Sistem za analizo rečnega prometa z uporabo YOLO detektorja

Žan Besednjak in Luka Šajn

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
E-pošta: zb3383@student.fri-lj.si

Povzetek

Zaradi večanja količine turizma in njegovega vpliva na okolje se je pojavila potreba po analizi rečnega prometa in s tem problem implementacije sistema za zajem podatkov, namenjenim analizi rečnega turizma. Sistem sestoji iz detektorja objektov, algoritma za sledenje, števca in podatkovne baze. Pogledali si bomo zajem in obdelavo podatkov za učenje in testiranje sistema.

Abstract

Due to more and more tourism and its impact on the environment, a need for analyzing river traffic arose and with it a problem of implementing a system to acquire data, intended for analyzing river tourism. The system is composed from a detector, tracker, counter and a database. We will look at acquisition and processing of data for training and testing our system.

1 Uvod

Turizem se iz leta v leto veča. Naj bo to poletno kopanje v morju, gorski vzponi ali rečne avanture s kajakom in kanujem. Kljub temu, da rečni turizem odpira nove karierne možnosti, prinaša dobiček in je dober način za rekreacijo in sprostitev, prinaša tudi nekaj pomanjkljivosti - onesnaževanje, spreminjanje okolja in nevarnosti za neizkušene turiste.

V nadaljevanju je predstavljen sistem za štetje in razvrščanje plovil, ki bo omogočal analizo rečnega prometa - gostoto prometa ter distribucijo prometa v dnevu in tednu. Nameščen bo na sistemu z video kamero, ki bo v realnem času prepoznaval rečna plovila, jih kategoriziral in podatke posredoval na strežnik v oblaku. S tem bomo zagotovili popolno anonimnost vseh udeleženih v rečnem prometu ter se obenem izognili hranjenju velike količine videoposnetkov.

2 Sestava sistema

Da lahko analiziramo rečni promet potrebujemo število različnih plovil, ki potujejo po reki. Za naše potrebe bomo te ločili na kajake 1a in rafte 1b. Sistem bomo razdelili na tri dele - zajem, obdelava in hranjene podatkov. Naše zanimanje bo predvsem na drugem delu - obdelavi podatkov.

(a) razred kajak

(b) razred raft

Slika 1: Razreda plovil

2.1 Strojna oprema

Poleg programske opreme potrebujemo določiti strojno opremo, s katero bomo lahko podatke dovolj hitro obdelovali in bo obenem imela dovolj nizko energetsko porabo za možnost napajanja na sončno energijo. Želimo namreč, da bi naš sistem bil kar se da neodvisen od okolja in čim manj zahteven za vzdrževanje (glej sliko 2). Ker bo v našem sistemu potrebna obdelava slik, več o tem v nadaljevanju, se odločimo, da bo osnova za naš sistem naprava NVIDIA® Jetson Nano™, ki vsebuje grafični procesor s podporo CUDA®, kar nam zagotavlja podporo za večino odprto kodnih orodji. Komunikacija s podatkovno bazo bo potekala po mobilnem omrežju, s čimer zagotovimo popolno brezžično delovanje.

2.2 Zajem podatkov

Pri zajemu podatkov se moramo odločiti za vrsto le-teh. V našem primeru smo kot senzor za zajem izbrali video kamero. Poleg tega, da je marsikatero orodje za klasifikacijo izdelano na osnovi slik in video posnetkov, nam video kamera omogoča enostaven način za spremljanje razvoja in delovanje sistema.

Naši podatki morajo biti raznoliki, da zagotovimo čim večjo prilagodljivost sistema in se izognemo pretiranemu prilagajanju na učne podatke. Iz tega razloga za učenje uporabimo podatke pridobljene na različnih predelih reke in v različnih vremenskih pogojih.

Slika 2: Sistem nameščen ob reki.

2.3 Hranjenje podatkov

Preden določimo kako bomo zastavili podsistem za obdelavo si poglejmo kako bomo podatke hranili. Ker nimamo veliko prostora za hranjenje vseh podatkov in želimo imeti več naprav na različnih lokacijah bomo podatke iz naprav posredovali v bazo podatkov v oblaku. Kot že omenjeno, bomo za posredovanje podatkov uporabili mobilno podatkovno omrežje, ki nam tako omogoča čim večjo mero prilagodljivosti okolja. Za obliko in zapis podatkov si želimo tako predstavitev, da bo razvidno število, razred in časovna oznaka detekcije.

2.4 Obdelava podatkov

V prejšnjih dveh poglavjih smo določili vrsto vhodnih podatkov ter kakšne izhodne podatke pričakujemo. Ostane nam implementacija podsistema, ki bo podatke obdelal na tak način, da bo upošteval naše zahteve. Časovna oznaka je precej trivialna. V kolikor naš sistem prepozna enega izmed iskanih razredov mu pripišemo trenutni čas. Za detekcijo uporabimo model za prepoznavanje na osnovi računalniške vida, ker je vhod v sistem slika. Z detektorjem smo rešili drugo zahtevo - razred. Ker je rezultat detektorja poleg razreda tudi okvir, ki določa pozicijo detekcije na sliki, lahko uporabimo algoritem za sledenje, ki bo določil, če je bila detekcija na sliki že prepoznana in je objekt že prištet, ali je objekt nova instanca ter ga moramo prišteti.

Iz navedenega določimo, da so potrebne komponente podsistema detektor, algoritem za sledenje in števec.

3 Model za prepoznavanje

Glavni del programske opreme je model za prepoznavanje plovil. Prepoznava in klasifikacija objektov je ključna za naš sistem, a je računsko najzahtevnejši del. Zato moramo biti pri izbiri modela posebno pozorni. Trenutna tehnologija prepoznavanja temelji na nevronskih mrežah [5].

Detektorji na principu konvolucijskih nevronskih mrež se v grobem delijo na dve veji, dvostopenjski in enostopenjski. Dvostopenjski detektorji, kot že ime pove, delujejo v dveh korakih. Najprej z različnimi tehnikami (CNN [5], HOG [3], SIFT [9]) zaznajo verjetna območja v sliki in jih označijo. V drugem koraku detektor predloge potrdi ali zavrne. Enostopenjski detektorji za razliko v istem koraku lokalizirajo in klasificirajo objekte. Tak pristop je hitrejši, a manj zanesljiv.

3.1 YOLO

Družina YOLO (You Only Look Once) spada med enostopenjske detektorje. Njihovo delovanje je bazirano na razdelitvi slike v mrežo celic in izvajanju predikcije na vsaki celici. Detektorje v večini sestavljajo konvolucijski in popolnoma povezani nivoji. YOLO (od različice 2 naprej) za detekcijo uporablja sidrane škatle, kar omogoči natančnejšo lokalizacijo prepoznanih objektov. Uporabili bomo različico YOLOv5 [7], ki sicer ni več med novejšimi, a je zaradi enostavne vključitve v Python okolje in manjše porabe virov idealna rešitev.

YOLOv5 je razdeljen na več različnih velikosti, ki se razlikujejo v številu nivojev in parametrov. Sestoji iz hrbtenice DarkNet53, vratu s SSP (Spatial Pyramid Pooling) in PANet (Path Aggregation Network) ter enake glave kot je v YOLOv4 detektorju [1]. Novosti in razlike YOLOv5 detektorja od prejšnjih različic so fokusni nivo (the focus layer), izločevanje občutljivosti mreže (Eliminating Grid Sensitivity) in Pytorch ogrodje.

4 Algoritem za sledenje

Naš detektor določi razred in lokacijo zaznanega objekta, a ne hrani informacij prejšnjih detekcij in jih med seboj ne povezuje. Zato bomo uporabili algoritem za sledenje, ki nam bo omogočil, da povežemo detekcije med zaporednimi slikami. Za to nalogo bomo uporabili algoritem za sledenje več objektom (MOT). Za evaluacijo algoritma se uporablja več različnih primerjalnih preizkusov. Po pregledu lestvice MOT20 [4] se odločimo, da bomo za naš algoritem izbrali sledilnik ByteTrack, predvsem zaradi hitrosti algoritma, ki je za naše namene in strojno opremo izjemno pomembna.

4.1 ByteTrack

ByteTrack [11] je algoritem za sledenje na osnovi zaznanih okvirjev. Za razliko od večine drugih sledilnikov ByteTrack ne zavrže detekcij, ki imajo nizko verjetnost.

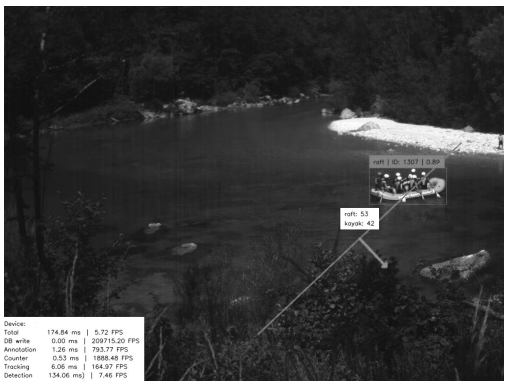
Algoritem za vsako sliko iz videa in pripadajočimi okvirji detekcij najprej predpostavi položaj za vsako sled

s Kalmanovim filtrom [10]. Nato poveže detekcije z visoko zanesljivostjo s pripadajočimi sledmi. Enako stori s preostalimi detekcijami z manjšo zanesljivostjo ter preostalimi sledmi. Na koncu pobriše preostale sledi in ustvari morebitne nove sledi iz detekcij, ki jih nismo povezali s prejšnjimi sledmi. Izhod sledilnika so objekti, katerih parametri so, tako kot prej, pozicija in velikost okvirja, razred in indeks sledi.

5 Štetje objektov

Zadnji korak, ki ga moramo narediti je preštevanje objektov. Postavili bomo navidezno črto, ki se razteza med rečnima bregovoma. Črto predstavimo z vektorjem, kar nam omogoči, da z množenjem tega vektorja in vektorja, ki poteka od začetka črte do našega objekta, določimo na kateri strani črte je naš objekt. Kar moramo nato storiti je, da za vsako sled hranimo dve zastavici, ki nam pomagata določiti, če je objekt na obeh straneh črte. Ko sta obe zastavici neke sledi označeni, objekt prištejemo.

Prištetemu objektu dodamo časovno značko ter ga shranimo v podatkovno bazo (glej sliko 3).



Slika 3: Prikaz delovanja programske opreme.

6 Podatki

Ko imamo izbran model, ga moramo naučiti prepoznavati želene objekte. Detektor se bo učil na slikah, ki jim bomo označili instance objektov ter pripadajoči razred.

6.1 Predpriprava podatkov

Zaradi velikega števila slik, ki jih dobimo z zajemom videa si želimo olajšati in poenostaviti anotiranje. Zato najprej izločimo vse podvojene zaporedne slike. S tem ne zmanjšamo le časa, ki ga potrebujemo za anotiranje podatkov, temveč tudi zmanjšamo prostor za shranjevanje in odstranimo večino irelevantnih podatkov - slik brez plovil.

Iz istega razloga uporabimo algoritem za prepoznavanje ozadja (podpoglavje 6.4.2), ki vse slike, ki vsebujejo le ozadje odstrani in dodatno zmanjša število slik za označevanje.

6.2 Označevanje

Za učenje detektorja potrebujemo zapis, ki bo predstavil objekte na slikah in lokacijo le-teh. Zapis bo v obliki 'C x y w h', kjer je C razred objekta, x in y koordinati okvirja zgornje leve točke okvirja objekta, w in h pa širina in višina. Orodji za označevanje podatkov je veliko, a nam njihovo označevanje kljub temu vzame veliko časa, saj moramo vsaki instanci določiti okvir in razred. Da bi si olajšali delo si zamislimo orodje za pohitritev tega postopka.

6.3 Avtomatsko označevanje

Označevanje velike količine podatkov je časovno dolgotrajen proces. Da bi si olajšali delo si poglejmo, kako lahko proces avtomatiziramo.

V osnovi je problem avtomatskega označevanja podoben osnovnemu problemu - prepoznavanju in lokaliziranju objektov na sliki. Ob predpostavki, da imamo model za prepoznavo, bi ga lahko uporabili za avtomatsko označevanje. Po označevanju bi imeli večjo množico podatkov na kateri bi ponovno učili model ter tega ponovno uporabili na novih podatkih. S tem dobimo cikel učenja modela na vedno večji množici podatkov.

Na naš model se seveda ne moremo zanesti, da bo vedno pravilno prepoznal in klasificiral objekt, saj bi ob morebitni napaki te postajale vedno večje in bi najverjetneje obtičali v lokalnem ekstremu. Da bi to preprečili bomo avtomatsko klasificirali le, če je detektor dovolj prepričan v pravilnost razreda. V nasprotnem primeru bomo zahtevali ročni vnos oziroma ročno potrjevanje razreda. Kljub ročnemu vnosu razreda smo objekt avtomatsko lokalizirali in mu določili okvir.

6.4 Prepoznavanje brez detektorja

Kljub predlagani rešitvi vseeno ostaja problem, kako naučiti začetni model. Začetno klasifikacijo bomo prepustili nam, a bi vseeno želeli avtomatsko zaznati objekte in jim določiti okvir.

Najprej lahko poskusimo s preprosto rešitvijo in poiščemo razlike med dvema zaporednima slikama. Ker se naša snemalna naprava ne giblje, imamo mirujoče ozadje. To pomeni, da bodo razlike v slikah predvsem gibajoči se objekti. Od tu naprej lahko z morfološkimi operacijami, odpiranjem in zapiranjem, odstranimo šum in zapolnimo luknje ciljnih polj. Pridobljena polja premajhne velikosti lahko sedaj odstranimo in ostalim očrtamo okvir. Rezultat je okvir, ki sovpada z zapisom, ki ga potrebuje naš model za prepoznavo.

Osnovni ideji dodamo tudi naslednje razširitve.

6.4.1 Maska

Prvi dodatek, našemu algoritmu za zaznavanje objektov je preprosta maska, ki izloči vse dele slike, ki jih nočemo uporabiti v našem modelu. V to so vključeni breg, nebo in okolica.

Masko lahko uporabimo tudi za rešitev z detektorjem. V primeru, da detektor napačno prepozna okolico kot enega izmed ciljnih razredov lahko to detekcijo odstranimo.

6.4.2 Odstranjevanje ozadja

Pomembno je seveda rešiti problem, kako razpoznati objekt v primeru, da premik na dveh zaporednih slikah ni dovolj zaznaven. Uporabili bomo podoben pristop kot [6].

Prvi korak je ekstrakcija ozadja. Rezultat, ki ga želimo je slika, ki bi jo uporabili namesto predhodne slike v videu. Ker so naši podatki zajeti od jutranjih do večernih ur želimo, da se model ozadja prilagaja na spremembe.

Za začetni model B_0 vzamemo prvo sliko videa. Nato za vsako novo sliko I_t izračunamo novi model B_t po sledeči formuli 1.

$$B_t = B_{t-1} + \frac{1}{t} * (I_t - B_{t-1})$$

$$\begin{aligned} &B_t : trenutni\ model\ ozadja \\ &B_{t-1} : prejšnji\ model\ ozadja \\ &I_t : trenutna\ slika \end{aligned} \tag{1}$$

Sedaj lahko namesto primerjanja slike s prejšnjo $I_t - I_{t-1}$, uporabimo izračunani model tako, da dobimo slikovne točke, ki so nam pomembne. Te označimo z belo barvo - 255, ostali del slike pa s črno - 0 2.

$$d(x,y) = \begin{cases} 255, & I_t - B_t > \Delta t \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Prag Δt določimo dinamično po formuli 3. Z dinamičnim pragom omogočimo prilagajanje na globalne spremembe v sliki in izboljšamo ekstrakcijo.

$$\Delta t = \lambda * (\frac{1}{N} * M) * \sum_{x=1}^{N} \sum_{y=1}^{M} |I_t(x,y) - B_t(x,y)|$$

$$\begin{aligned} &\lambda : koeficient \\ &N, M : velikost\ slike \\ &B_t : model\ ozadja \\ &I_t : trenutna\ slika \end{aligned} \tag{3}$$

Ko imamo slikovne točke zanimanja, ponovno uporabimo operacije morfologije, da zapolnimo luknje in odstranimo šum. Nato združimo povezane slikovne točke v skupine in ohranimo le dovolj velike, ki bi lahko predstavljale iskane objekte. Določimo jim okvir in s tem pridobimo želeni rezultat, koordinate in velikost objektov.

Dodatno s tem rešimo problem odstranjevanja zaporednih slik. Zaradi narave tekoče vode (odsevi, valovanje ipd.) se namreč dve zaporedni sliki lahko razlikujeta preveč, da bi jih odstranili.

Največja slabost takega pristopa je, da v primeru prekrivanja označimo dve instanci objekta kot eno samo.

6.4.3 Sledenje

Dodatno bomo uporabili področno sledenje (regional tracking) po podobnosti [2]. To nam omogoči, da razred objekta določimo le enkrat. Nato primerjamo lokacije objektov, ki jih dobimo s sledenjem in lokacije zaznanih objektov, ki jih določimo z odstranjevanjem ozadja ter ohranimo razred, ki smo ga določili na začetku.

7 Testiranje učinkovitosti

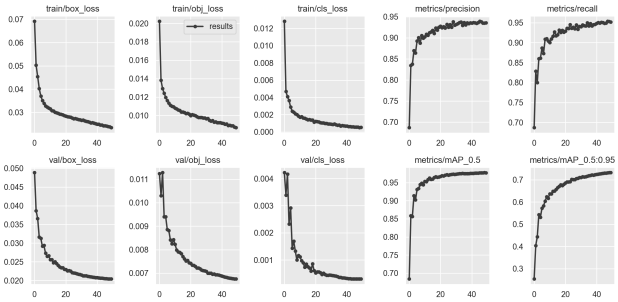
Za testiranje pripravljenega modela potrebujemo podatke, ki jih nismo uporabili za učenje, zato pred pričetkom učenja pripravimo testno množico, ki je ločena od ostalih podatkov.

Po 50 iteracijah učenja nevronske mreže na 30000 instancah objektov dobimo sledeče rezultate. Po metriki mAP_0.5 je zanesljivost našega detektorja 0.9771, po metriki mAP_0.5:0.95 pa 0.7298 1 4. Ostale metrike opisujejo prileganje okvirjev (val/box_loss), prepričanost v obstoj detekcije (val/obj_loss), pravilnost detekcije (val/cls_loss), natančnost (precision) in občutljivost (recall).

Na podlagi rezultatov je naš model boljši v primerjavi z uradnimi COCO [8] rezultati [7], a to je pričakovano, saj imamo v naši množici podatkov le dva razreda v primerjavi s COCO, ki vsebuje 91 različnih razredov.

Tabela 1: Rezultati

metrika	rezultat
box loss	0.0238
object loss	0.0087
class loss	0.0005
precision	0.93533
recall	0.95283
mAP_0.5	0.97758
mAP_0.5:0.95	0.73211



Slika 4: Grafični prikaz rezultatov 50 iteraciji učenja

8 Zaključek

V članku je predstavljeno delovanje in implementacija sistema za analizo rečnega prometa. Razložen je bil način prepoznavanja, sledenja in preštevanja plovil, določena je bila strojna oprema in omejitve, ki nam jih ta prinaša. Predvsem smo se osredotočili na postopek avtomatskega označevanja, ki nam omogoči, da minimiziramo trud priprave učne množice podatkov in za konec analizirali rezultate sistema.

Literatura

[1] Alexey Bochkovskiy, Chien-Yao Wang in Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. DOI: 10.48550/ARXIV.2004.10934. URL: https://arxiv.org/abs/2004.10934.

[2] Fei Chen in sod. "Visual object tracking: A survey". V: *Computer Vision and Image Understanding* 222 (2022), str. 103508. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2022.103508. URL: https://www.sciencedirect.com/science/article/pii/S1077314222001011.

[3] N. Dalal in B. Triggs. "Histograms of oriented gradients for human detection". V: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Zv. 1. 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[4] Patrick Dendorfer in sod. *MOT20: A benchmark for multi object tracking in crowded scenes*. 2020. arXiv: 2003.09003 [cs.CV].

[5] Ross Girshick in sod. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". V: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Jun. 2014.

[6] Larbi Guezouli in Hanane Belhani. "Automatic detection of moving objects in video surveillance". V: *2016 Global Summit on Computer & Information Technology (GSCIT)*. IEEE. 2016, str. 70–75.

[7] Glenn Jocher. *ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation — Zenodo*. hhttps://zenodo.org/record/7347926. 2022.

[8] Tsung-Yi Lin in sod. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[9] David G Lowe. "Distinctive image features from scale-invariant keypoints". V: *International journal of computer vision* 60 (2004), str. 91–110.

[10] Greg Welch, Gary Bishop in sod. "An introduction to the Kalman filter". V: (1995).

[11] Yifu Zhang in sod. *ByteTrack: Multi-Object Tracking by Associating Every Detection Box*. 2022. arXiv: 2110.06864 [cs.CV].