Corrección de Perspectiva con Homografía

Estimador de Velocidad de Pelota - Kristhian Santiago Palomino Fajardo

Febrero 2025

1 Introducción

La **homografía** es una transformación geométrica fundamental para corregir la perspectiva en imágenes cuando la cámara no está alineada ortogonalmente al plano de movimiento de la pelota. En este documento, se proporciona una implementación detallada para integrar esta corrección en el código ya existente del proyecto de estimación de velocidad de la pelota.

2 Relevancia de la Homografía en la Estimación de Velocidad

Cuando la cámara no está perfectamente alineada con el plano del movimiento de la pelota, las distancias en píxeles no reflejan con precisión la distancia real recorrida. Esto introduce errores en la conversión de píxeles a metros y afecta la estimación de la velocidad. Para solucionar este problema, se debe aplicar una transformación de homografía para rectificar la imagen antes de procesar los cuadros.

3 Fundamentos Matemáticos de la Homografía

La homografía es una transformación **proyectiva** que se define mediante una matriz de 3×3:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = H \begin{bmatrix} x \\ y \\ w \end{bmatrix} \tag{1}$$

Donde:

- (x, y) son las coordenadas originales de un punto en la imagen.
- (x', y') son las coordenadas corregidas tras la transformación.
- w' es un factor de normalización.
- H es la matriz de homografía calculada a partir de al menos 4 pares de puntos de correspondencia.

4 Implementación en el Código Existente

Para integrar la homografía en el código del proyecto, se deben seguir los siguientes pasos:

4.1 Definir Puntos de Correspondencia

Se deben seleccionar al menos cuatro puntos en la imagen original y sus correspondientes en la imagen corregida. Esto se puede hacer de manera manual o mediante detección automática si se tienen referencias conocidas en la escena.

4.2 Cálculo de la Matriz de Homografía en OpenCV

Se utiliza la función `cv2.findHomography()` para calcular la matriz de transformación:

```
import cv2
import numpy as np

def compute_homography():
    pts_src = np.array([[100, 50], [400, 50], [50, 300], [450, 300]], dtype=np.float3
    pts_dst = np.array([[100, 50], [400, 50], [100, 300], [400, 300]], dtype=np.float
    H, _ = cv2.findHomography(pts_src, pts_dst)
    return H

def apply_homography(frame, H):
    height, width = frame.shape[:2]
    return cv2.warpPerspective(frame, H, (width, height))
```

4.3 Integración en la Clase BallTracker

Dentro del código del `BallTracker`, la homografía debe ser aplicada a cada cuadro antes de procesar la detección de la pelota. Se modifica la función de preprocesamiento en el archivo correspondiente:

```
class BallTracker:
    def __init__(self, source=0, output_path=None):
        self.video = cv2.VideoCapture(source)
        self.H = compute_homography()
        ...

    def process_frame(self):
        ret, frame = self.video.read()
        if not ret:
            return None, None

        frame = apply_homography(frame, self.H)
        ...  # Continuar con la detección de la pelota
```

4.4 Ajuste de Líneas de Referencia

Después de aplicar la homografía, es necesario recalibrar las líneas de referencia para reflejar la escala real en la imagen corregida. Se recomienda medir la nueva distancia en píxeles y ajustar la conversión de unidades en el código.