KNN – Car keypoint detection

Dvořák Martin (xdvora2l)

Halva Vladislav (xhalva04)

2020

Abstrakt

Tento článek se zabývá problematikou stanovení pozice klíčových bodů (dále keypoints) vozidel z reálně získaného obrázku z provozu. Jsou zde prezentovány dvě kategorie přístupů, konkrétně metody založené na extrakci příznaků a metody využívající architekturu *stacked hourglass*, která je založena na opakovaném zpracování shora-dolů a zdola-nahoru a k učení modelu využívá klíčové body v podobě "tepelných map" (dále heatmaps). Vstupními prvky jsou běžně RGB obrázky vozidel různého rozlišení, získané a zpracované jako snímky z jedné nezávislé kamery. Každý snímek je posuzován nezávisle na ostatních. Dále se pracuje s předem označenými *keypoints* vozidel ve formě dvaceti bodů ve dvourozměrném souřadném systému. Dále je popsán námi zvolený přístup využívající zmíněný *stacked hourglass* model a experimenty provedené s tímto modelem. Na závěr jsou vyvozeny výsledky těchto experimentů.

Klíčová slova - zpracování obrazu, hluboké učení, konvoluční neuronové sítě, hodinový model, odhad polohy vozidla, klíčové body

1 Úvod

Odhad polohy a orientace vozidel je široce zkoumané téma, které je již nyní esenciální pro velké množství aplikací např. re-identifikace vozidel, rozpoznávání SPZ, implementace chytré dopravy, a samozřejmě největší aktuální téma, kterým je autonomní řízení vozidel. Zejména pro systémy spadající do kategorie autonomních vozidel je nutné *real-time* zpracování obrazu, kdy je kladen důraz na rychlost výpočtu a zároveň na vysokou kvalitu. Klíčové pro tyto aplikace je rozpoznání předmětů na vozovce a určení jejich co nejpřesnější pozice a orientace. Je nutné, aby systém chápal jednotlivé předměty získané kamerami a senzory, a dokázal z nich např. usoudit jakým směrem se dané vozidlo pohybuje, což může být výpočetně náročné. Pro zrychlení výpočtu se často používají techniky založené na zpracování obrazu, které z kamery tyto informace dokáží extrahovat. Aplikace technik založených na *deep learning* však v této problematice udělala obrovský krok dopředu a dnes v tomto odvětví dominují.

2 Existující přístupy

Existující metody řešení problému lze rozdělit do dvou hlavních kategorií. První kategorie metod zakládá na předzpracování snímku a extrakci příznaků. Tyto metody pracují s databází příznaků, ve které hledají shodu dominantních příznaků pro další zpracování vstupního obrázku. Druhou hlavní skupinu řešení představuje architektura *hourglass*, resp. *stacked hourglass*, která se inspiruje plně konvolučními sítěmi např. sítí *U-net* [2]. Tyto modely pracují na principu *pooling* a *up-sampling*, které představují jádro celého modelu.

2.1 Metody založené na extrakci příznaků

Tato kategorie zahrnuje mnoho různých přístup. První přístup o kterém se zmíníme jsou metody, které

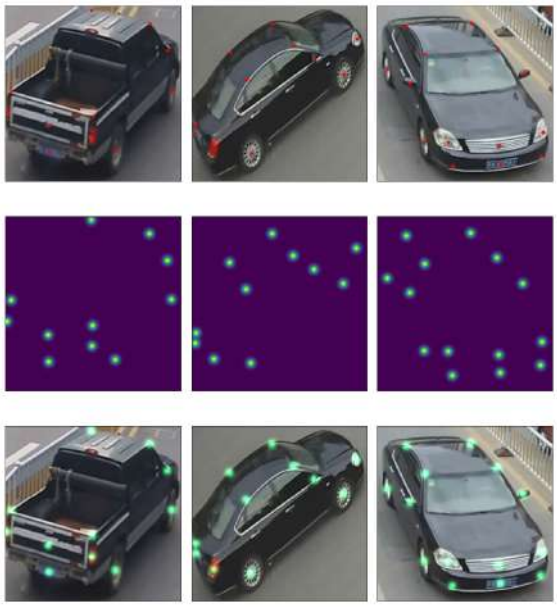

Figure 1: Použité obrázky a označené *keypoints* pocházejí z databáze VeRi-776 [1]. První řádek představuje zobrazení *keypoints* v původním obrázku. Druhá řada představuje vytvořené *heatmaps* sloučené do jednoho snímku pro konkrétní vozidlo zobrazené ve sloupci. Poslední řada ukazuje *heatmaps* v reálném snímku.

se pokouší snímek napasovat na předem vytvořené modely vozidel [3][4][5][6][7]. V předem připravené databázi se nacházejí 3D modely aut, které lze detekovat ve snímcích. Při nalezení shody auta z obrazu s objektem v databázi jsou dopočítány viditelnosti a *keypoints*. Pro tyto modely je společný rys *loss funkce*, která se skládá z několika lokálních funkcí pro daný podproblém.

Druhou často aplikovanou možností je předzpracování obrazu pomocí konvoluční části sítě následované plně propojenými vrstvami [8][9][10]. Tyto sítě vycházející z architektury *VGG* [11]. Jejich společným prvkem je rozdělení výpočtu do několika nezávislých bloků, které lze vykonávat paralelně. Model [8] například využívá při paralelním počítání sdílení vah v paměti. V důsledku mají obě větve sítě stejné váhy. Výstupem těchto sítí je vektor čísel, které reprezentují odhad požadované vlastnosti. Před finálním výpočtem se musí nacházet vrstva, která provádí sdružení výsledků z různých větví výpočtu. Tyto sítě jsou výpočetně méně náročně.

2.2 Metody založené na hodinovém modelu - *hourglass model*

Další skupinou jsou modely používající architekturu *hourglass*, která byla úspěšně použita například pro řešení problému detekce pozice lidí v obraze[12][13]. Tato architektura není komplikovaná a zároveň je schopna získat z obrazu mnoho informací na různých vrstvách abstrakce, které je schopna spojit do výstupu. Díky použití residuálních bloků je také možné vytvořit poměrně hluboké sítě. Výstup je realizován pomocí *heatmaps* nebo jiným kombinovaným přístupem (často kombinace heatmaps a vektoru příznaků). Hlavní výhoda této architektury spočívá v odhadu *keypoints* z celého snímku. Síť je tedy nucena pro řešení problému pochopit strukturu vozidel.[14][15][16][17]

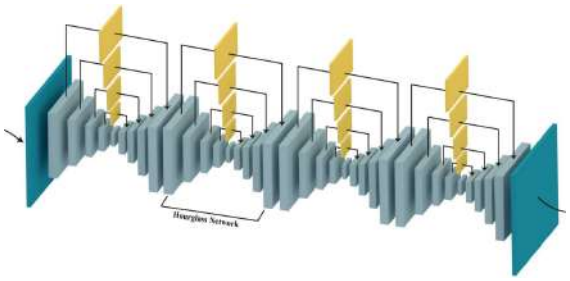

Figure 2: Grafické znázornění architektury *hourglass*, která je ve variantě čtyřikrát zduplikována a zapojena za sebe (tzv. *stacked hourglass*) [15].

2.2.1 Architektura

Model typicky na vstupu očekává obrázek specifikované velikosti. Samotnému modelu pak často předchází blok sestávající z konvoluční vrstvy a poolingu, který upraví rozměr a počet kanálů na požadovaný formát vstupu samotného *hourglass* modelu (počet zde vytvořených kanálů dále vysoce

ovlivňuje počet příznaků, které se je síť schopna naučit).

Hourglass model se skládá ze bottom-up a top-down části. Obě části využívají residuální bloky pro zpracování dat, v případě bottom-up části jsou však prokládány pooling vrstvami, které zmenšují rozměr na polovinu, a v případě top-down upsampling vrstvami, které naopak zdvojnásobují rozměr. Tyto části jsou vždy stejně dlouhé. V důsledku mají data na výstupu stejný rozměr, jako na vstupu modelu. Odpovídající si residuální vrstvy (stejná úroveň - stejný rozměr) jsou navíc vždy propojeny *skip vrstvou*, která slouží jako další residuální blok vyšší úrovně. *Skip vrstva* je v obrázku 2 zobrazena žlutým čtvercem.

Residuální blok se skládá ze třikrát se opakující posloupnosti vrstev *batch normalizace*, *ReLU* a *konvoluce*. Konvoluce mají rozměr 1×1, 3×3 a 1×1. Na výstupu této vrstvy se nachází residuální propojení se vstupem do tohoto složeného bloku. Vstupní obraz je tedy stejně velký jako výstupní.

Tato architektura se používá ve zřetězené formě *stacked hourglass*, obrázek 2 ukazuje zřetězení čtyř těchto bloků. Mezi dílčími *hourglass* architekturami se předávají data ve stejné velikosti. Tato velikost je určena vrstvou předzpracování, která se nachází před architekturou *stacked hourglasss*. Počet použitých kanálů například u modelu použitého pro detekci pozice osob v obraze roven 256 [12]. V rozšířené variantě tohoto modelu lze výstup posledního dílčího *hourglass* modelu rozdělit na dvě identické větve, kde v jedné je počítána klasická varianta s *heatmaps* a druhá větev počítá svůj výstup pomocí *plně propojených* vrstev.

2.2.2 Loss funkce

Jestliže jsou výstupem modelu *heatmaps*, tak chybová funkce představuje regresi hodnot, je tedy realizována pomocí rovnice 1 funkcí *mean square error*. Neznámá x reprezentuje odhadnutou hodnotu modelem (prediction) a neznámá gt představuje očekávanou hodnotu výstupu (ground truth). Propagace chyby je provedena z každého výstupu hodinového bloku směrem ke vstupu. Pokud se síť skládá ze čtyř zřetězených hodin je tedy chyba počítána ze čtyř míst (dílčích výstupů).

Pokud je přidána plně propojená vrstva, chybová funkce se skládá ze součtu *mean square error* a funkce *soft-max*, která je zobrazena rovnicí 2.

$$L_1 = \frac{1}{N}\sum_i (x_i - gt_i)^2 \quad (1)$$

$$L_2 = log(\frac{e^{y_i}}{\sum_i e^{y_i}}) \quad (2)$$

2.2.3 Heatmaps

Použití heatmaps napomáhá zrychlení konvergence řešení jelikož rozšiřuje oblast, kde je výsledek považován za správný, resp. blízko správnému řešení. Plocha snímků je příliš velká pro použití ground-truth v podobě jednoho bodu, proto je zde vhodné použití rozptylu, která právě heatmaps zavádějí. Heatmap proto místo jednoho bodu reprezentuje řešení v podobě dvoudimenzionální Gaussovy funkce s maximem v bodě, kde se nachází *keypoint* (toto místo je tedy střední hodnotou) a stanoveným rozptylem. Řešení pak v ideálním případě konverguje k maximu této funkce. V každé *heatmap* je zobrazen právě jeden *keypoint* formou této Gaussovy funkce. Pokud daný *keypoint* nebyl zaznačen v referenčních výstupech (ve většině případů se jedná o bod, který není viditelný) potom mapa neobsahuje žádné hodnoty, resp. jsou všechny nulové. Příklad konkrétní mapy je zobrazen na obrázku 3.

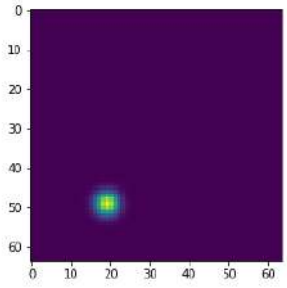

Figure 3: Grafické znázornění heatmap ukazující keypoint reprezentující levé přední kolo vozidla.

3 Popis řešení

V rámci našeho řešení byl vybrán přístup, který využívá *stacked hourglass* model z několika důvodů. Tento model je implementačně poměrně přímočarý a zároveň patří do kategorie *State of the Art*. Zároveň model nevyžaduje extrakci příznaků, či jinou podobnou aplikaci znalosti o doméně problematiky, či použití databáze modelů vozidel a podobně. Model také poskytuje širokou variabilitu ve nastavení jeho parametrů či jeho modifikaci pro konkrétní úlohu.

Vybraný model byl implementován v jazyce *Python 3* s použitím knihovny *Pytorch*. Samotný model je inspirovaný řešením [20]. Dále byla implementována část preprocessingu vstupních dat, včetně mechanismu efektivního načítání a transformací, a vytvoření heatmaps jako ground-truth a loss funkce odpovídající implementaci těchto map. Byl vytvořen trénovací algoritmus pracující s optimalizačním algoritmem ADAM a batch o velikosti 16. Loss funkce je realizována kombinovanou funkcí 1 pro heatmaps (kombinovanou ve smyslu kombinace výstupu sítě ve více bodech výpočtu).

K trénování a testování sítě byl použít dataset [1], který obsahuje trénovací set o velikosti 30 000 obrázků a testovací o velikosti 12 000 obrázků s již přiloženými soubory s označenými keypoints. Obrázky jsou pořízeny z několika různých kamer a obsahují širokou škálu typů vozidel, z různých úhlů pohledu a různých rozměrů.

Model byl implementován pro vstupní velikost obrázku 64×64. Trénovací a testovací sada disponuje proměnlivou velikostí obrázků. Fáze předzpracování tedy změní rozměr vstupních obrázků na požadovaný rozměr 64×64 a adekvátně také upraví hodnoty ground-truth, tedy pozice keypoints. Vrstva preprocessingu dále pracuje již se vstupem korektních rozměrů a transformuje jej pomocí několika konvolučních vrstev na požadovaný vstupní počet kanálů (v našem případě 128). V jednotlivých hodinových modelech je nastavena hloubka zanoření na hodnotu 4 (v sekcích *bottom-up* a *top-down* se tedy čtyřikrát změní rozměr obrazu a provede se stejný počet *residuálních bloků*. Hodnoty dále nezmíněných parametrů byly nastaveny dle použití v [20].

4 Experimenty a výsledky

Jako objekt námi provedených experimentů byl zvolen parametr počet *hourglass* modelů, tedy počet hodinových bloků, které tvoří *stacked hourglass* strukturu. Tento parametr byl zvolen zejména kvůli vysokému vlivu na dobu výpočtu a jeho vliv na přesnost je tak poměrně zajímavým rysem. Bylo provedeno trénování na modelech o velikosti 1, 2, 3 a 4 zřetězených hodin. Počet kanálů 256, který byl použit v rámci řešení úlohy pozice osob [13] nemohl být realizován z důvodu nízké velikosti GPU paměti v trénovacím a testovacím prostředí *Google Colaboratory*, proto bylo v rámci našich experimentů použito pouze 128 kanálů, tento počet by však neměl mít markantní dopad na přesnost vzhledem k i tak poměrně vysokému počtu.

Každý z uvedených modelů byl trénován na 100 epochách a průběžně po pěti trénovacích epochách vyhodnocován na testovacích datech. Výslednou úspěšnost v rámci těchto běhu lze vidět v grafu 4, který zobrazuje úspěšnost jednotlivých modelů v různých fázích trénování. Úspěšnost je uvedena pro všechny definované klíčové body, které jsou na obrázcích označeny viz 5.

Určení korektnosti pozice klíčového bodu z predikce, tedy heatmap, která je výstupem sítě, je určen jako vzdálenost maximální hodnoty *ground truth* a predikce sítě (jde tedy o eukleidovskou vzdálenost středních hodnot dvourozměrných Gaussovských funkcí). Následně je tato vzdálenost porovnána s prahem, který reprezentuje hranici tolerovaného nepřesnosti/šumu. Hranice byla empiricky odhadnuta a nastavena na hodnotu 3.0. Pokud je tedy vzdálenost menší než stanovená hranice, je bod označen jako správně určený a bude započten mezi korektní výsledky. Pokud hodnota překročí stanovenou hranici, bude bod označen jako špatně určený. Z provedených experimentů bylo zjištěno, že výsledné predikce vždy přesně nekopírují Gaussovské rozložení hustoty pravděpodobnosti a nemohla tak být použita varianta sumy rozdílu jednotlivých pixelů map, popřípadě suma druhých mocnin rozdílů. Nicméně i v případech, kdy predikce toto rozložení poměrně respektuje bylo určení tímto způsobem velice málo vypovídající o korektnosti

určení klíčového bodu.

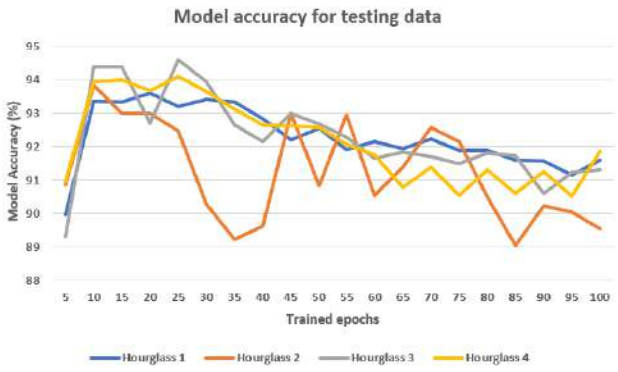

Figure 4: Úspěšnost testovaných modelů v různých fázích (epochách) trénování. Trénováno bylo na 100 epoch.

Pro každý model byl dále určen nejlepší počet trénovacích epoch podle celkové úspěšnosti dosahované na trénovací sadě. U modelu s pouze jedním hodinovým modelem bylo dosaženo nejlepších výsledků při dvaceti epochách. Pro model 2 hodin je to deset epoch. Model o 3 zřetězených hodin dosahoval nejlepší výsledky na dvaceti-pěti epochách. Největší model o čtyřech hodinách dosahoval nejlepší výsledky také na dvaceti-pěti epochách.

Všechny křivky, které lze vidět v 4 mají po začátečním rostoucím trendu dlouhodobě klesající směr. Tento trend lze přisuzovat přeučení sítě. Klesající trend u všech modelů provázejí lokální výkyvy. V závislosti na velikost sítě lze vidět, že modely potřebují k naučení poměrně malý počet epoch než bylo původně očekáváno. Podle nejlépe ohodnocených modelů můžeme pozorovat předpokládaný fakt, že je potřeba více epoch na trénování větší konvoluční sítě (více zřetězených hodin).

Nejlepší výsledky vykazuje model se třemi zřetězenými hodinami na prvních třiceti epochách s propadem na dvacáté epoše. Lze pozorovat téměř konstantní úspěšnost modelu se čtyřmi zřetězenými hodinami na prvních třiceti epochách. Pokud by hlavním pozorovacím metrem nebyla úspěšnost modelu, ale kombinace úspěšnosti a spotřebované energie na trénování a počítání ostrých výsledků, nejlepší kombinovaný výsledek by vykazovala síť s jedním hodinovým modelem. Tato síť představuje poměrně kvalitní výsledky s minimálně poloviční spotřebovanou energii pro výpočet.

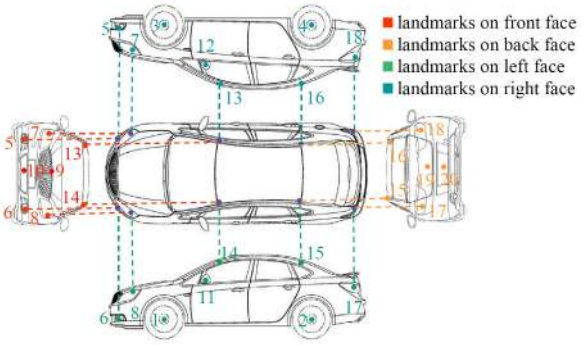

Figure 5: Grafické zobrazení jednotlivých *keypoint* bodů uvedených v tabulce 1.

Nižší zvolený počet kanálů, tedy 128 oproti 256, který byl použit při řešení úlohy určení pozice osob v [13] neměl na úspěšnost modelů zásadní vliv. Tato vlastnost může být ovšem mírně zavádějící vzhledem k poměrně vysoké toleranci zvolené při testování modelů, tedy 3.0, u obrázků velikosti $64x64$. Nicméně nízký vliv by mohl být způsoben také rozdílnou variabilitou vzhledu klíčových bodů u vozidel a osob, jelikož u vozidel je poměrně striktně definována geometrie vozu a tedy i vzájemná poloha klíčových bodů podléhá menším změnám, či změnám daným pouze úhlem pohledu či zkosením nebo škálováním obrazu. U osob je poloha jednotlivých klíčových bodů téměř libovolná kvůli různým pozicím ve kterých se osoby mohou nacházet a pro model je tak obtížnější osvojit si tyto závislosti. Pozitivní vliv má nižší počet kanálů také na rychlost výpočtu, jelikož se markantně sníží počet parametrů modelu.

Během trénování těchto modelů byla dále sledována úspěšnost určení pozice jednotlivých klíčových bodů. V tabulce 1 jsou kromě celkových nejlepších dosažených výsledků pro jednotlivé testované modely (tedy pro všechny klíčové body) uvedeny také, pro každý model, nejlepší dosažené

výsledky u jednotlivých klíčových bodů. V těchto výsledcích lze pozorovat, že úspěšnost u jednotlivých klíčových bodů se poměrně značně liší. Například pro keypoint s indexem 19, tedy logo v zadní části vozidla, je průměrná úspěšnost $99,55\%$, naopak u keypoint s indexem 16, tedy pravý horní roh střechy vozidla, je průměrná úspěšnost pouze 74.35%. Tyto rozdíly jsou nejspíše způsobeny různou variabilitou vzhledu a pozici klíčových bodů u různých typů vozidel.

Size	**1** (%)	**2** (%)	**3** (%)	**4** (%)
1	95,40	92,29	90,92	91,41
2	93,61	90,20	90,54	99,93
3	93,61	99,59	99,89	96,60
4	94,69	95,53	98,11	94,26
5	95,36	95,98	96,27	95,69
6	96,00	95,61	96,90	95,99
7	95,81	94,40	99,99	98,91
8	93,89	91,49	95,42	91,69
9	92,82	91,34	92,44	93,05
10	94,36	94,51	95,41	95,39
11	93,85	93,96	94,19	94,99
12	86,43	92,84	93,96	92,61
13	86,43	89,43	89,25	89,36
14	89,54	91,86	90,94	90,95
15	84,54	89,66	86,09	85,61
16	78,92	69,00	69,48	80,96
17	85,95	85,91	85,84	84,00
18	84,44	85,59	84,94	83,88
19	99,24	99,67	99,92	99,52
20	95,12	95,31	93,80	91,30
TOTAL	93,61	93,83	94,59	94,09

Table 1: Znázornění úspěšností modelů pro jednotlivé klíčové body.

5 Závěr

V rámci projektu do předmětu *konvoluční neuronové sítě* byla nastudována literatura na téma *Car keypoint detection*. Tato literatura byla shrnuta v úvodních kapitolách této zprávy. Vysoká pozornost byla věnována teoretické části popisující použitou technologii *stacked hourglass*. *Ground Truth* je implementována pomocí *heat* map. Tato architektura byla implementována v jazyce *Python 3* s využitím knihovny *PyTorch*. Trénovací, testovací a vyhodnocovací proces byl proveden v prostředí *Google Colaboratory*. V Experimentální části byly testovány čtyři různá nastavení parametrů předvedené architektury. Jejich vyhodnocení bylo provedeno iterativně po malém počtu epoch a následně byl vybrán zástupce za každý model, který dosahoval nejlepších výsledků. Na těchto modelech bylo provedeno podrobné ověření kvality, výsledky zaznamenává tabulka popisující úspěšnost pro každý testovaný model a také každý *keypoint* zvlášť. Z výsledků získaných při testování lze vidět, že na vzdory markantnímu rozdílu ve velikosti modelů jsou rozdíly v úspěšnosti velice malé. Nejlepší úspěšnosti bylo dosaženo u sítě se třemi zřetězenými hodinovými modely. Síť s pouze jedním hodinovým modelem, který je menší, rychlejší při vyhodnocování, a tak i méně náročný např. na spotřebovanou energii ale vykazuje téměř totožnou úspěšnost a tím pádem lze toto řešení považovat v tomto ohledu za optimální. Dále byla sledována úspěšnost při detekci jednotlivých klíčových bodů, která byla značně rozdílná a blíže popsána výše. Zajímavým poznatkem je fakt, že pro natrénování modelů k dosažení vysoké úspěšnosti bylo potřeba relativně málo trénovacích epoch.

References

[1] Wang, Zhongdao and Tang, Luming and Liu, Xihui and Yao, Zhuliang and Yi, Shuai and Shao, Jing and Yan, Junjie and Wang, Shengjin and Li, Hongsheng and Wang, Xiaogang.: Orientation Invariant Feature Embedding and Spatial Temporal Regularization for Vehicle Re-Identification. 2017.

[2] Ronneberger O., Fischer P., Brox T. (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab N., Hornegger J., Wells W., Frangi A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science, vol 9351. Springer, Cham

[3] Chabot, Florian & Chaouch, Mohamed & Rabarisoa, Jaonary & Teuliere, Celine & Chateau, Thierry. (2017). Deep MANTA: A Coarse-to-Fine Many-Task Network for Joint 2D and 3D Vehicle Analysis from Monocular Image. 1827-1836. 10.1109/CVPR.2017.198.

[4] Li, Qingnan & Hu, Ruimin & Chen, Yu & Chen, Yixin. (2019). Vehicle Pose Estimation Using Mask Matching. 1972-1976. 10.1109/ICASSP.2019.8683429.

[5] Y. Xiang, Wongun Choi, Y. Lin and S. Savarese. (2015). Data-driven 3D Voxel Patterns for object category recognition. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Boston. MA. pp. 1903-1911.

[6] Chen, Tao & Lu, Shijian & Fan, Jiayuan. (2017). S-CNN: Subcategory-Aware Convolutional Networks for Object Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence. PP. 1-1. 10.1109/TPAMI.2017.2756936.

[7] J. K. Murthy, G. V. S. Krishna, F. Chhaya and K. M. Krishna, "Reconstructing vehicles from a single image: Shape priors for road scene understanding," 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017, pp. 724-731.

[8] Liu, Xinchen & Liu, Wu & Mei, Tao & Ma, Huadong. (2016). A Deep Learning-Based Approach to Progressive Vehicle Re-identification for Urban Surveillance. Computer Vision – ECCV 2016. Springer International Publishing. 869-884. 10.1007/978-3-319-46475-6_53,

[9] Chen, Xiaozhi & Kundu, Kaustav & Zhang, Ziyu & Ma, Huimin & Fidler, Sanja & Urtasun, Raquel. (2016). Monocular 3D Object Detection for Autonomous Driving. 2147-2156. 10.1109/CVPR.2016.236.

[10] J. G. Lopez, A. Agudo, and F. Moreno-Noguer, "Vehicle pose estimation using g-net: Multi-class localization and depth estimation," in Artificial Intelligence Research and Development - Current Challenges, New Trends and Applications, CCIA 2018

[11] Karen Simonyan & Andrew Zisserman. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition.

[12] Newell, Alejandro & Yang, Kaiyu & Deng, Jia. (2016). Stacked Hourglass Networks for Human Pose Estimation. 9912. 483-499. 10.1007/978-3-319-46484-8_29.

[13] F. Moreno-Noguer, "3d human pose estimation from a single image via distance matrix regression," Computer Vision and Pattern Recognition (CVPR), 2017.

[14] Lopez, Javier & Agudo, Antonio & Moreno-Noguer, Francesc. (2019). Vehicle pose estimation via regression of semantic points of interest. 209-214. 10.1109/ISPA.2019.8868508.

[15] Ding, W., Li, S., Zhang, G., Lei, X., & Qian, H. (2018). Vehicle Pose and Shape Estimation Through Multiple Monocular Vision. 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO). 709-715.

[16] Laan, Chris,: Real-time 3D car pose estimation trained on synthetic data. 2019.

[17] S. Tulsiani and J. Malik, "Viewpoints and Keypoints," in Proc. 2015 IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR), Boston, USA, 2015, pp. 1510-1519

[18] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A simple yet effective baseline for 3d human pose estimation," in ICCV, 2017.

[19] Adrien Gaidon, Qiao Wang, Yohann Cabon, Eleonora Vig, "Virtual world as proxy for multi-object tracking analysis," Computer Vision and Pattern Recognition (CVPR), 2016.

[20] URL: https://github.com/princeton-vl/pytorch_stacked_hourglass