UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Trabajo fin de grado

Sistema de tracking y reidentificación de jugadores en fútbol amateur

Realizado por
Soriano Muñoz Juan Ignacio

Profesor encargado:
Luque Baena Rafael Marcos
Jerez Aragonés Jose Manuel

Departamento
Lenguajes y Ciencias de la Computación

MÁLAGA, DICIEMBRE de 2024

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

ESTUDIANTES DE INGENIERÍA BIOINFORMÁTICA

Sistema de tracking y reidentificación de jugadores en fútbol amateur

Trabajo fin de grado

Realizado por
Soriano Muñoz Juan Ignacio

Profesor encargado:
Luque Baena Rafael Marcos
Jerez Aragonés Jose Manuel

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, DICIEMBRE DE 2024

Contents

1 Introducción

2 Diario de avances

2.1 Semana 3-16 de marzo)

Por ahora lo que llevamos es un dataset hecho en roboflow con un partido de España contra Suiza. Realizamos capturas y dividimos el conjunto en training, validation y test.

Entrené el modelo de YOLO con este dataset revisado y el modelo no supo detectar bien el balón debido a la poca cantidad de imágenes donde se pueda ver bien la bola. El árbitro y los jugadores fueron bien detectados.

Se replanteó el objetivo del TFG. Se focalizará en la reidentificación de jugadores cuando salen fuera de plano y en el desarrollo de una aplicación que permita al usuario decidir si cuando se produce un cambio de identificador, mantenerlo o cambiarlo, creando un dataset revisado.

2.2 Semana 17-31 de marzo [3]

A la hora de medir resultados como estas gráficas:

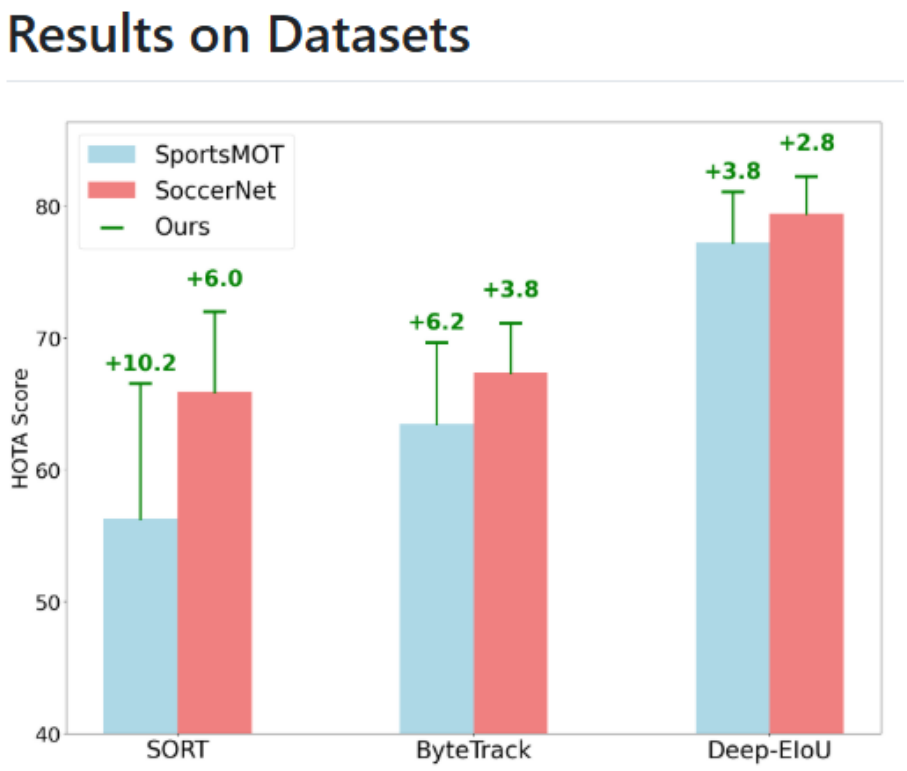

Figure 1: **Métricas gta_link**

Nos centraremos en la medida de los ID's intentando minimizarlos, lo máximo posible, ya que la métrica significa número de ids generados.

Me he puesto a mirar lo que hace exactamente el código del repositorio de gta-link. Y en resumidas cuentas necesitamos un dataset con una pred, ya realizada,

Table 1: Tracking performance on SportsMOT before and after applying our Global Tracklet Association (GTA) method.

Method	HOTA↑	AssA↑	IDF1↑	DetA↑	MOTA↑	IDs↓
SORT [26]	56.28	42.67	58.83	74.30	85.11	5180
SORT + GTA	**66.52** (+10.24)	**59.59** (+16.92)	**77.37** (+18.54)	74.29	**85.27**	**3547** (-1633)
ByteTrack [31]	63.46	51.81	70.76	77.81	94.91	3147
ByteTrack + GTA	**69.74** (+6.28)	**62.61** (+10.80)	**83.16** (+12.40)	77.72	**95.01**	**2107** (−1040)
Deep-EIoU [19]	77.21	67.63	79.81	88.22	96.30	2909
Deep-EIoU + GTA	**81.04** (+3.83)	**74.51** (+6.88)	**86.51** (+6.70)	88.21	**96.32**	**2737** (−172)

Figure 2: **Objetivo**

como es el caso de SoccerNet. Podemos descargar el dataset con el tracking realizado en los archivos .txt que se encuentran dentro de cada clip.

He utilizado las siguientes líneas de comando para realizar la descarga del dataset.

```
from SoccerNet.Downloader import SoccerNetDownloader
mySoccerNetDownloader = SoccerNetDownloader(LocalDirectory="path/to/SoccerNet")
mySoccerNetDownloader.downloadDataTask(task="tracking",
 split=["train","test","challenge"])
```

Posteriormente hice unos ajustes en el código para que cogiera los archivos `gt.txt` para que se tomara como referencia –**pred_dir tracking results directory**, ya que estos son archivos **MOT**, que guardan información sobre los **bounding box** de cada objeto de cada frame, conteniendo información sobre **frame**, **id**, **bb_left**, **bb_top**, **bb_width**, **bb_height**, **conf**, **x**, **y**, **z**, en este orden.

Estos son esenciales para que se pueda ejecutar el archivo `generate_tracklets.py` para generar los **tracklets**.

Un **tracklet** se genera al seguir a un objeto desde su detección en un fotograma hasta el siguiente. En este proceso, el código asocia las detecciones de objetos de cada fotograma con un identificador único (**ID**) para cada objeto, y los agrupa en una **"trayectoria"** que sigue ese objeto a lo largo del tiempo.

En el código:

- **Extracción de características:** El código utiliza un modelo de reidentificación de personas (`FeatureExtractor`) para extraer características visuales de cada objeto detectado en los fotogramas. Esto es útil para seguir objetos entre fotogramas, incluso cuando se producen cambios de apariencia debido a variaciones en la vista o el movimiento.

- **Cálculo de tracklets:** Para cada fotograma, el código agrupa las detecciones de objetos utilizando el identificador único (`track_id`). Si un objeto se detecta en un fotograma y luego aparece en otro, se agrega a un tracklet, que es una colección de todas las detecciones del mismo objeto a lo largo de varios fotogramas. Además, se guarda información como el puntaje de la detección y las características extraídas del modelo de reidentificación.

- **Salvado de tracklets:** Al final de cada secuencia de video o serie de fotogramas, los tracklets generados se guardan en un archivo `pickle` para su posterior uso, permitiendo analizar y trabajar con las trayectorias de los objetos en el futuro.

El programa `generate_tracklets.py`, lo que hará será generar unos ficheros `.pkl` que guardan los datos de las trayectorias de los objetos detectados a lo largo del tiempo. Esto incluye, por ejemplo, las **detecciones de objetos en cada fotograma**, las **características extraídas por el modelo de reidentificación**, y los **identificadores únicos (ID)** asignados a cada objeto (en formato binario).

Una vez que se tenga esos ficheros se realiza una **refinación de los tracklets** para evitar que se cambien los **ids con frecuencia**. Mejora los **tracklets** (trayectorias de objetos) generados por un **tracker en tareas de seguimiento de múltiples objetos (MOT)**. Utiliza dos componentes principales: el **Tracklet Splitter**, que divide **tracklets impuros** (con múltiples identidades) en subtracklets más precisos mediante **clustering (DBSCAN)**, y el **Tracklet Connector**, que fusiona **tracklets fragmentados** que pertenecen al mismo objeto basándose en **similitudes visuales y restricciones espaciales**. Los resultados refinados se guardan en archivos `.txt` en formato **MOT**, listos para su evaluación, visualización o análisis posterior. Este proceso optimiza la **precisión del seguimiento**, corrigiendo errores como **cambios de identidad** y **fragmentaciones**.

Ahora que tenemos un dataset etiquetado, tendremos que:

- **Encontrar una combinación de hiperparámetros óptima:** Con el objetivo de que el cambio de ids sea el mínimo posible.
- **Desarrollar una aplicación:** Con el objetivo de que avise al usuario sobre los cambios de id en los frames y confirme si está bien cambiado o debería conservarse el anterior.

2.3 Semana 31-6 de abril)

En estas semanas hemos estado invirtiendo tiempo en las siguientes cosas. En un principio lo que teníamos era un dataset preparado con los ficheros ground truth fruto de un tracker usado (como DeepEIoU). Pero dado un vídeo, no podíamos aplicar los programas de gta-link, por lo que teníamos que conseguir los ficheros MOT de alguna manera.

Había dos opciones. La primera aplicar un tracker y la segunda aplicar un modelo de YOLO.

Para la primera opción busqué en varios repositorios, que trataran con trackers, principalmente DeepEIoU, que es el tracker con el mejor rendimiento según las estadísticas de paperscode [2].

Rank	Model	HOTA↑	IDF1	AssA	MOTA	DetA	Extra Training Data	Paper	Code	Result	Year	Tags
1	**DeepEIoU + GTA**	81.0	86.5	74.5	96.3	88.2	✓	GTA: Global Tracklet Association for Multi-Object Tracking in Sports			2024	
2	**Deep HM-SORT**	80.1	85.2	72.7	96.6	88.3	✓	Deep HM-SORT: Enhancing Multi-Object Tracking in Sports with Deep Features, Harmonic Mean, and Expansion IOU			2024	
3	**AED**	79.1	81.8	70.1	97.1	89.4	✓	Associate Everything Detected: Facilitating Tracking-by-Detection to the Unknown			2024	

Figure 3: **Métricas DeepEIoU**

Tras encontrar un repositorio que trataba en específico con este [1]. Tuve que modificar código para que utilizara cpu, ya que estaba configurado para gpu de nvidia. Los paquetes no se me instalaban correctamente. Los inputs que utilizaba para los programas eran ficheros tipo .npy.

Dado que me estaba dando muchos problemas, después de dos días decidí que la mejor solución era utilizar un modelo de YOLO. Empecé utilizando un modelo de YOLO estandar capaz de detectar personas. El inconveniente de esto es que el video, estaba grabado desde un ángulo muy bajo, por lo que el modelo detectara las personas de las gradas y esto, a la larga, es un problema.

Entonces para resolver esto había dos opciones. La primera era realizar una segmentación del campo y la segunda entrenar un modelo de YOLO específico para jugadores. La segmentación del campo es demasiado costosa, porque no hay nada automatizado y sería segmentar infinidad de frames. Además de que es específica para un partido de fútbol en un ángulo en específico.Por lo tanto, la mejor opción para mi gusto es entrenar un modelo de YOLO con un dataset, por ahora, específico del partido de balonmano que me proporcionó mi tutor Jose Manuel Jerez. El dataset lo hice creando un programa utilizando las librerias de opencv (cv2) que extraía frames cada 20 segundos que transcurría de vídeo. Los 20 segundos previenen levemente más overfitting del que ocurre por tener frames del mismo partido.

Esta opción me permite etiquetar poco, ya que Roboflow tiene una herramienta de etiquetado automático una vez se han etiquetado unas cuantas imágenes, por lo que simplemente tuve que corregir aquellas imágenes con un mal etiquetado.

Tras tener el dataset etiquetado le añadí algunas augmentations.

A la hora de entrenar el modelo lo hice con 25 epochs. Tardó hora y media en entrenarse.

Apliqué los programas del repositorio de gta-link y los resultados del trackeo son muy buenos, pero la ReID no tiene nada que ver con los resultados tan consistentes del dataset de SoccerNet debido a que la grabación no es profesional.

Aun así tras 23 intentos observe que los parámetros que daban mejores resultados eran los siguientes: –use_split –min_len 100 –eps 0.8 –min_samples 10 –max_k 4 –use_connect –spatial_factor 1.0 –merge_dist_thres 0.7

Recientemente le he añadido frames del partido de fútbol 7 al dataset de roboflow. Y he creado un nuevo modelo de YOLO entrenado exclusivamente con ese dataset.

Los objetivos para la próxima semana:

- **Buscar como aplicar un tracker:** Después de aplicar un trackeo con YOLO, sería necesario aplicar DeepEIoU. Investigar si aplicar un doble gta-link puede funcionar para refinar los tracklets todavía más.

- **Seguir desarrollando la aplicación:** Con el objetivo de que avise al usuario sobre los cambios de id en los frames y confirme si está bien cambiado o debería conservarse el anterior. Ya tengo un programa prueba_revision_ReID.py, que funciona como prototipo, pero falta desarrollarlo mucho.

2.4 Semana 7-13 de abril)

Al final el proceso de usar el tracker es obligatorio, ya que los objetos son detectados por YOLO, pero luego los ids son colocados por el tracker, y de ahí saldrán ficheros tipo MOT. Entonces tras aplicar YOLO para tener ficheros con la info de

los bounding box sobre los jugadores intentaremos aplicarle un tracker y de ahí un modelo de ReID, para posteriormente utilizar gta.

Secuencia de trabajo: YOLO + DeepEIoU + GTA

References

[1] Hsiang-Wei Huang et al. "Iterative Scale-Up ExpansionIoU and Deep Features Association for Multi-Object Tracking in Sports". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2024, pp. 163–172.

[2] Papers With Code. *Multiple Object Tracking on SportsMOT*. Accessed: 2025-04-06. 2025. URL: `https://paperswithcode.com/sota/multiple-object-tracking-on-sportsmot`.

[3] Jiacheng Sun et al. "GTA: Global Tracklet Association for Multi-Object Tracking in Sports". In: *Proceedings of the Asian Conference on Computer Vision*. Springer, 2024, pp. 421–434.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA