TPre - Predicción de trayectoria de peatones con cámara frontal monocular

Alicia Jiajun Lorenzo Lourido
Grado de Robótica
Universidad de Santiago de Compostela
Lugo, España

Juan Daniel Suárez González
Grado de Robótica
Universidad de Santiago de Compostela
Lugo, España

***Resumen*—En este documento se plantea la posibilidad de predecir eficazmente y en tiempo real la trayectoria de múltiples peatones en un entorno urbano observado desde una cámara monocular instalada en un vehículo, identificando a cada peatón individualmente a lo largo del tiempo y estimando su posición en el presente y el futuro.**

https://github.com/ajlorenzo1315

1. Introducción

En el contexto de la conducción automática existe una gran preocupación por prevenir accidentes en los que personas puedan resultar heridas por una decisión errónea de un programa. Para ello no solamente es necesario conocer la posición de las personas en el entorno del vehículo en tiempo real, sino que se deben calcular las posibles posiciones futuras de dichas personas para evitar imprevistos.

1.1. Motivaciones

Nuestra motivación para plantear este trabajo es mejorar las condiciones de seguridad en la conducción autónoma con la menor cantidad de recursos físicos posible, lo que equivale a emplear una única cámara monocular.

Además, combinamos diferentes arquitecturas de red y algoritmos para obtener la salida deseada en vez de recurrir a la segmentación panóptica con modelos de varios cientos de millones de parámetros, que no eran admisibles para nuestro hardware. De este modo se minimizan costes computacionales y costes físicos, tanto en la obtención de información como en su posterior procesado.

2. Trabajos relacionados

En este campo, el estado del arte actual consiste en la utilización de arquitecturas complejas de segmentación panóptica del entorno que generan una imagen completa en vista cenital segmentada a partir de una serie de entradas de vistas frontales desde distintos ángulos [1] o con una única entrada frontal [2] [3].

3. Enfoque teórico

El problema principal consiste en que, dada una entrada compuesta por 8 fotogramas de dimensiones M x N, se debe obtener una salida de 12 listas de coordenadas correspondientes a las posiciones absolutas futuras de todas las instancias de peatones captadas. Para ello dividimos el problema en una serie de tareas que resolvemos individualmente para obtener el resultado final.

Añadimos que la cámara debe estar calibrada para evitar deformaciones en la imagen y errores adicionales en la predicción.

3.1. Procesado de cada fotograma

En las fases de detección y estimación de la profundidad llevamos a cabo una conversión de las coordenadas imagen a coordenadas reales posterior a la obtención de los valores X e Y relativos a la imagen original. El valor de la coordenada X se corresponde con el punto medio de cada instancia en vista frontal, y el valor de la coordenada Y se corresponde con el valor de distancia a la cámara calculado mediante estimación, acotado entre 0 y 255.

Las instancias obtenidas son procesadas con un formato legible para la predicción de trayectoria, convirtiéndose en listas de 4 parámetros en forma:

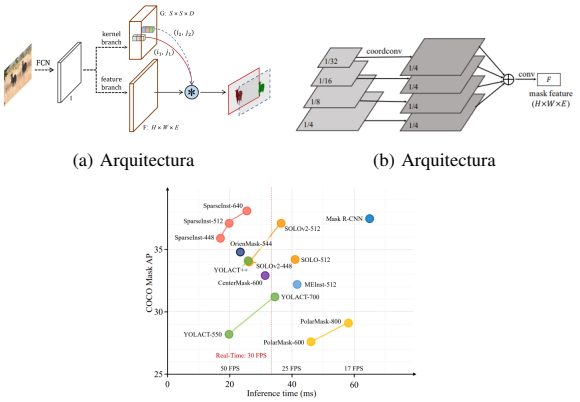
$$[n_{fotograma}, n_{ID}, X, Y]$$

Que serán agrupadas posteriormente en bloques de 8 números de fotograma, por lo que su tamaño será indefinido y estará correlacionado con el número de instancias por fotograma. De este modo se obtendrá un conjunto de puntos con los que predecir hasta 12 fotogramas de trayectorias futuras.

4. Metodología

4.1. Detección y segmentación de instancias - SOLOv2

El objetivo principal de esta parte es detectar a cada peatón observable en cada fotograma. Para ello empleamos



(a) Arquitectura (b) Arquitectura

(c) Velocidad vs precision

Figura 1: (a) I es la característica de entrada después de la extracción de la representación de la FCN-backbone. Las flechas discontinuas indican convoluciones. $k = i * S + j$; y '*' denota la operación de convolución dinámica. (b) Cada nivel de FPN (izquierda) se muestrea con convoluciones y bilineal upsampling hasta llegar a 1/4 de escala (medio). En el nivel FPN más profundo, concatenamos las coordenadas (x, y) y las características originales para codificar la información espacial. Después de la suma de elementos, una convolución de 1 × 1 se emplea para transformar a la función de máscara de salida designada $F \in R^{Al \times La \times An}$. (c) Las velocidades de inferencia son medidas en una NVIDIA 2080Ti.

la arquitectura SOLOv2 [4], que no solamente proporciona las coordenadas de las cajas asociadas a cada instancia, sino que además ofrece una segmentación para producir las máscaras correspondientes. Aunque se sacrifica velocidad de procesamiento, se garantizan resultados más precisos a la hora de estimar posteriormente la distancia a cada peatón, lo cual ha demostrado ser imprescindible en nuestro enfoque.

La arquitectura de la red consta de una capa completamente convolucional seguida de una bifurcación en un módulo de núcleos de máscaras y otra capa de características. El módulo de núcleos predice los pesos asociados a las posiciones de las máscaras, y la capa de características permite diferenciar entre las distintas instancias. Finalmente se lleva a cabo una supresión de no máximos matricial para eliminar máscaras y regiones redundantes y obtener una salida depurada deseada.

4.1.1. Entrenamiento.

La red se entrena con la siguiente función de pérdida:

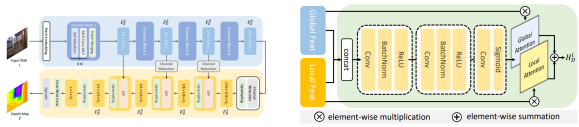
$$L = L_{cate} + \lambda L_{mask}$$

En la que L_{cate} es la pérdida focal convencional para clasificación semántica, y L_{mask} es la pérdida en las máscaras obtenida mediante el coeficiente de Sørensen-Dice, cuya fórmula es:

$$s = \frac{2|X \cap Y|}{|X| + |Y|}$$

Es decir, el doble de la intersección de 2 máscaras partido por la suma de ambas.

4.2. Estimación de la profundidad - GLPDepth



(a) Arquitectura (b) Arquitectura SFF

Figura 2: (a) Arquitectura general de la red propuesta. Los componentes principales de la arquitectura son el codificador, el decodificador y las conexiones de salto con módulos de fusión de características. (b) Descripción detallada del módulo SFF.

Para obtener la distancia a la cámara de cada instancia es necesario poder estimar su profundidad en la imagen. Para ello recurrimos a la arquitectura GLPDepth [5] construida sobre el principio de características globales y locales para generar un mapa de profundidad relativo a cada fotograma. Para extraer estas características se emplea una estructura codificador-decodificador ligera, en la que el codificador genera mapas de características en 4 escalas diferentes $(\frac{1}{4}, \frac{1}{8}, \frac{1}{16} y \frac{1}{32})$ y el decodificador restaura la dimensión de la imagen de entrada para obtener el mapa de profundidad. Además se agrega un módulo de fusión selectiva de características que asocia características locales y globales mediante mapas de atención, multiplicando los valores de ambas para dar mayor relevancia a su ubicación en la imagen.

4.2.1. Entrenamiento.

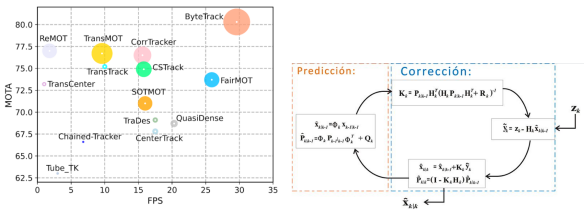
El modelo ha sido entrenado empleando los datasets NYU Depth V2 y KITTI, empleando una métrica de error logarítmica e invariante en la escala para determinar la distancia entre el mapa de profundidad predicho y el valor real de las profundidades en la imagen original, siguiendo la fórmula:

$$L = \frac{1}{n}\sum_i d_i^2 - \frac{1}{2n^2}(\sum_i d_i^2)$$

En la que $d_i = logy_i - logy_i^*$.

Además se emplea una técnica de aumento de datos conocida como *Vertical CutDepth* en la que se introducen imágenes parcialmente completadas con información del mapa de profundidad, respetando siempre la geometría vertical de la imagen. Esto implica que no se eliminan secciones verticales de la imagen para garantizar una mejor predicción, apoyándose en estudios realizados que observan que la estimación de la profundidad se realiza en una gran medida por medio de la posición vertical del objeto en la imagen, y no por medio de la textura o tamaño de este [6]

4.3. Seguimiento - BYTE



(a) MOT (b) filtro de Kalman

Figura 3: MOTA-IDF1-FPS comparaciones de diferentes rastreadores en el dataset de MOT17. El eje horizontal es FPS (velocidad de carrera), el eje vertical es MOTA y el radio del círculo es IDF1. Nuestro ByteTrack logra 80.3 MOTA, 77.3 IDF1 en el conjunto de prueba MOT17 con una velocidad de carrera de 30 FPS, superando a todos los rastreadores anteriores.

Para llevar a cabo el seguimiento de instancias nuevamente optamos por el Estado del Arte en este ámbito: el algoritmo BYTE [7]. Su funcionamiento consiste en la división de las instancias en 2 grupos según un umbral de fiabilidad y la aplicación de un filtro de Kalman sobre cada fotograma. Posteriormente se seleccionan 2 técnicas de asociación por similitud para emparejar las instancias. En primer lugar las instancias con mayores índices con las trayectorias generadas con el filtro de Kalman empleando la primera técnica, y posteriormente se emparejan las instancias con menor fiabilidad con las trayectorias no asociadas anteriormente mediante el segundo método. Aquellas instancias no asociadas se emplean para generar nuevas trayectorias en el futuro si vuelven a aparecer en un número determinado de frames.

Como primera técnica de asociación se emplean las características Re-ID, obtenidas a través del modelo FastReID [8], y como segunda técnica se emplea la medida IoU (*Intersection over Union*), que compara el porcentaje de coincidencia entre las posiciones de 2 instancias.

4.4. Mapeo

Para obtener las posiciones de las personas utilizaremos primero el sistema de coordenadas pixel puesto que nos facilita la obtección.

Para la obtención del valor en el eje X utilizamos la homografía para para saber su posición proyectada y esta a la vez se ponderará por el valor de distancia por pixel. Por otro lado para la obtención de la posición del eje Y utlizamos la media de la profundidad de cada persona. Para obtener este valor utilizamos la unión entre las mascaras y la prediccion del mapa de profundidad.

4.5. Predicción de trayectorias - ExpertTraj

Para predecir las futuras trayectorias de todas las instancias de las personas visibles empleamos el modelo ExpertTraj [9]. Los principios de este modelo son que, dada una serie de trayectorias iniciales, se calculan posibles objetivos de dichas trayectorias mediante búsqueda por similitud con un repositorio de trayectorias proporcionado previamente. Con estos objetivos se lleva a cabo una codificación de la información, que es suministrada a un módulo de atención social encargado de considerar las trayectorias del resto de peatones a la hora de producir los resultados. Una vez calculadas las trayectorias, se decodifica la información y se escoge entre todas las trayectorias calculadas una muestra de n trayectorias, que será la predicción final.

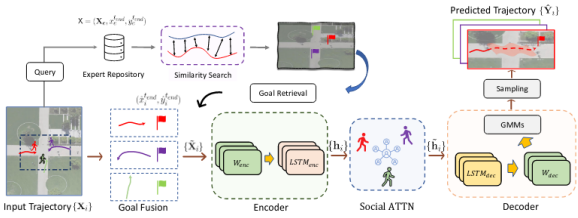


Figura 4: Arquitectura de red

Para predecir las futuras trayectorias de todas las instancias de las personas visibles empleamos el modelo ExpertTraj []. Los principios de este modelo son que, dada una serie de trayectorias iniciales, se calculan posibles objetivos de dichas trayectorias mediante búsqueda por similitud con un repositorio de trayectorias proporcionado previamente. Con estos objetivos se lleva a cabo una cod

Los resultados de este enfoque son sobresalientes y destaca como el estado del arte para predicción de trayectorias empleando un formato simple sin mapeo completo del entorno.

4.5.1. Entrenamiento.

El entrenamiento del modelo se lleva a cabo mediante los datasets SDD (*Stanford Drone Dataset*) y ETH/UCY, que contienen instancias en el formato:

$$[Frame_{ID}, ID_{Peatón} Ped_X Ped_y]$$

Estas instancias son procesadas en bloques de 8 fotogramas consecutivos para la trayectoria observable y los 12 fotogramas consecutivos siguientes, que se corresponden con la trayectoria real que se debe predecir. Se lleva a cabo un entrenamiento con el optimizador Adam, y las métricas de error empleadas para validar los modelos son:

- ADE (*Average Distance Error*): mide el promedio de error de distancia en cada punto de la predicción.
- FDE (*Final Distance Error*): mide el error de distancia en el destino de la predicción.

4.6. Nuestro modelo - TPre

Como ya se ha mencionado el modelo está conformado por diferentes sub-modelos que se encargan de las tareas más sencillas, siguiendo una estrategia de “divide y vencerás”.

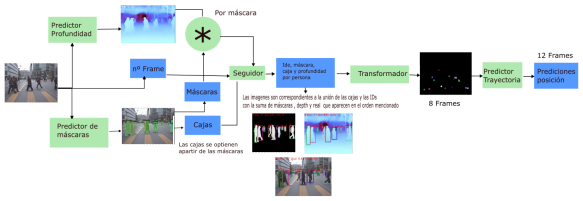


Figura 5: Arquitectura de TPre

4.6.1. Información.

La información entra al modelo en forma de fotogramas. Esta es introducida simultáneamente en la red de segmentación de instancias y en la red de estimación de profundidad, que proporcionan el mapa de profundidad equivalente al fotograma y las coordenadas y dimensiones en la imagen de las máscaras de instancias. Estos valores son suministrados al algoritmo de seguimiento para obtener las IDs vinculadas a cada máscara y un historial de profundidades para cada ID.

4.6.2. Superposición de instancias.

Uno de los principales obstáculos que nuestro modelo es capaz de superar es el de las instancias superpuestas. El seguidor de instancias se encarga de conservar las cajas e IDs de los peatones aunque estén totalmente ocultos, aunque hay otro principal inconveniente: estimar la distancia a la cámara de los peatones si no son visibles. Para ello recurrimos al historial de profundidades asociado a su ID.

Se calcula la distancia de la media de las profundidades guardadas (los últimos valores observables) a la profundidad obtenida, y si esta supera un umbral se descarta y se guarda como profundidad la media de las profundidades anteriores. De este modo se evita que se pueda malinterpretar la información o enviar saltos de posición bruscos al predictor de trayectorias.

4.6.3. Transformación de coordenadas.

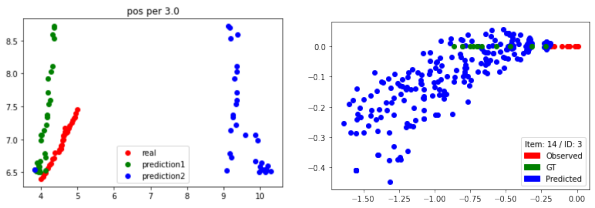
Para convertir las coordenadas a coordenadas reales empleamos algoritmos de regresión, ya que es necesario garantizar la velocidad del sistema para que sea eficiente en tiempo real. El objetivo es que, dadas las coordenadas reales de peatones en una secuencia de fotogramas anotados manualmente, y la misma secuencia de coordenadas estimadas a partir de los fotogramas, obtener una función que relacione ambos sistemas y que permita transformar las coordenadas entre ambos.

4.6.4. Entrenamiento.

El entrenamiento del modelo se lleva a cabo por partes mediante los datasets antes mencionados. Esto se debe a la dimensión total del mismo y a nuestras limitaciones computacionales.

Una de las grandes ventajas de este enfoque modular es que permite emplear una gran diversidad de datasets a la hora de minimizar errores. Además, en caso de surgir una nueva técnica que supere el estado del arte en cualquiera de las tareas descritas, reemplazarla en el modelo sería sencillo y no sería necesario entrenarlo de nuevo completamente.

5. Experimentos



(a) Posiciones (b) Predicción de la trayectoria

Figura 6: (a) Predicción de las posiciones con respecto a la real (b) Predicción de la trayectoria

Llevamos a cabo pruebas con secuencias de vídeo grabadas y anotadas manualmente para las trayectorias de las personas visibles, realizadas en una zona cuadriculada para tener conocimento de las dimensiones del espacio. Para validar la precisión de las medidas tomadas por el modelo empleamos la medida de error cuadrático medio (MSE) sobre los puntos empleando las anotaciones como medidas reales. De este modo podremos saber si el resultado de estimar las posiciones en tiempo real de nuestro modelo es acertado y puede emplearse para la predicción.

La importancia de esta métrica de error es considerable ya que el ruido en las estimaciones puede afectar a las predicciones futuras enormemente.

Persona	MSE depth	RMSE depth	MSE posx	RMSE pos x
Per 1	0.006	0.081	0.10	0.33
Per 2	0.37	0.61	0.83	0.91
Per 3	0.30	0.54	0.23	0.48
Per 4	0.10	0.32	0.14	0.38

Cuadro 1: MSE Y RMSE

Los valores obtenidos en la predicción para estos conjuntos de datos son:

ADE : 1,3153547521537017
FDE : 4,664200045608136

6. Conclusiones

Se ha propuesto un modelo modular capaz de estimar las coordenadas en tiempo real de peatones en el plano cenital, y de predecir sus trayectorias futuras. Sin embargo,

tras una exhaustiva comparación de modelos y pruebas realizadas no se ha alcanzado el estándar de 15 fotogramas por segundo correspondiente a ejecuciones en tiempo real, al menos con nuestro hardware (*NVidia RTX 3070 Laptop*).

Tambien hay que destacar que la superposición total de los peatones es un problema que no se puede solventar con la tecnología actual. Además cualquier tipo de superposición afecta a la predicción de profundidad. Por otro lado el que las prediciones de profundidad estén acotadas al valor de mapa de profundidades entre 0 y 255 impide una precisión total puesto que este valor se comporta de manera distinta segun nos alejamos de la cámara , lo que obliga a escoger entre una precisión cercana o una lejana.

Otro problema fundamental del proyecto es la incapacidad de realizar predicciones significativas en momentos futuros lejanos, ya que las predicciones están limitadas a 12 fotogramas.

Para resumir, aunque la propuesta resulta interesante, tiene varios problemas a la hora de conseguir una predicción precisa.

7. Trabajos futuros

Plantear una red neuronal que pueda obtener los valores reales de la profundidad y no en un mapa de calor. Además, sería práctico contar con una arquitectura capaz de predecir la proyección de la coordenada X para poder obtener su verdadero valor.

Referencias

[1] Gosala, N. & Valada, A. Bird's-Eye-View Panoptic Segmentation Using Monocular Frontal View Images. *IEEE Robotics And Automation Letters*. **7**, 1968-1975 (2022)

[2] Pan, B., Sun, J., Leung, H., Andonian, A. & Zhou, B. Cross-View Semantic Segmentation for Sensing Surroundings. *IEEE Robotics And Automation Letters*. **5**, 4867-4873 (2020)

[3] Mohan, R. & Valada, A. Efficientps: Efficient panoptic segmentation. *International Journal Of Computer Vision (IJCV)*. (2021)

[4] Wang, X., Zhang, R., Kong, T., Li, L. & Shen, C. SOLOv2: Dynamic and Fast Instance Segmentation. *Proc. Advances In Neural Information Processing Systems (NeurIPS)*. (2020)

[5] Kim, D., Ga, W., Ahn, P., Joo, D., Chun, S. & Kim, J. Global-Local Path Networks for Monocular Depth Estimation with Vertical CutDepth. *ArXiv Preprint ArXiv:2201.07436*. (2022)

[6] Van DIjk, T. & Croon, G. How do neural networks see depth in single images?. *IEEE Xplore*. (2020,2), https://ieeexplore.ieee.org/document/9009532/

[7] Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W. & Wang, X. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. *Proceedings Of The European Conference On Computer Vision (ECCV)*. (2022)

[8] He, L., Liao, X., Liu, W., Liu, X., Cheng, P. & Mei, T. FastReID: A Pytorch Toolbox for General Instance Re-identification. *ArXiv Preprint ArXiv:2006.02631*. (2020)

[9] He, Z. & Wildes, R. Where are you heading? Dynamic Trajectory Prediction with Expert Goal Examples. *Proceedings Of The International Conference On Computer Vision (ICCV)*. (2021)

[10] Mohamed, A., Zhu, D., Vu, W., Elhoseiny, M. & Claudel, C. Social-implicit: Rethinking trajectory prediction evaluation and the effectiveness of implicit maximum likelihood estimation. *ArXiv.org*. (2022,3), https://arxiv.org/abs/2203.03057

Apéndice A. Tablas y validaciones

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
box-based:							
Mask R-CNN [4]	Res-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN*	Res-101-FPN	37.8	59.8	40.7	**20.5**	40.4	49.3
MaskLab+ [30]	Res-101-C4	37.3	59.8	39.6	16.9	39.9	53.5
TensorMask [7]	Res-101-FPN	37.1	59.3	39.4	17.4	39.1	51.6
YOLACT [2]	Res-101-FPN	31.2	50.6	32.8	12.1	33.3	47.1
MEInst [9]	Res-101-FPN	33.9	56.2	35.4	19.8	36.1	42.3
CenterMask [31]	Hourglass-104	34.5	56.1	36.3	16.3	37.4	48.4
BlendMask [8]	Res-101-FPN	38.4	60.7	41.3	18.2	41.5	53.3
box-free:							
PolarMask [10]	Res-101-FPN	32.1	53.7	33.1	14.7	33.8	45.3
SOLO [1]	Res-101-FPN	37.8	59.5	40.4	16.4	40.6	54.2
SOLOv2	Res-50-FPN	38.8	59.9	41.7	16.5	41.7	56.2
SOLOv2	Res-101-FPN	39.7	60.7	42.9	17.3	42.9	57.4
SOLOv2	Res-DCN-101-FPN	**41.7**	**63.2**	**45.1**	18.0	**45.0**	**61.6**

Figura 7: Comparación de rendimiento para el modelo SOLOv2

Method	Params (M)	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	AbsRel ↓	RMSE ↓	$log_{10} \downarrow$
Eigen [Eigen *et al.*, 2014]	141	0.769	0.950	0.988	0.158	0.641	-
Fu [Fu *et al.*, 2018]	110	0.828	0.965	0.992	0.115	0.509	0.051
Yin [Yin *et al.*, 2019]	114	0.875	0.976	0.994	0.108	0.416	0.048
DAV [Huynh *et al.*, 2020]	25	0.882	0.980	0.996	0.108	0.412	-
BTS [Lee *et al.*, 2019]	47	0.885	0.978	0.994	0.110	0.392	0.047
Adabins[Bhat *et al.*, 2021]	78	0.903	0.984	0.997	0.103	0.364	0.044
DPT* [Ranftl *et al.*, 2021]	123	0.904	**0.988**	**0.998**	0.110	0.357	0.045
Ours	62	**0.915**	**0.988**	0.997	**0.098**	**0.344**	**0.042**

Figura 8: Comparación de rendimiento para el modelo GLPDepth

Tracker	MOTA↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDs↓
DeepSORT [70]	27.1	28.6	8.5%	41.5%	5894	42668	2220
MOTDT [12]	26.1	32.9	8.7%	54.6%	6318	43577	1599
IOUtracker [7]	38.6	38.6	28.3%	27.6%	9640	28993	4153
JDE [69]	33.1	36.0	15.1%	24.1%	9526	33327	3747
FairMOT [85]	35.0	46.7	16.3%	44.2%	6523	37750	**995**
CenterTrack [89]	40.9	45.1	10.8%	32.2%	3208	36414	1568
ByteTrack (Ours)	**61.7**	**63.1**	**38.3%**	**21.6%**	**2822**	**22852**	1031

Figura 9: Comparación de rendimiento para el seguimiento de instancias

	ETH	Hotel	Univ	Zara1	Zara2	Average	Average AMD/AMV
	ADE/ FDE AMD/ AMV KDE						
S-GAN [8]	0.81/1.52 3.94/0.373 5.02	0.72/1.61 2.59/0.384 3.45	0.60/1.26 2.37/0.440 2.03	0.34/0.69 1.79/0.355 0.68	0.42/0.84 1.66/0.254 -0.03	0.58/1.18 2.47/0.361 2.23	1.42
S-STGCNN [25]	0.64/1.11 3.73/0.094 6.83	0.49/0.85 1.67/0.297 1.56	0.44/0.79 3.31/0.060 5.65	0.34/0.53 1.65/0.149 1.40	0.30/0.48 1.57/0.103 1.17	0.44/0.75 2.39/0.104 3.32	1.26
Trajectron++ [32]	0.39/0.83 3.04/0.286 1.34	0.12/0.21 1.98/0.114 -1.89	0.20/0.44 1.73/0.228 -1.08	0.15/0.33 1.21/0.171 -1.38	0.11/0.25 1.23/0.116 -2.43	0.19/0.41 1.84/0.183 -1.09	1.01
ExpertTraj [39]	0.30/0.62 61.77/0.034 NaN	0.09/0.15 21.30/0.003 NaN	0.19/0.44 M/M M	0.15/0.31 32.14/0.005 NaN	0.12/0.24 M/M M	0.17/0.35 38.4/0.004 NaN	19.20
Social-Implicit(ours)	0.66/1.44 3.05/0.127 5.08	0.20/0.36 0.58/0.410 0.59	0.31/0.60 1.65/0.148 1.67	0.25/0.50 1.72/0.078 1.24	0.22/0.43 1.16/0.106 0.69	0.33/0.67 1.63/0.174 1.85	**0.90**

Figura 10: Comparación de rendimiento para la predicción de trayectorias