Možnosti ML na osnih signalih

Na zadnjem sestanku smo debatirali o možnosti ML na osnih signalih. Za detekcijo skupin itak ni težav, problem je identičen tistemu s CV, tudi vhodne podatke imamo vse. Vsekakor se ta del splača poskusiti.

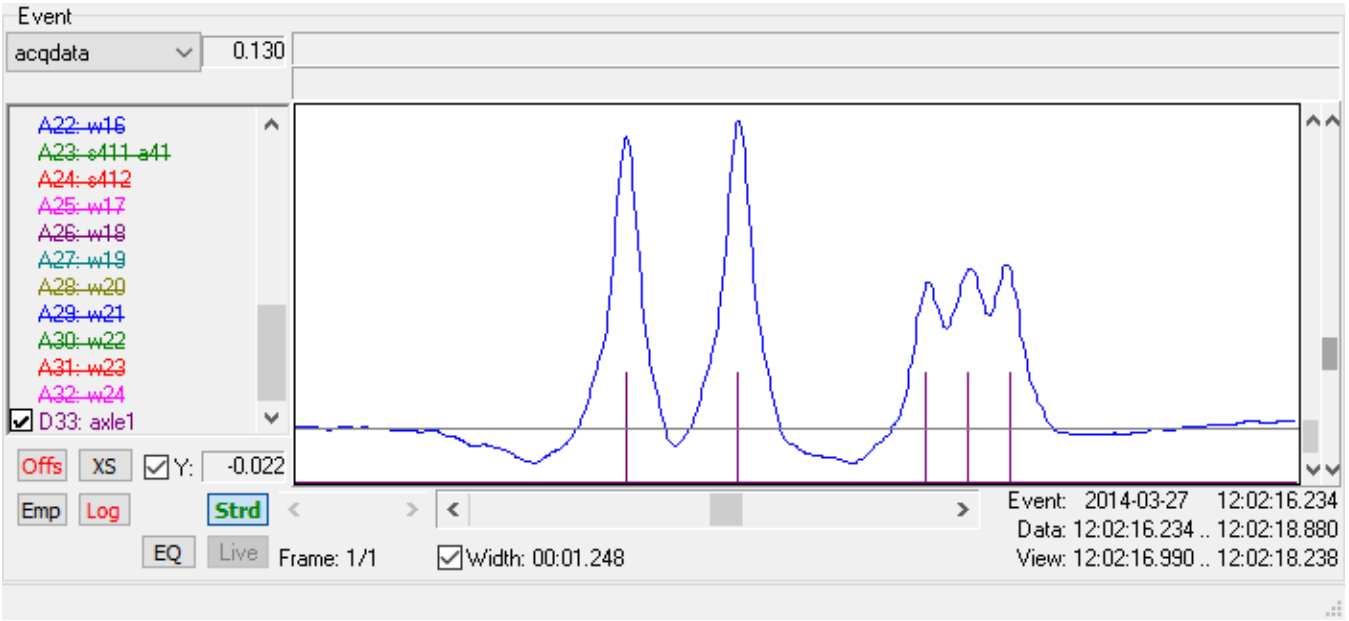
Sledi pa detajlna razlaga podatkov, ki jih imamo na voljo, ter predlog kako bi se z ML vseeno dalo določati tudi pozicije osi.

Osnovni podatki

SiWIM sistem da od sebe t.i. event-e E, kose podatkov, ki vsebujejo zajete signale, diagnostike generirane z različnimi procesnimi moduli ter podatki o vozilih. Simbolično bi relevantno informacijo v event-u zapisal kot $E = E(s, p, x)$, kjer so:

- s signal iz detektorjev osi. To je analogen signali, vzorčeni s 512 vzorci na sekundo.
- p pulzi (palčke), ki jih SiWIM detektira iz signala s. To je digitalen kanal, ki je interno predstavljen s spiskom parov, kjer posamezen par predstavlja številko vzorca začetka in konca pulza. Če je vozilo pravilno detektirano, je število pulzov enako številu osi.
- V event-u obstajata dva spiska vozil. Prvi je `detected_vehicles`, ki jih producira modul `vehicle_fad`. Modul `weigh` vzame vozila iz `detected_vehicles` ter tehtalne signale in z Moses-ovim algoritmom izračuna surove teže. Tako opremljena vozila shrani v `weighed_vehicles`. Vsako vozilo v spisku ima shranjeno spisek medosnih razdalj x.

Primera signalov `s` (z modro) in `p` (z vijolično) sta na sledeči sliki:



To ustreza vlačilcu s skupinami osi 113 in z medosnimi razdaljami $[3.5, 5.8, 1.4, 1.3]$ (v metrih).

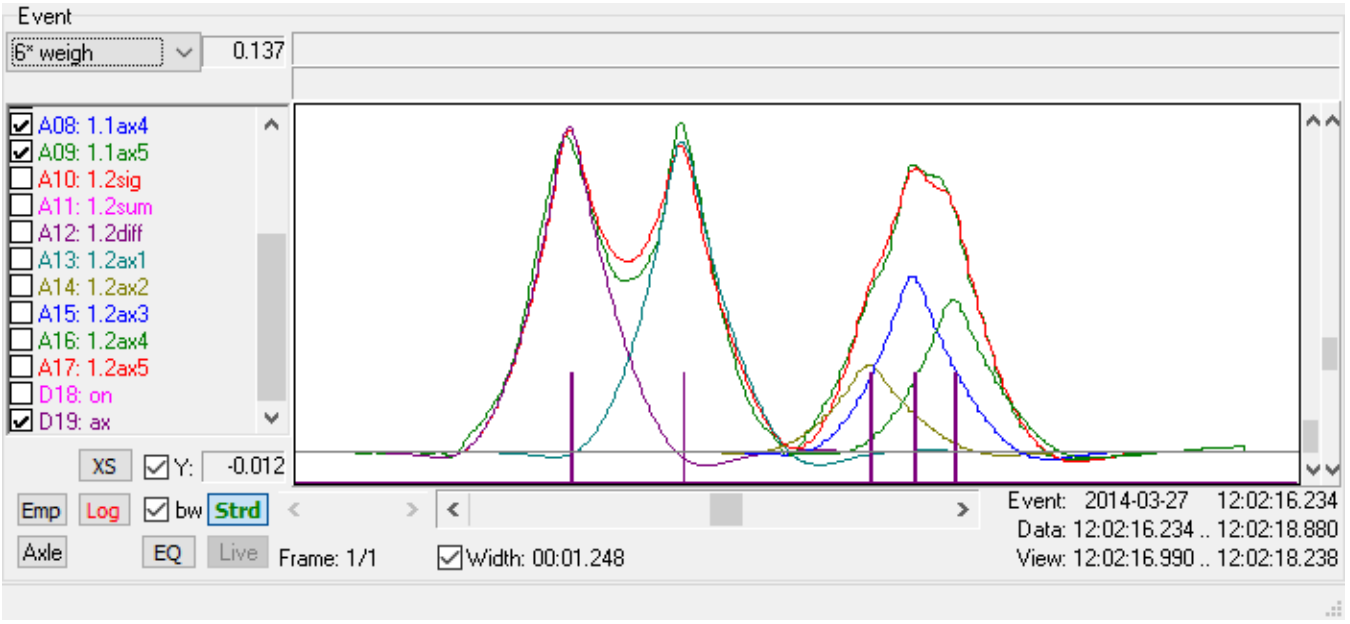
Obdelava

Obdelava od detektiranih vozil do končnih podatkov poteka v več korakih:

Neposredno tehtanje vozil

Prvi korak je neposredno tehtanje vozil. To se izvede tako, da se na merjeni signal $g(t_j); j = 1 \dots M$, kjer je M število vzorcev, prilagaja funkcijo $f(t_j) = \sum_{i=1}^{N} A_i I(v_i(t_j - t_i))$, kjer je N število osi, A_i posamezni osni pritisk, $I()$ vplivnica, v_i hitrost i-te osi in t_i prihod i-te osi v izhodišče. Z variiranjem A_i se minimizira $\chi^2 = \sum_{j=1}^{M} (g(t_j) - f(t_j))^2$ in na ta način izračuna osne pritiske.

Pri tehtanju se generira tudi diagnostika za tehtanje. Primer diagnostike za isti event, kot na prejšnji sliki, je:



Pulzi pri pozicijah osi so vijolični, merjeni signal $g(t)$ je zelen, prilagajani signal $f(t)$ rdeč. Ostalih 5 signalov so vplivnice na pravih mestih, pomnožene s posameznim osnim pritiskom — vsota teh signalov je $f(t)$.

Simbolično lahko napišemo da je tehtanje preslikava $W : E(s, p, x) \mapsto E(s, p, q, x)$, kjer smo s q označili pulze za osi uporabljene za tehtanje. Pri osnovnem tehtanju je $q = p$.

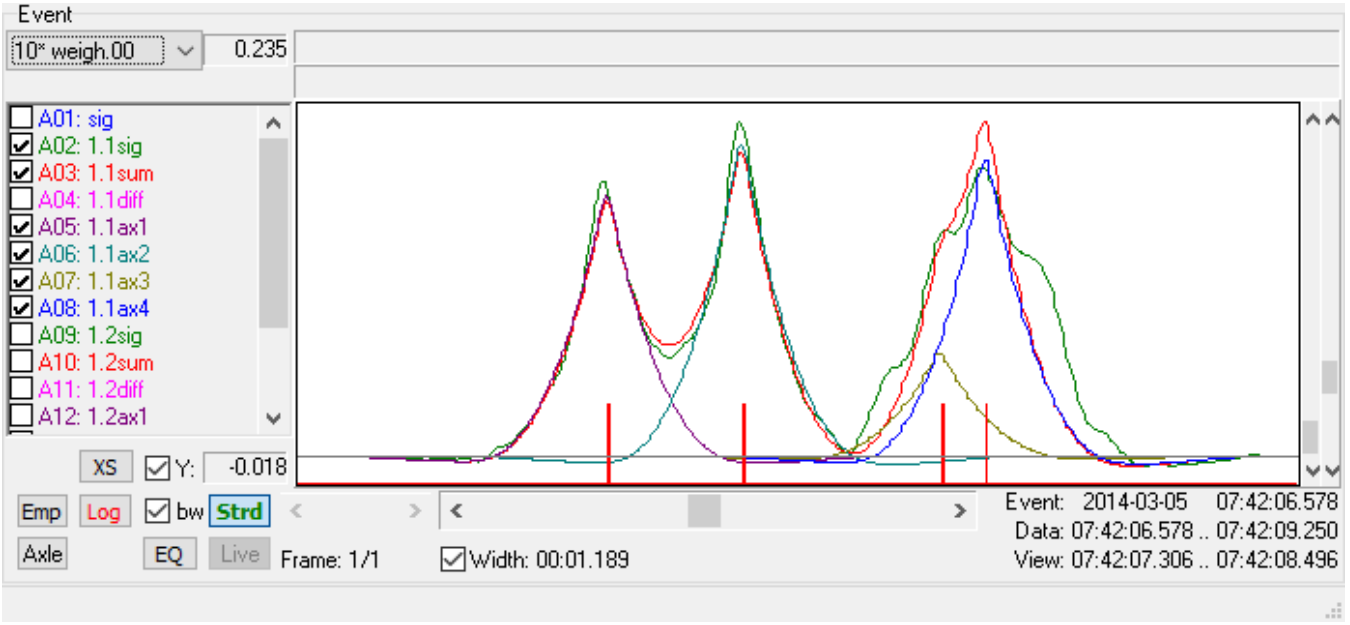
Rekonstrukcija vozil

Ideja je sledeča:

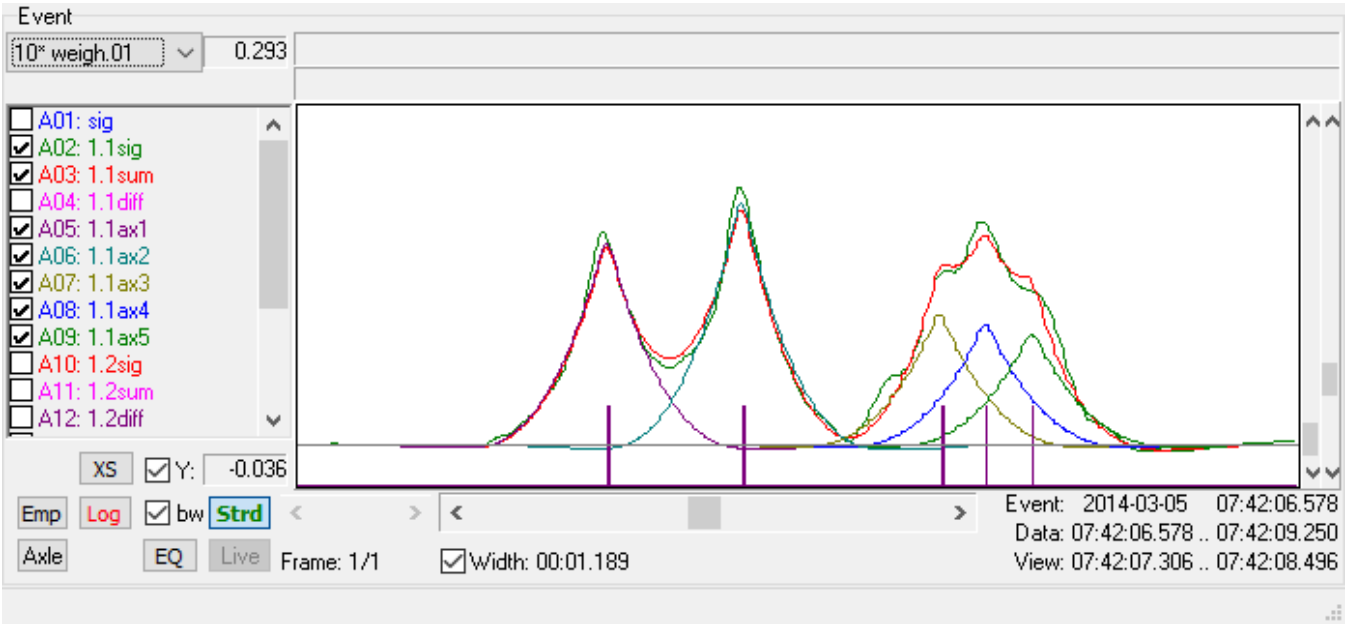
1. Izračunajo se osni pritiski z detektiranimi osmi in zabeleži χ_0^2.
2. Osne razdalje se primerja s spiskom pravil. Vsako pravilo vsebuje skupine osi ter minimalne in maksimalne medosne razdalje. Če bi vozilo, na primer, imelo namesto skupin 113 skupine 112 in bi medosne razdalje bile znotraj meja, bi se dodalo eno os na razdalji $1.35m$ pred in/ali tretjo osjo (odvisno od konkretnih razdalj), in tako "rekonstruirano" vozilo zopet stehtalo.
3. Tako dobljen χ_1^2 se primerja s χ_0^2 in, v kolikor je je χ_1^2 manjši od χ_0^2 in je relativna razlika večja od neke pred-določene vrednosti (po navadi 5%), se to rekonstruirano vozilo vzame kot novega kandidata za rekonstruirano vozilo.
4. Korake 2.-3. se ponavlja dokler vozilo še ustreza kakšnemu pravilu.
5. Kandidata pri katerem se χ^2 najbolj zmanjša, se vzame kot rekonstruirano vozilo.

Rekonstruiranim vozilom se nastavi zastavica `Flag_Vehicle_Reconstructed`.

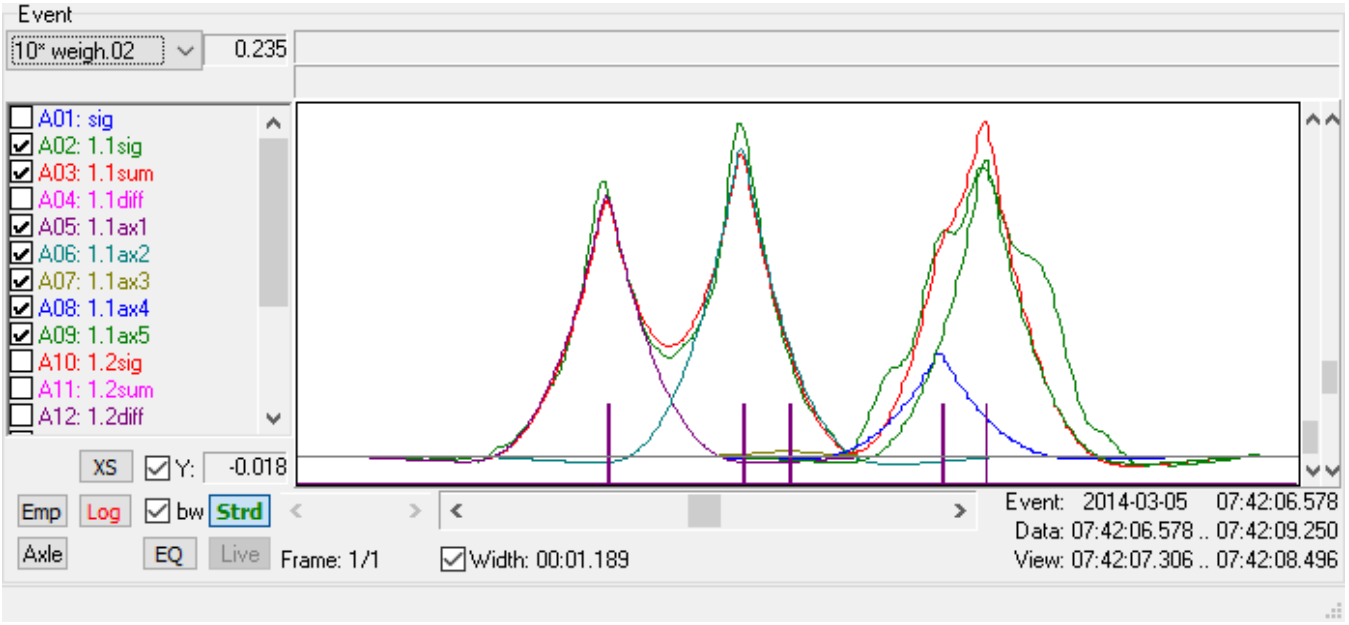
Primer rekonstrukcije je na naslednjih slikah. Na prvi je prikazano tehtanje z detektiranimi osmi (narisanimi z rdečo).



Vidi se, da je prvi del (vlačilec) v redu, pri polpriklopniku pa se prilagajana funkcija ne ujema najbolje. Na naslednji sliki je prvi poskus rekonstrukcije, pri katerem se doda zadnjo manjkajočo os v skupini. Ujemanje je že na oko precej boljše.



Nazadnje je poskus, pri katerem se drugi osi doda se eno os v skupino. To os je bila pri tehtanju izračunana skoraj 0, pri skupini pa je še vedno slabo ujemanje.



Ker je bilo zmanjšanje χ^2 največje pri pri prvem poskusu, je ta poskus tudi obveljal — končno vozilo ima skupine 113.

Če gledamo na rekonstrukcijo kot na preslikavo R, lahko napišemo $R : E(s, p, q, x) \mapsto E(s, p, q', x')$, pri čemer velja $x = x'$ in $q = q'$, če ne pride do rekonstrukcije. V nasprotnem primeru pa q' vsebuje nove pulze, x' pa nove medosne razdalje. V tem konkretnem primeru se število pulzov spremeni iz 4 v 5, $x = [3.9, 5.8, 1.3]$ pa v $x' = [3.9, 5.8, 1.3, 1.4]$.

Skripta QA

Čez tako dobljene podatke se zapelje Python skripta `qa.py`. Na osnovi hevrističnih pravil, nekaj povzetih iz projekta *BridgeMon*, nekaj pa dodanih z Alešem, se vsakemu vozilu nastavi ena izmed zastavic: zelena, oranžna ali rdeča. Zelena vozila so skoraj zagotovo OK, rdeča je zagotovo treba preveriti, oranžna pa so nekje vmes.

Signali in medosne razdalje se v tem koraku ne spreminjajo.

Strojni popravki

Tako stehtana vozila gredo skozi Python skripto `fix.py`, ki na podlagi hevrističnih pravil popravi vozila. Eden izmed vhodnih podatkov je tudi zastavica iz koraka QA. Simbolično lahko napišemo $F : E(s, p, q', x') \mapsto E(s, p, q', x'')$, kjer se x' in x'' lahko razlikujeta, če pride do popravka v lokacijah osi vozila. Lahko pa se, npr., spremeni samo kak osni pritisk, kar nas tukaj ne zanima — v tem primeru sta x' in x'' enaka.

Žal pa se q', tudi v primeru popravka medosnih razdalj, ne spremeni. V času, ko je bila pisana skripta, še ni bila napisana knjižnica *SiWIM-Pi*, s katero je moč brati, pisati in popravljati tudi event-e. Vse spremembe so se dogajale samo na tekstovnih NSWD datotekah, niso pa se mogle propagirati nazaj v event-e.

Popravljenim vozilom se nastavi zastavica `Flag_QA_Fixed`,

Ročni popravki

Nazadnje so se za Šentvid vozila pregledala in potrebi ročno obdelala s programom *SiWIM-D*. Tam je možno premikati osi levo, desno, gor, dol,... vozila združiti, ločiti,.., Pregledana niso bila vsa vozila, temveč tista najbolj kritična — z rdečo zastavico.

Simbolično $H : E(s, p, q', x'') \mapsto E(s, p, q''', x''')$. Spremenjenim vozilom se lahko nastavi ena ali več zastavic izmed `Flag_Classification_Manually_Changed`, `Flag_First_Axle_Position_Manually_Adjusted` in `Flag_Axles_And_Loads_Manually_Adjusted`.[1]

V primeru ročnih popravkov se tudi lokacije osi spremenijo iz q' v q'''. Vendar, kot sem že kakšno leto nazaj razložil, je pri premetavanju in shranjevanju podatkov prišlo do napake in event-i so izgubljeni. Zato je največ, kar imamo na Šentvidu, $E(s, p, q', x''')$, se pravi potencialno rekonstruirani osni pulzi iz koraka rekonstrukcije, q', brez informacije o strojnih in ročnih popravkih, ter strojno in ročno popravljene medosne razdalje.

Tudi če bi imeli shranjene signale q''', nam to samo po sebi ne bi pomagalo, saj nismo pregledali vseh vozil. Pa tudi, če je bilo vozilo videno, ne pa popravljeno, se to ni nikjer zabeležilo. To pomeni, da bi zagotovo vedeli le, če je vozilo bilo popravljeno. Če pa ne bi bilo označeno, bi to lahko pomenilo, da je bilo videno in je OK, ali pa da ni bilo videno in je napačno.

Ročno preverjanje

Nazadnje smo za pripravo ground truth CV ML podatkov iz množice vseh primernih meritev (shranjenih v datoteki `recognized_vehicles-ORIGINAL.json`, ki jo je zgeneriral Domen) izbrali samo tista vozila, ki so na pasu 1. Potem smo s pomočjo GUI skripte `label_braid_photos.py` pregledali precej fotografij, jih preverili in primerno označili.

Označena vozila lahko razdelimo v množici pravilno in nepravilno detektiranih vozil. Če je $x = x'''$, je vozilo v $\mathbb{P}$, drugače je v $\mathbb{N}$. Iz tega je nastala datoteka `metadata.hdf5`, ki jo je Domen uporabil kot vhod za treniranje NN, skupaj s fotografijami vozil.

Rekonstrukcija signala q'''

Če sem prav razumel, je problem v tem, da se pri vseh teh preslikavah transformira samo $x \mapsto x'''$, s čemer dobimo prave medosne razdalje, nimamo pa tudi $q \mapsto q'''$. Zato se ne da trenirati NN, da bi znal iz s predvideti q, Učil bi se samo na pravilno detektiranih primerih, ne pa tudi kako iz "zoprnih" signalov dobiti prave osi. Ampak mislim, da bi se dalo ta problem zlahka zaobiti. Za vozila iz množice $\mathbb{P}$ je zadeva preprosta, q''' je kar q, oziroma p.

Za vozila iz množice $\mathbb{N}$ pa bi bilo treba rekonstruirati q'''. Za to pa pride v poštev Andrejeva ideja z uporabo, npr., prve medosne razdalje kot referenco. Verjetno nikoli (ali pa vsaj v samo izjemno majhnem deležu primerov) namreč niso spremenjene prav vse medosne razdalje. Med medosnimi razdaljami x in x''' bi se poiskalo pare enakih medosnih razdalj in s pomočjo teh (ali verjetno najdaljše ustrezajoče izmed teh) določilo skalo signala q. S tem in z x''' pa so dani vsi podatki, s katerimi bi se dalo rekonstruirati signal q''', ki bi bil, poleg signala s, vhod v NN.

Tako imamo, PMSM, vse potrebne podatke, da bi iz signala lahko dobili pulze za določanje pozicij osi in s tem za boljše tehtanje.

Izbira podatkov

Najprej je izmed vseh podatkov o vozilih treba izbrati primerne za nadaljnjo obdelavo. Iz datotek `braid.nswd` v direktorijih `rp01` (rezultati neposrednega tehtanja z vklopljeno rekonstrukcijo) in `rp03` (rezultati popravljeni s `fix.py` in ročnimi popravki) smo izbrali tiste, ki so v obeh datotekah. To je nujno, ker strojni in ročni popravki včasih razdelijo vozila na eno ali več vozil ali združijo dve vozili v eno. Število takšnih vozil je 251328.

Od teh je treba izbrati samo tista vozila, ki so v datoteki `metadata.hdf5` (saj smo iz originalnih `braid.nswd` datotek vzeli le podmnožico vozil — glej lbp.pdf za detajle) skupaj 175926 vozil. Izmed teh vozil smo izločili tiste, ki:

- niso bili ročno preverjeni (polje `seen_by` je prazno) in
- pri katerih polje `axle_groups` *ni* prazno (število osi se ne ujema s sliko)

S takšno izbiro nam ostane množica 56224 vozil, ki:

- So šla skozi celotno procesno verigo, vključno z ročnimi popravki in
- se število osi dobljeno z rekonstrukcijo ter strojnimi in ročni popravki ujema s številom osi na sliki

s tem pa imamo, v danih okoliščinah, najboljši možen nabor za treniranje NN.

Vmesne datoteke

Rezultat izbire podatkov so štiri datoteke, `vehicles_for_axles-<KEY>.json`, kjer vrednosti `<KEY>` pomenijo:

asdf

: to je to

---

1. Pri pripravi datoteke `grp_and_fixed.hdf5`, poslane 3.12.2024, o kateri smo tudi včeraj govorili, sem pozabil na te zastavice. Bom, ko vzpostavim ML PC doma, še enkrat vse skupaj zgeneriral. Vidim, da sem tudi v `metadata.hdf` po nepotrebnem združil zastavice za ročne spremembe z zastavico `Flag_QA_Fixed`. Bom tudi to popravil, da bosta to ločena podatka. ↩