UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DETECCIÓN Y SEGUIMIENTO DE PERSONAS MEDIANTE TÉCNICAS DE FILTRADO

TESIS

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERÍA ELÉCTRICA

PRESENTA:

Alberto Navarrete Hernández

DIRECTOR DEL TRABAJO:
Dr. Jesús Savage Carmona
Facultad de Ingeniería

Ciudad Universitaria, CD. MX. , enero 2017

JURADO ASIGNADO:

Presidente: Dr. Boris Escalante Ramírez

Secretario: Dra. María Elena Martínez Pérez

Vocal: Dr. Jesús Savage Carmona

1er Suplente: Dr. Pablo Roberto Pérez Alcázar

2do Suplente: M. I. Larry Salguero Escobar

Lugar o lugares donde se realizó la tesis: Ciudad Universitaria.

TUTOR DE TESIS:

Dr. Jesús Savage Carmona

_____

FIRMA

Agradecimientos

A mi familia por el apoyo que siempre ha mostrado a lo largo de mis diferentes etapas en mi formación profesional y lo han mostrado en esta ocasión nuevamente.

A la División de Estudios de Posgrado de la Facultad de Ingeniería, por darme la oportunidad de formar parte de su programa de maestría. Así como a mis compañeros con los cuales tuve la oportunidad de convivir en este período.

Al Dr. Jesús Savage Carmona por el apoyo en este trabajo y la oportunidad de adquirir nuevas experiencias a lo largo de la maestría.

Se agradece a la DGAPA-UNAM por el apoyo proporcionado para la realización de esta tesis a través del proyecto PAPIIT IG100915 ”Desarrollo de técnicas de la robótica aplicadas a las artes escénicas y visuales”. Y finalmente al CONACYT, por el apoyo proporcionado.

Resumen

En el presente trabajo se desarrolla un sistema que tiene una fase para la detección y otra para el seguimiento de una persona, en una secuencia de imágenes obtenidas mediante el dispositivo Kinect de Microsoft. Dado que el dispositivo Kinect permite a través de sus sensores obtener información tridimensional de una escena, se hace uso de esta información y se combina con métodos de análisis de imágenes en dos dimensiones, para la detección de la persona y poder realizar el seguimiento.

Se muestran dos métodos para el seguimiento de personas: uno es el uso de un filtro adaptable LMS (por sus siglas en ingles Lest Mean Square) y el otro es el uso del filtro de partículas. Este último es una herramienta que ha sido considerada en diversos trabajos similares al presente. En lo referente al seguimiento de la persona, se propone el poder obtener una región de interés que contará con un punto central (centroide), el cual será parte fundamental en la actualización en tiempo real de la ubicación de la persona mediante los resultados obtenidos de las técnicas de seguimiento.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

En la actualidad una de las ramas de la robótica que se encuentra en constante progreso es el área de visión computacional o visión artificial y ésta es un área que sigue presentando grandes retos; dentro de ellos se encuentra como enfoque fundamental, el reconocimiento de objetos que están presentes en una escena determinada. Por otro lado, también la tecnología ha avanzado de tal manera que hoy en día es posible reconstruir escenas 3D, mediante cierto tipo de cámaras que cuenten con sensores capaces de medir la profundidad o a partir de la información 2D de las imágenes.

Los avances tecnológicos que existen actualmente en el campo de la robótica han permitido poder superar ciertos retos, ya que la gran mayoría de sistemas de robots, ya sea del tipo interactivo o no, tienen que resolver una cantidad común de problemas. Estos incluyen áreas de conocimiento (toma de decisiones, planeación), percepción (navegación, sensado del ambiente), acción (movilidad, manipulación) e interacción humano-robot (interface de usuario y elementos de entrada, despliegue de respuestas) [15].

Un robot socialmente interactivo debe percibir libremente e interpretar la actividad y el comportamiento humano, lo cual incluye detectar y reconocer gestos, monitorear y clasificar actividades, e incluso discernir intenciones y señales sociales; así como, medir la respuesta humana. Para interactuar de una manera mas natural con los humanos, se tiene la intención de que los robots perciban el mundo como nosotros mismos lo hacemos, es decir, captar e interpretar el mismo fenómeno que el humano observa.

Así mismo, uno de los grandes retos de la interacción humano-robot consiste en encontrar métodos que permitan el seguimiento de personas en la presencia de oclusiones, cambios de iluminación, el uso de cámaras en movimiento, y los diferentes escenarios que se puedan presentar [15].

Actualmente las aplicaciones posibles de los métodos de seguimiento de objetos son muy diversas y su uso puede encontrarse tanto en procesos industriales, como en los sistemas de videovigilancia. Una muestra se tiene en la fabricación de elementos para vehículos mediante el uso de robots, en la cual se tiene la necesidad de posicionar de manera precisa herramientas sobre diversas piezas de trabajo. A través del uso de estos métodos de

seguimiento se puede conocer la ubicación y posición de la pieza que se desea utilizar con respecto a la de la cámara y de esta manera saber como posicionar la herramienta necesaria para realizar el trabajo sobre la pieza en cuestión.

Con la tecnología actual, se tiene el uso de sensores de profundidad, que en conjunto con las características de las cámaras a color o cámaras RGB (por sus siglás del inglés Red, Blue y Green), son utilizados para detectar en tiempo real objetos, incluyendo a las personas. De esta manera, mediante un sistema que procesa la información de estos sensores, a los cuales se les nombra como sensores RGB-D, las personas pueden utilizar su cuerpo y sus movimientos para interactuar de manera eficiente con un dispositivo.

La creciente popularización de los sensores RGB-D ha generado un gran interés científico en desarrollar nuevas técnicas de procesamiento de imágenes y adaptar otras ya conocidas utilizando toda la información provista por estos sensores. La información de profundidad obtenida por un sensor RGB-D es un dato fundamental que nos permite encontrar la distancia de un objeto con respecto al sensor, pudiendo recuperar información tridimensional 3D junto con sus correspondientes valores RGB en tiempo real. La información de profundidad es una señal importante cuando una persona reconoce objetos debido a que los objetos puede que no contengan color y textura consistente pero deben ocupar una región íntegra en el espacio.

Un sistema de seguimiento consiste generalmente en tres etapas definidas que nos permiten detectar y seguir un objeto, ya sea en un video o una secuencia de imágenes en tiempo real. Estas etapas son: entrenamiento, detección y seguimiento. La etapa de entrenamiento consiste en obtener una representación del objeto al cual se pretenda seguir. La etapa de detección consiste en obtener la ubicación del objeto a seguir en una imagen dada. Y la etapa de seguimiento la cual depende que el objeto detectado sea seguido correctamente.

En el presente trabajo se realiza la implementación, evaluación y estudio de un sistema de seguimiento de una persona en un video, a partir de los datos de escenas obtenidas mediante los sensores de profundidad de bajo costo como lo son el dispositivo Kinect para Xbox 360, de la compañía Microsoft, o la cámara Xtion de la compañia ASUS.

Como anteriormente se habia mencionado, el uso de la información de la imagen RGB y la estimación de profundidad posibilita una mejor precisión en la detección y seguimiento. Una parte importante es la necesidad de contar con un método de detección preciso y robusto dado que se pretende lograr el mayor acierto en la etapa de seguimiento, así como la ubicación en la imagen proporcionada por el algoritmo de seguimiento.

1.1 Objetivos

El propósito de este trabajo es desarrollar un sistema de seguimiento de una persona, por lo cual se tiene como objetivo:

- Detectar y reconocer a una persona en escenarios complejos dentro de un área limitada en el espacio.

- Seguir a una persona en movimiento en una trayectoria no definida y en un escenario también no conocido.

Por lo tanto en este trabajo se tienen las etapas de un sistema de seguimiento las cuales particularmente son:

- Detectar la presencia de las personas dentro de un cierto escenario.

- Reconocer a la persona a la cual se pretende seguir con base en un conjunto de características que sean robustos a los diferentes cambios como lo son los variaciones de iluminación y cambios de escala.

- Realizar el seguimiento de la persona mediante el uso de un estimador del tipo Bayesiano, que en este caso se contará con el uso de un filtro partículas, debido a su gran capacidad para la estimación de la función de distribución probabilidad *a posteriori* a partir de los datos obtenidos mediante una observación. También se utiliza un filtro adaptable para considerarlo en la etapa de seguimiento.

1.2 Estructura del documento

El capítulo uno corresponde a una breve introducción y los objetivos de este trabajo.

En el capítulo dos se citan algunos de los trabajos que se han realizado a lo largo del tiempo en relación con la robótica, la visión computacional y el problema de la detección y seguimiento de objetos, así mismo se da una introducción de las herramientas matemáticas utilizadas para tratar de resolver la cuestión del seguimiento de objetos.

El capítulo tres cuenta con la información de los elementos tanto de software como hardware que son utilizados en la realización de este trabajo, y también son los más usados en el campo de la visión computacional.

El capítulo cuatro contiene una descripción más detallada de dos elementos que son utilizados principalmente en el sistema de detección de personas: se define el concepto de Histograma de Gradientes Orientados (HOG) y métodos que pueden ser utilizados para el sistema de seguimiento, en este caso se dan aspectos teorícos del filtro adaptable, el filtro de Kalman y el filtro de partículas.

En el capítulo cinco se explica el procedimiento de la detección y seguimiento de las personas, con base a la teoría descrita en gran parte del capítulo cuatro.

Se tiene el capítulo seis en el cual se muestran los resultados obtenidos mediante el sistema planteado para la resolución del problema en este trabajo.

Por último se dan las conclusiones del trabajo realizado y se proporcionan algunos detalles que pueden considerarse a futuro para lograr ampliar y mejorar el presente trabajo.

Capítulo 2

Fundamentos y antecedentes

En este capítulo se describe brevemente, la manera en que los robots han ido desarrollándose para poder interactuar con el medio que los rodea, es decir, identificar objetos y realizar una estimación en particular de la ubicación de los objetos. Considerando para ello ciertas características de los objetos, que en general su adquisición pueda ser realizada mediante diversos tipos de sensores. Por otra parte, se cuenta con una descripción de los métodos estadísticos, que han sido utilizados para la estimación de la posición de un objeto en movimiento, a partir de las características consideradas del objeto.

2.1 La robótica centrada en los humanos

La investigación en el área denominada "robótica centrada en los humanos" y "la interacción robot-humano" pretende alcanzar tareas como: el reconocimiento de expresiones, la comunicación en un alto nivel, el aprendizaje y desarrollo de características sociales, entre otras. En diversos trabajos se trata entonces de poder realizar las tareas mediante lo que se llamaría un robot de servicio, que en general, tiene que enfocar su atención sobre los humanos y estar al pendiente de su presencia. Es por lo tanto muy necesario, contar con un sistema de seguimiento o rastreo que sea capaz de proporcionar la información de la posición en relación al robot en tiempo real y de las diversas personas que se encuentren en el rango de visión del propio robot. Por lo cual resulta una tarea compleja, debido a que el comportamiento de las personas es frecuentemente impredecible [15].

Pfinder [31] es uno de los primeros sistemas desarrollados para el seguimiento de personas. El sistema fue utilizado con mucho éxito en aplicaciones tales como video juegos, en una interface de realidad virtual e incluso para el reconocimiento de gestos. El sistema era capaz de seguir a una persona en tiempo real a 10 fps con imágenes de 160x120 píxeles.

Su sistema utiliza un modelo de apariencia basado en las estadísticas del color y la forma, así su sistema cuenta con una cámara estacionaria y realiza la substracción del fondo para detectar una persona y representarla mediante "manchas" e inicializar el propio modelo. El seguimiento se procede mediante la predicción de posición de las "manchas" basado en un modelo de movimiento de velocidad constante y después actualiza los respectivos

modelos al realizar la clasificación de los píxeles.

En diversos trabajos se han utilizado diferentes métodos para poder resolver el problema de seguimiento de objetos y en muchos de los casos se han encontrado soluciones sujetas a diversas limitaciones; algunas de ellas, por ejemplo, realizan el seguimiento en condiciones sencillas mediante un robot estático o también incluyendo algunos sensores adicionales que son distribuidos en el ambiente sobre el cual se realiza el seguimiento.

Probablemente el primer robot diseñado para reconocer personas fue Polly [18], desarrollado en los laboratorios del MIT, mostrado en la figura 2.1.1. Contaba con un sistema de visión simple, una cámara apuntando en dirección hacia el piso con una resolución de 64x48 píxeles y a una velocidad de 15 fps. El sistema podía estimar de cierta forma la profundidad de diferentes objetos en el ambiente, al filtrar los píxeles correspondientes al piso. Esta aproximación asume que el ambiente es planar, de tal manera que la profundidad puede ser estimada a partir de la altura en el plano de la imagen, y que el piso tiene una textura distintiva que puede ser fácilmente separada de los objetos que se encuentra en el primer plano.

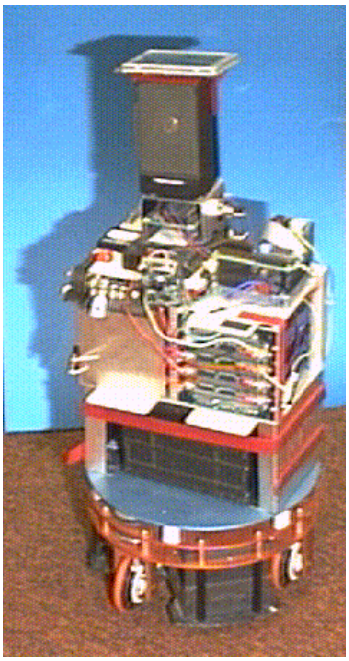


Figura 2.1.1: Robot Polly [18]

Basada en esta información de profundidad, Polly podía detectar objetos que correspondieran a las piernas de una persona, en un ambiente donde no estuvieran presentes objetos similares como las patas de una silla o una mesa. Los muros y cruces eran detectados por el mismo sistema de visión utilizando una aproximación similar. El sistema podía reconocer gestos simples como la agitación de pies, lo cual permitía una interacción simple con el usuario [18].

Los investigadores Blackburn y Nguyen en 1994 presentaron un robot móvil equipado

con un sistema de control y visión inspirado biológicamente, el cual podía separar el movimiento de un objeto, considerando el movimiento del ambiente provocado por el desplazamiento del robot. La velocidad del sistema era de 15 fps a una resolución de 128x128 píxeles. El sistema requirió que la velocidad del objeto seguido fuera considerablemente alta para separar el objeto del fondo; así, la aproximación solo podía ser efectiva en el caso del seguimiento de personas mientras éstas se estuvieran moviendo rápidamente de un lugar a otro [5].

El seguimiento de una persona permite a los robots de servicio elaborar un plan y adaptar sus movimientos de acuerdo a las personas que se encuentran a su alrededor o también pueden ser útiles en el caso de seguir a alguien a través de diferentes áreas en el interior de un edificio. Otro campo de aplicación que tiene relación con los robots de servicio, es el de la seguridad automática o remota, el cual pretende ser utilizado para monitorear áreas donde es complicado el uso de los sensores fijos.

Los elementos comúnmente utilizados para la tarea de segumiento en los robots de servcio, son los sensores láser y diferentes tipos de cámaras, como se puede observar en la figura 2.1.2. En algunos trabajos realizados mediante estos elementos, el sistema solamente detecta objetos en movimiento, realizando un seguimiento a través del uso de algoritmos heurísticos, y por otra parte el robot puede permanecer estático o realizar movimientos muy lentos. En estos casos, debido a que el rango de trabajo se encuentra limitado por el ángulo de visión de la cámara, se tiene por ende que es muy difícil el realizar el seguimiento de dos objetos o más al mismo tiempo [4].

Por otra parte, la mayoría de los sistemas de reconocimiento de rostro están basados en algoritmos de visión. Utilizando una imagen o una secuencia de imágenes, los métodos se pueden dividir en: métodos holísticos, métodos basados en características y los métodos híbridos. Los médotos holísticos, tienen la región del rostro completo como entrada y diferentes aproximaciones, como los que utilizan análisis de componentes principales, máquinas de soporte vectorial, algoritmos genéticos y redes neuronales artificiales; son utilizadas para realizar el reconocimiento.

Los métodos basados en características, se apoyan principalmente en el análisis de los ojos, la nariz o la boca. Ejemplos representativos incluyen métodos de acoplamiento gráfico, modelos ocultos de Markov y mapas de características auto-organizadas.

Por último se encuentran los métodos híbridos, cuya aproximación es similar al sistema de percepción humana, combinando el análisis del rostro completo así como características locales. A pesar de varios avances en esta área de investigación, se tiene un problema aún, el cual consiste en la sensibilidad a las variaciones de luminosidad y de la posicion del rostro.

Los investigadores Feyrer y Zell [13] propusieron un sistema basado en una detección jerárquica: color de la piel y señales de movimiento, las cuales eran utilizadas para seleccionar regiones de interés que después eran filtradas, dependiendo del contorno y la información estéreo. Existen también sistemas que combinan el color de la piel con los

Figura 2.1.2: Robot Pioneer [4]

datos de un sensor láser. En este tipo de sistemas, las regiones que hipotéticamente corresponden a personas son encontradas mediante la detección del color de la piel y los datos del láser son utilizados para determinar distancias y determinar su tamaño.

En el laboratorio de BioRobótica de la Facultad de Ingeniería se cuenta con el robot de servicio Justina, el cual ha sido parte de varias competencias donde se tiene el enfoque en los robots que operan dentro de un hogar, tratando de evaluar la capacidad de realizar actividades cotidianas de la manera más precisa posible. En la figura 2.1.3, se muestra al robot Justina en su participación en la competencia RoboCup

Figura 2.1.3: Robot Justina

La competencia RoboCup, tiene un gran reconocimiento a nivel mundial y es celebrada anualmente. Dentro de la categoria RoboCup@home, en la cual ha estado participando el equipo del laboratorio de BioRobótica de manera constante, hay una prueba que se denomina *Follow me*. Esta prueba *Follow me* consiste en reconocer y seguir a una persona, sin tener previamente conocimiento de ella, a través de un espacio público. Se tiene una interacción básica y algunos señalamientos para una navegación segura, en un ambiente con espacios reducidos y con la interveción de otras personas de manera indirecta.

La prueba entonces consiste, básicamente, en tres etapas que son: el comienzo, donde un referi y un operador realizan las condiciones necesarias para iniciar; la etapa de memorizar al operador, el cual puede contar con un cierto procedimiento por parte del robot y la última etapa que es la de seguimiento, donde el robot sigue al operador y además tiene que tratar con diversos obstáculos en diferentes secciones.

Entonces, en esta prueba, el operador tienen que actuar de manera natural y el robot utilizar los procedimientos necesarios para poder seguir a la persona en caso de algún obstáculo (obstrucciones en el camino ya sea mediante objetos o grupos de personas). Finalmente se califica el desempeño del robot mediante los tiempos requeridos en las diversas secciones que consituyen el recorrido y el grado de seguridad con el que se haya realizado.

2.2 Métodos estadísticos

En la gran mayoría de los sistemas, la información adquirida por los sensores pueden ser imprecisa o incluso errónea debido al ruido en los sensores, a un ambiente ruidoso y también debido a oclusiones dinámicas causadas por otros objetos y personas. Por lo tanto, para estimar de forma fiable la localización y el movimiento de personas u objetos, es necesario aplicar el procedimiento de seguimiento. El seguimiento permite la combinación de información de diferentes sensores, dando resultados más precisos y completos.

Entre las métodos utilizados para realizar la estimación del seguimiento, se encuentran los del tipo de inferencia Bayesiana y los relacionados al método de Monte-Carlo. La aproximación Bayesiana requiere de la representación probabilística del modelo dinámico, de las cuales las soluciones más óptimas para el estimador del tipo Bayesiano recursivo pueden ser obtenidas mediante ciertas suposiciones; ejemplo de este de tipo de aproximación son:

- El filtro de Kalman, cuya principal característica es considerar que las funciones de estado y de las mediciones son del tipo lineales, así como la representación del ruido que se presenta en el modelo es del tipo Gaussiano y de parámetros conocidos. El filtro de Kalman es un método muy popular en muchas aplicaciones de seguimiento [2].
- Métodos basados en "mallas", los cuales consideran que el espacio es discreto y consiste de un número finito de estados [2]. Estos métodos pueden llegar a ser computacionalmente ineficientes con el incremento en el tamaño del espacio de estados.
- Filtro de Bernes-Daum, cuya principal característica considera que el modelo es lineal, y éste es de una clase limitada de filtros para los cuales existe una solución exacta.

Los métodos presentados hasta ahora mencionados son frecuentemente inadecuados para aplicaciones en sistemas de seguimiento real que tienen que manejar las funciones del tipo no Gaussianas, fenómenos no lineales o no estacionarios. Otras soluciones utilizan métodos subóptimos, algunos de ellos son por ejemplo:

- Aproximaciones analíticas. Estos métodos están basados en el filtro de Kalman, pero se denomina filtro de Kalman extendido (EKF por sus siglas en inglés), cuya principal idea es tratar de hacer lineal, tanto el sistema no lineal como las funciones de medicion. El EKF utiliza un procedimiento de linearización basado en el primer término de la serie de Taylor y una extensión obvia es usar mas términos lo cual resulta en un EKF de mayor orden. Otra versión del tipo EKF es una variante iterativa que maneja la linearización de las ecuaciones de medidas, basado en la actualización del filtro [2].
- Aproximaciones numéricas. Estos métodos aplican integraciones numéricas para resolver las integrales encontradas en las ecuaciones Bayesianas. Estos también son denominados métodos basados en malla. El costo computacional de la aproximación se incrementa drásticamente con el incremento del tamaño del espacio de estados, una mayor dimensionalidad también afecta el radio de convergencia. El espacio de

estados debe estar predefinido y por lo tanto no puede ser particionado arbitrariamente sin un conocimiento a priori.

- Filtros de suma Gaussiana. Estos métodos también son conocidos generalmente como filtros de modelo múltiple. La idea clave es aproximar la probabilidad posterior por una mezcla de Gaussianas (una suma ponderada de funciones de densidad Gaussianas). Existe una versión estática para aproximar los parámetros en línea del filtro con un número fijo de componentes y un modelo dinámico.
- Muestreos de aproximación. Estos métodos incluyen otra variante del filtro de Kalman llamado filtro de Kalman Unscented (UKF por sus siglas en inglés) y los métodos de Monte Carlo. El método del UKF utiliza directamente un modelo no lineal, contrario al método EKF. El UKF representa la distribución Gaussiana con un conjunto mínimo de muestras, el cual es un poco mas que el número de muestras necesarias para el metodo de Monte Carlo. En cada instante de tiempo, el método UKF muestrea el estado alrededor de la estimación actual en una manera determinística. Cada muestra es actualizada usando un modelo de sistema no lineal y una nueva estimación es calculada después de incorporar las nuevas observaciones.

El método UKF produce una mejor aproximación que la del método EKF para sistemas no lineales, pero su complejidad computacional es más elevada que para el método EKF. El método UKF aún asume el uso de distribuciones de probabilidad Gaussiana, por lo tanto éste no puede manejar distribuciones multimodales.

El método de Monte Carlo [19], provee una solución aproximada basada en un muestreo de una distribución de probabilidad inicial, para el problema de estimación Bayesiano. La idea principal es representar la función de densidad *a posteriori* requerida por un conjunto de muestras aleatorias con ponderaciones asociadas, para calcular estimaciones basadas en estas muestras. Así, cuando el número de muestras llega a ser lo suficientemente grande, esta representación llega a ser equivalente a la densidad *a posteriori* verdadera. Simultáneamente el filtro se aproxima a un estimador Bayesiano óptimo. Estos métodos son mencionados con distintos nombres dependiendo del dominio donde estos son aplicados : filtro de partículas, filtros *bootstrap*, aproximación de partículas de interacción, el algoritmo de condensación en visión computacional y “supervivencia del más apto” en algoritmos genéticos.

En el año 1999 [6] se presenta un trabajo de una aproximación basado en el filtro de partículas para seguir varios objetos desde una plataforma móvil. El método utiliza la información de un escáner láser para detectar las piernas de las personas, usando un acoplamiento de escaneo entre escaneos consecutivos para separar las piernas del fondo, lo cual significa que la aproximación no puede detectar personas estáticas.

Para cada objeto se utiliza un filtro de partículas simple y un algoritmo del tipo JPDAF (del inglés “Joint probabilistic data association filter”) es el encargado de la asociación de datos. Este algoritmo explícitamente modela oclusiones para incrementar la robustez, basándose en la señal de movimiento. Este es uno de los trabajos que proporcionaron una representación rigurosa del problema de seguimiento a múltiples personas en el campo de la robótica móvil.

De lo descrito en este capítulo se consideró que para este trabajo, una forma práctica de realizar el reconocimiento de un un objeto, es ubicar a éste en una región de interés en una imagen junto con su información de color, que será parte fundamental de la etapa de seguimiento, la cual utiliza principalmente el método de Monte Carlo.

Capítulo 3

Elementos de software y hardware

En este capítulo se presenta una breve descripción de los aspectos relacionados con la visión computacional y dos temas que son principales en el análisis de imágenes, en un caso se definen los modelos de color y por otra parte se describe lo que son histogramas. Se exponen además algunas características de los elementos de software y hardware, que son utilizados generalmente para la adquisición de información 2D y 3D, que son usados para un análisis.

3.1 Visión computacional

El área de visión computacional consiste en realizar la transformación de los datos de una cámara fija o de un video en una nueva representación. Todas las transformaciones son realizadas para alcanzar alguna meta en particular. Los datos de entrada pueden consistir en alguna información contextual tal como indicar si "la cámara está montada en un automóvil" o si un "láser indica un objeto a un metro de distancia". La decisión, dependiendo del análisis que se este llevando a cabo, puede ser "hay una persona en la escena" o "existen 14 células tumorosas en esta diapositiva". Una nueva representación también consiste en cambiar una imagen de color a una imagen en escala de grises o remover el movimiento de una cámara de una secuencia de imágenes.

Debido a que los humanos contamos con un sistema visual, pareciera fácil pensar que las tareas de visión computacional son sencillas. Sin embargo, para encontrar un automóvil en una imagen, nuestro cerebro divide la señal de visión en varios canales en los cuales fluyen diferentes tipo de información. El cerebro cuenta con un sistema de atención que identifica partes importantes de la imagen para examinarlas mientras omite el análisis de otras áreas.

Existe una retroalimentación masiva del flujo de información visual, el cual es todavía poco entendido. Hay entradas asociativas generalizadas de los músculos de control de sensores y de todos los otros sentidos que permiten al cerebro dibujar diversas asociaciones relacionadas entre sí, hechas a partir de los años vividos en el ambiente. Las mallas de retroalimentación regresan a todas las etapas de procesamiento, incluyendo los propios

sistemas sensoriales (en este caso los ojos), los cuales mecánicamente controlan la iluminación a través del iris y sintonizan la recepción en la superficie de la retina.

En una máquina con un sistema de visión, hay una computadora que recibe solamente información en forma de malla que contiene solamente datos numéricos, ya sea de una cámara o un elemento de almacenamiento, por ejemplo, un disco. En realidad no existe un sistema de reconocimiento de patrones, ni tampoco una asociación interrelacionada por los años de experiencia. La información que recibe la computadora contiene, además, una gran cantidad de ruido y por lo tanto no proporciona la información como es en realidad.

El principal problema que se tiene con una imagen 2D de un mundo en 3D, es que, no hay una única forma de reconstruir la señal de la imagen en 3D. Ya que una imagen en 2D podría representar una combinación infinita de escenas 3D, aún si los datos que se tuvieran fueran precisos. El ruido contenido en las imágenes proviene de variaciones en el ambiente, tales como iluminación, el clima, reflexiones o movimientos; por otra parte, también se presentan imperfecciones en los lentes, en los ajustes mecánicos y en los medios de compresión después de capturar una imagen.

En el desarrollo de un sistema práctico, la información de contexto adicional que se pueda tener, puede ser utilizada para trabajar alrededor de las limitaciones impuestas por los sensores visuales. Esta información también puede ser modelada explícitamente mediante técnicas de aprendizaje de máquina.

Para resolver el problema del ruido normalmente se utilizan métodos estadísticos ya sea sobre una muestra o mediante el cálculo de variables estadísticas a través del tiempo. Otras técnicas utilizadas para compensar las distorsiones o eliminar el ruido son llevadas a cabo mediante la construcción de modelos explícitos aprendidos directamente de los datos con los que se cuentan, tal es el caso del uso de los parámetros de distorsión de la cámara; denominados parámetros intrínsecos y parámetros extrínsecos.

Debido a la manera en que el humano realiza el seguimiento de algo es a través de la visión, alrededor del año 2005 Zadjel [32], realizó un sistema de visión artificial el cual podía detectar y seguir varias personas mediante las características de movimiento y color, a través de algoritmos de redes bayesianas, operado desde la plataforma móvil de un robot y con una cámara.

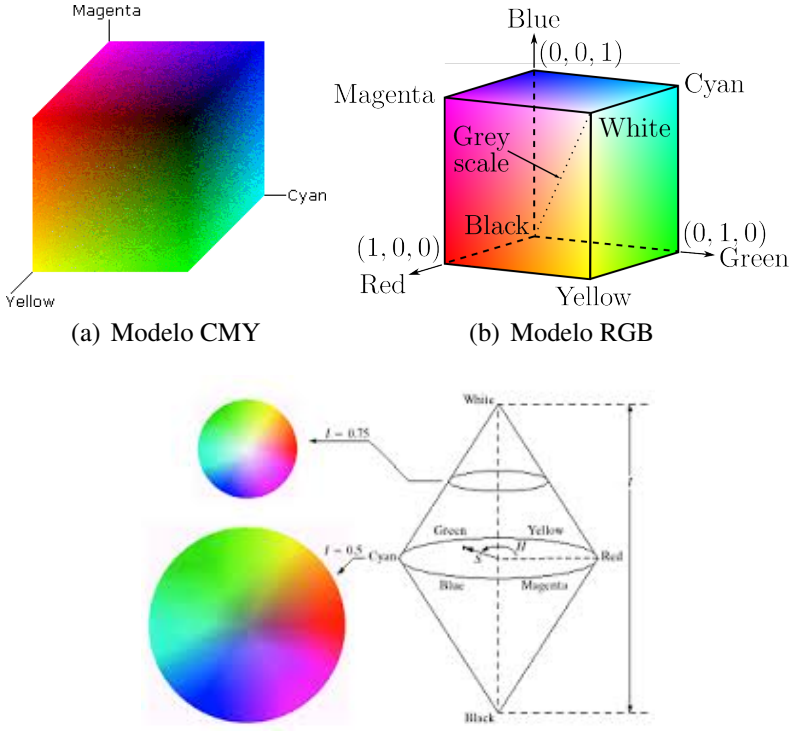
Así también se ha llegado a utilizar el método de visión panorámica, presentado en el trabajo de Chakravarty y Jarvis en el 2006 [7], el cual utilizaba una cámara panorámica estacionaria con la que el robot se podía dirigir, junto con un láser con el que obtenía la información espacial y realizaba el seguimiento.

3.1.1 Modelos de color

El propósito de un modelo de color o espacio de color es facilitar la especificación de colores dentro de un estándar, generalmente aceptado de cierta manera. En esencia, un modelo de color es un sistema de coordenadas y un subespacio dentro del cual cada color es representado por un punto [16].

La mayoría de los modelos de color utilizados están orientados hacia el uso de hardware, por ejemplo en el uso de monitores o de impresoras. En general se tiene el sistema de color RGB, que es utilizado en el procesamiento digital de imágenes, los modelos de color CMY (cyan, magenta, yellow) y CMYK (cian, magenta, yellow, black) se utilizan básicamente en las impresoras.

El modelo HSI (del inglés Hue, Saturation, Intensity) corresponde a la representación más cercana que se tiene acerca de la manera en que es interpretado y descrito el color por los humanos. Además, el modelo HSI tiene la ventaja que desacopla el color y la información en escala de grises de una imagen, facilitando el uso de técnicas de análisis de las imágenes en la escala de grises. En la figura 3.1.1 se puede observar las representaciones de los modelos de color.



(a) Modelo CMY (b) Modelo RGB

(c) Modelo HSI

Figura 3.1.1: Modelos de color [16]

Se puede realizar el cambio de un modelo de color a otro, usualmente se utiliza el modelo de color HSI en técnicas de procesamiento de imágenes. Entonces dada una imagen en el

formato RGB, el componente H de cada pixel es obtenido mediante la ecuación (3.1),

$$H = \begin{cases} \theta & \text{si } B \leq G \\ 360 - \theta & \text{si } B > G \end{cases} \tag{3.1}$$

donde el valor de θ está dada por la ecuación (3.2)

$$\theta = cos^{-1} \left\{ \frac{0.5[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{\frac{1}{2}}} \right\} \tag{3.2}$$

El valor del componente de saturación S se calcula mediante la euación (3.3)

$$S = 1 - \frac{3}{(R+G+B)}[\mathit{min}(R,G,B)] \tag{3.3}$$

Finalmente, el valor de la intensidad I está dadó por la ecuación (3.4)

$$I = \frac{1}{3}(R+G+B) \tag{3.4}$$

Se asume que los valores de los píxeles en el modelo RGB han sido normalizados de tal manera que se encuentran en el rango $[0, 1]$. Los valores del componente H pueden ser normalizados también en el rango de $[0, 1]$ al dividir entre 360 todos los valores obtenidos por la ecuación (3.1). Los otros dos componentes del modelo HSI se encontrarán normalizados si los valores de los píxeles RGB fueron normalizados.

3.1.2 Histogramas

Las funciones de transformación de intensidad, basadas sobre la información extraída de un histograma de intensidad de una imagen, desempeñan un papel central en el procesamiento de imágenes en áreas tales como la compresión, el mejoramiento de la imagen y la segmentación de una imagen.

El histograma de una imagen digital con *L* niveles de intensidad posibles en total dentro de un rango $[0, G]$ es definido como una función discreta:

$$h(r_k) = n_k \tag{3.5}$$

donde r_k es el *k*-ésimo nivel de intensidad dentro del rango $[0, G]$ y n_k es el número de píxeles en la imagen cuyo nivel de intensidad es r_k. El valor de *G* es de 255 para images con datos tipo *unit8*, 65535 para datos tipo *unit16* y uno para los datos flotante. En el caso de los datos tipo *unit8* y *unit16* se utiliza $G = L - 1$. A veces es necesario trabajar con histogramas normalizados, lo cual se obtiene simplemente al dividir todos los elementos

de $h(r_k)$ entre el número total de píxeles en la imagen, el cual tiene un valor conocido N. La ecuación para la normalización queda entonces como:

$$p(r_k) = \frac{h(r_k)}{N} = \frac{n_k}{N} \tag{3.6}$$

donde se tiene que $k = 0, 1, 2, \ldots, L-1$. De los conceptos de probabilidad se reconoce $p(r_k)$ como la probabilidad de ocurrencia del nivel de intensidad r_k.

También, a partir de la obtención de histogramas se pueden obtener los siguientes datos estadísticos, que llegan a ser considerados para cálculos que se requieran en alguna etapa de un proceso.

- Media: el valor promedio de todos los valores.
- Mínimo: el valor más pequeño de todos.
- Máximo: el valor más grande de todos.
- Desviación estándar: cantidad que expresa que tanto se extienden los valores alrededor de la media.
- Ancho de la clase: la distancia sobre el eje x entre el límite de la izquierda y el de la derecha en cada barra del histograma.
- Número de clases: el número de barras (incluyendo las de altura cero) en el histograma.

Con respecto a la comparación de dos histogramas, existen varias técnicas utilizadas para este fin, sin embargo, una de las mas utilizadas, es el coeficiente de Bhattacharyya [33]. Donde dadas dos distribuciones $p(u)$ y $q(u)$ se calcula el coeficiente mediante la ecuación:

$$\rho[p, q] = \int \sqrt{p(u)q(u)}du \tag{3.7}$$

Considerando que las distribuciones son del tipo discreto, es decir, $p = \{p^{(u)}\}_{u=1,\ldots,N}$ y $p = \{q^{(u)}\}_{u=1,\ldots,N}$, donde N es el número de clases, el cálculo simplemente se realiza de la siguiente manera:

$$\rho[p, q] = \sum_{u=1}^{N} \sqrt{p(u)q(u)} \tag{3.8}$$

Entre más grande sea el valor del coeficiente de Bhatacharyya, ρ, se tiene que las distribuciones son mas similares. Por otra parte, si además se tienen que las distribuciones están normalizadas, se puede calcular un valor al cual se le denomina distancia de Bhattacharyya, mediante la ecuación siguiente:

$$d = \sqrt{1 - \rho[pq]} \tag{3.9}$$

3.2 Kinect

El dispositivo Kinect fue diseñado y comercializado como un accesorio para la consola de videojuegos XBOX 360, con la intención de un nuevo estilo en la división de juegos de Microsoft sin la necesidad de tener que lanzar una nueva marca de consola de videojuego. La brillante idea era la de ofrecer una nueva y excitante forma de jugar para aquellos que tuvieran la consola XBOX 360, ya que el Kinect ofrece una interface natural sin cables, cargadores o controles y que con sólo mover una mano la interfaz respondiera adecuadamente. En la figura 3.2.1 se observan las partes principales que conforman el dispositivo Kinect.

Los requerimientos de hardware para el uso del dispositivo Kinect 360 en una computadora es necasario, contar con un puerto USB 2.0; una tarjeta gráfica capaz de manejar la información de imágenes 2D y 3D y por último una fuente de alimentación independiente para el uso del dispositivo Kinect 360.

El dispositivo kinect maneja una cámara RGB, un arreglo de micrófonos y un sensor de profundidad capaz de capturar el movimiento en 3D de un cuerpo completo. La consola de videojuegos cuenta con un sistema donde el dispositivo Kinect en conjunto con un software realiza la captura e interpretación de los datos durante su uso. La innovación y la proeza del dispositivo Kinect reside en su tecnología de sensado de profundidad. Un haz de luz infrarroja (IR) es producida y difundida a través de una habitación, enviando información codificada en la forma de patrones de luz que varían el tamaño de los puntos que cubren la habitación [17].

Figura 3.2.1: El sensor Kinect [23]

La información adquirida consiste en una nube de puntos, en la cual los datos con mayor precisión se encuentran a una distancia de 1.2m a 3.5m; así que, los datos son seleccionados y reenviados, transmitiendo la información tal que la distancia de cualquier objeto detectado está basado en la deformación de los patrones de luz IR. Dos cámaras complementarias, fabricadas con tecnología metal-óxido semiconductor; son utilizadas indivi-

dualmente para analizar mas a detalle estos datos obtenidos por los patrones de luz IR. La cámara RGB obtiene 30 frames por segundo de los eventos en tiempo real a una resolución de 640x480 píxeles.

Para poder utilizar el Kinect como elemento de control para diferentes tipos de actividades, es necesario el uso de los algoritmos que permitan realizar este trabajo. De esta forma se cuenta con el multilenguaje y multiplataforma llamada Open Natural Interaction (OpenNI), el cual define las API (Interfaz de Programación de Aplicaciones) para escribir aplicaciones que utilizan interacción natural.

OpenNI fue creada en el año 2010 y uno de sus principales miembros fue PrimeSense. Esta empresa liberó sus propios controladores de código abierto junto con un *middleware* de detección de movimiento denominado NITE. El software ha sido utilizado ampliamente entre la comunidad académica y algunos aficionados. Sin embargo, tras la adquisición por parte de Apple, el sitio web de OpenNI fue cerrado. Algunas organizaciones que utilizaban OpenNI conservaron la documentación para su uso en un futuro y se pueden encontrar actualmente en el sitio web: `http://www.structure.io/openni`.

Las API de OpenNI se componen de un conjunto de interfaces para escribir aplicaciones de interacción natural. La API estándar de OpenNI permite a los desarrolladores de aplicaciones realizar un seguimiento en tiempo real de escenas 3D utilizando los tipos de datos que se calculan a partir de la entrada de un sensor. Las aplicaciones pueden ser escritas independientemente de los proveedores de los sensores o middleware utilizado.

3.3 OpenCV

OpenCV es una fuente de código libre con librerías para el área de visión computacional, la cual se puede obtener de la web: `http://SourceForge.net/projects/opencvlibrary`. Las librerías se encuentran escritas en lenguaje de programación C y C++ y puede trabajarse bajo las plataformas de Linux, Windows y MacOS X. Existe también un desarrollo activo sobre las interfaces para Python, Ruby, Matlab y otros lenguajes de programación.

OpenCV fue diseñado para obtener una eficiencia computacional y con un enfoque muy profundo en aplicaciones que puedan ser realizadas en tiempo real. OpenCV está elaborada en un lenguaje de programación C optimizado de tal manera que se pueda tomar ventaja del uso de procesadores con varios núcleos. También se cuenta con la posibilidad de tener una optimización automática sobre arquitecturas con Intel, al tener las librerías IPP, los cuales consisten en rutinas optimizadas de bajo nivel en diversas áreas algorítmicas. OpenCV automáticamente usa las librerías apropiadas durante la ejecución de los programas si dichas librerías se encuentran instaladas.

Uno de los propósitos de OpenCV es el de proveer una infraestructura para el área de visión computacional fácil de utilizar, de tal manera que ayude a los usuarios a elaborar

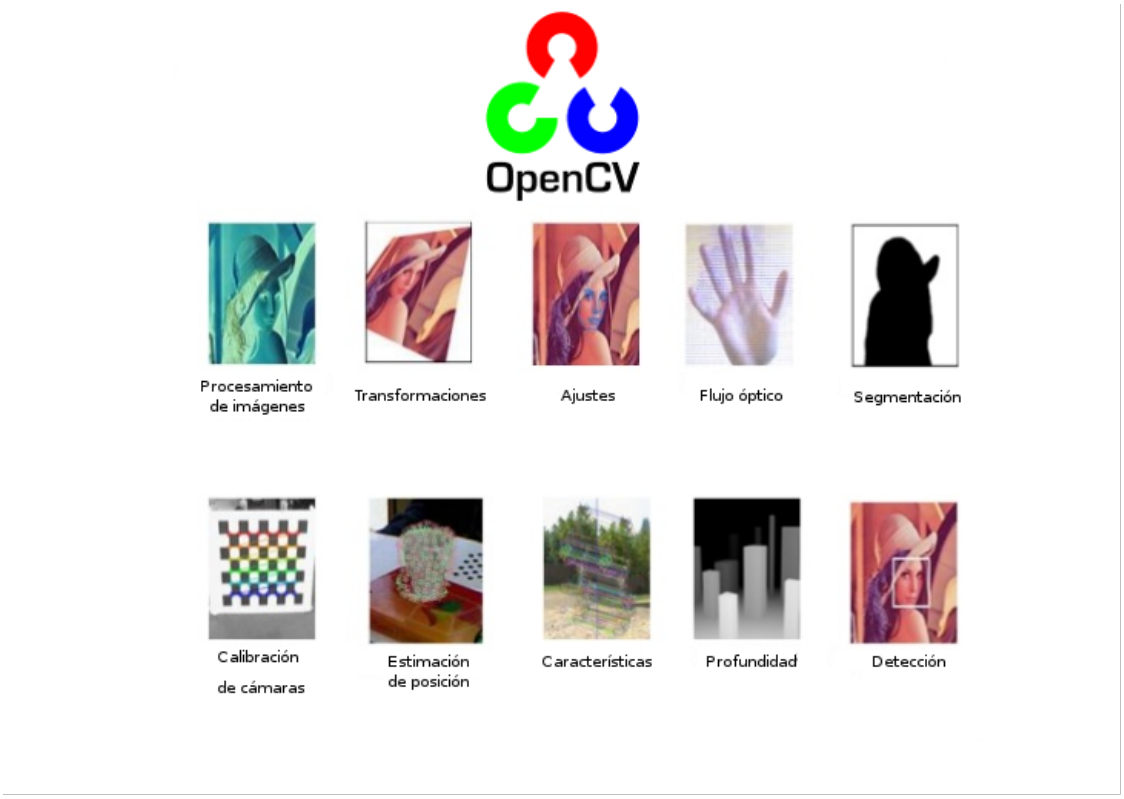


Figura 3.3.1: Librería Opencv [24]

aplicaciones de visión sofisticadas de una manera rápida. La librería de OpenCV cuenta con más de 500 funciones, las cuales abarcan las diversas áreas de visión tales como inspección de productos en fábricas, imágenes médicas, seguridad, interfase entre usuarios, calibración de cámaras, visión estéreo y robótica. En la figura 3.3.1 se indican algunas de las áreas de mayor uso mediante la librería OpenCV.

Desde el lanzamiento de la versión alfa en enero de 1999, OpenCV ha sido usada en diversas aplicaciones, productos y en algunas áreas de investigación. Ejemplos de algunas aplicaciones son: unión de imágenes en mapas de satélite, alineación de imágenes de escáner, reducción de ruido en imágenes médicas, análisis de objetos, sistemas de seguridad, aplicaciones militares, aplicaciones en vehículos áereos, terrestres y subacuáticos no tripulados.

3.4 PCL

Point Cloud Library (PCL) es una librería de código abierto para tareas de procesamiento de *nubes de puntos* y procesamiento geométrico, la cual contiene algoritmos para la estimación de superficies, registro entre nubes de puntos y segmentación de las mismas. PCL es una plataforma realizada en C++ y con lincencia BSD (por sus siglas en inglés Berkerly Software Distribution).

Los algoritmos que contiene la librería han sido ampliamente utilizados en diversas aplicaciones en el área de percepción de la róbotica; ya sea para filtrar el ruido en las nubes de puntos, unir nubes de puntos, segmentar lugares con ciertas características e incluso para crear nubes de puntos y poder visualizarlos.

El desarrollo de la librería PCL comenzó en marzo del año 2010 en el laboratorio Willow Garage (California, Estados Unidos); el proyecto inicialmente residía en un subdominio del laboratorio y luego se le colocó en un nuevo sitio web: `http://www.pointclouds.org` en el año 2011. El primer lanzamiento oficial de PCL versión 1.0 fue liberado en mayo del año 2011. En la figura 3.4.1 están algunas de las funciones que manejan la librería PCL.

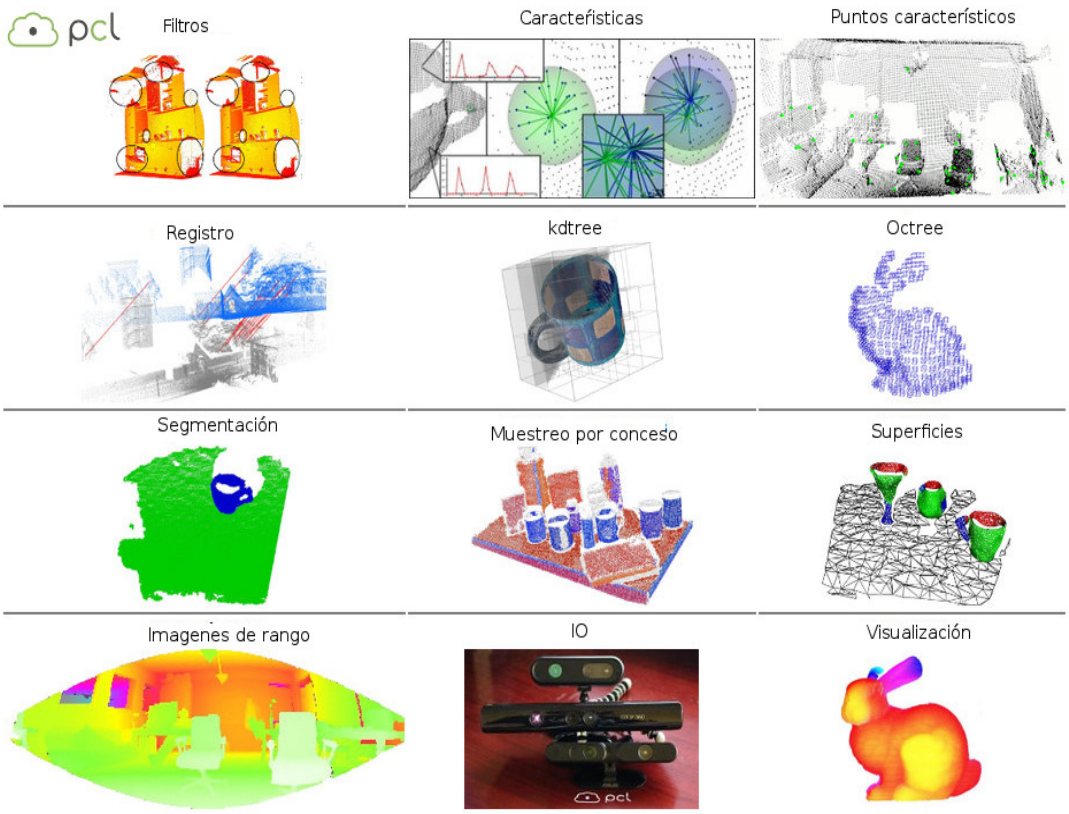


Figura 3.4.1: Librería PCL [26]

PCL es un conjunto de pequeños módulos [1] algunos de los cuales son :

– *libpcl_filters*, el cual contiene algoritmos para la eliminación de ruido y de valores atípicos (*outliers*). La implementación de estas eliminaciones se basan en la distribución de los puntos y la distancia con los puntos que se encuentran alrededor, considerando entonces un valor umbral para la eliminación.

- *libpcl_features*, realiza el análisis para la estimación de ciertas características a partir de los datos de la nube de puntos. Dichas características son representaciones en un cierto punto 3D, que describen patrones geométricos basados en la información en torno al punto analizado. El espacio de datos seleccionado alrededor del punto de análisis se refiere generalmente como los *k-vecinos*. Dos características más utilizadas de un punto son la estimación de la curvatura de la superficie y la normal en el punto analizado.
- *libpcl_keypoints*, cuenta con la implementación de algoritmos para la detección de puntos que se consideran como significativos. Estos puntos en un imagen se consideran como estables, distintivos y se pueden identificar utilizando un criterio de detección bien definido.
- *libpcl_registration*, consiste en poder realizar el registro de nubes de puntos para conjuntos de datos organizados o no organizados. La idea clave es identificar los puntos correspondientes entre los conjuntos de datos y minimizar la distancia entre los puntos correspondientes. Este proceso se repite dado que la búsqueda de correspondencia se ve afectada por la posición relativa y la orientación del conjunto de datos.
- *libpcl_kdtree*, proporciona la estructura de datos *kd-tree* que permite la búsqueda rápida de los vecinos más cercanos. La búsqueda de vecinos cercanos es una operación esencial cuando se trabaja con los datos de nubes de puntos y se puede utilizar para encontrar correspondencias entre grupos de puntos, descriptores característicos o definir el vecindario local en torno a un punto.
- *libpcl_octree*, proporciona métodos eficientes para la creación de una estructura de datos del tipo árbol jerárquico a partir de las nubes de puntos, esto permite una partición especial, submuestreo y operaciones de búsqueda en un conjunto de datos. Esta clase ofrece rutinas de búsqueda del vecino más cercano de manera más eficiente, tales como *"Neighbors whitin voxel search"*, *"K Neareast neighbor search"* y *"Neighbors within radius search"*.
- *libpcl_segmentation*, contiene algoritmos para la segmentación de una nube de puntos en grupos distintos. Estos algoritmos son los más adecuados para el procesamiento de datos que se encuentran formados por un número de regiones especialmente aisladas. En tales casos, la agrupación se utiliza a menudo para fragmentar la nube de puntos en sus partes constituyentes, que luego pueden ser procesados de manera independiente.
- *libpcl_sample_concensus*, contiene métodos llamados "Concenso de muestras" SAC (de su definición en inglés Sample Concensus) como el del tipo RANSAC (de su definición en inglés Random Sample Concensus) para modelos como planos y también se utilizan para el cálculo de superficies del tipo cilíndrico o esférico. Estos pueden combinarse libremente con el fin de detectar los modelos específicos y sus parámetros en las nubes de puntos.
- *libpcl_surface*, cuenta con los procedimientos para la reconstrucción de las superficies a partir de los datos obtenidos mediante escaneos en 3D. Dependiendo de la tarea en cuestión se hace una representación de una malla o superficie suavizada y remuestreada con normales. El suavizamiento y remuestreo puede ser un proceso importante si la nube es ruidosa o si se compone de varios escaneos que no están alineados perfectamente.

– *libpcl_io*, contiene las clases y funciones para la escritura y lectura de los datos contenidas en archivos de nubes de puntos (archivos tipo .pcd), así como la captura de datos a partir de una gran variedad de dispositivos de detección.

– *libpcl_visualization*, se realizó con el fin de ser capaz de crear prototipos y visualizar los resultados de los algoritmos que operan con los datos de la nube de puntos de manera más rápida. Se puede utilizar para mostrar imágenes en 2D y dibujar en pantalla formas básicas 2D. Esta clase cuenta con métodos para el procesamiento y la configuración de las propiedades visuales (colores, tamaños de puntos, opacidad, etc.) para cualquier conjunto de datos de puntos n-D en un cierto formato.

3.5 Robot Operating System (ROS)

Robot Operating System (ROS) es una plataforma, basado en la distribución linux Ubuntu para el desarrollo de robots, que provee la funcionalidad de un sistema operativo en un *clúster* heterogéneo. ROS se desarrolló originalmente en el año 2007 bajo el nombre de *Switchyard*, por el Laboratorio de Inteligencia Artificial de Standford, para dar soporte al proyecto del robot STAIR (Stanford Artificial Intelligence Robot). Desde el año 2008, el desarrollo continua primordialmente en Willow Garage (California, Estados Unidos), un instituto de investigación robótica con mas de veinte instituciones colaborando en un modelo de desarrollo federado.

ROS provee los servicios estándar de un sistema operativo tal como es el caso de la abstracción del hardware, el control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, paso de mensajes entre procesos y mantenimiento de paquetes. Está basado en una arquitectura de grafos donde el procesamiento toma lugar en los nodos que pueden, recibir, mandar y multiplexar mensajes de sensores, control, estados, planificaciones y actuadores entre otros.

Figura 3.5.1: Plataforma ROS [28]

Muchas instituciones han comenzado a desarrollar proyectos en ROS, así como diversas compañías han adaptado sus productos para que puedan ser utilizados en ROS. Razón por la cual cada día el número de elementos de esta plataforma va en aumento. La librería

está orientada hacia el sistema UNIX (Ubuntu linux), pero también se está adaptando a otros sistemas operativos como Windows, Mac OS, que son considerados como experimentales.

El *ros-pkg* es una comunidad de repositorios para desarrollar librerías de alto nivel. Muchas de las capacidades frecuentemente asociadas con ROS, tales como la librería de navegación y el visualizador Rviz (de su definición en inglés ROS Visualization), son desarrollados en este repositorio. Estas librerías proporcionan una poderosa herramienta para trabajar con ROS fácilmente, permitiéndonos saber que es lo que sucede en cada instante en el esquema de trabajo.

ROS promueve la reutilización del código de tal manera que los desarrolladores de robótica y los diversos científicos, pueden tomar el código de los repositorios, mejorarlo y compartirlo nuevamente. ROS ha lanzado diversas versiones desde su origen, por lo cual pueden ser incompatibles unas con otras. En la figura 3.5.1 se muestra el ícono de la penúltima versión. Actualmente se tiene la versión Jade liberada en mayo del año 2015.

Para poder realizar a cabo el presente trabajo se considera al dispositivo Kinect, como elemento único para poder obtener la información en 2D (imágenes) y al mismo tiempo contar con los datos 3D, que se representan mediante un conjunto o "nube" de puntos. Una vez adquirida ésta información, se usan en conjunto los tres elementos de software:

- PLC para realizar cálculos considerando la información 3D.
- OpenCV principalmente para analizar información 2D.
- ROS como elemento que une el análisis de toda la información.

Capítulo 4

Filtros y descriptor HOG

En el reconocimiento de objetos se utilizan los sistemas basados en detección de formas, en los cuales se seleccionan un conjunto de rasgos para discriminar los objetos que no son de interés con respecto a los que si lo son, a pesar de que los fondos estén saturados o las condiciones de iluminación no sean óptimas. Otro medio es emplear sistemas basados en descriptores. El reconocimiento de formas consiste en la detección de patrones de los objetos, quedando éstos representados por una colección de descriptores.

El punto esencial del reconocimiento de formas es la correcta clasificación del objeto en cuestión. Mediante un mecanismo de extracción de características se obtiene la información útil, eliminando aquella información que se considera como redudante e irrelevante. Finalmente se lleva a cabo la etapa de toma de desiciones en la cual un sistema de reconocimiento asigna a cada objeto su categoría o clase.

En la detección de personas en una imagen los descriptores pueden ser clasificados en dos categorías, la diferencia radica en como son medidas las características de los descriptores. La primer categoría que se utiliza es el analizar toda la imagen y llevar a cabo el análisis de componentes principales para poder tener un descriptor global. La segunda categoría es aquella donde el análisis de la imagen es realizado por regiones. La desventaja que se presenta con la primer categoría, es que al determinar características significantes, puede no obtenerse debido a la variación de la apariencia del objeto y a las condiciones de ilumincación. Y la segunda categoría no se ven tan afectados por este tipo de factores.

Por lo tanto el descriptor representará una región de la imagen que ha sido previamente extraída y cada región correspondiente a un objeto se caracteriza por un vector característico. Estos descriptores son conservados para posteriormente ser evaluados por un clasificador que determina la presencia o ausencia del objeto buscado, el cual en el caso del presente trabajo es detectar a una persona. El clasificador antes de su uso ha sido previamente entrenado a partir de un conjunto de ejemplos, representados por el descriptor a clasificar.

4.1 Descriptor Histograma de gradientes orientados (HOG)

Para la detección de personas en una imagen, últimamente se hace uso del decriptor "Histograma de gradientes orientados" (HOG por sus siglas en inglés) [34], el cual ha mostrado una gran eficiencia a la hora de utilizarlo en diversos sistemas. Existen dos clasificaciones para el descriptor HOG, el primero de ellos se denomina descriptor estático, debido a que los cálculos son realizados sobre una imagen. El segundo tipo de descriptor HOG se denomina descriptor de movimiento y éste se basa en el cálculo del descriptor sobre imágenes consecutivas derivadas de una secuencia de vídeo. Para el presente trabajo se hace uso del descriptor HOG del tipo estático, ya que solo se toma una imagen para realizar los cálculos correspondientes.

El descriptor HOG está basado en la idea de que la apariencia local del objeto y la forma del mismo pueden ser caracterizados frecuentemente de manera eficiente por la distribución de gradientes de intensidad local o la dirección de los bordes, aún sin un conocimiento preciso de las correspondientes posiciones de los gradientes o de los bordes [20]. En la práctica, la implementación se realiza al tomar la imagen, obtener la misma imagen en diferentes escalas, y en cada una de estas imágenes se procede a dividirlas en pequeñas regiones (denomindadas celdas), y para cada una de estas regiones acumular un histograma de gradientes unidimensional local sobre los píxeles de la región [9].

El primer paso para el cálculo del descriptor HOG es obtener la magnitud y orientación (ángulo) del gradiente . El segundo paso es obtener el histograma de orientación a partir de las magnitudes y las orientaciones. El tamaño usualmente utilizado para la detección de personas en una imagen es una ventana de 64 x 128 pixeles. Una caracterísa que se puede considerar como una desventaja es que normalmente se trata de detectar a las personas que se encuentran de pie en la imagen. Por lo tanto la región cubierta por la ventana es recorrida por bloques que se traslapan y cada uno de estos bloques consiste normalmente de cuatro celdas, cada celda es de tamaño 8x8 pixeles. En cada una de estas celdas se obtiene el histograma de gradientes.

En la mayoría de los casos se tiene que la región cubierta por la ventana consta de 3780 características [34]. En la figura 4.1.1 se muestra del lado izquierdo una imagen en la cual se tienen dos personas, una de ellas esta dentro de una región y en esta región se muestra una ventana que contiene las cuatro celdas. En la parte central de la imagen se muestra el ejemplo de los gradientes correspondientes a las celdas. Finalmente en la parte derecha se tiene el histograma de cada una de las cuatro celdas.

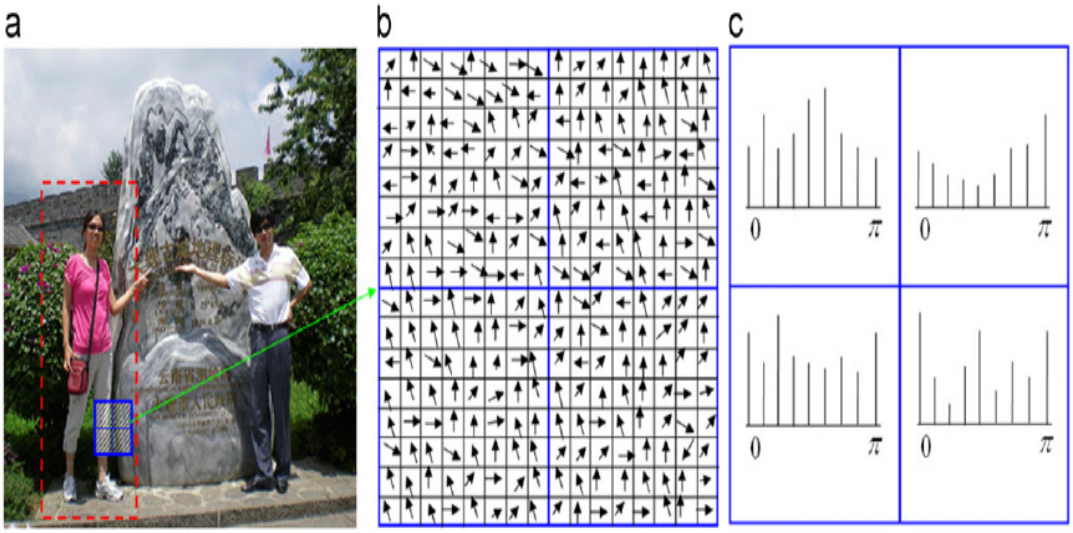


Figura 4.1.1: Cálculo del descriptor HOG [25]. (a) Una ventana de detección de 64x128 pixeles en la imagen y en su interior se tiene un bloque enmarcado en color azul , (b) el bloque de tamaño 16x16 pixeles el cual se divide en cuatro celdas y en cada una de ellas se muestra el gradiente correspondiente a cada píxel y (c) el histograma de gradientes que corresponden a cada una de las cuatro celdas.

En la figura 4.1.2 se puede observar un esquema completo de las etapas realizadas para el cálculo del descriptor HOG, una vez que se cuenta con la imagen a analizar. El primer paso consiste en realizar una normalización y equalizar la imagen para reducir la influencia de los efectos de iluminación, en la práctica se realiza una compresión tipo gamma. El segundo paso consiste en obtener el contorno, la silueta y alguna informacion de textura y realizar el cálculo de los gradientes.

El tercer paso consiste en poder realizar una codificación que permita conservar la información de la imagen y no se vea tan afectado a pequeños cambios debido a la apariencia de la imagen y en este caso se forma un histograma de gradientes para cada una de las celdas de la imagen. En el cuarto paso se realiza una normalización, el cual toma grupos locales de celdas a los cuales se les denomina bloques. Y el último paso consiste en agrupar todos los bloques y formar un vector característico para su uso en la etapa de clasificación de la imagen.

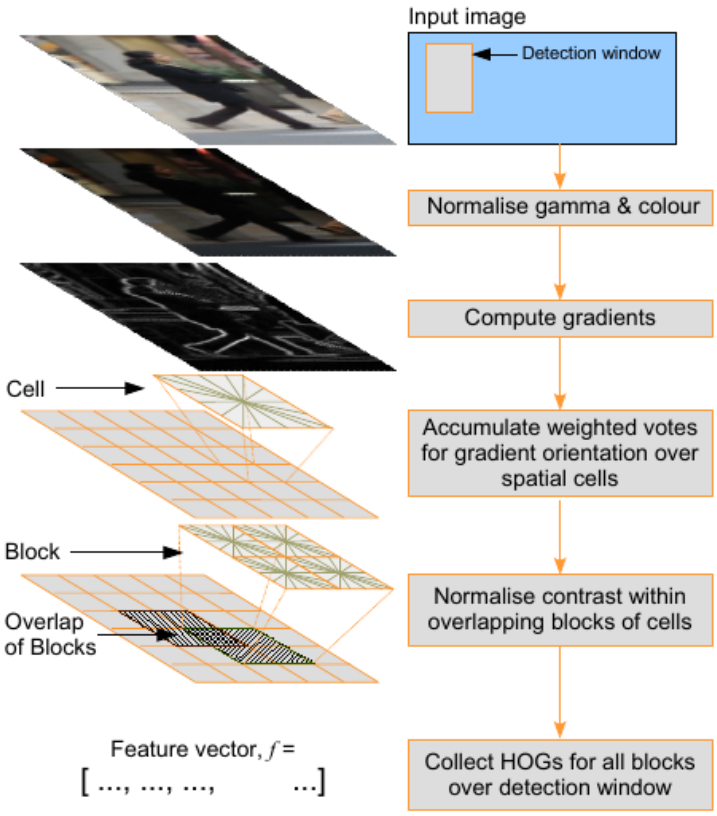


Figura 4.1.2: Diagrama del cálculo del descriptor HOG [9]

El sistema de reconocimiento usualmente utilizado para la detección de personas basados en el cálculo de histogramas orientados, cuenta con Máquinas de Soporte Vectorial o Máquinas de Vectores de Soporte (SVM de sus siglas en inglés), los cuales permiten la clasificación de si hay o no una persona en la región analizada. Las SVM son un conjunto algoritmos de aprendizaje supervisado empleados para la clasificación y la regresión desarrollados por Vladimir Vapnik y sus colaboradores [8].

Dado un conjunto de ejemplos de entrenamiento (imágenes modelo) se puede etiquetar el conjunto para definir las clases y poder entrenar una SVM para construir un modelo que prediga la clase de una nueva imagen muestra [25]. Por lo tanto, dado un conjunto de entrenamiento etiquetado $\{\mathbf{x}_i, y_i\}$, donde i =1,..., N; donde cada vector $\mathbf{x}_i$ de dimensión d representa las muestras del conjunto de entrenamiento, es decir, $\mathbf{x}_i \in R^d$ e $y_i = \{1, -1\}$. Una SVM considera que el mejor clasificador de datos es aquel hiperplano $\mathbf{w}^T\mathbf{x} + b = 0$ que máximice la distancia γ (o margen) con los puntos que estén mas cerca del plano, donde $\mathbf{w} \in R^d$ es un vector de parámetros que representa la normal al hiperplano y b se considera como un parámetro de sesgo. La distancia entre el plano y los puntos está dada por la ecuación:

$$\gamma = \frac{1}{\|\mathbf{w}\|^2} = \frac{1}{\langle \mathbf{w} \cdot \mathbf{w} \rangle} \tag{4.1}$$

donde $\langle \cdot \rangle$ representa el producto interno de dos vectores. Si las muestras de entrenamiento son linealmente separables, el problema de optimización de una SVM puede ser formulado mediante las siguientes ecuaciones:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq 1 \text{ para } y_i = +1 \tag{4.2}$$
$$\mathbf{x}_i \cdot \mathbf{w} + b \leq 1 \text{ para } y_i = -1 \tag{4.3}$$

combinando las ecuaciones 4.2 y 4.3 se llega a:

$$(y_i)(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geqslant 1, \quad i = 1, ..., N \tag{4.4}$$

Si las muestras contienen cierto ruido y también existen valores atípicos, el conjunto puede llegar a ser no linealmente separable. Para poder sobrellevar estos casos, se introducen variables estáticas ξ_i y los cálculos se dan de la siguiente manera:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq 1 \text{ para } y_i = +1 - \xi \tag{4.5}$$
$$\mathbf{x}_i \cdot \mathbf{w} + b \leq 1 \text{ para } y_i = -1 + \xi \tag{4.6}$$

de tal manera que la función a minimizar queda también afectada por esta variable, quedando entonces como

$$\frac{1}{\|\mathbf{w}\|} + C \sum_{i=1}^{N} \xi_i^2 \tag{4.7}$$

y nuevamente combinando las ecuaciones 4.5, 4.6 y 4.1 se tiene

$$(y_i)(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geqslant 1 - \xi_i, \quad i = 1, ..., N \tag{4.8}$$

donde *C* es un párametro determinado ya sea por usar un conjunto validado separado o alguna técnica de validación cruzada.

4.2 Filtro LMS (Least Mean Square)

Una de las herramientas muy utilizada en el análisis de señales en un proceso para la predicción de valores futuros, es la predicción lineal. En este tipo de análisis se considera una serie discreta de elementos $y(n), y(n-1), ..., y(n-M)$ los cuales representan los $M+1$ valores de un proceso hasta el tiempo n. Por lo tanto, se pueden utilizar los valores $y(n-1), y(n-2), ..., y(n-M)$ para estimar el valor de $y(n)$.

Se dice que se tiene una predicción lineal si el cálculo de la muestra $y(n)$ es realizada mediante una combinación lineal de las muestras anteriores, es decir, se tiene la siguiente ecuación:

$$\widehat{y}(n) = \sum_{k=1}^{M} w_{ok}^{*} y(n-k) \tag{4.9}$$

donde los valores de $w_{o1}, w_{o2}, ..., w_{oM}$ son coeficientes.

Dentro de los criterios que se tienen para validar la estimación obtenida, está el uso de una función de costo basado en el cálculo del error cuadrático medio (MSE por sus siglas en inglés), el cual puede ser definido como el valor esperado del error, cuya ecuación es:

$$J_{MSE} = E[|e(n)|^2] = E[|y(n) - \widehat{y}(n)|^2]. \tag{4.10}$$

Por lo tanto el problema de estimación se basa en poder determinar el valor de los coeficientes w, que se pueden considerar en conjunto como un vector $\mathbf{w}$, de tal manera que este vector minimice la función de costo $J_{MSE}(\mathbf{w})$. La solución óptima al problema de la estimación entonces determinará un valor $\mathbf{w}_{opt}$ de manera que éste sea el argumento mínimo de la función de costo. A partir de la expansión de la ecuación 4.10 se tiene una función del tipo cuadrático:

$$J_{MSE} = E[|y(n)|^2 - 2y(n)\mathbf{x}^T(n)\mathbf{w} + \mathbf{w}^T\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}]. \tag{4.11}$$

donde la solución óptima resulta ser el valor obtenido del gradiente de la función que es igual a:

$$\frac{\partial J_{MSE}}{\partial \mathbf{w}} = -2E[e(n)\mathbf{x}(n)] \tag{4.12}$$

E igualando a cero la ecuación anterior, el valor mínimo que se obtiene es :

$$E[e_{min}(n)\mathbf{x}(n)] = E\{[y(n) - \mathbf{w}_{opt}^T\mathbf{x}(n)]\mathbf{x}(n)\} = \mathbf{0}_{M\times 1} \tag{4.13}$$

Dado que la ecuación 4.13 se iguala a cero se tiene entonces:

$$E[\mathbf{x}(n)\mathbf{x}^T(n)]\mathbf{w}_{opt} = E[\mathbf{x}(n)y(n)]. \tag{4.14}$$

Considerando la matriz de autocorrelación y la matriz de correlación, definidas como :

$$\mathbf{R}_{\mathbf{x}} = E[\mathbf{x}(n)\mathbf{x}^T(n)] \text{ y } \mathbf{r}_{\mathbf{x}y} = E[\mathbf{x}(n)y(n)] \tag{4.15}$$

La solución a la ecuación 4.13 mediante la ecuación 4.15, asumiendo que la matriz $\mathbf{R}_{\mathbf{x}}$ es definida como positiva, se conoce como *filtro de Wiener* [11].

$$\mathbf{w}_{opt} = \mathbf{R}_{\mathbf{x}}^{-1}\mathbf{r}_{\mathbf{x}d} \tag{4.16}$$

A partir de las muestras mas recientes con que se cuentan, se pueden realizar dos tipos de estimaciones, la primera se denomina predicción lineal "*hacia adelante*" donde se utilizan un conjunto de muestras $x(n-1), x(n-2), ...$ para estimar el valor de $x(n+k)$ con $k \geq 0$. El segundo tipo es la predicción lineal "*hacia atrás*", en este caso las muestras $x(n-1), x(n-2), ..., x(n-M+1)$ son utilizadas para estimar el valor de $x(n-k)$ con $k \geq M$.

Para el caso de la predicción lineal hacia adelante se puede considerar el error de la siguiente forma :

$$e_{f,M} = x(n) - \sum_{j=1}^{M} w_j x(n-j) = x(n) - \mathbf{w}^T\mathbf{x}(n-1) \tag{4.17}$$

Para determinar el valor de los coeficientes óptimos de $\mathbf{w}_{f,M}$, se minimiza el error cuadrático medio (MSE). En este caso, la matriz de autocorrelación es :

$$E[\mathbf{x}(n-1)\mathbf{x}^T(n-1)] = E[\mathbf{x}(n)\mathbf{x}^T(n)] = \mathbf{R}_{\mathbf{x}} = \begin{bmatrix} r_x(0) & r_x(1) & \cdots & r_x(M-1) \\ r_x(1) & r_x(0) & \cdots & r_x(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_x(M-1) & r_x(M-2) & \cdots & r_x(0) \end{bmatrix} \tag{4.18}$$

Por otra parte, también se tiene la correlación definida mediante la siguiente ecuación:

$$r_f = E[\mathbf{x}(n-1)x(n)] = [r_x(1), r_x(2), \cdots, r_x(M)]^T \tag{4.19}$$

Así como en el caso del filtro Wiener la solución proporcionada será mediante los valores de $\mathbf{w}_{f,M}$, como en la ecuación 4.17 , $\mathbf{R}_{\mathbf{x}}\mathbf{w}_{f,M} = \mathbf{r}_f$.

Es posible obtener un valor de potencia que relaciona el valor de la autocorrelación en el tiempo inicial con la ponderación del vector de correlaciones.

$$P_{f,M} = r_x(0) - \mathbf{r}_f^T\mathbf{w}_{f,M} \tag{4.20}$$

Combinando las dos últimas ecuaciones se tiene la llamada *ecuación de Wiener-Hopf aumentada*

$$\begin{bmatrix} r_x(0) & \mathbf{r}_f^T \\ \mathbf{r}_f & \mathbf{R}_{\mathbf{x}} \end{bmatrix} \mathbf{a}_M = [P_{f,M} \quad \mathbf{0}_{L\times 1}] \tag{4.21}$$

donde $\mathbf{a}_M = [1 - \mathbf{w}_{f,M}^T]$.

En este proceso para calcular el valor $x(n)$ se puede entonces utilizar sólo los $M - i$ valores mas recientes, lo cual genera un error de predicción igual a :

$$e_{f,M-i} = x(n) - \sum_{j=1}^{M-i} w_j x(n-j) \tag{4.22}$$

utilizando el principio de ortogonalidad, se llega al siguiente resultado para la condición óptima en el intervalo $1 \leq i \leq M$.

$$E[e_{f,M}(n)e_{f,M-i}(n-i)] = E\left\{e_{f,M}(n)\mathbf{a}_{M-i}^T[x(n-i),\cdots,x(n-M)]^T\right\} = 0 \tag{4.23}$$

Por lo tanto, cuando $M \to \infty$, entonces $E[e_f(n)e_f(n-i)] = 0$, lo que significa que la secuencia de errores hacia adelante es asintóticamente un "ruido blanco" .

Considerando que un filtro adaptivo consiste en poder obtener un conjunto de coeficientes óptimos para cada instante de tiempo *n*, se realiza entonces el proceso de actualizar los coeficientes de manera que se tiene una función de la forma siguiente:

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \mu\nabla J_{MSE}(\mathbf{w}) \tag{4.24}$$

donde $\mu f(\mathbf{w}_n)$ es un factor de corrección que se aplica a los coeficientes $\mathbf{w}_n$. El método utilizado para aplicar dicho factor debe ser tal que el conjunto de coeficientes $\mathbf{w}_n$ converja a la solución óptima de Wiener de la ecuación 4.16.

El algoritmo LMS (por sus siglas en inglés de Least Mean Square), realiza una estimación instantánea de dichos parámetros mediante la siguiente regla de aprendizaje:

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \mu(\mathbf{R}_x - \mathbf{r}_f) = \mathbf{w}_n + \mu\mathbf{x}(n)e(n) \tag{4.25}$$

donde en este caso $e(n)$ es el error de predicción.

El algoritmo LMS cuenta con la característica de convergencia desde un conjunto de coeficientes iniciales $\mathbf{w}_0$ hasta el conjunto de coeficientes óptimos $\mathbf{w}_{opt}$, considerando el valor de μ .

El filtro LMS normalizado es una variante del algoritmo LMS, el cual hace que la constante μ sea dependiente los valores del vector $\mathbf{x}(n)$, de tal manera que los coeficientes son calculados de acuerdo a la siguiente ecuación:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu_n\mathbf{x}(n)e(n) \qquad donde \quad \mu_n = \frac{\mu}{\mathbf{x}^T(n)\mathbf{x}(n)} \tag{4.26}$$

En ocasiones los valores de $\mathbf{x}(n)$ pueden llegar a ser muy pequeños, de tal manera que esto causaría que la ganancia fuera demasiado grande. Para evitar esto se propone una constante que limite el problema de la ganancia, entonces se tiene la siguiente ecuación:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \frac{\mu_(n)}{\delta + \mathbf{x}^T(n)\mathbf{x}(n)}\mathbf{x}(n)e(n) \tag{4.27}$$

4.3 Filtro de Kalman

El filtro de Kalman y sus variantes es uno de los algoritmos mas populares en el campo del procesamiento de la información. Hoy en día se cuenta con este filtro en diferentes áreas como por ejemplo en elementos de navegación, juegos de computadora e incluso en los teléfonos móviles [12].

El modelo del filtro de Kalman asume que el estado de un sistema en el tiempo *t* surgió de un estado anterior en el tiempo $t-1$ de acuerdo a la ecuación siguiente:

$$\mathbf{x}_t = \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{u}_t + \mathbf{n}_t \tag{4.28}$$

donde:

- $\mathbf{x}_t$ es el vector de estados que contiene las variables de interés, ya sea posición o velocidad, en el tiempo *t*.
- $\mathbf{u}_t$ es el vector que contiene cualquier señal de entrada de control.
- $\mathbf{F}_t$ es la matriz de transición de estados, la cual aplica el efecto de cada parámetro del estado del sistema en el tiempo $t-1$ sobre el estado del sistema en el tiempo *t*.
- $\mathbf{B}_t$ es la matriz con las entradas de control las cuales aplican el efecto de cada parámetro de entrada en el vector $\mathbf{u}_t$ y sobre el sistema de estados.
- $\mathbf{n}_t$ es el vector que contiene el ruido del proceso para cada parámetro en el vector de estados. El ruido del proceso se asume que es una distribución normal multivariante con media cero y con una varianza dada por la matriz $\mathbf{Q}_t$.

Las mediciones del sistema también pueden ser manejadas de acuerdo al siguiente modelo

$$\mathbf{z}_t = \mathbf{H}_t\mathbf{x}_t + \mathbf{v}_t \tag{4.29}$$

- $\mathbf{z}_t$ es el vector de mediciones de la salida del sistema.

– $\mathbf{H}_t$ es la matriz de transformación que mapea los parámetros del vector de estado al dominio de mediciones.

– $\mathbf{v}_t$ es el vector que contiene los términos de ruido en las mediciones para cada observación en el vector de mediciones. Similar al ruido del proceso, se asume que este ruido es del tipo Gaussiano con media cero y con una covarianza $\mathbf{R}_t$.

El estado del sistema $\mathbf{x}_t$ no puede ser observado directamente y el filtro de Kalman provee un algoritmo para determinar una estimación $\hat{\mathbf{x}}_t$ al combinar los modelos del sistema y las mediciones con ruido de ciertos parámetros o funciones lineales de los parámetros. Las estimaciones de los parámetros de interés del vector de estados son proporcionadas entonces por las funciones de densidad de probabilidad (pdfs), en lugar de valores discretos [30].

Debido a que el filtro de Kalman está basado en distribuciones Gaussianas, es necesario conocer las varianzas y covarianzas de las funciones Gaussianas y estos valores pueden ser representados mediante una matriz $\mathbf{P}_t$, llamada matriz de covarianzas. Los términos en la diagonal principal de la matriz $\mathbf{P}_t$ son las varianzas asociadas con los términos correspondientes en el vector de estados, por lo tanto los valores fuera de la diagonal representan las covarianzas entre los términos del vector de estados.

El filtro de Kalman involucra dos etapas, la primera es denominada etapa de *predicción* y la segunda se denomina etapa de *actualización* de las mediciones. Para la etapa de predicción las ecuaciones del filtro de Kalman son:

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F}_t\hat{\mathbf{x}}_{t-1|t-1} + \mathbf{B}_t\mathbf{u}_t \tag{4.30}$$

$$\mathbf{P}_{t|t-1} = \mathbf{F}\mathbf{P}_{t-1|t-1}\mathbf{F}_t^T + \mathbf{Q}_t \tag{4.31}$$

La varianza associada con la predicción $\hat{\mathbf{x}}_{t|t-1}$ de un valor verdadero desconocido $\mathbf{x}_t$ está dado por:

$$\mathbf{P}_{t|t-1} = E[(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})^T] \tag{4.32}$$

y tomando la diferencia entre las ecuaciones 4.28 y 4.30 se llega al siguiente resultado

$$\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1} = \mathbf{F}(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1}) + \mathbf{n}_t \tag{4.33}$$

$$\begin{aligned} \Rightarrow \mathbf{P}_{t|t-1} &= E[(\mathbf{F}(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1}) + \mathbf{n}_t) \times (\mathbf{F}(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1}) + \mathbf{n}_t)^T] \\ &= \mathbf{F}(E[(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})])\mathbf{F}^T + \mathbf{F}E[(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})\mathbf{n}_t^T] \\ &\quad + E[\mathbf{n}_t(\mathbf{x}_{t-1} - \hat{\mathbf{x}}_{t-1|t-1}^T)]\mathbf{F}^T + E[\mathbf{n}_t\mathbf{n}_t^T] \end{aligned} \tag{4.34}$$

y dado que los errores de estimación de estado y el ruido del proceso no están correlacionados

$$\begin{aligned} E[(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})\mathbf{n}_t^T] = E[\mathbf{n}_t(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})^T] = 0 \\ \Rightarrow \mathbf{P}_{t|t-1} = \mathbf{F}(E[((\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})])\mathbf{F}^T + E[\mathbf{n}_t\mathbf{n}_t^T] \end{aligned} \tag{4.35}$$

Por otra parte, las ecuaciones para la actualización de las mediciones están dadas por

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{z}_t - \mathbf{H}_t\hat{\mathbf{x}}_{t|t-1}) \tag{4.36}$$

$$\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{K}_t\mathbf{H}_t\mathbf{P}_{t|t-1} \tag{4.37}$$

donde

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}_t^T(\mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t^T + \mathbf{R}_t)^{-1} \tag{4.38}$$

En relación con el desempeño computacional, el fitro de Kalman es muy eficiente. Actualmente se tienen algoritmos cuya complejidad de la inversión de la matriz es aproximadamente del orden *O(d)* para una matriz de tamaño $d \times d$.

La figura 4.3.1 presenta un escenario donde se da la localización unidimensional de un robot utilizando el filtro de Kalman, donde el robot se supone que hace un movimiento sobre el eje horizontal y esto se representa en la figura. En la figura 4.3.1 (a) se tiene una distribución inicial de la posición inicial, en la figura 4.3.1 (b) se ha agregado una distribución obtenida por las mediciones de los sensores del robot acerca de su posición, en la figura 4.3.1 (c) se combinan las distribuciones mediante el algoritmo de Kalman. En la figura 4.3.1 (d) se grafica la distribución una vez que el robot realiza un movimiento, en la figura 4.3.1 (e) nuevamente se realizan una segunda medición y se obtiene una distribución con cierta incertidumbre y por último la figura 4.3.1 (f) se obtiene nuevamente una distribución mediante el algoritmo de Kalman.

Entonces el robot tiene en un principio su localización dada por una distribución normal y posteriormente el robot realiza las mediciones con una incertidumbre asociada, de tal manera que el filtro de Kalman entonces integre la posición inicial y las mediciones para predecir la posición, a medida que el robot se desplaza se tiene que realizar un proceso similar y tratar de obtener de nuevo la estimación de la posición.

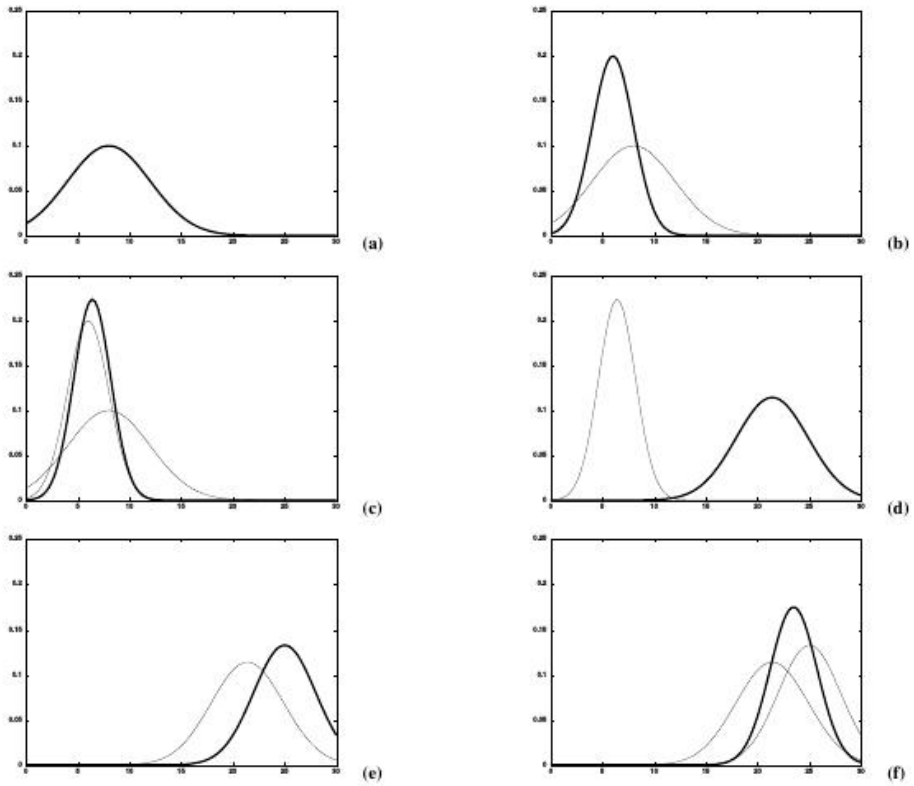


Figura 4.3.1: Ejemplo del filtro de Kalman [30]: (a) distribución de posición inicial, (b) se realiza la medición y se obtiene la distribución con una incertidumbre asociada, (c) distribución resultado del filtro de Kalman con base a (a) y (b), (d) distribución después de un desplazamiento, (e) una nueva medición con cierta incertidumbre y (f) la distribución final resultado de aplicar el filtro de Kalman

.

4.4 Filtro de partículas.

El filtro de partículas es una técnica general del método Monte Carlo para el manejo de la inferencia en los modelos espacio-estado, donde el estado de un sistema evoluciona en el tiempo y la información acerca del estado es obtenido mediante las mediciones que contienen cierto ruido, realizadas en cada instante de tiempo del proceso. El método de Monte Carlo se define como la técnica de aproximar una expectativa mediante la muestra media de una función de variables aleatoria simulada. En general, en un modelo de tiempo discreto espacio-estado, el estado de un sistema evoluciona de acuerdo a la siguiente ecuación:

$$\mathbf{x}_k = \mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{v}_{k-1}) \tag{4.39}$$

donde $\mathbf{x}_k$ es un vector que representa el estado de un sistema en el tiempo k, $\mathbf{v}_{k-1}$ es el vector que representa la distribución de una secuencia de ruido, $\mathbf{f}_k$ es una función posiblemente no lineal y dependiente del tiempo, tal que dicha función describe la evolución del vector de estado. El vector de estado $\mathbf{x}_k$ se asume que es no observable o latente. La

información acerca de $\mathbf{x}_k$ es obtenida únicamente a través de mediciones con ruido $\mathbf{z}_k$, las cuales están dados por la ecuación:

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{n}_k) \tag{4.40}$$

donde $\mathbf{h}_k$ es una función posiblemente no lineal y dependiente del tiempo que describe el proceso de mediciones y $\mathbf{n}_k$ es el vector que representa la distribución del ruido en una secuencia de mediciones.

El problema de filtrado involucra la estimación del vector de estado $\mathbf{x}_k$ en un tiempo *k*, considerando todas las mediciones e incluyendo aquellas del tiempo *k*, las cuales se denotan por $\mathbf{z}_{1:k}$. En un sentido Bayesiano, este problema puede ser formalizado como el cálculo de la distribución $p(\mathbf{x}_k|\mathbf{z}_{1:k})$, este cálculo puede ser llevado a cabo en dos etapas. La primera etapa se denomina proceso de *predicción*, en el cual se calcula $p(\mathbf{x}_k|\mathbf{z}_{1:k-1})$ a partir de la distribución $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})$ en el tiempo $k-1$, por la ecuación:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})d\mathbf{x}_{k-1} \tag{4.41}$$

donde $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})$ se asume que es conocida debido a la recursión y $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ es proporcionado por la ecuación 4.39. La distribución $p(\mathbf{x}_k|\mathbf{z}_{1:k-1})$ puede ser considerada como una distribución *a priori* sobre $\mathbf{x}_k$ antes de considerar las mediciones mas recientes $\mathbf{z}_k$.

La segunda etapa es denominada *actualización*, donde una distribución *a priori* es actualizada con la nueva medición $\mathbf{z}_k$, utilizando la regla de Bayes para obtener una distribución *a posteriori* sobre $\mathbf{x}_k$:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \propto p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) \tag{4.42}$$

En general, los cálculos de las etapas de predicción y de actualización no pueden ser llevados a cabo analíticamente, por lo tanto la necesidad de métodos de aproximación como lo es el muestreo de Monte Carlo. En algunos casos muy restrictivos, los cálculos pueden ser llevados a cabo de manera analítica.

Si la distribución en el tiempo $k-1$, $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})$, es del tipo gaussiano, la distribución en el próximo instante de tiempo, $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ puede mostrarse que también es del tipo gaussiano si las siguientes condiciones se cumplen:

- las funciones de evolución de estado y mediciones, $\mathbf{f}_k$ y $\mathbf{h}_k$ son lineales
- el ruido en las mediciones y en el estado, $\mathbf{v}_k$ y $\mathbf{n}_k$ son Gaussinas.

En este caso, las ecuaciones de las funciones de medición y de estado se reducen a las siguientes formas:

$$\mathbf{x}_k = F_k\mathbf{x}_{k-1} + \mathbf{v}_{k-1} \tag{4.43}$$
$$\mathbf{z}_k = H_k\mathbf{x}_k + \mathbf{n}_k \tag{4.44}$$

donde $\mathbf{v}_{k-1}$ y $\mathbf{n}_k$ son variables aleatorias Gaussianas y F_k y H_k son las matrices de medición y evolución del estado que se asume son conocidas. El cálculo de las ecuaciones de predicción y actualización en este caso tipo lineal y Gaussiano, conduce al algoritmo del filtro de Kalman, descrito en la sección 4.3.

Uno de los tipos mas básicos de muestreo Monte Carlo para el propósito dirigido es el denominado *Sequential Importance Sampling* (SIS). SIS considera la distribución total posterior en el tiempo $k-1$, $p(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1})$, antes que utilizar un distribución filtrada, $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})$, la cual es sólo la distribución marginal de la distribución posterior con respecto a $\mathbf{x}_{k-1}$. El concepto de SIS es aproximar la distribución posterior en el tiempo $k-1$, $p(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1})$, mediante un conjunto ponderado de muestras $\{\mathbf{x}^i_{0:k-1}, w^i_{k-1}\}^N_{i=1}$, las cuales son denominadas *partículas*, y actualizar estas *partículas* de manera recursiva, para obtener una aproximación a la distribución posterior en el próximo instante de tiempo $p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$.

SIS está basado en el muestreo de *importancia*, en el cual se trata de aproximar una distribución objetivo $p(x)$, utilizando muestras a partir de una distribución propuesta $q(x)$ denominada *densidad de importancia* [29]. El muestreo de importancia es usado generalmente cuando es difícil obtener una muestra directamente a partir de la distribución objetivo. Por lo tanto, si se dispone de una función $\pi(x)$ proporcional a $p(x)$, de la cual se tiene la manera de evaluar y además se tienen las muestras $x_i \backsim q(x), i = 1, ..., N$ extraídas a partir de la distribución propuesta $q(x)$, aplicados a la distribución posterior en el tiempo $k-1$, la importancia de muestro produce:

$$p(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1}) \approx \sum_{i=1}^{N} w^i_{k-1}\delta_{\mathbf{x}^i_{0:k-1}} \tag{4.45}$$

donde $\delta_{\mathbf{x}^i_{0:k-1}}$ es la función delta centrada en $\mathbf{x}^i_{0:k-1}$ y w_i es el peso de la i-ésima partícula ($w_i \propto \pi(x^i)/q(x^i)$) . La idea fundamental de SIS es actualizar las partículas $\mathbf{x}^i_{0:k-1}$ y sus *pesos* correspondientes de tal forma que éstos se puedan aproximar a la distribución posterior en el siguiente instante de tiempo: $p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$. Para llevar a cabo esto, lo primero que se tiene que asumir es que la distribución propuesta en el tiempo k puede ser factorizada tal que:

$$q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}) = q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k})q(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1}) \tag{4.46}$$

de esta manera se puede ampliar cada una de las partículas $\mathbf{x}^i_{0:k-1}$ a un nuevo estado $\mathbf{x}^i_k$ en el tiempo k a partir del muestreo de $q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k})$. Para actualizar los pesos, w^i_{k-1} en el tiempo k se calcula de la siguiente forma:

$$w^k_i \propto \frac{p(\mathbf{x}^i_{0:k}|\mathbf{z}_{1:k})}{q(\mathbf{x}^i_{0:k}|\mathbf{z}_{1:k})} \tag{4.47}$$

Se puede expresar la ecuación anterior en términos de lo que sería w_{k-1}^i, resultando:

$$w_k^i \propto w_{k-1}^i \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{0:k-1}^i, \mathbf{z}_{1:k})} \tag{4.48}$$

En la figura 4.4.1 se muestra de manera gráfica el concepto de lo que es el filtro de partículas, la figura 4.4.1(a) se ilustra una función de densidad f de una distribución de probabilidad que es la función objetivo y en la figura 4.4.1(b) se tiene la función propuesta g.

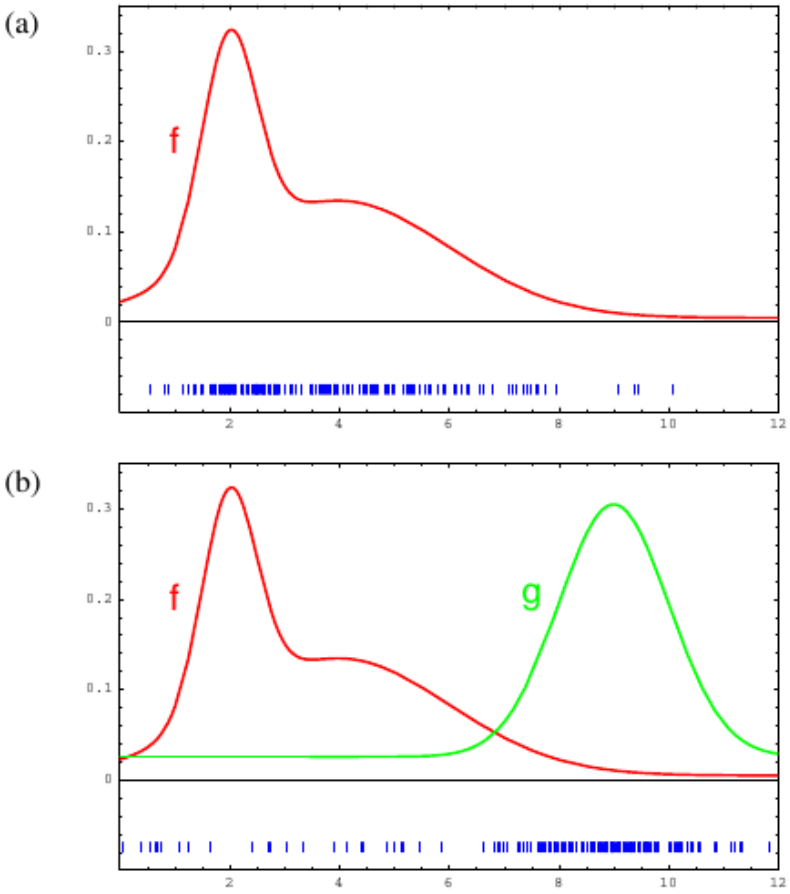


Figura 4.4.1: Filtro de partículas [30]. En (a) se tiene la función objetivo la función f la cual se busca aproximar. En (b) en lugar de tomar muestras a partir de esta función se hace uso de una función propuesta g, en la parte inferior de las gráficas se observan las muestras extraídas de la función propuesta.

Los pesos $\{w_k^i\}_{i=1}^N$ son normalizados de manera que la suma de todos estos es igual a 1. Si se asume que $q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k}) = q(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_{1:k})$, de tal forma que la distribución propuesta en el siguiente instante de tiempo solo depende del reciente estado y de la medición mas reciente, entonces solo es necesario mantener $\mathbf{x}_{k-1}^i$ y generar la partícula en el siguiente instante de tiempo a partir de la distribución $q(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_{1:k})$. Por lo tanto en este caso, las ecuaciones de actualización se simplifican quedando:

$$\mathbf{x}_k^i \sim q(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_k) \tag{4.49}$$

$$w_k^i \propto w_{k-1}^i \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k)} \tag{4.50}$$

A medida que k va en aumento sólo algunas partículas tendrán un peso muy significativo mientras otras partículas tendrán un valor muy pequeño, esto produce un problema denominado *degeneración*. Esta degeneración es medida al estimar el tamaño de las muestras efectivas:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N}(w_k^i)^2} \tag{4.51}$$

donde un valor pequeño de N_{eff} significa una gran variación en los pesos, por lo tanto una mayor degeneración.

Una manera de tratar con el problema de degeneración es mediante la técnica *remuestreo*, en este remuestro con reemplazo, se maneja un nuevo conjunto de N partículas de una aproximación discreta a la distribución $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ obtenida a partir de:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta_{\mathbf{x}_k^i} \tag{4.52}$$

El remuestreo es llevado a cabo cuando el tamaño efectivo de las muestras N_{eff} alcanza un valor menor a un cierto umbral. Se debe tomar en cuenta que dado que se esta realizando un muestreo con reemplazo, una partícula con una gran peso tiene mas probabilidad de ser extraída en varias ocasiones, mientras que partículas con pesos pequeños seguramente no serán extraídas. Con el remuestreo, el peso de las partículas llega a ser entonces $1/N$, entonces de esta forma el problema de degeneración es tratar de al eliminar las partículas con pesos muy pequeños.

Al evitar el problema de degeneración mediante el remuestreo se genera un nuevo problema denominado *empobrecimiento del muestreo*, esto es debido a que al ir eliminando cada vez las partículas con un peso pequeño, la diversidad de las partículas tenderá a disminuir después de llevar a cabo el remuestreo consecutivamente. Y en el caso extremo de esta situación todas las partículas *colapsarián* llegando a contar con sólo una partícula, lo cual afectaría negativamente la calidad de la aproximación ya que en lugar de contar con una cierto número de partículas que representen toda la distribución objetivo, se estaría tratando de representar la distribución mediante una sola partícula.

La mayoría de los algoritmos de filto de partículas cuentan con variaciones del método de SIS, uno de ellos es denominado SIR (de sus siglas en inglés *Sampling Importance Resampling*), en el cual la distribución propuesta $q(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_k)$ es considerada como la

distribución de transición de estado $p(\mathbf{x}_k|\mathbf{x}_{k-1}^i)$ y el remuestreo es aplicado en cada iteración. Por lo tanto en el algoritmo SIR, las ecuaciones de la etapa de actualización están dadas por :

$$\mathbf{x}_k^i \sim p(\mathbf{x}_k|\mathbf{x}_{k-1}^i) \tag{4.53}$$
$$w_k^i \propto p(\mathbf{z}_k|\mathbf{x}_k^i) \tag{4.54}$$

Cabe notar que en la ecuación 4.54 de actualizacion el término w_{k-1}^i ya no está presente debido a que después del remuestreo en el tiempo $k-1$ la suma de todos los pesos llega a ser igual a $1/N$. Las desventajas del algoritmo SIR son:

- la independencia de la distribución propuesta $p(\mathbf{x}_k|\mathbf{x}_{k-1}^i)$ de las observaciones $\mathbf{z}_k$ lo cual significa que la etapa de actualización del estado no toman en cuenta la información proporcionada por las observaciones
- el problema del empobrecimiento de las muestras el cual puede llegar a ser muy notable debido a que se realiza el remuestreo en cada iteración.

Otra variante del algoritmo SIS es el denominado *filtro partícula regularizado* [10], el cual también consta de un remuestreo, pero en este caso se trata de resolver el problema de empobrecimiento de muestras. En el método filtro de partículas regularizado se trata de aproximar la distribución total posterior mediante un kernel de densidad que utilizan las partículas en lugar de utilizar directamente las partículas. Por lo tanto se tiene:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N} w_k^i K(\mathbf{x}_k - \mathbf{x}_k^i) \tag{4.55}$$

donde $K(\mathbf{x}_k - \mathbf{x}_k^i)$ es una función kernel centrada en $\mathbf{x}_k^i$. Para el caso de pesos iguales es posible determinar analíticamente la forma y el ancho de banda de la función *kernel* óptima que minimiza una cierta distancia entre el filtrado original y la estimación del kernel de densidad obtenida mediante la ecuación 4.55 . El algoritmo del filtro de partículas regularizado consiste entonces de las ecuaciones de actualización del algoritmo SIS combinado con el remuestreo de N nuevas partículas de acuerdo a la ecuación 4.55 siempre que resulte que el valor de la muestra efectiva N_{eff} es menor a un cierto valor de umbral N_T.

En este capítulo se resalta que el descriptor HOG es la herramienta que permite la detección de una persona en una escena, ya que mediante este descriptor y su evaluación a través de una SVM, que ya cuenta con un entremamiento previo, es posible determinar y clasificar si una región podría contener a una persona. En el presente trabajo no se realiza un análisis de toda la escena, ya que se toman solo ciertas regiones que podrían contener a una persona. De tal manera que sólo se analizan regiones de tamaño 128×64, como se describe en la sección 4.1.

De los fitros que se explican un poco más a detalle en las secciones 4.2, 4.3 y 4.4. Se consideraron solamente el filtro LMS y el filtro de partículas, para poder realizar el seguimiento

de una persona detectada en escena.

Capítulo 5

Implementación del sistema

Dadas las herramientas necesarias que se tienen y se han descrito a grandes rasgos en los capítulos tres y cuatro, para poder llevar a cabo la detección y el seguimiento de una persona se procede a elaborar un sistema el cual cuenta con la capacidad de realizar el proceso en conjunto. En este caso se puede separar cada uno de las etapas para llevar a cabo el proceso. En la figura 5.0.1 se muestra un esquema general que indica el sistema utilizado.

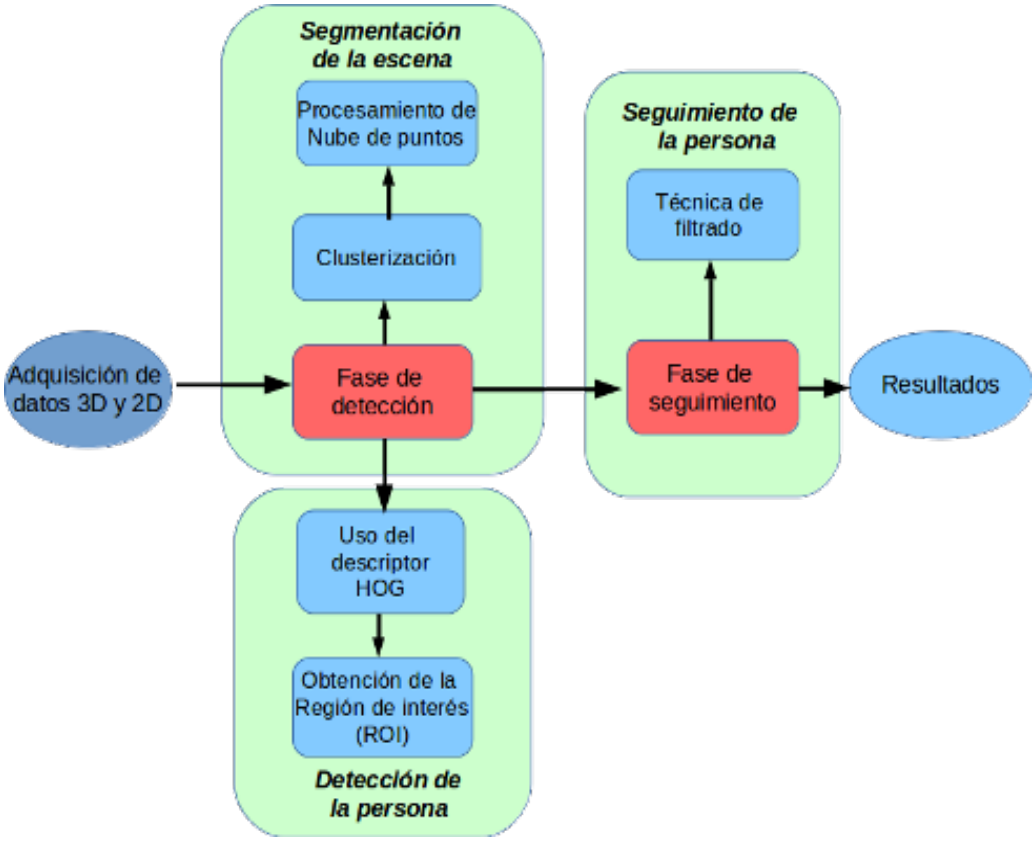


Figura 5.0.1: Esquema del sistema

El proceso de detección consta básicamente de tres etapas que son: detección de la superficie que será considerada como el piso, la segmentación de la nube de puntos para obtener los clusters que serán analizados y por último se realiza la posible identificación de los clusters considerados como personas.

Por otra parte, en el manejo del proceso de seguimiento, se consideran dos etapas las cuales son: el uso del filtro de partículas como parte fundamental en el seguimiento y el uso de histogramas que son considerados dentro del filtro de partículas, que están relacionados con el cálculo del peso de las partículas. En la figura 5.0.2 se tiene un esquema que muestra lo descrito anteriormente.

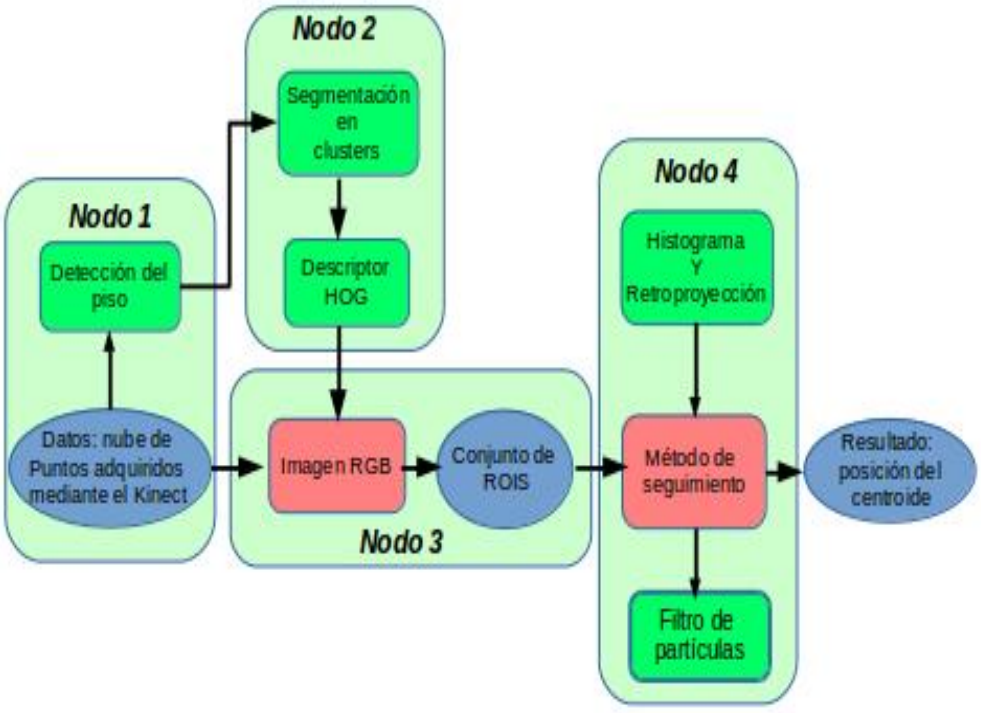


Figura 5.0.2: Etapas de la detección y seguimiento

5.1 Nodos en Robot Operating System (ROS)

En este trabajo se hace uso de lo que es la plataforma Robot Operating System (ROS) [28], por lo tanto se cuentan con los nodos que se sincronizan de tal manera que a pesar de realizar cada uno su función se logre el resultado deseado; esto es una de las facilidades con las que cuenta el entorno ROS y por lo tanto su elección para formar parte de este trabajo.

El primer nodo, es el de la adquisición de datos mediante el dispositivo Kinect, la plataforma ROS ya cuenta con un archivo que carga las funciones para la adquisición de la información y en este caso sólo se eligen los tópicos para obtener la imagen de color, los valores de disparidad y la nube de puntos.

El segundo nodo, tiene la función de trabajar con la nube de puntos para realizar la fase de la detección, ya que a partir de este nodo se obtiene parte de la información que determina la posibilidad de encontrar conjuntos de puntos en una escena que pueden corresponder a una persona y a partir de esto obtener una región de interés.

El tercer nodo, realiza el análisis de la información de la imagen RGB y las regiones de interés proporcionadas por el segundo nodo, el tercer nodo por lo tanto sólo nos permite verificar que la información proporcionada corresponda al de una persona. El resultado se puede observar en una imagen que es proporcionada por otro nodo, al cual le es enviado el resultado.

El cuarto nodo, es el que realiza el seguimiento de la persona detectada por los nodos anteriores, en este cuarto nodo solamente se utiliza la información de la imagen RGB, las regiones de interés y los valores de disparidad proporcionados por el dispositivo Kinect. El resultado final se muestra en la secuencia de imágenes, donde se marca la región de interés y el centroide de la misma.

Finalmente se cuenta con un nodo de apoyo para la visualización de imágenes RGB, que es utilizado en conjunto con cualquier otro nodo que proporcione la información necesaria para que este nodo pueda realizar su función.

5.2 Detección y seguimiento.

La fase de detección se lleva a cabo mediante la adquisición de datos a través del dispositivo Kinect considerando para ello los puntos en el espacio 3D y se hace uso de la clase *GroundBasePeopleDetector* de la librería Point Cloud Library (PCL) [26], que tiene como primer paso el poder dividir la escena en la que se está trabajando en diversos clusters y para cada uno de estos clusters se lleva a cabo un análisis para calcular la característica que indicará la posibilidad de que el cluster evaluado corresponda al de una persona.

Entonces lo primero que se obtiene es la nube de puntos a través del dispositivo Kinect, de la cual se eligen de manera manual tres puntos de la escena que correspondan al piso, y mediante estos tres puntos se podrán obtener los coeficientes del plano correspondiente al piso. Por lo tanto seleccionando seleccionando los puntos $A(x_1, y_1, z_1)$, $B(x_2, y_2, z_2)$ y $C(x_3, y_3, z_3)$ se puede resolver el sistema de ecuaciones siguiente para obtener los coeficientes *a, b, c, d* de un plano, dado por las ecuaciones:

$$\begin{aligned} ax_1 + by_1 + cz_1 + d = 0 \\ ax_2 + by_2 + cz_2 + d = 0 \\ ax_3 + by_3 + cz_3 + d = 0 \end{aligned} \tag{5.1}$$

Debido a que los puntos seleccionados generalmente contienen cierto ruido y por lo tanto no son precisos, la manera en que se obtienen los puntos que corresponden al plano del piso a partir de la nube de puntos, es utilizando el método de RANSAC (por sus siglas en inglés *Random Sample Consensus*) [14], el cual es un método iterativo para estimar los parámetros de un modelo matemático de un conjunto de datos observados que contiene ciertas anomalías. El uso de RANSAC es de gran utilidad ya que a partir de haber obtenido la ecuación del plano, en cada frame será utilizado para continuar obteniendo el plano. En la figura 5.2.1 se tiene la imagen de los puntos seleccionados.

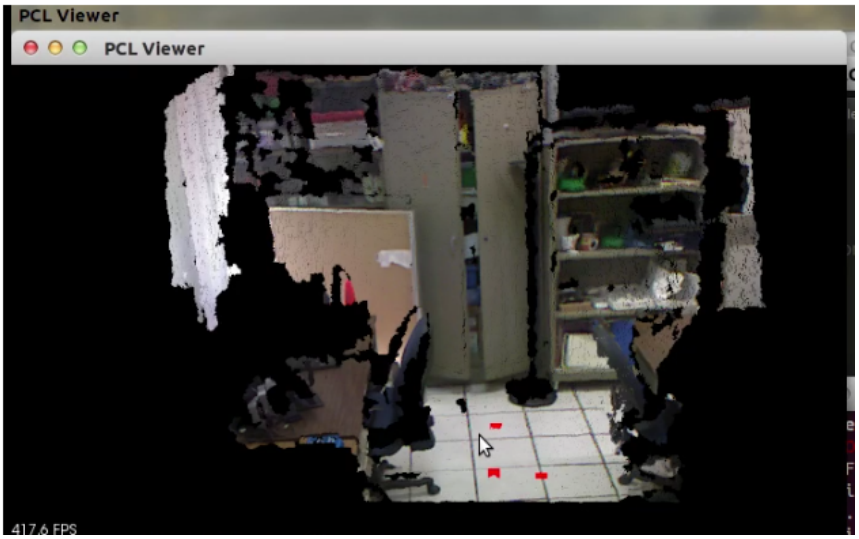


Figura 5.2.1: Puntos seleccionados en color rojo

Para poder obtener los *clusters* que se pueden formar dentro de una escena se utiliza la clase *EuclideanClusterExtraction* de la librería PCL. A partir del conjunto de puntos P obtenida mediante el dispositivo Kinect,se crea una lista C de grupos (clusters) y un conjunto Q de puntos que serán checados. Cada punto p_i del conjunto P se agrega a al conjuto Q y se obtienen los vecinos mas cercanos del p_i que se encuentran dentro de un radio r, hasta que todos los puntos vecinos de p_i que ya están considerados dentro del conjunto Q son verificados y en caso de que algún punto vecino de p_i no haya sido analizado se agrega al conjunto Q. Cuando todos los puntos del conjunto Q han sido analizados, se agrega a la lista de clusters y se reinicia el conjunto Q para ser verificado. Este proceso continua

hasta que todos los puntos p_i del conjunto P han sido analizados. Por lo tanto ésta clase se utiliza para dividir a la nube de puntos en clusters de acuerdo a una tolerancia definida por el tamaño del espacio (*voxel*) y poder remover aquellos clusters cuyo número de puntos esta fuera de ciertos límites establecidos.

La fase de evaluación de los clusters para poder tener una estimación de que estos corresponden a una persona, comienza al evaluar los clusters por una distancia medida a partir de la posición del dispositvo Kinect; una vez ordenados los clusters son evaluados para determinar su altura y ser eliminados aquellos que no cumplan con un cierto rango.

La clase *GroundBasedPeopleGround* permite realizar el análisis de los clusters que sean obtenidos mediante un proceso como el realizado por *EuclideanClusterExtraction*. La clase *GroundBasedPeopleGround* cuenta con la facilidad de calcular tres centroides que pertencen de una manera estimada al centro, la parte mas baja y la parte mas alta del cluster. La altura es calculada como la distancia entre el punto mas alto del cluster y lo que correspondería el plano del piso,es decir, dados los coeficientes del plano *(a, b, c, d)* y el punto $H = (x_h, y_h, z_h)$, la altura es calculada mediante la ecuación :

$$h = \frac{|ax_h + by_h + cz_h + d|}{(a^2 + b^2 + c^2)^{\frac{1}{2}}} \tag{5.2}$$

El centro es el punto geométrico de la forma del cluster, el cual es la media aritmética de todos los puntos del cluster. Entonces si se tiene un cluster con *k* puntos el centroide $cent = (x_c, y_c, z_c)$ se calcula como :

$$C = \frac{1}{k} \sum_{i=1}^{k} P_i \tag{5.3}$$

Debido a las características de los datos obtenido mediante el dispositivo Kinect, se tiene la ventaja de procesar también la información RGB para cada punto, por lo tanto es posible entonces obtener la correspondiente imagen en RGB, con la cual se procede a calcular el coeficiente del descriptor HOG que se lleva a cabo de la manera explicada la sección 4.1 (Descriptor HOG). Por una parte se tiene un archivo que ha sido obtenido al realizar el entrenamiento para lo que es el clasificador basado en una SVM y en este caso el parámetro con el que se cuenta para poder determinar que el cluster posiblemente pertence a una persona es el valor de un umbral llamado *min_confidence*.

Por lo tanto, de la nube de puntos se obtiene los clusters que pueden pertenecer a una persona, y los parámetros que se consideran para permitir realizar diferentes estimaciones en el proceso de detección basado en la clase *GroundBasedPeopleGround* se muestran en la tabla 1.

Asi mismo en la figura 5.2.2 se observa el resultado del proceso para estimar el lugar donde posiblemente se encuentra una persona, y también en la figura se puede observar

Parámetro	Valor por defecto
people/min_height	1.5m
people/max_height	2.3m
people/min_confidence	-1.5
people/head_centroide	true
cluster/min_points	30
cluster/max_points	600

Tabla 5.2.1: Tabla de parámetros.

Figura 5.2.2: Cluster que indica a una persona

un recuadro que delimita el cluster de puntos que resultaron con un valor de confidencia superior al establecido en el parámetro *people/min_confidence*.

Una vez obtenido el cluster correspondiente a la persona, se toman los centroides correspondientes al centro del cluster y a la parte mas alta del mismo, a partir de estos dos puntos se genera una región de interés (ROI por sus siglás en inglés), la cual será utilizada en la etapa de seguimiento (*tracking*). Debido a que los resultados se muestran en imágenes RGB se hace una conversión de puntos 3D a puntos 2D. La manera en que se hace dicha conversión es mediante los valores de los dos puntos correspondientes a los dos centroides del clúster clasificado como persona y mediante la matriz de parámetros intrínsecos del Kinect. Se realiza entonces el cálculo del valor correspondiente en 2D. Los cálculos son entonces a través de la ecuación :

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \frac{1}{Z} \tag{5.4}$$

Donde la pareja ordenada (u, v) representa el valor de las coordenadas del punto 2D y los valores (X, Y, Z) son los valores del punto en 3D. Así una vez obtenidos los valores de los

Figura 5.2.3: Región de interés del cluster detectado como posible persona

centroides se procede a realizar el cálculo de la región de interés de la siguiente manera:

$$\begin{aligned} R\_height &= cent_y - top_y \\ R\_width &= R\_height * 2/3 \\ R\_x &= \textit{max}(0, cent_x - R\_width/2) \\ R\_y &= \textit{max}(0, top_x) \end{aligned} \tag{5.5}$$

Donde *R_height* y *R_width*, son los valores de la altura y el ancho de la ROI, además *R_x* y *R_y* corresponden a las cordenadas del punto superior izquierdo de la ROI, todos estos valores tienen unidades de pixeles. El resultado se puede observar en la imagen RGB en la figura 5.2.3.

De esta forma los primeros tres nodos continuamente proporcionarán la información que será utilizada por el nodo que realiza la función de seguimiento de la persona detectada en la escena.

En el cuarto nodo cada instante se adquiere la imagen RGB , las regiones de interés y la información de disparidad, ésta última se puede obtener directamente a partir del dispositivo Kinect mediante la cámara IR. Lo primero que se procede a realizar es asociar las regiones de interés a un seguidor, este proceso de asociación consiste en tomar los valores de disparidad que corresponden a cada región y ser proyectadas a un plano XZ donde se localiza el punto que contiene el valor máximo en la región de interés, motivo por el cual el seguidor se dice que se realiza en el espacio 2D.

En la figura 5.2.4 se puede observar las mediciones de la geometría que son obtenidas

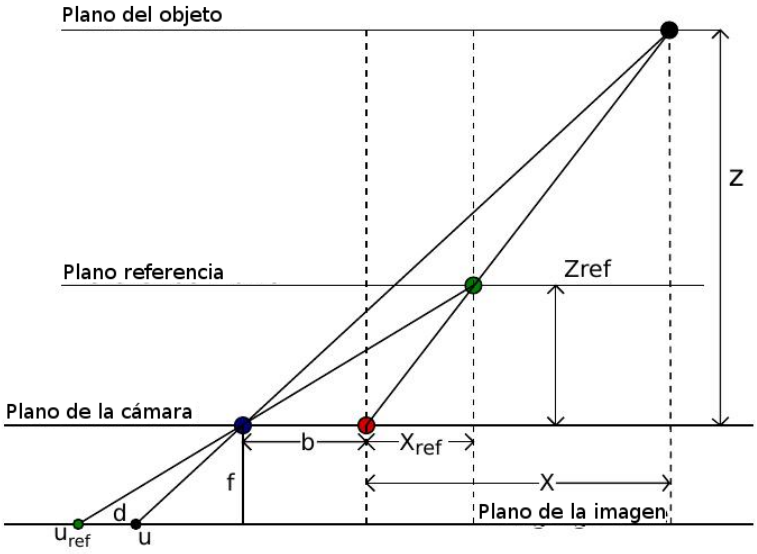


Figura 5.2.4: Geometría de disparidad [21]

mediante el dispositivo Kinect para derivar una relación entre el valor de disparidad *d* y el valor *Z*, el punto azul muestra la localidad de la cámara IR y el punto rojo es la localidad del proyector IR del dispositivo kinect. La distancia focal es f y la "*base*" (separación entre la cámara IR y el proyector IR) es *b*. Las imágenes son formadas a traves del modelo *pinhole* de la cámara IR (punto azul) sobre el plano de la imagen y la escena actual se encuentra delante del plano de la cámara.

El punto verde es la ubicación de una parte del patrón de puntos en el plano de referencia, mientras que el punto negro es la misma parte del patrón colocado en un plano alejado del plano de referencia. X_{ref} es la distancia horizontal del punto de referencia desde el proyector IR, y Z_{ref} es la distancia del punto de referencia. *Z* es el valor del punto que se quiere calcular, y *X* es la distancia horizontal a partir del proyector IR. La disparidad se define como el desplazamiento de pixeles, entre la imagen patrón en el plano de referencia y la imagen proyectada en el plano del objeto [21].

$$d = u - u_{ref} = \frac{bf}{Z} - \frac{b}{Z_{ref}} \tag{5.6}$$

$$Z = \frac{bf}{\frac{bf}{Z_{ref}} - Z} \tag{5.7}$$

Con la información de disparidad I_D se realiza el cálculo de un ángulo θ, que es utilizado para realizar una rotación de los valores de disparidad con respecto al eje X, debido a la posición de la cámara con respecto al eje Y. El procedimiento consiste en obtener un mapa de disparidad I_{vD}, realizando el análisis de I_D [22]. Se agrupan los pixeles que están por encima de un valor umbral a lo largo del eje X, que son los que conforman el mapa I_{vD}. Posteriormente se realiza el cálculo de una recta, haciendo uso de la transformada de

Hogh, la cual es una técnica que permite realizar el cálculo, en este caso de una recta, en términos de los parámetros de la pendiente m y la ordenada b.

Donde θ es el ángulo encontrado con base a la pendiente m, y éste es una aproximación que representa la inclinación debido a la posición de la cámara con respecto al plano del piso. Por lo que la transformación, en este caso sólo es una rotación con respecto al eje X y se lleva a cabo utilizando la ecuación:

$$\begin{bmatrix} x_{rot} \\ y_{rot} \\ z_{rot} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sen(\theta) \\ 0 & sen(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{5.8}$$

Con los valores de disparidad ya transformados y con la región de interés, se hace el cálculo de la ubicación de la región en el plano XZ, así también se obtiene el valor del centroide de la región. La región de interés se considera para realizar el cálculo de su histograma, el cual es considerado en lo consecutivo en la función del seguimiento.

Para el cálculo del histograma la región de interés se toma en la imagen RGB y ésta se convierte al sistema HSV, ya que en este sistema las variaciones de iluminación en la imagen se ven menos afectadas; sólo se utilizan los canales H y S para los cálculos del histograma, donde sólo se propone el conteo en cincuenta rangos en cada canal.

Por último una vez que se cuenta con un seguidor, se procede a la inicialización del filtro de particulas que comenzará el proceso de seguimiento. La estructura utilizada para el filtro es de la siguiente forma:

- Un vector *x* que contendrá el valor del centroide.
- Un valor de peso *weight* correspondiente a la partícula.
- Un valor de peso *preweight* que se utiliza para almacenar el peso anterior correspondiente a la partícula.

Otro parámetro a considerar para el caso del filtro de particulas es el número de partículas con el que se inicializa mismo, en este trabajo se considera $N = 500$ partículas. Con respecto al valor del peso para cada una de las partículas se considera el muestro aleatorio de una distribución uniforme $U(0, 1)$ como función de aproximación.

El nodo de seguimiento al recibir la primera región de interés, se inicializa un filtro de partículas y entonces se cuenta con un seguidor inicial que es almacenado, por lo tanto en la siguiente captura de información los nodos nuevamente proporcionarán regiones de interés que podrían considerarse como a una persona. De tal manera que el nodo de seguimiento realizará el cálculo de correspondencia entre las regiones de interés y el seguidor con el que se cuenta ya en este instante.

Para llevar a cabo tal correspondencia se hace una matriz de puntaje de la cantidad de regiones de interés con respecto a la cantidad de seguidores; en caso de que se contará en principio con más de uno. Entonces se realiza el cálculo de la distancia entre el centroide del seguidor y el centroide de cada una de las regiones de interés, que se coloca como el puntaje en la matriz y así de esta manera la región más cercana se considera que es la correspondiente con el seguidor, en la matriz de puntaje se elimina la columna de detecciones que ya ha sido relacionada con el seguidor.

En el nodo de seguimiento aparte de realizar la correspondencia entre las regiones de interés y el seguidor, se realiza la suma de un nuevo seguidor de manera similar al de la correspondencia, es decir, se calcula el puntaje y si este es menor a un cierto valor se dice que no hay una región de interés asociada, por lo cual se genera un nuevo seguidor y se inicializa por lo tanto un nuevo filtro de partículas.

La parte más importante es la etapa de actualización del seguimiento, ésta hace uso del histograma que se había calculado previamente y lo que se intenta es actualizar la región donde podría encontrarse en este caso la persona a seguir, para ello se hace uso de retroproyección.

La retroproyección de histogramas es usado en la segmentación de imágenes o para encontrar objetos en una imagen. Se crea una imagen del mismo tamaño que la de entrada pero de un solo canal, donde cada pixel corresponde a la probabilidad de ese pixel corresponda al objeto que se busca. La imagen de salida entonces contendrá al objeto en una tonalidad mas brillante comparada con el resto de la imagen [29].

A partir de un modelo del histograma del objeto a localizar en la imagen de prueba, se tiene que para cada pixel en ésta imagen $p(i,j)$ se obtiene la información que permite identificar el rango que le corresponde al pixel $(h_{i,j}, s_{i,j})$, en el modelo de histograma se busca el valor que corresponde en el rango $(h_{i,j}, s_{i,j})$ y este valor se almacena en la imagen de salida de la retroproyección, en la figura 5.2.5 se puede ver el ejemplo para encontrar la mano en una imagen de prueba.

Con respecto al modelo del filtro de partículas descrito en la sección 4.4 (Filtro de partículas), en la etapa de actualización básicamente se compone de dos partes, que es la propagación de la densidad (modelo de movimiento) $p(x_{t+1} \mid x_t)$ y la actualización de los pesos. La primera etapa consiste en actualizar el modelo de movimiento dado por la ecuación (5.9):

$$\mathbf{x}_{t+1}^n = \mathbf{A}\mathbf{x}_t^n + \mathbf{B}w \tag{5.9}$$

donde $\mathbf{A}$ define un el componente determinístico y el producto $\mathbf{B}w$ es el componente de ruido. En las pruebas realizadas en el presente trabajo se utiliza un modelo de ruido del tipo Gaussiano con media cero, es decir , $w \sim N(0, \Sigma)$ y simplemente se considera como $\mathbf{A} = \mathbf{I}$.

Para la actualización de los pesos de las partículas se hace uso de los valores de disparidad, que son considerados en principio como el modelo de densidad de observación. Cada

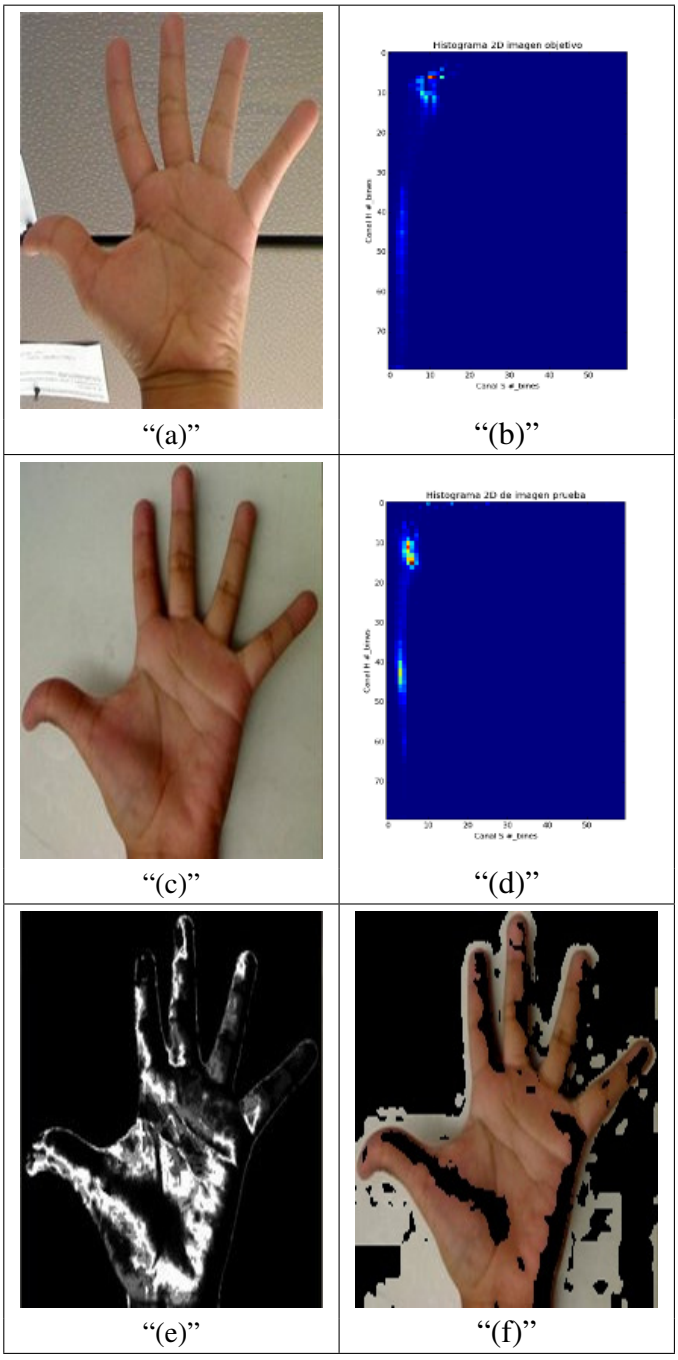


Figura 5.2.5: Retroproyección por histograma [24]. (a) muestra la imagen modelo , (b) es el histograma obtenido con respecto a dos canales de (a). (c) es una imagen prueba, (d) es el histograma de dos canales a partir de (c). (e) y (f) muestran el resultado de lo que es la retroproyección al utilizar los histogramas en (b) y (d)

partícula se actualiza entonces de acuerdo al algoritmo 1

Algorithm 1 Actualización de pesos

for i=1 : N **do**
    particle[i].weight = particle[i].weight*disparity(particle[i].x)
end for
sum_ weights = 0
for j=1 : N **do**
    sum_ weights = sum_ weights+particle[i].weight
end for
$factor = 1 / sum\_weights$
for k=1 : N **do**
    particle[i].weight = particle[i].weight * factor
end for
Cálculo de Neff
sum = 0
for j=1 : N **do**
    $sum = sum + (particle[i].weight)^2$
end for
$neff = \frac{1}{sum}$

Se considera que la cantidad suficiente de partículas con las que debe contar el filtro no debe ser menor al producto $N \times 0.75$, siendo *N* el número de partículas con las cuales se inicializa el filtro, en el caso de que se tenga un valor menor a éste se procede a realizarse el remuestro cuyo algoritmo 2 es:

Algorithm 2 Algoritmo de Remuestreo

$[\{x_j^k, w_j^k\}_{j=1}^N] =$ Remuestreo$[\{x_i^k, w_i^k\}_{j=1}^N]$
Inicializar la *cdf*: $c_1 = 0$
for i=2:N **do**
    cdf : $c_i = c_{i-1} + w_k^i$
end for
Elegir una muestra aleatoria: $u_1 \sim U(0, 1)$
for j=1:N **do**
    while $u_j > c_i$ **do**
        $i = i + 1$
    end while
    Asignar la muestra $x_k^j = x_k^i$
    Asignar el peso $w_k^j = N^{-1}$
end for

En el filtro de partículas se considera que ya existe una gran degradación cuando el número de partículas esta por debajo del valor igual a $N \times 0.04$, con este valor que se determina que el filtro ya no puede continuar con la tarea de seguimiento, y por lo tanto se elimina el filtro, pero el seguidor continua almacenado en el proceso. Sin embargo, dado que se cuenta con el proceso del cálculo de correspondencia es posible recuperar el seguidor y

asociarlo a una región de interés nuevamente, en esto también se considera un tiempo en el cual la persona podría darse el caso de que no pueda ser observado (*oclusión*) y vuelva a reaparecer en la escena analizada. Si dentro de ese lapso de tiempo no se puede realizar una correspondencia del seguidor, entonces este es eliminado del proceso.

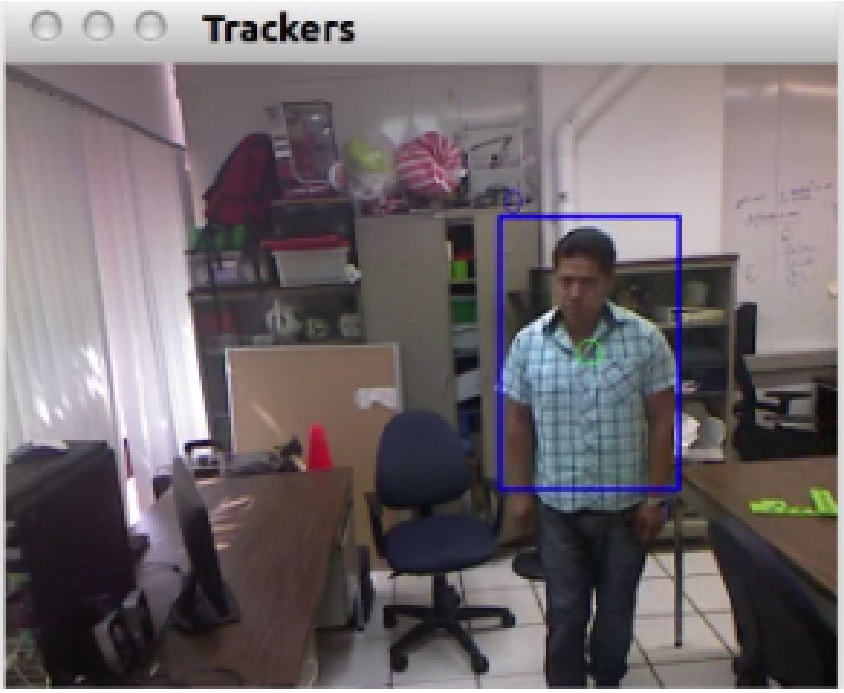


Figura 5.2.6: Resultado del seguidor

En la figura 5.2.6 se puede observar el resultado del seguidor, en el cual se enmarca con un color azul la región que determina la posición de la persona pronosticada por el filtro de partículas.

En este capítulo se han dado los detalles de la manera en que se lleva a cabo el proceso de detección de la persona, basado en el uso de algunas clases del software PCL. En lo que se refiere a la etapa de seguimiento, se explican las acciones que se tienen para el caso del método de filtro de partículas, donde sólo la región de interés es la que proporciona las características para estimar la posición de la persona . Cabe destacar que la implementación del sistema es el uso de la plataforma ROS, que a través de los nodos creados fluye la información para poder generar los resultados y poder evaluar de cierta manera el alcance obtenido.

Capítulo 6

Pruebas y resultados

Dentro de los métodos para poder realizar la etapa de seguimiento, como los descritos en el capítulo 4, en este trabajo se realizaron pruebas con el método del filtro LMS con el propósito de determinar si los resultados de este método es adecuado para la etapa de seguimiento. Un primer punto fue el manejar el filtro LMS (por sus siglas en inglés Least Mean Square) con señales conocidas y observar el comportamiento del filtro, con respecto al orden del filtro y el valor de la constante μ.

Considerando la ecuación 4.9 para realizar la estimación de un valor mediante la combinación de valores pasados ponderados por coeficientes, de manera vectorial puede ser definida como $\hat{u}(n) = \mathbf{u}(n-1) \cdot \mathbf{w}$, donde el orden del fitlro queda determinado por el número de valores pasados del vector $\mathbf{u}(n-1)$. Y también considerando la ecuación 4.25 se obtienen los valores correspondientes del vector **w**, tomando en cuenta el orden del filtro y el valor de la constante μ cuyo rango normalmente se encuentra entre $(0, 1]$.

En la figura 6.0.1 se presenta una señal polinomial, en la cual se realiza una estimación de la señal mediante el filtro LMS de orden tres con la constante $\mu = 0.001$. En la parte superior de la gráfica se muestran la señal polinomial junto con la estimación obtenida mediante el filtro. En la parte inferior se muestra el error obtenido al tomar solamente la diferencia entre cada valor de la señal polinomial y el valor estimado.

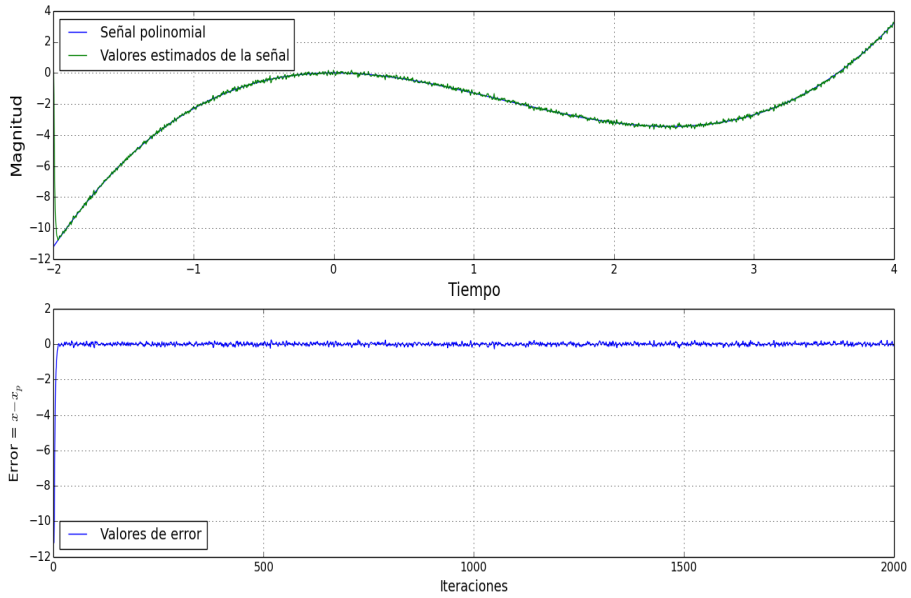


Figura 6.0.1: Filtro LMS de orden 3 $\mu = 0.001$ para señal polinomial

En la figura 6.0.2 se muestra una señal del tipo cosenoidal, y donde también su estimación se realiza mediante el filtro LMS de orden tres, con el valor de $\mu = 0.001$.

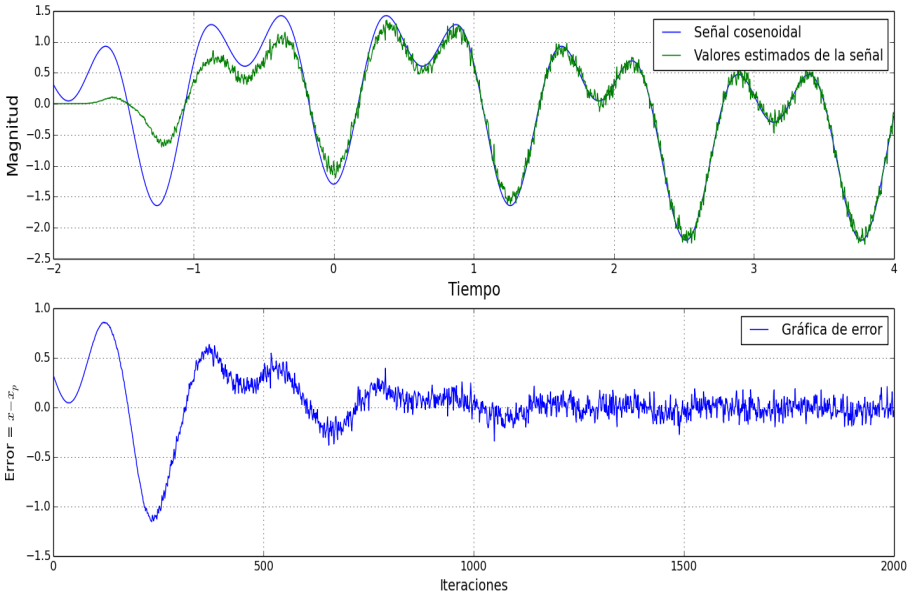


Figura 6.0.2: Filtro LMS orden 3 $\mu = 0.001$ para señal cosenoidal

En la figura 6.0.3 se tiene en la parte superior el error en porcentaje de la estimación para

la señal polinomial y en la parte inferior de la figura se muestra el error en porcentaje para la estimación de la señal senoidal. De estos valores de error se obtiene que en promedio para ambos casos, el porcentaje de error no es mayor al diez por ciento. En este caso el filtro LMS tiene un margen de error pequeño en cuanto a la estimación de valores para este tipo de funciones presentadas. Por lo tanto, considerando en principio que la trayectoria a estimar sería una línea recta, se procedió a realizar algunas pruebas para la etapa de seguimiento.

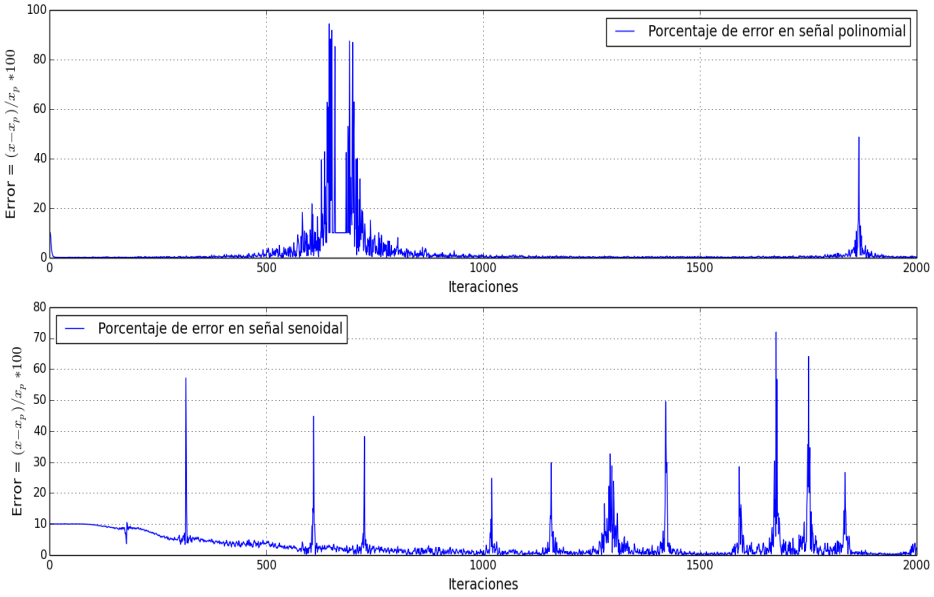


Figura 6.0.3: Porcentajes de error de estimación

El procedimiento de las pruebas consistió entonces en utilizar hasta el tercer nodo en el sistema ROS, por lo tanto se realizó la detección de lo que es el piso, para poder obtener la nube de puntos y procesarla hasta obtener el cluster correspondiente a una persona. Al tener la persona detectada por el sistema, se eligió un punto (*centroide*) que fue utilizado para aplicar el filtro LMS. Además considerando que en la etapa de seguimiento se utilizaría un punto 2D, se utilizó de manera independiente un filtro LMS para la estimación de cada uno los valores de las coordenadas X e Y.

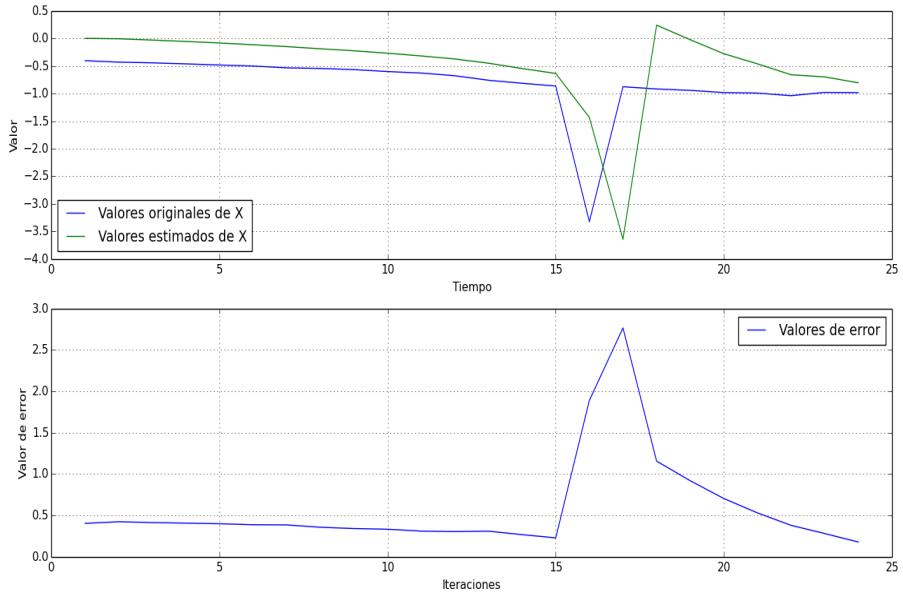


Figura 6.0.4: Filtro LMS de orden cuatro con $\mu = 0.08$ para la coordenada X

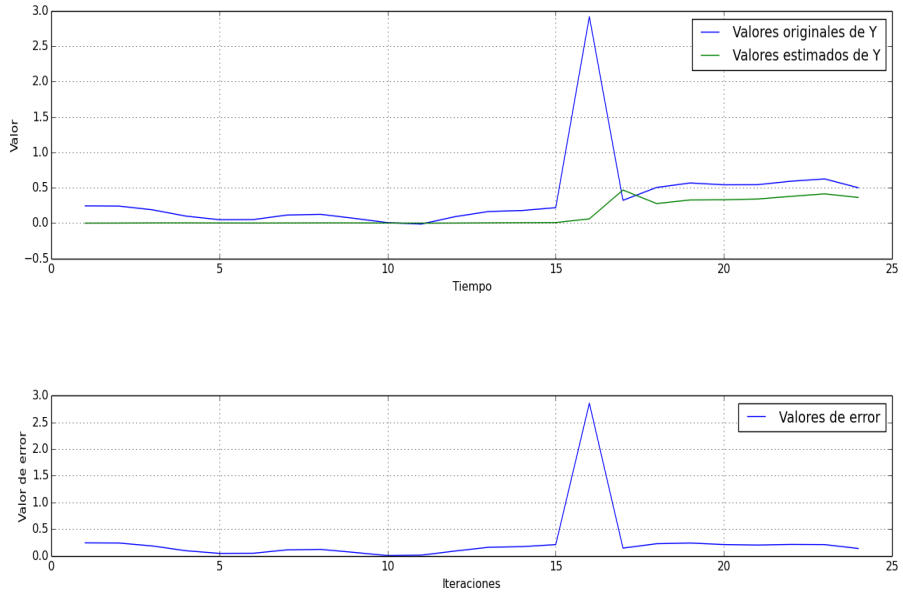


Figura 6.0.5: Filtro LMS de orden cuatro con $\mu = 0.004$ para la coordenada Y

En la figura 6.0.4 se muestra el resultado para la estimación de los valores de la coordenada X mediante un filtro LMS de orden cuatro con un valor de $\mu = 0.08$, en la parte superior están graficadas los valores obtenidos mediante el dispositivo Kinect y los valores

estimados. En la parte inferior solamente se muestra el error con base en la diferencia entre el valor observado y el valor estimado. En la figura 6.0.5 se muestra la estimación de los valores en Y mediante el filtro LMS de orden cuantro y con un valor de $\mu = 0.004$, en la parte superior están graficadas estos valores y en la parte inferior solamente la diferencia entre el valor observado y el valor estimado.

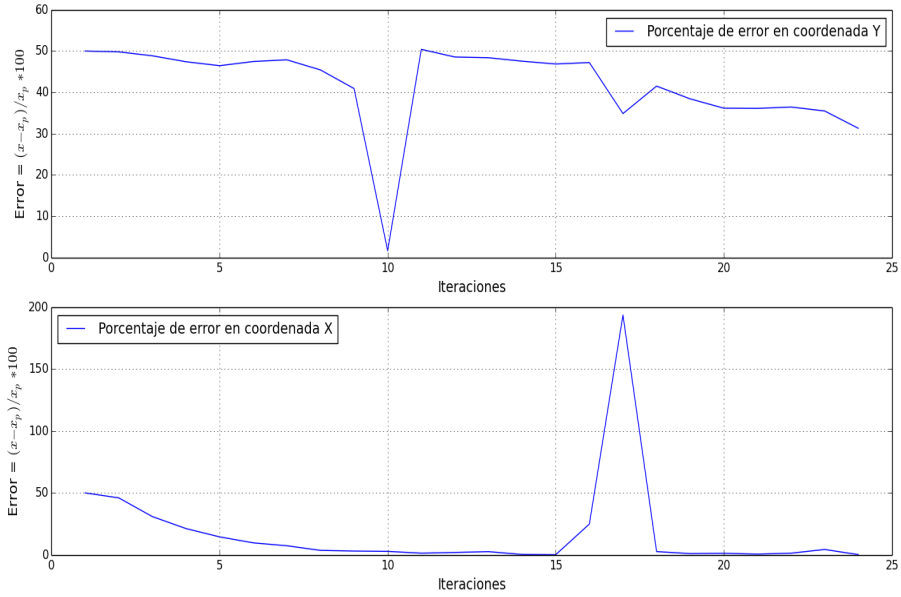


Figura 6.0.6: Porcentaje de error para el filtro LMS de orden cuatro

En la figura 6.0.6 se tiene la gráfica del porcentaje de error, en la parte superior corresponde a los valores de la coordenada Y y en la parte inferior los valores de la coordenada X. De estas gráficas se tiene que el porcentaje de error es mucho mayor al diez por ciento. Por lo tanto considerando que el orden del filtro es de orden tres, y teniendo en cuenta que el filtro LMS puede ser modificado en cuanto al número de coeficientes utilizados y también el valor de la constante μ es una variable, se realizaron cambios en estas variables del filtro LMS.

De las posibilidades del filtro LMS, en la figura 6.0.7 solamente se tiene el cálculo para la estimación de los valores de la coordenada X, mediante un filtro LMS de orden diez con un valor de $\mu = 0.01$ y la figura 6.0.8 se tiene la estimación de los valores de la coordenada Y con un filtro de orden diez con la constante $\mu = 0.01$.

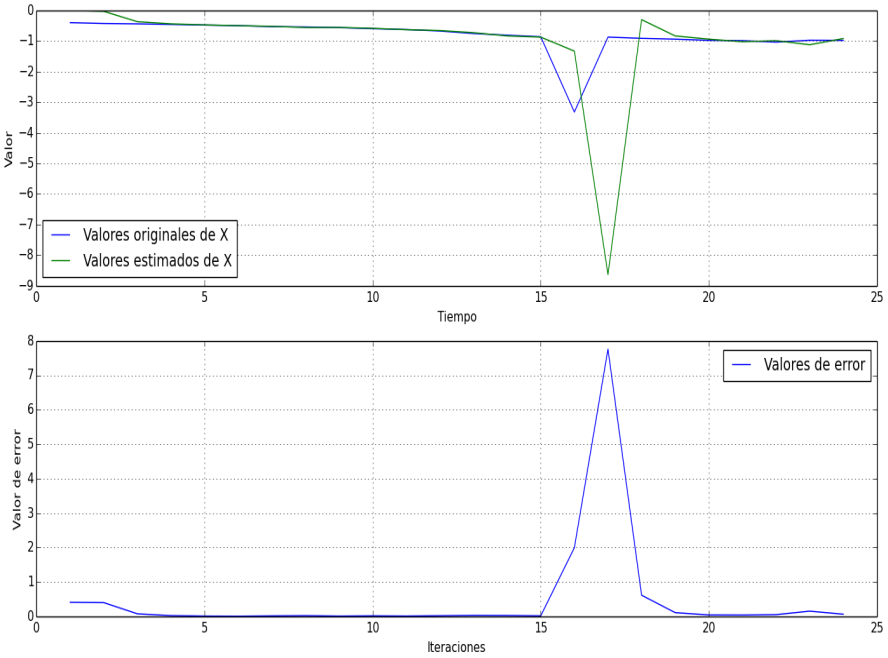


Figura 6.0.7: Filtro LMS de orden diez con $\mu = 0.01$ para la coordenada X

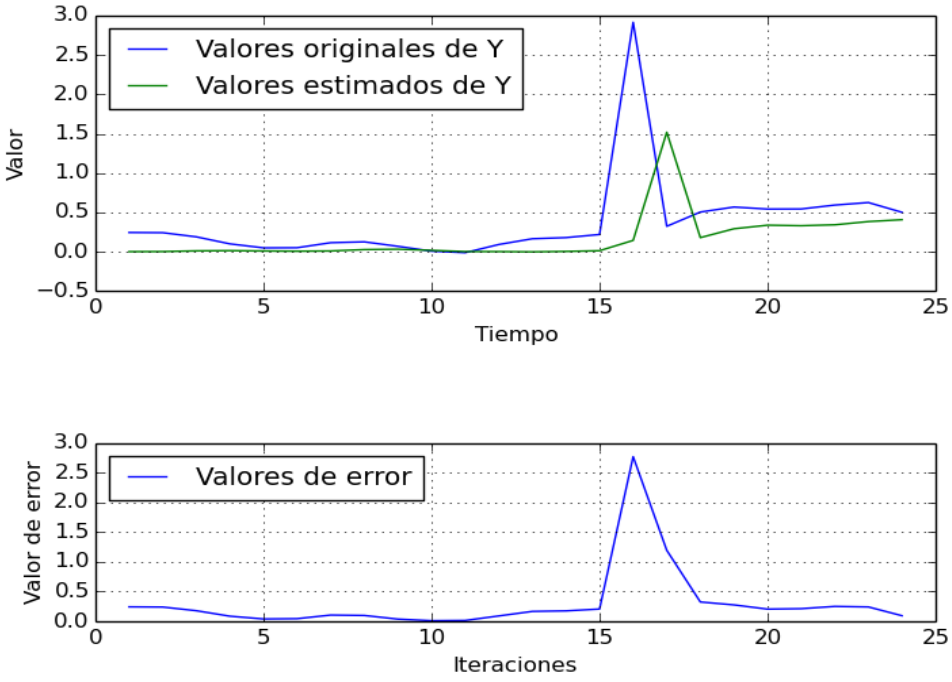


Figura 6.0.8: Filtro LMS de orden diez con $\mu = 0.01$ para la coordenada Y

En la figura 6.0.9 se tienen los porcentajes de error de la estimación, en la parte superior los correspondientes para la coordenada Y y en la parte inferior los correspondientes para la coordenada X.

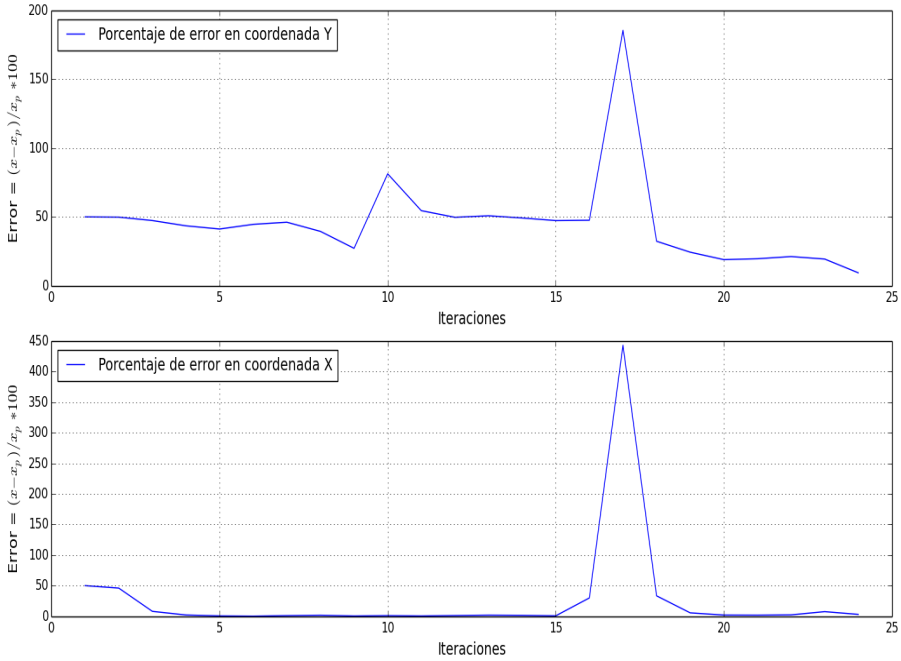


Figura 6.0.9: Porcentaje de errores para el caso del filtro LMS de orden diez

De la figura 6.0.9 se puede observar que el porcentaje de error que corresponde al valor de la coordenada Y, el filtro en los valores estimados distan mucho en la gran mayoría con respecto a los valores observados por el dispositivo Kinect y es muy notorio esta particularidad en comparación con los valores estimados de la coordenada X. Resultados similares se obtuvieron en varias pruebas realizadas para la trayectoria en línea. En la figuras 6.0.10 y 6.0.11 se grafican las estimaciones de una trayectoria donde se utiliza el filtro LMS de orden diez, con el valor de la constante $\mu = 0.001$. De manera similar se tiene que la estimación en la coordenada Y el error solo en la diferencia es aún mayor en la mayoría de los valores estimados. Sin embargo al variar el valor de μ se nota una mejor estimación, para el filtro de orden diez, en la figura 6.0.12, se muestra una gráfica donde la estimación se realiza con un filtro LMS de orden diez y la constante $\mu = 0.09$.

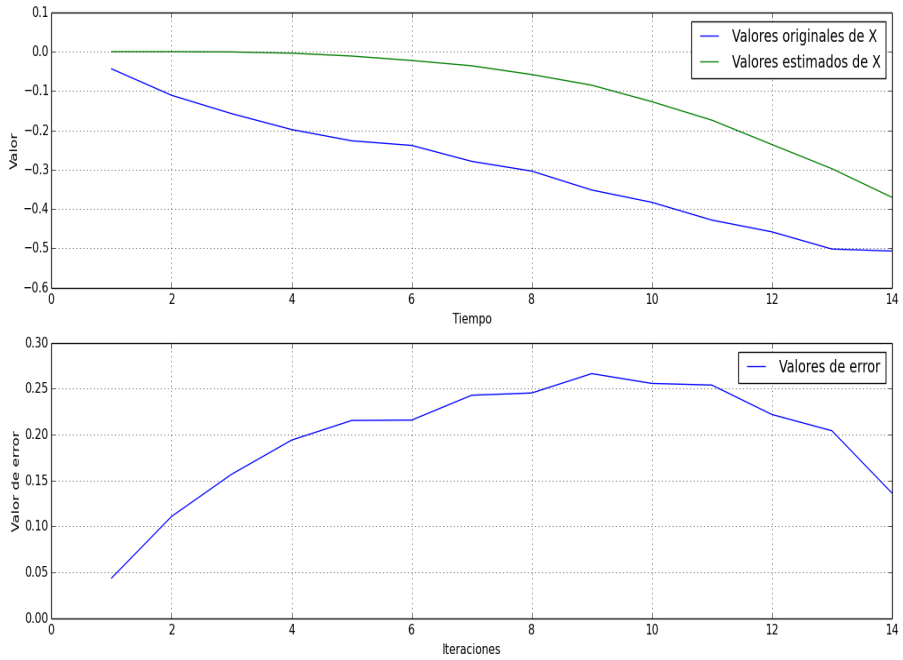


Figura 6.0.10: Filtro LMS de orden diez con $\mu = 0.01$ para la coordenada X

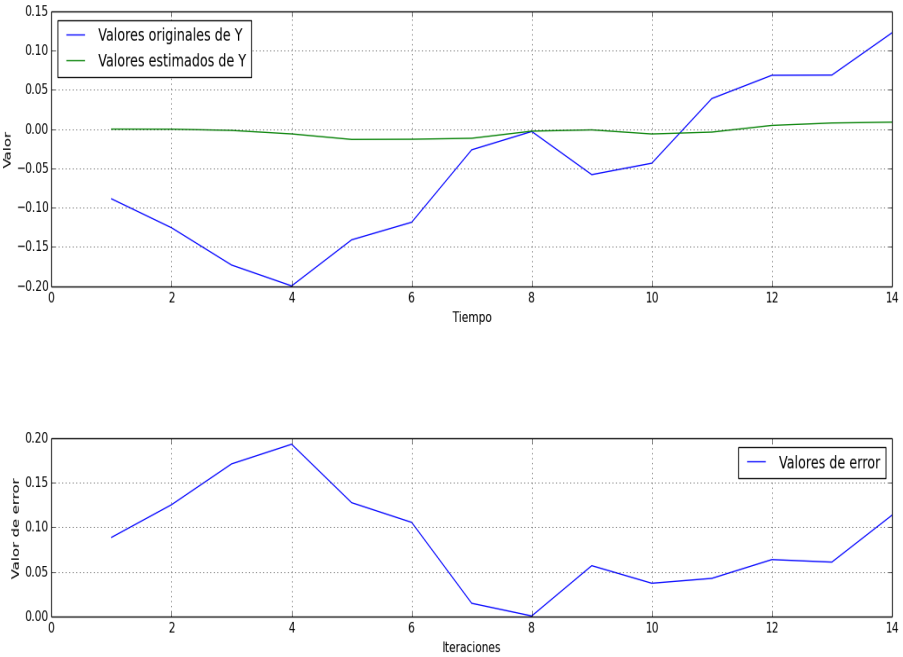


Figura 6.0.11: Filtro LMS de orden diez con $\mu = 0.01$ para la coordenada Y

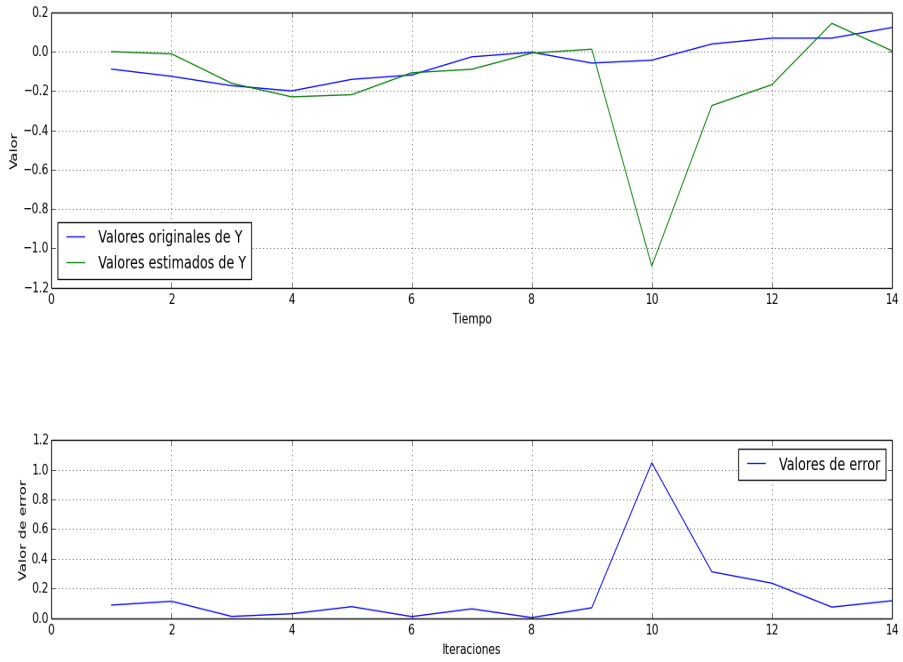


Figura 6.0.12: Filtro LMS de orden diez con $\mu = 0.09$ para la coordenada Y

De tal manera que se decidió utilizar un filtro LMS de orden mayor a cinco con un valor de la constante μ no mayor a una centésima, ya que al incrementar esta constante el filtro llega a variar de manera que los valores estimados tiene un porcentaje de error mucho mayor al cien por ciento. Esta característica se comienza a observar en la figura 6.0.12.

Por otra parte se procedió a realizar las pruebas con el uso del filtro de partículas, las cuales tienen un mayor enfoque en el presente trabajo.

Las pruebas se realizaron dentro del laboratorio de BioRobótica, ya que es un lugar que representa una escena con posibilidades de oclusión, así como también variaciones de iluminación. En estas pruebas el dispositivo kinect permaneció estático, sobre una base a una altura de 1.70 m, en lo cuanto a la distancia recorrida en algunas trayectorias se trató de que la distancia mínima fuera de un metro y la máxima de seis metros.

Las trayectorias presentadas en este trabajo fueron realizados en una escena en la que solamente aparece una persona. Las trayectorias que se llevaron a cabo principalmente fueron los recorridos en línea recta y trayectoría del tipo circular dentro del espacio permitido en la escena.

En la figura 6.0.13 se muestra una de las pruebas de la trayectoria en linea recta hasta una distancia no mayor de cinco metros, en cada una se muestra un cuadro en el cual está la detección de la persona identificada mediante un recuadro blanco, del lado izquierdo de

la figura 6.0.13 que corresponden entonces a los incisos (a), (c) y (e). El lado derecho de la figura 6.0.13 en los incisos (b), (d) y (f) contienen el mismo cuadro donde el seguidor determina la ubicación de la persona indicada por el recuadro azul. Por lo tanto las tres escenas muestran el inicio de la trayectoria, figura 6.0.13 (a) y (b), una posición a la mitad de la trayectoria, alejándose del dispositivo Kinect, figura 6.0.13 (c) y (d). Y por último se tiene un cuadro que muestra una posición del recorrido acercándose al dispositivo Kinect, 6.0.13 (e) y (f).

Dentro de los resultados obtenidos del seguidor, se realiza el cálculo de la distancia entre el centroide del detector y el seguidor,a partir de este dato se calcula un índice tomando como base el tamaño de la diagonal de la región de interés en la imagen del sistema de detección [27]. Para realizar el cálculo de la distancia entre centroides, se usa la ecuación:

$$d = \sqrt{(x_o - x_e)^2 + (y_o - y_e)^2} \tag{6.1}$$

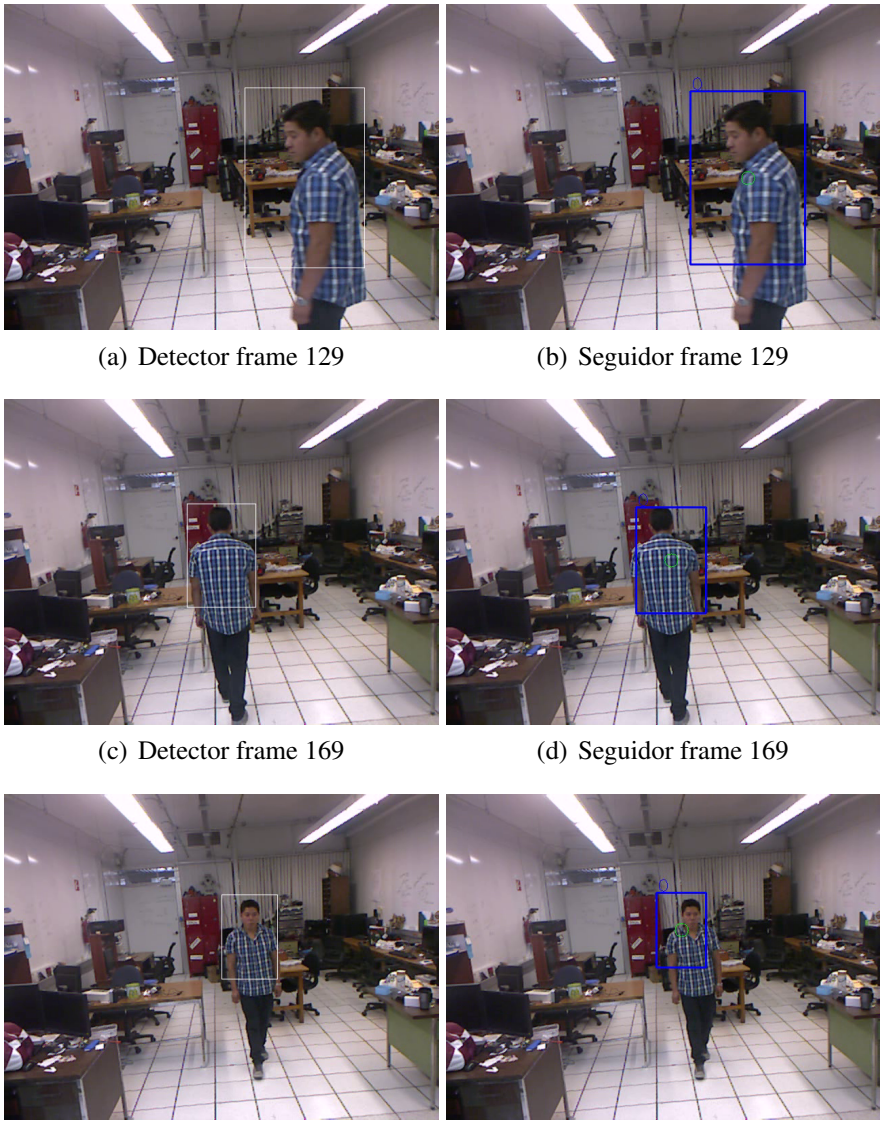
donde el centroide correspondiente a la observación es (x_o, y_o) y el centroide estimado es (x_e, y_e). De manera similar la distancia diagonal se calcula tomando un punto superior y uno inferior de la región de interés observada y utilizando nuevamente la ecuación 6.1. El índice con respecto a la distancia entre centroides es $I_e = d/D$, donde d es la distancia entre centroides y D es la longitud de la diagonal de la región de interés observada.

A su vez otro dato que es obtenido es el índice de traslape entre las regiones de interés. Considerando el área de la región de interés proporcionada por la altura y el ancho tanto en el caso de la observación como en el de la estimación, el índice es entonces $I_t = A_e/A_o$ donde el denominador es el área de la observación. Se considera que un índice cuyo valor es igual a o mayor a 0.5 el seguidor obtiene un resultado correcto en la predicción de la posición de la persona [3].

Para realizar el cálculo de estos índices se hace el análisis de ciertos cuadros de una secuencia de vídeo de alrededor de 500 cuadros. A partir de que se tenía la detección de la persona se marcaban manualmente cada diez cuadros las coordenadas de las regiones de interés en las imágenes 2D.

En la figura 6.0.14 se tienen los resultados obtenidos para el índice de traslape entre las regiones de interés I_t y en la figura 6.0.15 se tiene la gŕafica de los resultados para el índice de error entre la distancia entre centroides I_d, ambos resultados corresponden una trayectoria en línea recta.

En la figura 6.0.16 se muestra algunas de las imágenes de la trayectoria similar a un círculo, la cual fue llevada a una distancia no mayor a tres metros. El lado izquierdo de la figura que corresponden a los incisos (a), (c), (e) y (g) contienen la región de interés observada y se encuentra marcada en color blanco. El lado derecho de la figura correspondientes a los incisos (b), (d), (f) y (h) muestran los resultados de la estimación del seguidor, donde la región de interés se encuentra marcada con un color azul. De la misma forma son cuatro posiciones de todo la trayectoria realizada y que era en forma de círculo.



(a) Detector frame 129 (b) Seguidor frame 129

(c) Detector frame 169 (d) Seguidor frame 169

(e) Detector frame 331 (f) Seguidor frame 331

Figura 6.0.13: Trayectoría: línea recta

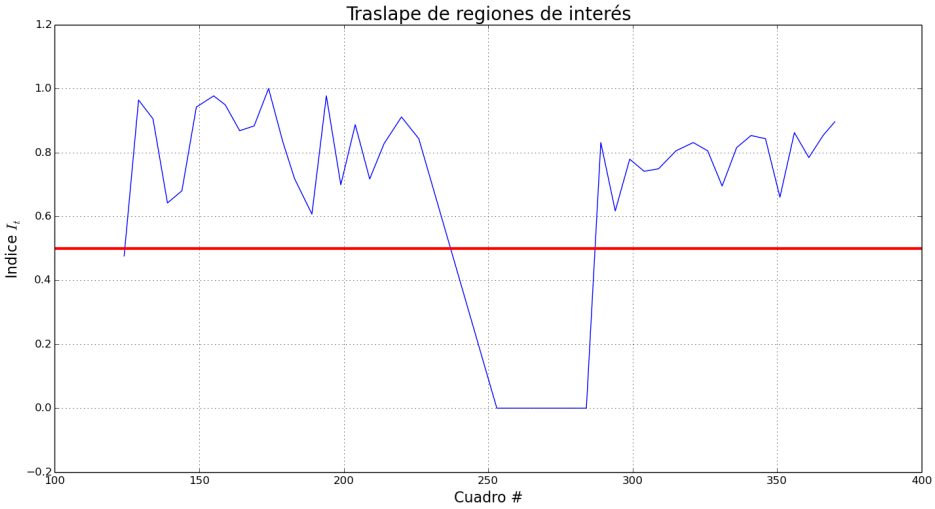


Figura 6.0.14: Índice de error de traslape

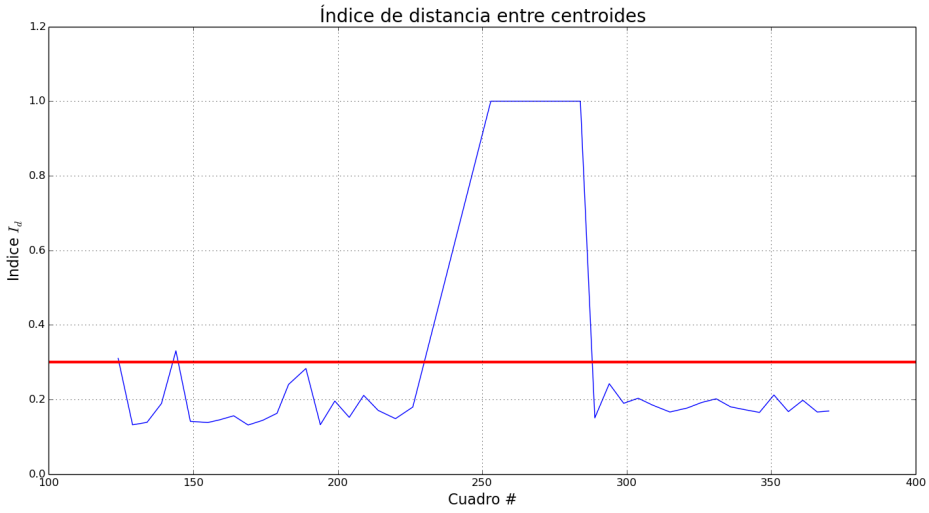
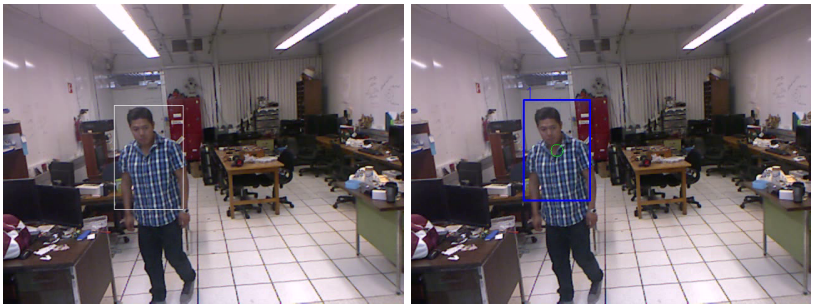


Figura 6.0.15: Índice error de distancia entre centroides

(a) Detector frame 148 (b) Seguidor frame 148

(c) Detector frame 221 (d) Seguidor frame 221



(e) Detector frame 543 (f) Seguidor frame 543

(g) Detector frame 595 (h) Seguidor frame 595

Figura 6.0.16: Trayectoría: aleatoria

En la figura 6.0.17 se tienen los resultados obtenidos para el índice de traslape entre las regiones de interés I_t y en la figura 6.0.18 se tiene la gŕafica de los resultados para el índice de error entre la distancia entre centroides I_d, ambos resultados corresponden una trayectoria en línea recta.

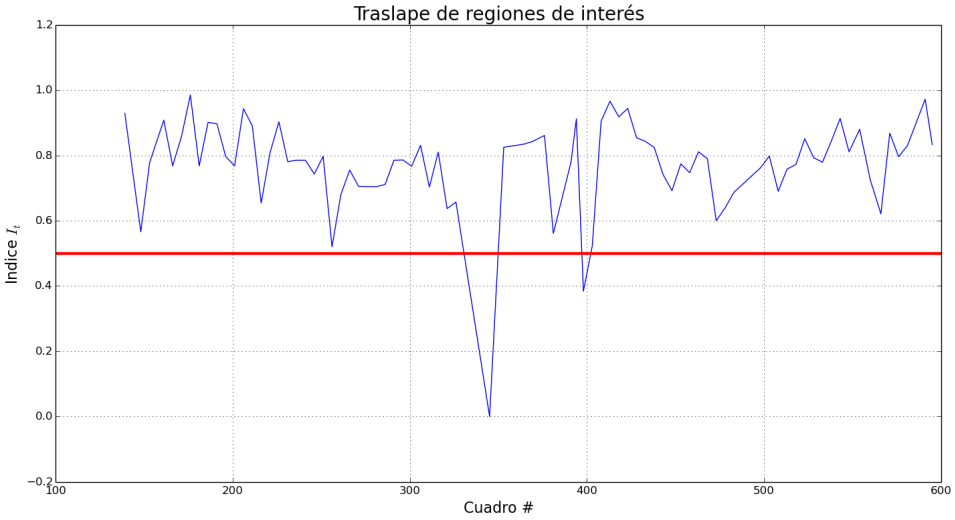
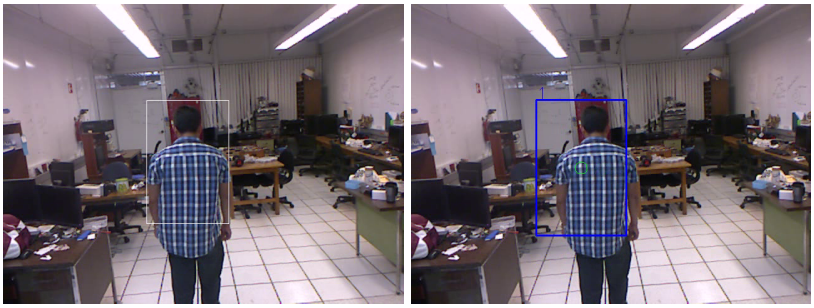


Figura 6.0.17: Índice de error de traslape

Figura 6.0.18: Índice de error de distancia entre centroides

La figura 6.0.19 cuenta algunas imágenes donde se trata de realizar una trayectoria aleatoria pero que contenga en primer lugar una trayectoria en línea recta y por otra parte una trayectoría circula la cual se realiza rodeando gran parte de un escritorio que se encontraba en la escena. En los incisos (a) y (c) corresponden a la trayectoría en línea recta y los incisos (e) y (g) son parte del trayecto de rodear el escritorio, todos éstos incisos contienen la región de interés observada y se encuentran marcados en color blanco. Lo correspondiente a la estimación del seguidor marcados en un recuadro de color azul corresponden a los incisos (b), (d), (f) y (h). Básicamente en esta parte de rodear el escritorio se alcanza una distancia mayor a cinco metros desde la posición de la cámara, con el fin de observar el comportamiento del sistema de seguimiento implementado.



(a) Detector frame 149 (b) Seguidor frame 149

(c) Detector frame 196 (d) Seguidor frame 196

(e) Detector frame 307 (f) Seguidor frame 307

(g) Detector frame 369 (h) Seguidor frame 369

Figura 6.0.19: Trayectoría: rodear escritorio

En la figura 6.0.20 se tienen los resultados obtenidos para el índice de traslape entre las regiones de interés I_t y en la figura 6.0.21 se tiene la gŕafica de los resultados para el índice de error entre la distancia entre centroides I_d, ambos resultados corresponden una trayectoria en línea recta.

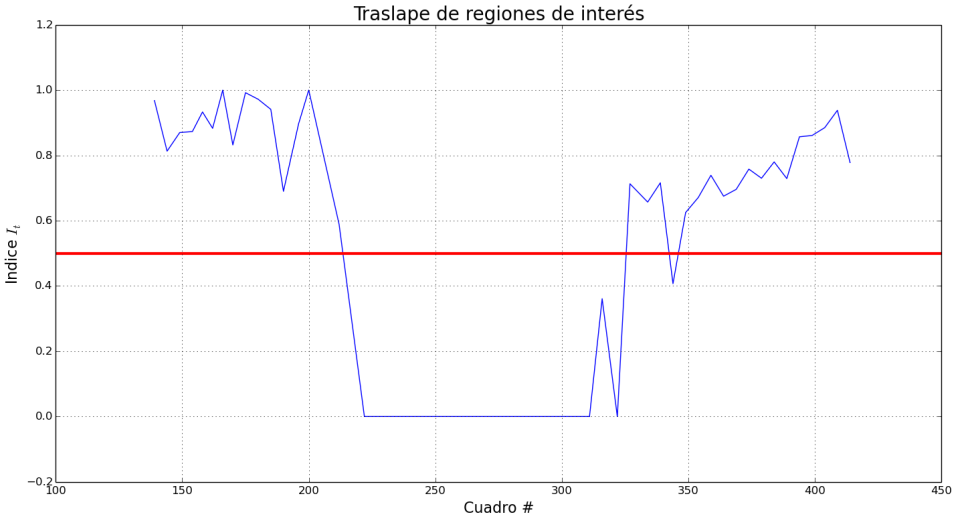


Figura 6.0.20: Índice de error de traslape

Figura 6.0.21: Índice de error de distancia entre centroides

De las diversas pruebas que se realizaron con los tres tipos de trayectorias descritas en las figuras 6.0.13, figura 6.0.16 y figura 6.0.19, en la tabla 6.0.1 los promedios de los índices tanto para la distancia entre centroides como para el traslape de las regiones de interés, cabe mencionar que en esta tabla se consideran para el caso del índice de traslape los valores que resultaron ser mayores a cero, así como aquellos índices cuyo valor fue mayor a 0.5 y para el caso del índice de distancia entre centroides se tomaron los valores que fueron menores a uno. En la tabla 6.0.1 se define L_R para la trayectoría en línea recta, C para la trayectora circular y R_E para el caso de rodear el escritorio. En la tabla 6.0.2 se tiene el resultado de los valores de la distancia entre los centroides y los valores de las

diagonales para cada una de las regiones de interés, obtenidos para las mismas trayectorias.

Parámetro	Promedio índice de distancia entre centroides	Promedio índice de traslape	Promedio índice de traslape mayor a 0.5
Trayectoria L_R	0.2079	0.6643	0.6308
Trayectoria L_R	0.1835	0.7116	0.6157
Trayectoria L_R	0.0950	0.6724	0.5829
Trayectoria L_R	0.1847	0.7083	0.5952
Trayectoria C	0.1918	0.6941	0.6427
Trayectoria C	0.1800	0.7174	0.6932
Trayectoria R_E	0.1741	0.7327	0.6561
Trayectoria R_E	0.1625	0.7290	0.6319

Tabla 6.0.1: Tabla de indices I_t e I_d.

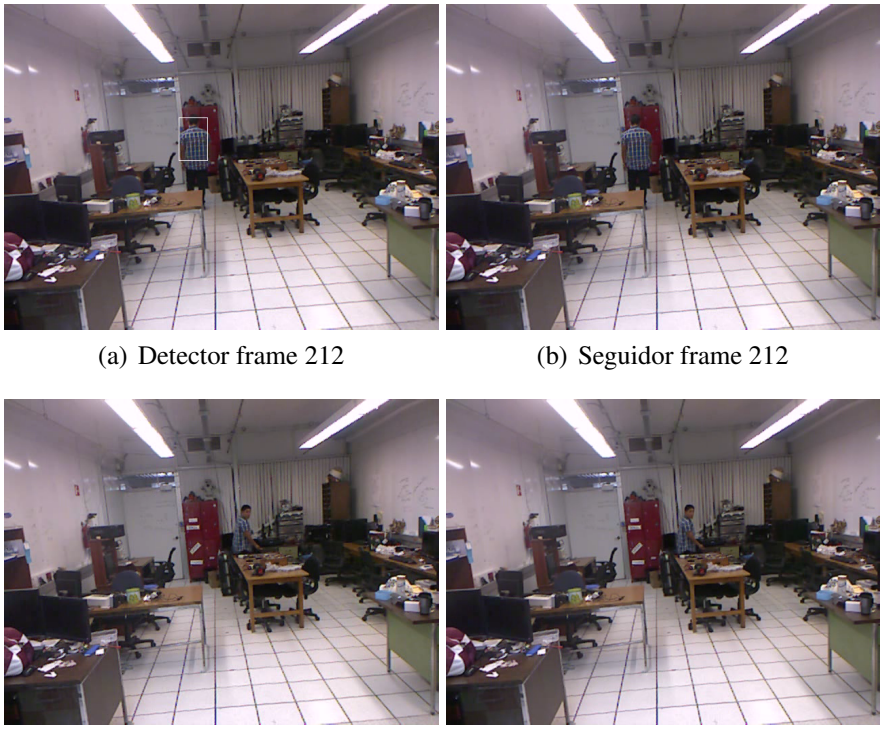
Parámetro	Distancia entre centroides	Diagonal de la ROI observada	Diagonal de la ROI estimada
Trayectoria L_R	45	212	190
Trayectoria L_R	28	157	163
Trayectoria L_R	32	162	162
Trayectoria L_R	13	162	168
Trayectoria C	35	207	200
Trayectoria C	33	193	185
Trayectoria R_E	32	177	177
Trayectoria R_E	27	165	179

Tabla 6.0.2: Tabla de diagonales y distancia entre centroides.

Así mismo de los resultados graficados en las figuras 6.0.20, 6.0.14 y 6.0.17 se puede observar que existen valores en los cuales el seguidor no obtuvo un resultado, es decir, se pierde el objeto a seguir y es instantes después que el seguidor logra reestablecer la predicción de la persona. Por lo tanto el valor del índice de traslape I_t en estos casos tiene un valor de cero, en el caso del indice de distancia entre centroides I_d el valor correspondiente es de uno.

Dos de los casos en los que era más frecuente, se presentan cuando la persona se encuentra más allá de los 5 metros de distancia y el segundo factor es debido aque el detector no realiza una correcta ubicación de la persona cuando ésta se haya de perfil a la cámara. En la figura 6.0.22 se muestra a la persona que esta a una distancia cercana a los cinco metros desde el dispositivo Kinect, donde la figura 6.0.22 (a) se tiene marcada a la persona detectada, pero en la figura 6.0.22 (b) no hay una estimación de la posición de la persona, este cuadro es parte de la trayectoria correspondiente a rodear el escritorio. En la figura 6.0.22 (c) se muestra un cuadro la persona está en una posición de perfil con respecto al dispositivo Kinect, donde no es posible realizar la detección y por lo tanto tampoco se

tiene en la figura 6.0.22 (d) no hay tampoco una estimación del seguidor.



(a) Detector frame 212 (b) Seguidor frame 212

(c) Detector frame 274 (d) Seguidor frame 274

Figura 6.0.22: Falla seguidor

Capítulo 7

Conclusiones

En el presente trabajo se ha implementado un sistema que combina el proceso de detección y seguimiento de una persona que se presente en alguna escena. Una vez que se llevaron a cabo las pruebas realizadas se ha visto la problemática que se tiene en el seguimiento de la persona, es decir, el proceso de detección es manejado con base en una clasificación de los clusters que fueron seleccionados como posibles candidatos a ser una persona y a pesar de que el descriptor HOG es uno de los descriptores mas utilizado para el proceso de detección, se ha visto que esto llega a tener fallas en ciertas condiciones.

En la etapa de detección también queda limitada al entrenamiento realizado para la SVM y otro apecto que limita la detección es el tener como referencia en todo momento la superficie que representa el piso. En la cuestión de hardware se ha utilizado una dispositivo que emite a partir de una cámara IR, se ve limitado a trabajar en escenas donde la iluminación tiene que ser del tipo artificial.

Otro punto a considerar a partir de los resultados obtenidos en relación con el sistema de detección, es que el dispositivo siempre se mantuvo estático, por lo tanto, el sistema de detección se ve afectado cuando la distancia entre el dispositivo Kinect y la persona es más de cinco metros, otro detalle en la etapa de detección que se pudo obervar de manera frecuente, consistió en que la persona al ser observada de perfil por el dispositivo Kinect generaba la ausencia de un area de interés para ser proporcionada al nodo de seguimiento. Esto puede se puede verificar de las figuras 6.0.20 y 6.0.14 donde hay un número de cuadros donde se tiene el valor del indice I_t igual a uno.

Con respecto a la etapa de seguimiento utilizando el filtro de partículas, se trato de utilizar un algoritmo para el proceso de remuestro, que permitiera tener la menor oscilación posible en el cálculo del centroide estimado, es decir, se puede observar que el número de particulas llega a ser menor al número suficiente para obtener la mejor estimación posible. Sin embargo, al obtener nuevamente la observación del centroide de la persona, el filtro no se desvía tan bruscamente, las figuras 6.0.21 y 6.0.15 muestran este punto en que el seguidor tiene un índice I_d que nunca es mayor a 0.5, claro que esto es sólo considerando todos los cuadros donde existe una estimación.

Considerando la tabla 6.0.1, en la cual sólo se consideran los resultados donde en la etapa de seguimiento se tiene una estimación, el índice I_d no es mayor a 0.5 y por otra parte el índice I_t es mayor a 0.5, cumpliendo por lo tanto una correcta estimación de la posición de la persona a seguir.

7.1 Trabajo a futuro

Dentro de las posibles mejoras que podrían realizarse al sistema se tienen con respecto a la detección:

- Aumentar la capacidad de la clasificación de la SVM, es decir, se podría realizar un entrenamiento que incluya la detección de la persona en un vista de perfil, que en este trabajo fue el que más relevancia tuvo.
- Poder combinar otro sistema de detección que no tenga en consideración la superficie de referencia, entonces tal vez se contaría con algún sistema que se tratara de identificar a la persona observando solamente la parte superior de su cuerpo (desde la cintura hasta la cabeza).

En relación a la fase de seguimiento y dado que el aspecto del filtro de partículas es la gran cantidad de información que se maneja se podría, por lo tanto obtener una mejora en el proceso de cálculos de los datos con una mejor programación del tipo paralelo.

Una de las pruebas que no se realizaron fue el de poder colocar el dispositivo kinect de tal forma que éste no permaneciera estático. Se tiene entonces la posibilidad de que el trabajo realizado sea una herramienta en un robot móvil. El laboratorio de BioRóbótica de la Facultad de Ingeniería cuenta con el robot Justina, por lo tanto se podrán realizar las pruebas y así poder generar una visión mas amplia de la efectividad del sistema de detección y seguimiento utilizado en el presente trabajo.

Bibliografía

[1] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation," *Robotics Automation Magazine, IEEE*, pp. 80 –91, 2012. 22

[2] M. S. Arulampalam, S. Maskell, and N. Gordon, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, vol. 50, pp. 174–188, 2002. 11

[3] F. Bashir and F. Porikli, "Performance Evaluation of Object Detection and Tracking Systems," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS2006)*, 2006. 67

[4] N. Bellotto and H. Hu, "Multisensor-based human detection and tracking for mobile service robots." *IEEE Trans. Systems, Man, and Cybernetics, Part B*, vol. 39, pp. 167–181, 2009. v, 8, 9

[5] M. R. Blackburn and H. G. Nguyen, "Autonomous visual control of a mobile robot," in *Proceedings of the 1994 Image Understanding Workshop*, 1994, pp. 1143–1150. 8

[6] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "Experiences with an interactive museum tour-guide robot," *Artif. Intell.*, vol. 114, 1999. 12

[7] P. Chakravarty and R. Jarvis, *Panoramic vision and laser range finder fusion for multiple person tracking*. IEEE, Institute of Electrical and Electronics Engineers, 2006, pp. 2949 – 2954. 15

[8] O. Chapelle, P. Haffner, and V. N. Vapnik, "Support vector machines for histogram-based image classification," *IEEE Transactions on Neural Networks*, vol. 10, pp. 1055–1064, 1999. 29

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, 2005, pp. 886–893. v, 27, 29

[10] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: fifteen years later," 2011. 42

[11] D. Elliott, *Handbook of Digital Signal Processing: Engineering Applications*, ser. Electronics & Electrical. Academic Press, 1987. 31

[12] R. Faragher, "Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes]." *IEEE Signal Process. Mag.*, vol. 29, pp. 128–132, 2012. 34

[13] S. Feyrer and A. Zell, "Detection, tracking, and pursuit of humans with an autonomous mobile robot," *in Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'99*, pp. 864–869, 1999. 8

[14] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, 1981. 47

[15] T. W. Fong, I. Nourbakhsh , and K. Dautenhahn, "A survey of socially interactive robots," *Robotics and Autonomous Systems*, vol. 42, pp. 143–166, 2003. 2, 6

[16] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., 2006. v, 16

[17] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced Computer Vision With Microsoft Kinect Sensor: A Review," *IEEE Transactions on Cybernetics*, vol. 43, pp. 1318–1334, 2013. 19

[18] I. Horswill, "Polly: A vision-based artificial agent," *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 824–829, 1993. v, 7

[19] F. James, "Monte carlo theory and practice," *Reports on Progress in Physics*, vol. 43, 1980. 12

[20] M. Kachouane, S. Sahki, M. Lakrouf, and N. Ouadah, "Hog based fast human detection," *CoRR*, vol. abs/1501.02058, 2015. 27

[21] K. Khoshelham and E. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," in *Sensors 2012, 12, 1437–1454. 2013*, 2012, p. 8238. v, 51

[22] R. Labayrade and D. Aubert, "Real time obstacle detection in stereovision on non flat road geometry through v-disparity representation," Proc. SPIE 238. 71 136 version 1 - 9 Sep 2011 LeGuilloux, Tech. Rep., 2002. 51

[23] Microsoft. (2013) Kinect for windows sensor components and specifications. [Online]. Available: https://msdn.microsoft.com/en-us/library/jj131033.aspx v, 19

[24] OpenCV. (2012) Opencv tutorials. [Online]. Available: http://docs.opencv.org/ v, v, 21, 54

[25] Y. Pang, Y. Yuan, X. Li, and J. Pan, "Efficient hog human detection," *Signal Process.*, vol. 91, pp. 773–781, 2011. v, 28, 29

[26] PCL. (2014) Pcl documentation. [Online]. Available: http://pointclouds.org/documentation/ v, 22, 46

[27] B. Purnama, B. Erfianto, and Y. Hafidz, "On the experiment of multi camera tracking using kalman filter and fov lines," in *Information and Communication Technology (ICoICT), 2014 2nd International Conference on*, 2014, pp. 297–301. 67

[28] ROS. (2013) Ros indigo. [Online]. Available: http://wiki.ros.org/indigo v, 24, 46

[29] M. J. Swain and D. H. Ballard, "Color indexing," *Int. J. Comput. Vision*, vol. 7, 1991. 39, 53

[30] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. v, v, 35, 37, 40

[31] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 780–785, 1997. 6

[32] W. Zajdel, Z. Zivkovic, and B. J. A. Kröse, "Keeping track of humans: Have I seen this person before?" in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005, April 18-22, 2005, Barcelona, Spain*, 2005, pp. 2081–2086. 15

[33] T. Zhang, S. Fei, X. Li, and H. Lu, "An improved particle filter for tracking color object," in *Intelligent Computation Technology and Automation (ICICTA), 2008 International Conference on*, vol. 2, 2008, pp. 109–113. 18

[34] Q. Zhu, Q. Zhu, S. Avidan, S. Avidan, M. chen Yeh, M. chen Yeh, K. ting Cheng, and K. ting Cheng, "Fast human detection using a cascade of histograms of oriented gradients," in *In CVPR06*, 2006, pp. 1491–1498. 27