

# **Estimación de la altura de una persona basada en un objeto de referencia**

**Profesora:** Maria Jose Jimenez Rodriguez

**Asignatura:** Procesamiento de Imágenes Digitales

**Alumnos:**

Darío López Villegas

Juan Javier Sánchez Portillo

Paulo Felipe Rezende da Silva

**Diciembre** 2025

# INDICE

1. Introducción
2. Planteamiento teórico
3. Implementación
4. Experimentación
5. Conclusiones

# INTRODUCCIÓN

- El proyecto se basa en la investigación "Calibration-Free Height Estimation for Person" de Takeda et al. (2024).
- **Objetivo de la Investigación:** Crear un método para estimar la altura humana sin necesidad de una calibración compleja de la cámara ni hardware industrial.

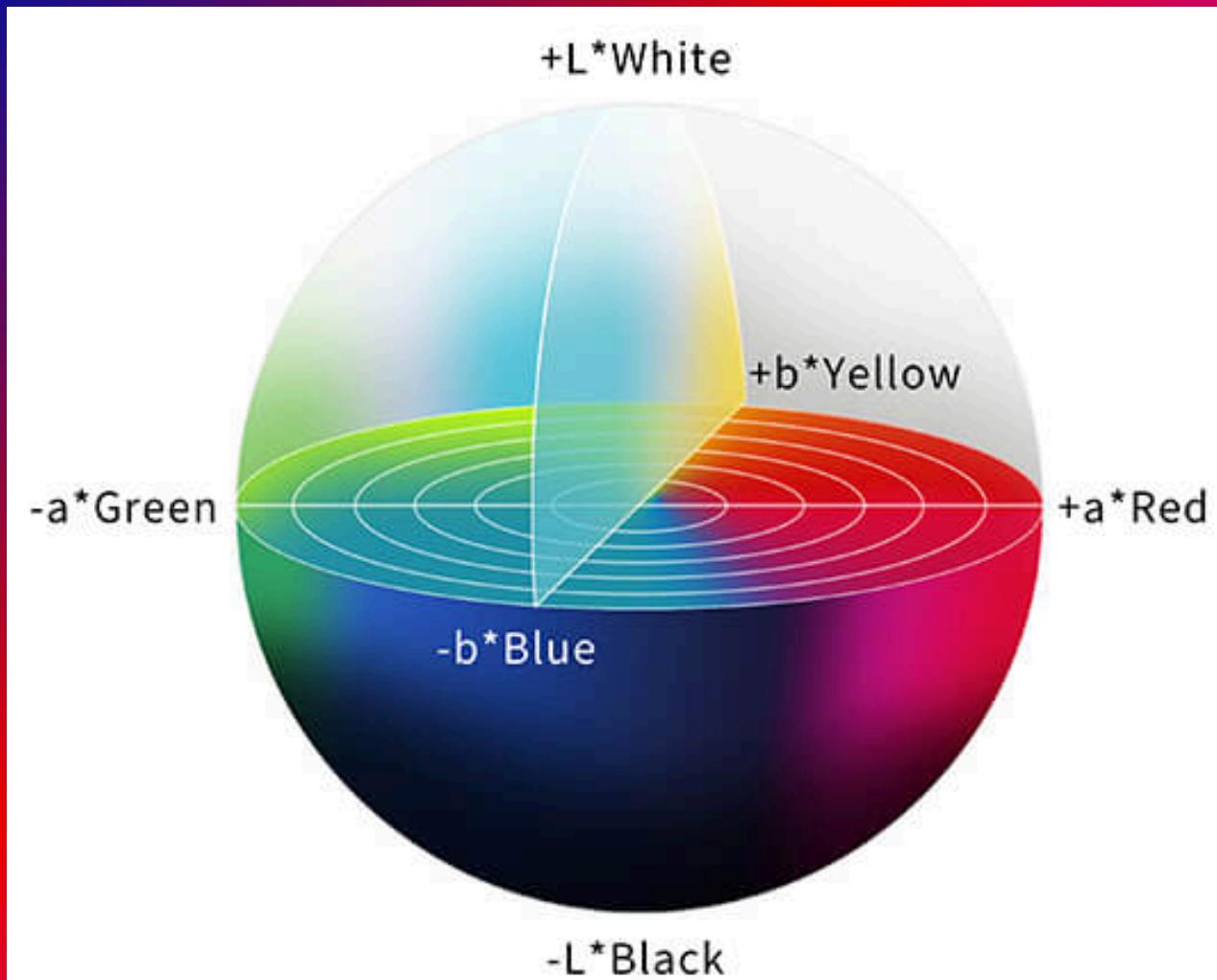
- **Premisa Principal:** Es posible obtener mediciones métricas precisas en imágenes 2D estableciendo una relación geométrica con un objeto de referencia.
- **Propósito de este Proyecto:** Validar experimentalmente si esta premisa teórica es viable en un entorno doméstico utilizando una webcam estándar de bajo coste.

- Solución "**Reference Object**": Para resolver la escala desconocida, el sistema introduce un objeto de dimensiones físicas conocidas en la misma escena que el sujeto.
- **Implementación Práctica:** Se utiliza un folio tamaño A4 ( $210 \times 297$  mm) como constante métrica para calcular la relación píxel/cm.

- **Implementación Práctica:** Se utiliza un folio tamaño A4 ( $210 \times 297$  mm) como constante métrica para calcular la relación píxel/cm.
- **Procesamiento:** El algoritmo combina la detección del humano y del objeto para inferir la altura real mediante proporcionalidad directa, mitigando la necesidad de sensores de profundidad.

# PLANTEAMIENTO TEORICO

## Técnicas de Procesamiento de Imagen Utilizadas



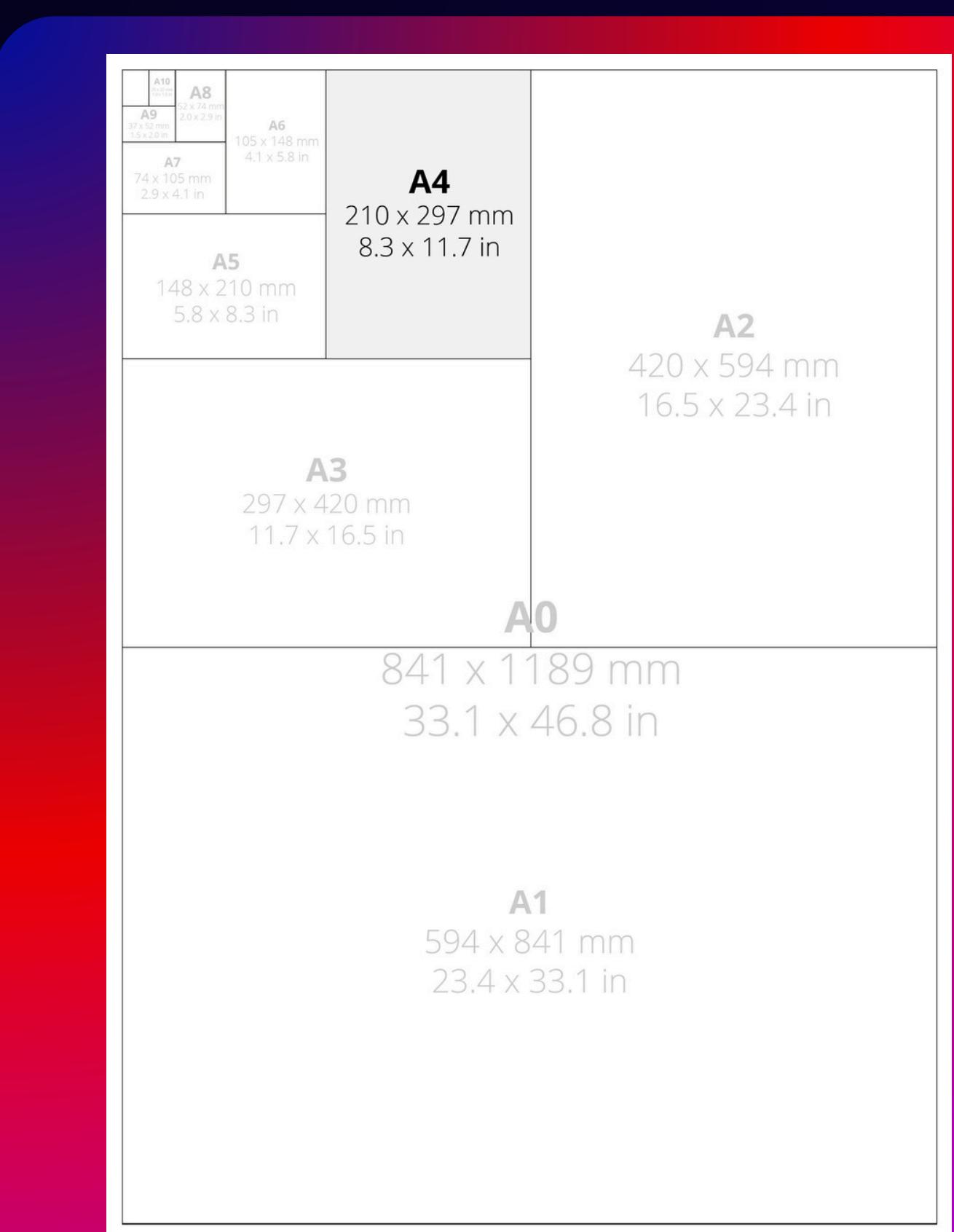
- Transformaciones de color (RGB → LAB)
- Linearización y Conversión sRGB → XYZ
- Conversión XYZ → LAB

## Técnicas de Procesamiento de Imagen Utilizadas

- Umbralización simple
- Operaciones morfológicas (dilatación)
- Detectores de bordes (Canny)
- Contornos y bounding boxes rotados (minAreaRect)

## Técnicas de Procesamiento de Imagen Utilizadas

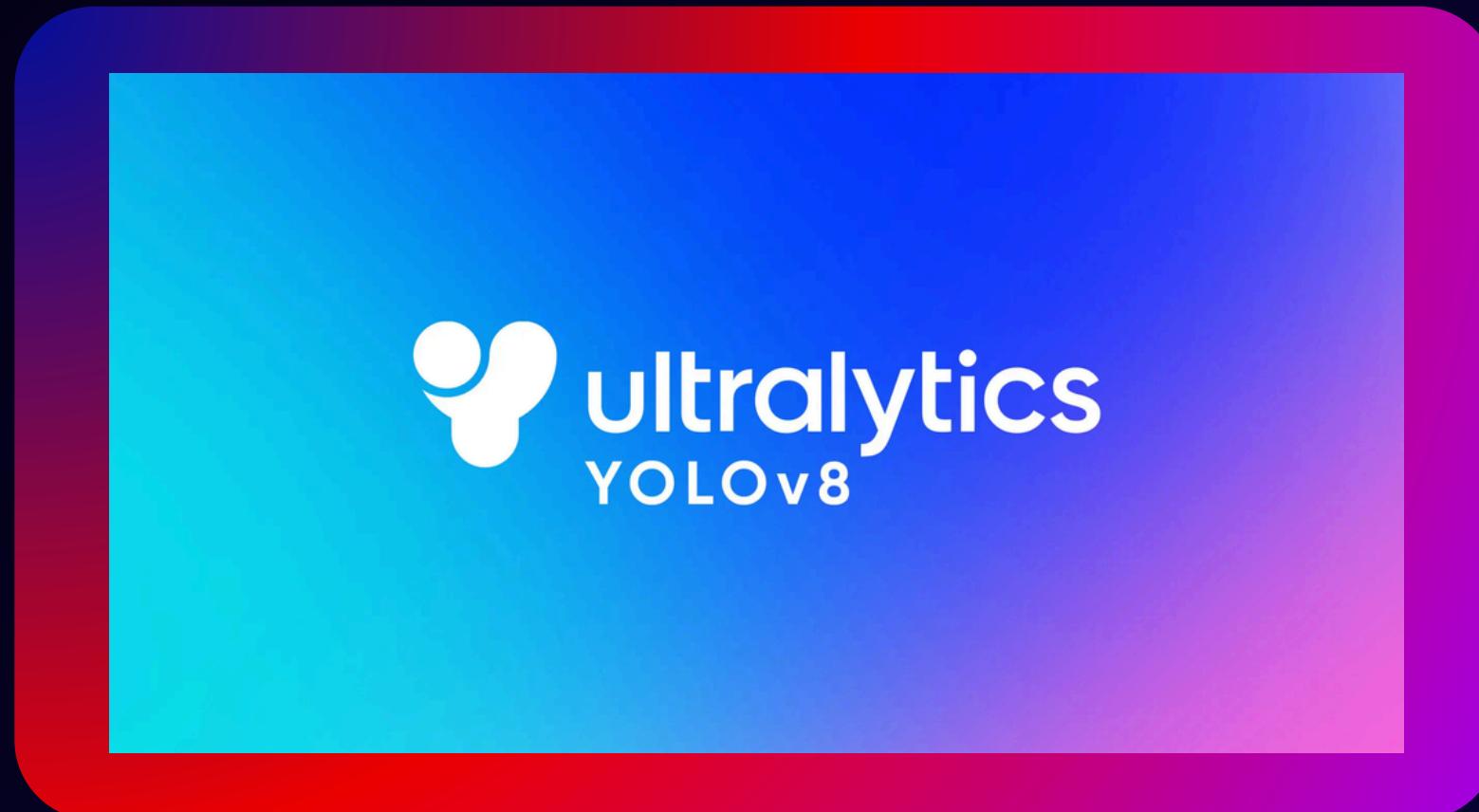
- Análisis Geométrico(ratio A4)



## 2.7 YOLOv8

## 2.7 YOLOv8

- El YOLOv8 es un modelo de detección de objetos que se caracteriza por combinar alta precisión con velocidad de procesamiento en tiempo real, siendo adecuado para uso en aplicaciones industriales, robótica, vigilancia y vehículos autónomos.



## 2.7 YOLOv8

### 2.7.1 Características de YOLOv8

- **Arquitectura Anchor-Free:** Elimina cajas de anclaje predefinidas, prediciendo directamente el centro del objeto para mayor generalización;
- **Utilización del módulo Backbone C2f:** Sustituye al módulo C3, mejorando la extracción de características y el flujo de gradientes con alta eficiencia;
- **Cabezal Desacoplado:** Separa las tareas de clasificación y regresión en ramas distintas para optimizar la precisión de cada una;

## 2.7 YOLOv8

### 2.7.1 Características de YOLOv8

- **Task-Aligned Assigner (TAL):** Asignación dinámica de etiquetas durante el entrenamiento, priorizando la calidad de la alineación entre clase y caja;
- **Utilización del módulo Backbone C2f:** Losses Avanzadas: Uso combinado de CIoU y DFL (Distribution Focal Loss) para un ajuste fino de los bordes de detección

## 2.7 YOLOv8

### Ventajas

- Precisión Superior
- Anchor-Free
- Versatilidad
- Ecosistema

### Inconvenientes

- Coste de Entrenamiento
- Objetos Pequeños
- Complejidad del Código

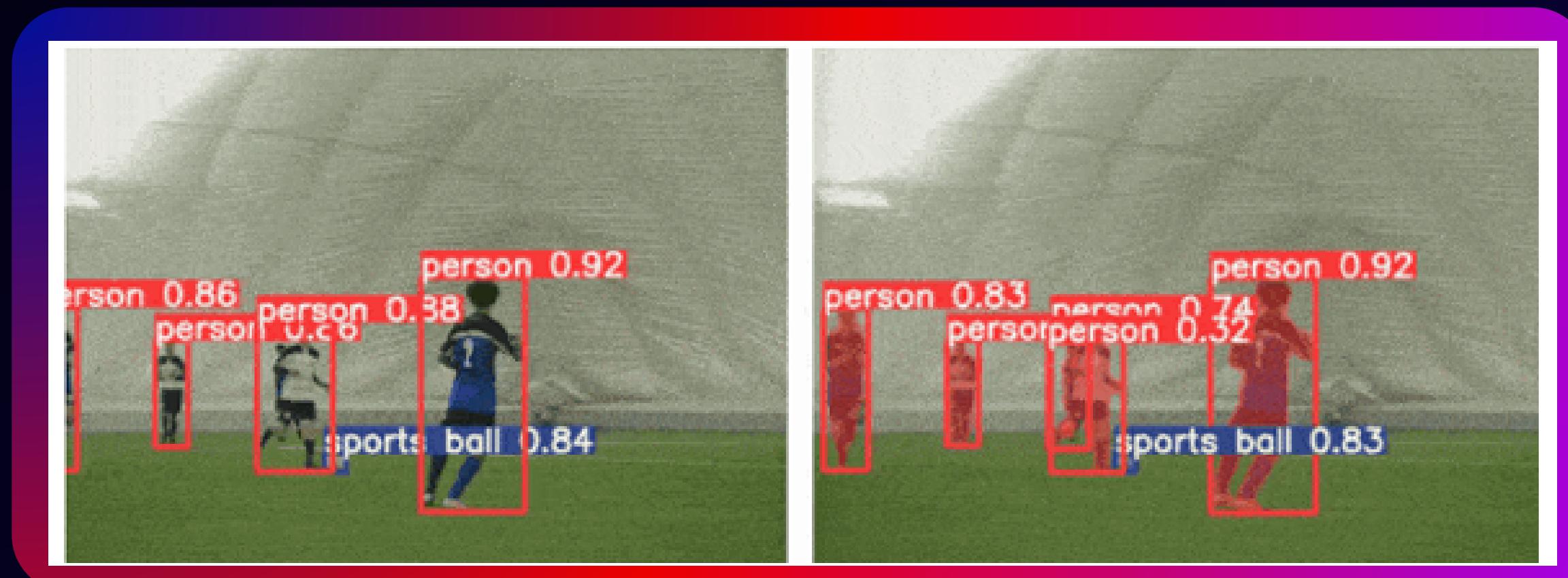
## 2.7 MODELOS YOLOv8 UTILIZADOS

- Best.py
- YOLOv8-Seg

## 2.7 YOLOv8-Seg

### ¿Qué es?

- Variante del modelo diseñada para segmentación de instancias: asigna una máscara de píxeles exacta a cada objeto, en lugar de una simple caja rectangular (bounding box).
- Produce simultáneamente tres salidas: Caja delimitadora, Clase predicha y Máscara de segmentación



## 2.7 YOLOv8-Seg

# Arquitectura Interna de YOLOv8-Seg

### **Backbone (Extracción de Características)**

- Utiliza el módulo C2f y convoluciones 3x3 para extraer mapas de características multiescala profundos.

### **Head de Detección (Clásico)**

- Rama estándar que predice las coordenadas de la bounding box y la probabilidad de clase del objeto.

## 2.7 YOLOv8-Seg

### Arquitectura Interna de YOLOv8-Seg

#### Head de Segmentación (Innovación)

- Mask Prototypes (Protos): Genera un conjunto de 32-64 máscaras base ("protos") mediante una rama convolucional.
- Mask Coefficients: Predice un vector de coeficientes para cada instancia detectada.
- Reconstrucción: La máscara final se obtiene combinando linealmente los protos con los coeficientes, recortándose luego a la caja delimitadora

## 2.7 YOLOv8-Seg

### Ventajas

- Precisión Espacial Real;
- Mediciones Métricas Fiables;
- Robustez en Posturas.

### Inconvenientes

- Coste Computacional;
- Sensibilidad Ambiental;
- Suavizado de Detalles.

# IMPLEMENTACIÓN

# Archivos utilizados

```

import sys
from typing import List, Tuple

import cv2
from ultralytics import YOLO

PESOS_YOLO = "best.pt"

def detectar_personas_y_mostrar(ruta_imagen: str) -> List[Tuple[float, float, float, float]]:
    # Cargar el modelo YOLO con los pesos entrenados.
    modelo = YOLO(PESOS_YOLO)

    # Ejecutar la detección sobre la imagen.
    # - conf: umbral de confianza mínimo para dibujar una detección.
    # - iou: umbral de solapamiento para filtrar cajas muy parecidas.
    resultados = modelo(ruta_imagen, conf=0.55, iou=0.7)

    # `resultados` es una lista; para una sola imagen usamos el primer elemento.
    resultado = resultados[0]

    # Extraer las cajas de detección en formato (x1, y1, x2, y2).
    # `boxes.xyxy` es un tensor; lo convertimos a lista de listas de floats.
    cajas_xyxy: List[List[float]] = resultado.boxes.xyxy.tolist() if resultado.boxes is not None else []

    # Convertimos cada caja a tupla para que sea más cómoda de manejar.
    cajas: List[Tuple[float, float, float, float]] = [
        (x1, y1, x2, y2) for (x1, y1, x2, y2) in cajas_xyxy
    ]

    # Imprimir por consola las cajas detectadas (bordes de detección).
    # Esto será útil para la segunda parte (detección del folio / referencia).
    if cajas:
        print("Cajas de detección encontradas (x1, y1, x2, y2) en pixeles:")
        for i, (x1, y1, x2, y2) in enumerate(cajas, start=1):
            print(f" Persona {i}: ({x1:.2f}, {y1:.2f}, {x2:.2f}, {y2:.2f})")
    else:
        print("No se han detectado personas en la imagen.")

```

# human-detector.py

```

def main() -> None: 1 usage  ↗ JuanJavier03
"""

Punto de entrada del script.

Lógica de argumentos:
- No se aceptan opciones ni flags.
- Solo se espera UN argumento posicional: la ruta de la imagen.
"""

# `sys.argv` contiene: [nombre_script, ruta_imagen]
if len(sys.argv) != 2:
    print("Uso incorrecto.\n")
    print("Forma correcta de ejecutar:")
    print("  python human-detector.py ruta/de/la_imagen.jpg")
    sys.exit(1)

ruta_imagen = sys.argv[1]

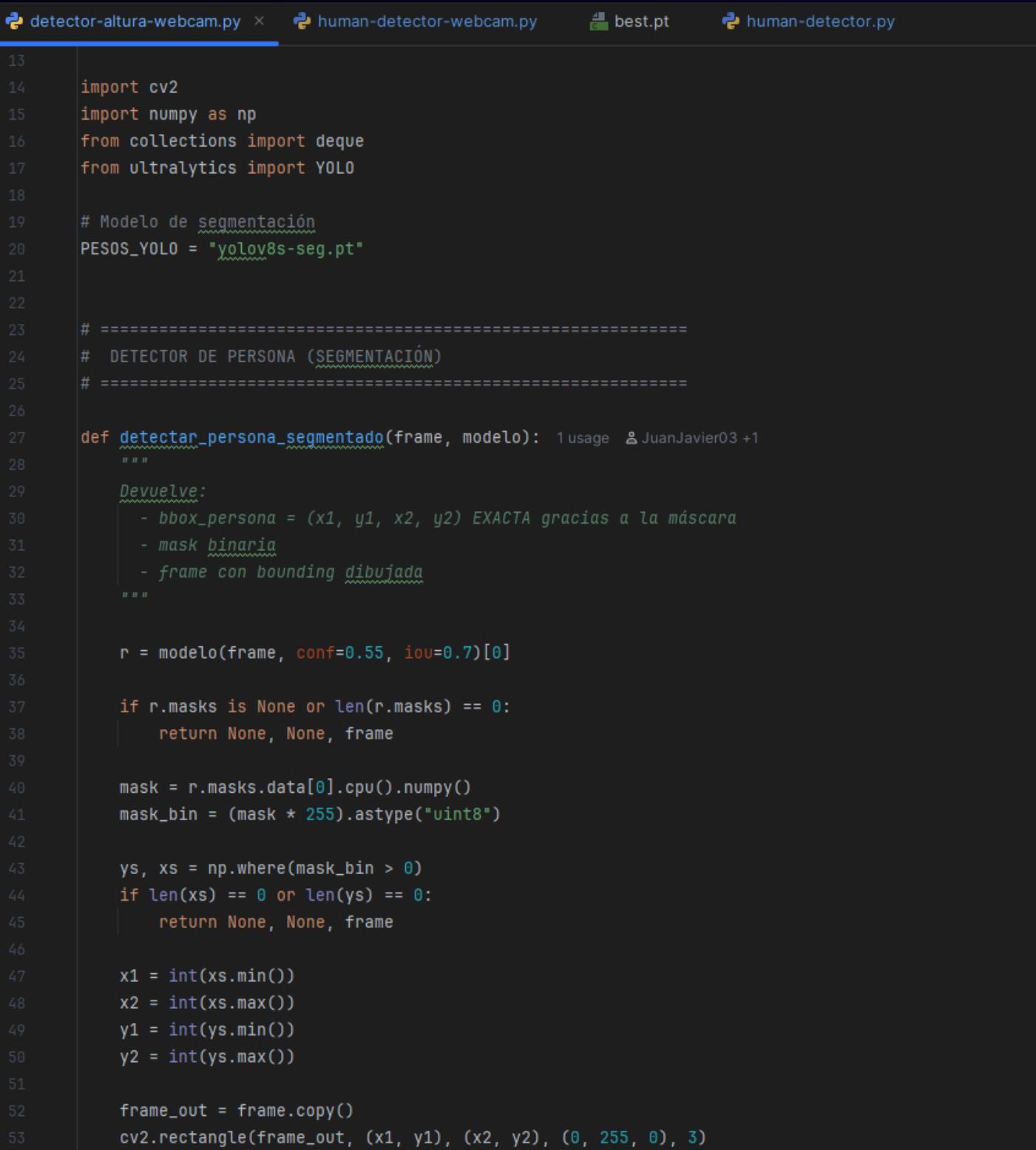
try:
    _ = detectar_personas_y_mostrar(ruta_imagen)
except FileNotFoundError:
    print(f"Error: no se encontró la imagen en la ruta: {ruta_imagen}")
    sys.exit(1)
except Exception as e:
    print("Ocurrió un error inesperado durante la detección:")
    print(e)
    sys.exit(1)

if __name__ == "__main__":
    main()

```

## Archivos utilizados

### detector-altura-webcam.py



```
detector-altura-webcam.py ×  human-detector-webcam.py ×  best.pt  human-detector.py
13
14 import cv2
15 import numpy as np
16 from collections import deque
17 from ultralytics import YOLO
18
19 # Modelo de segmentación
20 PESOS_YOLO = "yolov8s-seg.pt"
21
22
23 # =====
24 # DETECTOR DE PERSONA (SEGMENTACIÓN)
25 # =====
26
27 def detectar_persona_segmentado(frame, modelo): 1 usage  ↗ JuanJavier03 +1
28     """
29     Devuelve:
30     - bbox_persona = (x1, y1, x2, y2) EXACTA gracias a la máscara
31     - mask_binaria
32     - frame con bounding dibujada
33     """
34
35     r = modelo(frame, conf=0.55, iou=0.7)[0]
36
37     if r.masks is None or len(r.masks) == 0:
38         return None, None, frame
39
40     mask = r.masks.data[0].cpu().numpy()
41     mask_bin = (mask * 255).astype("uint8")
42
43     ys, xs = np.where(mask_bin > 0)
44     if len(xs) == 0 or len(ys) == 0:
45         return None, None, frame
46
47     x1 = int(xs.min())
48     x2 = int(xs.max())
49     y1 = int(ys.min())
50     y2 = int(ys.max())
51
52     frame_out = frame.copy()
53     cv2.rectangle(frame_out, (x1, y1), (x2, y2), (0, 255, 0), 3)
```

### 3.1 Detección de Humanos

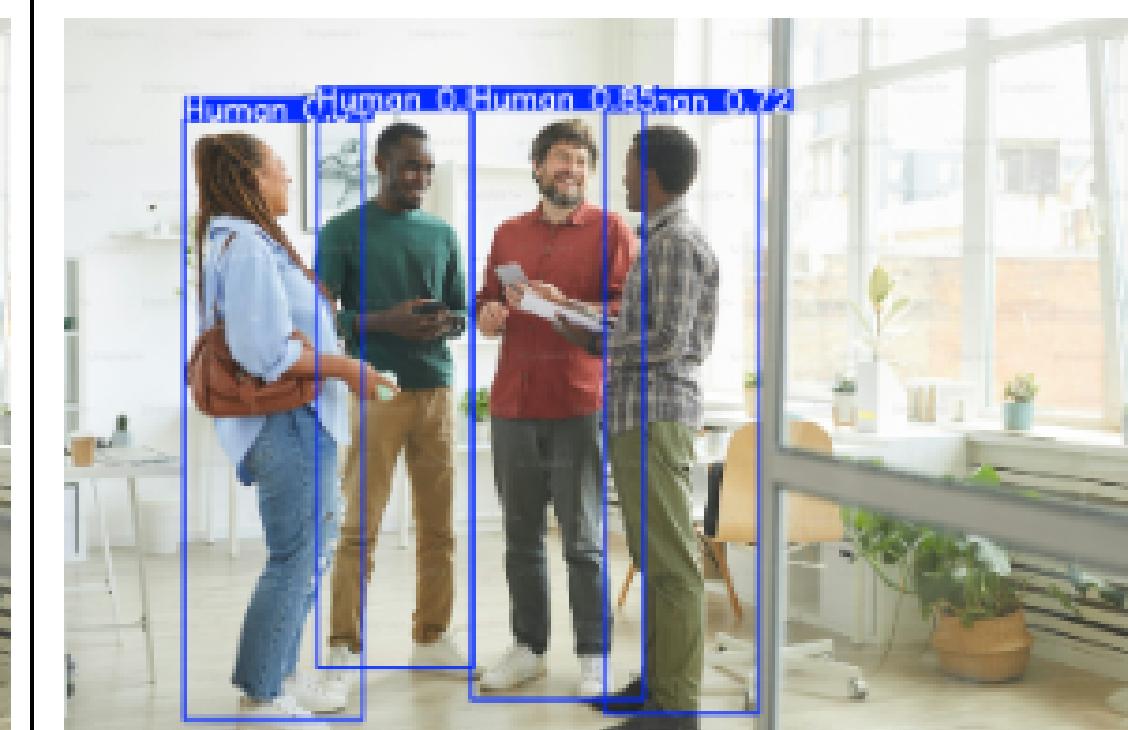
- **Recurso Base:** Adaptación del modelo pre-entrenado best.pt del repositorio YOLOv8-HumanDetection.

**Dataset del Modelo:** Entrenado con 4.200 imágenes, incluyendo data augmentation (rotación, saturación, blur).



### 3.1 Detección de Humanos

- **Lógica del Script (human-detector.py)**
  - **Entrada:** Imagen estática.
  - **Inferencia:** Aplicación del modelo con un umbral de confianza (threshold) de 0.55 para evitar falsos positivos
  - **Salida:** Extracción de coordenadas de las "bounding boxes" ( $x_1, y_1, x_2, y_2$ ).

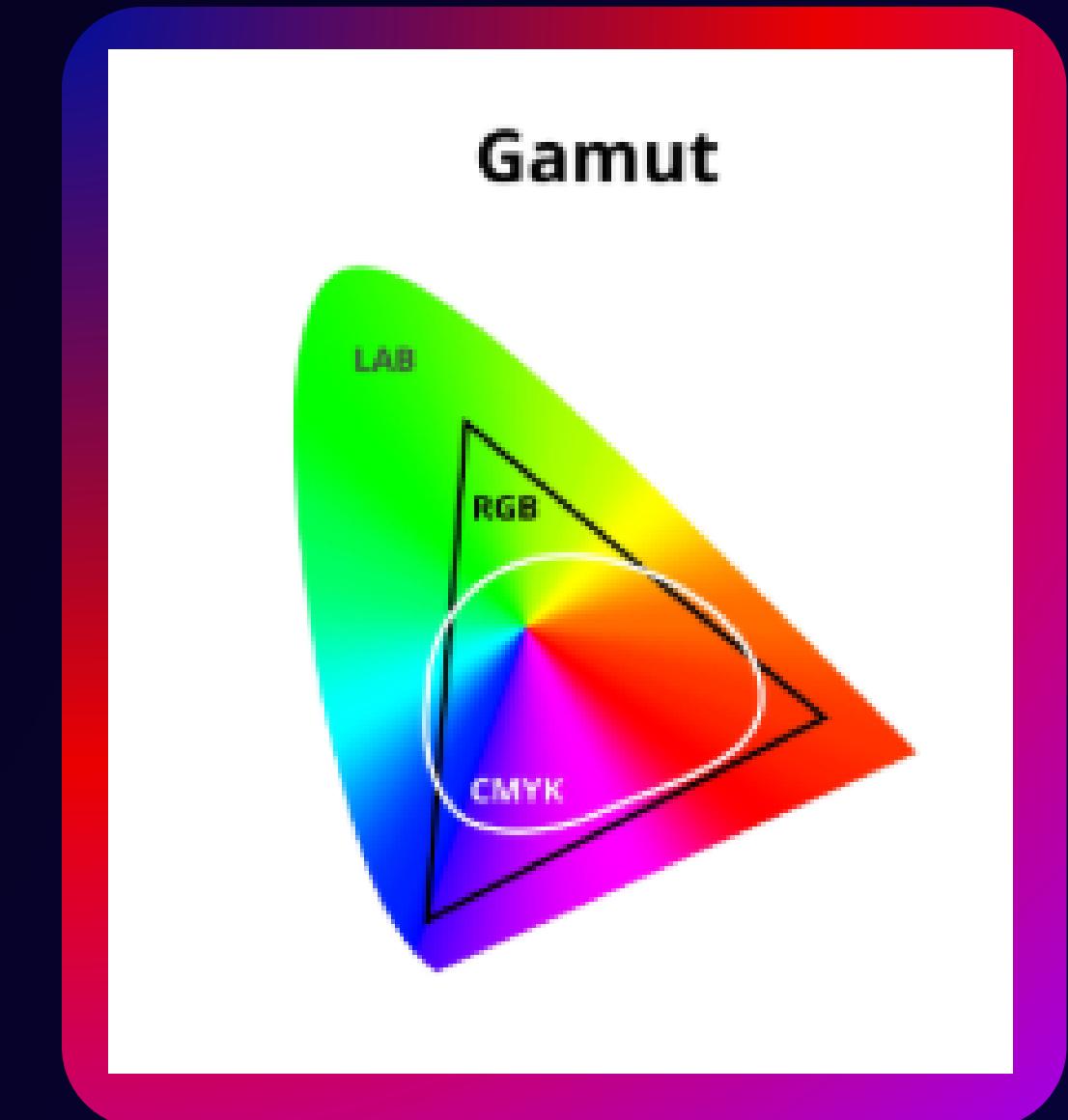


## 3.2 Detección del Referente (Folio A4)

- **Enfoque:** Visión Artificial Clásica (OpenCV) sobre la Región de Interés (ROI), sin Deep Learning en esta etapa..
- **Desafío:** Detectar un objeto blanco bajo iluminación variable.

### 3.2 Detección del Referente (Folio A4)

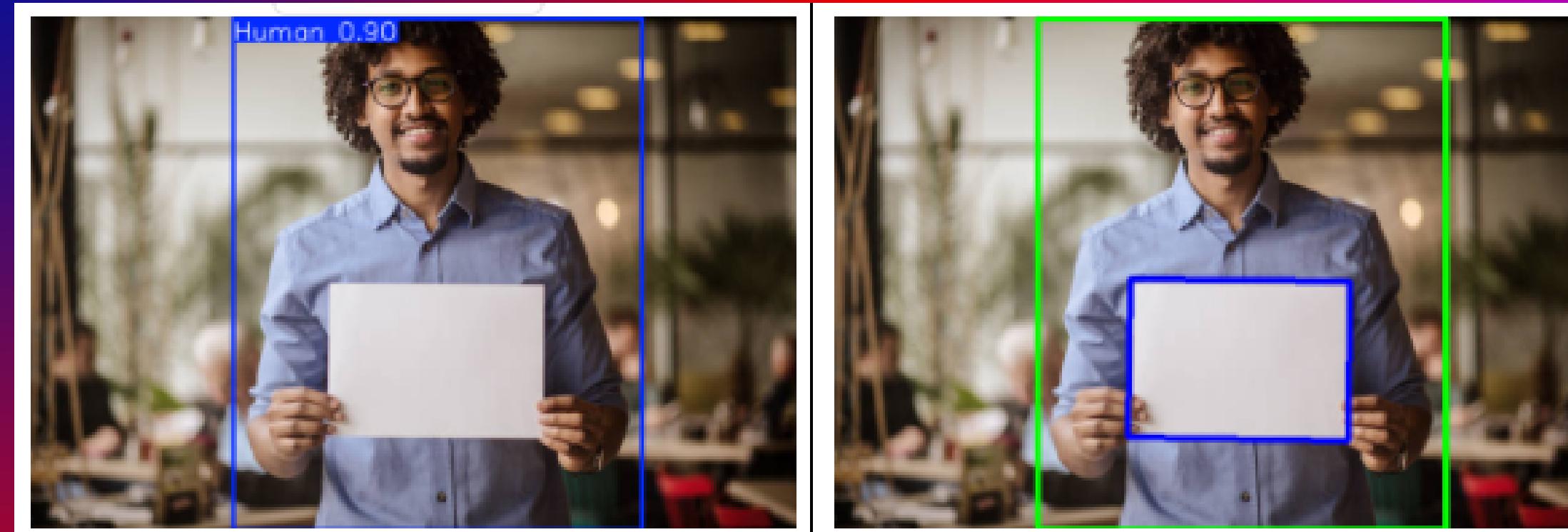
- **Solución de Color (RGB → LAB):**
  - Conversión al espacio de color CIELAB (LAB).
  - Separación de la Luminosidad (L) de la información de color (A, B).
  - Criterio: Búsqueda de objeto con cromaticidad neutra y alta luminosidad



## 3.2 Procesamiento Geométrico y Validación

- **Refinamiento de la Máscara:**
- Uso de detector de bordes Canny seguido de Dilatación para cerrar contornos.
- **Validación Geométrica:**
  - Uso de la función ***minAreaRect*** para obtener el rectángulo rotado mínimo.
  - Filtro de Ratio: El candidato solo se acepta si la proporción corresponde al estándar DIN A4.

## 3.2 Detección del Referente (Folio A4)



Comando de ejecución: **python folio-detector-pasos.py imagen.jpg x1 y1 x2 y2**

### 3.3 Implementación Final: Integración y Estimación

- Ejecución del ***detector-altura.py***
  1. **Detección Primaria (IA):** YOLO localiza al sujeto (confianza > 55% ) .
  2. **Recorte (ROI):** La búsqueda del folio ocurre solo dentro de la bounding box de la persona para reducir ruido .
  3. **Detección Secundaria:** Algoritmo de color LAB y geometría aplicado al recorte.

### 3.3 Implementación Final: Integración y Estimación

- Fórmula de Estimación (Regla de Tres)

$$\text{Altura}_{\text{cm}} = \frac{\text{Altura}_{\text{px\_persona}}}{\text{Largo}_{\text{px\_folio}}} \times 29.7$$

## 3.4 Transición al Tiempo Real: Validación Generalista

- **Justificación Técnica**

- Validación intermedia necesaria antes de trasladar la lógica de medición al vídeo.
- **Desafío:** El sistema debe procesar inferencia y renderizado a 15-30 FPS para fluidez .

- **Implementación (*object-detection.py*)**

- Se utiliza el modelo estándar YOLOv8 Nano **yolov8n.pt** en lugar del especializado **best.pt**.

### 3.4 Transición al Tiempo Real: Validación Generalista

- **Motivos de la Elección:**
- **Velocidad:** El modelo Nano es el más ligero, optimizado para inferencia rápida en hardware limitado (laptop sin GPU).
- **Robustez:** Capacidad de detectar 80 clases (dataset COCO) para probar el sistema ante fondos complejos.

### 3.4 Transición al Tiempo Real: Validación Generalista



Comando de ejecución: **python object-detection.py**

## 3.5 Detección Especializada en Tiempo Real

- **Fusión de Componentes (*human-detector-webcam.py*):**
  - Integración de la lógica de filtrado estricta en el bucle de vídeo.
  - Utilización del modelo adaptado *best.pt*.
- **Configuración Técnica:**
  - Resolución forzada de 1280x720 píxeles para mejor definición.
  - Mantenimiento de hiperparámetros validados ( $\text{conf}=0.55$ ,  $\text{iou}=0.7$ ).
  - Salida visual e impresión de coordenadas en tiempo real

## 3.6 Evaluación de Resultados y Limitaciones

- **Resultado de la Expansión:**
  - Teóricamente resolvió el problema geométrico.
  - En la práctica: Baja tasa de detección.
- **Causa del Fallo:** La ampliación del área introdujo "ruido" visual excesivo, confundiendo al algoritmo clásico de color y bordes.
- **Decisión:** Pivatar hacia una nueva solución.

### 3.7. Refinamiento del Sistema: Geometría, Segmentación y Nuevos Parámetros

- **Nuevo Modelo: YOLOv8-SEG (Segmentación)**
  - **Transición:** Se sustituyó la detección por cajas (bounding boxes) por segmentación de máscaras.
  - **Ventaja Crítica:** La máscara define el contorno exacto, permitiendo localizar el píxel real más alto (cabeza) y más bajo (pies) sin los márgenes de error de las cajas predictivas

### 3.7. Refinamiento del Sistema: Geometría, Segmentación y Nuevos Parámetros

#### Corrección Geométrica

- **Error Sistemático:** La proyección en perspectiva hace que la altura parezca menor porque los pies están más cerca de la cámara que el cuerpo.
- **Solución:** Se introdujeron las variables  $z$  (distancia cámara-persona) y  $L$  (longitud del pie) para compensar este acortamiento visual .

### 3.7. Refinamiento del Sistema: Geometría, Segmentación y Nuevos Parámetros

#### Corrección del Folio (Recuperación Ratio A4)

- **Limitación Previa:** La función `minAreaRect` no corregía la distorsión por inclinación del folio.
- **Algoritmo de Ajuste:** Se implementó una reconstrucción geométrica que obliga al rectángulo detectado a recuperar la proporción 1.414 del formato A4.
- **Beneficio:** Permite medir correctamente incluso si el usuario sostiene el folio inclinado o rotado.

## 3.7. Refinamiento del Sistema: Geometría, Segmentación y Nuevos Parámetros

### Eliminación de la Suela del Calzado

Se añadió un parámetro configurable que descuenta automáticamente la altura de la suela del cálculo final

## 3.7. Refinamiento del Sistema: Geometría, Segmentación y Nuevos Parámetros

### Parametrización Dinámica

- Flexibilidad
- Adaptabilidad

### 3.7. Refinamiento del Sistema: Geometría, Segmentación y Nuevos Parámetros

#### Resultado Final

- La integración de la segmentación precisa con las correcciones de perspectiva generó un modelo geométrico sólido.
- Las pruebas validaron que el sistema entrega mediciones coherentes y estables en tiempo real, superando las limitaciones de la visión clásica

# EXPERIMENTACIÓN

## Metodología Experimental

- **Referencia Real:** Alturas reales de los miembros del equipo para el cálculo del error:
  - Darío: 182 cm
  - Javier: 192 cm
  - Paulo: 180 cm

## 4.1 Validación Inicial en Entorno Controlado

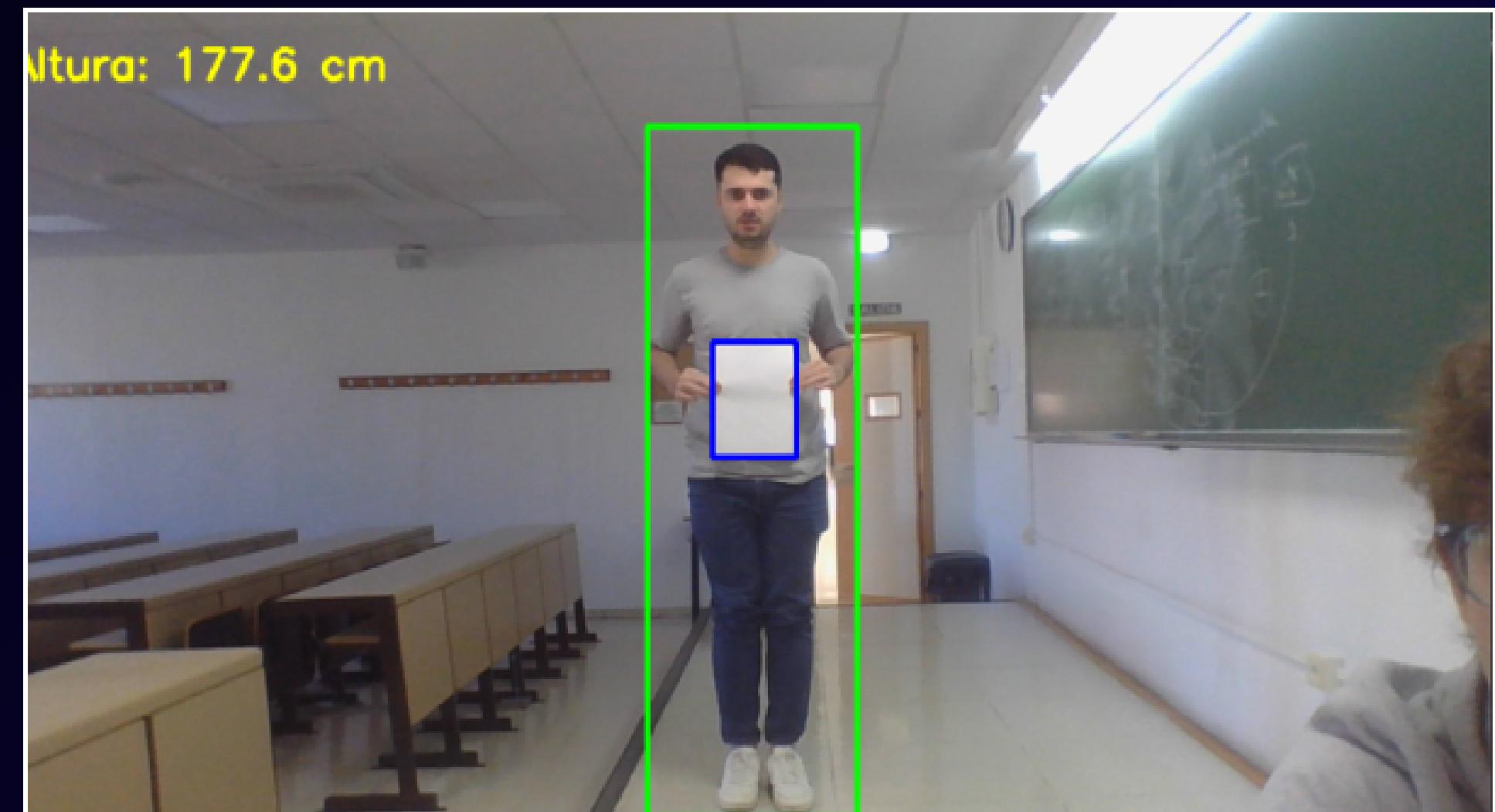
- **Contexto:** Domicilio particular (Pasillo), 21-23 de noviembre.
- **Variables Probadas:**
  - Distancia a la cámara (de 2m a +3m).
  - Condiciones de luz (Día/Noche)

## 4.2. Bloque 2: Pruebas de Integración

**Local y fecha:** Aula de clase A2.10, 26 de noviembre (Seguimiento 3)

### Resultados 1ª sesión

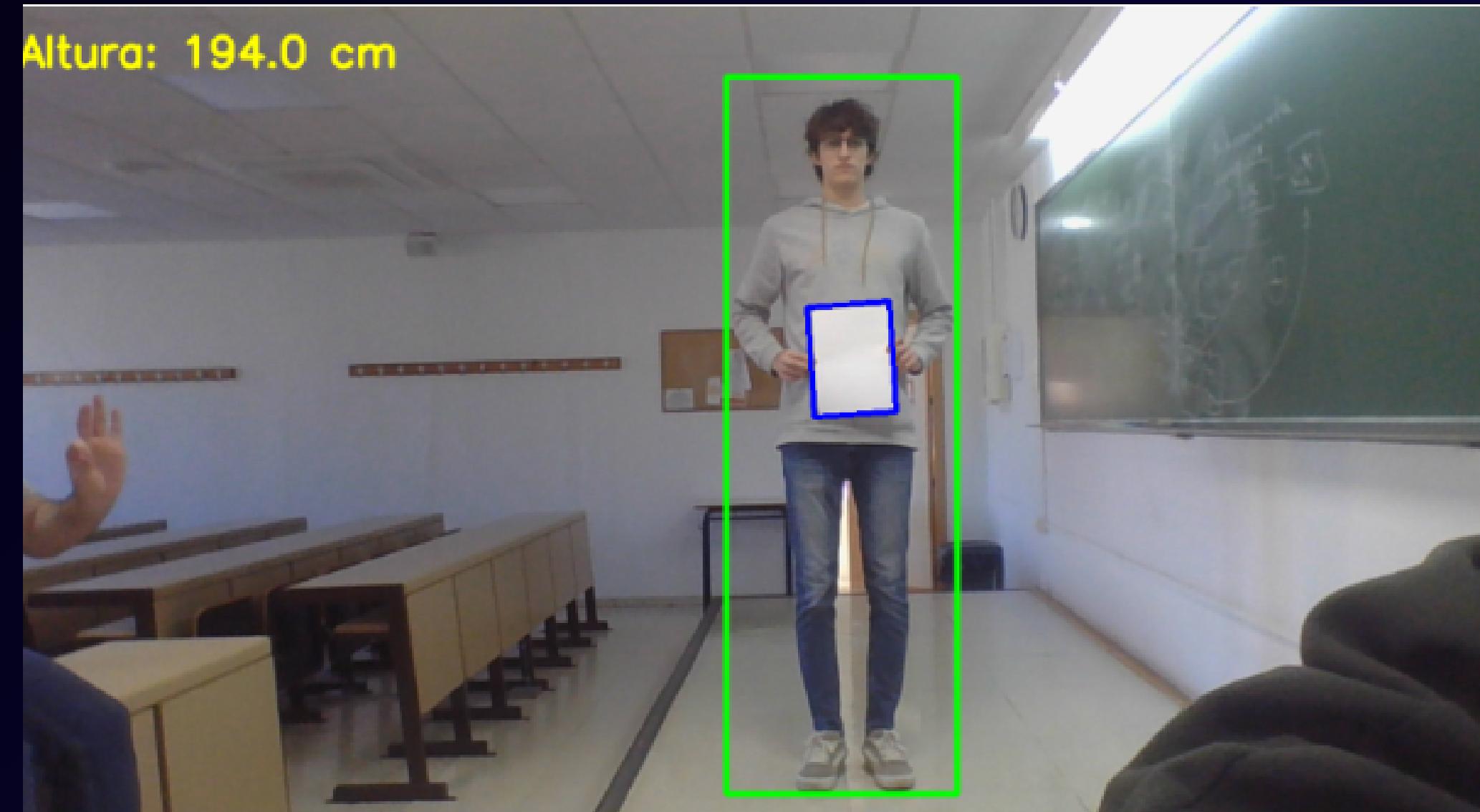
Sujeto Paulo (177cm):  
**Detección ~180-181cm  
(Aceptable).**



## 4.2. Bloque 2: Pruebas de Integración

### Resultados 1ª sesión

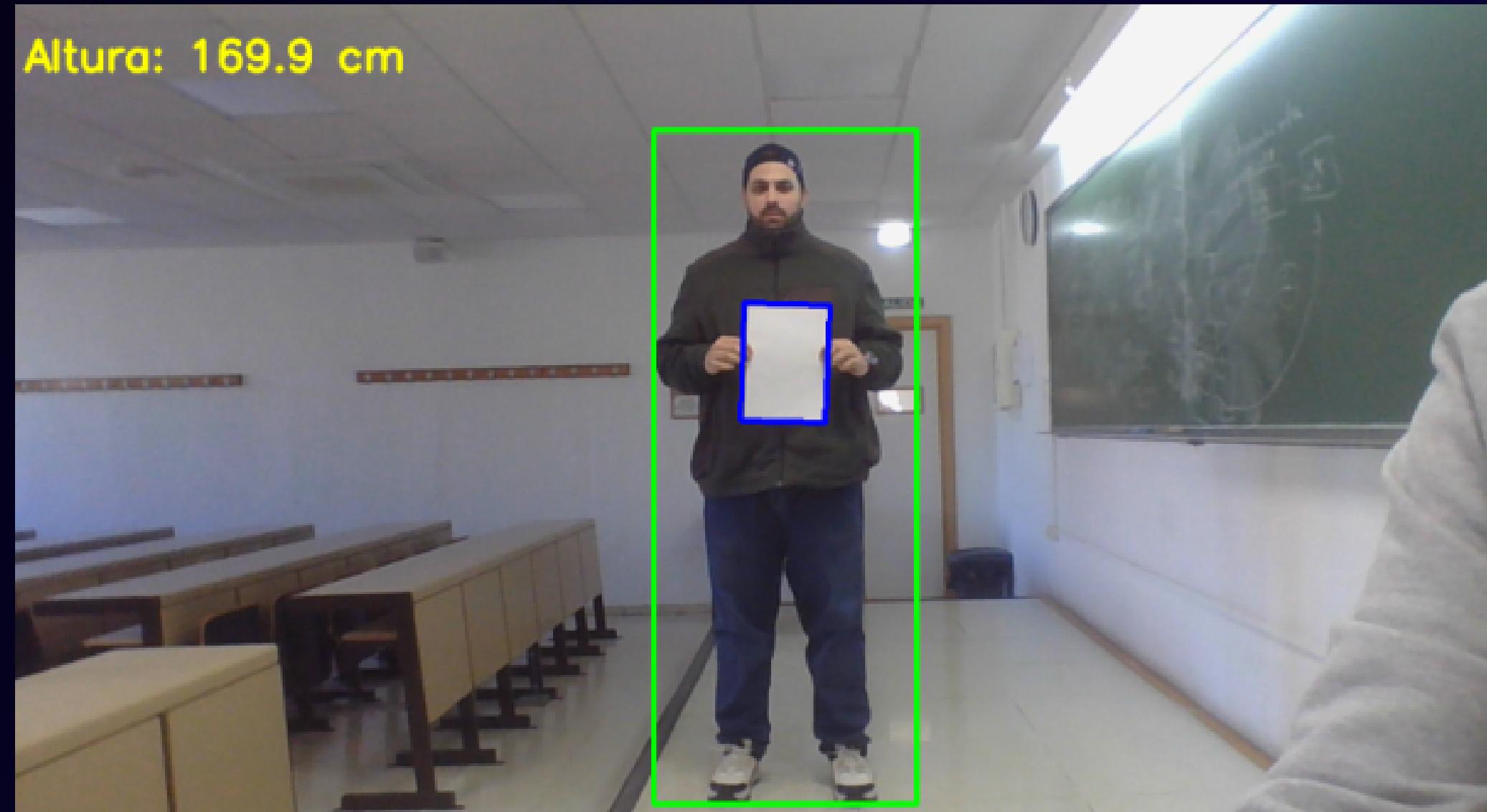
Sujeto Javier (194cm):  
**Error mínimo de +2 cm  
aprox. (~192cm).**



## 4.2. Bloque 2: Pruebas de Integración

### Resultados 1ª sesión

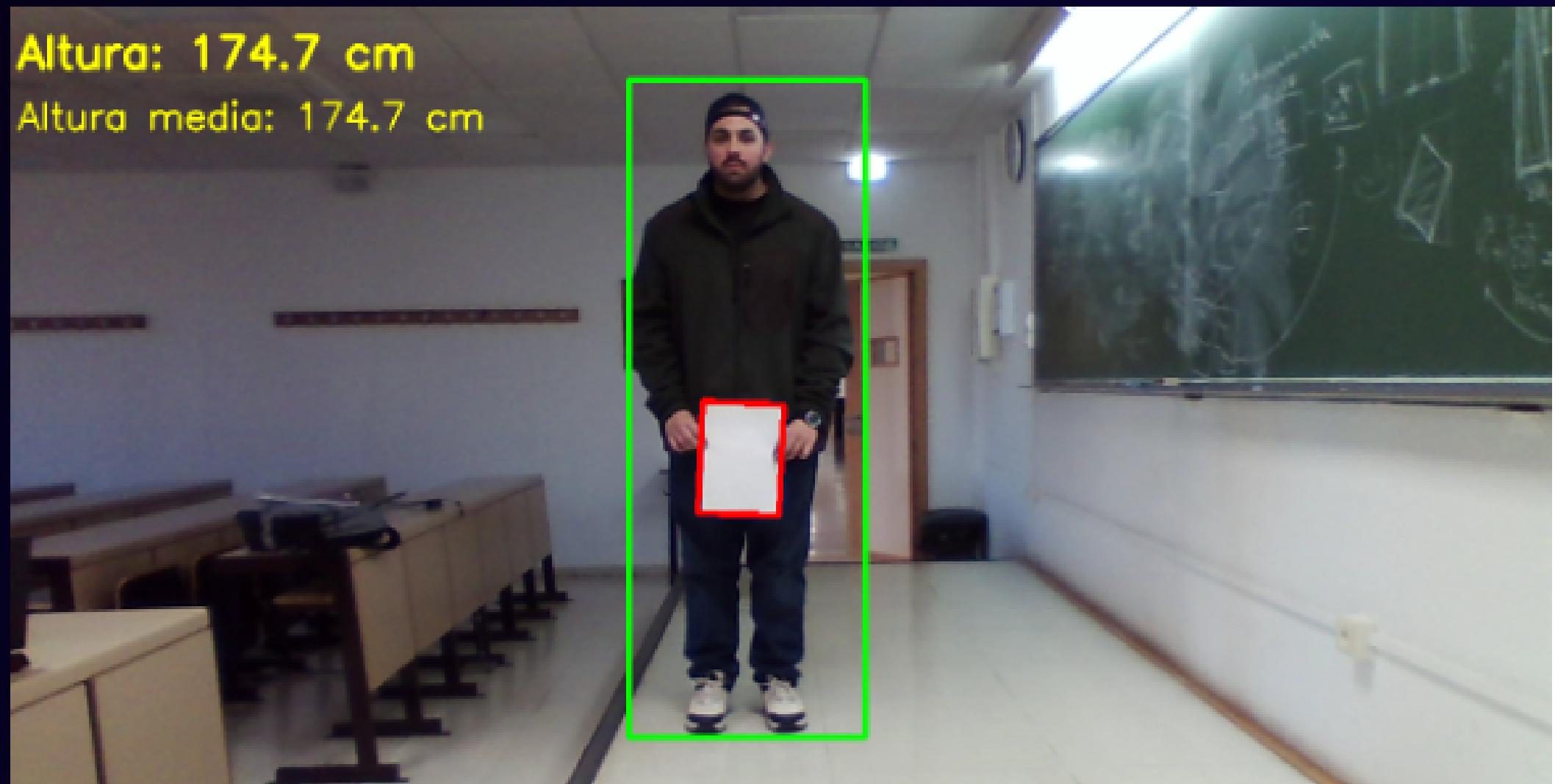
Sujeto Darío (169cm detectado vs 182cm real): **Fallo Crítico.**



## 4.2. Bloque 2: Pruebas de Integración

### Resultados 2ª sesión

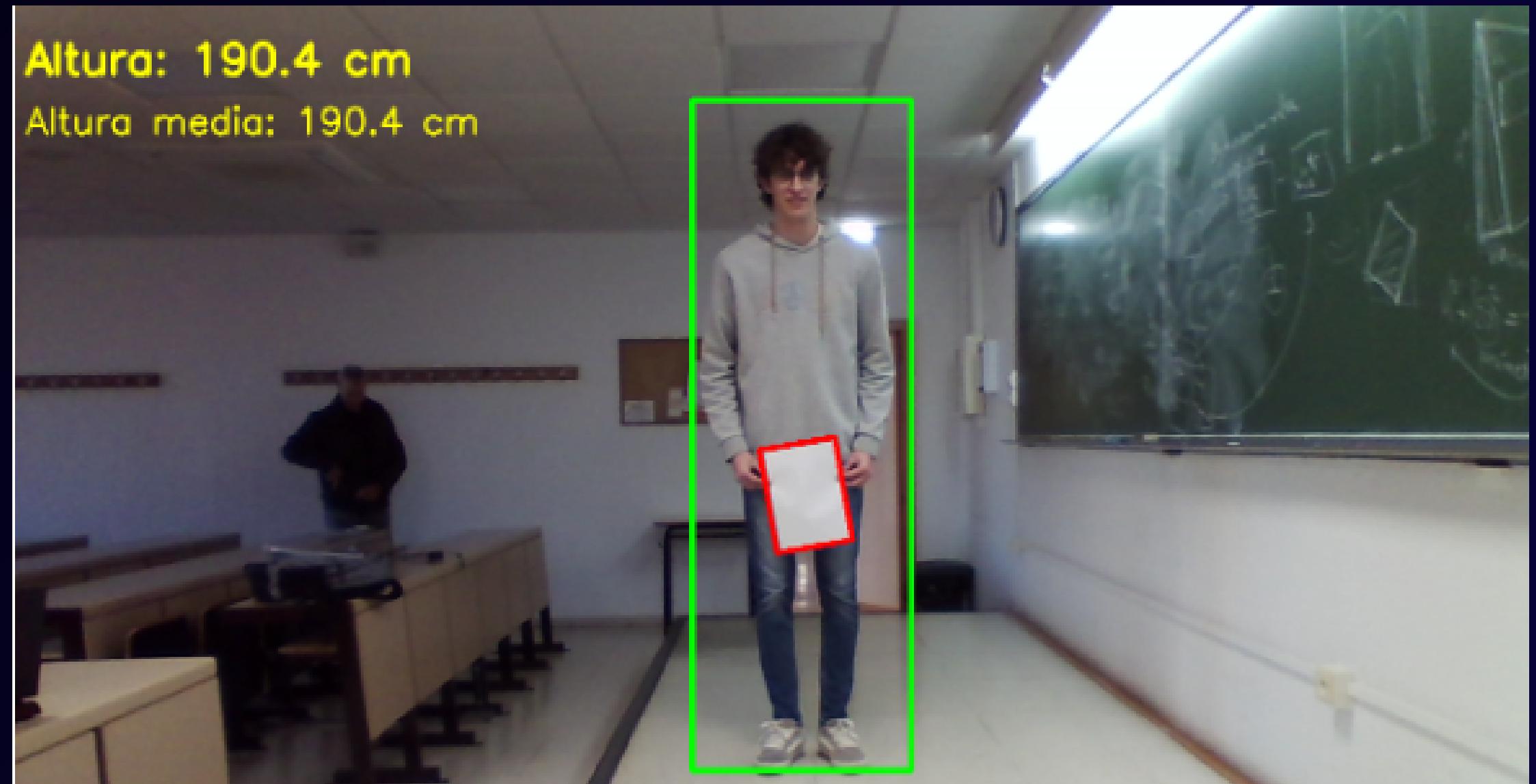
Sujeto Darío: 174,7cm  
detectado vs 182cm real



## 4.2. Bloque 2: Pruebas de Integración

### Resultados 2ª sesión

Sujeto Javier (190,4cm):  
**Error mínimo de -2 cm  
aprox. (~192cm).**



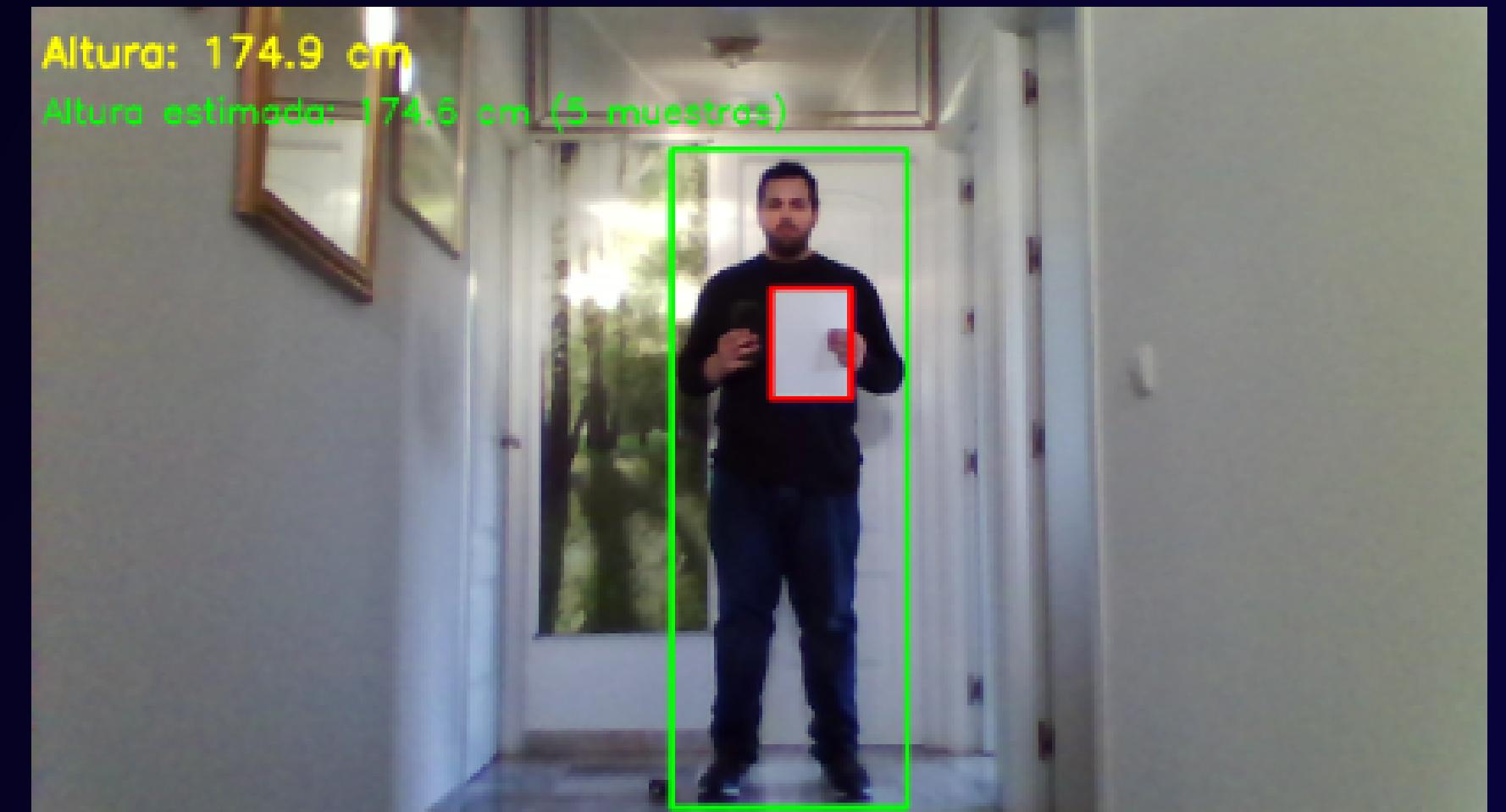
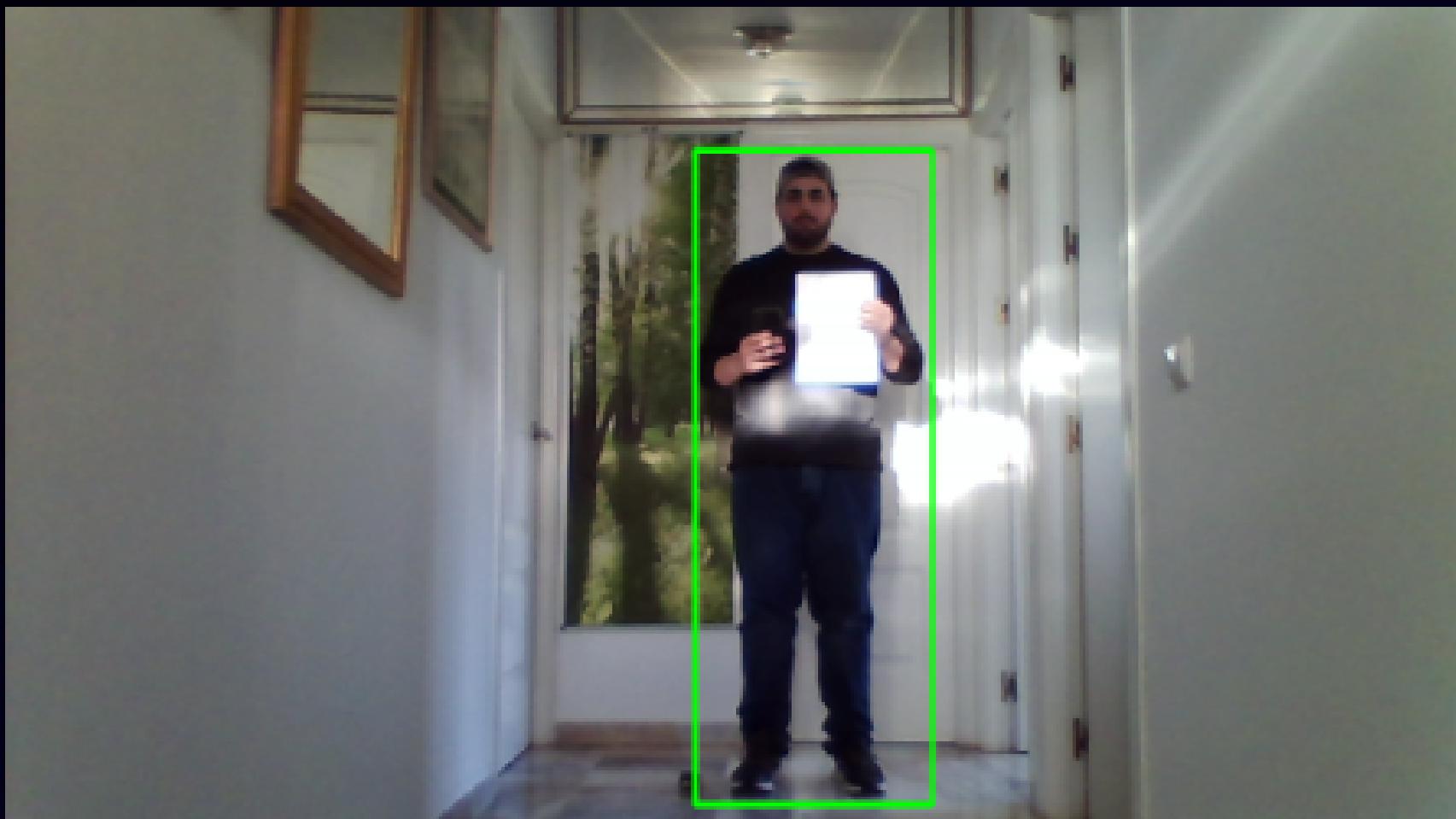
## 4.3. Bloque 3: Validación de Mejoras (ROI Expandido y Mediana)

- **Local y Fecha:** Domicilio, 2 de diciembre.
- **Objetivo:** Probar la expansión del ROI (Solución para el caso Darío) y filtro de mediana

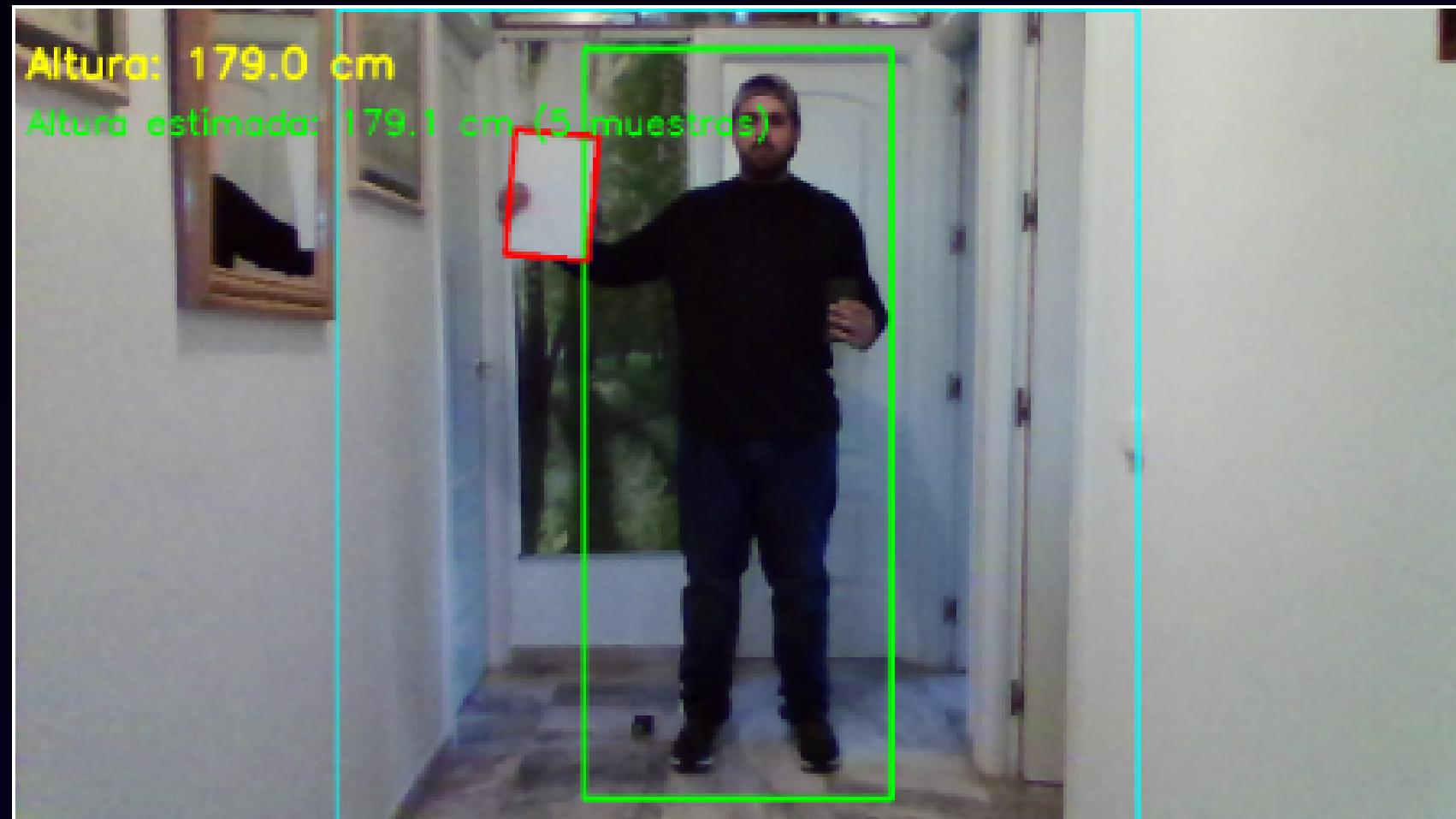
## 4.3. Resultados del bloque 3



## 4.3. Resultados del bloque 3



## 4.3. Resultados del bloque 3



## 4.2. Bloque 4: Validación Final y Comparativa de Escenarios

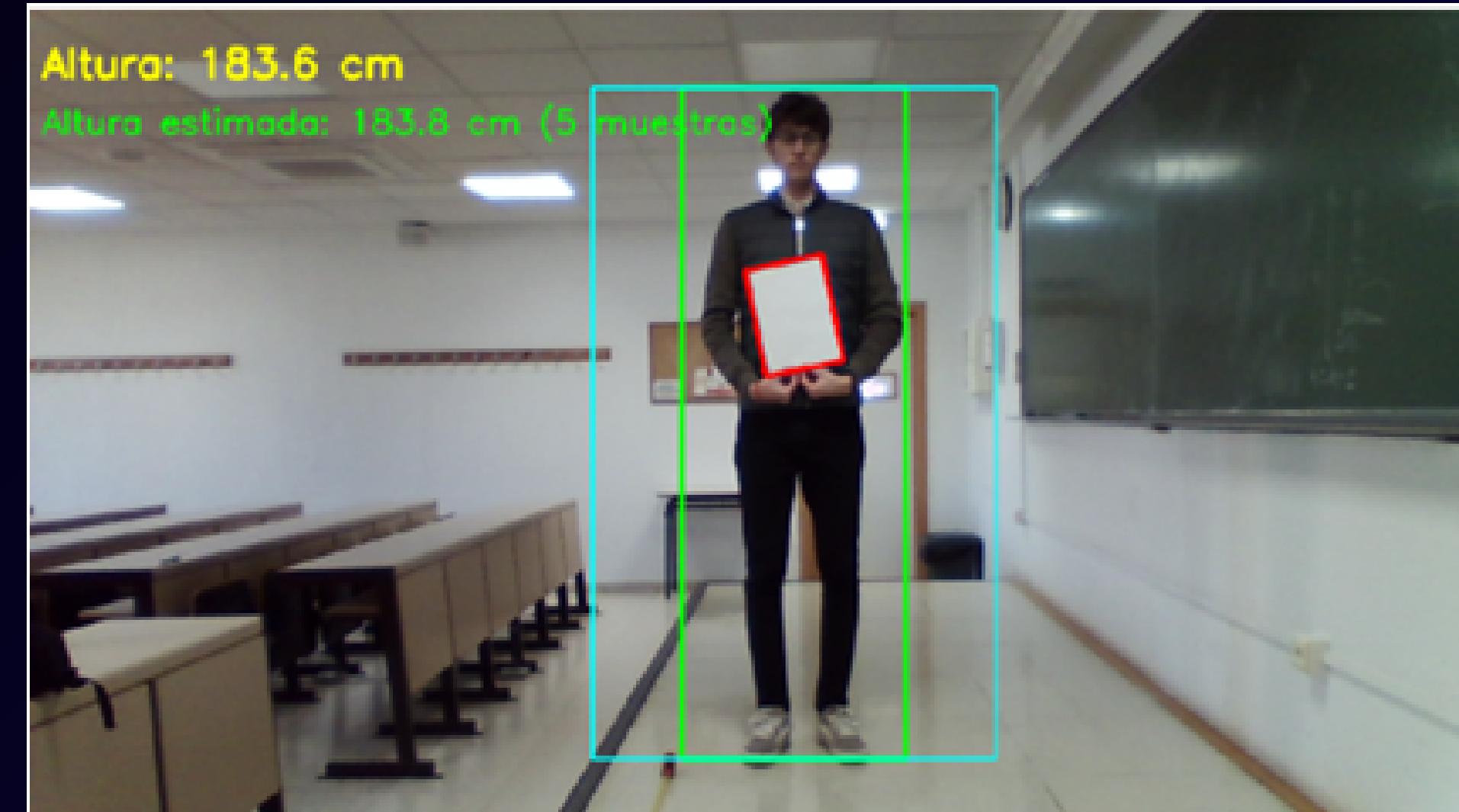
**Local y fecha:** Aula de clase A2.10, 03 de diciembre



## 4.2. Bloque 4: Validación Final y Comparativa de Escenarios

**Local y fecha:** Aula de clase A2.10, 03 de diciembre

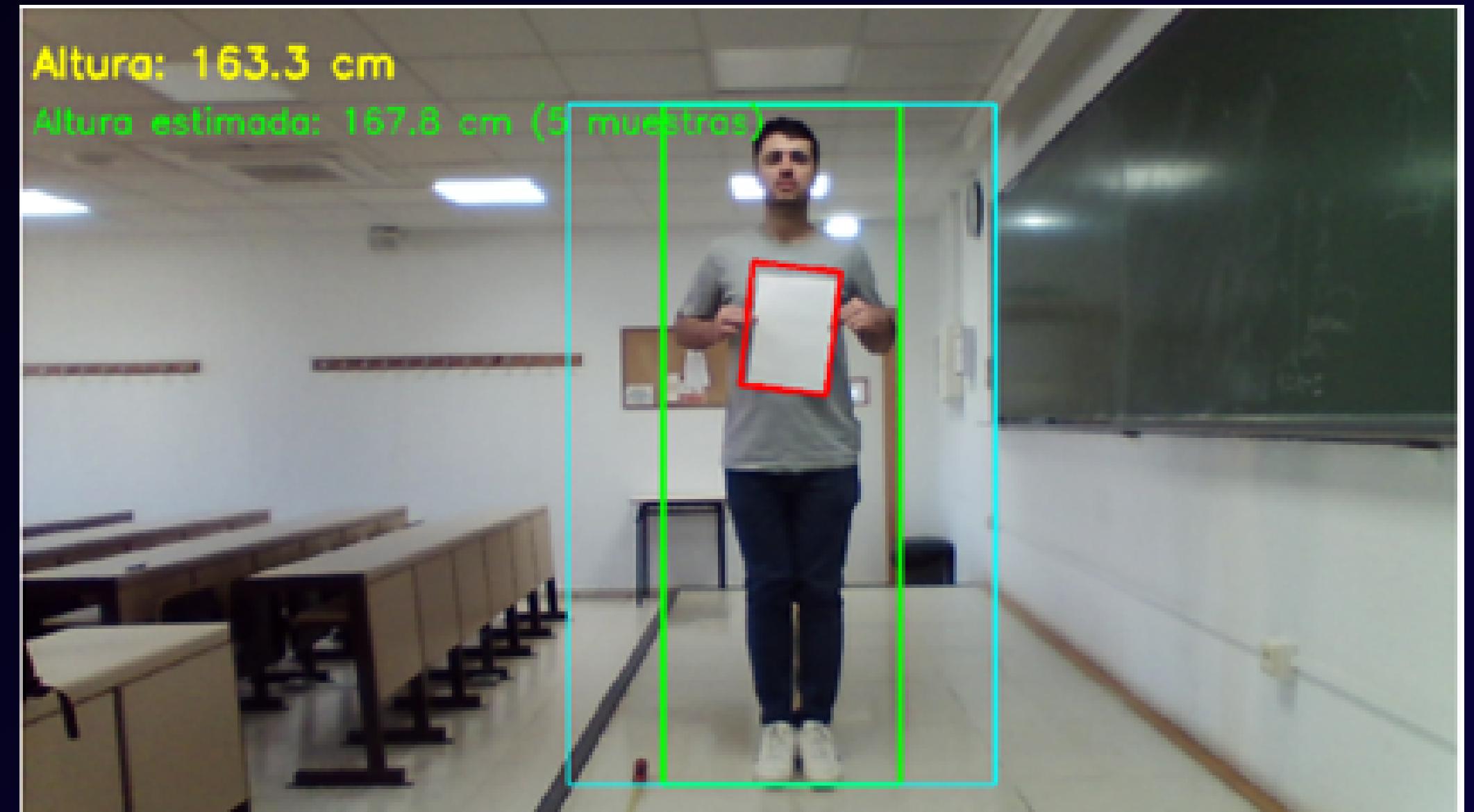
**Escenario iluminado**



## 4.2. Bloque 4: Validación Final y Comparativa de Escenarios

**Local y fecha:** Aula de clase A2.10, 03 de diciembre

**Escenario iluminado**



## 4.2. Bloque 4: Validación Final y Comparativa de Escenarios

**Local y fecha:** Aula de clase A2.10, 03 de diciembre

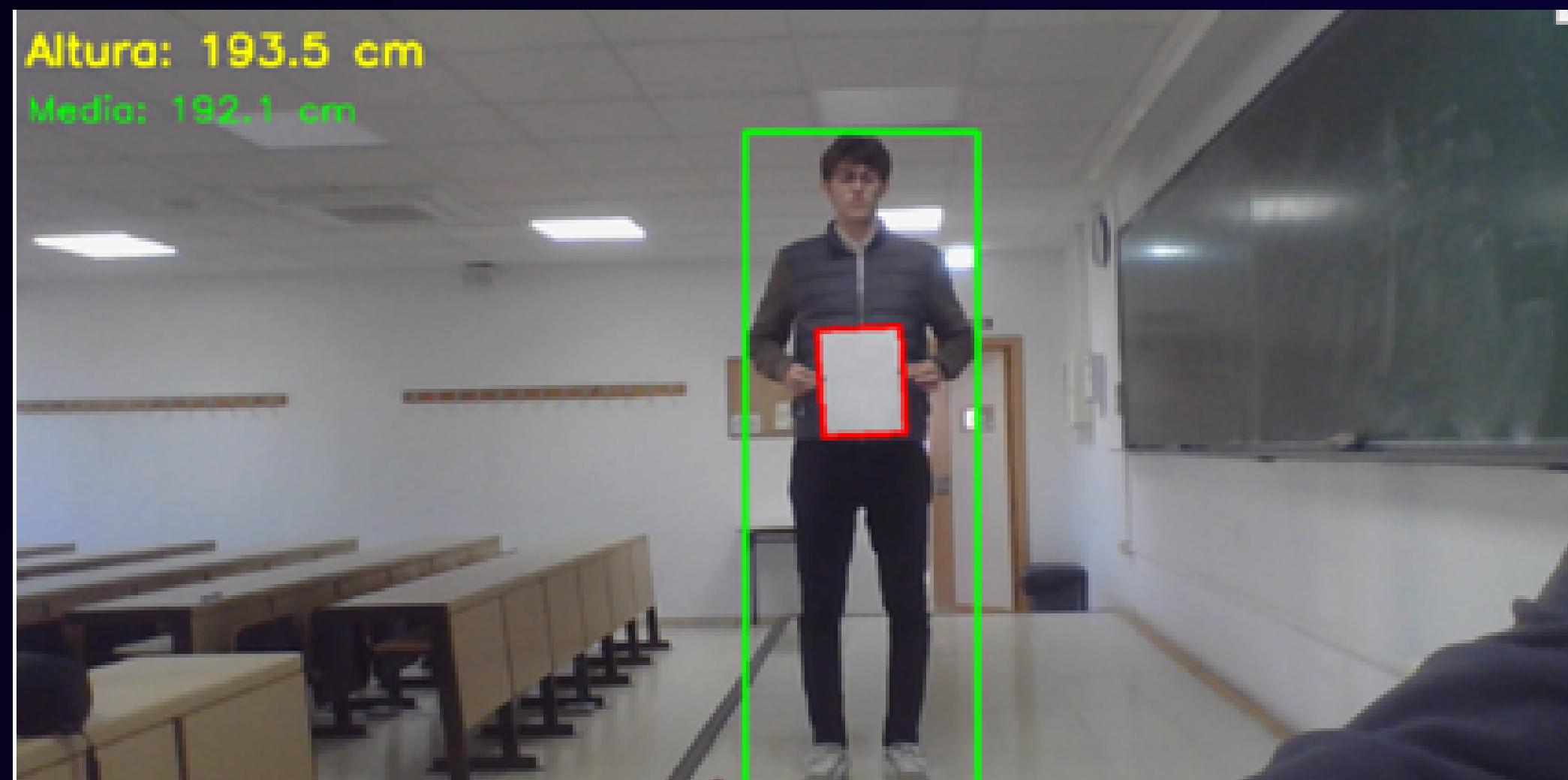
**Entorno iluminado y  
folio torcido**



## 4.2. Bloque 4: Validación Final y Comparativa de Escenarios

**Local y fecha:** Aula de clase A2.10, 03 de diciembre

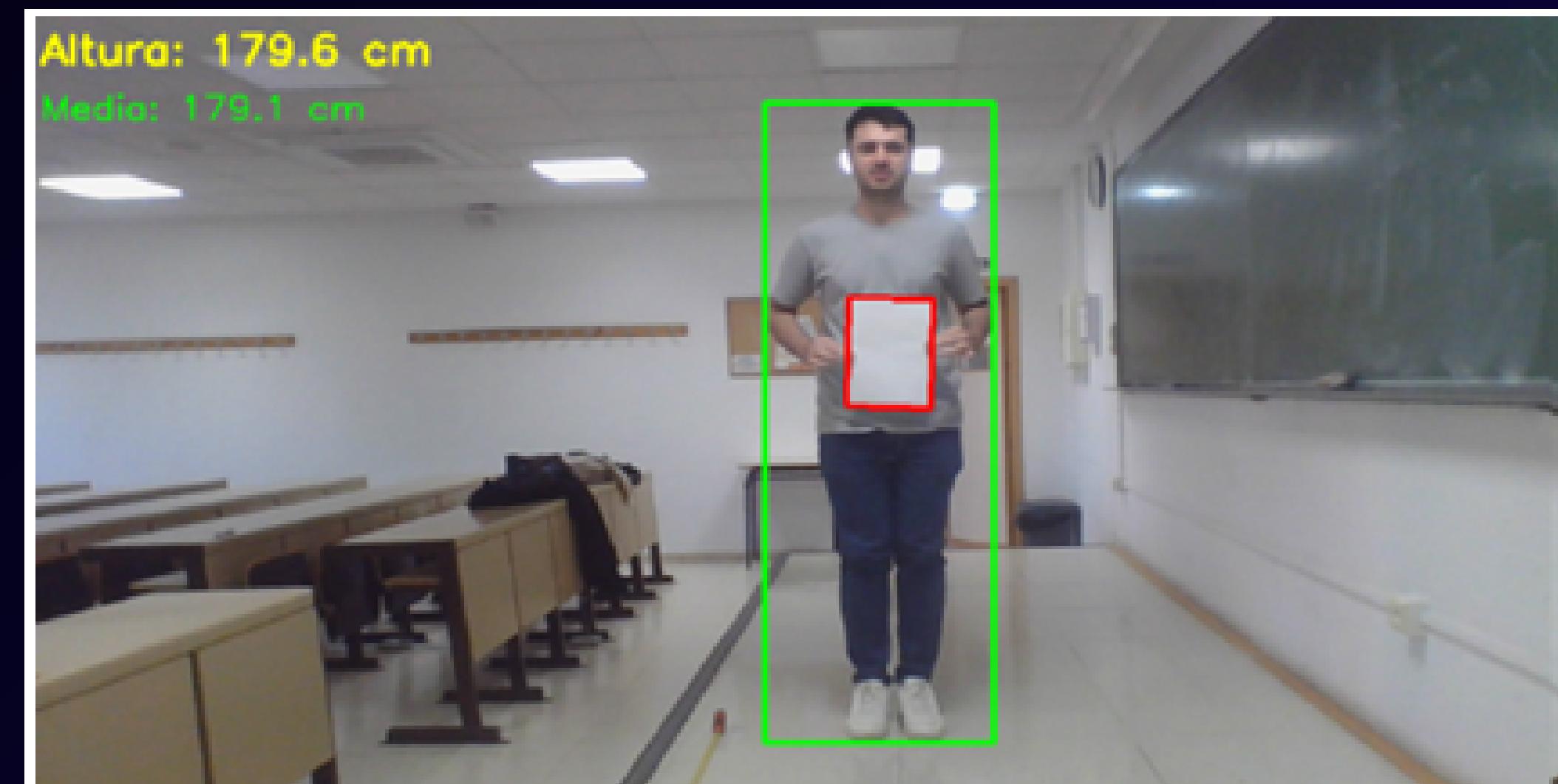
**Entorno iluminado y  
folio recto.**



## 4.2. Bloque 4: Validación Final y Comparativa de Escenarios

**Local y fecha:** Aula de clase A2.10, 03 de diciembre

**Entorno iluminado y  
postura correcta.**



## Validación Externa

- **Referencia Real:** Alturas reales de los miembros para el cálculo del error:
  - Yue Liu: 168 cm
  - Irene Ledi Leonés Léon: 161 cm
  - Enrique Grijuela Muñoz: 178 cm

## Validación Externa

**Local y fecha:** Aula de clase A2.10, 03 de diciembre

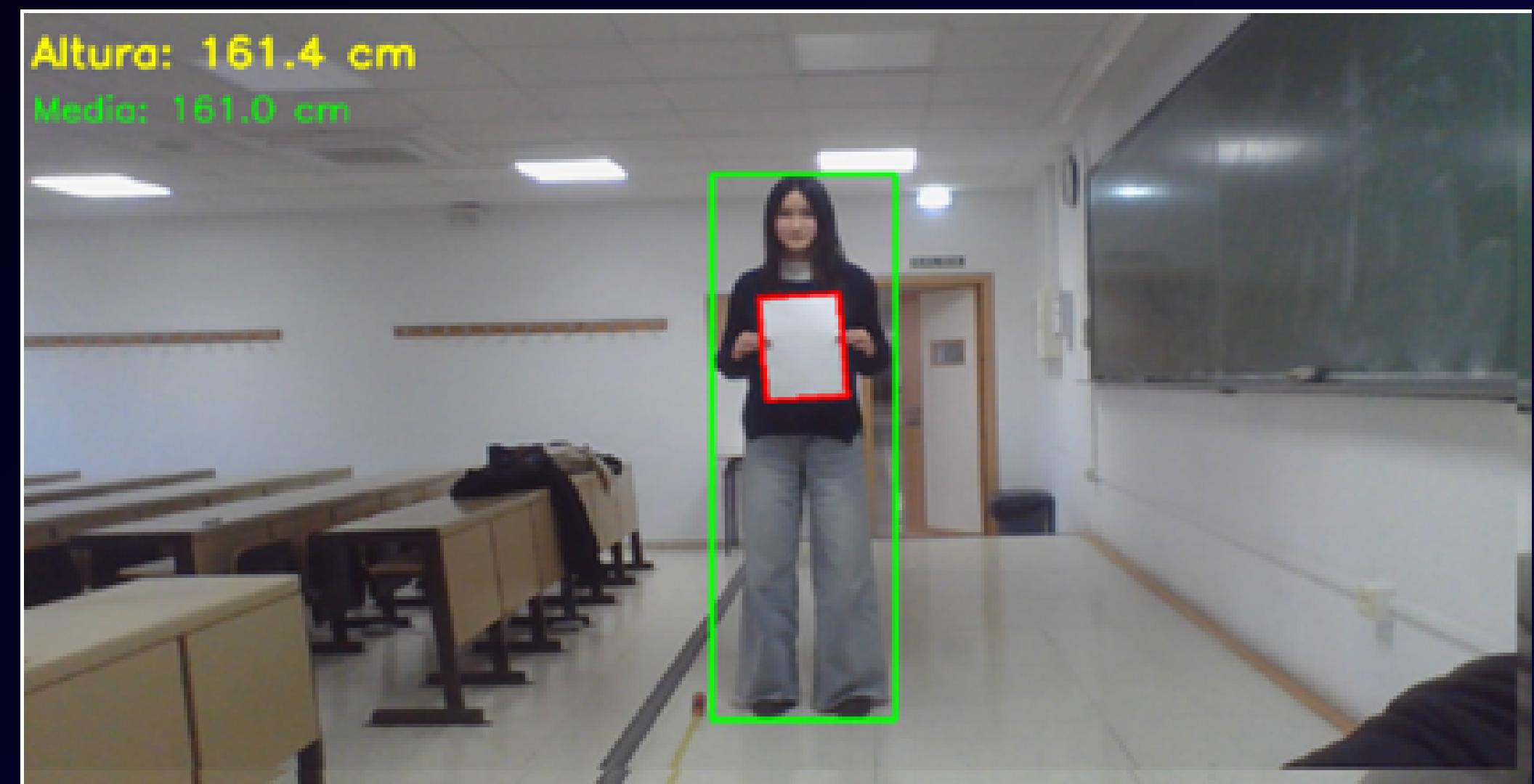
Yue: Altura calculada  
sin ningún error de  
aproximación



## Validación Externa

**Local y fecha:** Aula de clase A2.10, 03 de diciembre

Irene: Altura calculada  
sin ningún error  
presente.



## Validación Externa

**Local y fecha:** Aula de clase A2.10, 03 de diciembre

Enrique: Presenta un error de 1 - 2 cms



# CONCLUSIÓN

## Éxitos de la Implementación

- **Viabilidad del Sistema**
- **Solución de Escala**
- **Decisiones de Diseño Robustas**

## Limitaciones Críticas Encontradas

- **Errores de Profundidad (Depth Error)**
- **Insuficiencia del Modelo 2D:**

**¡Gracias!**

# BIBLIOGRAFIA

GONZALEZ, R. C.; WOODS, R. E. Digital Image Processing. 4.<sup>a</sup> ed. Pearson, 2018. (Capítulos 9 y 10).

CANNY, J. "A Computational Approach to Edge Detection". IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, n.<sup>o</sup> 6, 1986, pp. 679-698.

ULTRALYTICS. "Limiarización no Processamento de Imagens Explicada". Disponible en: <https://www.ultralytics.com/pt/blog/thresholding-in-image-processing> [Consulta: 26 noviembre 2025].

KAIZOUDOU. "From RGB to Lab\* color space". Disponible en: <https://kaizoudou.com/from-rgb-to-lab-color-space/> [Consulta: 26 noviembre 2025].

VISÃO COMPUTACIONAL. "Morfologia Matemática para Processamento de Imagens". Disponible en: <https://visaocomputacional.com.br/morfologia-matematica-para-processamento-de-imagens/> [Consulta: 24 noviembre 2025].

TIR (Universidad Federal de Pernambuco). "7. Morfología Matemática". Disponible en: <https://www.cin.ufpe.br/~tir/ComputacaoCientifica/7.Morfologia%20Matematica.pdf> [Consulta: 24 noviembre 2025].

PROGRAMAR FÁCIL. "Detector de bordes Canny, cómo contar objetos con OpenCV y Python". Disponible en: <https://programarfacil.com/blog/vision-artificial/detector-de-bordes-canny-opencv/> [Consulta: 25 noviembre 2025].

OPENCV. "Contour Features". Disponible en: [https://docs.opencv.org/4.x/dd/d49/tutorial\\_py\\_contour\\_features.html](https://docs.opencv.org/4.x/dd/d49/tutorial_py_contour_features.html) [Consulta: 25 noviembre 2025].

OPENCV. "Creating Bounding rotated boxes and ellipses for contours". Disponible en: [https://docs.opencv.org/3.4/de/d62/tutorial\\_bounding\\_rotated\\_ellipses.html](https://docs.opencv.org/3.4/de/d62/tutorial_bounding_rotated_ellipses.html) [Consulta: 25 noviembre 2025].

HALL OF PRINT. A4 Paper Size Guide. Disponible en: <https://www.hallofprint.com/a4-paper-size-guide> [Consulta: 23 noviembre 2025].

ORGANIZACIÓN INTERNACIONAL DE NORMALIZACIÓN (ISO). ISO 216:2007 Writing paper and certain classes of printed matter – Trimmed sizes – A and B series, and indication of machine direction. 2007.

STACKOVERFLOW. "What's the difference in results of cvBoundingRect and cvMinAreaRect?". Disponible en: <https://stackoverflow.com/questions/69911364/whats-the-difference-in-results-of-cvboundingrect-and-cvminarearect> [Consulta: 25 noviembre 2025].

# BIBLIOGRAFIA

Jocher, G., Chaurasia, A., & Qiu, J. (2023). YOLO by Ultralytics. Ultralytics.

ASKPYTHON. "3 Effective Methods for Applying Gaussian Filters to Images". AskPython [en línea]. [Consulta: 26 noviembre 2025]. Disponible en: <https://www.askpython.com/python-modules/applying-gaussian-filters-to-images>

RESEARCHGATE. "Figura 4: Supresión No Máxima a) Persona No. 1 b) Persona No. 2 y c) Persona No. 3". ResearchGate [en línea]. [Consulta: 26 noviembre 2025]. Disponible en: [https://www.researchgate.net/figure/Figura-4-Supresion-No-Maxima-a-Persona-No-1-b-Persona-No-2-y-c-Persona-No-3\\_fig4\\_272181378](https://www.researchgate.net/figure/Figura-4-Supresion-No-Maxima-a-Persona-No-1-b-Persona-No-2-y-c-Persona-No-3_fig4_272181378)

RESEARCHGATE. "Figura 5: Histéresis de Umbral a) Persona No. 1 b) Persona No. 2 y c) Persona No. 3". ResearchGate [en línea]. [Consulta: 26 noviembre 2025]. Disponible en: [https://www.researchgate.net/figure/Figura-5-Histeresis-de-Umbral-a-Persona-No-1-b-Persona-No-2-y-c-Persona-No-3\\_fig5\\_272181378](https://www.researchgate.net/figure/Figura-5-Histeresis-de-Umbral-a-Persona-No-1-b-Persona-No-2-y-c-Persona-No-3_fig5_272181378)

JISHUZHAN (KYLE). "OpenCV---minAreaRect". Jishuzhan [en línea]. 31 mayo 2025. Disponible en: <https://jishuzhan.net/article/1928627775321190401> [Consulta: 6 diciembre 2025].

MATHWORKS. "imfill - Fill image regions and holes". MATLAB Documentation [en línea]. Disponible en: <https://la.mathworks.com/help/images/ref/imfill.html> [Consulta: 5 diciembre 2025].

MATHWORKS. "Types of Morphological Operations - Morphological Dilation and Erosion". MATLAB & Simulink Documentation [en línea]. Disponible en: <https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html> [Consulta: 2 diciembre 2025].

J3LLY-BEEN. YOLOv8-HumanDetection [software]. Versión 1.0. GitHub, 2024. Licencia GPL-3.0. Disponible en: <https://github.com/J3lliy-Been/YOLOv8-HumanDetection> [Consulta: 5 diciembre 2025].

JUANJAVIER03. Human-Height-Detection-Using-Reference [software]. GitHub, 2024. Disponible en: <https://github.com/JuanJavier03/Human-Height-Detection-Using-Reference> [Consulta: 25 de noviembre 2025].