UNIVERSIDAD POLITÉCNICA DE VALENCIA

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

TRABAJO DE FIN DE MÁSTER

INTELIGENCIA ARTIFICIAL, RECONOCIMIENTO DE FORMAS E IMAGEN DIGITAL

“Detección y seguimiento de una persona en una habitación”

LORENA OLGUIN RIVAS

Supervisado por:

M. CARMEN JUAN LIZANDRA

Septiembre 2013

Todos nuestros sueños pueden convertirse en realidad

si tenemos el coraje de perseguirlos.

(Walt Disney)

RESUMEN

Existen diferentes métodos de detección de objetos utilizando diferentes dispositivos como cámaras web, Kinect, cámaras estereoscópicas, etc. La mayoría de estos estudios están enfocados a la detección de objetos con fines de seguridad en la sociedad, ya sea detectar robos, controlar el tráfico en las carreteras, contar personas en una escena, etc. Pero no existen sistemas para detectar alteraciones en el movimiento de las personas, o juegos para niños, pero con fines educativos. El presente trabajo final de Máster está motivado por la necesidad de seguimiento de una persona en una habitación para el proyecto CHILDMNEMOS.

A lo largo de esta tesina se han desarrollado cuatro algoritmos con el fin de conseguir detectar a una persona en movimiento con dispositivos ubicados en el techo de una escena, de tal forma que detecte a la persona verticalmente y no de frente. Para ello se realizó un análisis de posibles algoritmos que podrían responder a nuestra necesidad utilizando dos dispositivos diferentes de captura de fotogramas de video; utilizamos una cámara Logitech 9000 y Kinect. Además de usar librerías para visión artificial y para el desarrollo de aplicaciones para Kinect. Estas librerías son OpenCV y OpenNI, respectivamente.

Como pasos en los algoritmos desarrollados, se encuentran técnicas conocidas, como Sustracción de fondo, *Mixture of Gauss*, Filtro de Kalman; y también funciones de la librería OpenNI, que funcionan con Kinect. Se compararán los algoritmos y los dispositivos, para conocer qué algoritmo y qué dispositivo son los que ofrecen mejores resultados.

Al realizar los experimentos, se obtuvieron mejores resultados con un algoritmo desarrollado para Kinect, usando la técnica que se planteó, que consiste en analizar solo una región de la escena capturada por Kinect (Cabeza y Hombros de la persona) el cual permitió sustraer el fondo; además de usar filtros para eliminar el ruido y sombra; y para el seguimiento se usó el Filtro de Kalman.

ABSTRACT

There are different methods for detecting objects using different devices such as web cameras, Kinect, stereoscopic cameras, etc. Most of these studies are focused on detection of objects for security in society, whether detect theft, monitor traffic on the roads, counting people in a scene, etc. But there aren't systems to detect changes in the movement of people, or children's games, but for educational purposes. This Thesis is motivated by the need to track a person in a room for the project CHILDMNEMOS.

Throughout this thesis four algorithms have been developed in order to detect a person get moving with devices located in the ceiling of a scene, so the algorithms and the devices detects the person vertically and not in front of the person. This was achieved by an analysis of possible algorithms that could answer our need using two different devices capture video frames, we use a Logitech 9000 and Kinect camera. Besides using artificial vision and libraries for developing applications for Kinect. These libraries are OpenCV and OpenNI, respectively.

As steps in the algorithms developed are known techniques such as background subtraction, Mixture of Gauss, Kalman Filter, and OpenNI library functions that work with Kinect. Will compare the algorithms and devices for which algorithm and which device are those that offer better results.

In conducting the experiments, better results were obtained with an algorithm developed for Kinect, using the technique proposed, which consists of analyzing only a region of the scene captured by Kinect (Head and shoulders of the person) which allowed background subtraction in addition to using filters to remove noise and shadow, and for Tracking Kalman filter was used.

AGRADECIMIENTOS

Quisiera agradecer sinceramente a algunas personas, sin las que este trabajo no se hubiera podido llevar a cabo.

Primero agradecer a mi Directora de Tesina M. Carmen Juan, por su supervisión, por brindarme sus conocimientos, sus ideas y guías, base fundamental de esta tesina; además de su comprensión y paciencia a lo largo del trabajo.

A mis padres por apoyarme para realizar el Máster fuera de mi país, por su gran esfuerzo, por su comprensión, paciencia y confianza en mí. A Madelaine y Víctor, por el apoyo incondicional, tanto moral como académico. A Alba y Laura porque en los momentos malos con una sonrisa y una palabra conseguían hacerme sonreír y ser positiva. Al resto de mi familia en Bolivia, que a pesar de la distancia han estado presentes en todo momento a lo largo del Máster.

A las personas que se convirtieron en mi familia en España, por su amistad, cariño, y apoyo en todo momento.

A mis amigos y amigas en España, dentro y fuera del Máster, por su apoyo, comprensión y cariño, y especialmente por su hospitalidad en Valencia, y hacer que me sienta cómoda y en familia.

A mis amigos y amigas de Bolivia, por su apoyo moral y su amistad sincera.

Finalmente, a mi Abuelo, quien me apoyó, y confió en mí en todo momento. Esta tesina va dedicada a él, en su memoria; que hace 9 meses partió…

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I

INTRODUCCIÓN

1.1 Motivación

Actualmente, debido a la demanda que exige la sociedad en la seguridad, existen diferentes sistemas de supervisión de personas (p.e. video vigilancia, vigilancia ambiental o vigilancia militar). Estos sistemas, no solo vigilan los objetos (personas, autos, animales, etc.) que están en un lugar, también existen sistemas que realizan el conteo de objetos por un determinado lugar, sistemas de detección de objetos perdidos o robados, sistemas que controlan y vigilan los vehículos de forma aérea. Todos estos tipos de sistemas tienen fines diferentes, pero lo que tienen en común es realizar la detección y seguimiento de un individuo, analizar una escena determinada con dicho individuo mediante sensores y el uso de diferentes algoritmos.

Revisando el estado del arte sobre sistemas de detección y/o seguimiento, se ha observado que la mayoría de los sistemas se encargan de velar por la seguridad de las personas u objetos. Pero, no se han encontrado sistemas de este tipo que se usen para detectar alteraciones en el movimiento de las personas, es decir, control de niños en una habitación o trayectoria de una persona en una escena. Es por esta razón que se realizará un sistema de detección y seguimiento con un fin distinto a los ya existentes, el cual se mencionará más adelante.

Este trabajo final de máster está motivado por el proyecto CHILDMNEMOS, el cual tiene por objetivo principal desarrollar sistemas de Realidad Virtual y Aumentada que permitan a los profesionales evaluar la memoria espacial a corto plazo en niños sin patología previa identificada. Lo que se desea es comprender el desarrollo de la memoria y la orientación espacial en niños videntes y no videntes. Dentro las propuestas del proyecto se consideran estímulos auditivos que pueden ser utilizados con niños videntes y no videntes que se mueven en un entorno real.

Una de las propuestas del proyecto CHILDMNEMOS es desarrollar un sistema auditivo. En este caso se pretende identificar el movimiento de los usuarios en una habitación con sonidos; es decir, los sonidos estarán en diferentes lugares de la habitación y los niños tendrán que identificarlos caminando hacia el lugar de donde cree que el sonido se produjo. El sistema identificará su movimiento y se evaluará si el niño acertó con la posición del sonido o no.

Para realizar el sistema auditivo, mencionado anteriormente, primero se deben determinar qué técnicas se deberán usar para la detección y seguimiento de una persona, entonces lo que se desea realizar en el trabajo final de máster es un sistema de detección y seguimiento de una persona en una habitación, usando nuevas técnicas; pero a la vez evaluar distintos algoritmos para luego poder realizar una comparación y determinar cuál es el mejor.

En el primer año del proyecto CHILDMNEMOS, se ha desarrollado un sistema de Realidad Aumentada para evaluar la memoria espacial a corto plazo en niños. Este primer sistema de Realidad Aumentada está basado en marcadores. Los marcadores se introducen en varias cajas. Los niños tienen que abrir las cajas en un orden establecido y recordar los objetos que aparecen sobre los marcadores. Los objetos están modelados en 3D, y se observan a través de una *tablet*. La **Figura 1.1** muestra un ejemplo de dicho funcionamiento.

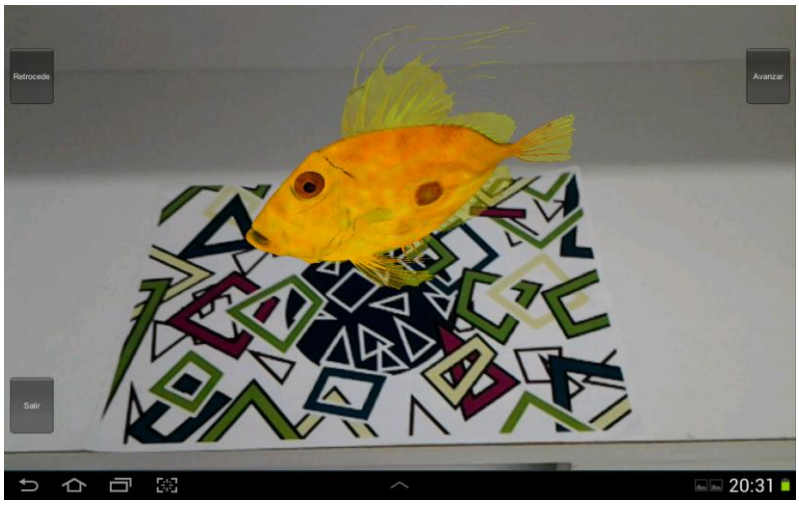


Figura 1.1. Visión del identificador a través de la tablet

Podríamos analizar el recorrido del niño en la habitación al observar los objetos en las cajas y luego el recorrido que realiza para indicar dónde se encuentra cada objeto

visto, y así poder analizar su memoria espacial a corto plazo. Este análisis se podría realizar con el sistema de detección y seguimiento que se plantea en este trabajo final de máster. Al igual que otra de las propuestas de CHILDMNEMOS; sistema auditivo; se realizará el seguimiento del niño en la habitación y de esta forma analizar su comportamiento al momento de recordar los objetos o sonidos, y evaluar su memoria espacial a corto plazo.

Los sistemas destinados a la supervisión de determinados eventos, tienen como objetivo proveer una interpretación automática de la escena basándose en la información adquirida por los sensores. Estos sistemas tienen básicamente 4 etapas: segmentación de primer plano o sustracción de fondo, identificación de regiones de interés, clasificación de esas regiones (personas u objetos) y, finalmente, la discriminación entre personas en movimiento para las zonas clasificadas como tales. La etapa de identificación de regiones de interés (llamada también de detección y segmentación de objetos de interés), en la secuencia de video capturada, es fundamental debido a que en las etapas posteriores de análisis como el seguimiento de objetos o personas, dependen directamente de la eficiencia de esta etapa.

Como se mencionó anteriormente, se pretende usar nuevas técnicas, técnicas que sean más precisas y fiables. Haciendo una exploración de nuevas técnicas para mejorar las existentes, se encuentra la posibilidad de usar el Sensor Kinect, dispositivo que proporciona información de la profundidad de cada uno de los elementos presentes en la escena

El lanzamiento del sensor Kinect en Noviembre del 2010, se lanzó pensando en la industria de los videojuegos. Pero, no tardaron en explorar el potencial del dispositivo; además que al tener un bajo coste y una precisión de datos aceptable, empezaron a usarlo en otro tipo de áreas como la robótica, medicina, informática, etc.

También se usará una Logitech Webcam Pro 9000, para hacer el mismo análisis que con Kinect. Se realizarán pruebas con diferentes algoritmos y con ambos dispositivos, de esta forma se obtendrán resultados de ambos que se compararán, y permitirá determinar qué dispositivo y qué algoritmo ofrece los mejores resultados.

Se realizará un estudio con 4 algoritmos planteados. Por un lado, 2 algoritmos (Algoritmo 1 y Algoritmo 4), estarán implementados para la Logitech Webcam Pro 9000 y 2 algoritmos (Algoritmo 2 y Algoritmo 3) implementados para la Kinect. El Algoritmo 1 y Algoritmo 4, realizan la detección mediante "*Mixture of Gauss*". El Algoritmo 2 y Algoritmo 3 utilizan, funciones específicas que contiene la librería

OpenNI. En los algoritmos mencionados, excepto en el algoritmo 3, el seguimiento se realiza mediante Kalman. El desarrollo del Trabajo final de Máster se realizará en C++, con el uso de las librerías OpenCV y OpenNI.

1.2 Objetivos e Hipótesis

El objetivo principal de este Trabajo Final de Máster es estudiar e implementar diversos algoritmos para la detección y seguimiento de una persona en una habitación, y compararlos para evidenciar pros y contras de cada uno de ellos.

Para lograr este objetivo se han planteado distintos objetivos secundarios:

- Revisar teóricamente la literatura existente sobre técnicas de detección y técnicas de seguimiento, para elegir los algoritmos que serán aplicados.
- Determinar los algoritmos para implementarlos.
- Estudiar al dispositivo de captura de información de profundidad (Sensor Kinect) para realizar la respectiva implementación.
- Determinar el entorno de desarrollo más adecuado para Kinect y la cámara, de acuerdo a sus prestaciones y a nuestros requerimientos.
- Determinar las librerías que serán necesarias para desarrollar los algoritmos.
- Estudiar las librerías OpenCV y OpenNI para desarrollar los algoritmos.
- Determinar el ambiente donde se realizarán las evaluaciones de los algoritmos.
- Realizar un análisis estadístico de los datos.

1.3 Estructura de la Tesina

El documento de la tesis está estructurado de la siguiente forma:

Capítulo 1 presenta la motivación, los objetivos y la estructura de este documento.

Capítulo 2 muestra el estado del arte, una visión detallada de la literatura relacionada con el estudio de técnicas de detección y seguimiento, además de información relacionada con el presente trabajo.

Capítulo 3 muestra la descripción de cada uno de los algoritmos estudiados y desarrollados.

Capítulo 4 muestra un análisis de los resultados que se obtuvieron con los diferentes algoritmos desarrollados.

Capítulo 5 finaliza el trabajo con un resumen de los principales logros del trabajo, las conclusiones y sugerencias para el trabajo futuro.

CAPÍTULO II

ESTADO DEL ARTE

2.1 Introducción

En este capítulo se comentan trabajos previos, sobre reconocimiento y seguimiento de personas y/o objetos, mediante cámaras. La mayoría de los sistemas hacen detección de partes específicas del cuerpo, o detectan objetos (coches, animales, etc.) con cámaras aéreas, y hacen el seguimiento de éstos; también existen sistemas que realizan conteo de personas u objetos así como también sistemas de video vigilancia, que son los más comunes. Existiendo sistemas que realizan la detección y seguimiento de una persona u objeto, pero no con las características precisas y finalidades del nuestro. A continuación se describen, en primer lugar, la visión artificial en general; posteriormente se hará una breve descripción de las fases de la detección y los sistemas de detección que existen actualmente; así como también las técnicas de seguimiento existentes, y finalmente sobre la información de profundidad.

2.2 Visión Artificial

En la década de los cincuenta, dentro de la comunidad científica se genera el interrogante de la posibilidad de enseñar a los ordenadores a realizar tareas asociadas con la inteligencia humana, entre las cuales está la capacidad de resolver problemas, analizar información visual o comprender lenguajes (Poppe, 2010). La visión artificial funciona de manera semejante con la visión humana (Moeslund et al., 2006); las computadoras y la visión humana trabajan de manera similar, en términos de funcionalidad, pero no tienen exactamente las mismas funciones (Yilmaz, 2006). El recurso básico para la visión artificial es una imagen obtenida mediante un dispositivo (cámara). La imagen es una matriz de puntos correspondientes al valor de una función bidimensional $f(x,y)$ donde x e y son coordenadas espaciales y el valor de f, representa el brillo de la imagen. En el caso de las imágenes a blanco y negro e

imágenes a color, corresponde a la combinación de tres matrices en el modelo de color RGB.

Actualmente la visión artificial se usa bastante en diferentes aplicaciones, sabemos que la visión artificial tiene como finalidad la extracción de información del mundo físico a partir de imágenes, utilizando para ello un ordenador (Velez et al., 2003). Un sistema de Visión Artificial actúa sobre una representación de una realidad que le proporciona información sobre colores, formas, brillo, etc. Estas representaciones suelen estar en forma de imágenes estáticas, escenas tridimensionales o imágenes en movimiento. El ser humano captura la luz a través de los ojos, y esta información circula a través del nervio óptico, hasta el cerebro que es donde se procesa la imagen. Es decir, el cerebro descompone la imagen en diferentes segmentos. Después del procesado, el cerebro interpreta la escena.

Las cámaras de video son algunos de los sensores más utilizados en un gran número de aplicaciones, especialmente para la vigilancia. Existe necesidad de desarrollar algoritmos para tareas tales como la detección, seguimiento, y el reconocimiento de objetos, haciendo uso de una cámara o redes distribuidas de cámaras. Para el uso de estos dispositivos, se debe resaltar el uso eficiente de las restricciones geométricas inducidas por los dispositivos de imagen para derivar algoritmos distribuidos para la detección de objetos, seguimiento y reconocimiento (Sankaranarayanan et al., 2008).

2.2.1 Detección y Seguimiento

Al ser la finalidad de la Visión artificial extraer información del mundo físico a través de imágenes, y como se mencionó anteriormente, funciona de forma similar que la visión Humana, se tienen 4 fases, que vienen a formar parte de la Detección de objetos **(Figura 2.1)**:

- **Captura:** consiste en la adquisición de las imágenes digitales mediante algún tipo de dispositivo o sensor.
- **Procesamiento:** esta segunda etapa consiste en el tratamiento de las imágenes, con el objetivo de facilitar las etapas posteriores. Es decir, es dónde, mediante filtros y transformaciones geométricas, se eliminan partes que no se desean en la imagen o se realzan partes que se necesitan (según la necesidad).

- **Segmentación:** esta etapa consiste en aislar los elementos que nos interesan de una escena (regiones de interés de la escena).
- **Reconocimiento o Clasificación:** por último en esta etapa, se pretende distinguir los objetos segmentados, gracias al análisis de ciertas características que se establecen previamente para diferenciarlos.

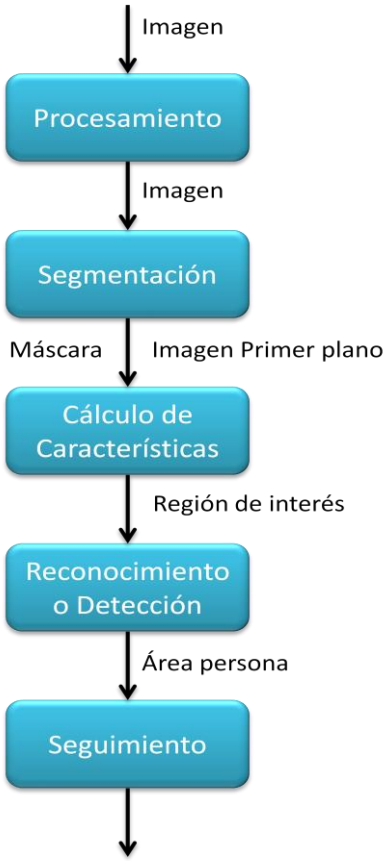


Figura 2.1. Fases de la detección de objetos

Estas cuatro fases, muchas veces no se siguen de manera secuencial, ya que en ocasiones debe existir una retroalimentación. Es decir, que es normal volver a la etapa de segmentación, si en la etapa de reconocimiento existe un error, o a la etapa de procesamiento, si es necesario. La secuencia de las fases depende también del sistema de detección que se desarrolle.

Actualmente existen diferentes técnicas para cada fase, y se pueden combinar de distintas maneras, dependiendo del tipo de detección que se desee. Pero se debe elegir la técnica que mejor se adecue al sistema de detección. Zhang et

al. (2004) indican que la detección de objetos, es un problema difícil debido a los desafíos que se tienen, es decir:

- Pose: el objeto es muy variable, debido a la posición que existe entre la cámara y el objeto.
- Presencia o ausencia de componentes estructurales: en los objetos de clase, algunos componentes pueden estar presentes o ausentes. Por ejemplo, bigote en la cara.
- Oclusión: los objetos pueden ser ocluidos por otros objetos.
- Condiciones de las imágenes: Diferentes iluminaciones y las características de la cámara pueden hacer que las imágenes sean diferentes.

Existen diferentes métodos para la detección, unos más usados que otros, además dependiendo del tipo de características que se tienen en el sistema. Karasulu (2010) realiza el estudio de diferentes técnicas usadas para la detección y seguimiento de objetos, dentro las cuales está la Sustracción de fondo, con el uso de una Gaussiana, o modelo de mezclas Gaussianas. Además menciona otras técnicas de detección de seguimiento como la segmentación de las imágenes, *Mean-Shift* y la estimación de densidad del kernel, *CAMShift*, y flujo óptico (**Figura 2.2**). Yilmaz et al. (2006) describen las diferentes técnicas existentes para la detección de objetos, donde también se menciona a la Sustracción de fondo.

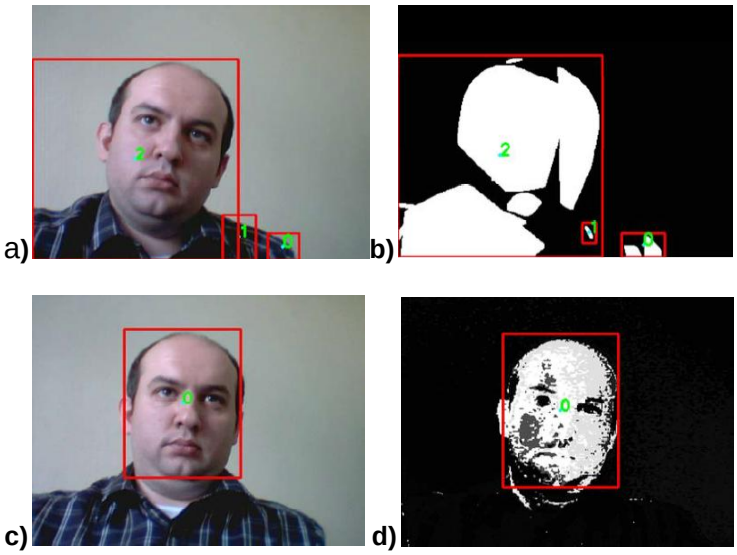


Figura 2.2. Estudio de Karasulu
a) Sustracción de fondo, b) Máscara de la sustracción de fondo, c) Seguimiento con CAMShift y d) Retroproyección de CAMShift

Córdoba (2012) presentó un sistema para la detección de robo o abandono de objetos en interiores (**Figura 2.3**), usando la cámara de profundidad de Kinect, la cual la integra al prototipo ya existente en el grupo VPU-Lab (*Video Processing and Understanding Lab*). El sistema realiza las etapas de: Detección del primer plano (*Mixture of Gauss*), Detección de *blobs* estacionarios y Clasificación de objetos. Una de las ventajas de usar la información de profundidad es que la máscara de *foreground* de profundidad no presenta falsos positivos, sin embargo, el caso opuesto (falsos negativos), sí es muy propenso a ocurrir debido a ciertas características de los objetos.

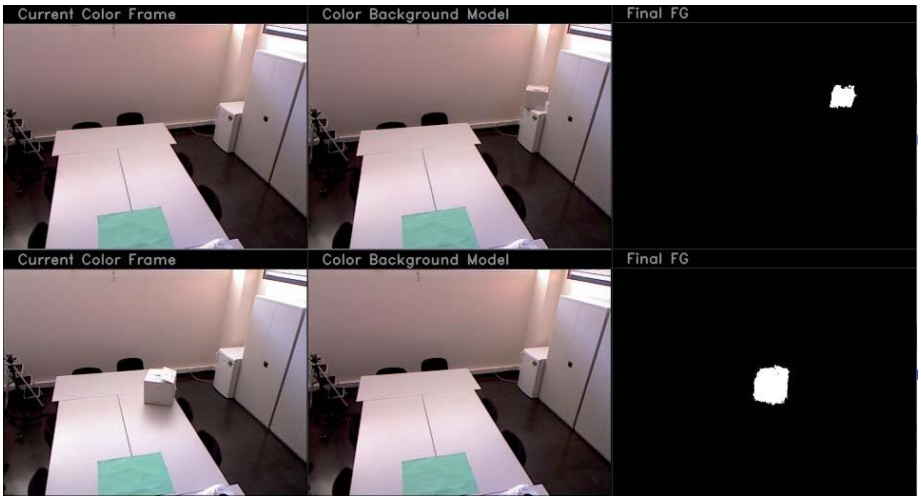


Figura 2.3. Córdoba, Detección de robo/abandono de objetos en el sistema.

González (2010), plantea un sistema de detección de personas, usando la segmentación del *Foreground* y *Background* usando como método la Sustracción de fondo. Además, estudió los algoritmos FGD (basado en el modelo de Bayes) y el MoG (basado en el modelo de mezclas Gaussianas), para elegir el mejor entre los dos. MoG tiene mejores resultados en cambios de iluminación, *foreground* estático, *foreground* dinámico solitario y en grupo, pero se elige a FGD, debido a que se busca obtener buenos resultados sobre todo a medias distancias. Con este algoritmo se consigue representar los contornos de manera mucho más fiel al tener menos ruido si bien se comporta peor en algunas situaciones concretas (personas lejanas, personas paradas desde un principio y personas con ropa semejante al fondo de la imagen).

Sanabria et al. (2011) indican que las aplicaciones inherentes a la seguridad han motivado el estudio de técnicas para realizar el seguimiento en ambientes

no controlados. El objetivo principal del seguimiento, es conocer en todo momento la región donde se produjo el movimiento y la trayectoria de un objeto detectado en cada uno de los fotogramas del video, en base a la región que se proyecta en cada instante de tiempo. Para esto primero se tiene una detección inicial del objeto, y posteriormente hacer el seguimiento de este objeto, para el cual existen diferentes técnicas. Las técnicas varían de acuerdo a las particularidades del movimiento y el ambiente en el cual se desarrolla, básicamente existen dos líneas: basado en un modelo planteado a partir del conocimiento del fenómeno (basado en un modelo), o la inferencia a partir de la información obtenida de los datos, es decir un análisis libre.

La información en tiempo real es muy usada para aplicaciones de video vigilancia, Xu et al. (2010) presentan un sistema de conteo, que consiste en cuatro módulos: extracción del primer plano, detección del componente cabeza – hombros, seguimiento y análisis de trayectoria (**Figura 2.4**). Para reducir el coste computacional, y enfrentar situaciones de vigilancia complejas para la extracción del primer plano, se seleccionó la técnica *Mixture of Gaussians*. Para el seguimiento se usa el filtro de Kalman, los beneficios que ofrece este filtro, se utilizan para eliminar falsos negativos y predecir la posición de candidatos no detectados (falsos positivos). Con estas técnicas se obtuvieron resultados satisfactorios en tiempo real.

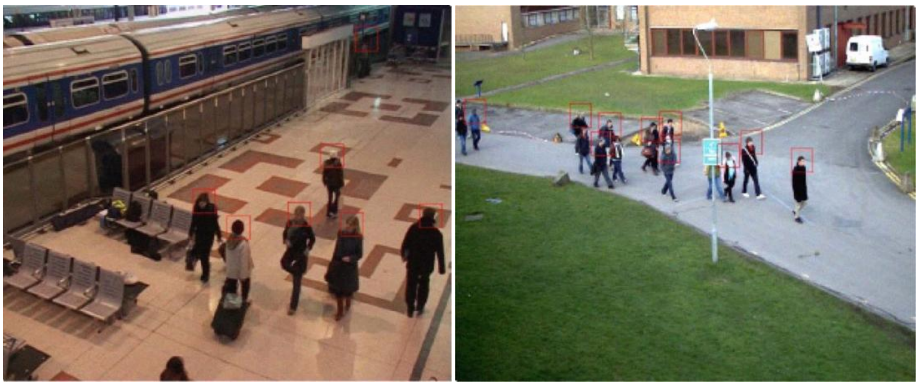


Figura 2.4. Xu et al., Sistema de conteo de personas.

Existen otro tipo de sistemas de Video vigilancia, que son para preservar la intimidad de las personas (Gutierrez, 2011), el cual realiza la detección de la persona, y oculta el rostro (**Figura 2.5**). Para la detección usa la sustracción de fondo usando la técnica *Mixture of Gaussians*, y además usa el algoritmo de

Viola Jones combinado con el Histograma de Gradientes Orientados para la detección de caras, y finalmente para el seguimiento usa el método *CAMShift*, con el uso de la librería OpenCV.

Figura 2.5. Gutierrez, Detección ocultando el rostro de la persona.

Garcia et al. (2012) realizan la detección que se basa en la búsqueda de las cabezas por medio de una correlación de la imagen ya pre-procesada con distintos patrones circulares. Utilizan estereovisión, ya que filtran las detecciones en función de la altura. El seguimiento se realiza con un algoritmo de múltiples hipótesis, basado en el Filtro de Kalman. Se consiguen tasas de detección entre el 87% y el 98% según el número de personas que atraviesan la zona de conteo a partir de vídeos capturados en escenarios reales.

En este trabajo final de máster, lo que nos interesa es que la cámara esté situada de forma cenital con respecto al suelo (perpendicular respecto del suelo y la imagen obtenida ofrece un campo de visión orientada de arriba – abajo), como en el trabajo de Rossi et al. (1994), se describe el desarrollo de un sistema capaz de detectar y seguir a las personas que se desplazan en una escena. Existe un módulo de detección de movimiento determina en primer lugar si una persona ha entrado en la escena. La estrategia empleada, es usar parte de la región de interés en el movimiento, que contiene suficiente información en términos de la variabilidad del nivel de gris. Es decir, que las regiones con alto contenido en frecuencia espacial o se consideran adecuados para continuar con el módulo de seguimiento. Si ninguna parte de la región cumple con el requisito mencionado anteriormente, la plantilla se omite y no se pasa al módulo de seguimiento. Y un módulo de seguimiento y predicción, que se basa en predecir la posición de la región de interés en la imagen siguiente de búsqueda, el uso de un criterio de distancia para determinar la mejor

coincidencia cerca de la posición predicha, y si se encuentra una coincidencia, mejorar la posición y actualizar la región para la nueva fase de búsqueda. Esto se puede aplicar sólo si se comprueba que el fondo no tiene una textura fuerte o, al menos localmente, los objetos son más variados que el propio fondo.

Un trabajo similar al nuestro es de Chen et al. (2006), donde realizan el conteo de personas que pasan a través de una puerta (**Figura 2.6**), y la cámara está situada de forma cenital; usando como técnica de detección la Sustracción de fondo, y para el seguimiento primero realiza un análisis del histograma HSI, esto para analizar la cromaticidad de la ropa de las personas, asociar un vector de color con cada persona según el color de su ropa, basado en una estrategia de intersección *bounding-box*.

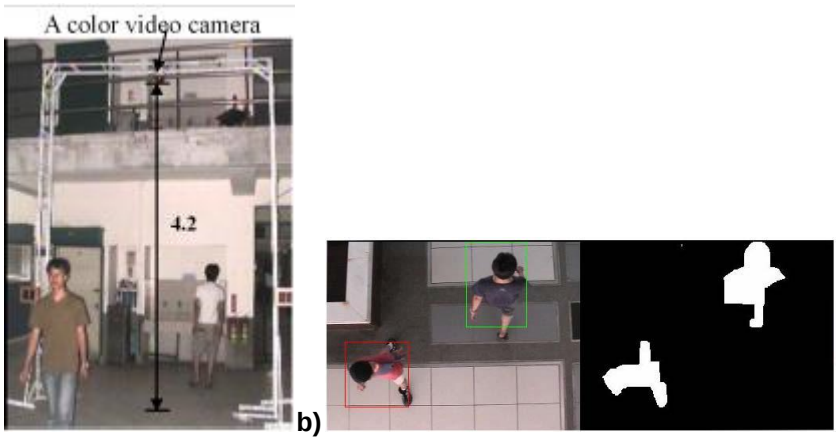


Figura 2.6. Chen et al., Conteo de personas
a) Puerta por la que atraviesan las personas b) Detección y Seguimiento con vectores de colores.

Yu et al. 2008 la estimación del número de personas que pasan por un portón o una puerta proporciona información útil para la vídeo vigilancia y aplicaciones basadas en monitoreo (**Figura 2.7 a)**). Es por eso que plantean un método para el conteo de personas, que se basa en la detección de bordes del *Foreground* y *Background*. En Barandiaran et al. (2008) que también tienen la cámara situada cenitalmente, realizan el conteo de personas, analizando una zona compuesta por un conjunto de líneas virtuales (**Figura 2.7 b)**). En ambos trabajos utilizan el flujo óptico que genera una persona al atravesar la zona de conteo para llevar a cabo la detección, y se obtuvieron resultados exitosos.

Revisando la literatura se han encontrado sistemas con los dispositivos orientados de esta forma, pero con diferentes características, y estos sistemas son los que tienen la visión aérea, o los UAV, vehículo aéreo no tripulado,

(*Unmanned Aerial Vehicle*), enfocados a la vigilancia militar, vigilancia de tráfico vehicular, vigilancia de animales, etc

a)  **b)** 

Figura 2.7. Conteo de personas a) Yu et al. 2008 b) Barandiaran el al. 2008.

Masoud et al. (2001) presentan un sistema en tiempo real para el seguimiento de objetos, en una secuencia de imágenes en escala de grises, adquiridas por una cámara fija. El objetivo es la integración de este sistema con una aplicación de control de tráfico, tales como un esquema de control de peatones en las intersecciones. El seguimiento a formas geométricas, se modela como un problema de optimización gráfica. Los peatones se modelan como *patches* rectangulares con un comportamiento dinámico. Se usa el filtro de Kalman, para estimar los parámetros de peatones. El sistema fue implementado en un Datacube MaxVideo 20, equipado con un cubo de datos Max860 y fue capaz de lograr un rendimiento máximo de más de 30 *frames* por segundo. Los resultados experimentales basados en las escenas de interior y exterior, demostraron la robustez del sistema en muchas situaciones como las oclusiones totales o parciales de los peatones.

Chan et al. (2008) presentan un sistema de preservación de la privacidad, para estimar el tamaño de los grupos, compuesto por peatones que caminan en diferentes direcciones. En primer lugar la multitud se segmenta en componentes de movimiento homogéneo, utilizando una mezcla del modelo de movimiento de texturas dinámicas; en segundo lugar, un conjunto de simples características se extrae de cada región segmentada, y la correspondencia entre las características y el número de personas por segmento, se realiza con un aprendizaje con el proceso de regresión de Gauss. Se obtienen buenos resultados, pero existen errores cuando bicicletas, *skateboarders*, atraviesan la vía rápidamente, pero nos dicen que estos errores se puedan subsanar añadiendo más componentes de *Mixture of Gaussian*, para la segmentación.

Khashman (2008) introduce un sistema de detección, extracción y reconocimiento automático de objetos en movimiento (aMORDERs), el cual tiene 3 fases, combinando primero el procesamiento de imágenes con reglas deterministas para determinar el movimiento del objeto, segundo extraer el objeto detectado y finalmente usa redes neuronales basadas en un algoritmo de aprendizaje para el reconocimiento del objeto extraído. La aplicación tuvo éxito, ya que tuvo una tasa de detección del 94.6% y además al usar redes neuronales, se puede fácilmente y rápidamente entrenarla para reconocer diferentes objetos.

Chhabraa (2009) en su sistema usa *Mixture of Gaussian*, es una técnica robusta y adaptable a los cambios de ambiente. En una evaluación cuantitativa, Sen-Ching et al. (2004) comparan *Mixture of Gaussian* con una mediana recursiva, obteniendo mejores resultados con *Mixture of Gaussian* ya que ofrece mayor precisión, además que el coste computacional es menor.

Breckon et al. (2009) realizan una detección de vehículos en tiempo real, de forma aérea, es decir mediante un UAV. Para lo cual utiliza un enfoque de dos etapas. La primera consiste en hacer uso del *Cascade Haar Classifier* para obtener regiones candidatas a ser vehículos, y una segunda etapa para la verificación de estas regiones, basándose en el tamaño rectangular, así se aseguran de que es un vehículo. En cuanto a los resultados que obtuvieron en este proyecto son exitosos, se logró detectar vehículos estáticos y en movimiento en entornos de diferente complejidad en términos de desorden. El método que usan, mejora las tasas de detección de vehículos individuales, y mejora la reducción de falsos positivos.

Los sistemas de imagen aérea de una sola vez, pueden funcionar, en entornos difíciles demasiado peligrosos para las aeronaves pilotadas o naves autónomas, por lo que este tipo de sistemas es útil en la asistencia para desastres, la exploración y algunos otros campos. Ge et al. (2009) desarrollan un método para el reconocimiento de un objetivo y seguimiento de secuencias de imágenes de este tipo de sistemas. El reconocimiento del objetivo se lleva a cabo con las características locales de los puntos más importantes. Usan el descriptor de imagen SIFT (*Scale Invariant Feature Transform*), capaz de detectar puntos característicos estables en una imagen de una sola vez.

Continuando con el análisis de videos aéreos, Huang et al. (2010) plantean una forma de detección de objetos en movimiento mediante un método hibrido, y toma como objeto a un vehículo, para esto lo primero que considera es alinear los cuadros de escena consecutivos del video utilizando la detección de esquinas de *Harris* y rastreo de *Kanade-Lucas-Tomasi* (KLT). Después aplica un método de diferenciación de marcos de imagen acumulativos, donde se obtienen los píxeles del objeto en movimiento basándose en el color de umbral y las regiones cerradas del vehículo y la carretera, con esto detectan el vehículo en movimiento, para obtener la posición y forma de este objeto utiliza un enfoque de *morphing*, que básicamente consiste en hacer un historial de regiones obtenidas. Por último se adaptó el filtro de Kalman para poder etiquetar el objeto y realizar un seguimiento.

En este tipo de sistemas también existen algunos problemas, como la resolución de las imágenes o el movimiento de la cámara. Xingbao et al. (2011) proponen un algoritmo basado en la detección de prominencia y el filtro de Kalman. Como resultado no solo se reduce el tiempo de cálculo, sino que también mejora la capacidad de adaptación al tiempo real. Lee (2011) propone un *framework* para el reconocimiento de las interacciones Persona – vehículo, por medio de video aéreo. Presenta un enfoque basado en lógica temporal que no requiere de entrenamiento a partir de ejemplos. A bajo nivel, se cuenta con programación dinámica para realizar el ajuste del modelo entre los vehículos seguidos y el renderizado 3D de los modelos de vehículos. En el nivel semántico, dada la localización de la región de interés, se verifica la historia de las interacciones hombre-vehículo con las definiciones de acontecimientos predefinidos.

Wang et al. (2011) usan un modelo pLSA (*Probabilistic Latent Semantic Analysis*), para descubrir el contexto semántico de los *patches* locales de una imagen y la imagen completa. Utilizando la relación entre ellos, se pueden detectar las regiones de interés. Los resultados experimentales muestran que es posible la detección de regiones de interés desde una visión aérea.

Choi et al. (2012) presentaron el diseño e implementación de un sistema de reconocimiento en video vigilancia aérea, utilizando Modelos de Entidad Relación (ERM). En este enfoque, se busca una actividad que sea equivalente a enviar una consulta a la RDBMS (*Relational DataBase Management System*).

Mediante la incorporación de datos de imágenes de referencia y de Información Geográfica, el seguimiento de objetos se asocia con significados físicos. Los resultados que se obtuvieron usando ERM, fueron que se identificaron actividades simples como “vuelta en U”, “2 point turn”, “3 point turn”, “entada – Salida”, “loop”, y “exceso de velocidad”, así como actividades complejas con múltiples actores. Las actividades se visualizan por un software disponible para la visualización de datos geoespaciales como Google Earth.

Babu (2012) propone un algoritmo basado en la detección de objetos móviles y el algoritmo de seguimiento de vehículos; está implementado con MATLAB, y lo que hace es descomponer un video AVI en componentes R, G y B. Se muestra la imagen en escala de grises, y se hace una sustracción de fondo, luego se elimina la sombra y se procede a realizar operaciones morfológicas para detectar el objeto y luego se calcula el área de dicho objeto **(Figura 2.7)**. Este algoritmo detecta la región en movimiento más pequeña, más lenta o rápida, con precisión. Trabaja en tiempo real o con vídeos.

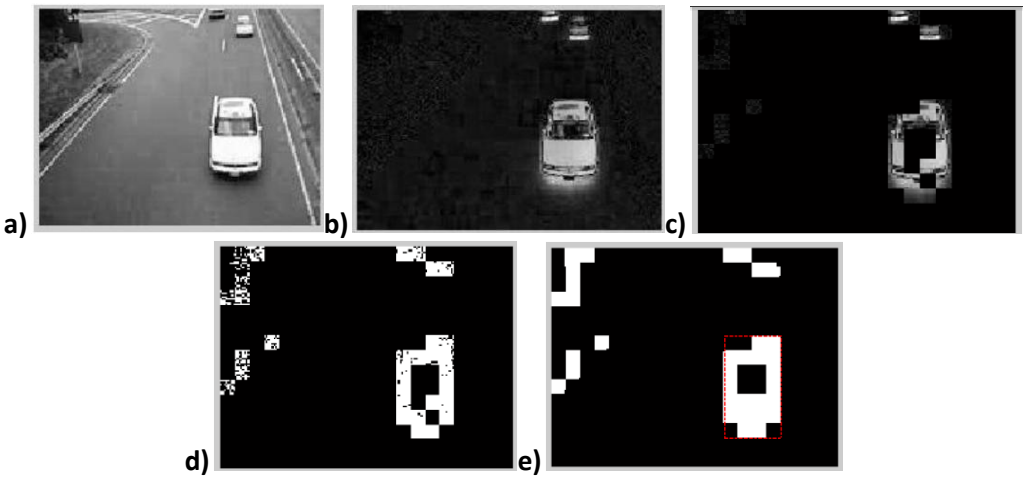


Figura 2.7. Detectar al objeto según algoritmo planteado por Babu (2012) a) Objeto en movimiento, b) Imagen con sombra, c) Eliminación de sombra, d) Después de comparar con el umbral de fondo, y e) Etiquetado con objetos en movimiento después de la transformación morfológica.

Rodríguez-Canosa et al. (2012) desarrollan un método en tiempo real para detectar y seguir objetos en movimiento (DATMO) desde un UAV con una sola cámara (**Figura 2.8**). Este método trata de crear un campo de flujo óptico artificial, mediante la estimación del movimiento de la cámara entre dos

fotogramas sucesivos; haciendo una comparación con un flujo óptico real. Usa el filtro de Kalman extendido para predecir el siguiente estado en el sistema.

Figura 2.8. Modelo de UAV "AscTec Pelican quadrotor"

2.3 Información de profundidad

Existen diferentes técnicas para capturar la profundidad de una determinada escena. Las técnicas más conocidas se dividen en 3 grupos: microondas, ondas de luz y ondas ultrasónicas. Como se puede ver en la **Figura 2.9**, el grupo más relacionado con la presente tesina es Ondas de Luz, ya que dentro de estos métodos, está el principio del funcionamiento del sensor Kinect. En este grupo se incluyen cuatro métodos basados en ondas de luz: triangulación (visión estéreo), ToF (*Time of flight*), luz estructurada y escaneo láser.

Estos métodos pueden ser clasificados en dos subgrupos: Activos (*Time of flight*, luz estructurada y escaneo láser) y pasivos (Triangulación). Los activos proyectan intencionalmente un haz de rayos luminosos sobre la escena para hacer que las características de la misma sean más fáciles de identificar; mientras que los métodos pasivos, intentan buscar correspondencias entre un par de imágenes, de las cuales no se conoce nada a priori.

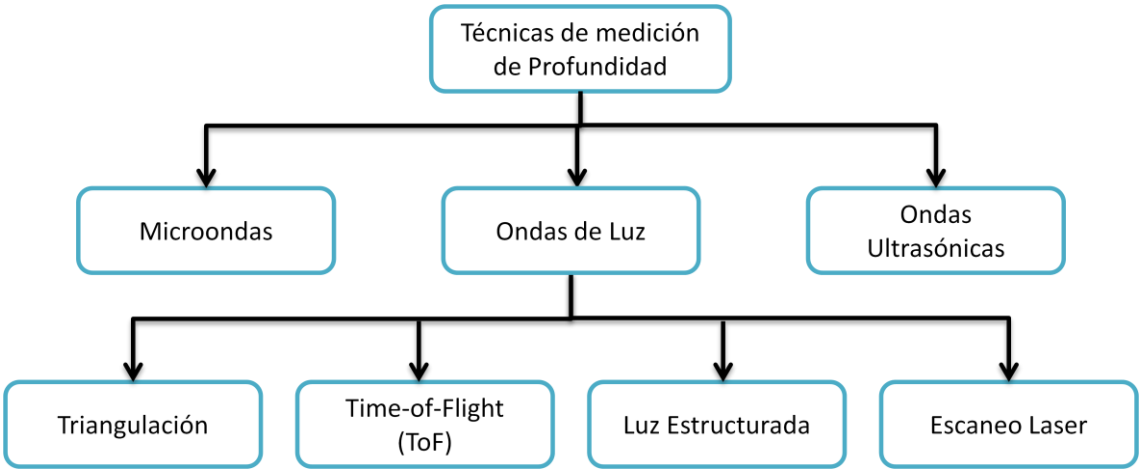


Figura 2.9. Técnicas de medición de Profundidad.

- **Triangulación**

  En este método la profundidad se obtiene considerando dos puntos de vista. Es decir, puntos de vista de las cámaras, con posiciones conocidas (Scharstein et al., 2002). Al obtener un par de imágenes, se intenta encontrar correspondencia entre Píxeles. Esto se puede realizar buscando características específicas de la escena, o eligiendo una ventana espacial arbitraria en una de las imágenes y buscando su correspondencia a través de la línea epipolar en la segunda imagen.

- **Escaneo láser (*Laser scanning*)**

  El escaneo láser, consiste en proyectar un haz de luz (láser) sobre los objetos, cuya profundidad se desea estimar, Davis et al. (2001).

- **ToF (*time of flight*)**

  Esta técnica consiste básicamente en medir el tiempo transcurrido desde que el haz de luz sale del proyector hasta que se refleja en el sensor. Es decir, el tiempo de viaje de la luz (Lee, 2012).

- **Luz estructurada**

  Luz estructurada es el principio que usa el sensor Kinect para su funcionamiento. Este método consiste en proyectar un patrón de luz sobre la escena y observar la deformación del patrón en la superficie de los objetos (Szeliski, 2010).

El patrón de luz es proyectado, por un proyector LCD o por un barrido láser. Una cámara ligeramente desplazada respecto del proyector captura la deformación de los puntos, y calcula la distancia de cada punto (Scharstein, 2003).

CAPÍTULO III

DESARROLLO

3.1 Introducción

En este capítulo se detallará el hardware y software necesarios para el desarrollo de los algoritmos, así como la descripción de desarrollo de cada algoritmo planteado. Como se mencionó anteriormente se desarrollarán 4 algoritmos, todos realizados en el lenguaje de programación C++, con las librerías OpenCV y OpenNI, según sean necesarias para cada algoritmo. Además se debe mencionar que estos algoritmos fueron desarrollados y analizados, en un mismo ambiente controlado, que viene a ser nuestro escenario para las validaciones posteriores.

3.2 Hardware y Software

Los dispositivos que se usarán en la tesina son de dos tipos, una Cámara Web Logitech QuickCam Pro 9000 y una Kinect. Ambos dispositivos se usarán para realizar la detección y seguimiento de la persona en tiempo real. La plataforma de programación que usamos es Microsoft Visual Studio 2010. El lenguaje de programación para el desarrollo de los algoritmos es C++, usando las librerías OpenCV y OpenNI.

3.2.1 Cámara Logitech

Muchas veces el funcionamiento de un algoritmo depende en gran medida de la calidad de la resolución de la imagen sobre la que se trabaja, debido a que facilita y acelera el proceso computacional de detección de los algoritmos. Al tener imágenes con una resolución aceptable, no existiría demasiado ruido en las mismas. En esta tesina se ha trabajado con una cámara Logitech QuickCam Pro 9000 (**Figura 3.1**). La cámara tiene las siguientes características:

- Óptica Zeiss® con enfoque automático

- Vídeo en alta definición (hasta 1600 x 1200) en nuestro caso tiene una resolución de 640x480
- Modo de pantalla panorámica de 720p (con sistema recomendado)
- Vídeo de hasta 30 *frames* por segundo
- Certificación USB 2.0 de alta velocidad

Figura 3.1. Cámara Logitech QuickCam Pro 9000

3.2.2 Kinect

Kinect, es un sensor desarrollado para la videoconsola Xbox 360 y desde junio del 2011 para PC (Windows 7 y Windows 8). Kinect fue lanzado en noviembre de 2010.

El sensor Kinect es una barra horizontal de aproximadamente 23 cm., conectada a una pequeña base circular con un eje de articulación de rótula, y está diseñado para ser colocado longitudinalmente por encima o por debajo de la pantalla. **(Figura 3.2).**

El dispositivo cuenta con una cámara RGB en resoluciones de 640x480 (VGA) y 1280x1024 píxeles, un sensor de profundidad (IR projector + IR camera) en resoluciones de 640x480 (VGA), 320x240 (QVGA) y 80x60 píxeles, un micrófono y un procesador personalizado que ejecuta el software patentado, que proporciona captura de movimiento de todo el cuerpo en 3D, reconocimiento facial y capacidades de reconocimiento de voz. El micrófono de Kinect permite a la Xbox 360 llevar a cabo la localización de la fuente acústica y la supresión del

ruido ambiente, permitiendo participar en el chat de Xbox Live sin utilizar auriculares. Algunas características en general de la Kinect son:

Sensores

- Lentes de color y sensación de profundidad
- Micrófono

Campo de visión

- Campo de visión horizontal: 57 grados
- Campo de visión vertical: 43 grados
- Rango de inclinación física: ± 27 grados
- Rango de profundidad del sensor: 1.2 – 3.5 metros

Data Streams (Flujo de datos)

- 320 × 240 a 16 bits de profundidad @ 30fps
- 640 × 480 32-bit de color @30fps
- Audio de 16-bit @ 16 kHz

Sistema de Seguimiento

- Rastrea hasta 6 personas, incluyendo 2 jugadores activos

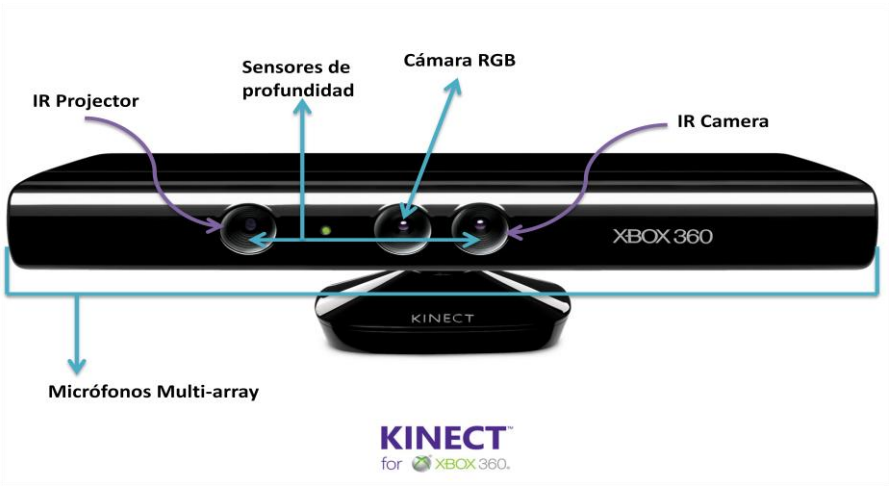


Figura 3.2. Sensor Kinect

El sensor de profundidad es un proyector de infrarrojos combinado con un sensor CMOS (*Complementary metal oxide semiconductor*) monocromo que permite a Kinect ver la habitación en 3D en cualquier condición de luz ambiental. El rango de detección de la profundidad del sensor es ajustable gracias al

software de Kinect capaz de calibrar automáticamente el sensor, basado en el ambiente físico del jugador, tal como la presencia de sofás.

3.2.3 OpenNI

Lanzado por Prime Sense como drivers para Kinect. El *Framework* OpenNI es un SDK de código abierto, para el desarrollo de sensores 3D, librerías middleware y aplicaciones.

OpenNI permite, por un lado, comunicarse con los sensores de audio, video y sensor de profundidad de Kinect, mientras que proporciona una API que sirve de puente entre el hardware del equipo, NITE Middleware y las aplicaciones e interfaces del Sistema Operativo. La idea es facilitar el desarrollo de aplicaciones que funcionen con interacción natural, llámese gestos y movimientos corporales.

Actualmente OpenNI permite la captura de movimiento en tiempo real, el reconocimiento de gestos con las manos, el uso de comandos de voz y utiliza un analizador de escena que detecta y distingue las figuras en primer plano del fondo. OpenNI facilita la obtención de imágenes de profundidad de Kinect, para ser procesadas por algunos algoritmos.

3.2.4 OpenCV

OpenCV es una biblioteca libre, de visión artificial desarrollada originalmente por Intel. OpenCV es multiplataforma, existiendo versiones para GNU/Linux, Mac OSX y Windows. Contiene funciones de programación para visión artificial en tiempo real, como el reconocimiento de objetos, calibración de cámaras, visión estéreo y visión robótica. Permite manipular imágenes de diferentes formatos, así como aplicar sobre ellas los algoritmos más comunes del tratamiento de imágenes.

OpenCV proporciona un entorno de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado, realizando su programación en código C y C++ optimizados, aprovechando además las capacidades que proveen los procesadores multi-núcleo.

En el presente trabajo la versión de OpenCV que se usa es: OpenCV 2.4.3 (Noviembre, 2012), usamos esta versión ya que incorpora más funciones, la compatibilidad con otras librerías como pueden ser OpenNI y SDK Kinect, que es otra librería de obtención de datos de Microsoft. Hasta la fecha, la versión más reciente de OpenCV es la 2.4.6 (Julio, 2013).

3.3 Algoritmos planteados

El desarrollo de los algoritmos se basa en un diagrama de procesos general, donde se incluye la detección y el seguimiento (**Figura 3.3**). Pero a la vez, cada algoritmo tiene su propio Diagrama de procesos, ya que usan diferentes procesos para realizar la Detección. El Seguimiento es el mismo en los algoritmos 1,2 y 4, y en el algoritmo 3 se utiliza el seguimiento que tiene incluido la librería OpenNI.

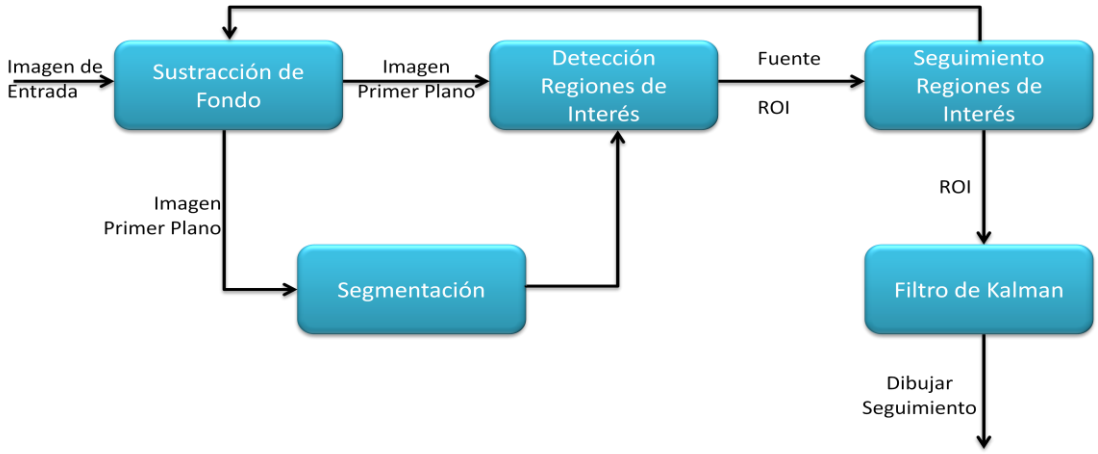


Figura 3.3. Diagrama de procesos de Detección y Seguimiento.

Los dispositivos, la Web Cam Logitech Pro 9000 y la Kinect, estarán colocados en el techo del mismo ambiente, en posición perpendicular al suelo (F**igura 3.4**), a una altura de 270 centímetros. Ambos tendrán la misma posición, de tal forma que enfoquen la misma escena (**Figura 3.5**); se debe mencionar que el campo de visión de Kinect es más amplio que de la cámara. Cabe recalcar que las escenas que se capturan con Kinect son solo con la cámara de profundidad.

Haciendo un breve resumen de los 4 algoritmos desarrollados, podemos decir que el Algoritmo 1 y Algoritmo 4 están desarrollados para ser aplicados a la Cámara Logitech Pro 9000, y con la librería OpenCV; mientras que los Algoritmos 2 y 3, están desarrollados para ser aplicados a la Kinect, con el uso de las librerías OpenCV y OpenNI. Lo que se realizó en los algoritmos para la detección es: para el Algoritmo 1 usamos "*Mixture of Gauss*"; para el Algoritmo 2 hacemos un análisis de la región de profundidad; para el Algoritmo 3, usamos funciones de OpenNI; y finalmente para el Algoritmo 4 usamos "*Mixture of Gauss*", añadiendo un "Identificador". Respecto al Seguimiento de la persona, para los Algoritmos 1,2 y 4, usamos el Filtro de Kalman,

mientras que para el Algoritmo 3, usaremos la técnica de seguimiento incorporada en la librería OpenNI.

Figura 3.4. Posición de Cámara Logitech y Kinect en el techo

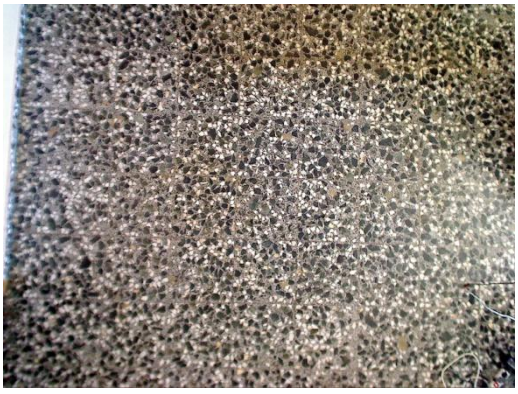
a)  b) 

Figura 3.5. Escena en la cual se desarrollan los algoritmos
a) Visión para la Cámara Logitech b) Visión para Kinect

Como se puede observar, en 3 casos, el seguimiento se realiza mediante el “Filtro de Kalman”, y es por esto que la descripción de esta etapa se realizará de forma general para los 3 algoritmos. La descripción del seguimiento del Algoritmo 3, se describe dentro del Algoritmo 3. A continuación se realiza una explicación sobre el desarrollo de los algoritmos y las diferencias que existen entre ellos, al momento de desarrollarlos. La descripción del seguimiento se encuentra después de describir cada uno de los algoritmos.

3.3.1 Algoritmo 1

Este algoritmo esta desarrollado con C++ y con la librería OpenCV, y será aplicado a la Web Cam Logitech Pro 9000. Para el desarrollo de este algoritmo existen diferentes procesos, los cuales podemos observar en el diagrama de procesos del algoritmo **(Figura 3.6)**, basado en el diagrama general, y posteriormente una explicación sobre el algoritmo

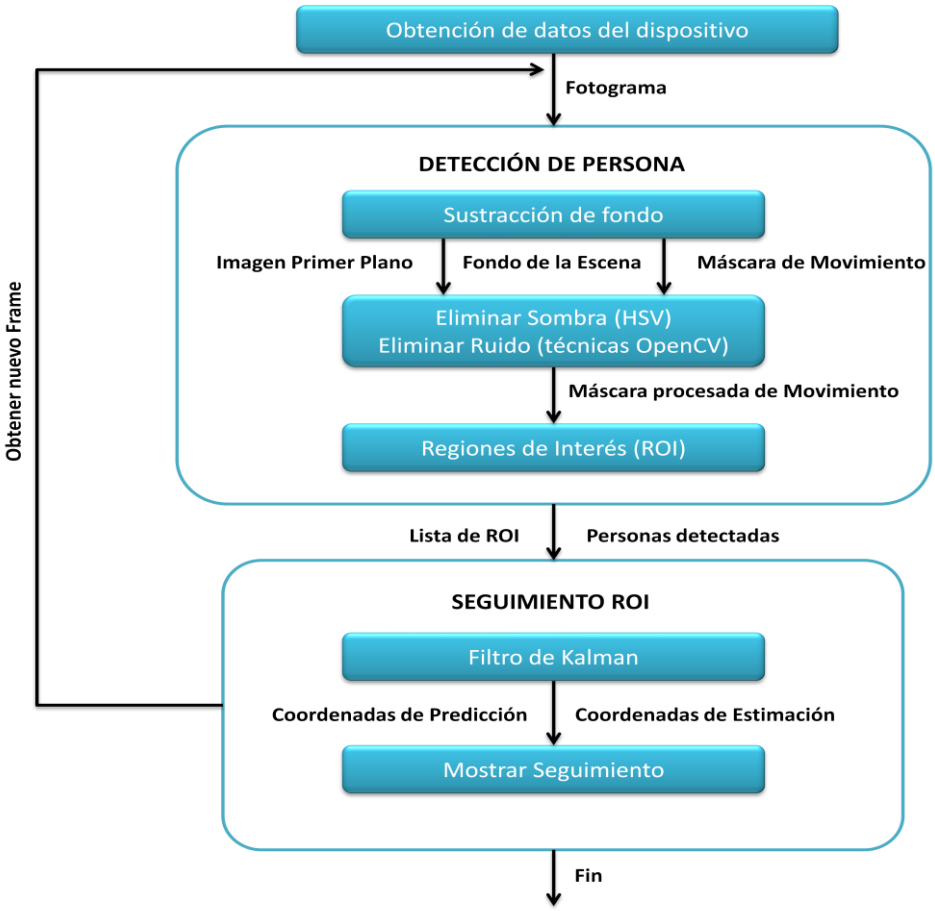


Figura 3.6. Diagrama de Procesos Algoritmo 1

a) Obtención de Datos en tiempo real

Este algoritmo planteado comienza con la extracción de la información de la escena con dicha cámara. Para la obtención de fotogramas de video la librería OpenCV brinda una función para usar el dispositivo en tiempo real o usar un video. La función es ***VideoCapture***, la cual debe ser iniciada, para después abrir el Dispositivo o el Video. Una vez abierto el dispositivo almacenamos el

fotograma capturado del dispositivo en una variable ***Mat*** la cual permite tener al mismo, almacenado en una matriz llamada ***frame***.

b) Sustracción de fondo

El primer paso para la "Detección", es realizar una sustracción de fondo de la escena, esto para poder detectar los objetos en movimiento (en nuestro caso personas). Para realizar este paso usamos un método de sustracción de fondo que se basa en el modelo *Mixture of Gauss* (MoG). Este método permite el aprendizaje del fondo de la escena mediante la combinación de distribuciones de Gauss. A partir de su aprendizaje, es capaz de extraer del fondo aquellos objetos en movimiento. La librería OpenCV, ya tiene implementado este método.

En base a los fotogramas obtenidos en la etapa anterior, estos deberán ser enviados al modelo, para analizar el fondo de la imagen y así capturar aquellos objetos que no pertenezcan al fondo. Para implementar este modelo debemos definir las variables **mascara, fondo, primer plano;** en las cuales se almacenaran los resultados obtenidos después de la sustracción. A continuación, se inicia el modelo de sustracción de fondo MoG, mediante la función ***BackgroundSubtractorMOG2***, la cual recibe como parámetros la matriz de Píxeles del fotograma obtenido para ser procesada, sustrayendo el fondo y obteniendo los objetos que no pertenezcan a fondo. Además también recibe el parámetro de que el fondo que se obtenga en todo el proceso sea fijo o se vaya actualizando constantemente. En este caso, se actualizara, también el MoG cuenta con funciones para eliminar la sombra del objeto en movimiento, que también se habilitará dicha función para que facilite la detección de la persona.

Una vez iniciada la función con dichos parámetros, se envían las variables ***frame*** y ***mascara*** al modelo iniciado, que en base al ***frame*** se devuelve el resultado en la variable ***mascara,*** la cual contiene al objeto en movimiento en la escena en formato binario. Con la función ***getBackgroundImage()***, obtenemos el fondo de la escena, la cual nos servirá posteriormente. Para el análisis de los objetos en movimiento dentro de la escena, se necesita que los datos estén legibles, y en base a las variables ***mascara*** y ***frame***, se realiza una sustracción de los píxeles activos en la máscara dentro del fotograma de video, por lo cual obtenemos una imagen en formato RGB del objeto en movimiento. Para ello utilizamos la función de OpenCV ***frame.copyTo(primer_plano,mascara)***, que hace una operación

lógica binaria ***AND*** entre la variable ***frame, mascara*** y ésta se almacenará en ***primer_plano***.

c) Eliminación de Sombra y Ruido

Después de obtener el ***fondo***, ***mascara*** e imagen ***primer_plano***, que son la base para la comprobación de la sombra en el objeto detectado, continuamos con el segundo paso, que es eliminar el ruido en la imagen de primer plano, y también eliminar la sombra que los objetos en movimiento proyectan. Éste último para tener una detección más precisa de la persona. Existen varios modelos para eliminar la sombra. En este caso, el método que se eligió, está basado en el análisis de la cromaticidad de la imagen.

El análisis de cromaticidad de la imagen, necesita convertir las imágenes de primer plano y el fondo, que normalmente se encuentran en un formato RGB, al modelo de color HSV (*Hue, Saturation, Value*), y así poder analizar por separado cada uno de sus canales. Este análisis se realiza píxel por píxel. Para ello recorremos la matriz de la imagen, donde está el objeto en movimiento ***mascara***, que está en formato binario. La primera comparación que se realiza es que el píxel de la máscara sea “1”, debido a que todo el Píxel “0” se considera como fondo y “1” se considera píxel del objeto en movimiento. Una vez que comprobamos lo dicho, se extraen los datos de las imágenes de fondo y primer plano ya convertidas a HSV, para obtener el valor del canal H, y así obtener la distancia entre ambos valores ***dh=absoluto(valorHPrimerPlano-valorHFondo)***. Al obtener la distancia, realizamos la siguiente comparación: ***min(dh, 180-dh) <= tauH***; donde obtenemos el menor valor de ambos; si este valor es menor al parámetro ***tauH***, se considera un pixel sombra en el canal H, y se procede a realizar la comparación del canal S. De la misma forma obtenemos los valores del canal H del píxel de ambas imágenes, y encontramos la distancia entre ambos (***ds = valorSPrimerPlano - valorSFondo***). Comparamos si **ds <= taus**, donde si se cumple, continuamos con la última comparación de canales, igual que las anteriores comparaciones obtenemos los valores del canal V del píxel y con esto realizamos la comparación de que si ***(valorVPrimerPlano/ valorVFondo) >= alpha AND (valorVPrimerPlano/ valorVFondo) <= beta***, si se cumple esta última condición, se trata de un píxel considerado como sombra del objeto, y en la matriz de píxeles ***mascara*** se cambia el valor de este píxel por “0”.

El método para determinar si un píxel corresponde a una sombra, que anteriormente lo usamos, se basa en la siguiente fórmula:

$$SP_t(x,y) = \begin{cases} 1 \text{ if } \alpha \leq \dfrac{I_t^V}{B_t^V} \geq \beta \\ \wedge \| \ I_t^S(x,y) - B_t^S(x,y) \ \| \leq \tau_S \\ \wedge \| \ I_t^H(x,y) - B_t^H(x,y) \ \| \leq \tau_H \\ 0 \text{ otherwise} \end{cases}$$

Donde:

SP_t : Píxel de sombra

I: Imagen de primer plano

B: fondo

H,S,V: los canales correspondientes a HSV

Esta fórmula se basa simplemente en la determinación de umbrales que definen a las sombras, diferenciándolas de los píxeles que efectivamente corresponden al objeto a segmentar.

Al obtener la máscara, aplicamos filtros para eliminar el ruido. Es decir, eliminar las pequeñas porciones en la imagen que fueron detectadas en la imagen de primer plano. Aplicamos estos filtros, para obtener una imagen, con las regiones más cerradas en la máscara y facilitar encontrar con mayor rapidez los contornos. Para esto usamos los siguientes filtros:

- ***Gaussian blur***. El desenfoque Gaussiano es un filtro que permite suavizar la imagen. En esencia, el efecto mezcla ligeramente los colores de los píxeles que estén vecinos el uno al otro, en un mapa de bits (**Figura 3.7**); haciendo que la imagen pierda algunos detalles minúsculos. De esta forma, hace que la imagen se vea más suave, aunque menos nítida o clara, respecto a que los bordes presentes en la imagen se ven afectados. Se genera un efecto similar al de una fotografía tomada con una cámara fotográfica desenfocada. Para usar este filtro en el algoritmo, utilizamos la función de OpenCV ***medianBlur()*** sobre la máscara.

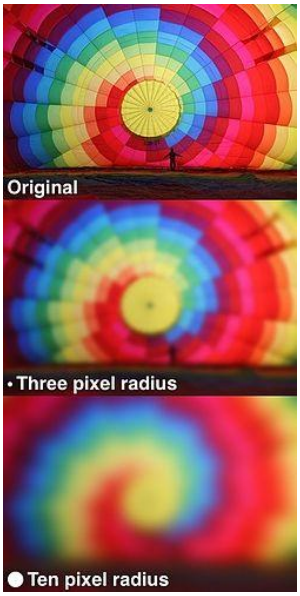


Figura 3.7. Desenfoque Gaussiano

- **Erosión (*Erode).*** Este filtro ensancha y realza las zonas claras de la capa activa o seleccionada. Es decir, para cada píxel de la imagen, se alinea el valor del píxel (luminosidad) con el valor más alto (el más claro) de los 8 píxeles circundantes (matriz 3x3). Así se añade un píxel claro sobre áreas claras; se borrará un píxel aislado en un fondo más claro y un área clara más grande, se dilatará en un píxel en todas las direcciones. Sobre un fondo sólido, este filtro puede suprimir el ruido **(Figura 3.8)**. Con este filtro se pretende mejorar los bordes de la imagen usando la función de OpenCV ***erode()***.

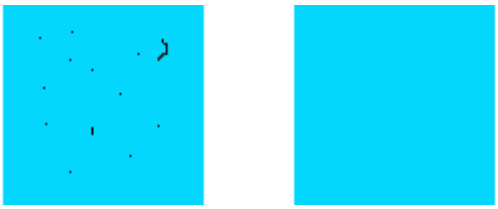


Figura 3.8. Erosión

- **Dilatación (*Dilate).*** Es la operación reciproca de *Erode*. Este filtro amplia y realza las zonas oscuras de la capa activa o seleccionada **(Figura 3.9)**. Es decir, para cada píxel de la imagen, alinea el valor del píxel (luminosidad) con el valor más bajo (el más oscuro), de los 8

píxeles circundantes (matriz 3x3). Así se añade un píxel oscuro en las áreas oscuras, y un píxel aislado en un fondo más claro se cambiara por un “gran píxel”, compuesto por 9 píxeles, y se añadirá ruido a la imagen. En otras palabras lo que se pretende con este filtro, es realzar el contorno del objeto, después de aplicar ***erode()***, y luego usando ***dilate()***. Al obtener la máscara libre de sombra y ruido, se podrán obtener regiones de interés de posibles candidatos, con mayor confiabilidad.

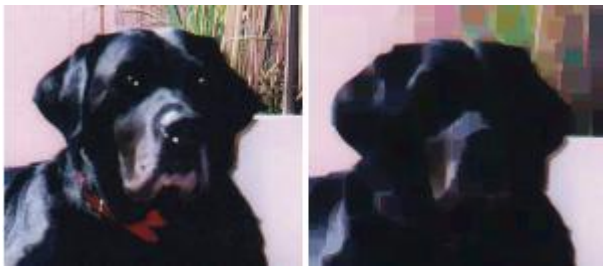


Figura 3.9. Dilatación

d) Regiones de interés (ROI)

Una vez eliminadas las sombras, realizamos el siguiente paso, que es la búsqueda de posibles candidatos (personas), llamadas Regiones de Interés (ROI). En base a la máscara se hace una búsqueda de aquellos contornos que tengan un perímetro mínimo. Esta búsqueda permitirá descartar las regiones pequeñas, ofreciendo una mejora en el proceso de detección, analizando solo aquellas regiones que puedan ser personas. Para la búsqueda, usamos la función ***findContours()*** sobre la ***máscara***; y los valores que obtenemos que son los contornos de la máscara, que se almacenarán.

Al obtener los contornos, se realiza un análisis del tamaño que poseen éstos, para lo cual calculamos la longitud de cada contorno y verificamos que los valores sean mayor a la cota mínima Si esto se cumple incluimos la región del contorno detectado dentro de una lista de posibles candidatos. La longitud del contorno y la superficie de la cota mínima fueron obtenidas en base a pruebas realizadas que se describen en el APÉNDICE B.

Esto permitirá convertir nuestra máscara en una imagen que contenga figuras geométricas del área de interés.

e) Validación de la lista de ROI

Para cumplir con la detección de la persona, se necesita realizar una última validación con la lista de Regiones de interés. Para ello, primero se hace una detección y conversión de bordes de cada candidato, usando la detección de bordes de Canny, que se basa en un algoritmo de múltiples fases para detectar un amplio rango de bordes.

Eso permitirá convertir a nuestros candidatos de la lista, en una imagen que contenga figuras geométricas del área de interés. Es decir, una imagen de bordes. OpenCV incluye la función Canny.

Con la lista de regiones de interés convertida en bordes, se buscan las circunferencias que puedan ser consideradas como la cabeza de una persona, para ello usamos la transformada de Hough, la cual permite detectar las circunferencias que existen en estos bordes, además la búsqueda se basa en un rango de radios, y con esto se filtra la lista de regiones de interés que no cumplan con el rango, y se eliminan de la lista. Finalmente la lista de regiones de interés contendrá a la persona detectada.

Para comprender gráficamente al algoritmo, mostramos una secuencia de imágenes, del funcionamiento del mismo. En la **Figura 3.10** se ve la Sustracción de fondo, con la máscara aplicada. En la **Figura 3.11** observamos la Región de interés, donde se ve a la persona detectada. En ambas figuras se puede observar que las dos primeras imágenes están vacías, esto es porque la persona aún no se encuentra dentro de la escena.

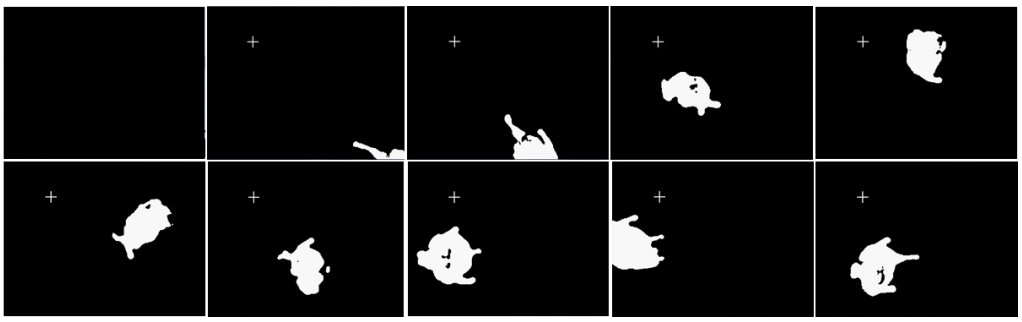


Figura 3.10. Máscara Algoritmo 1

Figura 3.11. ROI Algoritmo 1

3.3.2 Algoritmo 2

El algoritmo 2, al igual que el algoritmo 1, está desarrollado en C++, con la librería OpenCV, y además utilizamos la librería OpenNI, debido que este algoritmo está planteado para ser aplicado a Kinect. El algoritmo se basara básicamente en el análisis de la cámara de profundidad que tiene Kinect. Como se menciono anteriormente, el algoritmo 2, tiene procesos idénticos (**Figura 3.12**) al algoritmo 1; y es por este motivo solo se describirán los procesos diferentes del algoritmo 1.

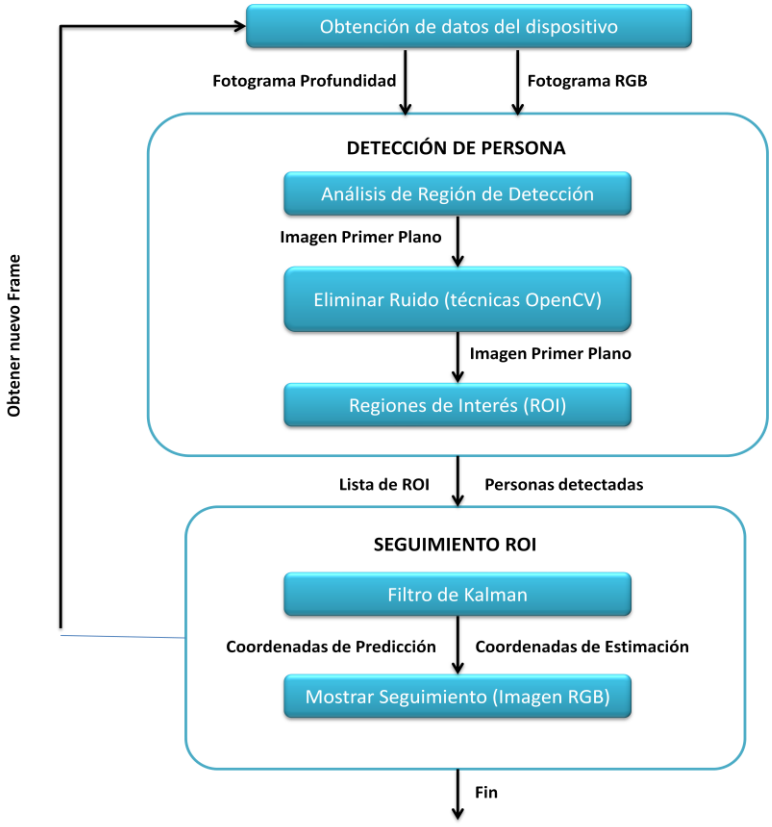


Figura 3.12. Diagrama de Procesos Algoritmo 2

a) Obtención de Datos en tiempo real

Lo primero que realizamos con este algoritmo es obtener los fotogramas de video de la cámara de profundidad "*Camera Depth Data*" de Kinect. Igual que el algoritmo 1 hacemos uso de la función ***VideoCapture()***. Este algoritmo puede capturar la información en tiempo real o Videos, previamente capturados con la cámara de profundidad que está en formato "*.oni".* Para capturar la información en tiempo real, se deberá usar OpenNI, es decir que debemos integrar ambas librerías, OpenCV con OpenNI. (APÉNDICE A.3)

Tenemos al dispositivo funcionando, y debemos almacenar la información en un formato que OpenCV pueda leer. Usamos dos variables ***Mat,*** una llamada ***depthMap***, la cual contendrá al fotograma de la cámara de profundidad y la otra ***bgrImage*** que contendrá al fotograma de la cámara RGB; esta última variable, la usamos solo para mostrar las áreas detectadas por el algoritmo. Pero, los procesos se hacen sobre la información de profundidad.

b) Análisis de Región de Detección (Sustracción de Fondo)

Una vez que ya obtenemos la información de profundidad de la cámara, realizamos el análisis de región de detección, similar a la sustracción de fondo. Se establece un rango de la distancia de cuan cerca y lejos se encuentra el objeto con respecto a la cámara de profundidad; esto permitirá establecer una región donde se encuentren los hombros y la cabeza de una persona, y descartar el resto de la escena. Creamos dos imágenes de umbral, las cuales se basan en los puntos de profundidad. La primera imagen será la cercanía en la que se encuentra el objeto a detectar con la cámara, usamos un parámetro llamado ***threshNear****;* y la segunda imagen es lo contraria. Es decir, cuán lejos está el objeto a detectar de la cámara, usamos el parámetro ***threshNear.*** Los datos de estos parámetros se obtuvieron previamente al calibrar la cámara de profundidad, basándonos en la altura a la cual se encuentra la cámara, y la altura de las personas, además considerando solamente los hombros y la cabeza. (APÉNDICE C)

Al obtener las imágenes de umbral, y unirlas, ya podemos establecer una imagen de primer plano, la cual contendrá a todo objeto que se mueva en el rango de área de profundidad teniendo en cuenta la distancia entre la cámara y la persona a detectar.

El resto del algoritmo es igual que el algoritmo 1. Es decir, que continuamos eliminando el ruido de la imagen de primer plano, usando los filtros mencionados en el paso **c)** del algoritmo 1. Para comprender gráficamente cómo funciona el algoritmo 2, en la **Figura 3.13**, observamos la información de profundidad, obtenida por el dispositivo, y en la **Figura 3.14**, la regiones de interés.

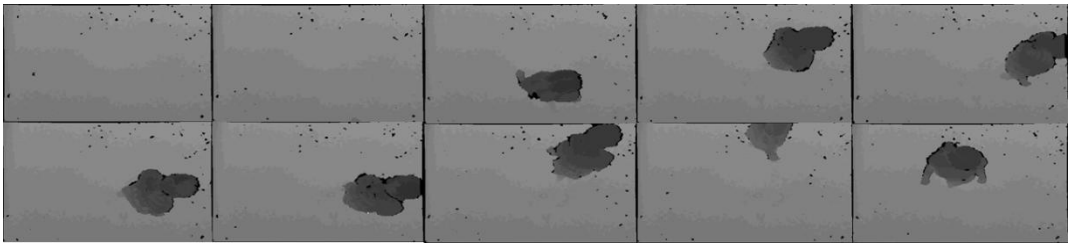


Figura 3.13. Información de Profundidad (Algoritmo 2)

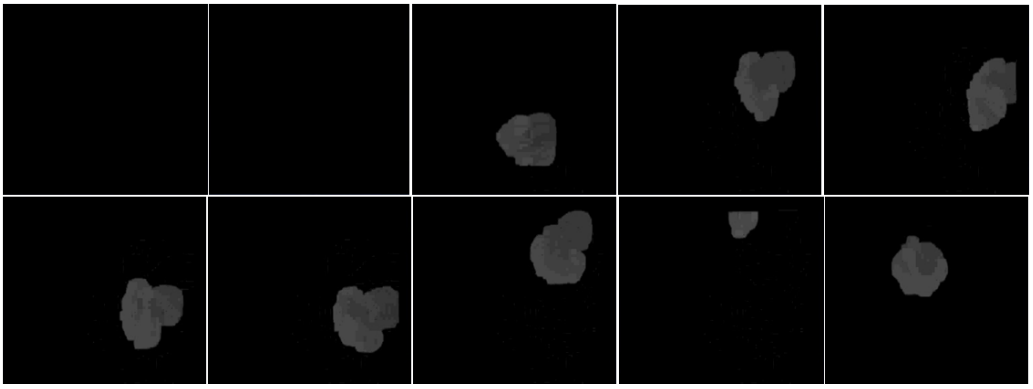


Figura 3.14. Regiones de Interés (Algoritmo 2)

3.3.3 Algoritmo 3

El desarrollo del algoritmo 3 es exclusivamente para Kinect, y está desarrollado con C++. A diferencia de los demás algoritmos, solo usamos la librería OpenNI. También usamos OpenGL para mostrar el seguimiento de la persona. OpenNI cuenta con un *middleware* para el análisis de cuerpo completo, el cual permite identificar cuerpos humanos en la escena (identificación de puntos específicos del cuerpo como la cabeza, el cuello, los hombros, etc.), y dentro de este *middleware*, también se cuenta con el seguimiento de la persona reconocida.

Para el algoritmo, utilizamos una aplicación de código abierto que provee OpenNI llamada *NiUserTracker*, cuyo objetivo es realizar el seguimiento de una persona en 3D. Las aplicaciones de código abierto de OpenNI, están desarrolladas para capturar a la persona u objeto de frente, y no así desde una

visión perpendicular. A continuación se da una explicación de los procesos del Algoritmo 3, con su respectivo Diagrama (**Figura 3.15**).

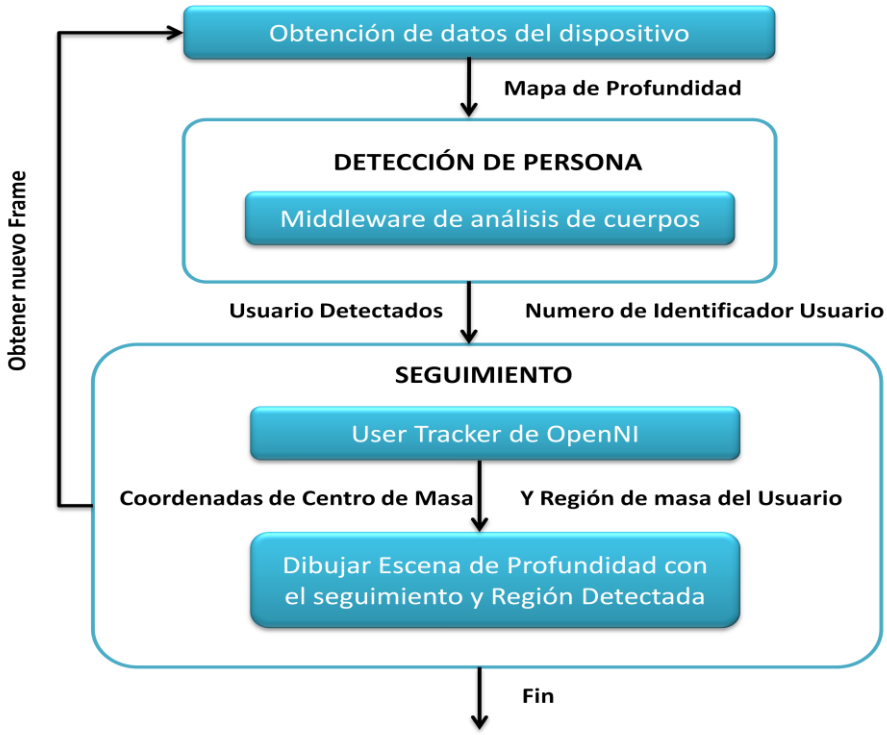


Figura 3.15. Diagrama de Procesos Algoritmo 3

a) Obtención de Datos en tiempo real

Para la obtención de información de profundidad de la escena, la librería OpenNI brinda una función para usar con el dispositivo Kinect en tiempo real o leer un video capturado por Kinect, en formato ".*oni*". Esta función es ***InitFromXmlFile()***, la cual permite conectarse con el dispositivo.

OpenNI trabaja toda su estructura en base a nodos de producción, que son conjuntos de componentes, que tienen un rol productivo en el proceso de creación de datos. Cada nodo de producción encapsula la funcionalidad que se relaciona con la generación de tipo específico de dato, y podrá proveer este dato a la aplicación u otro nodo de producción.

Una vez iniciada Kinect, continuamos obteniendo el fotograma de profundidad, para lo cual se define un nodo donde contendrá el mapa de profundidad, a este nodo lo llamamos ***g_DepthGenerator***, de esta forma ya podemos extraer el mapa de profundidad utilizando la función

g_Context.FindExistingNode(XN_NODE_TYPE_DEPTH,g_DepthGenerator), almacenando el mapa de profundidad en el nodo ***DepthGenerator***.

b) Detección del cuerpo de una persona

OpenNI como ya se mencionó con anterioridad, tiene un *middleware* para analizar en la escena la existencia de un cuerpo humano, el cual pusimos a prueba debido a que las librerías fueron desarrolladas para que Kinect se encuentre de frente a la persona, y en nuestro caso Kinect está ubicada en el techo, de forma cenital. Realizando las pruebas conseguimos que logre detectar la cabeza y hombros de una persona. Para realizar la detección con OpenNI, primero se debe crear un nodo llamado ***g_UserGenerator***, el cual contendrá las características de la persona detectada, como son tamaño de masa, coordenadas del punto central de la masa, etc.

Para utilizar el middleware, utilizamos la siguiente función ***g_Context.FindExistingNode(XN_NODE_TYPE_USER,g_UserGenerator)***. La detección de la persona se basará en el mapa de profundidad ***DepthGenerator***, que se lee constantemente y se analiza para generar como resultado, los datos acerca de la persona en la escena. Este nodo está en espera de detectar un cuerpo humano, cuando se detecta se le asigna un identificador numérico único y lo considerará a partir de ese momento como un usuario. Con ello ya tenemos la persona detectada en la escena.

c) Seguimiento de la persona

Al no usar OpenCV en este algoritmo, para realizar el seguimiento de la persona usamos funciones de seguimiento, incorporadas dentro de OpenNI. Después de detectar al usuario utilizaremos el mismo nodo ***g_UserGenerator*** para realizar el seguimiento.

Se debe tener en cuenta que esta aplicación reconoce diferentes usuarios, entonces cuando el mapa de profundidad $\boldsymbol{M_t}$, el nodo ***g_UserGenerator*** tiene detectados ***n*** usuarios; al recibir el siguiente mapa de profundidad $\boldsymbol{M_{t+1}}$, se buscará la posición de cada uno de los ***n*** usuarios en la escena. Para cada usuario $\boldsymbol{U_i}$ presente en $\boldsymbol{M_t}$ se realiza una búsqueda en $\boldsymbol{M_{t+1}}$ para encontrar cuál de los objetos (personas) presentes en el nuevo mapa corresponde a $\boldsymbol{U_i}$. Es posible que del instante ***t*** al instante ***t+1***, el usuario $\boldsymbol{U_i}$ haya realizado algún movimiento cambiando su posición en la escena (por ejemplo, moverse a la

izquierda o derecha, agacharse, levantar brazos, etc.). En la **Figura 3.16** se muestra un ejemplo ilustrado del seguimiento.

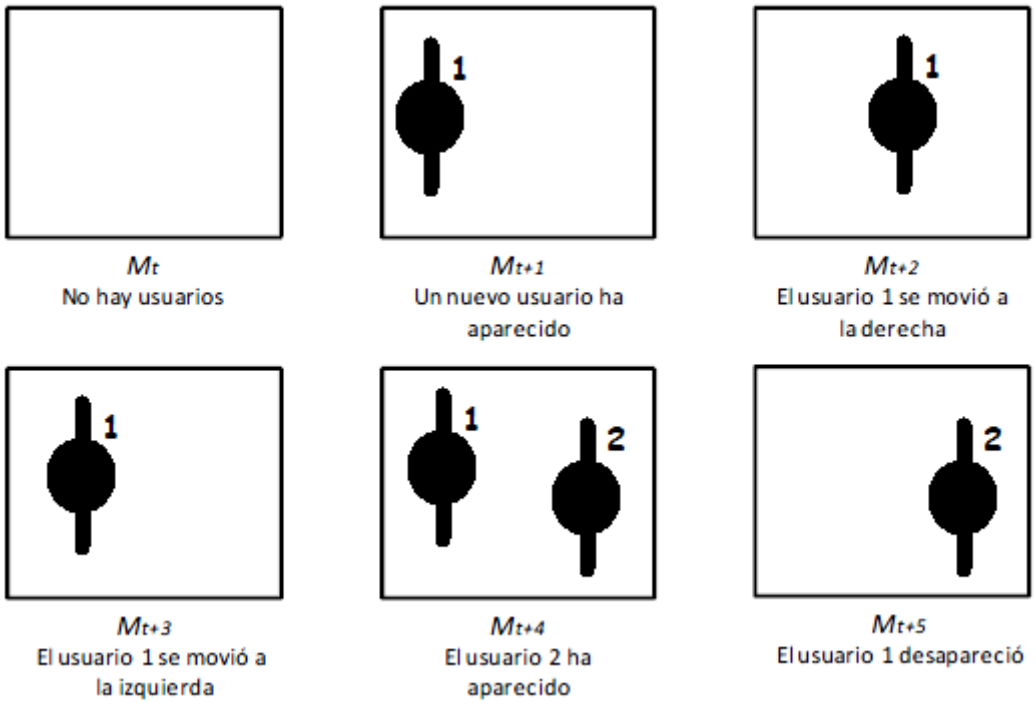


Figura 3.16. Seguimiento de la persona realizado por el nodo g_UserGenerator

Para utilizar el seguimiento, primero debemos registrar el identificador del usuario detectado. Para ello utilizamos la siguiente función del nodo ***g_UserGenerator.RegisterUserCallbacks(User_NewUser, User_LostUser, NULL, hUserCallBacks)***. Después, debemos iniciar la calibración de este usuario en la escena en base al centro de masa de mismo, con la función ***g_UserGenerator.GetSkeletonCap().RegisterToCalibrationStart(UserCalibration_CalibrationStart, NULL, hCalibrationStart)***. Si se completo esta calibración pasamos a hacer el seguimiento de esta persona utilizando la función ***g_UserGenerator.GetSkeletonCap().RegisterToCalibrationComplete(UserCalibration_CalibrationComplete, NULL, hCalibrationComplete).***

Una vez completado todo el análisis con el nodo ***g_UserGenerator*** obtendremos coordenadas del centro de masa del usuario, al cual se le está haciendo el seguimiento. Usando funciones de OpenGL, dibujamos el mapa de profundidad y dentro del mismo reflejamos las coordenadas del seguimiento mediante una línea de color y la región de la masa que ocupa en la escena.

En la **Figura 3.17**, observamos cómo la persona entra en la escena y la persona es detectada (con el color azul), y el seguimiento que se realiza de la persona detectada.

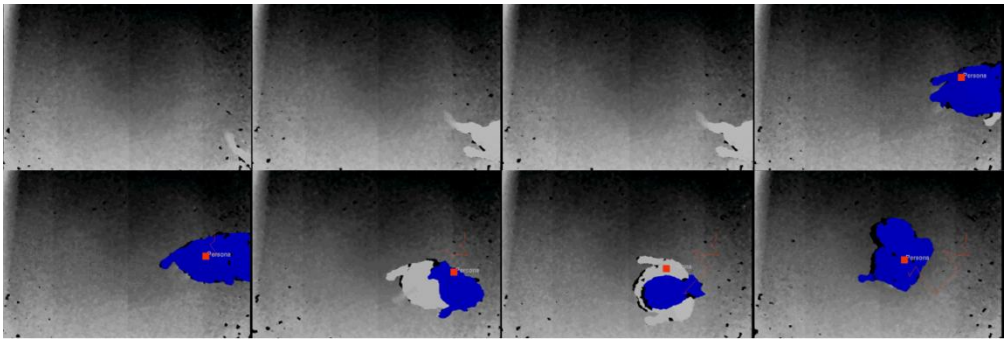


Figura 3.17. Detección y Seguimiento de la persona (Algoritmo 3)

3.3.4 Algoritmo 4

Este Algoritmo, también está desarrollado con C++ y la librería OpenCV, y esta desarrollado exclusivamente para la Web Cam Logitech 9000, como el algoritmo 1. Como se mencionó con anterioridad, los procesos de desarrollo son los mismos del primer algoritmo, con la diferencia que en este algoritmo, utilizamos un identificador dentro de la escena para que la detección se base en este identificador. Este proceso lo llevaremos a cabo utilizando un análisis de rangos de colores del formato HSV (*Hue, Saturatio Value*); basándonos en los colores de dicho identificador. Usamos este tipo de formato de imagen, debido a que HSV, permite obtener el color del objeto con mayor precisión, obteniendo colores puros, obviando la iluminación y brillo que se genere en el identificador.

En este caso el identificador es un “Gorro” de color Granate (guindo), que la persona lleva en su cabeza, de esta forma se realizará la detección de la persona. Los procesos de este algoritmo podemos observarlos en la **Figura 3.18**, ya que tiene un gran parecido con el del algoritmo 1, pasamos directamente a explicar los procesos diferentes.

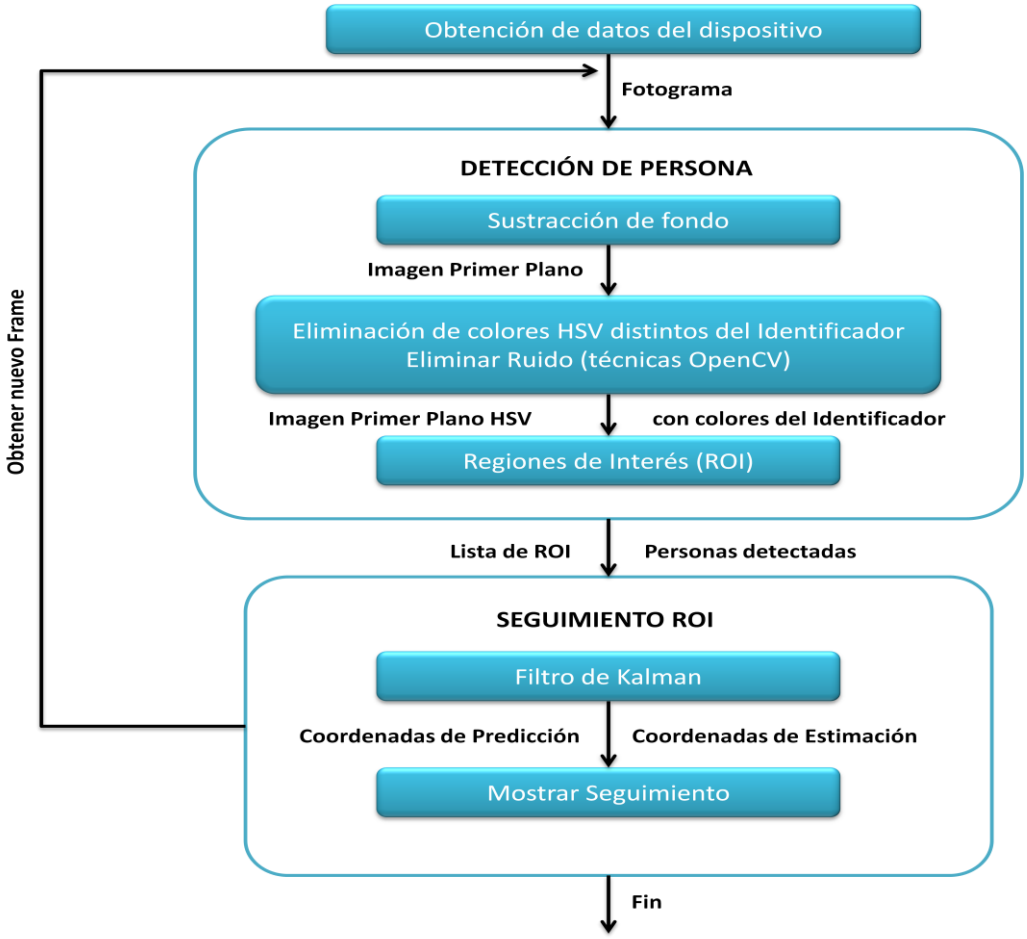


Figura 3.18. Diagrama de Procesos Algoritmo 4

d) Filtrado de colores HSV del Identificador y Eliminación de Ruido

Este proceso fue modificado con respecto al primer algoritmo para el reconocimiento del identificador. En base al ***primer_plano*** obtenido del proceso anterior, se inicia el filtro de colores HSV, para lo cual, primero se debe convertir dicha imagen a este formato; usando la función de OpenCV ***cvtColor()***. Al tener la imagen en el formato HSV, se obtiene el rango de colores HSV, que el identificador contiene; utilizamos un valor mínimo y máximo para los canales H, S y V; para establecer el rango se realizó un análisis de los colores HSV del identificador en la escena, además del tamaño que tiene este identificador dentro de la escena, mencionado en el APÉNDICE D.

OpenCV ofrece una función llamada ***inRange(),*** la cual se basa en encontrar determinados colores, en función de un rango específico y lo almacena en una

variable ***primer_plano_hsv,*** con lo cual filtramos los objetos en movimiento que no poseen los colores del identificador.

En cuanto a la eliminación de ruido se utiliza las mismas funciones que el primer algoritmo, pero incorporando un filtro más (***Gradiente Morfológico***), debido a que estamos trabajando con una imagen en formato HSV. La función ***Gradiente Morfológico***, básicamente realiza una diferencia entre la ***dilatación*** y ***erosión***, permitiendo encontrar el contorno de un objeto (**Figura 3.19)**. Con este filtro, además de que permite encontrar los contornos, también permite convertir la imagen a binaria.

Figura 3.19. Contorno de un objeto

e) Regiones de Interés (ROI)

Para obtener las regiones de interés, se realiza de la misma forma que el primer algoritmo, pero basándonos en la variable ***primer_plano_hsv,*** y utilizando el tamaño que el identificador tiene dentro de la escena. Con esto ya obtenemos la lista ROI de las personas detectadas y continuamos con el seguimiento que también se realiza con Kalman, como en los dos primeros algoritmos.

En el caso de este algoritmo, en la interfaz se tienen 4 ventanas, en la **Figura 20**, se muestra la sustracción de fondo, en la **Figura 3.21** el filtro HSV que se aplica, para detectar al identificador, y finalmente en la **Figura 3.22**, las regiones de interés.

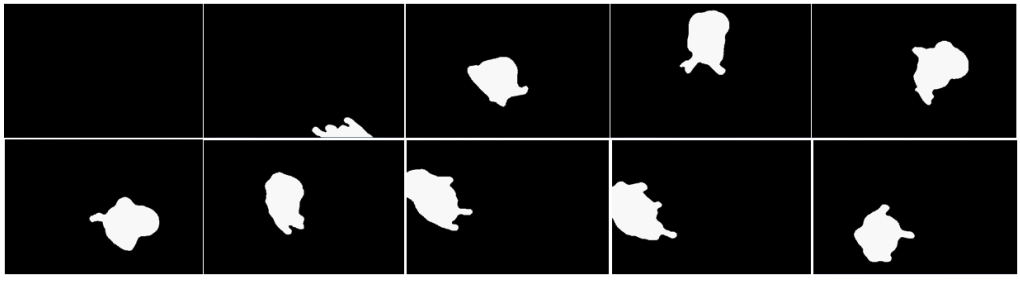


Figura 3.20. Sustracción de fondo (Algoritmo 4).

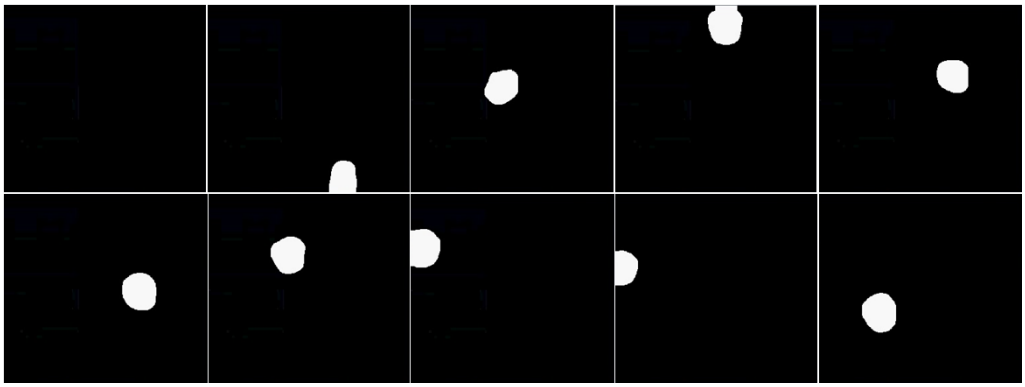


Figura 3.21. Sustracción de fondo, aplicando el filtro HSV (Algoritmo 4).

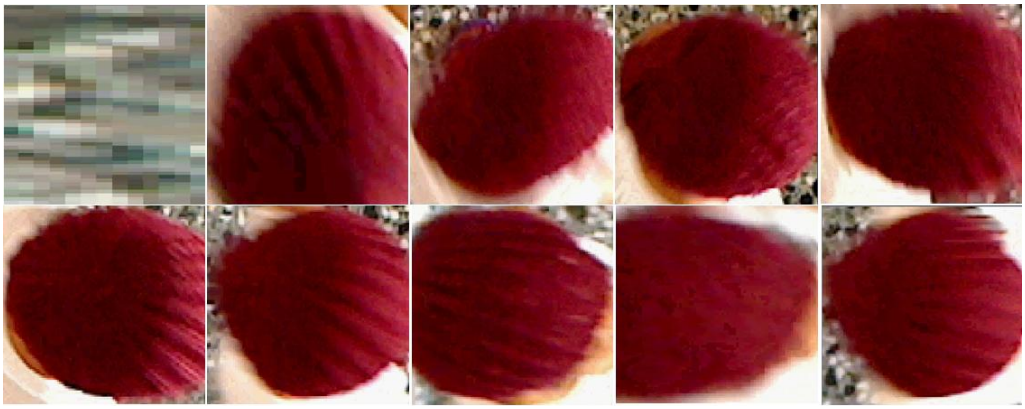


Figura 3.22. Regiones de interés (Algoritmo 4).

3.3.5 Seguimiento

Para realizar el seguimiento de la persona usamos el filtro de Kalman, el cual es un algoritmo recursivo que se utiliza para estimar la posición de un punto o característica en movimiento y la incertidumbre de la medida en una imagen posterior. El algoritmo consiste en buscar una característica (punto, borde, esquina, región, etcétera), en un área determinada de la imagen posterior, alrededor de la posición prevista, en la que estamos seguros de encontrar la característica dentro de un cierto grado de confianza.

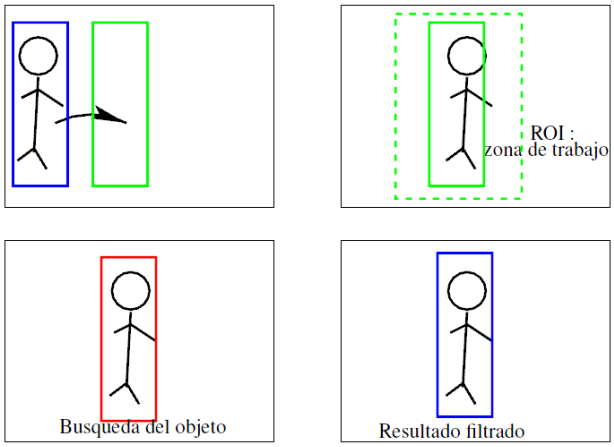


Figura 3.23. Seguimiento con el Filtro de Kalman.

Se inician los estimadores de estado de posición, después se envía la región en la cual se detectó a la persona, de esta región obtenemos el centro del contorno y el tamaño, la cual se enviará a la predicción a posteriori de Kalman. En base a esta predicción, se van almacenando todas las posiciones que se van retroalimentando y se hace una corrección para poder tener una mejor predicción a posteriori. La descripción del filtro de Kalman, con sus respectivas ecuaciones, podemos observarla en el diagrama de la **Figura 3.24**.

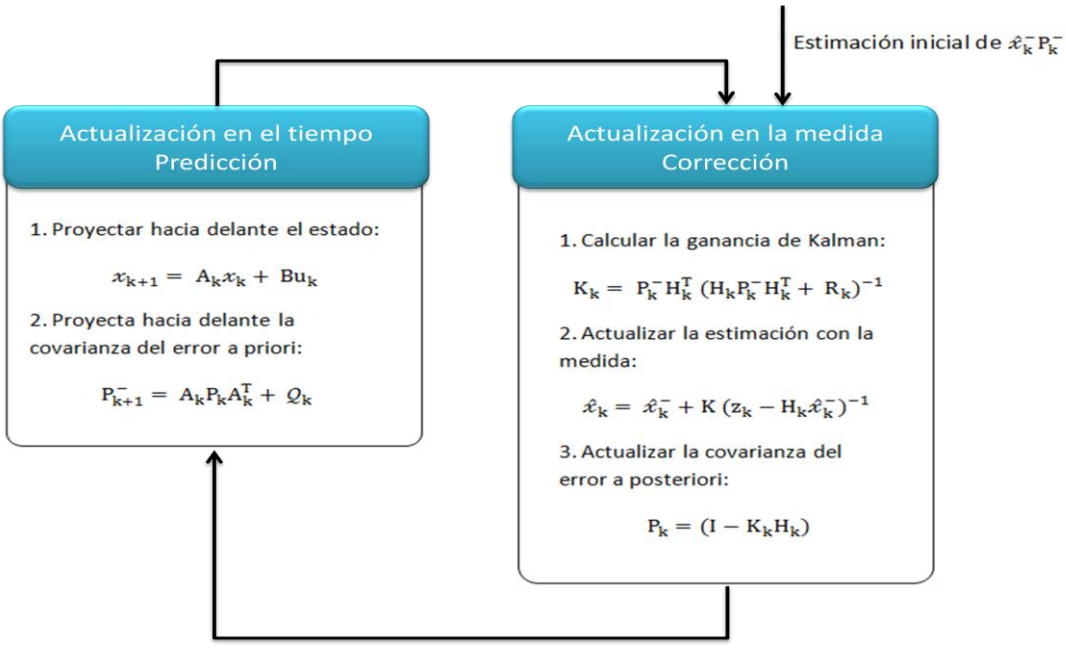


Figura 3.24. Filtro de Kalman

a) Inicialización del Filtro de Kalman

Para usar el filtro de Kalman, proporcionado por OpenCV, lo primero que debemos realizar es iniciarlo, para esto usamos ***KalmanFilter KF(4,2,0)***, donde 4 es la dimensión del estado, 2 la dimensión de la medición, y 0 es la dimensión del vector de control que no usamos. También se deben definir las matrices donde se almacenarán el estado (posición y velocidad), el proceso de ruido (covarianza) y las mediciones de la escena actual, ***Mat_<float> state***, ***Mat processNoise***, ***Mat_<float> measurement***; ya que se trabajará en un plano de dos dimensiones iniciamos las coordenadas del estado dentro del ***KF***, ***KF.statePre.at<float>(0) = 0; (Coordenada x)***, ***KF.statePre.at<float>(1) = 0; (Coordenada y)***, ***KF.statePre.at<float>(2) = 0***, ***KF.statePre.at<float>(3) = 0***.

También se debe iniciar la matriz de transición inicial y de covarianza, por defecto tomamos al azar estas matrices, y se definen de la siguiente manera:

KF.transitionMatrix = *(Mat_ <float>(4,4) << 1,0,1,0 0,1,0,1 0,0,1,0 0,0,0,1);

KF.processNoiseCov = *(Mat_ <float>(4,4) << 1,0,1,0 0,1,0,1 0,0,1,0 0,0,0,1);

b) Tratamiento de la Lista de Regiones de Interés

Una vez iniciado Kalman, continuamos con el proceso iterativo de *frames* de la escena; además de utilizar la lista de regiones que contiene a las personas detectadas ***contours***. Dentro del proceso iterativo, que contiene a la persona, antes de ser analizado por el filtro de Kalman, se necesita conocer el punto central de la región de la persona, para ello, primero se debe convertir la región a un conjunto de puntos o píxeles, a este proceso se conoce como “Rasterizar la imagen”. OpenCV tiene una función que permite rasterizar la región, además que facilita encontrar extremos y centros de la región. Éstos se almacenan, y se procede a encontrar el punto central de la región, para que el filtro de Kalman los pueda leer.

c) Actualización del Modelo y Seguimiento de la Persona

En la etapa anterior, obtenemos el punto central de la región detectada de la persona, lo siguiente que se realiza es una predicción y actualizar la variable del estado de predicción y las covarianzas, para saber dónde debería ir el punto en la siguiente imagen. En la primera iteración, como los valores fueron

introducidos al azar son irrelevantes. Pero a posteriori, los puntos introducidos serán reales de la dirección que está siguiendo la persona. Para dicho proceso, primero debemos realizar la predicción con la función ***predict()***, la cual obtiene una matriz con coordenadas y se almacena. Con esta matriz actualizamos los puntos de predicción. Envíamos el punto central de la persona a una variable de medición (***measurement***), y se actualizará el punto de medición del filtro de Kalman. Se deberá realizar una corrección en el filtro y así obtener la estimación. Una vez obtenida la estimación, realizamos la actualización de la covarianza, realizando este proceso recursivamente hasta que el seguimiento termine. En la pantalla de seguimiento, de cada algoritmo, se muestra con una línea de color celeste el punto central o medición de la persona detectada, y la línea de color verde es la estimación del seguimiento que está realizando el filtro de Kalman (**Figura 3.25)**.

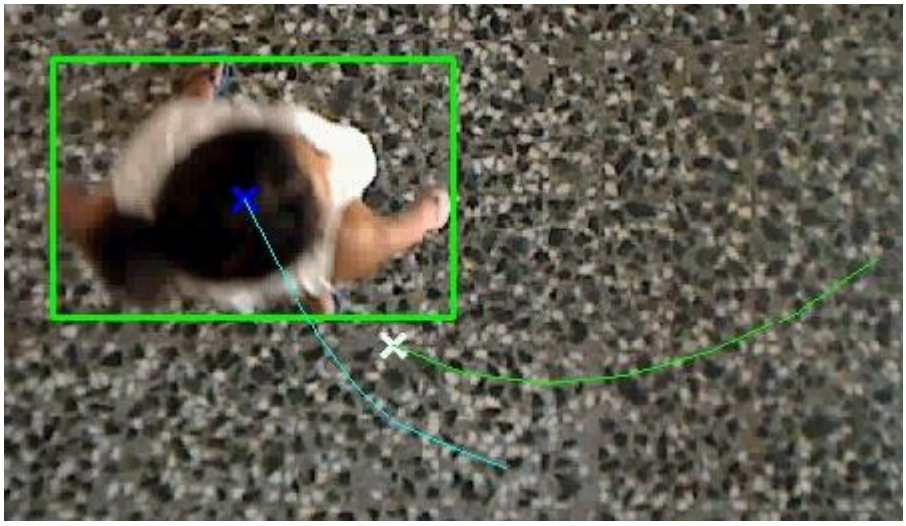


Figura 3.25. Seguimiento Algoritmos 1, 2 y 4.

El algoritmo realiza una corrección en base a la predicción y la medición, de tal forma que se obtenga la estimación, y con esto valor actualicemos la covarianza de error a posteriori. Finalmente el filtro de Kalman, concluye, y en pantalla se puede observar el trayecto que realiza la persona y el seguimiento en la escena.

Para el seguimiento realizado, después de la explicación anterior, podemos observar gráficamente, la detección y el seguimiento, para el Algoritmo 1 (**Figura 3.26**), Algoritmo 2 (**Figura 3.27**) y Algoritmo 4 (**Figura 3.28**).

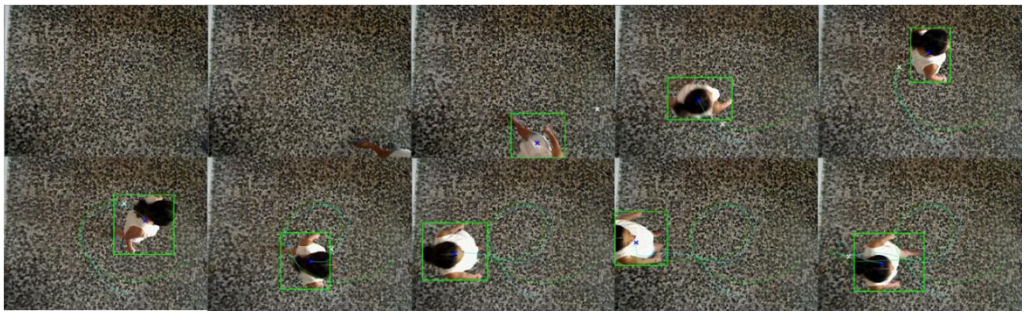


Figura 3.26. Detección y Seguimiento Algoritmo 1

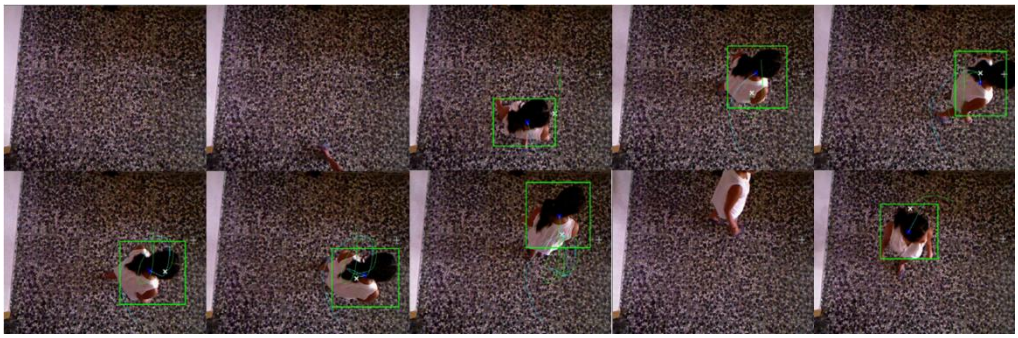


Figura 3.27. Detección y Seguimiento Algoritmo 2

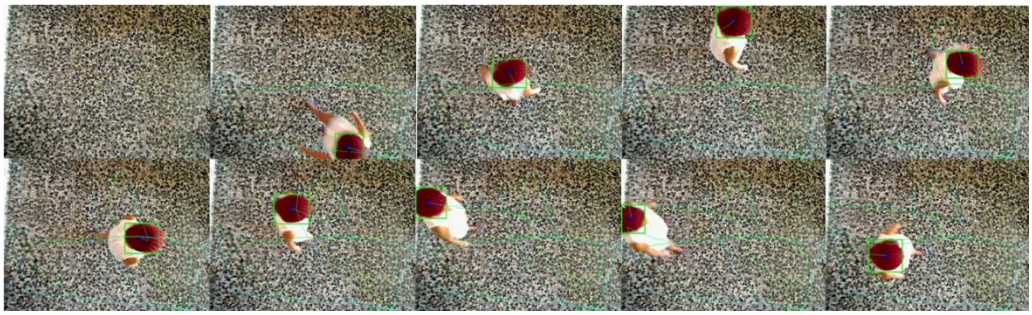


Figura 3.28. Detección y Seguimiento Algoritmo 4

CAPÍTULO IV

RESULTADOS

4.1 Introducción

Este capítulo está dedicado a la evaluación de los algoritmos propuestos en el capítulo 3. A continuación explicaremos las unidades de medida que vamos a emplear para medir la calidad de nuestros resultados en los experimentos. También explicaremos el método de análisis que usamos para los experimentos. Finalmente, una vez descritos los detalles necesarios para comprender la evaluación, se realizarán las diferentes pruebas (experimentos) con cada algoritmo propuesto.

4.2 Unidades de medida

Para medir la calidad de los resultados experimentales vamos a usar, básicamente el ancho y alto de la información obtenida, que en este caso son las imágenes capturadas por los dispositivos, es decir Alto y Ancho, y ambos estarán medidos en centímetros (cm.). La escena tiene un ancho de 300 centímetros, y un alto de 228 centímetros.

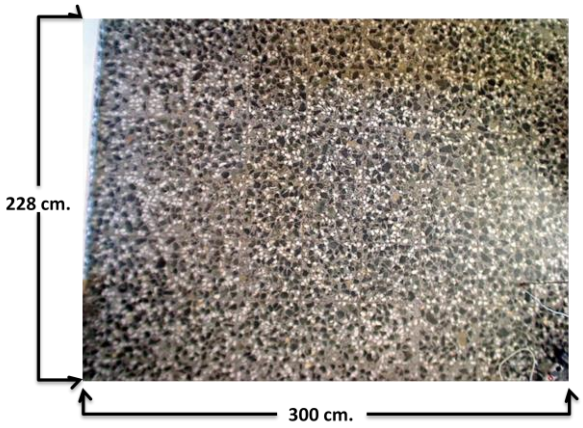


Figura 4.1. Medidas de la Escena (Centímetros)

Significa que los valores obtenidos en el análisis que se menciona posteriormente, tienen como unidad de medida los centímetros.

4.3 Método de análisis

Para analizar los resultados y saber qué algoritmo es el mejor, debemos comparar cada algoritmo, observar los datos que cada algoritmo captura y compararlos entre sí y con la trayectoria real. El método de análisis que usamos para los resultados es el "Área de una curva Trapezoidal", que consiste en dividir la curva en una serie de trapecios, cada uno con área = (altura media) (ancho), y a continuación realizar la suma de las áreas de los pequeños trapecios. Como fórmula para obtener el Área de los pequeños trozos, tenemos:

$$A_{Ti} = \left(\frac{Y_{i+1} + Y_i}{2}\right) * (X_{i+1} - X_i)$$

Donde:

A_{Ti}: Área del Trapecio

X_i: Ancho (Píxeles)

Y_i: Alto (Píxeles)

Para obtener el área de cada curva debemos sumar $A_{T1}, A_{T2}, A_{T3}, \dots, A_{Tn}$, teniendo:

$$A_{Curva} = \sum_{i}^{n} A_{Ti}$$

En nuestro caso tendremos 2 curvas por cada prueba (**Figura 4.2**) observamos el Área de una curva Trapezoidal**)** la curva de la trayectoria Original $A_{T_{Original}}$, y la curva obtenida por una de las pruebas de los algoritmos. Es por eso que también debemos aplicar las formulas a $A_{T_{Original}}$:

$$A_{T_{Original}} = \sum_{i=1}^{n-1} \left(\frac{Y_{i+1} + Y_i}{2}\right) * (X_{i+1} - X_i)$$

Esta fórmula también la usamos para obtener el área de la curva a comparar, con la original, es decir el área de la curva de prueba.

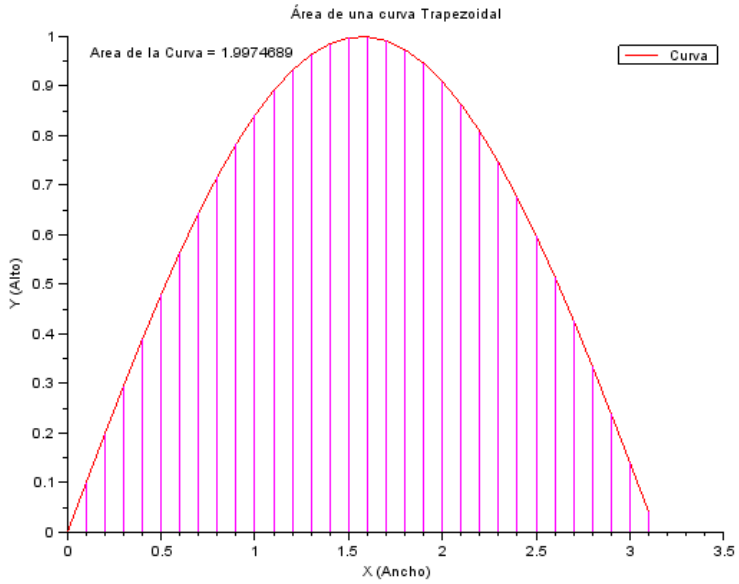


Figura 4.2. Área de una curva Trapezoidal (Centímetros).

Finalmente para obtener el A_R, Área entre las dos curvas (Trayectoria Original y la prueba), debemos aplicar lo siguiente:

$$A_R = abs(A_{T_{Original}} - A_{Curva})$$

En otras palabras, se obtienen dos curvas por prueba, una curva del trayecto original, y otra de los datos obtenidos por los algoritmos. Obtenemos el área de cada curva, y de esta forma, obtenemos el área que existe entre las curvas, de esta forma obtendremos la diferencia que existe entre la trayectoria original, y la trayectoria obtenida por los algoritmos.

Calculamos el Área media (AM):

$$AM = \frac{\sum_i^N A_{Ri}}{N}$$

Donde:

A_{Ri}: Área Resultante entre ambas curvas

N: Número de Muestras

Además usamos el Análisis de la varianza de un factor (ANOVA). Este análisis es una colección de modelos estadísticos y sus procedimientos asociados, en el cual la varianza está particionada en ciertos componentes debidos a diferentes variables explicativas. Este análisis nos permitirá conocer si existen diferencias estadísticas

significativas en los experimentos, si $p < 0.05$, significa que existe una diferencia estadística significativa.

El algoritmo que tenga el área media (AM) menor en comparación con el resto, será el que se aproxima más a la trayectoria real, por lo tanto será el algoritmo elegido. Para realizar los cálculos mencionados y hacer los gráficos, usamos el programa matemático Scilab conjuntamente con Excel.

4.4 Experimentos

Para realizar los experimentos se realizarán diferentes pruebas, es decir al ser algoritmos para la detección y seguimiento de personas en una escena, lo que se realizará básicamente es trazarse 3 trayectorias diferentes dentro de la escena, de tal forma que obtengamos los resultados de estas 3 trayectorias y también se conozca la trayectoria real. Se experimentarán las 3 trayectorias, con cada uno de los algoritmos y en tiempo real.

Se realizarán dos experimentos, y por cada experimento se obtendrán resultados para cada trayectoria. El primer experimento está enfocado para los algoritmos 1 y 2, y el segundo experimento se realizará con los algoritmos 1, 2 y 4. Se realizan estos experimentos de esta forma, ya que en el primer experimento la persona no llevará ningún tipo de identificador para ser detectada, y el Algoritmo 4, no funcionará en este experimento, ya que éste necesita un identificador. En el segundo experimento, la persona llevará el identificador consigo mismo, en este caso el Algoritmo 4, si será evaluado.

Para cada trayectoria, se repite el recorrido 10 veces con 2 personas diferentes, obteniendo un total de 20 resultados por trayectoria. En los resultados que obtenemos por cada recorrido tenemos las coordenadas del movimiento que realiza la persona.

El algoritmo 3 no se ha incluido en los experimentos. Dicho algoritmo presenta problemas de dos tipos. Por un lado, la iluminación existente en la escena, y por otro lado el movimiento de la persona al entrar en la escena.

En relación con los problemas de iluminación, se observa que cuando existen cambios de iluminación en la escena, el algoritmo no detecta correctamente a la persona, o el seguimiento es incorrecto, especialmente, si la escena es demasiado oscura.

En relación al segundo problema identificado, en varias ocasiones la persona debe realizar gestos llamativos en la escena para ser detectada. Es decir, la persona debe poner los brazos a los costados y levantarlos, como un gesto, o en otros casos, la

persona debe mover las manos, similar al movimiento que hacemos al saludar con la mano, pero por un periodo relativamente largo. Este problema se debe a que el *middleware* incluido en OpenNI, está desarrollado para capturar a la persona de frente, es decir que el sensor Kinect, debe estar frente a la persona. El *middleware* necesita un gesto conocido (mover la cabeza, mover las manos, etc.) para detectar a la persona. En nuestro caso, el sensor Kinect, se encuentra situado de forma cenital, y tiene una visión de la persona de forma perpendicular, y además la persona no realiza ningún gesto, ya que el resto de los algoritmos, no necesitan capturar un gesto para detectar a la persona.

La forma en la que está situada la Kinect, es el principal problema para que este algoritmo, no funcione correctamente. En la **Figura 4.3**, podemos observar la secuencia de imágenes, de detección y seguimiento que se realiza a la persona en la escena. Cuando la persona entra en la escena no es detectada, pasan 3 segundos para ser detectada, y además podemos ver que el centro de masa no se encuentra en el centro, pero el seguimiento se ha dibujado, y a los 10 segundos, la persona deja de ser detectada.

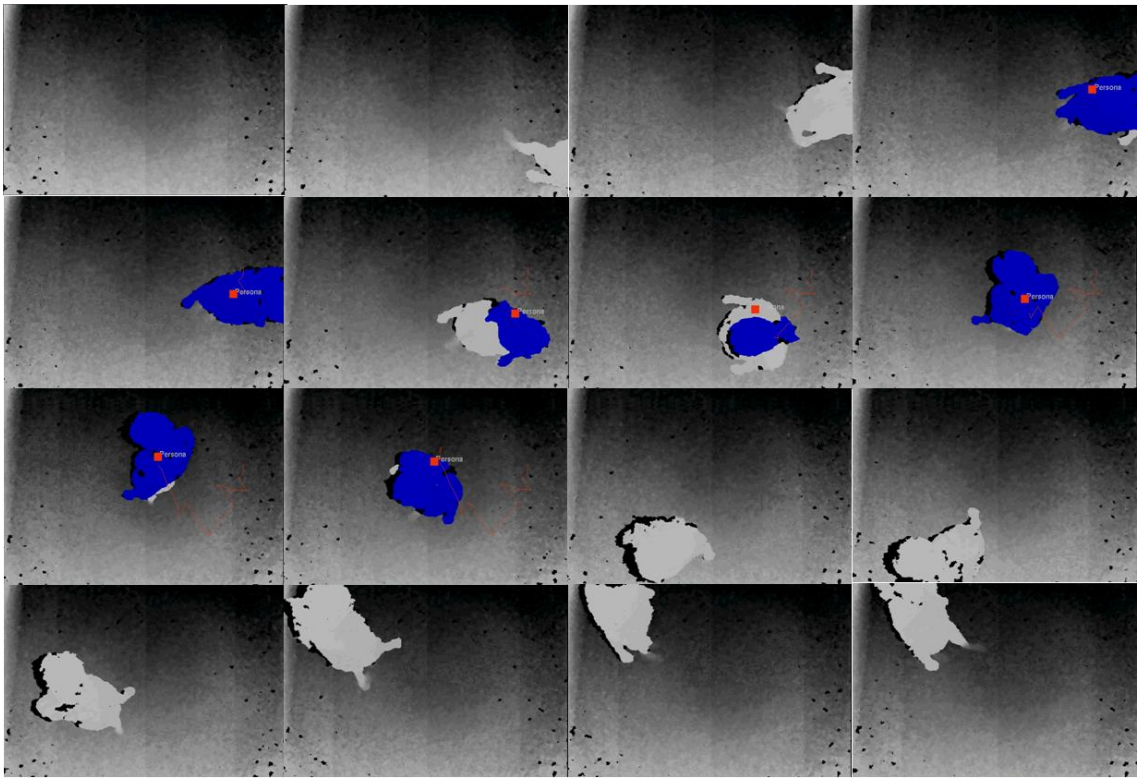


Figura 4.3. Detección y Seguimiento del Algoritmo 3.

En la **Figura 4.4**, observamos la detección de una persona en la misma escena, pero se observa que la persona hace gestos con los brazos para que sea detectada por el sensor, y en medio de la escena. Además de que en esta secuencia de imágenes, no existe seguimiento.

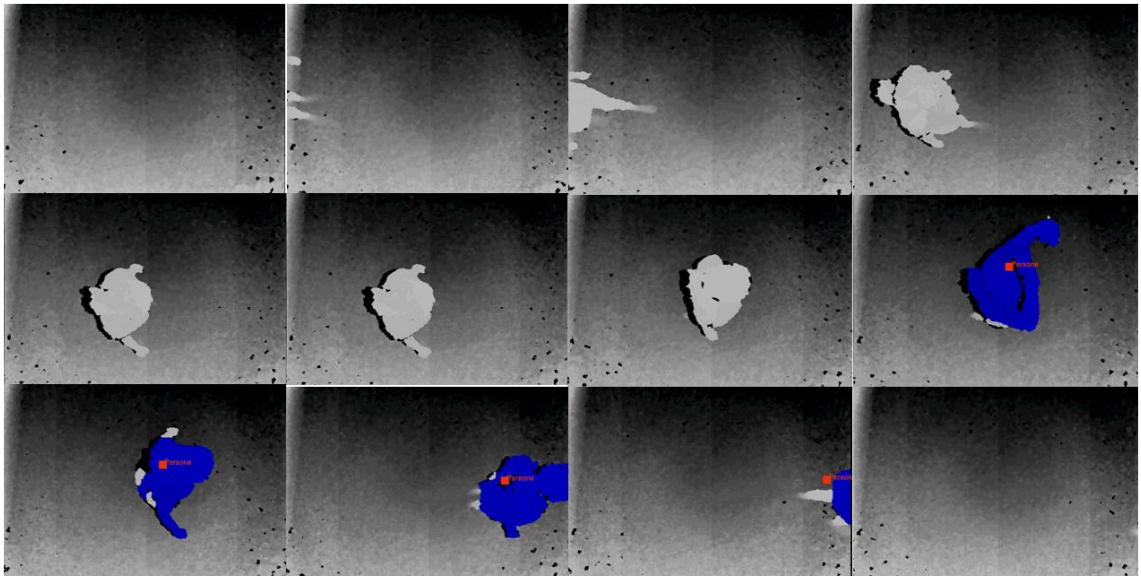


Figura 4.4. Detección mediante gestos del Algoritmo 3.

4.4.1 Trayectorias planteadas

Trayectoria 1:

El recorrido planteado, básicamente es lineal, de tal forma que se demuestra que los algoritmos detectan y siguen a la persona en la escena. Se desea mostrar una trayectoria simple para observar los resultados de los cuatro algoritmos. La trayectoria planteada la podemos observar en la **Figura 4.5**.

Figura 4.5. Trayectoria 1

Trayectoria 2:

El recorrido planteado es más complejo, es decir no es una línea recta. Se desea mezclar una línea recta con una especie de Zig-zag, pero sin que la persona, salga de la escena durante el recorrido, únicamente se sale de la escena en el

momento en el que la trayectoria se termina. La trayectoria planteada la podemos observar en la **Figura 4.6**.

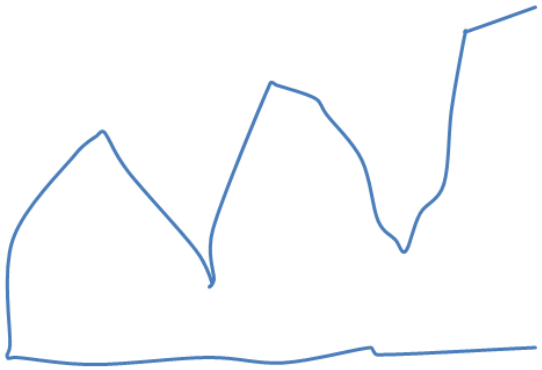


Figura 4.6. Trayectoria 2

Trayectoria 3:

El recorrido planteado, es diferente a los anteriores, ya que lo que se desea es analizar cómo reaccionan los algoritmos al momento de que la persona sale de la escena y entra de nuevo. Entonces en esta trayectoria se tiene una especie de dos “L’s”, ya que se entra en la escena, se camina en “L”, se sale de la escena, y al entrar de nuevo, se hace una “L” pero inversa a la anterior. La trayectoria planteada la podemos observar en la **Figura 4.7**.

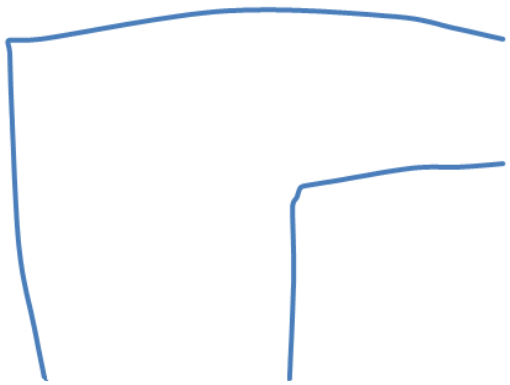


Figura 4.7. Trayectoria 3

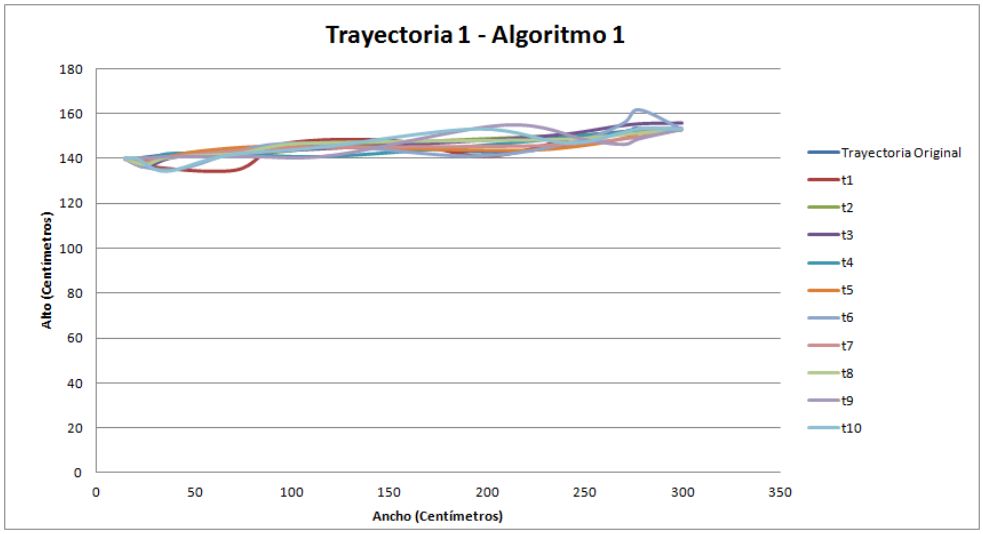
4.4.2 Experimento 1

En este primer experimento realizamos las pruebas con los algoritmos 1 y 2, con cada una de las trayectorias. Se realizaron 20 pruebas de cada trayectoria para

los dos algoritmos. Para cada trayectoria se aplicó el método de análisis mencionado anteriormente, de esta forma conoceremos cuál de los algoritmos trabaja mejor en este experimento.

Si observamos la **Figura 4.8**, tenemos las 10 trayectorias del usuario 1, para la trayectoria 1, analizadas por el algoritmo 1 y algoritmo 2, y se puede ver que las trayectorias en cada algoritmo son similares.

a)



b)

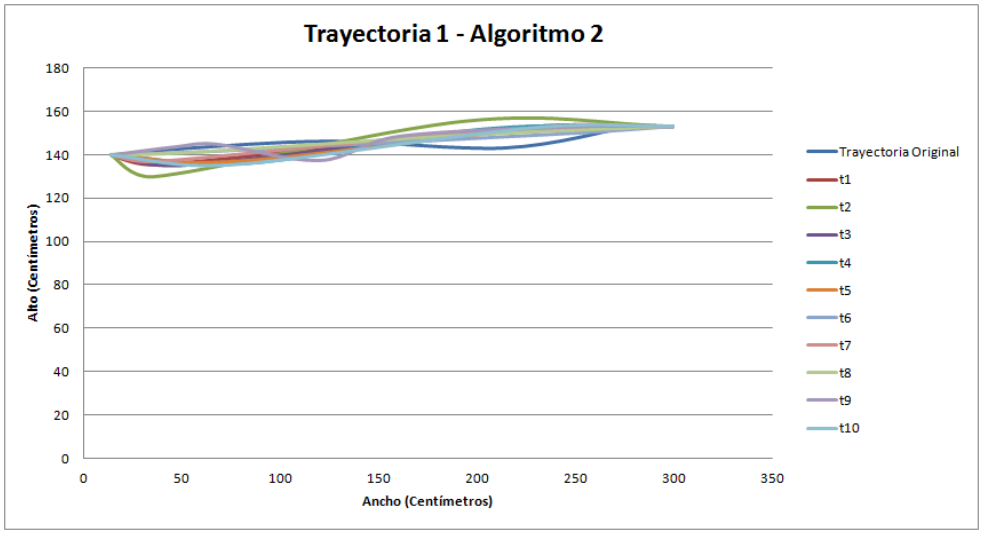


Figura 4.8. Trayectoria 1 del Usuario 1 para el Experimento 1
a) Algoritmo 1 b) Algoritmo 2

Los resultados que se han obtenido se muestran en la **Tabla 4.1** (Análisis de la varianza, Área Media y Desviación Estándar) las unidades de medida son los Centímetros. Básicamente nos basaremos en los resultados del área media

(AM), para realizar las comparaciones. Claramente podemos ver que en la trayectoria 1, AM es menor para los 2 algoritmos comparada con el resto de trayectorias, el motivo es porque esta trayectoria es simple, ya que es caminar de forma horizontal en la escena una vez, y sin volver a entrar en la escena. El AM para cada algoritmo, en la trayectoria 2 es más elevada, y es porque esta trayectoria es más compleja que la primera; pero en este caso también el algoritmo 2, tiene la media menor. Para la trayectoria 3, el AM es elevado con respecto a las trayectorias 1 y 2, y el motivo es porque esta trayectoria es más compleja, ya que la persona entra y sale de la escena. En general, el algoritmo que tiene la AM menor en las tres trayectorias, es el algoritmo 2.

	Algoritmo 1	**Algoritmo 2**	**F**	**p**
Trayectoria 1	206.35±78.75	184.72±68.15	0.34	0.57
Trayectoria 2	292.28±152.84	273.98±106.44	0.07	0.79
Trayectoria 3	481.51±142.56	388.34±123.53	1.93	0.18

Tabla 4.1. Análisis de la Varianza Experimento 1 (Centímetros).

También se ha llevado a cabo un análisis estadístico de los datos utilizando ANOVA. En la **Tabla 4.1**, se muestra dicho análisis. De los datos se observa que no existe una diferencia estadística significativa en ninguna de las trayectorias. Pero, si consideramos las medias, el algoritmo 2 obtiene una media menor para todas las trayectorias. Por lo tanto, considerando el análisis estadístico y las medias, se puede concluir que no existen diferencias estadísticas significativas entre los dos algoritmos, pero que la media del algoritmo 2 se aproxima más a la trayectoria original.

Si observamos gráficamente las trayectorias que realiza el Algoritmo 2 (**Figura 4.9**, **Figura 4.10** y **Figura 4.11**), son bastantes similares a la trayectoria original en las tres trayectorias, existiendo diferencias pequeñas. El algoritmo 1, también es similar a la trayectoria original, pero las diferencias son más notables que con el algoritmo 2.

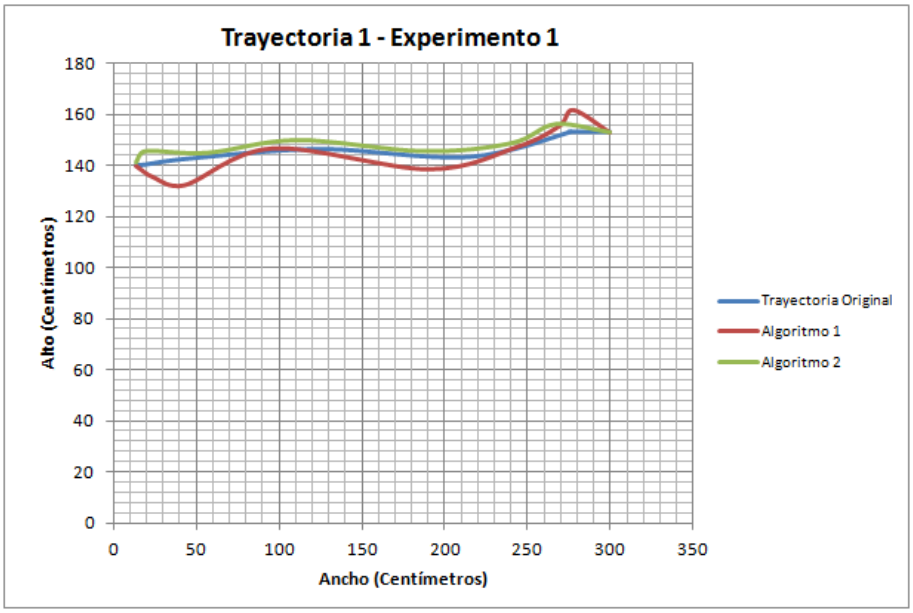


Figura 4.9. Resultados de la Media de Trayectoria 1

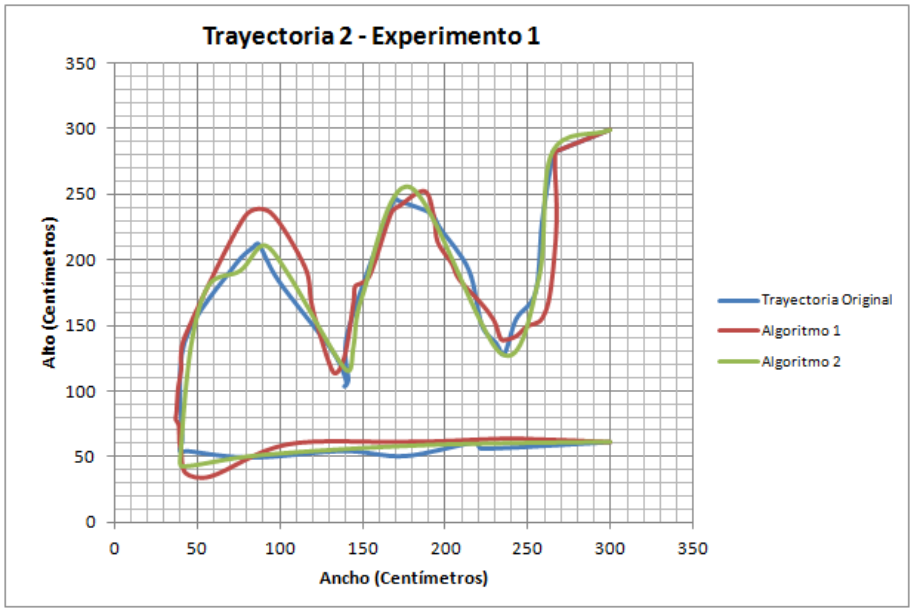


Figura 4.10. Resultados de la Media de Trayectoria 2

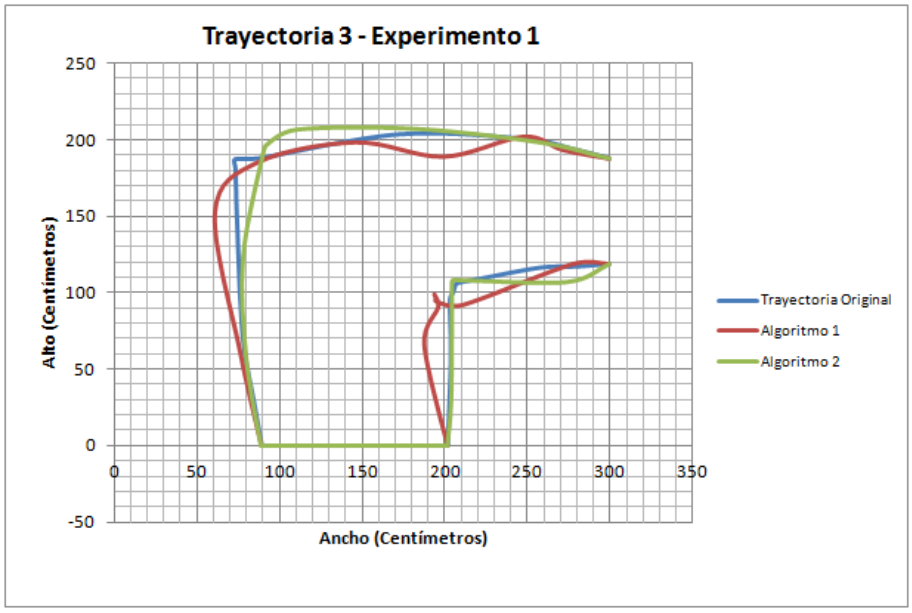


Figura 4.11. Resultados de la Media de Trayectoria 3

Dados los resultados obtenidos en este primer experimento, el algoritmo que se aproxima más a la trayectoria real en las tres trayectorias planteadas, es el Algoritmo 2.

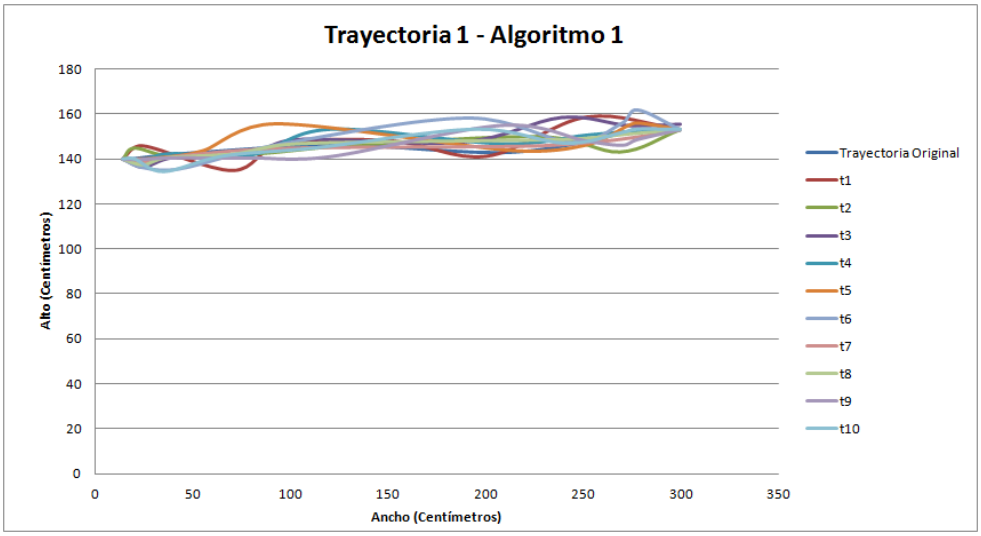
4.4.3 Experimento 2

En este segundo experimento realizamos las pruebas con los algoritmos 1, 2 y 4, y siguiendo las mismas trayectorias que en el experimento 1. Se realizaron 20 pruebas de cada trayectoria para los tres algoritmos. Para cada trayectoria se aplicó el método de análisis mencionado, así de esta forma conoceremos cuál de los algoritmos trabaja mejor en este experimento.

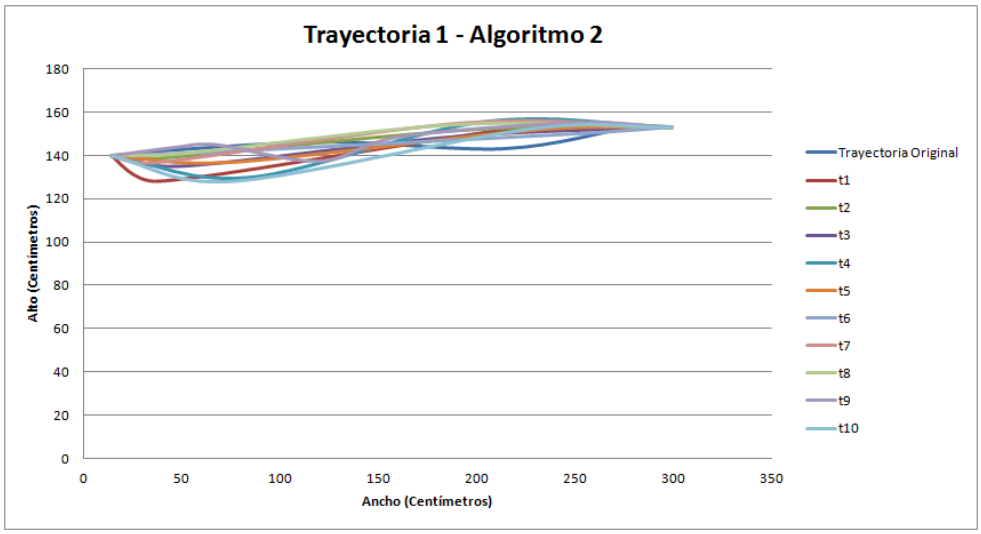
En este experimento existen dos algoritmos (1 y 4) que trabajan con el mismo dispositivo, es decir con la Cámara Logitech Pro 9000, y un algoritmo (2) con Kinect.

Si observamos la **Figura 4.12**, tenemos las 10 trayectorias del usuario 1, para la trayectoria 1, analizadas por los algoritmos 1, 2 y 4; y se puede ver que las trayectorias en cada algoritmo son similares.

a)



b)



c)

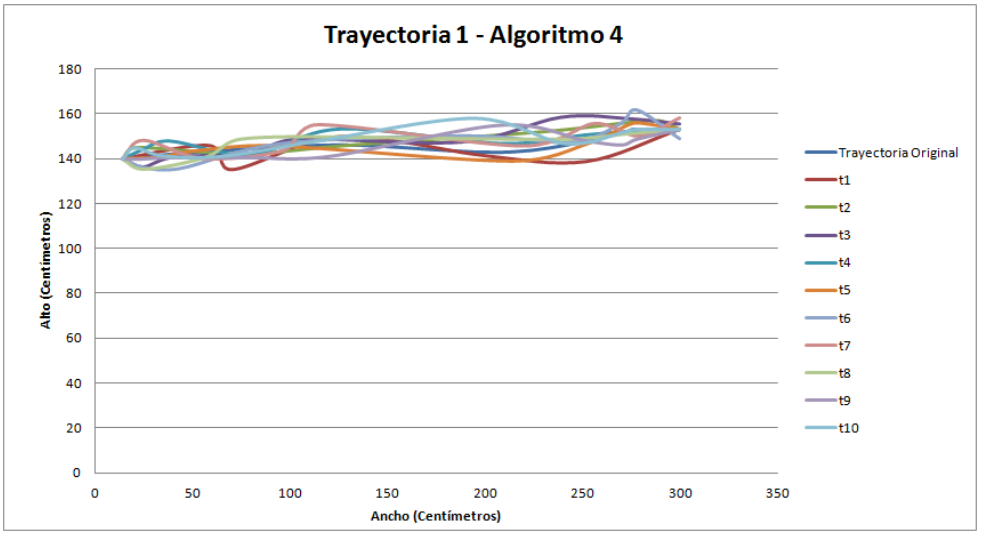


Figura 4.12. Trayectoria 1 del Usuario 1 para el Experimento 2
a) Algoritmo 1 b) Algoritmo 2 c) Algoritmo 4

Podemos observar en la **Tabla 4.2** (Área Media y Desviación Estándar), donde las unidades de medida de ambos son los centímetros, para cada uno de los algoritmos en cada trayectoria. En este experimento podemos observar al igual que en el experimento 1, que la AM en la trayectoria 1 para cada algoritmo es la menor, y es por el motivo que ya lo mencionamos con anterioridad, y es por el tipo de trayectoria, de la que se trata.

Para la trayectoria 2, la diferencia entre los algoritmos 1 y 2 con respecto a la trayectoria 1, la diferencia no es grande, pero el algoritmo 4, tiene una diferencia más elevada. Para la trayectoria 3, la diferencia en los 3 algoritmos es más grande que con respecto a las trayectorias anteriores, y el motivo es el cual mencionamos en el experimento 1.

	Algoritmo 1		**Algoritmo 2**		**Algoritmo 4**	
	AM	**DE**	**AM**	**DE**	**AM**	**DE**
Trayectoria 1	282.47	127.59	246.36	95.92	305,77	164,72
Trayectoria 2	319.99	72.89	261,93	109,12	408,99	147,87
Trayectoria 3	487.58	120.72	367,45	125,13	550.87	141.54

Tabla 4.2. Área Media (AM) y Desviación Estándar (DE) del Experimento 2 (cm.)

Si observamos la **Tabla 4.3**, que es el análisis de varianza entre los algoritmos 1 y 2; observamos que no existe una diferencia estadística significativa entre los algoritmos 1 y 2, a pesar de que para la trayectoria 3, el valor de p es 0.07. En la **Tabla 4.4**, tenemos los resultados del análisis de varianza de los algoritmos 2 y 4, observamos que para la trayectoria 3, existe diferencia estadística significativa. Esta trayectoria ya es compleja, pero si observamos los valores de AM, el algoritmo 4, tiene un AM muy elevado con respecto al algoritmo 2, esto significa que el algoritmo 4 tiene una gran diferencia entre las trayectorias del usuario y la trayectoria real. Y finalmente en la **Tabla 4.5**, observamos los resultados del análisis de varianza de los algoritmos 2 y 4, donde no existe diferencia estadística significativa.

	Algoritmo 1	Algoritmo 2	F	p
Trayectoria 1	282.47±127.59	246.36±95.92	0.51	0.48
Trayectoria 2	319.99±72.89	261.93±109.12	1.10	0.31
Trayectoria 3	487.57±120.72	367.45±125.13	3.61	0.07

Tabla 4.3. Análisis de la Varianza de Algoritmo 1 y Algoritmo 2 (Centímetros).

	Algoritmo 2	Algoritmo 4	F	p
Trayectoria 1	246.36±95.92	305.77±164.72	2.64	0.12
Trayectoria 2	261.93±109.12	408.99±147.87	4.58	0.05
Trayectoria 3	367.45±125.13	550.87±141.54	7.42	0.01**

Tabla 4.4. Análisis de la Varianza Algoritmo 2 y Algoritmo 4 (Centímetros).

** Indica diferencias estadísticas significativas

	Algoritmo 1	Algoritmo 4	F	p
Trayectoria 1	282.47±127.59	305.77±164.72	0.27	0.61
Trayectoria 2	319.99±72.89	408.99±147.87	2.92	0.10
Trayectoria 3	487.571±120.72	550.87±141.54	1.16	0.30

Tabla 4.5. Análisis de la Varianza Algoritmo 1 y Algoritmo 4 (Centímetros).

En general, en el experimento 2, el algoritmo que presenta el AM menor, es el algoritmo 2, seguido del algoritmo 1, y por último el algoritmo 4. En este experimento también se puede observar que la diferencia de resultados en los tres algoritmos es media, pero donde existe más diferencia de valores es en la trayectoria 3. Y además existiendo una diferencia estadística significativa entre el algoritmo 2 y algoritmo 4.

Si vemos gráficamente el recorrido que realiza cada uno de los algoritmos con respecto a la trayectoria original (**Figura 4.13, Figura 4.14** y **Figura 4.15**), observaremos cómo está trabajando cada algoritmo en cada trayectoria, y se entenderá claramente cómo trabaja cada algoritmo.

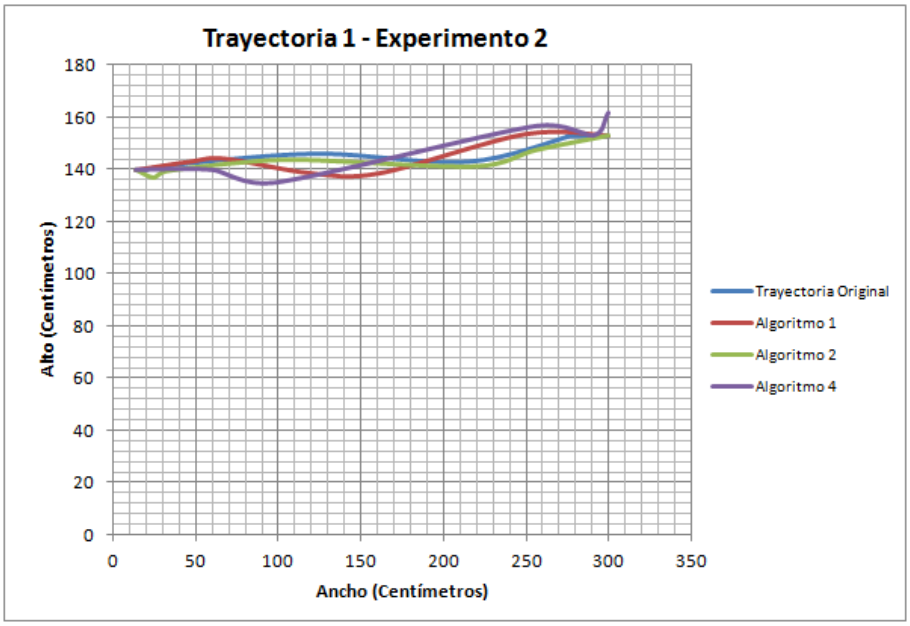


Figura 4.13. Resultados de la Media de Trayectoria 1

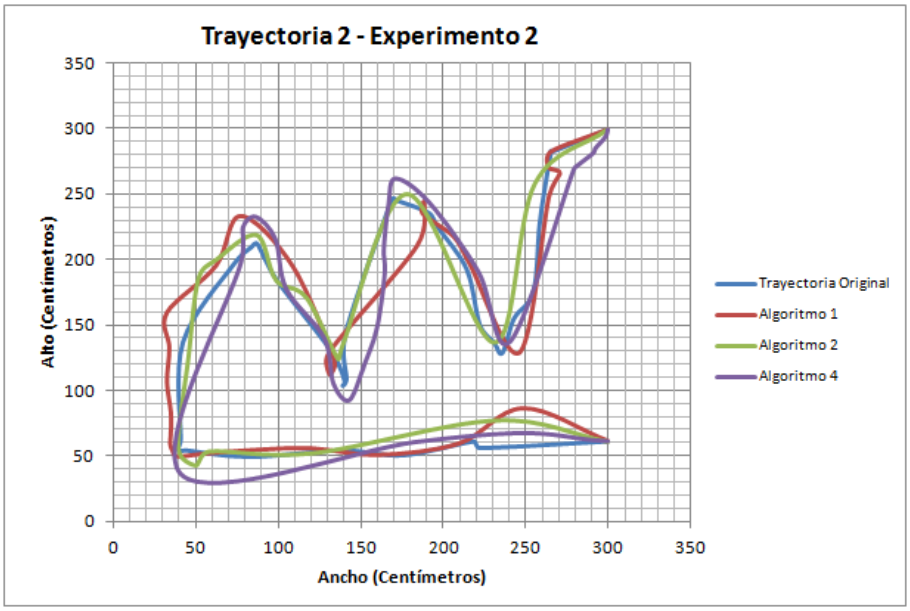


Figura 4.14. Resultados de la Media de Trayectoria 2

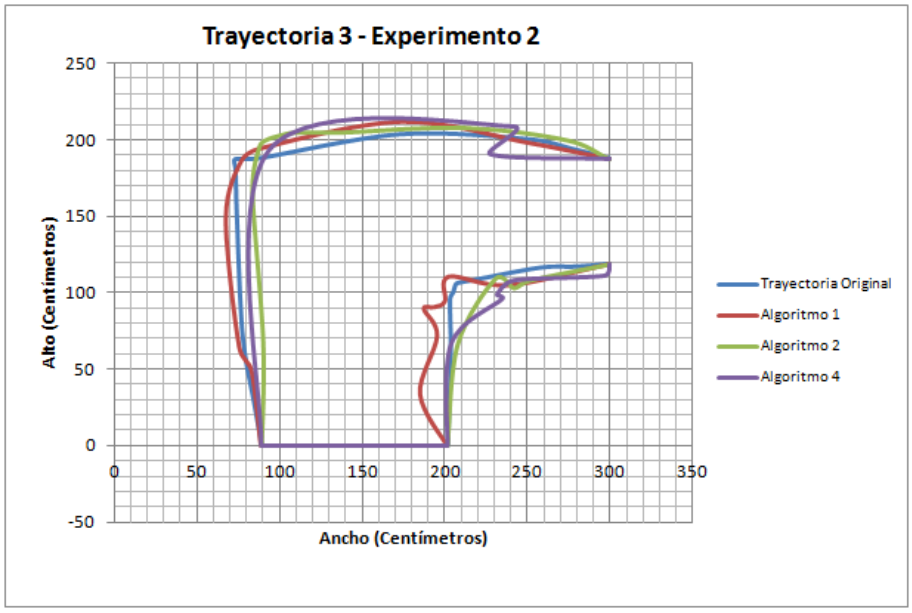


Figura 4.15. Resultados de la Media de Trayectoria 3

Se puede observar en las gráficas que en la trayectoria 1, la diferencia de trayecto es pequeña, para los tres algoritmos, lo que sucede es que se encuentra a una distancia de varios centímetros, y es por esta razón que AM en muchos casos es más elevado; gráficamente se ve pequeña, pero en sentido numérico, es alta. El algoritmo 1, no tiene problemas para detectar ni seguir a la persona, ya que el trayecto que realiza es similar, aunque con pequeñas variaciones, el algoritmo 2, al igual que el algoritmo anterior, hace la detección y el seguimiento sin ningún problema, y finalmente el algoritmo 4, no tiene mayor problema, el trayecto trazado es similar, pero con un leve desplazamiento más que el algoritmo 1.

Para la trayectoria 2, se puede observar claramente la diferencia que existe entre cada algoritmo. El algoritmo 1 tiene una diferencia media con respecto al algoritmo 2, que la trayectoria es similar a la original, a pesar de que existen sectores donde la diferencia es clara. El algoritmo 4 tiene una diferencia notable con respecto a la trayectoria original y al algoritmo 2.

En este experimento pudimos observar que el algoritmo 1, tiene valores más elevados que en el experimento 1, esto se debe a que en este experimento las personas llevaban un identificador en la cabeza, y esto afecta en la reacción que tiene el algoritmo con respecto al identificador.

El algoritmo 4, tuvo algunos problemas al momento de seguir a la persona, y la causa es porque los cambios de iluminación en la escena muchas veces hacen que el identificador se vea más oscuro de lo normal.

Dados los resultados obtenidos en el experimento 2, el algoritmo que se aproxima más a la trayectoria real es el Algoritmo 2. En ambos experimentos se eligió al algoritmo 2, por tener la media menor y por aproximarse más a las trayectorias reales.

CAPÍTULO V

CONCLUSIONES

5.1 Conclusiones

A lo largo de esta tesina se han realizado diferentes tareas que cumplen con los objetivos planteados. Después de una búsqueda exhaustiva de sistemas de detección y seguimiento de objetos, se encontraron las diferentes técnicas que existen, y qué técnicas son las más usadas para la detección, además de que las técnicas también dependen del tipo de detección, la posición de los dispositivos de captura, y de la escena. Con la información estudiada, se logró conocer los procesos necesarios para una detección y un seguimiento de objetos y qué técnicas son necesarias para realizar estos procesos. Al tener la información necesaria y haberla estudiado, se logró establecer las técnicas de detección y seguimiento, que más se adecuaban a nuestro escenario y dispositivos, y plantearnos los algoritmos que se desarrollarían, para después compararlos.

Con respecto a las librerías y el entorno de desarrollo, que serían necesarios para el desarrollo de los algoritmos, se pudo constatar que existen librerías enfocadas a la visión artificial, de tal forma que facilitan los procesos de captura, procesamiento de la información, despliegue de la misma, etc. OpenCV, es la librería que se eligió para el desarrollo de los algoritmos planteados, ya que cuenta con una amplia gama de funciones y algoritmos implementados, para la captura de información, procesamiento, y despliegue de la misma. Además ya que uno de los dispositivos usados sería Kinect, se hizo el estudio de otra librería llamada, OpenNI, la cual es compatible con Kinect, y facilita el desarrollo de aplicaciones para este dispositivo, además de ser capaz de trabajar conjuntamente con OpenCV.

En cuanto al desarrollo de los algoritmos, básicamente fueron planteados como se mencionó anteriormente, con el estudio realizado de las técnicas de detección y seguimiento. Los algoritmos fueron planteados con las técnicas más usadas para la detección y seguimiento de objetos. Se realizó una combinación de técnicas para los

algoritmos 1, 2 y 4, mientras que para el algoritmo 3, se puso a prueba las técnicas de detección y seguimiento de OpenNI.

Si nos centramos en los resultados obtenidos por los algoritmos podemos decir lo siguiente:

- El algoritmo 1, que usa como dispositivo de captura de información la Cámara Logitech Pro 9000; y el algoritmo 2, que usa como dispositivo de captura de información Kinect, funcionan como se deseaba. Se observó que cuando una persona entra en la escena, la detectan rápidamente, y empiezan a realizar el seguimiento de la persona, sin problemas. En el momento en el cual la cabeza de la persona sale de la visión de la cámara, la persona deja de ser detectada. Si la persona entra en la escena de nuevo, el algoritmo, la detecta y la continúa siguiendo.
- Para el Algoritmo 1, realizando la sustracción de fondo utilizando *Mixture of Gauss*, se obtuvieron buenos resultados, mejorando la detección con la eliminación de ruidos y sombras. En un principio, se hicieron pruebas sin el uso de las técnicas para eliminar ruido y sombras; éste detectaba a la persona, pero a la vez detectaba a cualquier objeto en movimiento en la escena, incluso a la misma sombra de la persona. Es por este motivo que se usaron los filtros mencionados en el capítulo 3. La detección y el seguimiento trabajan correctamente, y obtuvimos resultados aceptables.
- El Algoritmo 2, usa una técnica similar a la sustracción de fondo, pero en sí es una función combinada entre OpenCV y OpenNI que permite establecer una región de Análisis de la Cámara de Profundidad. Con esta técnica también obtuvimos buenos resultados, pero al igual que para el A1, se aplicaron filtros de eliminación de ruido, por las mismas razones que se mencionaron en el párrafo anterior. La detección y seguimiento, fueron satisfactorios, obteniendo buenos resultados, cumpliendo con nuestro objetivo principal
- El algoritmo 3, que usa como dispositivo de captura de información Kinect, no funciona como se esperaba. Cuando la persona entra en la escena, el algoritmo no la detecta rápidamente, incluso a veces no la detecta. Cuando el algoritmo si detecta a la persona, realiza el seguimiento, pero muchas veces realiza el seguimiento de forma errónea, ya que empieza a dibujar una trayectoria que la persona no ha realizado. Además que cuando la persona sale de la escena, y vuelve a entrar en ella, el algoritmo o la detecta con su centro de masa en un lugar erróneo, o no la detecta. Al realizar las pruebas se

observó también, que con este algoritmo, para que la persona sea detectada, muchas veces la persona tiene que hacer movimientos o gestos llamativos con las manos, pero estas acciones no deberían realizarse.

Las causas de que este algoritmo no funcione como se deseaba, es porque las funciones de OpenNI y Kinect, están desarrollados, para capturar información de la persona, pero situada frente a esta, de tal forma que al estar frente a la persona, detecta la cabeza, los pies, los brazos, las manos; y en nuestro caso, lo único que puede ver Kinect es la cabeza y hombros de la persona, también se pueden ver los pies, pero no asocia éstos con la cabeza. Dentro de la literatura, no se encontró información, sobre aplicaciones para Kinect, que sitúen el dispositivo de la forma que se sitúa en este trabajo, y que además detecten personas. Ya que el dispositivo esta desarrollado en un principio, para los juegos de videoconsola, y siempre tendrán al usuario de frente, y lo reconoce de esta forma, no se ha tenido en cuenta situar a la Kinect en el techo y de forma perpendicular al suelo.

- El algoritmo 4, que usa como dispositivo de captura de información la Cámara Logitech Pro 9000, detecta a la persona en el momento en el que el identificador está dentro la visión de la cámara, si el identificador se encuentra fuera de la visión de la cámara, la persona deja de ser detectada. El problema con este algoritmo es que si el identificador está muy oscuro, por la iluminación de la escena, no lo detecta, aunque la persona se encuentre dentro de la escena, pero no existen mayores problemas.

Podemos observar que el algoritmo 1 y el algoritmo 2, son los que funcionan correctamente, y no tienen inconvenientes al momento de detectar o de seguir. El Seguimiento con el Filtro de Kalman, en los algoritmos en los cuales se aplicó, funciona satisfactoriamente, y no así en el algoritmo 3, que usa una función de seguimiento de OpenNI. Los inconvenientes que se encontraron con los cuatro algoritmos son:

Un inconveniente que se encontró en los cuatro algoritmos, es que cuando la persona se queda en un lugar sin moverse y por un periodo largo, la persona deja de ser detectada, o es detectada incorrectamente, ya que empieza a detectar algún objeto que esté en la escena similar al de una cabeza.

Observamos al realizar las pruebas en cada experimento, que las trayectorias obtenidas de un mismo algoritmo y de una misma trayectoria, son diferentes, no en

su totalidad, ya que tienen la misma forma de la trayectoria original, pero las coordenadas van cambiando. Esto se debe a que una persona no camina de la misma forma en todas las pruebas, es decir que la persona camina más rápido, hace movimientos bruscos o más lentos, y lo más importante es el movimiento de la cabeza, ya que lo que se detecta en la persona, es principalmente ésta. Entonces cuando una persona mueve la cabeza de cierta forma, el centro de masa cambia, y las coordenadas de movimiento son diferentes. Este hecho sucede en los tres algoritmos evaluados.

Finalmente podemos concluir que el algoritmo 2, es el que obtuvo mejores resultados en ambos experimentos. Es decir, que el algoritmo 2, cumple con el objetivo principal de este trabajo. Al no existir diferencias estadísticas significativas más que entre los algoritmos 2 y 4, en la trayectoria 3, el Algoritmo 2, fue elegido porque las trayectorias trazadas por este algoritmo son similares a las trayectorias reales, además porque la media en cada trayectoria es menor que las medias de los algoritmos 1 y 4. El dispositivo elegido es Kinect, y es el algoritmo 2, el cual se podría aplicar al proyecto que motivó este Trabajo Final de Máster. El algoritmo 1, sería la segunda opción, y además cuenta con otro tipo de dispositivo.

5.2 Trabajo Futuro

Se plantean una serie de líneas de trabajo futuras, como continuación del presente trabajo:

- Principalmente, aplicar el algoritmo 2, al proyecto de CHILDMNEMOS, el cual desea detectar al niño y seguirlo, en una escena controlada, para evaluar su memoria espacial a corto plazo. Este sistema será enfocado a niños videntes y no videntes.
- Por otro lado, en el caso de la detección se puede utilizar otra técnica de detección, la cual se basa en el aprendizaje, y es la técnica llamada *Haar Training*, la cual también facilitaría la detección de una persona. Esta técnica también es bastante usada en sistemas de detección.
- En el desarrollo de los algoritmos, que utilizan la sustracción de fondo basado en Gauss Mixture, un problema importante reside cuando la persona permanece estáticamente mucho tiempo en la escena. Para solucionar este problema se podría añadir un módulo, que se encargue de solucionar este inconveniente, pues evidentemente aunque la persona se quede estática, es importante seguir detectándola de forma correcta.

- Se podría modificar los algoritmos, de tal forma que se pueda detectar y seguir a más de una persona a la vez, en una misma escena.
- Si se desea aplicar los algoritmos en un ambiente más grande, y tener una visión más amplia, se podría aumentar el número de dispositivos. Para ello, se tendrá que modificar el código fuente. Otra solución sería trabajar con un ordenador por dispositivo, y centralizar el análisis en uno solo.

REFERENCIAS

Agusti, M., & Armenteros, J. Interacción con OpenCV: Seguimiento de un objeto por color.UPV. http://riunet.upv.es/bitstream/handle/10251/12682/interaccionPorColor.pdf

Andreopoulos A., (2011). Active object recognition in theory and practice. *PhD Thesis*. Dept. of Computer Science & Engineering, York University.

Babu, G. S. (2012). Moving object detection using MATLAB. *International Journal of Engineering Research & Technology (IJERT)*, 1(6). http://www.ijert.org/browse/august-2012-edition?download=715%3Amoving-object-detection-using-matlab

Ballester, J., & Pheatt, C. (2013). Using the Xbox Kinect sensor for positional data acquisition. *American Journal of Physics*, 81: 71 – 77.

Barandiaran, J., Murguia, B., & Boto, F. (2008). Real-time people counting using multiple lines. *Ninth International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS),* pp. 159 – 162.

Bradski, G., & Kaehler, A., Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly Media, Inc. Publication, 1005 Gravenstein Highway North, Sebastopol, 2008.

Breckon, T. P., Barnes, S. E., Eichner, M. L., & Wahren, K. (2009). Autonomous real-time vehicle detection from a medium-level UAV. *24th International Unmanned Air Vehicle Systems*, pp. 29.1 – 29.9.

Chan, A. B., Liang, Z. S., & Vasconcelos, N. (2008). Privacy preserving crowd monitoring: Counting people without people models or tracking. *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1 – 7.

Chen, T. H., Chen, T. Y., & Chen, Z. X. (2006). An intelligent people-flow counting method for passing through a gate. *Conference on Robotics, Automation and Mechatronics*, pp. 1 – 6.

Chhabra, P., (2009). MSc. Thesis dissertation: An architecture for Video Change Detection for Unmanned Aerial Vehicle applications. A Thesis Submitted for the Degree of MSc Erasmus Mundus in Vision and Robotics (VIBOT). http://gradvibot.u-bourgogne.fr/thesis/PC_thesis.pdf

Choi, J., Dumortier, Y., Prokaj, J., & Medioni, G. (2012). Activity Recognition in Wide Aerial Video Surveillance Using Entity Relationship Models. *Technical Report.* University of Southern California.

http://iris.usc.edu/outlines/papers/2012/choi_dumort_prokaj_medioni_icmr.pdf

Córdoba, F., (2012). Detección de robo/abandono de objetos en interiores utilizando cámaras de profundidad. *Proyecto Fin de Carrera*. Universidad Autónoma de Madrid.

Crouzet, S. M., & Serre, T. (2011). What are the visual features underlying rapid object recognition?. *Frontiers in psychology*, 2 (326): 1 – 15.

Davis, J., & Chen, X. (2001). A laser range scanner designed for minimum calibration complexity. *Third International Conference on 3D Digital Imaging and Modeling*, pp. 91 – 98.

Ge, S., Xu, T. F., Ni, G. Q., & Liu, Y. H. (2009). Target recognition and tracking method for one-off aerial imaging system. *International Conference on Optical Instrumentation and Technology: Optoelectronic Imaging and Process Technology*, pp. 75131U - 75131U.

Gutierrez, A. (2011). Videovigilancia y privacidad. La ocultación del rostro en vídeo para el cumplimiento del derecho a la intimidad. *Trabajo de Fin de Master*. Universidad de Sevilla.

García, J., Gardel, A., Bravo, I., Luis Lázaro, J., Martínez, M., & Rodríguez, D. (2012). Detección y Seguimiento de Personas Basado en Estereovisión y Filtro de Kalman. *Revista Iberoamericana de Automática e Informática Industrial (RIAI)*, 9(4): 453 – 461.

Gąszczak, A., Breckon, T. P., & Han, J. W. (2011). Real-time people and vehicle detection from UAV imagery. *SPIE Conference Intelligent Robots and Computer Vision XXVIII: Algorithms and Techniques*, 78.

González, U. (2010). Detección de personas a partir de visión artificial. *Proyecto Fin de Carrera*. Universidad Carlos III de Madrid.

Huang, C. H., Wu, Y. T., Kao, J. H., Shih, M. Y., & Chou, C. C. (2010). A hybrid moving object detection method for aerial images. *Proceedings of the 11th Pacific Rim conference on Advances in multimedia information processing*, pp. 357 – 368.

Karasulu, B. (2010). Review and evaluation of well-known methods for moving object detection and tracking in videos. *Journal of aeronautics and space technologies*, 4(4): 11 – 22 .

Khashman, A. (2008). Automatic Detection, Extraction and Recognition of moving objects. *International Journal of Systems Applications, Engineering & Development*, 2(1): 43 – 51.

Khoshelham, K. (2011). Accuracy analysis of kinect depth data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, 38(5): 133 – 138.

Lee, J. T., Chen, C. C., & Aggarwal, J. K. (2011). Recognizing human-vehicle interactions from aerial video without training. *Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 53 – 60.

Lee, S. (2012). Depth camera image processing and applications. *International Conference on Image Processing (ICIP)*, pp. 545 – 548.

Masoud, O. & Papanikolopoulos, N. P. (2001). A novel method for tracking and counting pedestrians in real-time using a single camera. *Trans. Vehicular Technology*, 50 (5): 1267 – 1278.

Moeslund, T. B., Hilton, A., & Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2 – 3): 90 – 126.

Nixon, M., & Aguado, A. S. (2008). *Feature extraction & image processing*.

Poppe, R. (2010). A survey on vision-based human action recognition. *Image and vision computing*, 28(6): 976 – 990.

Rizzon, L., Massari, N., Gottardi, M., & Gasparini, L. (2009, May). A low-power people counting system based on a vision sensor working on contrast. *International Symposium on Circuits and Systems, (ISCAS)*, pp. 786 – 786.

Rodríguez-Canosa, G. R., Thomas, S., del Cerro, J., Barrientos, A., & MacDonald, B. (2012). A real-time method to detect and track moving objects (DATMO) from unmanned aerial vehicles (UAVs) using a single camera. *Remote Sensing*, 4(4): 1090 – 1111.

Rossi, M. & Bozzoli, A. (1994). Tracking and counting moving people. *International Conference on Image Processing (ICIP)*, 3: pp. 212 – 216.

Russell, S. J., Norvig, P., "Artificial Intelligence: A Modern Approach, Second Edition", Pearson Education, Inc., Upper Saddle River; New Jersey, 2003.

Sanabria, J. J., & Archila, J. F. (2011). Detección y análisis de movimiento usando visión artificial. *Scientia Et Technica*, 49: 180 – 188.

Sankaranarayanan, A. C., Veeraraghavan, A., & Chellappa, R. (2008). Object detection, tracking and recognition for multiple smart cameras*. Proceedings of the IEEE,* 96(10): 1606-1624*.*

Sen-Ching, S. C., & Kamath, C. (2004, January). Robust techniques for background subtraction in urban traffic video. *International Society for Optics and Photonics Electronic Imaging*, pp. 881 – 892.

Scharstein, D., & Szeliski, R. (2002) A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision (IJCV)*, 47(1 – 3): 7–42.

Scharstein, D., & Szeliski, R. (2003). High-accuracy stereo depth maps using structured light. *Conference on Computer vision and pattern recognition (CVPR)*, pp. 195 – 202.

Szeliski, R. (2010). Computer vision: Algorithms and applications.

Vélez, J. F., Moreno, A. B., Sánchez, & Á., Sánchez, J. L., 2º edición. *Visión por computador.*

Velasco, N. D. R. (2013). Desarrollo e implementación de un algoritmo para detección de objetos con tecnología KINECT. *Tesis Doctoral*.

Viola, P., Jones, M. & Snow, D. (2005). Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision,* 63(2): pp. 153 – 61.

Wang, J., Wang, Y., & Zhang, Z. (2011). Interesting region detection in aerial video using Bayesian topic models. *First Asian Conference on Pattern Recognition (ACPR)*, pp. 706-710.

Xingbao, W., Chunping, L., Gong, L., Long, L., & Shengrong, G. (2011, December). Pedestrian Recognition Based on Saliency Detection and Kalman Filter Algorithm in Aerial Video. *Seventh International Conference on Computational Intelligence and Security (CIS)*, pp. 1188 – 1192.

Xu, H., Lv, P., & Meng, L. (2010). A people counting system based on head-shoulder detection and tracking in surveillance video. *International Conference on Computer Design and Applications (ICCDA)*, (1): pp. 394 – 398.

Yilmaz, A., Javed, O., & Shah, M. (2006). Object tracking: A survey. *ACM Computing Surveys (CSUR)*, 38(4): 13 – 56.

Yu, S., Chen, X., Sun, W., & Xie, D., (2008). A robust method for detecting and counting people. *In: Proc. Int. Conf. Audio, Language and Image Processing ICALIP 2008*. pp. 1545–1549.

Zhang, W., & Zelinsky, G. (2004). Current Advances in Computer-based Object Detection and Target Acquisition. *Technical Report*. pp 1 – 25.

APÉNDICES

APÉNDICE A

CONFIGURACIÓN OPENCV Y OPENNI

Para poder configurar OpenCV y OpenNI primeramente debe instalarse ambas librerías, y después proceder a la configuración de ambos.

A.1. Instalación OpenCV y configuración Visual Studio

La instalación de OpenCV en este caso será para Windows 7 de 64 bits. Para esto seguimos los siguientes pasos:

- Instalar Visual Studio 2010 en nuestro ordenador.
- Necesitamos la librería de OpenCV, una versión estable, en este caso se usa la versión 2.4.3, que se puede descargar de :

  http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.3/
- Una vez descargado ejecutamos el *.exe*, nos pedirá donde queremos descomprimir y le damos la dirección C:\opencv\.
- Después de haberse extraído se debe configurar las variables del sistema, para que llame a las librerías de OpenCV, para eso se va a **Equipo** y seleccione **Propiedades del sistema > Configuración avanzada del sistema > Variables de entorno**.

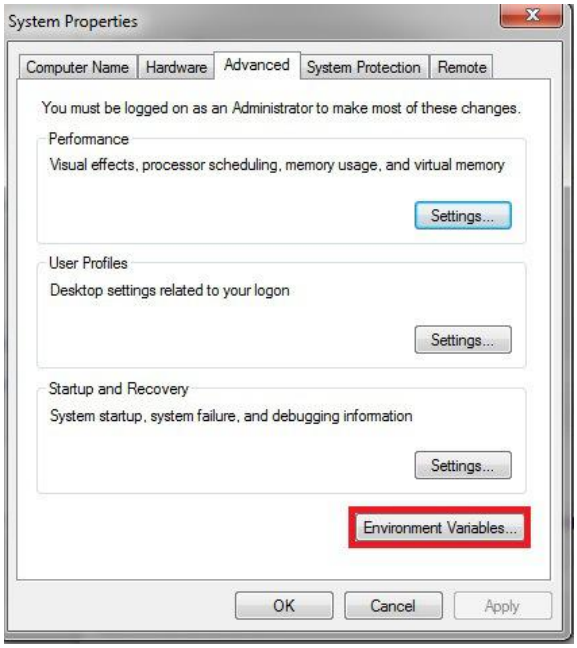


Figura A.1. Variables de Entorno

Y seleccionamos la variable Path y la editamos, donde agregaremos al final de la línea:

C:\opencv\build\x64\vc10\bin;

En caso de que se trate de un procesador de 32 bits, la única diferencia, es cambiar el x64 a x86.

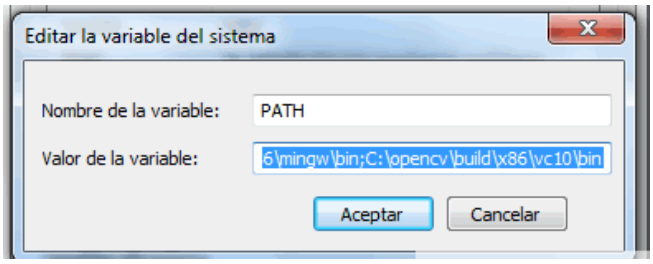


Figura A.2. Valor de la variable

A continuación, se configura Visual Studio 2010, para eso primero se crea un proyecto vacío, una vez creado, nos vamos a las propiedades del proyecto y se debe cambiar el proyecto a plataforma de 64 bits:

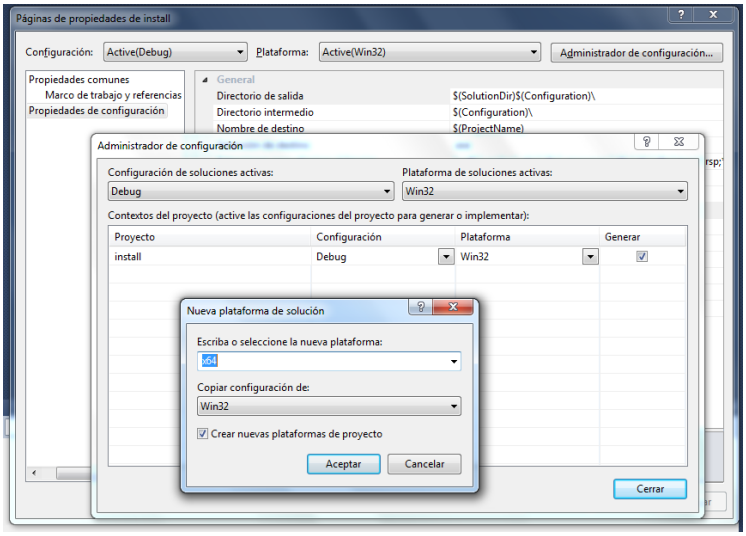


Figura A.3. Modificar a Plataforma x64

Se definen las librerías y las inclusiones que necesita OpenCV, para esto nos vamos a **Propiedades de configuración > Directorios de VC++ > Directorios de Archivos de inclusión** ahí agregamos los siguientes directorios:

C:\opencv\build\include;C:\opencv\include\opencv;

En esa misma ventana buscamos **Propiedades de configuración > Directorios de VC++ > Directorios de Archivos de biblioteca** y agregamos la siguiente dirección:

C:\opencv\build\x64\vc10\lib;

Y por último, **Propiedades de configuración > Vinculador > Entrada > Dependencias adicionales** y agregamos las siguientes líneas:

opencv_core243d.lib
opencv_imgproc243d.lib
opencv_highgui243d.lib
opencv_ml243d.lib
opencv_video243d.lib
opencv_features2d243d.lib
opencv_calib3d243d.lib
opencv_contrib243d.lib
opencv_flann243d.lib
opencv_gpu243d.lib
opencv_legacy243d.lib
opencv_objdetect243d.lib

Con esto ya tiene configurado, para que las aplicaciones se ejecuten en Visual Studio 2010 con OpenCV.

A.2. Instalación OpenNI

Al igual que la anterior librería se instalará para Windows 7 de 64 bits para lo cual se siguen los siguientes pasos:

- Se necesita **descargar OpenNI SDK v1.5.4.0 binary, PrimeSensor Modules OpenNI Drive , PrimeSense NITE Unestable Build**, todos deben ser para Windows x64, y se los podrá descargar en esta dirección:

  http://code.google.com/p/simple-openni/

- Todos estos paquetes deben ser instalados en el mismo orden que fueron mencionados y sin tener conectado el sensor Kinect al ordenador. Para establecer que OpenNI trabaje con el sensor Kinect, se debe sustituir archivos que corresponden a Kinect Sensor. Para ello supongamos que ha instalado todos los paquetes a sus carpetas predeterminadas. Vamos al zip que descargamos **PrimeSensor Modules OpenNI Drive** y buscamos la carpeta **\OpenNI\Data** dentro de esta se encuentra un archivo llamado **SamplesConfig.xml** este lo copiamos y reemplazamos en la siguiente dirección:

  C:\Program Files\OpenNI\Data

  Después de eso descargamos el archive XML de esta dirección:

  https://www.dropbox.com/s/n5etmxqot9e3gko/OpenNIKinectXML.zip

  Y lo reemplazamos todo el contenido de este zip dentro de la carpeta:

  C:\Program Files\PrimeSense\NITE\Data

A.3. Configuración de OpenCV y OpenNI

Después de instalar OpenCV y OpenNI, configuramos a ambos para que trabajen conjuntamente, seguimos los siguientes pasos:

- Instalar CMake para Windows, el cual lo puede descargar de:

  http://www.cmake.org/files/v2.8/

- Una vez instalado abrimos CMake, y es aquí donde se especifica donde se ha instalado OpenCV y después donde se compilará lo que generemos con CMake como se muestra en la siguiente imagen:

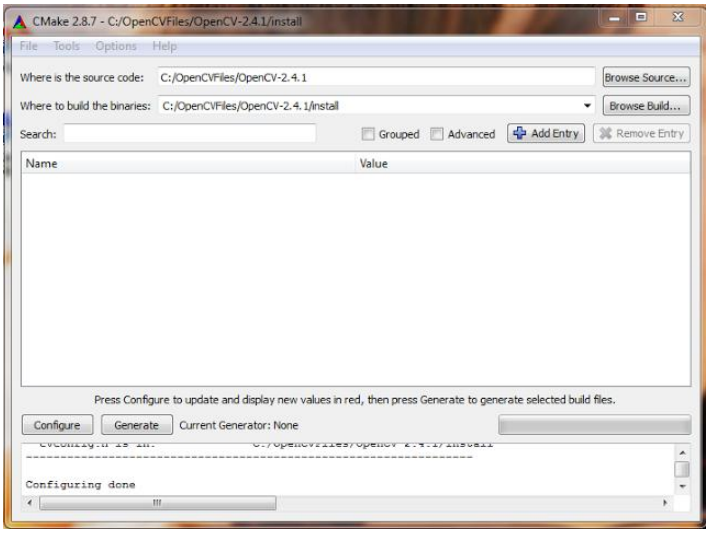


Figura A.4. Configuracion CMake (Paso 1)

Después hacemos click en **Configure**, nos aparecerá una pantalla donde tendremos que especificar el proyecto a generarse, como estamos trabajando con Visual Studio 2010 seleccionamos este tipo. Después le damos finalizar y nos volverá a aparecer la ventana cuando abrimos CMake pero con todas las librerías que tiene OpenCV y una vez más hacemos **Configure** para que se genere el proyecto. Después de que termine nos volverá a aparecer la ventana con las librerías de OpenNI sin marcar, las cuales debemos marcar para que se integre con las librerías de Opencv como se muestra en la siguiente pantalla.

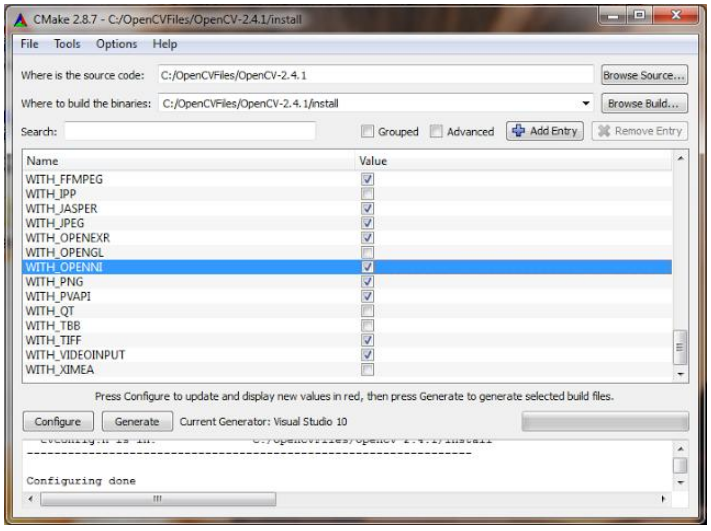


Figura A.5. Configuración CMake (Paso 2)

Y volvemos a hacer click en **Configure**, y nos mostrará las direcciones donde tenemos instalado OpenNI validamos estas direcciones y volvemos a apretar **Configure**.

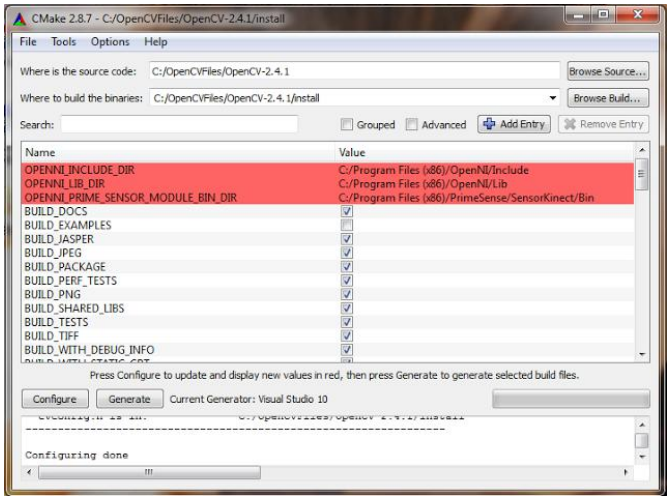


Figura A.6. Configuración CMake (Paso 3)

Con esto ya está listo para generarse el código que después compilaremos para generar las nuevas librerías donde se encuentren OpenCV y OpenNI.

- Buscamos la dirección donde se generó el proyecto y buscamos el archivo **OpenCV.sln** lo abrimos con Visual Studio**,** una vez abierto hacemos click en explorador de soluciones y en la barra de menú buscamos **Build > Build Solution,** y se generará la solución para compilar las librerías.
- Una vez que termine de generar la solución podemos recién generar las librerías conjuntas, para eso buscamos en explorador el siguiente archivo INSTALL y generamos la solución de este archivo:

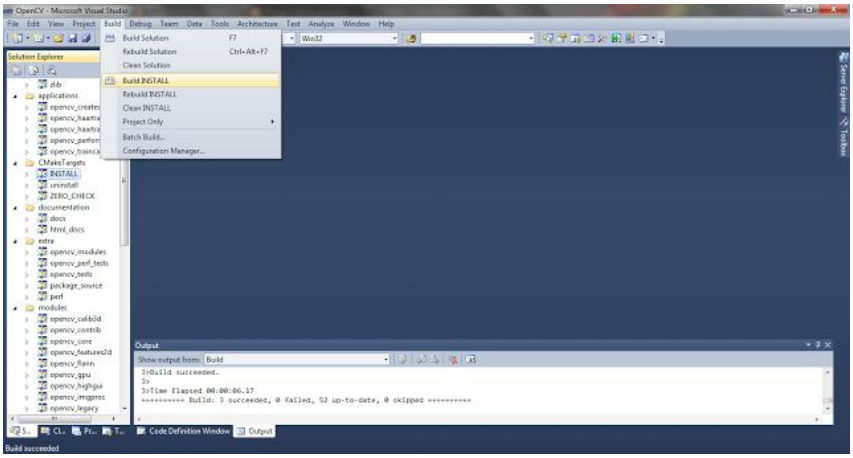


Figura A.7. Build Solution

Con esto ya habremos obtenido las librerías e inclusiones que se necesitan para ejecutar OpenCV y OpenNI juntos, estas se encuentran ubicadas donde está el proyecto en una carpeta llamada **Install** , de igual forma que configuramos el OpenCV en la Instalación volvemos a configurarlo pero tomando las librerías de esta carpeta y la configuramos de igual forma en el Visual Studio.

APÉNDICE B

LA LONGITUD DEL CONTORNO Y LA SUPERFICIE DE LA COTA MÍNIMA ALGORITMO 1

Para obtener la longitud de una persona, y así filtrar las regiones de interés para el algoritmo 1. En base al mismo algoritmo, después de hacer la sustracción de fondo realizamos la búsqueda de contornos y calculamos la longitud de cada contorno y guardamos las regiones que una persona tiene al pasar a través de la cámara, esto se guardó en un archivo de texto para luego hacer una análisis de cota mínima de las regiones de la persona. Este proceso se lo muestra en las siguientes figuras.

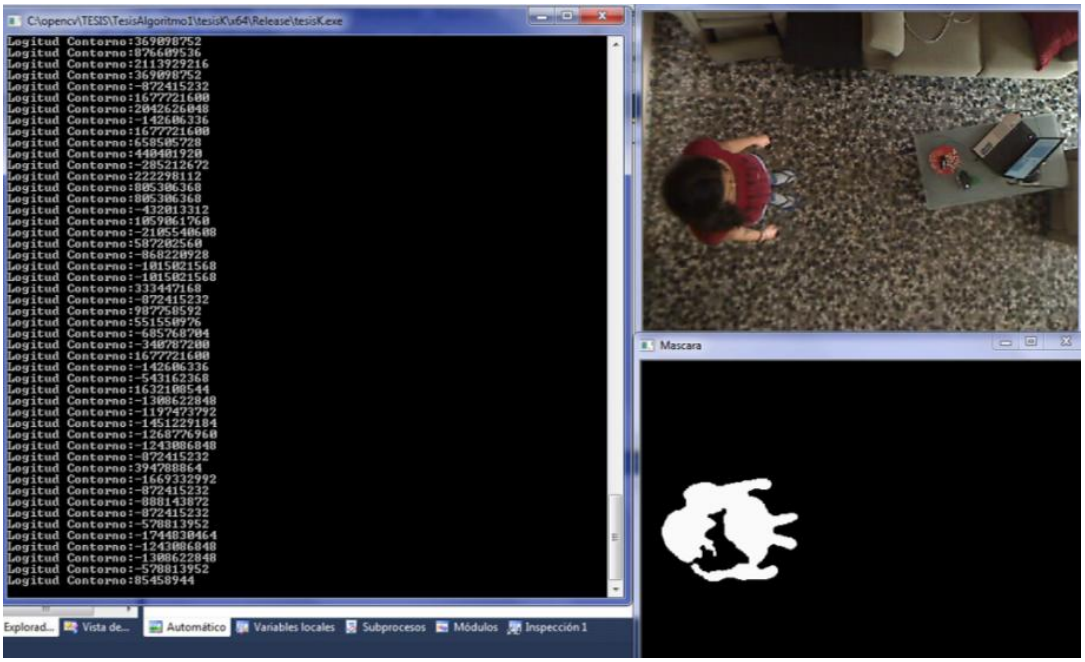


Figura A.8. Longitud de Contorno y máscara Algoritmo 1.

Como se observa tenemos 3 ventanas, la primera y la más grande muestra la longitud del contorno que está remarcando de color blanco en la venta **máscara,** y para estar seguros de que la región detectada en la **máscara** se trata de una persona tenemos la ventana **imagen** donde se muestra la escena capturada. Los datos mostrados en la ventana de la izquierda, se almacenan en un archivo de texto, el cual se analizará para obtener la cota mínima, que se muestra en la siguiente tabla:

Nº	Longitud del Contorno	Nº	Longitud del Contorno
1	1491	26	1512
2	1575	27	1522
3	1487	28	1459
4	1467	29	1571
5	1486	30	1524
6	1532	31	1591
7	1503	32	1496
8	1481	33	1482
9	1524	34	1555
10	1583	35	1522
11	1575	36	1454
12	1518	37	1545
13	1558	38	1569
14	1515	39	1513
15	1559	40	1476
16	1537	41	1535
17	1498	42	1515
18	1476	43	1514
19	1520	44	1490
20	1477	45	1550
21	1464	46	1519
22	1566	47	1463
23	1551	48	1598
24	1457	49	1472
25	1524	50	1488
Promedio		**1517,18**	
Máximo		**1598**	
Mínimo		**1454**	
Desviación		**38,8737832**	

Tabla A.1. Cota mínima Algoritmo 1

Como se observa en la **Tabla A.1** la cota mínima que tomaremos para la detección será el promedio obtenido de **1517,** es decir, que todo contorno que sea mayor a este valor, será considerado como el contorno de una persona.

APÉNDICE C

CALIBRAR DISTANCIA DE LA KIENCT A LA PERSONA

De igual forma que el apéndice anterior se desarrolló una herramienta para poder calibrar la distancia de la región de análisis con la cámara de profundidad del Kinect y obtener la longitud del contorno que la persona refleja en la escena.

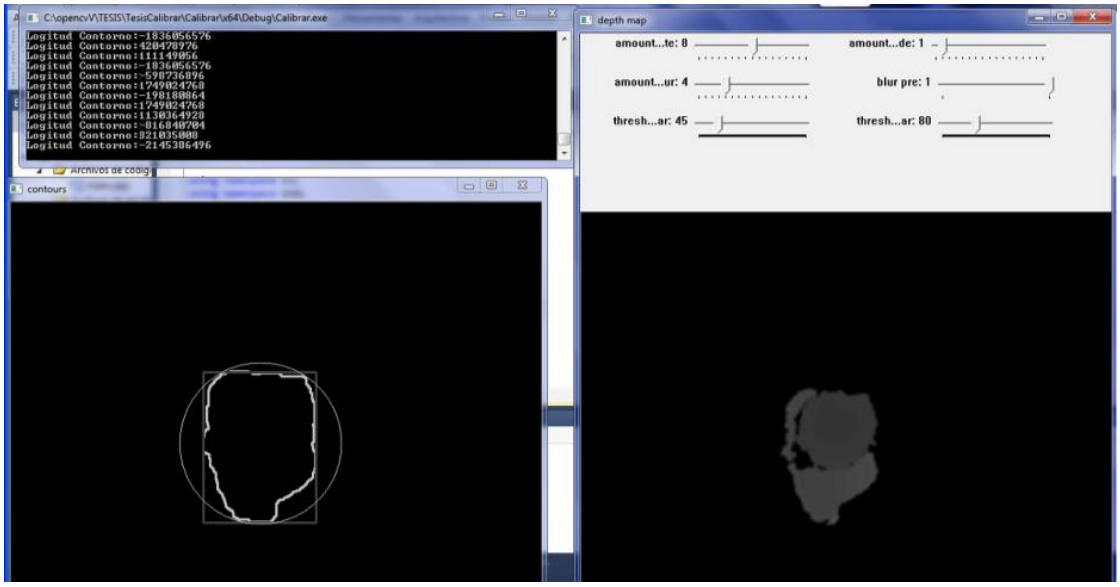


Figura A.9. Longitud de Contorno y máscara Algoritmo 2.

Como se observa en las imágenes, obtenemos los valores de ***threshNear*** que es igual a ***45*** y también el parámetro ***threshFear*** que es ***80.*** Estos parámetros sirven para establecer la región de análisis de la cámara de Kinect con respecto a la escena en la ventana ***depth map*** y en la misma ventana se puede observar cómo se filtró de la escena, los hombros y la cabeza de la persona. Ahora para tener una mejor detección y validación de que el objeto que pase sobre la cámara Kinect se trata de una persona al igual que el anterior Apéndice obtenemos la longitud del contorno que se ve en la ventana de ***contours,*** en base a esto pudimos establecer la siguiente tabla:

Nº	Longitud del Contorno	Nº	Longitud del Contorno
1	575	16	574
2	587	17	584
3	590	18	564
4	586	19	576

5	574	20	581
6	575	21	561
7	599	22	577
8	585	23	584
9	583	24	587
10	560	25	582
11	571	26	578
12	566	27	571
13	591	28	587
14	580	29	583
15	577	30	572
Promedio		**578,6**	
Máximo		**599**	
Mínimo		**560**	
Desviación		**9,1**	

Tabla A.2. Cota Mínima Algoritmo 2

Como se observa en la tabla obtenida el tamaño de la cota minima para el caso del algoritmo 2, el valor que sonsideramos para que se trate de una persona sera si la longitud de un contorno es mayor a ***578***, con esto estamos filtrando otros objetos que pasen a cierta distancia del la camara Kinect y no se sean una persona.

APÉNDICE D

APLICACIÓN PARA OBTENER LOS VALORES HSV PARA EL ALGORITMO 4

Para establecer los valores máximos y mínimos de los canales HSV de nuestro identificador, desarrollamos un programa que permita modificar estos valores hasta poder obtener solo el color del identificador esto se muestra a continuación:

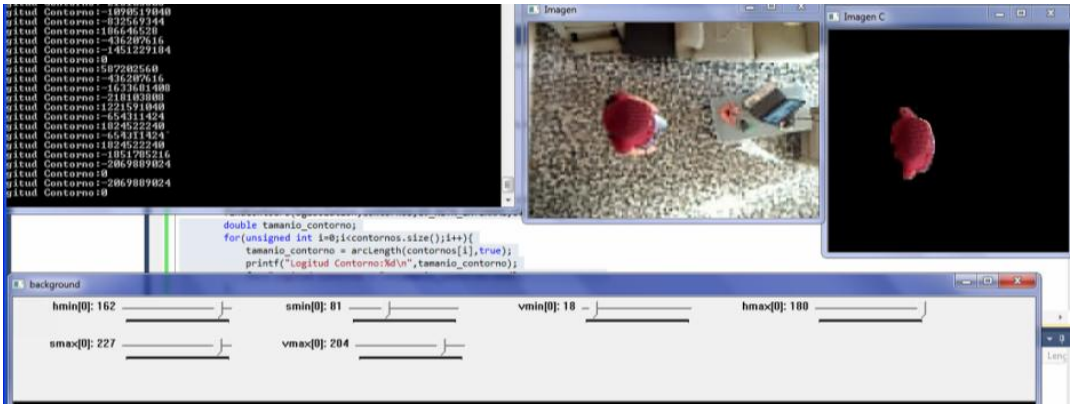


Figura A.10. Valores HSV del Identificador Algoritmo 4.

Como se observa en las Figura logramos obtener los valores HSV del identificador, que en nuestro caso es un "*gorrito*" de color granate. Este identificador podría ser otro, ya dependiendo de lo que cada persona desee. Para el algoritmo 4, se toman en cuenta los valores como mínimo ***162*** y máximo ***180*** para el canal H, para el canal S el valor mínimo ***81*** y máximo ***227,*** y por ultimo para el canal V el mínimo ***18*** y máximo ***204*** como se ve en la ventana de ***Background*** e ***Imagen C.***

Como los anteriores Apéndices igual obtenemos una cota mínima de longitud del contorno de nuestro identificador para poder filtrar estos contornos, esto se muestra en la siguiente tabla:

Nº	Longitud del Contorno	Nº	Longitud del Contorno
1	492	16	485
2	470	17	469
3	496	18	473
4	499	19	495
5	493	20	487
6	489	21	490
7	465	22	473
8	492	23	465

9	479	24	486
10	499	25	467
11	470	26	497
12	487	27	482
13	485	28	498
14	475	29	481
15	496	30	494
Promedio		**484,3**	
Máximo		**499**	
Mínimo		**465**	
Desviación		**11,14**	

Tabla A.3. Cota Mínima Algoritmo 4

Como se ve en la tabla la cota mínima de longitud para filtrar las regiones de interés tendrán que ser mayor a ***484,*** si la longitud de los contornos es mayor a ese valor se considera que esa región pertenece a nuestro identificador.