Universidad Nacional Autónoma de México

Posgrado en Ciencia e Ingeniería de la Computación

Rastreo y Evaluación de Gestos Manuales Usando Captura de Movimiento

T E S I S

Que para optar por el grado de:
Maestro en Ciencia e Ingeniería de la Computación

P R E S E N T A :

Marco Antonio Caballero Guerrero

Director de Tesis:
Dr. Jorge Alberto Márquez Flores
CCADET

Co-director de Tesis:
Dr. Alfonso Gastelum Strozzi
CCADET

Ciudad de México, Marzo 2018

Hoja de datos del Jurado

1. Alumno Caballero Guerrero Marco Antonio Universidad Nacional Autónoma de México Posgrado en Ciencia e Ingeniería en Computación 406061342
2. Presidente Dr. Alfonso Gastelum Strozzi
3. Vocal Dr. Jorge Alberto Márquez Flores
4. Secretario Dr. Miguel Ángel Padilla Castañeda
5. Suplente Dra. Fabiola Miroslaba Villalobos Castaldi
6. Suplente Dr. Patrice Jean Delmas
7. Trabajo escrito Rastreo y Evaluación de Gestos Manuales Usando Captura de Movimiento 104 páginas 2018

Agradecimientos

A Dios y a la vida misma, por permitirme seguir en este mundo experimentando cada uno de sus altibajos... sólo ellos saben con exactitud todo lo que he pasado y sentido estos últimos años.

A mis padres, a mi familia y a mis verdaderos amigos, por su amor, apoyo y comprensión, por los instantes de risas y aventuras... porque jamás se han dado por vencidos conmigo en los momentos difíciles.

A mis grandes maestros de vida, grandes sabios, a quienes admiro enormemente y de quienes espero seguir aprendiendo para alcanzar mi plenitud.

A mis profesores del posgrado, por su entrega y dedicación, por mostrarme nuevos horizontes y hacerme descubrir lo que quiero hacer realmente en un futuro... Gracias, Wendy.

Agradezco a Conacyt por brindarme la beca de estudios de posgrado y de esta manera enfocarme con más tranquilidad al proceso de investigación.

A mis tutores, el Dr. Jorge Márquez y el Dr. Alfonso Gastelum, por su paciencia, por sus recomendaciones, por su confianza y apoyo, por darnos la oportunidad de buscar un proyecto de investigación un poco distinto a lo acostumbrado.

Al IIMAS, a la UIDT del Hospital General de México "Dr. Eduardo Licéaga" y al CCADET, por permitirme hacer uso de sus instalaciones y equipo durante mis estudios y pruebas de laboratorio que requería para mis resultados.

A los doctores Miguel Padilla, Fabiola Villalobos y Patrice Delmas, por su asesoramiento y confianza en la revisión del trabajo realizado. Particularmente, gracias al Dr. Miguel y sus alumnos, por apoyarme en la realización de los experimentos necesarios en la UIDT.

A todos y cada uno, a quienes he podido conocer hasta ahora... sin esos lazos efímeros o muy significativos, mi vida no sería lo que es hoy. Quizás, nunca estaré preparado para dar el próximo paso... pero siempre lo intentaré al menos.

Marco

Resumen

En el presente escrito se expone el diseño y desarrollo de algoritmos para detección y rastreo de movimiento manuales, además se propone el uso de un sistema multivisión de captura basado en Kinects v2 y múltiples hilos de ejecución.

Para el proceso de segmentación de las regiones de interés se emplean marcadores pasivos en forma de anillos cubiertos con cinta reflejante que son distinguibles en las imágenes infrarrojas tomadas con los sensores correspondientes de los dispositivos.

La adquisición y sincronización de las cámaras de la herramienta se lleva a cabo a través de software y la arquitectura del sistema es adecuada para utilizar un número arbitrario de dispositivos, con la limitante del poder de cómputo del equipo empleado.

En cuanto a la utilidad final del sistema construido, éste fue aplicado para realizar un análisis cualitativo y cuantitativo del nivel de experiencia de especialistas en sutura de ojo por microcirugía con microscopio e instrumental.

Por último, aunque no es parte del alcance de esta tesis, es importante señalar que la herramienta podría extenderse para realizar reconocimiento de gestos en la ejecución de movimientos específicos para tareas de animación, simulación, lenguajes de señas, entre otras.

Índice general

1. Introducción **1**
- 1.1. Motivación . . . 2
  - 1.1.1. Gestos . . . 2
- 1.2. Rastreo de movimiento . . . 3
- 1.3. Definición del problema . . . 5
  - 1.3.1. Objetivos . . . 5
- 1.4. Estructura . . . 6

2. Antecedentes **7**
- 2.1. Algoritmos de segmentación . . . 8
  - 2.1.1. Vertientes . . . 9
  - 2.1.2. Simple Linear Iterative Clustering (SLIC) . . . 11
- 2.2. *Motion Tracking* . . . 13
- 2.3. Hand Tracking . . . 16
- 2.4. Kinect . . . 22
  - 2.4.1. Rastreo de gestos manuales . . . 22

3. Diseño y Desarrollo **28**
- 3.1. Adquisición de imagen . . . 28
- 3.2. Arquitectura multivisión . . . 30
  - 3.2.1. Calibración . . . 32
- 3.3. Método de detección . . . 35
  - 3.3.1. Pre-procesamiento . . . 35
  - 3.3.2. Segmentación . . . 38
- 3.4. Método de rastreo . . . 42

4. Experimentación y Resultados **44**
- 4.1. Recuperación de marcadores . . . 44
  - 4.1.1. Experimento 1 . . . 45
  - 4.1.2. Experimento 2 . . . 46
  - 4.1.3. Experimento 3 . . . 47
- 4.2. Análisis de movimientos manuales . . . 48
  - 4.2.1. Experimento A: Nudos . . . 48
  - 4.2.2. Experimento B: Suturas . . . 54
  - 4.2.3. Observaciones adicionales . . . 57

5. Conclusiones y trabajo a futuro **58**

Apéndices **60**

A. Gráficas de resultados de análisis de movimiento **60**

B. Código de adquisición de imágenes por multivisión **74**

C. Código de evaluación de movimiento **81**

Bibliografía **90**

Índice de figuras

1.1. Colección de gestos manuales distinguibles. 2
1.2. Traje con marcadores luminosos utilizados para rastreo de movimiento y proceso de generación de un modelo tridimensional asociado[(1)]. 4

2.1. Aplicaciones comunes de visión por computadora: (a) estéreo visión y reconstrucción 3D[(2)], (b) rastreo de elementos en movimiento[(3)], (c) segmentación y reconocimiento de objetos[(4)]. 7
2.2. Ejemplo de segmentación mediante superpíxeles. 8
2.3. Representación gráfica del relieve de una imagen, el cual es usado para segmentar por vertientes[(5)]: (a) imagen sintética de dos manchas negras, (b) línea de vertiente. 10
2.4. Ejemplo de segmentación por vertientes: (a) imagen con objetos en contacto, (b) binarización estándar conteniendo elementos unidos, (c) separación de los objetos y líneas de vertiente. 11
2.5. Segmentación por SLIC, mostrando superpíxeles de tamaño aproximado a 64, 256 y 1024 píxeles (Fuente: Achanta *et al.* [2012]). 13
2.6. Problemas presentes en el rastreo de objetos en movimiento: (a) desenfoque y oclusión de objetos[(6)], (b) desenfoque y cambios de iluminación en captura. . . . 14
2.7. Representación de objetos en movimiento a través de flujo óptico[(7)]. 14
2.8. Sustracción de fondo en una escena con objetos en movimiento[(8)]. 16
2.9. Superpíxeles de un brazo y mano calculados con SLIC[(9)]. 17
2.10. Resultado de la fusión de extracción de color y superpíxeles: imágenes de entrada (izquierda), extracción de color estándar (centro) y segmentación por método desarrollado por *Zhang y Huang* [2013]. 17
2.11. Comparación del método de rastreo propuesto por *Zhang y Huang* [2013] (izquierda), el método *mean-shift* (centro) y extracción estándar de color de la piel (derecha). 18
2.12. (a) Marcadores pasivos en forma de anillos de colores y (b) la distribución de las regiones en el espacio de color xyY (Fuente: Sasaki *et al.* [2009]). 19
2.13. Restricciones anatómicas para estimar las posiciones del dedo: (a) modelo 3D del dedo, (b) restricción de plano de movimiento (Fuente: Sasaki *et al.* [2009]). . . . 20
2.14. Resultado del modelo utilizando gaussianas anisotrópicas y con primitivas isotrópicas (Fuente: Sridhar *et al.* [2014]). 21
2.15. Método de rastreo de manos basado en suma de funciones gaussianas anisotrópicas (Fuente: Sridhar *et al.* [2014]). 22
2.16. Dispositivo infrarrojo Kinect v2. 22
2.17. Autoajuste del tamaño del cuadro de la ROI: (a) la mano está cerca del sensor y (b) la mano es alejada del sensor (Fuente: Bakar *et al.* [2014]). 24
2.18. (a) Secuencia de rotación de la mano y (b) rastreo de mano invariante a rotación con detección del centro de la palma (Fuente: Bakar *et al.* [2014]). 25

2.19. Guantes de 12 colores propuestos por Usachokcharoen *et al.* [2015]. 25
2.20. Proceso de obtención de la ROI de profundidad (Fuente: Usachokcharoen *et al.* [2015]). 26
2.21. Proceso de localización de la ROI de movimiento (Fuente: Usachokcharoen *et al.* [2015]). 26
2.22. Proceso de obtención de la ROI de color (Fuente: Usachokcharoen *et al.* [2015]). . 27

3.1. Características de las condiciones de laboratorio para la adquisición de imágenes: (a) tipo y color de superficies sugeridas, (b) arreglo de kinects. 29
3.2. Marcadores reflejantes para el sensor infrarrojo: (a) anillos ajustables para cada dedo, (b) los anillos son cubiertos con la cinta reflejante. 29
3.3. Imágenes de las manos con marcadores reflejantes; (a) frame RGB de la captura y (b) imagen infrarroja recuperada. 30
3.4. Guantes utilizados como recursos complementarios para el sistema de captura. . 30
3.5. Múltiples dispositivos conectados a un mismo equipo. 31
3.6. Ejemplo de sistema coordenado del sistema multivisión. 32
3.7. Distintas vistas del patrón de calibración con una misma cámara. 33
3.8. Calibración de dos capturas con múltiples Kinects. 33
3.9. Error medio de reproyección obtenido en la calibración de dos dispositivos para distintas capturas. 34
3.10. La ubicación del patrón de calibración en relación a las cámaras fijas. 34
3.11. El patrón calibrado con dos de los dispositvos. 35
3.12. Histograma de la imagen en grises invertida. 36
3.13. Aplicación de umbralizado estándar. 36
3.14. Filtrado de elementos pequeños: (a) imagen binaria con píxeles aislados y (b) erosión de componentes. 37
3.15. Asociación entre componentes conexas: (a) componentes originales de la imagen binaria y (b) remoción de componentes inválidas marcadas en rojo. 37
3.16. Máscara de imagen de fondo de las componentes: (a) la apertura remueve zonas ruidosas de los objetos y (b) la dilatación genera regiones correspondientes al fondo de los mismos. 38
3.17. Imagen de los objetos en primer plano: (a) la apertura se emplea para la transformada de distancia; (b) remoción de los bordes de los elementos de interés. . . . 39
3.18. Resultado de sustracción de fondo con respecto a los objetos de interés en primer plano. 39
3.19. Resultado de la segmentación aplicando *watershed*. 40
3.20. Algoritmo de *SLIC* sobre la imagen: (a) imagen binaria de componentes válidas, (b) generación de superpíxeles. 41
3.21. Filtrado de superpíxeles calculados: (a) todos los superpíxeles generados por *SLIC*, (b) los superpíxeles asociados a las componentes válidas. 41
3.22. Centroides de los marcadores segmentados. 42
3.23. Flujo óptico del movimiento rastreado mediante los marcadores. 42

3.24. Coordenadas espaciales de los marcadores: (a) posiciones en el plano desde la imagen infrarroja y (b) los valores de proximidad desde el sensor de profundidad. 43

4.1. Experimento 1: Segmentación de marcadores en ambas manos con dos dispositivos Kinect. 45
4.2. Experimento 1: Puntos característicos detectados para el flujo óptico. 45
4.3. Experimento 2: Segmentación de marcadores en ambas manos con dos dispositivos Kinect. 46
4.4. Experimento 2: Puntos característicos detectados para el flujo óptico. 46
4.5. Experimento 3: Segmentación de marcadores en ambas manos con dos dispositivos Kinect. 47
4.6. Experimento 3: Puntos característicos detectados para el flujo óptico. 47
4.7. Posiciones de los marcadores de un movimiento circular artificial; los puntos de un sólo color corresponden a cada marcador distinto. 48
4.8. Prueba de atadura de nudos; región de interés definida para la vista izquierda y derecha. 49
4.9. Coordenadas en el plano de los marcadores del usuario de prueba **PSEM**; el resto de figuras no citadas se ilustran en el Anexo A. 49
4.10. Aceleración de la mano izquierda del usuario **EEMA** y **PSEM**. 51
4.11. Velocidad de la mano izquierda del usuario **PSE-AH** y **PSE-S**. 53
4.12. Distribución de posiciones en el plano de los movimientos de la mano derecha del usuario **PSEM** y **PSE-S**. 53
4.13. Regiones de interés de la vista derecha de los especialistas **B** (izquierda) y **M** (derecha), respectivamente, en la prueba de sutura. 54
4.14. Posiciones de los marcadores de la mano izquierda del especialista **B** y **M** . . . 55
4.15. Comparación de las velocidades de la mano derecha entre el especialista **B** y **M**. 56
4.16. Comparación de las aceleraciones de la mano derecha entre el especialista **B** y **M**. 57

A.1. Coordenadas en el plano de los marcadores del especialista de prueba **M**. 60
A.2. Velocidades de los movimientos manuales del usuario de prueba **M**. 60
A.3. Aceleraciones de los movimientos manuales del usuario de prueba **M**. 61
A.4. Coordenadas en el plano de los marcadores del usuario de prueba **B**. 61
A.5. Velocidades de los movimientos manuales del usuario de prueba **B**. 62
A.6. Aceleraciones de los movimientos manuales del usuario de prueba **B**. 62
A.7. Coordenadas en el plano de los marcadores del usuario de prueba **EEBA**. . . . 63
A.8. Velocidades de los movimientos manuales del usuario de prueba **EEBA**. 63
A.9. Aceleraciones de los movimientos manuales del usuario de prueba **EEBA**. . . . 64
A.10. Coordenadas en el plano de los marcadores del usuario de prueba **EEBI**. 64
A.11. Velocidades de los movimientos manuales del usuario de prueba **EEBI**. 65
A.12. Aceleraciones de los movimientos manuales del usuario de prueba **EEBI**. 65
A.13. Coordenadas en el plano de los marcadores del usuario de prueba **EEMA**. . . . 66
A.14. Velocidades de los movimientos manuales del usuario de prueba **EEMA**. 66

A.15.Aceleraciones de los movimientos manuales del usuario de prueba **EEMA**. . . . 67
A.16.Coordenadas en el plano de los marcadores del usuario de prueba **PSEM**. . . . 67
A.17.Velocidades de los movimientos manuales del usuario de prueba **PSEM**. 68
A.18.Aceleraciones de los movimientos manuales del usuario de prueba **PSEM**. . . . 68
A.19.Coordenadas en el plano de los marcadores del usuario de prueba **PSE-AH**. . . 69
A.20.Velocidades de los movimientos manuales del usuario de prueba **PSE-AH**. . . . 69
A.21.Aceleraciones de los movimientos manuales del usuario de prueba **PSE-AH**. . . 70
A.22.Coordenadas en el plano de los marcadores del usuario de prueba **PSE-AM**. . . 70
A.23.Velocidades de los movimientos manuales del usuario de prueba **PSE-AM**. . . . 71
A.24.Aceleraciones de los movimientos manuales del usuario de prueba **PSE-AM**. . . 71
A.25.Coordenadas en el plano de los marcadores del usuario de prueba **PSE-S**. . . . 72
A.26.Velocidades de los movimientos manuales del usuario de prueba **PSE-S**. 72
A.27.Aceleraciones de los movimientos manuales del usuario de prueba **PSE-S**. . . . 73

Capítulo 1

Introducción

A lo largo de los años, el desarrollo de los sistemas de cómputo ha tenido como consecuencia una reducción de las dimensiones y peso de los circuitos digitales, permitiendo un incremento en la construcción y consumo de productos electrónicos y nuevas tecnologías, como computadoras portátiles, teléfonos y relojes inteligentes, tabletas, entre otros.

Al mismo tiempo, ha sido importante el proceso de investigación en el diseño y uso de estas tecnologías, haciendo énfasis en facilitar la comunicación entre las personas (usuarios) y las computadoras. Así, surge la rama conocida como interacción humano-computadora (*Human-Computer Interaction, HCI*), definida como el estudio de cómo la gente interactúa con dispositivos de cómputo y en qué medida éstos son desarrollados para que dicha interacción sea exitosa o satisfactoria para los seres humanos.

Algunas de las problemáticas que se intentan resolver en tal estudio se relacionan con cómo realizar el intercambio de información, la especificación de cuáles son las entradas requeridas y cómo son proporcionadas a un sistema, además de qué tipo de respuesta necesitamos y cómo representarla.

Adicionalmente, el avance tecnológico ha buscado la automatización de los procesos tanto para la simplificación de las tareas de los usuarios como para mejorar la precisión de los resultados en la realización de las mismas. En ese sentido, es común el uso de dos áreas de cómputo muy importantes, visión por computadora y reconocimiento de patrones, mediante las cuáles se lleva a cabo el procesamiento de colecciones de imágenes o videos con el fin de extraer, a través de un análisis específico, datos o características valiosas que puedan describir la presencia de ciertos elementos o eventos ocurriendo en las escenas capturadas.

1.1. Motivación

Los mecanismos actuales usados para interactuar con sistemas computacionales, como el uso de teclado, ratón, pantalla táctil, lápiz óptico, micrófono, cámara, etc, parecen suficientes para la mayoría de nuestros propósitos; sin embargo, en otras ocasiones la construcción de algoritmos y sistemas cuya interacción sea más sencilla o sin el uso de dispositivo alguno facilita el aprendizaje de los usuarios, en general, permitiendo además mejorar su desempeño en sus actividades cotidianas o ayudar en su inclusión en la sociedad, en particular; por ejemplo, manipulación de una computadora o aplicaciones[1] por medio de ciertos movimientos corporales, medición del rendimiento de un deportista, intérpretes de lenguajes de señas[2][3], etc.

1.1.1. Gestos

Un **gesto** (Figura 1.1) es una forma no verbal de comunicación en la cual acciones corpóreas visibles transmiten mensajes particulares, con o sin intervención del discurso hablado, permiten comunicar una variedad de pensamientos y sentimientos desde rebeldía y hostilidad hasta aprobación y afecto.

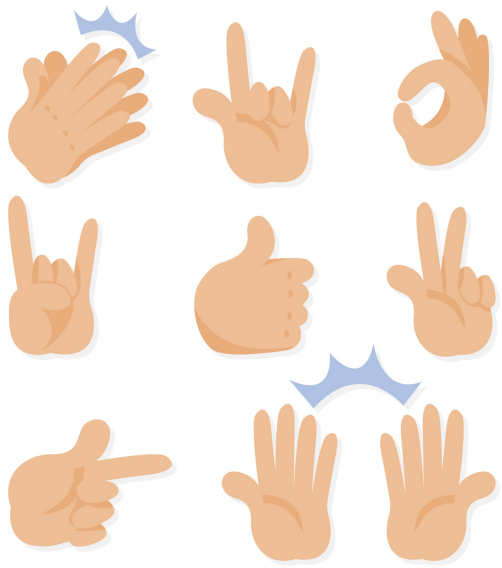


Figura 1.1: Colección de gestos manuales distinguibles.

Los gestos más comunes son conocidos como emblemas, símbolos o gestos citables; son gestos culturales o convenciones que pueden ser usados para reemplazar palabras (p.ej. ondeo de la palma para el 'Hola' o 'Adiós'). Un gesto simple emblemático puede tener distinto significado dependiendo del contexto cultural, variando desde algo agradable o complementario hasta algo altamente ofensivo.

Un *gesto dinámico* involucra cambios de posición sobre el tiempo, mientras que un *gesto estático* es observado enfáticamente en un cierto lapso. El ondeo de la mano en el gesto emblemático 'Adiós' es un ejemplo dinámico y la señal de 'Alto' es un ejemplo de gesto estático.

Debido a que los gestos incluyen movimientos de manos, cara y otras partes del cuerpo, proporcionan una forma natural, intuitiva, fácil y conveniente de interactuar con un sistema sin el uso de dispositivos adicionales.

Para entender un mensaje completo es necesario interpretar los gestos estáticos y dinámicos sobre un periodo de tiempo; este complejo proceso es conocido como **Reconocimiento Gestual**[4]. Algunas ventajas del reconocimiento gestual son[5]:

- No hay partes móviles; es decir, utilizar o no algún dispositivo auxiliar es posible; además, no implica (necesariamente) contacto entre el usuario y el dispositivo.

- Un sistema basado en gestos no requiere detectar o hacer sonidos, así que ruidos de fondo en captura de videos no afectan la información de entrada.

- Un sistema de reconocimiento de gestos podría controlar un número de dispositivos multimedia a través de la implementación de un conjunto de gestos intuitivos; por tanto, puede ser diseñado para parecer lo más natural posible a los usuarios de modo que disminuya el tiempo de su aprendizaje.

- Usar una interfaz de captura y reconocimiento de gestos ayudaría a vencer barreras de comunicación.

1.2. Rastreo de movimiento

La **captura o rastreo de movimiento** (*Motion Capture* o *Motion Tracking* en inglés) es el proceso de registrar el movimiento de objetos o personas; es utilizado en aplicaciones militares, de entretenimiento, deportivas o médicas, además incorpora y valida procesos relacionados con visión computacional y robótica. Cuando incluye movimientos de la cara y dedos o captura expresiones sutiles es referido como *performance capture*; en filmación y videojuegos es conocido como *match moving*[6].

Mediante la captura de movimiento se pueda generar un modelo digital que represente el cambio registrado en la posición de cierto(s) elemento(s) en la escena, para un análisis o aplicación posterior de los datos obtenidos.

El movimiento puede ser capturado y rastreado usando un sistema óptico o un sistema no óptico. Un sistema no óptico usualmente emplea sensores de inercia para medir tasas de rotación, las cuales son traducidas en movimientos corpóreos; los sistemas ópticos utilizan sensores de imagen y cámaras para rastrear movimientos en 3D.

Los sistemas ópticos con marcadores emplean elementos construidos con materiales pasivos (reflectivos) o activos (motorizados). Los marcadores son colocados en un traje o en el cuerpo del sujeto y rastreados empleando software de procesamiento de imágenes, actúan como puntos de referencia para la identificación de articulaciones particulares (joints) y extremidades mientras el sujeto se mueve (Figura 1.2). Estos dispositivos son comúnmente usados en proyectos a gran escala para la televisión y la industria fílmica.

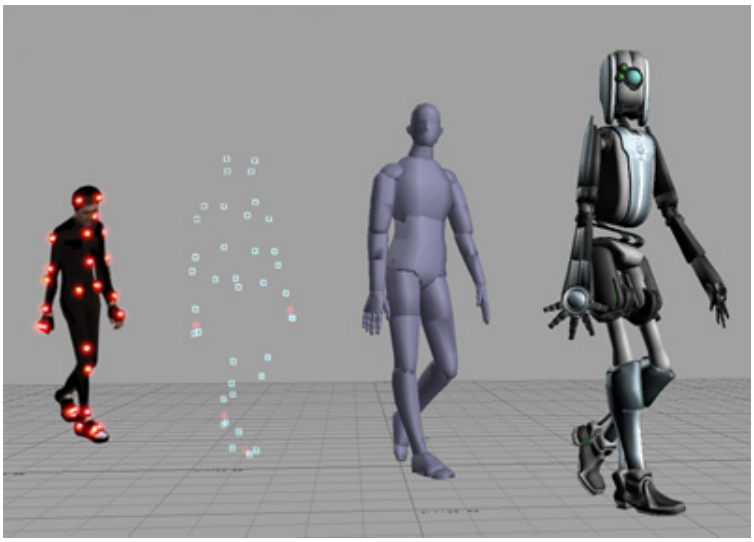


Figura 1.2: Traje con marcadores luminosos utilizados para rastreo de movimiento y proceso de generación de un modelo tridimensional asociado(1).

(1) https://en.wikipedia.org/wiki/Computer_animation

En el caso de los sistemas ópticos sin marcadores no se requiere que el sujeto utilice algún atuendo especial para realizar el rastreo, aquí los sensores ópticos se utilizan para determinar la profundidad de los objetos en el ambiente. Los sistemas infrarrojos facilitan la detección de la entrada de un sujeto en la escena para posteriormente identificar las articulaciones o extremidades del mismo.

1.3. Definición del problema

La finalidad primordial del proyecto de tesis consiste en implementar un método de *motion tracking* robusto, aplicando un sistema óptico multivisión, que permita el rastreo y recuperación de las posiciones de regiones de interés de las manos.

Para ello, el sistema empleará en general un conjunto de marcadores pasivos colocados en las manos, mismos que serán caracterizados para su detección y seguimiento y, finalmente, se llevará a cabo un análisis de los movimientos realizados (recuperación de posición, velocidad, aceleración, etc.) durante la ejecución de alguna destreza manual o gesto determinado.

1.3.1. Objetivos

El trabajo de tesis está dirigido por el diseño y construcción de tres módulos específicos:

- Los recursos y arquitectura del sistema multivisión para la captura de los movimientos manuales.
- Los algoritmos de detección y seguimiento de las regiones de interés correspondientes.
- El análisis y evaluación de los gestos simples asociados a ciertos movimientos.

En particular, los objetivos específicos son:

1. Diseño de la arquitectura del sistema multivisión.
2. Módulo de sincronización de múltiples dispositivos ópticos.
3. Diseño y construcción de los marcadores pasivos.
4. Adquisición de imágenes mediante el sistema multivisión.
5. Algoritmo de detección de regiones de interés.
6. Algoritmo de seguimiento de movimientos manuales.
7. Definición de experimentos para el rastreo de movimientos.
8. Captura de colección de imágenes de prueba y validación.
9. Análisis y evaluación de los movimientos manuales.
10. Pruebas con usuarios.

1.4. Estructura

El documento presentado está organizado de la siguiente manera: en el Capítulo 1 se enuncia una breve justificación de la intención y principales objetivos de la tesis. El Capítulo 2 corresponde al marco teórico, en el cual se describen generalidades sobre técnicas de captura y rastreo de movimiento, haciendo particularmente énfasis en propuestas para seguimiento de manos.

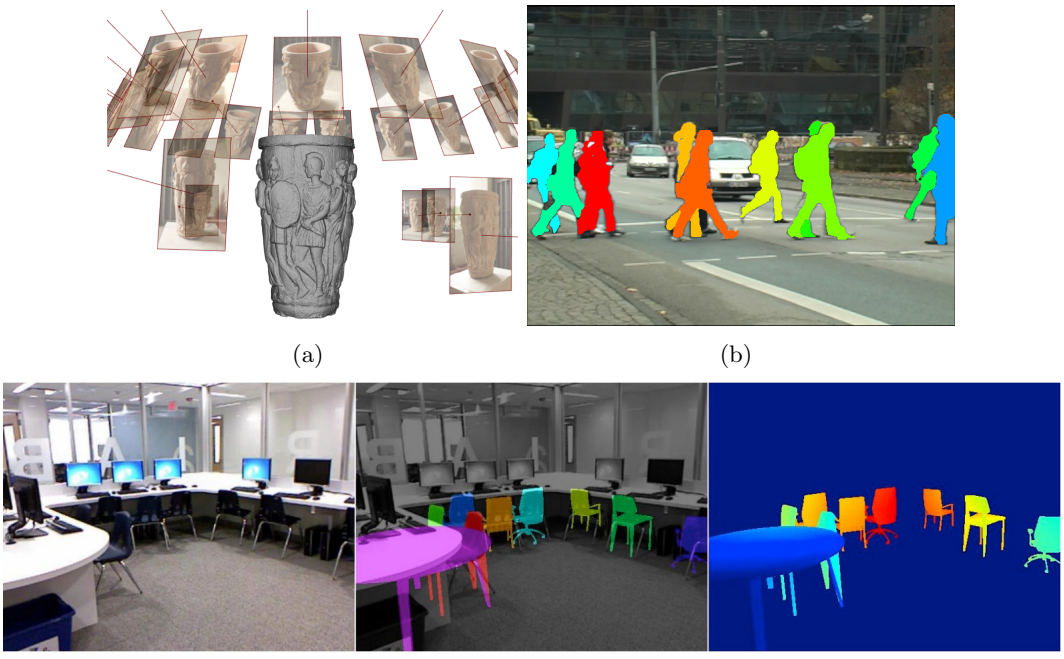
Dentro del Capítulo 3 se explica el diseño y desarrollo de los recursos, así como de la arquitectura del hardware usado y el software del sistema multivisión que será utilizado para hacer la detección de las regiones de interés y el rastreo de los movimientos manuales. Posteriormente, en el Capítulo 4 se muestran los experimentos y resultados obtenidos con el procesamiento hecho por el sistema construido.

Finalmente, en el Capítulo 5 se discuten las conclusiones acerca del desarrollo y los resultados correspondientes, además se listan las posibles mejoras y el trabajo a futuro.

Capítulo 2

Antecedentes

Los métodos de visión por computadora, comúnmente basados en el procesamiento y análisis de imágenes digitales, buscan emular el proceso de visión humano para caracterizar y describir el contenido de una imagen o de algunos de sus componentes (Figura 2.1). Sin embargo, los algoritmos dependen generalmente de las condiciones de la información de las capturas; por ejemplo, el ruido contenido en la imagen, la iluminación, las variaciones de color y contraste de los objetos, las formas de los mismos, entre otras.



(a) (b)

(c)

Figura 2.1: Aplicaciones comunes de visión por computadora: (a) estéreo visión y reconstrucción 3D[(2)], (b) rastreo de elementos en movimiento[(3)], (c) segmentación y reconocimiento de objetos[(4)].

(2) http://cs.bath.ac.uk/~nc537/opportunities.html (3) http://www.vision.ee.ethz.ch/~rhayko/

(4) https://people.eecs.berkeley.edu/~sgupta/

2.1. Algoritmos de segmentación

Normalmente, durante el procesamiento de las imágenes, los elementos encontrados corresponden a características estructurales tales como rectángulos, círculos, etc; o bien, a nivel de píxel como esquinas o bordes. Tomando en cuenta estos tipos de elementos se realiza la segmentación correspondiente, cuyas técnicas se dividen principalmente en dos grupos: las técnicas provenientes de la detección de contornos y aquellas que involucran crecimiento de regiones, entre las cuales se encuentran contornos activos, métodos de separación y mezclado, basados en gráficas y en funciones de energía, entre otros[7].

Una colección de algoritmos existentes mezclan ambas técnicas señaladas para obtener las regiones sobre una imagen; tales métodos utilizan las bases de segmentación por clústers y establecen el concepto de *superpíxeles*[8][9]. Un **superpíxel** puede definirse como un conjunto de píxeles que tienen características similares y que son usados como grupo para representar formas. Los superpíxeles (ver Figura 2.2) generalmente se emplean en segmentación por color, agrupando la redundancia de las imágenes y reduciendo la complejidad de las tareas de procesamiento posteriores.

Figura 2.2: Ejemplo de segmentación mediante superpíxeles.

De acuerdo a Achanta *et al.*[10], dentro de las características deseables de un buen algoritmo de segmentación por superpíxeles estarían las siguientes:

- Buen ajuste de los superpíxeles a las fronteras entre regiones.
- Si son empleados para reducir la complejidad computacional como un paso de

preprocesamiento, deben ser rápidos de calcular, utilizar memoria de forma eficiente y de uso simple.

- Cuando son usados para fines de segmentación, deben incrementar la rapidez de los algoritmos y mejorar la calidad de los resultados.

Asimismo, los métodos de segmentación capaces de producir superpíxeles pueden clasificarse en algoritmos basados en gráficas y los que utilizan ascenso del gradiente. Los primeros consideran cada píxel como el nodo de una gráfica y el peso de la arista entre dos nodos es proporcional a la similitud entre tales píxeles; los segmentos se obtienen minimizando una función de costo definida para la gráfica.

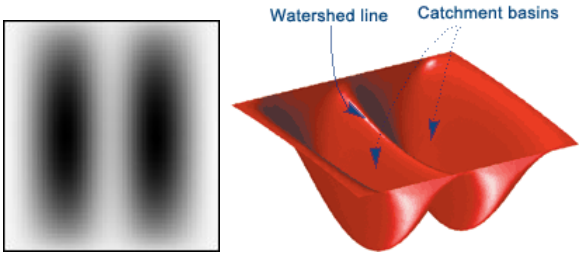
Por su parte, los algoritmos por descenso del gradiente ajustan clústers de iteraciones previas para obtener mejores segmentos hasta lograr el criterio de convergencia definido; dentro de éstos se encuentran Mean-Shift[11], Quick-Shift[12], Watershed[13], Turbopíxel[14] y SLIC[10], algunos de los cuales se describen en las siguientes secciones.

2.1.1. Vertientes

Una técnica propuesta por Vincent y Soille[13], con cierta similitud a crecimiento de regiones, es la segmentación basada en líneas divisorias de cuencas, mejor conocidas como vertientes (*watersheds*[1]), en la cual la transformación trata la imagen como un mapa topográfico utilizando el brillo de cada píxel para representar la altura en el mapa y encuentra las líneas formadas por las cimas de los crestas.

El método considera cada píxel como una gota de agua que desciende en un relieve topográfico hacia su mínimo más cercano, el cual se localiza al final del trayecto de la pendiente más pronunciada. La implementación de la transformada, introducida por Beucher y Meyer[15], simula un algoritmo de inundación sobre la imagen en escala de grises y mediante el gradiente de la misma, que finaliza al obtener diferentes vasijas inundadas que son separadas por las líneas de los bordes, de tal forma que cada vasija distinta corresponde a un segmento en la imagen (Figura 2.3).

[1] El nombre *watershed* proviene del concepto geológico que se refiere al límite de agua que desemboca y separa regiones de tierra (cuencas) vecinas.



(a) (b)

Figura 2.3: Representación gráfica del relieve de una imagen, el cual es usado para segmentar por vertientes[5]: (a) imagen sintética de dos manchas negras, (b) línea de vertiente.

(5) www.mathworks.com/company/newsletters/articles/the-watershed-transform-strategies-for-image-segmentation.html

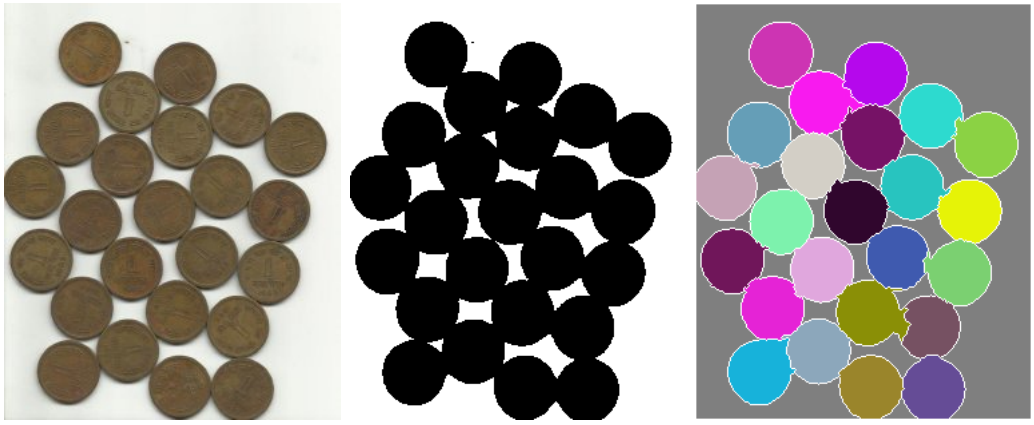
Para lograr la inundación se emplean puntos marcadores que serán los píxeles donde comenzará el proceso de inundación en cada región, los cuales pueden ser obtenidos como sigue:

- A partir de los mínimos locales del gradiente de la imagen; en este caso se produce una sobre-segmentación, por lo que es necesario aplicar un paso de mezcla de regiones.
- Los marcadores pueden ser especificados explícitamente por el usuario o determinados automáticamente mediate operadores morfológicos.

Los pasos del algoritmo de inundación se listan a continuación:

1. Se elige el conjunto de marcadores y a cada uno se le asigna una etiqueta.
2. Los píxeles vecinos de cada zona marcada se guardan en la cola de prioridad de acuerdo a la magnitud del gradiente del píxel.
3. Se extrae el píxel al frente de la cola de prioridad,
   a) si todos sus vecinos han sido marcados igual entonces se le asigna esa etiqueta al píxel.
   b) en caso contrario, todos los vecinos no marcados que no estén ya en la cola de prioridad son insertados.
4. Repetir el paso 3 hasta que la cola de prioridad quede vacía.

Como consecuencia de los pasos señalados, los píxeles no etiquetados son las líneas de vertientes; resultado sobre una imagen de ejemplo se muestra en la Figura 2.4.



(a) (b) (c)

Figura 2.4: Ejemplo de segmentación por vertientes: (a) imagen con objetos en contacto, (b) binarización estándar conteniendo elementos unidos, (c) separación de los objetos y líneas de vertiente.

2.1.2. Simple Linear Iterative Clustering (SLIC)

El método **SLIC**[10] es una adaptación de clusterización local de píxeles por *K-means*[16] para generación de superpíxeles de acuerdo a su similitud de color y proximidad en el plano de la imagen. Para ello, se utilizan tuplas en un espacio 5-D definido por los valores L, a, b en el espacio de color *CIELab* - perceptualmente uniforme para distancias de color pequeñas - así como las coordenadas x, y de los puntos.

El algoritmo realiza un paso de inicialización para obtener k centros de clúster $C_i = [l_i\ a_i\ b_i\ x_i\ y_i]^T$, muestreados sobre una rejilla regularmente espaciada por S posiciones, donde $S = \sqrt{N/k}$, con N el número de píxeles. Los centros se convierten en semillas de crecimiento de regiones, colocadas en las posiciones correspondientes a los píxeles con el gradiente más bajo en una vecindad de 3×3 para evitar centrar el superpíxel en un borde.

Cada i-ésimo píxel, en una región de traslape alrededor de cada centro C_k, se asocia al centro del clúster más cercano; ésto reduce el espacio de búsqueda y mejora la eficiencia del algoritmo reduciendo el número de distancias calculadas. La distancia D entre un centro de clúster y el píxel en proceso combina la distancia en el espacio de color d_c y la distancia euclidiana d_s, tal que

$$D = \sqrt{{d_c}^2 + \left(\frac{d_s}{S}\right)^2 m^2} \tag{2.1}$$

donde,

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2} \tag{2.2}$$

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{2.3}$$

Tomando la ecuación 2.1, el factor m pesa la importancia entre la similitud del color y proximidad espacial a partir de las expresiones 2.2 y 2.3, respectivamente. Si m es un número grande, los superpíxeles son más uniformes entre sí; mientras que para valores bajos de m los superpíxeles se ajustan mejor a las fronteras entre regiones, pero son menos regulares en tamaño y forma.

El algoritmo 1 resume los pasos de este método, mismo que requiere aplicar un paso final de componentes conexas para reunir aquellos píxeles aislados que no pertenecieron a la misma componente conexa del centro de clúster al que realmente corresponden.

Algoritmo 1 Algoritmo *SLIC* para segmentación por superpíxeles (Fuente: Achanta *et al.* [2012]).

Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular steps S
Move cluster centers to the lower gradient position in a 3×3 neighborhood
$\forall i \in I, l(i) = -1$, where i represents each pixel for the image I
$\forall i \in I, d(i) = \infty$
repeat
    for each cluster center C_k **do**
        for each pixel i in a $2S \times 2S$ region around C_k **do**
            Compute the distance D between C_k and i
            if $D < d(i)$ **then**
                set $d(i) = D$
                set $l(i) = k$
            end if
        end for
    end for
    Compute new cluster centers
    Compute residual error E
until $E \leq threshold$

En la Figura 2.5 pueden observarse superpíxeles de distintos tamaños sobre algunas imágenes usando SLIC.

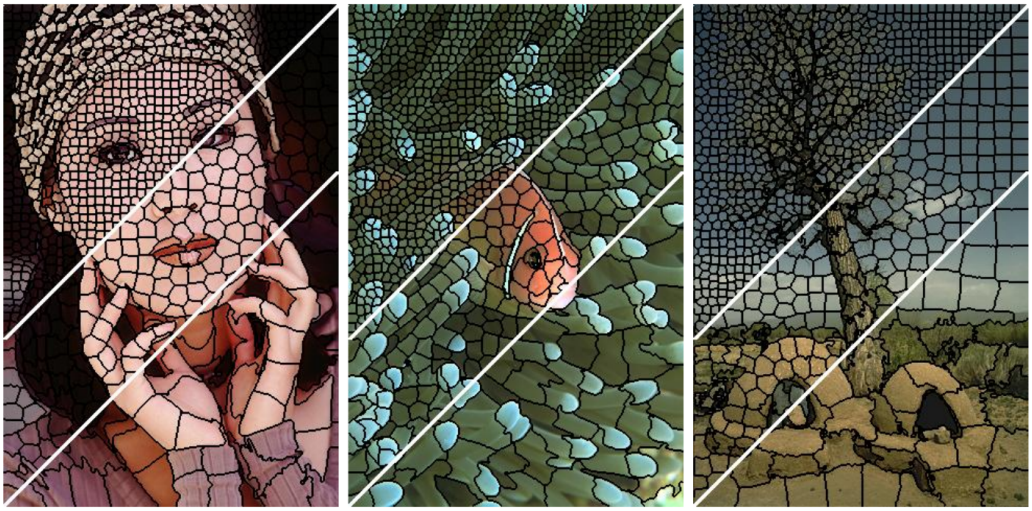


Figura 2.5: Segmentación por SLIC, mostrando superpíxeles de tamaño aproximado a 64, 256 y 1024 píxeles (Fuente: Achanta *et al.* [2012]).

2.2. *Motion Tracking*

Vemos que el proceso de detección de objetos en una imagen no es una tarea fácil, la cual adquiere mayor complejidad si tratamos con una secuencia de frames de un video almacenado o un flujo de datos adquiridos en vivo inclusive. Aquí, es necesario considerar variables adicionales sobre las imágenes resultantes que, además de problemas o posibles errores de origen tras la adquisición de los dispositivos, dependen de información relacionada al movimiento de los objetos.

Precisamente, el rastreo de objetos es un problema difícil debido principalmente a factores tales como la aparición, desaparición y oclusión de elementos, así como cambios de velocidad que pueden generar *blurring* y agregar variaciones de iluminación y presencia de sombras; o bien, el cambio de posición de los objetos, respecto a los dispositivos de captura, produce modificaciones en su forma, tamaño y rotaciones en escena (Figura 2.6(a)).

(a) (b)

Figura 2.6: Problemas presentes en el rastreo de objetos en movimiento: (a) desenfoque y oclusión de objetos[6], (b) desenfoque y cambios de iluminación en captura.

(6) https://www.viewbug.com/contests/motion-blur-photo-contest

En general, en la detección y análisis de objetos en movimiento, uno de los procesos más usados es el *flujo óptico* que consiste en la representación del cambio de posición a través de un campo vectorial, en el cual se muestra un patrón de movimiento aparente de puntos significativos entre frames consecutivos comparando las variaciones entre éstos. El flujo óptico considera que las intensidades de los píxeles de un objeto en particular no cambian entre frames consecutivos y las vecindades de tales píxeles tienen un movimiento similar (Figura 3.23).

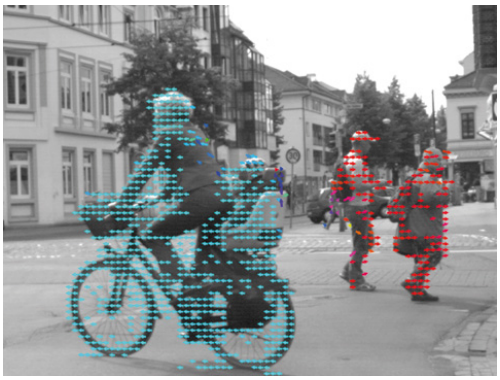


Figura 2.7: Representación de objetos en movimiento a través de flujo óptico[7].

(7) http://www.vision-systems.com/articles/2012/09/stemmer-imaging-readies-two-new-algorithms-for-vision-2012.html

El principio del flujo óptico, para el caso 2D sobre imágenes, toma la posición de un píxel (x, y, t) con intensidad $I(x, y, t)$ que se desplaza Δx, Δy y Δt entre los dos frames consecutivos, de modo que la constancia de brillo satisface la siguiente ecuación

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{2.4}$$

Asumiendo que el movimiento descrito es pequeño, la expresión 2.4 puede escribirse como

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + ... + \epsilon \tag{2.5}$$

tal que,

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0 \tag{2.6}$$

Dividiendo 2.6 por Δt y reescribiendo algunos valores se tiene

$$I_x u + I_y v + I_t = 0 \tag{2.7}$$

donde u y v con las componentes de la velocidad o flujo óptico de $I(x, y, t)$, mientras que I_x, I_y y I_t son las derivadas correspondientes $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial x}$ y $\frac{\partial I}{\partial x}$ en la imagen (x, y, t).

Los algoritmos estándar toman un conjunto de puntos a rastrear del frame en turno y obtienen los vectores de flujo óptico de esos puntos; sin embargo, eso funciona bien para movimientos pequeños y falla para cambios más largos. La solución propuesta por Lucas y Kanade[17] aplica el uso de pirámides de imágenes, subiendo en la pirámide para que los movimientos pequeños sean removidos y los movimientos largos se conviertan en pequeños.

El método Lucas-Kanade[18] utiliza una vecindad 3×3 alrededor de un píxel y se obtienen los valores I_x, I_y y I_t para cada uno de los 9 puntos; con éstos datos se aplica ajuste por mínimos cuadrados para calcular los valores solución u y v como se muestra en la ecuación 2.8

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \Sigma_i {I_{x_i}}^2 & \Sigma_i I_{x_i} I_{y_i} \\ \Sigma_i I_{x_i} I_{y_i} & \Sigma_i {I_{y_i}}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\Sigma_i I_{x_i} I_{t_i} \\ -\Sigma_i I_{y_i} I_{t_i} \end{bmatrix} \tag{2.8}$$

Otros métodos comunes consideran que en la escena estática los cambios de interés en los píxeles son generados por el cambio de posición de los objetos relevantes, por lo cual la idea es separar las características dinámicas en primer plano (*foreground*) del fondo fijo (*background*), a esta técnica se le conoce como *background subtraction* (Figura 2.8).

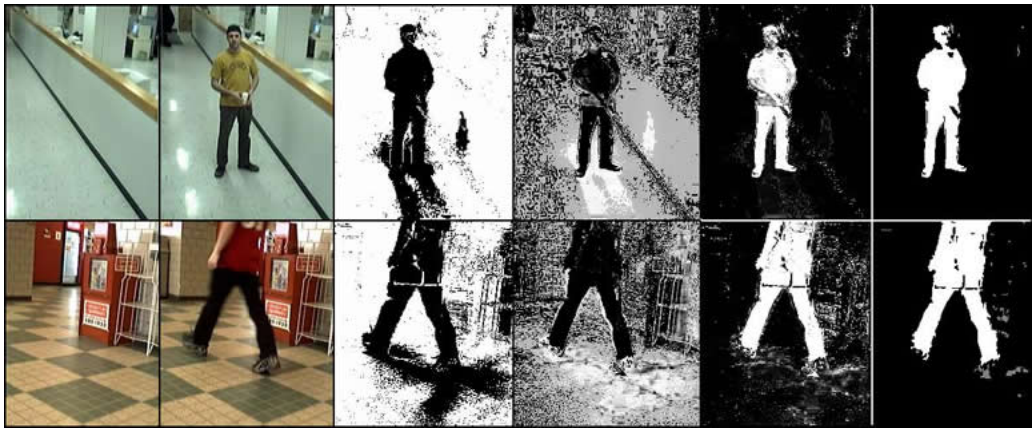


Figura 2.8: Sustracción de fondo en una escena con objetos en movimiento(8).

(8) https://www.cs.unc.edu/~lguan/Research.files/Research.htm

La mayoría de los algoritmos operan sobre cada pixel, utilizando mezcla de distribuciones gaussianas que son pesadas de acuerdo al tiempo proporcional de aquellos colores que aparecen en la escena, de manera que los colores de fondo probables son aquellos que permanecen estáticos y más tiempo[19].

2.3. Hand Tracking

Con respecto al rastreo de movimientos manuales, algunos de los principales problemas a resolver son la deformación, la interferencia de las condiciones del fondo de la escena y el cambio de tamaño de la mano. Por ello, muchos algoritmos están basados en la extracción de componentes de acuerdo al color, asumiendo que existen diferencias significativas entre el color de la piel de las regiones de interés - muñeca, palma y falanges - y el color de fondo. En consecuencia, es importante la selección de un espacio de color adecuado con independencia de iluminación para minimizar la interferencia de la luz en la extracción del color de la piel.

Otra forma de realizar la extracción del color de la piel consiste en un aprendizaje adaptativo en línea para aprender y actualizar iterativamente la distribución de los píxeles para determinar cuáles corresponden a zonas de piel de las manos y cuáles son fondo.

Considerando que la mayor causa de error en el rastreo se debe principalmente a la carencia de nivel de detalle expresivo de las características de las superficies bajo condiciones de deformación, autores como Zhang y Huang[20] sugieren clusterización iterativa SLIC (sección 2.1.2) para extraer características superpíxel de la(s) mano(s) (Figura 2.9) para mantener elementos estructurales en varios niveles.

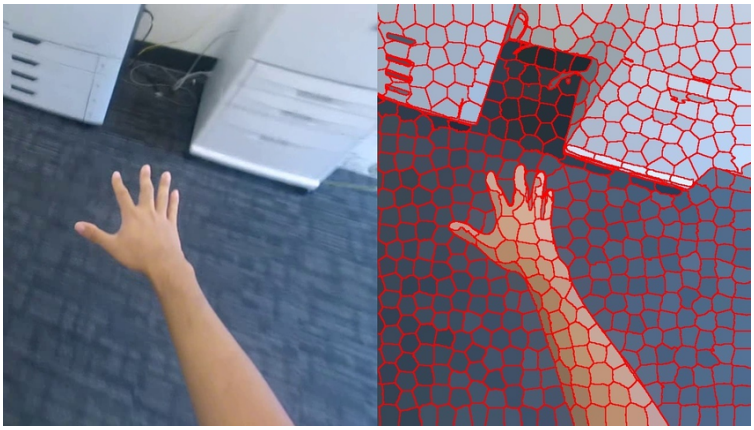


Figura 2.9: Superpíxeles de un brazo y mano calculados con SLIC[9].

(9) https://www.researchgate.net/publication/302954003_Hand-related_methods_in_Egocentric_Vision

Su algoritmo fusiona la extracción por color de la piel y los superpíxeles para mejorar la segmentación del área de la mano descartando el fondo (Figura 2.10). Posteriormente calcula un histograma de gradiente integrado (*gradient integrated histogram, GIH*), el cual es un tipo de vector de características que puede incorporar multiples orientaciones del gradiente y del histograma de intensidades. Finalmente, se utiliza una modificación del método *mean-shift*[21] para estimar la posición del objetivo (Figura 2.11).

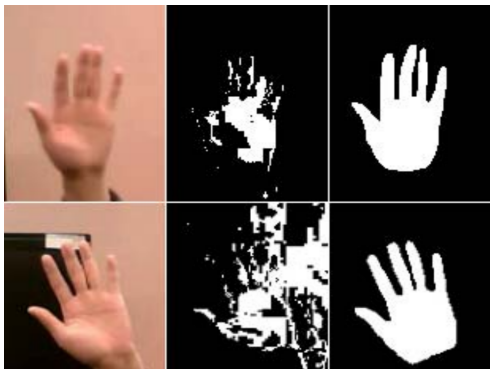


Figura 2.10: Resultado de la fusión de extracción de color y superpíxeles: imágenes de entrada (izquierda), extracción de color estándar (centro) y segmentación por método desarrollado por *Zhang y Huang* [2013].

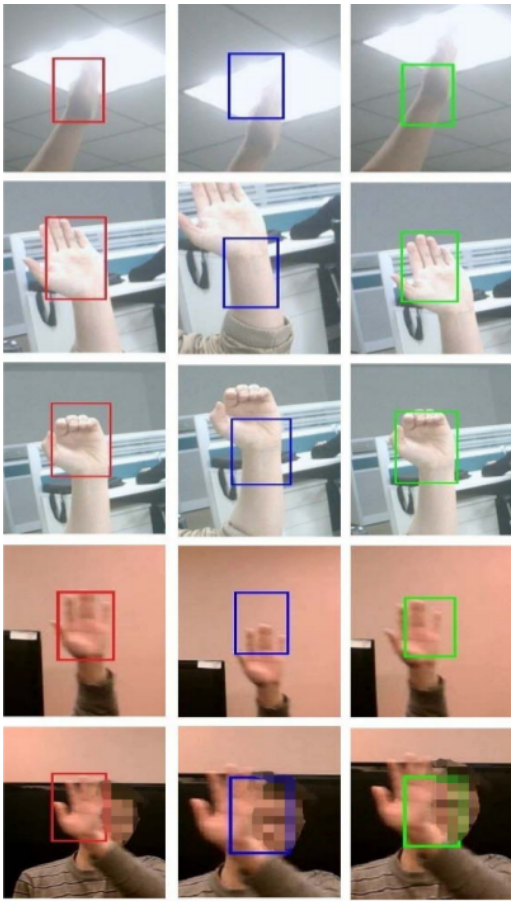
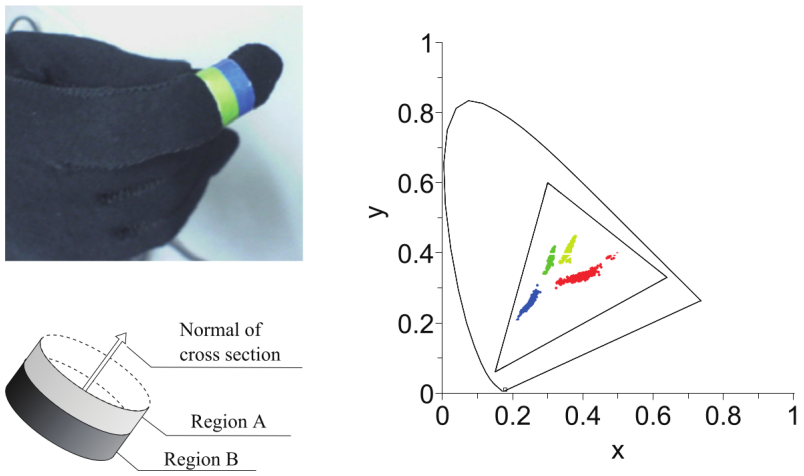


Figura 2.11: Comparación del método de rastreo propuesto por *Zhang y Huang* [2013] (izquierda), el método *mean-shift* (centro) y extracción estándar de color de la piel (derecha).

En algunos casos, también se han utilizado recursos adicionales para obtener un rastreo con mayor precisión de la posición de la mano; ejemplo de ésto es el trabajo realizado por Sasaki *et al.*[22] quienes emplean marcadores en forma de anillos con regiones de dos colores que al ser procesadas en el espacio de color xyY y bajo apropiadas condiciones de iluminación forman cúmulos de puntos debido a su similitud en tal espacio, cuya distribución no cambia fuertemente aún con variaciones de pose de los marcadores (Figura 2.12).



(a) (b)

Figura 2.12: (a) Marcadores pasivos en forma de anillos de colores y (b) la distribución de las regiones en el espacio de color xyY (Fuente: Sasaki *et al.* [2009]).

Una vez distinguidas las regiones de color correspondientes se extrae el borde definido por ambas, de tal forma que se pueda determinar el segmento de elipse visible en las capturas de las cámaras. La estimación de la elipse completa se calcula mediante ajuste por mínimos cuadrados, expuesto por Fitzgibbon *et al.*[23], al obtener el vector de parámetros $p = [a\ b\ c\ d\ e\ f]^T$ que satisfacen la ecuación 2.9 para todos los puntos.

$$p^T x = ax^2 + bxy + cy^2 + dx + ey + f = 0 \tag{2.9}$$

Para determinar el centro y vector normal de los marcadores se usa geometría proyectiva por estereo visión encontrando las dos elipses ajustadas sobre dos planos de imagen. Los centros de los anillos se calculan mediante la ecuación 2.10 mientras que el plano normal, en el cual está colocado el borde del círculo original, se expresa en las ecuaciones 2.11 y 2.12.

$$\mathbf{p}_c = -\begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix}^{-1} \begin{bmatrix} d/2 \\ e/2 \end{bmatrix} \tag{2.10}$$

$$\pi(\mu) = [\mathbf{e}'^T(\mathbf{H}(\mu) + (\mathbf{e}'^T\mathbf{C}'\mathbf{e}')\mathbf{A}), (\mathbf{e}'^T\mathbf{C}'\mathbf{e}')\|\mathbf{e}'\|^2]^T \tag{2.11}$$

$$\mathbf{H}(\mu) = [\mathbf{C}'\mathbf{e}']_x\mathbf{F} + \mu\mathbf{e}'(\mathbf{C}e)^T \tag{2.12}$$

donde $\mathbf{e}$ y $\mathbf{e}'$ con los epipolos, $\mathbf{F}$ la matriz fundamental, $[a]_x$ es la matrix antisimétrica[2] obtenida de a, mientras que μ es un parámetro dado por la ecuación 2.13 y $\mathbf{C}$ es la representación matricial de la cónica, que se muestra en la ecuación 2.14.

$$\mu^2[(\mathbf{Ce})(\mathbf{Ce})^T - (\mathbf{e}^T\mathbf{Ce})\mathbf{C}](\mathbf{e}'^T\mathbf{Ce}') = \mathbf{F}^T[\mathbf{C}'\mathbf{e}']_x\mathbf{C}'[\mathbf{C}'\mathbf{e}']_x\mathbf{F} \tag{2.13}$$

$$\mathbf{C} = \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix} \tag{2.14}$$

Por último, a partir de los centros y normales correspondientes, se utilizan restricciones anatómicas para estimar las posiciones $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ y $\mathbf{p}_4$ del dedo ilustradas en la Figura 2.13.

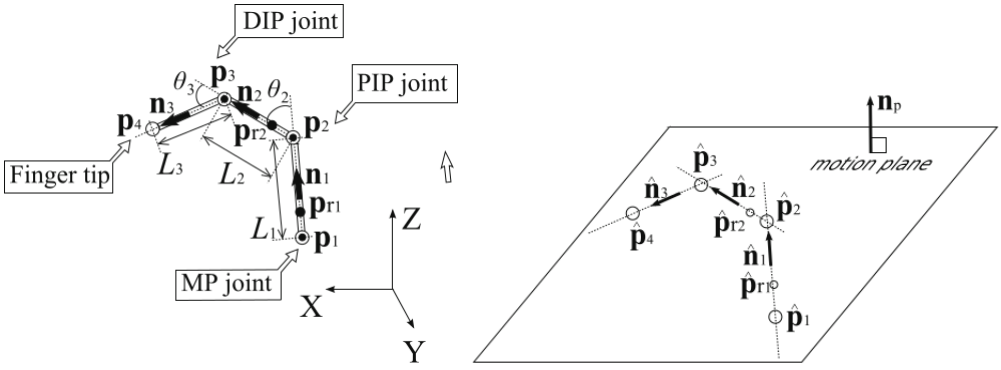


Figura 2.13: Restricciones anatómicas para estimar las posiciones del dedo: (a) modelo 3D del dedo, (b) restricción de plano de movimiento (Fuente: Sasaki *et al.* [2009]).

Por otro lado, en la literatura se reportan diversos tipos de métodos para realizar el rastreo de manos en tiempo real; en particular, Sridhar *et al.*[24] agrupa los algoritmos presentados en los últimos años, clasificándolos de la siguiente manera:

- **Generativos o basados en modelos**. Utilizan un modelo de la mano y sus articulaciones, usualmente un esqueleto cinemático, para estimar los parámetros de pose que mejor expliquen las observaciones de la imagen en turno. La estimación produce soluciones temporalmente suaves, pero requiere optimización local que en ocasiones converge a óptimos locales erróneos.

[2] Una matrix *antisimétrica* o *antimétrica* es una matrix cuadrada cuya transpuesta es igual al negativo de la matrix; i.e. se cumple la condición $A^T = -A$.

- **Discriminativos o por cuadros simples**. Asumen poco acerca de la coherencia temporal entre frames y utilizan modelos no paramétricos de la mano; las poses son inferidas mediante mapeo inverso de las características de la imagen hacia un espacio de configuraciones manuales existentes o aprendidas; sin embargo, los resultados son usualmente menos estables.
- **Híbridos**. Hace uso de un sistema multivisión y una cámara de profundidad; en este caso, el rastreo generativo está basado en una representación implícita de suma de gaussianas (*Sum of Gaussians, SoG*) y el rastreo discriminativo usa clasificadores para detectar las posiciones de los dedos.

El concepto de un modelo por suma de gaussianas se refiere a aproximar el volumen 3D de la mano a través de gaussianas isotrópicas[3] adjuntas a los huesos del esqueleto, asociando un color a cada gaussiana. En 2D, las imágenes son segmentadas en regiones de colores y cada región es aproximada por una gaussiana en el plano. El problema con una aproximación isotrópica es que se requieren muchas funciones gaussianas con pequeñas desviaciones estándar lo cual incrementa la complejidad computacional. Por esta razón, su modelo, denominado SAG (*Sum of anisotropic gaussians*), extiende SoG representando la forma de la mano en 3D mediante gaussianas *anisotrópicas*[4] , como se observa en la Figura 2.14.

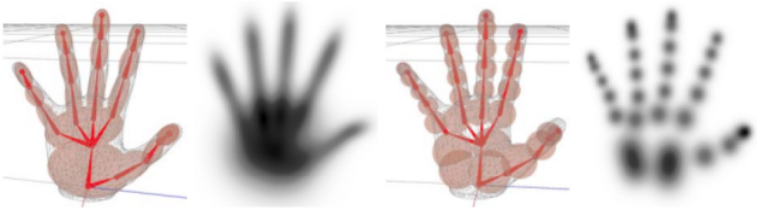


Figura 2.14: Resultado del modelo utilizando gaussianas anisotrópicas y con primitivas isotrópicas (Fuente: Sridhar *et al.* [2014]).

El uso de gaussianas anisotrópicas les permite establecer una función de ajuste de energía suave y que es analíticamente diferenciable para el modelo SAG bajo proyecciones de perspectiva de las cámaras a color empleadas y que permite optimización con un resolvedor iterativo basado en gradiente. El resumen del flujo del método se muestra en la Figura 2.15.

[3] Una gaussiana *isotrópica* es aquella en la cual su matrix de covarianza Σ es proporcional a la matrix identidad, ésto es $\Sigma = \sigma^2 I$.

[4] El filtrato *anistrópico* fue presentado por Perona y Malik[25] y es un filtro que elimina los efectos de dentado en las curvas o líneas diagonales de una imagen, reducciendo el suavizado y preservando detalles de los bordes.

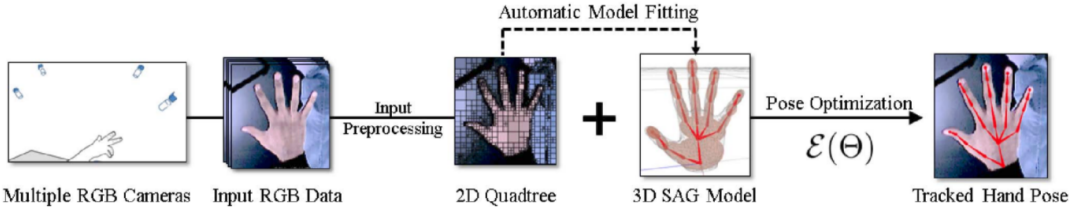


Figura 2.15: Método de rastreo de manos basado en suma de funciones gaussianas anisotrópicas (Fuente: Sridhar *et al.* [2014]).

2.4. Kinect

Motion tracking es un área de investigación muy importante actualmente, especialmente provocada por las grandes compañías desarrolladoras de videojuegos (Microsoft, Nintendo y Sony). Todos ellos han liberado consolas, a un costo relativamente bajo, con capacidades para rastreo de movimiento haciendo un trabajo razonable de captura.

El Kinect[26] es un dispositivo óptico desarrollado por Microsoft para la consola Xbox 360 (v1, Noviembre 2010) y para el Xbox One (v2, 2014) (Figura 2.16). Posee un emisor infrarrojo y una base de inclinación variable, además de un sensor infrarrojo de profundidad y un sensor de color, así como un arreglo lineal de 6 micrófonos.

Figura 2.16: Dispositivo infrarrojo Kinect v2.

2.4.1. Rastreo de gestos manuales

Aunque el Kinect fue diseñado originalmente para consolas de videojuegos y Windows PC's, posee su propio kit de desarrollo de software[27], habilitándolo como una tecnología de captura de movimiento 3D ampliamente utilizado en diversas áreas de investigación hoy en día.

Precisamente, dentro de las aplicaciones más frecuentes del Kinect se encuentra la interacción con sistemas de cómputo por medio de gestos prácticos manuales que permiten manejar elementos como cursores, controles de juegos, televisores de forma remota, entre otros. La información proporcionada por el Kinect puede ser utilizada para sobrellevar problemas de iluminación no uniforme en el ambiente, los distintos tonos de piel en las personas y las rotaciones de la mano durante la ejecución de algún movimiento.

Bakar *et al.*[28] desarrollaron un método de rastreo de manos invariante a rotación aprovechando el sensor de profundidad del Kinect. Un primer paso de inicialización busca el gesto de ondeo de la mano utilizando *NiTE*[5]. Cuando la posición de la mano es localizada y rastreada se define la región de interés alrededor del área correspondiente para evitar detectar que otros objetos moviéndose a la misma distancia sean considerados dentro del área indicada.

El problema con el movimiento es que el usuario puede desplazar la mano hacia el Kinect y, por tanto, el tamaño del rectángulo de la ROI no puede ser fijo; por ello, definen el contorno de la mano como el inverso de la distancia z_{depth} al sensor del dispositivo de modo que, como señala la ecuación 2.15, a una distancia de 1000 mm el tamaño del cuadro sea 80 para incluir adecuadamente la palma.

$$size_{square} = 1000 * 80/z_{depth} \tag{2.15}$$

Con el autoajuste del cuadro de la región de interés (Figura 2.17) se realiza el umbralizado de la mano utilizando los valores de profundidad de cada píxel y, tomando en cuenta que las manos están usualmente hacia el frente del cuerpo, se descartan elementos o ruido con colores similares.

[5] **NiTE**[29] es un *middleware* 3D de visión por computadora que utiliza los datos recolectados por un dispositivo de captura de imágenes para realizar tareas de localización y rastreo de manos, así como reconocimiento de algunos gestos.

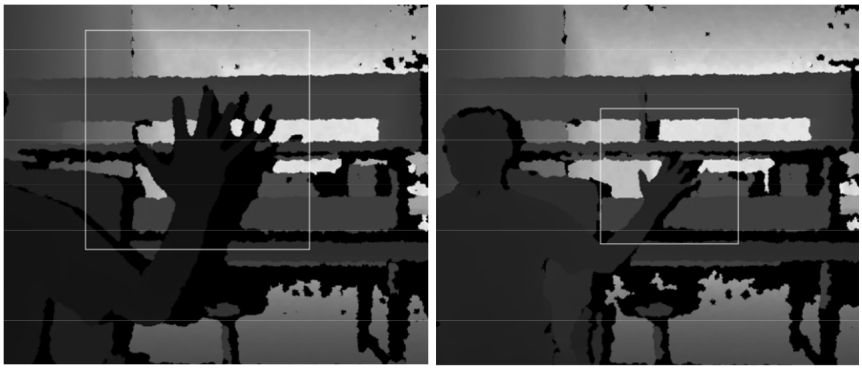
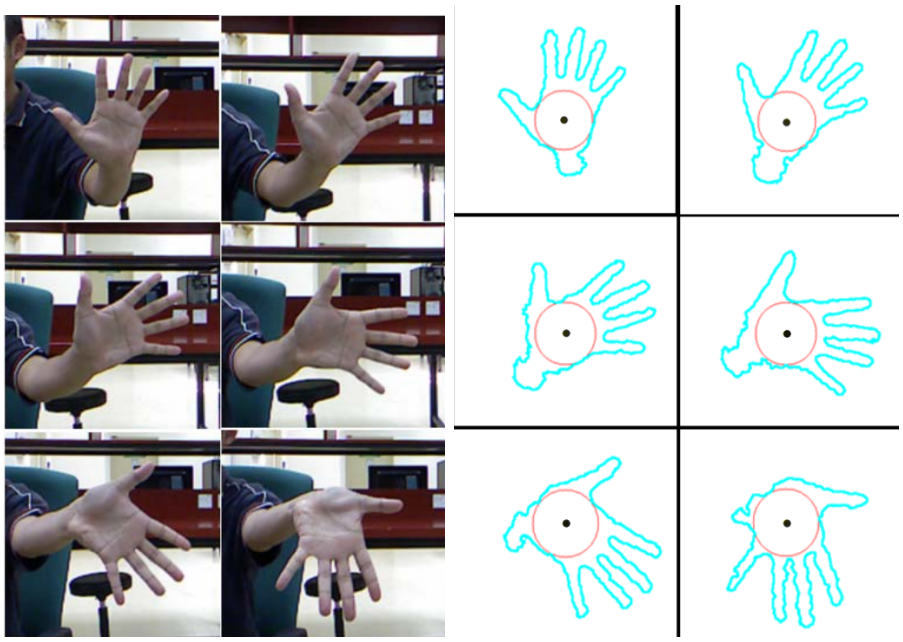


Figura 2.17: Autoajuste del tamaño del cuadro de la ROI: (a) la mano está cerca del sensor y (b) la mano es alejada del sensor (Fuente: Bakar *et al.* [2014]).

El último paso, descrito en el algoritmo 2, encuentra el radio y centro de la mano calculando la circunferencia contenida en la palma. Algunos resultados del rastreo invariante a rotación se ilustran en la Figura 2.18.

Algoritmo 2 Algoritmo para determinar el círculo aproximado de la palma (Fuente: Bakar *et al.* [2014]).

1: Set palm as: $Circle(P(0,0), r = -1)$
2: Start with point at the top left of tracking square
3: Loop through every 4th pixel, from top to bottom, left to right
4: Measure distances from current pixel to every 4th pixel on hand contour
5: Record the contour point with minimum distance, let them be point P and distance D
6: **if** $D > r$ **then**
7: palm is now: $Circle(P, D)$
8: **else**
9: continue
10: **end if**



(a) (b)

Figura 2.18: (a) Secuencia de rotación de la mano y (b) rastreo de mano invariante a rotación con detección del centro de la palma (Fuente: Bakar *et al.* [2014]).

Para mejorar la detección y rastreo de los gestos manuales, Usachokcharoen *et al.*[30] proponen aprovechar tanto el sensor de color como de profundidad utilizando guantes de colores especialmente diseñados (Figura 2.19), recuperando características de movimiento, profundidad y color para construir una representación de descriptores 3D de las manos y aplicarlo al reconocimiento de gestos.

Figura 2.19: Guantes de 12 colores propuestos por Usachokcharoen *et al.* [2015].

La ROI de profundidad se consigue mediante umbralizado de la zona con distancias cercanas a la mano aplicando un histograma de intensidades, en el cual el rango 0-60 fue dividido en espacios más pequeños correspondientes al cuerpo humano (Figura 2.20).

Figura 2.20: Proceso de obtención de la ROI de profundidad (Fuente: Usachokcharoen *et al.* [2015]).

Para la extracción de la ROI de movimiento, compararon dos frames de profundidad consecutivos para identificar las diferencias y asignar el área donde el movimiento se presentó. Las características de movimiento consisten en el número de píxeles en cada celda de una regilla pre-establecida (Figura 2.21).

Figura 2.21: Proceso de localización de la ROI de movimiento (Fuente: Usachokcharoen *et al.* [2015]).

Finalmente, con las posiciones de los píxeles de la mano recuperadas de la segmentación del frame de profundidad, se obtiene la región de los guantes que es procesada para remover aquellos puntos con colores fuera de los rangos de los 12 colores (Figura 2.22).

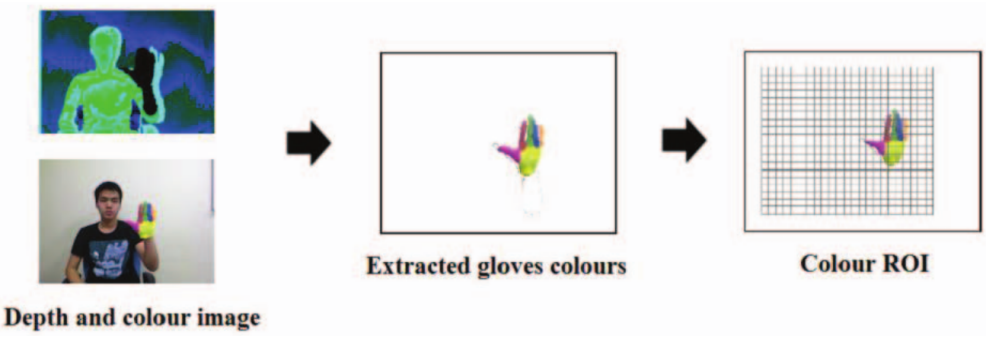


Figura 2.22: Proceso de obtención de la ROI de color (Fuente: Usachokcharoen *et al.* [2015]).

Capítulo 3

Diseño y Desarrollo

Con base en la revisión de las características y aplicaciones de los métodos descritos en el capítulo 2, en las siguientes secciones se detallan cada uno de los componentes del sistema y el método de tracking desarrollado en este trabajo, los cuales consideran restricciones particulares para la realización de algunas destrezas manuales específicas en condiciones controladas, tal como se muestra en las pruebas realizadas del capítulo 4.

Tomando en cuenta estas observaciones, a continuación se explican las especificaciones de la arquitectura multivisión propuesta, así como los algoritmos de preprocesamiento, segmentación y rastreo de los marcadores; el análisis de movimientos correspondiente será explicado en los resultados del capítulo 4.

3.1. Adquisición de imagen

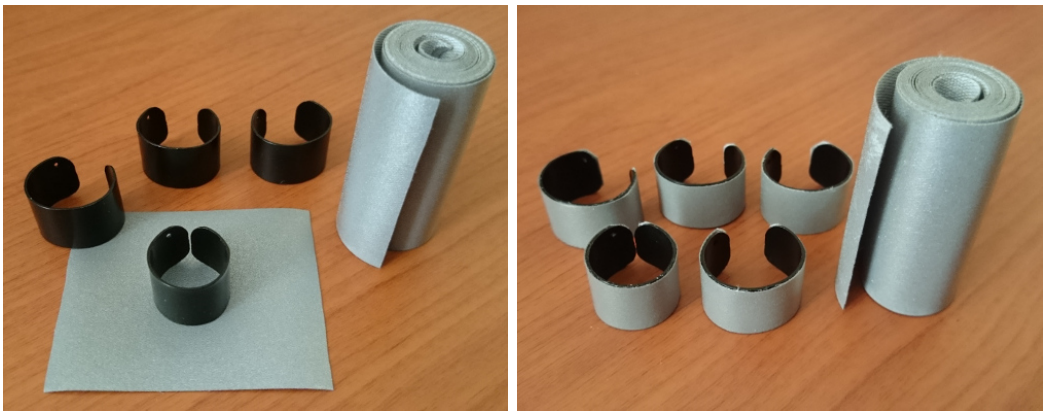
La primera parte consiste en obtener una colección de imágenes adecuadas para desarrollar los métodos de detección y rastreo. Para la captura de frames se utilizan elementos con colores contrastantes entre sí que permitan distinguir el fondo y las regiones de interés de las manos; ejemplo de ello son las superficies sobre las cuales se colocan las manos y se realizan los movimientos (Figura 3.1(a)).

Asimismo, un arreglo de Kinects v2 se coloca alrededor de la superficie de prueba (Figura 3.1(b)) y se almacenan los 3 tipos de imágenes que son proporcionados por cada dispositivo; la disposición del arreglo de los Kinects se describe en la sección 3.2.

(a) (b)

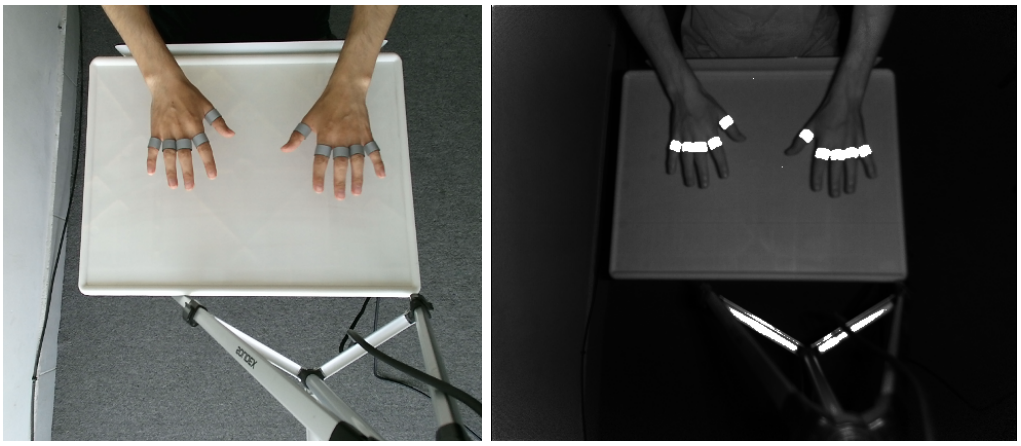
Figura 3.1: Características de las condiciones de laboratorio para la adquisición de imágenes: (a) tipo y color de superficies sugeridas, (b) arreglo de kinects.

Para la implementación se consideró el uso de marcadores pasivos en forma de anillos, que son colocados sobre las primeras falanges de los dedos de ambas manos. El conjunto de anillos de metal ajustables fue cubierto con cinta reflejante (Figura 3.2) con la finalidad de mejorar la precisión durante el proceso de segmentación de tales marcadores. Por sí mismo, el color de la cinta es distinguible del tono de la piel, en tanto que el material reflejante satura el sensor infrarrojo y los marcadores aparecen como regiones brillantes en las imágenes capturadas (Figura 3.3).



(a) (b)

Figura 3.2: Marcadores reflejantes para el sensor infrarrojo: (a) anillos ajustables para cada dedo, (b) los anillos son cubiertos con la cinta reflejante.



(a) (b)

Figura 3.3: Imágenes de las manos con marcadores reflejantes; (a) frame RGB de la captura y (b) imagen infrarroja recuperada.

Adicionalmente, se utilizaron guantes de látex en algunas capturas con el fin de determinar si ello permitía obtener un mejor contraste en la detección de los marcadores (Figura 3.4).

Figura 3.4: Guantes utilizados como recursos complementarios para el sistema de captura.

3.2. Arquitectura multivisión

El sistema utiliza un arreglo de múltiples Kinects conectados a un mismo equipo, para lo cual se requiere que éste último posea varios canales usb para la tranmisión de datos, de modo que las distintas vistas permitan proponer una solución para la oclusión de algún marcador respecto a determinada cámara (Figura 3.5).

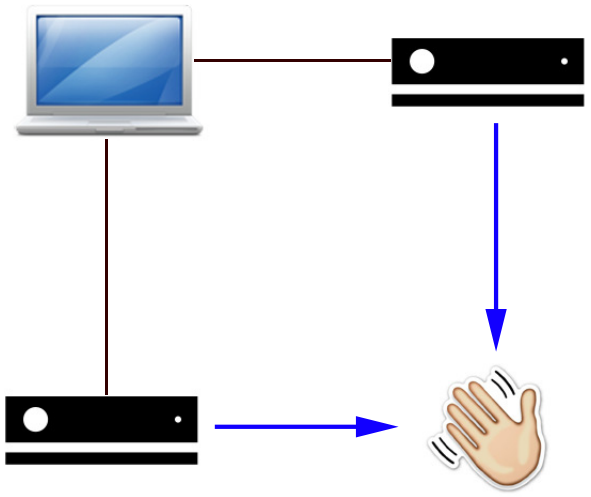


Figura 3.5: Múltiples dispositivos conectados a un mismo equipo.

Uno de los componentes más importante del sistema multivisión es el software de captura, construido con la biblioteca *libfreenect2*[31], que administra la conexión y comunicación con múltiples dispositivos. Para ello, el programa principal obtiene los números de serie de los Kinects e instancia un objeto para identificación, preparación y captura del dispositivo, proceso que es llevado a cabo por un hilo de ejecución de la biblioteca multi-threading de *boost*[32].

En cada iteración, y por cada instancia de captura, los frames son guardados en un búfer circular; una vez alcanzado el número de frames especificados se sincronizan los threads y, finalmente, se almacenan en disco las imágenes contenidas en los búfers. El proceso de adquisición por software se resume en los Algoritmos 3 y 4.

Algoritmo 3 Programa multivisión principal a través de sincronización de hilos.

```
1: Se verifica la conexión con los dispositivos
2: ksConnected.getSerials()
3: ksConnected.createDevices()
4: for i = 0 : num-dispositivos do
5:     deviceThreads.add(thread(ksConnected[i]))
6: end for
7: for i = 0 : num-dispositivos do
8:     deviceThreads.join()
9: end for
10: ksConnected.saveData()
```

Algoritmo 4 Función de captura por hilo de ejecución.

```
1: dataM[id].setCapacity()
2: while framecount < framemax do
3:     frames ← listener.waitForNewFrame()
4:     rgb ← frames[Frame::Color]
5:     iR ← frames[Frame::Ir]
6:     dP ← frames[Frame::Depth]
7:     imData.RGBM ← rgb
8:     imData.IRM ← iR
9:     imData.DPM ← dP
10:    dataM[id].add(imData)
11: end while
12: dataM[id].stop()
13: dataM[id].close()
```

3.2.1. Calibración

En general, la identificación de los Kinects es indistinta; sin embargo, se establece la posición de uno de éstos como origen del sistema coordenado; así, la ubicación de los otros puede obtenerse por visión estéreo tomando las imágenes del dispositivo de referencia y de cada una del resto de cámaras por separado (Figura 3.6).

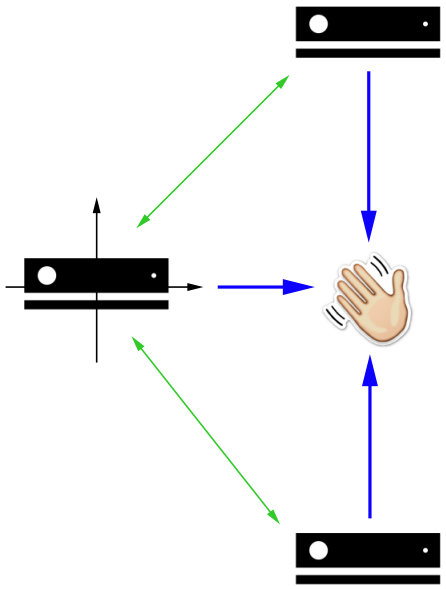


Figura 3.6: Ejemplo de sistema coordenado del sistema multivisión.

Para ello, el proceso de calibración se realiza capturando diferentes imágenes del patrón de cuadros del tablero de ajedrez (Figura 3.7); posteriormente, la obtención de la matriz fundamental y otros parámetros estéreo se hace mediante la herramienta de calibración de *Matlab*[33] empleando los datos de las cámaras correspondientes (Figura 3.8).

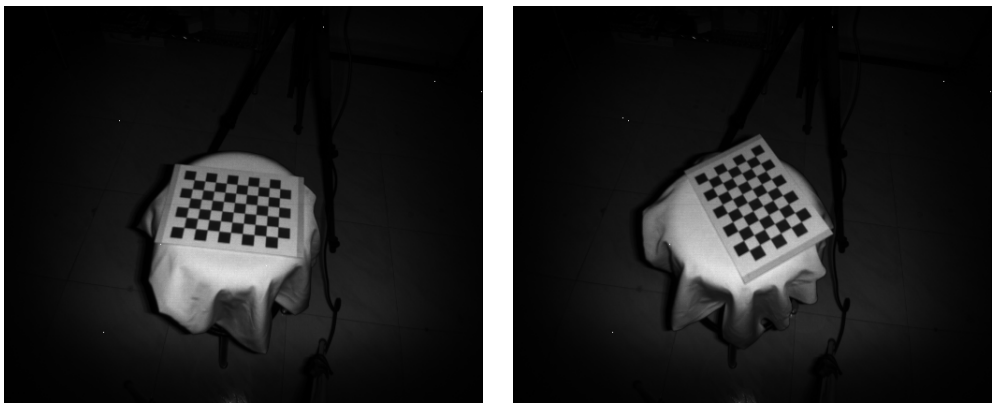


Figura 3.7: Distintas vistas del patrón de calibración con una misma cámara.

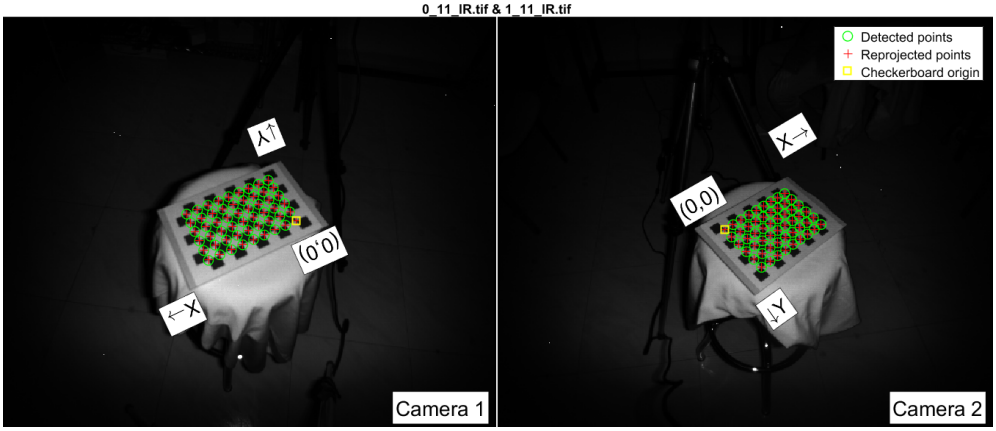


Figura 3.8: Calibración de dos capturas con múltiples Kinects.

Ejemplo del error medio de reproyección obtenido para dos de los dispositivos utilizados se muestra en la Figura 3.9, en la cual se muestran los pares de imágenes capturadas con ambas cámaras; las barras verticales en colores azul corresponden a las imágenes provenientes de la primer cámara, mientras que el error medio en píxeles para la segunda cámara se visualiza en colores naranja.

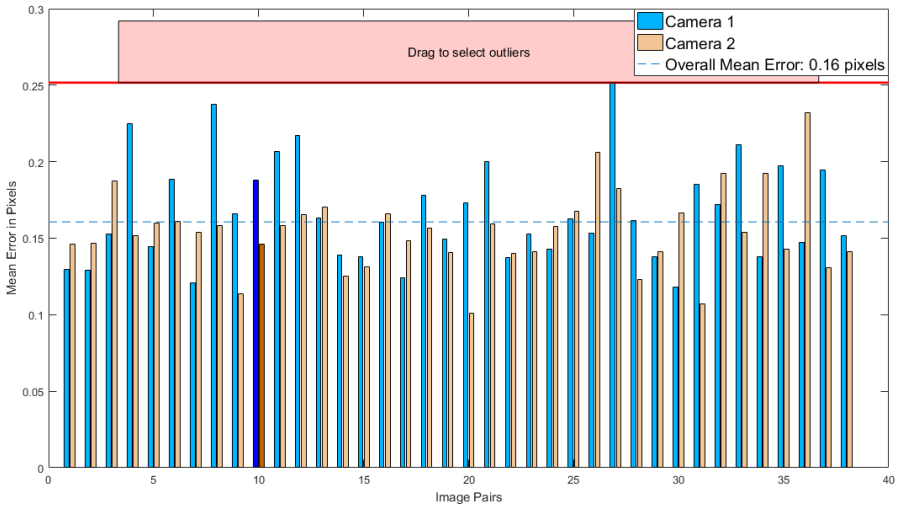


Figura 3.9: Error medio de reproyección obtenido en la calibración de dos dispositivos para distintas capturas.

La representación de los parámetros extrínsecos de los dispositivos puede observarse en la Figura 3.10 y la colocación de las cámaras respecto a la posición fija del patrón calibrado es mostrada en la Figura 3.11.

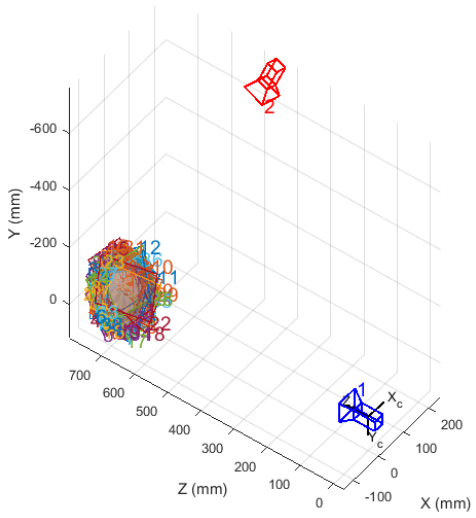


Figura 3.10: La ubicación del patrón de calibración en relación a las cámaras fijas.

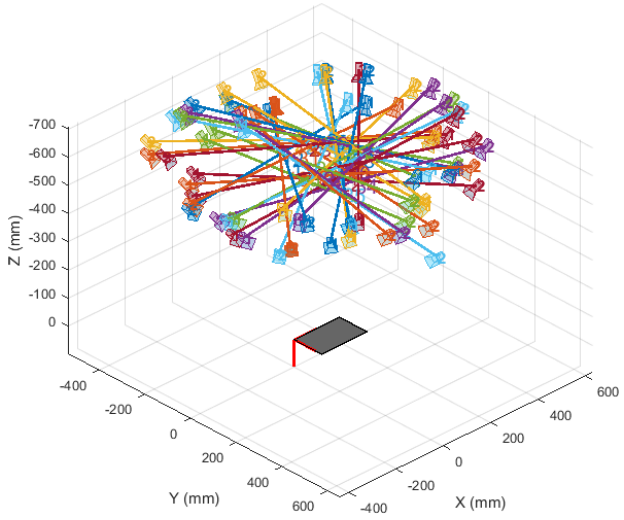


Figura 3.11: El patrón calibrado con dos de los dispositvos.

3.3. Método de detección

Para realizar eventualmente una adecuada evaluación de los movimientos se deben determinar las posiciones de los marcadores en cada una de los frames correspondientes; además, se requiere que el sistema pueda discernir cuál marcador es cada uno.

Durante el proceso se podrían realizar desplazamientos de forma arbitraria, pero se recomienda empezar con una pose fija con las palmas extendidas sobre la superficie de prueba, ya que la finalidad es segmentar correctamente, desde un inicio, todos los marcadores de ambas manos y asociar sus centros a sus zonas respectivas en la nube de puntos capturada con el sistema multivisión.

3.3.1. Pre-procesamiento

El procesamiento previo se hace sobre las imágenes infrarrojas, el gráfico de los histogramas nos muestra que el uso de éstas permite recuperar las regiones de interés deseadas (Figura 3.12), aunque se necesita un proceso de limpieza de posibles artefactos luminosos en la escena.

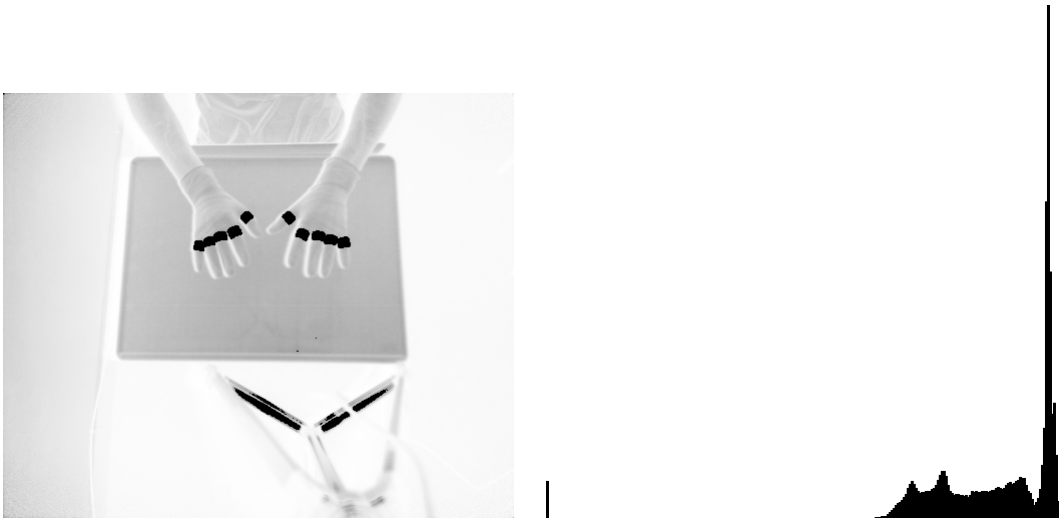


Figura 3.12: Histograma de la imagen en grises invertida.

El algoritmo de recuperación de componentes válidas se describe a continuación:

1. Un umbralizado estándar es utilizado para descartar los elementos en el fondo; particularmente, al emplear la cinta reflejante, sólo los marcadores y aquellos objetos o partes de ellos que aparecen brillantes satisfacen la binarización (Figura 3.13).

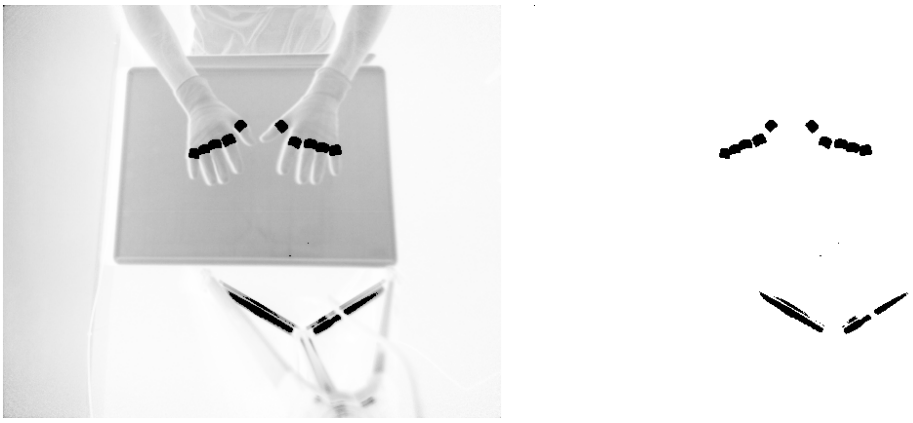


Figura 3.13: Aplicación de umbralizado estándar.

2. Para eliminar algunos puntos de luz o elementos pequeños se aplica un filtro de erosión (Figura 3.14).

(a) (b)

Figura 3.14: Filtrado de elementos pequeños: (a) imagen binaria con píxeles aislados y (b) erosión de componentes.

3. Se obtienen las componentes conexas de la imagen binaria original y de la imagen erosionada del paso anterior para asociar los puntos entre ambas imágenes, tal forma que aquellas componentes sin correspondencia son descartadas de la imagen binaria.

(a) (b)

Figura 3.15: Asociación entre componentes conexas: (a) componentes originales de la imagen binaria y (b) remoción de componentes inválidas marcadas en rojo.

4. Por último, se escalan al doble la imagen de entrada y la binaria resultante del último paso para obtener una máscara adecuada en el proceso de segmentación.

3.3.2. Segmentación

El método de detección se realiza por cada frame para recuperar con mayor precisión las posiciones de los centros de los marcadores utilizados.

Las regiones de interés del último paso del preprocesamiento son separadas y etiquetadas logrando la extracción de los marcadores individuales. Los algoritmos de segmentación utilizados y evaluados en este trabajo se detallan en los siguientes bloques.

✳ **WATERSHED**

1. Se construye una máscara binaria que representa el fondo de los objetos a separar, la cual se consigue aplicando un filtro de apertura seguido de una dilatación (Figura 3.16). La idea es que píxeles de regiones cerca de los centros de los objetos están en primer plano, mientras que los más alejados son fondo.

(a) (b)

Figura 3.16: Máscara de imagen de fondo de las componentes: (a) la apertura remueve zonas ruidosas de los objetos y (b) la dilatación genera regiones correspondientes al fondo de los mismos.

2. La imagen en primer plano se extrae usando transformada de distancia sobre la apertura y posteriormente un umbralizado apropiado para remover los bordes de los objetos (Figura 3.17).

(a) (b)

Figura 3.17: Imagen de los objetos en primer plano: (a) la apertura se emplea para la transformada de distancia; (b) remoción de los bordes de los elementos de interés.

3. A continuación, se realiza la resta de la imagen de fondo de la Figura 3.16(b) y la imagen de primer plano del paso anterior; el resultado es una imagen de crestas aumentadas que se muestra en la Figura 3.18.

Figura 3.18: Resultado de sustracción de fondo con respecto a los objetos de interés en primer plano.

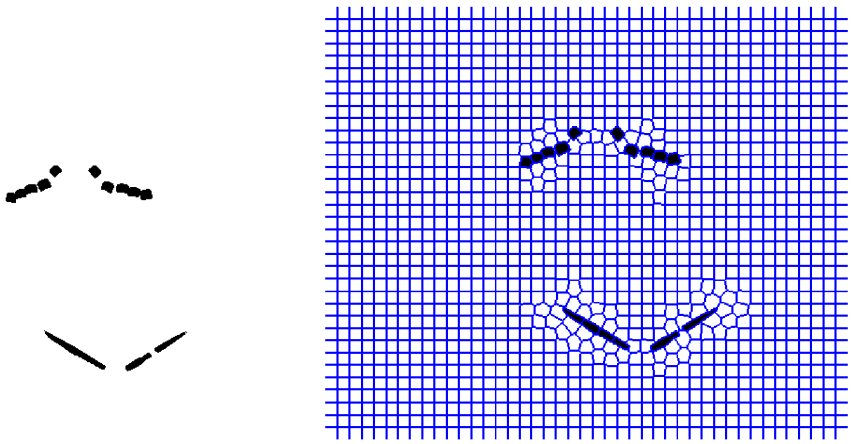
4. Tomando la imagen de crestas generada en el paso previo se aplica el algoritmo de *watershed* (vertientes) para conseguir la segmentación de los marcadores (Figura 3.19).

Figura 3.19: Resultado de la segmentación aplicando *watershed*.

✳ **SLIC**

1. Al igual que la segmentación por vertientes, se toma la imagen binaria del último paso del procesamiento y se obtienen las áreas de las componentes válidas para calcular un área promedio A, tal que $N = M/A$ es el número de superpíxeles aproximados que se pueden generar, donde M es el número total de píxeles en la imagen de entrada.

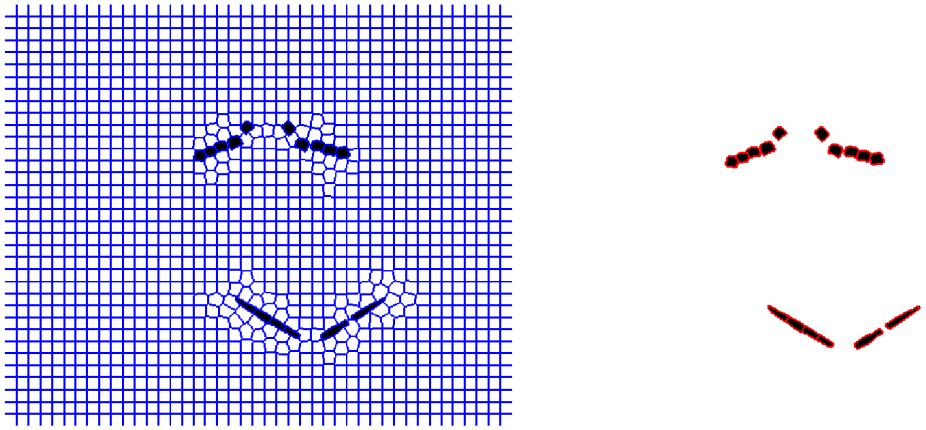
2. Aplicando el algoritmo *SLIC* se consiguen los superpíxeles que particionan la imagen binaria correspondiente, como se muestra en la Figura 3.20.



(a) (b)

Figura 3.20: Algoritmo de *SLIC* sobre la imagen: (a) imagen binaria de componentes válidas, (b) generación de superpíxeles.

3. Posteriormente, se realiza un filtrado de aquellos superpíxeles asociados a las manchas negras de las componentes válidas (Figura 3.21).



(a) (b)

Figura 3.21: Filtrado de superpíxeles calculados: (a) todos los superpíxeles generados por *SLIC*, (b) los superpíxeles asociados a las componentes válidas.

Tras la segmentación de las componentes, se ordenan de izquierda-derecha y arriba-abajo a partir de su ubicación en la imagen y se asigna una etiqueta de identificación.

3.4. Método de rastreo

Para realizar el rastreo durante el movimiento de los dedos de las manos se hará uso de la nube de puntos recolectada con los múltiples Kinects, caracterizando cada uno de los marcadores y determinando su posición espacial en cada captura posterior.

1. El primer paso tras la segmentación es determinar los centros de cada uno de los marcadores (Figura 3.22), los cuales son proporcionados como parámetros para el método de flujo óptico por pirámides de Lucas-Kanade de la biblioteca *OpenCV* [34] (Figura 3.23).

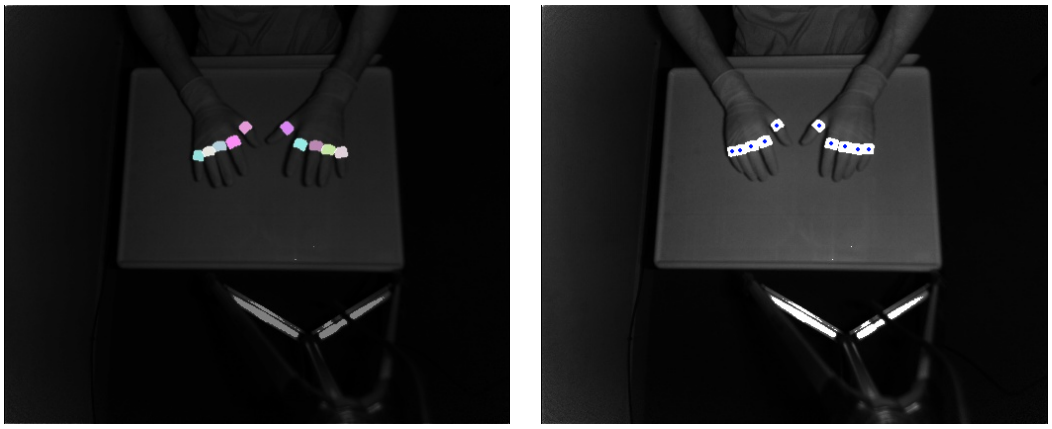


Figura 3.22: Centroides de los marcadores segmentados.

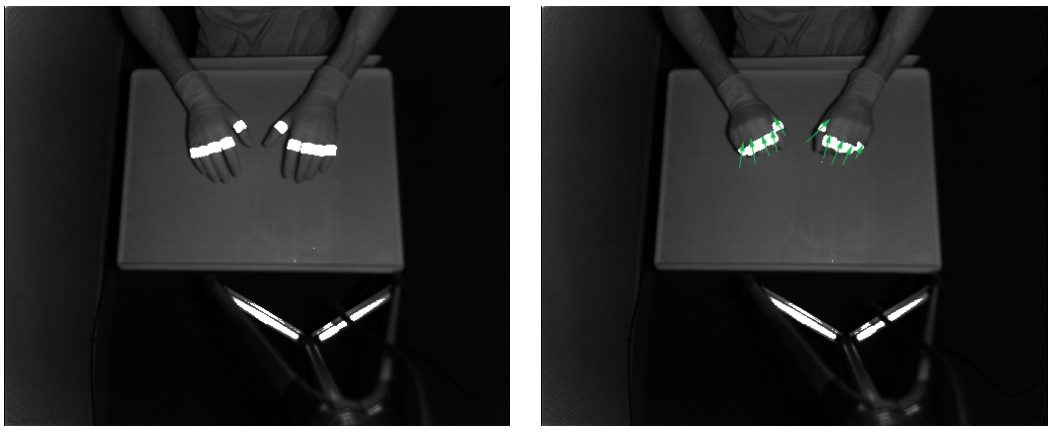
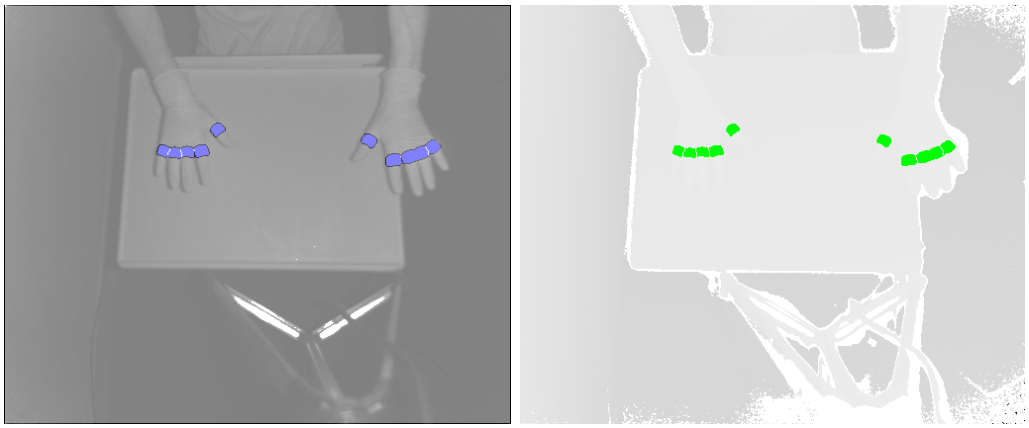


Figura 3.23: Flujo óptico del movimiento rastreado mediante los marcadores.

2. Al mismo tiempo, se transforman las posiciones en el plano de la imagen infrarroja y las distancias de la imagen del sensor de profundidad para construir las coordenadas espaciales (Figura 3.24).



(a) (b)

Figura 3.24: Coordenadas espaciales de los marcadores: (a) posiciones en el plano desde la imagen infrarroja y (b) los valores de proximidad desde el sensor de profundidad.

3. Por último, se asocian los identificadores de los marcadores a las coordenadas correspondientes y los tiempos de adquisición para guardar el registro en disco.

Capítulo 4

Experimentación y Resultados

A partir de la implementación hecha en el capítulo 3 se llevaron a cabo varios conjuntos de pruebas para identificar las fallas y hacer las mejoras correspondientes en el rastreo. Asimismo, recopilando las posiciones de los marcadores en cada paso de tiempo es posible realizar algunas mediciones de las distribuciones de esos puntos respecto a los movimientos realizados, mismos que se analizarán en la sección 4.2 para experimentos particulares.

En todas las pruebas se emplearon dos dispositivos Kinect v2 dispuestos hacia los lados de la superficie sobre la que son colocadas las manos, procurando que las cámaras fueran alineadas a una altura similar y con ángulo de visión parecido; no obstante, el número y posición de las cámaras puede extenderse en tanto las vistas sean útiles.

Asimismo, el número de marcadores cambió entre experimentos con la finalidad de evaluar tanto la segmentación de los mismos como la recuperación de las posiciones para el análisis mencionado, debido a que la principal meta del trabajo es analizar la precisión de los movimientos realizados.

4.1. Recuperación de marcadores

En la presente sección se muestra la segmentación de los anillos reflejantes durante algunas pruebas de laboratorio en las que, dada la tasa de muestreo del Kinect (30 fps) y considerando la capacidad de cómputo del equipo utilizado, el tiempo de captura fue de 20 segundos, suficiente para un total de 600 frames por cámara y por tipo de imagen.

4.1.1. Experimento 1

El primer conjunto de pruebas corresponde al diseño y desarrollo realizados, por lo que se emplearon 5 marcadores en cada mano para determinar la funcionalidad de los algoritmos. En la Figura 4.1 puede verse la segmentación de los marcadores en algunas capturas hechas con cada uno de los Kinect, mientras que en la Figura 4.2 se observan los centros de los marcadores que son utilizados como puntos característicos para al flujo óptico calculado.

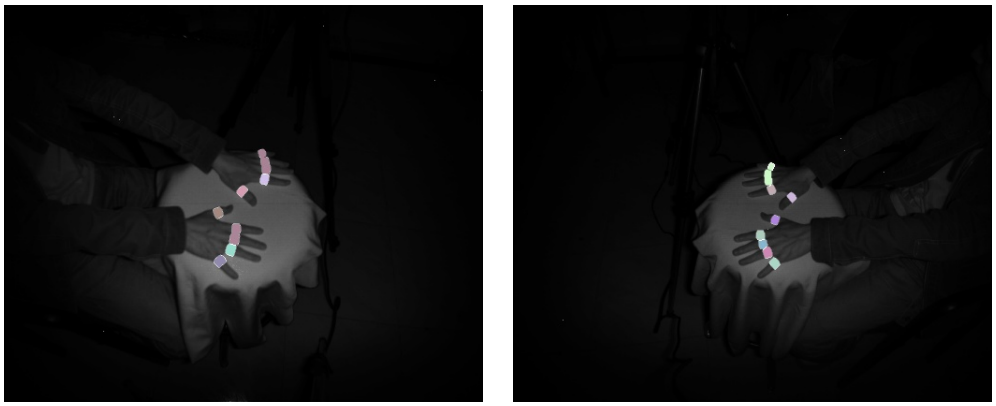


Figura 4.1: Experimento 1: Segmentación de marcadores en ambas manos con dos dispositivos Kinect.

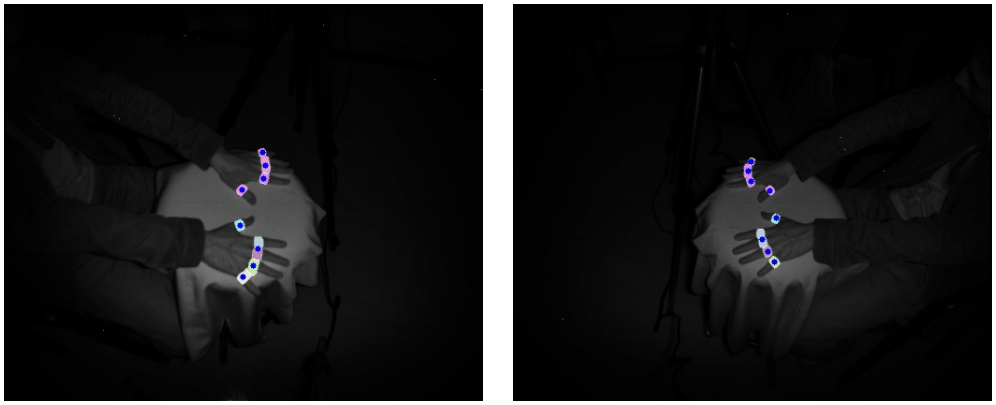


Figura 4.2: Experimento 1: Puntos característicos detectados para el flujo óptico.

4.1.2. Experimento 2

Para el segundo experimento se redujo el uso de marcadores a 3 por cada mano, colocados estratégicamente para mejorar las segmentaciones y, por tanto, la precisión de las posiciones. Además, se realizó una tarea de atar nudos para incrementar la complejidad de los movimientos en la ejecución.

De manera análoga, en la Figura 4.3 se muestra la recuperación de los marcadores en las vistas, en tanto que en la Figura 4.4 se ilustran los puntos característicos para el flujo óptico.

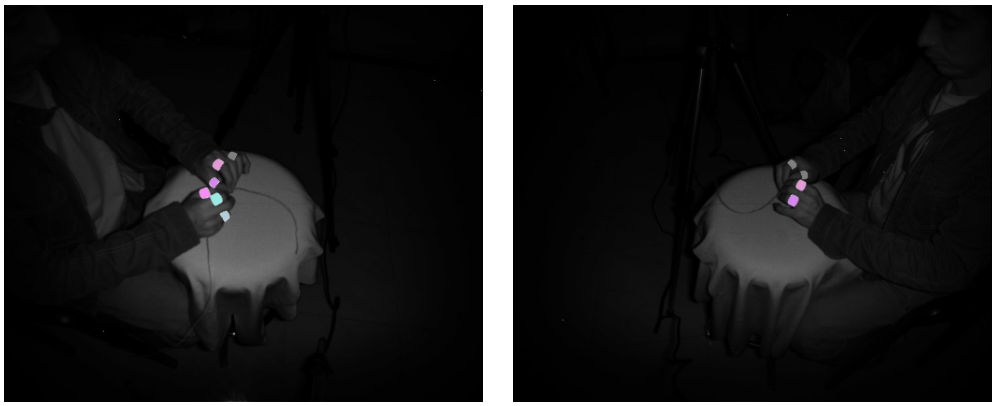


Figura 4.3: Experimento 2: Segmentación de marcadores en ambas manos con dos dispositivos Kinect.

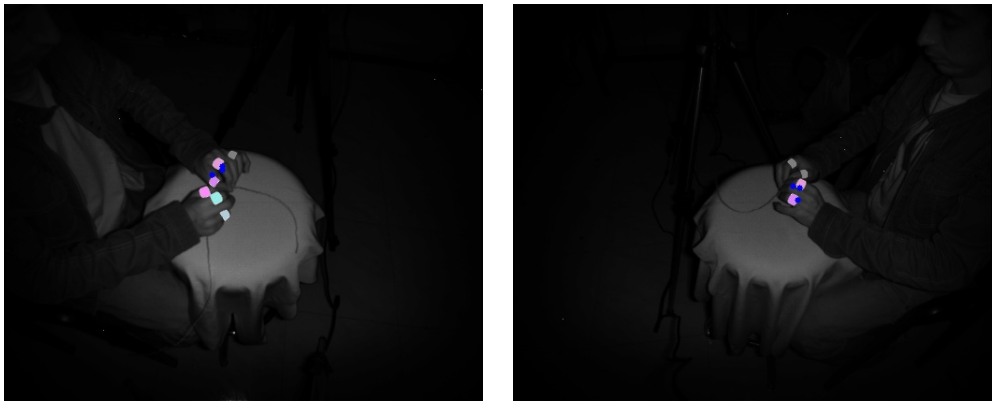


Figura 4.4: Experimento 2: Puntos característicos detectados para el flujo óptico.

4.1.3. Experimento 3

El tercer grupo de pruebas se llevó a cabo con tres marcadores por mano, pero los gestos capturados fueron hechos por usuarios externos a quienes se les permitió realizar movimientos arbitrarios con el fin de explorar espacios distintos. Muestra de los resultados, de movimientos realizados por uno de los usuarios, se tiene en las Figuras 4.3 y 4.4.

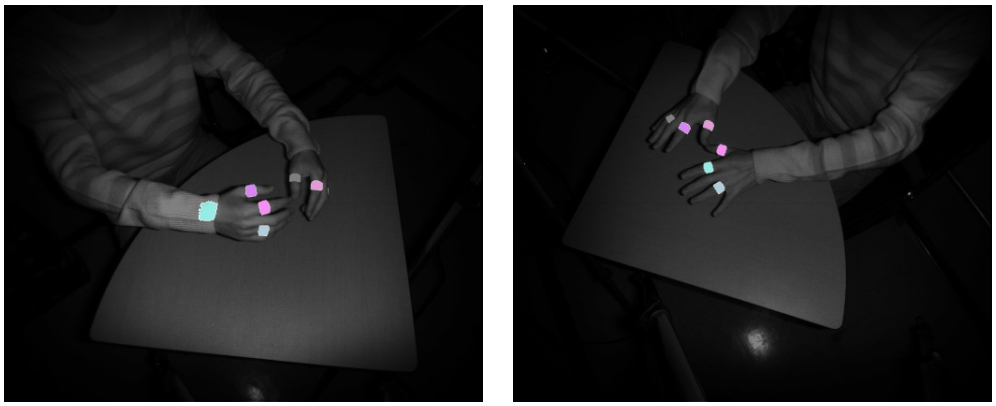


Figura 4.5: Experimento 3: Segmentación de marcadores en ambas manos con dos dispositivos Kinect.

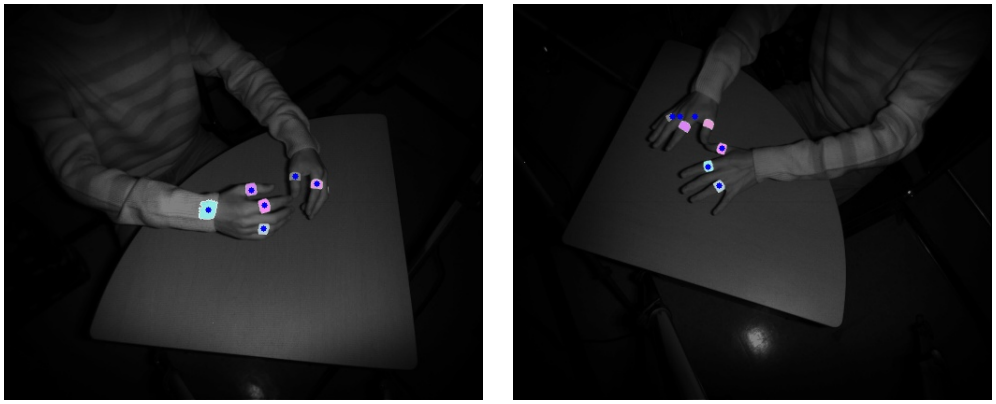


Figura 4.6: Experimento 3: Puntos característicos detectados para el flujo óptico.

4.2. Análisis de movimientos manuales

Con respecto a la aplicabilidad del sistema desarrollado, el prototipo fue utilizado para determinar si es posible evaluar cualitativa y cuantivativamente la habilidad de distintos usuarios para atar nudos, así como el nivel de experiencia de especialistas en sutura de ojo por microcirugía con microscopio e instrumental.

Para ello, a partir de las posiciones registradas para cada marcador se calcularon propiedades de velocidad y aceleración entre frames consecutivos. Ejemplo de una gráfica de distribución de posiciones en el plano de un movimiento circular artificial puede verse en la Figura 4.7, donde los puntos de los marcadores se visualizan con un color distinto entre sí.

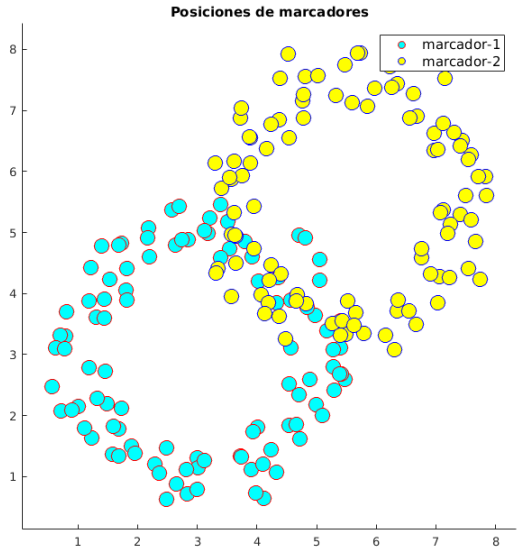
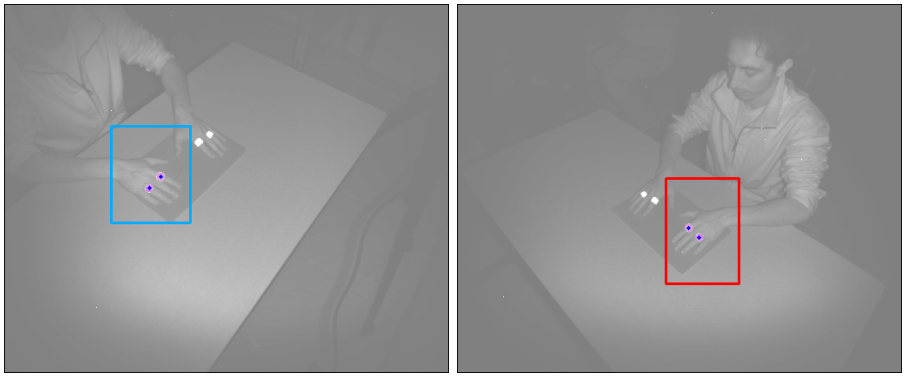


Figura 4.7: Posiciones de los marcadores de un movimiento circular artificial; los puntos de un sólo color corresponden a cada marcador distinto.

4.2.1. Experimento A: Nudos

Para esta prueba se consideró un grupo de usuarios con poca o nula experiencia en atadura de nudos, tarea que consistió en una captura de 10 segundos para intentar realizar el nudo. A cada usuario se le colocaron dos anillos reflejantes, en los dedos índice y anular de cada mano, y se le indicó comenzar de una posición particular con las palmas de las manos extendidas sobre una superficie de referencia, llevar a cabo el nudo y una vez hecho regresar las manos a una posición similar a la de inicio del experimento.

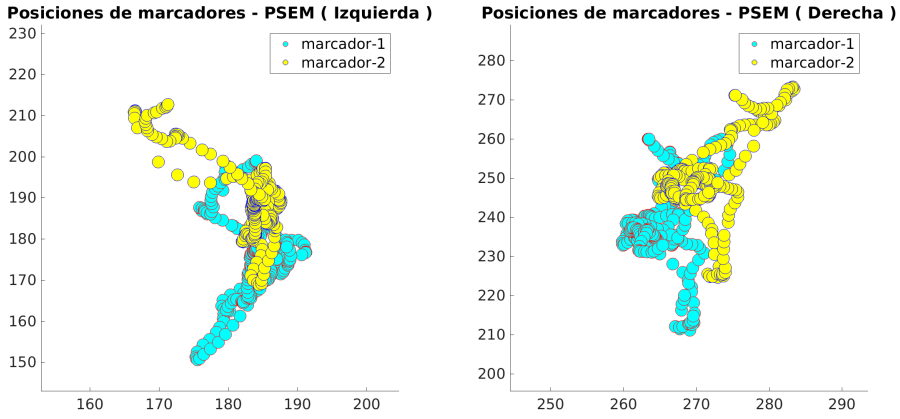
Para llevar a cabo las comparaciones entre los usuarios, se utilizaron las vistas para procesar únicamente los marcadores de la mano correspondiente; es decir, en las vistas de la mano izquierda se consideraron solamente los anillos de esa mano, definiendo para ello una región de interés menor en las imágenes, tal como se ilustra en la Figura 4.8.



(a) (b)

Figura 4.8: Prueba de atadura de nudos; región de interés definida para la vista izquierda y derecha.

En todos los casos, se graficaron las coordenadas en el plano correspondientes a los marcadores rastreados de cada mano (Figura A.16) y se registró el número de frames en los cuales la tarea fue completada; es decir, el tiempo en el que el usuario colocó las manos nuevamente sobre la superficie mencionada después de hacer el nudo.



(a) (b)

Figura 4.9: Coordenadas en el plano de los marcadores del usuario de prueba **PSEM**; el resto de figuras no citadas se ilustran en el Anexo A.

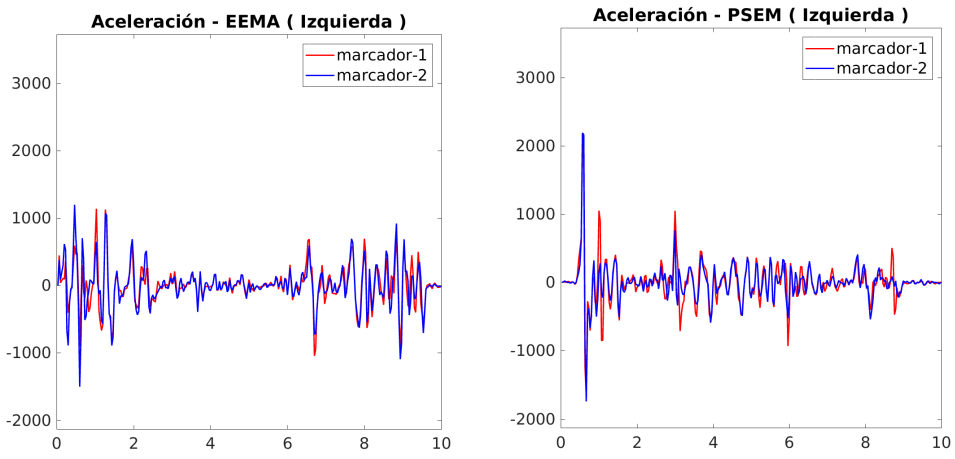
Con los datos recuperados se calculó la velocidad y aceleración instantáneas entre frames consecutivos; asimismo, se obtuvieron tales valores debido a que el cambio de posición y las propiedades señaladas se modifican durante la ejecución dependiendo de la destreza de cada usuario. En las tablas 4.1 y 4.2 se agrupan los valores de las dos sesiones de captura del experimento. Además, para obtener todas las gráficas, se definieron las áreas comunes de trabajo entre usuarios, asi como los máximos y mínimos de los valores de las propiedades evaluadas, con el fin de visualizar las diferencias entre sí.

Usuario	**Mano**	**No.Frames**	**Tiempo**	**Marc.**	**Dst(Avg)**	**Dst(Stdev)**
EEBA	Izquierda	245	8.166666667	1	9.5987	4.764
EEBA	Izquierda	245	8.166666667	2	13.2372	6.1613
EEBA	Derecha	245	8.166666667	1	10.0314	4.8403
EEBA	Derecha	245	8.166666667	2	13.0093	5.7645
EEBI	Izquierda	248	8.266666667	1	18.3091	10.6386
EEBI	Izquierda	248	8.266666667	2	16.578	8.7749
EEBI	Derecha	235	7.833333333	1	21.2977	10.689
EEBI	Derecha	235	7.833333333	2	22.0329	11.0901
EEMA	Izquierda	290	9.666666667	1	8.81	7.4302
EEMA	Izquierda	290	9.666666667	2	9.2748	6.9023
EEMA	Derecha	285	9.5	1	9.9966	5.5017
EEMA	Derecha	285	9.5	2	9.2898	4.4583
PSE-M	Izquierda	270	9	1	8.7279	6.7515
PSE-M	Izquierda	270	9	2	9.2163	7.0325
PSE-M	Derecha	260	8.666666667	1	9.0441	7.2765
PSE-M	Derecha	260	8.666666667	2	9.689	7.1087
PSE-AH	Izquierda	85	2.833333333	1	20.9447	8.1702
PSE-AH	Izquierda	85	2.833333333	2	21.6723	8.1682
PSE-AH	Derecha	170	5.666666667	1	23.0208	7.6713
PSE-AH	Derecha	170	5.666666667	2	23.7909	9.8759
PSE-AM	Izquierda	130	4.333333333	1	24.4421	10.7054
PSE-AM	Izquierda	130	4.333333333	2	26.3728	14.4111
PSE-AM	Derecha	175	5.833333333	1	36.8063	12.9248
PSE-AM	Derecha	175	5.833333333	2	37.9388	14.7198
PSE-S	Izquierda	185	6.166666667	1	8.4856	4.2517
PSE-S	Izquierda	185	6.166666667	2	8.6958	4.0816
PSE-S	Derecha	205	6.833333333	1	13.0859	7.8548
PSE-S	Derecha	205	6.833333333	2	11.8871	7.2446

Tabla 4.1: Tiempo de ejecución y distancia recorrida en los movimientos manuales de usuarios con nula o poca experiencia en la prueba de atadura de nudos.

De acuerdo a la tabla 4.1, en el caso de la primera sesión (primeros 4 usuarios), la mayoria utilizó más del 80 % del tiempo para completar la tarea y el promedio en el cambio de posiciones consecutivas de los marcadores es en algunos casos la mitad respecto a los usuarios de la segunda sesión; por ejemplo, los usuarios **EEBA** y **EEMA**, con mayor experiencia en la tarea, se desplazan menos que los usuarios **PSE-AH** y **PSE-AM** de la segunda sesión.

Por otra parte, la distancia promedio recorrida y la desviación estándar son casi iguales para los usuarios **EEMA** y **PSEM**; sin embargo, revisando la tabla 4.2 puede observarse que la aceleración del más experimentado (**EEMA**) es menor e incluso negativa, de manera que para completar el nudo tiende a detenerse más y la aceleración es baja durante el intervalo correspondiente a la atadura del nudo, en comparación con el usuario **PSEM** con mayor variación en su aceleración, tal como se muestra en la Figura 4.10.



(a) (b)

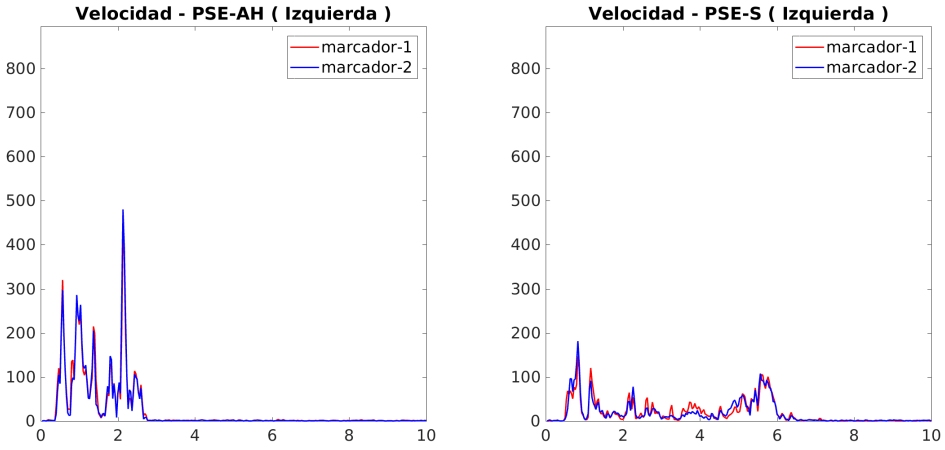
Figura 4.10: Aceleración de la mano izquierda del usuario **EEMA** y **PSEM**.

Respecto a los usuarios de la segunda sesión (últimos 3 usuarios de las tablas señaladas), el tiempo de realización del nudo oscila alrededor de los 6 segundos, por lo que sus velocidades y aceleraciones correspondientes son en conjunto mayores que el grupo de la primera sesión. Con base en esta observación, en el caso de la mano izquierda del usuario **PSE-AH** la ejecución registrada es menor a 3 segundos, lo cual confirma que además de haber realizado la tarea con mayor rapidez, ésto influyó en un problema en la adquisición puesto que la transición entre frames de la captura no es continua e incrementa las distancias entre marcadores de imágenes consecutivas y, por tanto, el cálculo de valores reportados en la Tabla 4.2. Una muestra comparativa

de la velocidad de este caso, respecto al mejor usuario de la segunda sesión (**PSE-S**), se ilustra en la Figura 4.12.

Usuario	**Mano**	**Marc.**	**Spd(Avg)**	**Spd(Stdev)**	**Acc(Avg)**	**Acc(Stdev)**
EEBA	Izquierda	1	27.3797	33.7582	0.25571	387.4389
EEBA	Izquierda	2	24.8585	32.2504	0.23826	328.5905
EEBA	Derecha	1	31.9006	30.5899	0.24233	419.2308
EEBA	Derecha	2	31.8976	33.3637	0.25734	439.971
EEBI	Izquierda	1	28.5794	36.6747	0.15425	431.4697
EEBI	Izquierda	2	28.3412	35.7603	0.06683	429.7061
EEBI	Derecha	1	30.1454	34.6704	0.00074859	339.9695
EEBI	Derecha	2	32.8161	38.4125	-0.0029838	408.0487
EEMA	Izquierda	1	24.6212	23.2972	-0.80981	294.419
EEMA	Izquierda	2	23.5929	23.4865	-0.78878	320.5731
EEMA	Derecha	1	24.4302	23.1428	-0.47052	353.5591
EEMA	Derecha	2	25.0605	26.5074	-0.28393	432.0847
PSE-M	Izquierda	1	24.8931	23.5869	0.32131	333.208
PSE-M	Izquierda	2	21.854	21.8122	0.19861	299.6808
PSE-M	Derecha	1	23.0347	20.2137	0.23841	238.4248
PSE-M	Derecha	2	21.9905	19.7339	0.24674	219.6462
PSE-AH	Izquierda	1	86.3188	89.2765	0.31224	1390.6392
PSE-AH	Izquierda	2	85.4951	94.0169	0.40471	1431.8405
PSE-AH	Derecha	1	46.633	57.9754	0.18386	770.1664
PSE-AH	Derecha	2	46.8258	52.6088	0.17147	535.7303
PSE-AM	Izquierda	1	83.1548	140.6596	0.48906	2109.6786
PSE-AM	Izquierda	2	85.2658	158.223	0.46561	2531.5332
PSE-AM	Derecha	1	71.435	100.4643	0.091642	1132.9564
PSE-AM	Derecha	2	71.7078	103.7959	0.038947	1205.2764
PSE-S	Izquierda	1	28.5526	27.7459	1.5874	369.4005
PSE-S	Izquierda	2	28.0083	29.5586	1.6321	340.7733
PSE-S	Derecha	1	26.124	22.9357	1.821	294.0929
PSE-S	Derecha	2	25.6941	23.2989	1.373	295.8754

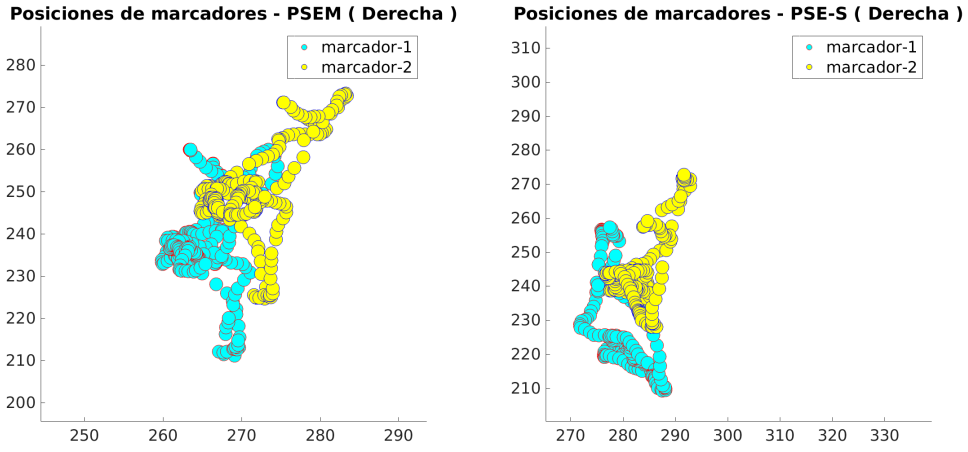
Tabla 4.2: Velocidad y aceleración de movimientos manuales de usuarios con nula o poca experiencia en la prueba de atadura de nudos.



(a) (b)

Figura 4.11: Velocidad de la mano izquierda del usuario **PSE-AH** y **PSE-S**.

Por último, en un par de casos, los usuarios con experiencia nula **PSEM** y **PSE-S**, sus velocidades son más bajas que el resto de los usuarios de la sesión de cada uno, mientras que sus aceleraciones promedio son más altas para ambos,; sin embargo, aún siendo usuarios sin experiencia en la tarea, las posiciones de sus movimientos están concentradas en una región reducida, en comparación con el resto de los usuarios de las sesiones respectivas.

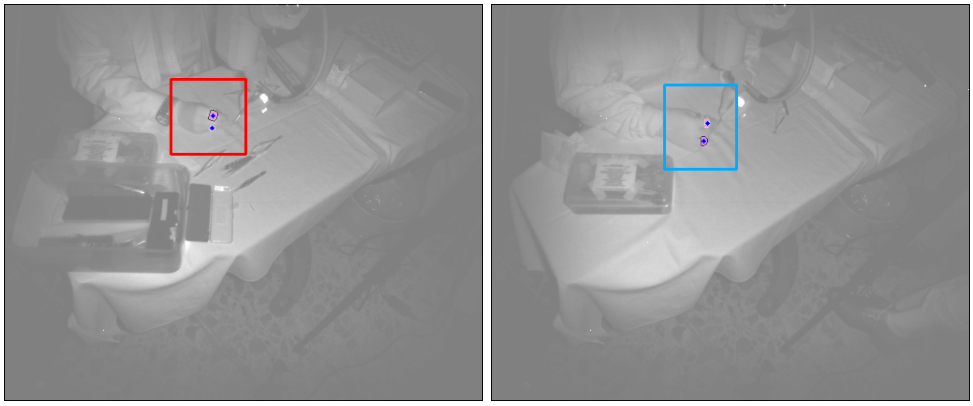


(a) (b)

Figura 4.12: Distribución de posiciones en el plano de los movimientos de la mano derecha del usuario **PSEM** y **PSE-S**.

4.2.2. Experimento B: Suturas

La prueba de sutura por microcirugía realizada, por 2 especialistas del Hospital General de México "Dr. Eduardo Licéaga", consistió en suturar 3 puntos de corte haciendo 3 nudos por cada uno (Figura 4.13). El tiempo máximo de captura de las manos fue de 180 segundos, durante el cual el especialista **B** sólo consiguió suturar un punto y el especialista **M** completó los 3 puntos en los primeros 125 seguntos. En resumen, los detalles más relevantes acerca de los participantes en el experimento se muestran en la Tabla 4.3.



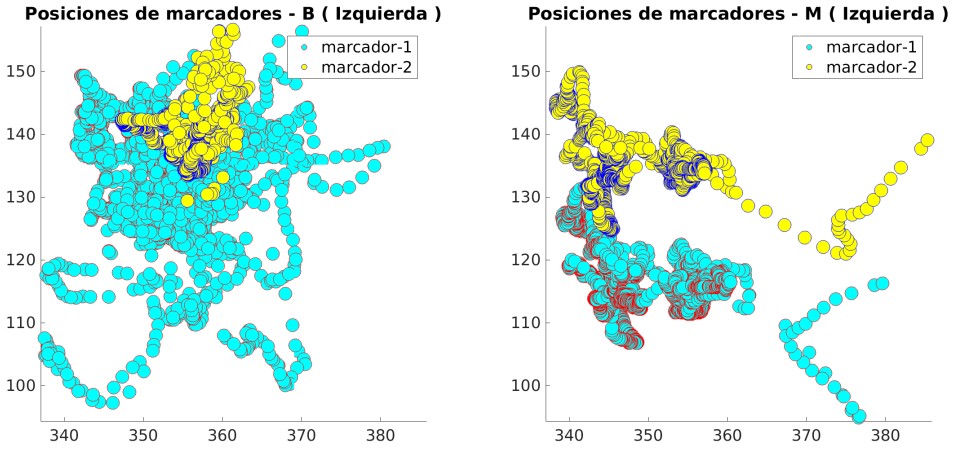
(a) (b)

Figura 4.13: Regiones de interés de la vista derecha de los especialistas **B** (izquierda) y **M** (derecha), respectivamente, en la prueba de sutura.

Especialista	**Experiencia (años)**	**Num. Puntos**	**Tiempo (seg.)**
B	3	1	180
M	> 6	3	125

Tabla 4.3: Datos y mediciones de los especialistas en la prueba de sutura por microcirugía; el especialista **M** cuenta con más de 6 años de experiencia, mientras que el especialista **B** con 3 años de experiencia con intervenciones esporádicas.

Tras recuperar las posiciones de los marcadores, se definió el área de trabajo común para los expertos, notando que el especialista **B**, además de emplear los 180 segundos para suturar sólo un punto, realizó mayor cambio de la mano en dicha área; por su parte, el recorrido más largo de la mano del especialista **M** ocurrió hacia el final del experimento cuando libera el instrumental quirúrgico (Figura 4.14).



(a) (b)

Figura 4.14: Posiciones de los marcadores de la mano izquierda del especialista **B** y **M**

De manera análoga al experimento de la sección 4.2.1, se registraron los tiempos y distancias de los marcadores de ambos especialistas, para calcular las estadísticas correspondientes mostradas en las tablas 4.4 y 4.5.

Usuario	Mano	No.Frames	Tiempo	Marc.	Dst(Avg)	Dst(Stdev)
M	Izquierda	3750	125	1	6.9431	3.4718
M	Izquierda	3750	125	2	6.6923	3.3505
M	Derecha	3750	125	1	4.5526	2.8357
M	Derecha	3750	125	2	4.1541	2.7856
B	Izquierda	5400	180	1	7.3864	4.9828
B	Izquierda	5400	180	2	3.7909	2.3393
B	Derecha	5400	180	1	9.0371	4.6211
B	Derecha	5400	180	2	8.2157	4.334

Tabla 4.4: Tiempo de ejecución y distancia recorrida en los movimientos manuales de los especialistas en la prueba de sutura por microcirugía.

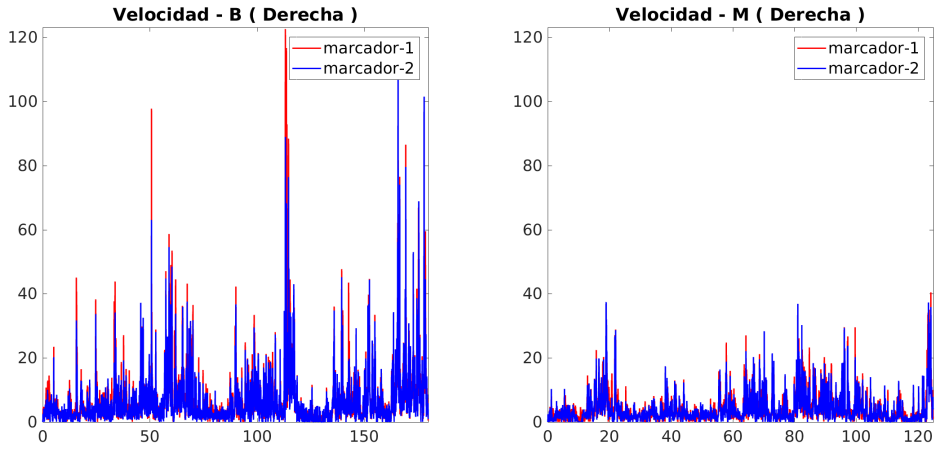
En particular, como los usuarios requirieron emplear las pinzas quirúrgicas, la sujeción de las mismas hizo que sus movimientos fueran más lentos y cortos para lograr la precisión necesaria para suturar; así, puede verse que incluso las desviaciones estándar en las distancias recorridas son similares y bajas en ambos casos.

Por otro lado, siguiendo la Tabla 4.5 se distingue que tanto las velocidades y aceleraciones del especialista **M** son en efecto menores respecto a los valores, de ambas manos, del especialista **B**.

Usuario	Mano	Marc.	Spd(Avg)	Spd(Stdev)	Acc(Avg)	Acc(Stdev)
M	Izquierda	1	4.8852	8.412	0.11912	93.7763
M	Izquierda	2	4.0931	7.0962	0.17921	79.0189
M	Derecha	1	4.0413	4.635	-0.0090361	61.0616
M	Derecha	2	3.827	4.3708	0.0048292	59.8448
B	Izquierda	1	11.1847	18.6654	0.11572	227.2029
B	Izquierda	2	5.928	10.9615	0.50116	137.1702
B	Derecha	1	7.3854	9.9987	-0.043218	118.0792
B	Derecha	2	6.6401	8.6781	-0.035565	104.9277

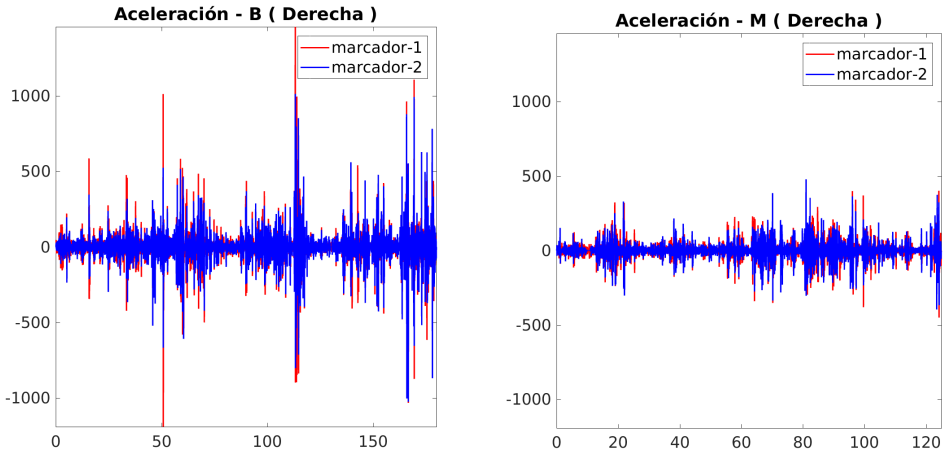
Tabla 4.5: Velocidad y aceleración de movimientos manuales de especialistas en la prueba de sutura por microcirugía.

En consecuencia, para este experimento específico y los usuarios registrados, el nivel de experiencia de uno de ellos le permite completar la tarea en menos tiempo y variación en sus movimientos en comparación con el especialista menos experimentado (Figuras 4.15 y 4.16).



(a) (b)

Figura 4.15: Comparación de las velocidades de la mano derecha entre el especialista **B** y **M**.



(a) (b)

Figura 4.16: Comparación de las aceleraciones de la mano derecha entre el especialista **B** y **M**.

4.2.3. Observaciones adicionales

Considerando los resultados presentados en las secciones 4.2.1 y 4.2.2 vemos que en conjunto sí existe una relación entre el nivel de experiencia en la atadura de nudos, como una tarea base, para evaluar eventualmente la destreza de prospectos en la realización de experimentos o eventos de suturas por microcirugía.

En las dos pruebas, las observaciones sobre usuarios con un cierto nivel y detenimiento para realizar las tareas satisfacen, en la mayoría de los casos, las mediciones analizadas.

No obstante, es importante señalar que la finalidad de la herramienta construida es únicamente evaluar ciertas características de los movimientos registrados para que expertos en el área sean quienes validen en determinado momento - sin necesidad de encontrarse presencialmente durante las pruebas - la habilidad de otros usuarios en la ejecución de las destrezas presentadas y sugeridas en este trabajo.

Capítulo 5

Conclusiones y trabajo a futuro

Se presentó el desarrollo de una colección de algoritmos de una herramienta utilizada para el seguimiento de marcadores pasivos con el fin de analizar la distribución espacial de movimientos manuales en un área de trabajo determinada.

La elección de los anillos y la cinta reflejante para el diseño de los marcadores permite la recuperación de las posiciones de los mismos en la mayoría de las capturas. Para ésto, se emplea el algoritmo de segmentación por vertientes o clusterización iterativa para separar los marcadores en aquellos casos donde sus proyecciones en las vistas los muestran unidos o muy cercanos entre sí.

Precisamente, el uso de múltiples cámaras ayuda a procesar marcadores que en algunas de las imágenes proporcionadas no aparecen o son más difíciles de recuperar. Es importante señalar que el número de dispositivos puede extenderse de forma que el proceso de rastreo se simplifique seleccionando la vista que mejor detalle proporcione sobre algunos marcadores; sin embargo, debe tomarse en cuenta la capacidad de cómputo para el almacenamiento temporal en los búfers de datos (Sección 3.2).

Por ahora, el procesamiento de la imagen se realiza usando únicamente la imagen infrarroja del Kinect, ya que la detección de los anillos es más sencilla en este tipo de imágenes; no obstante, el uso de la imagen RGB también es una buena opción aún con el color actual de la cinta reflejante, que es contrastante con el color de la piel de cualquier usuario, o bien, empleando guantes de látex.

Mediante las pruebas descritas en la sección 4.2 se logró presentar algunas observaciones y mediciones que ayudarían a determinar el grado de experiencia de médicos especializados en sutura por microcirugía, sin necesidad de requerir en el momento de un usuario más experimentado que, actualmente, evalúa los movimientos, su precisión y el tiempo de duración de la tarea de forma presencial.

A través del experimento mencionado se pudo notar que los especialistas con mayor experiencia ejecutan gestos manuales más lentos, suaves y realizan las suturas desplazando sus dedos en distancias cortas; incluso, las gráficas muestran que las posiciones de los marcadores se mantienen en una zona de trabajo en particular y no se mezclan entre sí. En cambio, especialistas con menos experiencia demoran en hacer los nudos de las suturas y las coordenadas de sus manos se registran y trasla-

pan en un área mayor.

Con el fin de mejorar y extender las capacidades del programa construido es necesario llevar a cabo pruebas adicionales que incorporen más Kinects; además, los algoritmos de segmentación y rastreo son independientes del tipo de dispositivo ya que sólo procesan las colecciones de imágenes capturadas, por lo que podría explorarse el uso de otro tipo de cámaras con mejor resolución o con mayor cuadros por segundo.

En ese sentido, dado que el proceso de adquisición utiliza una biblioteca específica para múltiples Kinect v2, sería útil desarrollar un *framework* más genérico que facilite la implementación del sistema multivisión en otras plataformas de diversos dispositivos.

Asimismo, se puede realizar segmentación mediante superpíxeles aplicando algún otro algoritmo citado en la sección 2.1 y aprovechar las imágenes a color de las cámaras correspondientes para mejorar el rastreo de los marcadores y que el método de flujo óptico pueda estimar los vectores de movimiento adecuadamente.

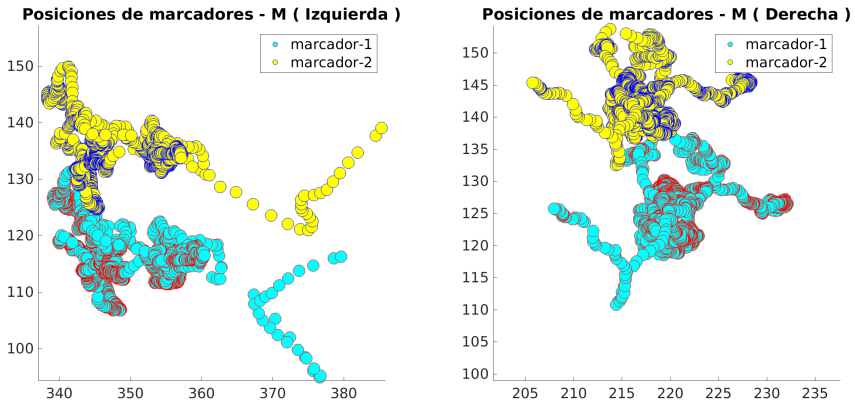
Por otro lado, nuevas pruebas de un mayor número de especialistas permitirían detectar otras características para evaluar su experiencia en microcirugía de forma más objetiva y que puedan compararse usuarios con el mismo nivel de experiencia para llevar a cabo el análisis cuantitativo pertinente.

Finalmente, podría utilizarse la herramienta para otras tareas como manipulación de dispositivos de cómputo mediante gestos específicos o llevarlo a la enseñanza del lenguaje de señas, por lo que sería necesario desarrollar o aplicar el motor de reconocimiento gestual apropiado para tales movimientos manuales.

Apéndice A

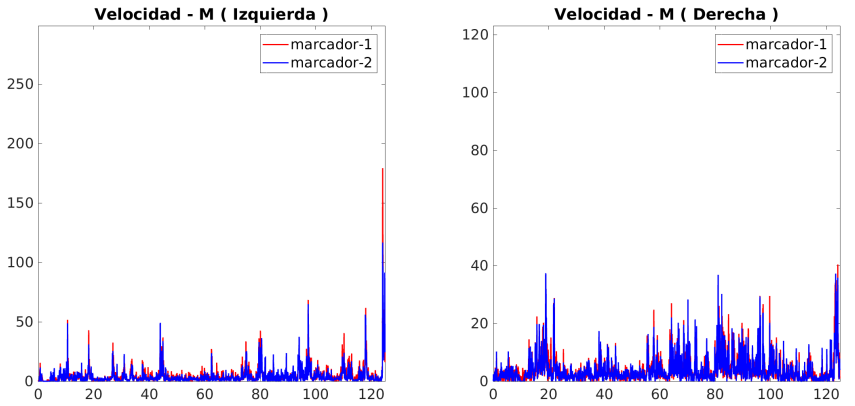
Gráficas de resultados de análisis de movimiento

1. Especialista **M**



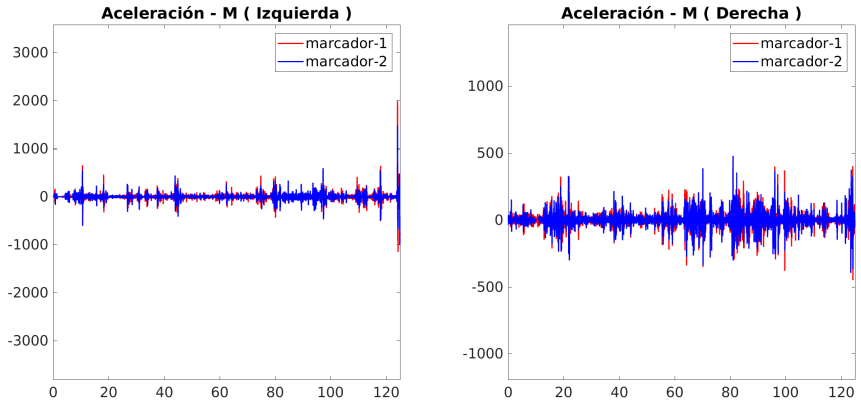
(a) (b)

Figura A.1: Coordenadas en el plano de los marcadores del especialista de prueba **M**.



(a) (b)

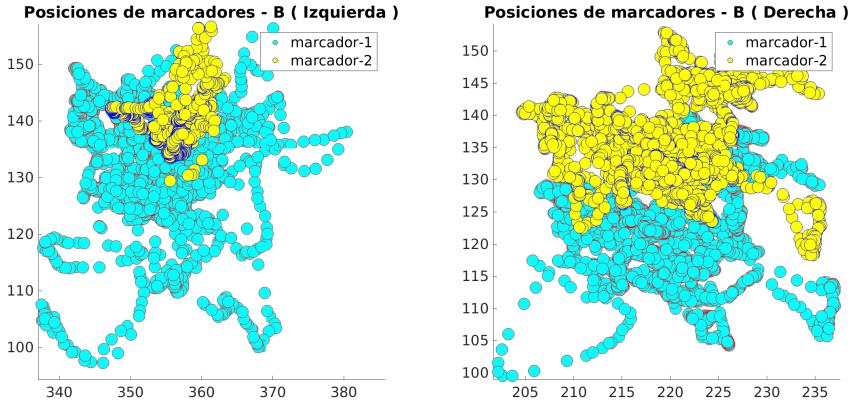
Figura A.2: Velocidades de los movimientos manuales del usuario de prueba **M**.



(a) (b)

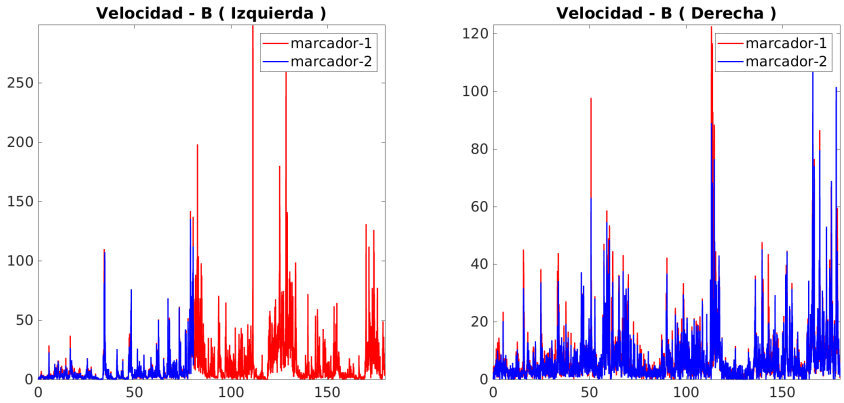
Figura A.3: Aceleraciones de los movimientos manuales del usuario de prueba **M**.

2. Especialista **B**



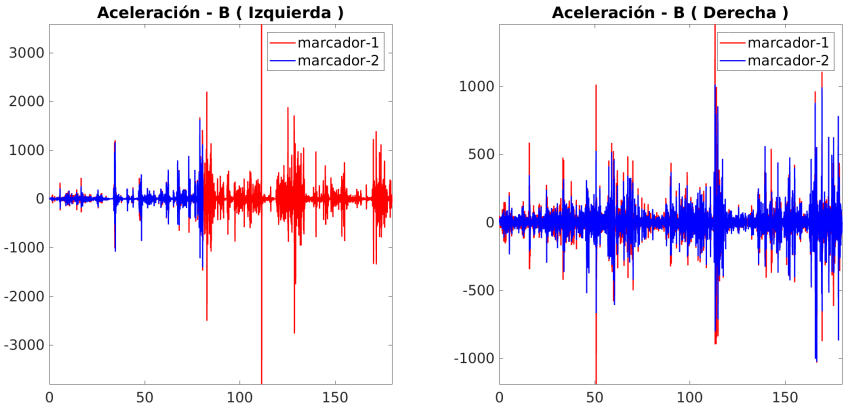
(a) (b)

Figura A.4: Coordenadas en el plano de los marcadores del usuario de prueba **B**.



(a) (b)

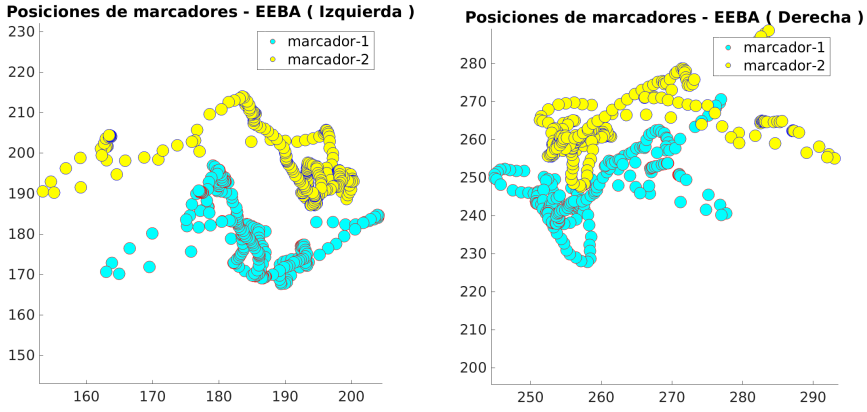
Figura A.5: Velocidades de los movimientos manuales del usuario de prueba **B**.



(a) (b)

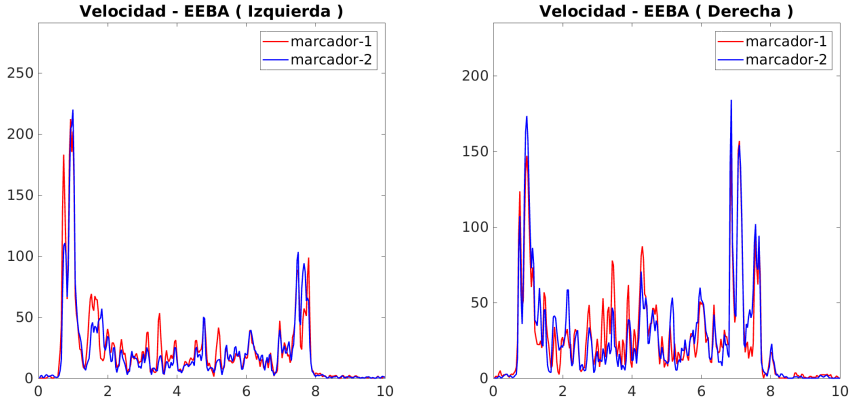
Figura A.6: Aceleraciones de los movimientos manuales del usuario de prueba **B**.

3. Usuario **EEBA**



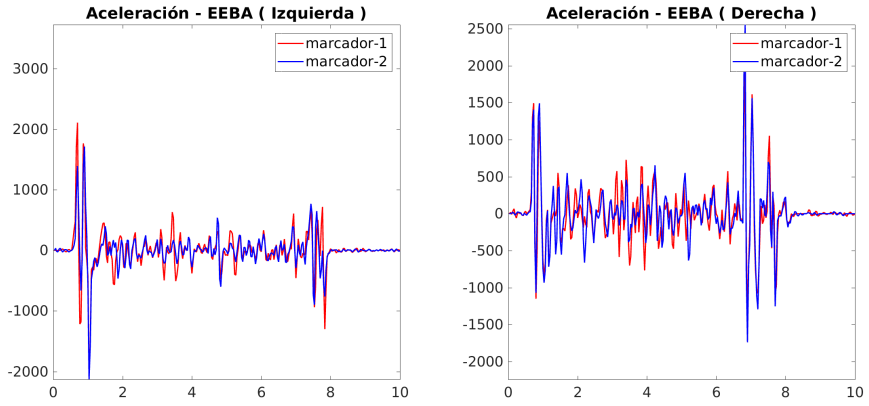
(a) (b)

Figura A.7: Coordenadas en el plano de los marcadores del usuario de prueba **EEBA**.



(a) (b)

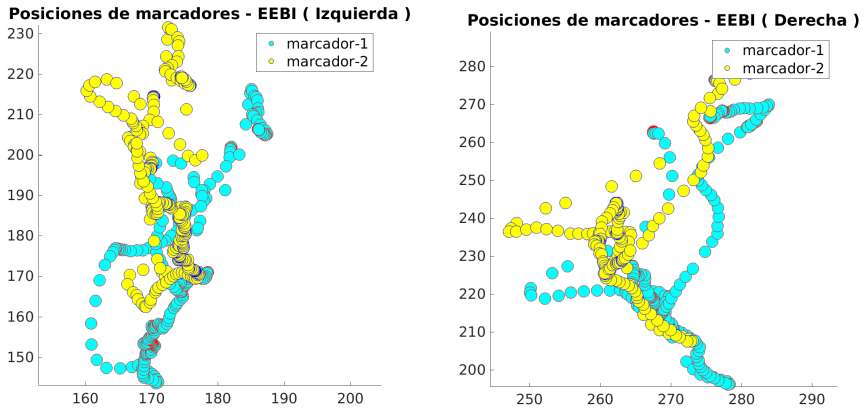
Figura A.8: Velocidades de los movimientos manuales del usuario de prueba **EEBA**.



(a) (b)

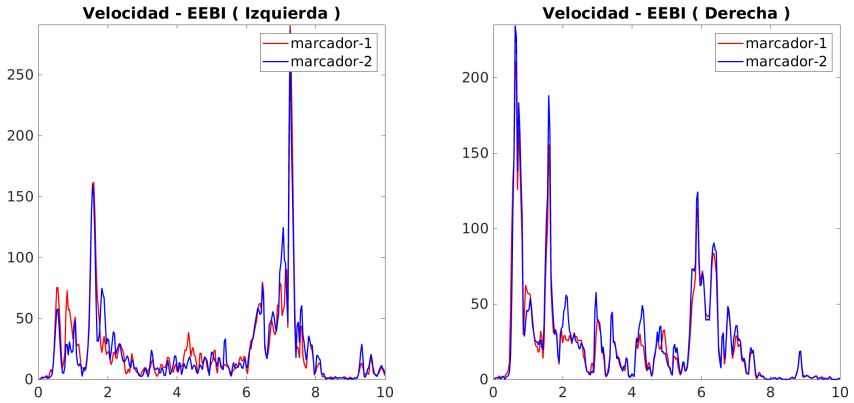
Figura A.9: Aceleraciones de los movimientos manuales del usuario de prueba **EEBA**.

4. Usuario **EEBI**



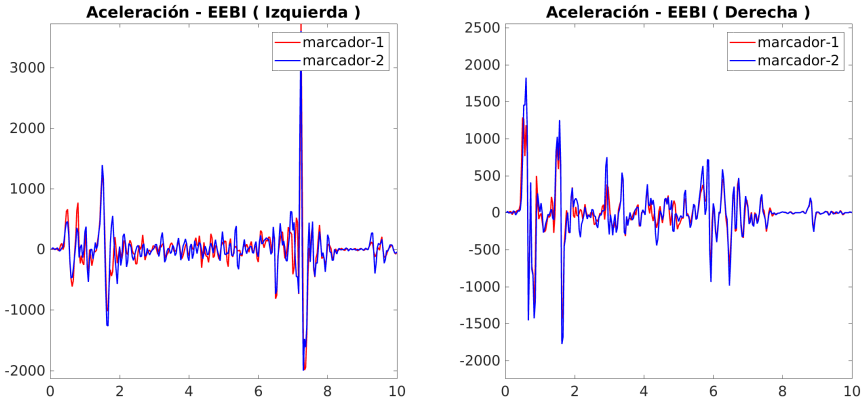
(a) (b)

Figura A.10: Coordenadas en el plano de los marcadores del usuario de prueba **EEBI**.



(a) (b)

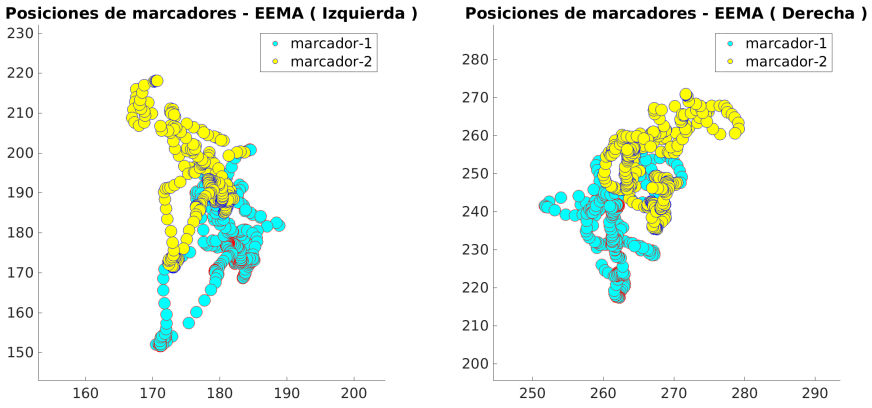
Figura A.11: Velocidades de los movimientos manuales del usuario de prueba **EEBI**.



(a) (b)

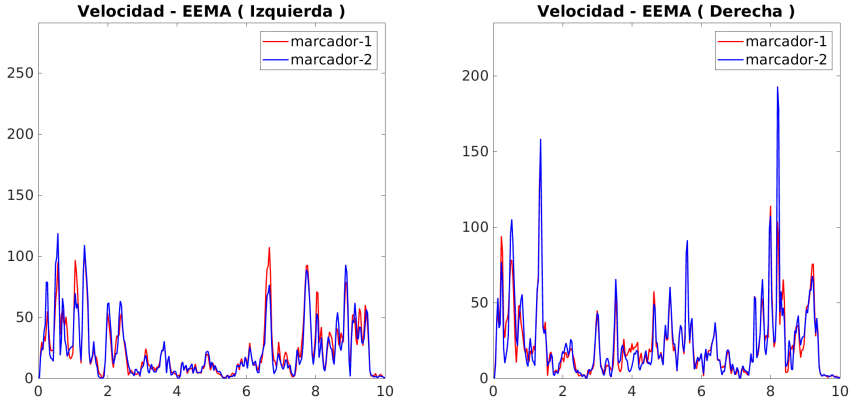
Figura A.12: Aceleraciones de los movimientos manuales del usuario de prueba **EEBI**.

5. Usuario **EEMA**



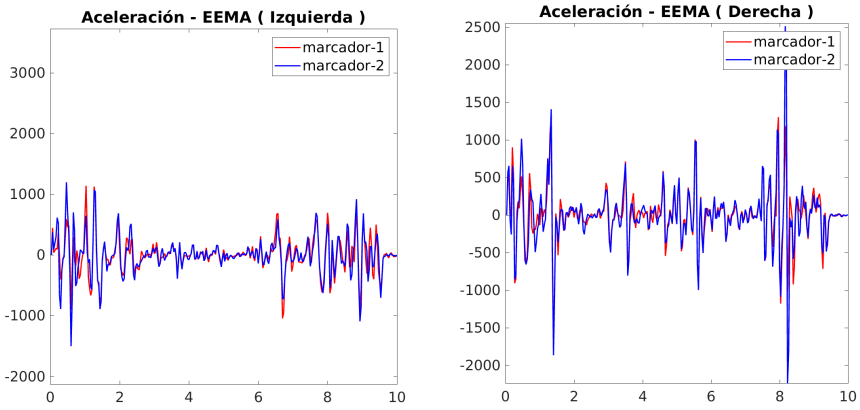
(a) (b)

Figura A.13: Coordenadas en el plano de los marcadores del usuario de prueba **EEMA**.



(a) (b)

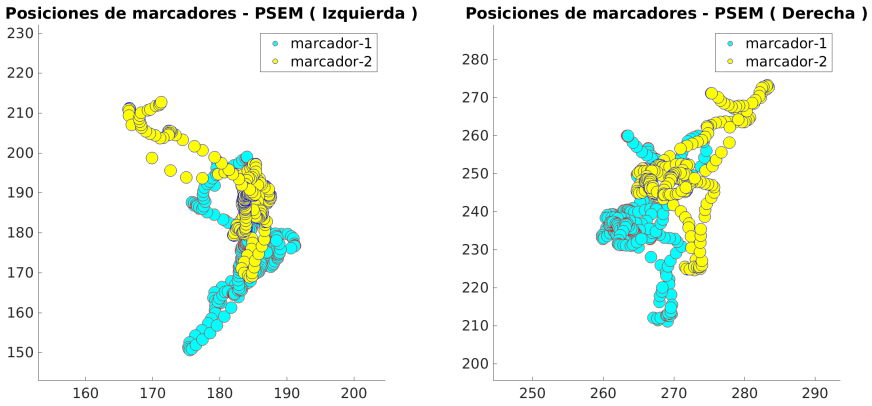
Figura A.14: Velocidades de los movimientos manuales del usuario de prueba **EEMA**.



(a) (b)

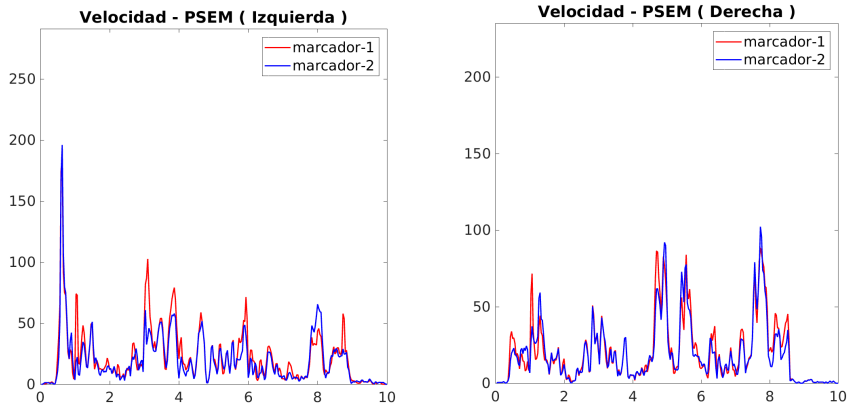
Figura A.15: Aceleraciones de los movimientos manuales del usuario de prueba **EEMA**.

6. Usuario **PSEM**



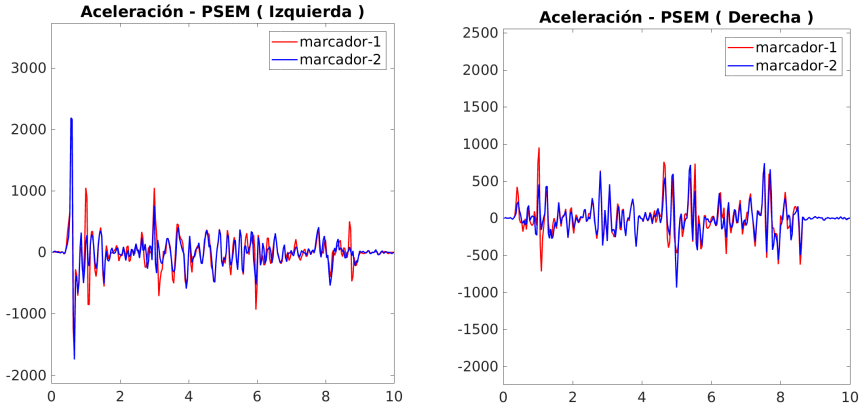
(a) (b)

Figura A.16: Coordenadas en el plano de los marcadores del usuario de prueba **PSEM**.



(a) (b)

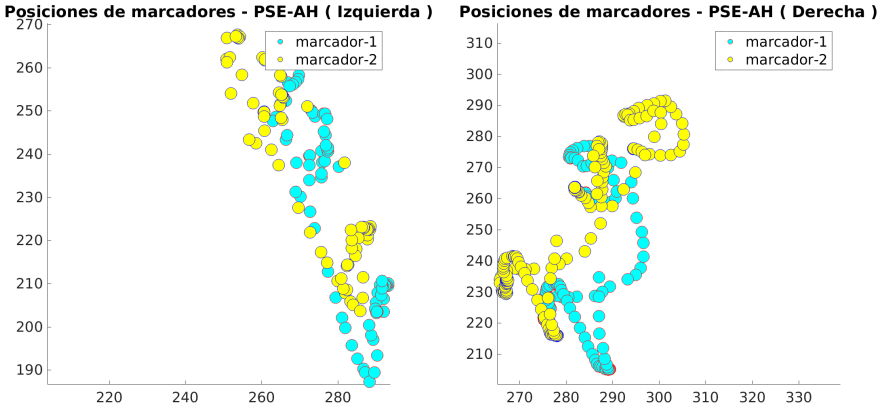
Figura A.17: Velocidades de los movimientos manuales del usuario de prueba **PSEM**.



(a) (b)

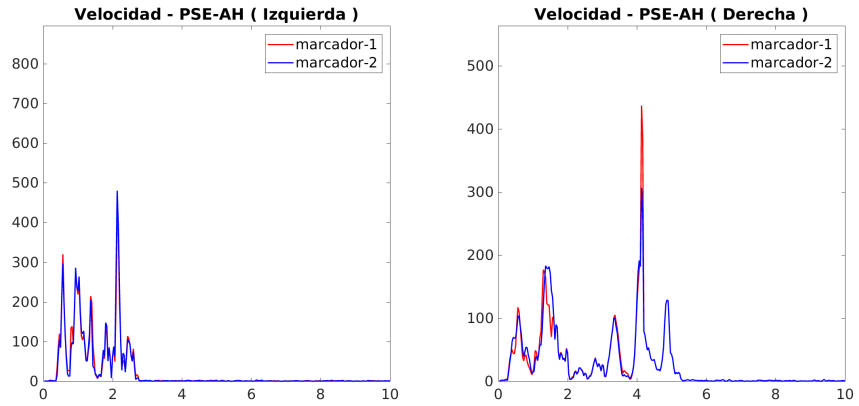
Figura A.18: Aceleraciones de los movimientos manuales del usuario de prueba **PSEM**.

7. Usuario **PSE-AH**



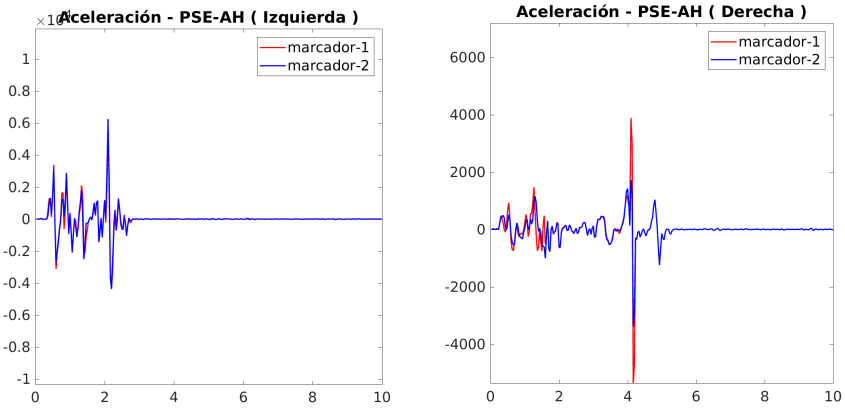
(a) (b)

Figura A.19: Coordenadas en el plano de los marcadores del usuario de prueba **PSE-AH**.



(a) (b)

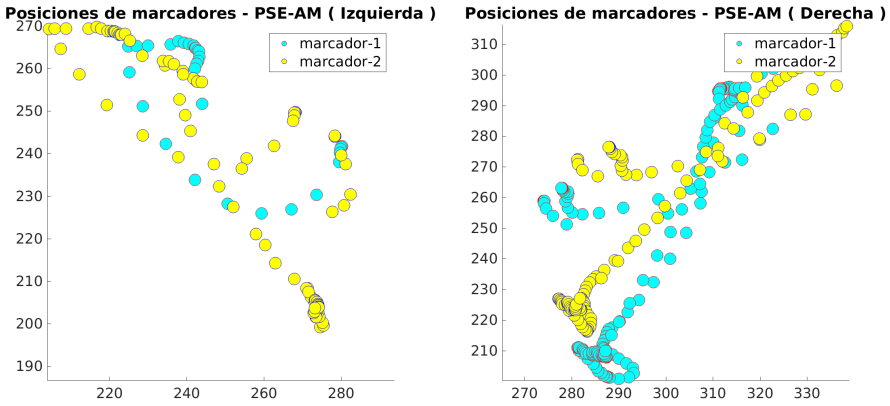
Figura A.20: Velocidades de los movimientos manuales del usuario de prueba **PSE-AH**.



(a) (b)

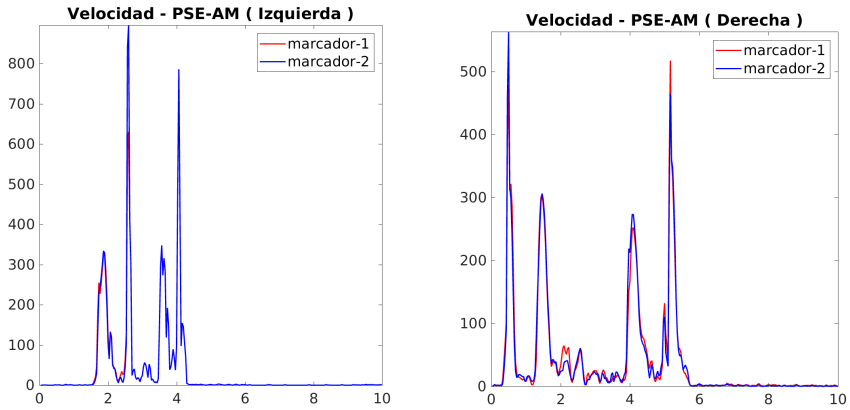
Figura A.21: Aceleraciones de los movimientos manuales del usuario de prueba **PSE-AH**.

8. Usuario **PSE-AM**



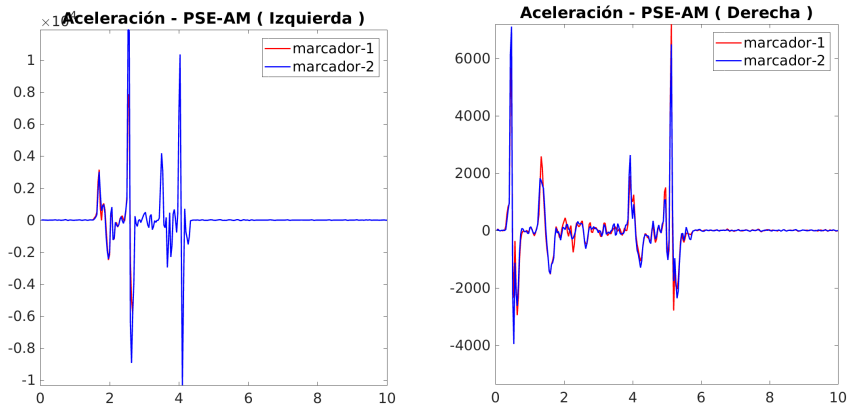
(a) (b)

Figura A.22: Coordenadas en el plano de los marcadores del usuario de prueba **PSE-AM**.



(a) (b)

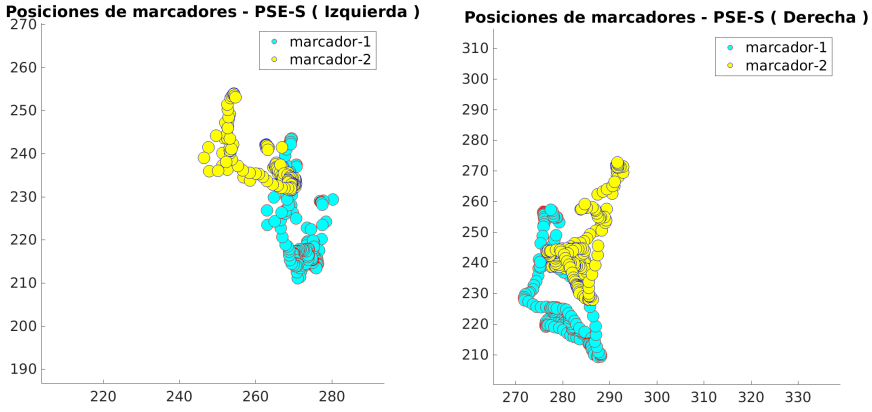
Figura A.23: Velocidades de los movimientos manuales del usuario de prueba **PSE-AM**.



(a) (b)

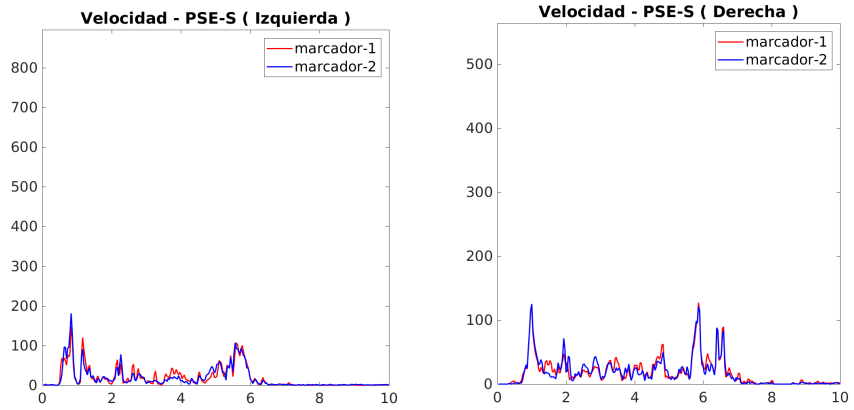
Figura A.24: Aceleraciones de los movimientos manuales del usuario de prueba **PSE-AM**.

9. Usuario **PSE-S**



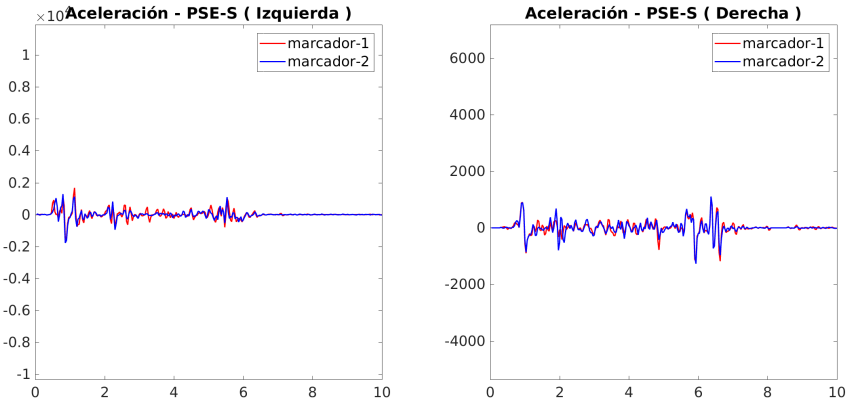
(a) (b)

Figura A.25: Coordenadas en el plano de los marcadores del usuario de prueba **PSE-S**.



(a) (b)

Figura A.26: Velocidades de los movimientos manuales del usuario de prueba **PSE-S**.



(a) (b)

Figura A.27: Aceleraciones de los movimientos manuales del usuario de prueba **PSE-S**.

Apéndice B

Código de adquisición de imágenes por multivisión

```
#include <boost/thread.hpp>
#include <boost/chrono.hpp>
#include <iostream>
#include <vector>

#include "Devices.hpp"
#include "DeviceControl.hpp"
#include "DataStructures.hpp"

/// [headers] libfreenect2
#include <libfreenect2/libfreenect2.hpp>
#include <libfreenect2/logger.h>

#include <opencv2/opencv.hpp>

std::vector<boost::thread *> DeviceThread;

void rundevicenum(EnumDevices &Ksconnected, int idnum)
{
    DeviceControl DevKinect;
    DevKinect.listeners(&Ksconnected, idnum);
    DevKinect.start(&Ksconnected, idnum);
    DevKinect.registrationini(&Ksconnected, idnum);
    DevKinect.capture(&Ksconnected, idnum);
}

int main(int argc, char *argv[])
{
    libfreenect2::setGlobalLogger(NULL);

    EnumDevices Ksconnected;
    Ksconnected.getserials();
    Ksconnected.CreateDevice();

    std::cout << " INFO::main.inicio-grabacion" << std::endl;

    for (int idx = 0; idx < Ksconnected.serials.size(); idx++)
        DeviceThread.push_back(new boost::thread(rundevicenum, boost::ref
            (Ksconnected), idx));
```

```
    for (int idx = 0; idx < Ksconnected.serials.size(); idx++)
          DeviceThread[idx]->join();

    std::cout << " INFO::main.fin-grabacion" << std::endl;

    Ksconnected.savedata();
    return 0;
}

#include <iostream>
#include "DeviceControl.hpp"

//C++ Member Initialization List it can be done with variables
//http://en.cppreference.com/w/cpp/language/initializer_list
DeviceControl::DeviceControl()    //: RGBbuff(2000), IRbuff(2000),
    DPbuff(2000), BfData(2000)
{
}

DeviceControl::DeviceControl(bool list1, bool list2)
{
    enable_rgb = list1;
    enable_depth = list2;
}

void DeviceControl::increment()
{
    int* p = InternalId.get();
   ++*p;
}

void DeviceControl::listeners(EnumDevices *DevSer, int idx)
{
    int types = 0;
    if (enable_rgb)
        types |= libfreenect2::Frame::Color;
    if (enable_depth)
        types |= libfreenect2::Frame::Ir | libfreenect2::Frame::Depth;

    listener.reset(new libfreenect2::SyncMultiFrameListener(types));
    DevSer->devs[idx]->setColorFrameListener(listener.get());
    DevSer->devs[idx]->setIrAndDepthFrameListener(listener.get());

    std::cout << "device id: " << idx << std::endl;
}

void DeviceControl::start(EnumDevices *DevSer, int idx)
{
    if (enable_rgb && enable_depth)
    {
        if (!DevSer->devs[idx]->start())
        {
```

```
            std::cout << "failure to start listeners!" << std::endl;
            exit(EXIT_FAILURE);
        }
    }
    else
    {
        if (!DevSer->devs[idx]->startStreams(enable_rgb, enable_depth))
        {
            std::cout << "failure to start listeners!" << std::endl;
            exit(EXIT_FAILURE);
        }
    }

    std::cout << "device serial: " << DevSer->devs[idx]->getSerialNumber
        () << std::endl;
    std::cout << "device firmware: " << DevSer->devs[idx]->
        getFirmwareVersion() << std::endl;
}

void DeviceControl::registrationini(EnumDevices *DevSer, int idx)
{
    registration.reset(new libfreenect2::Registration(DevSer->devs[idx
        ]->getIrCameraParams(), DevSer->devs[idx]->getColorCameraParams()
        ));
    undistorted.reset(new libfreenect2::Frame(512, 424, 4));
    registered.reset(new libfreenect2::Frame(512, 424, 4));
    depth2rgb.reset(new libfreenect2::Frame(1920, 1080 + 2, 4));
}

void DeviceControl::capture(EnumDevices *DevSer, int idx)
{
    size_t framecount = 0;
    DevSer->DataM[idx].set_capacity(3001);

    while ((framemax == (size_t)-1 || framecount < framemax))
    {
        if (!listener->waitForNewFrame(frames, 10 * 1000)) // 10 sconds
        {
            std::cout << "timeout!" << std::endl;
            exit(EXIT_FAILURE);
        }

        //libfreenect2::Frame *rgb = frames[libfreenect2::Frame::Color];
        libfreenect2::Frame *ir = frames[libfreenect2::Frame::Ir];
        libfreenect2::Frame *depth = frames[libfreenect2::Frame::Depth];
        MatArr BuffData;

        //cv::Mat(rgb->height, rgb->width, CV_8UC4, rgb->data).copyTo(
            BuffData.RGBM);
        cv::Mat(ir->height, ir->width, CV_32FC1, ir->data).copyTo(
            BuffData.IRM);
```

```
        cv::Mat(depth->height, depth->width, CV_32FC1, depth->data).
            copyTo(BuffData.DPM);
        MatArr ImData;

        //ImData.RGBM = BuffData.RGBM.clone();
        ImData.IRM = BuffData.IRM.clone();
        ImData.DPM = BuffData.DPM.clone();
        mutex.lock();

        //DevSer->DataV[idx].push_back(ImData);
        DevSer->DataM[idx].push_back(ImData);
        mutex.unlock();

        framecount++;
        listener->release(frames);
    }
    DevSer->devs[idx]->stop();
    DevSer->devs[idx]->close();
}

DeviceControl::~DeviceControl()
{
}
```

```
#pragma once
#include <boost/circular_buffer.hpp>
#include <boost/thread/tss.hpp>
#include <boost/thread/thread.hpp>
#include <libfreenect2/libfreenect2.hpp>
#include <libfreenect2/frame_listener_impl.h>
#include <libfreenect2/registration.h>
#include <opencv2/opencv.hpp>
#include "Devices.hpp"
#include "DataStructures.hpp"

class DeviceControl {
    private:
        /// [listeners]
        int types = 0;
        boost::thread_specific_ptr<libfreenect2::SyncMultiFrameListener>
            listener;
        /// [registration setup]
        boost::thread_specific_ptr<libfreenect2::Registration>
            registration;
        boost::thread_specific_ptr<libfreenect2::Frame> undistorted;
        boost::thread_specific_ptr<libfreenect2::Frame> registered;
        boost::thread_specific_ptr<libfreenect2::Frame> depth2rgb;

        ///mat containers
        //MatArr BuffData;

        ///printing
```

```
        boost::thread_specific_ptr<std::string> windowsname;
        boost::thread_specific_ptr<std::string> filename;
        boost::thread_specific_ptr<int> InternalId;

        std::string rgbwin = "RGB";
        std::string irwin = "ir";
        std::string dpwin = "depth";

    public:
        libfreenect2::FrameMap frames;
        bool enable_rgb = true;
        bool enable_depth = true;

        boost::mutex mutex;
        /// run
        size_t framemax = 300;

        //functions
        DeviceControl();
        DeviceControl(bool, bool);
        ~DeviceControl();

        void increment();
        void listeners(EnumDevices *DevSer, int idx);
        void start(EnumDevices *DevSer, int idx);
        void registrationini(EnumDevices *DevSer, int idx);
        void capture(EnumDevices *DevSer, int idx);
};

#include <iostream>
#include <stdlib.h>
#include "Devices.hpp"

EnumDevices::EnumDevices()
{
    /// [discovery]
    numdevices = freenect2.enumerateDevices();
    if (numdevices == 0)
    {
        std::cout << "no device connected!" << std::endl;
        exit(EXIT_FAILURE);
    }
}

void EnumDevices::getserials() {
    for (int id = 0; id < numdevices; id++) {
        serials.push_back(freenect2.getDeviceSerialNumber(id));
        pipeline[id] = 0;
    }
}

void EnumDevices::CreateDevice()
```

```
{
    for (int id = 0; id < numdevices; id++)
    {
        devs[id] = 0;

        if (pipeline[id]) {
            devs[id] = freenect2.openDevice(serials[id], pipeline[id]);
        }
        else {
            devs[id] = freenect2.openDevice(serials[id]);
        }

        if (devs[id] == 0)
        {
            std::cout << "failure opening device!" << std::endl;
            exit(EXIT_FAILURE);
        }
    }
}

void EnumDevices::savedata()
{
    for (int idx = 0; idx < serials.size(); idx++)
    {
        int InternalId = 0;
        std::string filename;
        cv::Mat temp1,temp2;

        while (!DataM[idx].empty())
        {
            MatArr BuffData = DataM[idx].front();
            DataM[idx].pop_front();
            InternalId++;

            //filename = "../images/" + boost::lexical_cast<std::string>(
                idx) + "_" + boost::lexical_cast<std::string>(InternalId) +
                 "_" + "RGB.png";
            //cv::imwrite(filename, BuffData.RGBM);
            filename = "../images/" + boost::lexical_cast<std::string>(idx
                ) + "_" + boost::lexical_cast<std::string>(InternalId) + "_
                " + "IR.png";
            BuffData.IRM.convertTo(temp1, CV_16U, 1, 0.0);
            cv::imwrite(filename, temp1);
            filename = "../images/" + boost::lexical_cast<std::string>(idx
                ) + "_" + boost::lexical_cast<std::string>(InternalId) + "_
                " + "DP.png";
            BuffData.DPM.convertTo(temp2, CV_16U, 1, 0.0);
            cv::imwrite(filename, temp2);
        }
    }
}
```

```
#pragma once
#include <boost/circular_buffer.hpp>
#include <boost/lexical_cast.hpp>
#include <opencv2/opencv.hpp>
#include <libfreenect2/libfreenect2.hpp>
#include <vector>
#include <array>
#include "DataStructures.hpp"

class EnumDevices {
    private:

    public:
        /// [context]
        libfreenect2::Freenect2 freenect2;
        std::string serial = "";
        std::vector<std::string> serials;
        int numdevices = 0;

        //devices
        libfreenect2::Freenect2Device *devs[10];
        libfreenect2::PacketPipeline *pipeline[10];

        //circular buffers per device
        using ArrMbuff = std::array<CBuff, 10>;
        using ArrVbuff = std::array<VBuff, 10>;
        ArrMbuff DataM;
        ArrVbuff DataV;

        EnumDevices();
        void getserials();
        void CreateDevice();
        void savedata();
};

#pragma once
#include <opencv2/opencv.hpp>
#include <libfreenect2/libfreenect2.hpp>

struct MatArr {
    //cv::Mat RGBM;
    cv::Mat DPM;
    cv::Mat IRM;
};

typedef boost::circular_buffer<MatArr> CBuff;
typedef std::vector<MatArr> VBuff;
```

Apéndice C

Código de evaluación de movimiento

```
% -*- MATLAB -*-
%
% Tracking - Evaluacion del movimiento
%
% Author: Marco A. Caballero Gro. (macfirstall_4@ciencias.unam.mx)
% Date: 25/Sep/2017
%

%
%Funcion principal del programa
%
%@param filename Archivo con las posiciones espaciales
%de los marcadores y el tiempo de adquisicion
%
%Ejemplo.-  main('test.csv', 0, ' test', 'L', 200);
%
function main(filename, numEx, idE, idH, nFrames)
    %Se verifica que el nombre del archivo no sea vacio
    if (strcmp(filename, ''))
        fprintf(' ---------------------------------------------- \n');
        fprintf(' **ERROR: Input filename was not specified \n');
        fprintf(' \n ');
        return;
    end

    suffixTitle = strcat(" - ", idE, " ( ", idH, " )");

    fps = 30;
    tInit = 1.0 / fps;
    tFin = nFrames / fps;
    framestep = 1;

    [numMarkers, V] = loadValues(filename);
    [pnts, results] = processValues(V, tInit, tFin, fps, framestep);
    statsProperties(numMarkers, pnts, results, idE, idH);
    plotProperties(V, results, tInit, tFin, fps, framestep, suffixTitle)
        ;
end
```

```
% -*- MATLAB -*-
%
% Tracking - Evaluacion del movimiento
%
% Author: Marco A. Caballero Gro. (macfirstall_4@ciencias.unam.mx)
% Date: 12/Dic/2017
%

%
% Funcion <plotProperties>
%
% @param V
%
function plotProperties(V, results, tInit, tFin, fps, framestep,
    suffixTitle)
    [k, numMarkers] = size(V);

    iInit = round(tInit * fps);
    iFin  = round( tFin * fps);


    nChnnls = 3;
    c = rand(numMarkers, nChnnls);
    c = [1.0 0.0 0.0 ; 0.0, 0.0, 1.0];

    m_name = {};
    prfx = 'marcador-';
    type = 'filled';
    type = 'o';

    fPos = [100 100 700 700];

    az = -45;
    el = 45;
    az = 0;
    el = 90;

    dlt = 0.5;
    cSize = 150;
    lWidth = 1.5;

    fontSize = 11;
    location = 'northeast';

    for m = 1 : numMarkers
       m_name{m} = strcat( prfx, num2str(m) );
     end


    xmins = []; xmaxs = [];
    ymins = []; ymaxs = [];
```

```
figure('pos', fPos);
for m = 1 : numMarkers
   m_coords = subsampleData(V{m}{1}, iInit, iFin, framestep);

   m_clr = c(m, :);

   xmins = [xmins min(m_coords(:, 1))];
   ymins = [ymins min(m_coords(:, 2))];

   xmaxs = [xmaxs max(m_coords(:, 1))];
   ymaxs = [ymaxs max(m_coords(:, 2))];

   h = scatter( m_coords(:, 1), m_coords(:, 2), cSize, m_clr, type )
       ;
   h.MarkerFaceColor = 1.0 - m_clr;
   hold on;
 end
hold off;

xmin = min(xmins) - dlt;  xmax = max(xmaxs) + dlt;
ymin = min(ymins) - dlt;  ymax = max(ymaxs) + dlt;

axis([xmin xmax ymin ymax]);
view(az, el);
legend(m_name, 'FontSize', fontSize, 'Location', location);
title(strcat('Posiciones de marcadores', suffixTitle));


spd = true;
acc = true;

if (spd)
   xmaxs = [];
   ymaxs = [];

   figure('pos', fPos);
   for m = 1 : numMarkers
      m_times = V{m}{2};
      m_clr = c(m, :);

      xmaxs = [xmaxs m_times(length(m_times))];
      ymaxs = [ymaxs max(results{m}('speed'))];

      plot(m_times, results{m}('speed'), 'Color', m_clr, 'LineWidth
          ', lWidth);
      hold on;
    end
   hold off;

   xmin = 0;  xmax = max(xmaxs);
```

```
        ymin = 0;   ymax = max(ymaxs) + dlt;

        axis([xmin xmax ymin ymax]);
        legend(m_name, 'FontSize', fontSize, 'Location', location);
        title(strcat('Velocidad', suffixTitle));
    end


    if (acc)
        xmaxs = [];
        ymins = [];   ymaxs = [];

        figure('pos', fPos);
        for m = 1 : numMarkers
            m_times = V{m}{2};
            m_clr = c(m, :);

            xmaxs = [xmaxs m_times(length(m_times))];

            ymins = [ymaxs min(results{m}('acceleration'))];
            ymaxs = [ymaxs max(results{m}('acceleration'))];

            plot(m_times, results{m}('acceleration'), 'Color', m_clr, '
                LineWidth', lWidth);
            hold on;
         end
        hold off;

        xmin = 0;   xmax = max(xmaxs);
        ymin = min(ymins) - dlt;   ymax = max(ymaxs) + dlt;

        axis([xmin xmax ymin ymax]);
        legend(m_name, 'FontSize', fontSize, 'Location', location);
        title(strcat('Aceleracion', suffixTitle));
    end

end
```

```
% -*- MATLAB -*-
%
% Tracking - Evaluacion del movimiento
%
% Author: Marco A. Caballero Gro. (macfirstall_4@ciencias.unam.mx)
% Date: 07/Dic/2017
%

%
% Funcion <statsProperties>
%
% @param numMarkers
%
function statsProperties(numMarkers, pnts, results, example, hand)
    delimiter = ',';

    header = strcat('Example', delimiter, 'Hand', delimiter, 'Marker');
    header = strcat( header, delimiter, 'Dst(Avg)', delimiter, 'Dst(
        Stdev)', delimiter, 'Spd(Avg)');
    header = strcat( header, delimiter, 'Spd(Stdev)', delimiter, 'Acc(
        Avg)', delimiter, 'Acc(Stdev)');
    header = strcat( header, delimiter, 'Smt(Avg)', delimiter, 'Smt(
        Stdev)', '\n');
    fprintf(header);

    content = strcat(example, delimiter, hand);

    for m = 1 : numMarkers
        m_coords = pnts{m};

        values = strcat( content, delimiter, num2str(m) );

        ctr = mean( m_coords(:, 1:2) );
        dx = m_coords(:, 1) - ctr(1);
        dy = m_coords(:, 2) - ctr(2);
        dst = sqrt( (dx .* dx) + (dy .* dy) );

        values = strcat( values, delimiter,
            num2str( mean(dst) ), delimiter,
            num2str( std(dst) ) );
        values = strcat( values, delimiter,          num2str( mean(results
            {m}('speed')) ), delimiter,          num2str( std(results{m}('
            speed')) ) );
        values = strcat( values, delimiter,  num2str( mean(results{m}('
            acceleration')) ), delimiter, num2str( std(results{m}('
            acceleration')) ) );
        values = strcat( values, delimiter,     num2str( mean(results{m}('
            smoothness')) ), delimiter,    num2str( std(results{m}('
            smoothness')) ) );
        values = strcat( values, '\n');
```

```
        fprintf(values);
    end

end

% -*- MATLAB -*-
%
% Tracking - Evaluacion del movimiento
%
% Author: Marco A. Caballero Gro. (macfirstall_4@ciencias.unam.mx)
% Date: 25/Sep/2017
%

%
% Funcion <subsampleData>
%
function result = subsampleData(data, iInit, iFin, framestep)
    [numData, c] = size(data);

    jFin = iFin;
    if (iFin > numData)
        jFin = numData;
    end

    result = [];
    for k = iInit : jFin
        if ( mod(k, framestep) == 0 )
            result = vertcat(result, data(k, 1 : c));
        end
    end
end

% -*- MATLAB -*-
%
% Tracking - Evaluacion del movimiento
%
% Author: Marco A. Caballero Gro. (macfirstall_4@ciencias.unam.mx)
% Date: 25/Sep/2017
%

%
% Funcion <processValues>
%
% @param V
%
% @return results
%
function [pnts, results] = processValues(V, tInit, tFin, fps, framestep
    )
    [k, numMarkers] = size(V);

    results = {};
```

```
    pnts = {};
    for m = 1 : numMarkers
        m_coords = V{m}{1};
        m_times = V{m}{2};

        props = getProperties(m_coords, m_times);
        pnts{m} = m_coords;
        results{m} = props;
    end
end

%
% Funcion <getProperties>
%
% @param coords
% @param times
%
% @return props
%
function props = getProperties(coords, times)
    [numPnts, nCols] = size(coords);

    spd = [0];
    acc = [0];
    smt = [0];

    dlt = 0.0001;

    fCoords = vertcat(coords, coords(numPnts, :));
    fTimes = vertcat(times, times(numPnts) + dlt);

    for i = 2 : numPnts
        dP1 = sum((fCoords(i, 1:2) - fCoords(i - 1, 1:2)).^2).^0.5;
        dP2 = sum((fCoords(i + 1, 1:2) - fCoords(i, 1:2)).^2).^0.5;

        dT1 = fTimes(i) - fTimes(i - 1);
        dT2 = fTimes(i + 1) - fTimes(i);

        v = ((dP1 / dT1) + (dP2 / dT2)) * 0.5;
        spd = [spd, v];
    end

    fSpd = [spd, spd(numPnts)];

    for i = 2 : numPnts
        dV1 = fSpd(i) - fSpd(i - 1);
        dV2 = fSpd(i + 1) - fSpd(i);

        dT1 = fTimes(i) - fTimes(i - 1);
        dT2 = fTimes(i + 1) - fTimes(i);
```

```
        a = ((dV1 / dT1) + (dV2 / dT2)) * 0.5;
        acc = [acc, a];
    end

    fAcc = [acc, acc(numPnts)];

    for i = 2 : numPnts
        dA1 = fAcc(i) - fAcc(i - 1);
        dA2 = fAcc(i + 1) - fAcc(i);

        dT1 = fTimes(i) - fTimes(i - 1);
        dT2 = fTimes(i + 1) - fTimes(i);

        s = ((dA1 / dT1) + (dA2 / dT2)) * 0.5;
        smt = [smt, s];
    end

    keys = {'speed', 'acceleration', 'smoothness'};
    values = {spd, acc, smt};

    props = containers.Map(keys, values);
end

% -*- MATLAB -*-
%
% Tracking - Evaluacion del movimiento
%
% Author: Marco A. Caballero Gro. (macfirstall_4@ciencias.unam.mx)
% Date: 25/Sep/2017
%

%
% Funcion <loadValues>
%
% @param filename
%
function [numMarkers, V] = loadValues(filename)
    data = readValues(filename);

    markers = unique(data(:, 1));
    [numMarkers, k] = size(markers);

    V = {};
    for m = 1 : numMarkers
        pp = data(:, 1) == m;

        coords = data(pp, 2 : 4);
        times = data(pp, 5);

        W = {};
        W{1} = coords;
        W{2} = times;
```

```
        V{m} = W;
    end
end

% -*- MATLAB -*-
%
% Tracking - Evaluacion del movimiento
%
% Author: Marco A. Caballero Gro. (macfirstall_4@ciencias.unam.mx)
% Date: 22/Oct/2017
%

%
% Funcion <readValues>
%
% @param filename
%
function data = readValues(filename)
    delimiterIn = ',';
    headerlinesIn = 0;
    data = importdata(filename, delimiterIn, headerlinesIn);
end
```

Bibliografía

[1] F Parada-Loira, E González-Agulla, and J L Alba-Castro. Hand gestures to control infotainment equipment in cars. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 1–6, 2014.

[2] K F Li, K Lothrop, E Gill, and S Lau. A Web-Based Sign Language Translator Using 3D Video Processing. In *Network-Based Information Systems (NBiS), 2011 14th International Conference on*, pages 356–361, sep 2011.

[3] Cao Dong, M C Leu, and Z Yin. American Sign Language alphabet recognition using Microsoft Kinect. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 44–52, 2015.

[4] R. Suriya and V. Vijayachamundeeswari. A survey on hand gesture recognition for simple mouse control. In *Information Communication and Embedded Systems (ICICES), 2014 International Conference on*, pages 1–5, Feb 2014.

[5] C J Cohen, G Beach, and G Foulk. A basic hand gesture control system for PC applications. In *Applied Imagery Pattern Recognition Workshop, AIPR 2001 30th*, pages 74–79, oct 2001.

[6] Wikipedia; the free encyclopedia. *Motion Capture from Wikipedia*. `https://en.wikipedia.org/wiki/Motion_capture`.

[7] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[8] X. Ren and J. Malik. Learning a classification model for segmentation. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 10–17 vol.1, Oct 2003.

[9] G. Mori. Guiding model search using segmentation. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1417–1423 Vol. 2, Oct 2005.

[10] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, Nov 2012.

[11] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

[12] Andrea Vedaldi and Stefano Soatto. *Quick Shift and Kernel Methods for Mode Seeking*, pages 705–718. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[13] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, Jun 1991.

[14] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2290–2297, Dec 2009.

[15] Serge Beucher and Fernand Meyer. *The morphological approach to segmentation: The watershed transformation*, volume Vol. 34, page 433–481. 01 1993.

[16] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.

[17] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[18] Open Source Computer Vision Library OpenCV 3.3. *Optical Flow from OpenCV*. `https://docs.opencv.org/trunk/d7/d8b/tutorial_py_lucas_kanade.html`.

[19] Pakorn Kaewtrakulpong and Richard Bowden. An improved adaptive background mixture model for realtime tracking with shadow detection. 05 2002.

[20] Z. Zhang and F. Huang. Hand tracking algorithm based on superpixels feature. In *Information Science and Cloud Computing Companion (ISCC-C), 2013 International Conference on*, pages 629–634, Dec 2013.

[21] R. T. Collins. Mean-shift blob tracking through scale space. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–234–40 vol.2, June 2003.

[22] Akinori Sasaki, Sho Yokota, Yasuhiro Ohyama, and Hiroshi Hashimoto. Hand pose estimation from camera images of ring-shaped markers. In *2009 ICCAS-SICE*, pages 3255–3259, Aug 2009.

[23] A. W. Fitzgibbon, M. Pilu, and R. B. Fisher. Direct least squares fitting of ellipses. In *Proceedings of 13th International Conference on Pattern Recognition*, volume 1, pages 253–257 vol.1, Aug 1996.

[24] S. Sridhar, H. Rhodin, H. P. Seidel, A. Oulasvirta, and C. Theobalt. Real-time hand tracking using a sum of anisotropic gaussians model. In *2014 2nd International Conference on 3D Vision*, volume 1, pages 319–326, Dec 2014.

[25] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, Jul 1990.

[26] Wikipedia; the free encyclopedia. *Kinect from Wikipedia*. `https://en.wikipedia.org/wiki/Kinect`.

[27] Microsoft; Windows Dev Center. *Kinect for Windows SDK*. `https://msdn.microsoft.com/library/dn799271.aspx`.

[28] M. Zabri Abu Bakar, R. Samad, D. Pebrianti, and N. L. Y. Aan. Real-time rotation invariant hand tracking using 3d data. In *Control System, Computing and Engineering (ICCSCE), 2014 IEEE International Conference on*, pages 490–495, Nov 2014.

[29] PrimeSense. *NiTE, Natural Interaction Middleware*. `http://openni.ru/files/nite/index.html`.

[30] P. Usachokcharoen, Y. Washizawa, and K. Pasupa. Sign language recognition with microsoft kinect's depth and colour sensors. In *2015 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 186–190, Oct 2015.

[31] OpenKinect. *libfreenect2, Driver for Kinect for Windows v2 (K4W2) devices*. `https://github.com/OpenKinect/libfreenect2`.

[32] Boost Org; Anthony Williams. *Boost C++ Libraries*. `http://www.boost.org/doc/libs/1_38_0/doc/html/thread.html`.

[33] Mathworks. *Stereo Calibration App from Matlab 2016*. `https://www.mathworks.com/help/vision/ug/stereo-camera-calibrator-app.html`.

[34] OpenCV Foundation and Itseez Inc. *OpenCV 3.1, Open Source Computer Vision Library*. `http://opencv.org/opencv-3-1.html`.