UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

INGENIERÍA ELÉCTRICA-PROCESAMIENTO DIGITAL DE SEÑALES

RECONOCIMIENTO DE OBJETOS EN MOVIMIENTO A PARTIR DE UN SISTEMA DE VISIÓN ESTEREOSCÓPICA

TESIS

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERÍA

PRESENTA:

HÉCTOR RODRIGO ARCE GONZÁLEZ

TUTOR PRINCIPAL:

JESÚS SAVAGE CARMONA

FACULTAD DE INGENIERÍA

MEXICO, Cd. Mx., 2018

Diciembre

JURADO ASIGNADO:

Presidente:	Dr. Pérez Alcázar Pablo Roberto.
Secretario:	Dr. Rivera Rivera Carlos.
Vocal:	Dr. Savage Carmona Jesús.
1er Suplente:	Dr. Escalante Ramírez Boris.
2o Suplente:	Dr. Arámbula Cosío Fernando.

Ciudad Universitaria, 2018.

TUTOR DE TESIS
JESÚS SAVAGE CARMONA

AGRADECIMIENTOS:

A mis padres y familiares por el apoyo que me han brindado a lo largo de mi trayectoria.

A Stephany por los momentos que hemos vivido juntos y el cariño incondicional brindado.

A Arturo, Daniela, Víctor, Alexis, Alejandro y José Luis por su amistad todos estos años.

A Ignacio y Byron, compañeros de posgrado, que ayudaron a resolver y generar dudas.

A los compañeros de laboratorio, especialmente a Reynaldo, José Luis, Jaime, Edgar, Jesús y Marco, por los consejos brindados durante mi tiempo en el mismo.

A mi tutor, Jesús Savage, por el espacio, tiempo y paciencia brindados durante el desarrollo de este trabajo.

Al jurado de este trabajo, por el tiempo y dedicación en la revisión del mismo.

A la Universidad por la tan diversa educacion obtenida.

Al CONACYT por el financiamiento otorgado durante el desarrollo de esta maestría.

Resumen

En el presente trabajo se elabora y prueba un algoritmo para la detección de objetos que se encuentren en estado de movimiento, indicando no solo el objeto reconocido sino también su localización en el espacio tri-dimensional. Se trabaja sobre un marco teórico de detección y reconocimiento de objetos, manipulación computacional de imágenes y métodos de adquisición de una representación en profundidad, debido a la ya mencionada necesidad de obtener una medición que nos indique su posición. El fin de este algoritmo no es solo satisfacer los parámetros ya mencionados sino también que sea viable para implementarse en dispositivos con baja capacidad de procesamiento, como sistemas embebidos; debido a esto también se realizan pruebas de desempeño sobre dispositivos con distintas capacidades de procesamiento y en diferentes resoluciones de adquisición de imagen, denotando con esto ultimo la dualidad de desempeño entre resolución y velocidad de adquisición. Al final de este trabajo se muestra que términos de resultados se logro obtener un algoritmo cuyo desempeño computacional resulta compatible con el requisito de implementación sobre sistemas embebidos, sin sacrificar la capacidad de detectar la posición física de los objetos en el espacio.

Índice general

Índice de figuras

CAPÍTULO 1

Introducción

1.1. Objetivos

Con este trabajo se buscó realizar el desarrollo de un sistema basado en visión estereoscópica el cual pudiese ser capaz de reconocer objetos en movimiento previamente definidos, así como su posición y dimensiones en el espacio, para posteriormente ser implementado en un robot móvil. Para este caso especifico, se plantea su uso principalmente sobre el humanoide tipo "Nimbro-Op", mostrado en la figura 1.1.

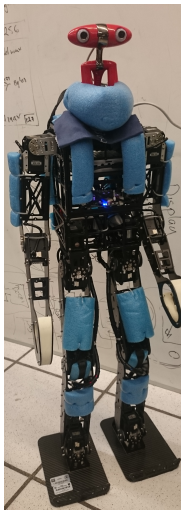


Figura 1.1: Robot Nimbro-Op, con cámara estereoscópica añadida

Los objetivos particulares fueron los siguientes:

1. Abordar el tema del reconocimiento de objetos en movimiento y métodos que se utilizan actualmente para realizarlo.
2. Detectar el tamaño y distancia a dichos objetos gracias al método de reconocimiento elegido.
3. Implementar el método de visión estereoscópica de la forma mas óptima posible para el problema planteado.
4. Desarrollar un sistema que sea capaz de detectar el movimiento de uno o varios objetos referentes a la cámara.
5. Reconocer objetos que se encuentran en estado de movimiento.
6. Utilizar el sistema desarrollado para poder ajustar los parámetros del mismo a través de los algoritmos y experimentaciones de prueba elegidos.
7. Desarrollar un sistema que sea capaz de detectar el movimiento de uno o varios objetos referentes a la cámara.

1.2. Hipótesis

Con este trabajo se busca resolver las siguientes interrogantes:

- ¿Es posible generar un algoritmo implementable con cualquier par de cámaras compatibles con la regulación UVC (*USB Video Class*), lo suficientemente ligero en ejecución como para poder instalarse en un sistema embebido para un robot móvil?
- De ser posible, ¿Que tan exacto será el seguimiento de objetos en movimiento al obtener una predicción de su desplazamiento junto con su posición en el espacio físico tridimensional?

1.3. Motivación

El presente trabajo se enfoca principalmente en la aplicación sobre hardware mayormente comercial de tal forma que la aplicación de los resultados tenga, en términos de eficiencia y costo, la menor cantidad de restricciones posibles, ya que muchos de los

algoritmos y programas similares consumen una cantidad considerable de recursos computacionales.

Además, aunque el sistema resultante se plantea como una aplicación sobre un robot móvil, también se realizó para no ser excluyente a otro tipo de aplicaciones en diferentes contextos, ya que ofrece flexibilidad en su ejecución al permitir realizar modificaciones de forma sencilla sobre las bases de datos y parámetros de los objetos en movimiento a reconocer.

En general, el reconocimiento de objetos en movimiento para este caso en particular se realiza teniendo presentes las siguientes consideraciones:

- El objeto debe estar bajo iluminación que no sature el sistema de visión.
- El objeto se encuentra dentro de un rango establecido de distancia máxima y mínima.
- El movimiento del objeto se encuentra dentro de cierto rango definido.

Durante el desarrollo de este trabajo se profundiza en el procedimiento ejecutado para obtener una mejor aproximación a estos rangos de operación.

1.4. Estructura de la tesis

Para el capítulo 2, Antecedentes, se describen los elementos que abordan los problemas de estimación computacional de movimiento, velocidad y posición sobre objetos, así como las diferentes formas de realizar visión computacional. También en este capítulo se describen trabajos previos y el estado del arte actual en el tema de reconocimiento del movimiento sobre objetos conocidos.

En el capítulo 3, Marco teórico, se abordan los conceptos mas utilizados en esta tesis, principalmente los utilizados para describir objetos de forma computacional.

El capítulo 4 contiene los elementos que conformaron al sistema final y el desarrollo de cada uno de los módulos que componen al mismo. También se definen los parámetros por los que se optaron y los rangos de operación.

La implementación del sistema se aborda en el capítulo 5, en la que se muestran los resultados junto con las comparaciones en diferentes entornos de prueba elegidos.

Finalmente el capítulo 6 muestra las conclusiones obtenidas de la realización del trabajo, así como recomendaciones para un posible trabajo futuro.

CAPÍTULO 2

Antecedentes

Para obtener un sistema de conocimiento ontológico del medio ambiente, es importante no solo considerar objetos en una posición estática, sino también objetos que se encuentren describiendo de forma dinámica una trayectoria en el espacio. Para realizar de forma exitosa el trabajo de detección de objetos en movimiento hasta ahora se utilizan 2 líneas de investigación principales respecto al orden para realizar dicha tarea, las cuales son:

1. Reconocimiento del objeto y la subsecuente predicción de su movimiento.
2. Detección de movimiento en la escena, para posterior clasificación de objetos.

La investigación realizada en este trabajo se basa principalmente en la segunda de estas tendencias, debido a la consideración de que para conocer la posición de un objeto en el espacio no basta con saber su ubicación en un espacio plano respecto al eje visual de una cámara, sino también su distancia a ésta. Además, la ventaja del segundo enfoque resulta evidente en términos de velocidad de procesamiento, ya que al considerar primero el movimiento se minimiza la cantidad de posiciones en las que el objeto en cuestión podría encontrarse, sin mencionar que al reducir el área de interés también se reducen la cantidad máxima de descriptores necesarios en los objetos de entrenamiento.

Una vez definido el enfoque adquirido para resolver este problema y las necesidades adicionales, es posible dividir el procedimiento en 3 secciones principales;

- Adquisición de los datos visuales; debido principalmente a la necesidad conjunta de obtener una imagen para clasificar objetos en movimiento y estimar también la posición espacial de éstos.

- Estimación de la posición del objeto en movimiento; Como el enfoque elegido propone primero definir el estado de movimiento, resulta primordial definir un algoritmo para generar una región de interés que involucre dicho estado.
- Posición y características espaciales del objeto detectado; Uno de los requisitos es el obtener la distancia espacial euclidiana hacia el objeto, por lo que resulta importante definir una forma de cuantificarlo.

2.1. Adquisición y descripción de la imagen

La teoría aplicada en este trabajo se basa fuertemente en la necesidad de generar la representación de una posición tri-dimensional a partir de estereoscopía, debido a esto, el realizar detección y descripción de características sobre las imágenes utilizadas para la generación de la estereoscopía resulta una prioridad, pues facilita el reconocimiento de los objetos dentro de la escena representada, así como la posición espacial de los mismos. De acuerdo con Tazi y Jain (2014) se presenta una dualidad, en cierta medida opuesta, entre generación de una representación de profundidad y generación de una región de interés por flujo de imágenes, partiendo del tamaño necesario de la linea base del sistema estereoscópico, a continuación una tabla de los parámetros que la resumen:

	Profundidad	Movimiento
Construcción desde el punto de vista directo de estereoscopía	Funcional	Sin relación directa
Abordaje teórico-computacional	Establecer una perspectiva para reconstrucción 3D sobre una Imagen	Imágenes o secciones tomadas en diferentes instantes de tiempo
Linea base ideal entre las cámaras para estereoscopía (En términos cualitativos)	Larga	Corta

Cuadro 2.1: Relación entre linea base, profundidad y movimiento

Debido a que establecer una perspectiva sobre un par de imágenes, tal y como lo exige la estereoscopía, no resulta excluyente a capturar imágenes en diferente instante de tiempo, se considera que entonces los requisitos de la estereoscopía son mas importantes. Dados los requisitos de estereoscopía en términos de la linea base ideal, la necesidad de una linea base corta se omite, en lugar de esto, se buscara obtener una cantidad continua de cuadros por segundo generando así el mismo resultado. Una vez definida la adquisición, la implementación de detectores y descriptores de características sobre la imagen queda a implementación bajo las siguientes dos restricciones:

1. Los detectores y descriptores deberán aplicarse sobre las imágenes destinadas a establecer la perspectiva tri-dimensional.

2. Dichos detectores y descriptores deben permitir cierta velocidad de procesamiento entre cuadros.

Es sobre estas dos principales premisas que se basa la búsqueda del detector/descriptor a implementar en este trabajo.

Respecto a la representación por profundidad a generar con estereoscopía, es obtenible a través de la disparidad entre las cámaras separadas por la ya mencionada linea base, para obtener esta disparidad las cámaras generalmente se describen previamente en términos de sus características internas y externas (relación en el espacio tri-dimensional); sin embargo, en muchas ocasiones, pese a la obtención de dichos parámetros, la disparidad resultante contiene bastante ruido, por lo que describir algo sobre ella en esos casos se vuelve complicado.

Para compensar una disparidad con poca calidad se han desarrollado dos principales vertientes de aplicación; filtrar la disparidad obtenida, o compensar el ruido a partir de patrones de búsqueda (Wan y Miao (2009)). El trabajo desarrollado se basa en la primer vertiente al aplicar un filtro de mínimos cuadrados ponderado, descrito mas adelante, pues la segunda vertiente aunque resulta conveniente para minimizar el error del cálculo de los parámetros, tiene el inconveniente de necesitar constantemente un proceso de adaptación tal como describe Zhang et al. (1995), dicho proceso de adaptación aumentaría significativamente la carga de procesamiento computacional del algoritmo y por tanto entorpecería el cumplimiento de los objetivos en términos de compatibilidad con sistemas embebidos comerciales.

Respecto a la Construcción de una imagen tridimensional a partir de un sistema de adquisición, pese a que la estereoscopía resulta ser el método elegido no es el único existente, a continuación se abordan brevemente los mas comúnmente utilizados.

2.1.1. Escáneres de luz estructurada

Estos dispositivos se componen de un proyector de luz en espectro no visible y una o varias cámaras que se encargan de medir la deformación en la proyección de la cual se calcula el volumen del objeto en la escena; posteriormente, esta información puede unirse con la imagen de una cámara RGB, construyendo así una escena tridimensional digital basada en la perspectiva visualizada del mundo físico. Uno de los dispositivos más utilizados con esta tecnología es el sensor kinect, que en su primer versión lo incorporaba. Su funcionamiento se ejemplifica en la figura 2.1.

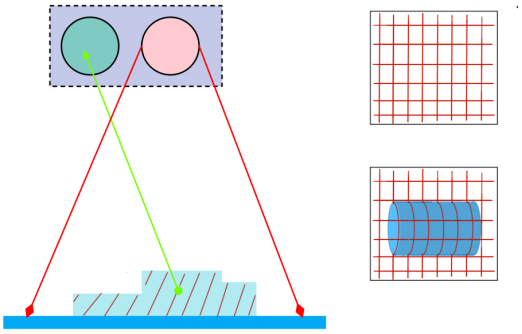


Figura 2.1: Diagrama de funcionamiento de un escáner de luz estructurada, a la derecha un ejemplo de proyección (superior) y su deformación generada por la presencia de un objeto bajo el escáner (inferior).

Una desventaja de adquirir una representación tridimensional con estos dispositivos es que la información en RGB es generalmente escasa, ya que la resolución espacial de las imágenes adquiridas con la cámara debe corresponder a la adquisición máxima del patrón de luz estructurada, sin mencionar que al ser información más independiente al color, está no es necesariamente similar en términos de la forma detectada. Además, al depender de un proyector de luz para detectar el patrón, se corre el riesgo de que el sensor detecte demasiados datos espurios en presencia de ciertas fuentes luminosas de incluso la luz natural misma.

2.1.2. Cámaras de tiempo de vuelo

Estas cámaras contienen sensores especiales para medir el tiempo de vuelo en la reflexión de una señal luminosa, su principio práctico es muy similar al de un sensor ultrasónico combinado con una cámara global shutter, sin embargo aquí el pulso de emisión-recepción es una onda de luz con alta frecuencia y conmutación de encendido (hasta 100MHz, realizado principalmente con LEDs y diodos láser), y la unidad de recepción es una cámara con arreglos en un filtro paso banda para restringir la señal de entrada a solo la luz reflejada con la frecuencia del pulso enviado. Lo obtenido por esta cámara receptora se traduce directamente a distancia por lo que la información adquirida es un mapa de distancias del mundo físico captado dentro de la perspectiva de la lente, en la figura 2.2 se muestra el comportamiento de la cámara.

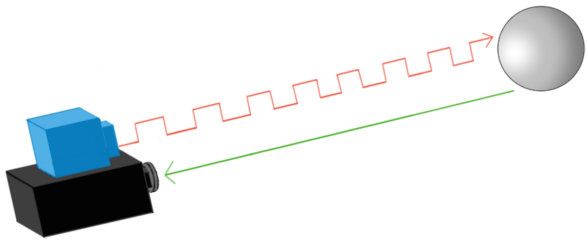


Figura 2.2: Esquema de operación de una cámara de tiempo de vuelo, los dispositivos para envió y recepción se consideran parte de una cámara

La desventaja principal de estos sensores es que debido a la necesidad de emitir una señal, la recepción puede contener muchos reflejos, atribuidos principalmente al rebote de la señal enviada. Otro inconveniente de utilizar este dispositivo es que en caso de necesitar información de color, se debe incorporar de forma paralela una cámara con sensor convencional (RGB), y éste a su vez debe ser capaz de capturar imágenes a gran velocidad, pues la cámara de tiempo de vuelo puede alcanzar hasta 160 cuadros por segundo y podría desfasarse fácilmente de una cámara RGB con una capacidad menor de cuadros.

2.1.3. Visión estereoscópica

Es una técnica mediante la cual se crean mapas de profundidad de una escena a partir de un par de imágenes RGB estándar, la cual para ser implementada, requiere el uso de dos dispositivos de adquisición de imágenes que operen en la misma banda de frecuencia de luz y que dichos dispositivos compartan una resolución espacial igual o similar, así como frecuencia de cuadros por segundo lo mas idéntica posible.

Conceptualmente, el método de visión estereoscópica es relativamente simple (Iocchi (1998)), los dos dispositivos de captura se colocan a una distancia constante entre ellos llamada *linea base*, la cual se encuentra definida por la resolución de los mismos; además, ambos deben colocarse paralelos al plano de la perspectiva a calcular, ya que al tenerlos en esta posición se pueden obtener las distancias de los objetos con volumen en la escena vista por ambas cámaras debido al cambio de perspectiva entre ellas tal y como se muestra en la figura 2.3.

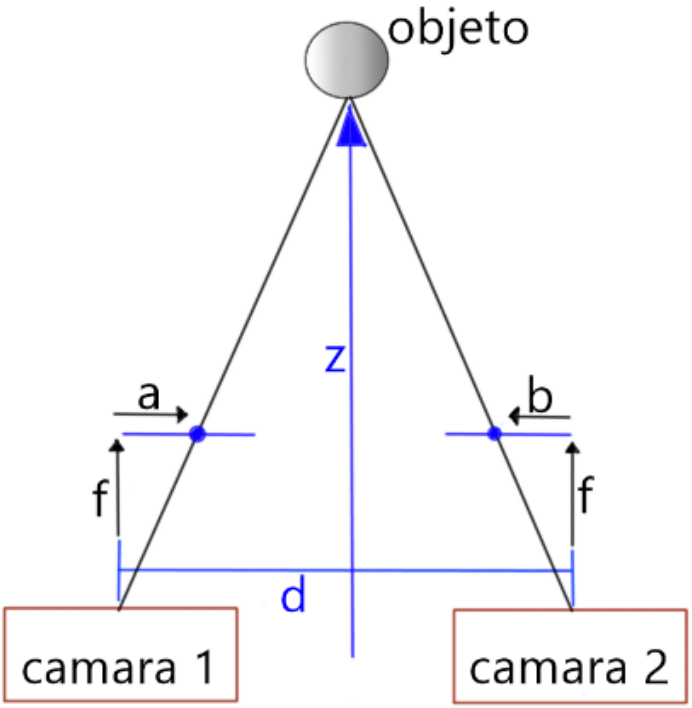


Figura 2.3: Composición geométrica de un sistema de visión estereoscópica

Cabe destacar que para este método es de gran importancia el rectificar las imágenes de entrada obtenidas por los dispositivos de adquisición, ya que al buscar compararlas entre sí éstas pueden presentar aberraciones geométricas, en la imagen siguiente (figura 2.4) se ilustra un ejemplo de dichas aberraciones.

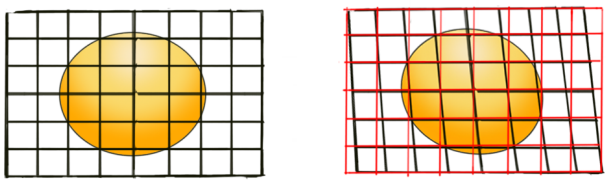


Figura 2.4: Distorsión de una imagen por desfase (skew). A la izquierda la matriz ideal, a la derecha la matriz en rojo muestra la matriz de salida, la negra la perspectiva inducida por el hardware de adquisición.

Para realizar la tarea de rectificar las imágenes es necesario describir un modelo físico de la cámara, así como de los parámetros de ésta respecto a la imagen generada, para finalmente considerar la distorsión de los píxeles a partir de la lente por la que atraviesa la luz reflejada de la escena a capturar.

Para el modelo de la cámara lo más utilizado es el modelo de *cámara pinhole*, el cual dicta que los rayos reflejados por la escena entran a la cámara a través de una apertura única sin lente y éstos se proyectan de forma invertida, quedando así una imagen bi-dimensional invertida en el eje vertical respecto a la original.

Considerando que los rayos del medio ambiente llevan cierta trayectoria y rotación hacia la cámara y posteriormente a la imagen, tal y como se muestra en las figuras 2.5 y 2.6, se toman en cuenta dos componentes diferentes para el modelo que realiza la generación de la representación digital, la parte que representa la posición de la cámara dentro del espacio físico en tres dimensiones (parámetros extrínsecos) y la que representa las características "internas" de la cámara como la longitud focal y el centro óptico (parámetros intrínsecos).

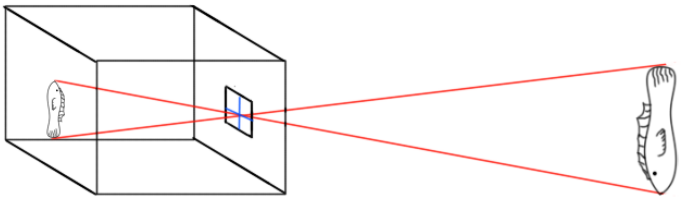


Figura 2.5: Proyección obtenida de una imagen real a un plano bi-dimensional a partir de una cámara de pin-hole

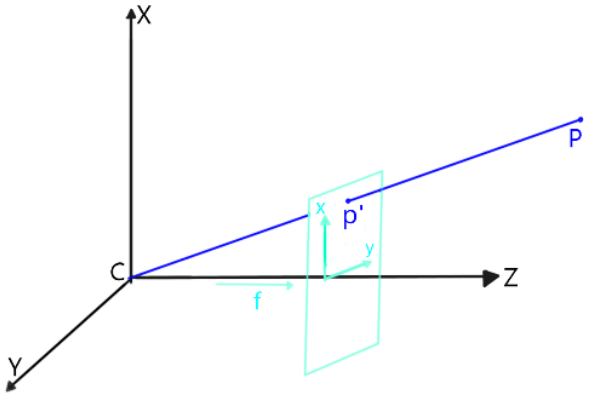


Figura 2.6: Proyección geométrica debido al modelo de pinhole

En términos parámetricos, la componente extrínseca del modelo convierte la posición espacial euclidiana (x,y,z), de la escena capturada respecto a la cámara, en una nueva posición considerando la rotación y traslación de la escena a la cámara (Jähne (2005)), pero aun tomando en cuenta que tanto la escena vista como la proyección son espacios

planos. Entonces la proyección de un punto $(X, Y, Z)^T$ en el plano de la imagen con una posición $(u, v)^T$ puede ser descrita como:

$$u = \frac{Xf}{Z} \text{ y } v = \frac{Yf}{Z}$$

Donde f denota la longitud focal. Para evitar una operación no lineal en la división, la relación anterior puede ser formulada utilizando un modelo proyectivo, quedando:

$$(\lambda u, \lambda v, \lambda)^T = (Xf, Yf, Z)^T$$

Que a su vez se representaría a partir de una notación matricial como:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Donde $\lambda = Z$ es el factor de escalamiento homogéneo. Ahora bien, considerando que de forma cartesiana el espacio definido posee el origen al centro del conjunto de los datos, mientras que generalmente los sistemas de sensado lo consideran en la esquina izquierda de todo el conjunto, se vuelve necesaria una conversión en el sistema coordenado para los puntos principales $(\sigma_x, \sigma_y)^T$. Colocando la posición del punto principal dentro de la matriz de proyección se obtiene con coordenadas homogéneas lo siguiente:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & \sigma_x & 0 \\ 0 & f & \sigma_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Mientras que los parámetros intrínsecos se emplean para escalar y posicionar los puntos tridimensionales dentro de un sistema bi-dimensional cartesiano, también se debe considerar dentro de este posicionamiento que la proyección de una imagen tridimensional puede acarrear cierta disimetría en los ejes del plano, la cual es muy importante estimar en el momento de generar todos los parámetros; a esta falta de simetría se le conoce como *skew*. En la ecuación anterior se considera solo la proyección del plano sin considerar dicho *skew*, para considerar esta imperfección matemáticamente se realiza lo siguiente:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & \tau & \sigma_x & 0 \\ 0 & \eta f & \sigma_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} K & O_3 \end{bmatrix} * P$$

$$P = (X, Y, Z, 1)^T$$

Considerando que P es el punto tridimensional previamente definido en coordenadas homogéneas. En términos del contenido de los parámetros, se tiene que la proporción cuadrada de los píxeles se define con $\eta = 1$, y la falta de inclinación en los ejes con $\tau = 0$.

Después de obtener el modelo físico de la cámara y los parámetros que caracterizan la imagen, aun resta obtener la distorsión generada por la lente (ver figuras 2.7 y 2.8) , ya que como se mencionó previamente, esto no se toma en consideración dentro del modelo de la *cámara pinhole*. Como tal, la lente puede causar dos tipos principales de distorsión: tangencial y radial. La primera se debe principalmente a que el lente no se ha colocado de forma paralela al sensor de imagen, mientras que la segunda se presenta por la curvatura en la fabricación de la lente.

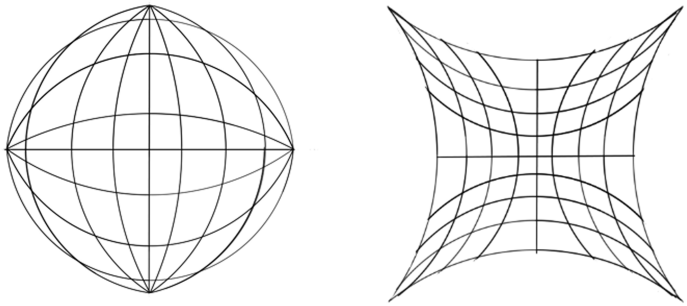


Figura 2.7: Ejemplo gráfico de distorsión tangencial y radial, a la izquierda distorsión de barril o radial positiva, a la derecha distorsión de alfiletero o radial negativa

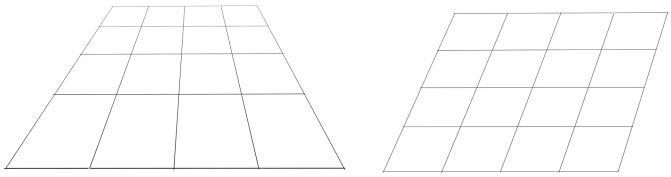


Figura 2.8: Comportamiento de la distorsión por perspectiva (izquierda) y por asimetría (derecha)

La aportación de los parámetros de estas distorsiones a nuestro modelo de calibración son principalmente el re-ordenamiento de las posiciones de los píxeles, obtenidas por el cambio de distorsión a lo largo de la lente, por lo que para la definición se pueden considerar dos tipos de posiciones, la distorsionada ($(x_d, y_d)^T$) y la rectificada ($(x_u, y_u)^T$), su relación se puede representar de la siguiente forma:

$$\begin{bmatrix} x_u - \sigma_x \\ y_u - \sigma_y \end{bmatrix} = L(r_d) \begin{bmatrix} x_d - \sigma_x \\ y_d - \sigma_y \end{bmatrix}$$

$$L(r_d) = 1 + k_1((x_d - \sigma_x)^2 + (y_d - \sigma_y)^2)^2$$

De esta ultima ecuación se puede resaltar que $L(r_d)$ simboliza la diferencia de posición entre el píxel distorsionado y el rectificado, tal que si $K_1 = 0$ entonces $x_u = x_d$ y $y_u = y_d$, por lo que no habría distorsión. Como puede verse, para poder calcular esta rectificación es muy importante considerar el píxel origen (σ_x, σ_y) previamente definido, pues a partir de éste se va realizando todo el mapeo del plano.

El obtener un mapa de profundidad con este método es mas complicado y computacionalmente costoso respecto a dispositivos como las cámaras de tiempo de vuelo y escáneres de luz estructurada, ya que en esos casos la profundidad se obtiene del sensado del dispositivo, mientras que en este caso primero se deben calcular las similitudes entre las imágenes capturadas y es con estas que se logra calcular la posible profundidad de la escena. Para encontrar las similitudes se emplean extractores y detectores de características sobre ambas imágenes bi-dimensionales, así como técnicas de búsqueda para empatar dichas características entre ambas. Una vez que se han encontrado las características, se procede a utilizar algoritmos de empate para relacionar geométricamente la posición de los píxeles entre las cámaras, a partir de la distancia física definida por la linea que las separa (linea base); a esta distancia entre los píxeles se le conoce como disparidad y su inverso es la profundidad del píxel en el mundo físico respecto a la cámara. Si se conocen las distancias que caracterizan físicamente a la cámara (Referirse a la imagen 2.3, pagina 9) y una vez calibrando la imagen obtenida, la profundidad hacia un punto se puede obtener a través de la ecuación:

$$d = z + f = \frac{f * d}{a - b}$$

2.2. Detección de movimiento en una escena

La base del algoritmo realizado en este trabajo yace principalmente sobre la detección del movimiento en una escena, para el posterior reconocimiento del objeto de interés, dicha detección del movimiento se agrupa por mascaras de movimiento vinculadas posteriormente a un objeto previamente entrenado para su reconocimiento y seguimiento, por medio de la predicción de la posición siguiente en la trayectoria descrita por el objeto respaldado por un filtro de Kalman.

La etapa de segmentación para el movimiento tiene una gran tendencia a realizarse por medio de la descomposición de la escena en zonas con diferente cuantización de mo-

vimiento, tal es el caso de Hariharakrishnan y Schonfeld (2005), cuyo algoritmo realiza una distinción entre zonas de movimiento y carentes del mismo; sin embargo, este algoritmo resulta muy simple en términos del producto del movimiento segmentado ya que la máscara resultante no es adecuada para verificar la clasificación del objeto detectado, pues en realidad la máscara se compone de un conjunto de zonas de movimiento activo.

De forma similar al trabajo mencionado , realizando también una distinción inicial sobre el movimiento en la escena, se tiene otra tendencia a clasificar la escena vista en grupos de movimiento trascendente, almacenando a su vez la posición y tamaño de cada grupo, tal es el caso de Mezaris et al. (2004), que realiza la aproximación de las regiones de forma bayesiana, teniendo así la capacidad de cuantificar la probabilidad de que un objeto comience a ser ocluido como parte del movimiento que esta describiendo.

Implementaciones mas completas de detección y seguimiento de objetos se han realizado en torno a la búsqueda de movimiento dentro del espacio tri-dimensional euclidiano, tal es el caso de Stein y Shashua (1998), cuyo trabajo destaca la búsqueda sobre una dimensión alta de forma directa implementando un filtro de Kalman sobre una nube de puntos existente. Aunque este método resulta muy atractivo también requiere un alto costo computacional, pues el solo hecho de identificar los datos de la nube de puntos requiere de una carga de procesamiento bastante grande respecto a las ya mencionadas implementaciones en espacio bi-dimensional.

La representación de movimiento en espacio tri-dimensional también se ha visto utilizada para generar la representación del objeto, en lugar de buscarlo dentro de la escena, tal es el caso de Moyung y Fieguth (2000), que valida primero la secuencia del movimiento y posteriormente que tanta relación se genera debido al empate estereoscópico, para finalmente empatar la secuencia de movimiento en cada tiempo t con la profundidad detectada en los mismos tiempos.

Como ya se mencionó en la sección previa a ésta, complementándose en esta misma, el método a abordar en este trabajo se basa principalmente en la estimación de una posición siguiente a través del filtro de Kalman por sobre un seguimiento por detección. A continuación se mencionan los aspectos generales de ambos métodos.

2.2.1. Seguimiento por detección

Una de las primeras técnicas para seguir un objeto dentro del flujo de una escena se conoce actualmente como “Seguimiento por detección”. Esta técnica se basa en simplemente generar información del movimiento de un objeto detectado al obtener una cantidad N de muestras a lo largo de un numero M consecutivo de cuadros, esperando que la cantidad de detecciones en N sea lo suficientemente significativa dentro de M como

para trazar una relación continua coherente entre el punto de detección inicial y el ultimo punto de referencia del objeto detectado. En general esta técnica es sencilla de aplicar si el desarrollo esta mayormente enfocado al área de la detección y descripción de objetos, ya que el elemento importante aquí es la posición misma detectada en el objeto conocido; sin embargo, debido a esto, también posee la desventaja de que se requiere detectar el objeto para forzosamente saber a futuro el movimiento a describir, lo que conlleva un alto peso computacional por cada cuadro del flujo de video.

2.2.2. Filtro de Kalman

El filtro de Kalman se utiliza para estimar un proceso empleando una forma de realimentación, ya que el proceso se divide en una secuencia de estados los cuales son estimados a través del tiempo mientras se obtienen simultáneamente muestras con ruido. Las ecuaciones que definen un filtro de Kalman (Brown y Hwang (2012)) caen en dos grupos, predecir y corregir, el primer grupo se encarga de calcular cierto error junto con estados anteriores para así tener estimados para cada paso siguiente, avanzando así el estado actual a lo largo del tiempo. Las ecuaciones para corrección de error son responsables de incorporar una nueva medida en el estimado anterior, de tal forma que el estimado siguiente mejore, realizando así la re-alimentación. El proceso descrito se ilustra también en la figura 2.9.

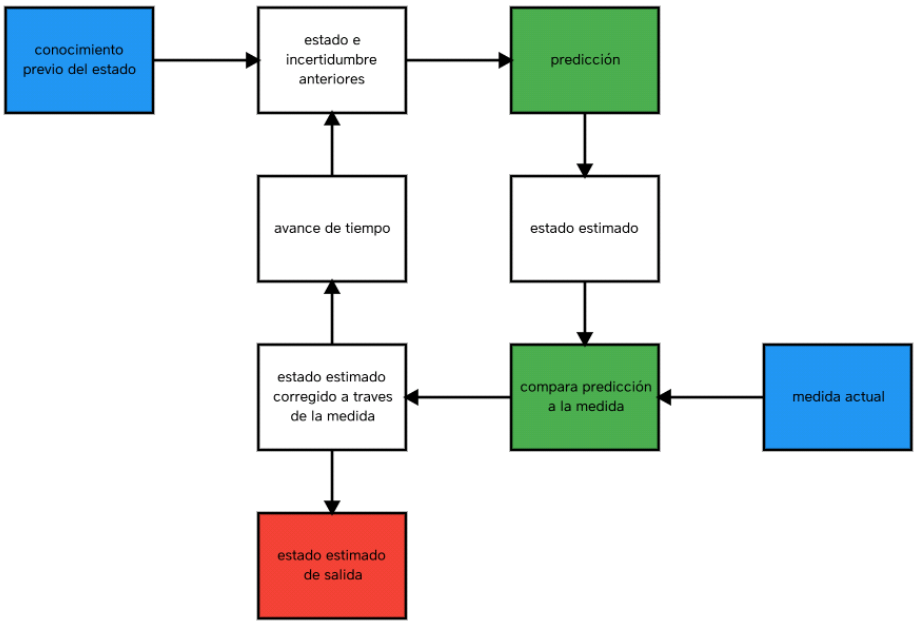


Figura 2.9: Algoritmo general para filtrado kalman, marcando las etapas de predicción y corrección

2.3. Interacción y localización de un robot móvil con el entorno

El avance actual de la robótica ha llevado a que la técnica conocida como Mapeo y localización simultanea o SLAM (simultaneous localization and mapping) sea una de las mas implementadas y estudiada; dicha técnica se basa en la creación y corrección en tiempo real de un mapa referido a un ambiente, esto a partir de la información sensorial del móvil que navega a través de dicho ambiente. En términos de implementación en este trabajo no se busca realizar dicha técnica, sin embargo se espera que esta implementación especifica sea escalable a dicho método tal y como se comenta en *trabajo a futuro*.

De acuerdo con Konolige et al. (2008), la visión estereoscópica resulta ventajosa en una aplicación de SLAM, pues las cámaras permiten detectar objetos a una mayor distancia, restringida no por un sistema de sensado activo sino por un conjunto de sensores de imagen, esto último es mucho mejor pues generalmente resulta mas sencillo conseguir un nuevo sistema de adquisición visual por sobre un nuevo sistema de sensado directo de distancia. Una ventaja más de utilizar la visión estereoscópica como método de adquisición es el hecho de que pueden detectarse un sin numero de texturas visualmente adquiribles, siendo la única limitante el desarrollo computacional en este ámbito.

CAPÍTULO 3

Marco teórico

3.1. Representación digital del medio ambiente

3.1.1. Imágenes 2D

Una imagen bi-dimensional es, como tal, la proyección de una escena física a una función discreta de 2 variables con divisiones equi-espaciadas tanto en su variable horizontal como vertical, a estas divisiones se les conoce formalmente como píxeles. Cuando se habla de una imagen, no solo basta definir sus propiedades geométricas, sino también el contenido, ya que una imagen puede representar no solo un valor por píxel, sino todo un conjunto de parámetros. En el caso de las representaciones bi-dimensionales se suele tener uno de dos tipos de valores contenidos: *monocromáticos* o *color*. Cuando hablamos de una imagen con valores monocromáticos tenemos que cada píxel representa un valor dentro de la escala de luz de una sola frecuencia del espectro visible, y esta frecuencia es la misma para todos los píxeles cambiando solo dicho valor, un ejemplo de esto se ilustra en la figura 3.1. Mientras que para una imagen a color, un píxel contiene los valores de más de una frecuencia, viéndose definidos a partir de un *espacio de color* el cual se encarga de marcar una relación con la cual las frecuencias base (colores) de dicho espacio generarán los colores restantes. Para fines de imágenes interpretables por el ojo humano, generalmente se consideran los valores del espacio RGB, pues éste plantea la generación de los diferentes colores del espectro visible a partir de combinaciones del rojo, verde y azul.

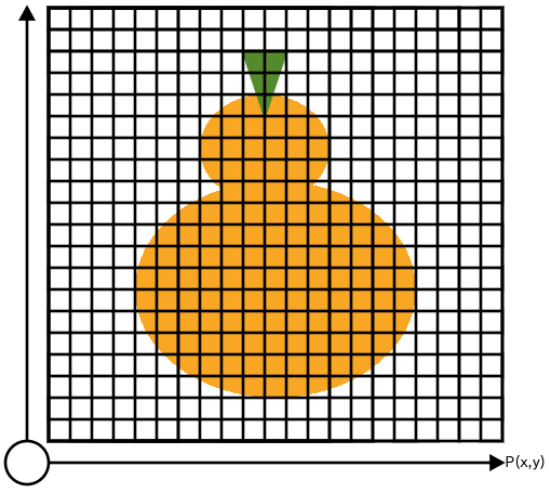


Figura 3.1: Imagen 2D, constituida por una matriz de elementos

3.1.2. Nube de puntos y Mapa de profundidad

La nube de puntos se define como un conjunto de puntos sin relación entre sí contenidos dentro de un espacio euclidiano, dichos puntos se encuentran representados dentro de un sistema de coordenadas discreto, el cual tiene su origen en el dispositivo de captura de los mismos, si dichos puntos se interconectan entre si, el resultado seria una superficie sin intersecciones internas entre los puntos.

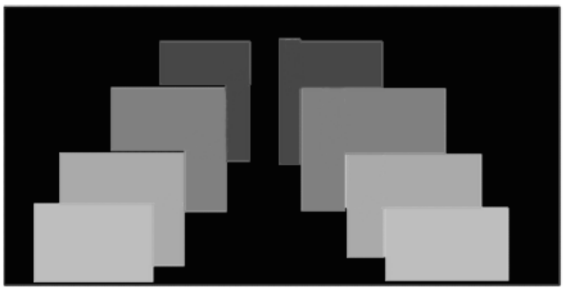


Figura 3.2: Ejemplo gráfico de la representación por diferencia de profundidad en 4 diferentes niveles

Un mapa de profundidad es la representación de un conjunto de puntos discretos o volúmenes dentro de un espacio euclidiano, considerando que éstos se perciben desde cierto punto de vista, tal y como se muestra en la figura 3.2. Si se habla de un mapa de

profundidad de puntos obtenidos a partir de una perspectiva, se obtiene entonces el mapa de las distancias desde el punto de adquisición hasta los objetos en dicha perspectiva, lo cual es muy empleado para adquirir información espacial a través de estereoscopia, pues ambos dispositivos de adquisición se colocan paralelos en una escena.

3.2. Procesamiento digital

3.2.1. Filtrado espacial

Algunas operaciones sobre vecindarios consideran los valores en el píxel de estudio así como los de sus vecinos, definiendo un espacio a partir de todas estas regiones consideradas. A partir de este espacio se genera una sub imagen relacionada al mismo con el fin de operarlos entre si. A dicha sub imagen, con las mismas dimensiones que el vecindario, se le conoce como filtro (Gonzalez y Woods (2018)), mascara, kernel o ventana, y sus valores son referidos como coeficientes en lugar de píxeles, se puede apreciar un ejemplo de máscara en la figura 3.3.

El uso de estas máscaras para realizar filtrado se basa en el concepto del uso de la transformada de fourier sobre una señal en el dominio de la frecuencia, sin embargo, en este caso, el filtrado se realiza directamente sobre el área de los píxeles cubiertos por la máscara definida en lugar de operar sobre una representación frecuencial, para distinguir entre ambas operaciones se utiliza el término de filtrado espacial en la primera. La mecánica en la aplicación de estos filtros consiste simplemente en mover la máscara de un punto de interés a otro a través de toda la imagen, mientras que en cada uno se calcula una "respuesta" a partir de la interacción tanto del punto de interés como los vecinos cubiertos por dicha máscara; la respuesta obtenida a partir de esta interacción es el resultado del filtrado. El filtro puede relacionarse de forma lineal o no lineal. En la figura 3.4 se puede ver el resultado de aplicar una mascara directamente sobre la totalidad de una imagen, sin realizar el mencionado barrido.

W11	W21	W31	W41	W51	W61
W12					⋮
W13					
W14					
W15					
W16	...				W66

Figura 3.3: Ejemplo global de máscara $N * N$, con N pesos

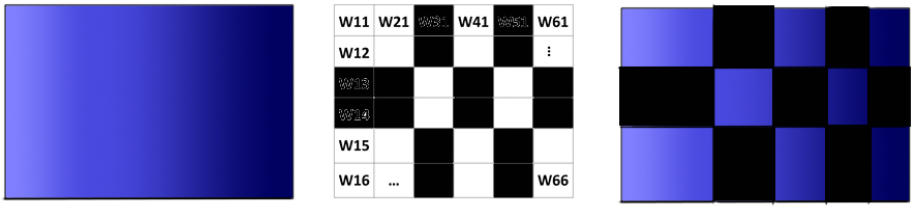


Figura 3.4: Ejemplo de operación directa con una máscara sobre una imagen de un degradado

Filtrado espacial lineal

En este caso se realiza la aplicación de un filtro similar a los utilizados en frecuencia, pero con una representación bi-dimensional, a dicha representación se le conoce como kernel. La función principal de estos filtros es la de restringir o modificar la escala de valores representados en las regiones espaciales de la imagen bi-dimensional, lo cual se logra de distintas formas y es por ello que hay una amplia variedad de filtros, la mayoría basados en el análisis de señales dentro del dominio de la frecuencia. Algunos de los filtros más utilizados (Lim (1990)) son:

- Filtro Gaussiano
- Filtro Gradiente
- Filtro de promedio ponderado
- Filtro paso banda
- Filtro Laplaciano
- Filtro de Wiener
- Filtro paso alta
- Filtro de restricción banda
- Filtro Derivativo
- Filtro de Media
- Filtro paso baja
- Filtro de Hilbert

La obtención del kernel de un filtro espacial basado en alguno de los filtros digitales anteriores es relativamente simple, visto dentro de un espacio tridimensional hay que

considerar la misma forma de onda proyectada en el plano perpendicular desde el centro de la señal, posteriormente la señal se discretiza para finalmente acotar la muestra dentro de una matriz. En la figura 3.5 se muestra un ejemplo de este proceso con una campana de Gauss.

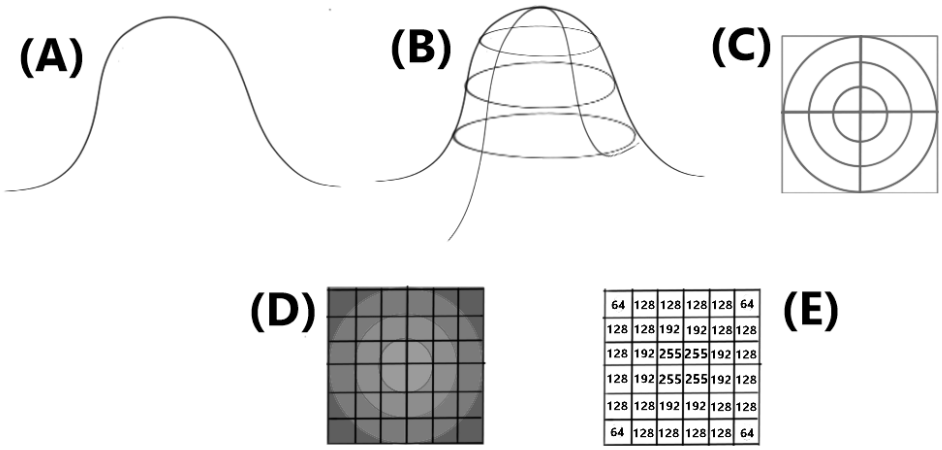


Figura 3.5:
A) Campana de Gauss utilizada en filtrado lineal para señales
B) Superficie de revolución de una campana de Gauss
C) Vista superior de la superficie
D) Discretización de la campana en 4 niveles (Marcados por intensidad de grises), y división en N=6 filas/columnas, donde N sera el tamaño del kernel cuadrado.
E) Kernel de un filtro Gaussiano, finalmente los valores discretos se cuantificaron a un nivel de 8 bits (0 a 255 en base decimal).

Al realizar una convolución entre la imagen y el kernel se obtendrá la imagen filtrada. Debido a que el tamaño del kernel influirá en la cantidad de píxeles con los que se debe operar cada píxel de la imagen original, se tiene que:

$$I_{salida}(x, y) = \sum_{u,v}^{N,M} I_{entrada} * g(u, v)$$

Donde g(u,v) es el kernel o filtro.

3.2.2. Filtrado no lineal: Procesamiento morfológico

El procesamiento morfológico (Haralick et al. (1987)) se basa en tomar una imagen binaria y un elemento estructural como entrada, combinándolos usando un conjunto de operaciones conocidas como intersección y unión. Estas operaciones procesan los objetos en la imagen de entrada basándose en su forma, que se codificará de acuerdo al elemento estructural.

Los elementos estructurales (también llamados kernel) utilizados para la codificación pueden tomar 2 tipos de arreglos matriciales, binarios y con peso, el primero siendo representado como una imagen blanco/negro y el segundo como una imagen en escala de grises, en ambos casos, en la figura 3.6 se muestran ejemplos de estos elementos. Al aplicar un elemento binario sobre una imagen, se considera que los píxeles blancos son parte de la región a mostrar, mientras que los píxeles negros, representando el fondo, eliminarán aquellos píxeles de la imagen con los que se operen.
Si en cambio el elemento estructural contiene pesos (es en escala de grises), entonces se considera que éste es una superficie tridimensional que se eleva desde el fondo (píxeles negros) hacia la cresta de la superficie (píxeles blancos). Al aplicar este operador no solo se mantienen o eliminan los píxeles en la imagen original, también se disminuyen en intensidad, entre más oscuro sea el píxel del kernel mas se disminuirá la intensidad del píxel correspondiente con esta. En general, ambos casos de kernel se pueden representar con la siguiente ecuación:

$$pixel_{nuevo} = pixel_{original} * pixel_{kernel}$$

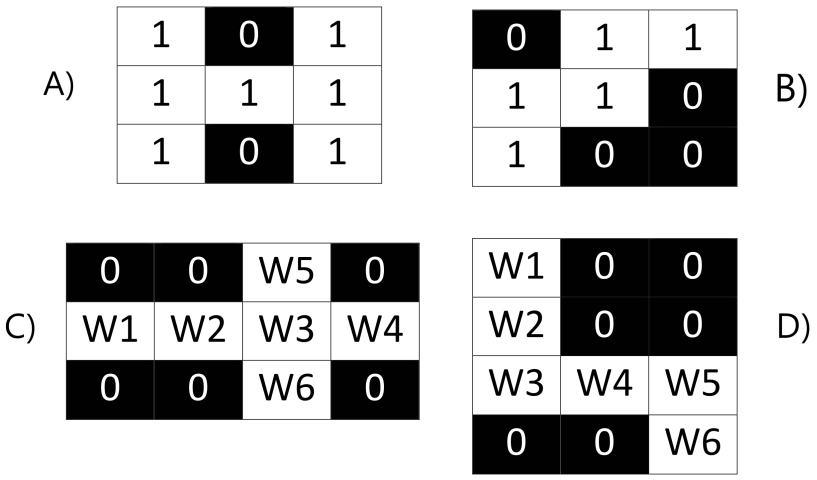


Figura 3.6: Diferentes ejemplos de elementos estructurales. A y B tienen un peso constante de 1, mientras que B y C son de peso W, su geometría siempre se contiene en una matriz al igual que las imágenes a tratar

Respecto a la aplicación del elemento estructural sobre la imagen lo único que se realiza es recorrer el centro del elemento estructural sobre cada uno de los píxeles de la imagen a operar, debido a esto es que generalmente los elementos estructurales tienen una cantidad impar de renglones y columnas, ya que es más sencillo considerar un centro fijo a simplemente operar toda la máscara sobre la imagen (Pratt (2001)). La operación más

básica se denomina *Hit and Miss*, basada en que si el píxel del operador se coloca sobre un píxel de la imagen, este se conserva si sus valores no son de fondo.

Más allá de la operación previamente mencionada, las más utilizadas son aquellas que alteran la geometría de la imagen al ensanchar o adelgazar ciertas áreas, tal es el caso de la operación dilatación, ilustrada en la figura 3.7, donde el resultante de aplicar el operador sobre una imagen es el ensanchamiento de sus zonas continuas. Visualmente, el resultado de una dilatación aplicada varias veces es el de hacer más grandes las zonas con área pequeña, al mismo tiempo que las zonas al rededor comienzan a reducirse debido a esto. En términos de características generales de la imagen, el resultado es similar al de aplicar un filtro de emborronamiento sobre la imagen.

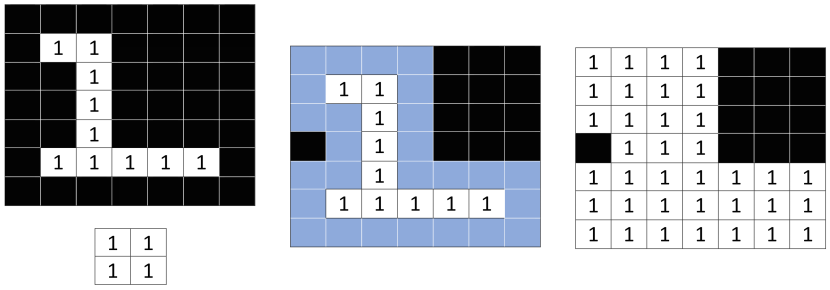


Figura 3.7: Izquierda: Imagen a tratar y elemento estructural a utilizar. Centro: Acción del elemento estructural. Derecha: Resultado de la operación dilatación

Otra operación común es la erosión, que puede considerarse el inverso de la dilatación debido al efecto visual y cuantitativo sobre los píxeles de la imagen. En este caso las regiones continuas que coinciden con las zonas del kernel se ven reducidas o adelgazadas (Figura 3.8). En términos de características pequeñas, éstas eventualmente se ven eliminadas al aplicar continuamente un proceso de erosión, es debido a esto que éste proceso resulta útil para separar objetos pequeños dentro de la imagen y eliminar imperfecciones, sin embargo al utilizarlo como filtro también se degrada la imagen pues regiones como los bordes llegan a volverse discontinuas.

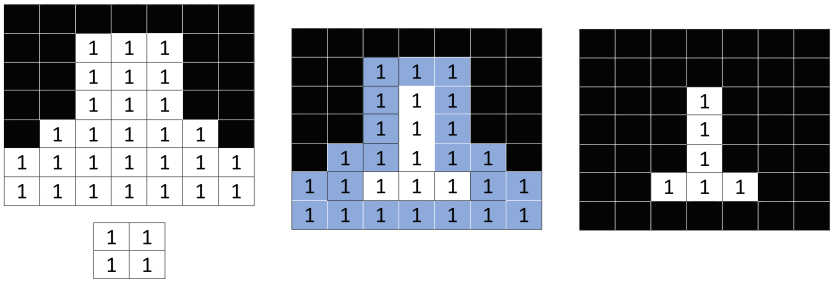


Figura 3.8: Izquierda: Imagen a tratar y elemento estructural a utilizar. Centro: Acción del elemento estructural. Derecha: Resultado de la operación erosión

Las 2 operaciones básicas mostradas anteriormente no se limitan a utilizarse individualmente, al realizarse secuencialmente se logra tener efectos más específicos sobre la imagen, tal es el caso de la operación apertura, que se logra al realizar en secuencia una erosión seguida por una dilatación con el mismo elemento estructural. El efecto de su aplicación es el de conservar elementos en la imagen que son más grandes al elemento estructural mientras se reducen los más pequeños a este, por lo que al aplicar esta operación resulta una buena forma de filtrado sobre ruido con pequeña distribución. Al realizar una apertura con cierto elemento estructural, si se intenta realizar de nuevo no habrá ningún cambio nuevo en los elementos más grandes de la imagen, a esta propiedad se le conoce como idempotencia. La figura 3.9 es un ejemplo de este proceso.

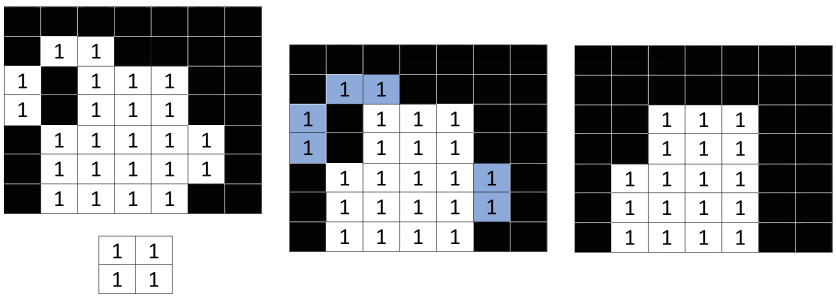


Figura 3.9: Izquierda: Imagen a tratar y elemento estructural a utilizar. Centro: Acción del elemento estructural. Derecha: Resultado de la operación apertura

Si se invierte el orden de aplicación en la erosión y dilatación, iniciando por la dilatación, se obtiene la operación conocida como cerradura (Figura 3.10). Contrario a la apertura, esta operación rellena los pequeños huecos que sean de un tamaño menor al del elemento estructural, esta vez sin afectar a los objetos de mayor tamaño que no posean algún posible vecino al que puedan unirse de cierta forma. El mayor inconveniente de

aplicar cerradura es la posible sobre escritura de características pequeñas por parte del entorno mismo que las contiene.

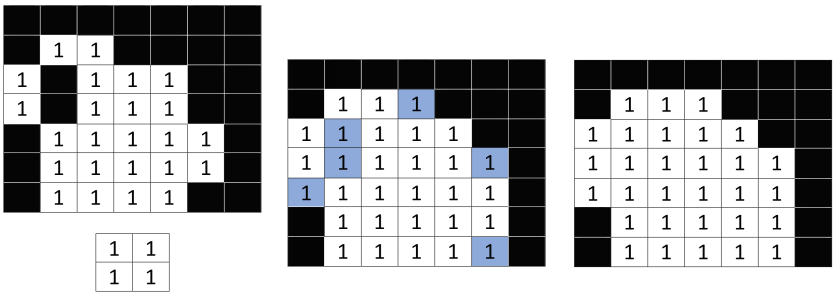


Figura 3.10: Izquierda: Imagen a tratar y elemento estructural a utilizar. Centro: Acción del elemento estructural. Derecha: Resultado de la operación

3.3. Clasificador: K vecinos más cercanos (*k nearest neighbors / knn*)

Es un método de clasificación utilizado principalmente cuando no hay conocimiento previo sobre la distribución de los datos, pero aun así, los datos están descritos por algunos parámetros cuantificables conocidos. Fue planteado originalmente en un reporte de la "US Air Force School of Aviation and Medicine" por Fix and Hodges en 1951 y desde entonces ha sido refinado hasta ser uno de los métodos más utilizados por su relativa sencillez respecto a otros.

La principal característica de este clasificador es que la medida realizada se basa, comúnmente, en las distancias euclidianas calculadas entre una muestra de prueba y las muestras de entrenamiento. Para esa medida se considera que una muestra i, de un conjunto máximo n, con j características, puede representarse como una variable $X_{i,j}$. Si se calcula una distancia l desde una muestra a otra, donde $l =!i$, la ecuación para obtener dicha distancia entre 2 muestras queda como:

$$d(\mathbf{x}_i, \mathbf{x}_l) = \sqrt{(x_{i1} - x_{l1})^2 + (x_{i2} - x_{l2})^2 + \cdots + (x_{ip} - x_{lp})^2}.$$

Esta ecuación de estimación de distancia entre muestras de p característica se realiza desde la muestra prueba y hasta el radio de un circulo de distancia k, teniendo que todas las muestras que queden dentro del radio previamente establecido serán las que le darán su clasificación a la muestra prueba. Debido a que la muestra prueba obtiene

su clasificación de las muestras de entrenamiento, resulta obvio decir que las muestras de entrenamiento ya fueron previamente clasificadas y por tanto este es un metodo de clasificación supervisado.

Si al realizar la definición del radio de distancia k, este se considera como 1, entonces se cae en la regla de vecinos más cercanos, la cual (Cover y Hart (1967)) dice que la clase de la muestra de prueba x es igual a la de su vecino mas cercano m.

$$d(\mathbf{m}_i, \mathbf{x}) = \text{mín}_j\{d(\mathbf{m}_j, \mathbf{x})\}.$$

Si se generaliza el análisis de esta ecuación para k muestras de entrenamiento, se puede decir que la clase de una muestra de prueba es igual a la clase mas frecuente dentro de un grupo de muestras de entrenamiento. En la figura 3.11 se ejemplifica este proceso de forma gráfica.

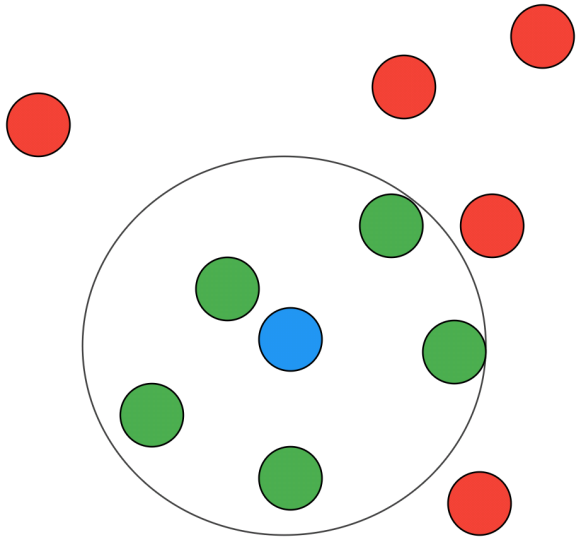


Figura 3.11: Ejemplo de clasificación a 2 clases, por medio de k-means

3.4. Descriptores

La tarea de detectar y posteriormente reconocer objetos a través de visión computacional o el mero procesamiento de imágenes actualmente se basa mayormente en descomponer dichos objetos en un conjunto de datos que sea fácil de describir, estos datos pueden provenir de diferentes características visuales, como bordes, color, tamaño, etc.,

y dada su naturaleza "clave" en el proceso de reconocimiento se denominan *Puntos clave* o *keypoint*. Para describir un *keypoint* y posteriormente detectarlo como un patrón consistente dentro de un nuevo grupo de información, en este caso una imagen, existen varios tipos de algoritmos, cada uno con diferentes parámetros a favor y en contra principalmente en términos de velocidad computacional, precisión de detección y cantidad de características detectadas. Entre los algoritmos desarrollados para este fin, los mas implementados y con mayor eficiencia computacional resultan ser los denominados SIFT, SURF y ORB, siendo este ultimo el implementado en este trabajo.

3.4.1. Transformada de características invariantes a escala (*Scale Invariant Feature Transform / SIFT*)

Este método de descripción (Lowe (2004)) transforma un conjunto de datos característicos contenidos en una imagen a coordenadas invariantes a escala relativas a zonas locales. Para realizar esto se emplea un método de filtrado en cascada aplicando al principio las operaciones más sencillas y dejando las más costosas para el final, ya que a estas solo llegarán los datos que cumplan con los requisitos iniciales del filtrado. Los niveles del filtro son:

1. Detección de espacio escala.
   En esta etapa se buscan datos que tengan existencia a través de diferentes escalas de la misma imagen y que a su vez puedan reconocerse en diferentes vistas de la misma, a los datos que cumplan este último requisito se les denomina candidatos a puntos clave. La comparación para obtener la posible existencia en escalas se implementa a través de la diferencia entre espacios-escala (Witkin (1983)) representada como función de diferencia de Gaussianas, ya que se ha demostrado que la función Gaussiana es la única valida como kernel espacio-escala (Lindeberg (1994)). Para realizar la diferencia, primero se convoluciona la imagen de entrada con funciones Gaussianas para producir imágenes separadas por escala en factor de 2, posteriormente se obtiene un espacio-escala de diferencias en un factor de octavas, a través de los resultados de las convoluciones.

2. Detección de máximos locales
   Después de localizar los candidatos a puntos clave, se realiza un filtrado a partir de la robustez de los mismos dentro de la imagen, para obtener esta medida se considera lo siguiente:

   - Frecuencia de muestreo por escala.
     Para cumplir esta medida se busca que el candidato aparezca de forma constante a través de las diversas escalas obtenidas previamente, aún si estas escalas son afectadas por artefactos como ruido.

- Frecuencia de muestreo en el dominio espacial.
  En este caso los candidatos deben aparecer en las escalas aún si la imagen original sufre variaciones de tamaño por medio de la mezcla de interpolación y suavizado espacial.

3. Asignación de posición al punto clave
   Para esto no solo se considera que el punto clave esta en ciertas coordenadas dada su aparición original, sino que se consideran los puntos locales respecto al punto clave original que cumplan con la aplicación del espacio escala en su máximo.

4. Asignación de orientación
   Finalmente, esta etapa complementa la anterior añadiendo robustez a los puntos clave posicionados al obtener la magnitud y ángulos entre el mismo punto clave y sus diferentes escalas, con estos se genera un histograma de orientación para finalmente considerar la clase mas alta y las consiguientes dentro del 80 % de su altura como las orientaciones mas probables. A partir de las orientaciones obtenidas se genera un punto clave por cada una de ellas respecto al punto clave original, teniendo así varios vectores de orientación.

3.4.2. Características robustas aceleradas (*Speeded Up Robust Features / SURF*)

En este caso se construye un descriptor y detector denominado *Speeded Up Robust Features* o SURF (Bay et al. (2006)) a partir de una detección basada en una medida simplificada de la matriz hessiana, mientras que la descripción es a partir de la generación de una distribución.

El uso de la matriz hessiana se realiza de forma similar al detector Laplace-Hessiano (Mikolajczyk y Schmid (2001)) que demuestra que los resultados de la matriz son precisos y de rápida obtención, sin embargo a diferencia del ya mencionado detector, la posición y escala de las características se obtienen directamente del determinante de la matriz, la cual se representa como:

$$\mathcal{H}(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix}$$

Donde $L_{xx}(x,\sigma)$ es la convolución de la derivada Gaussiana en segundo orden $\frac{\partial^2}{\partial x^2} g(\sigma)$ con la imagen I en el punto x. Para $L_{xy}(x,\sigma)$ y $L_{yy}(x,\sigma)$ el desarrollo operacional es similar.

Respecto a la invarianza a la rotación, se busca encontrar una orientación reproducible y para esto, se obtiene la respuesta a wavelets haar en la dirección horizontal y vertical sobre una región circular previamente seleccionada al rededor del píxel de interés x previamente mencionado.

Finalmente, para poder etiquetar al píxel x como un punto clave valido, se genera una descripción invariante a la iluminación y contraste. Para esto, la región circular previamente utilizada para encontrar la orientación del descriptor se convierte a una región cuadrada, aún centrada al rededor del punto de interés, la cual se divide en sub-regiones tamaño 4x4 para calcular sobre cada una de estas lo siguiente:

- Suma de la respuesta a wavelet Haar en dirección horizontal
- Suma de la respuesta a wavelet Haar en dirección vertical
- Suma del valor absoluto a la respuesta horizontal
- Suma del valor absoluto a la respuesta vertical

Con los resultados de dichos cálculos se genera un descriptor con 4 dimensiones (una por resultado), por tanto el tamaño total del descriptor de la característica referida al píxel x, contando sub-regiones, será de 64. Las respuestas wavelet ya son por sí mismas invariantes a cambios en iluminación, por lo que la invarianza a contraste se logra al convertir el descriptor en un vector unitario.

3.4.3. Descriptor binario *ORB*

Este algoritmo fue mostrado públicamente por primera vez en el 2011 con el artículo *ORB: An efficient alternative to SIFT or SURF*, en el cual se plantea como una alternativa viable a los ya existentes SIFT y SURF, no solo en términos de una mejor eficiencia computacional, sino también en términos comerciales al carecer de un registro de patente en contraste a los otros dos. La mayor ventaja obtenida por ORB sobre los algoritmos entonces existentes es un menor peso computacional al momento de almacenar las características percibidas en la imagen, lo que se traduce a su vez en una mayor velocidad de búsqueda y comparación de éstas entre un grupo de entrenamiento y una nueva escena presentada. Para lograr esto, ORB hecha mano de dos algoritmos ya desarrollados en ese entonces (Rublee et al. (2011)): el detector de esquinas denominado “FAST” y el descriptor binario de puntos “BRIEF”.

Por su parte FAST (Rosten y Drummond (2006)) se originó inicialmente de una propuesta basada en aprendizaje de máquina, como alternativa a detectores de borde

con mayor costo computacional como Harris, SUSAN o derivadas de gaussianas. Este algoritmo se basa en operar sobre un círculo de Bresenham con radio de 3 píxeles a partir de un píxel central de interés, el cual se iterará a través de todo el conjunto de píxeles en la imagen. Como definición, al píxel central de interés se le denomina *píxel p*. En la imagen siguiente (figura 3.12) se muestra un ejemplo de el recorrido de circular ya mencionado.

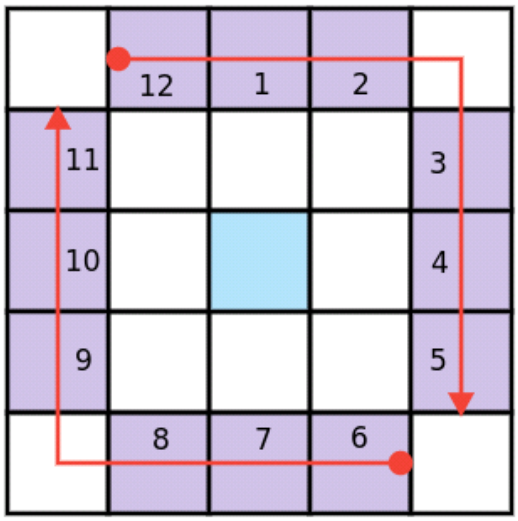


Figura 3.12: Búsqueda sobre un circulo de Bresenham a partir de un punto central

Si al analizar los píxeles que definen el borde del círculo mencionado, se encuentra que existe una cantidad continua n con el mismo valor de brillo entre dos píxeles sobre el perímetro y a su vez dicho valor es mayor o menor al del *píxel p* más cierto umbral, entonces estos píxeles contínuos se consideran como un borde.

Respecto a la definición de la zona de búsqueda para el algoritmo previo a la realización de la búsqueda de continuidades en el radio definido, se realiza una comprobación en la intensidad de los píxeles directamente verticales y horizontales al píxel de referencia p (Calonder et al. (2010)), si la diferencia entre ambos resulta muy grande en las 4 direcciones, el *píxel p* queda descartado completamente de la búsqueda; en caso de poseer un píxel candidato entonces la búsqueda solo se realiza referente a éste. Para definir dicha diferencia se utiliza el criterio siguiente:

$$V = max(\ \sum_{x \in S_{\text{bright}}} |I_{p \to x} - I_p| - t\ ,\ \sum_{x \in S_{\text{dark}}} |I_p - I_{p \to x}| - t)$$

Tal que $S_{\text{bright}} = \{x | I_{p \to x} \geq I_p + t\}$ y $S_{\text{dark}} = \{x | I_{p \to x} \leq I_p - t\}$, donde:

I es la imagen o zona de búsqueda sobre la que se recorrerá la búsqueda respecto al píxel p.

x es la posición siguiente adyacente al píxel p.

t es el valor seleccionado para generar el umbral de búsqueda previamente mencionado.

CAPÍTULO 4

Desarrollo

En este capitulo se aborda el desarrollo de los sistemas particulares utilizados por el algoritmo para el reconocimiento de los objetos en movimiento. A continuación se muestra un esquema mostrando cada uno de los niveles de procesamiento planteados:

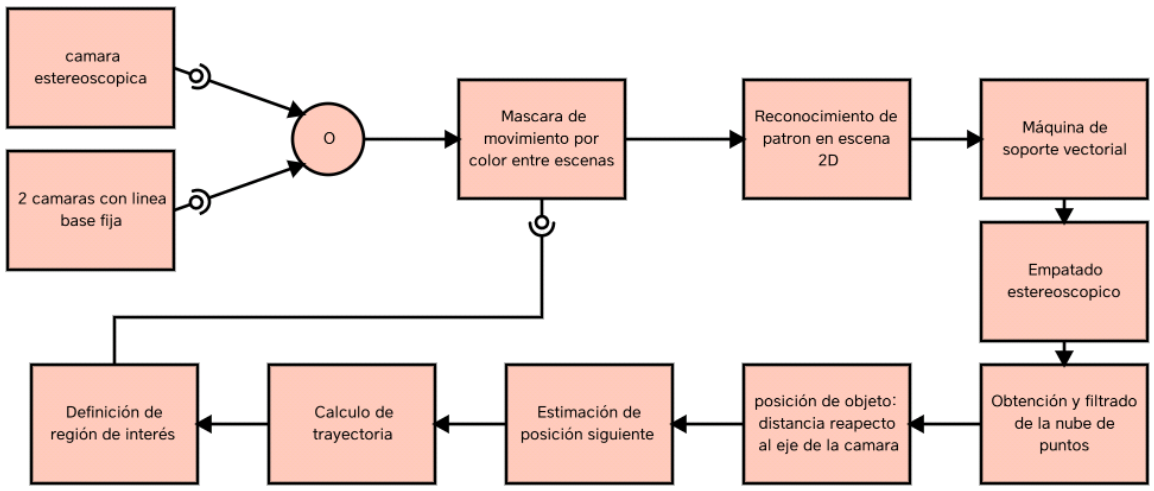


Figura 4.1: Diagrama del procedimiento seguido con el algoritmo de este trabajo

4.1. Adquisición

El sistema de adquisición elegido para este trabajo fue el de visión estereoscópica, pues se aprovecha la relación entre resolución y velocidad de adquisición, a diferencia de

un sensor de movimiento como el kinect, ya que este último presenta la desventaja de tener una resolución demasiado baja si lo que se quiere es realizar procesamiento sobre la imagen bi-dimensional adquirida.

La adquisición se realizó de forma directa desde el USB de la cámara estereoscópica, ya que el algoritmo se plantea compatible con cualquier dispositivo del estándar UVC. Para las pruebas principales de este trabajo se utilizaron 2 dispositivos con estas características, la cámara estereoscópica minoru 3D y la cámara estereoscópica ZED, cada una con las siguientes características:

Dispositivo/Características	Resolución Mínima	Resolución Máxima	Cuadros por Segundo Máximos
Minoru 3D	320 x 240	800 x 600	30
ZED	672 x 376	2208 x 1242	100

4.2. Detección y clasificación

4.2.1. Maquina de soporte Vectorial (MSV)

Se utiliza como técnica de clasificación y en ella, en términos simples, se considera que existe un espacio bi-dimensional que contiene a las muestras y observaciones del experimento a clasificar; cada una de estás muestras se considera como un vector de soporte debido a su posición en el espacio. Posteriormente en la técnica se dividen los grupos de vectores similares, creando así clases bien definidas.

Para realizar las divisiones pertinentes, la maquina de soporte vectorial utiliza un hiper-plano para dividir la información que contiene a las clases existentes, si las clases pueden dividirse por la acción directa de un hiper-plano entonces se dice que la MSV es lineal. En caso de que estas no puedan ser divididas por un plano de manera evidente, se puede modificar la representación de los datos al cambiar la interpretación de los parámetros que los representan, a esto último se le conoce como un cambio de dimensionalidad en el problema, yendo de una dimensionalidad $\mathbb{R}^D$ a una $\mathbb{R}^E$ tal que $E > D$. Desde el punto de vista matemático los datos representados en una variable x se convierten a una matriz hessiana en función de los mismos $x \to \Phi(x)$ (Liu et al. (2010)), entonces:

$$h_{ij} = y_i y_j \Phi(x_i)^T \Phi(x_j), \qquad i, j = 1, \ldots, N$$

Al realizar el cambio de dimensionalidad no es muy conveniente que éste sea de forma explicita un mapeo $\mathbb{R}^D \to \mathbb{R}^E$, pues computacionalmente los cálculos generarán una cantidad de datos muy grandes si la dimensionalidad original ya era grande, por ejemplo una transformación polinomial de segundo orden arrojará un conjunto con términos

constantes, lineales, cuadráticos y cuadráticos cruzados, por tanto para conjuntos de alta dimensionalidad la transformación explícita se vuelve computacionalmente ineficiente. Una forma implícita de realizar el cambio es a partir del *Truco del Kernel*, el cual dicta que existe una versión no lineal de cualquier clasificador lineal si se encuentra una transformación no lineal $\Phi(a)$ provista por un producto escalar que pueda ser expresado como:

$$K(a, b) = \Phi(a)\Phi(b)$$

Con lo anterior se reemplazan los productos escalares de la matriz hessiana por una función kernel $K(a, b)$ semi-definida positiva, donde $a, b \in \mathbb{R}^D$ pertenecen a la dimensión original:

$$h_{ij} = y_i y_j \Phi(x_i)^T \Phi(x_j)$$
$$h_{ij} = y_i y_j K(a, b), \qquad i, j = 1, \ldots, N$$

4.2.2. Detectores y clasificadores de características

Se realizaron pruebas sobre 4 diferentes combinaciones de descriptores de características y métodos de aproximación:

- SURF con método FLANN (Fast Approximate Nearest Neighbor)
- ORB con método FLANN
- SIFT con método FLANN
- SIFT con método FB (Fuerza Bruta)

Los cuales se probaron bajo 4 diferentes entornos de prueba:

- Características detectadas por segundo,
- Tolerancia al escalamiento (Respecto a la disminución en las características del primer punto)
- Tolerancia a la rotación (Respecto a la disminución en las características del primer punto)
- Tolerancia a la perspectiva (Respecto a la disminución en las características del primer punto)

A continuación se muestran los resultados de los experimentos implementados con el lenguaje C++ y la librería OpenCV versión 3 (OpenCV (2014)):

Características por segundo:

Figura 4.2: Características reconocidas en una región de la escena

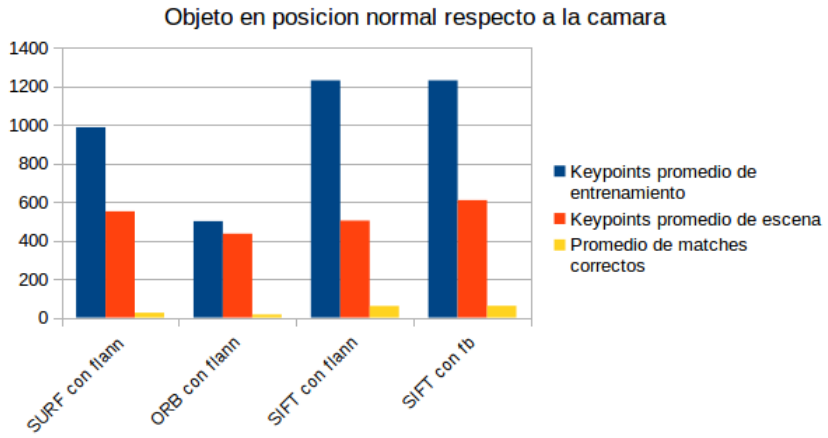


Figura 4.3: Características reconocidas (correctas contra totales)

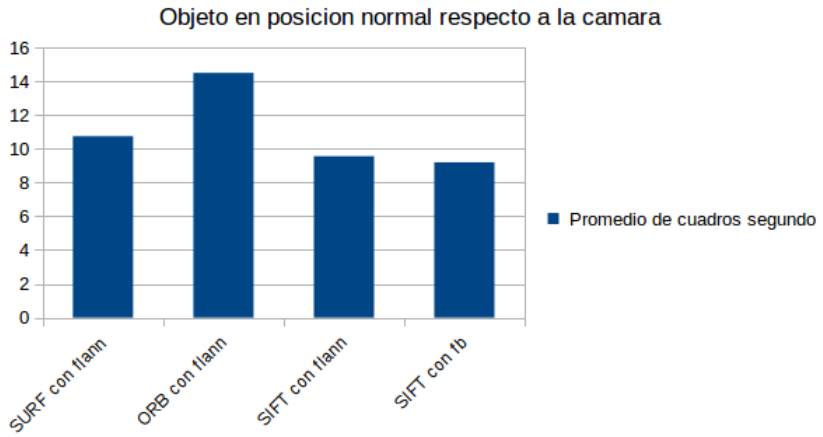


Figura 4.4: Velocidad de detección de características

Tolerancia al escalamiento:

Figura 4.5: Características reconocidas en una región de la escena sobre reducción del objeto

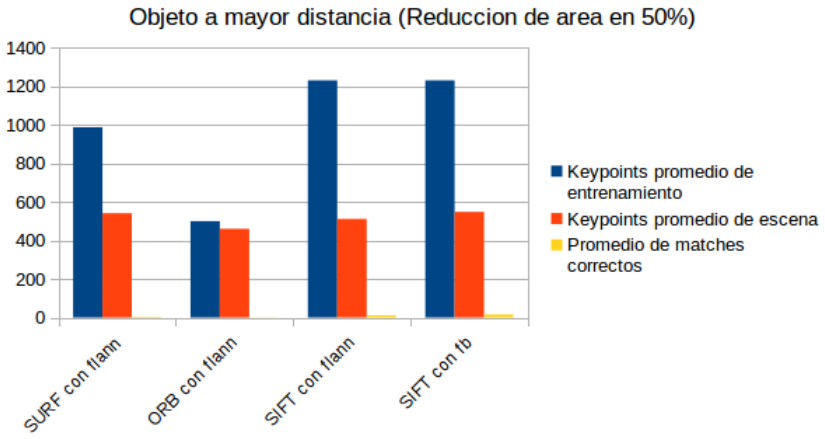


Figura 4.6: Características reconocidas en escalamiento (correctas contra totales)

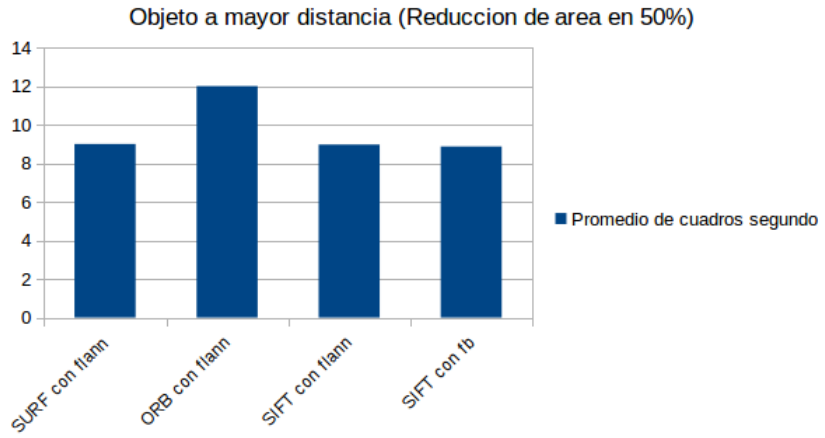


Figura 4.7: Velocidad de detección de características en escalamiento

Tolerancia a la rotación:

Figura 4.8: Características reconocidas en una región de la escena sobre rotación a 180 grados sobre el objeto

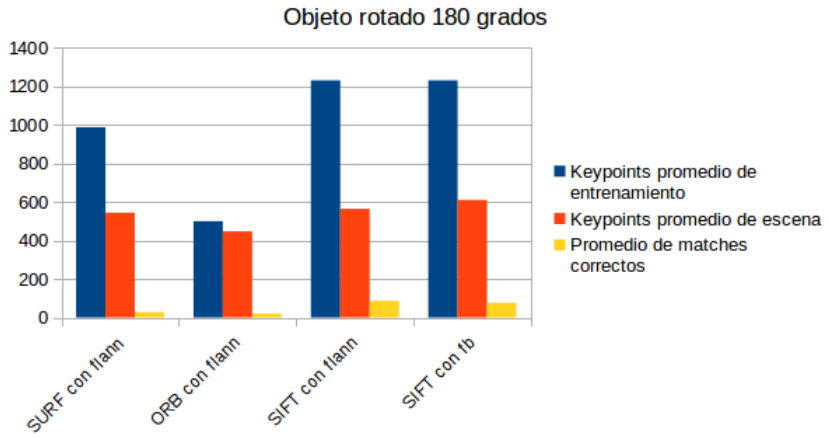


Figura 4.9: Características reconocidas en rotación(correctas contra totales)

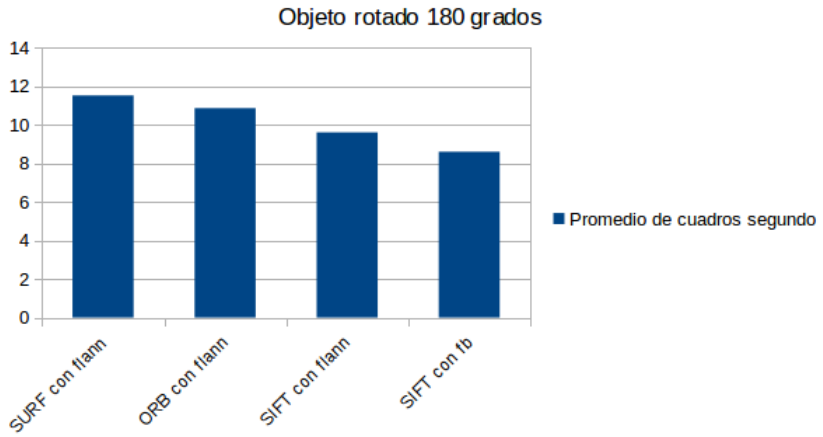


Figura 4.10: Velocidad de detección de características en rotación

Tolerancia a una perspectiva de 45 grados:

Figura 4.11: Características reconocidas en una región de la escena sobre objeto con una perspectiva en la profundidad

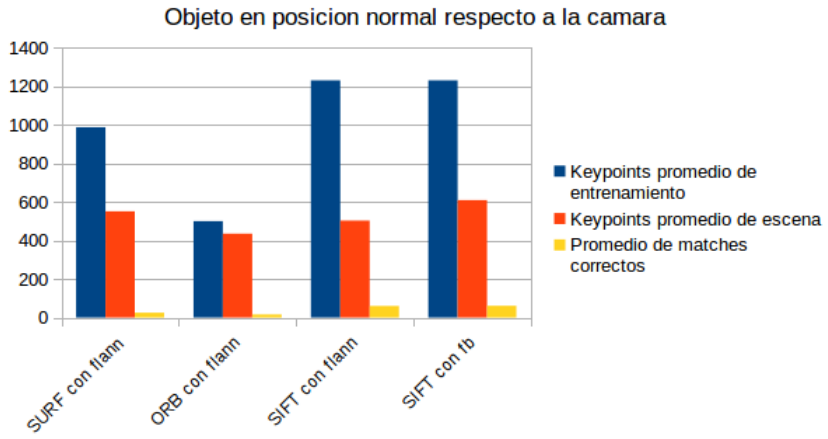


Figura 4.12: Características reconocidas en perspectiva (correctas contra totales)

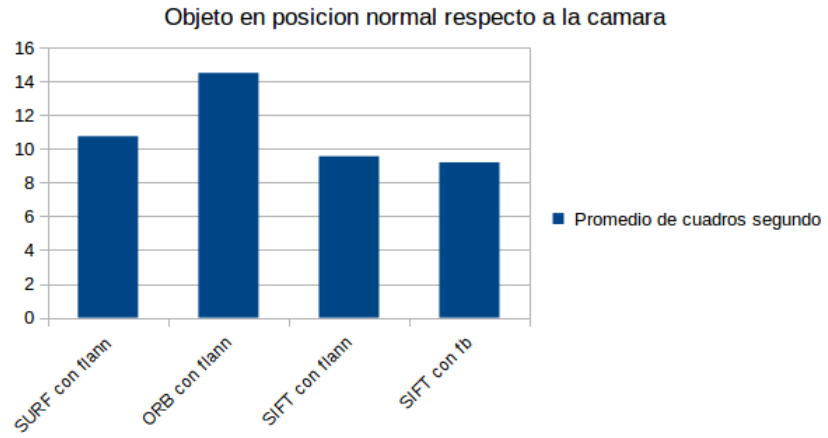


Figura 4.13: Velocidad de detección de características en perspectiva

Si se analizan estos resultados en términos de las gráficas obtenidas, se comprueba que de forma constante el descriptor con el que se obtiene una mayor cantidad de cuadros por segundo es ORB. En términos de la descripción de las imágenes este mismo descriptor es el que reconoce menos características, sin embargo el promedio de características entre la escena y el objeto se mantienen constantes, sin mencionar que estos promedios son muy similares entre sí, es por esta razón que el descriptor ORB se elige para los diferentes niveles de detección de características en el problema planteado.

4.3. Procesamiento del mapa de profundidad

En este caso es necesario un método de filtrado debido a la cantidad de ruido generado al obtener un mapa de disparidad de forma directa por estereoscopia. En lugar de recurrir a un suavizado directo sobre todo el mapa, se realiza un suavizado que preserve los bordes, ya que el resultado de la disparidad constituye mayormente una diferencia entre estos, el filtro que cumple perfectamente con esto último es el mínimos cuadrados ponderados (WLS). Este filtro obtiene su resultado al resolver un sistema lineal con una matriz laplaciana acotada. De forma matemática el filtrado se logra dada una imagen de entrada f y una imagen de referencia g, obteniendo una salida u al minimizar la función de energía de mínimos cuadrados ponderados (Elad (2002)):

$$J(u) = \sum_P \{(u_p - f_p)^2 + \lambda \sum_q [w_{p,q}(g)(u_p - u_q)^2]\}$$

La restricción del suavizado es dada al variar los pesos de la función $w_{p,q}$, la cual representa la similitud entre los pixeles p y q. Dicha función queda definida como:

$$w_{p,q}(g) = \exp(-||g_p - g_q||/\sigma_c)$$

Donde σ_c es el parámetro de rango dinámico. Además de esto se incluye un control de nivel de suavizado sobre la salida, el cual es denotado por λ en la ya mencionada ecuación de energía; si dicha ecuación se iguala a cero, la minimización de u se obtiene al resolver un sistema lineal de matrices acotadas:

$$f = (I + \lambda A)u$$

Donde u y f denotan vectores conteniendo los valores de color de u y f respectivamente. A representa la matriz laplaciana necesaria para obtener el suavizado, definida como:

$$A(m,n) = \begin{cases} \sum_{l \in N(m)} w_{m,l}(g), & n = m \\ -w_{m.n}(g), & n \in N(m). \\ 0, & \text{cualquier otro caso} \end{cases} \tag{4.1}$$

Despejando u de la ecuación de f, obtenemos entonces que el resultado final del suavizado se escribe como:

$$u(m) = ((I + \lambda A)^{-1} f)(m)$$

4.4. Cálculo de posición siguiente y Re-definición de región de interés

Se realiza la consideración de que el desplazamiento de un objeto sobre un plano puede ser modelado como un proceso lineal, el cual puede describirse por las siguientes dos ecuaciones:

$$\text{Ecuación de estado: } x_{k+1} = Ax_k + Bu_k + w_k$$

$$\text{Ecuación de salida: } y_k = Cx_k + z_k$$

Donde A, B y C son matrices, k es el índice del tiempo, x el estado del sistema, u una entrada al sistema, y una salida medida, mientras que w es el ruido del proceso y z el ruido de medida.
Se considera que el vector x contiene toda la información sobre el estado presente del sistema, sin embargo éste no puede ser medido directamente, por lo que se mide y, la cual es una función de x modificada por el ruido z, para posteriormente estimar la x misma.

Abordando el movimiento de un objeto en específico, podemos considerar la composición de los estados por la posición p y la velocidad v. La entrada u será la velocidad añadida (u ordenada en el caso de un móvil controlado), mientras que y sera la posición p medida. Si se considera que se pueden medir ambos cada T segundos entonces la velocidad estará gobernada por la siguiente ecuación:

$$v_{k+1} = v_k + Tu_k$$

Esto significa que la velocidad siguiente será igual a la velocidad presente mas la añadida u ordenada, multiplicada por la cantidad de pasos hacia adelante. El inconveniente en esta ecuación es su falta de precisión, pues no se considera que la velocidad puede ser perturbada por varias fuentes de ruido de índole física, por lo que al considerar una ecuación mas realista se obtiene:

$$v_{k+1} = v_k + Tu_k + n\check{v}_k$$

Donde nv es el ruido sobre la velocidad. Se puede obtener una ecuación similar para la posición p:

$$p_{k+1} = p_k + \tfrac{1}{2}T^2u_k + n\check{p}_k$$

Donde np es el ruido sobre la posición. Cabe aclarar que aquí el avance de la posición a lo largo del tiempo también se considera respecto a la velocidad, por lo que se realiza la conversión pertinente tal que la distancia recorrida sera $a(t) = s'(t)$ donde $s(t)$ es la distancia recorrida en función del tiempo y $a(t)$ es la velocidad como función del tiempo, por lo que $\int (Tu_k)dT = \frac{1}{2}T^2u_k$.

Definiendo lo anterior, se puede obtener un vector de estado x que consista de la posición y velocidad:

$$x_k = \begin{bmatrix} p_k \\ v_k \end{bmatrix}$$

Substituyendo las ecuaciones basándose en la posición, velocidad y el modelo lineal, queda finalmente:

$$x_{k+1} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} u_k + w_k$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} x_k + z_k$$

Dado que se requiere conocer el estimado de la posición siguiente x_{k+1}, entonces también x_k debe ser lo mas exacto posible. Para realizar esta tarea, se elige el filtro de kalman, ya que el algoritmo realizado obtiene medidas "actuales" de posición (y_k) con ruido por medición a compensar, dicha posición se obtiene a partir de una máscara de movimiento generada por el movimiento percibido por la cámara, el algoritmo global se basa en el diagrama 4.14.

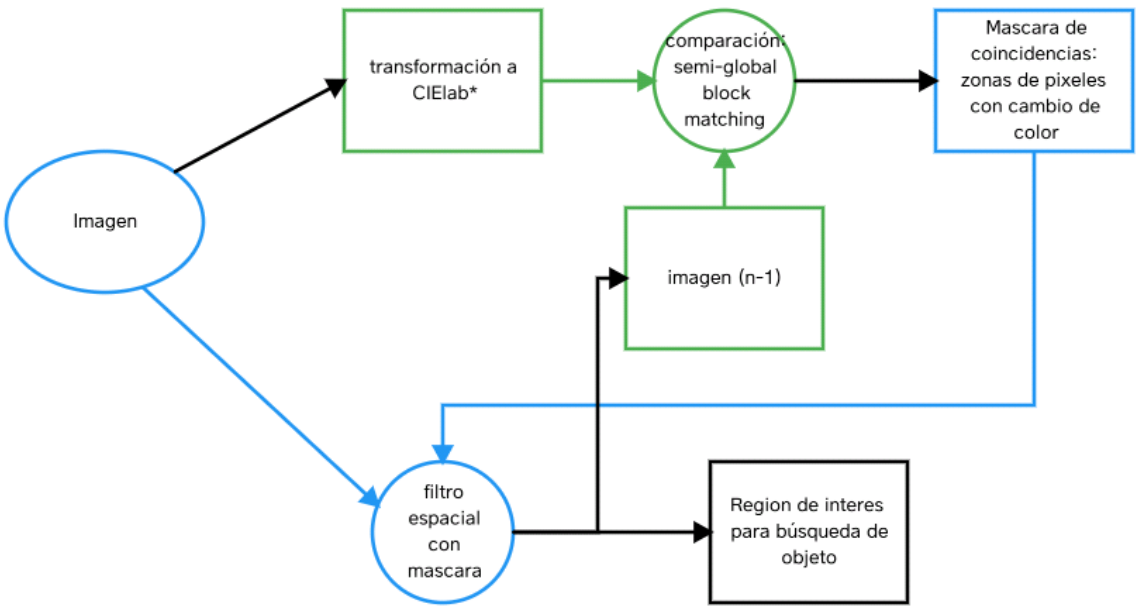


Figura 4.14: Algoritmo del cálculo de la máscara para la posición siguiente

Considerando las ecuaciones ya obtenidas para velocidad y posición, pero en forma meramente causal, se tiene que:

$$x_k = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} x_{k-1} + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} u_{k-1}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} x_{k-1} + z_k$$

A estas se les considera como las ecuaciones para actualización de tiempo del filtro discreto de Kalman.

Para obtener las ecuaciones de actualización de medida, se considera inicialmente que es necesaria una ecuación que calcule un estado posterior $\hat{v_k}$ a partir de una combinación lineal entre un estimado anterior $\hat{v_{k-1}}$ y una diferencia con pesos entre la medida actual z_k y la predicción de medida $H\hat{v_{k-1}}$. Realizando esto se llega a la ecuación:

$$\hat{v_k} = \hat{v_{k-1}} + K_k(z_k - H\hat{v_{k-1}})$$

Esta es la ecuación principal para la actualización de medida del filtro discreto de Kalman. A la diferencia $(z_k - H\hat{v_{k-1}})$ se le llama usualmente innovación de la medida, el residuo

de esta operación refleja que tanta diferencia existe entre la predicción de medida y la medida real. La matriz K añadida es la ganancia que minimiza la co-varianza del error de estimación posterior, esta co-varianza se expresa como:

$$P_{k-1} = E[(e_{k-1})(e_{k-1}^T)] \qquad \text{Dado que} \qquad e_{k-1} = x_k - \hat{x}_{k-1}$$

Para realizar la minimización de la co-varianza a partir de la ganancia K se realiza la sustitución de $\hat{v_k}$ sobre y_{k-1} y se obtiene el respectivo valor esperado (Brown y Hwang (2012)), quedando:

$$K_k = y_{k-1}H^T(Hy_{k-1}H^T + R)^{-1}$$

Con esto se puede interpretar el comportamiento del peso K, si la co-varianza de la medida del error R se aproxima a cero, la medida actual z_k se vuelve mas confiable, mientras que la predicción de medida $H\hat{x}_{k-1}$ pierde fiabilidad. De forma opuesta si la co-varianza del error estimado a priori y_{k-1} se aproxima a cero entonces la medida actual z_k es menos confiable mientras que la predicción $H\hat{x}_{k-1}$ aumenta su fiabilidad.

El paso final para las ecuaciones de actualización de medida es el de obtener un estimado del error a posterior, esto se puede hacer a través de:

$$y_k = (1 - K_k H)y_{k-1}$$

Cada vez que se actualizan las ecuaciones de medida y predicción, los valores de los estados anteriores se actualizan con los actuales, de tal forma que el filtro se actualiza en tiempo actual de forma recursiva.

CAPÍTULO 5

Pruebas y resultados

A continuación se muestran los resultados de cada etapa del sistema planteado, partiendo de la obtención general del mapa de profundidad a través de estereoscopía. Posteriormente se presentan las pruebas realizadas para el reconocimiento de objetos a partir de distintos descriptores de características y, finalmente, se muestra un esquema general de los experimentos planteados para la estimación del movimiento del objeto y la aplicación de los mismos.

5.1. Obtención de mapa de profundidad

El primer paso es delimitar el área donde se ha encontrado el objeto, para almacenar la posición bi-dimensional del mismo y posteriormente proyectarla al mapa de profundidad que se obtendrá. El método de reconocimiento del objeto se aborda mas adelante. En la figura 5.1 se muestra un ejemplo de la región de interés generada por el reconocimiento de un objeto entrenado. Para mas información respecto a la generación de los patrones de entrenamiento referirse al *Apéndice B*.

Una vez obtenida la posición del objeto, se considera una región de interés (figura 5.2), operando dicha región con una máscara del tamaño de la misma por sobre la imagen original (figura 5.3), al realizar esto se puede obtener una proyección que represente el área a revisar en términos de distancia dentro del mapa de profundidad.

Figura 5.1: Imágenes obtenidas de la cámara, con la región de interés marcada

Figura 5.2: Imagen recortada, con la parte interna de la región de interés

Figura 5.3: Región de interés obtenida

Aplicando un recorte sobre la disparidad obtenida con las cámaras, el cual tendrá a

la región de interés como plantilla, podemos obtener solo el área de interés en términos de disparidad, para posteriormente convertir ésta en el mapa de profundidad que nos interesa. A partir del mapa se puede estimar la distancia del objeto reconocido a la cámara. La disparidad del objeto ejemplo se muestra en la figura 5.4, mientras que el mapa de profundidad se encuentra en la figura 5.5.

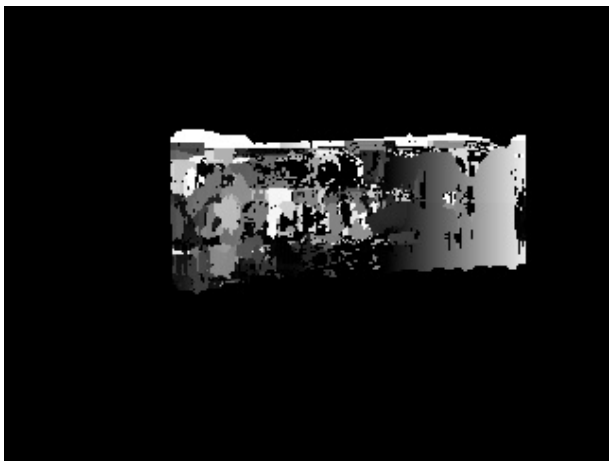


Figura 5.4: Disparidad obtenida

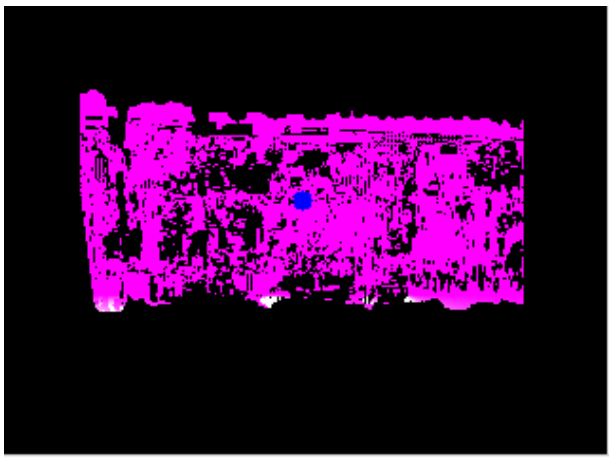


Figura 5.5: Mapa de profundidad

En el proceso de obtener la disparidad, es muy importante considerar la necesidad de filtrar las imágenes, que, como ya se mencionó previamente, en este desarrollo se optó por el filtro WLS (Weighted Least Squares). A continuación, en la figura 5.6, se muestra la diferencia entre aplicar el filtro y no hacerlo.

Figura 5.6: Disparidad obtenida
Izquierda: Imagen original izquierda.
Derecha inferior: Disparidad obtenida sin aplicar el filtro wls.
Derecha superior: Disparidad obtenida al aplicar el filtro wls.

Como se puede notar la mejora es sustancial y resulta una prioridad mantener esta etapa de procesamiento, ya que si por ejemplo el objeto de búsqueda en la escena fuese el termo de metal, en la disparidad sin filtrar no se muestra señal alguna de su existencia geométrica, mientras que en el caso de la disparidad filtrada resulta sencillo percibir visualmente la superficie de los objetos.

5.2. Detección de características

Para describir individualmente los objetos a reconocer, asi como para realizar la primera parte del empate estereoscópico, se optó por el uso de descriptores de características estándar como lo son SIFT, SURF y ORB. Después de varias pruebas se concluyó que debido a la necesidad de una implementación que fuese ligera en términos de procesamiento, el mejor descriptor para el trabajo sería ORB ya que es de tipo binario mientras que SIFT y SURF se basan en punto flotante y sus etapas de procesamiento en términos de cantidad poseen mas carga computacional. A continuación se muestran una serie de imágenes en las que se puede apreciar la cantidad de características que se obtienen por cada descriptor, pese a que SIFT y SURF obtienen más en la escena, visualmente se logra comprobar que varias de ellas resultan ser falsos positivos, sin mencionar el ya

planteado requisito de la carga de procesamiento (A mas características mayor tiempo de procesamiento).

- SIFT

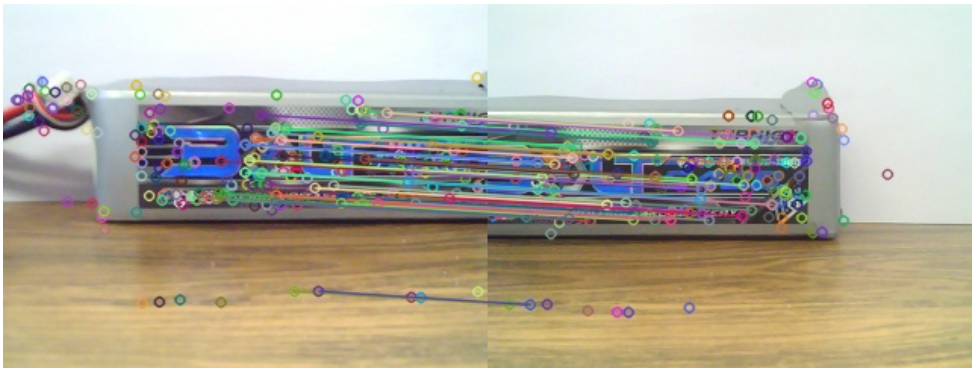


Figura 5.7: Empatado estereoscópico de características con descriptores SIFT

- SURF

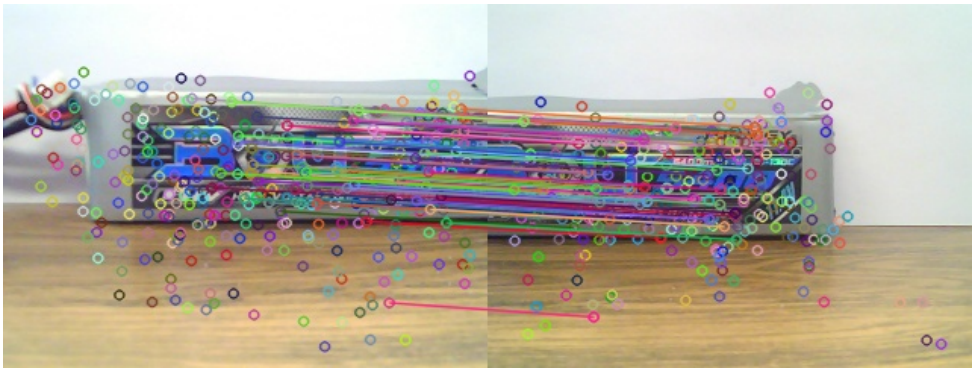


Figura 5.8: Empatado estereoscópico de características con descriptores SURF

- ORB

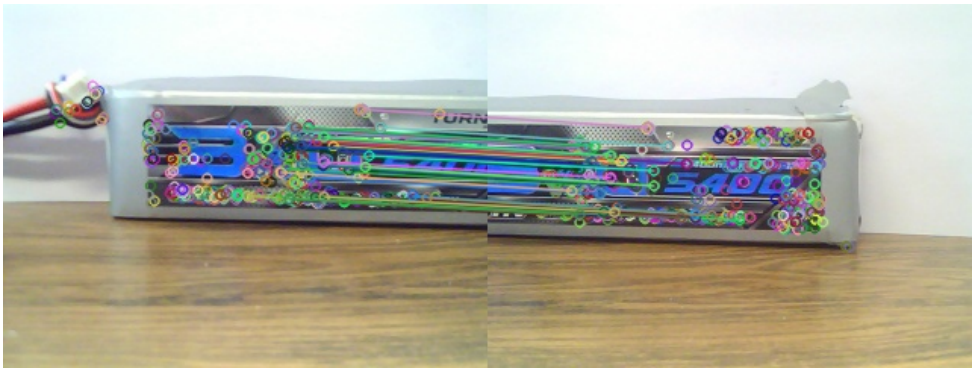


Figura 5.9: Empatado estereoscópico de características con descriptores ORB

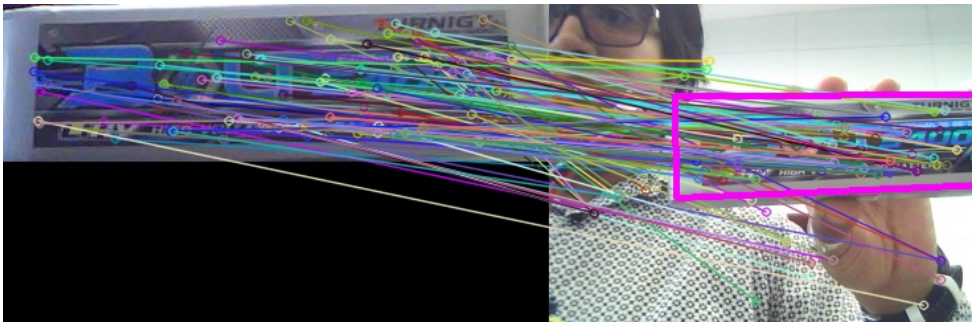


Figura 5.10: Reconocimiento de objeto respecto a características en imagen izquierda

Figura 5.11: Reconocimiento de objeto respecto a características en imagen derecha

5.3. Pruebas de movimiento

Además de los resultados arrojados a partir de cada modulo del sistema, se plantearon dos pruebas sobre el sistema total:

- Estimación de la posición de un objeto móvil en un eje:
  En esta prueba se plantea verificar la precisión y exactitud de las medidas obtenidas por el sistema de visión, para esto se utilizó un robot móvil con desplazamiento en un solo eje. Dicho robot fue visualizado desde una perspectiva superior y se almacenó de forma constante su propio movimiento con la ayuda de un acelerómetro y un codificador rotatorio. Después de realizar la prueba se procedió a empatar las mediciones tanto del sistema de visión como lo obtenido por el robot mismo.

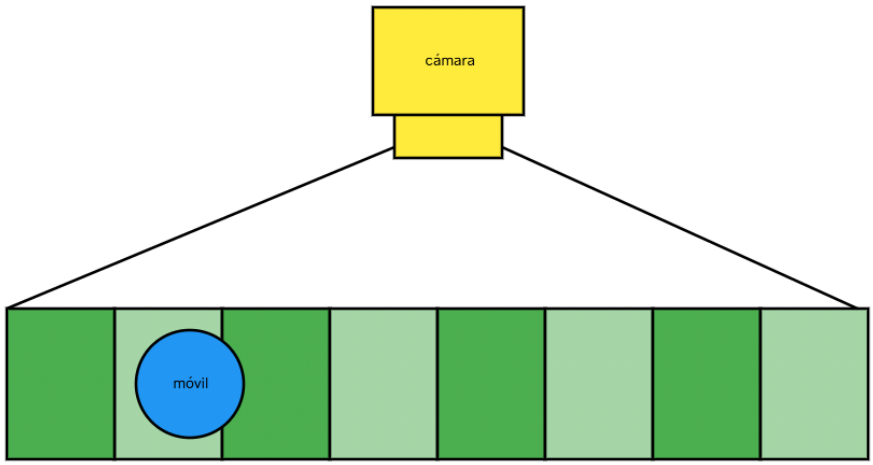


Figura 5.12: Ambiente planteado para la prueba de la posición y velocidad en un solo eje

  Esta primer prueba se utilizó también para calibrar el sistema de desplazamiento, ya que el eje fue marcado por una linea métrica, en la cual se verificó visualmente que el desplazamiento medido por el robot fuese coherente, para mas información respecto al como se realizaron las medidas por parte del robot referirse al *Apéndice A, sección 2.*

- Estimación de la posición de objetos en desplazamiento desde perspectiva estática:
  En esta prueba se planteo el uso de un robot móvil dentro de un ambiente conocido, junto con objetos adicionales, los cuales se desplazaron sobre un plano y fueron seguidos a partir de un punto de vista superior al mismo. El experimento se puede plantear de la siguiente forma:

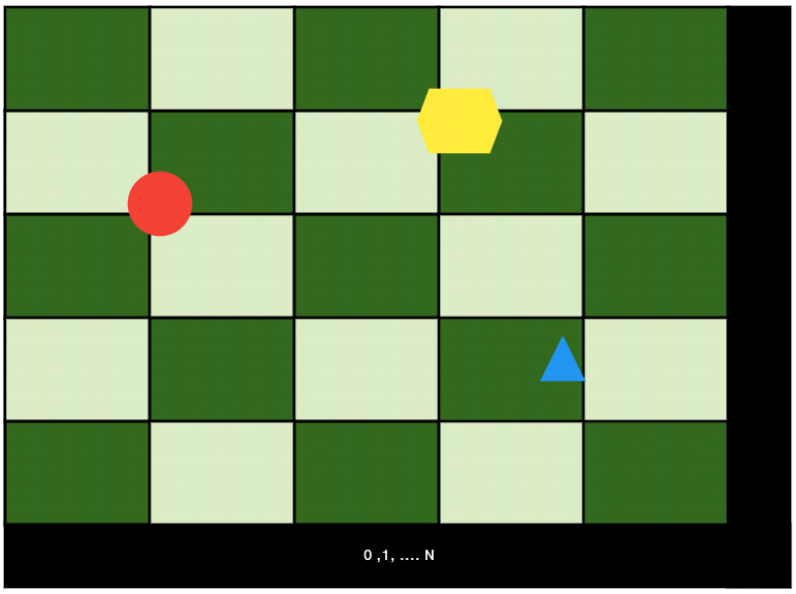


Figura 5.13: Ambiente planteado para la prueba de la posición y velocidad

El robot utilizado para ambos escenarios se muestra en la imagen 5.14, para mayor información respecto a su composición referirse al *Apéndice A*.

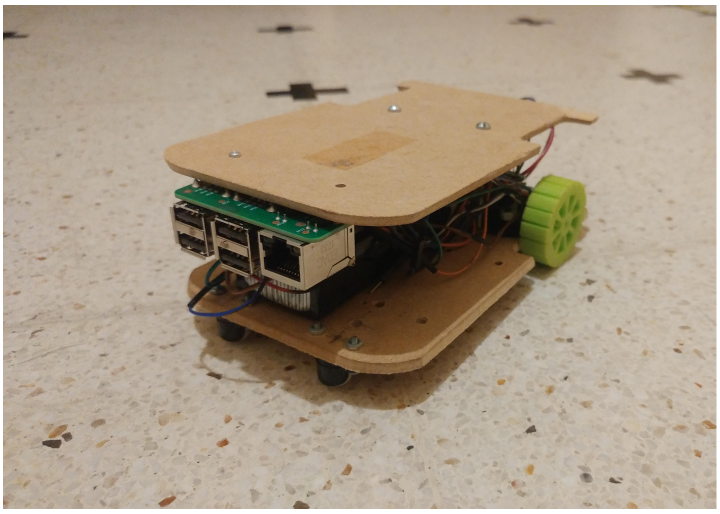
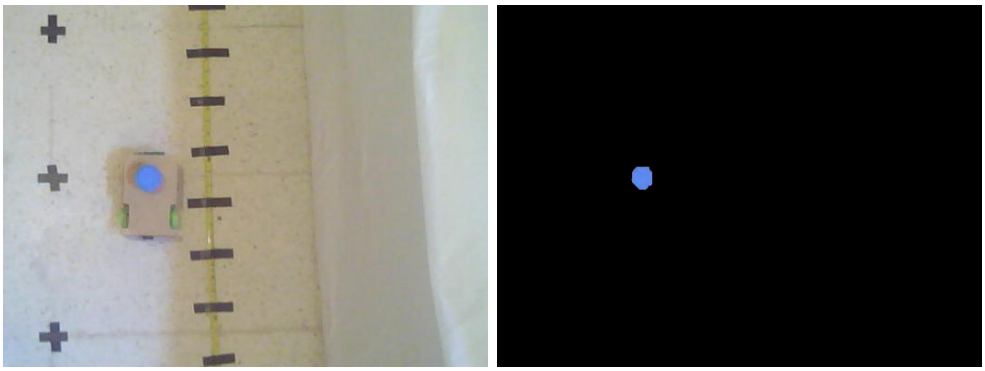


Figura 5.14: Robot utilizado para las pruebas de estimación planteadas

Las mediciones de salida se realizaron empatando el movimiento proporcionado por el acelerómetro del robot y la calculada por el sistema de visión computacional. Pese a que el ambiente de prueba consta de marcadores, estos se utilizaron solamente para calibrar

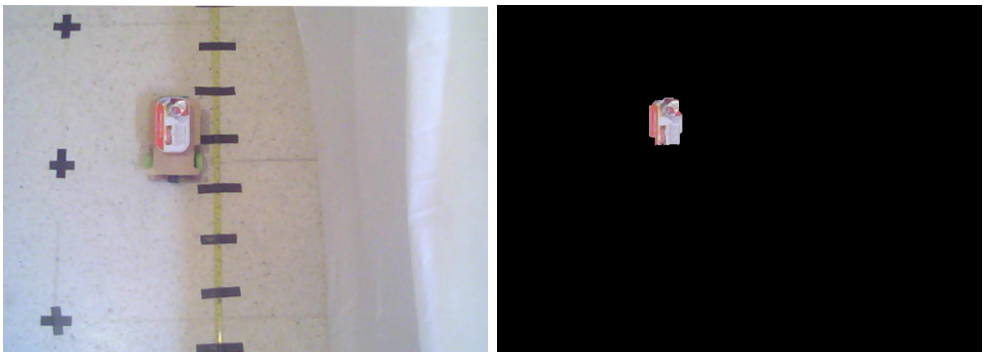
el sistema de medición incorporado en el robot para así evitar cualquier incoherencia en los datos de salida.

En términos prácticos, la configuración final de las pruebas realizadas, así como los resultados de salida para la máscara de movimiento, quedó de la siguiente forma:



(a) Imagen de entrada para cámara derecha. *(b) Salida de detección*

Figura 5.15: Prueba sobre un solo eje (Perspectiva Horizontal).

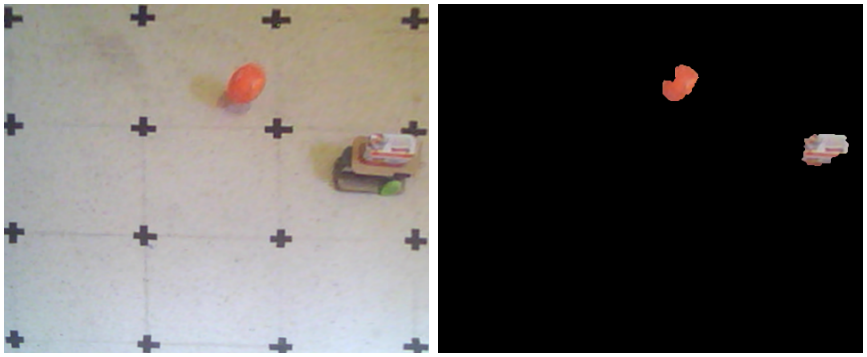


(a) Imagen de entrada para cámara derecha. *(b) Salida de detección*

Figura 5.16: Prueba sobre un solo eje (Perspectiva Horizontal).

(a) Imagen de entrada para cámara derecha. *(b) Salida de detección*

Figura 5.17: Prueba sobre un solo eje (Perspectiva Vertical).



(a) Imagen de entrada para cámara derecha. *(b) Salida de detección*

Figura 5.18: Estimación de varios objetos sobre mismo plano.

(a) Imagen de entrada para cámara derecha. *(b) Salida de detección*

Figura 5.19: Estimación de varios objetos sobre mismo plano.

5.3.1. Análisis de datos obtenidos

Las pruebas anteriores se utilizaron como metodo de medida para la cantidad obtenible de cuadros por segundo, velocidad máxima calculada y el promedio de error en la proyección de distancia calculada, las pruebas se realizaron sobre 3 diferentes unidades de procesamiento:

1. Broadcom BCM2837 4 x ARM Cortex A53, Quad Core 1.2GHz
2. Intel Core i3-6100U, Dual Core 2.3GHz
3. Intel Core i7-3615QM, Quad Core 2.3GHz

En las tablas 5.2, 5.3 y 5.1 se muestran los resultados obtenidos de cada combinación de pruebas, los datos mostrados para cada celda son el producto de un promedio de 200 repeticiones por cada combinación procesador-resolución, para obtener estas repeticiones se realizaron un mínimo de 5 ejecuciones sobre los ambientes planteados previamente, en cada una de esas ejecuciones se procuraron tomar 40 medidas teniendo así un total de 200. En los caso donde las medidas no lograron juntarse en 5 ejecuciones se aumentó la cantidad de éstas y por tanto, también de la toma de muestras.

Además de la tabla de datos muestreados, presento un análisis de los resultados obtenidos por cada rubro, en torno a la resolución utilizada en las pruebas:

1. Cantidad máxima de cuadros obtenidos por segundo (tabla 5.1)
2. Velocidad mínima calculada para los movimientos (tabla 5.2)
3. Promedio de error en el calculo de la distancia (tabla 5.3)

Posteriormente presento un análisis de estos mismos parámetros pero en torno a los procesadores utilizados.

Después de cada gráfica se explica brevemente el significado de los resultados representados en las mismas.

Resolución	Procesador utilizado	Cantidad de cuadros por segundo
640x480(Minoru 3D)	Intel Core i7 3615QM	27
640x480(Minoru 3D)	Intel Core i3 6100U	24
640x480(Minoru 3D)	Broadcom BCM2837 4 x ARM Cortex A53	21
672x376 (ZED)	Intel Core i7 3615QM	48
672x376 (ZED)	Intel Core i3 6100U	43
672x376 (ZED)	Broadcom BCM2837 4 x ARM Cortex A53	27
800x600(Minoru 3D)	Intel Core i7 3615QM	29
800x600(Minoru 3D)	Intel Core i3 6100U	28
800x600(Minoru 3D)	Broadcom BCM2837 4 x ARM Cortex A53	19
1280x720 (ZED)	Intel Core i7 3615QM	16
1280x720 (ZED)	Intel Core i3 6100U	20
1280x720 (ZED)	Broadcom BCM2837 4 x ARM Cortex A53	14
1920x1080 (ZED)	Intel Core i7 3615QM	6
1920x1080 (ZED)	Intel Core i3 6100U	9
1920x1080 (ZED)	Broadcom BCM2837 4 x ARM Cortex A53	6

Cuadro 5.1: Datos obtenidos de los experimentos del apartado anterior, resultados respecto a cuadros por segundo

Resolución	Procesador utilizado	Velocidad promedio mínima detectada (cm/s)
640x480(Minoru 3D)	Intel Core i7 3615QM	4
640x480(Minoru 3D)	Intel Core i3 6100U	4
640x480(Minoru 3D)	Broadcom BCM2837 4x ARM Cortex A53	4
672x376 (ZED)	Intel Core i7 3615QM	2
672x376 (ZED)	Intel Core i3 6100U	4
672x376 (ZED)	Broadcom BCM2837 4x ARM CortexA53	9
800x600(Minoru 3D)	Intel Core i7 3615QM	9
800x600(Minoru 3D)	Intel Core i3 6100U	9
800x600(Minoru 3D)	Broadcom BCM2837 4x ARM CortexA53	16
1280x720 (ZED)	Intel Core i7 3615QM	22
1280x720 (ZED)	Intel Core i3 6100U	16
1280x720 (ZED)	Broadcom BCM2837 4x ARM CortexA53	22
1920x1080 (ZED)	Intel Core i7 3615QM	30
1920x1080 (ZED)	Intel Core i3 6100U	28
1920x1080 (ZED)	Broadcom BCM2837 4x ARM CortexA53	30

Cuadro 5.2: Datos obtenidos de los experimentos del apartado anterior, resultados respecto a velocidad

Resolución	Procesador utilizado	Distancia máxima obtenida (cm)	Tolerancia promedio por medición del robot y cámara (%)
640x480(Minoru 3D)	Intel Core i7 3615QM	120(x) 80(y) 120(z)	3(x) 2(y) 13(z)
640x480(Minoru 3D)	Intel Core i3 6100U	120(x) 80(y) 120(z)	3(x) 3(y) 13(z)
640x480(Minoru 3D)	Broadcom BCM2837 4x ARM Cortex A53	100(x) 80(y) 120(z)	5(x) 3(y) 16(z)
672x376 (ZED)	Intel Core i7 3615QM	100(x) 65(y) 80(z)	6(x) 5(y) 11(z)
672x376 (ZED)	Intel Core i3 6100U	100(x) 65(y) 80(z)	6(x) 5(y) 13(z)
672x376 (ZED)	Broadcom BCM2837 4x ARM CortexA53	100(x) 65(y) 80(z)	7(x) 4(y) 14(z)
800x600(Minoru 3D)	Intel Core i7 3615QM	180(x) 130(y) 240(z)	3(x) 2(y) 9(z)
800x600(Minoru 3D)	Intel Core i3 6100U	180(x) 130(y) 240(z)	3(x) 3(y) 9(z)
800x600(Minoru 3D)	Broadcom BCM2837 4x ARM CortexA53	110(x) 110(y) 200(z)	5(x) 3(y) 10(z)
1280x720 (ZED)	Intel Core i7 3615QM	150(x) 140(y) 220(z)	11(x) 11(y) 7(z)
1280x720 (ZED)	Intel Core i3 6100U	150(x) 140(y) 220(z)	11(x) 11(y) 6(z)
1280x720 (ZED)	Broadcom BCM2837 4x ARM CortexA53	110(x) 110(y) 240(z)	17(x) 17(y) 9(z)
1920x1080 (ZED)	Intel Core i7 3615QM	240(x) 150(y) 350(z)	14(x) 14(y) 4(z)
1920x1080 (ZED)	Intel Core i3 6100U	240(x) 150(y) 350(z)	15(x) 15(y) 3(z)
1920x1080 (ZED)	Broadcom BCM2837 4x ARM CortexA53	240(x) 150(y) 320(z)	19(x) 19(y) 9(z)

Cuadro 5.3: Datos obtenidos de los experimentos del apartado anterior, resultados respecto a distancia calculada

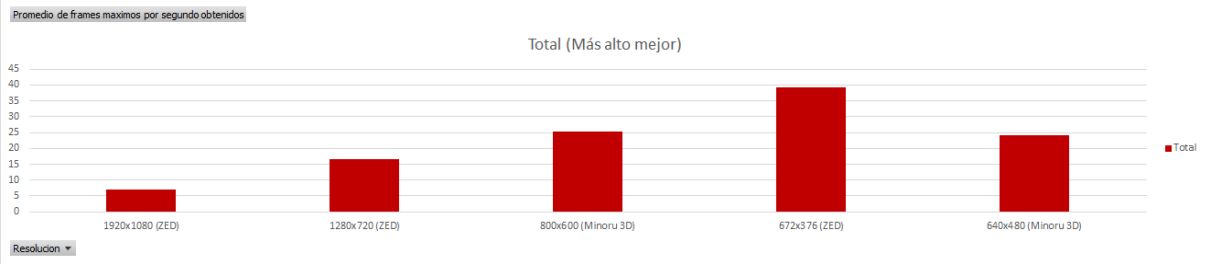


Figura 5.20: Resultados obtenidos en Resolución vs Promedio máximo de cuadros por segundo

En esta gráfica se presenta por cada resolución, ponderando en cada una los 3 procesadores utilizados, el promedio de cuadros por segundo máximos obtenibles, en este caso la mejor opción es la que presente el valor mas grande, debido a que entre más cuadros se obtengan más fluido sera el movimiento obtenido y por tanto en general habrá una mejor estimación de los movimientos. Cabe destacar que una restricción del algoritmo se presenta debido al hecho de que es necesario acumular cierta cantidad de cuadros para inicializar el sistema, por lo que muy pocos cuadros no realizaran un procesamiento fluido debido a esta inicialización.

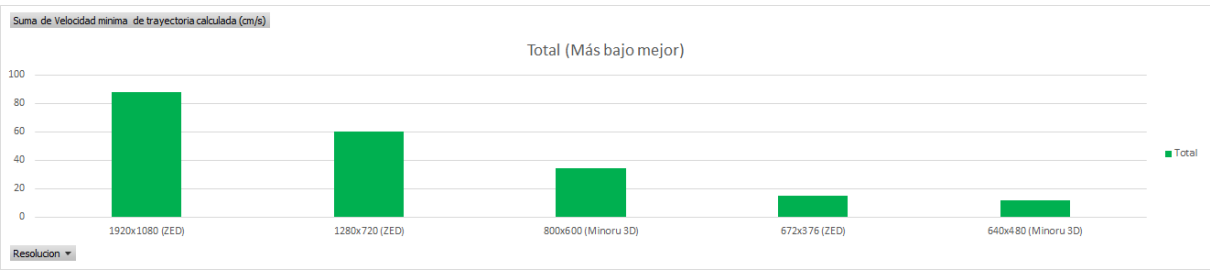


Figura 5.21: Resultados obtenidos en Resolución vs Promedio de velocidad mínima calculada para movimientos

En este caso la variable medida es el promedio de la velocidad mínima calculable para los movimientos, similar a la gráfica anterior en cada promedio se incorporan los 3 procesadores utilizados para las pruebas, en este caso el mejor resultado es el mas bajo, ya que se busca que la estimación del movimiento sea lo mas sensible posible respecto a los cambios de orientación y esto se logra detectando movimientos mas pequeñas.

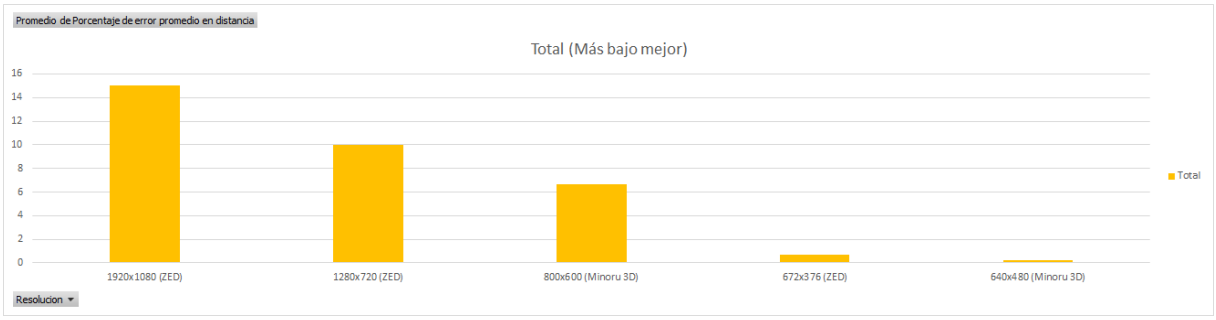


Figura 5.22: Resultados obtenidos en Resolución vs Promedio de error en proyección de distancia

En esta tercer gráfica se presenta el promedio de error de una proyección de distancia, es decir, el error obtenido en promedio al realizar continuamente la medida de distancia entre el objeto y la cámara. Dicha distancia se obtiene a partir de estereoscopía.

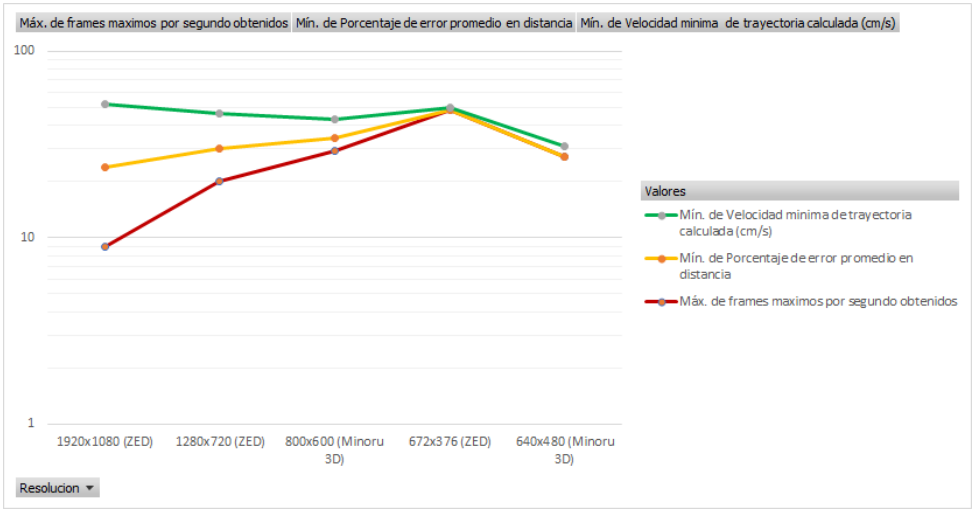


Figura 5.23: Gráfica logarítmica de resultados por resolución utilizada, usando datos de mejor caso

En este caso se muestran nuevamente los 3 parámetros graficados anteriormente, esta vez en la misma gráfica, debido a esto la escala utilizada es logarítmica. Además, se realizaron ciertas consideraciones para que esta gráfica representara el mejor caso posible de los datos, ya que en las gráficas anteriores lo mostrado era el promedio general por cada resolución.

Gracias a la sobreposición de los resultados y su escalamiento, podemos ver que a menor resolución mejora el porcentaje de error de la distancia calculada, sin embargo tanto la velocidad máxima calculable como la máxima distancia no obtienen mejores resultados a mejores resoluciones, al comparar gráficamente la relación de las tres variables se puede

notar que el resultado que mejor cumple con los mejores parámetros obtenidos es el de una resolución intermedia, a $672x376$ píxeles.

En la tabla 5.4 se muestra el valor de cada parámetro tanto para el mejor como para el peor caso.

Parámetro	Valor mejor caso	Valor peor caso
Porcentaje de error en distancia	Mínimo	Máximo
Velocidad mínima de movimientos	Mínimo	Máximo
Cuadros por segundo máximos	Máximo	Mínimo

Cuadro 5.4

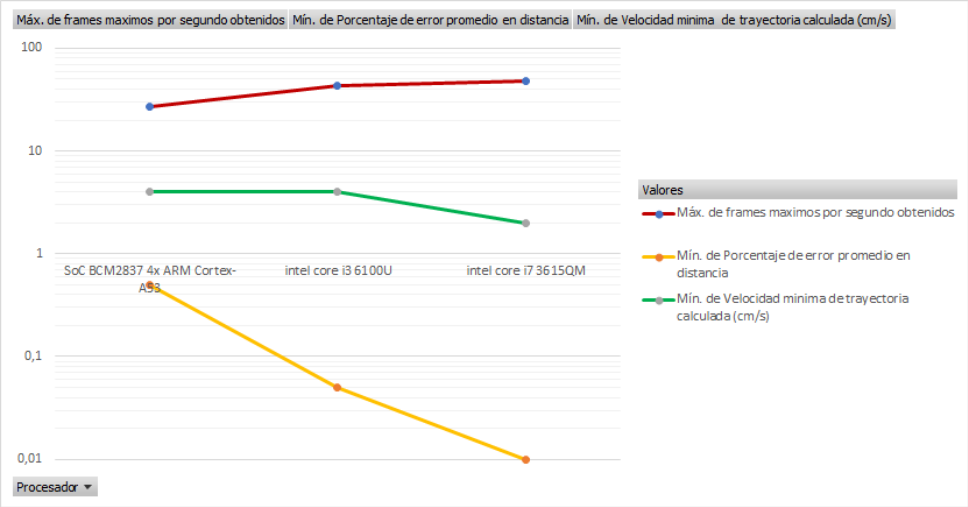


Figura 5.24: Gráfica logarítmica de resultados por procesador utilizado, usando datos de mejor caso

Esta gráfica también representa el mejor caso, solo que desde la perspectiva de cada uno de los procesadores utilizados, una interpretación sencilla de este resultado es que cada parámetro mejora a mayor procesamiento, siendo más beneficiada la cantidad de cuadros adquiridos, siguiendo la velocidad calculada y finalmente el porcentaje de error.

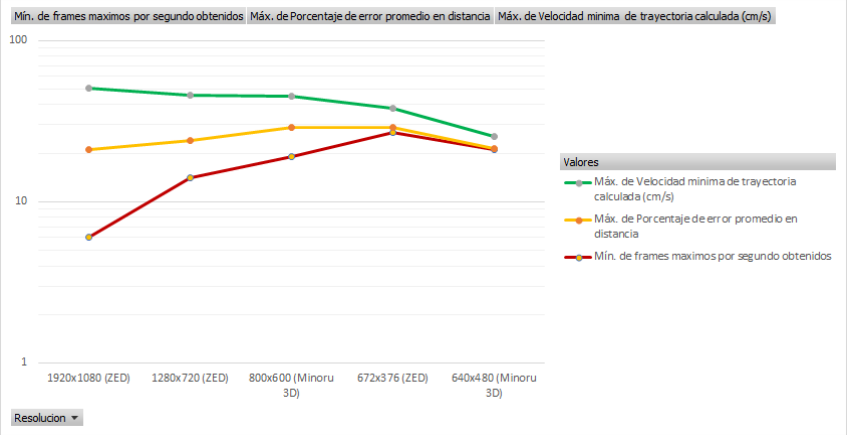


Figura 5.25: Gráfica logarítmica de resultados por resolución utilizada, usando datos de peor caso

Esta gráfica representa el peor caso para cada uno de los parámetros respecto a la resolución, se logra notar que el porcentaje de error y la cantidad de cuadros por segundo obtenidos no sufren un cambio significativo en su comportamiento respecto al mejor caso, sin embargo el comportamiento de la velocidad de movimiento calculada resulta muy afectada, mostrando que una mejora en la resolución no significa necesariamente una mejora en la velocidad calculada.

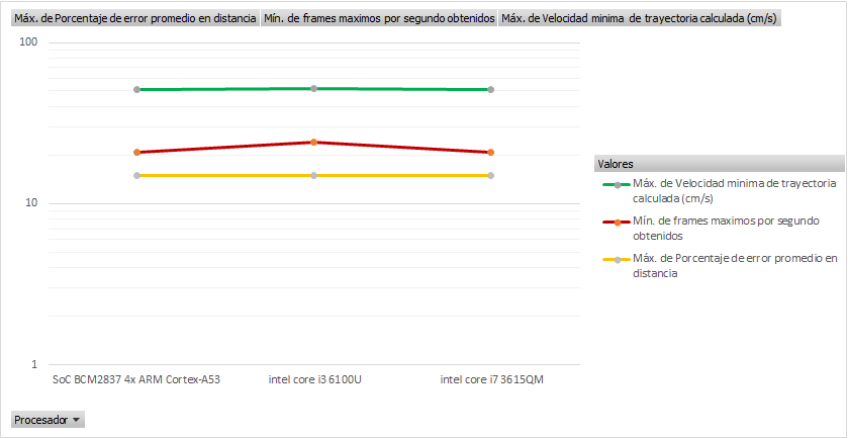


Figura 5.26: Gráfica logarítmica de resultados por procesador utilizado, usando datos de peor caso

La gráfica del peor caso para el uso de procesadores nos muestra que el valor de cada parámetro puede llegar a mantenerse de forma constante en su mínimo, por lo que se puede decir que el mejor indicador del desempeño obtenible se encuentra vinculado mayormente a la resolución utilizada y no al procesador en el que se implemente el algoritmo realizado en este trabajo.

CAPÍTULO 6

Conclusiones y trabajo futuro

6.1. Conclusiones

Con este trabajo de tesis se comprobó que es posible realizar un programa lo suficientemente ligero para ser ejecutado en un sistema embebido, esto se implemento en una tarjeta electrónica Raspberry Pi de forma exitosa, pues con dicho programa se obtuvieron medidas útiles para caracterizarlo. El algoritmo logro diseñarse para ser aplicable con cualquier par de cámaras y formar visión estereoscópica con estas, mas adelante se discuten los resultados pertinentes al proceso de empate estereoscópico. En términos del éxito en el seguimiento de objetos a partir de una predicción, así como de la obtención de su posición, resultó ser relativamente bueno, ya que aunque ambas tareas son realizables resultaron ser relativamente opuestas en términos de requisitos e implementación, ya que a mayor resolución la estereoscopía mejoraba, pero el seguimiento de los objetos resultaba entorpecido.

También se comprobó que aunque el mapa de profundidad obtenido a partir del empate estereoscópico en las cámaras suele tener una componente de ruido un poco más alta respecto a cámaras de tiempo de vuelo o escáneres de luz estructurada, el método de detección de movimiento resulta mucho mas rápido, ya que una de las ventajas más grandes al utilizar cámaras individuales para generar un mapa de profundidad es el hecho de que el mapa puede construirse hasta que éste sea realmente necesario, como por ejemplo en el instante de medir la distancia hacia el objeto de interés, ahorrando así tiempo de procesamiento al realizar la búsqueda dentro de un espacio tridimensional.

Resulta lógico el hecho de que este método sea mas rápido y, en la mayoría de los casos, mas preciso que los de seguimiento a partir de detección si consideramos el campo de la visión estereoscópica, ya que si la prioridad es discernir un objeto ontológico entonces se

requiere procesar primeramente las características de la escena y posteriormente conocer si estas se encuentran en proceso de movimiento; en cambio, si primero se procesa individualmente la interacción entre los cuadros provistos por las cámaras, se pueden obtener primero las áreas en proceso de movimiento y posteriormente que objetos se encuentran dentro de dichas áreas. En el caso de la detección del objeto ya en movimiento este resulta ser bastante ligero en tiempo de ejecución y sin tanto error en términos de la posición geométrica detectada (tal y como se muestra en las tablas y gráficas de resultados), sin embargo en caso de requerir una descripción basada en características mas especificas entonces lo conveniente seria complementar este algoritmo con uno de seguimiento por detección.

La etapa más sensible de este algoritmo resulta ser el empate estereoscópico, para el cual concluyo que hay 3 alternativas para realizar más rápidamente un mapa de profundidad a partir de estereoscopía, cada una con diferentes ventajas:

- Realizar la rectificación de las cámaras solo en términos de los parámetros extrínsecos, para posteriormente filtrar el mapa de profundidad. En este caso la generación del mapa resulta ser bastante rápida y el resultado otorga una idea bastante buena del volumen y posición de los objetos, sin embargo el escalamiento de la profundidad obtenida tendrá algunas aberraciones, si lo que se busca es la referencia entre objetos no es mucho problema, pero si el interés está en la distancia al objeto entonces no es muy recomendable en general y para nada recomendable si se utilizan 2 cámaras diferentes para la visión estereoscópica.
- Utilizar características globales por sobre semi-globales. Posterior a la rectificación se debe realizar el empate por algún método ya sea global o semi-global, en pruebas se corroboró la teoría de que en general resultan más rápidos los métodos globales, aunque siempre habrá una pérdida de resolución en términos de cuanto detalle se obtiene en el mapa de profundidad (Kanade y Okutomi (1994)).
- Utilizar una cámara que ya tenga un SDK desarrollado para la obtención del mapa. Éste es el caso de la cámara ZED ya que aunque el dispositivo en si no proporciona directamente el mapa de profundidad, las librerías que ésta incluye realizan la tarea rápidamente. El problema de esto es que resulta una solución muy especifica al hardware utilizado, pues por ejemplo en el caso de la cámara ZED nuevamente, se requiere uso de GPU.

A partir de las pruebas realizadas se puede concluir sobre los aspectos mas críticos del algoritmo en particular, por lo que a continuación enumero dichas Conclusiones:

1. Resolución en la detección:
   En general se podría decir que se logró una buena resolución de detección mínima de

movimiento gracias al uso de operadores morfológicos para filtrar la escena, sin embargo, la detección de objetos muy grandes tiene de cierta forma un efecto adverso sobre la imagen, ya que aunque en efecto son detectados, la región de interés generada debido al movimiento de los mismos resulta ser muy grande pues se considera dentro del movimiento la posición actual/posible del objeto y su posición anterior. Una forma de evitar este problema sería mejorando la discriminación entre fondo y objeto, pudiendo reducir aún más la región de interés. Respecto a la identificación del fondo, cabe destacar que muy probablemente seria de gran ayuda realizar primero una segmentación por flujo óptico (Liu et al. (1997)), pudiendo así separar regiones con diferente cuantificación de movimiento aunque éste sea constante en ambas.

2. Velocidad de detección: El algoritmo depende de principalmente 3 aspectos en este rubro, la velocidad de computo y la velocidad de adquisición por parte del hardware, dado que se logró obtener buenos resultados sobre un sistema embebido con capacidades relativamente reducidas, el escalarlo a un hardware mas potente debería presentar una mejora directa en la velocidad de detección, sin embargo hay que considerar que también se requiere un hardware de adquisición mas ad-hoc. Debido a cuestiones de disposición de hardware este algoritmo solo se probó con cámaras que poseen tecnología rolling-shutter (lo que a su vez lo hace más compatible en general), pero se sabe que una cámara global-shutter presentaría menos aberraciones entre cuadro (Saurer et al. (2013)), mejorando también los errores por posible ruido.

3. Discriminación entre objetos y fondo: Resulta bastante necesario la discriminación del fondo de los objetos en movimiento, pese a que en un principio se creyó que no presentaría un mayor problema resulta ser muy complicado en términos de textura e iluminación cuando tanto el objeto como la cámara se desplazan, eso sin mencionar los posibles errores por ruido debido a una lenta adquisición de datos. Si solo se detectan objetos desde una perspectiva fija esto no presenta ningún inconveniente para el algoritmo en general. La forma más óptima de lidiar con este rubro parece ser la contextualización de la escena.

6.2. Trabajo a futuro

Considero que se puede aligerar la carga del cálculo de la estimación de la posición siguiente, tanto en su seguimiento como en la definición de la región de interés si se opta por representar el desplazamiento del objeto de una forma diferente a la vectorial, en lo personal sugiero utilizar una representación tridimensional por códigos de cadena (Bribiesca (2000)), de esta forma sería mas fácil obtener una estadística del movimiento a partir del número de repeticiones en cada dirección, cabe destacar que aunque en general

las operaciones serían más rápidas y sencillas, el optar por una representación como esta significaría discretizar aun más la posición respecto a un desplazamiento continuo ya que los valores de movimiento solo podrían considerar opciones pre-definidas, a continuación presento una imagen basada en una descripción del desplazamiento por códigos de cadena con 4 direcciones definidas a *priori*.

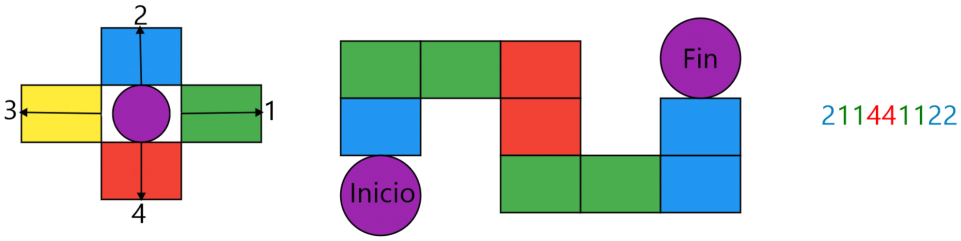


Figura 6.1: Izquierda: direcciones a considerar y su respectiva referencia numérica. Centro: Ejemplo de trayectoria descrita por el objeto circular. Derecha: Código de cadena resultante de la trayectoria descrita

Otro trabajo futuro a modo de extensión del actual puede resultar de aplicar sobre el algoritmo un método de diferenciación entre objetos en movimiento y posibles referencias espaciales, con esto se obtendría directamente una implementación de odometría visual, lo que a su vez podría escalarse en un metodo de SLAM. Las modificaciones para añadir a éste algoritmo capacidades de odometría visual deben integrarse en la parte de detección y almacenamiento de las posiciones, también es necesario añadir un módulo que proyecte como referencia todo el movimiento hacia un punto clave conocido. Sobre el primer requisito, detección, se debe extender la capacidad en objetos estáticos utilizando un detector de características aumentado, aunque esto signifique requerir una mayor capacidad de computo ya que el algoritmo actual perderá en gran parte su propiedad de ser ligero en procesamiento. Respecto a la necesidad de cambiar la referencia hacia un punto conocido, el añadir esta funcionalidad no significa ningún cambio al algoritmo actual, ya que la medición de las posiciones de salida, así como la re-alimentación de la región de interés, son accesibles por variables, por lo que el cálculo de proyección se puede realizar por un programa o funciones externas.

6.3. Recomendaciones de implementación

Como se muestra en el desarrollo y experimentación, el método de detección de objetos en movimiento descrito en este trabajo resulta muy dependiente del descriptor de características implementado así como del filtrado del mapa de disparidad, para el caso del descriptor de características es altamente recomendable que este sea binario debido a la velocidad de detección y entrenamiento, en caso de necesitar descriptores en punto flotante es indispensable almacenar los vectores de entrenamiento y utilizar una estructura para navegar mas rápidamente como por ejemplo en árboles.

Respecto al filtrado del mapa de disparidad resulta menos costoso en tiempo de ejecución utilizar un filtro basado en pesos, como el "wls" utilizado, que los basados en flujo óptico. Si se posee el suficiente poder de cómputo y la velocidad a detectar no es la máxima alcanzable en la menor resolución entonces la implementación del filtro para disparidad no resulta tan crítico.

Bibliografía

Bay, H., T. Tuytelaars, y L. V. Gool (2006), "Surf: Speeded up robust features." En *Computer Vision – ECCV 2006* (Aleš Leonardis, Horst Bischof, y Axel Pinz, eds.), 404–417, Springer Berlin Heidelberg, Berlin, Heidelberg.

Bribiesca, E. (2000), "A chain code for representing 3d curves." *Pattern Recognition*, 33, 755–765.

Brown, R. y P. Hwang (2012), *Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises*, 4a edicion. Jhon Wiley and Sons Inc, New York.

Calonder, M., V. Lepetit, C. Strecha, y P. Fua (2010), "Brief: Binary robust independent elementary features." En *Computer Vision – ECCV 2010* (Kostas Daniilidis, Petros Maragos, y Nikos Paragios, eds.), 778–792, Springer Berlin Heidelberg, Berlin, Heidelberg.

Cover, T. y P. Hart (1967), "Nearest neighbor pattern classification." *IEEE Transactions on Information Theory*, 13, 21–27.

Elad, M. (2002), "On the origin of the bilateral filter and ways to improve it." *IEEE Transactions on Image Processing*, 11, 1141–1151.

Gonzalez, R. y R. Woods (2018), *Digital Image Processing*, 4a edicion. Pearson.

Haralick, R., S.Sternberg, y X. Zhuang (1987), "Image analysis using mathematical morphology." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9, 532–550.

Hariharakrishnan, K. y D. Schonfeld (2005), "Fast object tracking using adaptive block matching." *IEEE Transactions on Multimedia*, 7, 853–859.

Iocchi, L. (1998), "Stereo vision: Triangulation." URL `www.dis.uniroma1.it/~iocchi/stereo/triang.html`.

Jähne, B. (2005), *Digital Image Processing*, 6a edicion. Springer-Verlag Berlin Heidelberg.

Kanade, T. y M. Okutomi (1994), "A stereo matching algorithm with an adaptive window: theory and experiment." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16, 920–932.

Konolige, K. (1998), "Small vision systems: Hardware and implementation." En *Robotics Research* (Yoshiaki Shirai y Shigeo Hirose, eds.), 203–212, Springer London, London.

Konolige, K., M. Agrawal, R. Bolles, C. Cowan, M. Fischler, y B. Gerkey (2008), *Outdoor Mapping and Navigation Using Stereo Vision*, 179–190. Springer Berlin Heidelberg, Berlin, Heidelberg, URL `https://doi.org/10.1007/978-3-540-77457-0_17`.

Lim, J. (1990), *Two-dimensional Signal and Image Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Lindeberg, T. (1994), "Scale-space theory: A basic tool for analysing structures at different scales." *Journal of Applied Statistics*, 21, 224–270.

Liu, H., T. Hong, M. Herman, y R. Chellappa (1997), "A general motion model and spatio-temporal filters for computing optical flow." *International Journal of Computer Vision*.

Liu, W., J. Principe, y S. Haykin (2010), *Kernel Adaptive Filtering: A Comprehensive Introduction*. Wiley.

Lowe, D. (2004), "Distinctive image features from scale-invariant keypoints." *International Journal of Computer Vision*, 60, 91–110, URL `https://doi.org/10.1023/B:VISI.0000029664.99615.94`.

Mezaris, V., I. Kompatsiaris, y M. G. Strintzis (2004), "Video object segmentation using bayes-based temporal tracking and trajectory-based region merging." *IEEE Transactions on Circuits and Systems for Video Technology*, 14, 782–795.

Mikolajczyk, K. y C. Schmid (2001), "Indexing based on scale invariant interest points." En *Computer Vision – ICCV*, volumen 1, 525–531.

Moyung, T. y P. Fieguth (2000), "Incremental shape reconstruction using stereo image sequences." En *Proceedings 2000 International Conference on Image Processing (Cat. No.00CH37101)*, volumen 2, 752–755 vol.2.

OpenCV (2014), "Referencia OpenCV 3." `https://docs.opencv.org/3.0-beta/modules/refman.html`.

Pratt, W. (2001), *Digital Image Processing: PIKS Inside*, 3a edicion. Jhon Wiley and Sons Inc, New York.

Proakis, J. y D. Manolakis (1996), *Digital Signal Processing (3rd Ed.): Principles, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Rosten, E. y T. Drummond (2006), "Machine learning for high-speed corner detection." En *Computer Vision – ECCV 2006* (Aleš Leonardis, Horst Bischof, y Axel Pinz, eds.), 430–443, Springer Berlin Heidelberg, Berlin, Heidelberg.

Rublee, E., V. Rabaud, K. Konolige, y G. Bradski (2011), "Orb: An efficient alternative to sift or surf." En *2011 International Conference on Computer Vision*, 2564–2571.

Saurer, O., K. Köser, J. Bouguet, y M. Pollefeys (2013), "Rolling shutter stereo." En *2013 IEEE International Conference on Computer Vision*, 465–472.

Stein, G. P. y A. Shashua (1998), "Direct estimation of motion and extended scene structure from a moving stereo rig." En *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*, 211–218.

Tazi, S. N. y V. K. Jain (2014), "Enhance matching in multi-dimensional image reconstruction using stereo image sequences." En *2014 International Conference on Information Systems and Computer Networks (ISCON)*, 33–38.

Wan, Y. y Z. Miao (2009), "3d scene reconstruction based on uncalibrated image sequences." En *2009 International Conference on Digital Image Processing*, 163–166.

Witkin, A.P. (1983), "Scale-space filtering." *International Joint Conference on Artificial Intelligence, Karlsruhe, Germany*, 1019–1022.

Zhang, Z., R. Deriche, O. Faugeras, y Q. Luong (1995), "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry." En *Elsevier, Artificial Inteligence Volume 78, Isues 1-2*, 87–119.

Apéndices

APÉNDICE A

Robot móvil usado en pruebas

A.1. Componentes del robot

El robot utilizado para verificar los experimentos mencionados en este trabajo se detalla a continuación, esperando sea útil en caso de necesitar reproducir los resultados obtenidos.

A.1.1. Componentes electrónicos

Función	Descripción del componente utilizado	Referencia
Controlador	Raspberry Pi 3 Modelo B Ver. 1.2	https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
Sensor Acelerómetro	Acelerómetro y Brújula 3D LSM303D	https://www.pololu.com/product/2127
Sensor codificador rotatorio	codificador rotatorio óptico con 5 dientes	https://www.pololu.com/product/2590
Regulador de Voltaje	Regulador de subida y bajada a 6V 2A S18V20F6	https://www.pololu.com/product/2577
Actuador: Motor	Micro motor de 6V reducción 100:1, con doble eje	https://www.pololu.com/product/2214
Puente H	Doble controlador de motores DRV8833	https://www.pololu.com/product/2130

A.1.2. Componentes mecánicos

No todos los componentes utilizados en este robot fueron comprados debido a las tareas a realizar, a continuación una tabla de los componentes tanto comprados como diseñados en especifico.

Función	Descripción del componente utilizado	Referencia
Rueda para movimiento libre	Soporte pololu para esfera de 3/8 con esfera	https://www.pololu.com/product/950
Soporte para motor	Soporte para micro-motor pololu, par extendido	https://www.pololu.com/product/1089
Base y tapa para robot	Base y tapa plana realizada en MDF	Ver diseño en figura A.1
Ruedas principales	Rueda semi-flexible de 3.8cm de diámetro	Ver diseño en figura A.2

A continuación una muestra de los diseños realizados y utilizados en el robot, la base para el robot fue realizada en MDF, mientras que la llanta fue impresa en FilaFlex (PLA Flexible) para evitar posibles derrapes en la trayectoria descrita.
Cabe destacar que se intentó realizar una versión impresa del soporte para motores, sin embargo el resultado no tuvo la suficiente densidad para soportar el movimiento por mucho tiempo.

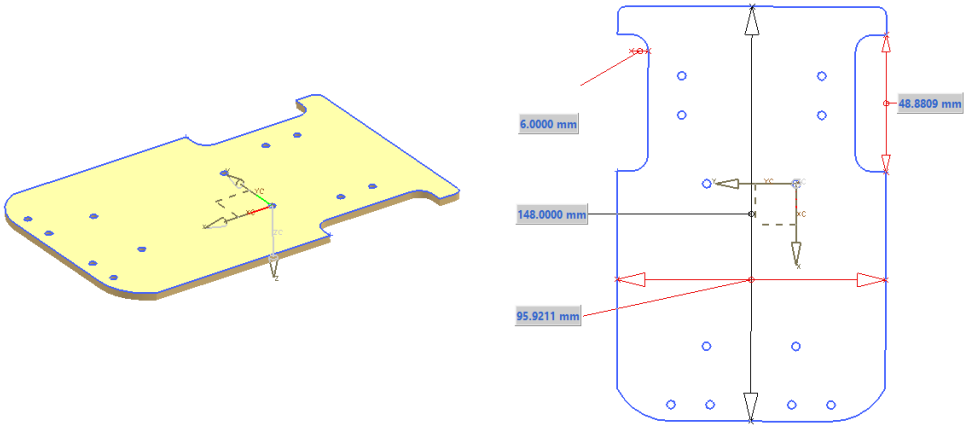


Figura A.1: Componente cortado con CNC en MDF utilizado tanto como base como nivel para colocar marcas en el robot

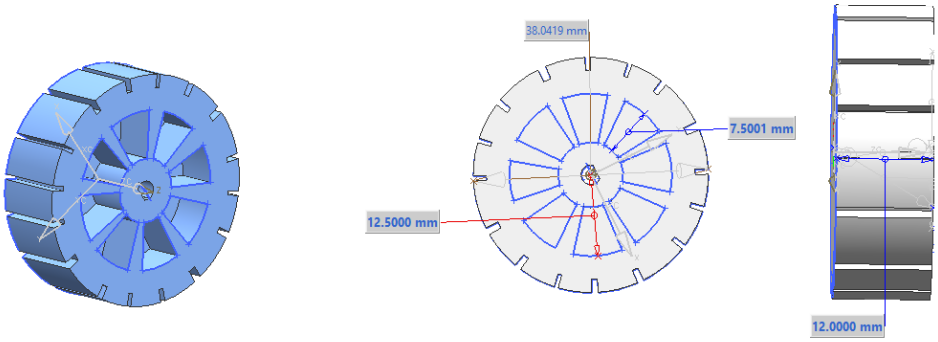


Figura A.2: Rueda impresa en FilaFlex.
Centro: Vista a lo ancho-largo.
Derecha: Vista de profundidad.

A.2. Algoritmos implementados

De forma global lo implementado en el robot móvil fueron los algoritmos mostrados en los siguientes apartados, cabe destacar que es necesario incluir este algoritmo en una ejecución ajena a cualquier otra operación lógica, utilizando solo los datos de medición de salida, ya que puede introducirse ruido por un retardo entre el cálculo y la medición, sobre todo en el caso de los codificadores rotatorios, donde la medición se realiza directamente a los sensores sin ningún otra electrónica de por medio:

A.2.1. Cuenta de movimiento en los codificadores rotatorios

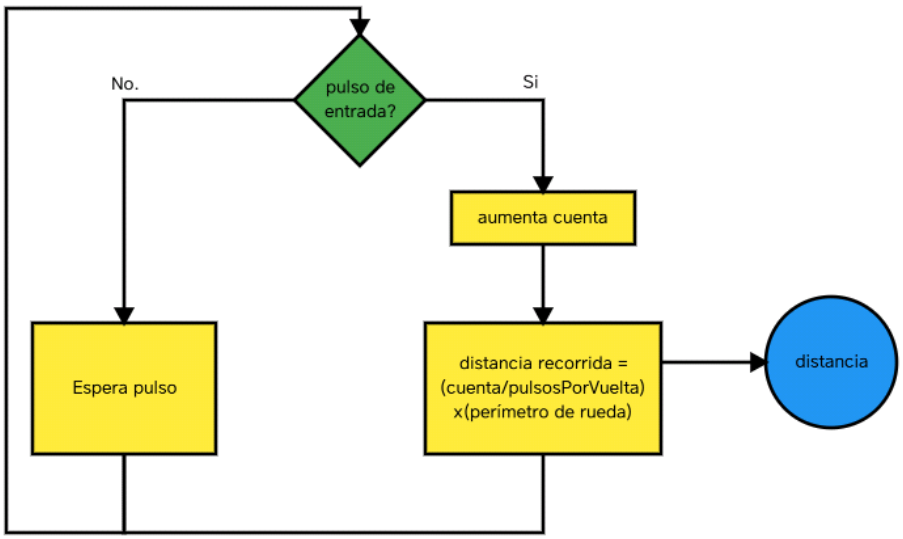


Figura A.3: Algoritmo aplicado para la medición de distancia avanzada con codificadores.

Antes de aplicar este algoritmo es vital conocer el perímetro de la llanta usada, así como de la cantidad de pulsos necesarios para una vuelta de 360 grados, esto varía dependiendo la resolución del codificador rotatorio. Posteriormente basta con aplicar el cálculo:

$$d = (\frac{\#pulsosMedidos}{\#pulsosPorVuelta}) * perimetroRueda$$

A.2.2. Acelerómetro

En este caso se optó por utilizar una variante de la librería creada por la empresa pololu (proveedora de los componentes electrónicos utilizados), ya que en ésta se sintetiza el uso de la comunicación I^2C. Dicha variante es desarrollada por la empresa Adafruit y se puede localizar (aún a julio del 2018) en el Repositorio de GitHub:

https://github.com/adafruit/Adafruit_Python_LSM303

Más allá del uso de este sensor, cabe aclarar que se utilizaron 2 métodos de lectura para compensar al acelerómetro por medio de los codificadores con un filtro de kalman, debido a la propagación del error de la medición individual con el acelerómetro al convertir la aceleración a distancia por medio de una segunda integral. De forma vectorial el cálculo aplicable con cualquier sensor queda de la siguiente forma:

$t = 0$)
Posición y Velocidad: $[x0, y0, z0]$, $[vx0, vy0, vz0]$.

$t = 1$)
Lectura de la aceleración promedio: $[ax1, ay1, az1]$
Vector de velocidad: $[vx1, vy1, vz1] =$
$$[vx0 + ax1 * (t1 - t0), vy0 + ay1 * (t1 - t0), vz0 + az1 * (t1 - t0)]$$
Velocidad promedio de $t = 0$ a $t = 1$: $[vx01, vy01, vz01] =$
$$[(vx0 + vx1)/2, (vy0 + vy1)/2, (vz0 + vz1)/2]$$
Posición en $t = 1$: $[x1, y1, z1] = [x0+vx01*(t1-t0), y0+vy01*(t1-t0), y0+vy01*(t1-t0)]$

Sustituyendo el vector de velocidad en $t = 1$ y la velocidad promedio de $t = 0$ a $t = 1$ en la ecuación de posición, no solo se resuelve esta última, sino también resulta en una propagación del error en t^2, es debido a esto la ya mencionada necesidad de una compensación vía codificadores rotatorios.

APÉNDICE B

Algoritmo para muestras de entrenamiento

El procesamiento a partir de la entrada de la cámara se realiza en el siguiente orden:

1. Obtención del nombre de la imagen a generar
2. Dilatación
3. Cerradura
4. Modificación de nitidez
   a) Emborronamiento Gaussiano:
   ventana ajustable, tamaño predeterminado igual a 3.
   b) Suma con peso:
   Imagen post-emborronamiento -0.5, imagen post-cerradura 1.5
5. Detector de bordes Canny:
   Umbrales de histeresis ajustables iniciando en 59, segundo umbral igual al doble del primero, operador de Sobel igual a 3.
6. Genera contornos a partir de Canny
7. Encuentra contorno mas largo: Solo contornos externos.
8. Máscara del contorno
9. Suma bitwise: Mascara del contorno con imagen original
10. Guardar imagen generada

Para las operaciones morfológicas utilizadas el elemento morfológico fue un rectángulo de tamaño (5 x 5) bits con centro en $(2, 2)$.

A continuación se muestra el resultado de aplicar el algoritmo.

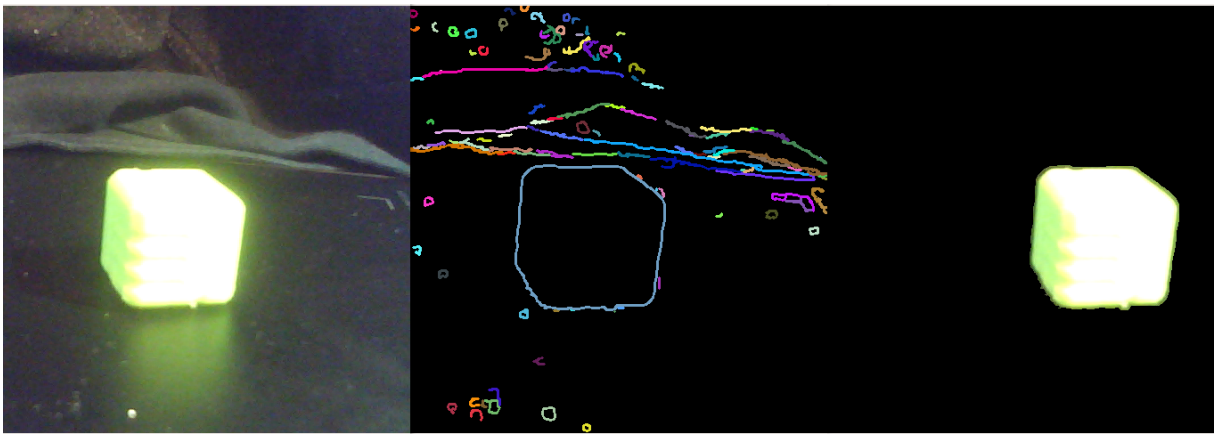


Figura B.1: Resultado de aplicar el algoritmo sobre un dado cubico color verde claro