УНИВЕРЗИТЕТ У БЕОГРАДУ
МАТЕМАТИЧКИ ФАКУЛТЕТ

Момир Аџемовић

ПРЕДИКЦИЈА ТРАЈЕКТОРИЈА ВИШЕ ОБЈЕКАТА НА СЦЕНИ

мастер рад

Београд, 2022.

Ментор:

др Младен Николић, ванредни професор
Универзитет у Београду, Математички факултет

Чланови комисије:

др Јована Ковачевић, доцент
Универзитет у Београду, Математички факултет

др Александар Картељ, доцент
Универзитет у Београду, Математички факултет

Датум одбране: 15. септембар 2022.

Посвета... у изради...

Наслов мастер рада: Предикција трајекторија више објеката на сцени

Резиме: У изради...

Кључне речи: машинско учење, аутономна вожња, растеризација, графовске неуронске мреже

Садржај

Глава 1

Увод

Аутономна вожња подразумева потпуну или парцијалну аутоматизацију процеса контроле возила коришћењем рачунарских и осталих технологија. Агент (возило) у овом случају мора да има способност опажања окружења и самосталног кретања кроз то окружење ради остваривања задатог циља (краткорочно: паркирање, скретање, ... дугорочно: транспорт, ...). Кључне компоненте које се интегришу у овај систем су:

- Детекција и праћење објеката у околини (опажање окружења са акцентом на покретне објекте)
- Разумевање њихових циљева, како би се сам циљ агента ускладио са њима (први корак за омогућавање самосталног кретања кроз окружење).

Прва компонента се реализује помоћу специјалних сензора као што су *LiDAR* сензори за конструкцију 3D мапе окружења, *Radar* сензора за детекцију удаљености и брзине објеката (погодан и у случају лошег времена) и камере који прикупљају 2D слике окружења. Саме камере помоћу метода машинског учења могу да извршавају исте послове као и радари (као што то човек ради користећи само вид). Циљ овог рада се односи на други део тј. анализа постојећих радова чија је тама разумевање циљева објеката на сцене где се агент налази. Будуће трајекторије тих објеката се посматрају као циљеви, а сама трајекторија агента се усклађује у односу на њих.

1.1 Аутономна вожња

Министарство саобраћаја САД и NHTSA (*National Highway Traffic Safety Administration*) је усвојила *Society of Automotive Engineers (SAE)* стандард који прописује 6 нивоа аутоматизације од нултог, где човек има потпуну контролу до петог, где рачунар самостално управља возилом:[1]

- **Ниво 0**: Човек има потпуну контролу над возилом
  - Возило може да има једноставније аутоматизације као што је на пример аутоматизовано кочење у хитним случајевима.
- **Ниво 1**: Човек и рачунар сарађују у процесу управљања возилом (сва одговорност је на возачу).
- **Ниво 2**: Рачунар има потпуну контролу, али човек мора да буде присутан и да реагује у било ком тренутку ако је то неопходно (сва одговорност је на возачу).
- **Ниво 3**: Рачунар има потпуну контролу ако су испуњени одређени услови, а човек је у том случају потпуно ослобођен свих дужности вожње.
  - Ако услови више не важе (нпр. пада снег), онда човек мора да преузме одговорност над возилом.
  - Када су испуњени услови, одговорност је на произвођачу софтвера за аутономну вожњу.
- **Ниво 4**: Слично као ниво 3, али возило је испрограмирано тако да се само заустави у случају да услови нису испуњени.
  - Када сви услови опет важе, возило-рачунар може да настави да врши свој задатак.
- **Ниво 5**: Рачунар има потпуну контролу над возилом у сваком тренутну.

1.2 Поставка проблема

Предвиђање трајекторија или предвиђање кретања једног агента на сцени са више покретних објеката (суседи аналогни агенту) се формално дефинише као предикција скупа вредности $\{(x_a^t, y_a^t) \mid t \in [0, T]\}$, где је T дужина трајекторије агента која се предвиђа, под претпоставком да су дате следеће информације:

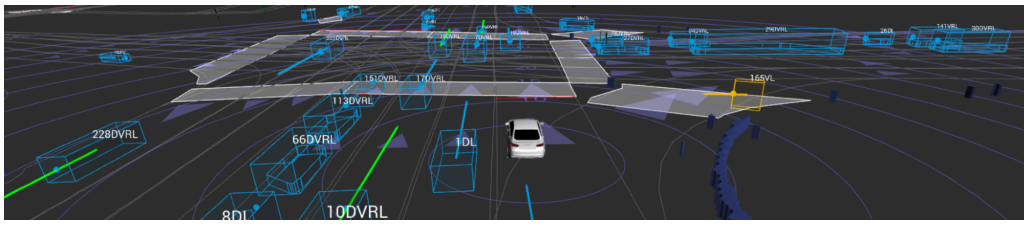
- Историја тог агента $A_h = \{(x_a^t, y_a^t) \mid t \in [-H, -1]\}$, где је H дужина историје трајекторије;
- Историја осталих покретних агената $\{T_{raj}^n \mid n \in N\}$, где је T_{raj}^n истог облика као и A_h, а N је скуп свих суседа;
- Окружење агента E - варира у односу на скуп података (путеви, пешачки прелази, семафори, ...).

Сам проблем је општији од примене у аутономног вожњи (предвиђање кретања пешака), али представља кључну компоненту у системима за аутономну вожњу.

1.3 *HD* мапе

Мапа са високим нивоом детаља (*eng. HD map*) је мапа путева са прецизношћу до пар центиметара и високим нивоом познавања окружења: позиције пешачких прелаза, позиције семафора, полигони путева, смерови улица итд... Битно је нагласити да су *HD* мапе супериорне у односу на *GPS* у смислу прецизности и разноврсности информација. Саме *GPS* мапе нису довољно прецизне да може да их користи систем за аутономну вожњу уместо људи који су значајно робуснији на грешке *GPS*-а (што може да буде и до неколико метара). Аналогија HD мапа са човеком и вожња добро познатим путем и вожња непознатим путем први пут у животу. Вожња добро познатим путем одговара знању које се добија из HD мапа.

Овај тип података даје јако велику количину информација које се могу процесирати у целокупном систему аутономне вожње. Два главна проблема и изазова овог типа података је њихово прикупљање (LiDAR системи су скупи) и одржавање, и обрада овако неструктуираних података. Алтернатива је коришћење камера и модела дубоког учења за прикупљање података о окружењу у реалном времену.



Слика 1.1: *HD* мапа - *Kaggle Lyft* [2]

На слици 1.1 се види пример визуализоване *HD* мапе. Бело возило је агент, а плави правоугаоници су суседи. Са слике се види да мапе садрже и полигоне пешачких прелаза и још неке додатне податке. Подаци ових мапа су често у *3D* формату тј. узима се у обзир и висина. У реалном времену се ови подаци могу комбиновати са улазима из камера.

Глава 2

Преглед основних градивних елемената

2.1 Неуронске мреже

TODO: Osnovno, MLP

2.2 Конволутивне неуронске мреже

TODO: Mozda se i ne budu koriscene

2.3 Графовске неуронске мреже

TODO: Osnovno, opšte, GCN, GraphSage

2.4 Механизам пажње

TODO: attention

Глава 3

Преглед претходних приступа

Техника за предикцију трајекторија више објеката које издвајају посебан објекат као агент, могу да се групишу грубо у четири групе:

- Технике прилагођене репрезентацијама временских серија;
- Технике засноване на растеризацији;
- Технике засноване на графовским репрезентацијама;
- Хибридне технике.

Постоје и општије технике које не разликују конкретно агента од осталих објеката у процесу предвиђања.

3.1 Технике прилагођене репрезентацијама временских серија

Група модела која је интуитивна као иницијални приступ овом проблему који се своди на предвиђање временских серија, јесу рекурентне неуронске мреже (*eng. RNN - Recurrent neural network*) и конволутивне мреже за једну димензију (*CNN - Convolutional Neural Network*). Често коришћена *LSTM* архитектура рекурентних неуронских мрежа погодна за кодирање динамике објеката. Како трајекторија једног објекта зависи од трајекторија осталих објеката, неопходно је да се користи механизам пажње као компонента модела која учи везе између различитих објеката на сцени. [3] [4] Мана овог скупа модела је што модели нису погодни за потпуно разумевање контекста сцене (објекти, возне линије, саобраћајни знакови, ...). Такође, једна од главних проблема оваквих директних приступа је неспособност модела да научи мултимодалну природу трајекторија на сцени.

Уместо учења модела који дају предикције трајекторија које се у просеку најбоље у тим случајевима (проблем мултимодалности), алтернатива су скуп архитектура заснованих на (условљеним) генеративним моделима који омогућавају генерисање произвољног броја трајекторија кроз узорковање из условљене расподеле (расподела је условљена историјом трајекторије, као и окружењем у којем се тај објекат налази). Пример је *Social GAN* [5] архитектура која узима у обзир претходне наведене услове и генерише „социјално прихватљиве“ трајекторије пешака. У случају возила са агентом, генеративни модели захтевају током процеса предикције напредније алгоритме узорковања ради оптимизације покривености трајекторијама[1].

3.2 Технике засноване на растеризацији

Растеризација подразумева трансформацију *HD* мапа у формат слике, где слике приказују сцене из птичје перспективе (*eng. BEV - Bird's-eye-view*). Предност овог формата је могућност учења контекста тих мапа коришћењем више слојева конволутивних неуронских мрежа. Овакве компоненте архитектуре (низ повезаних конволутивних мрежа од улаза) се често називају енкодери[2]. Конволутивне мреже нису ограничене да раде са RGB и сличним форматима слика тј. сваком пикселу може да се додели низ својстава. Нека од очигледнијих својстава су: да ли агент заузима тај пиксел, да ли сусед (неагент) заузима тај пиксел, да ли пиксел припада улици, ... Излаз CNN енкодера се углавном накнадно комбинује са осталим компонентама за генерисање резултата.

Излази модела такође могу да се представљају као слике тј. топлотне мапе *(eng. Heat Map)*, где се сваком пикселу додељује вероватноћа да се агент (или посматрани објекат) налази на тој локацији. Топлотне мапе се добијају декодерима који комплементирају енкодер компоненте. То значи да није неопходно да се експлицитно постави ограничење на одређени број излазних трајекторија модела. Проблем код више излазних трајекторија модела је то што може да изазове колапс моде (eng. *mode collapse*). Број трајекторија се овде не дефинише експлицитно и може да варира од једне сцене до друге. Наравно, не узимају се сви пиксели као кандидати, већ се примењује неки алгоритам узорковања. Алгоритам узорковања у том случају даје флексибилност, где може сам алгоритам да се имплементира тако да је најпогоднији за неку од метрика тј. даје боље резултате за једну метрику у односу на остале. [6] [7]

Архитектура *HOME* [6] користи овај принцим за паралелно кодирање растеризоване сцене и трајекторија објеката. Резултати компоненти се спајају и прослеђују као

[1]Покривеност се односи на метрике које узимају у обзир више од једне трајекторије, па је битно и да те саме трајекторије буду разноврсне

[2]Сам појам енкодера није везан само за појам конволутивних слојева, већ било које слојееве

улаз у декодер за генерисање топлотне мапе. Узорковањем тачака из топлотних мапа се добија скуп кандидата тачака. Главна претпоставка архитектура заснованих на топлотним мапама је: Ако знамо циљну тачку и историју трајекторије објекта, онда можемо једноставно да одредимо трајекторију до те циљне тачке[3]. За одређивање ових трајекторија могу да се користе једноставнији модели неуронских мрежа (нпр. потпуно повезане мреже).

Могуће је растеризовати потпуно податке тј. растеризовати и трајекторије као низ слика. Свака слика садржи скуп тачака које представљају локације објеката. Архитектура *CASPNet* [8] примењује CNN енкодер на свако стање и користи ConvLSTM [9] за разумевање темпоралних веза. На овај начин се извлаче својства динамичког дела сцена. На сличан начин је могуће и извући својства и за статички део (возне линије, возни површине, ...) користећи класичне конволутивне мреже. Комбинацијом ових података се генеришу трајекторије које су исто у растеризованом облику.

3.3 Технике засноване на графовским репрезентацијама

Мапе имају комплексну топологију. Технике засноване на растеризацији користе конволуцију која тешко извлачи потпуно семантику тих мапа, а такође су конволуције скупе операције. Алтернатива је моделовање мапа графовским структурама. Технике засноване на графовском репрезентацијом као улаз добијају стање мапе кодиране као граф и примењују моделе графовских неуронских мрежа. Две архитектуре које представљају основе за већину архитектура ове групе су *LaneGCN* [10] и *VectorNet* [11].

Мапа може да се моделује као скуп повезаних сложених линија (*end. polylines*), где сваком објекту одговара једна усмерена сложена линија. *VectorNet* је хијерархијска графовска неуронска мрежа која као улаз добија мапу која је моделована као скуп сложених линија, а као резултат даје скуп предикција трајекторија. Идеја је да се прво извуку својста из сложених линија појединачних објеката, а онда то пронађу одговарајуће међусобне везе између објеката и везе између објеката и возних линија. За проналазак међусобних веза се користи механизам пажње. [12] [11]

Архитектура *LaneGCN* нуди варијанту конволутивних графовских мрежа (*GCN - Graph Convolution Network*) која је специјализована за графове путних линија које имају различите типове веза. Користе се различите матрице повезаности за суседе, претходнике, следбенике (леви и десни) у контексту путева. За сваку матрицу повезаности може да се примени класичан GCN, а комбиновањем тих елемената је добија један *LaneGCN* слој. [10] Сам модел се своди на *GCN* са више матрица повезаности.

[3]Ова претпоставка је такође основа за многе друге методе

3.4 Хибридне технике

Хибридне технике користе комбинацију структура графова и BEV слика. *GOHOME* Модификована верзија *HOME* која уместо CNN енкодера и растеризованих слика мапа, користи *LaneGCN* архитектуру за енкодер. Заправо ансамбл ова два модела (*HOME, GOOME*) даје најбоље резултате по *MR* метрици[4].

3.5 Технике засноване на облацима тачака

Последња група нешто одступа од осталих али и даље даје добре резултате. Подаци се посматрају као облаци тачака (*eng. point cloud*) и примењују се технике намењене за такву структуру података. Основна архитектура је *TPCN* [13] која је заснована на *PointNet* [14], а већина осталих техника су „изведене“.

[4]Објашњење за ову метрике се налази у секцији за евалуацију

Глава 4

Метрике за евалуацију модела

Неке од стандардних метрика за евалуацију квалитета предикције трајекторија су „просечна грешка одступања“ (*eng. ADE - Average Displacement Error*) и „последња грешка одступања“ (*eng. FDE - Final Displacement Error*). У наставку се користе енглеске скраћенице *ADE* и *FDE*. Метрика *ADE* се добија упросечавањем еуклидског растојања између временски синхронизованих тачака трајекторија предикције и реализације. Метрика *FDE* узима у обзир само последњу тачку тј. еуклидско растојање између последње тачке реализације и предикције. [5] [3] У наставку се налазе формуле у случају да се посматра тачно један објекат (нпр. само агент):

$$ADE = \frac{1}{T} \sum_{k=1}^{T} \sqrt{(x_k - \hat{x}_k)^2 + (y_k - \hat{y}_k)^2}$$

$$FDE = \sqrt{(x_{last} - \hat{x}_{last})^2 + (y_{last} - \hat{y}_{last})^2}$$

Метрике се једноставно уопштавају у случајевима где постоји више објеката на сцени:

$$ADE = \frac{1}{T \times N} \sum_{n=1}^{N} \sum_{k=1}^{T} \sqrt{(x_k^n - \hat{x}_k^n)^2 + (y_k^n - \hat{y}_k^n)^2}$$

$$FDE = \frac{1}{N} \sum_{n=1}^{N} \sqrt{(x_{last}^n - \hat{x}_{last}^n)^2 + (y_{last}^n - \hat{y}_{last}^n)^2}$$

Ако узмемо у обзир претпоставку да се релативно детерминистичкі може da se одреди трајекторија ако је дата њена последња тачка (што је у случају возила разумна претпоставка), онда су ове две метрике корелисане у смислу да модел који има добар *FDE*, има и добар *ADE* (и супротно).

Ове једноставне метрике су погодне када претпостављамо да је расподела трајекторија унимодална тј. да је природа трајекторија претежно детерминистичка. Скупови података трајекторија могу да имају јачу стохастичку природу због стохастичке природе самих објеката (трајекторија) или непотпуних опажања окружења

(непотпуне информације). Пример таквог скупа података је скуп трајекторија пешака. Пешак који је прешао пешачки прелаз, може у том тренутку да скрене лево или десно. У том случају имају два вероватна сценарија за исту историју трајекторије (углавном немамо информације о циљевима самог пешака). [5] [15] У случају возила је ситуација мало блажа због претходне претпоставке, али и даље је тешко одредити крајњу тачку трајекторије која може значајно да варира (да ли возач жели да скрене десно или иде право, да ли вози брже или спорије, ...).

Скуп „најбољи из групе“ (*eng. „Best of Many“*) техника узимају у обзир мултимодалну природу расподела трајекторија. Модел може да генерише неколико различитих предикција трајекторија и за сваку трајекторију да да одговарајућу вероватноћу (поузданост). Као грешка се узима предикција која је најбоља по дефинисаном критеријуму (критеријум не мора да се поклапа са самом мером која се користи). [15] [3] Претходно наведене технике *ADE* и *FDE* се уопштавају у *minADE* и *minFDE*. Због једноставности узимају се у обзир облици са једним објектом: [16] [15]

$$minADE = ADE(\arg\min_{\hat{T}^k_{raj}} FDE(\hat{T}^k_{raj}, T_{raj}), T_{raj}), k \in \{1, ..., K\}$$

$$minFDE = \min FDE(\hat{T}^k_{raj}, T_{raj}), k \in \{1, ..., K\}$$

Уколико модел генерише више од K трајекторија, узима се и обзир првих K по поузданости. У специјалном случају када је $K = 1$, онда *minFDE* постаје *FDE*, а *minADE* се и даље разликује по избору „главне“ трајекторије. Проблем са *minADE* і *minFDE* је у томе што не узимају у обзир остале трајекторије поред најбоље и самим тим се не прави разлика између предикције са свим добрим трајекторијама и предикције са једном добром трајекторијом. [16] Друга замерка овим метрикама је што не узимају у обзир поузданост предикција након филтрирања K трајекторија. Уколико је најбоља трајекторија прецизна, желимо и даље да знамо да ли је модел сигуран или је добар резултат последица „среће“. Модификоване метрике: [6]

$$p\text{–}minADE = ADE(\hat{T}^{min}_{traj}, T_{raj}) - \ln P(\hat{T}^{min}_{traj}|E_{nv})$$

$$p\text{–}minFDE_{prob} = FDE(\hat{T}^{min}_{traj}, T_{raj}) - \ln P(\hat{T}^{min}_{traj}|E_{nv})$$

Овде је $\hat{T}^{min}_{traj}$ трајекторија која има најбољу FDE оцену, а $p(\hat{T}^{min}_{traj}|E_{nv})$ је условљена вероватноћа те трајекторије $\hat{T}^{min}_{traj}$ у односу на стање окружења E_{nv}. Уколико је мала грешка по метрици ADE (FDE) за одговарајућу трајекторију, али њена одговарајућа вероватноћа има малу вредност, онда негативан логаритам те вероватноће има велику вредност. [3] У импементацији се ова вероватноћа ограничава са доње

стране, како не би дошло до прекорачења због велике апсолутне вредности након примене логаритма на веома мале вредности.

Уместо упросечавања *L2* растојања у сваком кораку, могу да се броје кораци у којима трајекторије одступају за више од MR_{thresh}. Мотивација за ову метрику је чињеница да одступање које је 1 или 2 метра од реализације није толико релевантно у односу на велику разлику одступања. [6] Такође постоји верзија метрике која узима у обзир вероватноћу и кажњава предикцију модела ако је добра, а модел је ипак несигуран.

$$MR = \sum_{k=1}^{T} I(||\hat{T}_{traj}^{k} - T_{raj}||_2 \geq MR_{thresh})$$

$$\begin{gathered} MR_{prob} = \sum_{k=1}^{T} I(||\hat{T}_{traj}^{k} - T_{raj}||_2 \geq MR_{thresh}) \\ +I(||\hat{T}_{traj}^{k} - T_{raj}||_2 < MR_{thresh}) \cdot (1.0 - P(\hat{T}_{traj}^{k}|E_{nv})) \end{gathered}$$

У случају *Argoverse* скупа података, за параметар MR_{thresh} се узима 4 пиксела тј. 2 метра у реалном свету.

Све до сада наведене метрике су опште примене на било које објекте за које се предвиђају трајекторије. Пошто је агент углавном возило, може да се анализира да ли предвиђена трајекторија скреће са пута. Због тога се уводи метрика „сагланост са продучјем вожње“ (*eng. DAC - Drivable Area Compilance*), која одређује учесталост трајекторија које нису скренуле са пута од изабраних K трајекторија: [3]

$$DAC = \frac{DAC_{occurences}}{T}$$

У евалуацији модела се узимају у обзир све метрике. За параметар K се узима вредност 6.

Глава 5

Припрема података

Основни скуп података за тренирање и тестирање техника предикције трајекторија је *Argoverse Motion Forecasting* скуп података који се састоји од 324 хиљаде детаљних мапа саобраћаја (*eng. „HD maps“*) које садрже геометријске и семантичке податке сцена. Постоје две *HD* сцене за градове Питсбург и у Мајами. Коришћењем аутономних возила су генерисани сценарији који представљају неколико узастопних слика сцена (у табеларном формату) на деловима мапа. Сви детаљи о овом скупу података се могу пронаћи на адреси www.argoverse.org [3].

Кључне информације које се издвајају из сваког сценарију су:

- Мапа сценарија (Питсбург или Мајами);
- Трајекторије агената;
- Трајекторије осталих објеката на сцени;
- Путеви - возне (централне) линије.

5.1 Претпроцесирање података

Подаци сваког сценарија се векторизују и чувају у полу-структуираном формату. За парсирање и обраду улазних података се користи *argoverse API* интерфејс. Циљ овог корака процесирања података је векторизација података и релативно је независан од модела који се користи.

Трајекторија агента[1] се дели на два дела: историја (својства) и реализација (будуће вредности). Реализација се састоји од N_r опажања x и y координата тј. облик

[1]Низ (x, y) тачака, где је приближна временска разлика између две тачке око 0.1 секунде

реализације је $(N_r, 2)$.. Историја се аналогно формира да садржи историју N_h опажања x и y релативних координата. Овај део трајекторије иде непосредно пре реализације. Посматрамо следеће случајеве:

- Постоји више од $N_h + N_r$ опажања: Одбацује се реп трајекторије (првих неколико вредности хронолошки гледано);
- Постоји мање од $N_{hmin} + N_r$ опажања: Сценарио се одбацује (сматра се да је невалидан);
- Постоји између $N_{hmin} + N_r$ и $N_h + N_r$ опажања: реп трајекторије се допуњава до димензије $N_h + N_r$ посматрано као да објекат мирује у тим тренуцима.

Коначно, облик историје је $(N_h, 3)$, где трећа вредност означава да ли је опажање право (1) или допуњено (0).

Све координате се нормализују тако да су релативне у односу на последње опажање у трајекторији историје агента. Последње опажање историје агента има координату (0, 0) у новом координатном систему, а све остале тачке се транслирају за вектор $-C$, где је C оригинална позиција последњег опажања. Координатни систем се такође ротира тако да се вектор који је одређен првом и последњом тачком историје трајекторије (поједностављен правац трајекторије) поклапа са y-осом сличко као и у оригиналном *Vectornet* раду [11]. Последња трансформација је скалирање свих координата са $\frac{1}{25}$, где је ова вредност добијена анализом стандардних девијација свих тачака на сценарију (узета је просечна вредност свих сценарија). Ове трансформације су кључне за перформансе модела, јер је сам проблем који се решава значајно једноставнији:

- Применом транслације модел не мора да учи где се агент налази, јер је то увек (0, 0) координата;
- Применом ротације моделу је олакшано учење усмерења трајекторије тј. моделу је лакше да одреди оријентацију агента;
- Применом скалирања се добијају стабилнији градијенти.

Трајекторије суседних објеката се деле на два дела аналогно трајекторији агента. Неопходно је да се синхронизују трајекторије суседних објеката по временским ознакама (eng. *timestamp*) са трајекторијом агента, јер не постоји у сваком тренутку исти број објеката на сцени. Након синхронизације се трајекторије деле на историју и реализацију и проверава се да ли дужине тих делова задовољавају критеријуме:

- Уколико је дужина трајекторије историје краћа од N_{homin}, онда се објекат одбацује;

- Уколико је дужина трајекторије реализације краћа од N_{romin}, онда се објекат одбацује.

Као додатна провера, за сваки сусед се провера растојање од агента. Уколико је сусед превише далеко, онда се се он одбацује. Критеријум за одбацивање суседа узима у обзир брзину агента (по x и y оси одвоједно) и растојање њихових последњих опажања у трајекторији историје. Уколико неки од следећих услова није испуњен, онда се сусед игнорише у сценарију: $\frac{O_n^x}{A_s^x} \leq T_{steps}$, $\frac{O_n^y}{A_s^y} \leq T_{steps}$, где је O_n^x (O_n^y) нормализована x (y) координата суседа, $\overset{x}{s}$ ($\overset{y}{s}$) је наивно апроксимирана[2] брзина агента по x (y) оси и T_{steps} је параметар толеранције. Трајекторије се секу или допуњавају до фиксног облика. Векторизован облик: $(N_n, N_h, 3)$ и $(N_n, N_r, 3)$, где је N_n број судедних објеката.

На основу локације агента се издвајају сегменти централних линија (возне путање) које нису даље од агента за више од D_{lsinit}. Уколико нема пронађених сегмената централних линија, онда се вредност за D_{lsinit} множи K_{ls}[3] пута до највише D_{lsmax} (ако и даље нема сегмената, онда се сценарио одбацује). За сваки сегмент се чува низ од 10 (x, y) координата приширених са метаподацима:

- *is_ intersection* - да ли се сегмент сече са неким сегментом,
- *turn_ right* - да ли је у питању скретање у десно,
- *turn_ left* - да ли је у питању скретање у лево,
- *turn_ none* - да ли нема стретања,
- *is_ traffic_ control* - да ли постоји контрола саобраћаја.

Коначан облик је $(N_{ls}, 10, 7)$.

Скуп кандидата централних сегмената линија за предикције трајекторија (сегменти на којим можемо да очекујемо): Постоји коначан број централних сегмената линија по којој објекат може да се креће у скоријој будућности, због чега је корисно да се као улаз у модел (директно или индиректно) користе централне линије као кандидати. Основа алгоритма за проналазак ових кандидата се налази у *argoverse* интерфејсу [3]. Кандидати се проналазе коришћењем трајекторије историје агента. Коначан векторизован облик је: $(N_c, N_r, 3)$, где је N_c број пронађених кандидата, N_r дужина трајекторије реализације. Пошто се путеви допуњавају по потреби до димензије N_r, користи се трећа координата за маску.

Argoverse интерфејс није увек у могућности да нађе све адекватне кандидате централних сегмената линија. За додатне кандидате централних линија се користи

[2]Брзина се апроксимира као просек промена координата у трајекторији историје

[3]D_{lsinit} и K_{ls} су фиксне вредности у *argoverse* интерфејсу

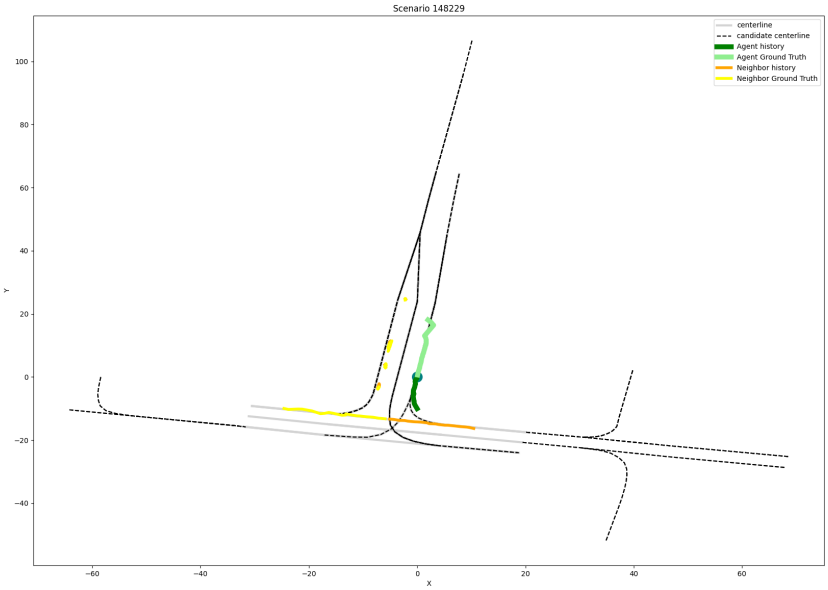
Ознака параметра	Објашњење
N_r	Дужина трајекторије реализације (део који се предвиђа)
N_h	Дужина трајекторија историје
N_{hmin}	Минимална дужина трајекторије историје пре допуњавања
N_{hmin}	Минимална дужина трајекторије историје суседа пре допуњавања
N_{rmin}	Минимална дужина трајекторије реализације суседа пре допуњавања
T_{steps}	Умножак максималног растојања до сегмента пута
D_{lsmax}	Максимално растојање до пута

Табела 5.1: Преглед параметара припреме података

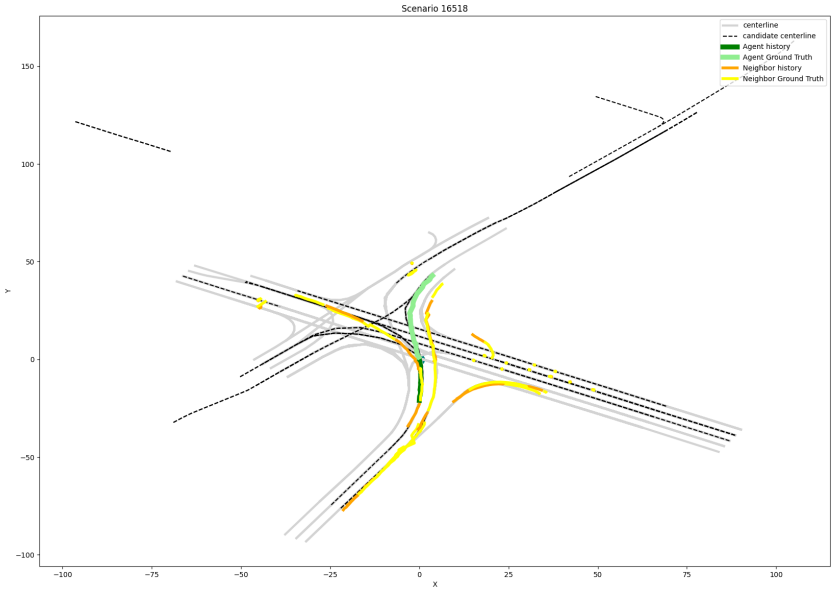
модел константне брзине са примењеном трансформација скалирања и ротација самих брзина. Пример: Ако је брзина кандидата $(1, 0)$, onda се у разматрање узимају и вредности $(2, 0)$ (скалирање са 2), $(\frac{1}{2}, 0)$ (скалирање са $\frac{1}{2}$), $(0, 1)$ (ротација за 90 степени), $(\sqrt{2}, \sqrt{2})$ (ротација за 45 степени), ... На основу брзине се добија апроксимирана позиција последње тачке трајекторије помоћу модела константне брзине: $T \cdot V_i$, где је T дужина трајекторије која се предвиђа, а V_i брзина.

Погледати табелу 5.1 за преглед свих параметара процеса.

На сликама 5.1 и 5.2 се налазе примери два визуализована сценарија након претходне припреме. У овом формату нису прикази делови сцене на којој је могућа вожња, али постоје (централне линије) тј. путање по којима се возила најчешће крећу. Изузеци су у случају неких скретања, промени линија, ...



Слика 5.1: Визуализација припремљених података - Пример 1



Слика 5.2: Визуализација припремљених података - Пример 2

Глава 6

Техника заснована на разумевању контекста обрадом сцене представљене графом

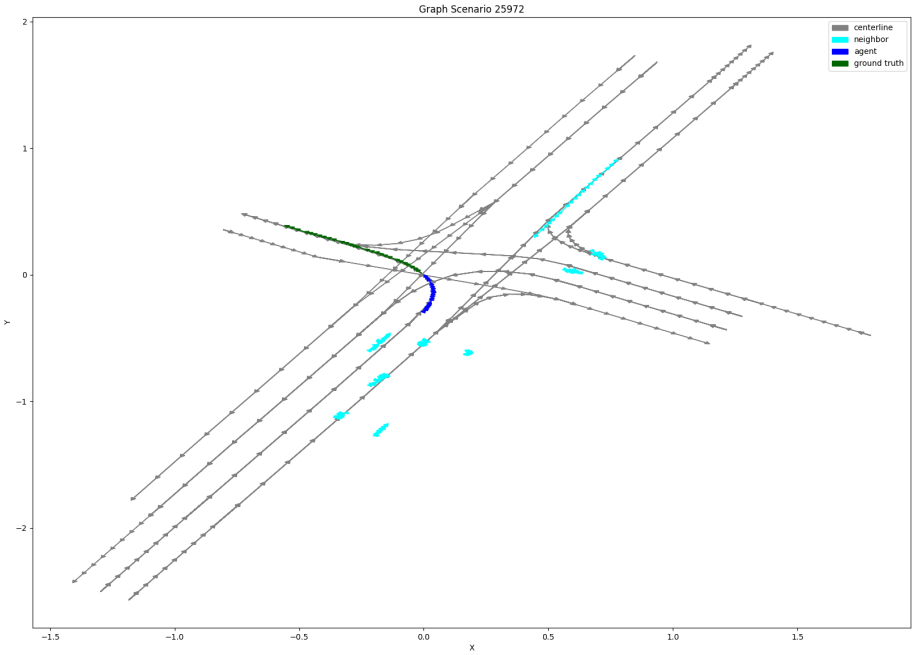
Независно од конкретног скупа података *HD* мапа, сваки сценарио може да се лепо представи графовском структуром која повезује тачке на сцени, где свака тачка има своја својства:

- Трајекторија агента и објеката је скуп тачака усмереног графа - сложена незатворена линија (*eng. polyline*) (сваки чвор је повезан са следећим).
- Путеви могу да се представе на исти начин као и објекти, али разлика би била у самим својствима чворова помоћу којих се путеви разликују од објеката.
- Пешачки прелаз је полигон, који може да се представи као сложена затворена линија.

Овакав приступ представљања сцена на апстрактном нивоу је лако прилагодљив било ком скупу података. Начин даље трансформације компоненти овог графа зависи од самог модела. [11]

6.1 Модел *VectorNet*

Као што је у уводном делу ове гране објашњено, сваку трајекторију и сегментн пута на сцени можемо да представимо сложеном линијом. Сваки чвор сложене линија садржи издвојена својстава. *VectorNet* је хијерархијска графовска неуронска мрежа која се састоји из три компоненте: [11]



Слика 6.1: Визуализација репрезентације сложеним линијама

- Конструкција подграфа: Агрегација сложених линија у један вектор који се даље посматра као чвор у глобалном графу;
- Моделовање интеракција на високом нивоу у глобалном графу;
- Предикција трајекторија за чворове глобалног графа који одговарају агентима.

Подграф

Свака сложена линија на сцени се посматра као посебан граф. Применом варијанте графовске неуронске мреже се издвајају битна својства и везе између чворова. Резултат се на крају агрегира како би се добио један вектор који представља репрезентацију

Архитектура подграфа је варијанта *GCN* архитектуре [17] са више слојева. Визуализација једног слоја се види на слици 6.2. Сваки слој може да се представи следећом формулом:

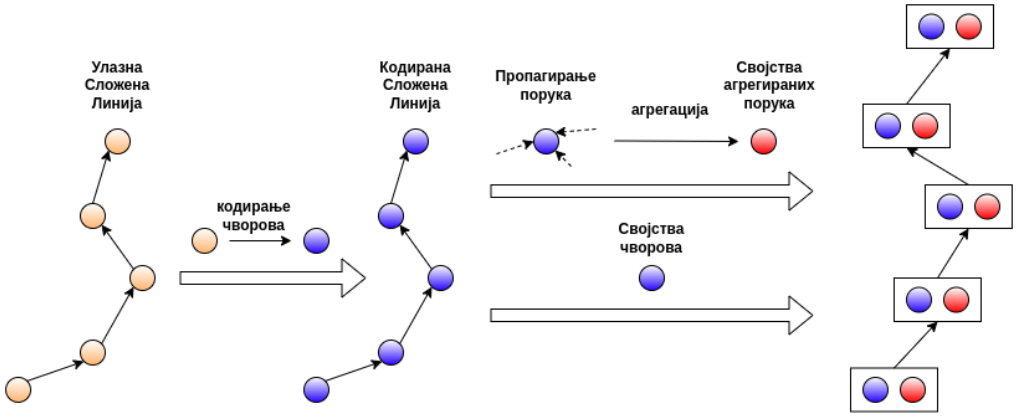
$$v_i^{(l+1)} = \rho_{rel}(g_{enc}(v_i^{(l)},\ \rho_{agg}(A, \{g_{enc}(v_j^{(l)})\})))$$

Овде је $v_i^{(l)}$ вектор својства чвора из претходног слоја ($v_i^{(0)}$су улазни подаци), g_{enc} слој за енкодирање својства чвора, ρ_{agg} операција агрегације порука добијених од суседа, A је матрица повезаности сложене линије, а ρ_{rel} операција обједињавања агрегираних порука суседа и својства самог чвора. Избор операција је произвољан. Конкретно: [11]

- g_{enc} је потпуно повезана неуронска мрежа;
- ρ_{agg} је максимум (*max pool*);
- ρ_{rel} је конкатенација својста чвора и агрегираних порука;
- A је у оригиналном раду матрица повезаности потпуно повезаног графа. Алтернативе су повезаност једног чвора са следећим или свим следећим.

Вектор сложене линије се добија агрегацијом добијеног графа последњег слоја:

$$P_{feat} = \rho_{agg}(\{v_i^{(L)}\})$$



Слика 6.2: Визуализација једног слоја

Из угла имплементације, улаз у ову компоненту је модела је вектор димензије (B, P, T, F), где је B димензија подскупа података једног корака, P је број сложених линија у једном графу, T је дужина сложене линије, а F је број својстава. Након L слојева се добија вектор димензије $(B, P, T, F \cdot 2^L)$ који се агрегира на нивоу чворова на вектор димензије $(B, P, F \cdot 2^L)$[1]. Агрегирану сложену линију посматрамо као један чвор у глобалном графу.

[1]Претпоставка је да свака сложена линија има исти број својстава и да сваки граф има исти број сложених линија

Глобални граф интеракција

Глобални граф интеракција се посматра као класичан граф са матрицом повезаности A преко које може да се користи хеуристика, као што је на пример удаљеност чворова на сцени. У оригиналном раду се ради једноставности глобални граф посматра као потпуно повезан граф без тежина. [11]

$$\{P_i^{(l+1)}\} = GNN(\{P_i^{(l)}\}, A)$$

Графовска неуронска мрежа је имплементирана као механизам пажње [12]:

$$GNN = softmax(\frac{P_Q P_K^T}{\sqrt{d_K}})P_V$$

Циљ ове компоненте је разумевање интеракција између агената и осталих објеката на сцени: који објекти у околини су у интересу, који путеви у околини су у интересу, ...

Предикција трајекторија

За предикцију трајекторија се филтрирају чворови агената из глобалног графа интеракције. За сваког агента се примењује призвољан *decoder* модел чији је излаз предикција трајекторије. Најједноставнији приступ је коришћење потпуно повезане неуронске мреже под претпоставком да су тачке трајекторије међусобно независне. Овај корак се замењује напреднијим приступоп у модификованој верзији описаној у следећој секцији.

Функција грешке

За функцију грешке предикције трајекторија L_{traj} се користи *huber* функција грешке тј. хибрид *L1* и *L2* функције грешке. У процесу тренирања може да се дода задатак **естимације недостајућих чворова глобалног графа интеракција**. Насумично се бирају чворови у графу и замаскирају се његова својства (множењем са нулом). Сваком чвору се додају две „нове“ вредности које су једнаке минималној вредности свакој од координата почетних тачака сложене линије. [2]. Нове вредности представљају идентификаторе тих сложених линија који се користе приликом тренирања модела за естимацију недостајућих чворова. Циљ овог задатка је форсирање бољег разумевања веза између трајекторија и генерално између сложених линија у графу. За функцију грешке овог задатка (L_{node}) се користи *L1* функција грешке. [11]

[2]Опис својстава сложених линија је описан у секцији за припрему података

$$L = L_{traj} + \alpha \cdot L_{node}$$

6.2 Модификована верзија *VectorNet* са узоркованим крајњим тачкама трајекторија

Претходно поменута верзија *VectorNet* модела може да се унапреди додавањем доменског знања. Један приступ је да се користе унапред дефинисани предлози трајекторија који се користе као основе за генерисање предикција трајекторија. Предлози се бирају на основу анализе података (кластеровање). Ова метода је аналогна примени предлога („сидра“ - *eng. anchors*) која се користи у детекцији објеката.

Алтернатива која се исто заснива на предлозима („сидрима“) је да уместо предлога трајекторија користе предлози крајњих тачака трајекторија. Модел који генерише трајекторију на основу крајње тачке и својства из *VectorNet* се лако тренира, али квалитет целог система доста зависи од квалитета узорковања тих предлога.

Архитектура *tnt: Target-driveN Trajectory Prediction* [**?**] се заснива баш на тој идеји:

1. Разумевање контекста помогућу *VectorNet* модела (основа);
2. Генерисање корекција и поузданости за сваки предлог крајње тачке трајекторије;
3. Узорковање модификованих предлога на основу поузданости;
4. Естимација трајекторије до сваког изабраног предлога крајње тачке.
5. Естимација вероватноћа за сваку од добијених трајекторија;
6. Филтрирање трајекторија на основу вероватноћа.

Уместо последња два корака могу да се користе вероватноће (поузданости) предлога крајњих тачака, али из добре крајње тачке не следи нужно да је добијена трајекторија такође квалитетна.

Овде *VectorNet* чини језгро архитектуре, а сви наредни кораци могу да се имплементирају потпуно повезаним неуронским мрежама и једноставним детерминистичким алгоритмима. Главни изазов ове архитектуре је у алгоритму за генерисање предлога [3] и балансирању параметара функција грешака приликом учења модела. Ако се не учи *естимације недостајућих чворова глобалног графа интеракција*, онда се функција грешке има следећи облик:

[3] Алгоритам је укратко описан у секцији за иницијалну припрему података (TODO: вероватно ће бити модификован и детаљније објашњен)

$$L = \lambda_1 \cdot L_{offsets} + \lambda_2 \cdot L_{tarconf} + \lambda_3 \cdot L_{trajde} + \lambda_4 \cdot L_{trajconf}$$

Нека је $T_{anchors}$ скуп свих предлога крајњиј тачака трајекторија и нека је P_{gt} истинита крајња тачка трајекторије. Тада се изваја из скупа $T_{anchors}$ елемент $P_{closest}$ који је најближи истинитој крајњој тачки[4]. Тада је:

$$L_{offsets} = H_{delta}(P_{closest\_offset}, \hat{P}_{closest\_offset})$$

где је H_{delta} *huber* (TODO: dodati formulu u dodatku) функција грешке са параметром *delta*, $P_{closest\_offset} := P_{gt} - P_{closest}$, а $\hat{P}_{closest\_offset}$ је предикција тог одступања. Циљ је да баш тај најближи предлог има највећу поузданост, па се функција грешке за поузданост дефинише на следећи начих:

$$L_{tarconf} = BCE(P_{closest\_onehot}, \hat{P}_{confs})$$

где BCE је бинарна унакрсна ентропија, $P_{closest\_onehot}$ индекс елемента $P_{closest}$ у *onehot* формату[5] и $\hat{P}_{confs}$ поузданост модела за сваки предлог (за сваки предлог се даје поузданост из интервала $[0, 1]$). Функција грешке за трајекторије је аналогна као и за предлога крајњих тачака. Састоји се из функције грешке за одступање трајекторија од реализације и оцене поузданости модела за сваку од тих трајекторија.

$$L_{trajde} = H_{delta}(_{traj}, \hat{T}_{traj})$$

где је $_{traj}$ истинита вредност трајекторије, а $\hat{T}_{traj}$ је њена предикција. Приликом учења се за естимацију трајекторије $\hat{T}_{traj}$ узима истинита крајња тачка трајекторије како би учење било стабилније. Ова техника се зове „Учитељско форсирање“ [18]. Последња компонента се односи на поузданост модела за сваку естимацију трајекторија. За сваку естимирану трајекторију се рачуна максимално растојање између свих упарених тачака естимиране трајекторије и истините вредности трајекторије тј. $D(T_{traj}, \hat{T}_{raj}) := max(||T^k_{traj} - \hat{T}^k_{traj}||^2_2)$. Тада се „истинита расподела“ P_{traj_confs} одређује као *softmax* ових негативних вредности.

$$L_{trajconf} = BCE(P_{traj\_confs}, \hat{P}_{traj\_confs})$$

[4]Овде не узимамо у обзир поправке, већ детерминистичке вредност. Уколико бисмо узели у обзир са поправкама, онда процес учења постаје тежак због шума (мења се индекс најближе тачке). Ово можда не би био толики проблем да су остале трајекторије другачије дефинисане тј. независне од промене изабране, најближе тачке.

[5]Индекс се представља као низ димензије броја класа (предложених крајњих тачака у овом случају) где су свуда нуле сем на локацији која одговара вредности тог индекса.

6.3 Припрема података

Припрема података у овој секцији се наставља на иницијални процес претпроцесирања података и своди се на трансформацију свих елемената *HD* мапа у сложене линије. Да би било могуће спојити више сценарија у један подскуп података за једну итерацију тренирања, неопходно је да се испуне одређена ограничења:

- Сваки сценарио мора да има исти број сложених линија (N_p). Ако је број сложених линија већи од N_p, онда се вишак сложених линија одбацује, при чему се води рачуна о приоритету сложеих линија: агент, остали објекти, путеви, путеви кандидати. Ако је број сложених линија мањи од N_p, онда се скуп допуњава сложеним линијама са нула чворовима.

- Свака сложена линија мора да има исти број чворова (N_n). Ово се решава аналогно броју сложених линија.

- Сваки чвор сложених линија мора да има исти број својстава (N_f). Сваки чвор има иста својства, са тим да се својства допуњавају нулама ако немају смисла за тај тим сложених линија (агент и путеви немају иста својства).

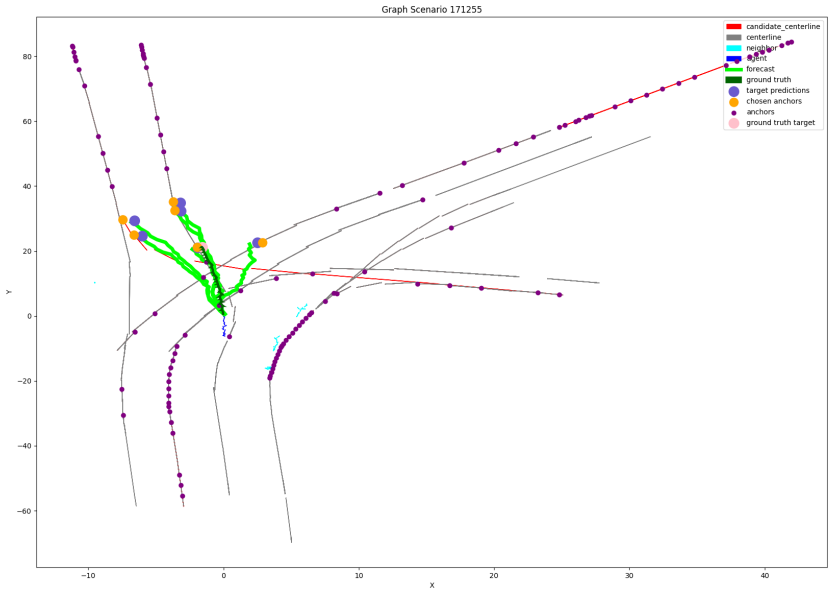
За сваки чвор сложене линије има следећа својства:

- Координате x и y: Просек претходне и тренутне тачке трајекторије - 2 скалара;
- Смер по x и y: Разлика тренутне и претходне тачке трајекторије - 2 скалара;
- Тип објекта као *onehot* вектор (агент, сусед - објекат, пут, пут кандидат) - 4 скалара;
- Метаподаци путних чворова
  - Да ли се сегмент пресеца са неким другим сегментом - 1 скалар;
  - Да ли постоји контрола саобраћаја - 1 скалар;
  - Смер као *onehot* вектор (нема, десно, лево) - 3 скалара
- Да ли је чвор прави (1) или вештачки (0) - 1 скалар.

Коначан формат улазних података у *VectorNet* је $(B, N_p, N_n, 14)$, где је B број сценарија у једном подскупу података, а N_p и (N_n) су параметри.

6.4 Експерименти и резултати

На слици 6.3 је приказан пример примене *TNT-VectorNet* архитектуре.



Слика 6.3: Пример резултата за сценарио 171255 - $minADE_6 = 0.6, minFDE_6 = 1.0$. Сивом бојом су обојени путеви, тамно плавом је обојена историја трајекторија агента, светло плавом су обојене историје суседних објеката, црвеном линијом су обојени путеви кандидати. Мали љубичасти кругови су узорковани предлози, велики плави кругови су изабрани узорковани предлози са највећом поузданошћу, а велики наранџасти кругови су модификовани изабрани предлози. Светло зеленом бојом су представљене предикције за сваку од изабраних крајњих тачака, а тамно зеленом бојом су обојени истинити резултати.

Глава 7

Техника заснована на разумевању контекста обрадом растеризоване сцене

TODO: За овај део бих као енкодер такође користио VectorNet уместо растеризоване сцене уз растеризоване топлотне мапе.

Глава 8

Закључак

У изради...

Библиографија

[1] Y. Huang and Y. Chen, "Autonomous driving with deep learning: A survey of state-of-art technologies," in *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2020. Arxiv preprint: 2006.06091.

[2] "Lyft motion prediction for autonomous vehicles - kaggle competition." https://www.kaggle.com/c/lyft-motion-prediction-autonomous-vehicles.

[3] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," in *CVPR*, 2019. Arxiv preprint: 2103.11624.

[4] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces."

[5] Gupta, Agrim, Johnson, Justin, Fei-Fei, Li, Savarese, Silvio, Alahi, and Alexandre, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, no. CONF, 2018.

[6] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "Home: Heatmap output for future motion estimation," 2021. Arxiv preprint: 2105.10968.

[7] X. Zhou, D. Wang, and P. Krahenbuhl, "Objects as points (centernet)," 2019. Arxiv preprint: 1904.07850.

[8] M. Schafer, K. Zhao, M. Buhren, and A. Kummert1, "Context-aware scene prediction network (caspnet)," 222. Arxiv preprint: 2101.06933.

[9] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. kin Wong, and W. chun Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," 2015. Arxiv preprint: 1506.04214.

[10] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, "Learning lane graph representations for motion forecasting," in *ECCV*, 2020. Arxiv preprint: 2007.13732.

[11] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," in *CVPR*, 2020. Arxiv preprint: 2005.04259.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017. Arxiv preprint: 1706.03762.

[13] M. Ye, T. Cao, and Q. Chen, "Tpcn: Temporal point cloud networks for motion forecasting," in *CVPR*, 2021. Arxiv preprint: 2103.03067.

[14] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017. Arxiv preprint: 1612.00593.

[15] M. F. Apratim Bhattacharyya, Bernt Schiele, "Accurate and diverse sampling of sequences based on a "best of many" sample objective," in *CVPR*, 2018. Arxiv preprint: 1806.07772.

[16] Chen, Guangyi, Li, Junlong, Zhou, Nuoxing, Ren, Liangliang, Lu, and Jiwen, "Personalized trajectory prediction via distribution discrimination," in *ICCV*, 2021.

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the 5th International Conference on Learning Representations*, 2016. Arxiv preprint: 1609.02907.

[18] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks. neural computation," in *Neural Computation (Volume: 1, Issue: 2, June 1989)*, 1989. ISSN 0899-7667.

Биографија аутора

Момир Аџемовић У изради...