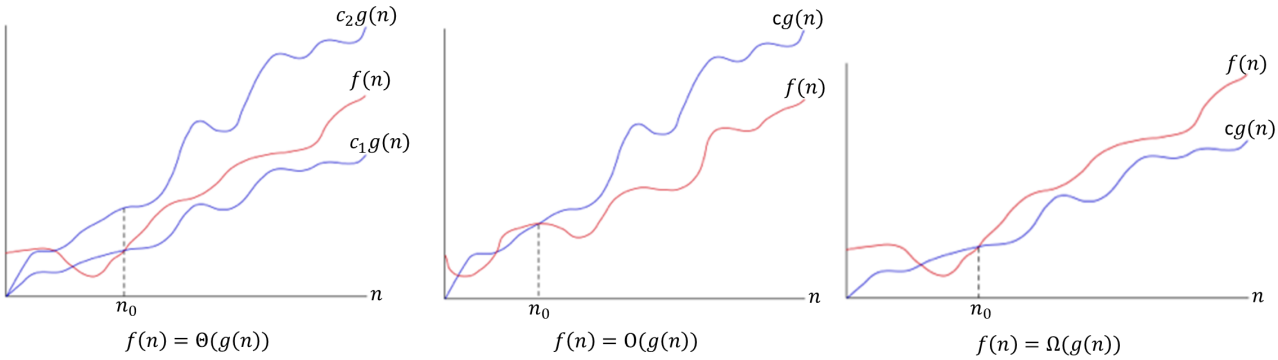

Figura 10: Ejemplos gráficos de las notaciones asintóticas. La notación-Θ impone límites superior e inferior a una función a partir de un punto n_0. La notación-O impone un límite superior (peor caso) a una función a partir de un punto n_0. La notación-Ω impone un límite inferior (mejor caso) a una función a partir de un punto n_0. Imagen recuperada de (Cormen *et al.*, 2001), pág. 41.

2.2.2. Análisis de algoritmos paralelos

El propósito del procesamiento paralelo es realizar el procesamiento más rápido que si se hiciera con un solo procesador al utilizar múltiples procesadores. Tres factores han influenciado el uso del cómputo paralelo en años recientes: el costo del hardware se ha abaratado, la tecnología para construir circuitos integrados a gran escala (VLSI, por sus siglas en inglés) produce circuitos cada vez con un mayor número de procesadores en

un circuito más pequeño, y el mejor tiempo de ciclo de una computadora con procesador tipo von-Neumann está llegando a sus límites. Por lo tanto es necesario el uso de computadoras con múltiples procesadores (JáJá, 1992).

Una *computadora paralela* es una colección de procesadores, normalmente del mismo tipo, interconectados de cierta forma que permitan la coordinación de sus actividades para el intercambio de información.

Al igual que para los algoritmos seriales, existen una variedad de modelos de cómputo; sin embargo, este trabajo se enfoca en el modelo más cercano a los procesadores de múltiples núcleos de hoy en día, el modelo de máquina de acceso aleatorio paralelo (PRAM, por sus siglas en inglés). En este modelo todos los procesadores operan sincronizadamente bajo un mismo reloj bajo instrucciones SIMD, una instrucción—múltiples datos, por sus siglas en inglés; y el modelo de memoria se define como MIMD, múltiples instrucciones—múltiples datos, por sus siglas en inglés. Del mismo modo, al analizar algoritmos paralelos contamos con un número de variables extra que cuando lo hacemos en algoritmos seriales, como el número de procesadores, capacidades de la memoria local, sistema de comunicación y protocolos de sincronización.

Sea P un problema computacional dado y n el tamaño de su entrada; luego, un algoritmo resuelve el problema P secuencialmente en $\mathbb{T}^*(\mathrm{n})$, no habiendo otro algoritmo que lo resuelva más rápido. Sea entonces A un algoritmo paralelo que resuelva P en tiempo $\mathbb{T}_p(\mathrm{n})$ en una computadora paralela con p procesadores. Luego, la *aceleración* en resolver A está definida por

$$S_p(\mathrm{n}) = \frac{\mathbb{T}^*(\mathrm{n})}{\mathbb{T}_p(\mathrm{n})}\,. \tag{58}$$

Idealmente, se esperaría que cuando $p = 1$ el algoritmo se ejecutara en $\mathbb{T}^*(\mathrm{n})$; sin embargo, esto no siempre es posible por los varios factores que se describieron anteriormente. Otra medida de rendimiento del algoritmo paralelo A es la *eficiencia*, definida como

$$E_p(\mathrm{n}) = \frac{\mathbb{T}_1(\mathrm{n})}{p\mathbb{T}_p(\mathrm{n})}\,. \tag{59}$$

Esta medida provee un indicador de cómo se utilizan los p procesadores en el algoritmo. Un valor para $E_p(\mathrm{n})$ cercano a 1 para p procesadores, indica que el algoritmo A

se ejecuta aproximadamente p veces más rápido usando p procesadores que usando un procesador. Esto significa que cada procesador está trabajando en cada paso del algoritmo.

En el modelo PRAM todos los procesadores comparten un espacio de memoria llamada *memoria global*, y tienen una memoria reducida para almacenar los datos con los que se encuentran trabajando llamada *memoria local*. Está de más decir que solo el procesador p_i puede accesar su memoria local; pero todos los procesadores pueden accesar la memoria global, en la cual puede haber conflictos de lectura/escritura. Debido a ello existen algunas variaciones del modelo PRAM. Las principales variaciones son:

1. **EREW PRAM.** Lectura exclusiva y escritura exclusiva, solo permite el acceso a un solo procesador a una localidad de memoria en un tiempo dado.
2. **CREW PRAM** Lectura concurrente y escritura exclusiva. Solo permite el acceso a lecturas concurrentes.
3. **CRCW PRAM** Lectura concurrente y escritura concurrente. Existen tres variaciones de este modelo dependiendo cómo se manejan las escrituras concurrentes:
   a) **Común** Solo permite la escritura concurrente si todos los procesadores intentan escribir el mismo dato.
   b) **Arbitrario** Permite escribir a un procesador arbitrario.
   c) **Prioritario** Permite la escritura al procesador con el índice más pequeño.

De la misma manera que medimos el tiempo de ejecución para algoritmos seriales en la RAM, en algoritmos paralelos usando la PRAM utilizamos un paradigma de trabajo-tiempo ($\mathbb{WT}$, por sus siglas en inglés), el cual provee una descripción del algoritmo paralelo de dos niveles: superior e inferior. El nivel superior evita detalles del algoritmo, mientras el nivel inferior sigue el principio de calendarización.

Sea $\mathbb{W}_i(\mathrm{n})$ el número de operaciones ejecutadas en la unidad de tiempo i, donde $1 \leq i \leq \mathbb{T}(\mathrm{n})$, si se simulan cada conjunto de $\mathbb{W}_i(\mathrm{n})$ operaciones en $\lceil \frac{\mathbb{W}_i(\mathrm{n})}{p} \rceil$ pasos paralelos por p procesadores, para cada $1 \leq i \leq \mathbb{T}(\mathrm{n})$. Si la simulación es exitosa, el algoritmo PRAM

respectivo con p-procesadores toma $\sum_i \lceil \frac{\mathbb{W}_i(\mathrm{n})}{p} \rceil \leq \sum_i \left(\lfloor \frac{\mathbb{W}_i(\mathrm{n})}{p} \rfloor + 1 \right) \leq \lfloor \frac{\mathbb{W}(\mathrm{n})}{p} \rfloor + \mathbb{T}(\mathrm{n})$ pasos paralelos.

- **Nivel superior** Describe el algoritmo en términos de una secuencia de unidades de *tiempo* ($\mathbb{T}$), donde cada unidad de tiempo puede incluir una cantidad de operaciones concurrentes. Se define como *trabajo* ($\mathbb{W}$) realizado por un algoritmo paralelo como el número total de operaciones realizadas.
- **Nivel inferior** Sea un algoritmo en el paradigma $\mathbb{WT}$ que corre en $\mathbb{T}(\mathrm{n})$ unidades de tiempo realizando $\mathbb{W}(\mathrm{n})$ operaciones; utilizando el principio de calendarización, casi siempre es posible adaptar el algoritmo para que se ejecute en una PRAM con p-procesadores en $\lfloor \frac{\mathbb{W}(\mathrm{n})}{p} \rfloor + \mathbb{T}(\mathrm{n})$ pasos paralelos.

Los algoritmos paralelos no mencionan la cantidad de procesadores que utilizan, de este modo se calcula un rendimiento donde se supone ya sea una cantidad infinita de procesadores o una cantidad de procesadores p suficiente para que el paso que requiera de la mayor cantidad de procesadores puede ejecutarse en un instante.

2.2.3. Ejemplo de algoritmo

En este apartado se presentan de forma sencilla los conceptos de esta sección mediante la presentación de un algoritmo en su forma serial y paralela. Cabe destacar que existen dos formas de presentar pseudocódigo, la primera es mediante código informal como el presentado más abajo; la segunda es mediante una descripción de los pasos del algoritmo sin presentarlo como pasos en un algoritmo. El algoritmo presentado es la suma S de $n - 2^k$ números guardados en un arreglo A. Para el cual se realiza el cálculo de tiempo de procesamiento y trabajo, este último para el caso paralelo.

Algoritmo 1 Suma serial

Entrada: $n = 2^k$ números guardados en el arreglo A
Salida: La suma $S = \sum_{i=1}^{n} A(i)$

```
1: S = 0
2: para i = 1 a n hacer
3:     S+ = A(i)
4: fin para
```

Como puede observarse, es muy sencillo calcular el tiempo de procesamiento del algoritmo serial, pues se añade cada elemento a S en cada una de las n iteraciones. Luego $\mathbb{T} = \mathrm{O(n)}$.

Algoritmo 2 Suma paralela con $n/2$ procesadores

Entrada: $n = 2^k$ números guardados en el arreglo A
Salida: La suma $S = \sum_{i=1}^{n} A(i)$

```
1: S = 0
2: i = obtenerProcesador()
3: para h = 1 hasta log n hacer
4:    si i ≤ n/2^h entonces
5:       A(i) = A(i) + A(n/2^h + i)
6:    fin si
7: fin para
8: S = A(1)
```

El algoritmo paralelo no menciona la cantidad de procesadores utilizados ni cómo se asignan, se asume que cada operación utiliza un procesador por operación. Solo se calcula en término de unidades de tiempo, cada cual puede incluir cualquier número de operaciones concurrentes. Primeramente tenemos un ciclo que toma $h = \log \mathrm{n}$ unidades de tiempo (en todos los algoritmos se supone que $\log \mathrm{n} = log_2 \mathrm{n}$, a menos que se indique lo contrario), en cada unidad de tiempo se ejecutan $\mathrm{n}/2^h$ operaciones concurrentes, por lo tanto tenemos $\mathbb{W}(\mathrm{n}) = 2 + \sum_{j=1}^{\log \mathrm{n}} (n/2j) = \mathrm{O(n)}$, siendo las dos operaciones la inicialización de S y la asignación del resultado; el tiempo de procesamiento es claramente $\mathbb{T}(\mathrm{n}) = \mathrm{O}(\log \mathrm{n})$. Para el cálculo de la complejidad de algoritmos, generalmente se descartan las constantes y los términos más pequeños, dejando solamente el término mayor indicando la cota superior del algoritmo.

2.3. Fundamentos tecnológicos

2.3.1. GPU

Los microprocesadores con una unidad central de proceso (CPU) incrementaron rápidamente el desempeño de cómputo y redujeron el costo de elaboración de aplicaciones computacionales. Estos microprocesadores permitieron a las computadoras de escritorio alcanzar billones de operaciones de punto flotante (GFLOPS), permitiendo a los programas de computadora incrementar su funcionalidad, mejores interfaces de usuario, y generar mejores y más confiables resultados.

Problemas como el consumo de energía y disipación de calor han limitado el incremento de frecuencia de reloj y la cantidad de trabajo efectivo que puede realizarse por ciclo del reloj en un solo CPU. Por lo tanto, los fabricantes de microprocesadores han cambiado el modelo de fabricación en el cual se tienen múltiples unidades de procesamiento, o núcleos de procesamiento, en un solo chip para incrementar el poder de procesamiento. La mayoría de las aplicaciones de cómputo son escritas como programas secuenciales, como fue descrito por John von Neumann en 1945. Los programas secuenciales solo se ejecutan en uno de los núcleos de procesamiento, por lo tanto no tienen ningún incremento de desempeño; por otro lado, los programas paralelos mejorarán su desempeño con cada generación de microprocesadores, pues varios hilos de ejecución pueden cooperar para terminar el trabajo más rápido.

Desde el 2003, los microprocesadores se han dividido en dos categorías: *multinúcleo* (CPU multinúcleo), los cuales mantienen la velocidad de ejecución de programas secuenciales en múltiples procesadores; *muchos-núcleos* (GPU), se centra en el rendimiento de procesamiento de aplicaciones paralelas, estos microprocesadores contienen una gran cantidad de procesadores pequeños. Existe una brecha importante entre el desempeño de una GPU de muchos-núcleos y un CPU de propósito general multinúcleo; la diferencia máxima de rendimiento entre una GPU y un CPU multinúcleo para el cálculo de operaciones de punto flotante es de aproximadamente 10 a 1. Esto se debe a la diferencia de diseño entre ambos tipos de microprocesadores, como se muestra en la Figura 11.

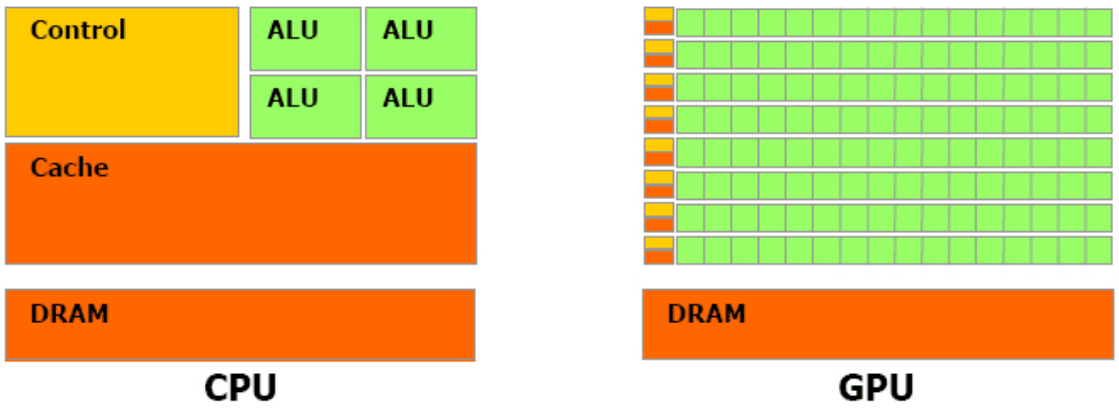

Figura 11: Arquitectura básica de un CPU y una GPU. Imagen recuperada de (Kirk y Hwu, 2010), pág. 4.

El diseño del CPU se encuentra optimizado para la ejecución de código secuencial; utiliza lógica de control que permite que un solo hilo se ejecute en paralelo mientras mantiene la apariencia de ejecución secuencial; para reducir el retraso de acceso a los datos utiliza un repositorio grande de memoria para el almacenamiento local de estructuras de datos y variables. En la actualidad, existen microprocesadores multinúcleo de propósito general de 2 (Intel Core 2 Duo, AMD K8) hasta 8 núcleos (Intel Core i7-5960X), simulando a su vez de 4 a 16 núcleos respectivamente.

El diseño de las GPU se moldeó gracias al rápido crecimiento de la industria de video juegos. Las GPUs tienen la ventaja que un gran número de hilos de ejecución realizan cálculos mientras una pequeña cantidad se ocupa de los lentos accesos a memoria, minimizando de esta forma la lógica de control requerida por cada hilo de ejecución. Las GPUs contienen además una pequeña memoria local que le permite reducir los requerimientos de ancho de banda de las aplicaciones, al reducir la cantidad de hilos que tienen que acceder a la memoria del sistema para la copia de información y dedicar este tiempo para el cálculo de operaciones de punto flotante. Las GPUs fueron diseñadas como máquinas de cómputo numérico, teniendo un bajo rendimiento en aplicaciones secuenciales diseñadas para un CPU. Generalmente se usa un GPU para ejecutar las partes secuenciales de un programa y una GPU para las partes numéricamente intensivas (Kirk y Hwu, 2010).

Arquitectura de GPU

Al momento de programar en una GPU se debe tener en cuenta la arquitectura y estructura de ésta, pues aunque cuenta con un gran número de núcleos o unidades de cómputo, también cuenta con memoria limitada y una serie de otras limitaciones que deben ser consideradas. Una GPU se compone de tres partes básicas, las unidades de cómputo, memoria y el canal de comunicación con el sistema. La Figura 12 muestra la arquitectura básica de una GPU.

Un dispositivo, o GPU, contiene un número establecido de unidades de cómputo, cada una de las cuales contiene un gran número de unidades de proceso que trabajan mediante el paradigma de una instrucción sobre múltiples datos (SIMD); esto quiere decir, que se ejecuta una sola instrucción, por ejemplo multiplicación por un escalar, a una gran

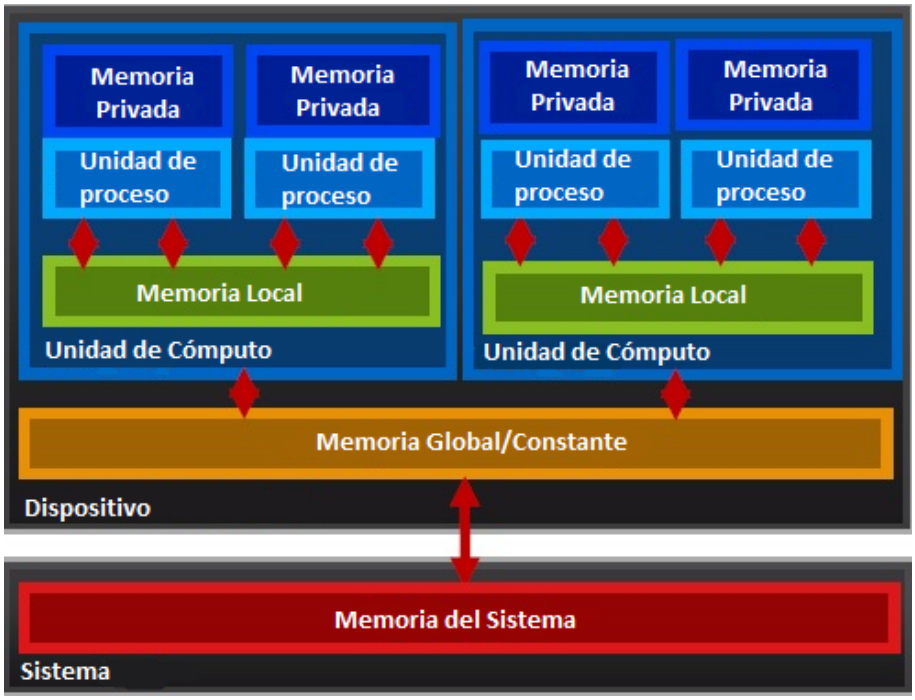

Figura 12: Arquitectura de una GPU. Cada unidad de cómputo contiene un número constante de unidades de proceso, cada uno con su memoria privada y memoria local compartida, así como memoria global compartida entre las unidades de cómputo. Imagen recuperada de `http://sabia.tic.udc.es/gc/trabajos%202011-12/ATIvsCUDA/arquitectura.html`**.**

cantidad de datos en memoria de forma concurrente. Las unidades de cómputo trabajan por separado, generalmente realizando el mismo tipo de operaciones, sin embargo, es posible que una unidad de cómputo termine antes que las demás y continúe la ejecución de instrucciones sobre otro conjunto de datos. Es importante conocer y tener en cuenta la cantidad de unidades de cómputo y unidades de proceso que contiene la GPU sobre la que se trabaja; de esta manera se puede optimizar el trabajo por unidad de tiempo que realiza cada unidad de cómputo.

Una GPU contiene tres tipos de memoria interna. Cada unidad de proceso posee una pequeña memoria local (generalmente 32 KB), la cual es utilizada para almacenar los datos que se procesarán en este núcleo, es la memoria más rápida y solo es visible por su unidad de proceso asociada. La memoria compartida es la segunda memoria más rápida, es visible por todas las unidades de proceso en una unidad de cómputo, se utiliza para almacenar vectores y estructuras de datos que se procesarán en la unidad de cómputo. La memoria global es la más lenta, esta es visible por todas las unidades de cómputo, contiene todas las estructuras de datos que se han copiado desde el sistema para ser ejecutadas en la GPU, en esta memoria se almacenan los resultados finales del trabajo

de cada unidad de cómputo. Las GPUs modernas tienen memorias globales que van de 1 GB a 12 GB.

Es necesario conocer el tamaño de las memorias internas de la GPU, pues como es limitada, hay ocasiones en que se deben copiar datos desde y hacia la memoria del sistema por el canal de comunicación (buffer) con la GPU. La copia de datos entre el sistema y la GPU es muy tardada, lo cual, si se realiza con mucha frecuencia, puede afectar gravemente el desempeño del algoritmo. Lo más recomendable es transferir los datos necesarios a la GPU al inicio del algoritmo y transferir el resultado final al término del algoritmo, utilizando al mínimo el canal de comunicación entre el sistema y la GPU.

Lenguajes de programación paralela

En las últimas décadas se han propuesto varios lenguajes de programación paralela. Entre los lenguajes más utilizados se encuentran la Interfaz de Transferencia de Mensajes (MPI, por sus siglas en inglés) utilizado en grupos escalables de computadoras, y OpenMP para sistemas multinúcleo de memoria compartida. MPI no tiene un modelo de memoria compartida, todos los datos se comparten mediante intercambio explícito de mensajes, lo cual provoca que traducir una aplicación a MPI requiera una gran cantidad de esfuerzo. OpenMP soporta memoria compartida, sin embargo, este lenguaje fue diseñado para trabajar con procesadores multinúcleo; por lo tanto, no se encuentra preparado para procesamientos que requieren una gran cantidad de hilos de ejecución.

Cuando las GPUs empezaron a tener un gran auge en la industria computacional, los programadores tenían que pasar a través de los APIs (Interface de Programación de Aplicaciones) de procesamiento gráfico para poder acceder a los núcleos de una GPU y utilizarlos para aplicaciones en general. Recientemente NVIDIA y Khronos desarrollaron dos lenguajes de programación paralela que se han vuelto el estandar de la industria: CUDA y OpenCL respectivamente. Estos lenguajes permiten el acceso tanto a los núcleos, registros y todos los niveles de memoria de una GPU y con el paso de los años se han ido refinando para aprovechar al máximo los recursos de las nuevas tarjetas gráficas que se han ido desarrollando.

CUDA es un lenguaje propietario de NVIDIA, y aunque solo se puede ejecutar en esta arquitectura es altamente eficiente. Por otro lado, OpenCL permite realizar programación

paralela heterogénea; esto quiere decir que OpenCL puede ejecutarse en una gran variedad de arquitecturas paralelas con ligeras modificaciones o ninguna en absoluto; entre las arquitecturas que soporta se encuentran GPUs, CPUs, FPGAs, Cross Bars, por mencionar algunas (Kirk y Hwu, 2010).

En este trabajo utilizamos como lenguaje de programación paralela OpenCL debido a que la tarjeta gráfica utilizada es una ATI RADEON HD 6450, la cual no es soportada por CUDA. El ciclo de programación paralela puede ser muy abrumador en un inicio, pues es un nuevo paradigma de programación y deben tomarse en consideración una serie de requisitos e *ideologías* que no existen en la programación serial. Primeramente debe inicializarse el ambiente de programación paralela para que el sistema sepa que se va a utilizar el GPU, este es un proceso algo tedioso y difícil de entender al inicio; sin embargo, una vez programado una vez, es posible reutilizarlo con ligeras modificaciones dependiendo de la aplicación.

Los algoritmos de OpenCL pueden encontrarse dentro de una variable de texto o en un archivo por separado, el cual es compilado por el GPU antes de la ejecución del algoritmo, definiendo de esta forma el *programa*, este programa puede contener una o más funciones paralelas. Para la ejecución de un algoritmo paralelo primero debemos copiar los datos de la memoria del sistema a la memoria global de la GPU y/o reservar espacio en la memoria global para depositar los resultados; es importante señalar que debemos conocer el espacio que ocuparán nuestros cálculos, pues la memoria no puede ser reservada dinámicamente. Una vez copiados los datos, le indicamos a la GPU qué algoritmo queremos que ejecute, con qué datos, y cuántos hilos de ejecución queremos ejecutar; la GPU asigna los hilos de ejecución a los núcleos de tal manera que se aproveche al máximo la capacidad del GPU, también es posible especificar cómo queremos que se ejecuten los hilos en las diferentes unidades de cómputo para tener más control del proceso.

El CPU funciona como la base del proceso, pues es quien *lanza* los trabajos a la GPU de manera secuencial. Toda tarea enviada a la GPU entra en una pila de ejecución tipo FIFO (primero que entra primero que sale); luego, cada tarea almacenada espera en la fila a que se liberen los recursos necesarios en la GPU para ser procesadas. Esto puede resultar confuso para quien inicia en la programación paralela, pues parece un paradigma secuencial; lo que se debe entender es que son los algoritmos que se ejecutan en la

GPU los que tienen un caracter paralelo, no la manera en que se envían estas tareas al GPU; por ejemplo, sea X una secuencia desordenada de números aleatorios y queremos sumar los n números más grandes, la primera tarea sería ejecutar un algoritmo paralelo de ordenamiento, y después ejecutar un algoritmo paralelo de suma sobre los n números más grandes; aunque el ordenamiento y la suma son algoritmos paralelos, su ejecución fue secuencial.

2.3.2. Cámaras de profundidad: Microsoft Kinect

Las cámaras de profundidad o de rango son dispositivos que en lugar de desplegar una imagen en niveles de intensidad o a color, como normalmente se hace; nos muestra una imagen en escala de grises, en la cual el nivel de gris nos indica la distancia entre los objetos en la escena y la cámara. Las cámaras de profundidad constaban de tecnología especializada, lo cual las convertía en dispositivos económicamente no viables; sin embargo, en 2010, Microsoft lanzó al mercado su cámara Kinect a un precio accesible tanto para grupos de investigación como para el público en general. La cámara de Microsoft propició a otras compañias a desarrollar sus propias cámaras de profundidad. La Figura 13 muestra las partes principales del sensor Kinect, las cámaras de la competencia presentan una estructura similar.

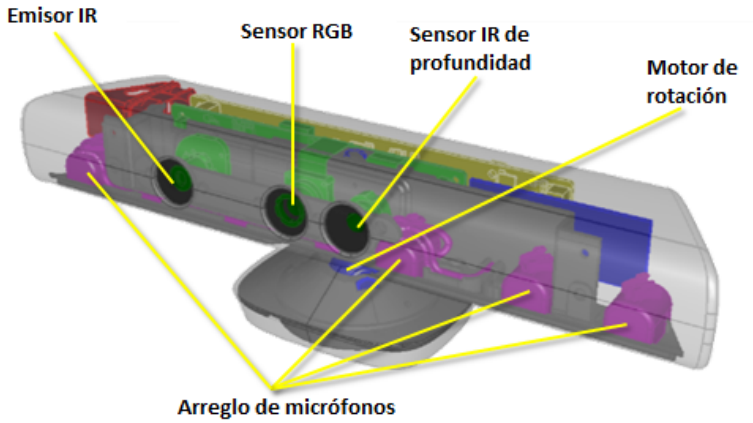

Figura 13: Componentes del sensor Microsoft Kinect. Imagen recuperada de `https://msdn.microsoft.com/en-us/library/jj131033.aspx`**.**

Existen dos esquemas para medir la profundidad mediante sensores infrarrojos: luz estructurada o escaner de laser, y Tiempo-de-Vuelo (ToF, por sus siglas en inglés). Actualmente el Kinect cuenta con dos versiones, el Kinect v1.0 que funciona mediante luz

estructurada, y el modelo más reciente, el Kinect v2.0 que funciona mediante ToF. Este trabajo utiliza videos obtenidos con el Kinect v1.0, el cual de aquí en adelante será referido solamente como Kinect. Los sistemas de luz estructurada, como el Kinect, usan un patrón irregular fijo que consiste en un gran número de puntos provenientes de un láser infrarrojo. El campo infrarrojo del Kinect es una matriz de puntos aleatorios de 3×3, en la cual cada zona tiene diferente intensidad y un punto de calibración central, como se muestra en la Figura 14. El patrón irregular permite una mejor discriminación que un patrón regular donde las intensidades son uniformes en toda la matriz. El sensor infrarrojo determina la diferencia entre el rayo de luz emitido y la posición del punto de luz; debido a que debe haber espacio entre dos puntos adyacentes que forman parte de la luz estructurada y el espacio debe ser lo suficientemente ancho para que el sensor de luz pueda distinguirlos, aproximadamente 1 de cada 20 pixeles en la imagen tiene una medida real de profundidad, los pixeles adyacentes deben ser interpolados (Langmann *et al.*, 2012; Jiang, 2015).

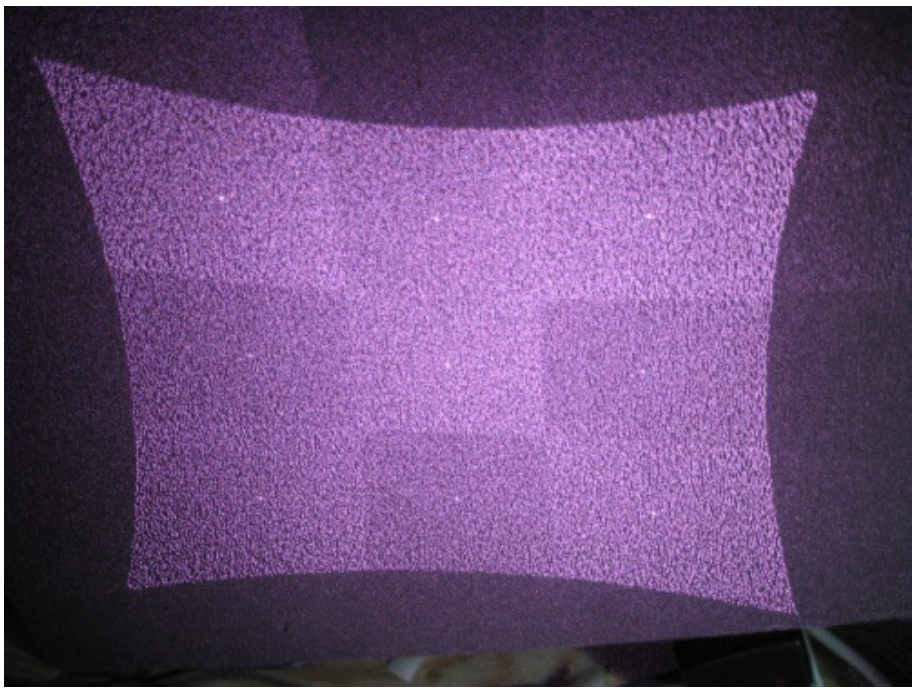

Figura 14: Patrón de luz estructurada emitida por el Kinect de Microsoft. Puede apreciarse la matriz de 3×3 zonas con diferente intensidad y el punto central de calibración. Imagen recuperada de (Jiang, 2015) pág. 68.

El sistema de Tiempo-de-Vuelo tiene dos principios de operación: pulso de luz, el cual realiza mediciones en pequeños intervalos de tiempo para obtener una resolución que corresponda a unos cuantos centímetros de profundidad; y amplitud modulada de

onda continua, la cual realiza mediciones mediante el cambio de fase entre la emisión y recepción de luz modulada, la cual corresponde al Tiempo-de-Vuelo y a su vez a la profundidad. La Figura 15 muestra cómo funciona el sistema de Tiempo-de-Vuelo.

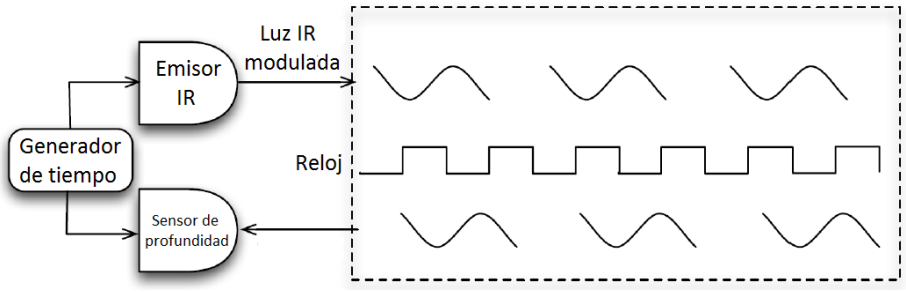

Figura 15: Esquema del modelo de Tiempo-de-Vuelo. Imagen recuperada de (Lun y Zhao, 2015) pág. 6.

Sea A la respuesta del sistema cuando el emisor está encendido y B cuando está apagado. A contiene tanto la luz ambiental como del emisor IR, B solo contiene la luz ambiental. Luego, la diferencia $A-B$ da como salida la luz modulada reflejada por el emisor infrarrojo, el cual puede ser usado para calcular de manera precisa la profundidad, y su magnitud $\| A-B \|$ proporciona una imagen infrarroja de alta calidad sin luz ambiental, lo cual la vuelve invariante a iluminación. La Figura 16 muestra los mapas de profundidad producidas por el Kinect v1.0 y Kinect v2.0 (Langmann *et al.*, 2012; Lun y Zhao, 2015).

La calidad de los mapas de profundidad del Kinect v2.0 son superiores a los de la versión 1.0. Al momento de escribir este trabajo el Kinect v2.0 es exclusiva de la consola Xbox One de Microsoft, con planes de lanzar la versión para Windows en un futuro próximo, lo cual dará lugar a mejoras sustanciales en los algoritmos que usen mapas de profundidad. Además, el *punto de referencia* utilizado en este trabajo utiliza el Kinect v1.0; motivando de esta manera a realizar nuevos experimentos en el futuro con la nueva versión del Kinect. Por lo tanto, solo se discutirán algunas características del Kinect v1.0.

El Kinect usa un elemento óptico difractivo y un láser infrarrojo para generar el patrón irregular de luz estructurada. Incorpora un módulo RGB y un módulo de niveles de gris con un filtro infrarrojo, el cual determina la disparidad entre los puntos de luz emitidos y su posición observada. La triangulación de la profundidad de un objeto en una escena se realiza al identificar los puntos de luz sobre el objeto, para lo cual es mejor utilizar un

Figura 16: Diferencias entre las imágenes IR y mapas de profundidad de las dos versiones del Kinect. Como puede apreciarse, el Kinect v2.0 ofrece imágenes de una mejor calidad, aunque de menor resolución espacial. Imagen recuperada de (Lun y Zhao, 2015) pág. 7.

patrón irregular a uno regular. El cálculo de la profundidad en metros se calcula como

$$\iota(d) = 0.181 \tan(0.161d + 4.15)\,, \tag{60}$$

donde d es un valor entero de profundidad en metros (Langmann *et al.*, 2012).

Aplicaciones

El análisis de mapas de profundidad tiene algunas aplicaciones interesantes, entre las que se encuentran (Jiang, 2015):

- Suavizado.
- Cómputo de características locales (geometría diferencial).
- Detección de bordes.
- Detección de puntos de interés.
- Segmentación de imagen.

- Registro de mapas de profundidad (fusión).
- Flujo de profundidad.
- Estimación de poses.

Capítulo 3. Algoritmos de correspondencia

En este capítulo se presenta el Estado-Del-Arte de los algoritmos de correspondencia. Aunque SIFT se desarrolló hace ya más de una década, sigue siendo considerado como el algoritmo más importante en lo que respecta a correspondencia. En la década pasada se realizaron innumerables variaciones de SIFT con el objetivo de mejorar el rendimiento, principalmente en tiempo de ejecución, pues SIFT genera vectores multidimensionales como descriptor que toma mucho tiempo en procesarse; tomando algunas ideas de SIFT e innovando en algunas otras áreas surge SURF, un algoritmo de características que mejora el tiempo de procesamiento al utilizar imágenes integrales y formar su espacio de escala mediante una pirámide de filtros; sin embargo, esta mejora impactó en la precisión del algoritmo. Finalmente, en 2010 ORB aparece como una alternativa a SIFT y SURF, utilizando un descriptor binario y características tipo FAST. Por el momento, estos tres algoritmos son la base de la investigación en correspondencia en los cuales los demás algoritmos se basan parcialmente o simplemente se les añaden características para hacerlos más robustos en sus puntos débiles.

3.1. Transformada de características invariante a escala (SIFT)

SIFT es un algoritmo que extrae características invariantes distintivas de imágenes que pueden ser usadas para realizar correspondencia confiable entre objetos o escenas en imágenes. Las características extraídas de las imágenes son invariantes a escala y rotación; además, proveen correspondencia robusta aún en condiciones de ruido y ligeros cambios de iluminación (Lowe, 2004).

A continuación se describen la serie de pasos que sigue SIFT para extraer los descriptores de características que pueden ser usados para realizar la búsqueda de correspondencia entre dos o más imágenes.

Descriptor

Esta primera etapa del algoritmo trata de detectar localizaciones en la imagen que sean repetibles e invariantes a cambios de escala y bajo diferentes vistas del objeto; para esto se hace la búsqueda a través de un espacio de escala utilizando un kernel Gaussiano. El espacio de escala de la imagen se define como una función $L(x, y, \xi)$ que

se produce por la convolución de un kernel Gaussiano de escala variable $G(x, y, \xi)$ y la imagen de entrada $I(x, y)$

$$L(x, y, \xi) = G(x, y, \xi) \otimes I(x, y)\,, \tag{61}$$

donde $\otimes$ es la operación de convolución en x y y, y el kernel Gaussiano está definido por

$$G(x, y, \xi) = \frac{1}{\sqrt{2\pi}\xi} \mathrm{e}^{-(x^2+y^2)/2\xi^2}\,. \tag{62}$$

El espacio de escala en SIFT difiere del concepto en una forma particular, ya que cuenta con un conjunto de escalas y un conjunto de octavas por escala. En este algoritmo la escala es la convolución de la imagen con el kernel Gaussiano como se mostró anteriormente, y una octava puede definirse como un muestreo de la imagen a la mitad del tamaño de la octava anterior, este concepto puede apreciarse claramente en la Figura 17.

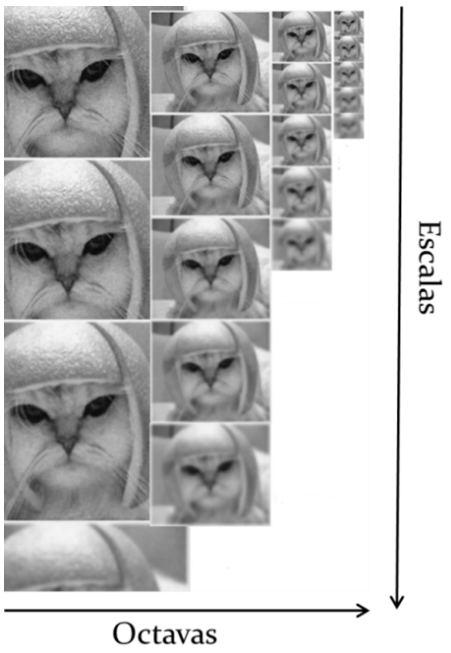

Figura 17: Espacio de escalas. Imagen recuperada de `http://www.aishack.in/2010/05/sift-scale-invariant-feature-transform/`**.**

Para la localización de características estables en el espacio de escala se realiza una Diferencia de Gaussianas (DoG) entre las escalas de la imagen, esta operación es una aproximación del Laplaciano de Gaussianas (LoG) como se muestra en la Figura 18;

para después extraer los pixeles extremos, máximos y mínimos, de entre las imágenes resultantes de la DoG, esta operación se realiza al comparar cada pixel con sus 26 vecinos como se muestra en la Figura 19, cabe destacar que esta operación no se realiza en las imágenes en los extremos de la DoG.

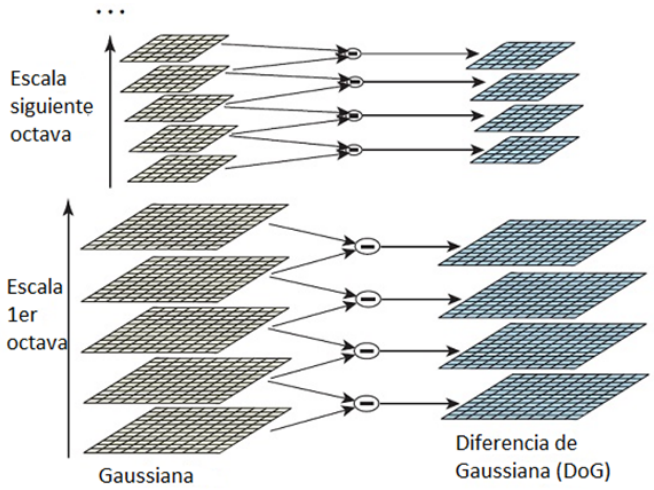

Figura 18: Diferencia de Gaussianas (DoG) sobre las escalas en cada octava de la imagen. Imagen recuperada de (Lowe, 2004), pág. 96.

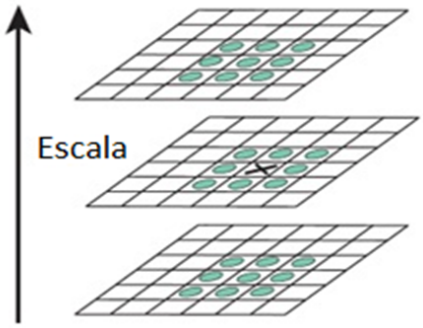

Figura 19: Localización de máximos y mínimos sobre el resultado de la DoG. Comparación del pixel X con sus 26 vecinos. Imagen recuperada de (Lowe, 2004), pág. 97.

Por cada máxima/mínima extraída, a las cuales de aquí en adelante se les llamarán características, se extraen las magnitudes y orientaciones de todos los pixeles a su alrededor con un radio de 1.5σ, dependiendo de su escala sobre las imágenes generadas por la DoG.

Con estos datos se forma un histograma de gradiente orientado (HoG) de 36 compartimientos ($10°$ por compartimiento), donde cada instancia que entra en un compartimiento del histograma es pesado por su magnitud correspondiente. Del histograma resultante la

característica toma la orientación del compartimiento que resulte ser el máximo del histograma y se crean características nuevas con la orientación de los compartimientos que cuenten con al menos el $80\,\%$ de la altura del compartimiento máximo como se muestra en la Figura 20, con lo cual se tienen características iguales con orientaciones diferentes.

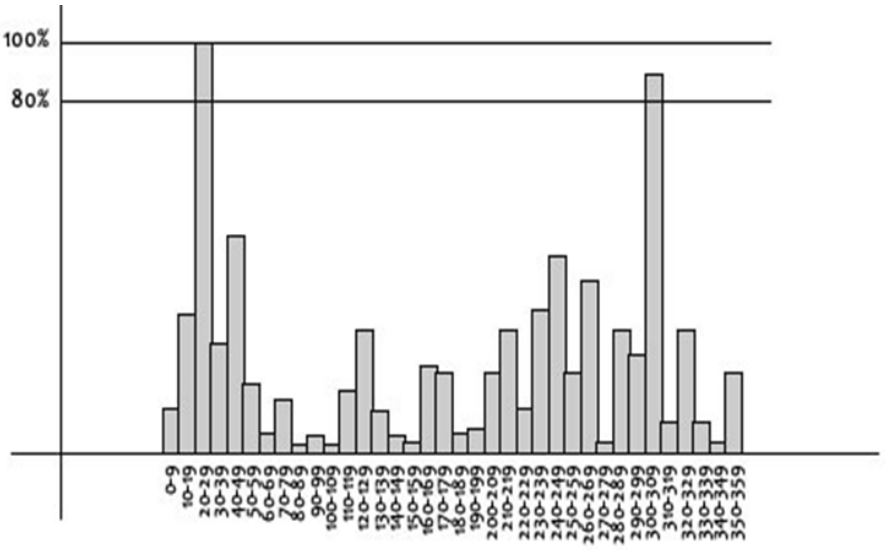

Figura 20: HoG en el cual se asigna la orientación de $20° - 29°$ y se crea una nueva característica con orientación $300° - 309°$. Imagen recuperada de `http://www.aishack.in/2010/05/sift-scale-invariant-feature-transform/`**.**

Para formar los descriptores de características se sigue un procedimiento similar a la asignación de orientaciones. Por cada característica se obtienen las magnitudes y orientaciones en un vecindario de 16×16 alrededor de la característica, luego se formarán 16 HoGs de 8 compartimientos de la misma forma que se realizó anteriormente formando un vector de 128 dimensiones como se muestra en la Figura 21, a este vector se le llama descriptor.

Correspondencia

Una imagen de 500×500 pixeles puede generar 2000 descriptores aproximadamente, estos descriptores se comparan uno por uno contra los descriptores de otras imágenes donde se busca algún objeto. Esta búsqueda de correspondencia entre vectores se realiza mediante el algoritmo de vecinos cercanos. Sin embargo, los resultados pueden dar a lugar a una gran cantidad de características correspondientes, para filtrar estas correspondencias se evalúa la cercanía entre los vectores de características, sobreviviendo solo

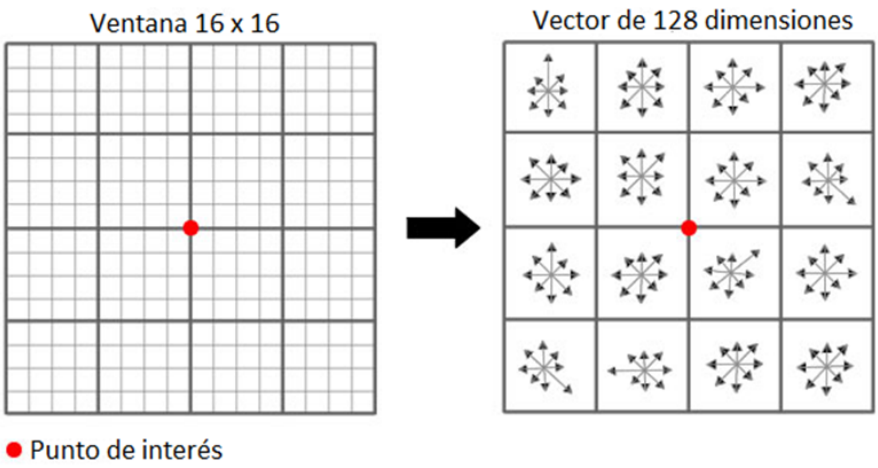

Figura 21: Descriptor de características SIFT. Imagen recuperada de `http://www.aishack.in/2010/05/sift-scale-invariant-feature-transform/`**.**

aquellas correspondencias en las que sus dos vecinos más cercanos no sobrepasen de un umbral establecido; Lowe establece este umbral en 0.6.

3.2. Características robustas aceleradas (SURF)

SURF (Bay *et al.*, 2008) es un algoritmo invariante a escala y rotación que se basa en el uso de imágenes integrales para las operaciones de convolución. Este algoritmo tiene sus bases en SIFT por lo que muchas de las operaciones que realiza son, si no idénticas, muy parecidas al algoritmo de Lowe.

Sin embargo, sacrificando en parte la precisión de SIFT, SURF opera mucho más rápido gracias al uso de imágenes integrales, a su innovativa forma de crear el espacio de escala y a su descriptor reducido de 64 dimensiones. A continuación se detalla el procedimiento que sigue SURF.

Descriptor

Para crear el espacio de escala, en lugar de filtrar la imagen una y otra vez con kernels Gaussianos y muestreando la imagen a la mitad de su tamaño original para cada octava, SURF opta por otra medida que es computacionalmente más eficiente. SURF crea una pirámide de kernels que aproximan al Laplaciano de Gaussianas incrementando cada vez el tamaño del kernel, de esta forma la convolución se lleva a cabo entre la imagen y la pirámide de kernels dando como salida el espacio de escalas. La Figura 22 muestra este concepto y los kernels resultantes de esta operación. La escala ξ se calcula mediante la

siguiente fórmula

$$\xi_{aprox} = Tamaño\ del\ kernel \frac{\xi\ base}{Tamaño\ del\ kernel\ base}\,. \tag{63}$$

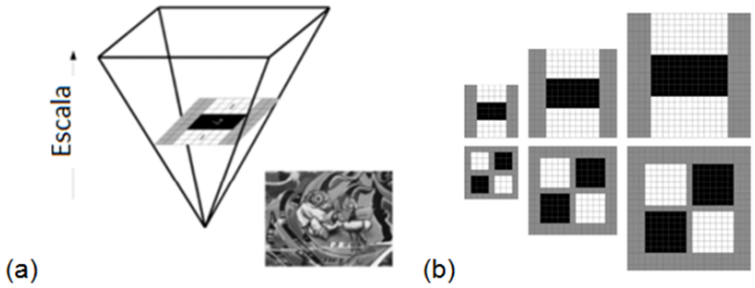

Figura 22: Espacio de escala de SURF. (a) Generación de la pirámide de kernels Gaussianos. (b) Conjunto de kernels Gaussianos que se convolucionarán con la imagen. Imagen recuperada de (Bay *et al.*, 2008), pág. 349.

Una vez generado el espacio de escala cada imagen resultante se evalúa por medio de un umbral y todos aquellos puntos que no superen este umbral son eliminados dejando solo las características más fuertes; al incrementar disminuyen el número de características y viceversa. Después de este paso se realiza, al igual que en SIFT, la supresión de características no máximas comparando las características de las escalas internas con sus 26 vecinos, como muestra en la Figura 19, esto provee un mapa de características reducido.

Teniendo un grupo selecto de características, el siguiente paso es asignarle una orientación que pueda ser repetible bajo distintas circunstancias, para ello se utilizan los filtros tipo—Haar (Figura 23) de tamaño 4 en un radio de tamaño 6 alrededor de la característica en la imagen integral, donde es la escala en la cual la característica fue detectada. Los filtros tipo—Haar proveen los gradientes de la imagen con los cuales se extraen las respuestas al filtro en las direcciones x y y de cada pixel en la zona seleccionada. Para seleccionar la orientación dominante se rota una ventana de $\frac{\pi}{3}$ sobre el origen de la característica sumando las respuestas al filtro y extrayendo su magnitud y orientación (Figura 24). La orientación de la característica será aquella del vector que tenga la mayor magnitud.

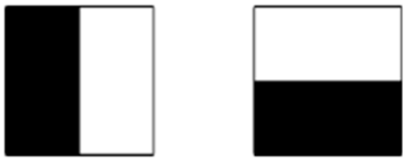

Figura 23: Filtros tipo—Haar. Izquierda, gradiente en x, derecha gradiente en y. La zona negra $= 1$, zona blanca $= -1$. Los filtros tipo—Haar en imágenes integrales requieren solo 6 operaciones. Imagen recuperada de (Bay *et al.*, 2008), pág. 351.

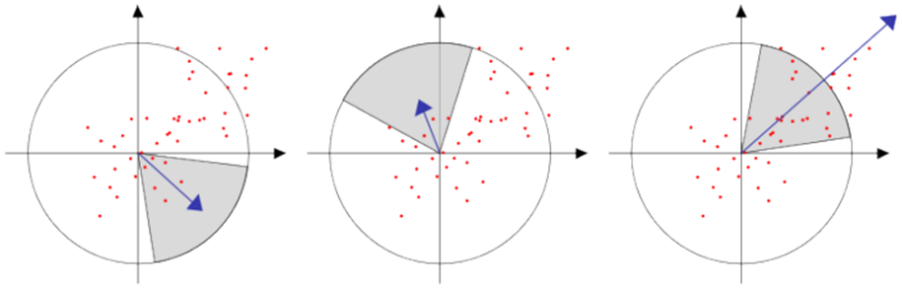

Figura 24: Asignación de orientación. La ventana rota alrededor de la característica sumando las respuestas de los filtros Haar y obteniendo un vector por zona, aquel con la mayor magnitud proporciona su orientación a la característica. Imagen recuperada de (Bay *et al.*, 2008), pág. 351.

Para extraer el descriptor SURF se construye una ventana cuadrada de tamaño 20ξ (donde ξ es la escala en la cual se localizó la característica) alrededor de la característica y orientada en la dirección calculada anteriormente. Esta ventana se divide en 16 zonas cuadradas regulares las cuales se filtran con filtros Haar de tamaño 2 obteniendo de esta forma los siguientes componentes por cada zona como lo definen (Bay *et al.*, 2008).

$$zona = \left\{ \sum dx, \sum dy, \sum |dx|, \sum |dy| \right\} . \tag{64}$$

De tal manera que cada zona contribuye 4 componentes al descriptor, por lo tanto el descriptor SURF se convierte en un vector de $16 \times 4 = 64$ dimensiones (Figura 25).

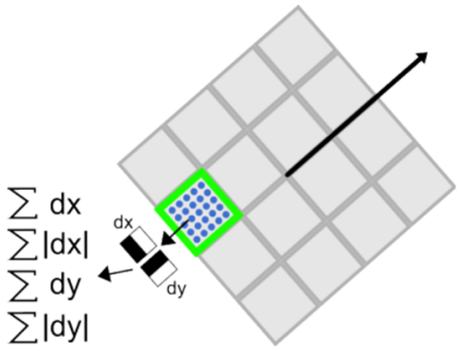

Figura 25: Componentes del descriptor. Por cada zona se calculan las respuestas a los filtros Haar obteniendo 4 componentes por zona. Imagen recuperada de (Bay *et al.*, 2008), pág. 352.

Correspondencia

Este paso se lleva a cabo por medio del algoritmo de vecinos cercanos tomando en cuenta el signo de la respuesta con los kernels de Gaussianas, de tal forma que solo se comparan aquellos que tengan el mismo signo pues las características siempre se ubican en zonas claras u oscuras que darán lugar a respuestas con signos diferentes, de tal forma que al comparar de esta manera el proceso se acelera.

3.3. FAST orientado y BRIEF rotado (ORB)

Orb es presentado como una alternativa a los algoritmos del Estado-Del-Arte en correspondencia. Rublee *et al.* (2011) presentan su algoritmo bajo la premisa de ser tan eficiente o más eficiente que SIFT y tan rápido como SURF. ORB se basa en dos algoritmos bien conocidos, el detector de características FAST (Rosten y Drummond, 2006) y el descriptor BRIEF (Calonder *et al.*, 2010); estas técnicas presentan buen desempeño y un bajo costo. El algoritmo utiliza variaciones de estas técnicas, añadiendo orientación a las características de FAST y rotación al descriptor BRIEF. La descripción de los elementos del algoritmo se presentan a continuación.

FAST orientado

Normalmente las características FAST no tienen orientación. Estas características toman un parámetro, el umbral de intensidad entre el pixel central y los pixeles alrededor de éste dentro de un anillo, ORB utiliza un anillo de radio 9. Para calcular N puntos de interés, primero se establece el umbral lo suficientemente pequeño para obtener más de N puntos, luego se ordenan por medio de la medida de Harris y se seleccionan los mejores N puntos. Se genera una pirámide de escala, en cada escala se producen puntos de interés para dotar al algoritmo de invarianza a escala.

El algoritmo utiliza el *centroide de intensidad* para calcular la orientación de bordes. El centroide de intensidad asume que la intensidad de un borde se encuentra desplazado del centro del fragmento produciendo un vector del centro al centroide, este vector se usa para calcular la orientación de la siguiente manera: primero se calculan los momentos del fragmento

$$m_{pq} = \sum_{x,y} x^p y^q (x, y)\,, \tag{65}$$

luego, con estos momentos se encuentra el centroide

$$\mathrm{C} = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) , \tag{66}$$

siendo la orientación del fragmento

$$\theta = \arctan \left(\frac{m_{01}}{m_{10}} \right) . \tag{67}$$

Para mejorar la invarianza a rotación, se mantienen las coordenadas x y y dentro de la región circular de radio r al calcular los momentos.

BRIEF rotado

El descriptor BRIEF es una cadena de bits de un fragmento de la imagen extraida de un conjunto de pruebas de intensidad binarias. Sea un fragmento suavizado fp; una prueba binaria ϖ se define como

$$\varpi(fp; x, y) := \begin{cases} 1 & fp(x) < fp(y) \\ 0 & fp(x) \geq fp(y) \end{cases} , \tag{68}$$

donde $fp(x)$ es la intensidad de fp en el punto x. La característica se define como un vector de n pruebas binarias

$$L_n(fp) := \sum_{1 \leq i \leq n} 2^{i-1} \varpi(fp; x_i, y_i) , \tag{69}$$

se selecciona un vector de longitud $n = 256$ con una distribución Gaussiana alrededor del centro del fragmento.

Para hacer a BRIEF invariante a rotación en plano, se gira el descriptor de acuerdo a la orientación de los puntos de interés FAST. Para cada conjunto de características de n pruebas binarias con posición (x_i, y_i) se define una matriz de $2 \times n$

$$S = \begin{pmatrix} x_1, ..., x_n \\ y_1, ..., y_n \end{pmatrix} . \tag{70}$$

Usando la orientación θ del fragmento y la matriz de rotación R_θ correspondiente, se obtiene la versión *girada* S_θ de S

$$S_\theta = R_\theta S\,, \tag{71}$$

luego, el operador BRIEF girado se vuelve

$$l_n(fp, \theta) := L_n(fp) \text{ donde } (x_i, y_i) \in S_\theta\,. \tag{72}$$

Se discretiza el incremento del ángulo a 12°. El conjunto de puntos en S_θ se usa para calcular el descriptor.

Capítulo 4. Algoritmos de seguimiento

En este capítulo se presentan los algoritmos de seguimiento del Estado-Del-Arte contra los que comparamos el algoritmo propuesto. Se seleccionaron los algoritmos que al día de hoy tienen más auge en el área de seguimiento de objetos en la literatura bajo dos esquemas diferentes. Primero, se presentan los algoritmos OMIL y TLD, los cuales utilizan múltiples instancias del objeto a seguir y cuentan con módulos de aprendizaje en línea, modificando sus descriptores al tiempo que la secuencia de video transcurre. Después, se presentan los algoritmos OAPF y RGBDOcc+OF, los cuales incluyen el procesamiento de mapas de profundidad para añadirle robustez al proceso de localización y seguimiento del objeto.

4.1. Aprendizaje de múltiples instancias en línea (OMIL)

OMIL es un algoritmo de seguimiento que se basa en la idea de extraer un conjunto de descriptores *correctos*, en lugar de instancias individuales del objeto, en un vecindario cercano a la *zona verdadera* del objeto y etiquetarlos dentro de una *bolsa positiva* de descriptores. De la misma manera, se crean *bolsas negativas* de descriptores apartados de la zona verdadera del objeto, con el objetivo de discriminar al objeto del fondo mediante clasificadores (Babenko *et al.*, 2009).

El concepto de *aprendizaje de múltiples instancias* fue introducido por Viola *et al.*(2007) para la detección de objetos, argumentando que la detección de objetos tiene ambigüedades inherentes que lo hacen más difícil de clasificar usando métodos tradicionales. Babenko extiende estas ideas para adaptarlas al seguimiento de objetos.

Este algoritmo se enfoca más en la descripción de un modelo de apariencia del objeto dinámico; es decir, que cambia a lo largo de la secuencia de video al añadir nuevas bolsas de descriptores correctas con las características del objeto mientras va cambiando conforme pasa el tiempo. El uso de múltiples instancias y bolsas de descriptores conlleva algunos problemas: primeramente, si la localización del objeto no es precisa, la siguiente bolsa obtiene instancias subóptimas, que al pasar el tiempo se degradan gradualmente; segundo, varias bolsas pueden ser etiquetadas como correctas, luego el algoritmo debe decidir cuál es la *más* correcta.

A continuación se describe el algoritmo OMIL.

Descriptor y modelo de movimiento

El descriptor de imagen se basa en un conjunto de vectores de características tipo Haar aleatoriamente calculadas de cada instancia seleccionada para las bolsas positivas y negativas. Cada característica consiste de 2 a 4 rectángulos, donde el valor de la característica es la suma ponderada de los pixeles en todos los rectángulos. Estas características son calculadas de forma eficiente mediante el uso de imágenes integrales (Viola y Jones, 2001).

El modelo de apariencia se compone de un clasificador discriminativo capaz de resolver $\mathrm{p}(y = 1 \mid x)$, donde x es un fragmento de la imagen y y es una variable binaria que indica la presencia del objeto de interés en el fragmento de la imagen. A cada paso t, el seguidor mantiene la localización del objeto $l_i^\star$. Sea $l(x)$ la localización del fragmento x en la imagen; en cada cuadro de video se seleccionan un conjunto de fragmentos $X^s = \{x \mid s > \| \, l(x) - l_{t-1}^\star \, \|\}$ a una distancia s de la localización actual del seguidor y se calcula $\mathrm{p}(y \mid x)$ para cada parche en el conjunto. Luego, se utiliza un algoritmo voráz para actualizar la localización del seguidor.

Un modelo de seguimiento es usado para mantener la localización del seguidor en el tiempo t con igual probabilidad de aparecer dentro de un radio s de la localización del objeto al tiempo $t-1$

$$\mathrm{p}(l_t^\star \mid l_{t-1}^\star) \propto \begin{cases} 1, & si \;\; \| \, l_t^\star - l_{t-1}^\star \, \| < s \\ 0, & \text{de otra forma} \end{cases} . \tag{73}$$

Los datos de entrenamiento que conforman al modelo de apariencia tienen la forma $\{(X_1, y_1), ..., (X_n, y_n)\}$, donde $X_i = \{x_{i1}, ..., x_{im}\}$ es una bolsa, y su etiqueta y_i está dada por

$$y_i = \max_j (y_{ij}) \, . \tag{74}$$

Para la actualización del modelo de apariencia, se seleccionan un conjunto de fragmentos de imagen $X^r = \{x | r > \|l(x) - l_i^\star\|\}$, donde $r < s$ es un radio entero, esta bolsa se etiqueta como positiva; la bolsa negativa se selecciona de un subconjunto de fragmentos dentro

de una región anular que satisfaga $X^{r,\beta} = \{x|\beta > \|l(x) - l_i^\star\| > x\}$, donde β es un escalar. Este proceso puede verse en la Figura 26.

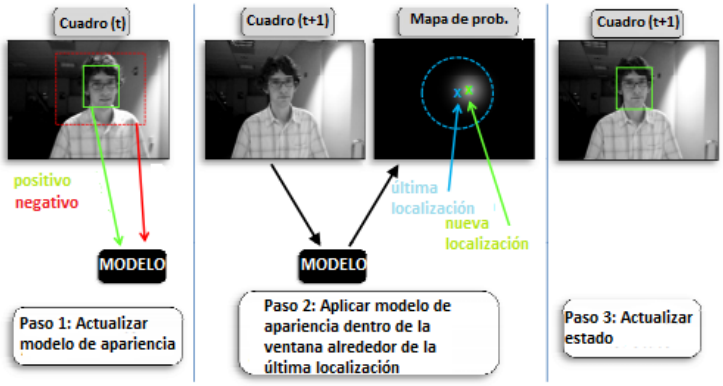

Figura 26: Modelo de seguimiento voráz de OMIL. Imagen recuperada de (Babenko *et al.*, 2009), pág. 2.

Clasificador

Una estimulación se refiere a combinar un conjunto de clasificadores débiles $\mathbf{h}(x)$ en un clasificador aditivo fuerte

$$\mathbf{H} = \sum_{k=1}^{K} \alpha_k \mathbf{h}_k(x)\,, \tag{75}$$

donde α_k son escalares de ponderación. Luego, se analiza estadísticamente el proceso con el objetivo de optimizar una función de pérdida J. Los clasificadores débiles se seleccionan secuencialmente para satisfacer

$$(\mathbf{h}_k, \alpha_k) = \max_{\mathbf{h} \in H, \alpha} J(\mathbf{H}_{k-1} + \alpha \mathbf{h}) \tag{76}$$

donde $\mathbf{H}_{k-1}$ es un clasificador fuerte compuesto de los primeros $k-1$ clasificadores débiles y $\mathbf{H}$ es el conjunto de todos los clasificadores. Los clasificadores son entrenados con instancias positivas y negativas de los cuadros anteriores para determinar la etiqueta que se le asignará a la bolsa.

4.2. Seguimiento-Aprendizaje-Detección (TLD)

TLD es un algoritmo relativamente nuevo que alcanzó una gran popularidad en los últimos 5 años, pues presenta una propuesta en la cual utiliza tres sistemas ejecutándose simultáneamente compartiendo información, lo cual lo hace un sistema robusto, con el

objetivo de corregir errores. El enfoque de Kalal *et al.* (2012) es que un algoritmo de seguimiento o de detección pueden resolver el problema de seguimiento continuo de manera independiente; pero, si trabajan de manera conjunta, se benefician mutuamente.

El algoritmo constra de tres partes. El seguidor (T), estima el movimiento del objeto en cuadros de video consecutivos bajo la premisa que el movimiento entre cuadros es limitado y el objeto se encuentra visible; el seguidor es propenso a fallar y a no recuperarse si el objeto sale del cuadro de video. El detector (D), trata cada cuadro independientemente y escanea la imagen completa para localizar todas las apariencias que han sido observadas y aprendidas en cuadros anteriores; el detector comete dos tipos de errores, falsos positivos y falsos negativos. El aprendizaje (L), observa el desempeño de los otros dos sistemas, estima los errores del detector y genera instancias de entrenamiento para evitar que los errores ocurran en el futuro, supone que los otros dos sistemas pueden fallar. La Figura 27 muestra la relación entre estos 3 elementos.

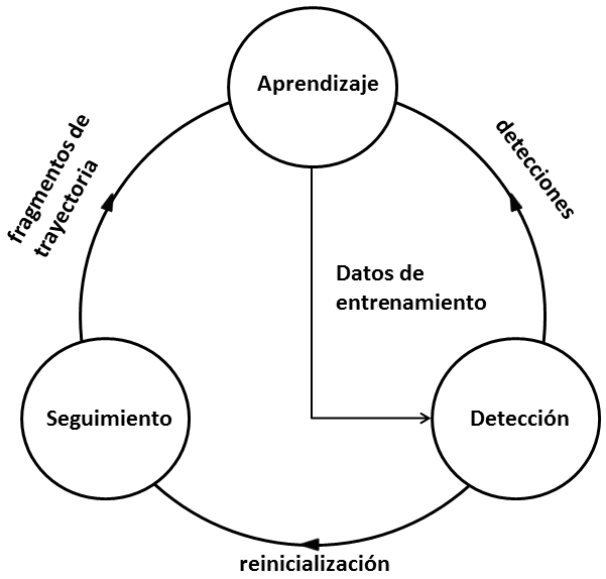

Figura 27: Relación entre el seguidor, detector y módulo de aprendizaje de TLD (Kalal *et al.*, 2012), pág. 3.

En cada instante de tiempo, el objeto es representado por su estado, ya sea un cuadro delimitador o una bandera que indica que el objeto no se encuentra visible. La caja delimitadora mantiene un tamaño fijo. Cada instancia del objeto es representada por un fragmento de imagen p dentro del cuadro delimitador del objeto y muestreado a una re-

solución de 15×15 pixeles. La similitud entre dos fragmentos p_i y p_j está definida por

$$S(\mathrm{p}_i, \mathrm{p}_j) = 0.5(NCC(\mathrm{p}_i, \mathrm{p}_j) + 1)\,, \tag{77}$$

donde NCC es el Coeficiente de Correlación Normalizado, por sus siglas en inglés.

El descriptor del objeto O es una estructura que representa al objeto y sus alrededores observados hasta el momento. Es un conjunto de fragmentos positivos y negativos $M = \mathrm{p}_1^+, ..., \mathrm{p}_m^+, \mathrm{p}_1^-, ..., \mathrm{p}_n^-$ donde p^+ y p^- representan fragmentos del objeto y del fondo respectivamente.

El algoritmo TLD se describe a continuación.

Detector

El detector busca en la imagen o cuadro de video mediante una ventana deslizante, y por cada fragmento en el cual es subdividida la imagen decide si el objeto está presente o no. Cada cuadro de video es de 240×320 pixeles. Debido al gran número de fragmentos de imagen a evaluar, el algoritmo utiliza un descriptor de tres etapas (Figura 28), donde cada una rechaza el fragmento o lo pasa a la siguiente etapa:

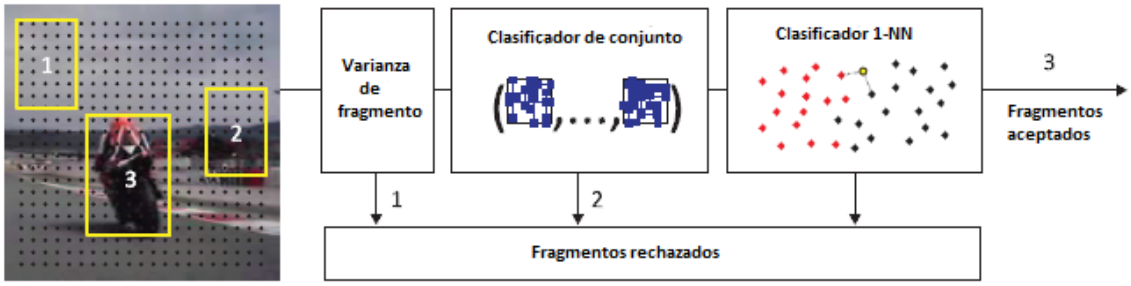

Figura 28: Diagrama de bloques del detector (Kalal *et al.*, 2012), pág. 8.

1. **Varianza de fragmento** En esta etapa se rechazan todos aquellos fragmentos que su varianza de valores de escala de gris de sus pixeles sea menor al $50\,\%$ de la varianza del fragmento que fue seleccionado para seguimiento del cuadro anterior. Dado que $\mathrm{E}(\mathrm{p})$ puede ser calculada fácilmente mediante imágenes integrales; luego, la varianza del fragmento p puede calcularse como $\mathrm{E}(\mathrm{p}^2) - \mathrm{E}^2(\mathrm{p})$.

2. **Clasificador de conjunto** Este clasificador consiste de **h** clasificadores base que

realizan un conjunto de comparaciones de pixel que tiene como salida un código binario x, los cuales corresponden a un arreglo de probabilidades *a posteriori* $P_i(y|x)$, donde i es el clasificador base y $y \in \{0, 1\}$. Cada clasificador se genera fuera de línea como preprocesamiento y quedan estáticos mientras se ejecute el algoritmo. La probabilidad se calcula como $P_i(y|x) = \frac{\#\mathrm{p}}{\#\mathrm{p}+\#\mathbf{h}}$, donde $\#\mathrm{p}$ y $\#\mathbf{h}$ corresponden a la cantidad de fragmentos positivos y negativos respectivamente, que fueron asignados al mismo código binario.

3. **Clasificador de vecinos cercanos** Este clasificador decide las cajas delimitadoras del objeto utilizando el descriptor. Un fragmento es clasificado como objeto si su similitud relativa $S^r(\mathrm{p}, \mathrm{O}) > \theta_{NN}$, donde $\theta_{NN} = 0.6$. Los fragmentos clasificados como positivos representan la respuesta del objeto al detector.

Seguidor

El seguidor de TLD se basa en el seguidor de Flujo de Mediana extendido con detección de fallas (Kalal *et al.*, 2010). El seguidor representa al objeto como una caja delimitadora y estima su movimiento entre cuadros de video consecutivos. Internamente, el seguidor estima el desplazamiento de un conjunto de puntos dentro de la caja delimitadora del objeto, estima su confiabilidad y vota con el $50\,\%$ de los desplazamientos más confiables utilizando la mediana. El seguidor supone la visibilidad del objeto en el cuadro de video, por lo tanto, falla en caso de que el objeto sea bloqueado o salga fuera del cuadro de video.

Aprendizaje

La tarea del componente de aprendizaje es inicializar el detector del objeto en el primer cuadro y actualizarlo en ejecución utilizando dos tipos de algoritmos *expertos*: experto-P y experto-N. En el primer cuadro, el componente de aprendizaje entrena al detector mediante instancias sintetizadas de la primera caja delimitadora del objeto; genera 20 versiones de ésta mediante diversas transformaciones geométricas: desplazamiento, cambios de escala, rotación en plano, ruido Gaussiano aditivo; esto resulta en un conjunto de 200 fragmentos positivos. Los fragmentos negativos se recolectan de los alrededores de la caja delimitadora inicial; a estos fragmentos no se les aplican transformaciones geométri-

cas. Después de la inicialización, el detector está listo para ejecutarse con los módulos de actualización de los expertos P-N.

- **Experto-P** El objetivo de este experto es descubrir la presencia del objeto en el cuadro de video. El desafío de este experto es identificar las partes confiables de la trayectoria del seguidor y utilizarla para la generación de instancias positivas de entrenamiento. La trayectoria se mantiene confiable mientras no sea reinicializada o el seguidor identifique su propia falla. Si la posición actual es confiable, el experto-P genera un conjunto de instancias positivas y actualiza el modelo del objeto y al clasificador de conjunto.
- **Experto-N** Este clasificador genera instancias de entrenamiento negativas. Su objetivo es descubrir el fondo contra el cual el detector debe discriminar. El experto-N supone que el objeto solo puede ocupar un espacio de la imagen; luego, si la localización del objeto es conocida, los alrededores del objeto se etiquetan como negativos.

Los expertos N-P se ejecutan concurrentemente. Los resultados de los expertos se muestra en la Figura 29.

4.3. Seguimiento por RGBD, oclusión y flujo óptico (RGBDOcc+OF)

El trabajo de Song y Xiao (2013) expone un algoritmo que toma ventaja tanto de la información 2D como de la información 3D que obtiene de las cámaras de profundidad. Este algoritmo funciona en base a un conjunto de descriptores del objeto: histogramas de gradiente orientado de la imagen así como del mapa de profundidad; generación de un modelo 3D de la escena mediante nube de puntos, el cual es escaneado por una *ventana deslizante* de tres dimensiones; manejo de la oclusión mediante la distribución de profundidad del objeto y el fondo; y el uso del algoritmo de flujo óptico para mantener el seguimiento del objeto.

Este algoritmo, aunque no es de los primeros que utilizan mapas de profundidad para la detección de objetos, proporciona dos variantes de detector, uno que solo utiliza información 2D y otro que utiliza información 3D; con lo cual demuestran, hasta cierto punto,

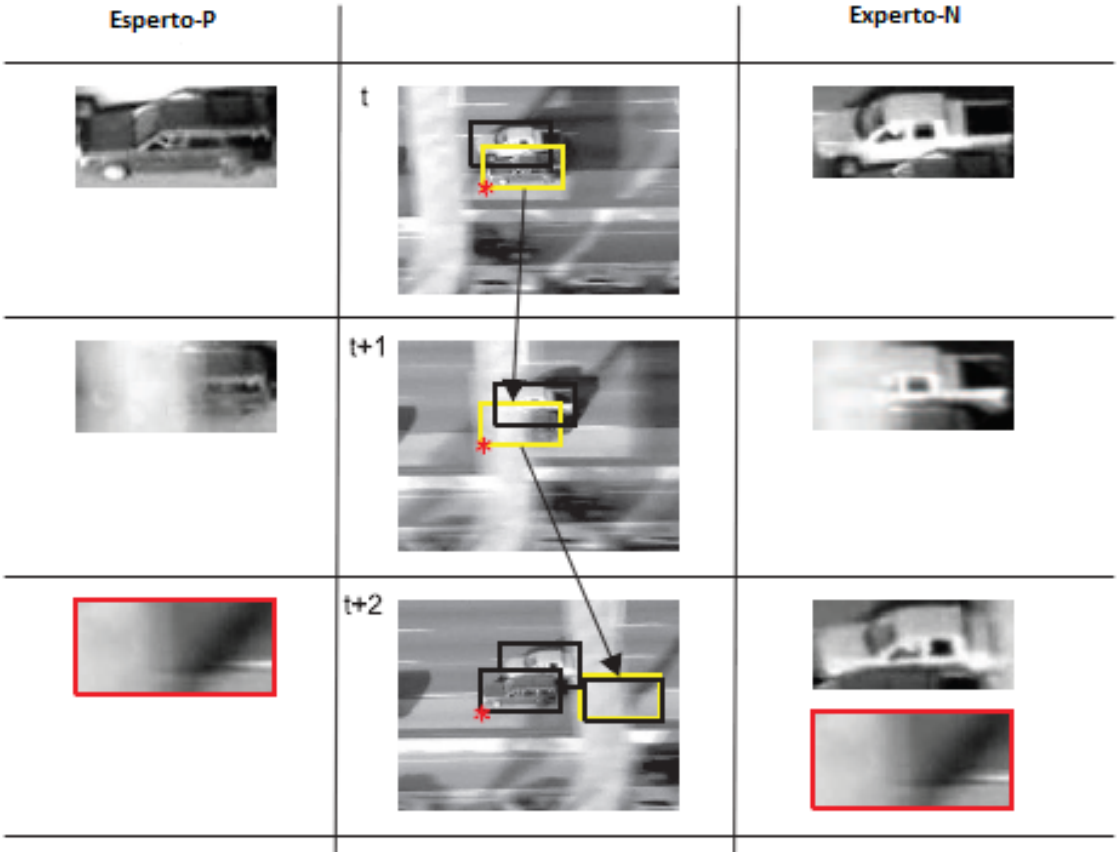

Figura 29: Salida de los expertos P-N, la columna de en medio muestra la compensación de errores (Kalal *et al.*, 2012), pág. 7.

que el añadir información de profundidad en el algoritmo le provee mayor robustez tanto para la localización del objeto como para su descripción y manejo de oclusión, comparado con algoritmos que no utilizan esta información. Aún y cuando el algoritmo presenta un buen desempeño, la basta cantidad de elementos que contiene hacen que afecte su tiempo de procesamiento.

A continuación se presentan dos tipos de configuraciones del algoritmo. La primera utiliza seguimiento por plantilla con algunas características del mapa de profundidad; la segunda, está basada en una nube de puntos 3D que proporciona una caja delimitadora 3D. Ambos utilizan un mecanismo para el manejo de oclusión.

Seguimiento por detección

Se construye un modelo de discriminación del objeto a seguir (un descriptor o conjunto de descriptores, ver Figura 30), y se utiliza para clasificar al objeto en cuadros posteriores. El modelo del objeto contiene los siguientes elementos:

- El primer descriptor es un histograma de gradiente orientado calculado de las imágenes RGB y de profundidad; el HOG de profundidad es tratado como una imagen en escala de gris; el HOG conjunto (RGBD HOG) describe las tecturas locales del objeto así como su forma tridimensional, mejorando de esta manera la robustez contra la iluminación, falta de textura y similitud de color contra el fondo de la imagen.

- Se divide el espacio tridimensional en un conjunto de celdas cúbicas mediante la nube de puntos 3D que proporciona la cámara de profundidad, generando celdas 3D a color y capturando la forma de los objetos dentro de la celda. Cada celda contiene tres características: un histograma de color, el número de puntos en la celda, y la forma 3D de los objetos en la celda.

- Entrenamiento y detección por medio de una *máquina de soporte vectorial* (SVM). Cada uno de los elementos anteriores, RGBD HOG y detector de nube de puntos, entrenan un clasificador SVM. En el primer cuadro se selecciona una caja delimitadora que contenga al objeto, ésta entra al SVM como instancia de entrenamiento positiva; las instancias de entrenamiento negativas se eligen mediante cajas delimitadoras aleatorias dentro del cuadro de video. En cuadros subsecuentes, los descriptores calculados son convolucionados con valores ponderados de la SVM regresando un conjunto de descriptores confiables, seleccionando como el objeto aquel que obtenga el mayor puntaje. Finalmente la SVM es reentrenada usando la caja delimitadora del resultado positivo.

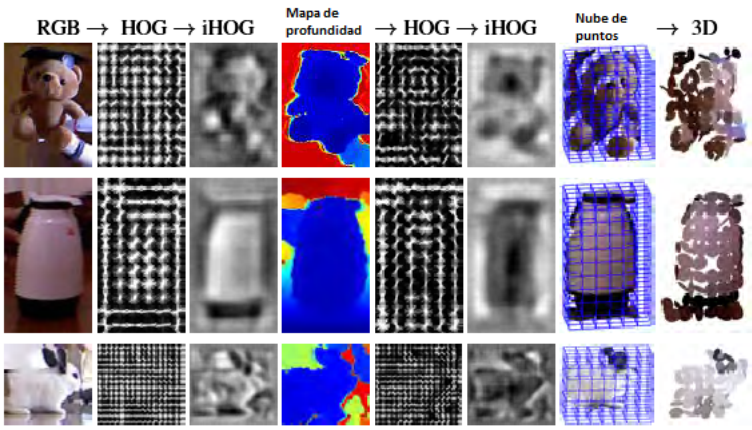

Figura 30: Descriptores del modelo de objeto, las primeras 6 columnas muestran los HOGs RGB y de profundidad, las últimas dos la división en celdas de nube de puntos y el modelo 3D del objeto (Song y Xiao, 2013), pág. 3.

Seguidor de punto

Se utiliza un seguidor de flujo óptico de desplazamiento largo sobre los datos RGB en cuadros consecutivos para datos en 2D. Para seguimiento en 3D se utiliza un algoritmo iterativo de punto más cercano (ICP), el cual calcula iterativamente una transformación rígida que minimiza la suma del error cuadrático medio (MSE) entre dos conjuntos de puntos.

Se asume que el objeto tiene movimiento y velocidad limitados, luego, el seguidor busca al objetivo en el vecindario de la última posición correcta del objeto. La caja delimitadora que obtenga el mayor valor es enviada al módulo de revisión de oclusión, en caso de no estar bloqueada, se considera como una salida positiva del algoritmo. Tanto el detector como el seguidor son inicializados con la información del primer cuadro de video.

Manejo de oclusión

El tratamiento de la oclusión se vuelve directo si se usa información de profundidad. Para detectar la oclusión, se supone que el objeto es el elemento que se encuentra más cerca de la cámara cuando no esta ocluido en la caja delimitadora, se calcula entonces un histograma de profundidad con un pico máximo con la profundidad del objeto. Si otro elemento aparece en la caja delimitadora frente al objeto, se inicia el estado de oclusión, luego, aparece otro pico en el histograma de profundidad, disminuyendo el pico del objeto si el nuevo elemento empieza a bloquearlo, como se muestra en la Figura 31.

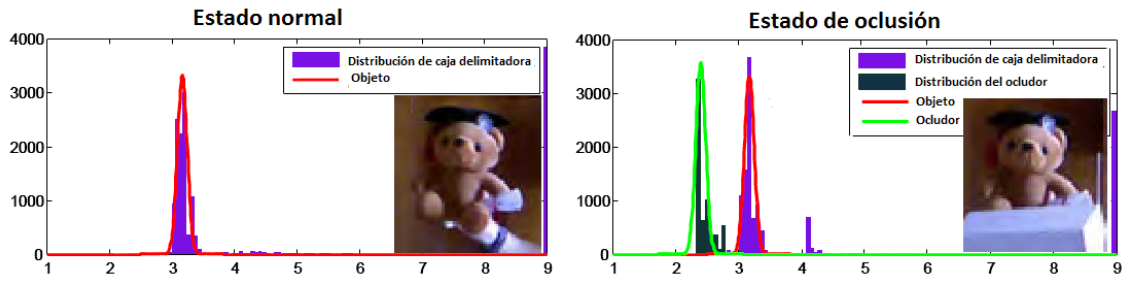

Figura 31: Distribución de profundidad del objeto. A la izquierda, estado normal. A la derecha, en estado de oclusión (Song y Xiao, 2013), pág. 4.

En el cuadro i, el histograma de profundidad dh_i de todos los pixeles en la caja delimitadora puede aproximarse como una distribución Gaussiana $dh_i \sim (\mu_i, \sigma_i^2)$, y la probabilidad de oclusión en el cuado $Occ_i = \sum_{d=0}^{\mu_i - \sigma_i} dh_i(d) / \sum_d dh_i(d)$, donde $dh_i(d)$ es el

valor de profundidad d en el cuadro i, $d = 0$ es la profundidad de la cámara y $\mu_i - \sigma_i$ es el umbral para considerar un punto como parte del elemento oclusor. Los pixeles en la caja delimitadora con una profundidad menor que la del objeto se consideran parte del elemento oclusor. La profundidad del objeto se actualiza cada cuadro, de manera que un objeto que se acerca o aleja de la cámara no sea tratado como oclusión.

El modelo del oclusor es inicializado en cuanto el sistema entra al estado de oclusión, y su posición es actualizada mediante un seguidor de flujo óptico. Cuando el objeto ha sido ocludido, una lista de posibles candidatos son seleccionados por el detector o por una búsqueda local alrededor del ocludor mediante la segmentación de la imagen RGB y la imagen de profundidad. Los candidatos resultado de la segmentación son juzgados por el clasificador SVM. El sistema sale del estado de oclusión cuando un candidato obtiene un puntaje alto y su área visible es comparable al momento antes de entrar al estado de oclusión.

4.4. Seguimiento por flujo óptico conciente de oclusión (OAPF)

Meshgi *et al.* (2014) proponen un seguidor que detecta oclusiones y realiza una recuperación rápida si entra en estado de oclusión. El algoritmo se basa en seguidores de flujo de partículas (Beauchemin y Barron, 1995) para otorgarle un tratamiento probabilístico a la oclusión. El flujo de partículas, es una forma recursiva de los filtros Bayesianos, ha sido utilizado para analizar series de imágenes, con una mayor ventaja en su aplicación a escenarios no lineales y no Gaussianos.

En caso de entrar a una oclusión, el algoritmo cambia su comportamiento a uno más adecuado para su manejo. Los algoritmos de seguimiento por detección, como los de flujo de partículas, cuentan con una serie de problemas durante el estado de oclusión como: falta de un módulo de oclusión, desvío del modelo, óptimos locales en el espacio de características, ruido y otras transformaciones geométricas. Con el objetivo de solucionar algunos de estos problemas, se utiliza la información de profundidad en conjunto con la información de color, movimiento de la cámara y diferenciación espacial de objetos similares.

Algoritmo

Se consideran como partículas a un conjunto de fragmentos de la imagen sobre y alrededor del objeto a seguir. Se inicializan un subconjunto de estas partículas como ocludidas, las cuales comienzan a esparcirse aleatoriamente por la imagen. Si estas partículas se encuentran con un oclusor, permanecen en este estado mientras otras se unen a ese subconjunto. Si una cantidad grande de partículas han sido marcadas como ocludidas, el objeto entra en estado de oclusión y el modelo se detiene; en tal caso, las partículas que se mueven aleatoriamente por la imagen, se empiezan a esparcir hacia afuera desde la última posición conocida del objeto buscando en un área cada vez más grande en cada cuadro de la secuencia de video.

En caso de que una partícula encuentre un objeto parecido al objetivo, esta partícula se replica, el objeto cambia su estado de oclusión y el seguimiento continua. En todo momento cuando no existe oclusión, el algoritmo se comporta de forma similar al seguidor de flujo de partículas (PFT). El comportamiento del algoritmo puede apreciarse en la Figura 32.

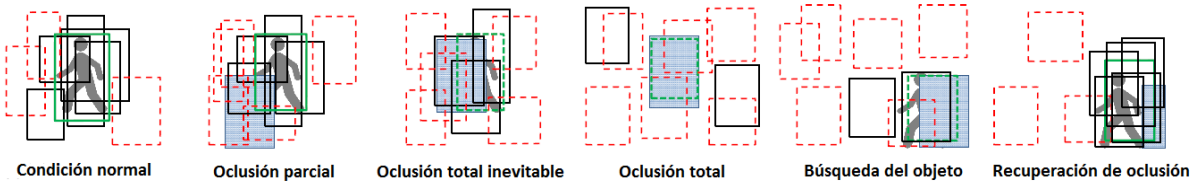

Figura 32: Fases del manejador de oclusión. De izquierda a derecha: el objeto sin oclusión, inicia la oclusión, varias partículas encuentran al ocludor, estado de oclusión, las partículas buscan al objeto desde la última posición conocida, se encuentra al objeto y dichas partículas se replican (Meshgi *et al.*, 2014), pág. 3.

La observación del modelo se divide en dos casos: caso de oclusión $p(Y_t|B_t, Z_t = 1, \vartheta_t)$, y caso de no oclusión $p(Y_t|B_t, Z_t = 0, \vartheta_t)$. El primero permite al seguidor comportarse de manera *exploratoria* durante una oclusión, el segundo propicia un proceso de correspondencia basado en fragmentos entre la partícula B_t y el fragmento ϑ_t,

$$p(I_t|X_t, \vartheta_t) \propto (1 - Z_t)p(Y_t|B_t, Z_t = 0, \vartheta_t) + p(Y_t|B_t, Z_t = 1, \vartheta_t)\,, \tag{78}$$

donde X_t es una partícula en el instante t, B_t es una caja delimitadora que contiene una variable binaria de oclusión Z_t, la imagen $I_t = I_{t,rgb}, I_{t,d}$ contiene componente de color

(rgb) y profundidad d y Y_t un fragmento de imagen dentro de una caja delimitadora.

La probabilidad del caso de oclusión sigue una distribución uniforme, mientras que el caso de no oclusión se deriva de la fusión de diversas características de la imagen. Suponiendo la independencia entre M características, la probabilidad está dada por

$$p(Y_t|B_t, Z_t = 0, \vartheta_t) \propto \prod_{i=1}^{M} \exp\left(-\frac{D_i(f_i(Y_t), \vartheta_{i,t})}{\varrho_i}\right), \tag{79}$$

donde $D_i(f_i(Y_t), \vartheta_{i,t})$ mide la similitud entre la característica f_i extraida del fragmento Y_t y la sección correspondiente del vector de características del fragmento $\vartheta_{i,t}$; ϱ_i es un factor de ponderación; y M es el número de características utilizadas, entre las que se encuentran: proyección de confianza, histograma de color, histograma de profundidad, mapa de bordes, histogramas de gradientes orientados, parámetros de forma 3D, e histograma de texturas.

Si se supone la independencia temporal de la caja delimitadora B_t y la bandera de oclusión Z_t, el modelo de movimiento está dado por

$$p(X_{t+1}|X_t) = p(B_{t+1}, Z_{t+1}|B_t, Z_t) = p(B_{t+1}|B_t)p(Z_{t+1}|Z_t)\,. \tag{80}$$

La estimación de la caja delimitadora B_t y la bandera de oclusión Z_t se calculan mediante la esperanza respecto a la distribución *a posteriori* $p(B_t, Z_t|I_1, ..., I_t)$, luego

$$\mathrm{E}[B_t|I_1, ..., I_t, Z_t = 0] \approx \sum_{j\in\mathfrak{P}_t} B_{j,t} \frac{p(I_t|B_t = B_{j,t}, Z_{j,t} = 0, \vartheta_t)}{\sum_{j'\in\mathfrak{P}'} p(I_t|X_{j',t}, \vartheta_t)} = \widehat{B}_t\,, \tag{81}$$

$$\mathbf{u}(\mathrm{E}[Z_t|I_1, ...I_t] - \delta_{occ}) \approx \mathbf{u}\left(\sum_{j=1}^{N} Z_{j,t} \frac{p(I_t|B_t = B_{j,t}, Z_{j,t}, \vartheta_t)}{\sum_{j'=1}^{N} p(I_t|B_t = Bj', t, Z_t = Z_{j',t}, \vartheta_t)} - \delta_{occ}\right) = \widehat{Z}_t\,, \tag{82}$$

donde $X_{j,t} = \{B_{j,t}, Zj, t\}$ es una muestra de $p(X_t|I_1, ..., I_{t-1}, \vartheta_t)$, δ_{occ} es el umbral de oclusión, $\mathfrak{P}_t$ es el conjunto de partículas no ocludidas, $\mathbf{u}(x)$ es uno si x es positiva y cero de lo contrario.

El modelo del objeto (ϑ) es inicializado por las características detectadas en la caja delimitadora $X_1 = \{B_{init}, Z_1 = 0\}$ usando un detector o una inicialización por el usuario

(B_{init}),

$$\vartheta_1 = \{\vartheta_{1,i}\} = \{f_i(Y_1)\} = \{f_i(B_{init})\},\ i = 1, ..., M\,; \tag{83}$$

para $t > 1$ cada fragmento es actualizado individualmente, a menos que sea detectada una oclusión

$$\vartheta_{i,t+1} = \begin{cases} \vartheta_{i,t} & , si\ \widehat{Z}_t = 1 \\ \lambda_i f_i(\widehat{Y}_t) + (1 - \lambda_i)\vartheta_{i,t} & , si\ \widehat{Z}_t = 0 \end{cases}, \tag{84}$$

donde λ_i es un *factor de olvido* y $\widehat{Y}_t$ es un fragmento de imagen estimado dentro de la caja delimitadora $\widehat{B}_t$.

Capítulo 5. Evaluación

En este capítulo no se pretende exponer experimentos ni resultados de los algoritmos presentados en este trabajo. El propósito de este capítulo pretende presentar los criterios de evaluación, así como las bases de datos y puntos de referencia utilizados para evaluar las características de nuestra propuesta. En la primera sección se presentan los criterios de evaluación del **detector**, el elemento de correspondencia de nuestro algoritmo; el detector se evalúa mediante un conjunto de imágenes sintéticas sobre las que se tiene un completo control. En la segunda sección se presentan los criterios de evaluación del **seguidor**; éste se evalúa usando un punto de referencia que incluye, por la naturaleza de las secuencias de video, varias deformaciones geométricas, las cuales tendrían un costo de evaluación exhaustivo separarlas en cada secuencia de video, por lo que cada secuencia se evalúa *como es* en su totalidad teniendo en cuenta ciertas características clave inherentes al problema de seguimiento.

La Tabla 1 muestra las características del equipo con el que se realizaron los experimentos, mientras que la Tabla 2 muestra las características de la tarjeta gráfica.

Tabla 1: Características del equipo de cómputo

Procesador	Intel Core i7-2600, 3.4 GHz
Memoria	16 GB DDR3 SDRAM
Sistema operativo	Windows 7 Home Premium 64-bits
Lenguaje	C++
Extensiones	OpenCV 2.4 OpenCL 1.2

Tabla 2: Tarjeta gráfica (GPU)

Modelo	ATI Radeon HD 6450
Versión de OpenCL	v1.2
Unidades de cómputo	2
Unidades de proceso	160
Frecuencia de reloj	625 MHz
Memoria global	1 GB
Memoria local	32 KB
Ancho de banda	9.28 GB/seg

5.1. Evaluación del detector

5.1.1. Base de datos

Para la evaluación de los algoritmos de correspondencia se utiliza la base de datos *Amsterdam Library of Object Images* (ALOI)[1]. ALOI es una colección de imágenes a color de mil objetos pequeños (Figura 33), capturados con un gran número de variaciones como: punto de vista, ángulo de iluminación, color de la iluminación, e incluso imágenes estereo.

Figura 33: Ejemplo de algunos de los objetos en la base de datos. Imagen recuperada de `http://aloi.science.uva.nl/`**.**

La variación del punto de vista de cada objeto consta de 72 direcciones, rotando el objeto $5°$ respecto a la posición anterior, obteniendo de esta manera un conjunto de imágenes con rotación fuera de plano con rango $[0°, 355°]$, ver Figura 34. Por cada objeto se tomaron las variaciones de $0°$ a $35°$ de rotación fuera de plano; adicionalmente, se generaron artificialmente rotaciones en plano con las mismas características, variaciones de $5°$.

Para medir las variaciones de iluminación se cuenta con 5 fuentes de iluminación sobre un arco alrededor del objeto a la misma distancia y 3 cámaras para capturar las imágenes desde diferentes ángulos. Cada instancia del objeto es capturada con solo

[1] `http://aloi.science.uva.nl/`

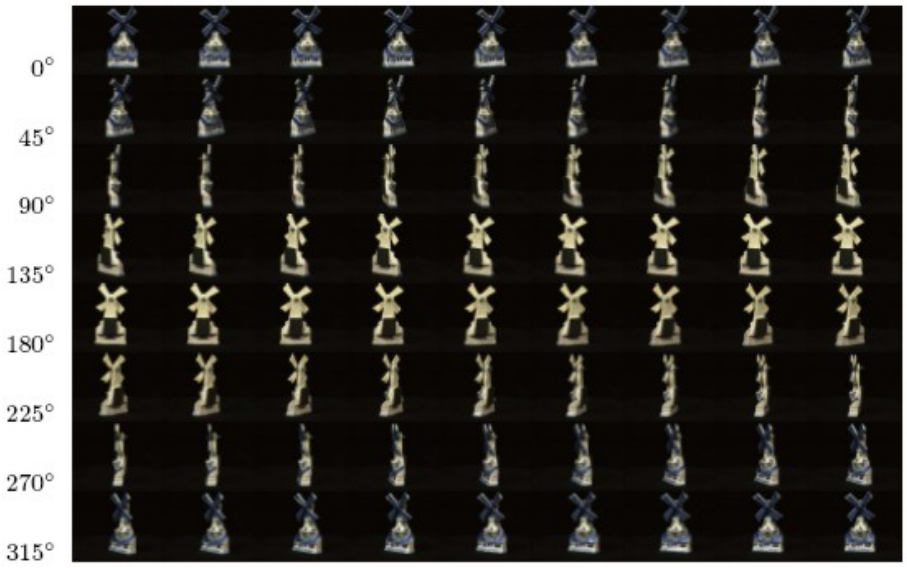

Figura 34: Variaciones de punto de vista. Imagen recuperada de `http://aloi.science.uva.nl/`**.**

una de las fuentes de iluminación activa ($\mathrm{I}1 - \mathrm{I}5$); las 3 cámaras están dispuestas a $0°(c1), 15°(c2), 30°(c3)$, siendo el objeto el que gira, luego el objeto siempre muestra la misma apariencia ante la cámara pero su iluminación varía $15°$ respecto a la imagen anterior. Se presentan otras 3 condiciones de iluminación, cuando las dos lámparas de la derecha se encuentran encendidas ($\mathrm{I}6$), cuando las dos lámparas de la izquierda se encuentran encendidas ($\mathrm{I}7$), y finalmente cuando todas las lámparas se encuentran encendidas ($\mathrm{I}8$). La Figura 35 muestra las variaciones de iluminación. Para este trabajo se tomó el conjunto de variaciones de iluminación $c1\mathrm{I}1 - c1\mathrm{I}8$. No se especifica la cantidad de iluminación de cada cámara en la base de datos.

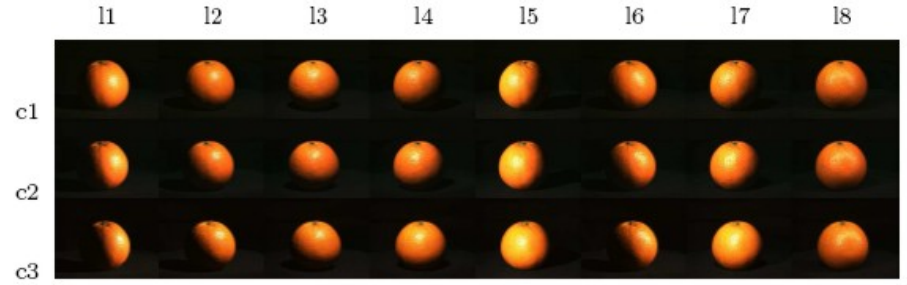

Figura 35: Variaciones de iluminación. Imagen recuperada de `http://aloi.science.uva.nl/`**.**

Cada imagen de la base de datos se encuentra en tres resoluciones: 768×576 pixeles, 384×288 pixeles y 192×144 pixeles, seleccionando esta última resolución para los experimentos. Además, para cada resolución cuenta con máscaras binarias de cada imagen en la base de datos para poder insertar fácilmente los objetos en cualquier imagen. Dado

que la base de datos solo contiene objetos, se seleccionaron 10 escenas variadas de libre acceso en Internet de 1663×965 pixeles cada una, en las cuales se insertaron 10 objetos distintos en diversas posiciones de la imagen con las variaciones antes mencionadas para su evaluación, obteniendo de esta manera 100 instancias diferentes por evaluar. Adicionalmente se realizaron experimentos agregando ruido blanco a las imágenes con un rango de $\sigma_n \in [0, 35]$.

5.1.2. Criterios de evaluación

El detector es la base de todo algoritmo de seguimiento por detección; luego, primeramente debemos asegurarnos que éste sea robusto a diversas transformaciones geométricas, cambios de iluminación y ruido. Para ello se diseñaron experimentos controlados, en los cuales en cada imagen se conoce exactamente cuántos grados de rotación tiene el objeto (en plano y fuera de plano), cambios de escala, posición de la fuente de iluminación y cantidad de ruido blanco (σ_n). De esta manera tenemos una visión objetiva de la robustez del algoritmo de detección, el cual puede ser luego utilizado en secuencias de video del mundo real en las que no tenemos control de estas situaciones, pues son tomadas de un punto de referencia.

El detector evalua la presencia de un objeto dentro de una imagen que contiene a dicho objeto. Dado que tenemos un gran número de variables, se grafica el porcentaje de imágenes donde se encuentra el objeto contra la cantidad de variación que se esté evaluando: rotación, perspectiva, escala, ruido blanco, iluminación. Al graficar estas cantidades tenemos en la vertical el porcentaje de cuadros donde se identificó correctamente al objeto con cierta característica, por ejemplo rotación a $25°$, y en la horizontal todas las instancias del factor a evaluar, por ejemplo los grados de rotación. Se define como el mejor algoritmo de detección aquel que obtenga los mayores puntajes en las gráficas y sea más consistente en general.

5.2. Evaluación del seguidor

5.2.1. Punto de referencia

La evaluación de las secuencias de video se realiza utilizando el punto de referencia *Princeton Tracking Benchmark*[2], el cual consta de 100 secuencias de video, cada cuadro de las cuales se compone de una imagen RGB y un mapa de profundidad. Las secuencias de video fueron adquiridas con el Kinect v1.0 de Microsoft, en ambientes cerrados en los cuales se tuviera un mejor control de la iluminación y con los objetos en un rango de distancias entre 0.5 a 10 metros (Figura 36).

Figura 36: Ejemplo de cuadros de video de la base de datos de Princeton Tracking Benchmark. Cada cuadro contiene información RGB con su mapa de profundidad correspondiente, éste último mapeando los niveles de profundidad a un campo de colores. Imagen recuperada de `http://vision.princeton.edu/projects/2013/tracking/dataset.html`**.**

De entre las 100 secuencias de video, 5 de ellas cuentan con anotaciones manuales de la posición real de los objetos mediante cajas delimitadoras; éstas son seleccionadas en cada cuadro por uno de los autores de los videos para mantener la consistencia de los datos. Cuando ocurre una oclusión, la caja delimitadora se limita a la porción visible del objeto. En cada cuadro de video los objetos se anotan mediante las coordenadas de la esquina superior izquierda, longitud horizontal y longitud vertical de la caja delimitadora $BB = [x, y, h, v]$. Cada secuencia de video tiene una resolución estandar de 640×480 pixeles tanto para la imagen RGB como para el mapa de profundidad, con una velocidad de muestreo de 30 cuadros por segundo (FPS); aunque el Kinect puede capturar imágenes de mayor resolución, la velocidad de muestreo se reduce hasta a 12 FPS; además, el tener la información RGBD en la mismas dimensiones facilita el proceso espacial de las secuencias de video.

[2]http://vision.princeton.edu/projects/2013/tracking/index.html

La base de datos abarca una gran cantidad de características como son:

- **Tipo de objeto** Se divide en tres categorías: objetos rígidos, animales y humanos. Los animales y los humanos contienen una gran variedad de partes deformables, lo que dificulta el seguimiento. Los objetos rígidos, como juguetes y caras, solo presentan rotaciones y traslaciones, por lo que es más sencillo trabajar con éstos.
- **Tipo de escena** Cada secuencia de video tiene diferente cantidad de objetos estáticos y/o dinámicos; además, la cámara puede ser móvil o fija.
- **Presencia de oclusión** La mayoría de las secuencias contienen cierto grado de oclusión de los objetos a seguir, el tiempo de oclusión puede ser de unos cuantos cuadros (milisegundos) a un par de segundos; durante este tiempo el objeto puede cambiar su apariencia.
- **Variaciones de la caja delimitadora** La caja delimitadora puede cambiar de tamaño a lo largo de la secuencia, dependiendo de la posición y escala del objeto en cada cuadro de video.

5.2.2. Criterios de evaluación

Las cinco secuencias que cuentan con anotaciones cuadro por cuadro son: *bear_front*, *child_no1*, *face_occ5*, *new_ex_occ4*, *zcup_move_1*. Estas secuencias contienen prácticamente todas las características mencionadas anteriormente, además contienen anotaciones cuadro por cuadro de la posición verdadera del objeto. La evaluación de resultados se realiza con estas cinco secuencias, pues nos permiten realizar comparaciones más objetivas con los otros algoritmos al tener una base no discresional del desempeño de los algoritmos. A continuación se describen las cinco secuencias de validación:

1. ***bear_front*** Esta secuencia muestra a un oso de peluche de frente, el cual es movido a través de la escena, el oso es ocludido en varias ocasiones por una caja blanca; además de la obvia traslación del oso, éste también presenta ligeron cambios de escala, rotación en plano y ligeros cambios de apariencia por rotación fuera de plano. El fondo de la escena es de color café, lo cual puede dificultar un poco la

discriminación cuando se usa color. Figura 38. La secuencia tiene una longitud de 295 cuadros.

2. ***child_no1*** Esta secuencia muestra a un niño que camina de un lado de la escena al lado contrario acercándose un poco a la cámara. En el trayecto el niño cambia constantemente su apariencia hasta llegar a su posición final en la cual se pone en cuclillas de espalda. No hay oclusión. Figura 39. La secuencia tiene una longitud de 164 cuadros.

3. ***face_occ5*** En esta secuencia se muestra el rostro de una persona, el cual es constantemente ocludido por un libro desde diferentes ángulos. El objetivo es la cara de la persona. Figura 40. La secuencia tiene una longitud de 330 cuadros.

4. ***new_ex_occ4*** Esta secuencia muestra un pasillo con varias personas caminando de un lado al otro de la escena. El objetivo es una mujer que camina del lado izquierdo al derecho de la escena, siendo ocludida por un hombre en los últimos cuadros de video, saliendo de la oclusión poco antes de terminar la secuencia. Figura 41. La secuencia tiene una longitud de 51 cuadros.

5. ***zcup_move_1*** En esta secuencia se presenta una tetera, la cual es trasladada por la escena con ligeros cambios de escala y rotación. Esta secuencia no presenta oclusión. Figura 37. La secuencia tiene una longitud de 370 cuadros.

Figura 37: zcup_move_1

Para evaluar el desempeño de los algoritmos se usa el criterio del desafío PASCAL VOC (Everingham *et al.*, 2010), en el cual se toman las cajas delimitadoras del objeto, la de posición verdadera y la generada por el algoritmo y son juzgadas como verdaderos positivos o falsos positivos al calcular la cantidad de empalme de éstas. Para considerarse

Figura 38: bear_front

Figura 39: child_no1

Figura 40: face_occ5

Figura 41: new_ex_occ4

como una detección positiva, el área de empalme ov_i entre la caja delimitadora de la posición verdadera BB_{gt} y la caja delimitadora generada por el algoritmo BB_{tar} en el

cuadro i debe exceder el 0.5 ($50\,\%$) al aplicar la fórmula

$$ov_i = \begin{cases} \frac{área(BB_{tar} \cap BB_{gt})}{área(BB_{tar} \cup BB_{gt})}, & \text{si existen } BB_{tar} \text{ y } BB_{gt} \\ 0, & \text{de otra manera} \end{cases}, \tag{85}$$

donde $BB_{tar} \cap BB_{gt}$ representa la intersección de las cajas delimitadoras y $BB_{tar} \cup BB_{gt}$ representa la unión de las cajas delimitadoras. Mediante este criterio se genera un gráfico donde se mide el empalme (horizontal) contra el porcentaje de cuadros que cumplen con cierto empalme (vertical). Por ejemplo, suponga que un objeto tiene un $70\,\%$ de empalme con el cuadro delimitador verdadero, luego en el gráfico sumará uno en cada casilla horizontal, de 0 hasta 0.7, y así cada cuadro de la secuencia de video. El desempeño del algoritmo en una secuencia de video se mide como el área bajo la curva del gráfico de empalme desde el umbral (0.5) hasta el máximo (1.0) (Song y Xiao, 2013; Everingham *et al.*, 2010).

La segunda métrica de desempeño utilizada es la cantidad de errores que comete el algoritmo, estos errores se clasifican en tres tipos (ver Figura 42):

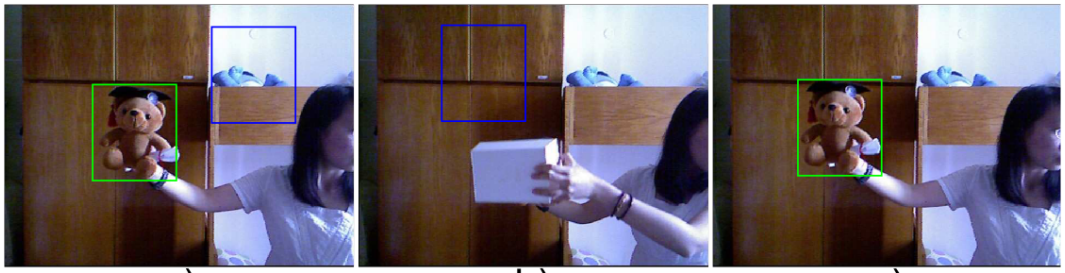

Figura 42: Descripción gráfica de los Tipos de Errores. La caja delimitadora verdadera es el recuadro verde y la caja delimitadora del algoritmo es el recuadro azul. a)Error Tipo I, b)Error Tipo II, c)Error Tipo III

- **Error Tipo I** El objeto está visible, pero la caja delimitadora del objeto no se empalma con la caja delimitadora verdadera.
- **Error Tipo II** El objeto no esta visible y el algoritmo da como resultado una caja delimitadora.
- **Error Tipo III** El objeto esta visible, existe la caja delimitadora verdadera y el algoritmo no produce una caja delimitadora.

Finalmente, se presenta el intervalo de confianza de las secuencias sobre el valor medio de empalme, para determinar si los algoritmos son realmente superiores unos de otros. Se utiliza el $95\,\%$ de nivel de confianza,

$$\overline{X} - 1.96\frac{S}{\sqrt{\mathrm{n}}} \leq \mu \leq \overline{X} + 1.96\frac{S}{\sqrt{\mathrm{n}}} \tag{86}$$

donde S es la desviación estándar de la cantidad de muestras n, $\overline{X}$ es la media muestral calculada y μ es la media real.

Capítulo 6. Algoritmo de seguimiento basado en ventanas circulares

En este capítulo se presenta el algoritmo propuesto, el cual será denominado como CWTA[1], donde el algoritmo CWMA[2] (Miramontes-Jaramillo *et al.*, 2013) es su componente de correspondencia. Todo algoritmo de seguimiento está compuesto de tres partes: el modelo del objeto, o representación de objeto, el cual define las estructuras y la forma con las cuales se representará e identificará el objeto de interés dentro de la imagen; la identificación del objeto, la cual en seguimiento por detección se realiza fundamentalmente por detección o correspondencia, para lo cual se define un descriptor, el cual se busca para el objeto de interés dentro de la escena; y el modelo de movimiento, el cual se encarga de predecir la posición en la cual el objeto se encontrará en los cuadros de video subsecuentes, normalmente se utiliza información histórica de la secuencia de video. Cada uno de estos elementos será descrito a detalle en este capítulo.

En capítulos anteriores se revisaron los algoritmos del Estado-Del-Arte respecto a correspondencia y seguimiento. En este capítulo se comparan éstos algoritmos con CWTA utilizando las métricas y bases de datos expuestos en el capítulo anterior. En cada sección se revisarán algunas ideas que se exploraron a lo largo del trabajo de tesis, la selección de características y estructuras que conforman el algoritmo final son señaladas y justificadas en cada caso.

Como en todo algoritmo de seguimiento por detección, se describirá primero el descriptor utilizado, el cual es invariante a la forma de la ventana seleccionada para el procesamiento del algoritmo; luego se describe el modelo del objeto; continuando con el algoritmo de correspondencia para la detección (Miramontes-Jaramillo *et al.*, 2014), con modificaciones en la forma de la ventana y el uso de la GPU para alcanzar el procesamiento en tiempo real; finalizando con el modelo y técnicas de seguimiento.

6.1. Descriptor

El descriptor es una estructura o un conjunto de características que tienen como objetivo *describir* inequívocamente el área de la cual se extraen. Los descriptores son fre-

[1]Circular Window Tracking Algorithm
[2]Circular Window Matching Algorithm

cuentemente vectores de operaciones matemáticas y estadísticas que se aplican sobre el vecindario de un pixel o a un conjunto de pixeles, algunos ejemplos de descriptores sencillos son el histograma de color o intensidad de los pixeles en un área específica, la extracción de un conjunto de estadísticos en un pequeño vecindario (ventana) en un área, entre otros; descriptores más elaborados incluyen las características SIFT (Lowe, 2004), SURF (Bay *et al.*, 2008), histogramas de gradiente orientado (HOG) (Dalal y Triggs, 2005). Por lo general el área de interés son los objetos que se quieren buscar en la imagen o en la secuencia de video, sin embargo, muchos algoritmos extraen descriptores del fondo de la imagen como entrenamiento con ejemplos negativos.

En este trabajo elegimos trabajar con histogramas de gradiente orientado pues presentan varias características benéficas para la discriminación positiva de los objetos. Existen varias formas de representar los histogramas de gradiente, ya sea solo la distribución de los ángulos de orientación de cada pixel, o la suma de la magnitud de cambio correspondiente a la orientación del pixel; en cada caso el valor del ángulo determina la casilla correspondiente en el histograma. Los HOGs nos permiten reducir la dimensionalidad de la información de la imagen, proporcionando una *firma* o *descripción* del área de donde se extrajo; donde los bordes y esquinas detectadas son las más distintivas, siendo muy difícil duplicar esta firma por otra área diferente a la del objetivo.

Para el cálculo del gradiente en cada pixel utilizamos un operador de gradiente sencillo, ya que se busca conservar el cambio marcado en bordes y esquinas, e incluso de la textura, preservando la calidad de las características (Dalal y Triggs, 2005). Se pueden utilizar otros operadores como Sobel, Robinson o Kirsch (Pratt, 2007); sin embargo, estos contienen un efecto de suavizado que pueden destruir las estructuras finas de los bordes y esquinas. Los operadores utilizados en direcciones horizontal y vertical se definen como

$$g_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}, \quad g_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}. \tag{87}$$

Una vez definidos los operadores de gradiente, los aplicamos a cada pixel para obtener el valor de máximo cambio en las direcciones horizontal (x) y vertical (y), luego se calcula

la magnitud y la orientación de cada pixel redefiniendo las Ecuaciones (35) y (36) como

$$mag(x,y) = \sqrt{g_x^2 + g_y^2}\,, \tag{88}$$

$$ori(x,y) = \arctan\left(\frac{g_y}{g_x}\right)\,, \tag{89}$$

donde la orientación está dada en radianes en un rango de $[-\pi/2, \pi/2]$; sin embargo, para la generación de un histograma es preferible que el ángulo se encuentre en grados en un rango de $[0°, 359°]$, para ello deben tomarse consideraciones respecto al signo de g_x, g_y y los casos especiales cuando g_x toma el valor de 0, de la siguiente manera

$$ori'(x,y) = \frac{180}{\pi} * \begin{cases} \arctan\left(\frac{g_y}{g_x}\right) & \text{si } g_x > 0 \text{ y } g_y \geq 0 \\ \arctan\left(\frac{g_y}{g_x}\right) + 2\pi & \text{si } g_x > 0 \text{ y } g_y < 0 \\ \arctan\left(\frac{g_y}{g_x}\right) + \pi & \text{si } g_x < 0 \\ +\frac{\pi}{2} & \text{si } g_x = 0 \text{ y } g_y > 0 \\ +\frac{3\pi}{2} & \text{si } g_x = 0 \text{ y } g_y < 0 \\ \text{indeterminado} & \text{si } g_x = 0 \text{ y } g_y = 0 \end{cases}. \tag{90}$$

La Figura 43 muestra gráficamente los conceptos de gradiente y magnitud[3].

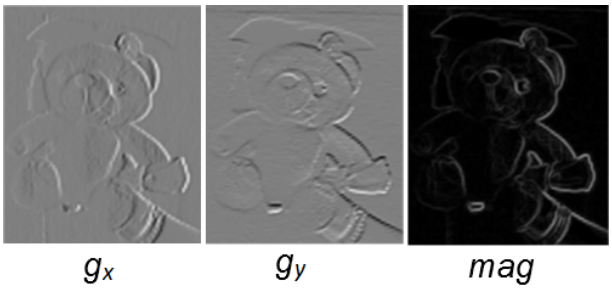

Figura 43: Ejemplo de gradientes g_x y g_y, y la magnitud mag con una cuantización de valores en el rango $[0, 255]$ para su correcto despliegue en pantalla. Los gradientes direccionales muestran los cambios prominentes en la imagen, la magnitud muestra la fuerza de estos cambios, definiendo bordes y esquinas.

Una imagen puede verse afectada por diversas condiciones, entre ellas, errores del sensor, ligeros cambios de iluminación entre dos cuadros consecutivos y condiciones ambientales. Cada una de éstas puede afectar el valor de pixel ligeramente, aplicando a

[3]La figura de orientación no se presenta pues involucraría un gráfico con flechas en cada pixel indicando la dirección del gradiente, dicho gráfico resultaría poco práctico.

su vez ligeros cambios en los gradientes del pixel, y a su vez la casilla del histograma, pues el ángulo variará. Un histograma de 360° es poco recomendable; en su lugar se propone seleccionar una cantidad de casillas Q determinado por la desviación estándar del ruido σ_n en la imagen

$$Q = \left\lceil \frac{360}{\sigma_n} \right\rceil . \tag{91}$$

El valor de Q compensa los pequeños cambios de orientación introducidos por factores externos. La operación de *techo* permite obtener un número entero de casillas.

Debido a la naturaleza circular del histograma de gradiente orientado, las orientaciones en la frontera entre 360° y 0° no pueden ser clasificados en la misma casilla Q_i, por tal motivo aplicamos la siguiente operación que permite compensar esta característica circular,

$$\varphi(x, y) = \left\lfloor \frac{Q}{360} ori'(x, y) + \frac{1}{2} \right\rfloor , \tag{92}$$

donde el factor $\frac{1}{2}$ rota el origen del histograma con un giro contra las manecillas del reloj, de tal forma que los valores en la frontera tengan la misma casilla. El cálculo de la magnitud y la orientación de gradiente es trivial en una GPU ($\mathrm{O}(1)$), pues se realiza la misma operación en cada pixel del fragmento de la imagen, de tamaño $\mathrm{n} = w \times h$, concurrentemente, como se muestra en el siguiente algoritmo:

Algoritmo 3 Gradiente de imagen con n procesadores

Entrada: n valores de pixel en una matriz A
Salida: Matrices de magnitud (mag) y orientación (φ)

1: $p =$ obtenerProcesador$()$
2: $i =$ obtenerPosiciónHorizontal(p)
3: $j =$ obtenerPosiciónVertical(p)
4: $gx(i, j) = -A(i - 1, j) + A(i + 1, j)$
5: $gy(i, j) = -A(i, j - 1) + A(i, j + 1)$
6: $mag(i, j) = \sqrt{g_x^2 + g_y^2}$
7: $ori'(i, j) = \arctan\left(\frac{g_y}{g_x}\right) * \frac{180}{\pi}$
8: $\varphi(i, j) = \left\lfloor \frac{Q}{360} ori'(i, j) + \frac{1}{2} \right\rfloor$

Existe un número constante de pasos; se puede ver claramente que el tiempo de procesamiento es $\mathbb{T}(\mathrm{n}) = \mathrm{O}(1)$; este proceso se ejecuta n veces, una vez por procesador, por lo tanto el trabajo realizado es $\mathbb{W}(\mathrm{n}) = \mathrm{O}(n)$.

Mediante el cómputo de gradientes radiales (Takacs *et al.*, 2013) es posible obtener orientaciones de gradiente invariantes a rotación. El procesamiento de gradientes radiales depende de la posición física del pixel respecto al centro del fragmento a procesar; por lo tanto, no es viable utilizarlo en nuestro trabajo dado que en el proceso de correspondencia se realiza un procesamiento iterativo con el objetivo de reducir el número de operaciones que se ejecutan; los gradientes radiales no nos permiten realizar este procesamiento iterativo, pues hay que recalcular todos los gradientes cada vez que la ventana avance pixel a pixel por el fragmento, como se verá en las siguientes secciones. Sin embargo, es interesante tener en cuenta estos gradientes para futuras referencias. Sea R un disco cerrado con radio r y centro c, y sea $p_i = (x, y)$ un punto dentro del disco; En cada punto p se definen dos vectores ortogonales relativos al origen, si $(u, v) = p - c$ y $\psi = \arctan(v/u)$, entonces

$$rad = (\cos\psi, \sin\psi)\,, \tag{93}$$

$$tan = (-\sin\psi, \cos\psi)\,, \tag{94}$$

donde rad y tan son las direcciones radial y tangencial en $p - c$; la Figura 44 muestra los ángulos $\psi \in R$.

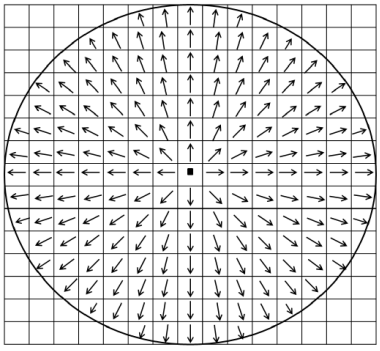

Figura 44: Máscara de ángulos radiales en R.

El tamaño de R debe ser igual que el fragmento utilizado. Sea T la matriz que transforma los gradientes cartesianos a gradientes radiales, dada por

$$T = \begin{bmatrix} rad \\ tan \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix}. \tag{95}$$

Aplicando la matriz de transformación al gradiente $g = [g_x, g_y]$ se obtiene

$$g' = Tg^{\mathrm{T}} = \begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} g_x \\ g_y \end{bmatrix}. \tag{96}$$

El resto del procesamiento continúa de la misma manera que los gradientes normales para obtener mag y φ. Nuevamente, los gradientes radiales son ilustrativos y una buena opción en caso de encontrar una técnica para calcularlos iterativamente solo en la frontera del fragmento.

En este trabajo se decidió construir los histogramas de gradiente orientado por medio de *voto de magnitud*. Cada pixel tiene una orientación y magnitud correspondientes, mientras que el primero representa la casilla en el histograma, el segundo representa el valor que se añade a esta casilla. Sin embargo, en la práctica, la orientación de un pixel difícilmente va a corresponder a una sola casilla; luego, el valor correspondiente a la magnitud se divide proporcionalmente entre las dos casillas en el histograma más cercanas a la orientación del pixel como se muestra en la Figura 45. La proporción de magnitud correspondiente a cada casilla se definen como

$$\lfloor MAG(x,y) \rfloor = mag(x,y) \cdot \{1 - [\varphi(x,y) - \lfloor \varphi(x,y) \rfloor]\}, \tag{97}$$

y

$$\lceil MAG(x,y) \rceil = mag(x,y) \cdot [\varphi(x,y) - \lfloor \varphi(x,y) \rfloor]. \tag{98}$$

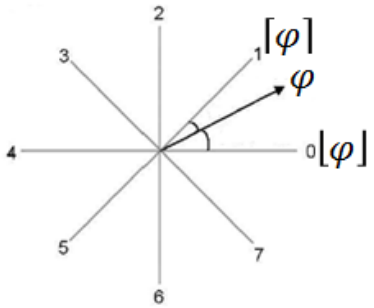

Figura 45: Ilustración de las dos casillas más cercanas a φ: $\lfloor\varphi\rfloor$ y $\lceil\varphi\rceil$. Imagen recuperada de (Rodríguez y Perronnin, 2008), pág. 4.

Suponiendo que el histograma empieza vacío, cada pixel proporciona su información al histograma de gradiente orientado mediante dos operaciones dadas por

$$HOG(\lfloor\varphi(x,y)\rfloor) += (\lfloor\varphi(x,y)\rfloor) + \lfloor MAG(x,y)\rfloor\,, \tag{99}$$

y

$$HOG(\lceil\varphi(x,y)\rceil) += (\lceil\varphi(x,y)\rceil) + \lceil MAG(x,y)\rceil\,. \tag{100}$$

Finalmente, se calcula un histograma centrado y normalizado, de la siguiente manera,

$$\overline{HOG}(\varphi) = \frac{HOG(\varphi) - Media}{\sqrt{Var}}\,, \tag{101}$$

donde $Media$ y Var son la media muestral y la varianza del histograma. Existen maneras más elaboradas de construir histogramas de gradiente orientado (Dalal y Triggs, 2005); sin embargo, son computacionalmente exhaustivas y no presentan ventajas significativas al histograma de gradiente orientado sencillo presentado en esta sección.

El algoritmo para calcular el histograma se compone de dos partes, la primera parte divide los datos de entrada n en $\log \mathrm{n}$ subconjuntos iguales, con un procesador operando sobre cada subconjunto para generar un histograma local de manera serial, lo cual toma $\mathbb{T}(\mathrm{n}) = \mathrm{O}(\frac{\mathrm{n}}{\log \mathrm{n}})$ pasos con $\mathbb{W}(\mathrm{n}) = \mathrm{O}(\mathrm{n})$ operaciones. La segunda parte utiliza un conjunto de procesadores ($\frac{\log \mathrm{n}}{2}$) por cada casilla Q del histograma, cada procesador suma la casilla correspondiente de todos los histogramas locales utilizando el Algoritmo 2 formando un histograma global; luego la suma de las casillas en los histogramas locales toma $\mathbb{T}(\mathrm{n}) = \mathrm{O}(\log\log \mathrm{n})$ pasos con $\mathbb{W}(\mathrm{n}) = \mathrm{O}(Q \log \mathrm{n})$ operaciones. Finalmente el tiempo de ejecución es $\mathbb{T}(\mathrm{n}) = \mathrm{O}(\frac{\mathrm{n}}{\log \mathrm{n}})$ pasos con $\mathbb{W}(\mathrm{n}) = \mathrm{O}(\mathrm{n})$ operaciones cuando $n > Q$ o $\mathbb{W}(\mathrm{n}) = \mathrm{O}(Q \log \mathrm{n})$ cuando $n \leq Q$.

6.2. Modelo de objeto

En esta sección se discute la forma en que se representa a los objetos de interés en el algoritmo. Se discute la estructura y sus características, así como la forma de la ventana que se eligió para obtener el mejor desempeño posible.

La idea principal es crear una estructura que represente al objeto de interés inequívocamente, y que sea lo suficientemente flexible para adaptarse a ligeros cambios en el objeto. La primera idea fue la de utilizar una ventana rectangular que abarcara la totalidad del objeto. Sin embargo, cuando el objeto sufre deformaciones geométricas como la rotación, la ventana rectangular puede perder información valiosa e introducir errores a los datos del objeto (Figura 46a). La segunda idea fue la de utilizar un disco circunscrito en un rectángulo abarcando todo o la mayor parte del objeto, en este caso la información del objeto permanecerá invariante hasta cierto punto dentro de la ventana circular (Figura 46b). Estas dos ideas fueron exploradas en su momento; sin embargo, se descartaron por el hecho de que a lo largo de una secuencia de video el forndo del objeto cambia constantemente, lo cual introduce errores del fondo al descriptor del objeto. Por lo tanto, nos decidimos por una tercera opción, utilizar un conjunto de discos dentro del objeto abarcando tanta área como sea posible, de esta forma se garantiza que la información dentro de los discos es del objeto sin información del fondo (Figura 46c); en este caso no se consideró el uso de un conjunto de rectángulos por las deficiencias inherentes de rotación del primer caso.

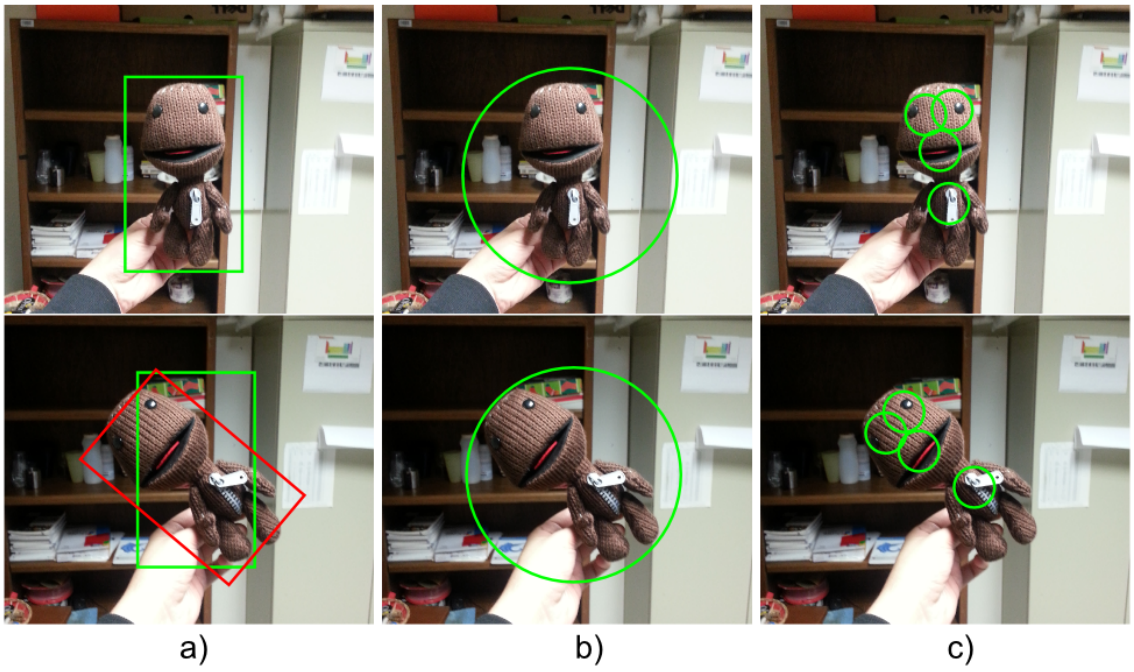

Figura 46: Tipos de posibles estructuras de descripción del objeto. La imagen de arriba representa la posición estándar, la imagen de abajo representa al objeto rotado. La columna a) muestra la ventana rectangular, la columna b) muestra una ventana circular global, y la columna c) muestra un conjunto de ventanas circulares.

Como puede verse, un conjunto de ventanas circulares tiene varias ventajas sobre una sola ventana circular o sobre ventanas rectangulares. El modelo de objeto es una estructura geométrica basada en ventanas circulares. Sea W_i un conjunto de discos cerrados, dados por

$$W_i = \{(x, y) \in \mathbb{R} \,|\, (x - x_i)^2 + (y - y_i)^2 \leq r^2, i = 1, ..., M\}\,. \tag{102}$$

Es importante notar, que tener solamente un conjunto de discos no es suficiente para tener un modelo robusto. Posteriormente, calculamos la distancia D_{ij} entre los centros de cada par de discos, como se muestra a continuación

$$D_{ij} = \left\{ \sqrt{(x_i - x_j)^2 + (x_i - x_j)^2},\, i = 1, ..., M;\, j = i + 1, ..., M \right\}\,, \tag{103}$$

y los ángulos entre los centros de cada tres discos adyacentes θ_i (Miramontes-Jaramillo *et al.*, 2013),

$$\theta_i = \left\{ \cos^{-1} \left[\frac{D^2_{i,j+1} + D^2_{i,j+2} - D^2_{i,j+3}}{2D^2_{i,j+1} D^2_{i,j+2}} \right],\, i = 1, ..., M - 2 \right\}\,, \tag{104}$$

donde (x, y) son las coordenadas del centro y r el radio de los discos, la estructura se muestra en la Figura 47.

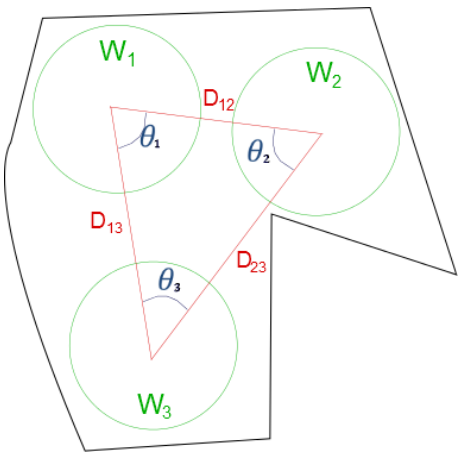

Figura 47: Estructura geométrica del modelo del objeto definida dentro del área de soporte del objeto.

El Algoritmo 4 muestra el cálculo de la ventana circular. Es sencillo ver que el tiempo de ejecución es constante $\mathbb{T}(\mathrm{n}) = \mathrm{O}(1)$, mientras que el trabajo $\mathbb{W}(\mathrm{n}) = \mathrm{O}(\mathrm{n})$, dado que se ejecuta por cada pixel en la ventana. La estructura geométrica consta de los vectores W_i, D_{ij}, θ_i, los cuales se pueden calcular en tiempo constante $\mathbb{T}(\mathrm{n}) = \mathrm{O}(1)$ aplicando las

Ecuaciones (102), (103) y (104), el trabajo depende de la cantidad de discos a calcular $\mathbb{W}(\mathrm{n}) = \mathrm{O}(M)$.

Algoritmo 4 Ventana circular con n procesadores, donde $\mathrm{n} = (2r)^2$

Entrada: Tamaño del radio r
Salida: Un vector C, donde el índice del vector corresponde a la longitud de la ventana $(2r)$ y el valor es la distancia del centro de la ventana hacia el perímetro del círculo (máximo $\pm r$.

1: $p =$ obtenerProcesador$()$
2: $i =$ obtenerPosiciónHorizontal$(p) - r$ //Posición centrada
3: $j =$ obtenerPosiciónVertical$(p) - r$ //posición centrada
4: **si** $\sqrt{i^2 + j^2} \leq r \ y \ \sqrt{(i-1)^2 + j^2} > r \ y \ \sqrt{(i+1)^2 + j^2} < r$ **entonces**
5: $\quad C(i) = j$
6: **fin si**

Dado que el círculo es una figura simétrica, solo se requiere un vector con la distancia del centro al perímetro, luego, puede calcularse el círculo de forma horizontal, si los índices del vector son el eje y, o vertical, si los índices son el eje x. Este principio es utilizado por el algoritmo de correspondencia para realizar la actualización iterativa de los histogramas.

Es importante que los discos sean de la misma dimensión, dado que facilita el procesamiento en una GPU. El tamaño y posición de los discos en el objeto son seleccionados manualmente, por falta de un algoritmo adecuado de segmentación y cálculo de estos discos; el radio de los discos es seleccionado de tal manera que se facilite su procesamiento en la GPU. Dado que los objetos en una secuencia de video sufren ligeras modificaciones geométricas, sus características tienen un grado de flexibilidad ϵ respecto al cuadro de video anterior, luego, la estructura se va modificando poco a poco a lo largo de la secuencia.

Se seleccionan un número M de ventanas circulares dentro del objeto, calculando un histograma de gradiente orientado en las áreas circulares de cada disco M_i, formando de esa manera el modelo del objeto. Es importante recordar que la información dentro de cada disco es invariante a ligeras deformaciones geométricas y los HOGs calculados son invariantes a rotación, simplemente tienen un desplazamiento circular igual a la rotación del objeto. Lo ideal es encontrar todos los discos en el cuadro de video dado que cada disco contiene información del objeto; sin embargo, en algunos casos solo un subconjunto

de los discos está visible o es encontrado, luego, la estructura nos permite determinar la posición del resto de los discos para determinar el fragmento de búsqueda del objeto para el siguiente fragmento.

6.3. Correspondencia

En las secciones anteriores se presentó el descriptor basado en histogramas de gradiente orientado y el modelo del objeto, que consta de una estructura de discos con distancias y ángulos conocidos, donde se extrae un HOG de cada uno de los M discos dentro del objeto. Antes de comenzar el proceso de correspondencia y seguimiento, se supone que existe el modelo del objeto conformado por dicha estructura geométrica, donde el tamaño del disco se utiliza como ventana deslizante en el proceso de correspondencia. El proceso comienza con el preprocesamiento del cuadro de video para reducir los efectos de ruido e iluminación, para después buscar el objeto por medio de un proceso iterativo y paralelo de ventanas deslizantes.

6.3.1. Preprocesamiento

6.3.1.1. Reducción de ruido

Uno de los primeros pasos en el procesamiento de señales digitales es el tratamiento del ruido, con el objetivo de atenuar lo más posible sus efectos. El ruido modifica las características de una imagen. Afectando el valor de los pixeles, la magnitud y orientación de sus gradientes en función de la desviación estándar del mismo ruido. En este trabajo suponemos que las imágenes y secuencias de video están contaminadas por ruido blanco aditivo, el cual es el tipo de ruido más común, causado por el dispositivo de adquisición de imágenes.

El ruido blanco es un proceso aleatorio el cual no está correlacionado, por lo tanto, su función de autocorrelación es la función delta de Kronecker y su espectro de varianza es una constante como se muestra en la Figura 48. Esto significa que el proceso contiene todas las frecuencias y todas muestran el mismo valor en el espectro de varianza. Como se muestra en la Figura 49, la varianza del ruido puede ser calculada mediante la interpolación lineal de los valores vecinos en el origen de la autocorrelación en la imagen ruidosa

para calcular la varianza de la imagen ideal, donde la diferencia de ésta con el origen de la autocorrelación se considera la varianza del ruido blanco.

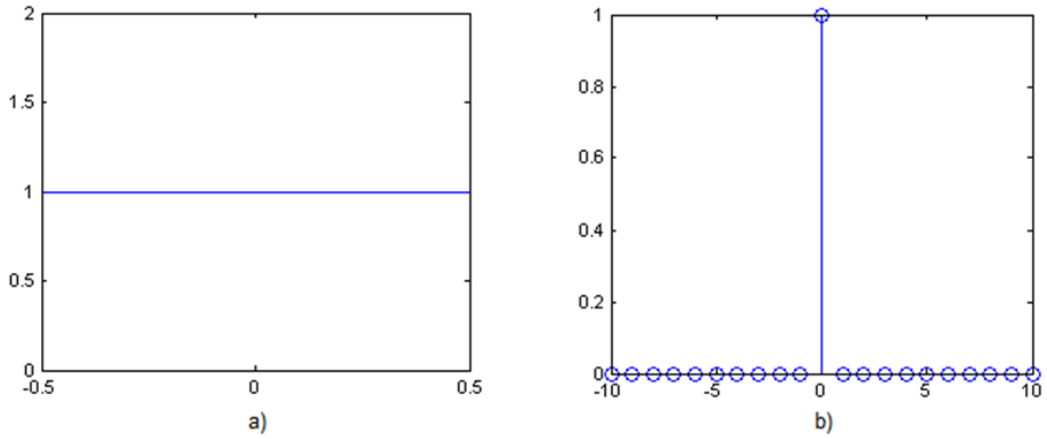

Figura 48: a) Espectro de varianza de ruido blanco. b) Autocorrelación de ruido blanco.

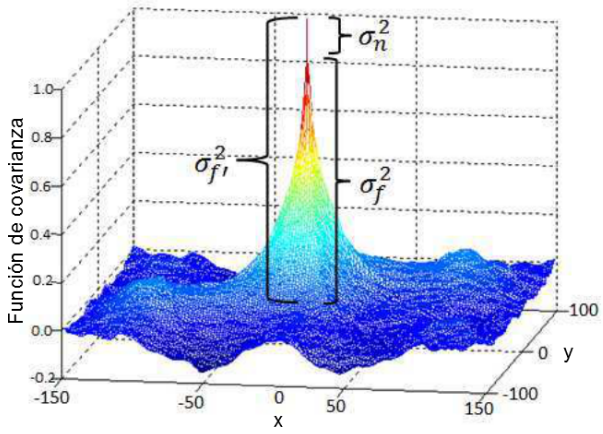

Figura 49: Función de autocorrelación de $f'(x)$, la imagen observada. Estimación de la varianza σ_f^2 de la imagen ideal $f(x)$ en el origen del mapa de autocorrelación y de la varianza del ruido σ_n^2.

El proceso de autocorrelación es muy costoso computacionalmente; en el espacio de frecuencias la operación de correlación se reduce a una multiplicación punto por punto,

$$\mathrm{F}(\omega) = \mathcal{F}\{f(x)\}\,, \tag{105}$$

$$\mathrm{EP}(\omega) = \mathrm{F}(\omega)\mathrm{F}^*(\omega)\,, \tag{106}$$

$$\mathrm{A}(\varrho) = \mathcal{F}^{-1}\{\mathrm{EP}(\omega)\}\,, \tag{107}$$

donde $\mathrm{F}(\omega)$ es la transformada de Fourier de $f(x)$, $\mathrm{EP}(\omega)$ es el espectro de varianza $|\mathrm{F}(\omega)|$ y $\mathrm{A}(\varrho)$ es el mapa de autocorrelación que se obtiene mediante la transformada inversa de

Fourier de $\mathrm{EP}(\omega)$. La complejidad de esta parte radica principalmente en la transformada de Fourier, la cual conlleva $\mathbb{T}(\mathrm{n}) = \mathrm{O}(\log \mathrm{n})$ pasos con $\mathbb{W}(\mathrm{n}) = \mathrm{O}(\mathrm{n} \log \mathrm{n})$ operaciones (JáJá, 1992). La varianza de la imagen ruidosa se encuentra en el origen del mapa de correlación, aplicando interpolación lineal en los puntos adyacentes se obtiene la varianza de la imagen *ideal*

$$\sigma_{f'}^2 = \mathrm{A}(0)\,, \tag{108}$$

$$\sigma_f^2 = 2\mathrm{A}(1) - \mathrm{A}(2)\,, \tag{109}$$

la desviación estándar del ruido es entonces

$$\sigma_n = \sqrt{\sigma_{f'}^2 - \sigma_f^2}\,, \tag{110}$$

la desviación estándar estimada (σ_n) se toma como parámetro del filtro Gaussiano utilizado para la remoción del ruido en la imagen. El ruido blanco afecta la orientación del gradiente del pixel; con el objetivo de compensar este error, el número de direcciones Q del histograma se calcula en función de la desviación estándar del ruido como

$$Q = \left\lceil \frac{360}{\sigma_n} \right\rceil\,. \tag{111}$$

Las operaciones de multiplicación y filtraje se realizan en tiempo constante $\mathbb{T}(\mathrm{n}) = \mathrm{O}(1)$ pasos con $\mathbb{W}(\mathrm{n}) = \mathrm{O}(\mathrm{n})$ operaciones. Por lo tanto, el tiempo total del algoritmo es $\mathbb{T}(\mathrm{n}) = \mathrm{O}(\log \mathrm{n})$ pasos con $\mathbb{W}(\mathrm{n}) = \mathrm{O}(\mathrm{n} \log \mathrm{n})$ operaciones.

6.3.1.2. Variaciones de iluminación

La variación de iluminación es un problema complicado en procesamiento de imágenes. En general existen dos métodos para la normalización de la iluminación: basados en el modelo de iluminación, dicho modelo es difícil de obtener para aplicaciones en condiciones no controladas; y basado en la transformación de la imagen sin tener información *a priori* del modelo de iluminación, el cual se usa en el preprocesamiento de las imágenes para normalizar la iluminación.

La corrección gamma ha adquirido relevancia en los últimos años debido al uso comercial de las imágenes digitales. La corrección gamma corrige la iluminación general de

una imagen (Goel *et al.*, 2011). Un sensor de imágenes captura el valor de intensidad directamente de un haz de luz, y lo ajusta de forma lineal, el ojo humano tiene la capacidad de compensar la iluminación automáticamente, de tal manera que la luminancia percibida no sea abrumadora. En este trabajo aplicamos la corrección gamma en la imagen para intensificar el contraste en la imagen resultante. Los valores de gamma para corrección de imágenes digitales se encuentran en el rango de $\gamma = \left[\frac{1}{2.2}, \frac{1}{2.6}\right]$ (Po-Ming y Hung-Yi, 2005). Estos valores de gamma son muy costosos de calcular computacionalmente, luego, la corrección gamma puede ser aproximada mediante la raíz cuadrada de cada valor de pixel modificando el rango del pixel de $[0, 255]$ a punto flotante de $[0, 1.0]$ (Dalal y Triggs, 2005). El resultado de la corrección gamma se muestra en la Figura 50, y es calculada mediante

$$I_S = I_E^{\gamma}\,, \tag{112}$$

donde I_S e I_E son las imágenes de salida y de entrada respectivamente. Este méto-

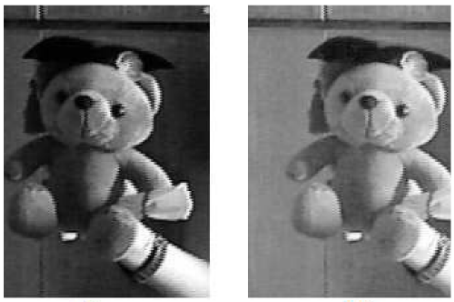

Figura 50: a) Imagen original. b) Imagen con corrección gamma. La imagen corregida muestra un mejor contraste en los bordes del objeto, lo cual mejora la información de gradiente obtenida.

do reduce las variaciones de iluminación y efectos locales de sombreado. Una imagen con iluminación no uniforme tiene iluminación uniforme local sobre áreas pequeñas de la imagen como los fragmentos de imagen a procesar con CWTA o sobre una ventana deslizante. Por lo tanto, el procesamiento de búsqueda en áreas pequeñas de la imagen en lugar de la imagen completa proporciona un beneficio en la compensación de la iluminación. En una GPU la corrección gamma toma $\mathbb{T}(\mathrm{n}) = \mathrm{O}(1)$ pasos con $\mathbb{W}(\mathrm{n}) = \mathrm{O}(\mathrm{n})$ operaciones, ya que se realiza el mismo conjunto de operaciones básicas sobre todos los pixeles concurrentemente.

Para mejorar aún más la robustez a variaciones de iluminación utilizamos imágenes de profundidad adquiridas de sensores Kinect. La información de profundidad es robusta a la

iluminación ambiental, ya que no es adquirida mediante datos RGB como otras técnicas de profundidad como las imágenes estereo, las cuales son sensibles a la iluminación desde su adquisición. Otro beneficio de las imágenes de profundidad es la segmentación de áreas de búsqueda, pues es posible definir esta área de acuerdo a la profundidad del objeto en el cuadro de video anterior, lo cual ayuda a aceptar o descartar fragmentos de imagen antes de que el proceso de correspondencia empiece, lo cual mejora el tiempo de respuesta cuando existe oclusión total del objeto en una imagen.

6.3.2. Detección

Una vez preprocesada la imagen, se procede a localizar el objeto de interés en ésta. Para ello contamos con un conjunto de ventanas con una separación equidistante entre ellas, cada ventana es del mismo tamaño que un disco de la estructura geométrica del modelo del objeto con radio r. La posición y cantidad de estas ventanas depende directamente de las capacidades de la GPU y sus tres niveles de memoria, siendo la más importante la memoria global, donde por cada ventana se guardan un conjunto de variables: estructura geométrica con sus M HOGs, imágenes de gradiente g_x y g_y, imágenes de magnitud y orientación, HOGs de cada ventana de búsqueda, mapas de resultado, entre otros.

En caso de tener una ventana por cada pixel, se pueden calcular todos los HOGs al mismo tiempo, para lo cual se requiere una gran cantidad de memoria global, de lo contrario, es necesario que las ventanas sean deslizantes, calculando en cada iteración el HOG correspondiente. En este trabajo utilizamos una distancia de un pixel entre ventanas, calculando una fila completa de ventanas deslizantes sobre la imagen. Por cada ventana se calcula el histograma de gradiente orientado concurrentemente descrito en la Sección 6.1.

Una vez que se ha calculado el conjunto de histogramas iniciales, el cómputo de los histogramas en las siguientes iteraciones en la posición k de cada ventana deslizant se realiza de manera iterativa pixel a pixel en dirección vertical u horizontal hasta que toda

la imagen haya sido cubierta.

$$HOG_i^k\left(\varphi\left(x,y\right)\right) = \begin{array}{l} HOG_i^{k-1}\left(\varphi\left(x,y\right)\right) - \sum\limits_{(x,y)\in OutP_i^{k-1}} Out\varphi_i^{k-1}\left(x,y\right) + \\ \sum\limits_{(x,y)\in InP_i^k} In\varphi_i^k\left(x,y\right) \end{array}, \tag{113}$$

donde $OutP_i^{k-1}$ es el conjunto de valores de pixel que salen del histograma correspondientes a la mitad del perímetro en la ventana deslizante en la iteración $k-1$, esto es,

$$Out\lfloor\varphi_i^{k-1}\left(x,y\right)\rfloor = \left\{\lfloor MAG_i^{k-1}\left(x,y\right)\rfloor \mid (x,y)\in OutP_i^{k-1}\right\}, \tag{114}$$

$$Out\lceil\varphi_i^{k-1}\left(x,y\right)\rceil = \left\{\lceil MAG_i^{k-1}\left(x,y\right)\rceil \mid (x,y)\in OutP_i^{k-1}\right\}, \tag{115}$$

e InP_i^k el conjunto de valores entrantes al histograma cuyos valores de pixel pertenecen a la mitad del perímetro de la ventana deslizante en la iteración k, dado por

$$In\lfloor\varphi_i^{k}\left(x,y\right)\rfloor = \left\{\lfloor MAG_i^{k}\left(x,y\right)\rfloor \mid (x,y)\in InP_i^{k}\right\}, \tag{116}$$

$$In\lceil\varphi_i^{k}\left(x,y\right)\rceil = \left\{\lceil MAG_i^{k}\left(x,y\right)\rceil \mid (x,y)\in InP_i^{k}\right\}. \tag{117}$$

El cómputo iterativo del histograma permite procesar rápidamente la imagen, dado que solo $2\pi r$ pixeles en la ventana circular son procesados en lugar de los πr^2 que hay en la ventana en cada iteración. El algoritmo de actualización del histograma se ejecuta en tiempo constante $\mathbb{T}(\mathrm{n}) = \mathrm{O}(1)$ con $\mathbb{W}(\mathrm{n}) = (O)(s\pi r)$ operaciones, donde s es la cantidad de histogramas a procesar concurrentemente, ya sea en vertical o en horizontal, como se describe a continuación:

Algoritmo 5 Actualización de histograma con πr procesadores

Entrada: Vectores de perímetro de entrada E y salida S, histograma hog, mapa de orientación φ y magnitud mag
Salida: El histograma hog actualizado

1: $i =$ obtenerProcesador$()$
2: MAG_S = obtener$Out\varphi(i, S(i), mag, \varphi)$
3: $hog(\varphi) -= MAG_S$
4: MAG_E = obtener$In\varphi(i, E(i), mag, \varphi)$
5: $hog(\varphi) += MAG_E$

Originalmente, el histograma en la ventana deslizante se basaba en el conteo de orientaciones y no por voto de magnitud compartida; luego, para obtener un histograma más robusto se filtraba mediante un umbral de mediana aquellos pixeles cuya magnitud no superara la mediana, esto hacía al algoritmo más complejo pues se requería ordenar los datos de magnitud para calcular la mediana y a cada iteración actualizarla para mantener histogramas coherentes a los datos de la posición actual de la ventana. El uso del voto de magnitud compartida proporciona robustez al histograma, pues todos los pixeles dentro del disco proporcionan información, además el cálculo iterativo mejora al eliminar la necesidad del ordenamiento de datos.

La Figura 51 muestra la actualización de los histogramas de gradiente con una ventana circular; si bien el cálculo con una ventana rectangular es más sencillo pues se conoce el número de filas y columnas de la ventana, al calcular la estructura geométrica se generaron vectores con las posiciones del perímetro del círculo, las cuales pueden usarse indiferentemente para deslizamiento vertical u horizontal, reduciendo la complejidad de la búsqueda del perímetro al indexado de los vectores previamente calculados.

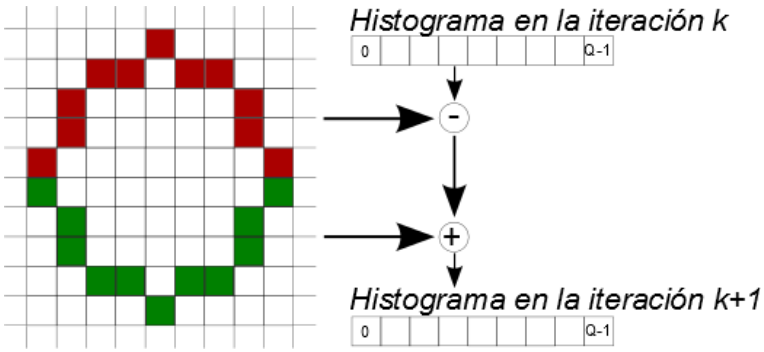

Figura 51: Actualización recursiva del histograma sobre las columnas.

En cada iteración k se procesa un histograma de *escena*, es decir, de la imagen o cuadro de video a procesar. La correspondencia se lleva a cabo al correlacionar los histogramas del i-*ésimo* disco de la estructura geométrica con los histogramas de escena de la iteración k mediante el uso de la transformada de Fourier y el Teorema de Correlación,

$$C_i^{jk}(\alpha) = IFT[HS_{jk}(\omega)HO_i^*(\omega)]\,, \tag{118}$$

donde $HS_k(\omega)$ es la transformada de Fourier del histograma de gradiente orientado centrado y normalizado dentro de la j-*ésima* ventana circular en la k-*ésima* iteración sobre la

imagen correlacionado a $HO_i(\omega)$, que es la transformada de Fourier de la i-*ésima* ventana circular de la estructura del modelo del objeto; el asterisco $(*)$ denota el complejo conjugado. El rango de valores de los mapas de correspondencia se encuentra en $[-1, 1]$. El algoritmo que domina este proceso es la transformada de Fourier de los histogramas, ya que la multiplicación se realiza en tiempo constante con Q procesadores y Q operaciones; luego, el algoritmo se ejecuta en $\mathbb{T}(n) = \mathrm{O}(\log Q)$ pasos con $\mathbb{W}(n) = \mathrm{O}(s^2 Q \log Q)$ operaciones, donde s es la dimensión de un lado de la imagen, es decir, la cantidad de histogramas generados, se supone que la diferencia en las dimensiones de la imagen no es sustancial.

Luego, se crean M mapas de correlación, uno por cada disco de la estructura del objeto, como se muestra en la Figura 52; estos mapas se utilizan para encontrar el mejor punto de correspondencia del objeto en la imagen de búsqueda.

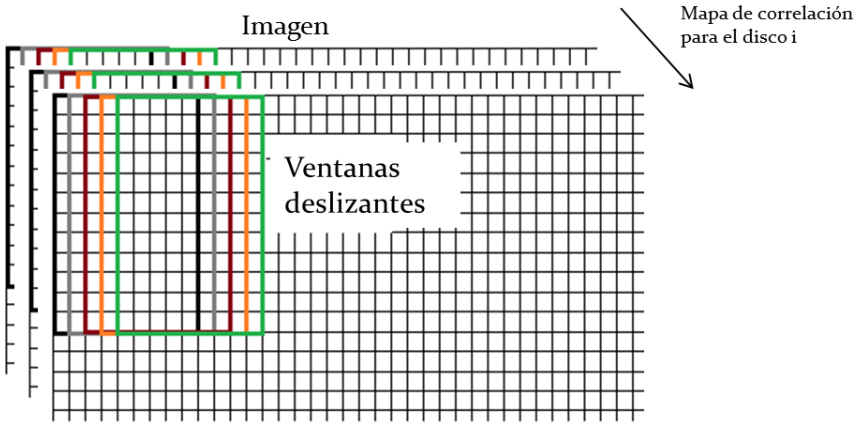

Figura 52: Mapas de correlación generados por la correlación entre las ventanas deslizantes con cada uno de los histogramas en la estructura geométrica.

La zona de correspondencia del objeto se encuentra mediante una búsqueda de los valores máximos de correlación P_i en cada uno de los M mapas de correlación. Se realiza una búsqueda de reducción de espacio de manera iterativa, tomando la información de la estructura geométrica en cada paso para reducir el espacio de búsqueda del siguiente disco. El *pico* de correlación se calcula como

$$P_i = \max_{\alpha} \{C_i(\alpha)\} \,. \tag{119}$$

La Figura 53 muestra un ejemplo del processo de correspondencia de M-pasos descrito a continuación:

1. Buscar la máxima correspondencia del primer disco en la estructura geométrica en el primer mapa de correlación C_1 para obtener el primer pico de correlación P_1; sin embargo, lo que obtenermos es un conjunto de picos de correlación superiores a un umbral Th definido para filtrar los picos de mayor valor. El objetivo es rechazar la mayoría de puntos de correlación del C_1^{jk}. Esta etapa toma $\mathbb{T}(\mathrm{n}) = \mathrm{O}(1)$ pasos con $\mathbb{W}(\mathrm{n}) = \mathrm{O}(\mathrm{n})$ operaciones, pues compara todos los valores en C_1 contra el umbral para determinar el conjunto de datos que se seguirán procesando.

2. En la segunda fase, se buscan los picos de correlación correspondientes al segundo disco en la estructura geométrica en el mapa de correlación C_2; por cada punto aceptado en C_1 se busca en el radii de esta posición en C_2 de acuerdo a la distancia entre los centros de los discos $D_{ij} \pm \epsilon$, aplicando el umbral para mantener los mejores puntos en cada zona que satisfagan el criterio de selección, se rechazan los puntos de la primera fase que no satisfagan la segunda fase. Tanto las distancias como los ángulos en la estructura geométrica no son absolutos, se aplica un poco de flexibilidad ϵ a la zona de búsqueda por cuestiones de ligeras deformaciones geométricas. Esta etapa toma $\mathbb{T}(\mathrm{n}) = \mathrm{O}(1)$ pasos con $\mathbb{W}(\mathrm{n}) = \mathrm{O}(2\pi r)$ operaciones por cada elemento del paso anterior, el cual se supone es un número pequeño, por lo tanto una constante, por lo que se descarta.

3. Para el tercer disco se cuenta tanto con la información correspondiente a la distancia entre centros $D_{ij} \pm \epsilon$, así como los ángulos $\theta_i \pm \epsilon$; dejando solo dos posibles posiciones en las que puede encontrarse el pico máximo de correlación para este conjunto de puntos en C_3. Por supuesto, este punto debe superar también el umbral de selección para ser considerado. Esta etapa y las siguientes toman $\mathbb{T}(\mathrm{n}) = \mathrm{O}(1)$ pasos con $\mathbb{W}(\mathrm{n}) = \mathrm{O}(1)$ operaciones, pues solo deben buscar en un par de pequeñas zonas en un conjunto limitado de elementos.

4. En los mapas de correlación C_4 a C_M existe solo una pequeña región que puede ser evaluada de acuerdo a la estructura geométrica.

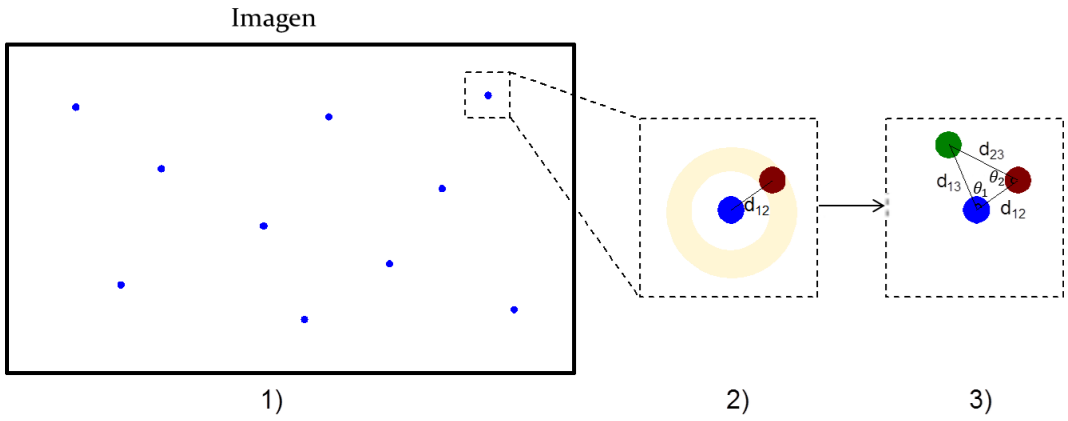

Figura 53: Correspondencia de reducción de espacio para $M = 3$. 1) Buscar los mejores candidatos del primer disco en la estructura geométrica en el primer mapa de correlación. 2) Por cada punto seleccionado, buscar el segundo disco en el radii de éste de acuerdo a la distancia entre centros correspondiente en la estructura geométrica en el segundo mapa de correlación. 3) Usando los ángulos y distancias de la estructura geométrica buscar el tercer disco en las dos posiciones disponibles en el tercer mapa de correlación. La máxima distribución conjunta de los picos de correlación de todas las ventanas indican la presencia del objeto.

Cada fase de la búsqueda de correspondencia se lleva a cabo usando un conjunto de procesadores de la GPU, donde cada uno analiza de manera concurrente un conjunto de picos de correlación y sus alrededores en los M mapas de correlación hasta que todos los histogramas de la estructura geométrica han sido evaluados. La decisión final acerca de la presencia del objeto se toma considerando la distribución conjunta de los picos de correlación de todos los discos en la estructura.

6.3.3. Resultados

En el Capítulo 5 se vieron los criterios de evaluación del algoritmo, en este apartado se verán resultados del algoritmo de correspondencia de acuerdo a los criterios de la Sección 5.1.

El desempeño del algoritmo de correspondencia propuesto (CWMA) se compara contra SIFT (Lowe, 2004), SURF (Bay *et al.*, 2008) y ORB (Rublee *et al.*, 2011). Los parámetros de estos algoritmos son los recomendados por sus respectivos artículos, siendo los que muestran el mejor desempeño. La Tabla 3 muestra la configuración de parámetros utilizada en el algoritmo de correspondencia desarrollado.

Las Figuras 54-58 presentan los resultados de las evaluaciones hechas sobre los algoritmos de correspondencia. Estos resultados fueron obtenidos de ambientes sintéticos y controlados para hacer un cálculo parcial de las capacidades del detector del algoritmo

Tabla 3: Parámetros de CWMA

Parámetro	**Valor**
Ventanas en la estructura	$M = 2$
Radio del disco	$r = 32$ pixeles
Casillas de histograma	$Q = \lceil 360/\sigma_n \rceil$
Umbral de decisión	$Th = 0.8$

de seguimiento. Como puede apreciarse, CWMA presenta resultados competitivos contra el que es considerado el mejor algoritmo de correspondencia, SIFT, en todas las áreas evaluadas. El algoritmo propuesto muestra una mayor estabilidad en rotaciones fuera de plano al no decaer como lo hacen los demás algoritmos, y a diferentes fuentes de iluminación debido a la corrección gamma realizada.

Hace unos cuantos años, SURF era considerado una alternativa rápida de SIFT; sin embargo, ORB muestra superioridad en casi todas las áreas, excepto tolerancia a ruido. CWMA es, sin embargo, una mejor alternativa a estos otros algoritmos. Todos los algoritmos presentan básicamente el mismo desempeño referente a ligeros cambios de escala, ruido y fuentes de iluminación; respecto a rotación, puede apreciarse que los algoritmos generalmente se degradan muy rápido cuando el objeto rota fuera de plano, excepto CWMA que se mantiene estable; mientras que en la rotación en plano, los algoritmos basados en características obtienen buenos resultados cada $90°$, pero se degradan en los ángulos intermedios, lo que no pasa tan notoriamente en CWMA, pues la información de la estructura no se ve afectada por el fondo de la imagen y el contenido de cada disco es constante.

Finalmente, la Figura 59 muestra el tiempo de procesamiento de los algoritmos de correspondencia evaluados. La gráfica muestra el tiempo promedio de detección del objeto de interés en una imagen de 1663×965 pixeles. Como se puede apreciar SURF es el algoritmo más veloz; sin embargo, como se vió anteriormente, es el que presentó el menor desempeño entre los algoritmos evaluados. SIFT aunque fue el mejor algoritmo, su tiempo de procesamiento es extremadamente alto, no es recomendable para aplicaciones en tiempo real. Luego, CWMA y ORB son los que presentan un mejor desempeño con el mejor tiempo de procesamiento, siendo el primero el más balanceado pues compite directamente con SIFT en calidad y con SURF en tiempo de procesamiento.

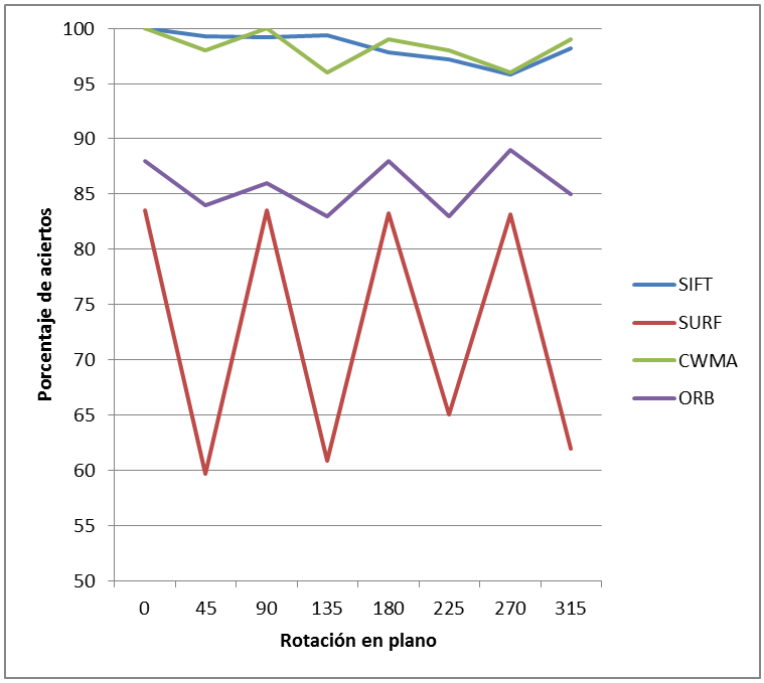

Figura 54: Desempeño de los algoritmos de correspondencia evaluados en rotación en plano.

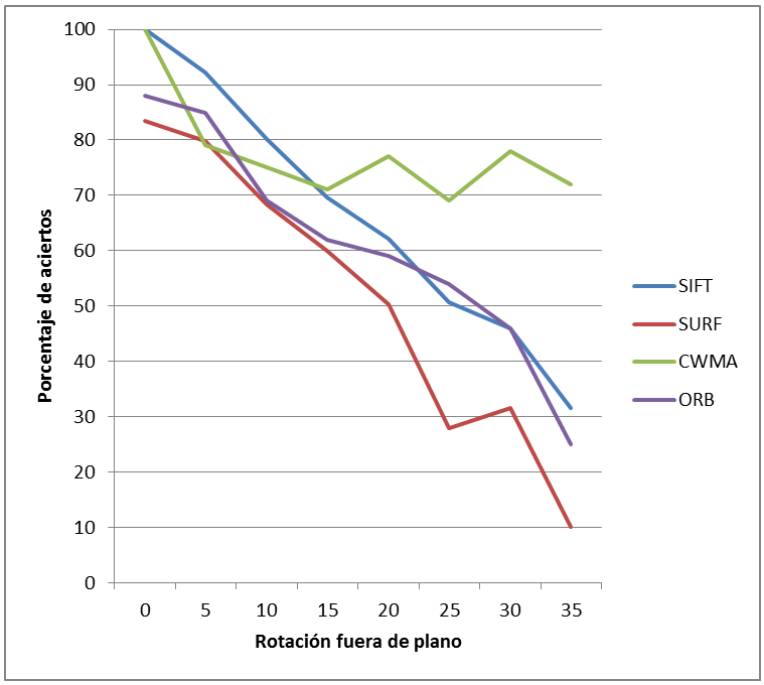

Figura 55: Desempeño de los algoritmos de correspondencia evaluados en rotación fuera de plano, o cambio de perspectiva.

Cabe destacar, que aunque el algoritmo se procesa en 1.23 segundos en estas imágenes, los cuadros de video utilizados en seguimiento son de 640×480 pixeles, de los cuales, en la mayoría de los casos, solo se procesa un fragmento donde se predice la posición del objeto; todas estas condiciones permiten que el algoritmo de seguimiento se ejecute en tiempo real (30 FPS $\sim 0.03\overline{3}$ segundos por cuadro de video).

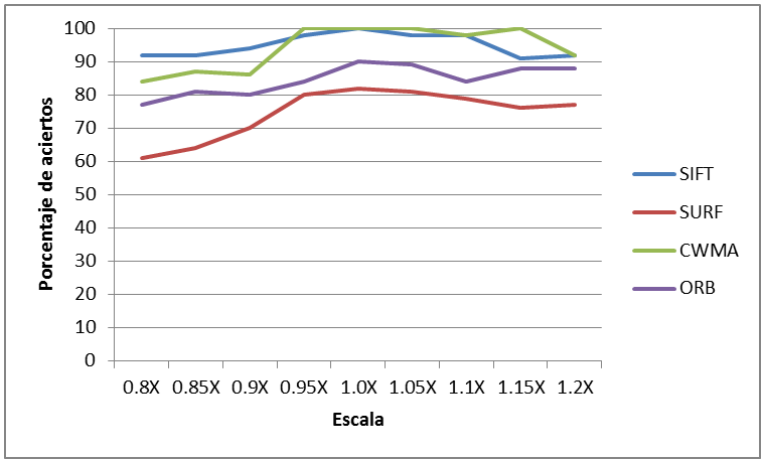

Figura 56: Desempeño de los algoritmos de correspondencia evaluados en ligeros cambios de escala.

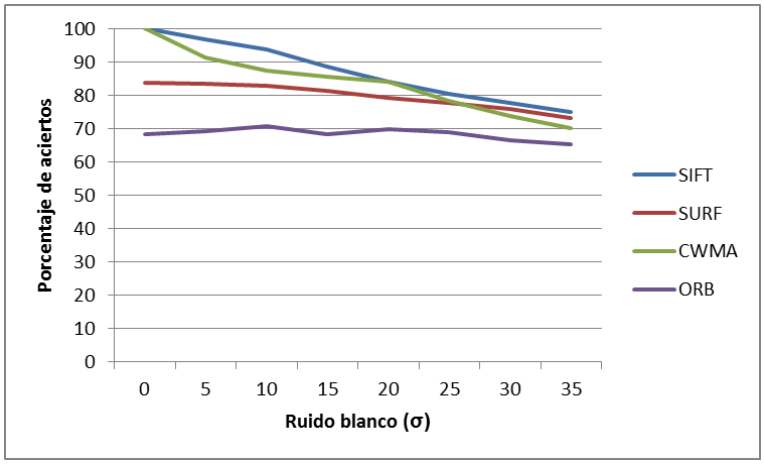

Figura 57: Desempeño de los algoritmos de correspondencia evaluados en tolerancia a ruido blanco.

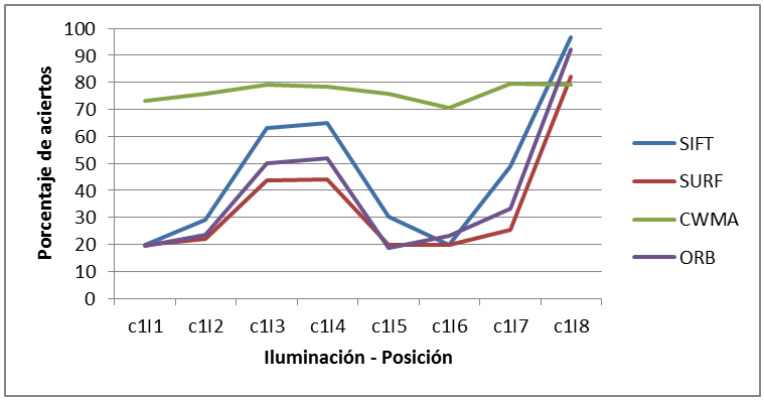

Figura 58: Desempeño de los algoritmos de correspondencia evaluados en tolerancia a iluminación.

6.4. Seguimiento

La última etapa del algoritmo es, una vez ubicado el objeto, seguirlo a través de la secuencia de video; o, en caso de perderlo, ubicarlo de nuevo y continuar con el procesa-

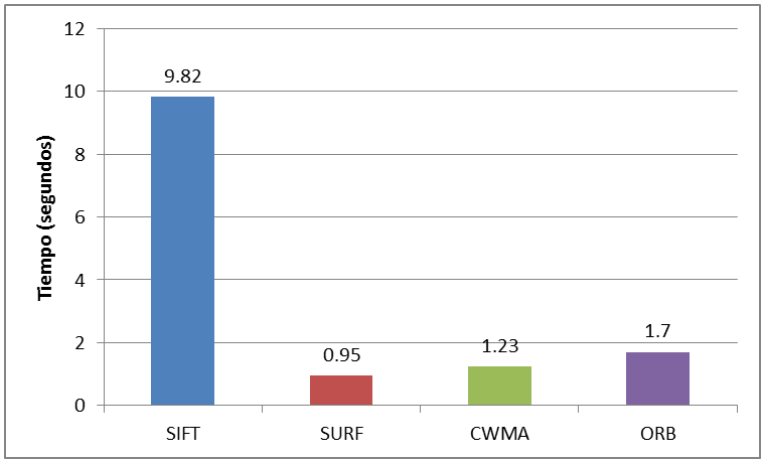

Figura 59: Tiempo de procesamiento de los algoritmos de correspondencia evaluados en imágenes de 1663×965 pixeles.

miento en el siguiente cuadro. Existen dos formas de iniciar un algoritmo de seguimiento, sin y con inicialización. La primera es cuando se desconoce la posición del objeto en el primer cuadro y se tiene que buscar en toda la imagen; la segunda es cuando se tiene conocimiento de la posición del objeto en el primer cuadro y se continúa el seguimiento en el siguiente cuadro. La segunda forma es la más utilizada en el área de seguimiento, en este trabajo se conoce la posición del objeto en el primer cuadro de video.

El proceso de seguimiento es relativamente sencillo: localizar al objeto mediante correspondencia, estimar la posición del objeto en el siguiente cuadro, repetir hasta terminar con la secuencia de video. En esta etapa se realizan dos procedimientos para seguir al objeto. El primero es estimar la posición del objeto mediante un modelo de movimiento, teniendo en cuenta la posición del objeto en cuadros anteriores y generar un fragmento de búsqueda dentro del cuadro; una vez obtenida la posición estimada se utiliza el mapa de profundidad del cuadro actual para ajustar el fragmento de acuerdo a la profundidad actual del objeto, la cual varía ligeramente cuadro por cuadro.

El primer proceso se realiza de forma serial en el CPU, pues son pocas operaciones las que se deben realizar, sería un desperdicio el uso del GPU; mientras que el segundo utiliza el GPU para procesar y comparar todos los pixeles del mapa de profundidad concurrentemente. Es importante destacar el tamaño que toma el fragmento del cuadro de video. El fragmento es 1.5 veces el tamaño de la caja delimitadora que circunscribe a la estructura del objeto; el fragmento crece un $10\,\%$ del tamaño del radio del disco por cada cuadro consecutivo donde no se haya encontrado el objeto por un periodo de 10

cuadros; si el objeto no fue encontrado para entonces, se realiza una búsqueda global en la imagen hasta recuperar al objeto.

6.4.1. Modelo de movimiento

Una vez localizado el objeto en un cuadro k, el algoritmo de correspondencia devuelve la posición de cada uno de los discos encontrados. La predicción de la localización del objeto para los cuadros subsecuentes se basa en series de tiempo al ajustar la dirección del movimiento de cada disco encontrado en x y y a una curva polinomial (Gupta, 2005). Sea el movimiento del objeto representado por la siguiente ecuación:

$$x = a + b\mathrm{k} + c\mathrm{k}^2\,, \tag{120}$$

donde x es la posición del objeto en dirección horizontal, se realiza el mismo procedimiento para la dirección vertical y; k es el cuadro de video donde se predice el movimiento; a, b, c son parámetros de segundo orden de la función de la parábola. El objetivo es ajustar el movimiento del objeto a la función de una parábola usando la regresión cuadrática de mínimos cuadrados (Apéndice A), obteniendo el siguiente sistema de ecuaciones

$$\sum x = ma + b\sum \mathrm{k} + c\sum \mathrm{k}^2\,, \tag{121}$$

$$\sum x\mathrm{k} = a\sum \mathrm{k} + b\sum \mathrm{k}^2 + c\sum \mathrm{k}^3\,, \tag{122}$$

$$\sum x\mathrm{k}^2 = a\sum \mathrm{k}^2 + b\sum \mathrm{k}^3 + c\sum \mathrm{k}^4\,, \tag{123}$$

donde m es la cantidad de cuadros de video que se toman para realizar la predicción. Debe definirse una cantidad fija de cuadros de video previos al actual para predecir la nueva posición, es preferible que esta cantidad sea un número impar, pues simplifica el sistema de ecuaciones haciendo los términos $\sum \mathrm{k}$ y $\sum \mathrm{k}^3$ igual a 0. Gupta (2005) menciona que $m = 5$ es suficiente para realizar una buena predicción, con $\mathrm{k} = [-2, 2]$, tomando como el cuadro a predecir como $k = 3$ en cada iteración. Luego, el sistema de ecuaciones para calcular los coeficientes a, b y c queda dado por

$$a = \frac{\sum x - c\sum \mathrm{k}^2}{m}\,, \tag{124}$$

$$b = \frac{\sum x\mathrm{k}}{\sum \mathrm{k}^2}\,, \tag{125}$$

$$c = \frac{m\sum x\mathrm{k}^2 - \sum \mathrm{k}^2 \sum x}{m\sum \mathrm{k}^4 - \left(\sum \mathrm{k}^2\right)^2}\,. \tag{126}$$

Se resuelve el sistema de ecuaciones para cada ventana circular M de la estructura del objeto en las direcciones x y y. Una vez calculada la nueva posición de cada disco en la estructura geométrica, se genera una caja delimitadora que la circunscriba, agregando a cada lado el tamaño del radio para compensar por cambios de dirección y velocidad que sufra el objeto. Dado el sistema de ecuaciones, el cálculo de de los coeficientes a, b y c toma $\mathbb{T}(\mathrm{n}) = \mathrm{O}(1)$ pasos, dado que los factores $\sum \mathrm{k}^2$, $\sum \mathrm{k}^4$ y $\left(\sum \mathrm{k}^2\right)^2$ son valores constantes, los demás términos se calculan con una cantidad constante de pasos (5); y aunque se realiza $2M$ veces, M es una cantidad pequeña y constante, por lo que el tiempo total de cómputo de la predicción es constante.

6.4.2. Mapa de profundidad

Una vez establecido el fragmento del cuadro de video estimado por el modelo de movimiento, se revisa el mapa de profundidad del nuevo cuadro de video con dos objetivos: primero, establecer si el objeto esta siendo ocludido parcial o totalmente; y segundo, en caso de no ser ocludido o tener oclusión parcial, ajustar la posición del fragmento del cuadro de video de tal manera que éste contenga toda el área visible del objeto.

En el primer cuadro de la secuencia se genera un histograma de profundidad, el cual contiene la distribución de profundidad dentro de los discos del modelo del objeto como se muestra en la Figura 60a, se guarda información del histograma como la posición del pico, media y desviación estandar. Esta información se actualiza cuadro a cuadro, pues la posición del pico del histograma varía cuando el objeto se acerca y se aleja de la cámara, y la densidad de pixeles baja cuando el objeto es ocludido.

Al momento que el objeto entra a estado de oclusión, la información de profundidad del oclusor aparecerá antes que la información del objeto, pues el oclusor se encuentra más cerca del sensor (Song y Xiao, 2013), como se muestra en la Figura 60b; al mismo

tiempo, la información del objeto disminuye, pues hay una menor cantidad de pixeles con la profundidad del objeto. En caso de que el pico del objeto baje de un umbral Th_{Occ} se considera que el objeto está completamente ocludido y se guarda toda su información para el proceso de recuperación.

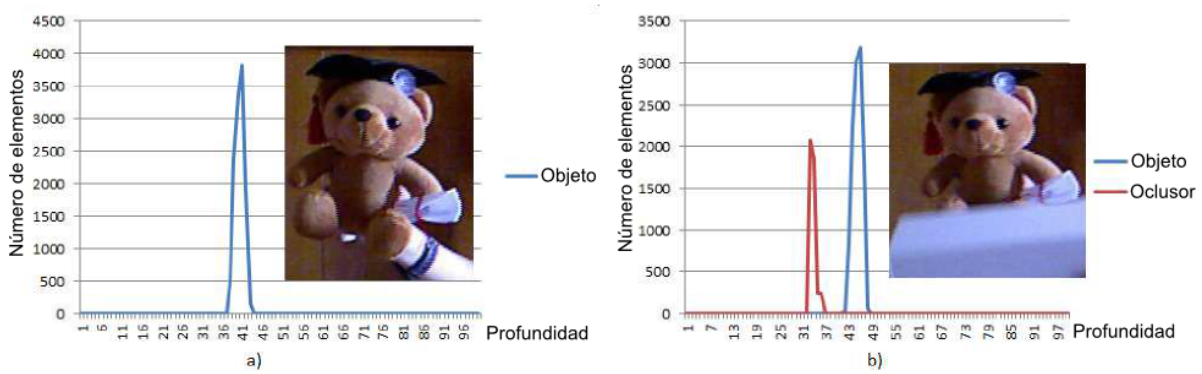

Figura 60: a) Histograma de profundidad del objeto. b) Histograma de profundidad del objeto y el oclusor. La distribución del oclusor aparece antes que la del objeto pues se encuentra más cercano a la cámara.

En caso de que el objeto se encuentre completamente ocludido, se busca en el mapa de profundidad, alrededor del oclusor, información de profundidad del objeto; de no encontrarse información o si el pico del objeto no supera el umbral, no se ejecuta el módulo de correspondencia y se analiza el siguiente cuadro de video, aumentando el tamaño del fragmento por $0.1r$ hasta un tamaño máximo del fragmento de $2r$; luego, se busca en el cuadro de video completo. En caso de que la información del objeto supere el umbral, se actualiza la información de profundidad del objeto y se vuelve a ejecutar el módulo de correspondencia.

El algoritmo requiere como mínimo encontrar uno de los discos de la estructura, calculando la posición del resto de los círculos o un área donde pudieran encontrarse el resto de los círculos para generar el siguiente fragmento a procesar, que sea lo suficientemente grande para contener al objeto completo. El proceso continua hasta que todos los cuadros de la secuencia han sido analizados. El resultado del algoritmo es la colección de posiciones de la estructura del objeto, las cuales muestran la trayectoria de éste en la secuencia de video.

El algoritmo de oclusión y ajuste del fragmento tiene dos variantes: si había oclusión o no en el cuadro anterior. En caso de no haber oclusión se realiza una búsqueda en los

límites del objeto en el fragmento; primero se busca en el perímetro del fragmento; este paso es trivial, toma $\mathbb{T}(\mathrm{n}) = \mathrm{O}(1)$ pasos con $\mathbb{W}(\mathrm{n}) = \mathrm{O}(4s)$ operaciones, donde s es el lado del fragmento.

En caso de no contener información del objeto no se hacen modificaciones a ese lado del fragmento, pues el objeto se encuentra dentro del límite, en caso de haber información del objeto, se busca la profundidad del objeto desde ese punto del perímetro al límite de la imagen, obteniendo la máxima (o mínima) posición de pixel en horizontal o vertical donde se encontró información de profundidad del objeto, ajustando el tamaño del fragmento con esta nueva posición. Todas las operaciones toman tiempo constante con n operaciones,

Algoritmo 6 Ajuste de fragmento con n procesadores

Entrada: Un sector de búsqueda de la imagen E, información de profundidad del objeto d
Salida: Coordenadas ajustadas del fragmento de búsqueda
1: $i =$ obtenerProcesador()
2: **si** $E(i) = d.profundidad()$ **entonces**
3: (x, y) = obtenerminMax$(E(i).x, E(i).y)$
4: **fin si**
5: ajustarFragmento(x, y)

excepto el máximo/mínimo que es una operación de reducción que toma la misma forma que el Algoritmo de Suma (2); por lo tanto, el ajuste del fragmento toma $\mathbb{T}(\mathrm{n}) = \mathrm{O}(\log \mathrm{n})$ pasos con $\mathbb{W}(\mathrm{n}) = \mathrm{O}(\mathrm{n})$ operaciones. En caso de haber oclusión, el algoritmo es similar, solo que el límite se extiende $2r$ del límite del oclusor, pues el objeto aparecerá poco a poco a partir de los límites del oclusor.

6.4.3. Resultados

En esta sección se presentan los resultados de la evaluación a los algoritmos de seguimiento. El estudio realizado por Song y Xiao (2013) presenta el conjunto de referencia utilizado en nuestros experimentos; así como resultados de un conjunto de algoritmos de seguimiento. Se seleccionaron los algoritmos con el mejor desempeño dentro de las dos categorías que considera el estudio: aquellos que no usan información de profundidad, OMIL (Babenko *et al.*, 2009) y TLD (Kalal *et al.*, 2012); y aquellos que hacen uso de la información de profundidad, RGBDOcc+OF (Song y Xiao, 2013), OAPF (Meshgi *et al.*, 2014) y el algoritmo propuesto CWTA.

Este estudio proporciona el resultado general de los algoritmos en cuanto al área de intersección promedio de las 5 secuencias etiquetadas, así como los errores cometidos; a estos resultados se añade una comparación del área bajo la curva y el valor medio de intersección con intervalos de confianza. Finalmente, se presentan estos mismos resultados secuencia por secuencia solo de los algoritmos que utilizan información de profundidad; estos resultados son proporcionados por Meshgi *et al.* (2014), el cual no toma en cuenta para su comparación a OMIL o a TLD.

La Figura 61 muestra como los algoritmos que utilizan información de profundidad se desempeñan sustancialmente mejor que aquellos que no tienen esta característica, esto se debe principalmente a su discapacidad de retener al objeto cuando existe oclusión y/o recuperarlo en caso de oclusión. Es interesante ver como los algoritmos que manejan información de profundidad tienen un desempeño similar; sin embargo, llegan a un punto, alrededor del $70\,\%$ de empalme, en el cual OAPF y RGBDOcc+OF comienzan a decaer rápidamente, mientras que CWTA tiene una disminución de empalme más gradual y estable.

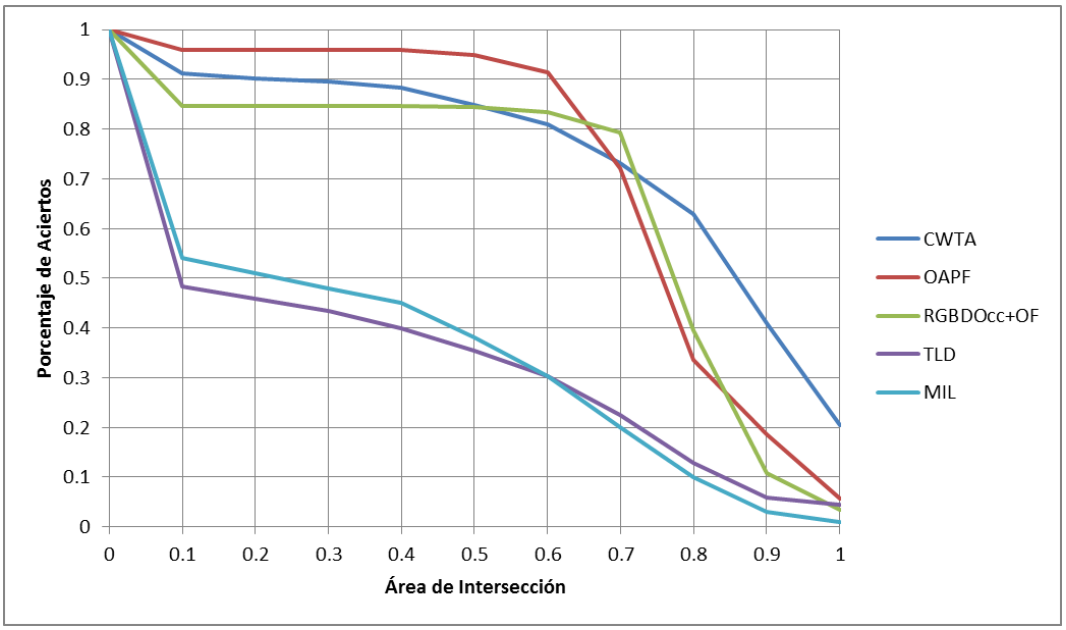

Figura 61: Comparación del área de intersección promedio de los algoritmos.

La Figura 62 muestra el área bajo la curva de intersección, el cual define la calidad del algoritmo. Esta medida se toma desde un umbral del $50\,\%$ de empalme. Nuevamente, puede apreciarse como los algoritmos que utilizan información de profundidad superan a

los que no. En esta gráfica puede verse que el algoritmo propuesto supera por aproximadamente 0.5 a OAPF; sin embargo, como se puede apreciar en la Figura 63, donde se muestra el valor medio de empalme con sus intervalos de confianza al $95\,\%$ (ver Tabla 4). Los algoritmos CWTA y OAPF tienen un desempeño prácticamente igual, por lo que no puede definirse un mejor algoritmo; luego se dice que ambos algoritmos son competitivos.

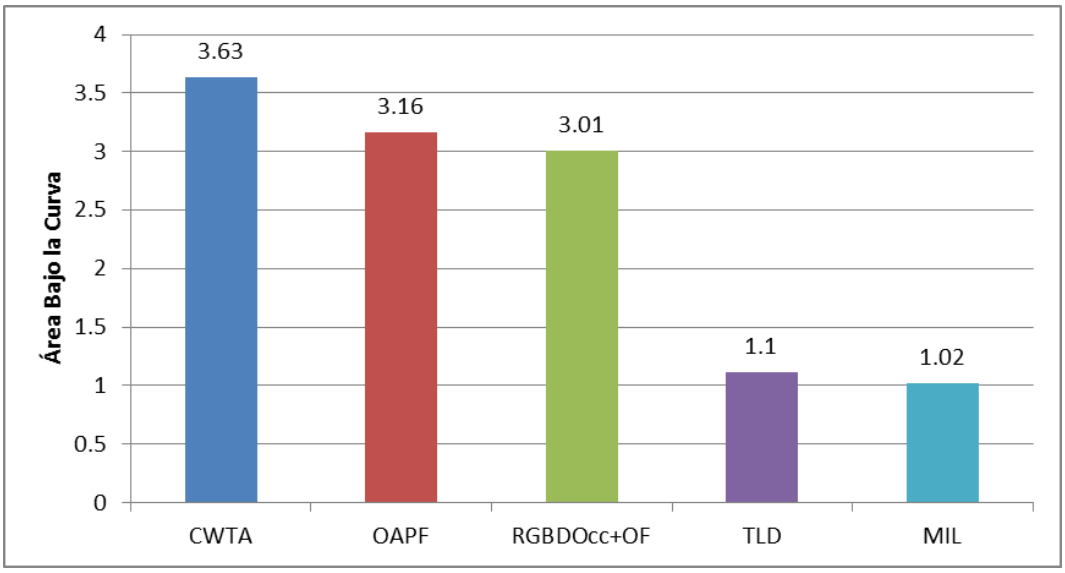

Figura 62: Comparación del área bajo la curva sobre un umbral del $50\,\%$.

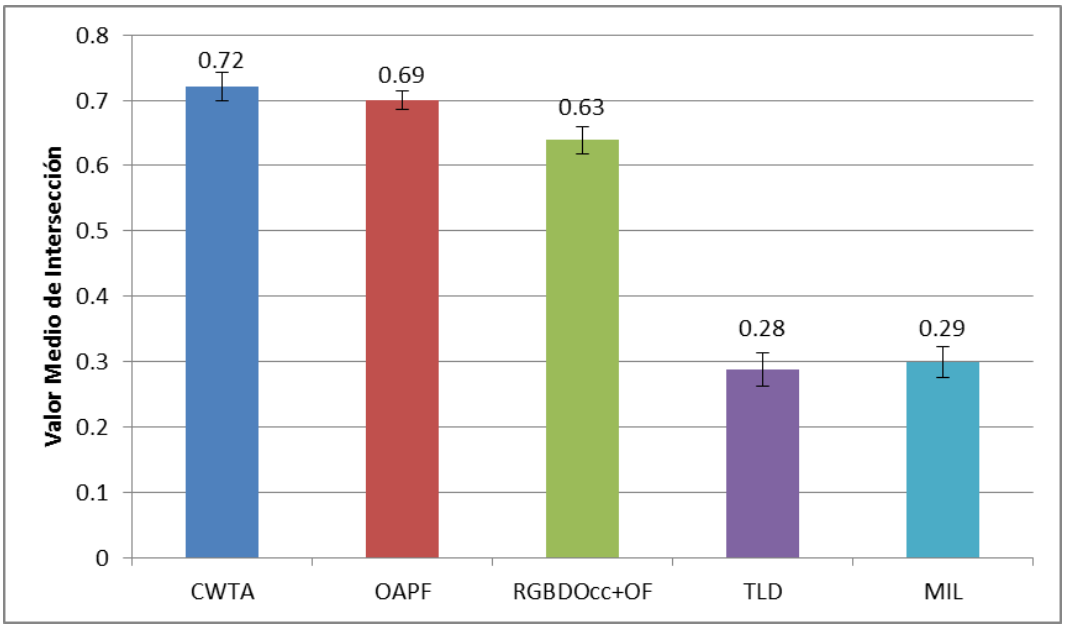

Figura 63: Intervalos de confianza ($95\,\%$) del valor medio de intersección.

Tabla 4: Intervalos de confianza ($95\,\%$) del valor medio de intersección

Algoritmo	Media	Intervalo de Confianza ($95\,\%$)
CWTA	0.72	$[0.69, 0.74]$
OAPF	0.69	$[0.68, 0.71]$
RGBOcc+OF	0.63	$[0.61, 0.66]$
TLD	0.28	$[0.26, 0.31]$
OMIL	0.29	$[0.27, 0.32]$

Finalmente, la Figura 64 muestra los errores cometidos por los algoritmos, como se describe en la Sección 5.2.2. Como puede apreciarse, CWTA comete la menor cantidad de errores de entre los algoritmos evaluados. El único problema encontrado en el algoritmo propuesto fue en la secuencia *face_occ5*, en la cual se cometen varios errores Tipo I, dado que la profundidad de la cara de la persona tiene la misma profundidad que el rostro (Figura 65), luego, al ocurrir la oclusión el algoritmo encuentra un histograma parecido en la blusa y sigue sobre esta posición durante varios cuadros. Por otra parte, la cantidad de errores cometidos es mínima.

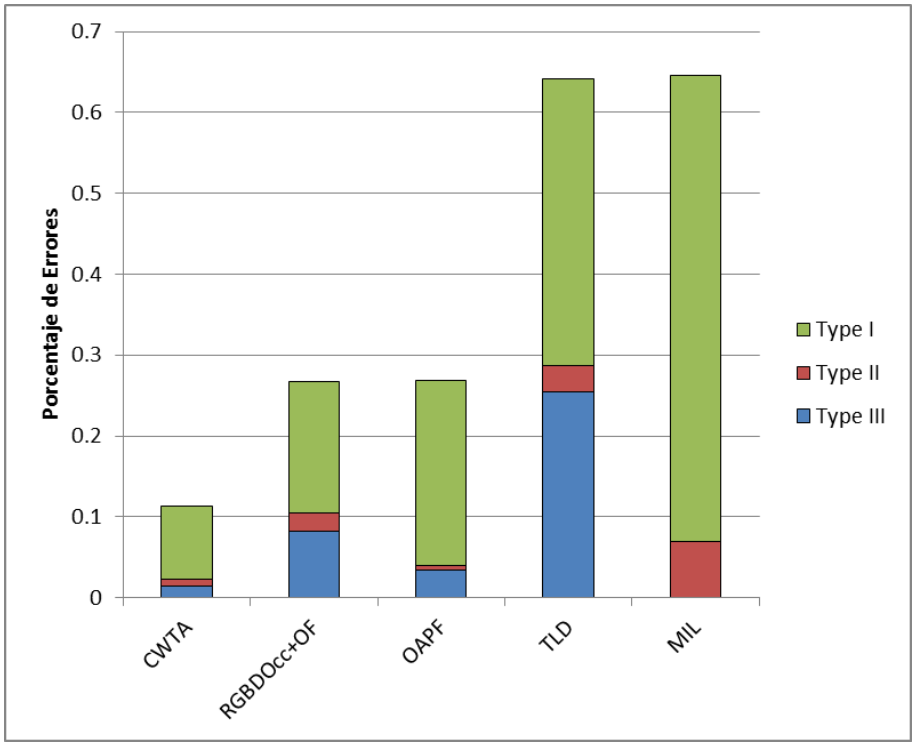

Figura 64: Error promedio.

A continuación se presenta un desgloce de las secuencias de video y sus resultados de manera individual, solo con los algoritmos CWTA, RGBDOcc+OF (Song y Xiao, 2013) y OAPF (Meshgi *et al.*, 2014).

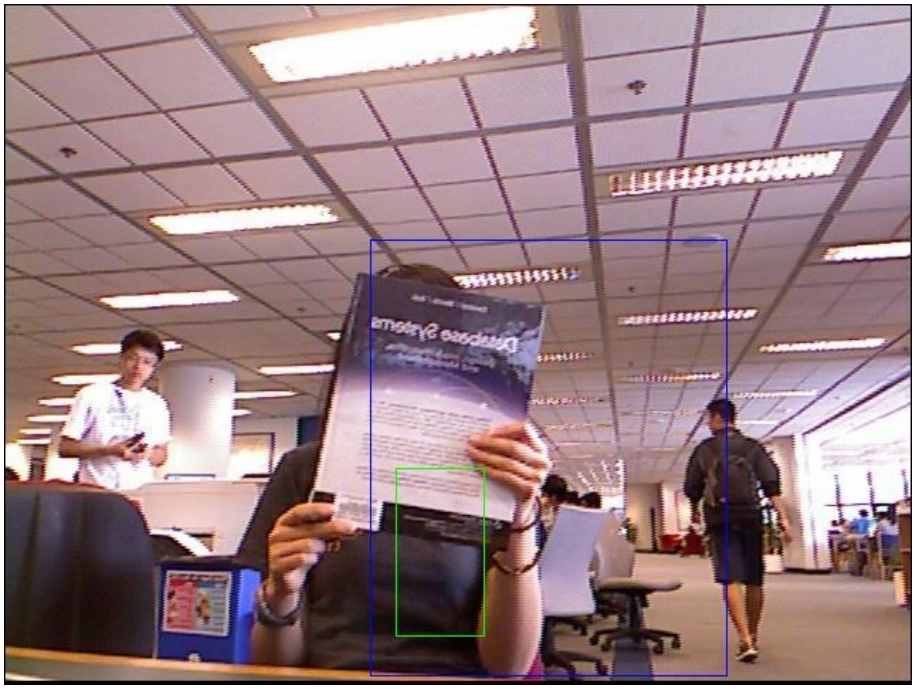

Figura 65: Ejemplo de Error Tipo I en la secuencia *face_occ5*.

Las Figuras 66-70 muestran el análisis del área de intersección de las secuencias de video de la Sección 5.2.2. Como se muestra en las figuras, los tres algoritmos tienen un desempeño similar; sin embargo, en la mayoría de los casos el algoritmo propuesto tiene la mayor cantidad de cuadros con mayor intersección. Las Figuras 66 y 70 muestran el mejor desempeño de CWTA sobre el umbral del $50\,\%$, mientras que el resto de las secuencias es similar en los tres algoritmos.

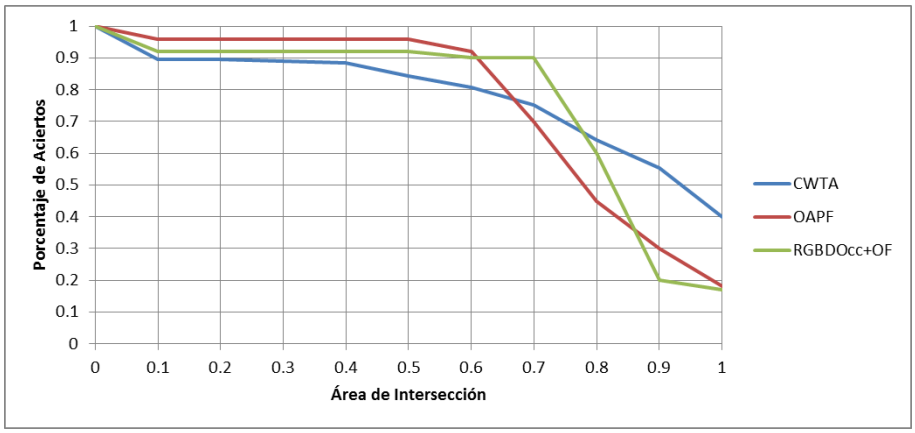

Figura 66: Comparación del área de intersección de la secuencia *bear_front*.

Las Figuras 71-75 muestran el área bajo de curva de cada secuencia, donde se ve que el algoritmo propuesto tiene los mejores resultados de entre los algoritmos analizados; sin embargo, la diferencia no es significativa para determinar al mejor algoritmo, hasta que

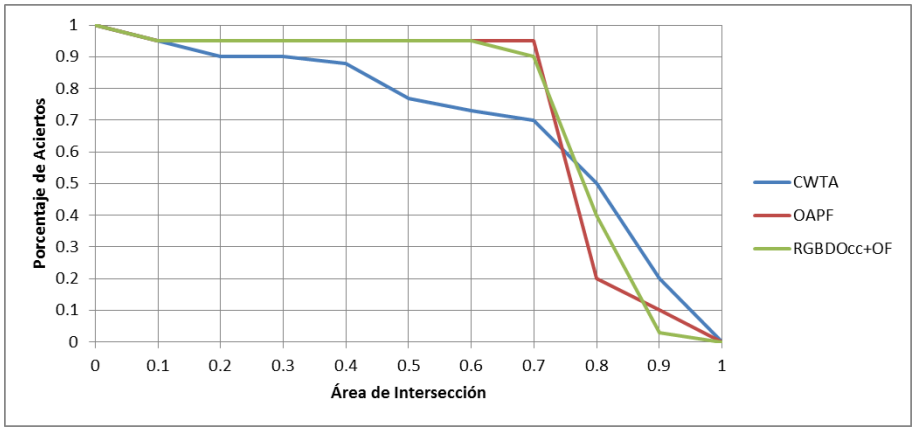

Figura 67: Comparación del área de intersección de la secuencia *child_no1*.

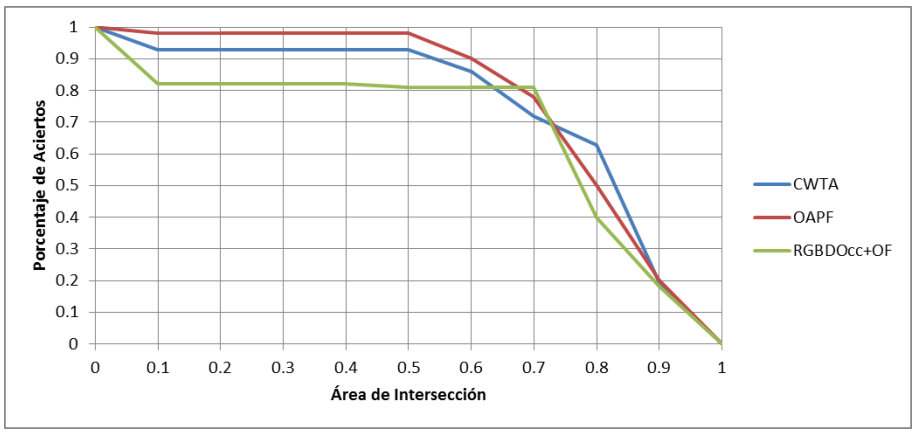

Figura 68: Comparación del área de intersección de la secuencia *face_occ5*.

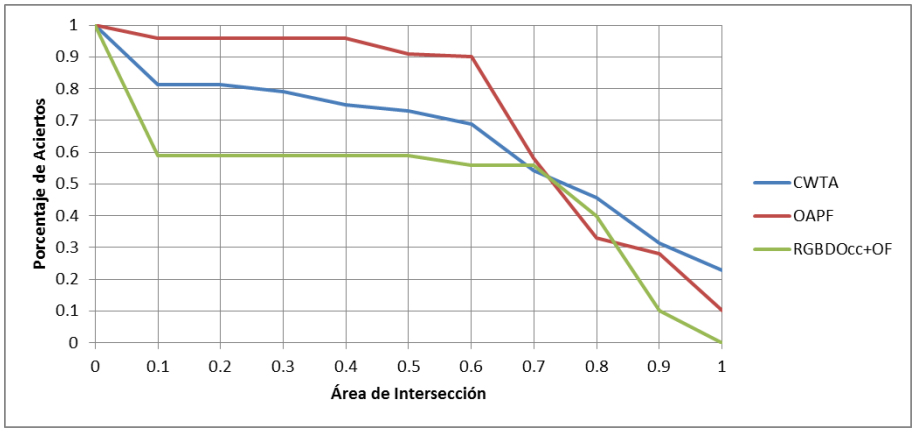

Figura 69: Comparación del área de intersección de la secuencia *new_ex_occ4*.

se realiza el análisis conjunto como al inicio de este capítulo.

Finalmente, las Figuras 76-80 muestran el valor medio de intersección y sus intervalos de confianza al $95\,\%$ (ver Tabla 5). Donde se muestra que los tres algoritmos tienen generalmente el mismo desempeño.

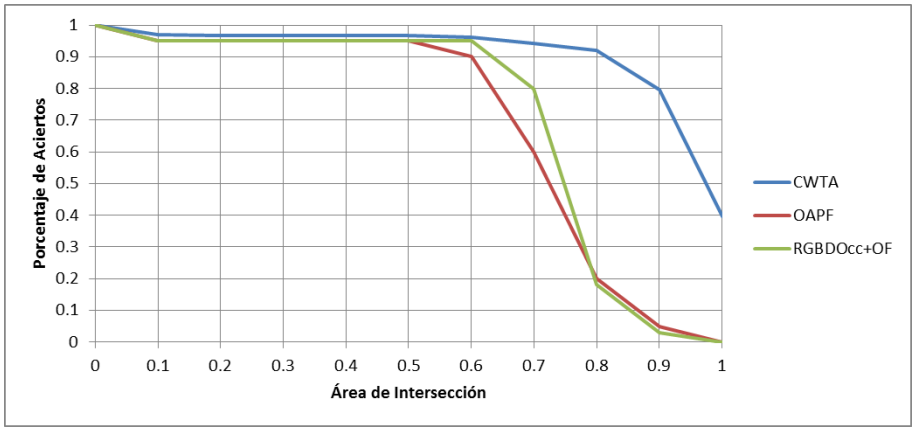

Figura 70: Comparación del área de intersección de la secuencia *zcup_move_1*.

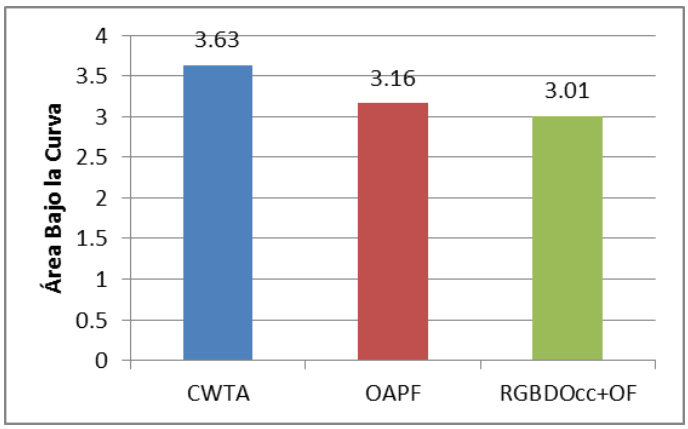

Figura 71: Comparación del área bajo la curva sobre un umbral del $50\,\%$ de la secuencia *bear_front*.

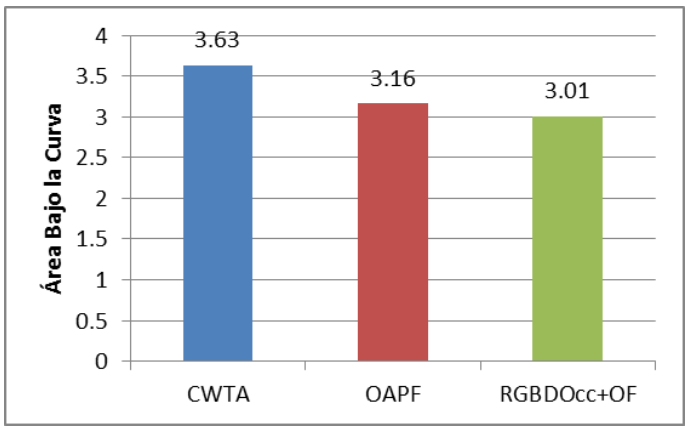

Figura 72: Comparación del área bajo la curva sobre un umbral del $50\,\%$ de la secuencia *child_no1*.

El algoritmo propuesto muestra el mejor desempeño en general, como se vio al inicio de este capítulo; siendo también el más sencillo, contando con un solo descriptor y técnicas sencillas de evaluación y seguimiento; pero es lo suficientemente robusto contra ruido y variaciones ligeras a iluminación y cambios de perspectiva o rotaciones fuera de plano y a ligeros cambios de escala, y completamente invariante a rotación en plano. Además, el

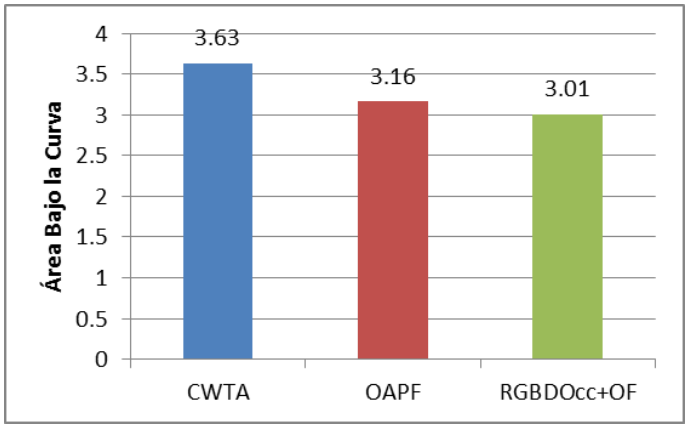

Figura 73: Comparación del área bajo la curva sobre un umbral del $50\,\%$ de la secuencia *face_occ5*.

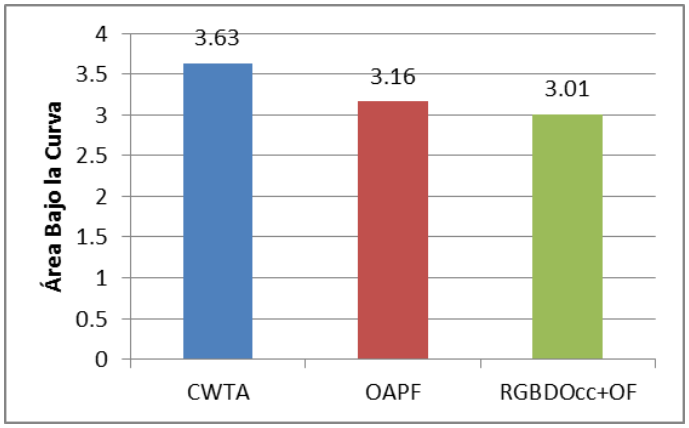

Figura 74: Comparación del área bajo la curva sobre un umbral del $50\,\%$ de la secuencia *new_ex_occ4*.

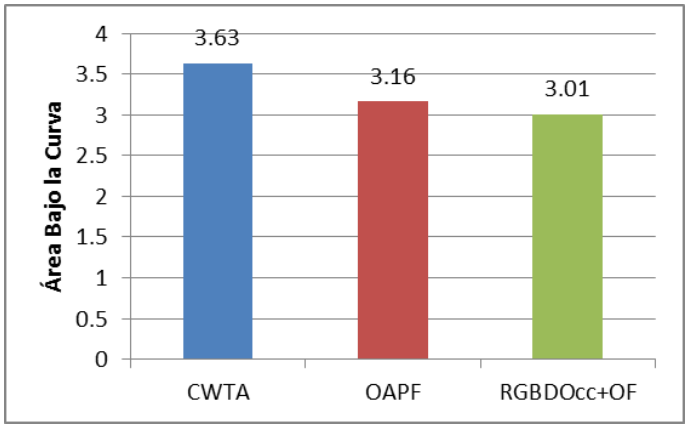

Figura 75: Comparación del área bajo la curva sobre un umbral del $50\,\%$ de la secuencia *zcup_move_1*.

procesamiento paralelo permite la ejecución del algoritmo en un rango de 28 a 32 FPS en una GPU de gama baja, pudiendo alcanzar un mejor desempeño con una GPU de gama alta.

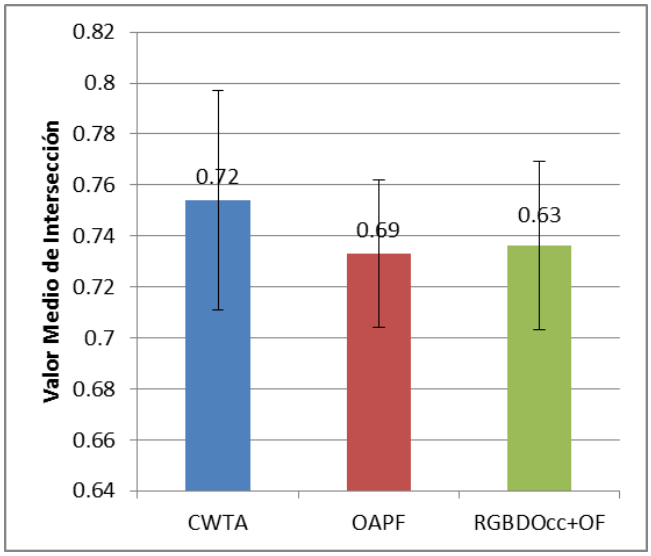

Figura 76: Intervalos de confianza ($95\,\%$) del valor medio de intersección de la secuencia *bear_front*.

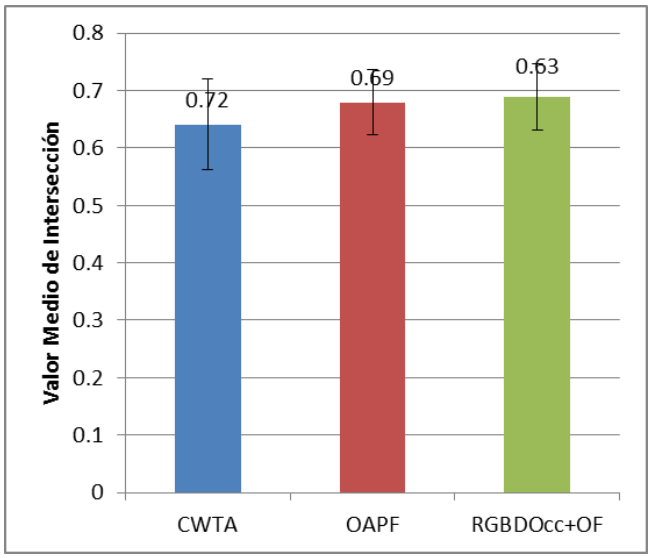

Figura 77: Intervalos de confianza ($95\,\%$) del valor medio de intersección de la secuencia *child_no1*.

Tabla 5: Intervalos de confianza ($95\,\%$) del valor medio de intersección de las cinco secuancias de video

	CWTA		OAPF		RGBDOcc+OF	
Secuencia	**Media**	**IC ($95\,\%$)**	**Media**	**IC ($95\,\%$)**	**Media**	**IC ($95\,\%$)**
bear_front	0.75	$[0.71, 0.79]$	0.73	$[0.70, 0.76]$	0.73	$[0.70, 0.76]$
child_no1	0.64	$[0.56, 0.72]$	0.67	$[0.62, 0.73]$	0.68	$[0.63, 0.74]$
face_occ5	0.70	$[0.66, 0.73]$	0.72	$[0.69, 0.75]$	0.62	$[0.57, 0.67]$
new_ex_occ4	0.61	$[0.51, 0.71]$	0.68	$[0.62, 0.74]$	0.44	$[0.33, 0.55]$
zcup_move_1	0.88	$[0.85, 0.90]$	0.64	$[0.62, 0.67]$	0.69	$[0.66, 0.72]$

La Tabla 6 muestra un resumen de la complejidad teórica de los módulos del algoritmo presentado. Como puede verse, el algoritmo propuesto tiene un tiempo de ejecución teórico de $\mathbb{T}(\mathrm{n}) = \mathrm{O}(log\mathrm{n})$ pasos con $\mathbb{W}(\mathrm{n}) = \mathrm{O}(\mathrm{n}\log\mathrm{n})$ operaciones.

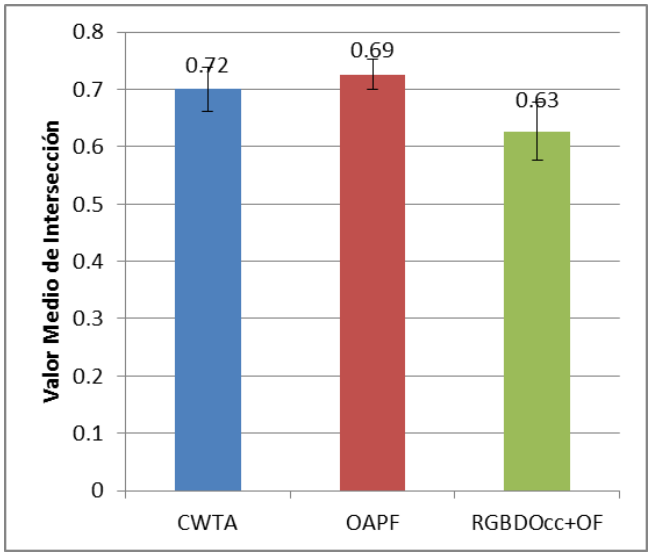

Figura 78: Intervalos de confianza ($95\,\%$) del valor medio de intersección de la secuencia *face_occ5*.

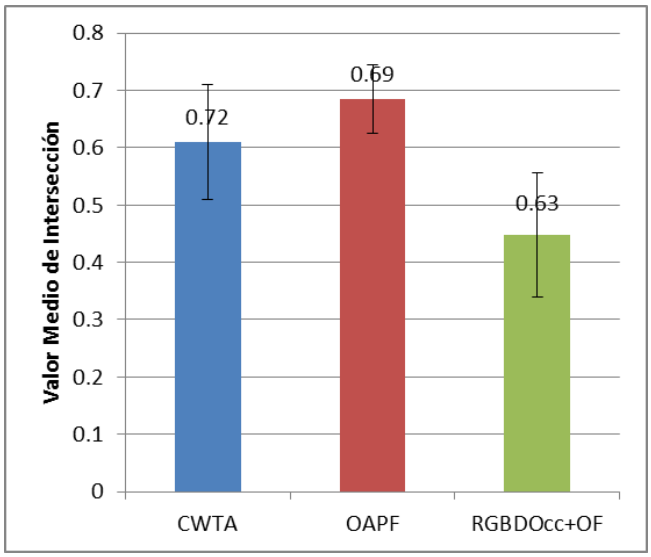

Figura 79: Intervalos de confianza ($95\,\%$) del valor medio de intersección de la secuencia *new_ex_occ4*.

Tabla 6: Complejidad teórica del algoritmo

Algoritmo	$\mathbb{T}(\mathrm{n})$	$\mathbb{W}(\mathrm{n})$
Gradiente	$\mathrm{O}(1)$	$\mathrm{O}(\mathrm{n})$
Histograma	$\mathrm{O}(\mathrm{n}/\log \mathrm{n})$	$\mathrm{O}(\mathrm{n})$ ó $\mathrm{O}(Q \log \mathrm{n})$
Cómputo del disco	$\mathrm{O}(1)$	$\mathrm{O}(\mathrm{n})$
Cálculo de la estructura	$\mathrm{O}(1)$	$\mathrm{O}(M)$
Ruido	$\mathrm{O}(\log \mathrm{n})$	$\mathrm{O}(n \log \mathrm{n})$
Iluminación	$\mathrm{O}(1)$	$\mathrm{O}(\mathrm{n})$
Actualización de histograma	$\mathrm{O}(1)$	$\mathrm{O}(s\pi r)$
Mapa de correlación	$\mathrm{O}(\log Q)$	$\mathrm{O}(sQ \log Q)$
Correspondencia	$\mathrm{O}(1)$	$\mathrm{O}(\mathrm{n})$
Predicción	$\mathrm{O}(1)$	
Ajuste del fragmento	$\mathrm{O}(\log \mathrm{n})$	$\mathrm{O}(\mathrm{n})$

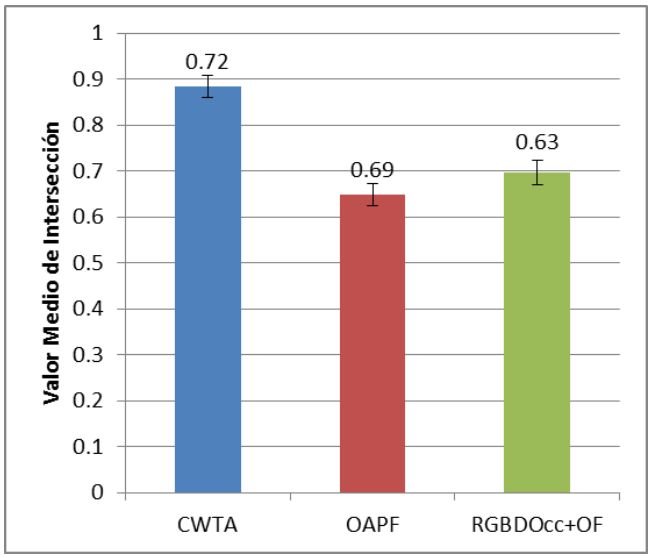

Figura 80: Intervalos de confianza (95 %) del valor medio de intersección de la secuencia *zcup_move_1*.

Capítulo 7. Conclusiones

La motivación del desarrollo de la ciencia y tecnología en la actualidad se ve influenciada por el mejoramiento de la calidad de vida de los seres vivos; así pues, el avance de la ciencia del procesamiento digital de imágenes para visión nos permite obtener herramientas que nos permitan *ver más allá* de lo que podemos ver sin ninguna influencia tecnológica.

En el presente trabajo de investigación se desarrolló un algoritmo de seguimiento de objetos basado en correspondencia. Dicho algoritmo contiene tres módulos principales: el modelo del objeto, el detector basado en correspondencia y el seguidor basado en un modelo de series de tiempo y rectificación por mapas de profundidad.

Para el modelo del objeto se eligió una estructura geométrica de múltiples ventanas en forma de discos dentro del área de soporte del objeto, con distancias y ángulos entre discos conocidos. La elección de la forma de la ventana es importante, pues es la información que será analizada por el algoritmo, y una ventana circular mantiene la información invariante a rotaciones en plano. Dentro de cada disco se decidió extraer histogramas de gradiente orientado, pues estos son robustos a cambios de iluminación, ruido y son fácilmente ajustables para proporcionar mayor robustez contra rotaciones en plano.

El detector se basa en el algoritmo CWMA, es la base del algoritmo de seguimiento. Utilizando ventanas circulares deslizantes, calcula iterativamente histogramas en el cuadro actual, los cuales se comparan con los histogramas de la estructura del objeto. Es importante que este módulo sea tolerante a distorciones geométricas y robusto a ruido e iluminación. Como muestran los resultados, CWMA muestra los resultados más estables y constantes contra los algoritmos del Estado-Del-Arte evaluados, en cuanto a cambios en la fuente de iluminación y ruido blanco gracias a los módulos de preprocesamiento; de la misma manera es el algoritmo más estable a rotaciones tanto en plano como fuera de plano y ligeros cambios de escala.

El seguidor tiene como objetivo mantener la posición de cada cuadro donde se localizó el objeto, y en base a éstas predecir la posición que tendrá en el siguiente cuadro. Los cambios bruscos de dirección provocan que un modelo de movimiento no sea 100 % confiable, por lo que en el algoritmo propuesto se tomó un fragmento del cuadro alrededor

del área donde debería estar el objeto en el siguiente cuadro, y es rectificado mediante la información de profundidad del objeto, cambiando la posición y/o tamaño del fragmento a evaluar en el siguiente cuadro.

Los experimentos muestran una mejora sustancial de los algoritmos que usan información de profundidad contra aquellos que no la usan, pues con ésta es posible hacer una segmentación por profundidad de la escena de acuerdo a la profundidad del objeto, analizando estas zonas existe una mayor probabilidad de éxito. De entre los algoritmos evaluados, el algoritmo propuesto, CWTA, presenta los mejores resultados en general, aunque por un corto margen, teniendo los algoritmos del Estado-Del-Arte y el algoritmo propuesto un desempeño comparable.

Para el desarrollo del algoritmo presentado se utilizaron dos tecnologías de vanguardia, la cámara Kinect de Microsoft que proporciona imágenes confiables de profundidad a un bajo costo; sin embargo, hasta el momento la distancia de profundidad no es muy grande (< 5 metros); y las tarjetas gráficas (GPU) para el cómputo de datos en general y no solo para gráficos como fue originalmente concebida; las GPUs tienen sus desventajas también, las cuales se deben comprender para sacar el mayor provecho del dispositivo; la tasa de transferencia de la memoria del sistema computacional a la GPU y viceversa, hace que sea mejor programar todos los módulos posibles en la GPU, aunque los algoritmos sean altamente seriales. Por ello, al usar GPUs debe planearse con mucho cuidado y tener en cuenta el tamaño de cada estructura y variable que mandamos a la unidad, pues esta es limitada.

El uso de estos avances tecnológicos proporciona más ventajas que desventajas. Usada adecuadamente, la GPU permite procesar concurrentemente cada pixel de la imagen para la extracción del descriptor, luego procesar un gran número de histogramas y realizar las comparaciones. Esta concurrencia le permite a un algoritmo complicado poderse procesar en tiempo real (30 FPS) o más rápido.

Se debe destacar que el algoritmo propuesto maneja un descriptor menos elaborado que algoritmos como OAPF y RGBDOcc+OF; sin embargo, obtiene resultados similares o superiores. Además, el algoritmo propuesto tiene la capacidad de tener un mejor desempeño si se usara una GPU de gama alta (20+ unidades de cómputo), en lugar de

la GPU de gama baja (2 unidades de cómputo) que utilizamos en este trabajo. La complejidad teórica del algoritmo se basa mediante el marco trabajo tiempo, ejecutándose en $\mathbb{T}(\mathrm{n}) = \mathrm{O}(log\mathrm{n})$ pasos con $\mathbb{W}(\mathrm{n}) = \mathrm{O}(\mathrm{n}\log\mathrm{n})$ operaciones.

7.1. Trabajo futuro

- Implementar un modelo de objeto dinámico como se usa en OMIL y TLD, con el objetivo de tener el descriptor más reciente del objeto, junto con un historial de descriptores, que se puedan utilizar para mejorar el reconocimiento con grandes cambios en la perspectiva del objeto.
- Implementar un clasificador más robusto para la correspondencia del objeto, como máquinas de soporte vectorial, de tal forma que se pueda entrenar en línea con fragmentos falsos y verdaderos para mejorar la identificación y clasificación del objeto.
- Implementar el algoritmo para que trabaje en tiempo real para seguir múltiples objetos, actualmente es posible identificar varios objetos; pero lo hace un objeto a la vez, luego el tiempo de procesamiento se multiplica por la cantidad de objetos.
- Desarrollar un algoritmo robusto de segmentación y generación automática de la estructura geométrica, ya que actualmente elegimos el tamaño del radio y la cantidad de discos dentro del objeto de tal forma que ocupen la mayor área de éste.
- Implementar el algoritmo en CUDA, pues se sabe que tiene mejor desempeño que OpenCL ya que es un lenguaje completamente dedicado a GPU.

7.2. Publicaciones derivadas de este trabajo

Revista

- D. Miramontes-Jaramillo, V. Kober, V. Díaz-Ramírez, y V. Karnaukhov, “A novel image matching algorithm based on sliding histograms of oriented gradients,” Journal of Communications Technology and Electronics 59(12), 1446–1450 (2014).

- D. Miramontes-Jaramillo, V. Kober, V. Dıaz-Ramırez, y V. Karnaukhov, “A new descriptor-based tracking algorithm using a depth camera,” Journal of Communications Technology and Electronics 61(6), (2016) (en prensa).

Congreso

- D. Miramontes-Jaramillo y V. Kober, “A fast kernel tracking algorithm based on local gradient histograms,” En: SPIE Optical Engineering + Applications. Applications of Digital Image Processing XXXVI, 8856, 885618_1–885618_8 (2013).
- D. Miramontes-Jaramillo, V. Kober, y V. H. Díaz-Ramírez, “CWMA: Circular Window Matching Algorithm,” En: Progress in Pattern Recognition, Image Analysis, Computer Vision and Applications, 8258, 439–446 (2013).
- D. Miramontes-Jaramillo, V. Kober, y V. H. Díaz-Ramírez, “Multiple objects tracking with HOGs matching in circular windows,” En: SPIE Optical Engineering + Applications. Applications of Digital Image Processing XXXVII, 9217, 92171N_1–92171N_8 (2014).
- D. Miramontes-Jaramillo, V. Kober, y V. Karnaukhov, “Fast image matching with sliding HOGs,” En: Pattern Recognition and Information Processing, 2014, 187–190 (2014).
- D. Miramontes-Jaramillo, y V. Kober, “A Robust Tracking Algorithm Based on HOGs Descriptor,” En: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, 54–61 (2014).
- D. Miramontes-Jaramillo, V. Kober, y V. H. Díaz-Ramírez, “Rotation invariant tracking algorithm based on circular HOGs,” En: Pattern Recognition, 115–124 (2015).
- D. Miramontes-Jaramillo, V. Kober, y V. H. Díaz-Ramírez, “Robust illumination-invariant tracking algorithm based on HOGs,” En: SPIE Optical Engineering + Applications. Applications of Digital Image Processing XXXVIII, 95991Q_1–95991Q_8 (2015).
- A. I. Mendoza-Morales, D. Miramontes-Jaramillo, y V. Kober, “Illumination-invariant hand gesture recognition,” En: SPIE Optical Engineering + Applications. Optics and Photonics for Information Processing IX, 9598, 95980W_1–95980W_8 (2015).

- D. Miramontes-Jaramillo, y V. Kober, “Real-time tracking based on rotation-invariant descriptors,” En: 2015 International Conference on Computational Science and Computational Intelligence, 543-546 (2015).
- B. A. Echeagaray-Patrón, D. Miramontes-Jaramillo, y V. Kober, “Conformal parameterization and curvature analysis for 3D facial recognition,” En: 2015 International Conference on Computational Science and Computational Intelligence, 843-844 (2015).

Lista de referencias bibliográficas

Aguilar-González, P. M. y Kober, V. (2012). Design of correlation filters for pattern recognition using a noisy reference. *Opt. Comm.*, **285**(5): 574–583.

Babenko, B., Ming-Hsuan, Y., y Belongie, S. (2009). Visual tracking with online multiple instance learning. En: *IEEE Conf. on Comp. Vis. and Patt. Rec.*. pp. 983–990.

Bay, H., Ess, A., Tuytelaars, T., y Gool, L. V. (2008). Surf: Speeded up robust features. En: *Comp. Vis. and Imag. Under.*. Vol. 110, pp. 346–359.

Beauchemin, S. S. y Barron, J. L. (1995). The computation of optical flow. *ACM Comput. Surv.*, **27**(3): 433–466.

Bovik, A. (2000). *Handbook of Image & Video Processing*. Academic Press.

Bovik, A. (2009). *The Essential Guide to Image Processing*. Academic Press.

Boyle, R. y Thomas, R. (1988). *Computer Vision: A first Course*. Blackwell Scientific Publications.

Buchanan, A. y Fitzgibbon, A. (2007). Combining local and global motion models for feature point tracking. En: *Comp. Vis. and Patt. Rec.*. pp. 1–8.

Calonder, M., Lepetit, V., Strecha, C., y Fua, P. (2010). Brief: Binary robust independent elementary features. En: *Euro. Conf. on Comp. Vis.*. pp. 778–792.

Chapra, S. C. (2012). *Applied Numerical Methods with MATLAB for Engineers and Scientists*. McGraw-Hill, tercera edición.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., y Stein, C. (2001). *Introduction to Algorithms*. McGraw-Hill, segunda edición.

Dalal, N. y Triggs, B. (2005). Histograms of oriented gradients for human detection. En: *Comp. Vis. and Patt. Rec.*. Vol. 1, pp. 886–893.

Díaz-Ramírez y Kober, V. (2007). Adaptive phase-input joint transform correlator. *Appl. Opt.*, **46**(26): 6543–6551.

Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J., y Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *Int. Jour. of Comp. Vis.*, **88**(2): 303–338.

Fieguth, P. y Terzopoulos, D. (1997). Color-based tracking of heads and other mobile objects at video frame rates. En: *Comp. Vis. and Patt. Rec.*. pp. 21–27.

Freedman, D., Pisani, R., y Purves, R. (1998). *Statistics*. W.W. Norton, tercera edición.

Goel, T., Nehra, V., y Vishwakarma, V. P. (2011). Comparative analysis of various illumination normalization techniques for face recognition. *Int. Jour. of Comp. App.*, **28**(9): 1–7.

Gonzalez, R. C. y Woods, R. E. (2002). *Digital Image Processing*. Prentice Hall, segunda edición.

Gupta, P. (2005). *Comprehensive Business Statistics*. Laxmi Publications.

Haritaoglu, I., Harwood, D., y Davis, L. (2000). W4: real-time surveillance of people and their activities. En: *IEEE Trans. on Patt. Anal. and Mach. Intell.*. Vol. 22, pp. 809–830.

Intille, S., Davis, J., y Bobick, A. (1997). Real-time closed-world tracking. En: *Comp. Vis. and Patt. Rec.*. pp. 697–703.

Jähne, B. (2002). *Digital Image Processing*. Springer, quinta edición.

JáJá, J. (1992). *An Introduction to Parallel Algorithms*. Addison-Wesley.

Jiang, X. (2015). Workshop: Depth image acquisition and analysis. En: *Pattern Recognition*. pp. 1–156.

Kalal, Z., Mikolajczyk, K., y Matas, J. (2010). Forward-backward error: Automatic detection of tracking failures. En: *Int. Conf. on Patt. Rec.*. pp. 23–26.

Kalal, Z., Mikolajczyk, K., y Matas, J. (2012). Tracking-learning-detection. En: *IEEE Trans. Pattern Anal. Mach. Intell.*. Vol. 34, pp. 1409–1422.

Kirk, D. B. y Hwu, W. W. (2010). *Programming Massively Parallel Processors*. Morgan Kaufmann.

Kober, V., Mozerov, M., y Alvarez-Borrego, J. (2001). Nonlinear filters with spatially connected neighborhoods. *Opt. Eng.*, **40**(6): 971–983.

Langmann, B., Hartmann, K., y Loffeld, O. (2012). Depth camera technology comparison and performance evaluation. En: *Proc. of the 1st Int. Conf. on Patt. Rec. Appl. and Meth.*.

Liao, C., Wang, G., Miao, Q., Wang, Z., Shi, C., y Lin, X. (2011). Dsp-based parallel implementation of speeded-up robust features. En: *IEICE Trans. on Inf. and Sys.*. Vol. E94-D, pp. 930–933.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Int. Jour. on Comp. Vis.*, **2**: 91–110.

Lun, R. y Zhao, W. (2015). A survey of applications and human motion recognition with microsoft kinect. *Int. Jour. of Patt. Rec. and Art. Intell.*, **29**(5): 1–49.

Manzur, T., Zeller, J., y Serati, S. (2012). Optical correlator based target detection recognition, classification, and tracking. En: *Appl. Opt.*. Vol. 51, pp. 4976–4983.

Meshgi, K., Maeda, S., Oba, S., Skibbe, H., Li, Y., y Ishii, S. (2014). Occlusion aware particle filter tracker to handle complex and persistent occlusions. En: *Comp. Vis. and Imag. Under.*. Under review.

Miramontes-Jaramillo, D., Kober, V., y Díaz-Ramírez, V. (2013). Cwma: Circular window matching algorithm. En: *Proc. 18th Iberoam. Cong. in Patt. Rec.*. Vol. LNCS 8258, pp. 439–446.

Miramontes-Jaramillo, D., Kober, V., Díaz-Ramírez, V. H., y Karnaukhov, V. (2014). A novel image matching algorithm based on sliding histograms of oriented gradients. *Jour. of Comm., Tech. and Elect.*, **59**(12): 1446–1450.

Nejhum, S., Ho, J., y Yang, M. (2010). Online visual tracking with histograms and articulating blocks. En: *Comp. Vis. and Img. Underst.*. pp. 901–914.

Ortiz, R. (2012). Freak: Fast retina keypoint. En: *Proc. of the 2012 IEEE Conf. on Comp. Vis. and Patt. Rec.*. pp. 510–517.

Ouerhani, Y., Jridi, M., Alfalou, A., y Brosseau, C. (2013). Optimized preprocessing input plane gpu implementation of an optical face recognition technique using a segmented phase only composite filter. *Opt. Comm.*, **289**: 33–44.

Pearson, K. (1895). Contributions to the mathematical theory of evolution. ii. skew variation in homogeneous material. En: *Philos. Trans. of the Royal Soc. A: Math., Phys. and Eng. Sci.*. Vol. 186, pp. 343–414.

Pitas, I. y Venetsanopoulos, A. N. (1992). Order statistics in digital image processing. En: *Proc. of the IEEE*. Vol. 80, pp. 1893–1921.

Po-Ming, L. y Hung-Yi, C. (2005). Adjustable gamma correction circuit for tft lcd. En: *IEEE Symp. On Circ. and Syst.*. pp. 780–783.

Pratt, W. K. (2007). *Digital Image Processing*. John Wiley & Sons.

Rodríguez, J. A. y Perronnin, F. (2008). Local gradient histogram features for word spotting in unconstrained handwritten documents. En: *Int. Conf. on Front. In Handwritting Rec.*.

Rosenkrantz, W. A. (2008). *Introduction to Probability and Statistics for Science, Engineering and Finance*. CRC Press.

Rosten, E. y Drummond, T. (2006). Machine lealearning highspeed corner detection. En: *Euro. Conf. on Comp. Vis.*. Vol. 1.

Rublee, E., Rabaud, V., Konolige, K., y Bradski, G. (2011). Orb: an efficient alternative to sift or surf. En: *IEEE Int. Conf. on Comp. Vis.*. pp. 2564–2571.

Sanders, J. y Kandrot, E. (2010). *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley Professional.

Sarvaiya, J. N., Patnaik, S., y Bombaywala, S. (2009). Image registration by template matching using normalized cross-correlation. En: *Int. Conf. on Adv. in Comp., Contr., and Telecomm. Tech.*. pp. 819–822.

Sato, K. y Aggarwal, J. (2004). Temporal spatio-velocity transform and its application to tracking and interaction. *Comp. Vis. Img. Underst.*, **96**(2): 100–128.

Sbalzarini, I. y Koumoutsakos, P. (2005). Feature point tracking and trajectory analysis for video imaging in cell biology. *Jour. of Struct. Bio.*, **151**(2): 182–195.

Schweitzer, H., Bell, J., y Wu, F. (2002). Very fast template matching. En: *Eur. Conf. on Comp. Vis.*. pp. 358–372.

Sethi, I. y Jain, R. (1987). Finding trajectories of feature points in a molecular image sequence. En: *IEEE Trans. on Patt. Anal. and Mach. Intell.*. Vol. 9, pp. 56–73.

Solomon, C. y Breckon, T. (2011). *Fundamentals of Digital Image Processing*. Wiley-Blackwell.

Song, S. y Xiao, J. (2013). Tracking revisited using rgbd camera: Unified benchmark and baselines. En: *Proc. Of 14th IEEE Int. Conf. on Comp. Vis.*. pp. 233–240.

Sonka, M., Hlavac, V., y Boyle, R. (2008). *Image Processing, Analysis, and Machine Vision*. Thomson.

Takacs, G., Chandrasekhar, V., Tsai, S., Chen, D., Grzeszczuk, R., y Girod, B. (2013). Fast computation of rotation-invariant image features by approximate radial gradient transform. En: *IEEE Trans. Imag. Proc.*. Vol. 22, pp. 2970–2982.

Talu, F., Turkoglu, I., y Cebeci, M. (2011). A hybrid tracking method for scaled and oriented objects in crowded scenes. En: *Exp. Syst. with App.*. Vol. 38, pp. 13682–13687.

Taneja, H. C. (2008). *Advanced Engineering and Mathematics*, Vol. 2. I. K. International.

Tate, R. y Northern, J. (2008). Fast template matching system using vhdl. En: *IEEE Region 5 Conf.*. pp. 1–5.

Verma, R. y Ali, J. (2013). A comparative study of various types of image noise and efficient noise removal techniques. *Int. Jour. of Adv. Res. in Comp. Sci. and Soft. Eng.*, **3**(10): 617–622.

Viola, P. y Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. En: *Comp. Vis. and Patt. Rec.*. Vol. 1, pp. 511–518.

Viola, P., Platt, J. C., y Zhang, C. (2007). Multiple instance boosting for object detection. En: *Adv. in Neur. Info. Proc. Sys.*. Vol. 18, pp. 1417–1426.

Yilmaz, A., Javed, O., y Shah, M. (2006). Object tracking: A survey. En: *ACM Computer Surveys*. Vol. 38, p. 45.

Zalesky, B. A. y Lukashevich, P. V. (2011). Scale invariant algorithm to match regions on aero or satellite images. En: *Proc. of Patt. Rec. and Inf. Proc.*. pp. 25–30.

Apéndice A. Regresión cuadrática de mínimos cuadrados

La idea de la regresión cuadrática es minimizar la suma de los cuadrados (Chapra, 2012). Para un polinomio de segundo orden, el mejor ajuste al polinomio se da al minimizar

$$S_r = \sum \left(y - a_0 - a_1 x - a_2 x^2\right)^2 . \tag{127}$$

En general, para un polinomio de orden m, la minimización se da por

$$S_r = \sum \left(y - a_0 - a_1 x - a_2 x^2 - ... - a_m x^m\right)^2 . \tag{128}$$

Para generar el ajuste de mínimos cuadrados, se toma la derivada de la ecuación anterior respecto a cada uno de los coeficientes desconocidos del polinomio

$$\frac{\partial S_r}{\partial a_0} = -2 \sum \left(y - a_0 - a_1 x - a_2 x^2\right) , \tag{129}$$

$$\frac{\partial S_r}{\partial a_1} = -2 \sum x \left(y - a_0 - a_1 x - a_2 x^2\right) , \tag{130}$$

$$\frac{\partial S_r}{\partial a_2} = -2 \sum x^2 \left(y - a_0 - a_1 x - a_2 x^2\right) . \tag{131}$$

Estas ecuaciones se igualan a cero y crean el siguiente conjunto de ecuaciones normales

$$\begin{array}{rcrcrcl} (n) a_0 & + & \left(\sum x\right) a_1 & + & \left(\sum x^2\right) a_2 & = & \sum y \\ \left(\sum x\right) a_0 & + & \left(\sum x^2\right) a_1 & + & \left(\sum x^3\right) a_2 & = & \sum xy \\ \left(\sum x^2\right) a_0 & + & \left(\sum x^3\right) a_1 & + & \left(\sum x^4\right) a_2 & = & \sum x^2 y \end{array} \tag{132}$$

en forma matricial tenemos

$$\begin{bmatrix} n & \sum x & \sum x^2 \\ \sum x & \sum x^2 & \sum x^3 \\ \sum x^2 & \sum x^3 & \sum x^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum y \\ \sum xy \\ \sum x^2 y \end{bmatrix} . \tag{133}$$