UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Aplicación software para la determinación de altura de salto deportivo vertical

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR/A: Pellicer Coves, Bàrbara

Tutor/a: Mossi García, José Manuel

CURSO ACADÉMICO: 2023/2024

Resumen

El salto vertical es una prueba física muy utilizada en diversos ámbitos, tanto deportivos como laborales. Se emplea para conocer la potencia del tren inferior del individuo.

El objetivo de este proyecto es desarrollar un software para analizar vídeos de salto vertical de un/a deportista realizado junto a una cinta métrica y determinar automáticamente la altura alcanzada. Para ello, se debe analizar la secuencia de imágenes para localizar el instante de máxima altura alcanzada. Se reconocerá automáticamente la altura del salto mediante la utilización de OCR. El desarrollo de este proyecto se ha llevado a cabo en dos fases: una primera fase de creación de un código con la herramienta MATLAB, y una segunda fase en la que se realiza una adaptación del código mediante el lenguaje Python para transformarlo en una aplicación para dispositivos móviles en un futuro, donde solo sería necesaria la utilización de una cinta métrica y un móvil con su cámara para la grabación de los saltos.

Palabras clave: salto vertical, medida, altura, automática, vídeo, cinta métrica, programa.

Resum

El bot vertical és una prova física molt utilitzada en diversos àmbits, tant esportius com laborals. S'utilitza per conèixer la potència del tren inferior de l'individu.

L'objectiu d'aquest projecte és desenvolupar un programa per analitzar vídeos de bot vertical d'un/a esportista realitzat al costat d'una cinta mètrica i determinar automàticament l'alçada assolida. Per a això, s'ha d'analitzar la seqüència d'imatges per localitzar l'instant de màxima alçada assolida. Es reconeixerà automàticament l'alçada del bot mitjançant la utilització d'OCR. El desenvolupament d'aquest projecte s'ha dut a terme en dues fases: una primera fase de creació d'un codi amb l'eina MATLAB, i una segona fase en la qual es realitza una adaptació del codi mitjançant el llenguatge Python per transformar-lo en una aplicació per a dispositius mòbils en un futur, on només seria necessària la utilització d'una cinta mètrica i un mòbil amb la seua càmera per a la gravació dels bots.

Paraules clau: bot vertical, mesura, alçada, automàtica, vídeo, cinta mètrica, programa.

Abstract

The vertical jump is a physical test widely used in various areas, such as in sport and work. It is used to know the power of the individual's lower body.

The aim of this project is to develop software to analyse videos of an athlete's vertical jump made next to a tape measure and automatically determine the height reached. To do this, the sequence of images must be analysed to locate the moment of maximum height reached. The jump height will be automatically recognised by using OCR. The development of this project has been carried out in two phases: a first phase of creation of a code with the MATLAB tool, and a second phase

in which an adaptation of the code is made using the Python language to transform it into an application for mobile devices in the future, where it would only be necessary to use a tape measure and a mobile phone with its camera to record the jumps.

Keywords: vertical jump, measurement, height, automatic, video, tape measure, program.

RESUMEN EJECUTIVO

La memoria del TFG del Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación debe desarrollar en el texto los siguientes conceptos, debidamente justificados y discutidos, centrados en el ámbito de la ingeniería de telecomunicación

CONCEPT (ABET)	**CONCEPTO (traducción)**	**¿Cumple? (S/N)**	**¿Dónde? (páginas)**
1. IDENTIFY:	1. IDENTIFICAR:		
1.1. Problem statement and opportunity	1.1. Planteamiento del problema y oportunidad	S	1-2
1.2. Constraints (standards, codes, needs, requirements & specifications)	1.2. Toma en consideración de los condicionantes (normas técnicas y regulación, necesidades, requisitos y especificaciones)	S	4-6, 33
1.3. Setting of goals	1.3. Establecimiento de objetivos	S	3
2. FORMULATE:	2. FORMULAR:		
2.1. Creative solution generation (analysis)	2.1. Generación de soluciones creativas (análisis)	S	9-11
2.2. Evaluation of multiple solutions and decision-making (synthesis)	2.2. Evaluación de múltiples soluciones y toma de decisiones (síntesis)	S	9-11, 23-26
3. SOLVE:	3. RESOLVER:		
3.1. Fulfilment of goals	3.1. Evaluación del cumplimiento de objetivos	S	31-34
3.2. Overall impact and significance (contributions and practical recommendations)	3.2. Evaluación del impacto global y alcance (contribuciones y recomendaciones prácticas)	S	34

Índice

Figuras

Código

Capítulo 1. Introducción

El deporte, tal como define la RAE, es una "actividad física, ejercida como juego o competición, cuya práctica supone entrenamiento y sujeción a normas" [1]. El deporte ha estado presente en todas las culturas y civilizaciones, desde la más antigua hasta la más actual.

Los beneficios de la práctica de actividades deportivas son de tres tipos, psicológicos, sociales y fisiológicos, porque mejora el bienestar, el estado físico y la salud de los practicantes [2], como ya pensaba Dudley Sargent. Sargent (1849-1924) fue un educador estadounidense pionero en la introducción de la actividad física como medicina preventiva y de rehabilitación [3]. Estudió medicina y fue contratado en 1880 por la Universidad de Harvard para enseñar entrenamiento físico [3]. Además, animaba a todo el mundo a realizar deporte, sin importar la habilidad, género o etnia [3]. Sargent observó que "más 75% de los jóvenes de [la sociedad americana] de ambos sexos tienen una mala postura" que conduce a problemas físicos [4]. Antes de trabajar en Harvard, abrió un gimnasio en la ciudad de Nueva York, donde recogía datos sobre la evolución de sus clientes y, asimismo, estudiaba cómo afectaba la actividad física en la rehabilitación de las personas enfermas [3]. Para conocer y evaluar el estado físico realizaba diferentes pruebas, entre ellas, el salto vertical, siendo el primero en describirlo en 1921 [5].

El salto vertical es un tipo de salto con gran relevancia en deportes como el baloncesto, el voleibol y el fútbol americano. No solo lo practican los deportistas, ya que se trata de una prueba física que también deben superar los aspirantes al cuerpo de policía y las Fuerzas Armadas. Este salto se utiliza para medir la potencia del tren inferior y su esfuerzo se concentra en el "tobillo, la rodilla y la cadera" [6].

La prueba consiste en medir la distancia saltada. Para conseguirlo se dispone un metro en la pared y el deportista se sitúa junto a él con el brazo y la mano estirados. Se apunta la altura en la que se encuentra la punta de los dedos. A continuación, el deportista salta lo más alto posible, anotando la altura máxima a la que ha llegado. Con la diferencia de alturas, se obtiene la distancia saltada, con la que luego se puede calcular la fuerza del tren inferior. A continuación, en la figura 1, se muestra una descripción visual del salto vertical.

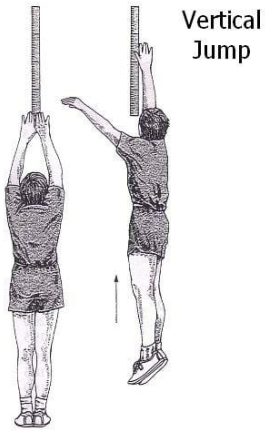


Figura 1. Descripción visual del salto vertical [7]

Para realizar estas pruebas en el ámbito profesional se dispone de aparatos especializados como Vertec® [8], figura 2. Consiste en una vara vertical con un conjunto de varillas horizontales en su parte superior, separadas cierta distancia entre ellas. La altura a la que se encuentran las varillas horizontales es regulable [8].

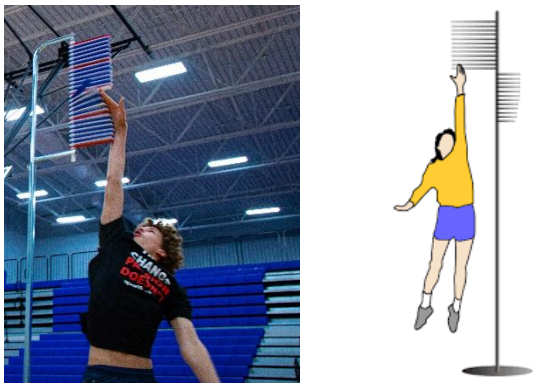


Figura 2. Vertec® [9], [5]

Otra técnica de medida de la distancia saltada es la alfombra de salto de contacto, Just Jump Mat System [10], figura 3. Midiendo el tiempo que los pies están en el aire, calcula la altura saltada [10].

Figura 3. Just Jump Mat [10]

En ambos casos, se requiere el uso de material técnico para la medición y éstos son costosos, su precio puede llegar a los $1000 (930€). Esto puede resultar un inconveniente para algunos deportistas amateurs y profesionales, asociaciones deportivas y centros educativos.

Por tanto, en este proyecto se propone la creación de una aplicación software para la medición automática de la altura del salto vertical sobre un metro en la pared. Las principales ventajas que aportaría serían el acceso económico a una herramienta de trabajo, que no requiere de equipos técnicos especializados, ya que solo sería necesario un metro y un teléfono móvil, y la gran accesibilidad al público en general.

Capítulo 2. Objetivos

2.1 Generales

El objetivo general de este proyecto es la obtención del punto más alto al que llega la mano en un salto vertical sobre un metro. A partir de la grabación de un vídeo, se realiza la extracción del fotograma con el punto más alto y su lectura sobre el metro.

Sin embargo, para llegar a alcanzar este objetivo general, hay que cumplir los distintos objetivos particulares que se describen a continuación.

2.2 Particulares

Los objetivos particulares son:

- El desarrollo del algoritmo base con MATLAB como una primera aproximación al problema aprovechando la rapidez de desarrollo que permiten sus herramientas y entorno.
- El desarrollo posterior del algoritmo con Python para estar más cerca de su posible desarrollo como aplicación.

Estos dos objetivos engloban unos objetivos básicos que se deben cumplir en ambos casos:

- La detección del ángulo de rotación de la cinta métrica.
- La detección de los bordes de la cinta.
- La utilización del OCR para el reconocimiento de caracteres.
- La detección de las líneas milimétricas y la clasificación en clases.
- La asignación de valor a la última línea para obtener el resultado.

Capítulo 3. Herramientas

Este capítulo describe las herramientas utilizadas en los diferentes procesos de este proyecto: la captación del salto y el desarrollo de la aplicación.

3.1 Captación del salto

La captación del salto consiste en la grabación del salto vertical. Para ello, se necesita una cinta métrica apoyada o pegada en la pared. Con ella, se observa la distancia a la que llega la mano durante el salto. Además, se requiere de una cámara para grabar el vídeo. Esta cámara se ha pensado para que sea de un dispositivo como un teléfono móvil o una Tablet. En este estudio, los elementos utilizados han sido:

- Cintas métricas: Flexómetro de goma con freno 25 mm – 8 m de Stein Tools. Aunque se ha realizado alguna prueba con el metro de papel 100 cm de IKEA y reglas escolares.
- Cámara: Teléfonos móviles Samsung Galaxy A53 y A55.

3.2 Desarrollo de la aplicación software

El desarrollo de la aplicación software ha constado de dos partes: el desarrollo de un programa base en MATLAB y el desarrollo de este mismo programa en Python.

3.2.1 MATLAB

MATLAB (Matrix Laboratory) es una plataforma de programación con un lenguaje basado en matrices y creado por MathWorks. Con ella se ha desarrollado el programa base. La Universitat Politècnica de València, UPV, dispone de una licencia que permite utilizar todos los productos de MATLAB, por tanto, su uso en este proyecto ha sido gratuito. Su elección se ha basado en la familiarización que ya se tenía con esta herramienta antes de empezar el proyecto y su capacidad para trabajar el procesamiento de imágenes y la visión artificial. Se muestra el icono de MATLAB en la figura 4.

Las librerías especiales utilizadas son las siguientes:

- Computer Vision Toolbox
  Esta librería tiene las funciones para el desarrollo de aplicaciones de visión artificial. Por tanto, proporciona la función del OCR (Optical Character Recognition, reconocimiento óptico de caracteres) que se usará para la lectura de los números de la cinta métrica.
  Funciones usadas: ocr.
- Image Processing Toolbox
  Esta librería se utiliza, como bien dice su nombre, para el procesamiento de imágenes.
  Funciones usadas: bwskel, edge, hough, houghlines, houghpeaks, imbinarize, imcomplement, imgaussfilt, imrotate, imshow.
- MATLAB
  Con las funciones y operaciones matemáticas.
  Funciones usadas: videoReader, hasFrame, readFrame, xor, mean, plot, maxk, length, min, imresize, im2gray, norm, find, abs, size, str2double, ismember, issorted, isequal, sum, max, ceil, round, zeros, prctile.

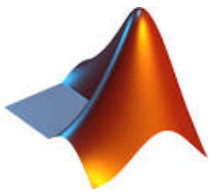


Figura 4. Icono de MATLAB [11]

3.2.2 Python

Python es un lenguaje de alto nivel orientado a objetos, desarrollado por Guido Van Rossum, de código abierto y administrado por Python Software Foundation, siendo su descarga totalmente gratuita. Si bien antes de empezar el proyecto no se había tenido demasiado contacto con este lenguaje, se ha decidido llevar a cabo el desarrollo más profesional del programa con esta herramienta, cuyo icono se muestra en la figura 5, por su amplio uso en el campo de desarrollo de aplicaciones software.

Para el desarrollo de la aplicación se ha necesitado la utilización de librerías externas. Las librerías utilizadas han sido:

- Imutils
  Contiene funciones básicas para el procesamiento de imágenes, como la rotación y la "esqueletonización".
  Funciones usadas: rotate, skeletonize.
- Matplotlib
  Esta librería se utiliza para graficar los datos. En este caso se ha utilizado tanto para puntos como imágenes.
  Funciones usadas: plt, show, subplots.
- NumPy
  Se utiliza para la computación de matrices.
  Funciones usadas: mean, degrees, round, abs, rot90, array, arange, isin, sum, min, argmin, max, argmax, uint8, zeros_like, percentile, zeros, ceil, nonzeros.
- OpenCV
  Open-Source Computer Vision es una de las librerías más grandes en procesamiento de imágenes y visión artificial.
  Funciones usadas: videoCapture, threshold, bitwise_not, bitwise_xor, imshow, resize, Canny, GaussianBlur, HoughLinesP, line, shape, cvtColor, rectangle.
- Pillow (Python Imaging Library)
  Tiene funciones para el procesamiento de imágenes.
  Funciones usadas en las pruebas iniciales, no en la versión final.
- Pytesseract
  Se trata de una interfaz para el Tesseract-OCR, es decir, para la herramienta de lectura de texto en imágenes.
  Funciones usadas: Image_to_string, image_to_data.
- Skimage
  Esta librería recoge funciones para el procesamiento de imágenes.
  Funciones usadas: hough_line, hough_line_peak.

Se han utilizado también los paquetes:

- Math
  El paquete proporciona funciones matemáticas.
  Funciones usadas: ceil.
- Re
  Da acceso a operaciones con expresiones regulares similares a las de Perl (lenguaje de programación enfocado a la manipulación de texto).
  Funciones usadas: findall, match.

Figura 5. Icono de Python [12]

El editor de código fuente utilizado para Python ha sido el Visual Studio Code (VS Code). Este editor fue desarrollado por Microsoft. Permite la programación con distintos lenguajes, como Python, Java y C/C++. Además, es gratuito y compatible con distintos sistemas operativos. Su icono se muestra en la figura 6.

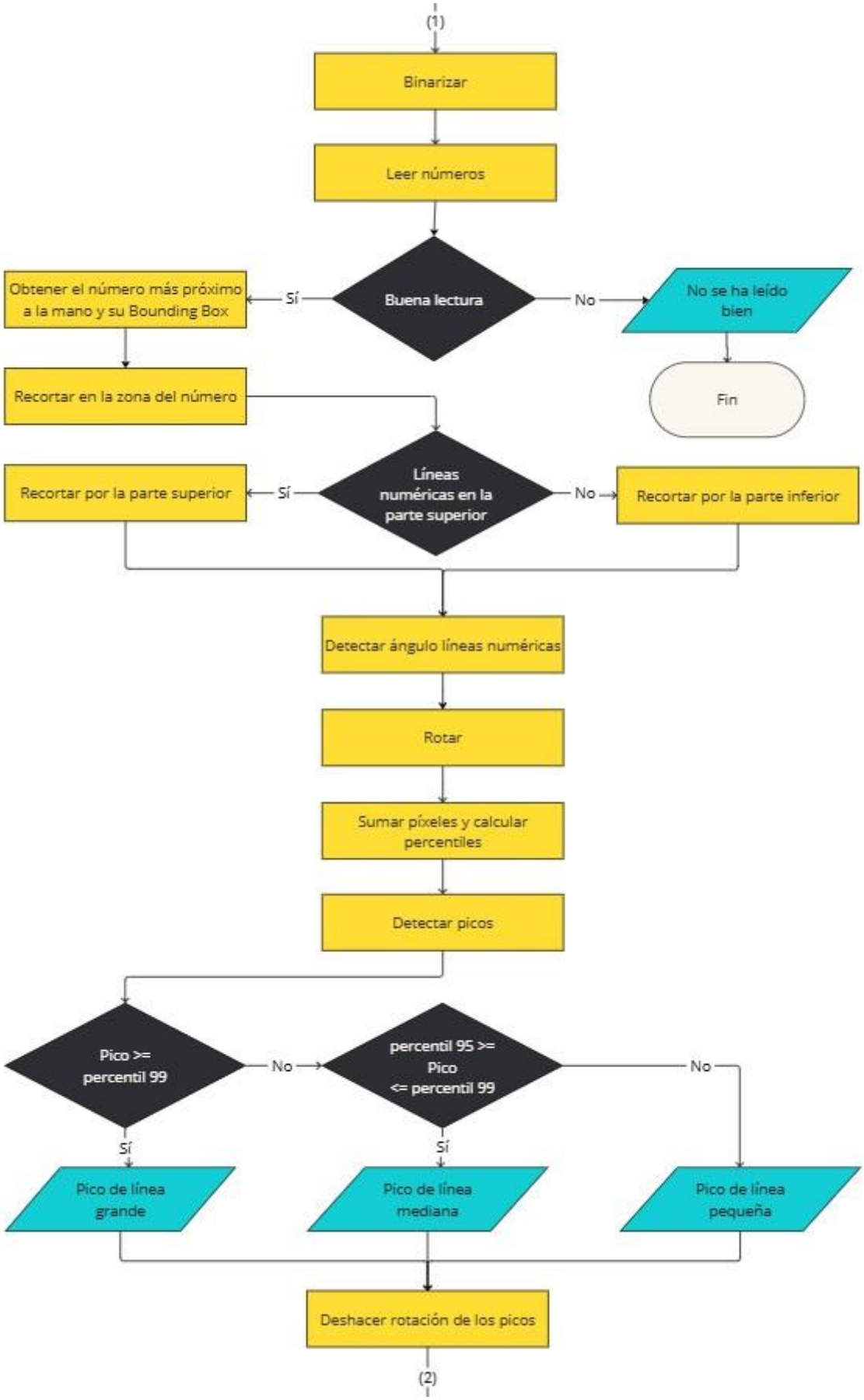
Figura 6. Icono de Visual Studio Code [13]

Capítulo 4. Desarrollo

En este capítulo se detalla el proceso seguido en el desarrollo de la aplicación, desde la lectura del vídeo hasta la obtención del resultado final. Consta de dos partes: la realización de un primer programa en MATLAB como base y el paso del código a Python para un desarrollo que está un paso más cerca de convertir el algoritmo en aplicación.

4.1 Diagrama de bloques

A continuación, en la figura 7, se presenta un diagrama de flujo como esquema general que sigue el programa. Este diagrama se ha realizado mediante la plataforma de colaboración visual Miro, la cual cuenta con una herramienta de diagramas online.


(1)
Binarizar
Leer números
Buena lectura
Sí
No
Obtener el número más próximo a la mano y su Bounding Box
No se ha leído bien
Fin
Recortar en la zona del número
Líneas numéricas en la parte superior
Sí
No
Recortar por la parte superior
Recortar por la parte inferior
Detectar ángulo líneas numéricas
Rotar
Sumar píxeles y calcular percentiles
Detectar picos
Pico >= percentil 99
No
percentil 95 >= Pico <= percentil 99
No
Sí
Sí
Pico de línea grande
Pico de línea mediana
Pico de línea pequeña
Deshacer rotación de los picos
(2)

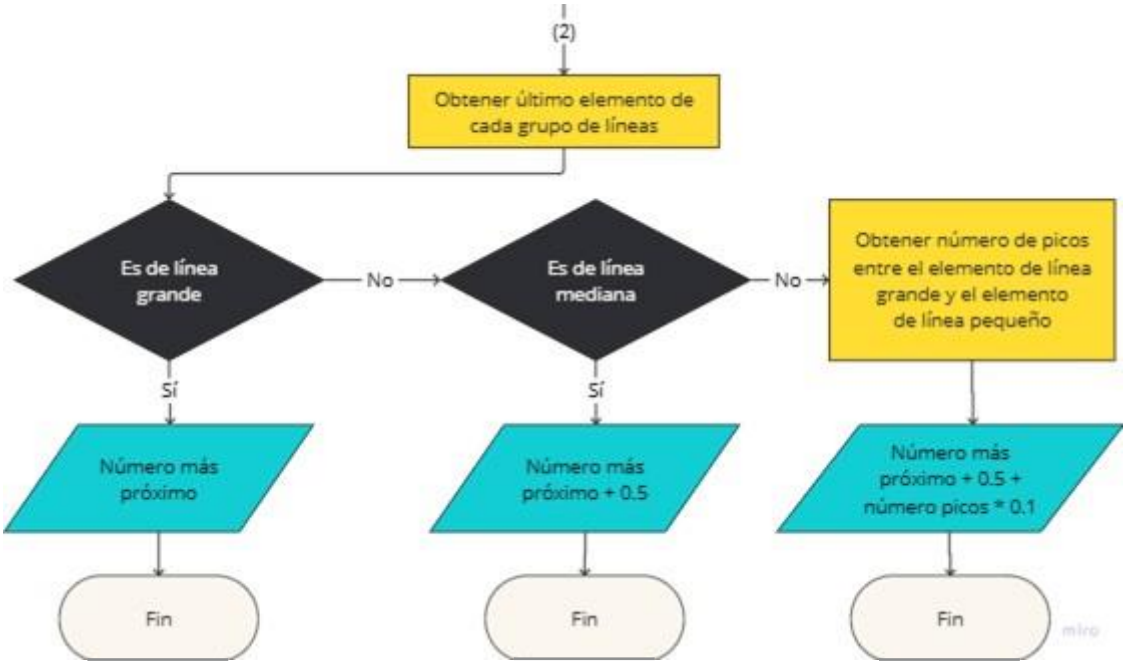


Figura 7. Diagrama de bloques que sigue el código [Fuente propia]

4.2 Desarrollo del código con MATLAB

El primer esbozo del código se ha diseñado en MATLAB. En los siguientes puntos se detalla su funcionamiento siguiendo el orden de ejecución del programa.

4.2.1 Determinación del fotograma con el punto más alto

La detección del fotograma que contiene el punto más alto al que llega la mano es el primer objetivo del programa y se realiza tras la lectura del vídeo.

La búsqueda de este fotograma se ha probado de tres maneras distintas: detección de objetos, sustracción de fotogramas consecutivos y sustracción de fotogramas con un fotograma de referencia.

Con la primera opción, se binarizan los fotogramas del vídeo y para cada uno se debe encontrar el objeto (la mano, al menos). Para ello, se necesita aplicar aperturas con elementos estructurantes y analizar las propiedades del objeto. Después, analizar el Bounding Box (caja rectangular que encierra el objeto, cuadro delimitador mínimo) y su altura mínima, para comparar fotograma a fotograma cuál tiene esa altura mínima, que corresponde con el número de fila menor, y que, por tanto, este sería el fotograma buscado.

La segunda opción se basa en la velocidad y movimiento de la mano. Antes de saltar, no hay movimiento, por tanto, tampoco velocidad. Cuando se está saltando hay movimiento, pero cuando llega al punto más alto, la velocidad es cero y el movimiento nulo. Al caer, vuelve a haber velocidad, por lo que también hay movimiento. Figura 8.

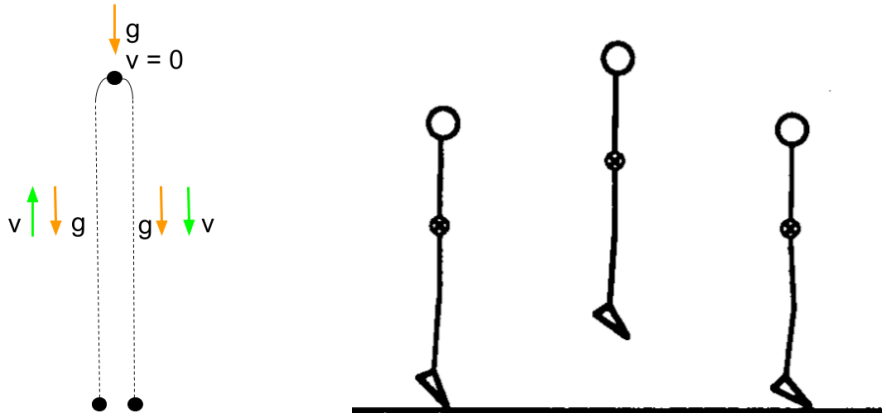


Figura 8. Velocidad del salto vertical [14] y representación del salto [15]

Así, después de binarizar los fotogramas, se calcula la diferencia entre ellos, se calcula la media de esa diferencia y se almacena en un vector que contendrá todas las medias. Cuando termina de calcular todas las diferencias y sus medias, busca los primeros máximos. Teniendo en cuenta que, aproximadamente, el fotograma que contiene el punto más alto está por la mitad del vídeo, se extrae un pico en la primera mitad (sin tener en cuenta el valor máximo que corresponde al inicio del vídeo) y otro en la segunda mitad. Finalmente, el fotograma buscado es el que tiene la media mínima entre ambos, como se observa en la figura 9, donde los picos se encuentran en los fotogramas 38 y 70 y el mínimo entre ambos en el 55, el fotograma con el punto más alto al que se llega.

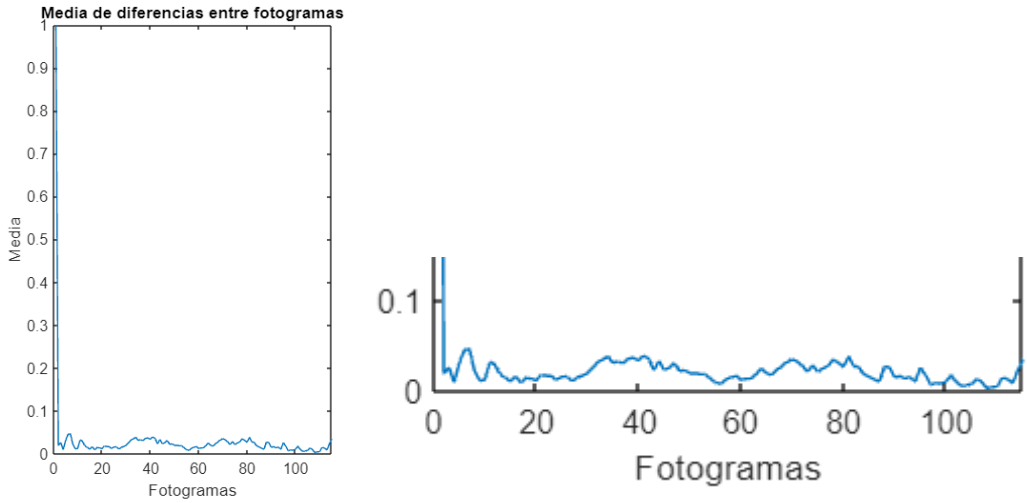


Figura 9. Medias de las diferencias entre fotogramas por fotograma (izqda.) y ampliación de dicha gráfica (dcha.) [Fuente propia]

El tercer proceso es muy similar al anterior, con la diferencia de que en este, la sustracción se realiza entre el fotograma actual y un fotograma de referencia, que suele ser el primero. De este modo, buscamos la media máxima (de nuevo, sin tener en cuenta el valor del primer fotograma), resultando una gráfica como la descrita en la figura 10, donde el valor máximo se encuentra en el fotograma 55.

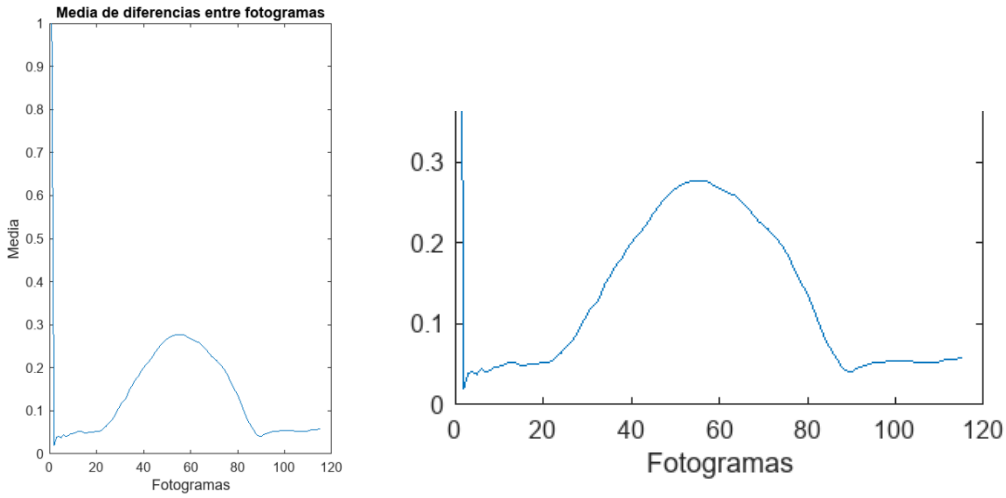


Figura 10. Medias de las diferencias entre fotograma actual y el de referencia (izqda.) y ampliación de dicha gráfica (dcha.) [Fuente propia]

En las siguientes imágenes, figura 11, se muestran los tres resultados para el mismo vídeo.

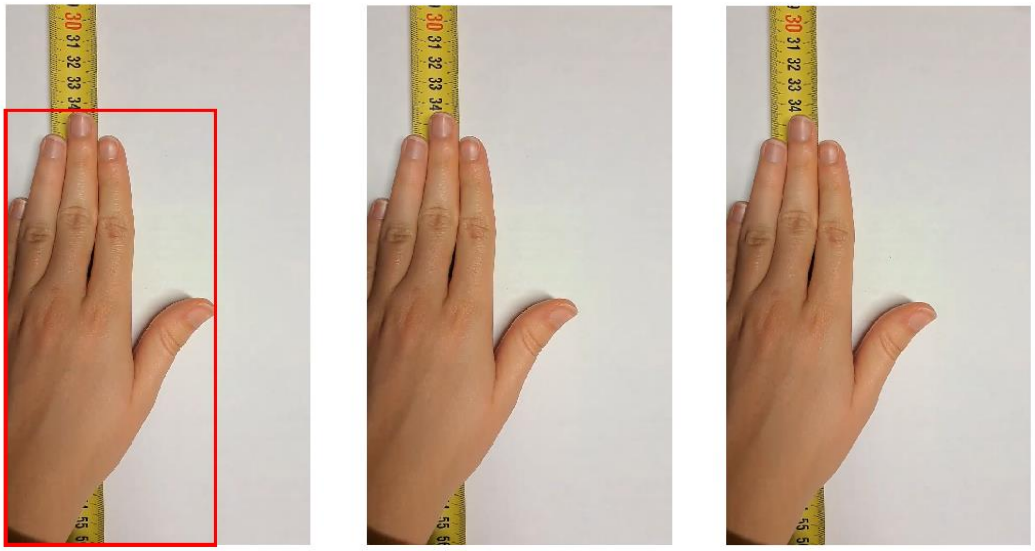


Figura 11. Fotograma con el punto más alto detectado mediante detección de objetos (izqda.) y diferencia de fotogramas consecutivos (centro) y diferencia con un fotograma de referencia (dcha.) [Fuente propia]

Como se observa, el resultado es el mismo. No obstante, tras probar los tres procesos en distintos vídeos, se decide continuar el proyecto con la detección del fotograma mediante la diferencia respecto a un fotograma de referencia. Los motivos que llevan a esta decisión son los siguientes:

- El proceso de detección de objetos, aunque en términos de cantidad de líneas de código ha sido similar a los otros dos, requiere más procesamiento de imagen y análisis por lo que resulta un proceso más lento.
- La obtención del fotograma mediante sustracción de imágenes, tanto en las consecutivas como en la de referencia, es más robusto, ya que no depende de que el objeto cumpla las condiciones pedidas.
- Realizadas las pruebas con los dos procesos anteriores, se ha observado que la detección del fotograma mediante sustracción de imágenes con una de referencia obtiene mejores resultados.

4.2.2 Detección de los bordes de la cinta métrica

Primero, se reduce el tamaño de la imagen para que el proceso sea más rápido, ya que se reduce el número de píxeles que hay que recorrer. En la detección de los bordes se ha utilizado la transformada de Hough, que permite detectar rectas.

El funcionamiento de la transformada de Hough se basa en la parametrización de una recta con las coordenadas polares de la recta perpendicular a ésta que pasa por el origen de coordenadas (4.1) [16].

$$\boldsymbol{\rho = x * cos(\theta) + y * sin(\theta)} \qquad \textbf{(4.1)}$$

Θ representa el ángulo de la perpendicular con el origen, mientras ρ es la distancia desde el origen hasta la recta, figura 12.

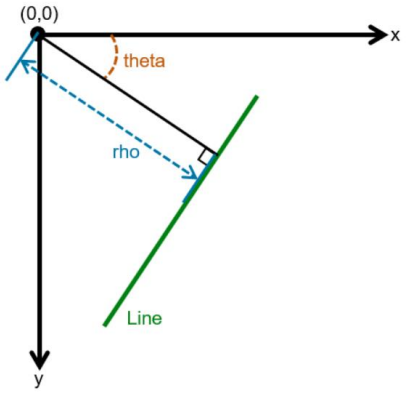


Figura 12. Representación de las variables de Hough [17]

Un punto (x,y) de la imagen es una sinusoide (Θ,ρ) en el espacio de Hough. En cambio, una recta en la imagen es un punto en la transformada. Este punto corresponde con una intersección de ondas sinusoidales [16], se muestra en la figura 13.

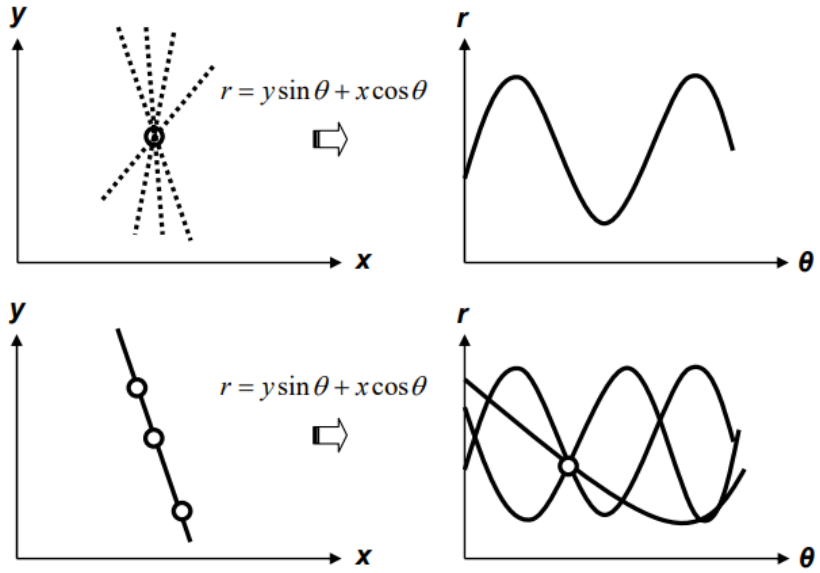


Figura 13. Representación de los puntos y rectas en la imagen (izqda..) y en el espacio de Hough (dcha.) [16]

El espacio de Hough se representa mediante una matriz inicializada a cero. En la imagen se buscan los píxeles marcados como bordes, y para cada uno, se resuelve la ecuación (4.1), creando una

sinusoide. Por cada valor que le corresponda en el espacio de Hough, se incrementa en uno, lo que indica la cantidad de píxeles que corresponden a una misma recta. Finalmente, se dejan solo los que cumplen los requisitos definidos en la entrada. Como se observa es un método robusto, ya que los puntos de ruido no influyen y las rectas con puntos ausentes pueden ser identificadas siempre que haya suficientes puntos para calcularla [18].

La entrada a esta función de Hough es una imagen binaria donde están identificados los bordes del metro, por lo que se debe convertir a escala de grises, filtrar y binarizar la imagen. En este caso se ha filtrado con 'imgaussfilt', un filtro gaussiano 2D para reducir el ruido y se ha binarizado con 'edge' y su método 'Canny', que devuelve la imagen binaria con unos (1) en los bordes detectados mediante máximos locales del gradiente de la imagen, como se observa en la figura 14 (centro).

Se aplica la transformada de Hough, como se observa en el extracto de código 1, donde 'BW' es la imagen de entrada de 'hough' en la que se buscan las rectas, imagen binaria resultante del proceso de filtrado y binarización. La función 'houghpeaks' devuleve una matriz con las coordenadas de los tres picos donde más sinusoides han intersectado y la función 'houghlines' extrae los segmentos de rectas, como se observa en la figura 14 (derecha).

Figura 14. Imagen antes del procesado (izqda.), imagen filtrada y binarizada (centro) e imagen con los bordes marcados (drcha.) [Fuente propia]

```
% Detecció marges de la regla
BW = edge(imgaussfilt(im2gray(imA3)),'canny');
figure, imshow(BW)


[H2,T2,R2] = hough(BW, 'RhoResolution',0.5,'Theta',[-20:0.1:20]);
P2  = houghpeaks(H2,3,'threshold',ceil(0.1*max(H2(:))))

lines = houghlines(BW,T2,R2,P2,'FillGap',5,'MinLength',7);
```

Código 1. Detección de bordes con la Transformada de Hough

Los márgenes se guardan para su futuro uso como altura mínima y máxima y anchura mínima y máxima, lo que equivale a un rectángulo.

Sin embargo, al probar con distintas imágenes tanto con la mano como sin ella, en algunas surgen problemas con la detección de los bordes. En algunos casos los márgenes de la cinta métrica se cogen únicamente de una de las líneas encontradas ya que se considera que es la única que cumple las condiciones lo que provoca que al recortar la imagen quede solo esta línea. Esto dificulta la lectura de los números, que se soluciona buscando dentro de las líneas detectadas otra línea que realmente cumpla con lo que se busca. Es decir, se debe buscar una línea cuyo ángulo sea similar y que esté a cierta distancia de la encontrada, como en la figura 15.

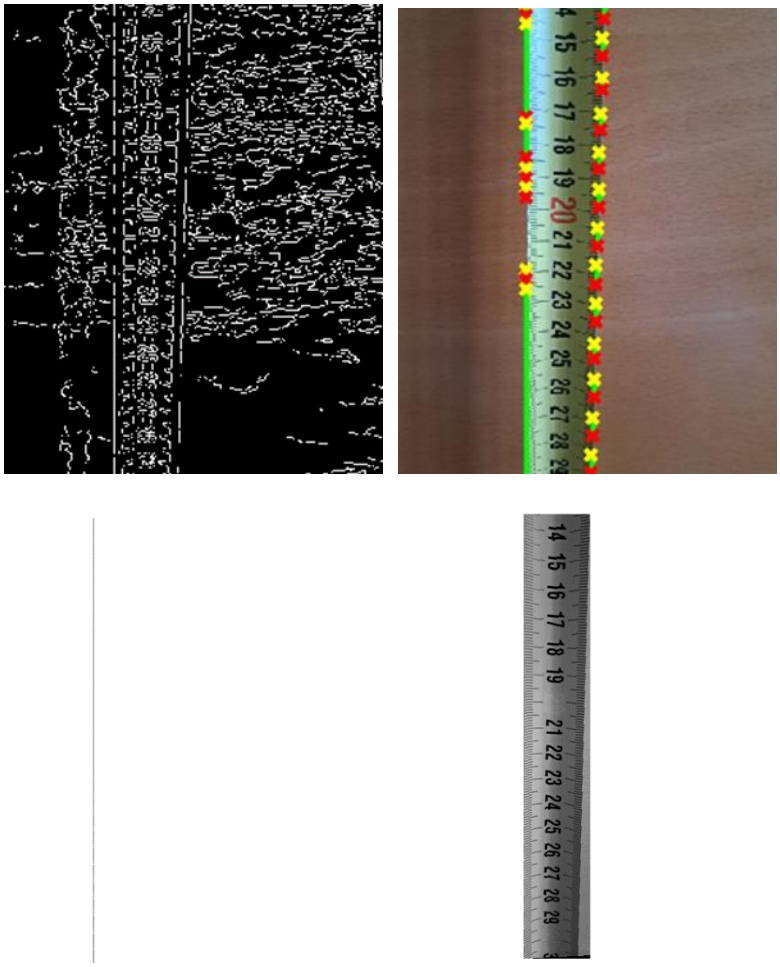


Figura 15. Entrada a la transformada de Hough (arriba, izqda.), detección de líneas (arriba, dcha.), recorte por las primeras esquinas encontradas (abajo, izqda.) y recorte después de aplicar las soluciones (abajo, dcha.) [Fuente propia]

De todos modos, antes de recortar la imagen, se realizan unas comprobaciones de tamaño, donde se termina de perfilar el valor de los puntos de los bordes.

Con los márgenes calculados, se trabaja con la imagen original, por lo que las cuatro esquinas que marcan los bordes se transforman para su equivalente en la original (se debe recordar que la imagen había sido reducida de tamaño para agilizar el proceso), y se recorta; como se muestra en código 2.

```
im_regla_og =
imA_00(round(alt_min/0.125):round(alt_max/0.125),round(anch_min/0.125):round(anch_max/0.125));
```

Código 2. Recortar imagen original

4.2.3 Lectura mediante OCR

El primer paso ha sido la comprobación del funcionamiento de la función OCR. Para ello, se han hecho fotografías de distintos tipos de reglas con diferentes orientaciones (vertical y horizontal, pero siempre alineadas con la cámara) y distintos tipos de letra. Aunque no es perfecto, funciona bien en general, como se observa en la figura 16.

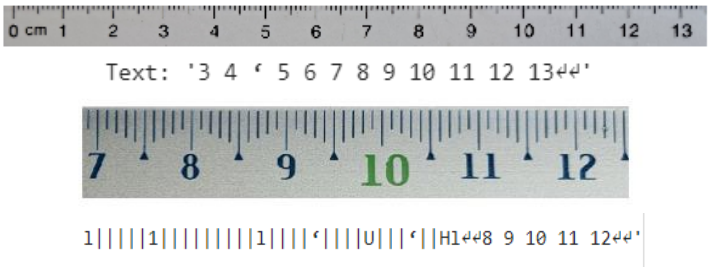


Figura 16. Ejemplos de lecturas del OCR de MATLAB [Fuente propia]

Con la comprobación del buen funcionamiento del OCR de MATLAB, se le aplica de nuevo a la imagen una rotación, ya que como los metros están posicionados verticalmente, sus números están girados, y se debe colocar el metro en horizontal para la correcta lectura de los números. Seguidamente, se realiza dicha lectura de mediante el OCR.

Al probar con distintas imágenes, se ha observado como en algunas no se leía ningún número a pesar de que en la imagen los números parecen ser bastante leíbles. Por tanto, se han planteado soluciones y la solución tomada ha sido aumentar el contraste entre los números y el fondo. Con la ejecución del código se ha observado como se solucionan los problemas de lectura, figuras 17 y 18.

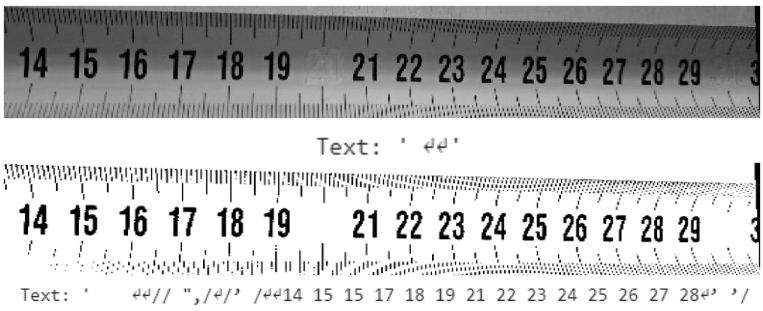


Figura 17. Lectura sin mano antes de binarizar (arriba) y después (abajo) [Fuente propia]

Figura 18. Lectura con mano antes de binarizar (izqda.) y después (dcha.) [Fuente propia]

Entonces, esta solución se consigue binarizando la imagen. Este proceso se ha realizado con unos umbrales que dependen de la media de la imagen. Así, si la imagen de entrada tiene sombras, su media es menor y el umbral que se le asigna es menor, y viceversa. A continuación, como se muestra en el código 3, para obtener la imagen binarizada (‘imagenBW’), se le aplica a la imagen (‘im_regla_h’) la función ‘imbinarize’ que binariza la imagen con el umbral localmente adaptativo (lo calcula la función, aunque denominado igual en el código, no es el nombrado

anteriormente), el 'ForegroundPolarity dark' que indica que el primer plano, los números en este caso, son más oscuros que el fondo, y la 'Sensitivity' que es el factor de sensibilidad que tiene el umbral adaptativo (entre 0 y 1). Los factores elegidos, aquí sí, el umbral asignado según la media, para la binarización no son muy altos, debido a que, a mayor valor, más píxeles serían considerados de primer plano y podría conllevar a que píxeles de fondo fuesen confundidos por píxeles de primer plano.

```
mitja = mean(im_regla_h(:))


if mitja < 115
    umbral = 0.3
elseif mitja >= 115 && mitja <= 150
    umbral = 0.4
else
    umbral = 0.5
end



imagenBW = imbinarize(im_regla_h,'adaptive','ForegroundPolarity','dark','Sensitivity',umbral);
imshow(imagenBW)
```

Código 3. Binarización previa a la lectura de los números

Una vez hecha la lectura, se debe comprobar que haya sido buena, es decir, que los números leídos se correspondan a la realidad. Se crea un vector de números con el que se compara el vector de números leídos con el OCR. Así se sabe cuáles son los correctos, código 4. Además, se comprueba que los números dados por correctos estén en orden ascendente, que no tenga números repetidos o no consecutivos. De este modo, se crea un vector con los valores desde el primer leído hasta el último, donde los correctos tienen su valor y los que no, se identifican como número erróneo. Después de volver a compararlos con el vector de números, se obtiene el número de coincidentes. Si son tres o más, se da por buena la lectura. En este momento, si la lectura es fallida, se notifica y se termina la ejecución del código.

```
[elem1,~] = ismember(leidos,numeros) % Indica els elements i localitzacions dels
números coincidents en els dos vectors
coincidentes = sum(elem1)
buenaLectura = (coincidentes>=3) % Donar per bona la lectura si els números
coincidents són 3 o més
```

Código 4. Obtención de números correctos y validación de lectura

4.2.4 Detección y clasificación de las líneas milimétricas

La detección de las líneas milimétricas del metro es uno de los procesos más importantes y se basa en la suma de píxeles de la imagen por columnas. Sin embargo, antes de realizar la suma, se debe determinar el número más próximo a la mano. Cuando la lectura es dada por buena, el último número leído es el más próximo a la mano. Por tanto, ya solo queda recortar la imagen alrededor de este número para la detección de las líneas milimétricas.

En los resultados del OCR de MATLAB, se proporcionan los Bounding Boxes de los números leídos, por lo que se debe buscar este Bounding Box, que indicará por donde se debe recortar la imagen. El Bounding Box es un vector de cuatro valores: la esquina superior izquierda (x,y), la anchura del cuadro y su altura. Por tanto, se recorta la imagen por las anchuras (eje X), respetando la altura para no cortar las líneas milimétricas.

El siguiente paso es recortar la imagen para mostrar únicamente las líneas, ya que para obtener buenos resultados en las sumas, interesa que sea lo único que se muestre. Se busca, con suma

valores de los píxeles por filas, la fila del borde de la cinta métrica, que será cero o casi cero debido a que es una fila de píxeles negros, y una fila blanca entre la línea más larga y el número, que tendrá un valor alto por su presencia de unos. Así se recorta la imagen por estas dos filas delimitadoras y solo quedan en la imagen las líneas. Figura 19.

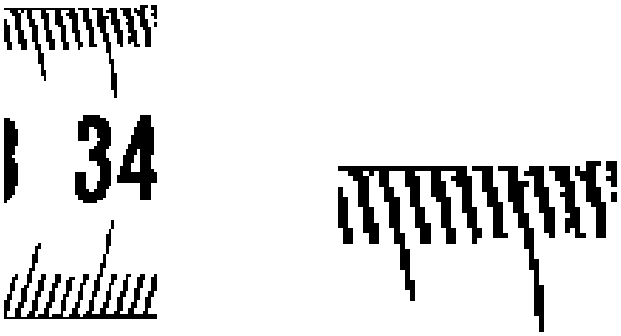


Figura 19. Recorte por los valores del eje X del Bounding Box (izqda.) y recorte únicamente con las líneas milimétricas (dcha.) [Fuente propia]

El siguiente paso ha sido 'esqueletonizar' la imagen para que los objetos, en este caso las líneas, se queden en sus 'huesos', como una línea muy fina. Este proceso ayuda a la detección del ángulo de inclinación de estas líneas y la rotación de la imagen, ya que se desea que estas líneas estén totalmente rectas para sumar sus píxeles por columnas (o eje X de la imagen) y obtener la forma de las líneas, como se muestra en la figura 20. Además, la imagen rotada se recorta ligeramente por los lados para evitar picos falsos en la suma. Aquí MATLAB ha generado algunos problemas a la hora de obtener el resultado final, no obstante, se ha solucionado realizando el recorte dos veces, con distintos valores. Como se recoge en el apartado 4.2.5 (Obtención de resultados).

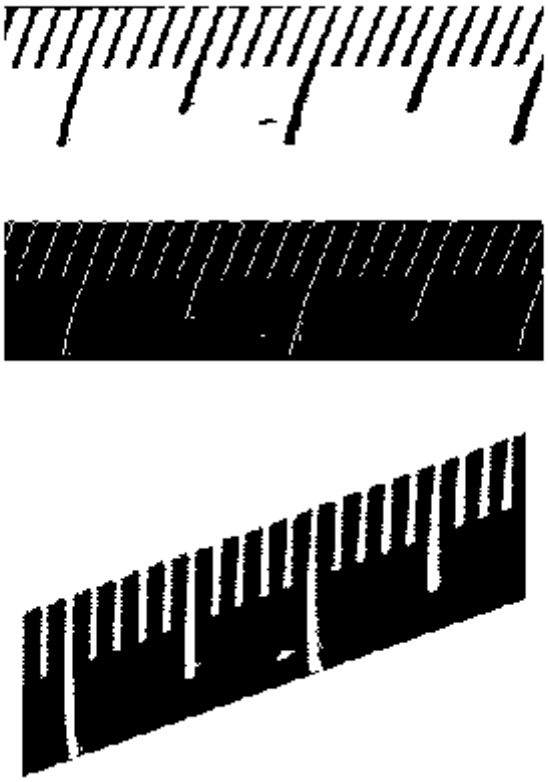


Figura 20. Segmento de imagen con líneas milimétricas (arriba.), 'esqueletonización' de la imagen (centro) y rotación y recorte (abajo.) [Fuente propia]

Se ha elegido utilizar el complemento de la imagen para pasar los píxeles blancos a negros y los negros a blancos. De ese modo, las líneas milimétricas quedan marcadas en blanco. Al tratarse de una imagen lógica, los píxeles negros valen 0, mientras que los blancos valen 1. La suma así es más clara, ya que las líneas son máximos y simulan las líneas en realidad, como se muestra en la figura 21.

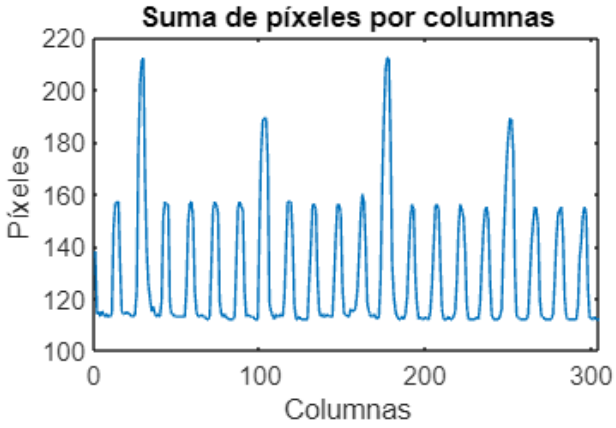


Figura 21. Suma de píxeles por columnas [Fuente propia]

A continuación, se buscan los máximos locales. Cada punto es el último píxel perteneciente a una línea. Este proceso se ha llevado a cabo mediante un bucle 'for' y buscando el máximo en áreas de tamaños condicionados (tres píxeles a izquierda y derecha) y condicionando que éste máximo sea mayor a la media las sumas, como se muestra en el código 5. El vector con los resultados de las sumas es 'c' y la media 'media'. Los máximos locales están representados por 'puntosmax' y 'posiciones', siendo los valores y las posiciones, respectivamente.

```
media = mean(c)
% Creem i inicialitzem variables
puntosmax = zeros(size(c));
posiciones = zeros(size(c));


plot(c)
title('Máximos locales')
xlabel('Columnas')
ylabel('Píxeles')
hold on
for m = 1:length(c)

    if c(m) > media
        if m==1 || m==2 || m==3 || m==length(c)-2 || m==length(c)-1 || m==length(c)
            % Esta condició és per a que no done error la següent
        elseif c(m+1)>=c(m) || c(m+2)>=c(m) || c(m+3)>=c(m) || c(m-1)>c(m) || c(m-2)>c(m) ||
c(m-3)>c(m)
            % Si hi ha prop un pic més alt (un resultat de suma major),
            % no pintarà l'actual
        else
            puntosmax(m) = c(m); % Valor màxim en este tros
            posiciones(m) = m; % POsició d'este màxim
```

```
            plot(posiciones(m),puntosmax(m),'r*')
        end
    end
end

hold off
```

Código 5. Máximos locales

A continuación, se deben clasificar los puntos en uno de los tres grupos posibles: líneas grandes, líneas medianas y líneas pequeñas. Denominamos líneas grandes a aquellas líneas que se corresponden con los números enteros (1, 2, 3, …), líneas medianas a los medios (1’5, 2’5, …) y líneas pequeñas al resto (1’1, 1’2, 1’3, 1’4, 1’6, 1’7, 1’8, 1’9, 2’1, …). Esta clasificación se ha realizado comparando el valor del punto con distintos percentiles, en este caso el 95 y el 98, como se ve en el extracto de código siguiente, código 6, donde los percentiles llevan el nombre de ‘percentil95’ y ‘percentil98’ refiriéndose al percentil que su propio nombre indica, y ‘lin_pequenas’, ‘lin_medianas’ y ‘lin_grandes’ los vectores que indican el tipo de línea con sus puntos correspondientes.

```
% Calculem el percentil per a separar els diferents tipus de ratlles
percentil98 = prctile(c,98)
percentil95 = prctile(c,95)


lin_pequenas = zeros(2,length(c)); % Fila 1: posición (x) - Fila 2: valor (y)
lin_medianas = zeros(2,length(c)); % Fila 1: posición (x) - Fila 2: valor (y)
lin_grandes = zeros(2,length(c));  % Fila 1: posición (x) - Fila 2: valor (y)


plot(c)
title('Clasificación de puntos')
xlabel('Columnas')
ylabel('Píxeles')
hold on
for m = 1:length(puntosmax)
    if puntosmax(m) >= percentil98
        lin_grandes(:,m) = [m;puntosmax(m)];
        plot(lin_grandes(1,m),lin_grandes(2,m),'r*')
    elseif puntosmax(m) >= percentil95 && puntosmax(m) <= percentil98
        lin_medianas(:,m) = [m;puntosmax(m)];
        plot(lin_medianas(1,m),lin_medianas(2,m),'g*')
    elseif puntosmax(m) < percentil95 && puntosmax(m) > 1
        lin_pequenas(:,m) = [m;puntosmax(m)];
        plot(lin_pequenas(1,m),lin_pequenas(2,m),'m*')
    end
end
hold off
```

Código 6. Clasificación de los máximos locales

Seguidamente, en la figura 22, se observan las gráficas resultantes de la ejecución de los códigos 5 y 6.

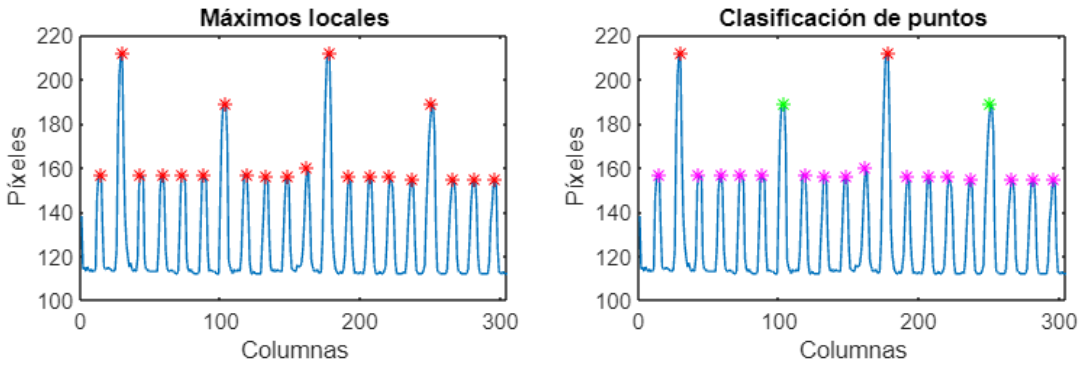


Figura 22. Máximos locales (izqda.) y su clasificación (dcha.) [Fuente propia]

Ahora, se debe deshacer la rotación que se había realizado para obtener los puntos que marcan las líneas para conseguir estos puntos en la imagen original. Los nuevos puntos se consiguen mediante la multiplicación por la matriz de rotación (4.2), donde φ es el ángulo de rotación. En este caso, como es el proceso inverso al que se había realizado, será este ángulo en negativo (-φ).

$$\mathbf{rotMat} = \begin{pmatrix} \boldsymbol{cos(\varphi)} & \boldsymbol{sin(\varphi)} \\ \boldsymbol{-sin(\varphi)} & \boldsymbol{cos(\varphi)} \end{pmatrix} \quad \textbf{(4.2)}$$

Como el centro de rotación está situado en el centro de la imagen y de los vectores, primero, se les resta a los puntos actuales el centro de la imagen actual. Luego, se multiplican por la matriz de rotación, y finalmente, se les suma el centro de la nueva imagen. Como se muestra en código 7, los puntos a rotar son los de los vectores 'lin_pequenas', 'lin_medianas' y 'lin_grandes'; los obtenidos por la rotación 'lin_pequenas_inv', 'lin_medianas_inv' y 'lin_grandes_inv'; y los centros son 'esteCentro' para la imagen actual y 'cc' para la imagen original.

En este caso, solo se ha usado la componente X (posición de los puntos en el eje X) en la multiplicación. Esto se debe a que no es necesaria la otra componente, ya que para la obtención del punto más cercano a la mano solo es necesario saber la posición en X y al grupo al que pertenecen los puntos. En todo caso, si se quisiera observar estos puntos sobre toda la imagen, solo haría falta multiplicar por toda la matriz de rotación en lugar de únicamente por la componente X. Figura 23.

```
lin_pequenas_centrat = zeros(size(lin_pequenas));
lin_medianas_centrat = zeros(size(lin_medianas));
lin_grandes_centrat = zeros(size(lin_grandes));
for m = 1:length(lin_pequenas)
    if lin_pequenas(1,m) ~= 0
        lin_pequenas_centrat(:,m) = lin_pequenas(:,m) - esteCentro;
    elseif lin_medianas(1,m) ~= 0
        lin_medianas_centrat(:,m) = lin_medianas(:,m) - esteCentro;
    elseif lin_grandes(1,m) ~= 0
        lin_grandes_centrat(:,m) = lin_grandes(:,m) - esteCentro;
    end
end
lin_pequenas_inv_rotat = zeros(size(lin_pequenas_centrat));
lin_medianas_inv_rotat = zeros(size(lin_medianas_centrat));
lin_grandes_inv_rotat = zeros(size(lin_grandes_centrat));
for m = 1:length(lin_pequenas_centrat)

%     Per a que es veja en la imatge sols de ratlletes:
```

```
    lin_pequenas_inv_rotat(:,m) = [rotMatInv(1,1),0;rotMatInv(2,1),0]*lin_pequenas_centrat(:,m);
    lin_medianas_inv_rotat(:,m) = [rotMatInv(1,1),0;rotMatInv(2,1),0]*lin_medianas_centrat(:,m);
    lin_grandes_inv_rotat(:,m) = [rotMatInv(1,1),0;rotMatInv(2,1),0]*lin_grandes_centrat(:,m);
end
lin_pequenas_inv = zeros(size(lin_pequenas_inv_rotat));
lin_medianas_inv = zeros(size(lin_medianas_inv_rotat));
lin_grandes_inv = zeros(size(lin_grandes_inv_rotat));
for m = 1:length(lin_pequenas_inv_rotat)
    if lin_pequenas_inv_rotat(1,m) ~= 0
        lin_pequenas_inv(:,m) = round(lin_pequenas_inv_rotat(:,m) + [cc(2);cc(1)]);
    elseif lin_medianas_inv_rotat(1,m) ~= 0
        lin_medianas_inv(:,m) = round(lin_medianas_inv_rotat(:,m) + [cc(2);cc(1)]);
    elseif lin_grandes_inv_rotat(1,m) ~= 0
        lin_grandes_inv(:,m) = round(lin_grandes_inv_rotat(:,m) + [cc(2);cc(1)]);
    end
end


imshow(imagenBB)
hold on
plot(lin_pequenas_inv(1,:),lin_pequenas_inv(2,:),'m*')
plot(lin_medianas_inv(1,:),lin_medianas_inv(2,:),'g*')
plot(lin_grandes_inv(1,:),lin_grandes_inv(2,:),'r*')
hold off
```

Código 7. Rotación de los puntos para la obtención de sus equivalentes en la imagen original

Figura 23. Puntos marcadores de las líneas y sus tipos sobre la imagen original: multiplicados por la componente X (izqda.) y multiplicados por toda la matriz de rotación (dcha.) [Fuente propia]

4.2.5 Obtención del resultado

Finalmente, se debe asignar al punto buscado, que es el punto más cercano a la mano, su valor real. De los vectores de las tres clases de puntos, se guarda el último valor, es decir, el de la última posición, ya que es el más cercano a los dedos. De entre los tres, se vuelve a coger el de mayor posición y a éste se le asigna su valor. El valor depende del tipo de línea que sea. Si es una línea grande el punto vale lo mismo que el número más próximo (como se ha dicho anteriormente, es el último número leído). Si se trata de una línea mediana, su valor es el número más próximo más 0'5. Y si es una línea pequeña, es el número más próximo más la distancia entre esta línea pequeña y la última línea grande, por 0'1. Este proceso se muestra en el código 8.

```
% Mirem quin és l'última ratlla i eixa serà on es troba la mà
if num_grandes > num_medianas && num_grandes > num_pequenas
    mano = proxim;
elseif num_medianas > num_grandes && num_medianas > num_pequenas
```

```
    mano = proxim + 0.5;
else
    dist = length(find(lin_pequenas(num_grandes:num_pequenas)));
    mano = proxim + dist*0.1;
end
% RESULTAT
mano
```

Código 8. Asignación del valor correspondiente al punto buscado

En el apartado 4.2.4. (Detección y clasificación de las líneas milimétricas) se expone que el recorte de la imagen de líneas milimétricas se realiza dos veces porque la obtención de resultados varía según la cantidad recortada. En consecuencia, el cálculo de este punto final también se realiza dos veces. Para cada valor de reducción o recorte de la imagen se almacena el resultado final. Por último, se escoge el resultado de mayor valor. Figura 24.

Figura 24. Resultado [Fuente propia]

4.3 Desarrollo del código con Python

La segunda fase del proyecto ha sido pasar el código a Python, esto permite preparar el código para que en un futuro se convierta en una aplicación para un dispositivo móvil.

Para llevar a cabo el desarrollo del código en Python, ya se partía de la base realizada en MATLAB, por lo que el trabajo principal de este punto ha sido la adaptación del código realizado con esta herramienta al lenguaje Python. Por tanto, también se va a abordar este apartado siguiendo el funcionamiento del código.

4.3.1 Determinación del fotograma con el punto más alto

El primer paso es leer el vídeo y buscar el fotograma con el punto más alto. El proceso para llegar hasta él es el mismo que MATLAB, como se recoge en el apartado 4.2.1. Este proceso en Python se muestra en el código 9.

```
# Lectura del vídeo
v = cv2.VideoCapture('video02.mp4')


# Creació i inicialització de variables
numeroframe = 0
numFrames = int(v.get(cv2.CAP_PROP_FRAME_COUNT)) # Número de frames
frame_mean_array = []
# Lectura de frames i operacions
while v.isOpened():
    ret,frame = v.read()
```

```
    if not ret:
        break # Surt si no hi ha més frames

    if numeroframe == 0:
        frame_act_g = frame[:,:,1]
        _,BW_act = cv2.threshold(frame_act_g,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
        BW_ref = cv2.bitwise_not(BW_act)
    else:
        frame_act_g = frame[:,:,1]
        _,BW_act = cv2.threshold(frame_act_g,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
        BW_act = cv2.bitwise_not(BW_act)

        # Se calcula la diferència entre fotogrames
        frame_resta_act = cv2.bitwise_xor(BW_act,BW_ref)
        # Calcula la mitja de la diferència i l'emmagatzema en un vector
        # per a treballar amb ell més avant
        frame_mean_act = np.mean(frame_resta_act[:])
        frame_mean_array.append(frame_mean_act)

    # Actualitzem algunes variables
    numeroframe += 1

frame_mean_array = np.array(frame_mean_array)
plt.plot(frame_mean_array)
plt.show()

frame_mean_max = max(frame_mean_array)
nFrame = np.argmax(frame_mean_array)+1
print('nFrame: ',nFrame)

# Se mostra el fotograma del punt més alt
v.set(cv2.CAP_PROP_POS_FRAMES, nFrame-1)
ret, vDib = v.read()
cv2.imshow('vDib', vDib)
cv2.waitKey()

v.release()
```

Código 9. Lectura del vídeo y determinación del fotograma

4.3.2 Detección de los bordes de la cinta métrica

La reducción de tamaño de la imagen, su filtrado y binarización sigue el mismo proceso realizado en MATLAB. A continuación, en el código 10, se presenta un extracto del algoritmo para la obtención de la imagen ‘BW’ como resultado del filtrado y binarización de la imagen ‘imA2’.

```
# Buscar angle de rotació per a posar la regla recta
BW = cv2.Canny(cv2.GaussianBlur(cv2.cvtColor(imA2,cv2.COLOR_BGR2GRAY), (5,5),0), 50, 150)
```

Código 10. Filtrado y binarización de la imagen

La detección de márgenes para rotar la imagen también se realiza mediante la transformada de Hough, en este caso el del paquete de 'skimage', que adopta la siguiente forma, código 11. Para la obtención de las líneas de los bordes del metro se utiliza el de OpenCV, código 12.

```
H1, T1, R1 = hough_line(BW)
P1 = hough_line_peaks(H1, T1, R1, threshold=math.ceil(0.1 * H1.max()),num_peaks=3)
angulo = np.degrees(P1[1][0])
```

Código 11. Detección del ángulo mediante la transformada de Hough de 'skimage'

```
lines = cv2.HoughLinesP(BW,0.5,np.pi*20/180,threshold=1,minLineLength=7,maxLineGap=5)
```

Código 12. Obtención de líneas mediante Hough de OpenCV

Estos pasos se aprecian a continuación en la figura 25.

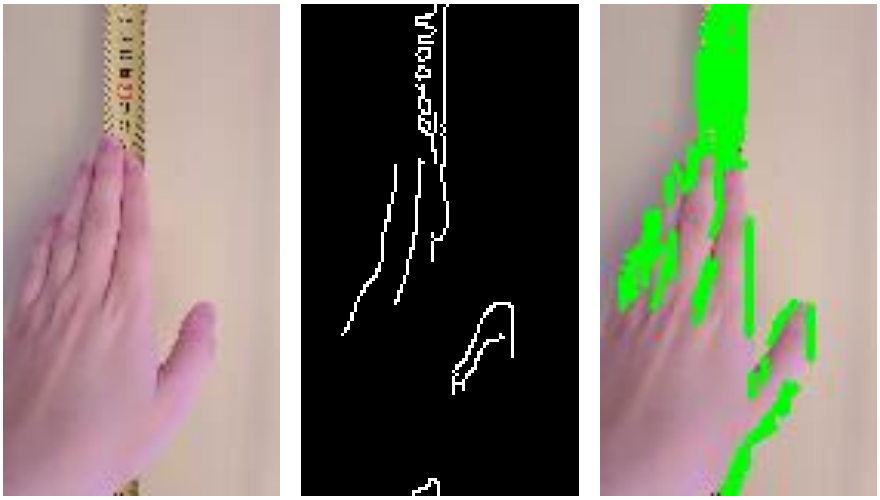


Figura 25. Imagen reducida de tamaño (izda.), filtrada y binarizada (centro) y detección de líneas (dcha.) [Fuente propia]

4.3.3 Lectura mediante OCR

En Python, la lectura de números es más precisa como se verá más adelante, por lo que no ha sido necesario añadir a la detección de márgenes la repetición del proceso de detección de líneas con dos ángulos distintos.

Después de recortar la imagen original por las esquinas adecuadas, se prepara la imagen para la lectura. Como en la base realizada en MATLAB, se binariza la imagen con un umbral, que, aunque es fijo en la función de binarización, es variable en cada imagen. Binarizando se consigue destacar los números y líneas sobre el fondo. A diferencia del desarrollo en MATLAB, el umbral escogido aquí indica el valor límite del píxel para considerarse negro o blanco. Como se observa en el extracto de código 13, 'imagenBW' es la imagen resultante de la binarización de la imagen 'im_regla_h_gris', que es una imagen en escala de grises, con 'umbral' que es el valor utilizado para clasificar los píxeles, y el resultado se muestra en la figura 26. Además, se le introduce el valor máximo que tendrán los píxeles que superen este umbral, en este caso será 255 (píxel blanco), si no lo superan tendrán valor 0 (píxel negro). Esta binarización es la básica y está indicada en el último parámetro, 'cv2.THRESH_BINARY'.

```
media_gris = np.mean(im_regla_h_gris)
print('media_gris: ',media_gris)
if media_gris < 130:
    umbral = 80
elif media_gris >= 130 and media_gris <= 150:
    umbral = 127
```

```
else:
    umbral = 150

_ ,imagenBW = cv2.threshold(im_regla_h_gris,umbral,255,cv2.THRESH_BINARY)
```

Código 13. Binarización previa a la lectura de los números

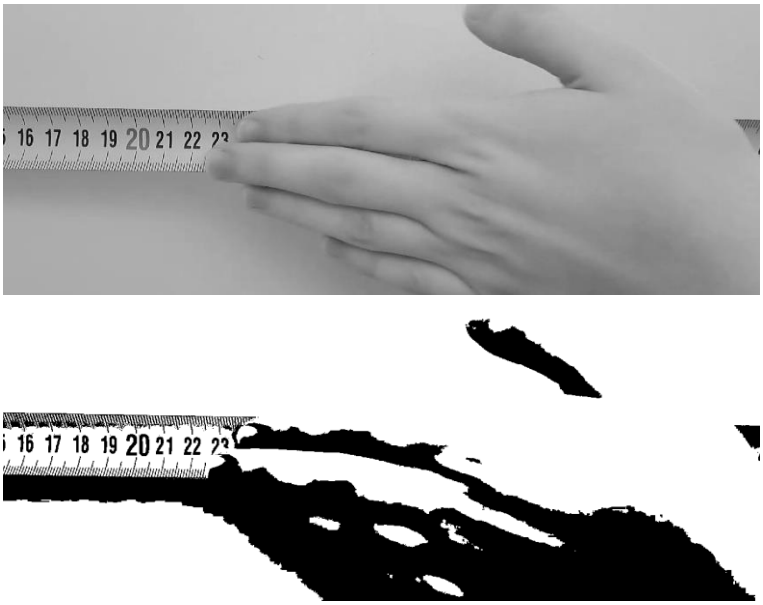


Figura 26. Imagen en escala de grises (arriba) y su binarización (abajo) [Fuente propia]

A continuación, se leen los números. Para la lectura, se utiliza 'pytesseract.image_to_string'. Del texto extraído, se buscan solo los que sean dígitos, y seguidamente se dividen en grupos de números consecutivos, mostrado en el código 14 y la figura 27.

```
# Primera lectura de la regla
numeros = pytesseract.image_to_string(imagenBW, config="--psm 6")
# Selecció únicament dels números
numeros_solo = re.findall(r'\d+', numeros)

# Funció per a detectar les seqüències consecutives de números
def consecutivos(lista_num):
    # Crear variables
    secuencias_consecutivas = []
    secuencia_actual = []

    lista_num = [int(num) for num in lista_num] # Passar d'string a integer els elements de
'lista_num'

    for num in lista_num:
        # Si 'secuencia_actual' està buida o l'element és consecutiu:
        if (not secuencia_actual) or (num == secuencia_actual[-1] + 1):
            secuencia_actual.append(num) # Afegeix element a 'secuencia_actual'
        else:
            # Si no és consecutiu i 'secuencia_actual' té més dos o més elements:
```

```
            if len(secuencia_actual) > 1:
                secuencias_consecutivas.append(secuencia_actual) # Guarda la sequencia actual
            secuencia_actual = [num] # Crea una sequencia nova amb l'element actual

    # Mirar si queda una altra sequencia després del bucle, si hi ha s'ha de guardar
    if len(secuencia_actual) > 1:
        secuencias_consecutivas.append(secuencia_actual)

    return secuencias_consecutivas

# Resultat final de la lectura
secuencias_consecutivas = consecutivos(numeros_solo)
print("secuencias_consecutivas: ", secuencias_consecutivas)
```

Código 14. Lectura de los números

```
numeros:  ATTA HST
16 17 18 19 2021 22 —= . ‘

numeros_solo:  ['16', '17', '18', '19', '2021', '22']
secuencias_consecutivas:  [[16, 17, 18, 19]]
```

Figura 27. Lectura del texto y extracción de las secuencias de números consecutivos [Fuente propia]

Por lo general, la lectura del OCR Pytesseract de Python es mejor que la de MATLAB. De todos modos, se debe comprobar que entre los números leídos y los esperados hay, al menos, 3 coincidentes. Si se cumple la condición, el programa continúa, si no lo hace, se termina la ejecución y se notifica que la lectura no es buena.

4.3.4 Detección y clasificación de líneas milimétricas

El siguiente paso es buscar la zona del número más próximo y recortar la imagen. El OCR de MATLAB proporciona los datos de los Bounding Boxes de los elementos leídos. En cambio, ‘pytesseract.image_to_string’ no los proporciona. Para solucionar este problema, ya que es necesario el Bounding Box del número más próximo para recortar la imagen, se ha realizado una segunda lectura con ‘pytesseract.image_to_data’. Esta función sí que proporciona los datos necesarios. Se ha pensado dejar solo esta función para la lectura de los números, sin embargo, como en algunas ocasiones los resultados de lectura con ‘pytesseract.image_to_string’ son mejores, finalmente, se ha decidido dejar las dos lecturas para después escoger el valor óptimo, figura 28.

```
secuencias_consecutivas:  [[16, 17, 18, 19]]
textos_cons:  [[16, 17, 18, 19, 20, 21, 22]]
```

Figura 28. Lectura con ‘pytesseract.image_to_string’ (arriba) y con ‘pytesseract.image_to_data’ (abajo) [Fuente propia]

Esta segunda lectura, con ‘pytesseract.image_to_data’, también se debe filtrar para obtener solo los dígitos. Además, en algunas ocasiones, los Bounding Boxes no son del todo correctos, porque dos números los reconoce como uno solo y, por tanto, el Bounding Box tiene el doble del tamaño esperado al rodear dos números en vez de uno, o también porque detecta la mano como parte del texto y resulta un Bounding Box con un tamaño excesivo.

El primer problema se soluciona en el momento de guardar Bounding Boxes, ya que cuando se encuentre un número leído con cuatro dígitos se separa el texto en dos números de dos dígitos

cada uno y su Bounding Box se divide por la mitad. Esto mejora también el reconocimiento de números porque se recuperan números "perdidos", figura 29.

```
textos:  ['16', '17', '18', '19', '2021', '22']
textos cons:  [[16, 17, 18, 19]]

textos:  ['16', '17', '18', '19', '20', '21', '22']
textos_cons:  [[16, 17, 18, 19, 20, 21, 22]]
```

Figura 29. Lectura de los números sin separación (arriba) y con separación (abajo) [Fuente propia]

El segundo problema se soluciona al recortar la imagen. La altura (eje Y) se recorta con los valores del penúltimo Bounding Box (el último es el problemático) y añadiendo un margen para recortar todo el metro y no solo el número. Los valores en el eje X van desde el valor de X de la esquina superior izquierda que se obtiene del Bounding Box del número más próximo hasta este valor más la anchura del penúltimo Bouning Box. A estos también se les da un margen.

Con el recorte de la imagen por el número más próximo, se recortan las líneas milimétricas, un proceso que se realiza como en MATLAB, buscando la línea negra del borde y una línea blanca entre las líneas milimétricas y el número, como se describe en el apartado 4.2.4 (Detección y clasificación de las líneas milimétricas). A continuación, se "esquelenotiza" la imagen, se detecta el ángulo de rotación de las líneas y se rota la misma, figura 30. De nuevo, al igual que en MATLAB, se recortan ligeramente los bordes de esta imagen para que creen picos falsos en la suma. Sin embargo, en Python ha sido suficiente recortar una única vez.

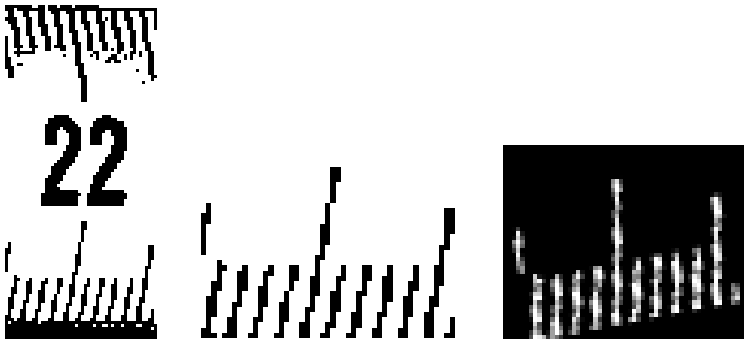


Figura 30. Recorte por la zona próxima a la mano (izqda.), recorte de las líneas milimétricas (centro) y rotación de éstas (dcha.) [Fuente propia]

La detección de picos se ha llevado a cabo también con la detección de máximos locales y su clasificación en clases se ha realizado con la comparación de sus valores, con la única diferencia del cambio de valores de los percentiles, que en este caso se ha utilizado el percentil 95 y el 99, como se muestra en el código 15.

```
# Assignació tipus a cada pic
percentil99 = np.percentile(c,99)
percentil95 = np.percentile(c,95)

lin_pequenas = np.zeros((2,len(c)))
lin_medianas = np.zeros((2,len(c)))
lin_grandes = np.zeros((2,len(c)))

fig,ax = plt.subplots()
ax.plot(c_aux1)
posC = 0
for m in puntosmax:
    if m >= percentil99:
```

```
        lin_grandes[:,posC] = [posC,m]
    elif m >= percentil95 and m <= percentil99:
        lin_medianas[:,posC] = [posC,m]
    elif m < percentil95 and m > 1:
        lin_pequenas[:,posC] = [posC,m]
    posC += 1

ax.plot(lin_grandes[0],lin_grandes[1],'r*')
ax.plot(lin_medianas[0],lin_medianas[1],'g*')
ax.plot(lin_pequenas[0],lin_pequenas[1],'m*')
plt.show()
```

Código 15. Clasificación de líneas milimétricas

La rotación de estos puntos se ha realizado mediante la multiplicación de la matriz de rotación, y el proceso seguido ha sido el mismo que para la base desarrollada en MATLAB, como se ha citado anteriormente en el apartado 4.2.4. En Python, se muestra el desarrollo en el código 16 y los resultados en la figura 31.

```
# Rotació punts
lin_pequenas_centrat = np.zeros_like(lin_pequenas)
lin_medianas_centrat = np.zeros_like(lin_medianas)
lin_grandes_centrat = np.zeros_like(lin_grandes)
for m in range(lin_pequenas.shape[1]):
    if lin_pequenas[0,m] != 0:
        lin_pequenas_centrat[:,m] = lin_pequenas[:,m] - esteCentro
    elif lin_medianas[0,m] != 0:
        lin_medianas_centrat[:,m] = lin_medianas[:,m] - esteCentro
    if lin_grandes[0,m] != 0:
        lin_grandes_centrat[:,m] = lin_grandes[:,m] - esteCentro
lin_pequenas_inv_rotat = np.zeros_like(lin_pequenas_centrat)
lin_medianas_inv_rotat = np.zeros_like(lin_medianas_centrat)
lin_grandes_inv_rotat = np.zeros_like(lin_grandes_centrat)
for m in range(lin_pequenas_centrat.shape[1]):
    lin_pequenas_inv_rotat[:,m] = np.dot(rotMatInv,lin_pequenas_centrat[:,m])
    lin_medianas_inv_rotat[:,m] = np.dot(rotMatInv,lin_medianas_centrat[:,m])
    lin_grandes_inv_rotat[:,m] = np.dot(rotMatInv,lin_grandes_centrat[:,m])
lin_pequenas_inv = np.zeros_like(lin_pequenas_inv_rotat)
lin_medianas_inv = np.zeros_like(lin_medianas_inv_rotat)
lin_grandes_inv = np.zeros_like(lin_grandes_inv_rotat)
for m in range(lin_grandes_inv_rotat.shape[1]):
    if lin_pequenas_inv_rotat[0,m] != 0:
        lin_pequenas_inv[:,m] = np.round(lin_pequenas_inv_rotat[:,m] + np.array([cw2,ch2]))
    elif lin_medianas[0,m] != 0:
        lin_medianas_inv[:,m] = np.round(lin_medianas_inv_rotat[:,m] + np.array([cw2,ch2]))
    if lin_grandes[0,m] != 0:
        lin_grandes_inv[:,m] = np.round(lin_grandes_inv_rotat[:,m] + np.array([cw2,ch2]))
fig,ax = plt.subplots()
ax.imshow(imagenBB_rec2)
ax.plot(lin_pequenas_inv[0],lin_pequenas_inv[1],'m*')
```

```
ax.plot(lin_medianas_inv[0],lin_medianas_inv[1],'g*')
ax.plot(lin_grandes_inv[0],lin_grandes_inv[1],'r*')
plt.show()
```

Código 16. Rotación de puntos

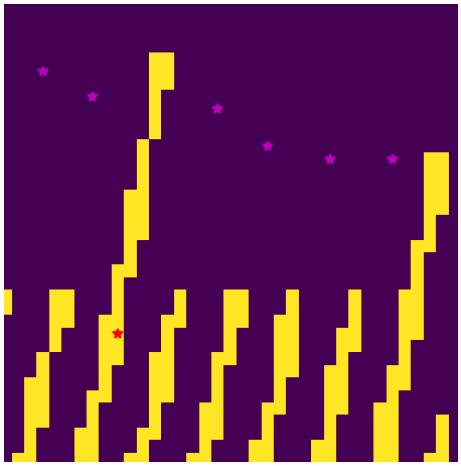


Figura 31. Representación de los puntos sobre la imagen [Fuente propia]

4.3.5 Obtención del resultado

La última parte del programa es la obtención del resultado y sigue la misma lógica que se había programado en el código de MATLAB. Se coge el último valor de posiciones, que corresponde a la posición de la última línea milimétrica, de cada tipo de línea. De las tres opciones posibles, se coge la de mayor valor de posición. Si es del grupo de líneas grandes, el resultado es el valor del número más próximo. Si es una línea mediana, es el valor del número más próximo más 0'5. Finalmente, si es del tipo de línea pequeña, hay dos opciones: que esté entre 0'1 y 0'4 o entre 0'6 y 0'9. Si está entre 0'1 y 0'4, el resultado es el valor del número más próximo más el número de líneas que hay entre la última línea grande y la línea en cuestión por 0'1. Si está entre 0'6 y 0'9, es el valor del número más próximo más el número de líneas que hay entre la última línea mediana y la línea pequeña buscada por 0'1. Se muestra el proceso en el código 17.

```
# Assignem el valor a l'última ratlleta detectada
if num_grandes > num_medianas and num_grandes > num_pequenas:
   mano = proxim
elif num_medianas > num_grandes and num_medianas > num_pequenas:
   mano = proxim + 0.5
else:
   if num_grandes > num_medianas:
      dist = len(np.nonzero(lin_pequenas[0,num_grandes:num_pequenas+1])[0])
      mano = proxim + dist*0.1
   else:
      dist = len(np.nonzero(lin_pequenas[0,num_medianas:num_pequenas+1])[0])
      mano = proxim + 0.5 + dist*0.1
print('mano: ',mano)
```

Código 17. Asignación del valor correspondiente al punto buscado

Y el resultado se muestra a continuación en la figura 32.

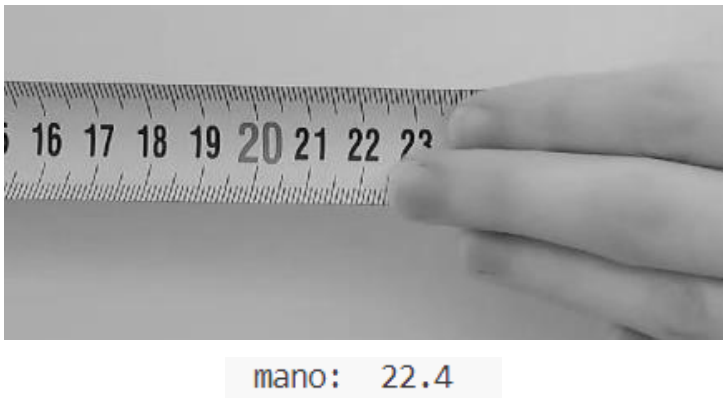


Figura 32. Resultado [Fuente propia]

Capítulo 5. Resultados

En este capítulo se exponen los resultados extraídos de la ejecución del programa con distintos vídeos.

Como se ha visto en el capítulo 4 (Desarrollo), el programa funciona con simulaciones de salto. En estas simulaciones, el movimiento de la mano es más lento y la grabación más cercana a la acción. Por eso, han servido como punto de partida para la programación del algoritmo. Aunque los resultados no siempre son tan ajustados, son una buena aproximación, figura 33.

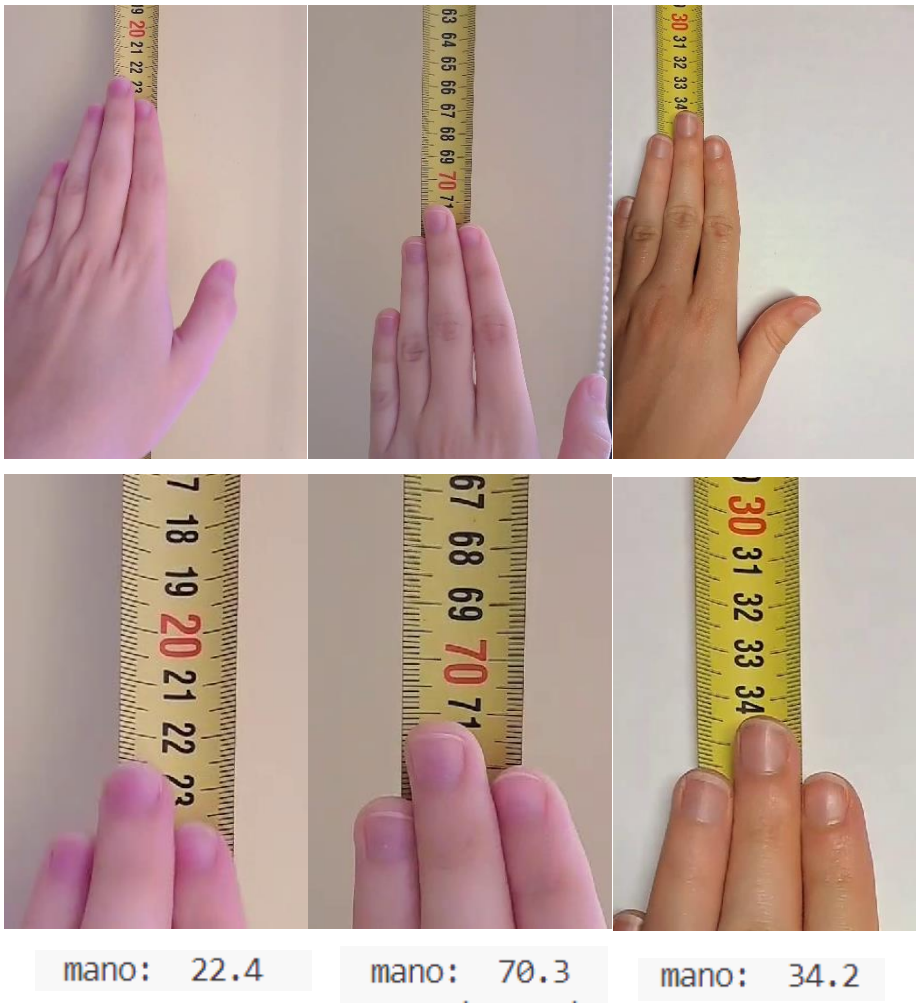


Figura 33. Salto (arriba), ampliación (centro) y resultado (abajo) [Fuente propia]

Por tanto, se prueba ahora con saltos de verdad, donde el movimiento es bastante más rápido y la cámara no está tan cerca.

Después de varias pruebas, algunos vídeos han resultado exitosos, mientras que otros, no los ha leído bien. Esto se debe a distintos factores, como la sombra provocada por la situación del metro respecto a la luz (aunque visualmente, no era muy pronunciada) y que influye en la binarización, como en la figura 34; y el recorte de la cinta métrica o las líneas milimétricas. Por tanto, sería un área de mejora en la continuación del proyecto.

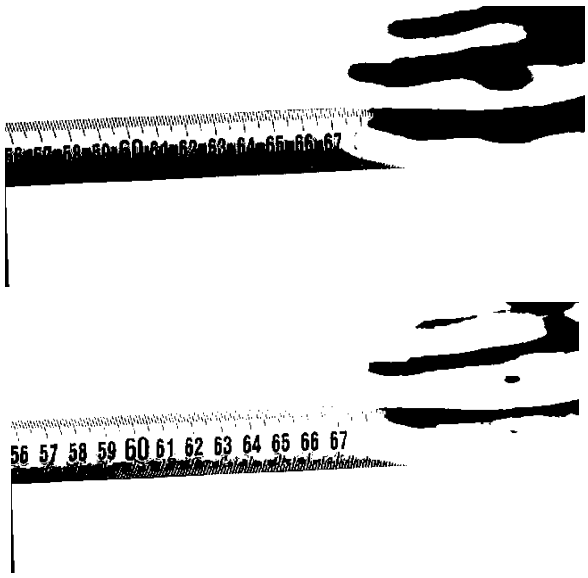


Figura 34. Cambio de umbral de binarización [Fuente propia]

Primero se ha probado con el metro de IKEA, pero los números no resultaban leíbles porque las líneas milimétricas y la numeración son muy finas. Ver figura 35.

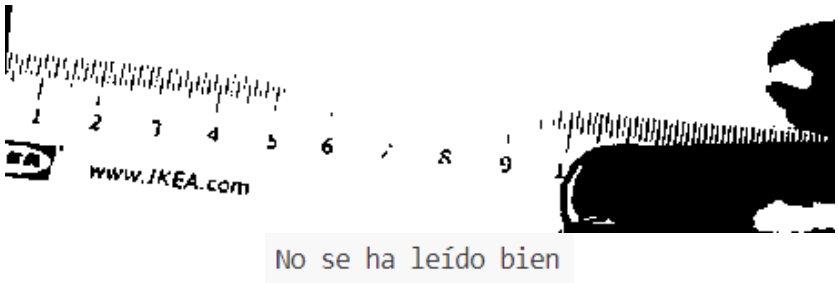


Figura 35. Lectura sobre el metro de IKEA [Fuente propia]

Después, se ha probado con la cinta métrica. Con éste, los números eran más visibles. Se muestran los resultados en la figura 36.

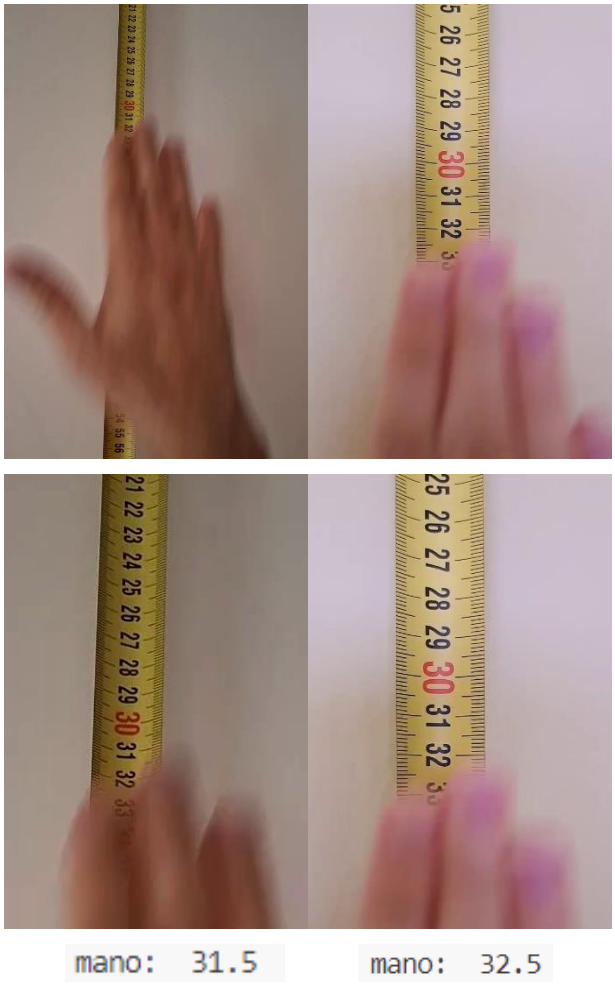


Figura 36. Salto (arriba), ampliación (centro) y resultado (abajo) [Fuente propia]

Una vez analizados los resultados obtenidos con las diferentes pruebas realizadas se ha comprobado que ha existido un mejor funcionamiento del programa cuando se han dado las siguientes situaciones:

- Cámara con buena resolución.
- Cámara cerca de la acción.
- Cinta métrica con números leíbles.
- Cámara lo más estable posible.
- Desplazamiento de la mano por encima de la cinta métrica.
- Buena iluminación.
- Trabajar sobre una pared lisa, evitando líneas que puedan confundir.

Capítulo 6. Conclusiones y líneas futuras

Al finalizar este proyecto se pueden extraer distintas conclusiones. Se puede afirmar que se han cumplido los objetivos propuestos. Y aunque el funcionamiento no ha sido exitoso en el cien por cien de los vídeos realizados, se ha conseguido crear una buena base para empezar.

Para obtener buenos resultados en todos los casos, o en su gran mayoría, se podrían realizar mejoras. Estas podrían ser, entre otras, perfeccionar el ajuste del umbral de binarización antes de la lectura, ajustar el recorte de la imagen y de las líneas milimétricas, desarrollar una buena lectura con distintos tipos de cinta métrica y distintos fondos, la lectura de la cinta métrica en sentido contrario al programado en este proyecto, lectura de los números con la cámara a mayor distancia y conseguir, en todos los vídeos, una mejor aproximación del resultado a la realidad.

Una línea futura sería que midiese la distancia saltada. Esto se conseguiría desarrollando el código para que leyese la posición de la mano antes del salto y llevando a cabo una resta con el punto del salto que ya proporciona este proyecto.

Otra línea de desarrollo consistiría en transformar este proyecto en una aplicación para dispositivos móviles, como teléfonos y tabletas. De ese modo, no sería necesario disponer del código y las herramientas MATLAB y Python para su ejecución. Esto permitiría a todas las personas que lo necesiten, especialmente los entrenadores y deportistas de actividades en las cuales el salto vertical es de gran importancia, como lo es el baloncesto, el disponer de una aplicación de bajo coste económico, así como de cómodo y fácil uso.

Finalmente, me gustaría añadir que este proyecto me ha proporcionado una visión más amplia de las aplicaciones de la ingeniería de telecomunicaciones en el mundo real. Además, he podido ampliar mis conocimientos sobre una parte de ella que me parecía muy interesante, pero no había podido experimentar en gran medida: el procesamiento de imágenes. También me ha introducido en un lenguaje de programación que antes no había utilizado y del que ahora ya tengo una base con la que puedo contar para mi futuro profesional.

Capítulo 7. Bibliografía y fuentes

[1] «Deporte,» Real Academia Española, [En línea]. Available: https://dle.rae.es/deporte. [Último acceso: 20 06 2024].

[2] «Beneficios de la actividad física,» Ministerio de Sanidad, [En línea]. Available: https://www.sanidad.gob.es/areas/promocionPrevencion/actividadFisica/guiaPadresMadres/beneficiosActividadFisica.htm. [Último acceso: 20 06 2024].

[3] M. Chalufour, «A Revolutionary Idea,» Boston University, 2020-2021. [En línea]. Available: https://www.bu.edu/sargent/about-us/our-publications/inside-sargent/inside-sargent-2020/a-revolutionary-idea/. [Último acceso: 19 06 2024].

[4] D. A. Sargent, «The Physical Test of a Man,» *American Physical Education Review*, nº 26, pp. 188-194, 1921.

[5] R. Wood, «Vertical Jump Test,» Topend Sports Website, 2008. [En línea]. Available: https://www.topendsports.com/testing/tests/vertjump.htm. [Último acceso: 18 06 2024].

[6] J. Colomer, «Salto Vertical: ¿Cómo medir la potencia del tren inferior?,» Blog HSN, [En línea]. Available: https://www.hsnstore.com/blog/deportes/fitness/salto-vertical/. [Último acceso: 19 06 2024].

[7] «How does the Jump Scan measure Jump Height?,» Sparta Science, [En línea]. Available: https://success.spartascience.com/en/knowledge/how-does-sparta-measure-jump-height. [Último acceso: 20 junio 2024].

[8] R. Wood, «Vertec: Vertical Jump Equipment,» Topend Sports Website, 2008. [En línea]. Available: https://www.topendsports.com/testing/products/vertical-jump/vertec.htm. [Último acceso: 19 junio 2024].

[9] «Vertec™ Vertical Jump Trainer,» Sports Imports, [En línea]. Available: https://www.sportsimports.com/product/vertec-jump-trainer/. [Último acceso: 19 junio 2024].

[10] R. Wood, «Just Jump System: Vertical Jump Equipment,» Topend Sports Website, 2008. [En línea]. Available: https://www.topendsports.com/testing/products/vertical-jump/justjump.htm. [Último acceso: 19 junio 2024].

[11] «MATLAB,» Wikipedia, [En línea]. Available: https://es.wikipedia.org/wiki/MATLAB. [Último acceso: 18 junio 2024].

[12] «Python (programming language),» Wikipedia, [En línea]. Available: https://en.wikipedia.org/wiki/Python_%28programming_language%29. [Último acceso: 18 junio 2024].

[13] [En línea]. Available: https://uxwing.com/wp-content/themes/uxwing/download/brands-and-social-media/visual-studio-code-icon.png. [Último acceso: 18 junio 2024].

[14] «Repaso de caída libre (artículo),» Khan Academy, [En línea]. Available: https://es.khanacademy.org/science/ap-physics-1/ap-one-dimensional-motion/falling-objects-ap-physics/a/freefall-ap1. [Último acceso: 18 junio 2024].

[15] «Physics questions and answers,» Chegg, 30 septiembre 2020. [En línea]. Available: https://www.chegg.com/homework-help/questions-and-answers/problem-analysis-

standing-vertical-jump-athlete-diagram-shows-stages-called-squat-jump-exe-q57266107. [Último acceso: 18 junio 2024].

[16] J. M. Martínez Sánchez, «Tema 6: Extracción de características geométricas,» Universidad Autónoma de Madrid, Diciembre 2010. [En línea]. Available: http://arantxa.ii.uam.es/~jms/taps/teoria/TAPS_Tema6_2010_1-90(bn).pdf. [Último acceso: 13 junio 2024].

[17] «Hough transform,» MathWorks, [En línea]. Available: https://es.mathworks.com/help/images/ref/hough.html. [Último acceso: 13 junio 2024].

[18] L. M. Jiménez García, «umh1782 2021-22 Lección 009-6 - Segmentación de Imágenes: Transformada de Hough,» Universidad Miguel Hernández de Elche, 15 marzo 2022. [En línea]. Available: https://www.youtube.com/watch?v=R1X1Uec6qkw. [Último acceso: 13 junio 2024].

Anexo I. Objetivos de Desarrollo Sostenible de la Agenda 2030

Relación del trabajo de fin de grado con los Objetivos de Desarrollo Sostenible de la Agenda 2030.

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. Fin de la pobreza.				X
ODS 2. Hambre cero.				X
ODS 3. Salud y bienestar.	X			
ODS 4. Educación de calidad.	X			
ODS 5. Igualdad de género.				X
ODS 6. Agua limpia y saneamiento.				X
ODS 7. Energía asequible y no contaminante.				X
ODS 8. Trabajo decente y crecimiento económico.				X
ODS 9. Industria, innovación e infraestructuras.	X			
ODS 10. Reducción de las desigualdades.	X			
ODS 11. Ciudades y comunidades sostenibles.				X
ODS 12. Producción y consumo responsables.				X
ODS 13. Acción por el clima.				X
ODS 14. Vida submarina.				X
ODS 15. Vida de ecosistemas terrestres.				X
ODS 16. Paz, justicia e instituciones sólidas.				X
ODS 17. Alianzas para lograr objetivos				X

Este proyecto cumple a un nivel alto con los siguientes ODS de la Agenda 2030:

- ODS 3. Salud y bienestar.
  Se trata del desarrollo de una aplicación software que ayuda a analizar resultados de la actividad de salto vertical a aquellas personas que practican deportes o trabajan en puestos en los que dicha actividad es muy importante.
- ODS 4. Educación de calidad.
  Las características de la aplicación permitirían un acceso fácil a una herramienta de evaluación de resultados a un tipo de pruebas físicas que son realizadas en los centros educativos.
- ODS 9. Industria, innovación e infraestructuras.
  Está pensado para que, con un desarrollo posterior, se pueda aplicar en los dispositivos móviles, siendo una forma novedosa de obtener los resultados del salto vertical.
- ODS 10. Reducción de las desigualdades.
  Al ser una aplicación destinada a utilizarse en los dispositivos móviles, solo se necesitaría una cinta métrica para efectuar las mediciones, lo que economiza su coste y generaliza su uso.