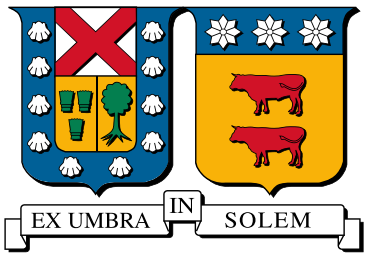
UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

DEPARTAMENTO DE ELECTRONICA

VALPARAISO - CHILE



"ANÁLISIS DE TROTE SIN MARCADORES EN PLANO SAGITAL CON DEEPLABCUT"

LUCAS FELIPE CORTÉS GUTIÉRREZ

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICO, MENCIÓN COMPUTADORES, SUBMENCIÓN CONTROL E INSTRUMENTACIÓN

PROFESOR GUIA: PHD. MATIAS ZAÑARTU
PROFESORES CORREFERENTES: PHD. ALEJANDRO WEINSTEIN
MSC. IVER CRISTI

JUNIO 2022

Agradecimientos

Agradezco por sobre todo a mi familia, mi madre *Nancy* y mis hermanos *Mariano* y *Sebastián*, por su apoyo incondicional durante mi tiempo como estudiante, y sin cuya ayuda difícilmente hubiera podido terminar este largo proceso.

Al centro clínico *You Just Better*, por la disposición de sus instalaciones, y a todos los voluntarios quienes de forma desinteresada cedieron el uso de su imagen para esta investigación.

A *Iver Cristi*, por su constante preocupación, disposición, y ayuda, que demostró ser instrumental a la hora de solucionar problemas y de realizar la validación clínica de los resultados.

A *Nicole Cedeño*, por su paciencia, ánimo, y dedicación, siempre dispuesta a ayudar en lo que fuera necesario, cuyo apoyo incesante permitió que mis esfuerzos llegaran a buen puerto.

A *Matías Zañartu*, quien por lejos ha sido el profesor más dedicado que he tenido, siempre dispuesto a hacer todo lo posible por solucionar los problemas que se me presentaron, agradezco profundamente la ayuda y el voto de confianza depositado en mí.

A *Alejandro Weinstein*, quien siempre se mostró muy entusiasmado con mis resultados y nunca dudó en expresarlo, agradezco sus consejos y críticas constructivas, así como su disposición y ayuda en lo que fuera necesario.

A la empresa *Lanek* por ofrecerme un espacio en el cual poder trabajar, aprender y darme la oportunidad de ser proactivo, además de brindarme la flexibilidad necesaria para compatibilizar mis responsabilidades académicas y laborales.

A mi familia,

a mis amigos,

y a todo quien me acompañó durante este viaje.

Análisis de Trote sin Marcadores en Plano Sagital con DeepLabCut

Lucas Felipe Cortés Gutiérrez

Memoria para optar al título de Ingeniero Civil Electrónico, mención Computadores, submención Control e Instrumentación.

Universidad Técnica Federico Santa María

Profesor Guía: PhD. Matías Zañartu

Junio 2022

Resumen

En el campo de la medicina existe la necesidad de estudiar el movimiento de las personas mediante el uso de videos. Aquí nace el uso de marcadores activos y pasivos para realizar seguimiento de los movimientos del usuario, sin embargo, estos presentan una serie de problemas importantes, entre los que destacan su tamaño, voluptuosidad, incomodidad y desprendimiento accidental durante el uso.

Es en este contexto que surge la pregunta, ¿se puede obtener un desempeño comparable al uso de marcadores prescindiendo de estos? en caso contrario, ¿vale la pena la pérdida de precisión versus la comodidad del usuario?

Para la realización de un análisis más rápido y consistente se recurre a herramientas computacionales modernas, como la visión por computador y la inteligencia artificial, con especial énfasis en las implementaciones de redes neuronales en *TensorFlow*. En este caso se hará uso de una herramienta recientemente desarrollada, *DeepLabCut*.

Palabras Clave: Marcadores activos, Visión por computador, Inteligencia artificial, Redes Neuronales, TensorFlow, DeepLabCut.

Sagital Plane Markerless Gait Analysis with DeepLabCut

Lucas Felipe Cortés Gutiérrez

Thesis for the fulfillment of the B.S. in Electronic Engineering, major in Computers, minor in Control and Instrumentation degree (6 year program).

Universidad Técnica Federico Santa María

Advisor: PhD. Matías Zañartu

June 2022

Abstract

In the medical field there is a need to study the movement of people through the use of videos. Here is where the use of active and passive markers to track the movements of the user was born, however, these have a number of major problems, among which are their size, bulkiness, discomfort and accidental detachment during use.

It is in this context that the question arises, can a comparable performance to the use of markers be obtained without using them? if not, is it worth the loss of accuracy or reliability versus user comfort?

Modern computational tools, such as computer vision and artificial intelligence, with special emphasis on neural networks and *TensorFlow* implementations, are used for faster and more consistent analysis. In this case, we will make use of a recently developed tool, *DeepLabCut*.

Keywords: Active markers, Computer vision, Artificial intelligence, Neural networks, TensorFlow, DeepLabCut.

Glosario

closed-source De código cerrado. 7

dataset Conjunto de datos capturados para la realización del experimento. 5

downsampling Proceso de submuestreo de una señal con el fin de reducir su dimensionalidad. 5

Kinect Cámara 3D con detección de profundidad desarrollada por *Microsoft* para su línea de productos *Xbox*. 9

marcador activo Marcador de posición iluminado que se adhiere al cuerpo o a la ropa del paciente. 8, 11

marcador pasivo Marcador de posición no-iluminado o reflectante que se adhiere al cuerpo o a la ropa del paciente. 9, 12

markerless Sistema que no hace uso de marcadores. 6

open-source De código abierto. 12

plano sagital En anatomía, aquellos planos perpendiculares al suelo y en ángulo recto con los planos frontales, que dividen al cuerpo en mitades, izquierda y derecha. 5

ResNet-50 Red neuronal residual de 50 capas. 18

semi-markerless Sistema que hace menor uso de marcadores. 6

Acrónimos

AC3E Advanced Center for Electrical and Electronic Engineering. 8

AI Artificial Intelligence. 2

ARBA Advanced Running Biomechanical Analysis. ix, 2, 5, 6, 8, 16, 17, 24–28, 46, 49, 124

CV Computer Vision. 1

DLC DeepLabCut. 4–6, 16, 17, 24–28, 49, 112

dtw() Distance Between Signals Using Dynamic Time Warping. 40

EMG Electromiografía. 12, 13

IMU Inertial Measurement Unit. 12, 13

ResNet Residual Neural Network. 18

RGB-D Red Green Blue - Depth. 9

ROI Region Of Interest. 18

SDK Software Development Kit. 3

UTFSM Universidad Técnica Federico Santa María. 8

UV Universidad de Valparaíso. 8

Índice de contenidos

1. Introducción **1**

1.1. Contexto 1

1.2. Problema a solucionar 2

1.3. Alternativas de solución 2

1.4. Características de comparación 3

1.5. Características de comparación 4

1.6. Selección 4

2. Objetivos del trabajo **5**

2.1. Trabajo a desarrollar 5

2.2. Evaluaciones a realizar 6

2.3. Resultados esperados 6

3. Estado del arte **7**

3.1. ARBA 8

3.2. MotionMetrix 3D Running Gait 9

3.3. GaitON Running Gait Analysis 11

3.4. Qualisys Track Manager 12

3.5. Noraxon-Ninox 13

3.6. Vicon Motion Systems 15

4. Metodología **16**

4.1. Objetivo 1 - Protocolo de evaluación y captura de datos 16

4.2. Objetivo 1 - Protocolo de procesamiento de datos 17

4.3. Objetivo 2 - Métricas y estadísticas de comparación 24

5. Resultados **25**

6. Conclusiones **49**

Referencias **50**

7. Anexos **52**

7.1. Imágenes . . . 52

7.1.1. Sujeto 4, plano sagital derecho . . . 52

7.1.2. Sujeto 5, plano sagital izquierdo . . . 62

7.1.3. Sujeto 5, plano sagital derecho . . . 72

7.1.4. Sujeto 6, plano sagital izquierdo . . . 82

7.1.5. Sujeto 7, plano sagital izquierdo . . . 92

7.1.6. Sujeto 8, plano sagital izquierdo . . . 102

7.2. Códigos . . . 112

Índice de tablas

1.1. Comparación de alternativas. . . . 4

5.2. Errores de evaluación de la red. . . . 25

5.3. Errores FFT, sujeto 5R. . . . 38

5.4. Errores ángulo articular, sujeto 5R. . . . 39

5.5. Distancias absolutas y medias, sujeto 5R. . . . 46

5.6. Resumen errores ángulo articular. . . . 46

5.7. Resumen distancias absolutas y medias eje x. . . . 47

5.8. Resumen distancias absolutas y medias eje y. . . . 47

7.9. Errores, Sujeto 4R. . . . 58

7.10. Errores ángulo articular, Sujeto 4R. . . . 59

7.11. Distancias absolutas y medias, Sujeto 4R. . . . 61

7.12. Errores, Sujeto 5L. . . . 68

7.13. Errores ángulo articular, Sujeto 5L. . . . 69

7.14. Distancias absolutas y medias, Sujeto 5L. . . . 71

7.15. Errores FFT, Sujeto 5R. . . . 78

7.16. Errores ángulo articular, Sujeto 5R. . . . 79

7.17. Distancias absolutas y medias, Sujeto 5R. . . . 81

7.18. Errores, Sujeto 6L. . . . 88

7.19. Errores ángulo articular, Sujeto 6L. . . . 89

7.20. Distancias absolutas y medias, Sujeto 6L. 91
7.21. Errores, Sujeto 7L. 98
7.22. Errores ángulo articular, Sujeto 7L. 99
7.23. Distancias absolutas y medias, Sujeto 7L. 101
7.24. Errores, Sujeto 8L. 108
7.25. Errores ángulo articular, Sujeto 8L. 109
7.26. Distancias absolutas y medias, Sujeto 8L. 111

Índice de figuras

3.1. Puntos anatómicos a seguir. 7
3.2. Funcionamiento del sistema *Advanced Running Biomechanical Analysis (ARBA).* 8
3.3. Funcionamiento del sistema MotionMetrix. 10
3.4. Funcionamiento del sistema GaitON. 11
3.5. Funcionamiento del sistema Qualisys. 12
3.6. Ejemplos del sistema Noraxon-Ninox. 13
3.7. Funcionalidades del sistema Noraxon-Ninox. 14
3.8. Ejemplos del sistema *Vicon Capture.U.* 15
4.9. Protocolo de evaluación y captura de datos. 16
4.10. Protocolo de procesamiento de datos. 17
4.11. Ejecución del software. 19
4.12. Pestaña de bienvenida. 19
4.13. Gestión de proyectos. 19
4.14. Extracción de frames. 20
4.15. Etiquetado de frames. 20
4.16. Ejemplo de etiquetado. 20
4.17. Creación de datos de entrenamiento. 21
4.18. Entrenamiento de red. 21
4.19. Evaluación de red. 21
4.20. Editor de videos. 22
4.21. Analizar videos. 22

4.22. Refinar tracklets. 22
4.23. Crear videos de salida. 23
4.24. Extraer frames aislados. 23
4.25. Refinar etiquetas. 23
4.26. Protocolo de comparación de resultados y métricas. 24
5.27. Posiciones x-y originales. 26
5.28. Posiciones x-y alineadas. 27
5.29. Posiciones x-y filtradas. 28
5.30. Histogramas de posición sujeto 5R. 29
5.31. Histogramas de posición alineados sujeto 5R. 30
5.32. Histogramas de posición filtrados sujeto 5R. 31
5.33. FFT rodilla x, sujeto 5R. 32
5.34. FFT rodilla x, sujeto 5R. 33
5.35. FFT tobillo x, sujeto 5R. 34
5.36. FFT tobillo y, sujeto 5R. 35
5.37. FFT metatarso x, sujeto 5R. 36
5.38. FFT metatarso y, sujeto 5R. 37
5.39. Comparación de ángulos y error medio del sujeto 5R. 39
5.40. Medida de distancia absoluta entre las señales, rodilla eje x. 40
5.41. Medida de distancia absoluta entre las señales, rodilla eje y. 41
5.42. Medida de distancia absoluta entre las señales, tobillo eje x. 42
5.43. Medida de distancia absoluta entre las señales, tobillo eje y. 43
5.44. Medida de distancia absoluta entre las señales, metatarso eje x. 44
5.45. Medida de distancia absoluta entre las señales, metatarso eje y. 45
5.46. Videos de salida, sujeto 5R. 48
7.47. Sujeto 4R. 52
7.48. Gráficas de posición, Sujeto 4R. 53
7.49. Histogramas de posición, Sujeto 4R. 54
7.50. Histogramas de posición alineados, Sujeto 4R. 55
7.51. Histogramas de posición filtrados, Sujeto 4R. 56
7.52. FFT de las señales, Sujeto 4R. 57

7.53. Comparación de ángulos y error medio, Sujeto 4R. 59
7.54. Distancia absoluta entre las señales, Sujeto 4R. 60
7.55. Videos de salida, Sujeto 4R. 61
7.56. Sujeto 5L. 62
7.57. Gráficas de posición, Sujeto 5L. 63
7.58. Histogramas de posición, Sujeto 5L. 64
7.59. Histogramas de posición alineados, Sujeto 5L. 65
7.60. Histogramas de posición filtrados, Sujeto 5L. 66
7.61. FFT de las señales, Sujeto 5L. 67
7.62. Comparación de ángulos y error medio, Sujeto 5L. 69
7.63. Distancia absoluta entre las señales, Sujeto 5L. 70
7.64. Videos de salida, Sujeto 5L. 71
7.65. Sujeto 5R. 72
7.66. Gráficas de posición, Sujeto 5R. 73
7.67. Histogramas de posición, Sujeto 5R. 74
7.68. Histogramas de posición alineados, Sujeto 5R. 75
7.69. Histogramas de posición filtrados, Sujeto 5R. 76
7.70. FFT de las señales, Sujeto 5R. 77
7.71. Comparación de ángulos y error medio, Sujeto 5R. 79
7.72. Distancia absoluta entre las señales, Sujeto 5R. 80
7.73. Videos de salida, Sujeto 5R. 81
7.74. Sujeto 6L. 82
7.75. Gráficas de posición, Sujeto 6L. 83
7.76. Histogramas de posición, Sujeto 6L. 84
7.77. Histogramas de posición alineados, Sujeto 6L. 85
7.78. Histogramas de posición filtrados, Sujeto 6L. 86
7.79. FFT de las señales, Sujeto 6L. 87
7.80. Comparación de ángulos y error medio, Sujeto 6L. 89
7.81. Distancia absoluta entre las señales, Sujeto 6L. 90
7.82. Videos de salida, Sujeto 6L. 91
7.83. Sujeto 7L. 92

7.84. Gráficas de posición, Sujeto 7L. 93
7.85. Histogramas de posición, Sujeto 7L. 94
7.86. Histogramas de posición alineados, Sujeto 7L. 95
7.87. Histogramas de posición filtrados, Sujeto 7L. 96
7.88. FFT de las señales, Sujeto 7L. 97
7.89. Comparación de ángulos y error medio, Sujeto 7L. 99
7.90. Distancia absoluta entre las señales, Sujeto 7L. 100
7.91. Videos de salida, Sujeto 7L. 101
7.92. Sujeto 8L. 102
7.93. Gráficas de posición, Sujeto 8L. 103
7.94. Histogramas de posición, Sujeto 8L. 104
7.95. Histogramas de posición alineados, Sujeto 8L. 105
7.96. Histogramas de posición filtrados, Sujeto 8L. 106
7.97. FFT de las señales, Sujeto 8L. 107
7.98. Comparación de ángulos y error medio, Sujeto 8L. 109
7.99. Distancia absoluta entre las señales, Sujeto 8L. 110
7.100Videos de salida, Sujeto 8L. 111

Índice de códigos

7.1. Importación de datos 112
7.2. Alineación de señales 113
7.3. Filtrado de las señales 116
7.4. Gráfico de posiciones 116
7.5. Histograma de posiciones 119
7.6. Cálculo de FFT 121
7.7. Cálculo de ángulo 122
7.8. Cálculo de distancias absolutas 124
7.9. Extracto de implementación de servicios 124
7.10. Servicio de filtrado de datos en Python 126
7.11. Servicio gráfico en Python 135

7.12. Servicio de reportería en Python 145

1. Introducción

1.1. Contexto

El movimiento es fundamental para el desarrollo de todo ser humano en un contexto personal, social y económico. Padecer alguna patología que limite el movimiento normal se traduce en una significativa disminución de salud y calidad de vida. Es por esta razón que se realizan análisis biomecánicos en un entorno clínico de forma habitual, y la cantidad de personas (pacientes, deportistas) que los requieren ya sea por orden médica, o por iniciativa propia han aumentado considerablemente en los últimos años, así como también lo han hecho las tecnologías disponibles para facilitar dicho análisis.[1–6]

Es en este contexto que nace la iniciativa de integrar tecnologías modernas existentes en el análisis biomecánico, el cual por lo general es realizado manualmente por un especialista. Entre las tecnologías aplicadas están el análisis de imágenes y la *Computer Vision (CV)*, con estas se logra hacer un seguimiento preciso a una serie de marcadores activos posicionados en puntos de interés sobre el cuerpo humano. Estas pueden ser aplicadas tanto en 2D como en 3D, siendo la alternativa bidimensional la más común, debido a su fácil implementación y al uso de hardware no especializado. En cuanto a la alternativa tridimensional, el análisis espacial del movimiento supone nuevos desafíos, desde la adquisición de hardware especializado y prohibitivo en precio, a la complejidad de su implementación en software, teniendo que lidiar con herramientas matemáticas avanzadas necesarias para interpretar el desfase espacial de múltiples cámaras en información tridimensional de posición del usuario. Este tipo de sistemas son por lo general muy precisos y complejos.

Lo anterior es, sin embargo, una solución imperfecta, ya que con la llegada de nuevas soluciones también vienen nuevos problemas, entre los que se puede mencionar:

- Incomodidad ante el uso de los marcadores.
- Su voluptuosidad.
- Desprendimiento accidental durante su uso.
- Reducida autonomía antes de que necesiten ser recargados (lo que impone una cota superior a la cantidad de exámenes que se pueden realizar consecutivamente).

- Poca disponibilidad para realizar reemplazos.
- Limitadas opciones de reparación debido a la especialización, especificidad y dimensionalidad de sus componentes.
- La sensibilidad a condiciones ambientales que presentan las cámaras al momento de capturar el movimiento de los marcadores en condiciones lumínicas no ideales.

Muchos de estos problemas pueden ser resueltos mediante el uso de herramientas de *Artificial Intelligence (AI)*.

1.2. Problema a solucionar

En la actualidad, es de interés clínico el análisis del movimiento del paciente. Este proceso requiere de un etiquetado, seguimiento y análisis manual por parte de un profesional del área de la salud. Mediante dicho procedimiento se consigue reconstruir y modelar la cinemática de los miembros y calcular los ángulos que son formados por dichos puntos de interés, para así analizar el trote del usuario, además de apoyar en el diagnóstico médico y en la prevención y recuperación de lesiones.

1.3. Alternativas de solución

DeepLabCut

Es un método eficiente de estimación de posición 3D[7] en múltiples sujetos[8] sin marcadores, basado en redes neuronales[9], es *open-source, GPLv3*[1], además de ser rápido y robusto.

Este software es capaz de hacer un seguimiento preciso a partes específicas de diversos animales[10–12], ya sea con modelos pre-entrenados[13, 14] o con un entrenamiento propio. Existe la posibilidad de crear un *dataset* específico para el problema a solucionar, entrenar la red con el mismo y comparar los resultados con el sistema de *ARBA*(Advanced Running Biomechanical Analysis). Este paso de solución aplica para todas las alternativas propuestas.

[1] https://github.com/DeepLabCut/DeepLabCut

YOLOv3

Sistema *open-source, GPLv3*[2], es un sistema de detección de objetos en tiempo real que funciona en base a la aplicación de una red neuronal singular a la imagen completa. Esta red divide la imagen en regiones, predice las *bounding boxes* y las probabilidades para cada región, las que son pesadas de acuerdo a sus probabilidades. La aplicación de una sola red a toda la imagen otorga a este sistema clasificador una gran velocidad[15].

Microsoft Kinect

Sistema con *Software Development Kit (SDK) open-source, MIT*[3], que a diferencia del resto de las alternativas, esta cámara es capaz de medir profundidad con gran precisión, y ha sido utilizada en múltiples ocasiones para el análisis de trote, tanto en su versión 1[16], como en su versión 2[17]. El principal problema recae en que *Microsoft* descontinuó la producción del dispositivo, por lo que su uso a futuro parece incierto.

1.4. Características de comparación

En la subsección 1.3 se analizaron diferentes alternativas de solución para el problema planteado, y para cada alternativa propuesta se crearon listas de ventajas y desventajas, además de un cuadro comparativo de características y la ponderación (en porcentaje) de importancia de cada una de ellas.

Se evaluará cada característica de acuerdo a la tabla 1.1, con un puntaje de 0 % a 100 %, se ponderarán los resultados y se les asignará una nota final.

[2] https://pjreddie.com/darknet/yolo/
[3] https://github.com/microsoft/Azure-Kinect-Sensor-SDK

1.5. Características de comparación

Tabla 1.1: Comparación de alternativas.

Característica	**DeepLabCut**	**YOLOv3**	**Kinect**	**Ponderación**
Open-source	✓	✓	✓✗	20 %
Tiempo de entrenamiento	✓✗	✓✗	✓	5 %
Modelos pre-entrenados	✓	✓	✓	15 %
Hardware no especializado	✓	✓	✗	20 %
Desarrollo activo	✓	✓✗	✗	10 %
Precisión	✓✗	✓✗	✓	10 %
Documentación	✓	✓	✓	10 %
Relevancia/novedad	✓	✓✗	✓✗	10 %
Nota final	92,5	82,5	55	

1.6. Selección

En base a la nota obtenida, la flexibilidad que ofrece, la alternativa de uso de un modelo pre-entrenado, y la posibilidad de utilización de hardware no especializado, se determina que la alternativa de solución seleccionada es ***DeepLabCut (DLC)***. Además, la novedad del sistema permite realizar una investigación de mayor relevancia.

2. Objetivos del trabajo

El objetivo general es procesar el trote en plano sagital humano mediante una red neuronal en *DLC*, para en análisis biomecánico.

Como objetivos específicos se tiene:

Acondicionamiento: Acondicionar la red neuronal de *DLC*[4] al caso humano.

Validación: Validar clínicamente la fiabilidad del seguimiento mediante *DLC* versus el sistema *ARBA*.

2.1. Trabajo a desarrollar

El trabajo a desarrollar consistió en la recolección de datos de entrenamiento, i.e. la captura de videos del *plano sagital* en distintas condiciones lumínicas/ambientales para conseguir así un producto más robusto, luego de esto se realizó un *downsampling* de los videos obtenidos, rescatando 20 cuadros utilizables para el entrenamiento por video, cada uno de estos cuadros fue etiquetado manualmente con los puntos anatómicos de interés para el especialista, los que se utilizaron como referencia para el entrenamiento del sistema, el cual se extendió por un tiempo de 4 meses, periodo en el cual se obtuvo validación clínica por parte de un especialista.

Tareas asociadas a los objetivos

- Conseguir voluntarios (acondicionamiento).
- Tomar videos de voluntarios (acondicionamiento).
- Crear *dataset* (acondicionamiento).
- Crear datos de entrenamiento mediante el *downsampling* de los videos capturados (acondicionamiento).
- Etiquetar datos de entrenamiento (acondicionamiento).
- Realizar entrenamiento (acondicionamiento).
- Evaluar modelo obtenido (validación).
- Analizar videos (validación).

[4] www.mackenziemathislab.org/deeplabcut

- Etiquetar videos (validación).
- Comparar resultados (validación).

2.2. Evaluaciones a realizar

La principal forma de evaluación corresponderá a una comparación de desempeño entre los datos entregados por *DLC* versus *ARBA*. Para esto, se deberá contar con la validación de un profesional de la salud en el área kinesiológica.

Se espera lograr una comparación matemática objetiva entre ambos sistemas, mediante el contraste de los factores de forma resultantes entregados por los gráficos de posición el el tiempo, además de una comparativa cualitativa del error medio entre los ángulos articulares formados.

2.3. Resultados esperados

Se espera poder realizar un análisis en profundidad, con fundamentos matemáticos del movimiento de las extremidades inferiores en el plano sagital, y, que a través del entrenamiento y múltiples pruebas realizadas mediante *DLC* en videos en un entorno clínico y en condiciones del mundo real, se obtenga un desempeño comparable al método del sistema *ARBA* y así poder implementar una versión *markerless* o *semi-markerless* a futuro.

3. Estado del arte

En los últimos años han nacido diversas soluciones que hacen uso de herramientas computacionales modernas. En el mercado existen variadas opciones que dan solución al problema planteado, sin embargo, en general, son de un alto costo comercial tanto en software como en hardware especializado y su implementación no es sencilla. Adicionalmente, estos sistemas suelen ser incapaces de incorporar nuevas funcionalidades y sumado a lo anterior, estas soluciones son en su gran mayoría *closed-source* y no pueden ser mejoradas ni modificadas. Esto decanta en una serie de problemas en lo que refiere a privacidad, desde el manejo o manipulación de la información del usuario hasta problemas con la recolección de datos.

Cabe destacar que todas las soluciones y sistemas presentados a continuación tiene distintas ventajas y desventajas, lo que hace necesaria e imperante la realización de investigación exhaustiva que favorezca en el desarrollo de nuevas tecnologías de análisis de movimiento.

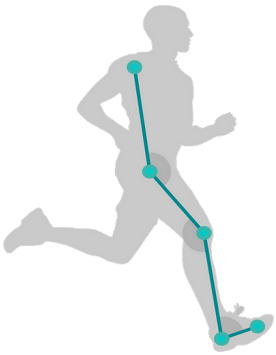
Figura 3.1: Puntos anatómicos a seguir.

3.1. ARBA[5]

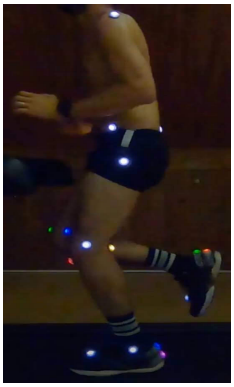
Uno de los sistemas que resuelve este problema es *ARBA*, desarrollado por la empresa Lanek[6] *spin off* del *Advanced Center for Electrical and Electronic Engineering (AC3E)*[7] compuesto po académicos de la *Universidad Técnica Federico Santa María (UTFSM)*[8] y la *Universidad de Valparaíso (UV)*[9]. Este sistema funciona en base a cámaras, *marcadores activos* y un software desarrollado para la adquisición de datos de posición de los distintos puntos de interés, con el fin de realizar seguimiento al movimiento de los marcadores, analizar el trote del usuario y apoyar al diagnóstico médico en la prevención de lesiones.

Actualmente *ARBA* es capaz de entregar los siguientes resultados:

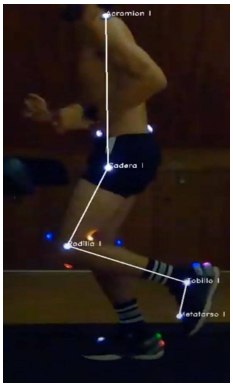
- Cálculo de posiciones anatómicamente correctas de los puntos de interés a medir.
- Cálculo de ángulos articulares de los puntos respectivos, filtrando el movimiento del músculo por sobre el hueso.
- Frecuencia y velocidad del ciclo de carrera.
- Identificación de puntos de contacto y despegue.



(a) Mockup del sistema *ARBA*.



(b) Posicionamiento de marcadores.



(c) Video procesado.

Figura 3.2: Funcionamiento del sistema *ARBA*.

[5] https://ac3e.usm.cl/moderno-sistema-de-evaluacion-de-trote-promete-prevenir-lesiones
[6] www.lanek.cl
[7] https://ac3e.usm.cl
[8] www.usm.cl
[9] www.uv.cl

3.2. MotionMetrix 3D Running Gait[10]

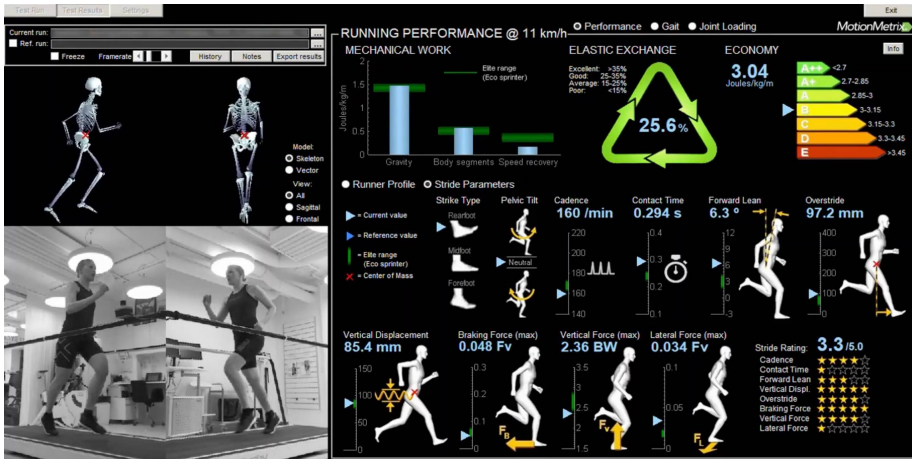
Empresa Sueca dedicada al análisis del trote. El objetivo de este sistema es ayudar a corregir la postura y posición de los miembros al correr, calculando las fuerzas ejercidas en las articulaciones, mediante cámaras capaces de detectar profundidad (*Kinect*, con la cual se han realizado diversas investigaciones relacionadas al campo del análisis de trote, tanto en su versión 1[16] como en su versión 2[17]) y *marcadores pasivos*. El uso de estos hace de la captura de movimiento una tarea simple y rápida.

El sistema cubre todos los aspectos esenciales de la mecánica del trote tales como:

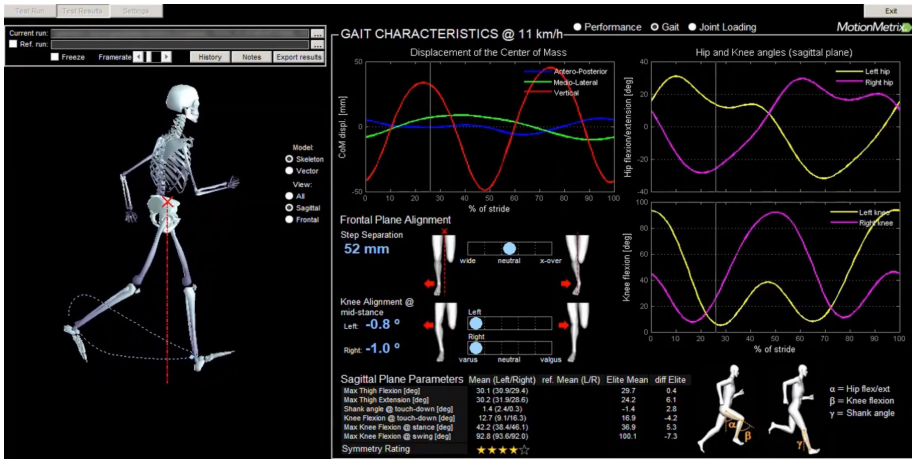
- Cálculo de trabajo mecánico e intercambio de energía elástica.
- Parametrización espacio-temporal del paso y reacción de fuerza-suelo.
- Cinemática de andada y carga articular para evaluación de riesgo de lesión.
- Data normalizada de corredores de más de 50.000 corredores de elite.
- Evaluación de perfil único de corredor que permite la predicción de velocidad, rendimiento, propensión a lesiones, etc.
- Comparación con *overlay* de datos de antes y después.
- Generación de gráficos en base a más de 60 parámetros de velocidad.
- Entrega de reporte exhaustivo y video procesado.

Además, durante los últimos años, cámaras de profundidad de este tipo *Red Green Blue - Depth (RGB-D)*, han sido validadas clínicamente en múltiples oportunidades por estudios externos[18].

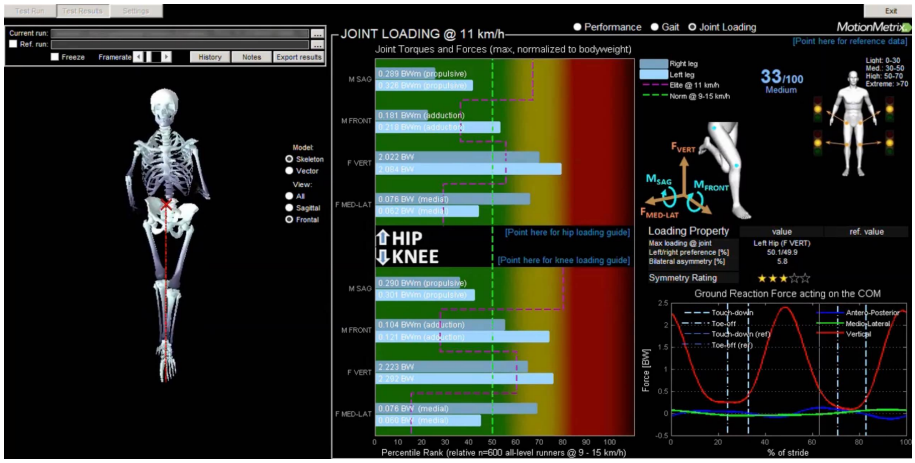
[10] www.motionmetrix.se/3-products/results-summary



(a) Rendimiento de carrera.



(b) Características de trote.

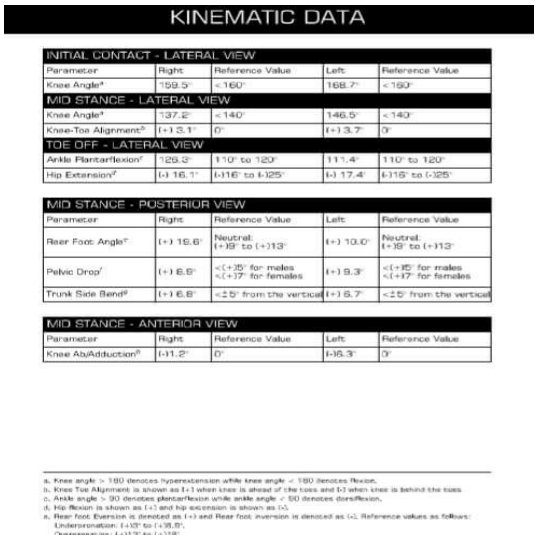


(c) Carga de articulaciones.

Figura 3.3: Funcionamiento del sistema MotionMetrix.

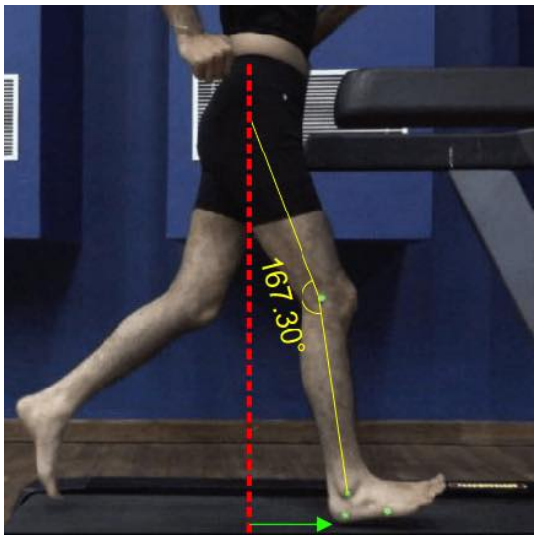
3.3. GaitON Running Gait Analysis[11]

Software desarrollado por la empresa *Auptimo*, basada en Delhi, India. Este sistema es capaz de realizar la captura, análisis y reportes del movimiento del usuario. Para esto se usa una trotadora y cuatro cámaras en diferentes posiciones (norte, sur, este y oeste del usuario). Este sistema también utiliza *marcadores activos*.

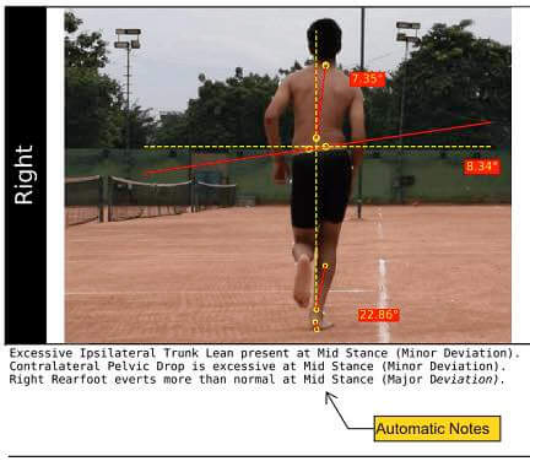


(a) Reporte kinesiológico.

(b) Comparación entre inclinaciones laterales maximales.



(c) Identificación de fallas biomecánicas.



(d) Documentación automática de fallas identificadas.

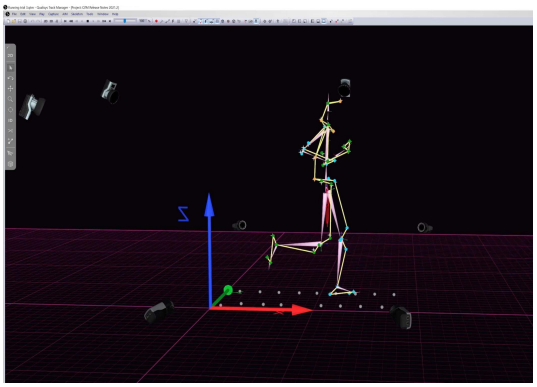
Figura 3.4: Funcionamiento del sistema GaitON.

[11] www.auptimo.com/running-analysis

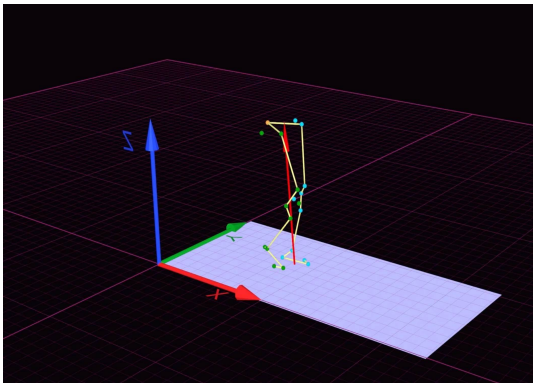
3.4. Qualisys Track Manager[12]

Sistema de captura de movimientos desarrollado por la empresa Sueca *Qualisys*. Parte de su implementación es *open-source*[13], hace uso de *marcadores pasivos* y múltiples cámaras (de 8 a 12) para la recolección precisa de datos de posición en 3D (típicamente a 300$[FPS]$) en conjunto con una trotadora especialmente diseñada, capaz de transmitir información sobre la fuerza de contacto del corredor con el suelo, esto sumado a una serie de hasta 32 sensores (*Electromiografía (EMG)*, *Inertial Measurement Unit (IMU)*, giroscopio, elementos que han sido clínicamente validados para el análisis de marcha[19]) capaces de realizar una sincronización en tiempo real al software de adquisición de datos, esto entregaría una precisión sub-milimétrica.

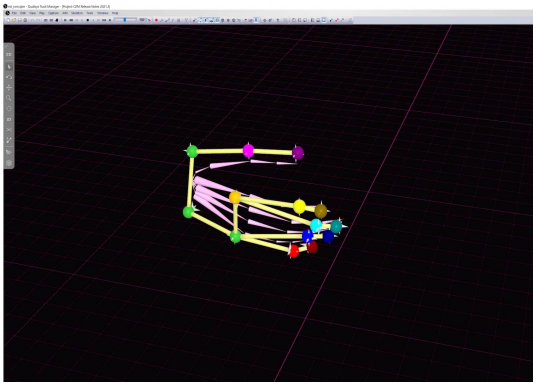
(a) Captura de datos.



(b) Reconstrucción esquelética.



(c) Cálculo de fuerza de contacto.



(d) Análisis de precisión.

Figura 3.5: Funcionamiento del sistema Qualisys.

[12] www.qualisys.com/software/qualisys-track-manager
[13] https://github.com/qualisys

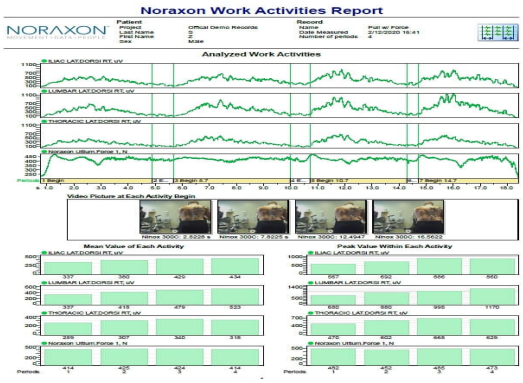
3.5. Noraxon-Ninox[14]

El sistema Ninox pertenece a la empresa Estadounidense Noraxon. Este sistema cuenta con un amplio abanico de funcionalidades, entre las que se pueden destacar: Análisis de patrones de activación promedio mediante *EMG*, análisis de equilibrio y balanceo, ajuste de bicicleta, plantillas ortopédicas a medida, análisis de amplitud *EMG*, biofeedback *EMG*, análisis de ergonomía y factores humanos, análisis de fatiga, análisis de marcha, pruebas isokinéticas, pruebas isométricas, análisis de salto, terapia de piso pélvico, análisis de rango de movimiento, retorno al juego, análisis de trote, pruebas de simetría y coordinación, análisis de lanzamiento, tiro y golpe.

El grueso de estas funcionalidades hace uso de cámaras de alta velocidad, sensores de *EMG*, fuerza, presión e *IMU* con una frecuencia de muestreo no menor a los 400$[Hz]$.

(a) Análisis de lanzamiento de bola.

(b) Captura de datos *raw*.

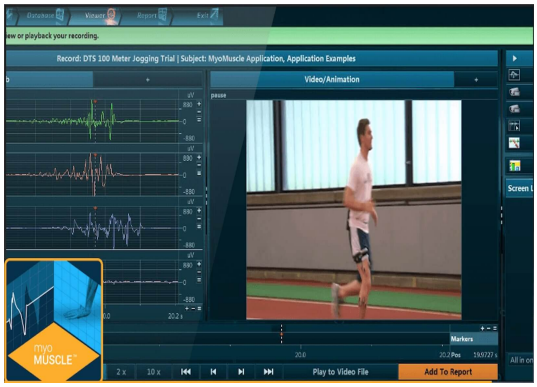


(c) Captura de datos de *EMG*.

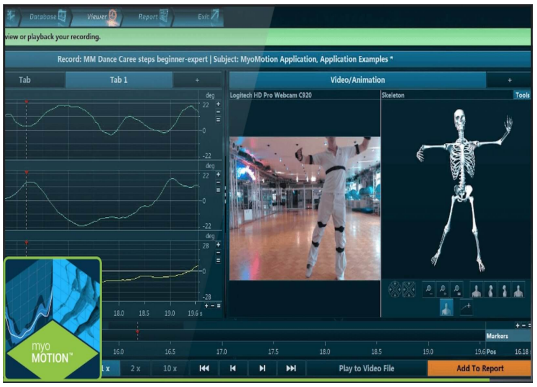
(d) *Biofeedback*.

Figura 3.6: Ejemplos del sistema Noraxon-Ninox.

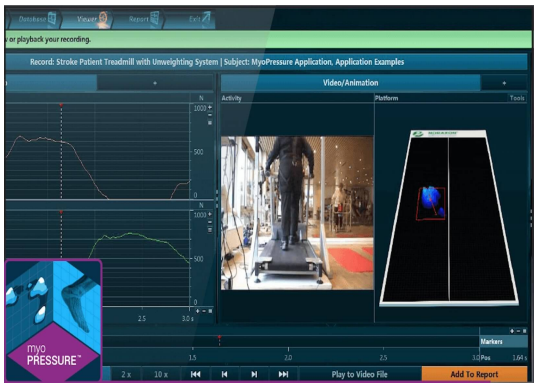
[14] https://www.noraxon.com/gait-analysis-running-analysis



(a) Electromiografía.

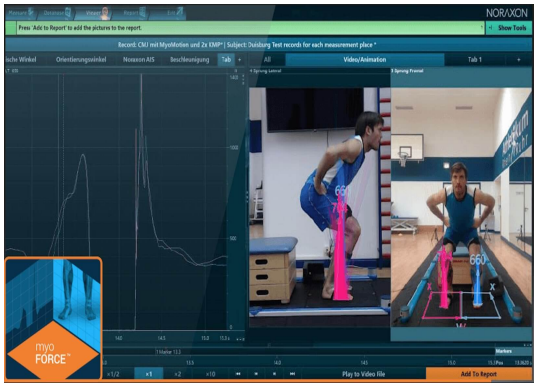


(b) Captura de movimiento 3D.

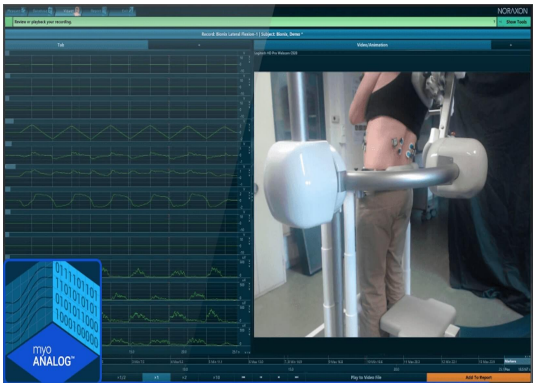


(c) Captura de datos de presión.

(d) Análisis de video.



(e) Análisis de fuerzas.



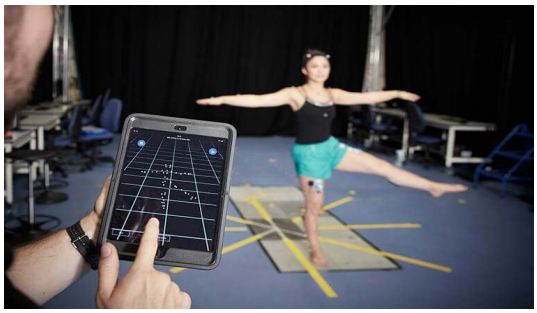
(f) Sensado externo.

Figura 3.7: Funcionalidades del sistema Noraxon-Ninox.

3.6. Vicon Motion Systems[15]

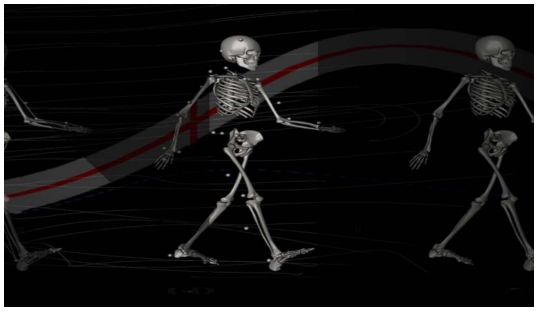
Vicon cuenta con cuatro soluciones de captura de movimiento[16], *Shōgun*[17], *Nexus*[18], *Tracker*[19] y *Capture.U*[20]. Este último software es utilizado tanto en el análisis del rendimiento deportivo[21] como en análisis de trote y control neuromotor[22].

Este sistema hace uso de cámaras y sensores inerciales propietarios[23] en combinación con marcadores pasivos. El software es capaz de hacer un seguimiento preciso al movimiento de humanos y objetos inertes. Además promete ser capaz de realizar este seguimiento con una precisión media de $0.017[mm]$, equivalente a $\frac{1}{5}$ del grosor de un cabello humano.

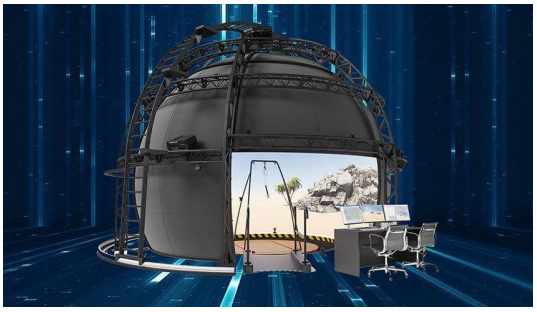


(a) Captura precisa de movimiento.

(b) Sensor inercial y marcadores pasivos.



(c) Reconstrucción esquelética.



(d) Condiciones ambientales controladas.

Figura 3.8: Ejemplos del sistema *Vicon Capture.U*.

[15] https://www.vicon.com
[16] https://www.vicon.com/software/
[17] https://www.vicon.com/software/shogun/
[18] https://www.vicon.com/software/nexus/
[19] https://www.vicon.com/software/tracker/
[20] https://www.vicon.com/software/captureu/
[21] https://www.vicon.com/applications/life-sciences/sports-performance
[22] https://www.vicon.com/applications/life-sciences/gait-analysis-neuroscience-and-motor-control
[23] https://www.vicon.com/hardware/

4. Metodología

A continuación se detalla la metodología utilizada durante este estudio, la que comprende los distintos protocolos utilizados en el diseño experimental y las métricas de comparación entre ambos sistemas.

Los participantes que formaron parte de este estudio consistieron en un grupo heterogéneo de personas de diferentes géneros, en condición física y estructura dentro de la media, de edades entre 18 y 30 años, con un n-muestral de 20, para un total de 36 videos de entrenamiento.

Todos los datos de entrenamiento y/o evaluación de este estudio fueron capturados en el centro clínico ***You Just Better***[24] en la comuna de Las Condes, Santiago, Chile, entre agosto y diciembre de 2021.

Para este caso particular, la adaptación de la red *DLC* corresponde a un entrenamiento, procesamiento y validación profesional.

4.1. Objetivo 1 - Protocolo de evaluación y captura de datos

El protocolo de evaluación y captura de datos obedece al siguiente diagrama de flujo:

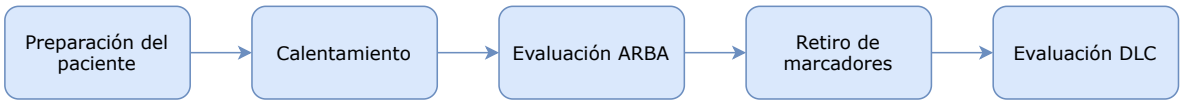


Figura 4.9: Protocolo de evaluación y captura de datos.

Preparación del paciente: El especialista se encarga de posicionar los marcadores en los puntos anatómicos correctos y despejar cualquier parte de la ropa del paciente que pudiera intervenir u ocluir en algún momento la visibilidad del marcador.

Calentamiento: El especialista instruye al paciente realizar un trote suave hasta alcanzar los 10$[km/h]$.

Evaluación ARBA: Se realiza la captura de video (10[s]) con el sistema *ARBA*. Las características del video capturado son las siguientes:

- Cámara: GoPro HERO7 Black

[24] http://justbetter.cl

- Contenedor: MP4
- Codec: H.264
- Resolución: 1920x1080p
- Framerate: 120[FPS]

Retiro de marcadores: Se retiran los marcadores anteriormente posicionados y se instruye al paciente continuar con el trote a 10$[km/h]$.

Evaluación DLC: Se realiza la captura de video (10[s]) sin marcadores a fin de ser utilizados como datos de entrenamiento y/o evaluación. Es importante mencionar que la evaluación *DLC* y la evaluación *ARBA* se realizan en igualdad de condiciones, sin modificar la luminosidad del ambiente ni los parámetros de captura de video de las cámaras.

4.2. Objetivo 1 - Protocolo de procesamiento de datos

El protocolo de procesamiento de datos obedece al siguiente diagrama de flujo:

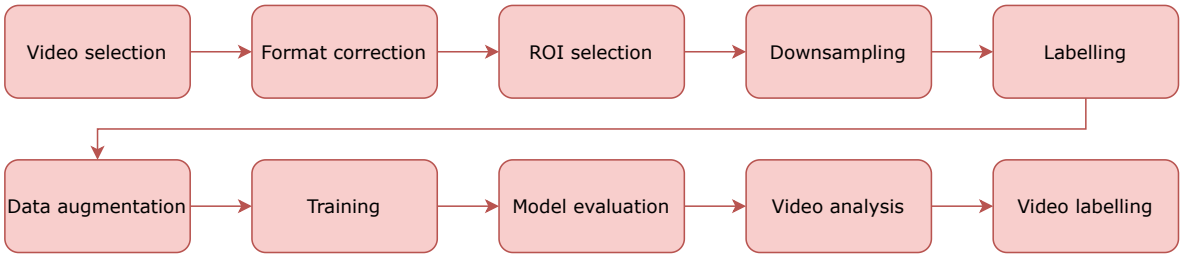


Figura 4.10: Protocolo de procesamiento de datos.

Video selection: Este paso consiste en la selección y separación de los videos obtenidos, decidir cuales pasarán a formar parte del detaset de entrenamiento y cuales formarán el dataset de evaluación.

Format correction: Se realiza una corrección de formato a cada video, tanto del dataset de entrenamiento como del de evaluación. En específico se utiliza el software ***HandBrake***[25] para convertir los videos a un formato con las siguientes características[20]:

- Contenedor: MP4
- Codec: H.264

[25] https://handbrake.fr

- Resolución: 1920x1080p
- Framerate: 120[FPS]

ROI selection: Se selecciona la *Region Of Interest (ROI)* de cada video con el fin de alivianar la carga de entrenamiento al sistema, además este paso produce mejores resultados al tener datos más acotados.

Downsampling: Se realiza un submuestreo de los videos del dataset de entrenamiento, en dónde se extraen 20 cuadros de cada uno.

Labelling: Se etiquetan manualmente los cuadros obtenidos en el paso anterior.

Data Augmentation: Se realiza un proceso artificial de aumentación de datos. En este caso se utiliza el método ***imgaug***[26] (entre las opciones: imgaug, tensorpack) sobre una *ResNet-50* (entre las opciones: dlcrnet_ms5, resnet50, resnet101, resnet152, mobilenet_v2_1.0, mobilenet_v2_0.75, mobilenet_v2_0.5, mobilenet_v2_0.35, efficientnet-b0, efficientnet-b3, efficientnet-b6).

Training: Se realiza el entrenamiento con la *Residual Neural Network (ResNet)* seleccionada, en modo *multi-animal* y los datos de entrenamientos aumentados generados en el paso anterior. Este entrenamiento se lleva a cabo por exactamente 1.030.000 épocas en un computador con las siguientes características:

- CPU: Intel Core i7-11700F @2.5GHz, caché 16MB, socket LGA 1200.
- GPU: EVGA NVIDIA RTX3060 12GB XC.
- RAM: G.Skill DDR4 32GB (4x8) @3200MHz Flare X.

Model evaluation: Se realiza una evaluación del modelo generado con los datos de entrenamiento.

Video analysis: Se realiza el análisis de los videos del dataset de evaluación. Con esto se extraen los puntos de interés encontrados por la red.

Video labelling: Se grafican y etiquetan los puntos encontrados en el paso anterior sobre los videos del dataset de evaluación.

[26] https://github.com/aleju/imgaug

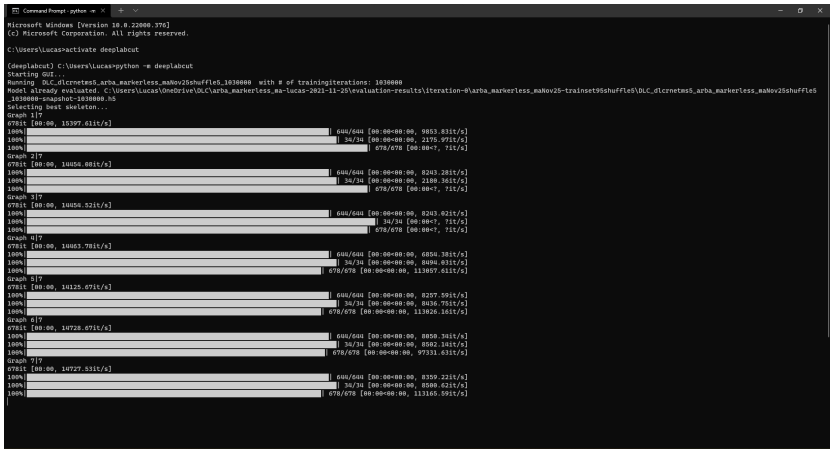


Figura 4.11: Ejecución del software.

Figura 4.12: Pestaña de bienvenida.

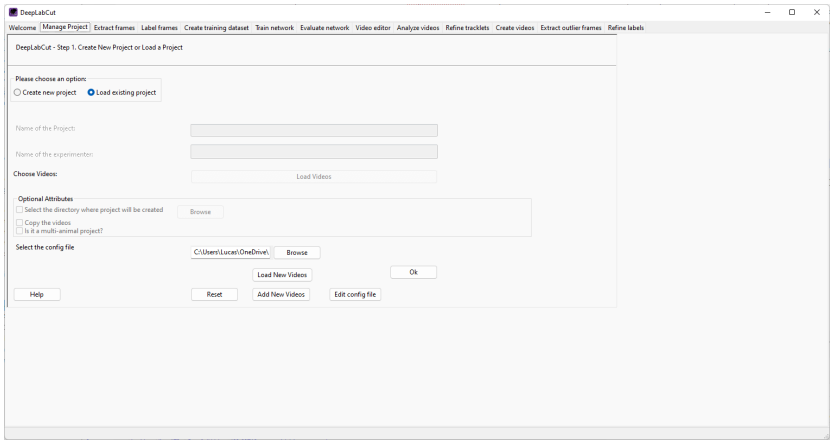


Figura 4.13: Gestión de proyectos.

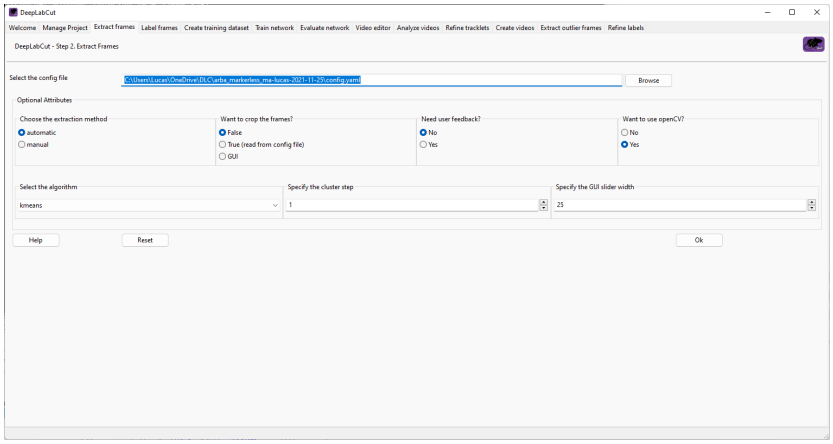


Figura 4.14: Extracción de frames.

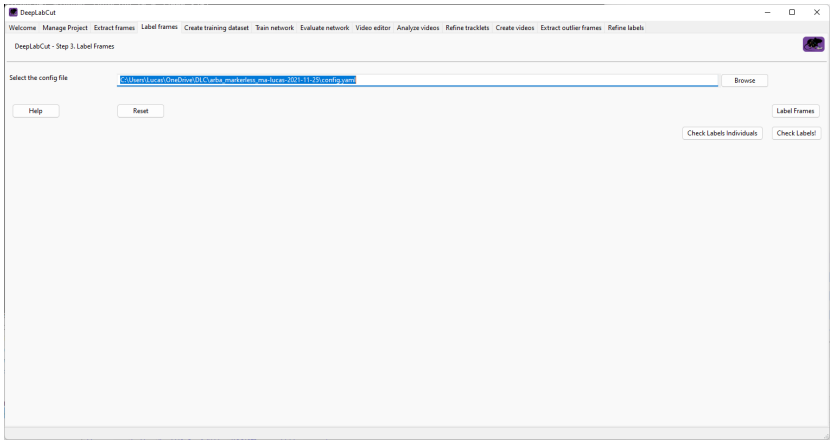


Figura 4.15: Etiquetado de frames.

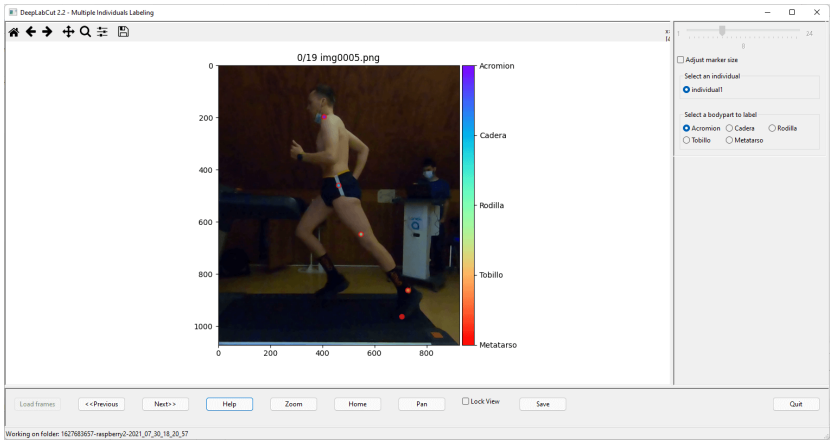


Figura 4.16: Ejemplo de etiquetado.

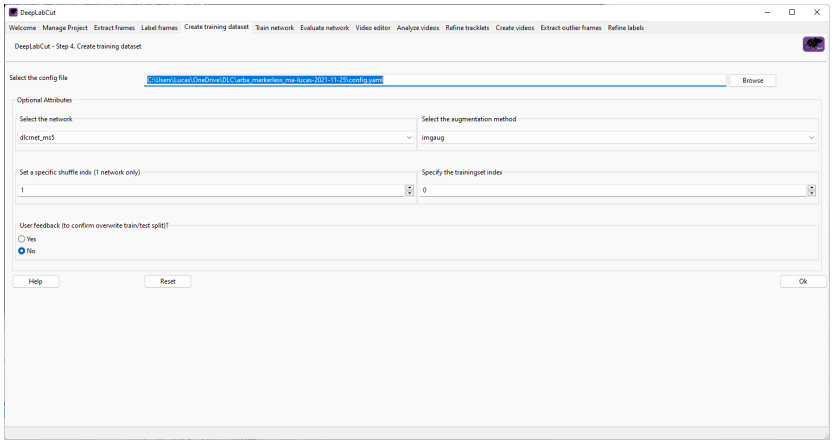


Figura 4.17: Creación de datos de entrenamiento.

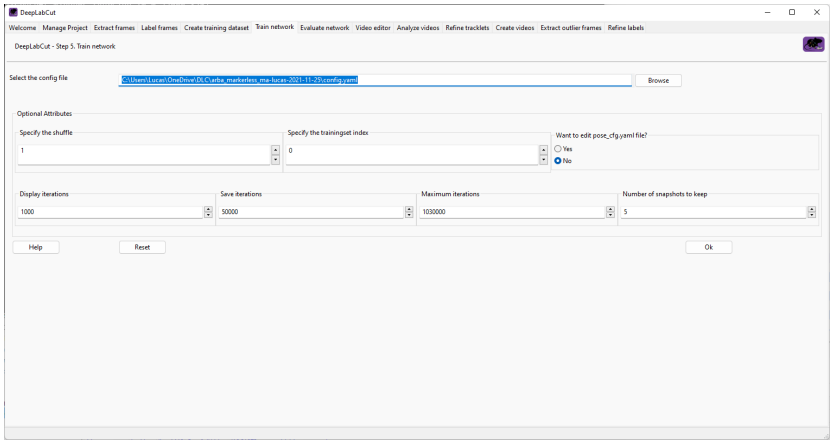


Figura 4.18: Entrenamiento de red.

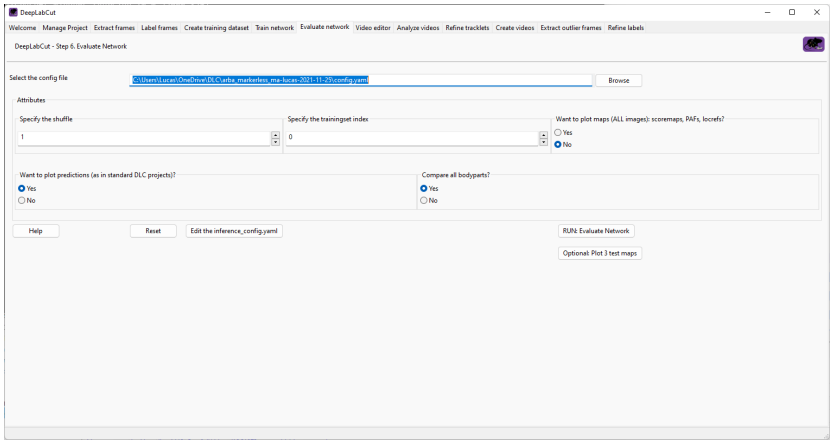


Figura 4.19: Evaluación de red.

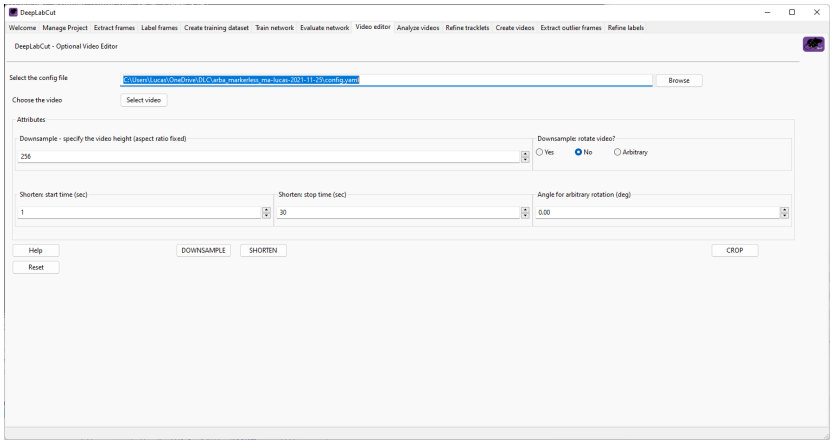


Figura 4.20: Editor de videos.

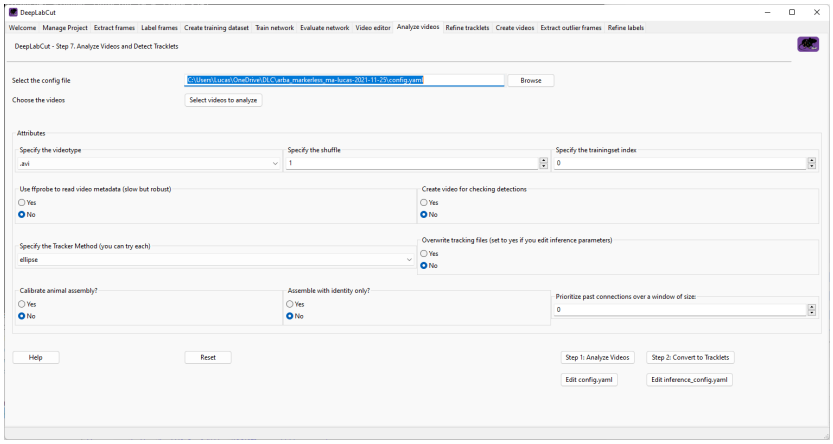


Figura 4.21: Analizar videos.

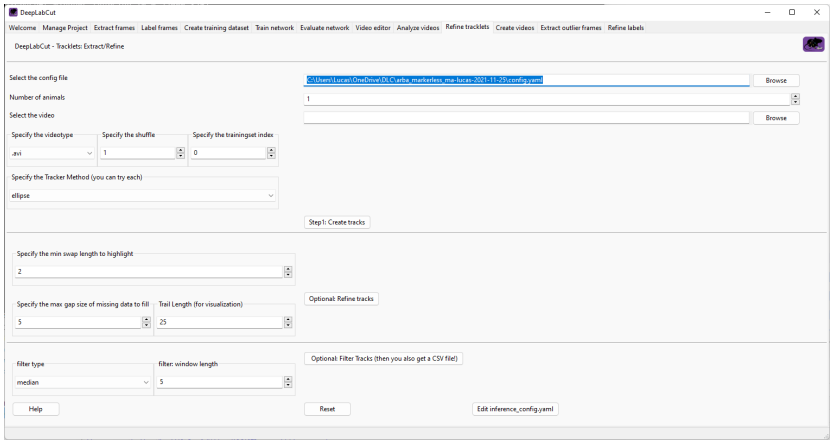


Figura 4.22: Refinar tracklets.

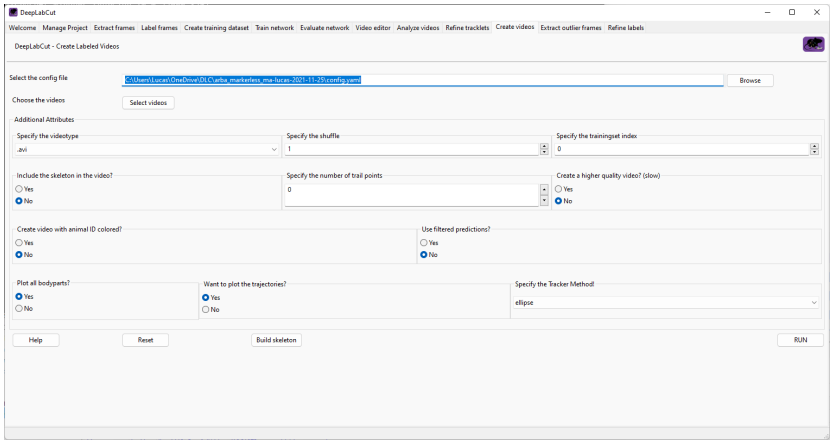


Figura 4.23: Crear videos de salida.

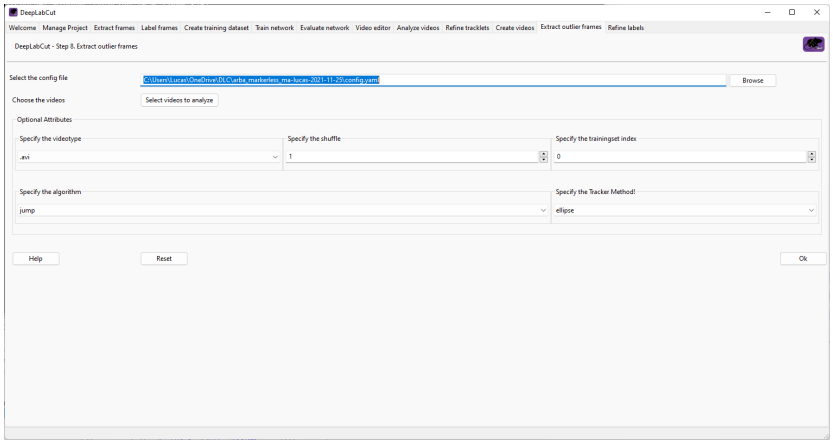


Figura 4.24: Extraer frames aislados.

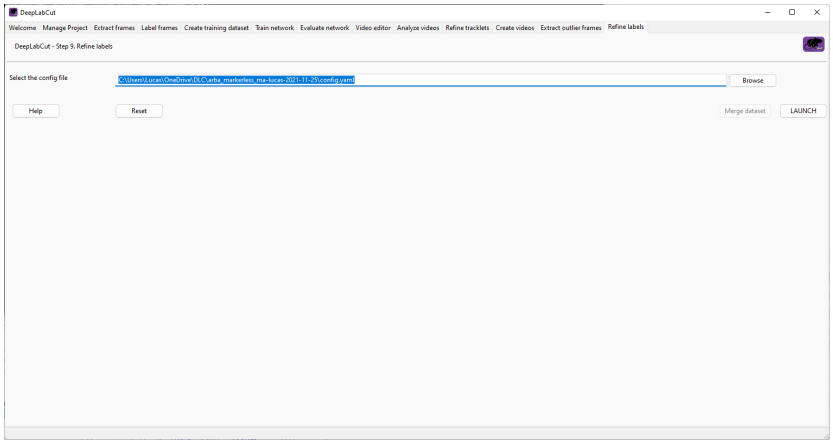


Figura 4.25: Refinar etiquetas.

4.3. Objetivo 2 - Métricas y estadísticas de comparación

El protocolo de comparación de resultados obedece al siguiente diagrama de flujo:

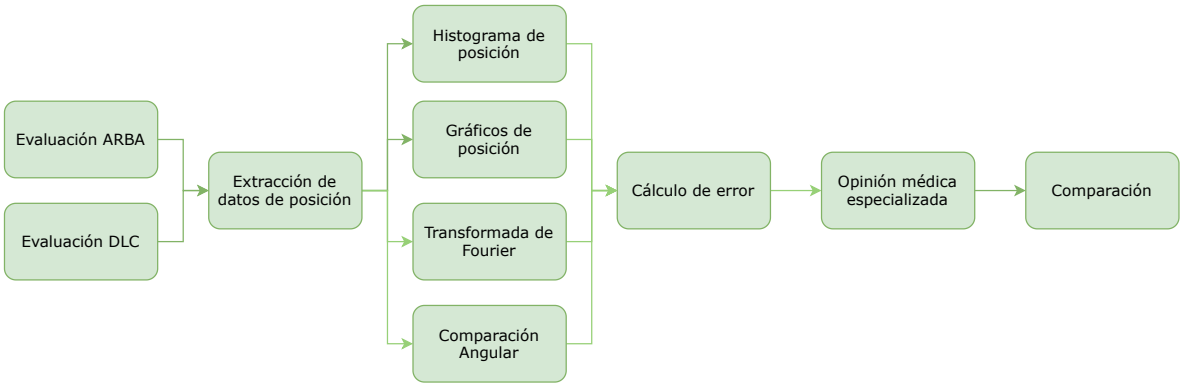


Figura 4.26: Protocolo de comparación de resultados y métricas.

Evaluación ARBA: Se realiza la captura de video con el sistema *ARBA*.

Evaluación DLC: Se realiza la captura de video con el sistema *DLC*.

Extracción de datos de posición: Se extraen los datos de posición de los videos analizados con ambos sistemas.

Gráficos de posición: Se producen gráficos de posición de los datos extraídos a modo de realizar una comparación visual simple entre los sistemas.

Histograma de posición: Se producen histogramas de posición de los datos extraídos a modo de realizar una comparación estadística.

Transformada de Fourier: Se producen gráficos de las transformadas de Fourier de los datos extraídos a modo de realizar una comparación entre sus frecuencias fundamentales, armónicos y su amplitud.

Comparación angular: Se comparan los ángulos articulares entre los datos extraídos.

Cálculo de error: Se calcula el error medio entre los ángulos calculados con las posiciones extraídas de ambos sistemas.

Opinión médica especializada: Se consulta la opinión de un especialista en el área kinesiológica a fin de validar los resultados obtenidos.

Comparación: Se realiza una comparación que toma en cuenta las métricas (histograma de posición, factor de forma gráfico, error medio) y la opinión del especialista.

5. Resultados

En cuanto al primer objetivo, el entrenamiento se realiza de forma exitosa. El software entrega los resultados de evaluación de red de 14.66 pixeles en el dataset de entrenamiento y 3.56 en el dataset de prueba (distribuidos en 95 % train y 5 % test). La tabla 5.2 contiene los datos de errores calculados por la red en cada marcador.

Tabla 5.2: Errores de evaluación de la red.

Marcador	**Rodilla**	**Tobillo**	**Metatarso**
RMSE	5.90121611	3.033265127	4.189546504

En cuanto al segundo objetivo, en primera instancia se importan los datos en *MATLAB* mediante el código 7.1.

Para la realización de un análisis cualitativo válido se realizan ciertas correcciones a los datos capturados. La primera de estas correcciones corresponde a una interpolación de los datos faltantes de *DLC* mediante al función *fillmissing()* de *MATLAB* mediante el método cúbico *spline*, esto permite otorgar completitud a los datos a utilizar, y es el símil de la interpolación realizada sobre los datos de *ARBA*. Este paso se corresponde con el código 7.10 en el caso de ARBA y el código 7.1 para el caso de DLC.

El análisis es inviable si los datos obtenidos no se encuentran en fase, por lo que se realiza una alineación y recorte de los datos extremos mediante las funciones *finddelay()* y *alignsignals()* de *MATLAB*. Este paso se puede ver en el código 7.2.

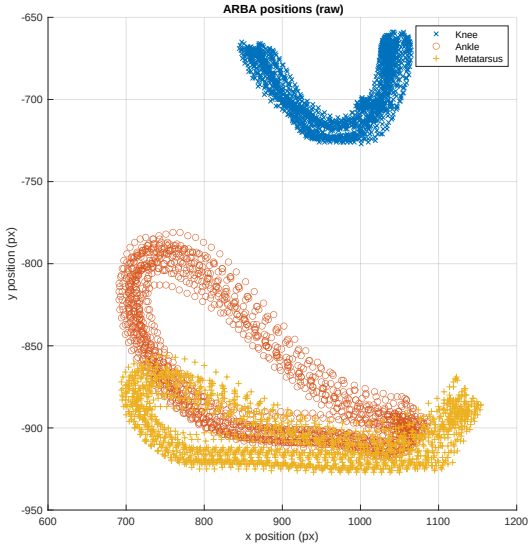
Habiéndose alineado y recortado las señales se procede al filtrado de las mismas. En este caso se utiliza un símil del método de filtrado de datos de *ARBA* (código 7.10), es decir, un filtro pasabajos de tipo *Butterworth*, de orden 8, con frecuencia de corte de 6$[Hz]$. El código correspondiente a este paso es el 7.3.

Con esto es posible graficar las posiciones de los puntos de interés de 3 formas diferentes, datos crudos, datos alineados y datos filtrados. Véase código 7.4.

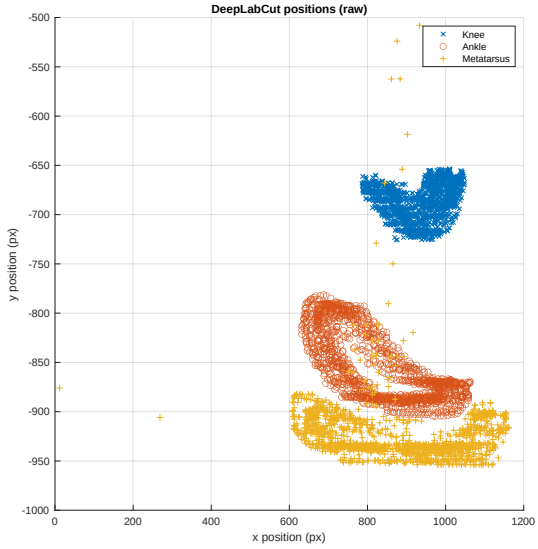
El primer caso a analizar corresponde al sujeto número 5, en plano sagital derecho.

En la imagen 5.27 se pueden apreciar las gráficas de comparación de posición de los marcadores

encontrados por el sistema *ARBA* con los encontrados por *DLC*. Estos corresponden a los puntos originales, es decir, los datos no se encuentran modificados de ninguna manera, solo se les ha aplicado una interpolación a modo de conseguir completitud. Además, se puede apreciar un ruido de seguimiento del marcador "Metatarso" por parte de *DLC*. Se eliminarán los datos incorrectos mediante el proceso de alineación, recorte y filtrado.



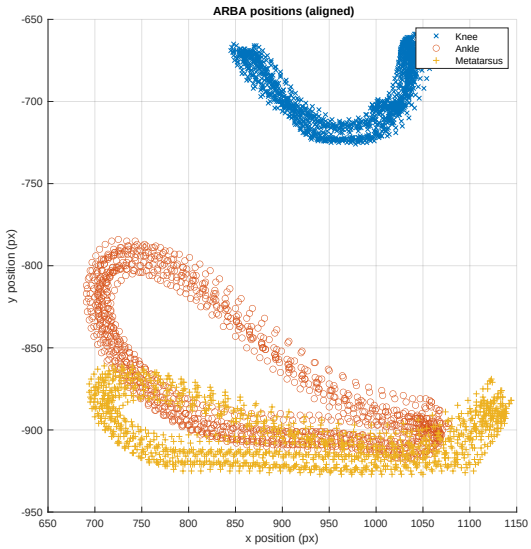
(a) *ARBA*.



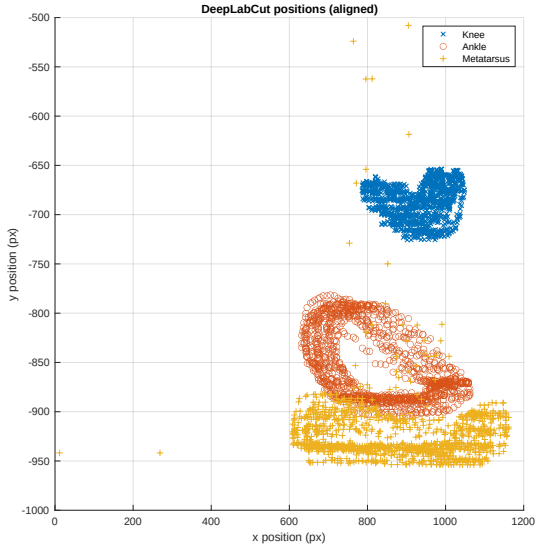
(b) *DLC*.

Figura 5.27: Posiciones x-y originales.

A continuación se procede a graficar las mismas posiciones, pero alineadas por fase. El resultado se puede apreciar en la figura 5.28, nótese que el ruido de seguimiento persiste.



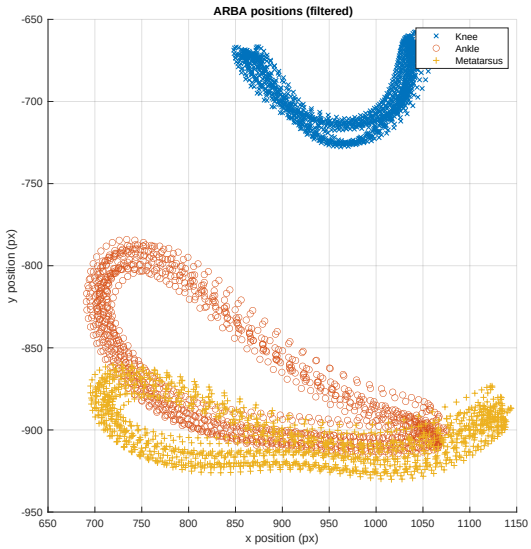
(a) *ARBA*.



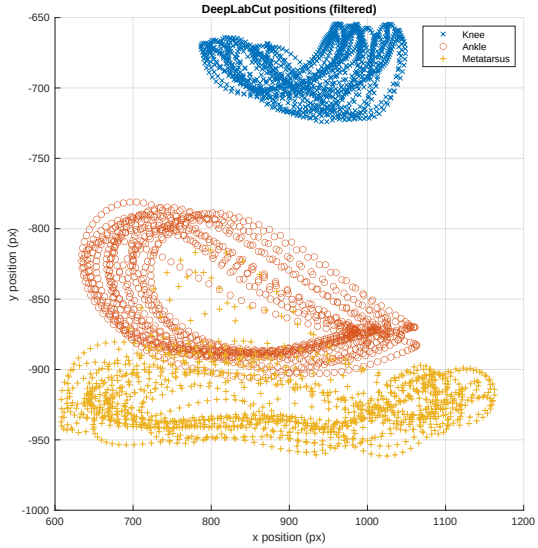
(b) *DLC*.

Figura 5.28: Posiciones x-y alineadas.

Luego de esto, se pueden graficar los mismos datos, pero filtrados en frecuencia. En la figura 5.29 se puede apreciar como los puntos de seguimiento incorrecto desaparecen por completo.



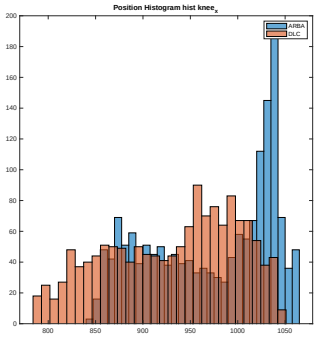
(a) *ARBA*.



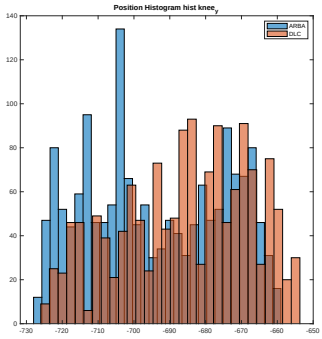
(b) *DLC*.

Figura 5.29: Posiciones x-y filtradas.

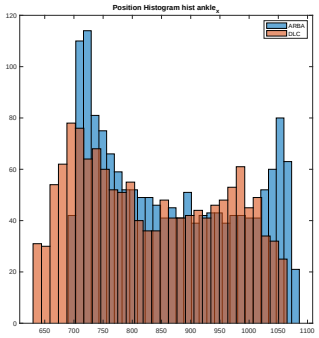
El siguiente método de análisis (código 7.5) corresponde a la comparación de los histogramas de posición de cada marcador. En la figura 5.30 se puede apreciar que no existe una fuerte correlación entre los datos sin procesar, esto se debe principalmente a dos factores, falta de alineación y ruido DC.



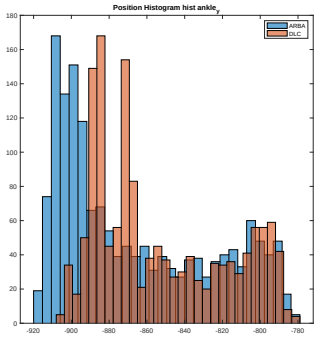
(a) Rodilla x.



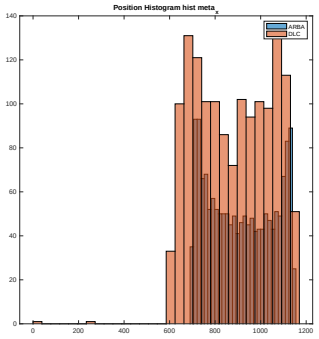
(b) Rodilla y.



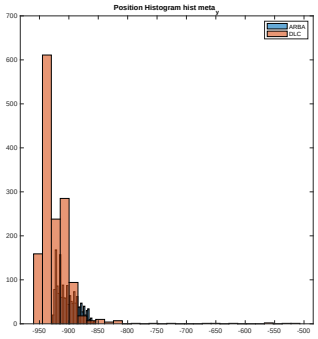
(c) Talón x.



(d) Talón y.



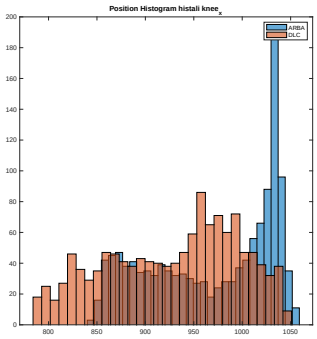
(e) Metatarso x.



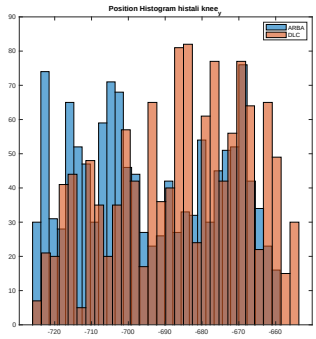
(f) Metatarso y.

Figura 5.30: Histogramas de posición sujeto 5R.

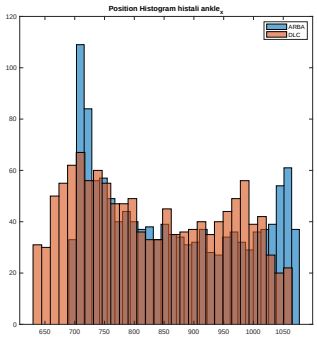
Al comparar la figura 5.31 con la figura 5.30 se puede apreciar que no existe una mejoría significativa entre los datos originales y los datos alineados (para propósito del histograma), esto es debido a que los valores del histograma no discriminan diferencias de fase/alineación.



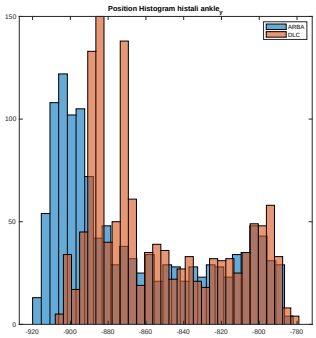
(a) Rodilla x.



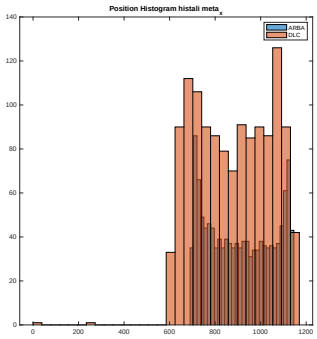
(b) Rodilla y.



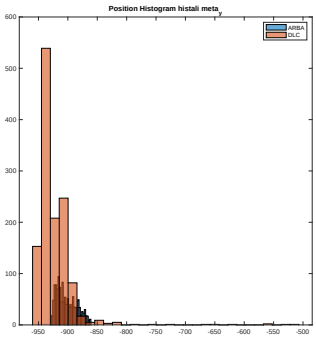
(c) Talón x.



(d) Talón y.



(e) Metatarso x.

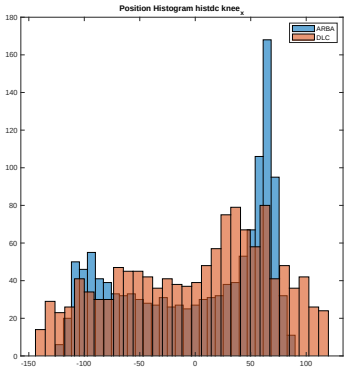


(f) Metatarso y.

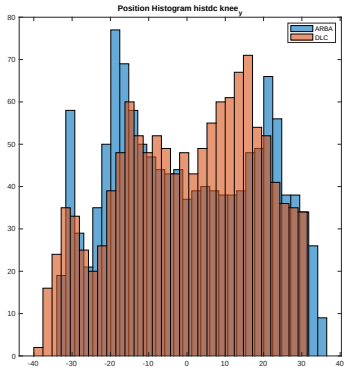
Figura 5.31: Histogramas de posición alineados sujeto 5R.

Sin embargo, luego de aplicar un filtro DC a los datos de posición de cada marcador se puede

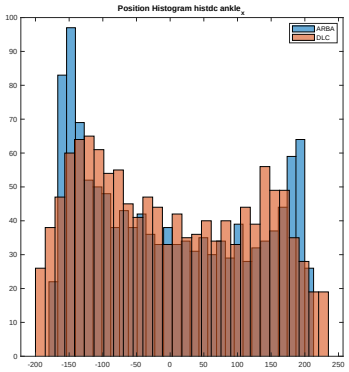
apreciar una correlación mucho más marcada entre los datos. Véase la figura 5.32, dónde se puede apreciar que el seguimiento hecho por ambos sistemas es estadísticamente comparable.



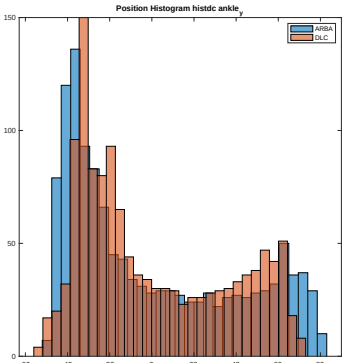
(a) Rodilla x.



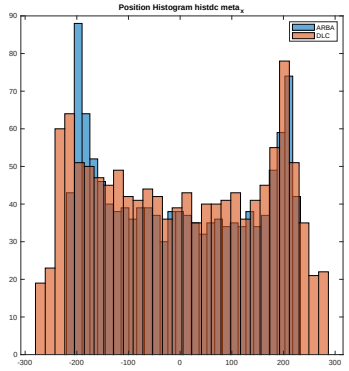
(b) Rodilla y.



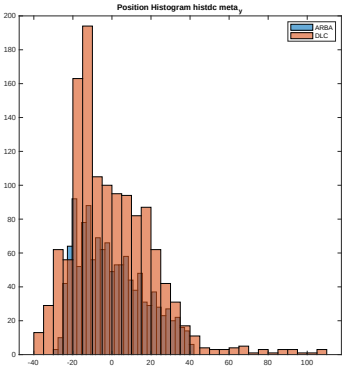
(c) Talón x.



(d) Talón y.



(e) Metatarso x.



(f) Metatarso y.

Figura 5.32: Histogramas de posición filtrados sujeto 5R.

Posterior a esto se aplica una FFT (véase código 7.6) a los vectores tratados (interpolados y filtrados), y se analiza la distribución de frecuencias de las mismas.

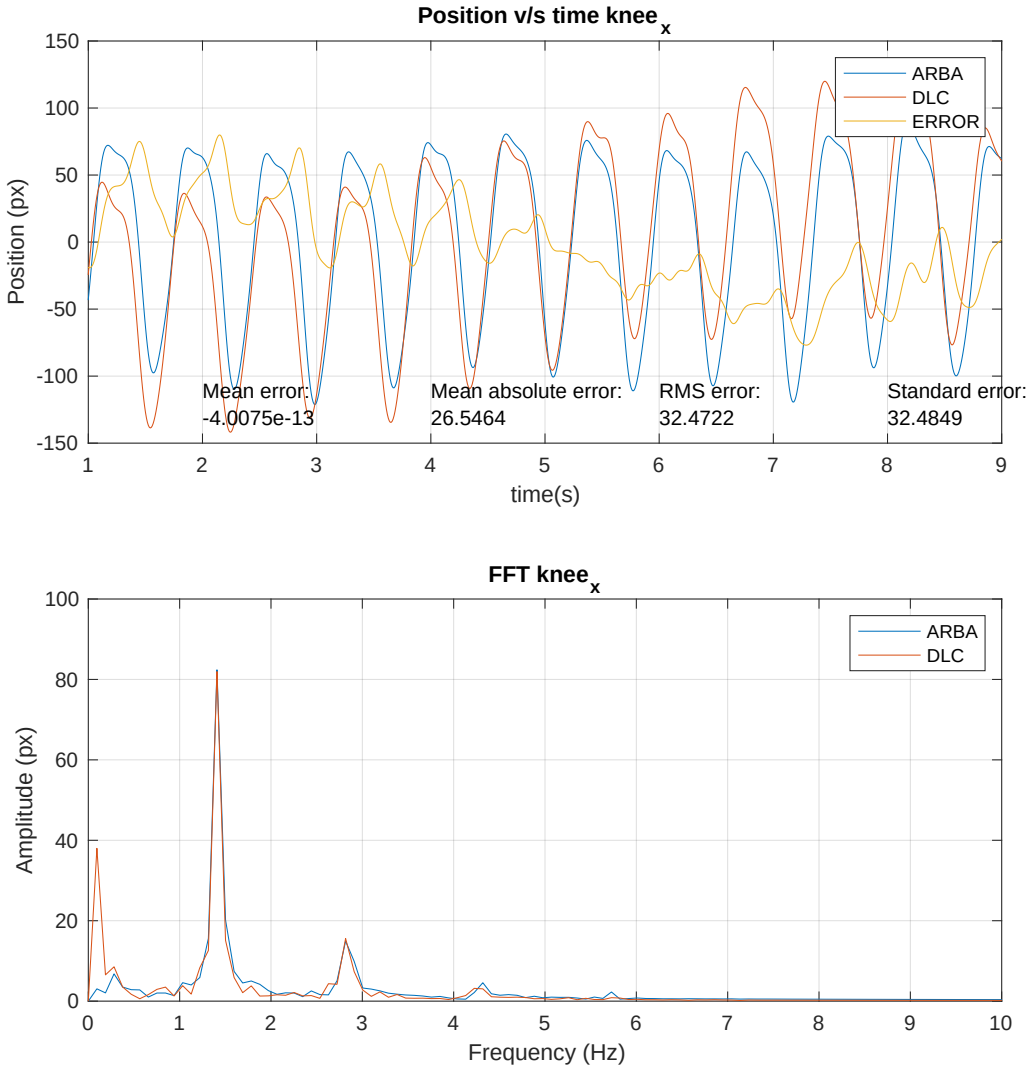


Figura 5.33: FFT rodilla x, sujeto 5R.

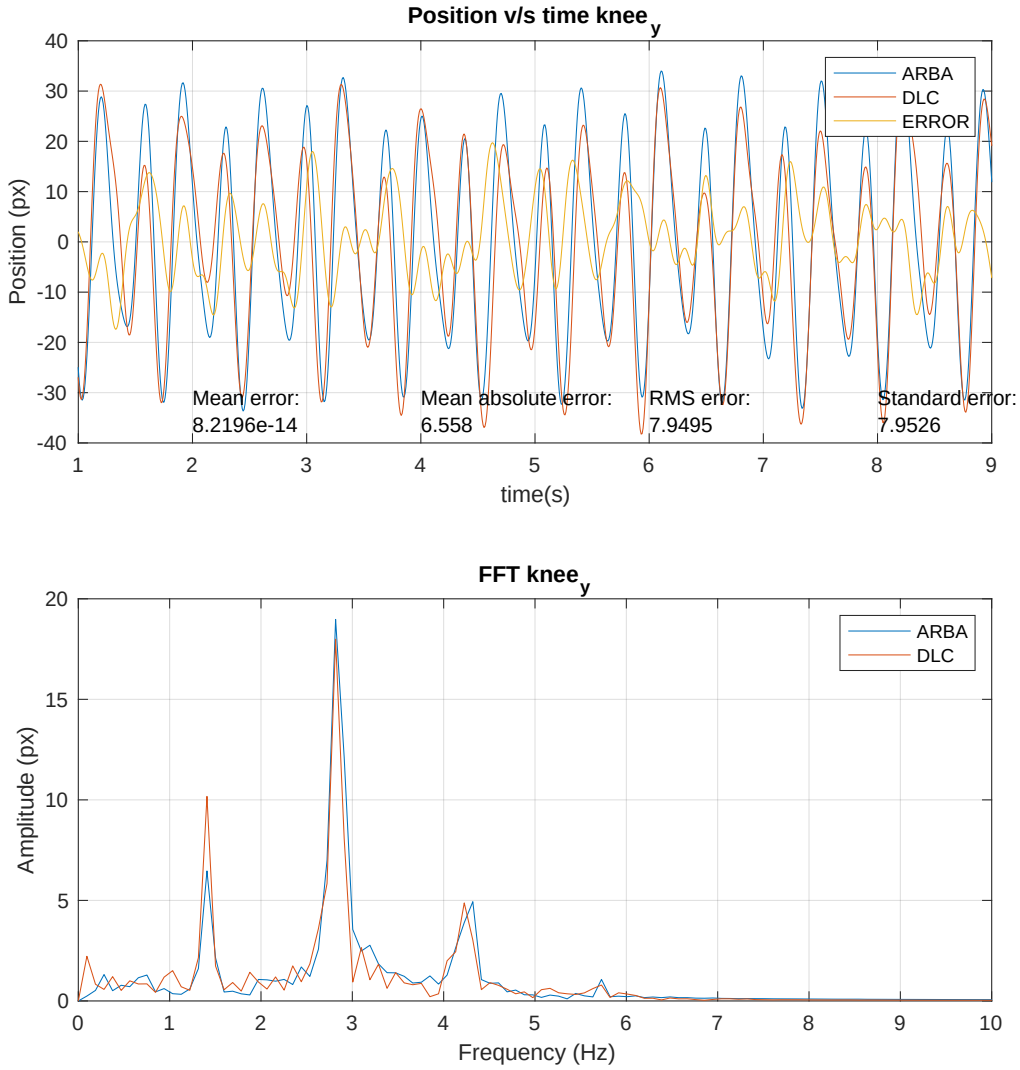


Figura 5.34: FFT rodilla x, sujeto 5R.

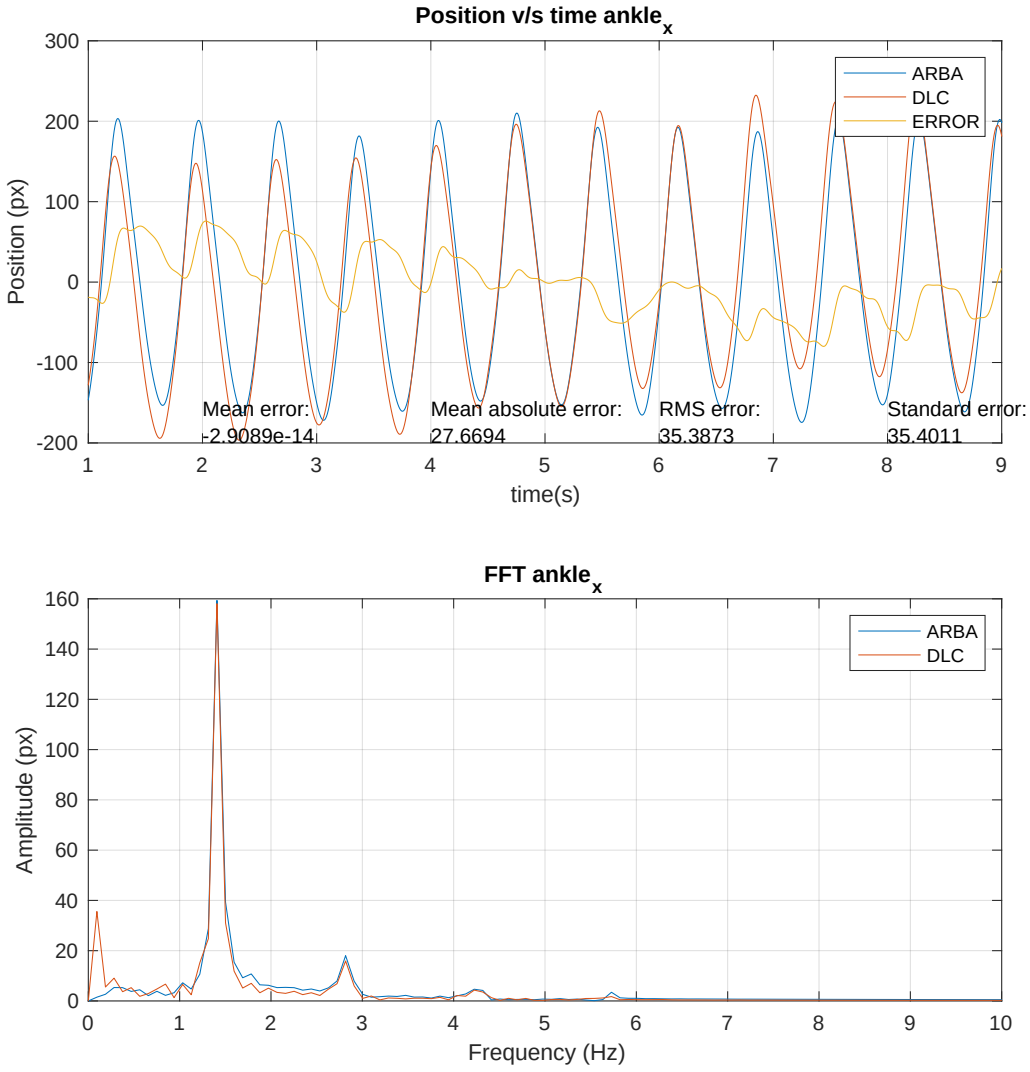


Figura 5.35: FFT tobillo x, sujeto 5R.

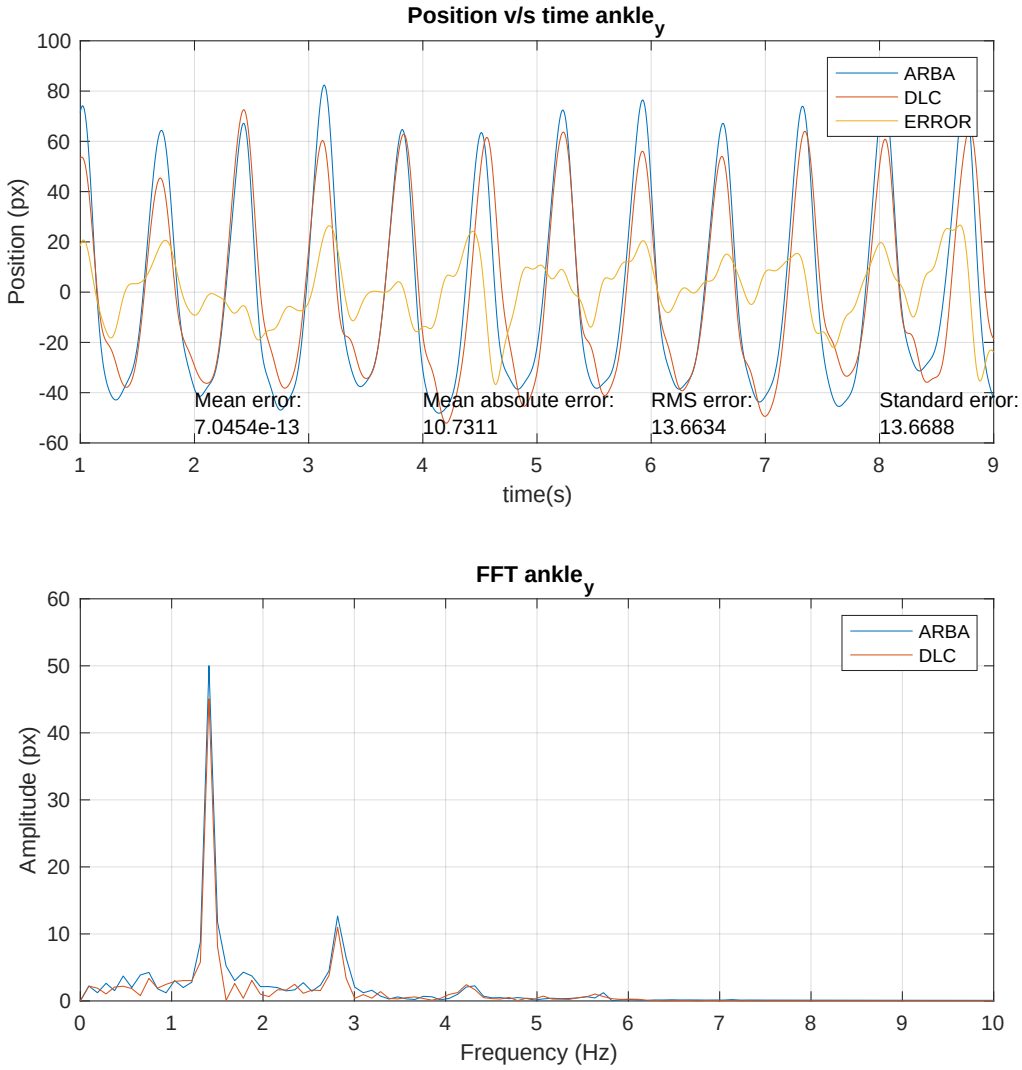


Figura 5.36: FFT tobillo y, sujeto 5R.

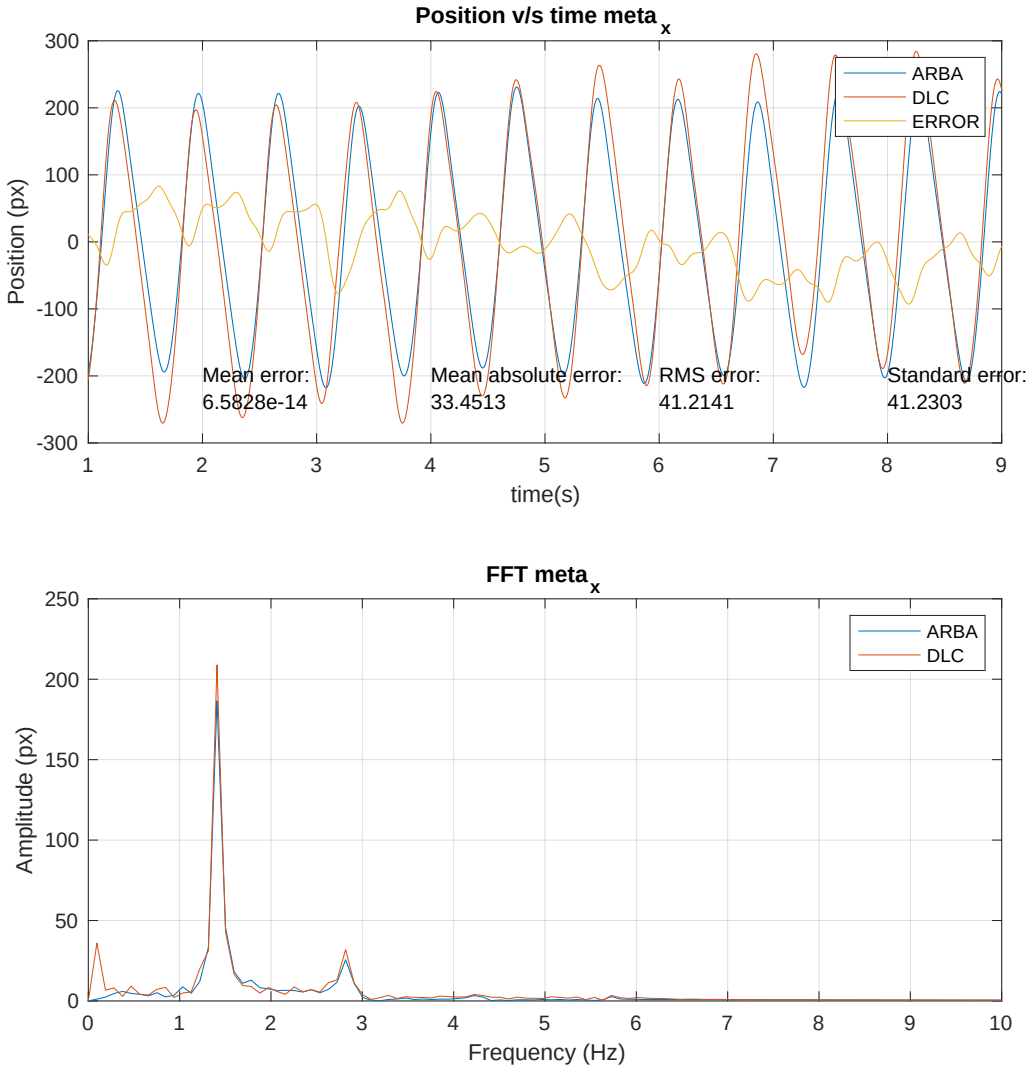


Figura 5.37: FFT metatarso x, sujeto 5R.

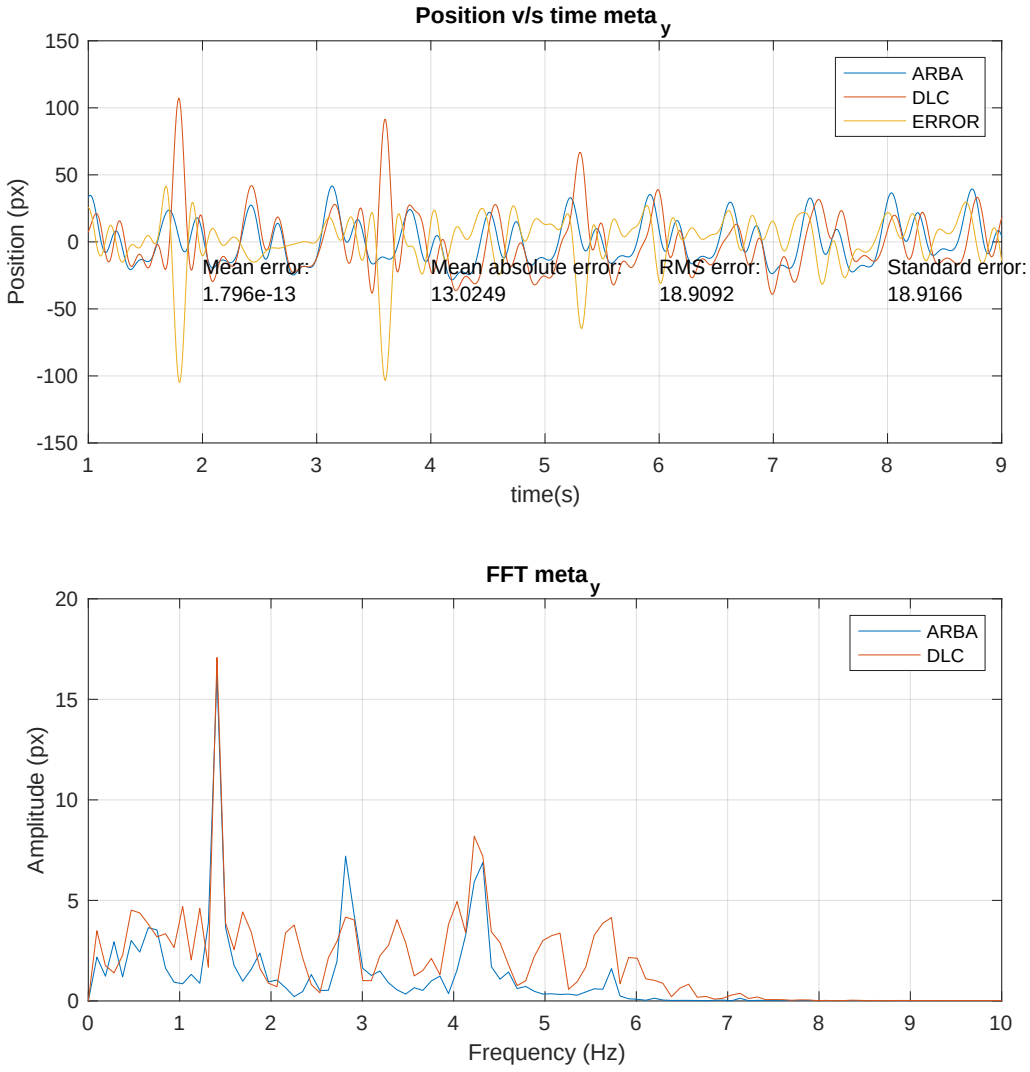


Figura 5.38: FFT metatarso y, sujeto 5R.

Al analizar las figuras 5.33, 5.34, 5.35, 5.36, 5.37 y 5.38 se puede apreciar que luego de haber tratado los datos estos se vuelven virtualmente idénticos, las amplitudes y fases de sus movimientos son lo suficientemente similares, solo presentándose errores relativamente bajos, a continuación se detallan en el cuadro 5.3 los errores encontrados entre las señales.

Tabla 5.3: Errores FFT, sujeto 5R.

Dato	Eje	Tipo	Valor
Rodilla	x	Mean	4.0075e-13
Rodilla	x	Mean abs	26.5464
Rodilla	x	RMSE	32.4722
Rodilla	x	Standard	32.4849
Rodilla	y	Mean	8.2196e-14
Rodilla	y	Mean abs	6.558
Rodilla	y	RMSE	7.9495
Rodilla	y	Standard	7.9526
Tobillo	x	Mean	-2.9089e-14
Tobillo	x	Mean abs	27.6694
Tobillo	x	RMSE	35.3873
Tobillo	x	Standard	35.4011
Tobillo	y	Mean	7.0454e-13
Tobillo	y	Mean abs	10.7311
Tobillo	y	RMSE	13.6634
Tobillo	y	Standard	13.6688
Metatarso	x	Mean	6.5828e-14
Metatarso	x	Mean abs	33.4513
Metatarso	x	RMSE	41.2141
Metatarso	x	Standard	41.2303
Metatarso	y	Mean	1.796e-13
Metatarso	y	Mean abs	13.0249
Metatarso	y	RMSE	18.9092
Metatarso	y	Standard	18.9166

En donde todos los errores (en pixeles) son bajos, en especial el error medio, además, el servicio de reportería (código 7.12) informa que el ciclo de carrera tiene un largo de 80 frames, sabiendo que los videos fueron grabados a 120$[fps]$ se calcula que la frecuencia del trote es de 1.5$[Hz]$, esto se comprueba en todos los espectros en Fourier graficados, los cuales presentan una frecuencia fundamental de 1.5$[Hz]$ y armónicos en 3$[Hz]$, 4.5$[Hz]$ y en algunos casos 6$[Hz]$.

El siguiente paso de comparación es el cálculo del ángulo articular formado por los vértices (rodilla, tobillo, metatarso). Se grafica en la figura 5.39 con el código 7.7 y se calculan los errores correspondientes en la tabla 5.4.

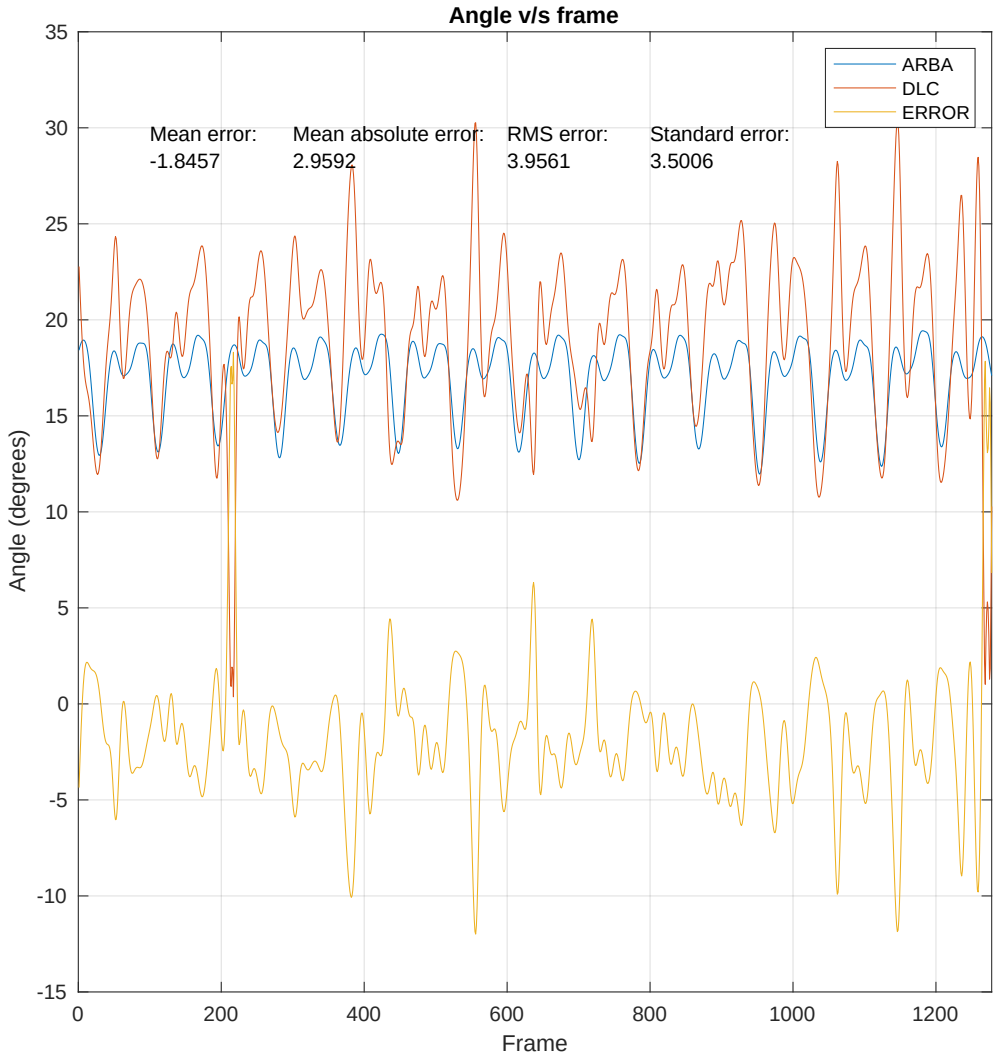


Figura 5.39: Comparación de ángulos y error medio del sujeto 5R.

Tabla 5.4: Errores ángulo articular, sujeto 5R.

Tipo	**Mean**	**Mean abs**	**RMSE**	**Standard**
Valor	-1.8457	2.9592	3.9561	3.5006

Se puede apreciar que el error (en grados) es de baja magnitud, por lo que se considera que el sistema hace un buen seguimiento de los puntos anatómicos requeridos.

Como última característica de comparación se utiliza la función *Distance Between Signals Using Dynamic Time Warping (dtw())*[21] de *MATLAB*, para obtener la distancia absoluta entre las señales. Para esto se utiliza el código 7.8.

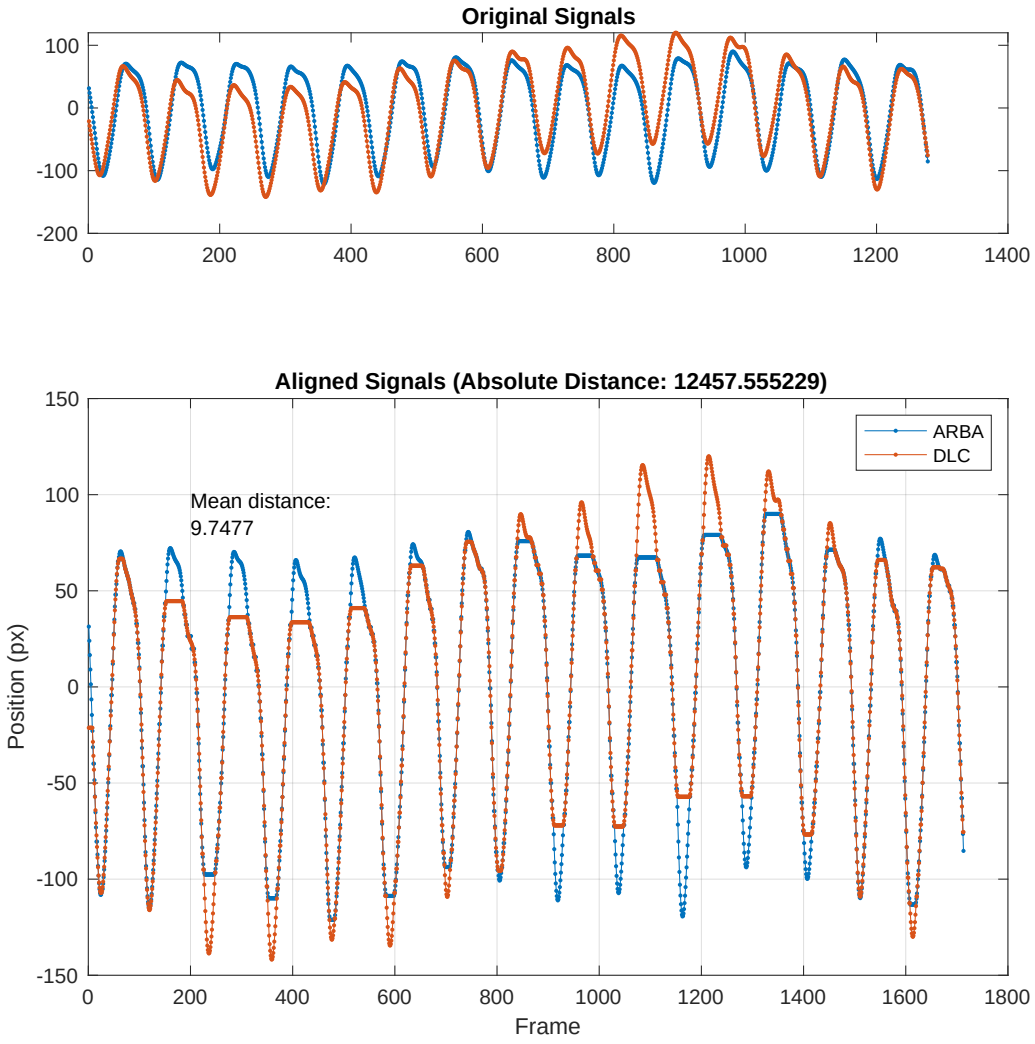


Figura 5.40: Medida de distancia absoluta entre las señales, rodilla eje x.

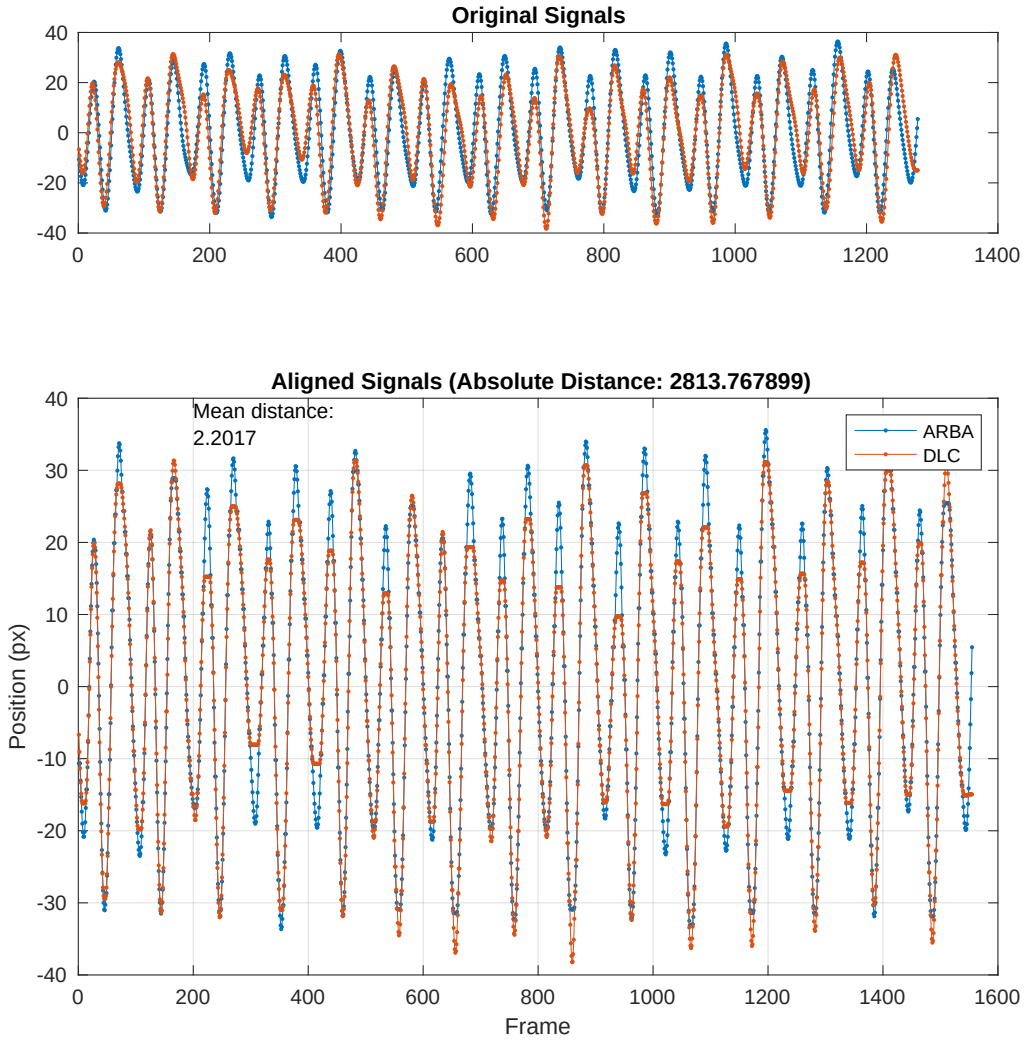


Figura 5.41: Medida de distancia absoluta entre las señales, rodilla eje y.

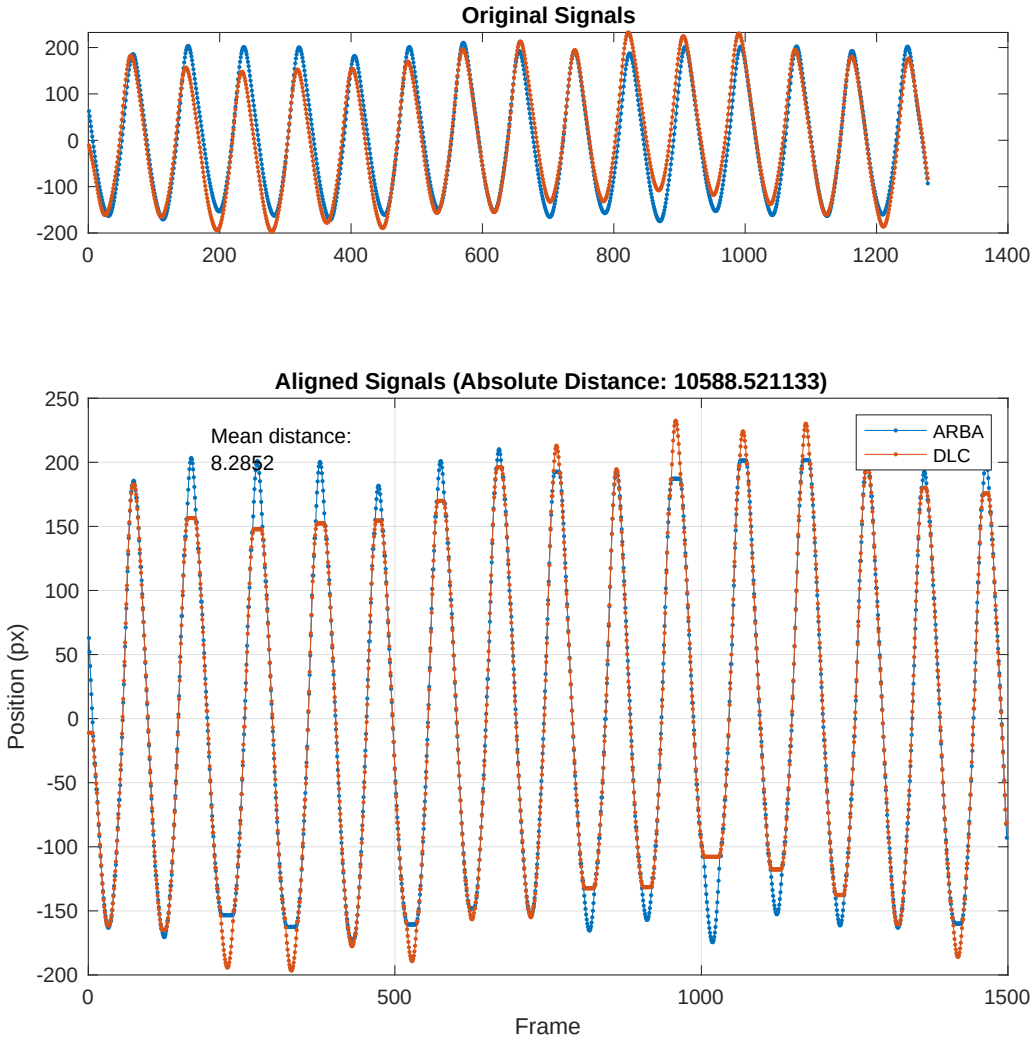


Figura 5.42: Medida de distancia absoluta entre las señales, tobillo eje x.

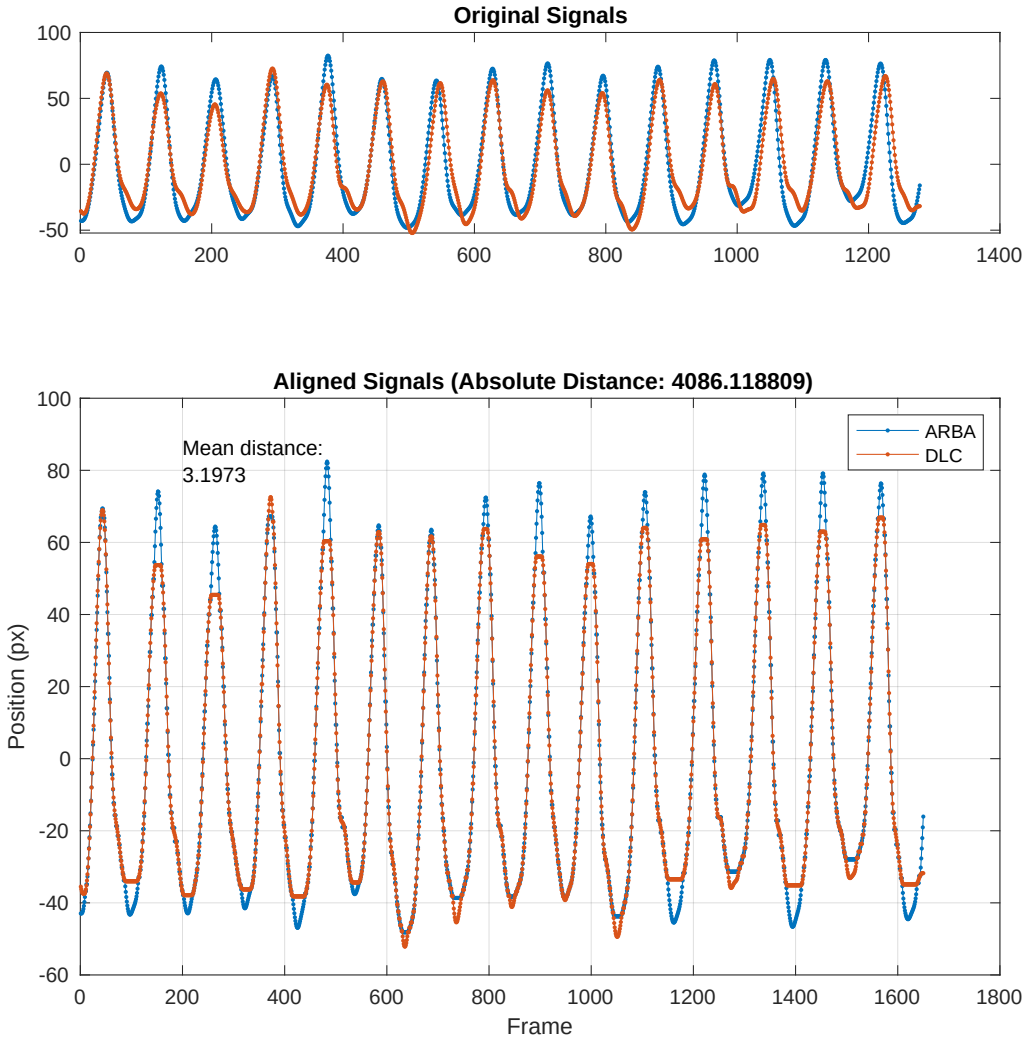


Figura 5.43: Medida de distancia absoluta entre las señales, tobillo eje y.

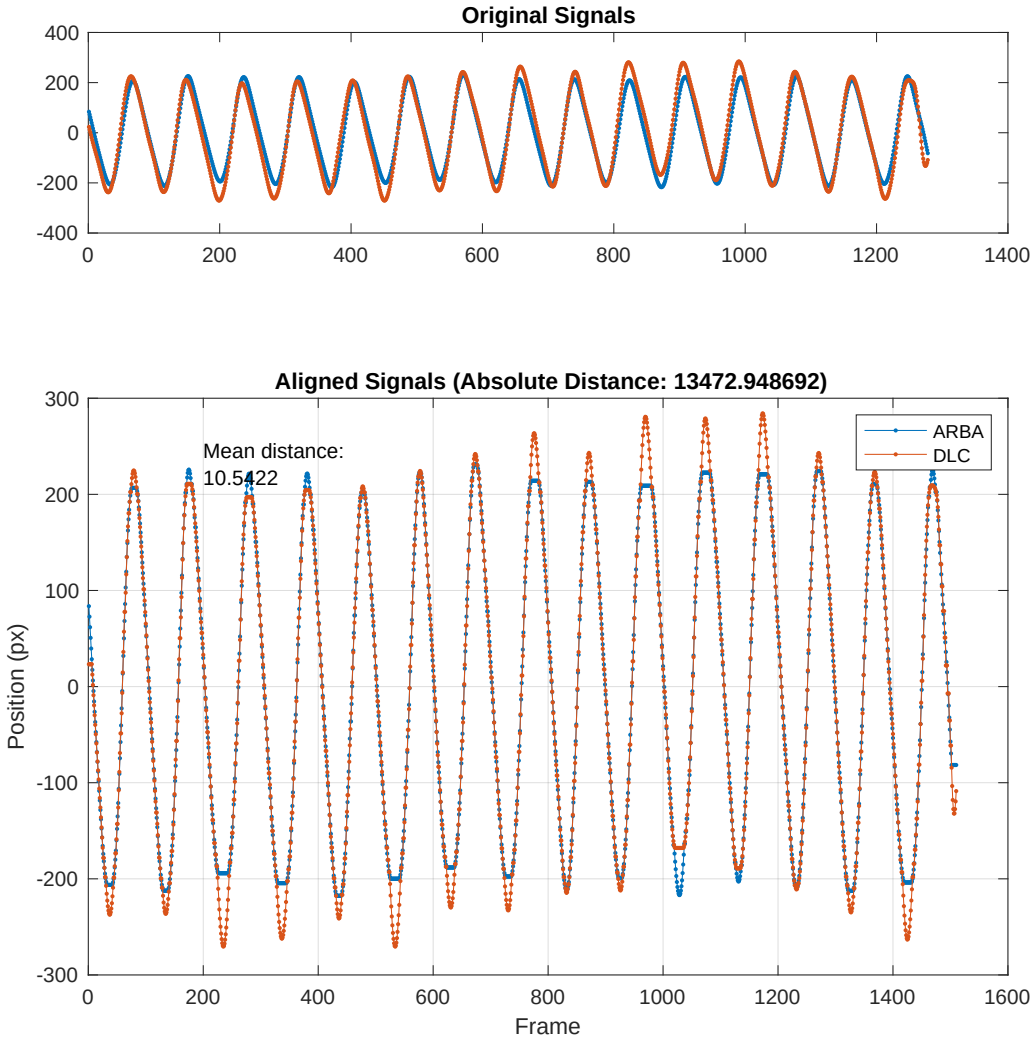


Figura 5.44: Medida de distancia absoluta entre las señales, metatarso eje x.

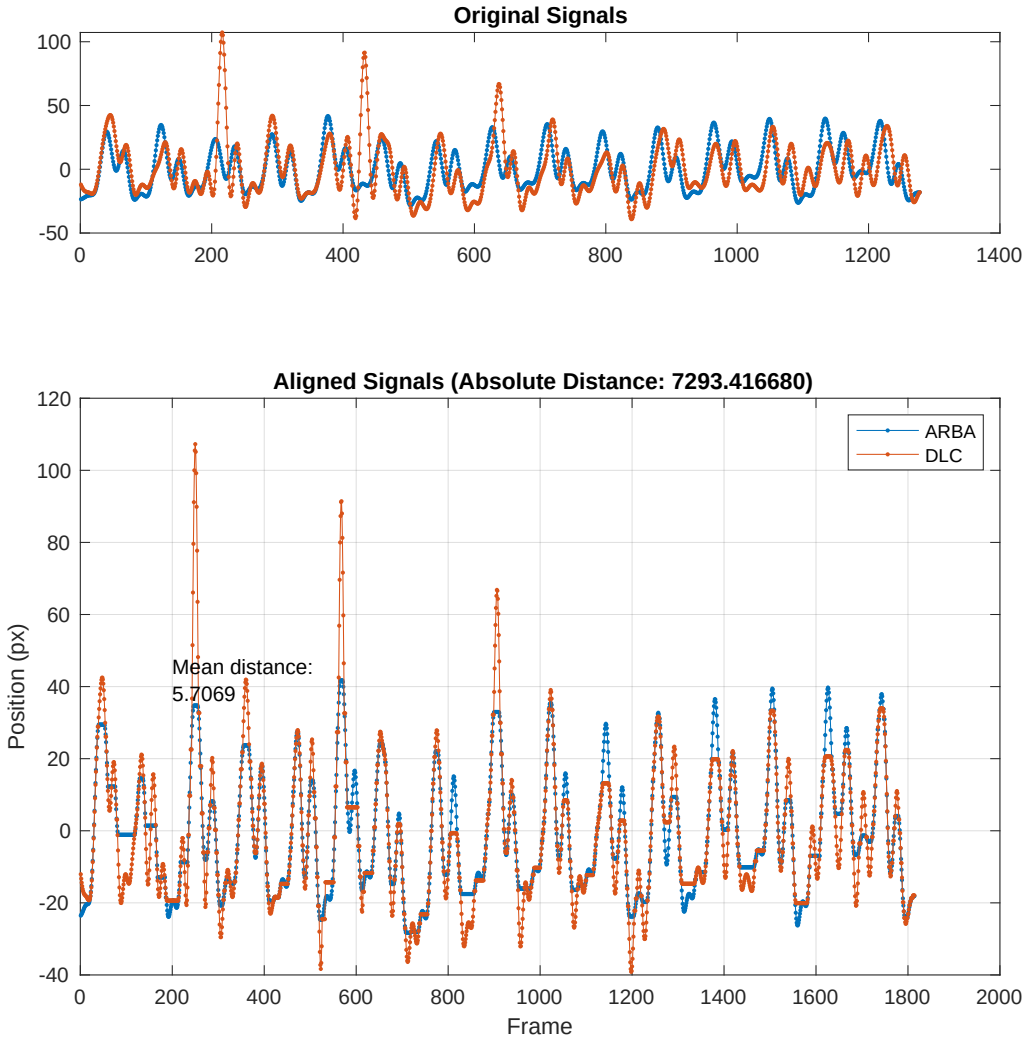


Figura 5.45: Medida de distancia absoluta entre las señales, metatarso eje y.

Tabla 5.5: Distancias absolutas y medias, sujeto 5R.

Dato	eje	Error absoluto	Error medio
Rodilla	x	12457.5552	9.7477
Rodilla	y	2813.7678	2.2017
Tobillo	x	10588.5211	8.2852
Tobillo	y	4086.1188	3.1973
Metatarso	x	13472.9486	10.5422
Metatarso	y	7293.4166	5.7069

En la tabla 5.5 se pueden apreciar los errores absolutos acumulados de las distancias entre cada una de las señales, además de un error medio de cada muestra. El error que surge al normalizar la distancia absoluta en el número de muestras varía entre [2.2 - 10.5] pixeles, siendo una muy buena aproximación del seguimiento real realizado por *ARBA*.

Tabla 5.6: Resumen errores ángulo articular.

Sujeto	Mean	Mean abs	RMSE	Standard
4R	-3.5013	3.6903	4.5319	2.8785
5L	-1.2312	4.5095	6.6198	6.5067
5R	-1.8457	2.9592	3.9561	3.5006
6L	-4.6468	5.3236	6.3438	4.3205
7L	-4.6213	7.0753	8.9921	7.7183
8L	-4.6268	5.9601	7.0165	5.2771
Promedio	-3.4122	4.9197	6.2434	5.0336

Tabla 5.7: Resumen distancias absolutas y medias eje x.

Sujeto	Dato	Eje	Error abs.	Error medio
4R	Rodilla	x	7296.1344	6.328
4R	Tobillo	x	6348.4345	5.506
4R	Metatarso	x	7956.9806	6.9011
5L	Rodilla	x	7828.2832	9.4888
5L	Tobillo	x	6396.8318	7.7537
5L	Metatarso	x	9934.9525	12.0424
5R	Rodilla	x	12457.5552	9.7477
5R	Tobillo	x	10588.5211	8.2852
5R	Metatarso	x	13472.9486	10.5422
6L	Rodilla	x	5203.6526	4.5328
6L	Tobillo	x	7045.4360	6.1371
6L	Metatarso	x	14376.8548	12.5234
7L	Rodilla	x	4452.8186	5.1958
7L	Tobillo	x	4431.2697	5.1707
7L	Metatarso	x	7967.6097	9.2971
8L	Rodilla	x	12083.2171	10.1796
8L	Tobillo	x	7762.3540	6.5395
8L	Metatarso	x	12085.1579	10.1813
Promedio			8760.5006	8.1306

Tabla 5.8: Resumen distancias absolutas y medias eje y.

Sujeto	Dato	Eje	Error abs.	Error medio
4R	Rodilla	y	2432.5498	2.1098
4R	Tobillo	y	3200.6438	2.7759
4R	Metatarso	y	4345.4864	3.7689
5L	Rodilla	y	2689.1060	3.2595
5L	Tobillo	y	3167.5975	3.8395
5L	Metatarso	y	4002.2471	4.8512
5R	Rodilla	y	2813.7678	2.2017
5R	Tobillo	y	4086.1188	3.1973
5R	Metatarso	y	7293.4166	5.7069
6L	Rodilla	y	3328.8578	2.8997
6L	Tobillo	y	3697.1338	3.2205
6L	Metatarso	y	5576.1980	4.8573
7L	Rodilla	y	1477.5741	1.7241
7L	Tobillo	y	1492.0876	1.7411
7L	Metatarso	y	4316.7412	5.037
8L	Rodilla	y	2875.8425	2.4228
8L	Tobillo	y	4179.0524	3.5207
8L	Metatarso	y	8628.0307	7.2688
Promedio			3866.8028	3.5779

Finalmente, se renderizan los videos de salida (figura 5.46) con los datos de posición obtenidos con cada método para ser analizados por un especialista, quien concluye: "Mi opinión clínica es que los resultados son promisorios dado que se puede desarrollar un sistema de análisis sin marcadores, facilitando al evaluación clínica y así optimizar procesos".

(a) ARBA. **(b)** DLC.

Figura 5.46: Videos de salida, sujeto 5R.

6. Conclusiones

En base a los datos de la tabla 5.2 se considera que se logró un entrenamiento lo suficientemente bueno para el propósito del análisis del trote humano.

Tomando en consideración la distribución de los histogramas de las figuras 7.51, 7.60, 7.69, 7.78, 7.87 y 7.96 se puede concluir que el seguimiento realizado por ambos sistemas es estadísticamente comparable, por lo que en un principio los resultados de posición calculados no presentan una diferencia significativa en cuanto a precisión.

Luego de analizar el espectro de las distintas señales (representadas en las figuras figuras 7.52, 7.61, 7.70, 7.79, 7.88 y 7.97) se puede concluir que la distribución en frecuencia y amplitud de los pares de señales analizadas son iguales, es decir, sus espectros son idénticos. Esto refuerza la noción de que el seguimiento hecho por ambos sistemas es, una vez más, comparable, encontrando que ambas señales poseen la misma frecuencia fundamental y armónicos.

Sin embargo, la evidencia más significativa de la correlación entre las señales corresponde a la comparación del ángulo articular (figuras 7.53, 7.62, 7.71, 7.80, 7.89 y 7.98), expuesta en la tabla 5.6, presentando errores medios que varían en magnitud entre 1.2312 y 8.9921 grados.

Otro factor a favor corresponde a las distancias calculadas en las tablas 7.11, 7.14, 7.17, 7.20, 7.23 y 7.26 que dan muestra de la insignificante diferencia absoluta presentada entre ambos métodos de medición. Para una mejor visualización de esto se presentan los resultados condensados en las tablas 5.7 y 5.8, en dónde los errores promedio corresponden a 8.1306 y 3.5779 pixeles, los cuales representan errores porcentuales de 0.4234 % en el eje x y 0.3312 % en el eje y.

En base a estos resultados se espera implementar el sistema *DLC* dentro de *ARBA* para conseguir una versión semi-markerless, reduciendo con esto significativamente el uso de marcadores activos en el proceso. Otra opción es el uso de marcadores pasivos, que queda como caso de estudio a futuro.

Referencias

[1] M. Roberts, D. Mongeon, and F. Prince, "Biomechanical parameters for gait analysis: a systematic review of healthy human gait," *Physical Therapy and Rehabilitation*, vol. 4, no. 1, p. 6, 2017. [Online]. Available: http://www.hoajonline.com/phystherrehabil/2055-2386/4/6

[2] F. Coutts, "Gait analysis in the therapeutic environment," *Manual Therapy*, vol. 4, no. 1, pp. 2–10, Feb. 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1356689X99800034

[3] "Biomecánica de la marcha atlética. Análisis cinemático de su desarrollo y comparación con la marcha normal." [Online]. Available: https://www.medigraphic.com/cgi-bin/new/resumen.cgi?IDARTICULO=78993

[4] L. E. C. Bravo, J. A. T. Ortiz, and L. F. V. Tamayo, "Análisis Biomecánico de Marcha Humana a Través de Técnicas de Modelaje," *Entre Ciencia e Ingeniería*, vol. 6, no. 12, pp. 29–35, 2012. [Online]. Available: https://revistas.ucp.edu.co/index.php/entrecienciaeingenieria/article/view/663

[5] "Análisis biomecánico en la marcha deportiva entre deportistas de iniciación y alto rendimiento." [Online]. Available: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S0864-03002018000200002

[6] E. C. Moreno, "Análisis biomecánico de la marcha en rampa," http://purl.org/dc/dcmitype/Text, Universidad de Zaragoza, 1997. [Online]. Available: https://dialnet.unirioja.es/servlet/tesis?codigo=203869

[7] T. Nath, A. Mathis, A. C. Chen, A. Patel, M. Bethge, and M. W. Mathis, "Using DeepLabCut for 3D markerless pose estimation across species and behaviors," *Nature Protocols*, vol. 14, no. 7, pp. 2152–2176, Jul. 2019. [Online]. Available: http://www.nature.com/articles/s41596-019-0176-0

[8] J. Lauer, M. Zhou, S. Ye, W. Menegas, T. Nath, M. M. Rahman, V. D. Santo, D. Soberanes, G. Feng, V. N. Murthy, G. Lauder, C. Dulac, M. W. Mathis, and A. Mathis, "Multi-animal pose estimation and tracking with DeepLabCut," Mathis Laboratory of Adaptative Motor Control, Tech. Rep., Apr. 2021, type: article. [Online]. Available: https://www.biorxiv.org/content/10.1101/2021.04.30.442096v1

[9] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge, "DeepLabCut: markerless pose estimation of user-defined body parts with deep learning," *Nature Neuroscience*, vol. 21, no. 9, pp. 1281–1289, Sep. 2018. [Online]. Available: https://www.nature.com/articles/s41593-018-0209-y

[10] M. W. Mathis and A. Mathis, "Deep learning tools for the measurement of animal behavior in neuroscience," *Current Opinion in Neurobiology*, vol. 60, pp. 1–11, Feb. 2020, arXiv: 1909.13868. [Online]. Available: http://arxiv.org/abs/1909.13868

[11] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "DeeperCut: A Deeper, Stronger, and Faster Multi-Person Pose Estimation Model," *arXiv:1605.03170 [cs]*, Nov. 2016, arXiv: 1605.03170. [Online]. Available: http://arxiv.org/abs/1605.03170

[12] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. Gehler, and B. Schiele, "DeepCut: Joint Subset Partition and Labeling for Multi Person Pose Estimation," *arXiv:1511.06645 [cs]*, Apr. 2016, arXiv: 1511.06645. [Online]. Available: http://arxiv.org/abs/1511.06645

[13] A. Mathis, T. Biasi, S. Schneider, M. Yüksekgönül, B. Rogers, M. Bethge, and M. W. Mathis, "Pretraining boosts out-of-domain robustness for pose estimation," *arXiv:1909.11229 [cs]*, Nov. 2020, arXiv: 1909.11229. [Online]. Available: http://arxiv.org/abs/1909.11229

[14] A. Mathis, S. Schneider, J. Lauer, and M. W. Mathis, "A Primer on Motion Capture with Deep Learning: Principles, Pitfalls and Perspectives," *Neuron*, vol. 108, no. 1, pp. 44–65, Oct. 2020, arXiv: 2009.00564. [Online]. Available: http://arxiv.org/abs/2009.00564

[15] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," Apr. 2018. [Online]. Available: https://arxiv.org/abs/1804.02767v1

[16] M. Gabel, R. Gilad-Bachrach, E. Renshaw, and A. Schuster, "Full body gait analysis with Kinect," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Aug. 2012, pp. 1964–1967, iSSN: 1558-4615.

[17] J. Latorre, C. Colomer, M. Alcañiz, and R. Llorens, "Gait analysis with the Kinect v2: normative study with healthy individuals and comprehensive study of its sensitivity, validity, and reliability in individuals with stroke," *Journal of NeuroEngineering and Rehabilitation*, vol. 16, no. 1, p. 97, Dec. 2019. [Online]. Available: https://jneuroengrehab.biomedcentral.com/articles/10.1186/s12984-019-0568-y

[18] S. Chakraborty, A. Nandy, T. Yamaguchi, V. Bonnet, and G. Venture, "Accuracy of image data stream of a markerless motion capture system in determining the local dynamic stability and joint kinematics of human gait," *Journal of Biomechanics*, vol. 104, p. 109718, May 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021929020301342

[19] "CONSIDERACIONES PARA EL ANÁLISIS DE LA MARCHA HUMANA. TÉCNICAS DE VIDEOGRAMETRÍA, ELECTROMIOGRAFÍA Y DINAMOMETRÍA." [Online]. Available: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1909-97622008000100004

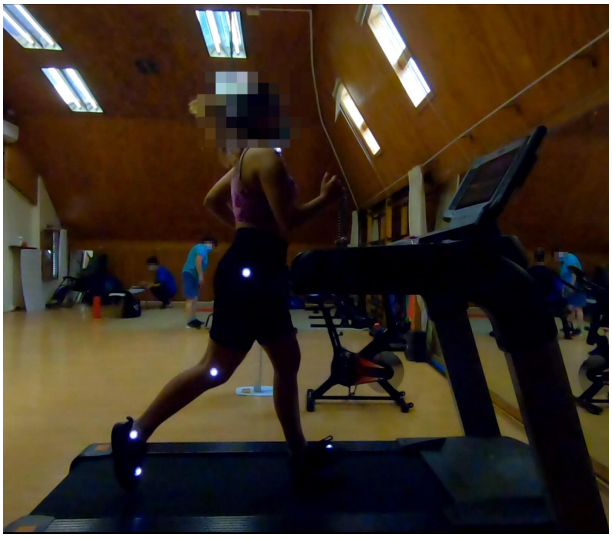
[20] A. Mathis and R. Warren, "On the inference speed and video-compression robustness of DeepLabCut," Mathis Laboratory of Adaptative Motor Control, Tech. Rep., Oct. 2018, type: article. [Online]. Available: https://www.biorxiv.org/content/10.1101/457242v1

[21] "Distance between signals using dynamic time warping - MATLAB dtw." [Online]. Available: https://www.mathworks.com/help/signal/ref/dtw.html

7. Anexos

7.1. Imágenes

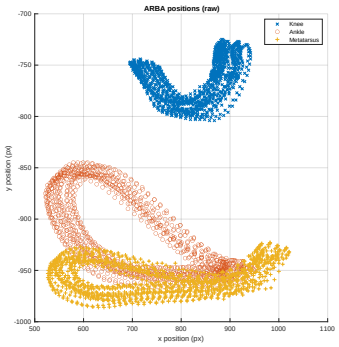
7.1.1. Sujeto 4, plano sagital derecho



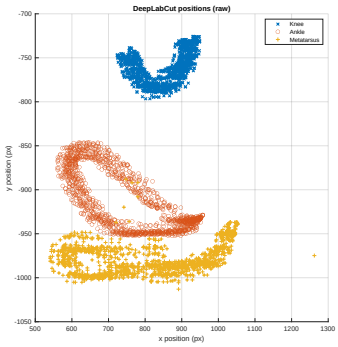
(a) ARBA.

(b) DLC.

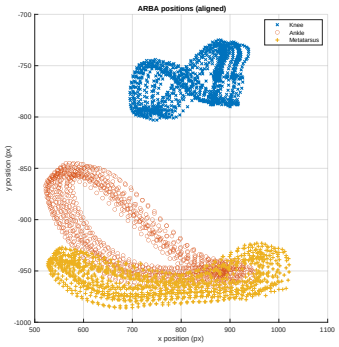
Figura 7.47: Sujeto 4R.



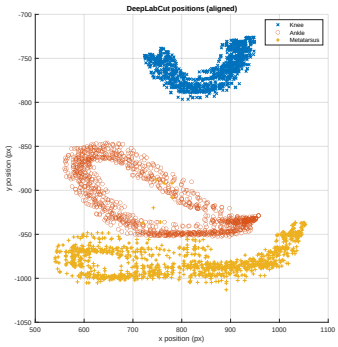
(a) ARBA, posición original.



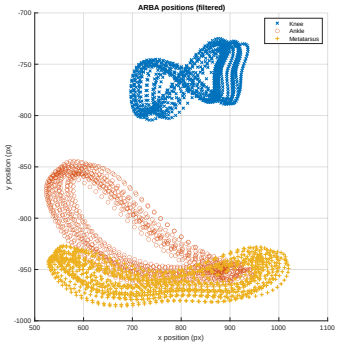
(b) DLC, posición original.



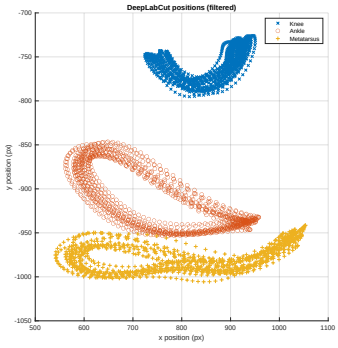
(c) ARBA, posición alineada.



(d) DLC, posición alineada.

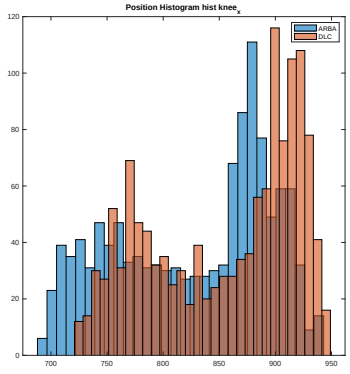


(e) ARBA, posición filtrada.

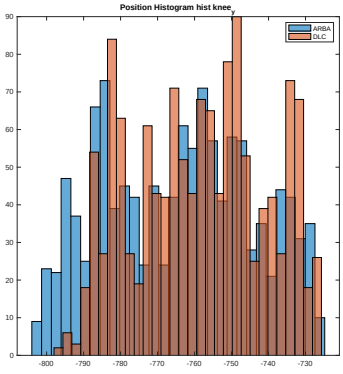


(f) DLC, posición filtrada.

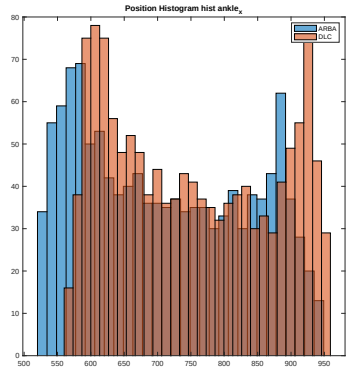
Figura 7.48: Gráficas de posición, Sujeto 4R.



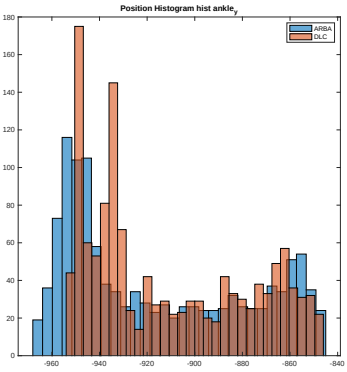
(a) Rodilla x.



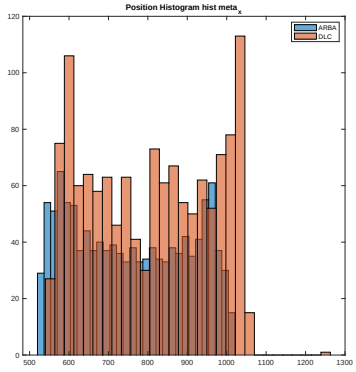
(b) Rodilla y.



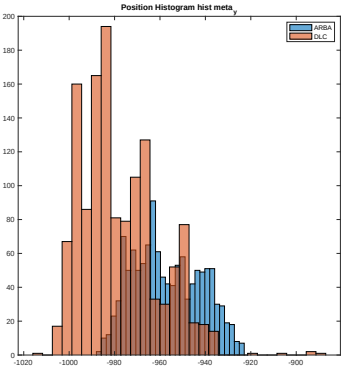
(c) Tobillo x.



(d) Tobillo y.

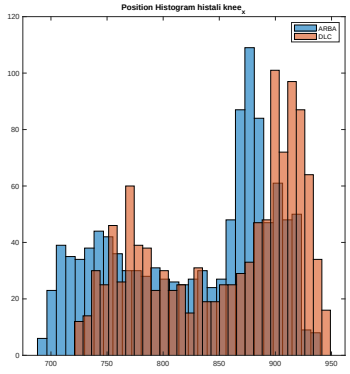


(e) Metatarso x.

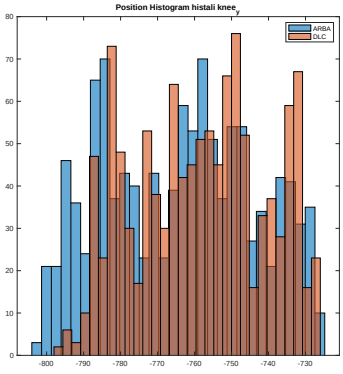


(f) Metatarso y.

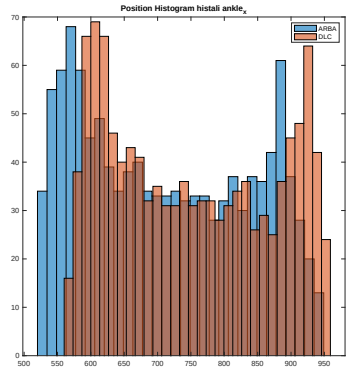
Figura 7.49: Histogramas de posición, Sujeto 4R.



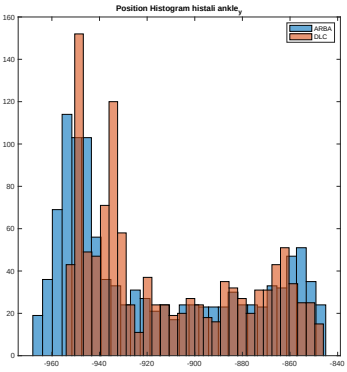
(a) Rodilla x.



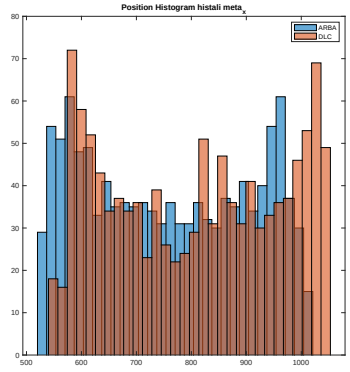
(b) Rodilla y.



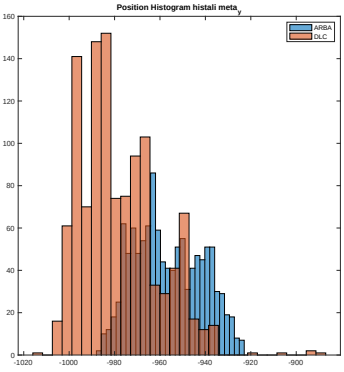
(c) Tobillo x.



(d) Tobillo y.

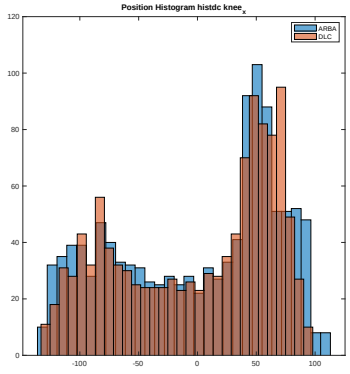


(e) Metatarso x.

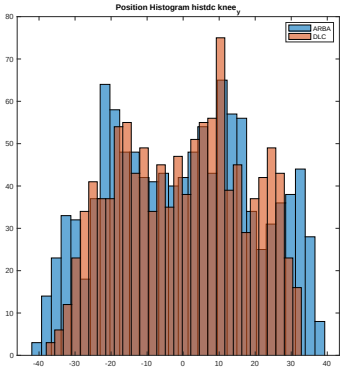


(f) Metatarso y.

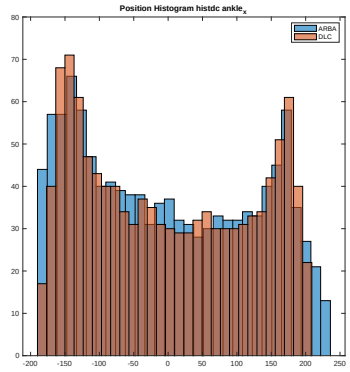
Figura 7.50: Histogramas de posición alineados, Sujeto 4R.



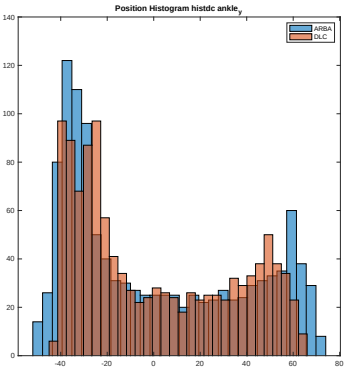
(a) Rodilla x.



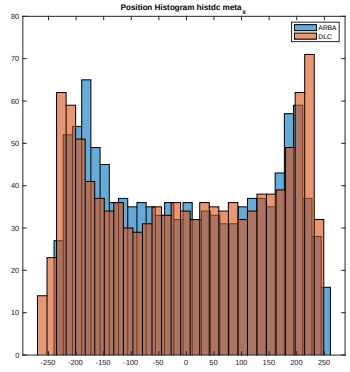
(b) Rodilla y.



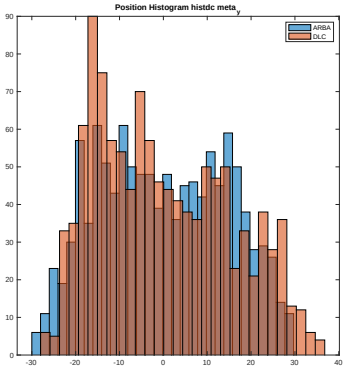
(c) Tobillo x.



(d) Tobillo y.

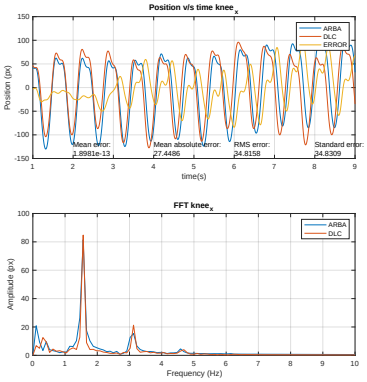


(e) Metatarso x.

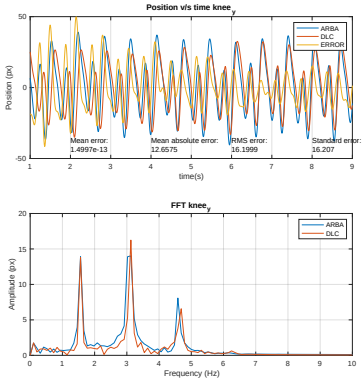


(f) Metatarso y.

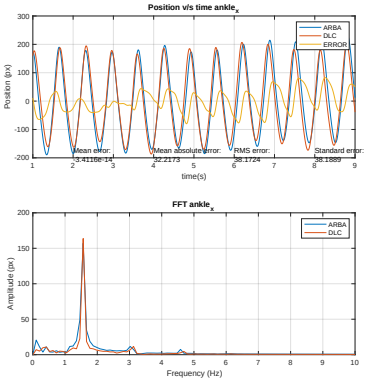
Figura 7.51: Histogramas de posición filtrados, Sujeto 4R.



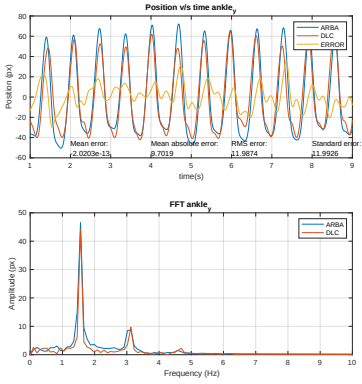
(a) Rodilla x.



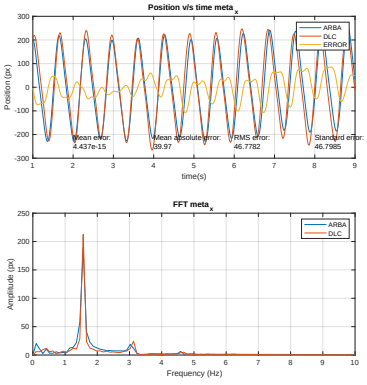
(b) Rodilla y.



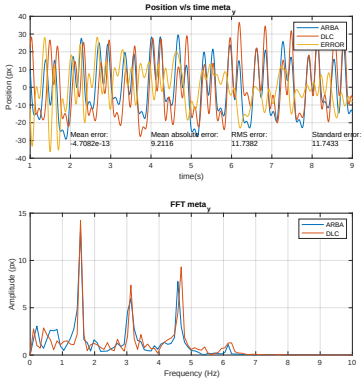
(c) Tobillo x.



(d) Tobillo y.



(e) Metatarso x.



(f) Metatarso y.

Figura 7.52: FFT de las señales, Sujeto 4R.

Tabla 7.9: Errores, Sujeto 4R.

Dato	Eje	Tipo	Valor
Rodilla	x	Mean	1.8981e-13
Rodilla	x	Mean abs	27.4486
Rodilla	x	RMSE	34.8158
Rodilla	x	Standard	34.8309
Rodilla	y	Mean	1.4997e-13
Rodilla	y	Mean abs	12.6575
Rodilla	y	RMSE	16.1999
Rodilla	y	Standard	16.207
Tobillo	x	Mean	-3.4116e-14
Tobillo	x	Mean abs	32.2173
Tobillo	x	RMSE	38.1724
Tobillo	x	Standard	38.1889
Tobillo	y	Mean	-2.0203e-13
Tobillo	y	Mean abs	9.7019
Tobillo	y	RMSE	11.9874
Tobillo	y	Standard	11.9926
Metatarso	x	Mean	4.437e-15
Metatarso	x	Mean abs	39.97
Metatarso	x	RMSE	46.7782
Metatarso	x	Standard	46.7985
Metatarso	y	Mean	-4.7082e-13
Metatarso	y	Mean abs	9.2116
Metatarso	y	RMSE	11.7382
Metatarso	y	Standard	11.7433

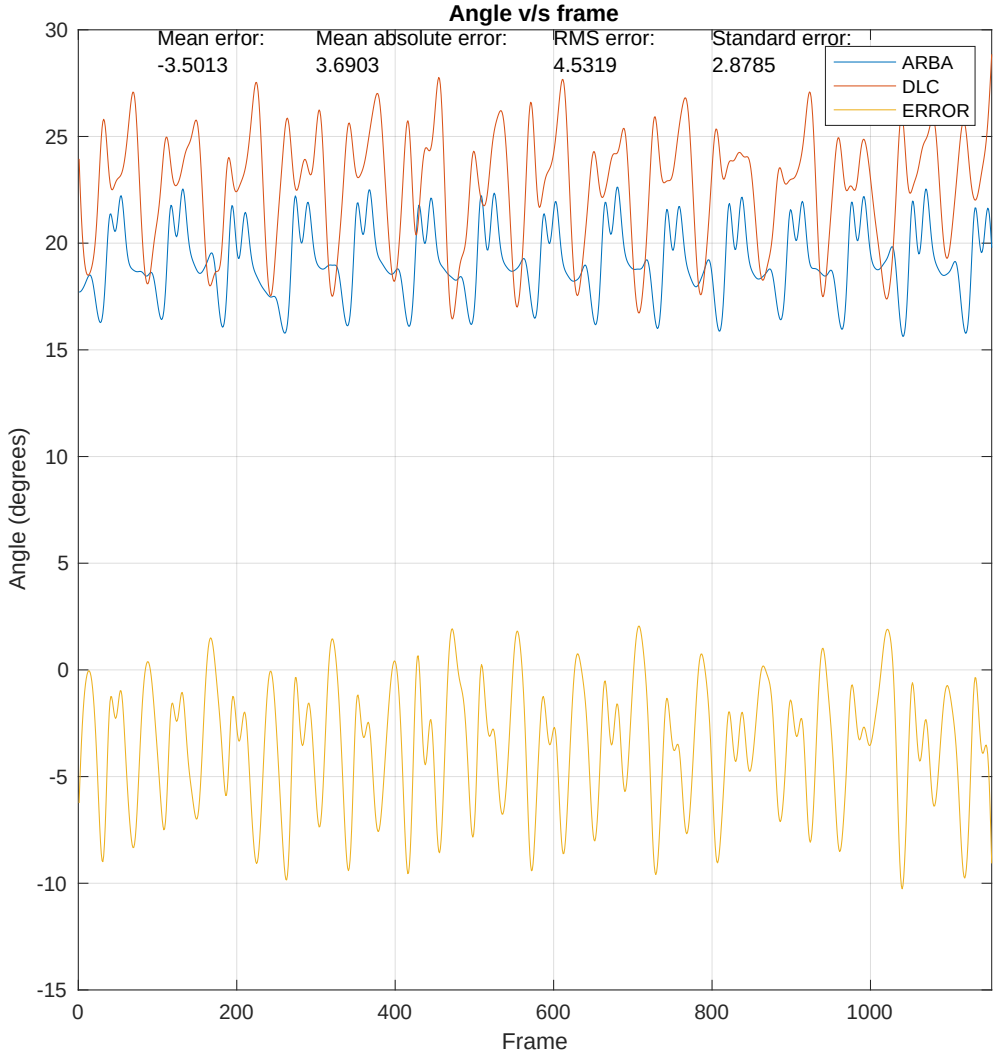
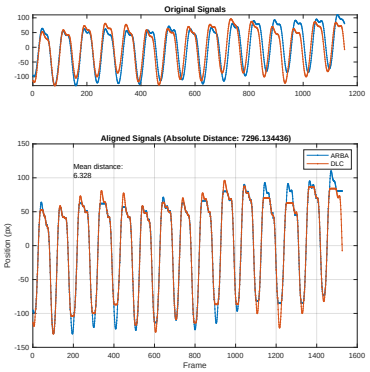


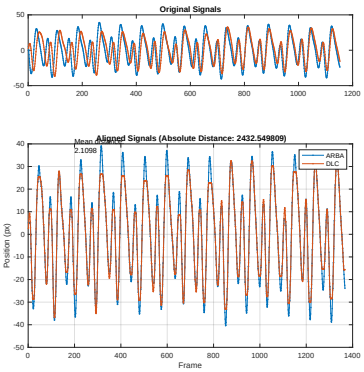
Figura 7.53: Comparación de ángulos y error medio, Sujeto 4R.

Tabla 7.10: Errores ángulo articular, Sujeto 4R.

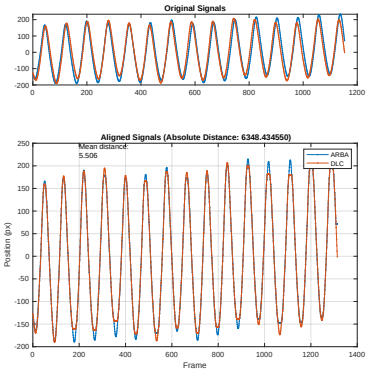
Tipo	Mean	Mean abs	RMSE	Standard
Valor	-3.5013	3.6903	4.5319	2.8785



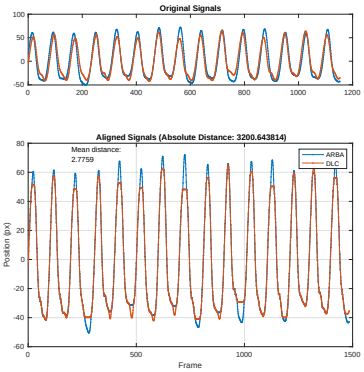
(a) Rodilla x.



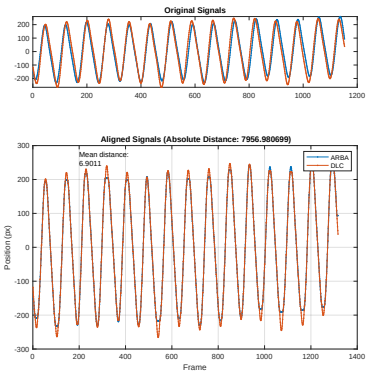
(b) Rodilla y.



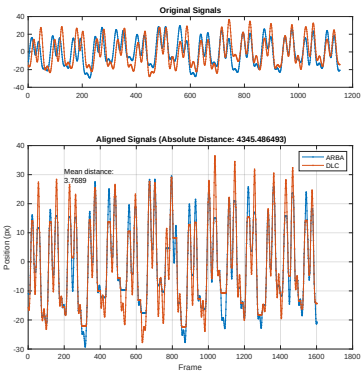
(c) Tobillo x.



(d) Tobillo y.



(e) Metatarso x.



(f) Metatarso y.

Figura 7.54: Distancia absoluta entre las señales, Sujeto 4R.

Tabla 7.11: Distancias absolutas y medias, Sujeto 4R.

Dato	Eje	Error absoluto	Error medio
Rodilla	x	7296.1344	6.328
Rodilla	y	2432.5498	2.1098
Tobillo	x	6348.4345	5.506
Tobillo	y	3200.6438	2.7759
Metatarso	x	7956.9806	6.9011
Metatarso	y	4345.4864	3.7689

(a) ARBA. **(b)** DLC.

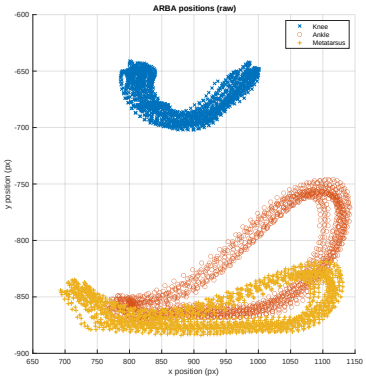
Figura 7.55: Videos de salida, Sujeto 4R.

7.1.2. Sujeto 5, plano sagital izquierdo

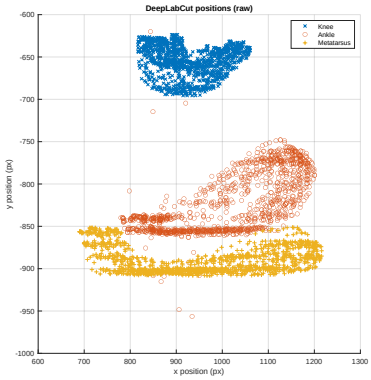
(a) ARBA.

(b) DLC.

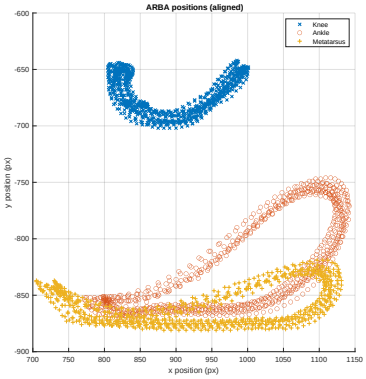
Figura 7.56: Sujeto 5L.



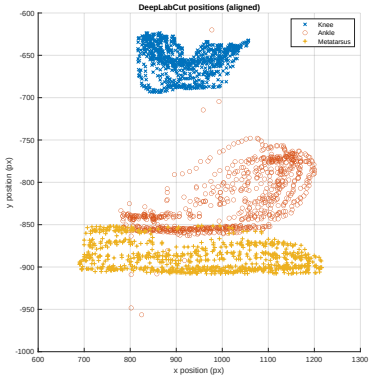
(a) ARBA, posición original.



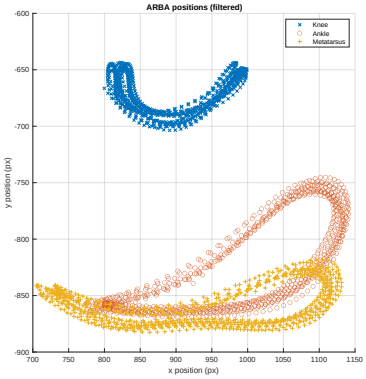
(b) DLC, posición original.



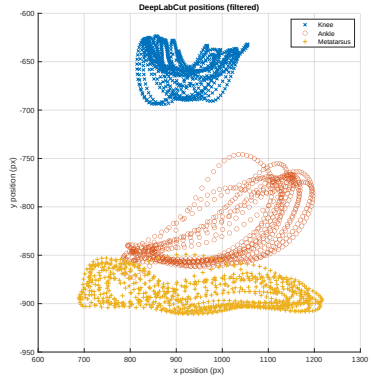
(c) ARBA, posición alineada.



(d) DLC, posición alineada.

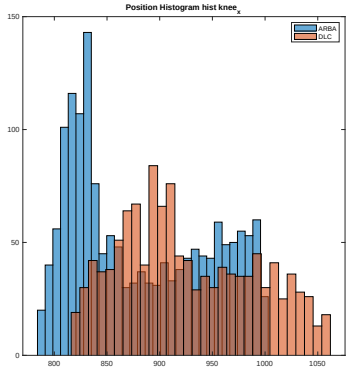


(e) ARBA, posición filtrada.

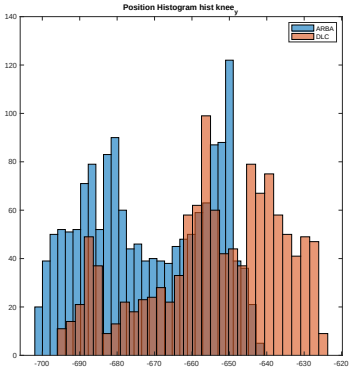


(f) DLC, posición filtrada.

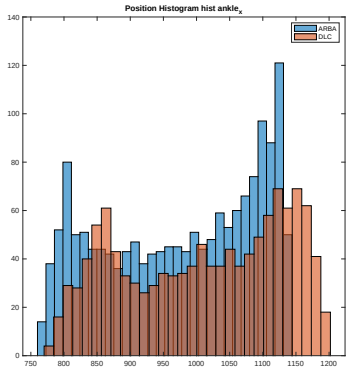
Figura 7.57: Gráficas de posición, Sujeto 5L.



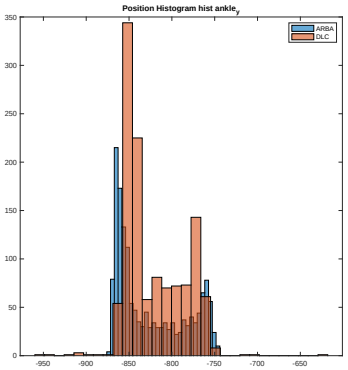
(a) Rodilla x.



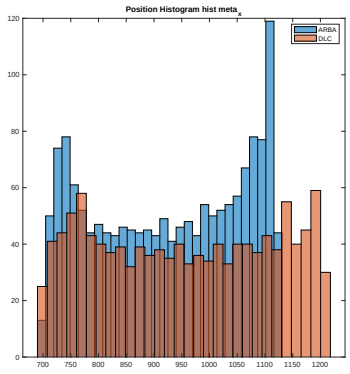
(b) Rodilla y.



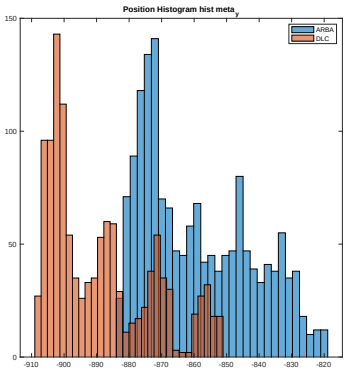
(c) Tobillo x.



(d) Tobillo y.

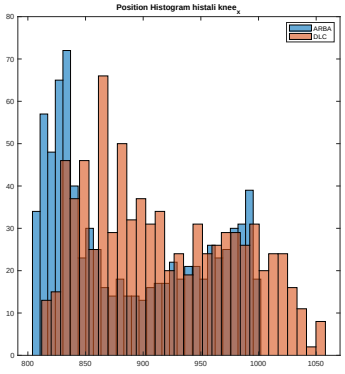


(e) Metatarso x.

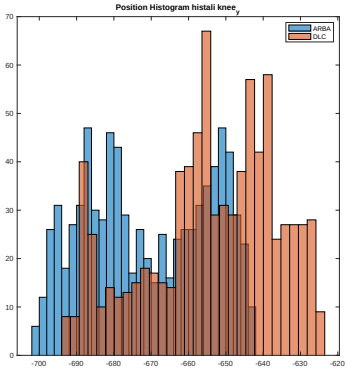


(f) Metatarso y.

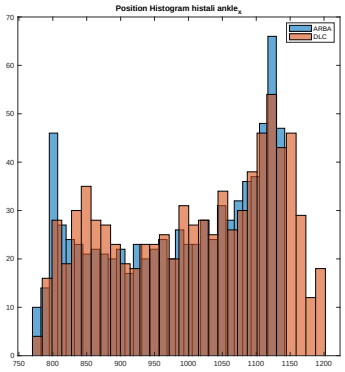
Figura 7.58: Histogramas de posición, Sujeto 5L.



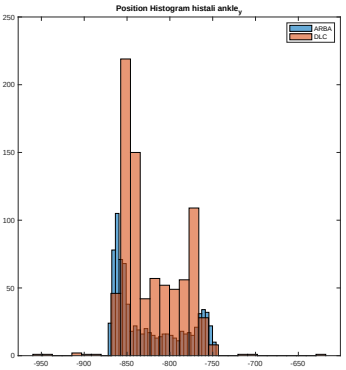
(a) Rodilla x.



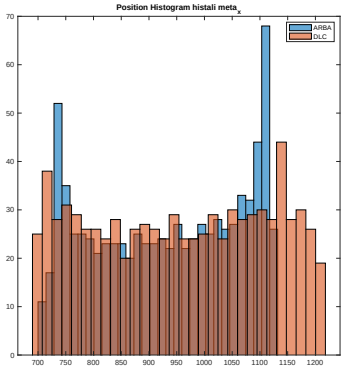
(b) Rodilla y.



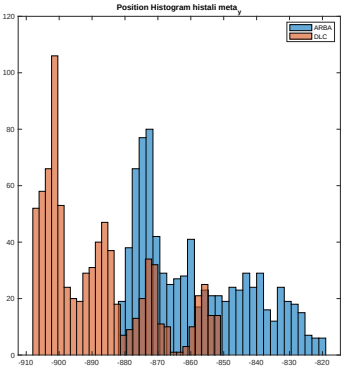
(c) Tobillo x.



(d) Tobillo y.

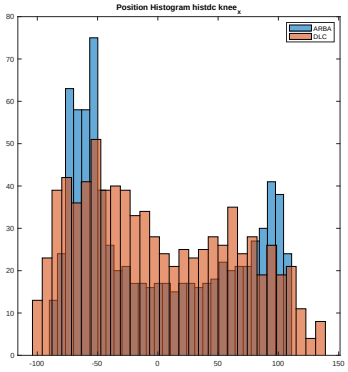


(e) Metatarso x.

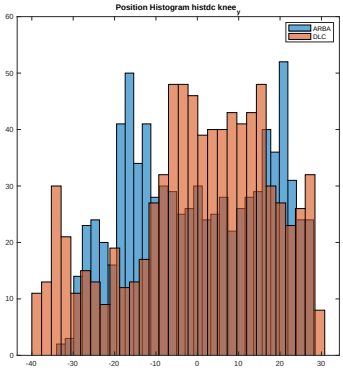


(f) Metatarso y.

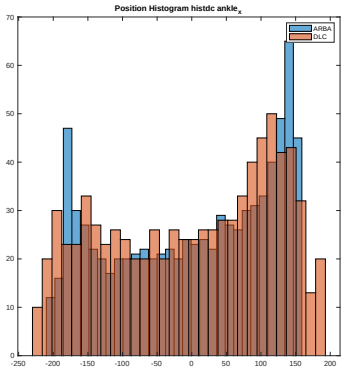
Figura 7.59: Histogramas de posición alineados, Sujeto 5L.



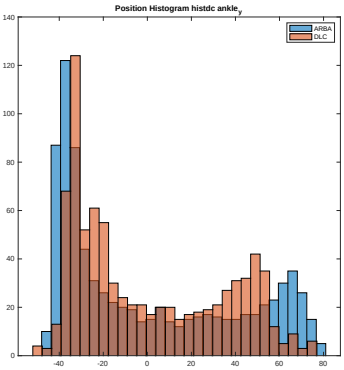
(a) Rodilla x.



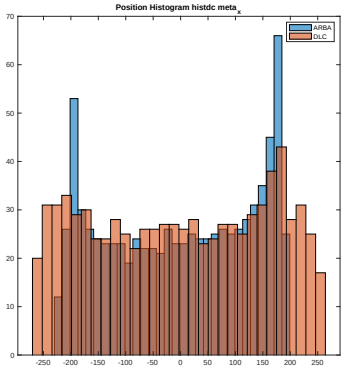
(b) Rodilla y.



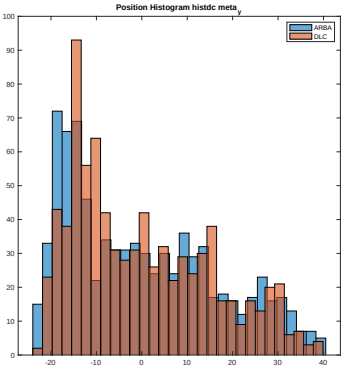
(c) Tobillo x.



(d) Tobillo y.

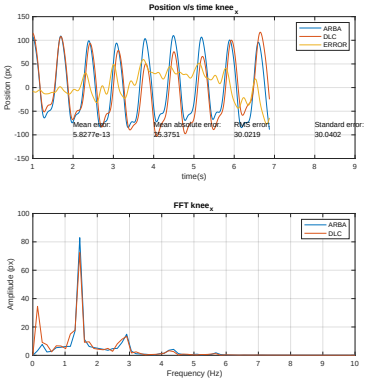


(e) Metatarso x.

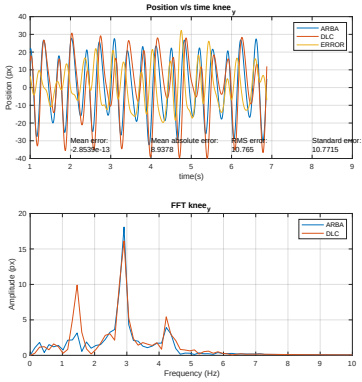


(f) Metatarso y.

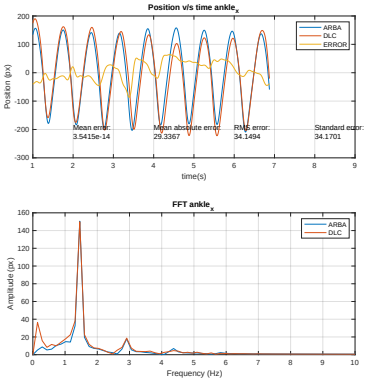
Figura 7.60: Histogramas de posición filtrados, Sujeto 5L.



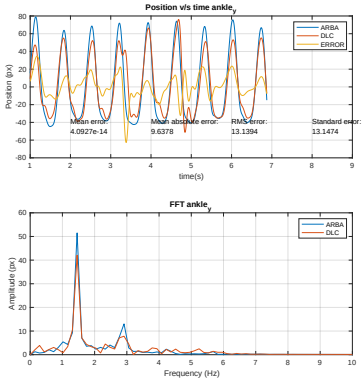
(a) Rodilla x.



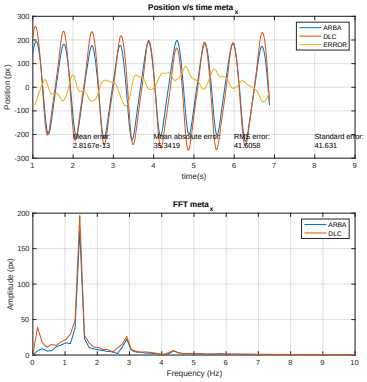
(b) Rodilla y.



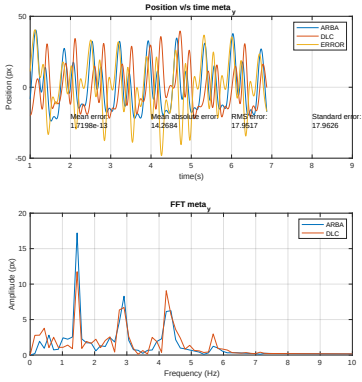
(c) Tobillo x.



(d) Tobillo y.



(e) Metatarso x.



(f) Metatarso y.

Figura 7.61: FFT de las señales, Sujeto 5L.

Tabla 7.12: Errores, Sujeto 5L.

Dato	Eje	Tipo	Valor
Rodilla	x	Mean	-1.8109e-13
Rodilla	x	Mean abs	29.3713
Rodilla	x	RMSE	35.171
Rodilla	x	Standard	35.1837
Rodilla	y	Mean	-2.7762e-13
Rodilla	y	Mean abs	12.8425
Rodilla	y	RMSE	15.4967
Rodilla	y	Standard	15.5023
Tobillo	x	Mean	1.4999e-13
Tobillo	x	Mean abs	30.8233
Tobillo	x	RMSE	55.1228
Tobillo	x	Standard	55.1428
Tobillo	y	Mean	4.4468e-14
Tobillo	y	Mean abs	12.3718
Tobillo	y	RMSE	16.1181
Tobillo	y	Standard	16.1239
Metatarso	x	Mean	6.1373e-13
Metatarso	x	Mean abs	36.1252
Metatarso	x	RMSE	42.9388
Metatarso	x	Standard	42.9544
Metatarso	y	Mean	6.8418e-13
Metatarso	y	Mean abs	10.8679
Metatarso	y	RMSE	13.9565
Metatarso	y	Standard	13.9616

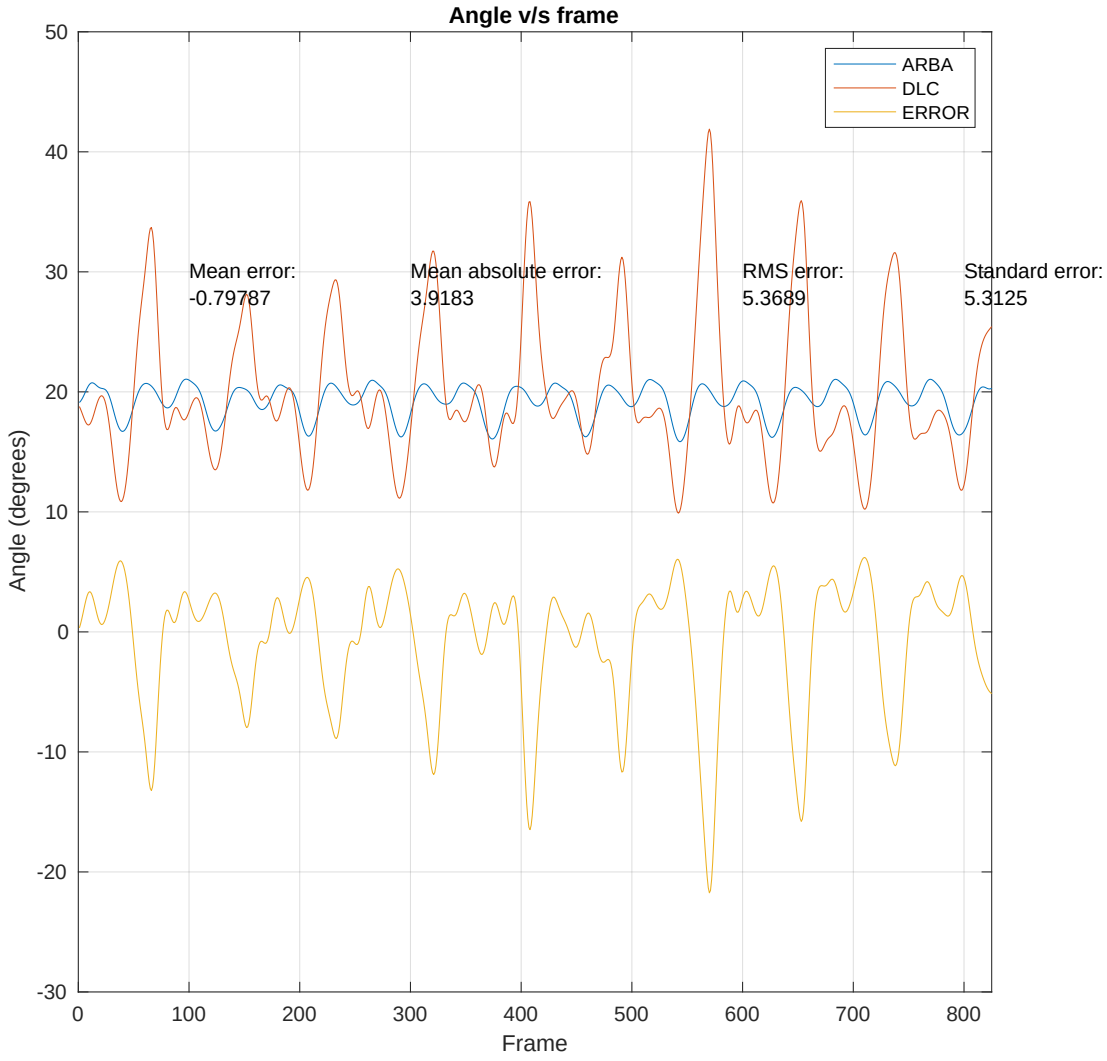
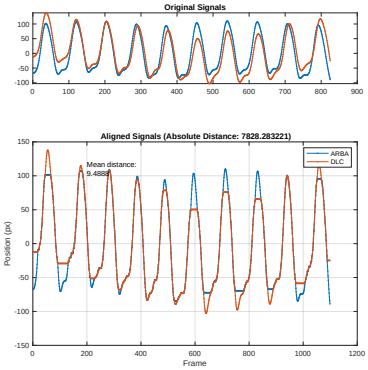


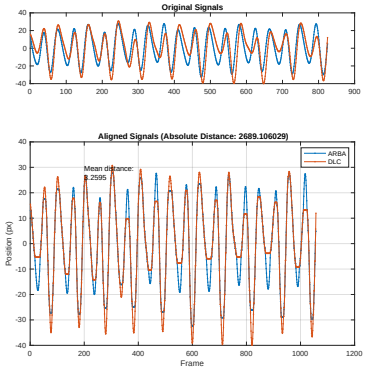
Figura 7.62: Comparación de ángulos y error medio, Sujeto 5L.

Tabla 7.13: Errores ángulo articular, Sujeto 5L.

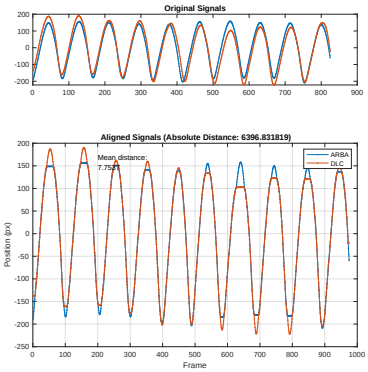
Tipo	Mean	Mean abs	RMSE	Standard
Valor	-1.2312	4.5095	6.6198	6.5067



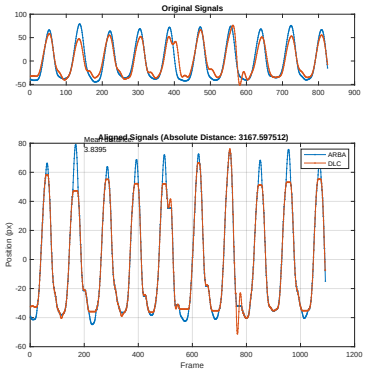
(a) Rodilla x.



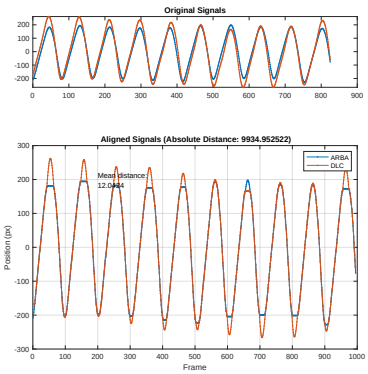
(b) Rodilla y.



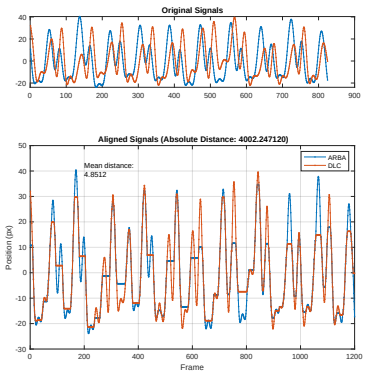
(c) Tobillo x.



(d) Tobillo y.



(e) Metatarso x.

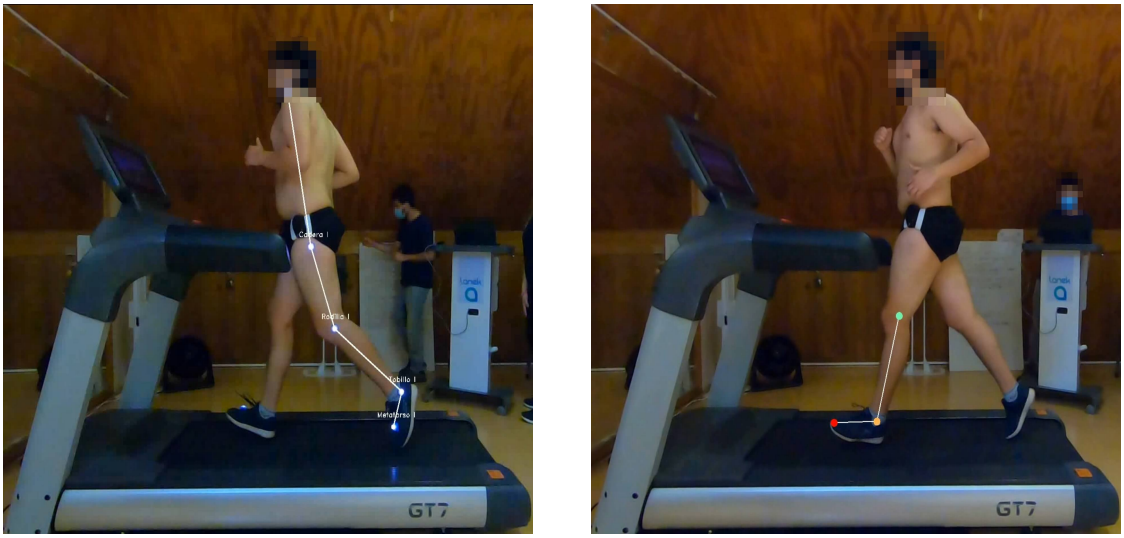


(f) Metatarso y.

Figura 7.63: Distancia absoluta entre las señales, Sujeto 5L.

Tabla 7.14: Distancias absolutas y medias, Sujeto 5L.

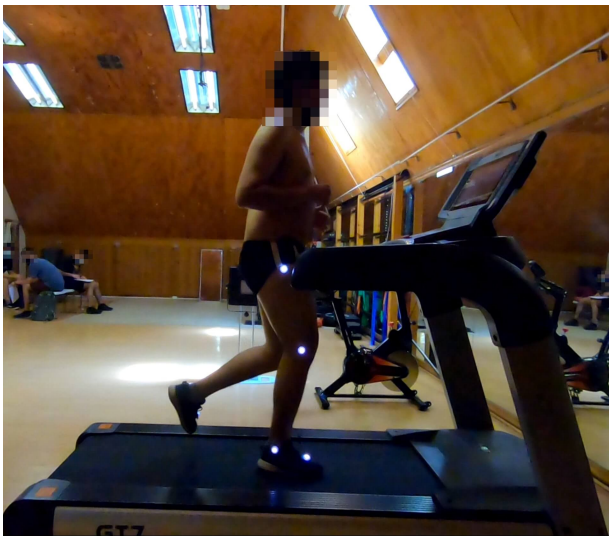
Dato	Eje	Error absoluto	Error medio
Rodilla	x	7828.2832	9.4888
Rodilla	y	2689.1060	3.2595
Tobillo	x	6396.8318	7.7537
Tobillo	y	3167.5975	3.8395
Metatarso	x	9934.9525	12.0424
Metatarso	y	4002.2471	4.8512



(a) ARBA. **(b)** DLC.

Figura 7.64: Videos de salida, Sujeto 5L.

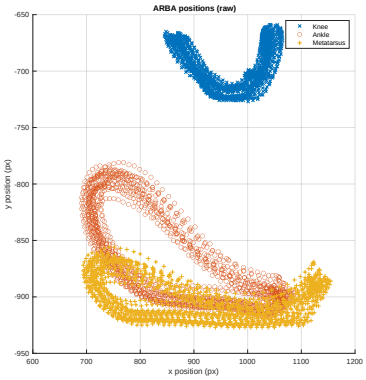
7.1.3. Sujeto 5, plano sagital derecho



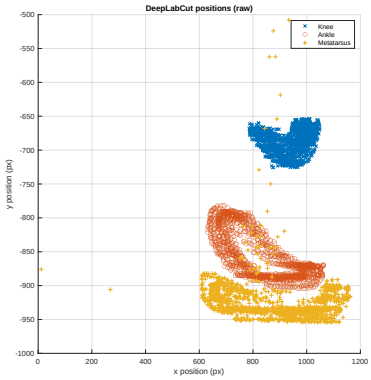
(a) ARBA.

(b) DLC.

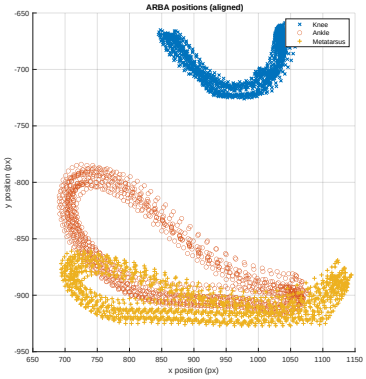
Figura 7.65: Sujeto 5R.



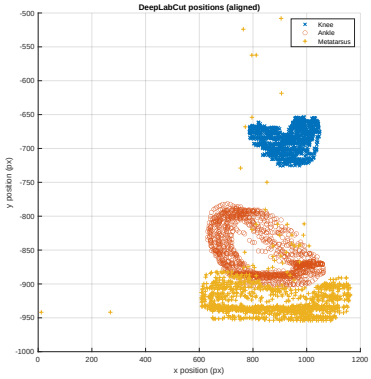
(a) ARBA, posición original.



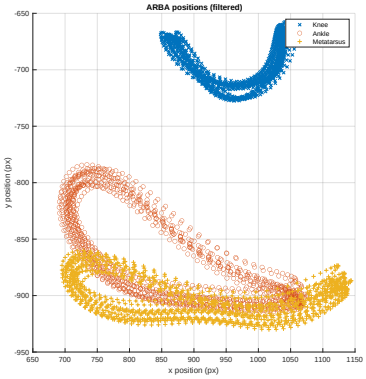
(b) DLC, posición original.



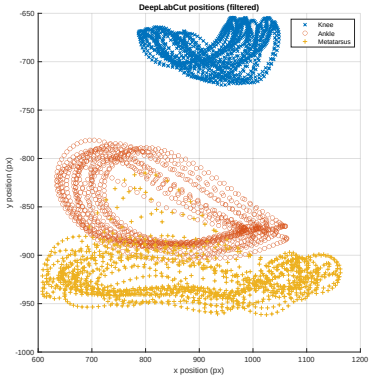
(c) ARBA, posición alineada.



(d) DLC, posición alineada.

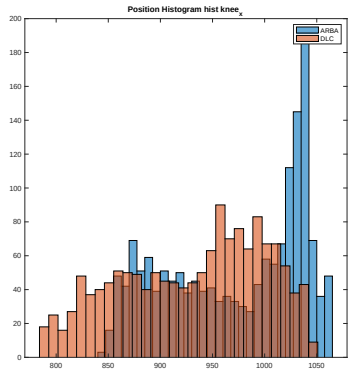


(e) ARBA, posición filtrada.

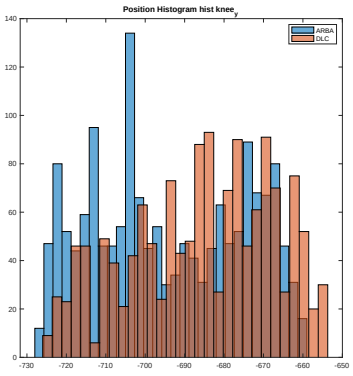


(f) DLC, posición filtrada.

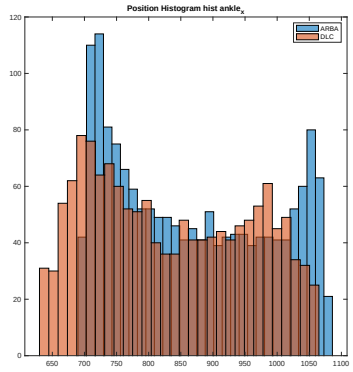
Figura 7.66: Gráficas de posición, Sujeto 5R.



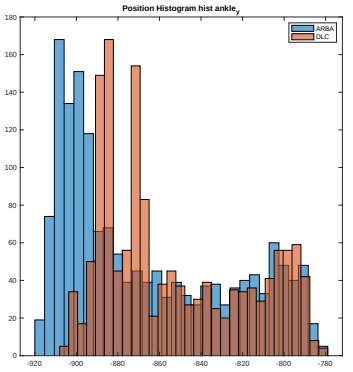
(a) Rodilla x.



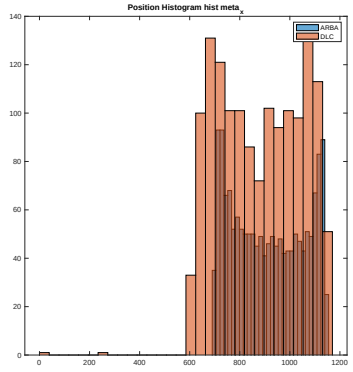
(b) Rodilla y.



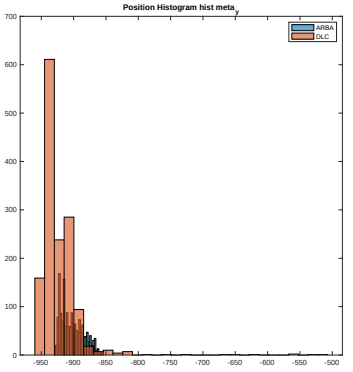
(c) Tobillo x.



(d) Tobillo y.

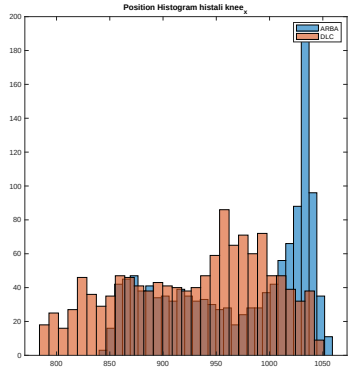


(e) Metatarso x.

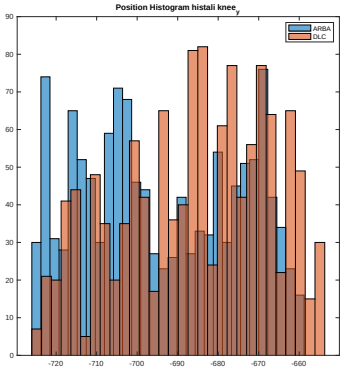


(f) Metatarso y.

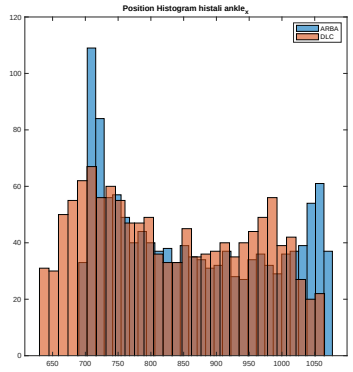
Figura 7.67: Histogramas de posición, Sujeto 5R.



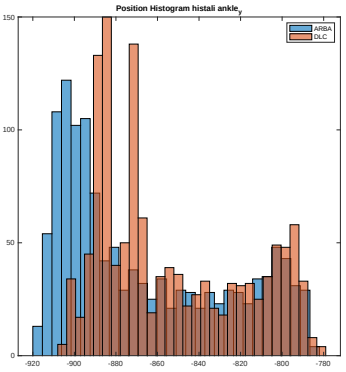
(a) Rodilla x.



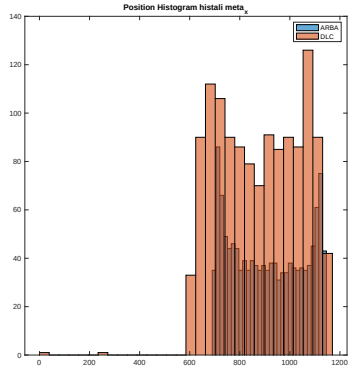
(b) Rodilla y.



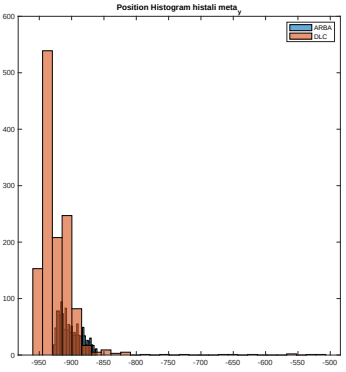
(c) Tobillo x.



(d) Tobillo y.

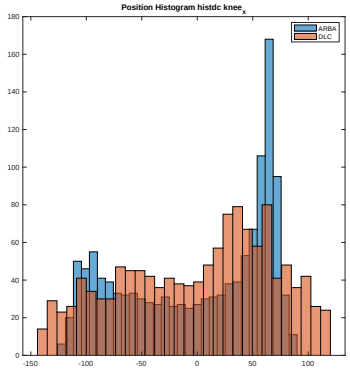


(e) Metatarso x.

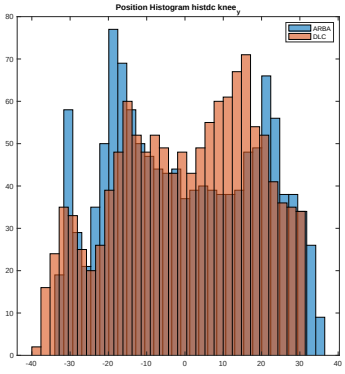


(f) Metatarso y.

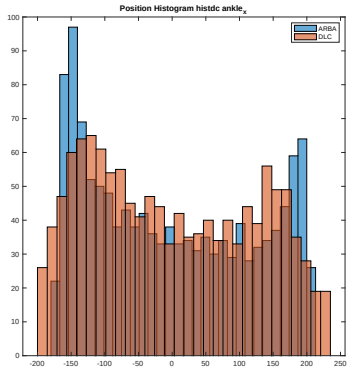
Figura 7.68: Histogramas de posición alineados, Sujeto 5R.



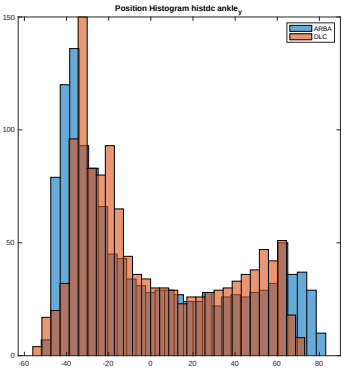
(a) Rodilla x.



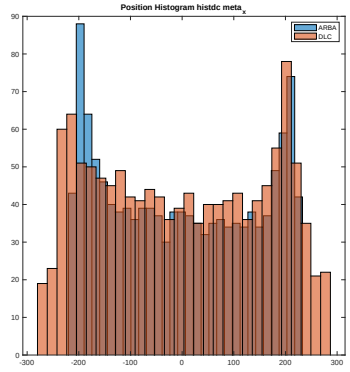
(b) Rodilla y.



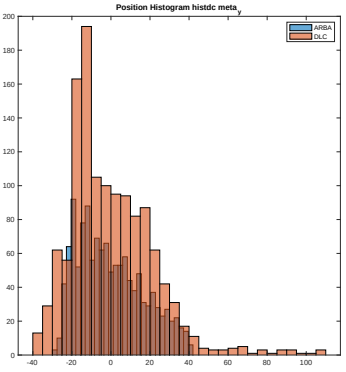
(c) Tobillo x.



(d) Tobillo y.

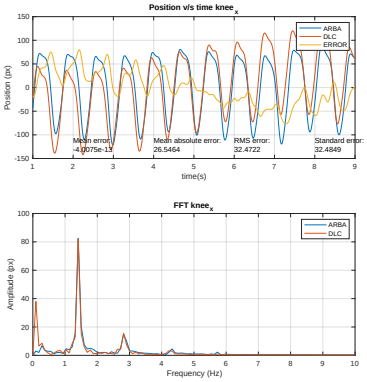


(e) Metatarso x.

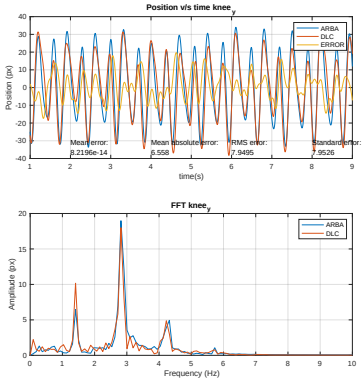


(f) Metatarso y.

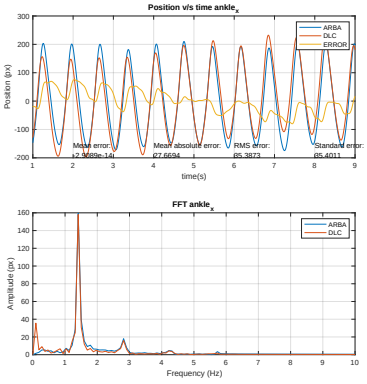
Figura 7.69: Histogramas de posición filtrados, Sujeto 5R.



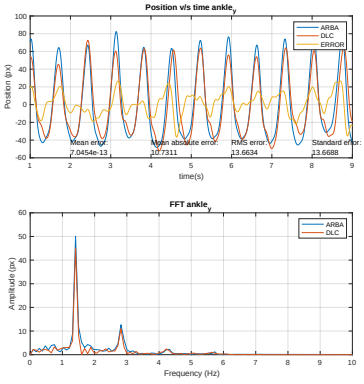
(a) Rodilla x.



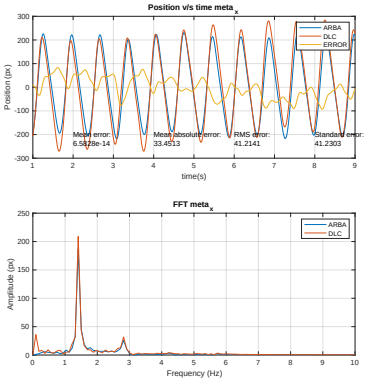
(b) Rodilla y.



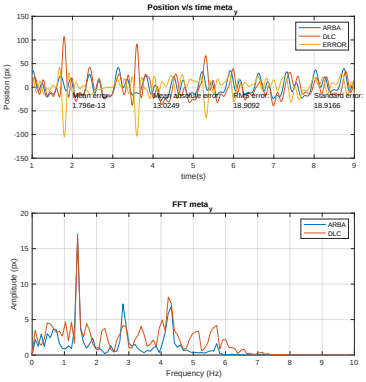
(c) Tobillo x.



(d) Tobillo y.



(e) Metatarso x.



(f) Metatarso y.

Figura 7.70: FFT de las señales, Sujeto 5R.

Tabla 7.15: Errores FFT, Sujeto 5R.

Dato	Eje	Tipo	Valor
Rodilla	x	Mean	4.0075e-13
Rodilla	x	Mean abs	26.5464
Rodilla	x	RMSE	32.4722
Rodilla	x	Standard	32.4849
Rodilla	y	Mean	8.2196e-14
Rodilla	y	Mean abs	6.558
Rodilla	y	RMSE	7.9495
Rodilla	y	Standard	7.9526
Tobillo	x	Mean	-2.9089e-14
Tobillo	x	Mean abs	27.6694
Tobillo	x	RMSE	35.3873
Tobillo	x	Standard	35.4011
Tobillo	y	Mean	7.0454e-13
Tobillo	y	Mean abs	10.7311
Tobillo	y	RMSE	13.6634
Tobillo	y	Standard	13.6688
Metatarso	x	Mean	6.5828e-14
Metatarso	x	Mean abs	33.4513
Metatarso	x	RMSE	41.2141
Metatarso	x	Standard	41.2303
Metatarso	y	Mean	1.796e-13
Metatarso	y	Mean abs	13.0249
Metatarso	y	RMSE	18.9092
Metatarso	y	Standard	18.9166

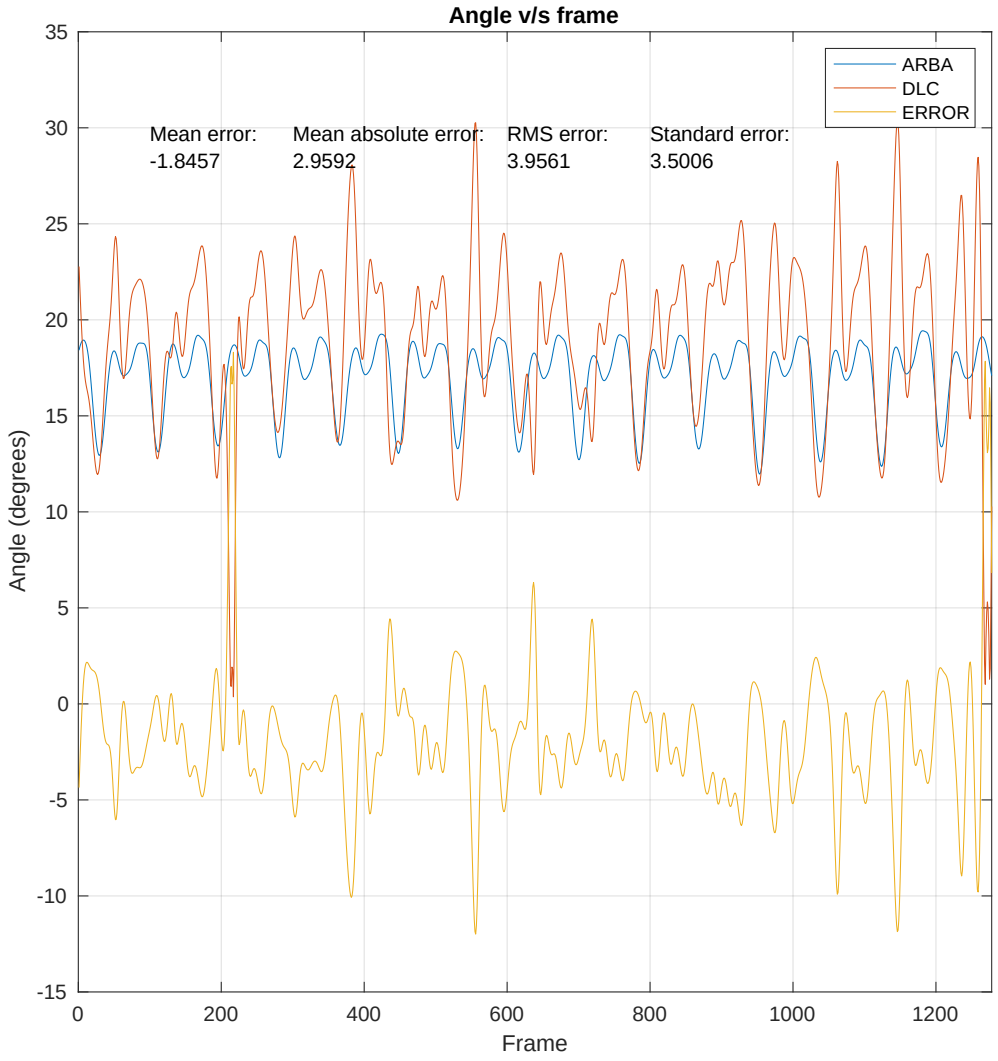
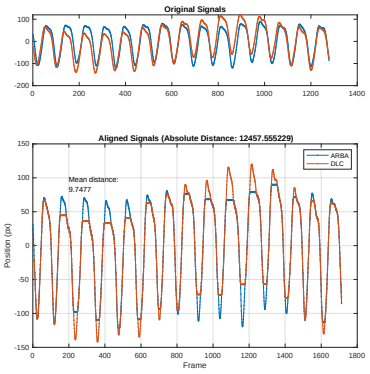


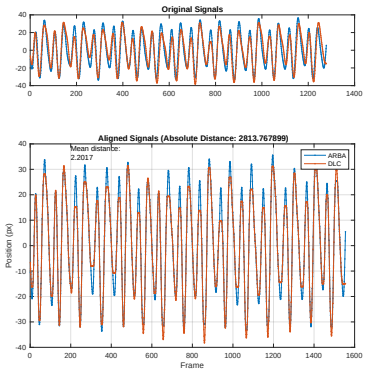
Figura 7.71: Comparación de ángulos y error medio, Sujeto 5R.

Tabla 7.16: Errores ángulo articular, Sujeto 5R.

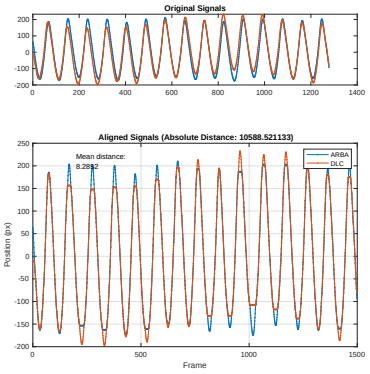
Tipo	Mean	Mean abs	RMSE	Standard
Valor	-1.8457	2.9592	3.9561	3.5006



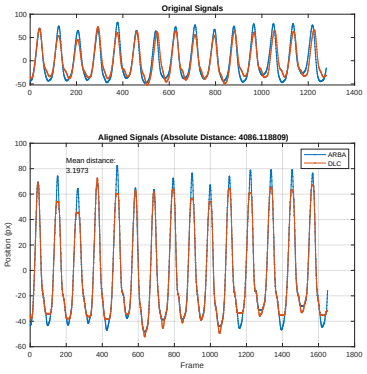
(a) Rodilla x.



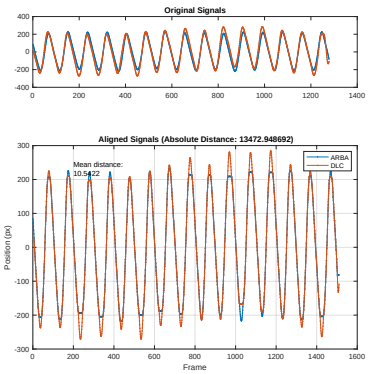
(b) Rodilla y.



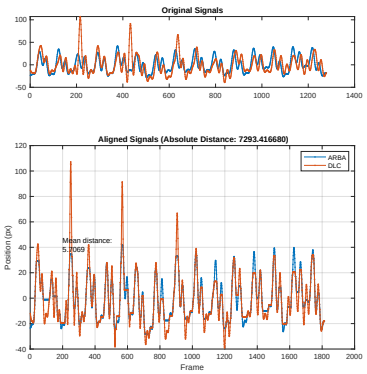
(c) Tobillo x.



(d) Tobillo y.



(e) Metatarso x.



(f) Metatarso y.

Figura 7.72: Distancia absoluta entre las señales, Sujeto 5R.

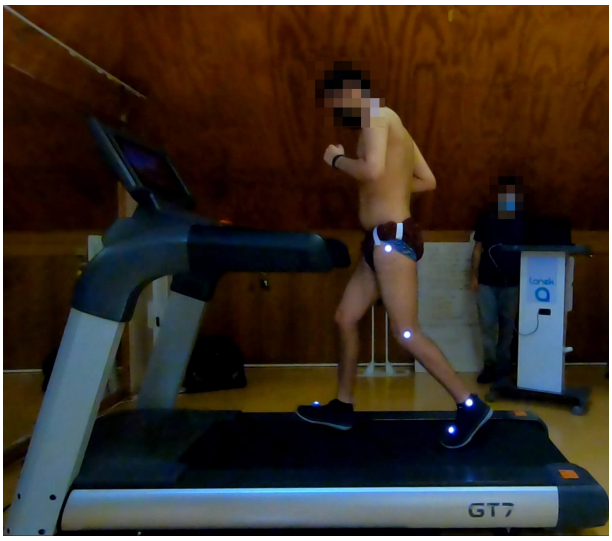
Tabla 7.17: Distancias absolutas y medias, Sujeto 5R.

Dato	Eje	Error absoluto	Error medio
Rodilla	x	12457.5552	9.7477
Rodilla	y	2813.7678	2.2017
Tobillo	x	10588.5211	8.2852
Tobillo	y	4086.1188	3.1973
Metatarso	x	13472.9486	10.5422
Metatarso	y	7293.4166	5.7069

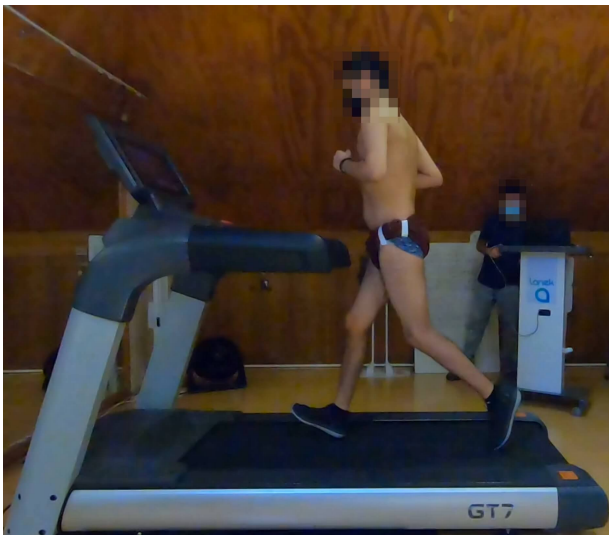
(a) ARBA. **(b)** DLC.

Figura 7.73: Videos de salida, Sujeto 5R.

7.1.4. Sujeto 6, plano sagital izquierdo

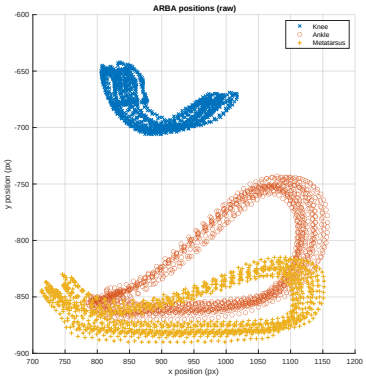


(a) ARBA.

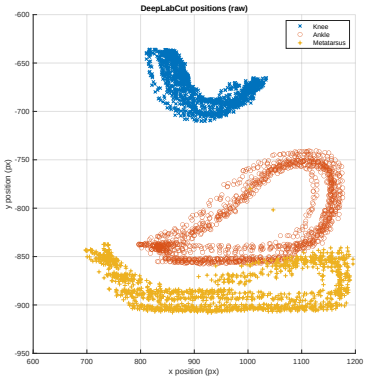


(b) DLC.

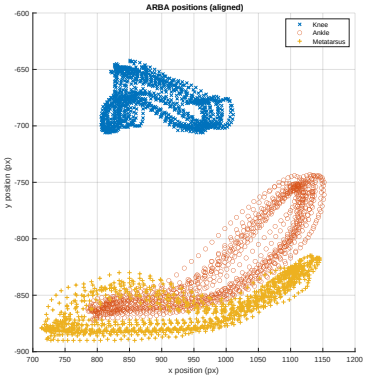
Figura 7.74: Sujeto 6L.



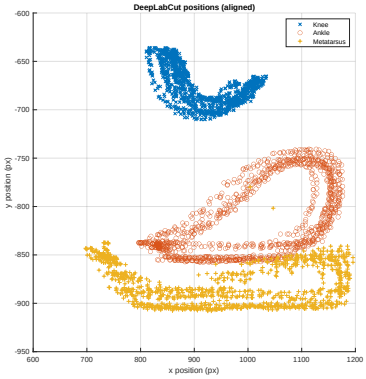
(a) ARBA, posición original.



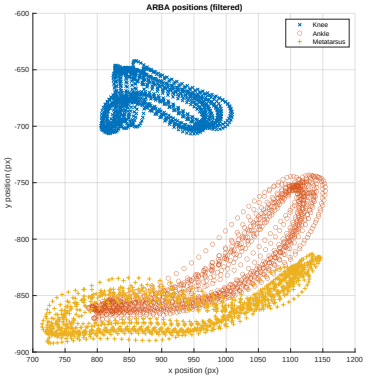
(b) DLC, posición original.



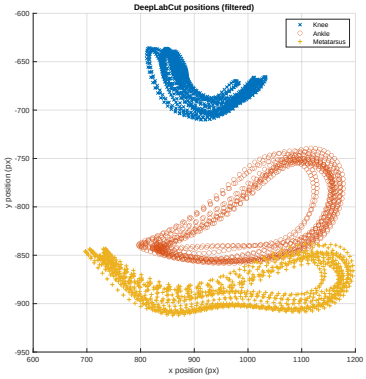
(c) ARBA, posición alineada.



(d) DLC, posición alineada.

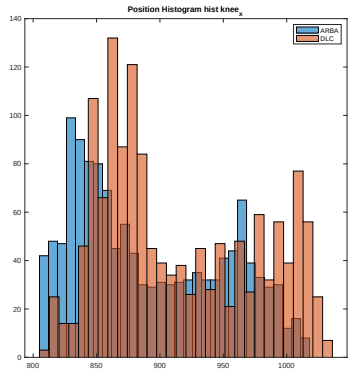


(e) ARBA, posición filtrada.

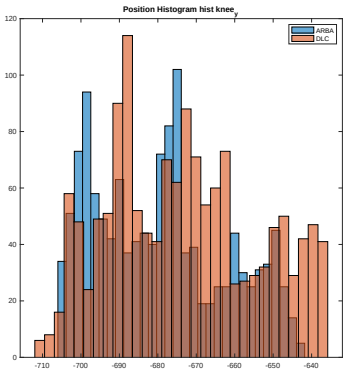


(f) DLC, posición filtrada.

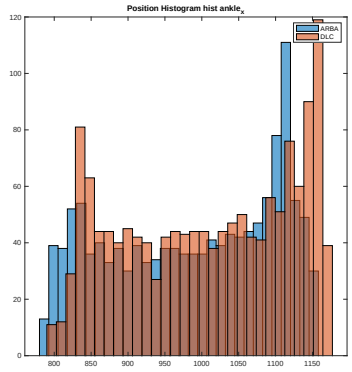
Figura 7.75: Gráficas de posición, Sujeto 6L.



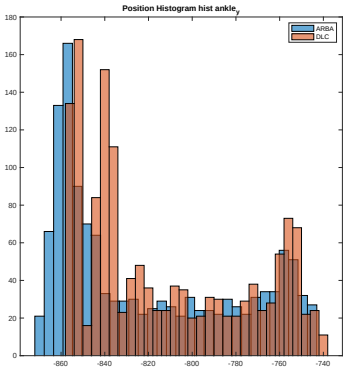
(a) Rodilla x.



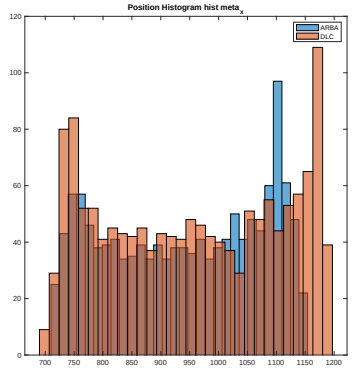
(b) Rodilla y.



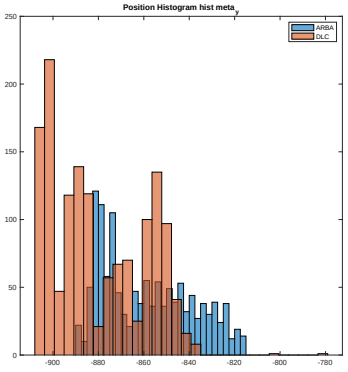
(c) Tobillo x.



(d) Tobillo y.

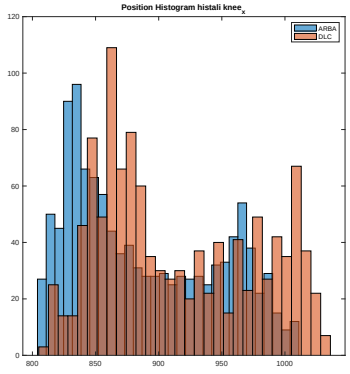


(e) Metatarso x.

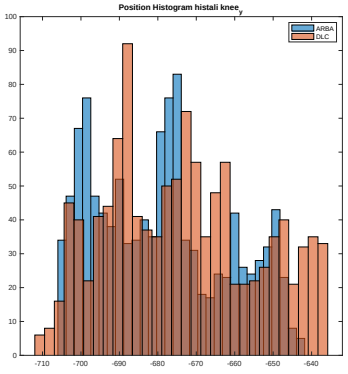


(f) Metatarso y.

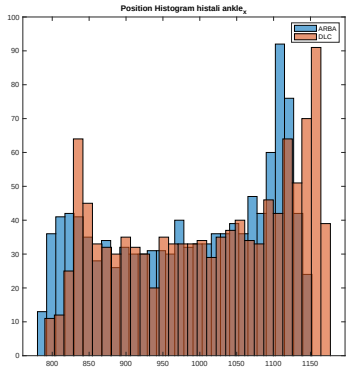
Figura 7.76: Histogramas de posición, Sujeto 6L.



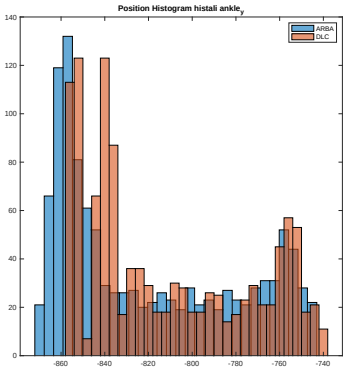
(a) Rodilla x.



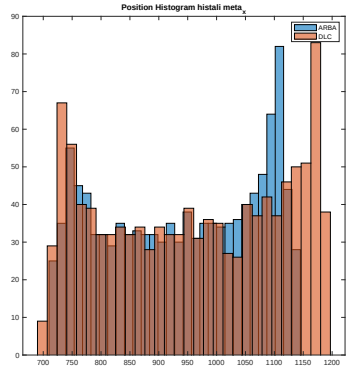
(b) Rodilla y.



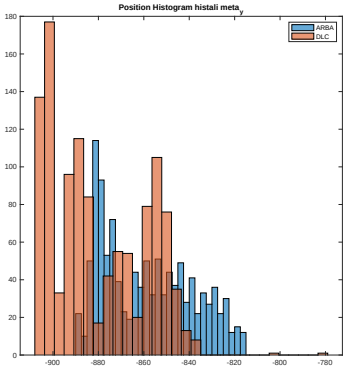
(c) Tobillo x.



(d) Tobillo y.

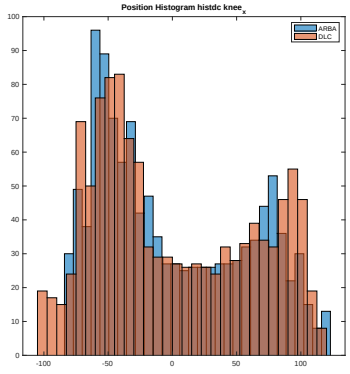


(e) Metatarso x.

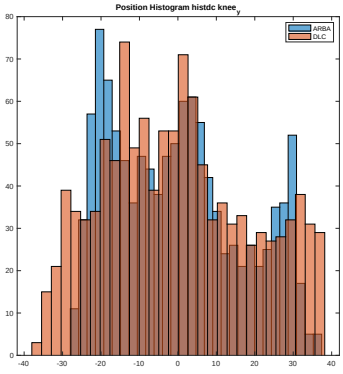


(f) Metatarso y.

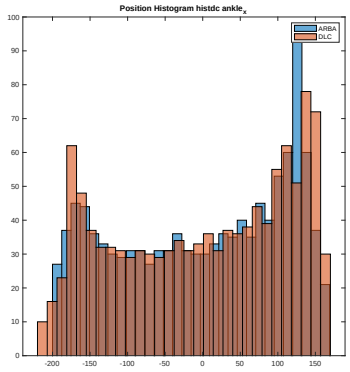
Figura 7.77: Histogramas de posición alineados, Sujeto 6L.



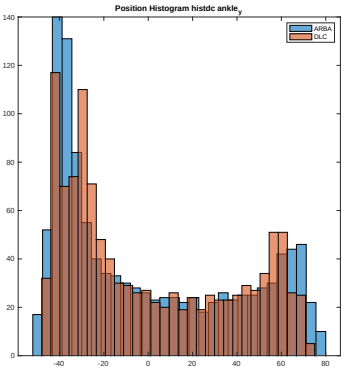
(a) Rodilla x.



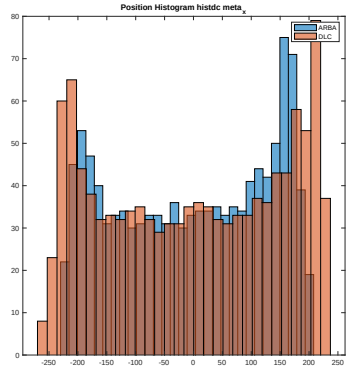
(b) Rodilla y.



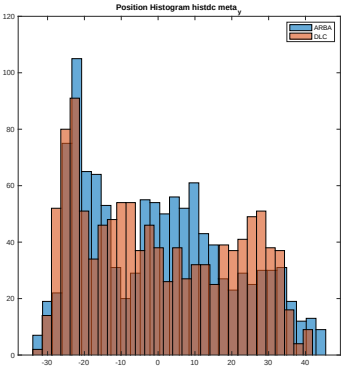
(c) Tobillo x.



(d) Tobillo y.

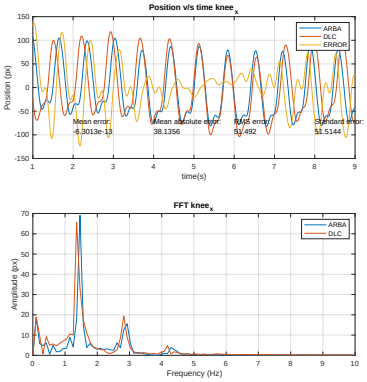


(e) Metatarso x.

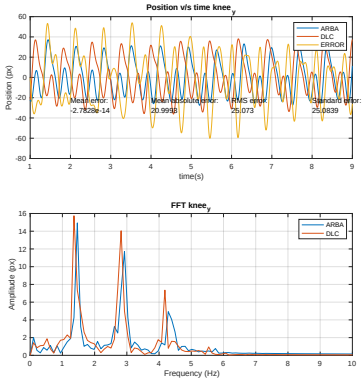


(f) Metatarso y.

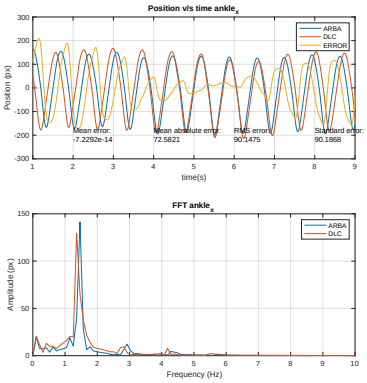
Figura 7.78: Histogramas de posición filtrados, Sujeto 6L.



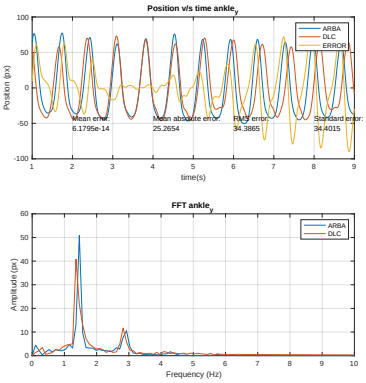
(a) Rodilla x.



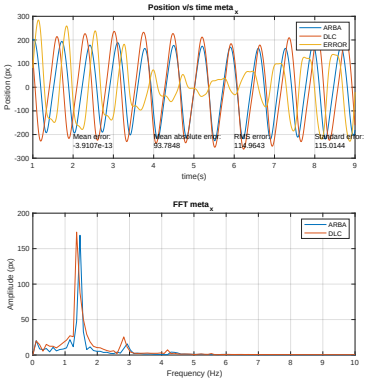
(b) Rodilla y.



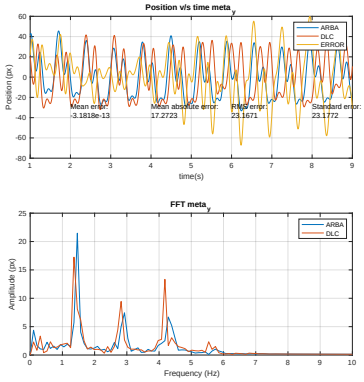
(c) Tobillo x.



(d) Tobillo y.



(e) Metatarso x.



(f) Metatarso y.

Figura 7.79: FFT de las señales, Sujeto 6L.

Tabla 7.18: Errores, Sujeto 6L.

Dato	Eje	Tipo	Valor
Rodilla	x	Mean	-6.3013e-13
Rodilla	x	Mean abs	38.1356
Rodilla	x	RMSE	51.492
Rodilla	x	Standard	51.5144
Rodilla	y	Mean	-2.7828e-14
Rodilla	y	Mean abs	20.9993
Rodilla	y	RMSE	25.073
Rodilla	y	Standard	25.0839
Tobillo	x	Mean	-7.2292e-14
Tobillo	x	Mean abs	72.5821
Tobillo	x	RMSE	90.1475
Tobillo	x	Standard	90.1868
Tobillo	y	Mean	6.1795e-14
Tobillo	y	Mean abs	25.2654
Tobillo	y	RMSE	34.3865
Tobillo	y	Standard	34.4015
Metatarso	x	Mean	-3.9107e-13
Metatarso	x	Mean abs	93.7848
Metatarso	x	RMSE	114.9643
Metatarso	x	Standard	115.0144
Metatarso	y	Mean	-3.1818e-13
Metatarso	y	Mean abs	17.2723
Metatarso	y	RMSE	23.1671
Metatarso	y	Standard	23.1772

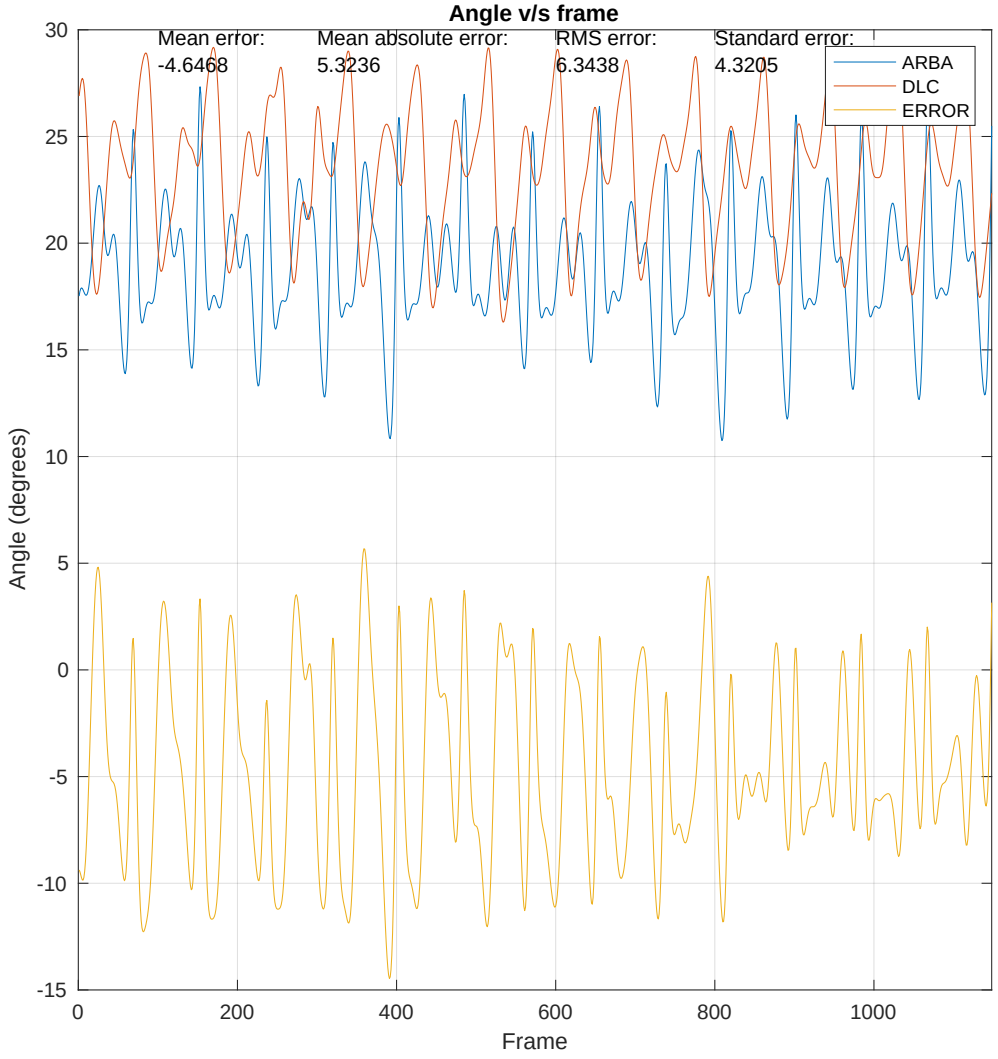
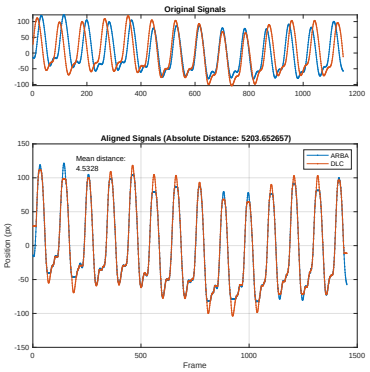


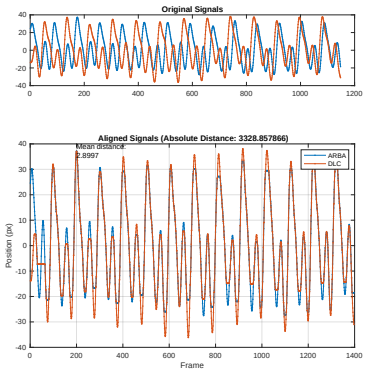
Figura 7.80: Comparación de ángulos y error medio, Sujeto 6L.

Tabla 7.19: Errores ángulo articular, Sujeto 6L.

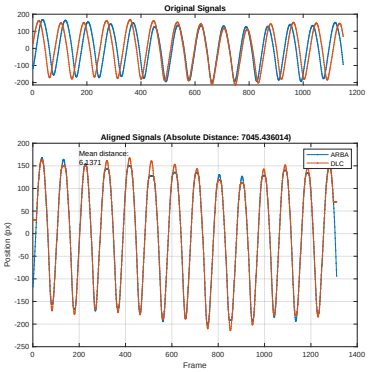
Tipo	Mean	Mean abs	RMSE	Standard
Valor	-4.6468	5.3236	6.3438	4.3205



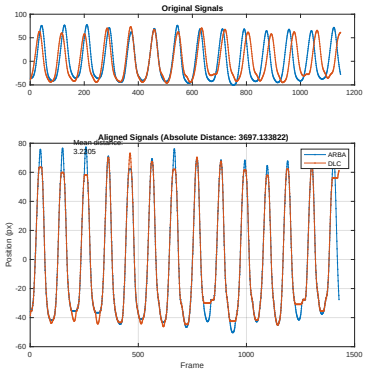
(a) Rodilla x.



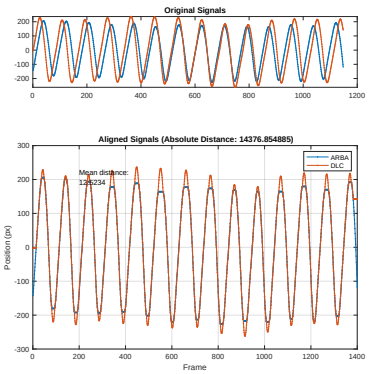
(b) Rodilla y.



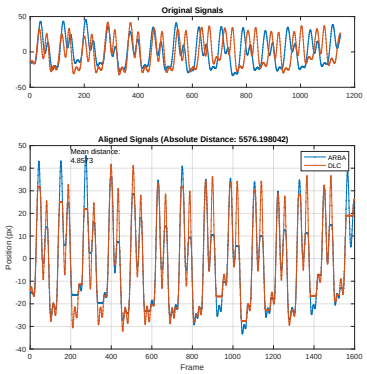
(c) Tobillo x.



(d) Tobillo y.



(e) Metatarso x.

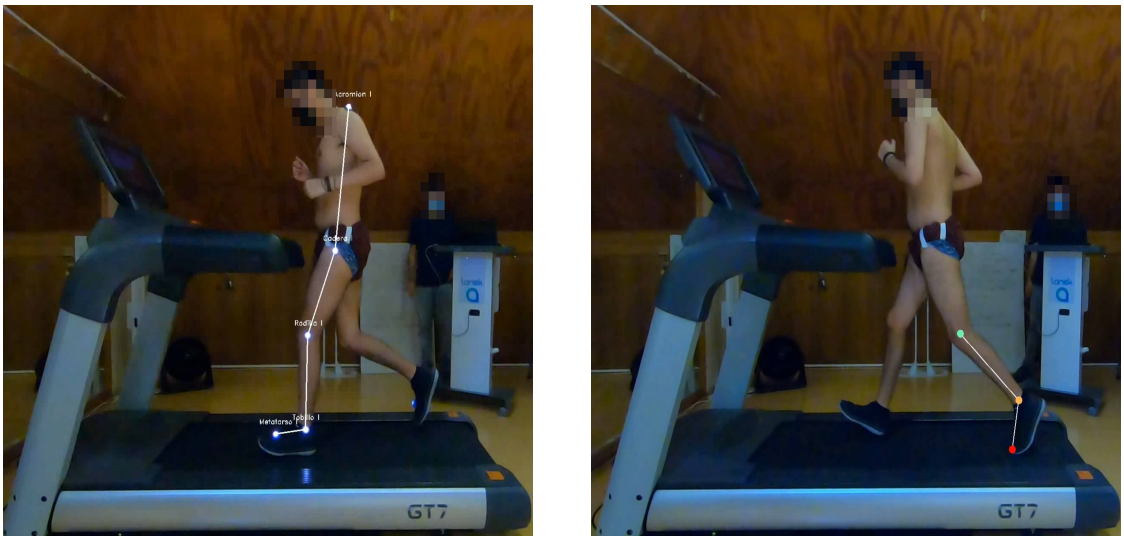


(f) Metatarso y.

Figura 7.81: Distancia absoluta entre las señales, Sujeto 6L.

Tabla 7.20: Distancias absolutas y medias, Sujeto 6L.

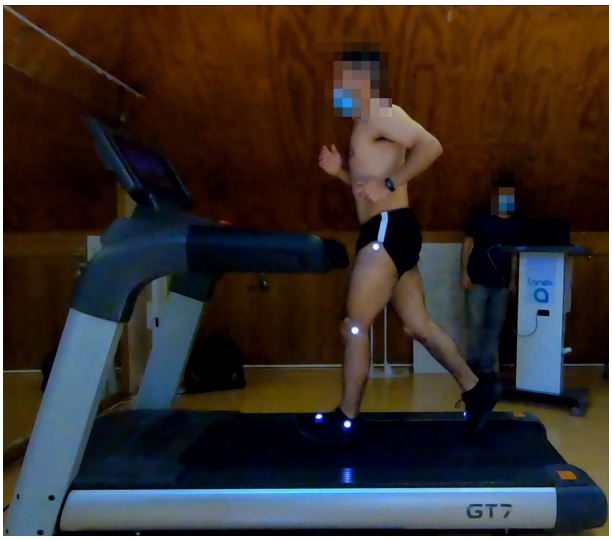
Dato	Eje	Error absoluto	Error medio
Rodilla	x	5203.6526	4.5328
Rodilla	y	3328.8578	2.8997
Tobillo	x	7045.4360	6.1371
Tobillo	y	3697.1338	3.2205
Metatarso	x	14376.8548	12.5234
Metatarso	y	5576.1980	4.8573



(a) ARBA. **(b)** DLC.

Figura 7.82: Videos de salida, Sujeto 6L.

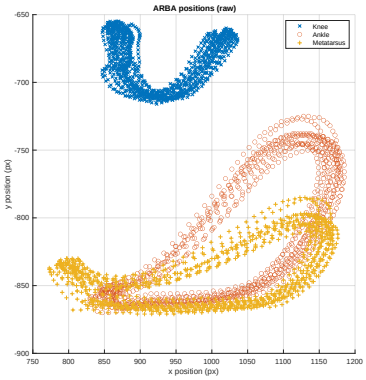
7.1.5. Sujeto 7, plano sagital izquierdo



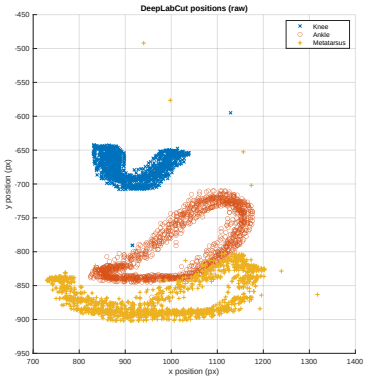
(a) ARBA.

(b) DLC.

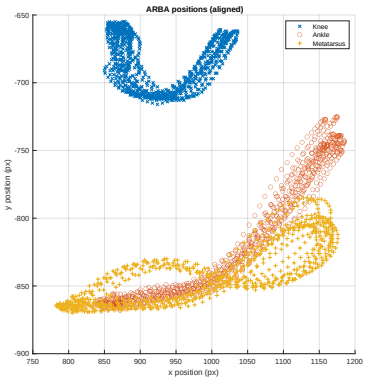
Figura 7.83: Sujeto 7L.



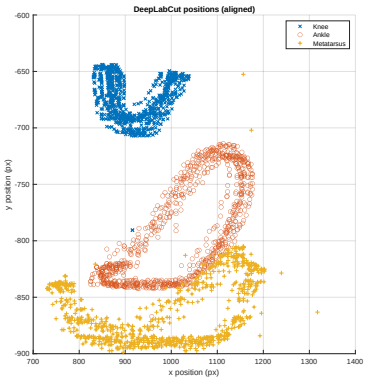
(a) ARBA, posición original.



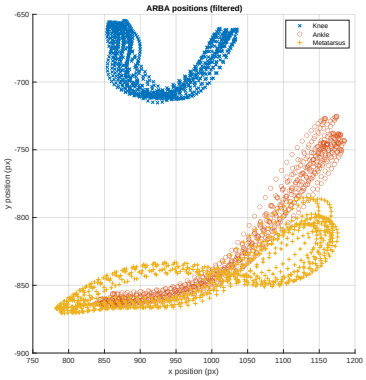
(b) DLC, posición original.



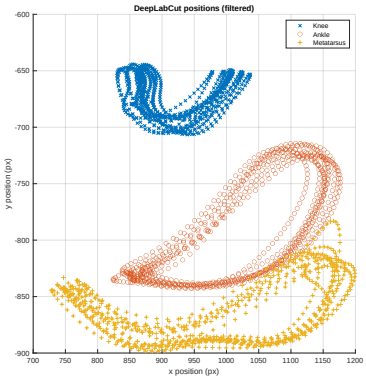
(c) ARBA, posición alineada.



(d) DLC, posición alineada.

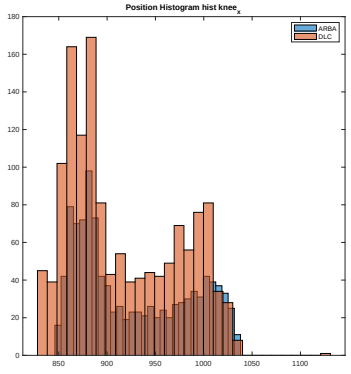


(e) ARBA, posición filtrada.

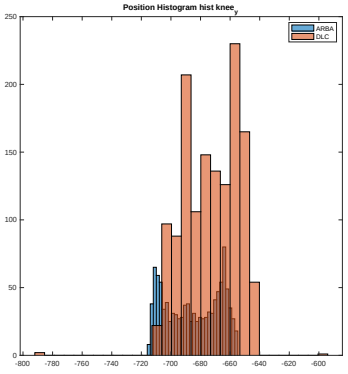


(f) DLC, posición filtrada.

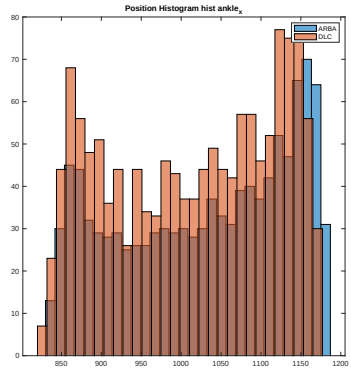
Figura 7.84: Gráficas de posición, Sujeto 7L.



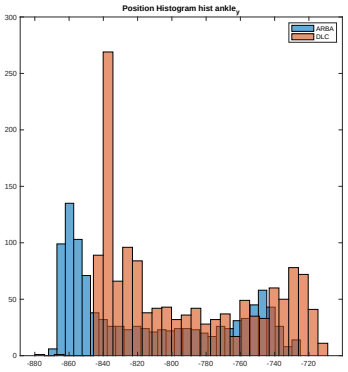
(a) Rodilla x.



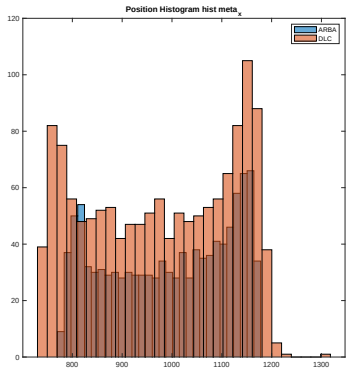
(b) Rodilla y.



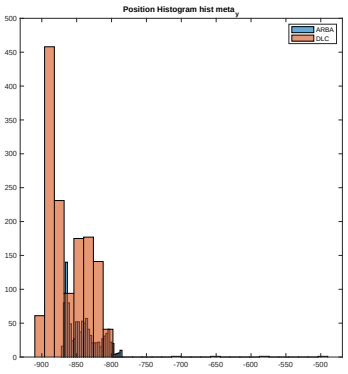
(c) Tobillo x.



(d) Tobillo y.

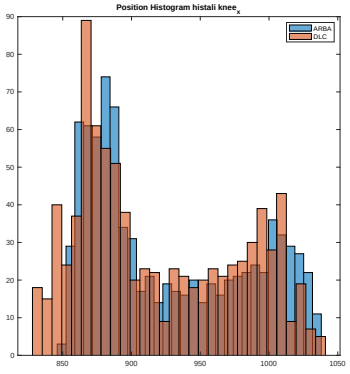


(e) Metatarso x.

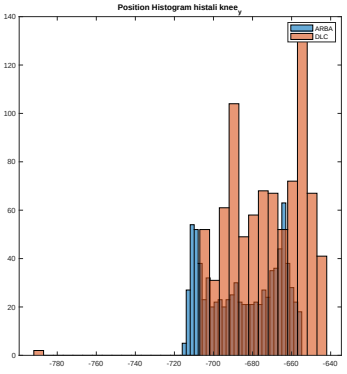


(f) Metatarso y.

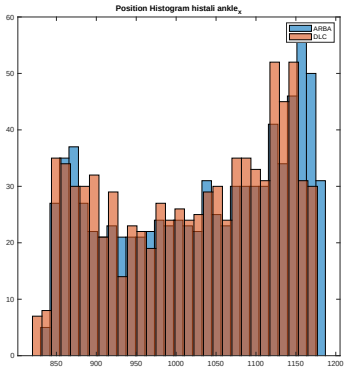
Figura 7.85: Histogramas de posición, Sujeto 7L.



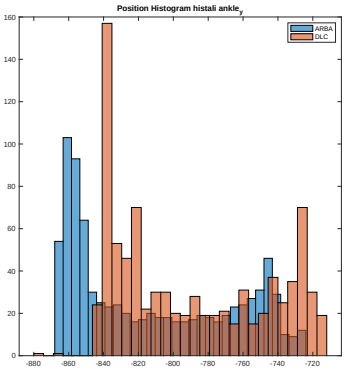
(a) Rodilla x.



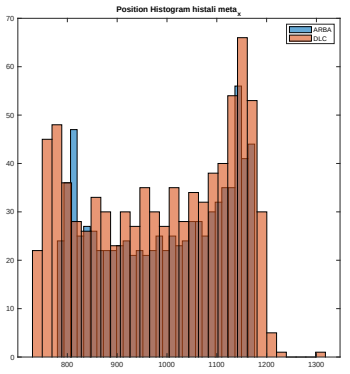
(b) Rodilla y.



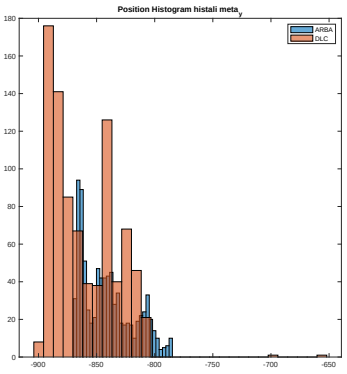
(c) Tobillo x.



(d) Tobillo y.

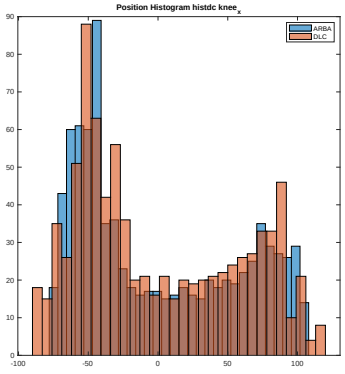


(e) Metatarso x.

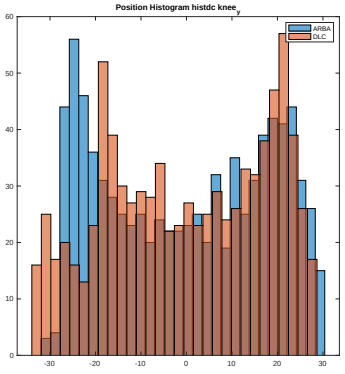


(f) Metatarso y.

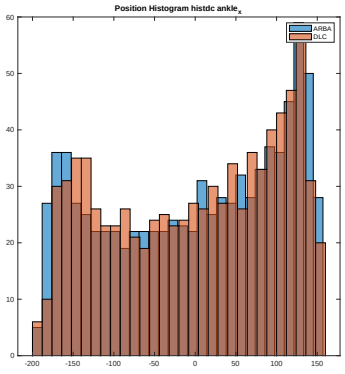
Figura 7.86: Histogramas de posición alineados, Sujeto 7L.



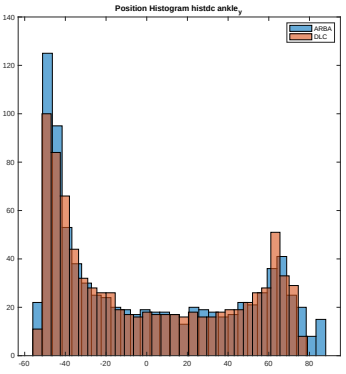
(a) Rodilla x.



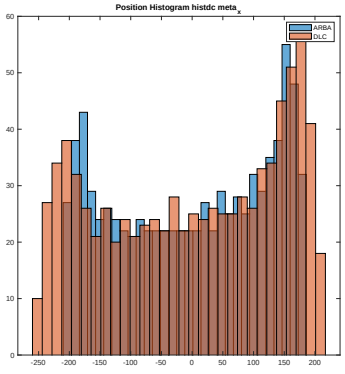
(b) Rodilla y.



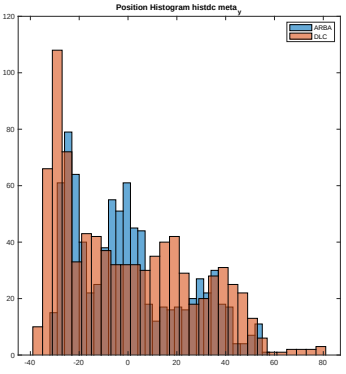
(c) Tobillo x.



(d) Tobillo y.

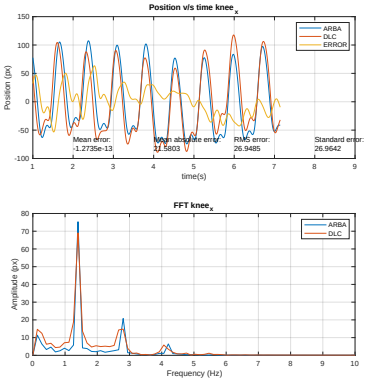


(e) Metatarso x.

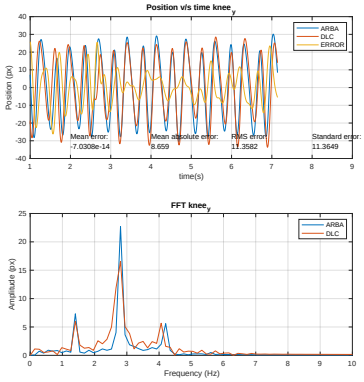


(f) Metatarso y.

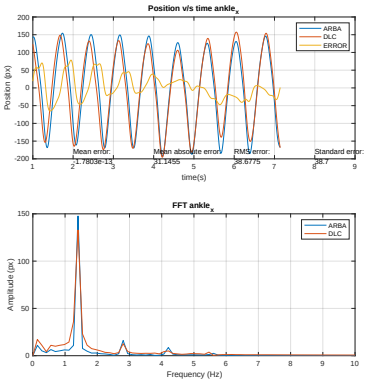
Figura 7.87: Histogramas de posición filtrados, Sujeto 7L.



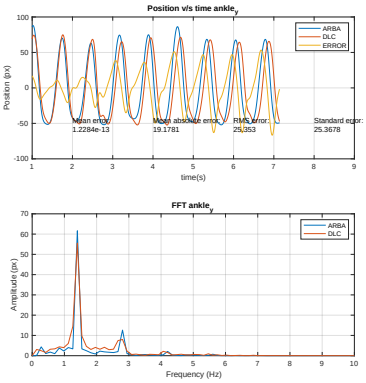
(a) Rodilla x.



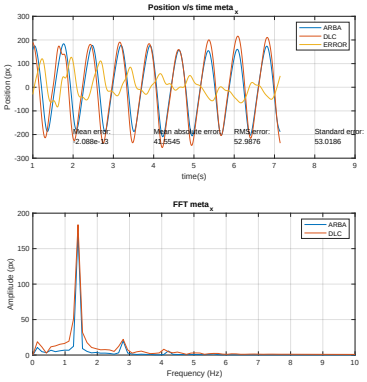
(b) Rodilla y.



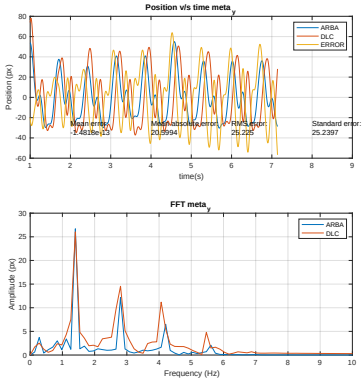
(c) Tobillo x.



(d) Tobillo y.



(e) Metatarso x.



(f) Metatarso y.

Figura 7.88: FFT de las señales, Sujeto 7L.

Tabla 7.21: Errores, Sujeto 7L.

Dato	Eje	Tipo	Valor
Rodilla	x	Mean	-1.2735e-13
Rodilla	x	Mean abs	21.5803
Rodilla	x	RMSE	26.9485
Rodilla	x	Standard	26.9642
Rodilla	y	Mean	-7.0308e-14
Rodilla	y	Mean abs	8.659
Rodilla	y	RMSE	11.3582
Rodilla	y	Standard	11.3649
Tobillo	x	Mean	1.7803e-13
Tobillo	x	Mean abs	31.1455
Tobillo	x	RMSE	38.6775
Tobillo	x	Standard	38.7
Tobillo	y	Mean	1.2284e-13
Tobillo	y	Mean abs	19.1781
Tobillo	y	RMSE	25.353
Tobillo	y	Standard	25.3678
Metatarso	x	Mean	-2.088e-13
Metatarso	x	Mean abs	41.5545
Metatarso	x	RMSE	52.9876
Metatarso	x	Standard	53.0186
Metatarso	y	Mean	-1.4818e-13
Metatarso	y	Mean abs	20.5994
Metatarso	y	RMSE	25.225
Metatarso	y	Standard	25.2397

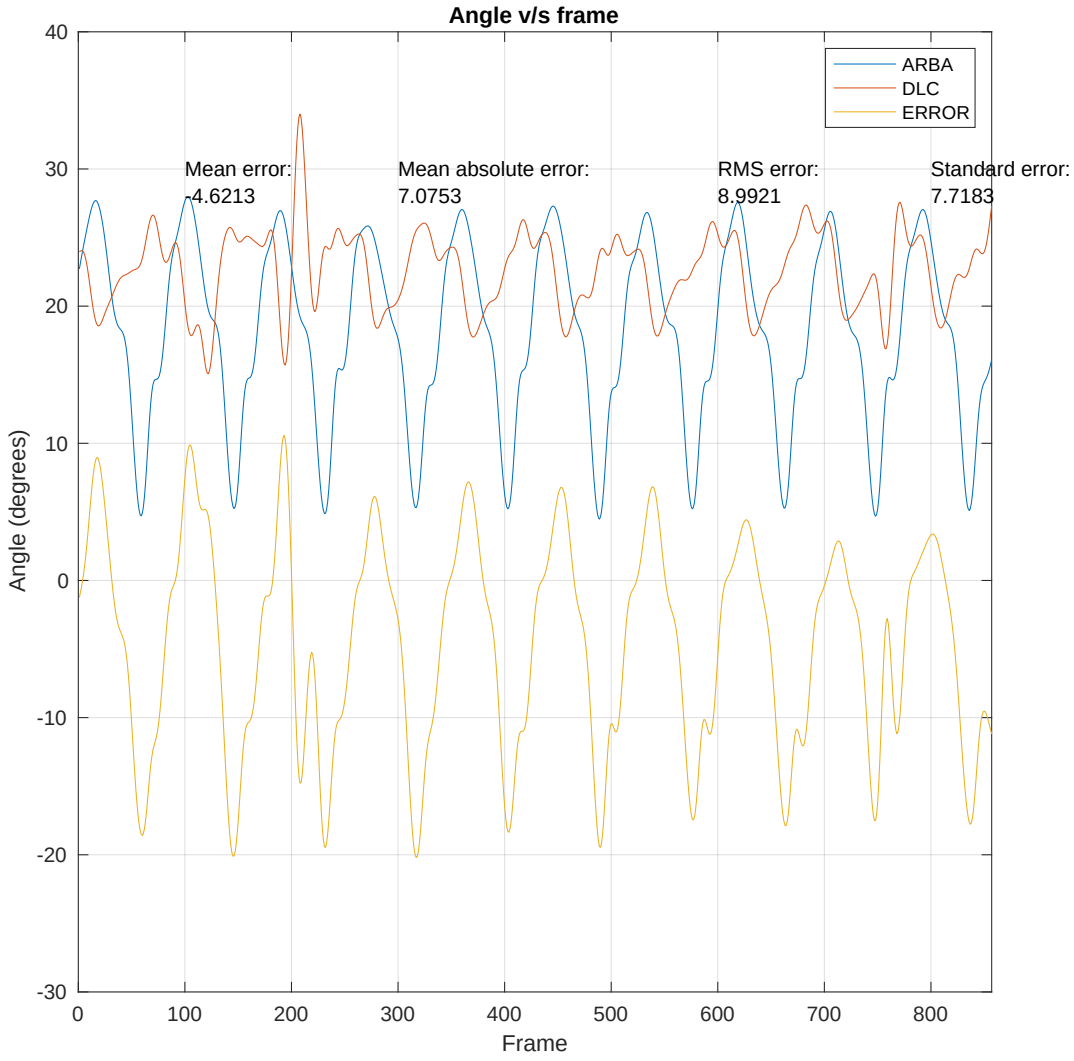
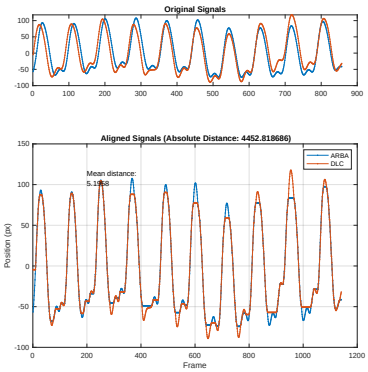


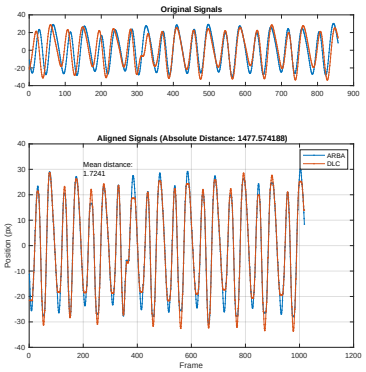
Figura 7.89: Comparación de ángulos y error medio, Sujeto 7L.

Tabla 7.22: Errores ángulo articular, Sujeto 7L.

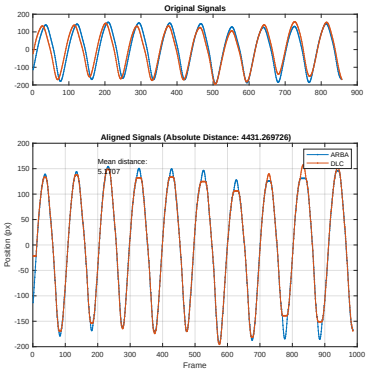
Tipo	Mean	Mean abs	RMSE	Standard
Valor	-4.6213	7.0753	8.9921	7.7183



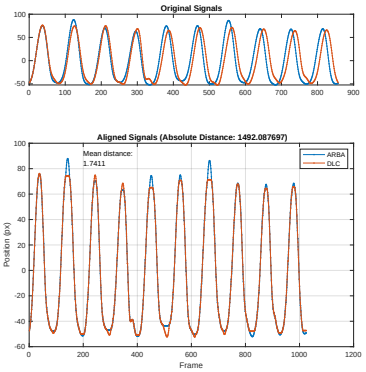
(a) Rodilla x.



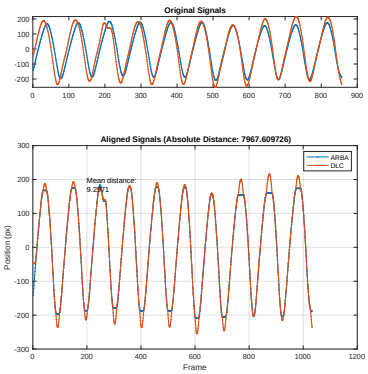
(b) Rodilla y.



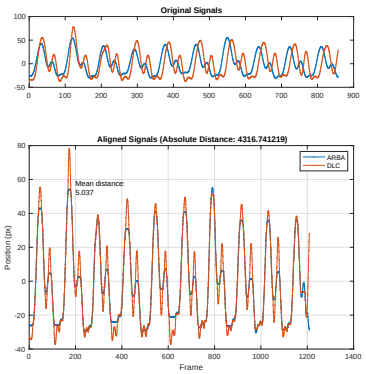
(c) Tobillo x.



(d) Tobillo y.



(e) Metatarso x.

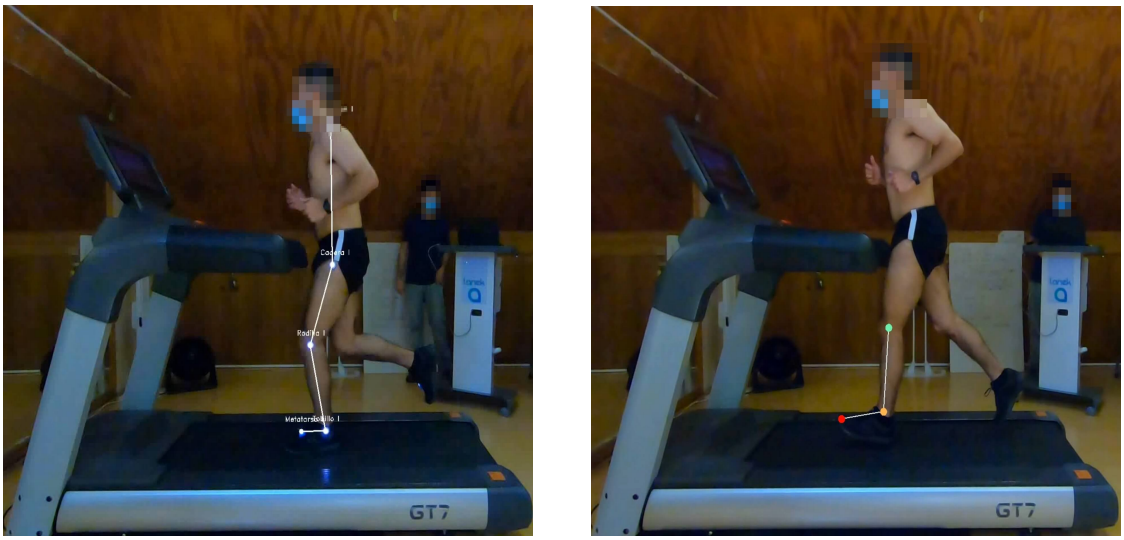


(f) Metatarso y.

Figura 7.90: Distancia absoluta entre las señales, Sujeto 7L.

Tabla 7.23: Distancias absolutas y medias, Sujeto 7L.

Dato	Eje	Error absoluto	Error medio
Rodilla	x	4452.8186	5.1958
Rodilla	y	1477.5741	1.7241
Tobillo	x	4431.2697	5.1707
Tobillo	y	1492.0876	1.7411
Metatarso	x	7967.6097	9.2971
Metatarso	y	4316.7412	5.037



(a) ARBA. **(b)** DLC.

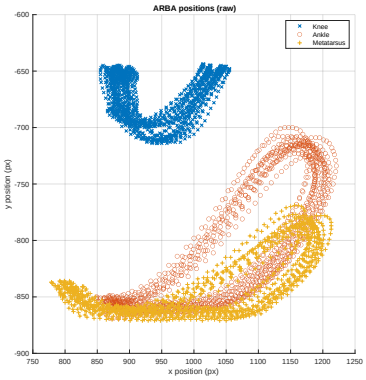
Figura 7.91: Videos de salida, Sujeto 7L.

7.1.6. Sujeto 8, plano sagital izquierdo

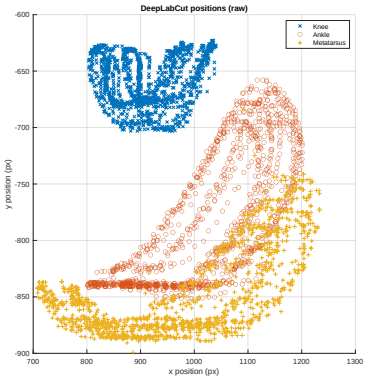
(a) ARBA.

(b) DLC.

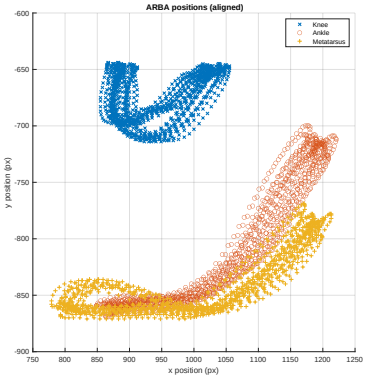
Figura 7.92: Sujeto 8L.



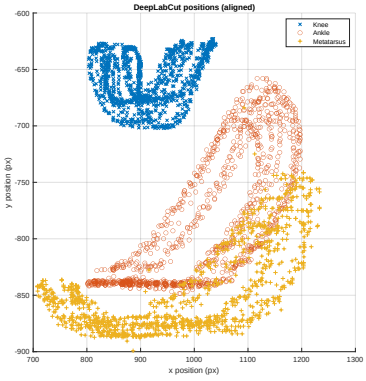
(a) ARBA, posición original.



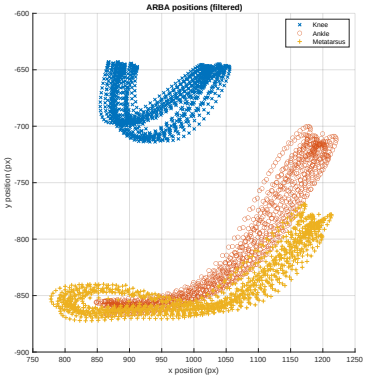
(b) DLC, posición original.



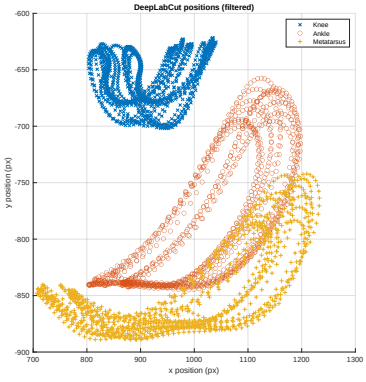
(c) ARBA, posición alineada.



(d) DLC, posición alineada.

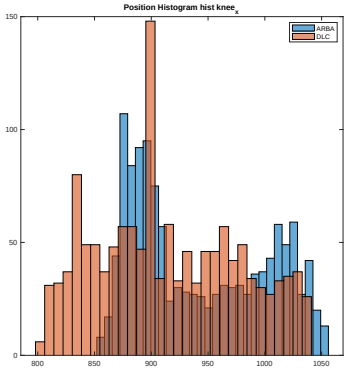


(e) ARBA, posición filtrada.

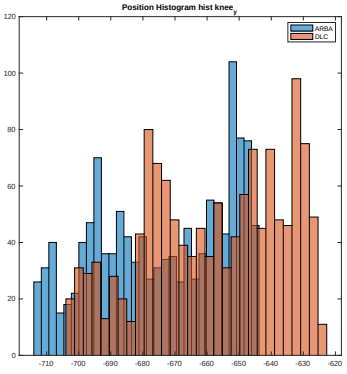


(f) DLC, posición filtrada.

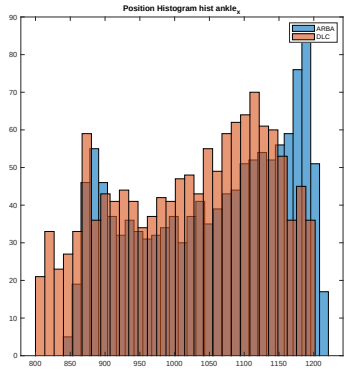
Figura 7.93: Gráficas de posición, Sujeto 8L.



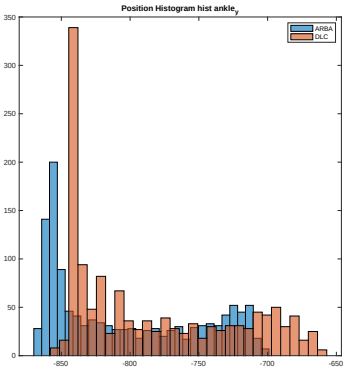
(a) Rodilla x.



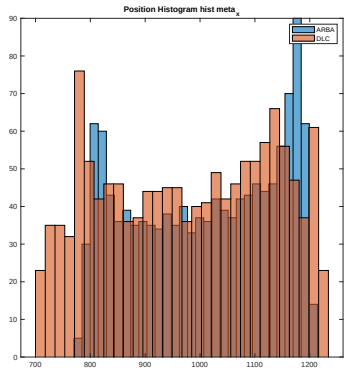
(b) Rodilla y.



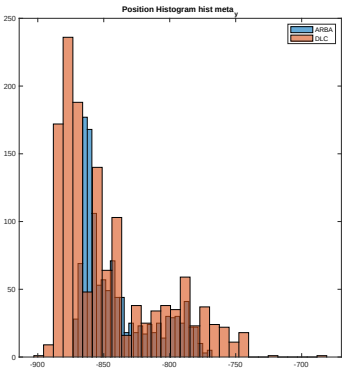
(c) Tobillo x.



(d) Tobillo y.

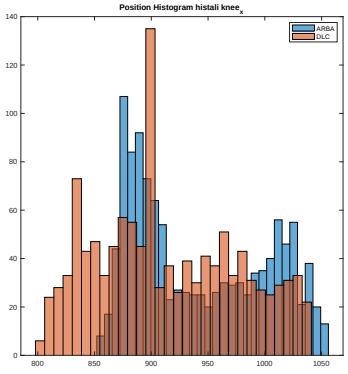


(e) Metatarso x.

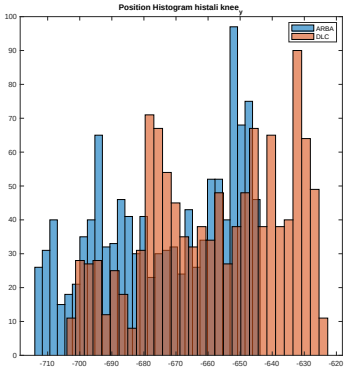


(f) Metatarso y.

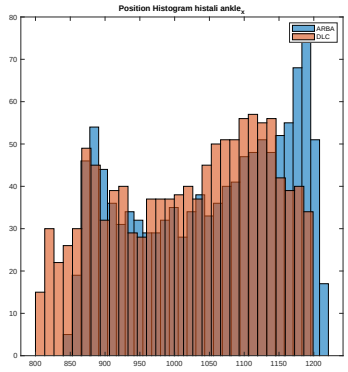
Figura 7.94: Histogramas de posición, Sujeto 8L.



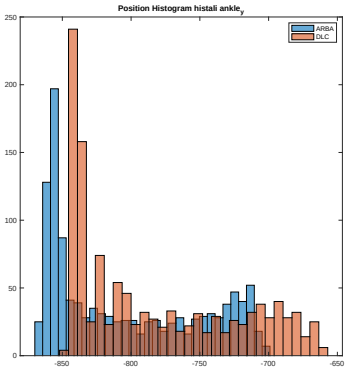
(a) Rodilla x.



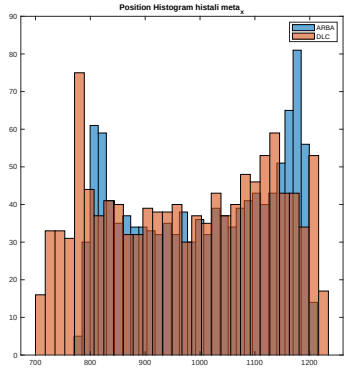
(b) Rodilla y.



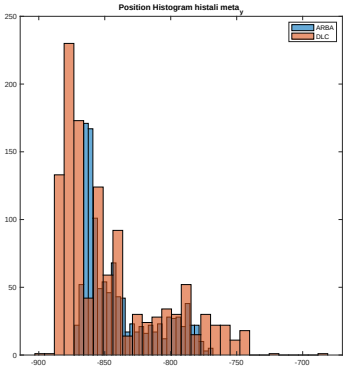
(c) Tobillo x.



(d) Tobillo y.

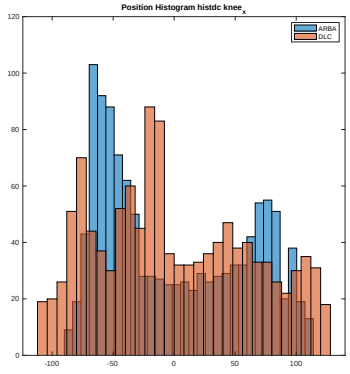


(e) Metatarso x.

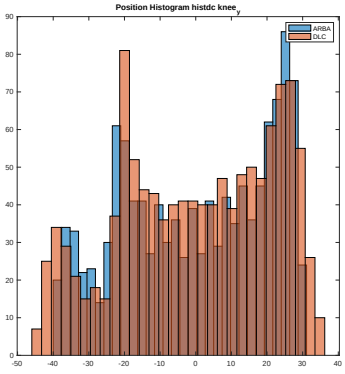


(f) Metatarso y.

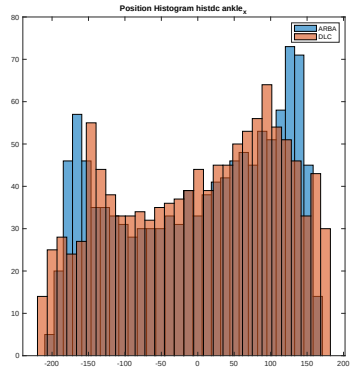
Figura 7.95: Histogramas de posición alineados, Sujeto 8L.



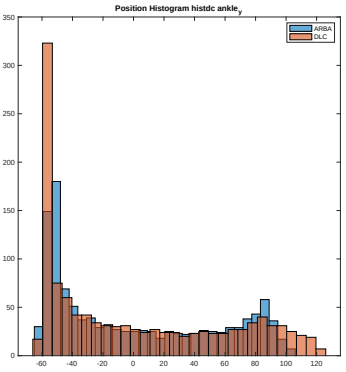
(a) Rodilla x.



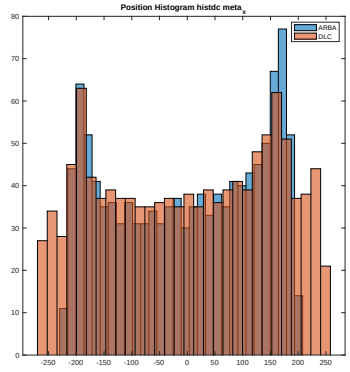
(b) Rodilla y.



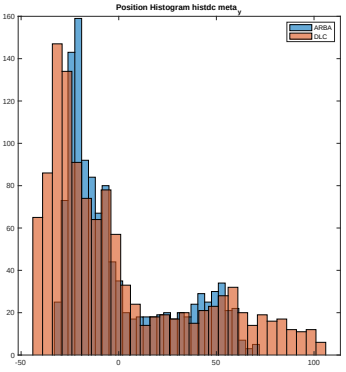
(c) Tobillo x.



(d) Tobillo y.

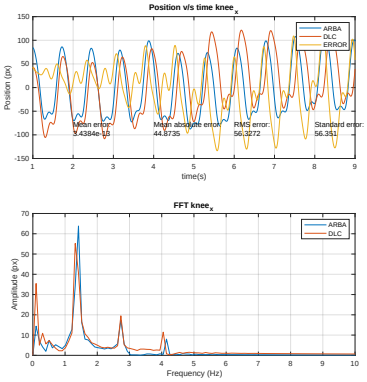


(e) Metatarso x.

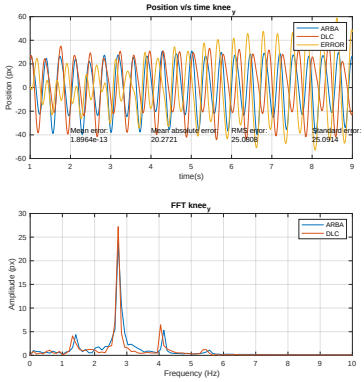


(f) Metatarso y.

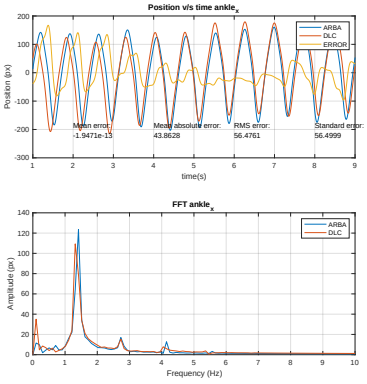
Figura 7.96: Histogramas de posición filtrados, Sujeto 8L.



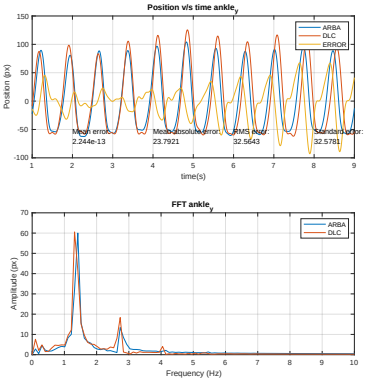
(a) Rodilla x.



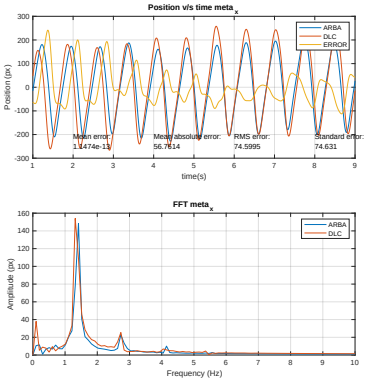
(b) Rodilla y.



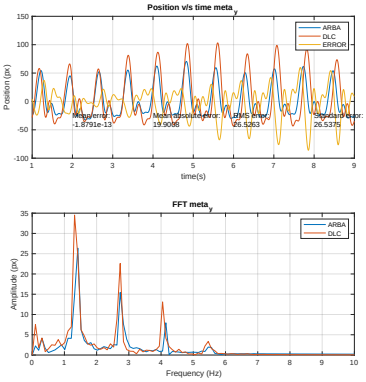
(c) Tobillo x.



(d) Tobillo y.



(e) Metatarso x.



(f) Metatarso y.

Figura 7.97: FFT de las señales, Sujeto 8L.

Tabla 7.24: Errores, Sujeto 8L.

Dato	Eje	Tipo	Valor
Rodilla	x	Mean	3.4384e-13
Rodilla	x	Mean abs	44.8735
Rodilla	x	RMSE	56.3272
Rodilla	x	Standard	56.351
Rodilla	y	Mean	1.8964e-13
Rodilla	y	Mean abs	20.2721
Rodilla	y	RMSE	25.0808
Rodilla	y	Standard	25.0914
Tobillo	x	Mean	-1.9471e-13
Tobillo	x	Mean abs	43.8628
Tobillo	x	RMSE	56.4761
Tobillo	x	Standard	56.4999
Tobillo	y	Mean	2.244e-13
Tobillo	y	Mean abs	23.7921
Tobillo	y	RMSE	32.5643
Tobillo	y	Standard	32.5781
Metatarso	x	Mean	1.1474e-13
Metatarso	x	Mean abs	56.7614
Metatarso	x	RMSE	74.5995
Metatarso	x	Standard	74.631
Metatarso	y	Mean	-1.8791e-13
Metatarso	y	Mean abs	19.9008
Metatarso	y	RMSE	26.5263
Metatarso	y	Standard	26.5375
Ángulo	-	Mean	-4.6268
Ángulo	-	Mean abs	5.9601
Ángulo	-	RMSE	7.0165
Ángulo	-	Standard	5.2771

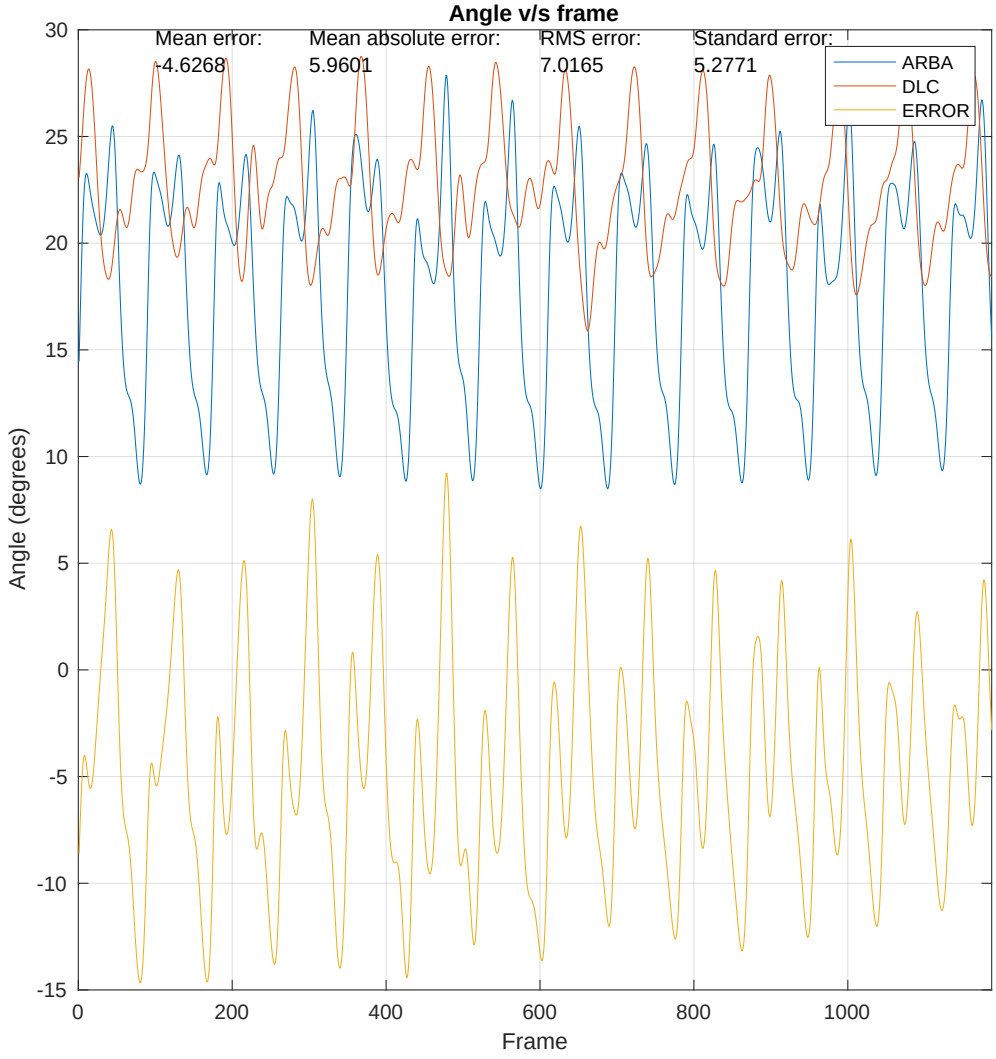
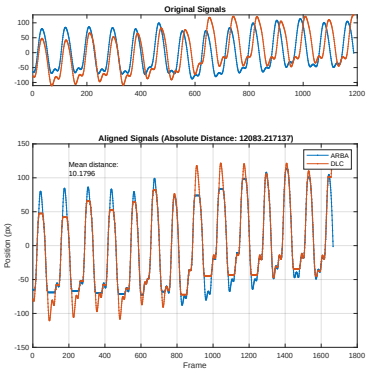


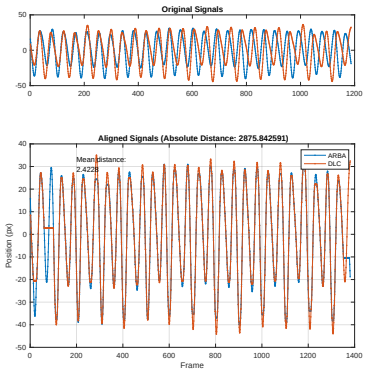
Figura 7.98: Comparación de ángulos y error medio, Sujeto 8L.

Tabla 7.25: Errores ángulo articular, Sujeto 8L.

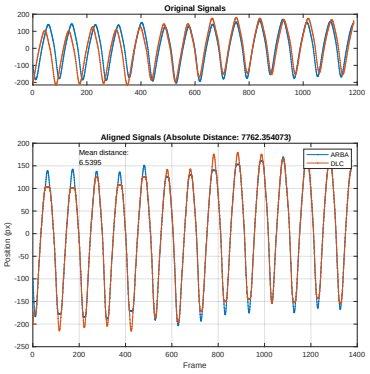
Tipo	Mean	Mean abs	RMSE	Standard
Valor	-4.6268	5.9601	7.0165	5.2771



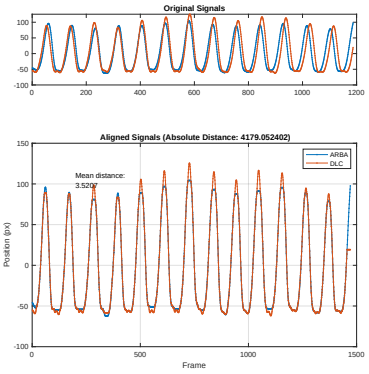
(a) Rodilla x.



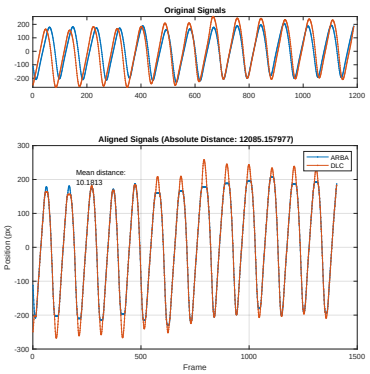
(b) Rodilla y.



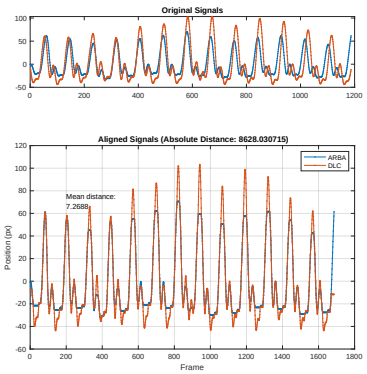
(c) Tobillo x.



(d) Tobillo y.



(e) Metatarso x.

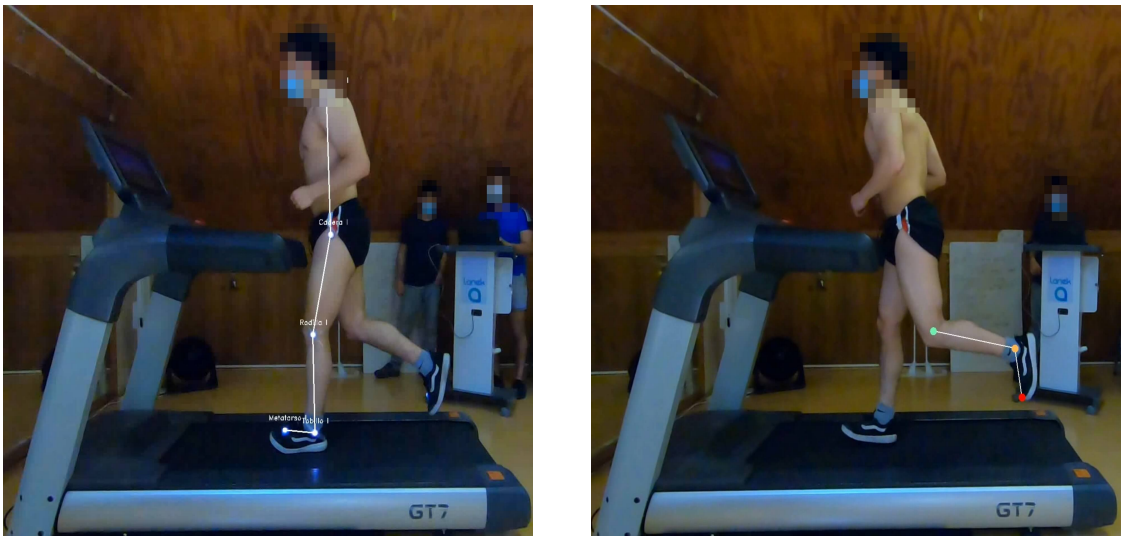


(f) Metatarso y.

Figura 7.99: Distancia absoluta entre las señales, Sujeto 8L.

Tabla 7.26: Distancias absolutas y medias, Sujeto 8L.

Dato	Eje	Error absoluto	Error medio
Rodilla	x	12083.2171	10.1796
Rodilla	y	2875.8425	2.4228
Tobillo	x	7762.3540	6.5395
Tobillo	y	4179.0524	3.5207
Metatarso	x	12085.1579	10.1813
Metatarso	y	8628.0307	7.2688



(a) ARBA. **(b)** DLC.

Figura 7.100: Videos de salida, Sujeto 8L.

7.2. Códigos

Para el procesamiento del sistema *DLC* se desarrollaron los siguientes códigos en *MATLAB*:

Código 7.1 : Importación de datos

```
%% Data import

% Clears memory, closes all figures.
clc
clear
close all

% Selects subject to evaluate.
sujeto = "p5r";
str_arba = ['arba/',sujeto,'_arba_raw.csv'];
str_dlc = ['dlc/',sujeto,'
    _dlcDLC_dlcrnetms5_arba_markerless_maNov25shuffle5_1030000_el_filtered.csv'];

% ARBA data import
M_arba = readmatrix(join(str_arba,''));
t_arba = M_arba(:,1);

% "spiline" is used to interpolate missing values.
x_arba_knee = fillmissing(M_arba(:,10),"spline");
y_arba_knee = fillmissing(-M_arba(:,11),"spline");
x_arba_ankle = fillmissing(M_arba(:,12),"spline");
y_arba_ankle = fillmissing(-M_arba(:,13),"spline");
x_arba_meta = fillmissing(M_arba(:,14),"spline");
y_arba_meta = fillmissing(-M_arba(:,15),"spline");

% DLC data import
M_dlc = readmatrix(join(str_dlc,''));
t_dlc = M_dlc(:,1);

% "spiline" is used to interpolate missing values.
x_dlc_knee = fillmissing(M_dlc(:,8),"spline");
y_dlc_knee = fillmissing(-M_dlc(:,9),"spline");
x_dlc_ankle = fillmissing(M_dlc(:,11),"spline");
y_dlc_ankle = fillmissing(-M_dlc(:,12),"spline");
x_dlc_meta = fillmissing(M_dlc(:,14),"spline");
y_dlc_meta = fillmissing(-M_dlc(:,15),"spline");

%% Data processing
```

```
% Data fixing, values missing from start or end can't be interpolated.
start= 100;
stop=1300;
x_dlc_knee = x_dlc_knee(start:stop);
y_dlc_knee = y_dlc_knee(start:stop);
x_dlc_ankle = x_dlc_ankle(start:stop);
y_dlc_ankle = y_dlc_ankle(start:stop);
x_dlc_meta = x_dlc_meta(start:stop);
y_dlc_meta = y_dlc_meta(start:stop);
%%
% Delay calculation

x_knee_delay = finddelay(x_arba_knee,x_dlc_knee);
x_ankle_delay = finddelay(x_arba_ankle,x_dlc_ankle);
x_meta_delay = finddelay(x_arba_meta,x_dlc_meta);

y_knee_delay = finddelay(y_arba_knee,y_dlc_knee);
y_ankle_delay = finddelay(y_arba_ankle,y_dlc_ankle);
y_meta_delay = finddelay(y_arba_meta,y_dlc_meta);

% If the delay is negative the delay calculation must be reversed.
inverted = false;
if (x_knee_delay < 0 || x_ankle_delay < 0 || x_meta_delay < 0 || y_knee_delay < 0 ||
    y_ankle_delay < 0 || y_meta_delay < 0 )
    inverted = true;
    x_knee_delay = finddelay(x_dlc_knee,x_arba_knee);
    x_ankle_delay = finddelay(x_dlc_ankle,x_arba_ankle);
    x_meta_delay = finddelay(x_dlc_meta,x_arba_meta);

    y_knee_delay = finddelay(y_dlc_knee,y_arba_knee);
    y_ankle_delay = finddelay(y_dlc_ankle,y_arba_ankle);
    y_meta_delay = finddelay(y_dlc_meta,y_arba_meta);
end

max_delay = max([x_knee_delay,x_ankle_delay,x_meta_delay,y_knee_delay,y_ankle_delay,
    y_meta_delay]);
```

Código 7.2 : Alineación de señales

```
% Data aligning
if inverted == false
    [x_arba_knee_align,x_dlc_knee_align] = alignsignals(x_arba_knee,x_dlc_knee);
```

```
    [x_arba_ankle_align,x_dlc_ankle_align] = alignsignals(x_arba_ankle,x_dlc_ankle);
    [x_arba_meta_align,x_dlc_meta_align] = alignsignals(x_arba_meta,x_dlc_meta);

    [y_arba_knee_align,y_dlc_knee_align] = alignsignals(y_arba_knee,y_dlc_knee);
    [y_arba_ankle_align,y_dlc_ankle_align] = alignsignals(y_arba_ankle,y_dlc_ankle);
    [y_arba_meta_align,y_dlc_meta_align] = alignsignals(y_arba_meta,y_dlc_meta);
end

if inverted == true
    [x_dlc_knee_align,x_arba_knee_align] = alignsignals(x_dlc_knee,x_arba_knee);
    [x_dlc_ankle_align,x_arba_ankle_align] = alignsignals(x_dlc_ankle,x_arba_ankle);
    [x_dlc_meta_align,x_arba_meta_align] = alignsignals(x_dlc_meta,x_arba_meta);

    [y_dlc_knee_align,y_arba_knee_align] = alignsignals(y_dlc_knee,y_arba_knee);
    [y_dlc_ankle_align,y_arba_ankle_align] = alignsignals(y_dlc_ankle,y_arba_ankle);
    [y_dlc_meta_align,y_arba_meta_align] = alignsignals(y_dlc_meta,y_arba_meta);
end

% Data trimming
if inverted == false
    x_dlc_knee_align = x_dlc_knee_align(max_delay+1:length(x_arba_knee_align)-
    x_knee_delay);
    x_dlc_ankle_align = x_dlc_ankle_align(max_delay+1:length(x_arba_ankle_align)-
    x_ankle_delay);
    x_dlc_meta_align = x_dlc_meta_align(max_delay+1:length(x_arba_meta_align)-
    x_meta_delay);

    x_arba_knee_align = x_arba_knee_align(max_delay+1:length(x_arba_knee_align)-
    x_knee_delay);
    x_arba_ankle_align = x_arba_ankle_align(max_delay+1:length(x_arba_ankle_align)-
    x_ankle_delay);
    x_arba_meta_align = x_arba_meta_align(max_delay+1:length(x_arba_meta_align)-
    x_meta_delay);

    y_dlc_knee_align = y_dlc_knee_align(max_delay+1:length(y_arba_knee_align)-
    y_knee_delay);
    y_dlc_ankle_align = y_dlc_ankle_align(max_delay+1:length(y_arba_ankle_align)-
    y_ankle_delay);
    y_dlc_meta_align = y_dlc_meta_align(max_delay+1:length(y_arba_meta_align)-
    y_meta_delay);

    y_arba_knee_align = y_arba_knee_align(max_delay+1:length(y_arba_knee_align)-
    y_knee_delay);
    y_arba_ankle_align = y_arba_ankle_align(max_delay+1:length(y_arba_ankle_align)-
```

```
        y_ankle_delay);
        y_arba_meta_align = y_arba_meta_align(max_delay+1:length(y_arba_meta_align)-
        y_meta_delay);

        t_dlc_trim = t_dlc(max_delay+1:min(length(t_arba),length(t_dlc)));
        t_arba_trim = t_arba(max_delay+1:min(length(t_arba),length(t_dlc)));
42  end

    if inverted == true
        x_arba_knee_align = x_arba_knee_align(max_delay+1:length(x_dlc_knee_align)-
        x_knee_delay);
        x_arba_ankle_align = x_arba_ankle_align(max_delay+1:length(x_dlc_ankle_align)-
        x_ankle_delay);
47      x_arba_meta_align = x_arba_meta_align(max_delay+1:length(x_dlc_meta_align)-
        x_meta_delay);

        x_dlc_knee_align = x_dlc_knee_align(max_delay+1:length(x_dlc_knee_align)-
        x_knee_delay);
        x_dlc_ankle_align = x_dlc_ankle_align(max_delay+1:length(x_dlc_ankle_align)-
        x_ankle_delay);
        x_dlc_meta_align = x_dlc_meta_align(max_delay+1:length(x_dlc_meta_align)-
        x_meta_delay);
52
        y_arba_knee_align = y_arba_knee_align(max_delay+1:length(y_dlc_knee_align)-
        y_knee_delay);
        y_arba_ankle_align = y_arba_ankle_align(max_delay+1:length(y_dlc_ankle_align)-
        y_ankle_delay);
        y_arba_meta_align = y_arba_meta_align(max_delay+1:length(y_dlc_meta_align)-
        y_meta_delay);

57      y_dlc_knee_align = y_dlc_knee_align(max_delay+1:length(y_dlc_knee_align)-
        y_knee_delay);
        y_dlc_ankle_align = y_dlc_ankle_align(max_delay+1:length(y_dlc_ankle_align)-
        y_ankle_delay);
        y_dlc_meta_align = y_dlc_meta_align(max_delay+1:length(y_dlc_meta_align)-
        y_meta_delay);

        t_dlc_trim = t_dlc(max_delay+1:min(length(t_arba),length(t_dlc)));
62      t_arba_trim = t_arba(max_delay+1:min(length(t_arba),length(t_dlc)));
    end
```

Código 7.3 : Filtrado de las señales

```
% Data filtering

% 8 order lowpass 6[Hz] Butterworth filter @120fps.
fc = 6;
fs = 120;
order = 8;
[b,a] = butter(order,2*fc/fs);

x_dlc_knee_filt = filtfilt(b,a,x_dlc_knee_align);
y_dlc_knee_filt = filtfilt(b,a,y_dlc_knee_align);
x_dlc_ankle_filt = filtfilt(b,a,x_dlc_ankle_align);
y_dlc_ankle_filt = filtfilt(b,a,y_dlc_ankle_align);
x_dlc_meta_filt = filtfilt(b,a,x_dlc_meta_align);
y_dlc_meta_filt = filtfilt(b,a,y_dlc_meta_align);

x_arba_knee_filt = filtfilt(b,a,x_arba_knee_align);
y_arba_knee_filt = filtfilt(b,a,y_arba_knee_align);
x_arba_ankle_filt = filtfilt(b,a,x_arba_ankle_align);
y_arba_ankle_filt = filtfilt(b,a,y_arba_ankle_align);
x_arba_meta_filt = filtfilt(b,a,x_arba_meta_align);
y_arba_meta_filt = filtfilt(b,a,y_arba_meta_align);

x_dlc_knee_dc = x_dlc_knee_filt - mean(x_dlc_knee_filt);
y_dlc_knee_dc = y_dlc_knee_filt - mean(y_dlc_knee_filt);
x_dlc_ankle_dc = x_dlc_ankle_filt - mean(x_dlc_ankle_filt);
y_dlc_ankle_dc = y_dlc_ankle_filt - mean(y_dlc_ankle_filt);
x_dlc_meta_dc = x_dlc_meta_filt - mean(x_dlc_meta_filt);
y_dlc_meta_dc = y_dlc_meta_filt - mean(y_dlc_meta_filt);

x_arba_knee_dc = x_arba_knee_filt - mean(x_arba_knee_filt);
y_arba_knee_dc = y_arba_knee_filt - mean(y_arba_knee_filt);
x_arba_ankle_dc = x_arba_ankle_filt - mean(x_arba_ankle_filt);
y_arba_ankle_dc = y_arba_ankle_filt - mean(y_arba_ankle_filt);
x_arba_meta_dc = x_arba_meta_filt - mean(x_arba_meta_filt);
y_arba_meta_dc = y_arba_meta_filt - mean(y_arba_meta_filt);
```

Código 7.4 : Gráfico de posiciones

```
%% Position plots

figure
```

```
scatter(x_arba_knee,y_arba_knee,"x")
hold on
scatter(x_arba_ankle,y_arba_ankle,"o")
hold on
scatter(x_arba_meta,y_arba_meta,"+")
grid on
legend('Knee','Ankle', 'Metatarsus')
title ('ARBA positions (raw)');
xlabel('x position (px)');
ylabel('y position (px)');
%xlim([650 1200])
%ylim([-950 -650])
set(gcf, 'PaperPosition', [0 0 20 20]); %Position plot at left hand corner with width
    5 and height 5.
set(gcf, 'PaperSize', [20 20]); %Set the paper to have width 5 and height 5.
saveas(gcf,join([sujeto,'_arba_raw_pos'],''), 'pdf'); %Save figure


figure
scatter(x_dlc_knee,y_dlc_knee,"x")
hold on
scatter(x_dlc_ankle,y_dlc_ankle,"o")
hold on
scatter(x_dlc_meta,y_dlc_meta,"+")
grid on
legend('Knee','Ankle', 'Metatarsus')
title ('DeepLabCut positions (raw)');
xlabel('x position (px)');
ylabel('y position (px)');
%xlim([650 1200])
%ylim([-950 -650])
set(gcf, 'PaperPosition', [0 0 20 20]); %Position plot at left hand corner with width
    5 and height 5.
set(gcf, 'PaperSize', [20 20]); %Set the paper to have width 5 and height 5.
saveas(gcf,join([sujeto,'_dlc_raw_pos'],''), 'pdf'); %Save figure

% Aligned plots
figure
scatter(x_arba_knee_align,y_arba_knee_align,"x")
hold on
scatter(x_arba_ankle_align,y_arba_ankle_align,"o")
hold on
scatter(x_arba_meta_align,y_arba_meta_align,"+")
grid on
```

```
legend('Knee','Ankle', 'Metatarsus')
title ('ARBA positions (aligned)');
xlabel('x position (px)');
ylabel('y position (px)');
%xlim([650 1200])
%ylim([-950 -650])
set(gcf, 'PaperPosition', [0 0 20 20]); %Position plot at left hand corner with width 5 and height 5.
set(gcf, 'PaperSize', [20 20]); %Set the paper to have width 5 and height 5.
saveas(gcf,join([sujeto,'_arba_ali_pos'],''), 'pdf'); %Save figure

figure
scatter(x_dlc_knee_align,y_dlc_knee_align,"x")
hold on
scatter(x_dlc_ankle_align,y_dlc_ankle_align,"o")
hold on
scatter(x_dlc_meta_align,y_dlc_meta_align,"+")
grid on
legend('Knee','Ankle', 'Metatarsus')
title ('DeepLabCut positions (aligned)');
xlabel('x position (px)');
ylabel('y position (px)');
%xlim([650 1200])
%ylim([-950 -650])
set(gcf, 'PaperPosition', [0 0 20 20]); %Position plot at left hand corner with width 5 and height 5.
set(gcf, 'PaperSize', [20 20]); %Set the paper to have width 5 and height 5.
saveas(gcf,join([sujeto,'_dlc_ali_pos'],''), 'pdf'); %Save figure

% Filtered plots
figure
scatter(x_arba_knee_filt,y_arba_knee_filt,"x")
hold on
scatter(x_arba_ankle_filt,y_arba_ankle_filt,"o")
hold on
scatter(x_arba_meta_filt,y_arba_meta_filt,"+")
grid on
legend('Knee','Ankle', 'Metatarsus')
title ('ARBA positions (filtered)');
xlabel('x position (px)');
ylabel('y position (px)');
%xlim([650 1200])
%ylim([-950 -650])
set(gcf, 'PaperPosition', [0 0 20 20]); %Position plot at left hand corner with width
```

```
    5 and height 5.
set(gcf, 'PaperSize', [20 20]); %Set the paper to have width 5 and height 5.
saveas(gcf,join([sujeto,'_arba_filt_pos'],''), 'pdf'); %Save figure

figure
scatter(x_dlc_knee_filt,y_dlc_knee_filt,"x")
hold on
scatter(x_dlc_ankle_filt,y_dlc_ankle_filt,"o")
hold on
scatter(x_dlc_meta_filt,y_dlc_meta_filt,"+")
grid on
legend('Knee','Ankle', 'Metatarsus')
title ('DeepLabCut positions (filtered)');
xlabel('x position (px)');
ylabel('y position (px)');
%xlim([650 1200])
%ylim([-950 -650])
set(gcf, 'PaperPosition', [0 0 20 20]); %Position plot at left hand corner with width
    5 and height 5.
set(gcf, 'PaperSize', [20 20]); %Set the paper to have width 5 and height 5.
saveas(gcf,join([sujeto,'_dlc_filt_pos'],''), 'pdf'); %Save figure

close all
```

Código 7.5 : Histograma de posiciones

```
%% Histogram analysis

% Plots the position histogram of the raw, aligned and DC filtered data
variables_title = ["hist", "histali", "histdc"];
variables_text = ["knee_x" "knee_y" "ankle_x" "ankle_y" "meta_x" "meta_y"];
variables_arba_raw = [x_arba_knee y_arba_knee, x_arba_ankle y_arba_ankle, x_arba_meta,
     y_arba_meta];
variables_dlc_raw = [x_dlc_knee y_dlc_knee, x_dlc_ankle y_dlc_ankle, x_dlc_meta,
    y_dlc_meta];
variables_arba_ali = [x_arba_knee_align y_arba_knee_align, x_arba_ankle_align
    y_arba_ankle_align, x_arba_meta_align, y_arba_meta_align];
variables_dlc_ali = [x_dlc_knee_align y_dlc_knee_align, x_dlc_ankle_align
    y_dlc_ankle_align, x_dlc_meta_align, y_dlc_meta_align];
variables_arba_dc = [x_arba_knee_dc y_arba_knee_dc, x_arba_ankle_dc y_arba_ankle_dc,
    x_arba_meta_dc, y_arba_meta_dc];
variables_dlc_dc = [x_dlc_knee_dc y_dlc_knee_dc, x_dlc_ankle_dc y_dlc_ankle_dc,
    x_dlc_meta_dc, y_dlc_meta_dc];
```

```
for j = 1:length(variables_title)
        variable_title = variables_title(j);
        if j == 1
            variables_arba = variables_arba_raw;
            variables_dlc = variables_dlc_raw;
        end
        if j == 2
            variables_arba = variables_arba_ali;
            variables_dlc = variables_dlc_ali;
        end
        if j == 3
            variables_arba = variables_arba_dc;
            variables_dlc = variables_dlc_dc;
        end
    for i = 1:length(variables_text)
        variable_text = variables_text(i);
        variable_arba = variables_arba(:,i);
        variable_dlc = variables_dlc(:,i);
        figure
        histogram(variable_arba,30)
        hold on
        histogram(variable_dlc,30)
        title (join(['Position Histogram',variable_title,variable_text],' '));
        legend('ARBA','DLC')
        set(gcf, 'PaperPosition', [0 0 20 20]); %Position plot at left hand corner
    with width 5 and height 5.
        set(gcf, 'PaperSize', [20 20]); %Set the paper to have width 5 and height 5.
        saveas(gcf,join([sujeto,variable_title,variable_text],'_'), 'pdf'); %Save
    figure
    end
end

close all

%% Fourier analysis

variables_text = ["knee_x" "knee_y" "ankle_x" "ankle_y" "meta_x" "meta_y"];
variables_arba = [x_arba_knee_dc y_arba_knee_dc, x_arba_ankle_dc y_arba_ankle_dc,
    x_arba_meta_dc, y_arba_meta_dc];
variables_dlc = [x_dlc_knee_dc y_dlc_knee_dc, x_dlc_ankle_dc y_dlc_ankle_dc,
    x_dlc_meta_dc, y_dlc_meta_dc];
```

Código 7.6 : Cálculo de FFT

```
for i = 1:length(variables_text)
    variable_text = variables_text(i);
    variable_arba = variables_arba(:,i);
    variable_dlc = variables_dlc(:,i);
    t = t_arba;
    z = variable_arba;
    z2 = variable_dlc;
    Err = z-z2;
    mean_err = mean(Err);
    mean_abs_err = mean(abs(Err));
    RMSE = sqrt(mean(Err.^2));
    std_err = std(Err);
    L = numel(z);
    fps = 120;
    time_s = linspace(0,L/fps,L);
    Ts = mean(diff(t))/fps;
    Fs = 1/Ts;
    Fn = Fs/2;
    FTs = fft(z)/L;
    FTs2 = fft(z2)/L;
    Fv = linspace(0,1, fix(L/2)+1)*Fn;
    Iv = 1:numel(Fv);
    figure
    subplot(2, 1, 1);
    plot(time_s,z);
    hold on
    plot(time_s,z2);
    hold on
    plot(time_s,Err)
    set(gca,'xlim',[1 9]);
    grid;
    title (join(['Position v/s time',variable_text],' '));
    xlabel('time(s)');
    ylabel('Position (px)');
    legend('ARBA','DLC','ERROR')
    txt_me = {'Mean error:',num2str(mean_err)};
    txt_mea = {'Mean absolute error:',num2str(mean_abs_err)};
    txt_rmse = {'RMS error:',num2str(RMSE)};
    txt_stde = {'Standard error:',num2str(std_err)};
    text(2,min(variable_arba),txt_me)
```

```
        text(4,min(variable_arba),txt_mea)
        text(6,min(variable_arba),txt_rmse)
        text(8,min(variable_arba),txt_stde)
        subplot(2, 1, 2);
        plot(Fv, abs(FTs(Iv))*2);
        hold on
        plot(Fv, abs(FTs2(Iv))*2);
        xlim([0 10])
        grid
        title (join(['FFT',variable_text],' '));
        xlabel('Frequency (Hz)');
        ylabel ('Amplitude (px)');
        legend('ARBA','DLC')
        set(gcf, 'PaperPosition', [0 0 20 20]); %Position plot at left hand corner with
        width 5 and height 5.
        set(gcf, 'PaperSize', [20 20]); %Set the paper to have width 5 and height 5.
        saveas(gcf,join([sujeto,'fft',variable_text],'_'), 'pdf'); %Save figure
    end

    close all

    %% Angle calculation

    % ARBA vertices definition
    P0_arba = [x_arba_knee_filt,y_arba_knee_filt];
    P1_arba = [x_arba_ankle_filt,y_arba_ankle_filt];
```

Código 7.7 : Cálculo de ángulo

```
    P2_arba = [x_arba_meta_filt,y_arba_meta_filt];

    % DLC vertices definition
    P0_dlc = [x_dlc_knee_filt,y_dlc_knee_filt];
    P1_dlc = [x_dlc_ankle_filt,y_dlc_ankle_filt];
    P2_dlc = [x_dlc_meta_filt,y_dlc_meta_filt];

    for i = 1:length(P0_arba)
        angle_arba(i) = rad2deg(atan2(abs(det([P2_arba(i,:)-P0_arba(i,:);P1_arba(i,:)-
        P0_arba(i,:)])),dot(P2_arba(i,:)-P0_arba(i,:),P1_arba(i,:)-P0_arba(i,:))));
    end

    for i = 1:length(P0_dlc)
        angle_dlc(i) = rad2deg(atan2(abs(det([P2_dlc(i,:)-P0_dlc(i,:);P1_dlc(i,:)-P0_dlc(i
```

```
    ,:)])),dot(P2_dlc(i,:)-P0_dlc(i,:),P1_dlc(i,:)-P0_dlc(i,:))));
end

Err = angle_arba-angle_dlc;
mean_err = mean(Err);
mean_abs_err = mean(abs(Err));
RMSE = sqrt(mean(Err.^2));
std_err = std(Err);

% Angle plot

figure
plot(angle_arba)
hold on
plot(angle_dlc)
hold on
plot(Err)
xlim([0 length(angle_arba)])
grid on
title ('Angle v/s frame');
xlabel('Frame');
ylabel('Angle (degrees)');
legend('ARBA','DLC','ERROR')
txt_me = {'Mean error:',num2str(mean_err)};
txt_mea = {'Mean absolute error:',num2str(mean_abs_err)};
txt_rmse = {'RMS error:',num2str(RMSE)};
txt_stde = {'Standard error:',num2str(std_err)};
text(100,29,txt_me)
text(300,29,txt_mea)
text(600,29,txt_rmse)
text(800,29,txt_stde)
set(gcf, 'PaperPosition', [0 0 20 20]); %Position plot at left hand corner with width
    5 and height 5.
set(gcf, 'PaperSize', [20 20]); %Set the paper to have width 5 and height 5.
saveas(gcf,join([sujeto,'_angle'],''), 'pdf'); %Save figure

close all

%% Distance between signals using dynamic time warping

variables_text = ["knee_x" "knee_y" "ankle_x" "ankle_y" "meta_x" "meta_y"];
variables_arba = [x_arba_knee_dc y_arba_knee_dc, x_arba_ankle_dc y_arba_ankle_dc,
    x_arba_meta_dc, y_arba_meta_dc];
variables_dlc = [x_dlc_knee_dc y_dlc_knee_dc, x_dlc_ankle_dc y_dlc_ankle_dc,
```

```
    x_dlc_meta_dc, y_dlc_meta_dc];
```

Código 7.8 : Cálculo de distancias absolutas

```
for i = 1:length(variables_text)
    variable_text = variables_text(i);
    variable_arba = variables_arba(:,i);
    variable_dlc = variables_dlc(:,i);
    dist = dtw(variable_arba,variable_dlc,'absolute');
    mean_dist = dist/length(variable_arba);
    figure
    dtw(variable_arba,variable_dlc,'absolute')
    txt_dist = {'Mean distance:',num2str(mean_dist)};
    text(200,max(variable_arba),txt_dist)
    legend('ARBA','DLC')
    xlabel('Frame');
    ylabel('Position (px)');
    grid on
    set(gcf, 'PaperPosition', [0 0 20 20]); %Position plot at left hand corner with
    width 5 and height 5.
    set(gcf, 'PaperSize', [20 20]); %Set the paper to have width 5 and height 5.
    saveas(gcf,join([sujeto,'dist',variable_text],'_'), 'pdf'); %Save figure
end

close all
```

Para el manejo de datos y creación de videos con el sistema *ARBA* se implementa el siguiente extracto de código en Python:

Código 7.9 : Extracto de implementación de servicios

```
# Initializes filter service
filter = Filter(
    data_raw_name=self.save_data_as,
    data_int_name=self.data_int_path,
    data_flag_name=self.data_flag_path,
    data_error_name=self.data_error_path,
    data_filt_name=self.data_filt_path,
    data_out_name=self.data_out_path,
    method="polynomial",
    order_int=3,
```

```
        cutoff=6.0,
        fs=120.0,
        order_filt=8,
        cleanup=True,
        plane=plane,
    )

    # Creates output dataframe
    if neural_parameters.get("filter_data"):
        filter.process_data()

    if graphics_parameters.get("filter_video") and neural_parameters.get(
        "filter_data"
    ):
        dataframe = self.data_out_path  # Filtered data.

    if not graphics_parameters.get(
        "filter_video"
    ) or not neural_parameters.get("filter_data"):
        dataframe = self.save_data_as  # Raw data.

    # Initializes graphics service
    graphics = Graphics(
        video_path=self.video_path,
        start_frame=request["video_parameters"].get("start_frame"),
        stop_frame=request["video_parameters"].get("stop_frame"),
        video_output_fr=graphics_parameters.get("video_output_fr"),
        video_out_path=self.video_out_path,
        skipped_frames=neural_parameters.get("skipped_frames"),
        convert=graphics_parameters.get("convert_video"),
        text_id="Exam ID: " + str(exam_id),
        logo_path=self.logo_path,
        font_path=self.font_path,
        dataframe=dataframe,
        plane=plane,
        filter=graphics_parameters.get("filter_video"),
        post_video_path=self.relative_filt_video_output_path,
        save_temp_video_as=self.video_out_path,
        save_video_as=self.save_video_as,
        relative_video_output_path=self.relative_video_filter_path,
        draw_marker_names=graphics_parameters.get("draw_marker_names"),
        draw_keypoints=graphics_parameters.get("draw_keypoints"),
        draw_skeleton=graphics_parameters.get("draw_skeleton"),
        draw_angles=graphics_parameters.get("draw_angles"),
```

```
        draw_error=graphics_parameters.get("draw_error"),
        draw_graphics=graphics_parameters.get("draw_graphics"),
        display=graphics_parameters.get("display"),
58  )

    # Creates output video
    if graphics_parameters.get("create_video"):
        graphics.create_video()
63
    # Initializes report service
    report = Report(
        data_out_name=self.data_out_path,
        video_out_path=self.video_out_path,
68      report_path=self.save_report_to,
        plane=plane,
        exam_id=exam_id,
        video_id=video_id,
    )
73  report.create_report()
```

Este código hace uso de los servicios implementados en los códigos 7.10, 7.11 y 7.12.

Código 7.10 : Servicio de filtrado de datos en Python

```
    # -*- coding: utf-8 -*-
2   """
    @file    : filter_service.py
    @brief   : Process CSV data.
    @date    : 2021/12/09
    @version : 4.0.0
7   @author  : Lucas Cortes.
    @contact : lucas.cortes@lanek.cl
    @bug     : None.
    """

12  import os
    import time
    import logging
    import numpy as np
    import pandas as pd
17  from contextlib import suppress
    from scipy.signal import butter, filtfilt
```

```
from source.services.OpenCV.opencv_service import OpenCV
import source.services.Neural.constants.constants as constants
from source.services.Utils.Helpers.helpers_service import Helpers


class Filter:
    def __init__(
        self,
        data_raw_name,
        data_int_name,
        data_flag_name,
        data_error_name,
        data_filt_name,
        data_out_name,
        method,
        order_int,
        cutoff_low,
        cutoff_high,
        fs,
        order_filt,
        cleanup,
        plane,
    ):
        logging.debug(
            f"[Filter] Initializing filter service for dataframe: {os.path.basename(
    data_raw_name)}"
        )
        self.data_raw_name = data_raw_name
        self.data_int_name = data_int_name
        self.data_flag_name = data_flag_name
        self.data_error_name = data_error_name
        self.data_filt_name = data_filt_name
        self.data_out_name = data_out_name
        self.method = method
        self.order_int = order_int
        self.cutoff_low = cutoff_low
        self.cutoff_high = cutoff_high
        self.fs = fs
        self.order_filt = order_filt
        self.cleanup = cleanup
        self.plane = plane
        # self.process_data()

    def get_marker_pos(idx, dataframe):
```

```
            markers = {}
            for descriptor in dataframe:
                if descriptor in constants.marker_components:
                    data_pos = dataframe[descriptor][idx]
                    if descriptor in constants.marker_x_list:
                        data_x = data_pos
                    if descriptor in constants.marker_y_list:
                        data_y = data_pos
                        with suppress(Exception):
                            # Stores coordinates for skeleton reconstruction and angle
        calculation.
                            markers[constants.marker_components[descriptor]] = [
                                int(data_x),
                                int(data_y),
                            ]
            return markers

        # Defines secondary angle drawing function.
        def calculate_angles(self):
            dataframe = pd.read_csv(self.data_filt_name)
            os.remove(self.data_filt_name)
            for angle_name, angle_info in constants.reporting_angles[self.plane].items():
                angle_list = {}
                for idx in range(len(dataframe["frame"].values.tolist())):

                    markers = Filter.get_marker_pos(
                        idx=idx,
                        dataframe=dataframe,
                    )

                    marker_names = [a_name, b_name, c_name] = angle_info[0]
                    are_visible = [
                        name in markers or name == "Vertical" or name == "Horizontal"
                        for name in marker_names
                    ]
                    if all(are_visible):
                        phase = angle_info[1]
                        sign = angle_info[2]
                        multiplier = 1
                        marker_a = markers[a_name]
                        marker_b = markers[b_name]
                        x_a, y_a = marker_a
                        x_b, y_b = marker_b
                        if c_name == "Vertical":
```

```
                            x_c, y_c = x_b, y_a
                            if x_b > x_a:
                                multiplier = -1
                        elif c_name == "Horizontal":
                            x_c, y_c = x_a, y_b
                            if y_b > y_a:
                                multiplier = -1
                        else:
                            marker_c = markers[c_name]
                            x_c, y_c = marker_c
                        angle = Helpers.find_angle([x_a, y_a], [x_b, y_b], [x_c, y_c])
                        angle = sign * (multiplier * angle - phase)
                        angle_list[dataframe[angle_name][idx]] = angle
                dataframe[angle_name].replace(angle_list, inplace=True)

            dataframe.to_csv(self.data_filt_name, index=False)

        # Defines butterworth lowpass filter function.
        def butter_lowpass_filter(data, cutoff, fs, order):
            nyq = 0.5 * fs  # Nyquist Frequency
            normal_cutoff = cutoff / nyq  # Normalise frequency
            # Get the filter coefficients
            b, a = butter(order, normal_cutoff, btype="low", analog=False)
            y = filtfilt(b, a, data)  # Filter data
            return y

        # Defines butterworth highpass filter function.
        def butter_highpass_filter(data, cutoff, fs, order):
            nyq = 0.5 * fs  # Nyquist Frequency
            normal_cutoff = cutoff / nyq  # Normalise frequency
            # Get the filter coefficients
            b, a = butter(order, normal_cutoff, btype="high", analog=False)
            y = filtfilt(b, a, data)  # Filter data
            return y

        def butter_bandpass_filter(data, cutoff_low, cutoff_high, fs, order):
            nyq = 0.5 * fs # Nyquist Frequency
            normal_low = cutoff_low / nyq # Normalise frequency
            normal_high = cutoff_high / nyq # Normalise frequency
            b, a = butter(order, [normal_low, normal_high], btype='band', analog=False)
            y = filtfilt(b, a, data)  # Filter data
            return y

         # Defines butterworth filter function.
```

```
        def butter_filter(data, cutoff, fs, order, filt_type):
            nyq = 0.5 * fs  # Nyquist Frequency
            normal_cutoff = cutoff / nyq  # Normalise frequency
            # Get the filter coefficients
            b, a = butter(order, normal_cutoff, btype=filt_type, analog=False)
            y = filtfilt(b, a, data)  # Filter data
            return y

        # Defines butterworth bandpass filter function.
        def butter_bandpass_filter(data, cutoff_low, cutoff_high, fs, order):
            x = Filter.butter_filter(data, cutoff_high, fs, order, "high")
            y = Filter.butter_filter(x, cutoff_low, fs, order, "low")
            return y
            """
            x = Filter.butter_lowpass_filter(data, cutoff_low, fs, order)
            y = Filter.butter_highpass_filter(x, cutoff_high, fs, order)
            return y
            """

        # Defines data interpolation function.
        def interpolate_data(self):
            # Data reading
            data_raw = pd.read_csv(self.data_raw_name)
            data_int = data_raw.interpolate(method=self.method, order=self.order_int)
            data_int.to_csv(self.data_int_name, index=False)

        # Defines data filtering function.
        def filter_data(self):
            # Data reading
            data_int = pd.read_csv(self.data_int_name)
            start = False
            frames = data_int["frame"]
            for descriptor in data_int:
                if descriptor != "frame":
                    down_limit = 0
                    up_limit = -1
                    found = False
                    data_temp = data_int[descriptor]
                    data_temp_list = data_temp.values.tolist()  # [0:-1]
                    total = len(data_temp_list)
                    for i in range(total):
                        value = data_temp_list[i]
                        if value == value and found == False:
                            down_limit = i
```

```
                            found = True
                        if value != value and found == True:
                            up_limit = i - 1
                            found = False
                    data_temp_list = data_temp_list[down_limit:up_limit]

                    data_temp_filt = Filter.butter_bandpass_filter(
                        data_temp_list, self.cutoff_low, self.cutoff_high, self.fs, self.
        order_filt
                    )
                    """
                    data_temp_filt = Filter.butter_lowpass_filter(
                        data_temp_list, self.cutoff_low, self.fs, self.order_filt
                    )
                    """

                    low_array = np.full(down_limit, np.nan)

                    up_array = np.full(total - up_limit - 1, np.nan)

                    data_temp_filt = np.concatenate(
                        (low_array, data_temp_filt, up_array), axis=0
                    )

                    data_frame = pd.DataFrame(data_temp_filt, columns=[str(descriptor)])

                    if start:

                        data_filt = pd.concat([data_filt, data_frame], axis=1)

                    else:
                        data_filt = data_frame
                    start = True

            data_filt = pd.concat([frames, data_filt], axis=1)

            data_filt.to_csv(self.data_filt_name, index=False)

        # Defines data error calculation function.
        def error_data(self):
            # Data reading
            data_int = pd.read_csv(self.data_int_name)
            data_filt = pd.read_csv(self.data_filt_name)
            for descriptor in data_int:
```

```
237             data_error = []
                for i in range(len(data_int[descriptor])):
                    data_error.append(
                        abs(data_filt[descriptor][i] - data_int[descriptor][i])
                    )
242
                data_error = pd.DataFrame(data_error, columns=[str(descriptor)])

                data_old = data_new if descriptor != "frame" else data_int[descriptor]
                data_new = (
                    pd.concat([data_old, data_error], axis=1)
                    if descriptor != "frame"
                    else data_old
                )

252         data_new.to_csv(self.data_error_name, index=False)

        # Defines data flag creation function.
        def flag_data(self):
            # Data reading
257         data_raw = pd.read_csv(self.data_raw_name)
            data_int = pd.read_csv(self.data_int_name)
            # frame_flag = [False] * len(data_int["frame"])
            for descriptor in data_int:
                data_flag = []
262             for i in range(len(data_int[descriptor])):
                    flag = True if pd.isna(data_raw[descriptor][i]) else False
                    data_flag.append(flag)

                data_flag = pd.DataFrame(data_flag, columns=[str(descriptor)])
267
                data_old = data_new if descriptor != "frame" else data_int[descriptor]
                data_new = (
                    pd.concat([data_old, data_flag], axis=1)
                    if descriptor != "frame"
272                 else data_old
                )
                # for i in range(len(data_flag)):
                # frame_flag = frame_flag or data_flag

277         # data_new.insert(1, 'frame_FLAG', frame_flag)
            data_new.to_csv(self.data_flag_name, index=False)

        # Defines data out creation function.
```

```
    def out_data(self):
        # Data reading
        data_int = pd.read_csv(self.data_int_name)
        data_filt = pd.read_csv(self.data_filt_name)
        data_error = pd.read_csv(self.data_error_name)
        data_flag = pd.read_csv(self.data_flag_name)
        for descriptor in data_int:
            data_int_out = pd.DataFrame(data_int[descriptor], columns=[str(descriptor)
])

            data_filt_out = pd.DataFrame(
                data_filt[descriptor].tolist(), columns=[str(descriptor) + "_FILT"]
            )

            data_error_out = pd.DataFrame(
                data_error[descriptor].tolist(), columns=[str(descriptor) + "_ERROR"]
            )

            data_flag_out = pd.DataFrame(
                data_flag[descriptor].tolist(), columns=[str(descriptor) + "_FLAG"]
            )

            data_old = data_new if descriptor != "frame" else data_int[descriptor]
            data_new = (
                pd.concat(
                    [
                        data_old,
                        data_int_out,
                        data_filt_out,
                        data_error_out,
                        data_flag_out,
                    ],
                    axis=1,
                )
                if descriptor != "frame"
                else data_old
            )

        # data_new.insert(1, "frame_FLAG", data_flag["frame_FLAG"])
        data_new.to_csv(self.data_out_name, index=False)

    # Defines CSV cleaning function.
    def cleanup_data(self):
        os.remove(self.data_int_name)
```

```
            os.remove(self.data_filt_name)
            os.remove(self.data_error_name)
            os.remove(self.data_flag_name)
327         os.remove(self.data_raw_name)

        # Defines data processing function.
        def process_data(self):
            tic = time.time()
332
            self.interpolate_data()
            self.filter_data()
            self.calculate_angles()
            self.error_data()
337         self.flag_data()
            self.out_data()

            if self.cleanup:
                self.cleanup_data()
342
            tac = time.time()
            print("Data processing time:", tac - tic)
```

Código 7.11 : Servicio gráfico en Python

```python
# -*- coding: utf-8 -*-
"""
@file    : graphics_service.py
@brief   : Renderizes video with CSV data.
@date    : 2021/10/04
@version : 4.0.0
@author  : Lucas Cortes.
@contact : lucas.cortes@lanek.cl
@bug     : None.
"""

import os
import cv2
import time
import logging
import subprocess
import numpy as np
import pandas as pd
from typing import Tuple
from contextlib import suppress

from source.services.OpenCV.opencv_service import OpenCV
import source.services.Neural.constants.constants as constants
from source.services.Utils.Helpers.helpers_service import Helpers


class Graphics:
    def __init__(
        self,
        video_path,
        start_frame,
        stop_frame,
        video_output_fr,
        video_out_path,
        skipped_frames,
        convert,
        text_id,
        logo_path,
        font_path,
        dataframe,
        plane,
```

```
            filter,
            post_video_path,
            save_temp_video_as,
            save_video_as,
            relative_video_output_path,
            draw_marker_names=True,
            draw_keypoints=True,
            draw_skeleton=True,
            draw_angles=True,
            draw_error=False,
            draw_graphics=True,
            display=False,
        ):
            logging.debug(
                f"[Graphics] Initializing graphics service for video: {os.path.basename(
        video_path)}"
            )
            self.video_path = video_path
            self.start_frame = start_frame
            self.stop_frame = stop_frame
            self.video_output_fr = video_output_fr
            self.video_out_path = video_out_path
            self.skipped_frames = skipped_frames
            self.convert = convert
            self.text_id = text_id
            self.logo = cv2.imread(logo_path, -1)
            self.font_path = font_path
            self.dataframe = pd.read_csv(dataframe)
            self.plane = plane
            self.filter = filter
            self.post_video_path = post_video_path
            self.save_temp_video_as = save_temp_video_as
            self.save_video_as = save_video_as
            self.relative_video_output_path = relative_video_output_path
            self.draw_marker_names = draw_marker_names
            self.draw_keypoints = draw_keypoints
            self.draw_skeleton = draw_skeleton
            self.draw_angles = draw_angles
            self.draw_error = draw_error
            self.draw_graphics = draw_graphics
            self.display = display
            # self.create_video()

        # Defines progress bar function.
```

```
    def printProgressBar(
        self,
        iteration,
        total,
        prefix="",
        suffix="",
        decimals=1,
        length=100,
        fill="X",
        printEnd="\r",
    ):

        percent = ("{0:." + str(decimals) + "f}").format(
            100 * (iteration / float(total))
        )
        filledLength = int(length * iteration // total)
        bar = fill * filledLength + "-" * (length - filledLength)
        print(f"\r{prefix} |{bar}| {percent}% {suffix}", end=printEnd)
        # Print new line on completion
        if iteration == total:
            print()

    # Defines error fetching function
    def get_error(self, total_frames):
        error_values = {}
        for descriptor in self.dataframe:
            if descriptor in constants.angle_list:
                data_error = self.dataframe[descriptor + "_ERROR"][
                    max(self.skipped_frames, self.start_frame) : max(
                        self.stop_frame, total_frames
                    )
                ].tolist()
                mean_error = np.nanmean(data_error)
                max_error = np.nanmax(data_error)
                error_values[descriptor] = {mean_error, max_error}
        return error_values

    # Defines graphics drawing function.
    def draw_graphics_fun(self, frame, frame_num, text, logo):
        # Draw frame number on top left corner.
        OpenCV.draw_frame_number(
            frame=frame,
            current_frame="Frame: " + str(frame_num),
        )
```

```
            # Draws exam_id information.
            OpenCV.draw_info(
                frame=frame,
                text=text,
                position=(150, 15),
                bgr=(100, 0, 255),
                font_scale=0.5,
                font_thickness=2,
            )

            # Overlays lanek logo
            OpenCV.overlay_logo(frame=frame, logo=logo, x_offset=0, y_offset=0, scale=
        0.15)

            return frame

        # Position fetching function
        def get_marker_pos(self, idx):
            markers = {}
            markers_filt = {}
            for descriptor in self.dataframe:
                if descriptor in constants.marker_components:
                    data_flag = False
                    data_pos = self.dataframe[descriptor][idx]
                    if filter:
                        data_pos_filt = self.dataframe[descriptor + "_FILT"][idx]
                        data_flag = self.dataframe[descriptor + "_FLAG"][idx]
                    if descriptor in constants.marker_x_list:
                        data_x = data_pos
                        if filter:
                            data_x_filt = data_pos_filt
                    if descriptor in constants.marker_y_list:
                        data_y = data_pos
                        if filter:
                            data_y_filt = data_pos_filt
                        with suppress(Exception):
                            # Stores coordinates for skeleton reconstruction.
                            markers[constants.marker_components[descriptor]] = [
                                int(data_x),
                                int(data_y),
                                data_flag,
                            ]
                            if filter:
```

```
                                markers_filt[constants.marker_components[descriptor]] = [
                                    int(data_x_filt),
                                    int(data_y_filt),
                                    data_flag,
                                ]
            # TODO: revisar
            return markers, markers_filt

        # Defines marker name drawing function.
        def draw_marker_names_fun(self, frame, markers, color: Tuple[int, int, int] = None
        ):
            for marker in markers.keys():
                if color is not None:
                    text_color = color
                else:
                    text_color = constants.aba_markers[self.plane][marker]
                    text_color = Helpers.get_color_code(text_color)
                data_x, data_y, data_flag = markers[marker]
                coordinates = [int(data_x) - int(len(marker) * 3.5), int(data_y) - 20]
                text_color = (100, 0, 255) if data_flag else text_color
                # Draws markers names
                OpenCV.draw_text(
                    frame=frame,
                    text=marker,
                    position=coordinates,
                    bgr=text_color,
                )
            return frame

        # Defines marker keypoint drawing function.
        def draw_keypoints_fun(self, frame, markers, color: Tuple[int, int, int] = None):
            for marker in markers.keys():
                if color is not None:
                    circle_color = color
                else:
                    circle_color = constants.aba_markers[self.plane][marker]
                data_x, data_y, data_flag = markers[marker]
                coordinates = [int(data_x), int(data_y)]
                circle_color = (100, 0, 255) if data_flag else circle_color
                # Draws marker keypoints
                OpenCV.draw_circle(
                    frame=frame,
                    position=coordinates,
                    color=circle_color,
```

```
                )
            return frame

        # Defines skeleton drawing function
        def draw_skeleton_fun(self, frame, markers, color="white", thickness=2):
            skeleton = {}
            for name, marker in markers.items():
                last_x, last_y, _ = marker
                skeleton[name] = [last_x, last_y]
            connected_part_names = constants.aba_connected_markers[self.plane]
            adjacent_keypoints = Helpers.get_skeleton_adjacent_keypoints(
                pose=skeleton, connected_part_names=connected_part_names
            )
            markers = {}
            return OpenCV.draw_lines(
                frame=frame,
                keypoints=adjacent_keypoints,
                color=color,
                thickness=thickness,
            )

        # Defines primary angle drawing function.
        def draw_angles_fun(
            self, frame, idx, title_text, filter, x_pos=100, y_pos=150, title=100, jump=30
        ):
            data_flag = False
            for descriptor in self.dataframe:
                # Draws plane information
                OpenCV.draw_info(
                    frame=frame,
                    text="Plano: " + self.plane + title_text,
                    position=(x_pos, title),
                    bgr=(255, 200, 0),
                    font_scale=0.8,
                    font_thickness=1,
                )
                if descriptor in constants.angle_list:
                    data_ang = self.dataframe[descriptor][idx]
                    if filter:
                        data_ang = self.dataframe[descriptor + "_FILT"][idx]
                    with suppress(Exception):
                        data_flag = self.dataframe[descriptor + "_FLAG"][idx]
                    color_text = (100, 0, 255) if data_flag else (255, 200, 0)
                    # ang_to_text = "ocluido" if (pd.isna(data_ang) or data_flag) else str
```

```
        (round(data_ang,1))
                    ang_to_text = descriptor + ": " + str(round(data_ang, 1))

                    # Draws angle information
                    OpenCV.draw_info(
                        frame=frame,
                        text=ang_to_text,
                        position=(x_pos, y_pos),
                        bgr=color_text,
                        font_scale=0.8,
                        font_thickness=1,
                    )
                    y_pos += jump
            return frame

        # Defines error drawing function.
        def draw_error_fun(self, frame, error_values, x_pos=400, y_pos=150, jump=30):
            for descriptor in self.dataframe:
                if descriptor in constants.angle_list:
                    mean_error, max_error = error_values[descriptor]

                    error_to_text = (
                        "mean: "
                        + str(round(mean_error, 1))
                        + ", max: "
                        + str(round(max_error, 1))
                    )

                    # Draws error information
                    if filter:
                        OpenCV.draw_info(
                            frame=frame,
                            text=error_to_text,
                            position=(x_pos, y_pos),
                            bgr=(200, 255, 0),
                            font_scale=0.8,
                            font_thickness=1,
                        )

                    y_pos += jump
            return frame

        # Defines ffmpeg conversion function.
        def convert_video(self):
```

```
            # Starts timer.
            tic = time.time()
            logging.debug(f"[NeuralController] Converting video: {self.post_video_path}")
            p1 = subprocess.Popen(
306             f"ffmpeg -i {self.save_temp_video_as} -vcodec libx264 {self.save_video_as}
        ",
                shell=True,
            )
            p1.wait()
            os.remove(self.save_temp_video_as)
311         post_video_path = self.relative_video_output_path
            logging.debug(f"[NeuralController] Converted video: {self.post_video_path}")
            tac = time.time()
            print("Conversion time:", tac - tic)

316     # Defines video creation function.
        def create_video(self):

            # Starts timer.
            tic = time.time()
321
            # Fixes extremes.
            fix = 2

            # Calls OpenCV.
326         open_cv = OpenCV(
                input_video_path=self.video_path,
                start_frame=self.start_frame,
                stop_frame=self.stop_frame,
            )
331         total_frames = open_cv.total_frames()
            fps, width, height = open_cv.get_video_info()

            # Defines ouput video parameters.
            output_video = OpenCV.create_output_video(
336             framerate=self.video_output_fr,
                path=self.video_out_path,
                resolution=(width, height),
            )

341         # Creates frame list
            frames = self.dataframe["frame"].values.tolist()
            first_frame = frames[0]
            frames = pd.Index(frames)
```

```
346             # Gets error
                if self.draw_error:
                    error_values = self.get_error(total_frames=total_frames)

                while True:
351                 ret, frame, frame_num = open_cv.read_frame()
                    if frame_num >= first_frame:

                        # Creates index constant.
                        idx = frames.get_loc(frame_num)
356
                        # Calculates markers positions.
                        markers, _ = self.get_marker_pos(idx=idx)

                        # Draws markers.
361                     if self.draw_marker_names:
                            self.draw_marker_names_fun(
                                frame=frame,
                                markers=markers,
                                color=(255, 255, 255),
366                         )

                        if self.draw_keypoints:
                            self.draw_keypoints_fun(
                                frame=frame,
371                             markers=markers,
                                # color=(255,255,255),
                            )

                        # Draws skeleton.
376                     if self.draw_skeleton:
                            self.draw_skeleton_fun(
                                frame=frame,
                                markers=markers,
                                color="white",
381                             thickness=2,
                            )

                        # Draws angles.
                        title_text = " (filtered position)" if filter else " (raw angle)"
386                     if self.draw_angles:
                            self.draw_angles_fun(
                                frame=frame,
```

```
                            idx=idx,
                            title_text=title_text,
                            filter=self.filter,
                            x_pos=100,
                            y_pos=150,
                            title=100,
                            jump=30,
                        )

                    if self.draw_error:
                        self.draw_error_fun(
                            frame=frame,
                            error_values=error_values,
                            x_pos=400,
                            y_pos=150,
                            jump=30,
                        )

                    # Draws graphics.
                    if self.draw_graphics:
                        self.draw_graphics_fun(
                            frame=frame,
                            frame_num=frame_num,
                            text=self.text_id,
                            logo=self.logo,
                        )

                    # Shows the live processing of the video file.
                    if self.display:
                        OpenCV.show_display(id="Rendering", frame=frame, fps=fps)
                        if cv2.waitKey(int(1000 / fps)) == ord("q"):
                            break


                    # Progress bar feedback
                    with suppress(Exception):
                        last_frame = (
                            total_frames - self.skipped_frames - 2 * fix
                            if (self.skipped_frames > self.start_frame)
                            else total_frames - 2 * fix
                        )
                        self.printProgressBar(
                            frame_num - max(self.skipped_frames, self.start_frame),
                            last_frame,
```

```
                            prefix="Writing:",
                            suffix="Complete",
                            length=50,
                        )

                    # Saves frame.
                    output_video.write(frame)

                    # Checks if it's done processing.
                    if open_cv.is_last_frame(frame_num + fix):
                        open_cv.release_video()
                        output_video.release()
                        cv2.destroyAllWindows()
                        break

            # Stops processing timer.
            tac = time.time()
            print("Rendering time:", tac - tic)

            if self.convert:
                self.convert_video(self)
```

Código 7.12 : Servicio de reportería en Python

```
# -*- coding: utf-8 -*-
"""
@file    : report_service.py
@brief   : Creates video report based on CSV data.
@date    : 2021/09/29
@version : 1.0.0
@author  : Lucas Cortes.
@contact : lucas.cortes@lanek.cl
@bug     : None.
"""

import os
import cv2
import time
import json
import logging
import numpy as np
import pandas as pd
from contextlib import suppress
```

```
from scipy.signal import find_peaks

from source.services.OpenCV.opencv_service import OpenCV
import source.services.Neural.constants.constants as constants
from source.services.Utils.Helpers.helpers_service import Helpers


class Report:
    def __init__(
        self, data_out_name, video_out_path, report_path, plane, exam_id, video_id
    ):
        logging.debug(
            f"[Report] Initializing report service for video: {os.path.basename(
video_out_path)}"
        )
        self.data_out_name = data_out_name
        self.video_out_path = video_out_path
        self.report_path = report_path
        self.plane = plane
        self.exam_id = exam_id
        self.video_id = video_id
        os.makedirs(self.report_path, exist_ok=True)
        # frame_path = os.path.join(self.report_path, self.video_id)
        # os.makedirs(frame_path, exist_ok=True)

    # Defines candidate searching function.
    def get_candidates(self):
        dataframe = pd.read_csv(self.data_out_name)
        markers = {}
        for descriptor in dataframe:
            if descriptor in constants.marker_components:
                data = dataframe[descriptor].values.tolist()
                data_inverted = []
                for item in data:
                    data_inverted.append(-item)
                # local_zero = np.diff(np.sign(np.diff(data))).nonzero()[0] + 1
                # local_minima = (np.diff(np.sign(np.diff(data))) > 0).nonzero()[0] +
1
                # local_maxima = (np.diff(np.sign(np.diff(data))) < 0).nonzero()[0] +
1
                local_maxima, _ = find_peaks(data)
                local_minima, _ = find_peaks(data_inverted)

                if descriptor in constants.marker_x_list:
```

```
                    x_index_min = local_minima.tolist()  # [1:-1]
                    x_index_max = local_maxima.tolist()  # [1:-1]

                if descriptor in constants.marker_y_list:
                    y_index_min = local_minima.tolist()  # [1:-1]
                    y_index_max = local_maxima.tolist()  # [1:-1]

                    markers[constants.marker_components[descriptor]] = {
                        "x_min": x_index_min,
                        "x_max": x_index_max,
                        "y_min": y_index_min,
                        "y_max": y_index_max,
                    }

        return markers

    # Defines frame extraction function.
    def get_frames(self, markers):
        print("Extracting frames...")
        for marker in markers.keys():
            for component in markers[marker]:
                video = cv2.VideoCapture(self.video_out_path)
                video.set(cv2.CAP_PROP_POS_FRAMES, markers[marker][component])
                _, frame = video.read()
                frame_path = os.path.join(
                    self.report_path, self.video_id, str(marker + component) + ".jpg"
                )
                cv2.imwrite(frame_path, frame)
        print("Frame extraction completed")
        video.release()

    # Defines frame extraction function.
    def get_angles(self, markers):
        dataframe = pd.read_csv(self.data_out_name)
        angles = {}
        for marker in markers.keys():
            for component in markers[marker]:
                index = markers[marker][component]
                key = str(marker + "_" + component)  # markers[marker][component]
                found = False
                for descriptor in dataframe:
                    if descriptor in constants.angle_list:
                        value = round(
                            dataframe[descriptor + "_FILT"].values.tolist()[index], 1
```

```
                            )
                            if found:
107                             angles[key].update({descriptor: value})
                            if not found:
                                angles[key] = {descriptor: value}
                                found = True
            return angles
112
        def dump_json(self, markers):
            markers["exam_id"] = self.exam_id
            markers["video_id"] = self.video_id
            report_path = os.path.join(self.report_path, self.video_id + ".json")
117         if os.path.exists(report_path):
                print("WARNING: overwriting file")
                os.remove(report_path)
            with open(report_path, "x") as outfile:
                json.dump(markers, outfile, indent=4)
122
        # Defines data reporting function.
        def create_report(self):
            markers = self.get_candidates()
            # self.get_frames(markers)
127         # angles = self.get_angles(markers)
            self.dump_json(markers)
```