

# FNavodila za uporabo LBP (Label Braid Photos)

---

Jan Kalin [jan.kalin@zag.si](mailto:jan.kalin@zag.si)

## Zgodovina izdaj aplikacije in dokumentacije

v1.13.2, 4. junij 2024

- Dodan je razdelek [Izbira časovnega intervala](#)

v1.13.1, 3. junij 2024

- Dodan je razdelek [Opis podatkov v datoteki metadata.hdf](#)

v1.13, 28. maj 2024

- Dodana oznaka [YOLO Error](#)
- Oznaka *Več vozil* je prestavljena med [WIM oznake](#)

v1.12, 22. maj 2024

- Dodan [Auto Brightness %](#) pri nalaganju slik
- Dodana je možnost [Expected Vehicle Type](#) v okvirčku *Select vehicle groups for labelling*
- Oznaka [Cannot Label](#) se sedaj imenuje *Cannot label / not found*
- Dodane bližnjice za [dvignjene osi](#)
- Spremenjena je postavitev kontrol za označevanje tako, da so najpogosteje uporabljene čim bližje sliki

v1.11, 21. maj 2024

- Poleg timestamp-a vozila se izpiše tudi timestamp slike, oboje v bolj prijaznem formatu

v1.10, 20. maj 2024

- Dodan [Auto Contrast %](#) pri nalaganju slik

v1.9, 16. maj 2024

- [Dvoklik na sliki](#) poveča kontrast ali svetlost za 125%

v1.8.1, 15. maj 2024

- Dodana so navodila za [lokalno zaganjanje aplikacije](#)

v1.8, 15. maj 2024

- Dodana sta preprosta [popravka slik](#)
- Dodana je možnost za [zaklep datoteke metadata.hdf5](#)

v1.7, 14. maj 2024

- Dodane so [smernice za označevanje](#)
- Uvedena je oznaka *Multiple vehicles*, opisana je pri [splošnih oznakah](#)
- Dodani sta oznaki *Reconstructed* in *Fixed* ter opis le-teh v razdelku [rekonstruirane in popravljene osi](#)

- Gumb za [hitro izbiro fotografije](#) je odstranjen. Namesto tega se sliko izbere s tipko `<Enter>` v vnosnem polju
- Bugfix: pri praznem seznamu medosnih razdalj (posledica diskrepance med NSWID in EVENT datotekama zaradi popravkov osi) je prišlo do prekinitve izvajanja aplikacije

v1.6, 14. maj 2024

- Dodana funkcija [Zoom](#)

v1.5, 13. maj 2024

- Dodano vnosno polje *Jump to photo* za hitro izbiro fotografije

v1.4, 10. maj 2024

- Premešan interface, sedaj se slika veča z oknom

v1.3, 27. april 2024

- Predelava za novo označevanje (brez grupiranja v avtobuse in tovornjake)
- Dodan tip `other`, oznake barv, komentar,....

v1.2, 29. marec 2024

- Dodana razlaga ADMP/CF v razdelku ADMPs

v1.1, 27. marec 2024

- Velike množice vozil (npr., tovornjaki s skupinam 113) so razdeljene na podmnožice moči 1000
- Bližnjice so brez `<Alt>`
- Nov način opisovanja dvignjenih osi
- Dodana oznaka **Nekonsistentni Podatki**

v1.0, 20. marec 2024

- Začetna izdaja

## Uvod

---

Ta dokument opisuje uporabo vizualne aplikacije Label Braid Photos, ki se uporablja za označevanje fotografij vozil stehanih z SiWIM B-WIM. Namen aplikacije je označiti napačno označene fotografije, skupaj z napačno detektiranimi vozili, da se zgenerira "ground truth" nabor podatkov za učenje AI.

## Namestitev

---

### Python

Aplikacija je napisana v Pythonu in testirana z verzijo 3.9. Načeloma bi novejša verzija morala tudi podpirati to aplikacijo. Prvi korak je torej inštalacija Pythona.

## Zunanje Python knjižnice

Poleg knjižnic, ki se inštalirajo kot del osnovne inštalacije Pythona, aplikacija uporablja še zunanje knjižnice: `matplotlib`, `PyQt5`, `numpy` in `pandas`. Te je treba inštalirati, detajli so odvisni od distribucije Pythona.

## Lokalne Python knjižnice

Za branje SiWIM binarnih datotek se uporablja `siwim-pi`. Dostopen je na `M:\disk_600_konstrukcije\Jank\siwim-pi`. Če se poganja aplikacijo direktno z diska `M:`, inštalacija ni potrebna, drugače se odpre `cmd` v tem direktoriju in požene `pip install` . (pika je pomembna).

## Skripta in podatki

Skripta se nahaja na `M:\disk_600_konstrukcije\Jank\braid_photo\label_braid_photo.py`.

V direktoriju `M:\disk_600_konstrukcije\Jank\braid_photo\data` so podatki:

- `*.nswd` v katerih so podatki o stehtanih vozilih
- `recognized_vehicles.json` vsebuje podatke o vozilih, ki jih je kategorizirala AI aplikacija
- `vehicle2event.json` vsebuje povezavo med vozilom in *event*-om — binarno datoteko v kateri so shranjeni signali in ostala informacija.
- Najpomembnejša datoteka je `metadata.hdf5`, v katero se shranjujejo rezultati ročnega označevanja vozil. Če se izgubi ali pokvari ta datoteka, bo ves do tedaj vložen trud zaman.

Če je Python pravilno inštaliran, bi moral dvoklik na skripto le-to zagnati.

## Lokalno zaganjanje aplikacije

Pri vsakem zagonu mora aplikacija naložiti okoli 100MB podatkov v datotekah `recognized_vehicles.json` in `vehicle2event.json`. Če je omrežna povezava počasna, lahko to precej upočasni zagon aplikacije (po zagonu je vseeno, saj se ti dve datoteki naložita samo enkrat).

Alternativa je, da se aplikacijo in ti dve datoteki prekopira na lokalni disk. Če je, na primer, direktorij z aplikacijo `D:\braid_photo`, morata biti drevesna struktura in lokacije datotek sledeče:

```
D:\braid_photo
├─data
│   ├──recognized_vehicles.json
│   └─vehicle2event.json
├─doc
│   └─lbp.pdf
├─label_braid_photos.py
├─locallib.py
└─main_window_ui.py
```

Slabost takšne postavitve je, da je treba ob nadgradnji aplikacije ponovno prekopirati datoteke `*.py` in `*.pdf` (datoteke `*.json` se ne spreminjajo), prednost pa precejšnji prihranek časa pri zagonu aplikacije. Na avtorjevem domačem PCju na hitri internetni povezavi (750/100 Mbps) lokalni zagon traja okoli 5 sekund, zagon z VPN omrežnega pa pol minute.

## Zaklepanje datoteke `metadata.hdf5`

Občasno (zelo redko) bo potrebna daljša obdelava, pri čemer se vsebina datoteke ne sme spreminjati. V tem primeru se bo ob zagonu aplikacije (ali med samim izvajanjem) pojavil opozorilen dialog:

V tem primeru je treba zapustiti aplikacijo in počakati, da se datoteka odklene. Načeloma bo zaklep objavljen vnaprej.

## Predpriprava na zagon

Za polno funkcionalnost aplikacije je treba priklopiti nekaj omrežnih diskov pod točno določenimi imeni:

- `\\mn-620-modeli.zag.si\siwim` kot **S:**
- `\\mn-620-modeli.zag.si\nfs-siwim` kot **T:**
- `\\mn-620-modeli.zag.si\braid` kot **B:**

V kolikor ne morete priklopiti diskov se oglasite pri avtorju.

Opozorilo: po reboot/sleep/hibernation,... je treba klikniti na vsakega izmed teh diskov v Explorerju. Windows imajo namreč to grdo navado, da disk dejansko priklopijo šele potem, ko v Explorerju klikneš nanj. Če pa poskuša kakšna aplikacija priti do diska pred tem, pride do napake.

## Uporaba več uporabnikov hkrati

Aplikacija skrbi za to, da jo lahko uporablja več uporabnikov hkrati. V zelo redkih primerih se lahko zgodi, da se zatakne pri dostopu do diska. Lahko pa seveda pride tudi do težav s povezavo v omrežje.

V splošnem to ni problem za podatke, ker gre večinoma za branje. Izjema je datoteka `metadata.hdf5` zaradi:

- Beleženja zadnjega dostopa takoj, ko se slika odpre
- Shranjevanja sprememb oznak.

Če aplikacija tega ne more narediti, opozori z dvema piskoma in z izpisom na konzoli. Priporočeno je, da se takrat aplikacijo zapusti in razišče izvor težav, saj se spremembe ne bodo pisale v datoteko.

## Uporaba aplikacije

### Zagon

Pri zagonu aplikacije se odpre konzola (`CMD` okno), v katerem se najprej izpiše

```
Loading recognized_vehicles.json, done.  
Loading vehicle2event.json, done.
```

potem pa se pojavi glavno okno aplikacije.

Konzole ne zapirajte, saj se s tem zapre tudi glavno okno. V konzoli se tudi izpišejo nekatere napake pri izvajanju (recimo da neke datoteke ne more najti), pa tudi, če pride do kakšne napake pri samem izvajanju aplikacije.

Če se zgodi to, prosim avtorju pošljite screenshot konzole in opis tega, kaj ste delali, ko je do napake prišlo.

## Zagon aplikacije z ročnim odpiranjem konzole

Težava pri zaganjanju z dvoklikom na `.py` datoteko je, da se pri napaki v aplikaciji zapre tudi konzola, s tem pa tudi izgine informacija o napaki. TBD je prestrezanje takšnih napak, vendar to ni najlažje.

Zato je bolje aplikacijo zagnati tako, da se konzolo ročno odpre — [10 načinov kako se zažene konzola](#) — potem pa v konzoli vtipka sledeče vrstice in vsako zaključí z `<ENTER>` :

- `M:`
- `cd \disk_600_konstrukcije\Jank\braid_photo`
- `python label_braid_photos.py`

Pri napaki se bo izpisal t.i. *stack trace* — spisek klicev funkcij in vrstic kje je šlo kaj narobe. Primer:

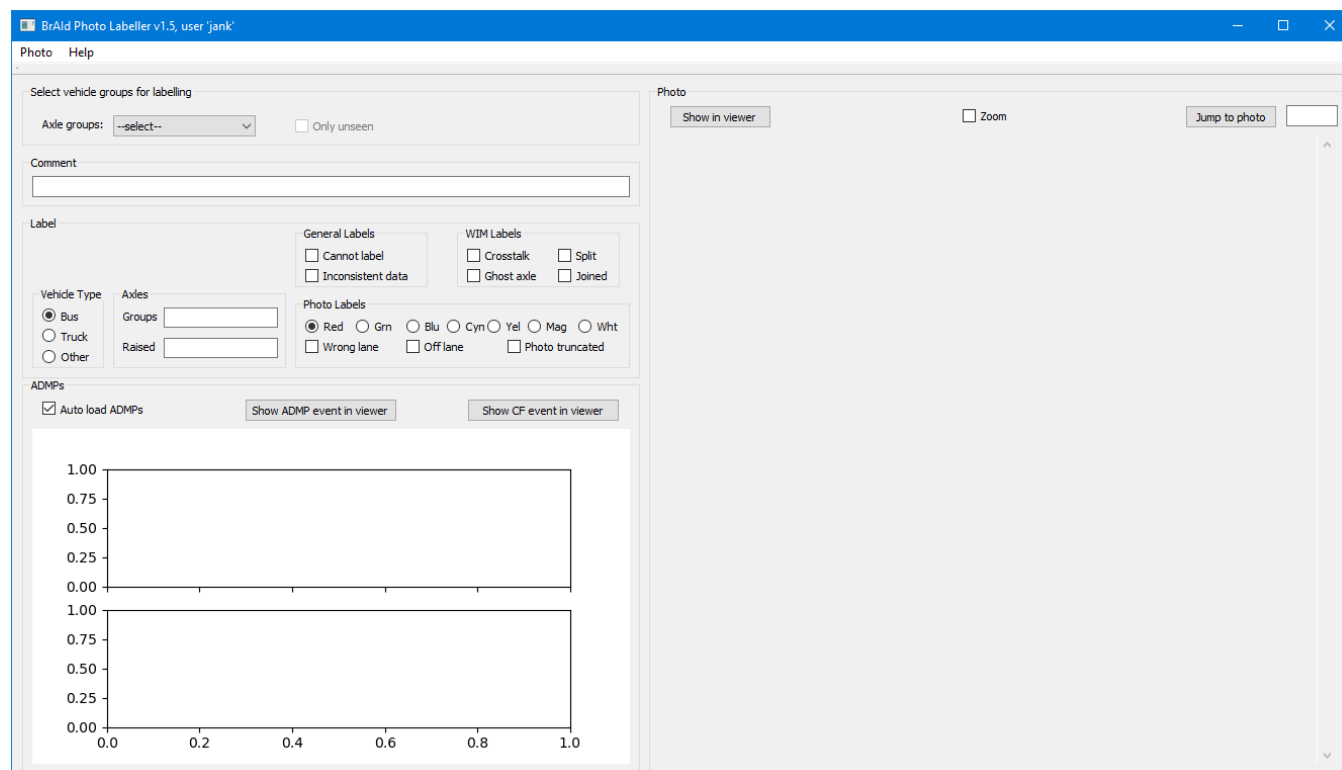
```
M:\disk_600_konstrukcije\Jank\braid_photo>python label_braid_photos.py
Loading recognized_vehicles.json, Traceback (most recent call last):
  File "M:\disk_600_konstrukcije\Jank\braid_photo\label_braid_photos.py", line 71, in
<module>
    rvs_loaded = json.load(f)
  File "C:\Python39\lib\json\__init__.py", line 293, in load
    return loads(fp.read(),
OSError: [Errno 22] Invalid argument

M:\disk_600_konstrukcije\Jank\braid_photo>
```

To prosim prekopirajte — tekst se označi s potegom z miške s pritisnjeno levo tipko in spravi na clipboard s tipko `<Enter>` — in pošljite avtorju.

Potem lahko aplikacijo zoper poženete z `<Up-Arrow>` in `<Enter>`.

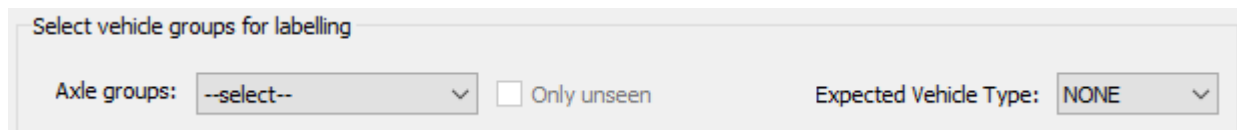
## Glavno okno



## Meni

V meniju sta dve postavki. **Photo** vsebuje postavke za premikanje po fotografijah in nastavljanje oznak, vendar so vse postavke dosegljive tudi s pomočjo bližnjic. Za hitro pomoč je spisek bližnjic dosegljiv v meniju **Help|Shortcuts**.

### Select vehicle groups for labelling



Tukaj se izbere množico vozil za označevanje. Struktura podatkov o slikah je bila določena na FAMNIT na osnovi uporabe AI klasifikacije slik, tega se drži tudi aplikacija. Glavna delitev na je skupine osi vozil. Primer je 113, ki predstavlja klasičen vlačilec s polpriklopnikom (šleper po domače).

Velike množice vozil (npr., tovornjaki s skupinami 113) so razdeljene na podmnožice moči 1000. S tem je lažje načrtovati in razdeliti obdelavo med več ljudi, saj je obdelovanje različnih podmnožic varno. Hkrati pa predvidevamo, da 1000 vozil predstavlja za približno uro dela, če predpostavimo 3.6 sekunde za povprečen pregled in morebitni popravek enega vozila. V tem primeru so vnosi v polju *Axle groups*: oblike, npr., **113 [02/13] (1000)**, kar pomeni druga podmnožica (z močjo 1000) izmed 13 podmnožic vozil s skupinami 113.

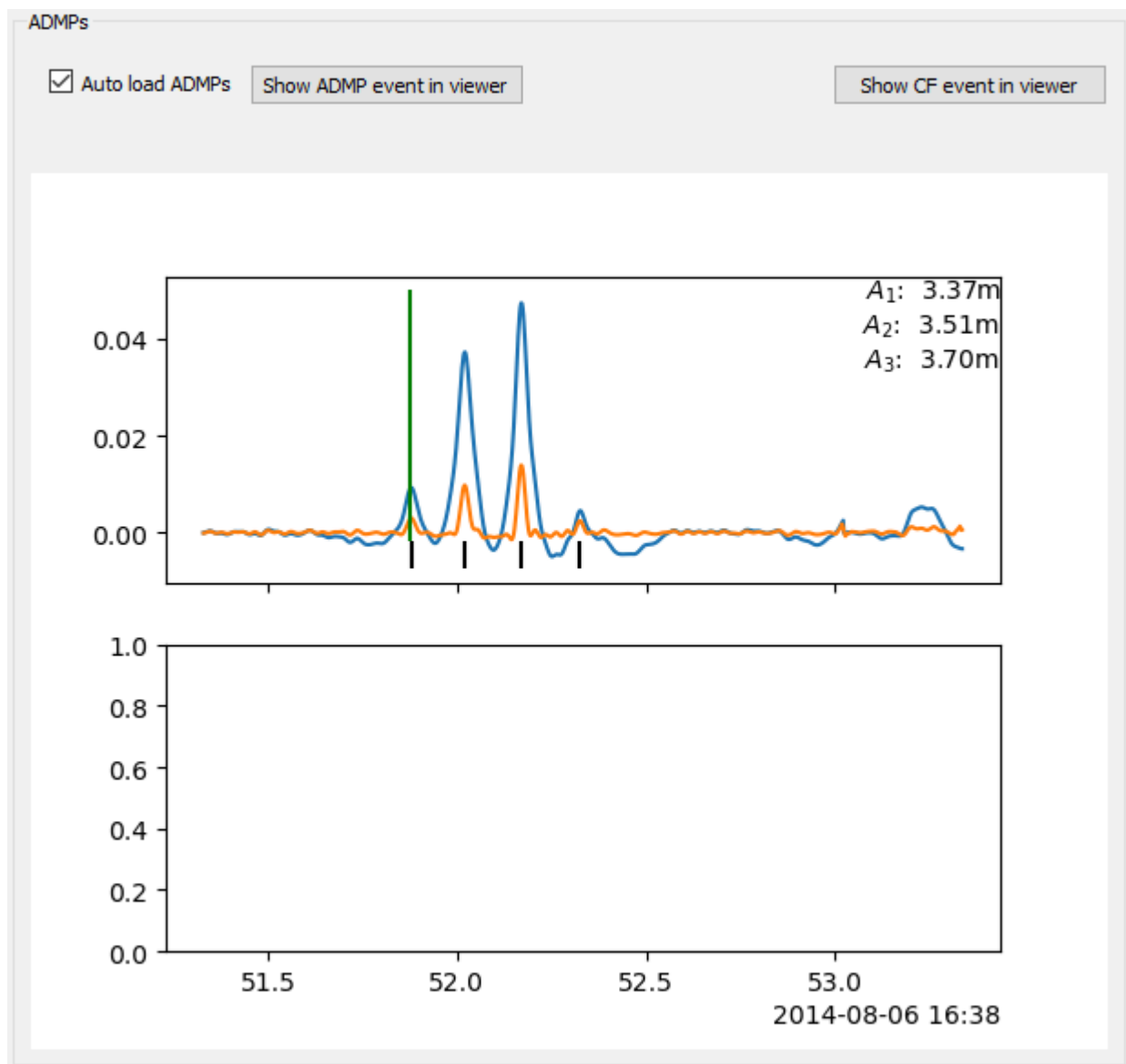
Potrditveno polje *Only unseen* omogoča nalaganje samo tistih slik, ki jih še noben ni videl. Trenutno je ta možnost onemogočena, ker zmeša oštevilčenje slik. Ponovno bo omogočena skupaj z možnostjo *Only with comments*.

Z izbiro ene izmed možnosti s spiska *Expected Vehicle Type* lahko nastavimo pričakovan tip vozila. Če je tip vozila drugačen, se prikaže na rdečem ozadju, kot je to prikazano na naslednji sliki. Če so tipi v določeni skupini vozil pretežno mešani, morda to ni tako uporabno, pri skupinah osi, npr., 113, kjer avtobusa ni, pa lahko precej zmanjša možnost da spregledamo napačno označeno vozilo.

expected\_type

### ADMPs

V tem razdelku se lahko vidi signale iz detektorjev osi (ADMP kanalov). Primer je na sledeči sliki.



Zgornji graf je za pas 1, spodnji za pas 2. Na grafih je z modro narisan originalni signal, z oranžno filtriran signal, s črnimi črtami detektirane osi, ter z zeleno črto timestamp obravnavanega vozila. Ob desnem robu so izpisane medosne razdalje vozila.

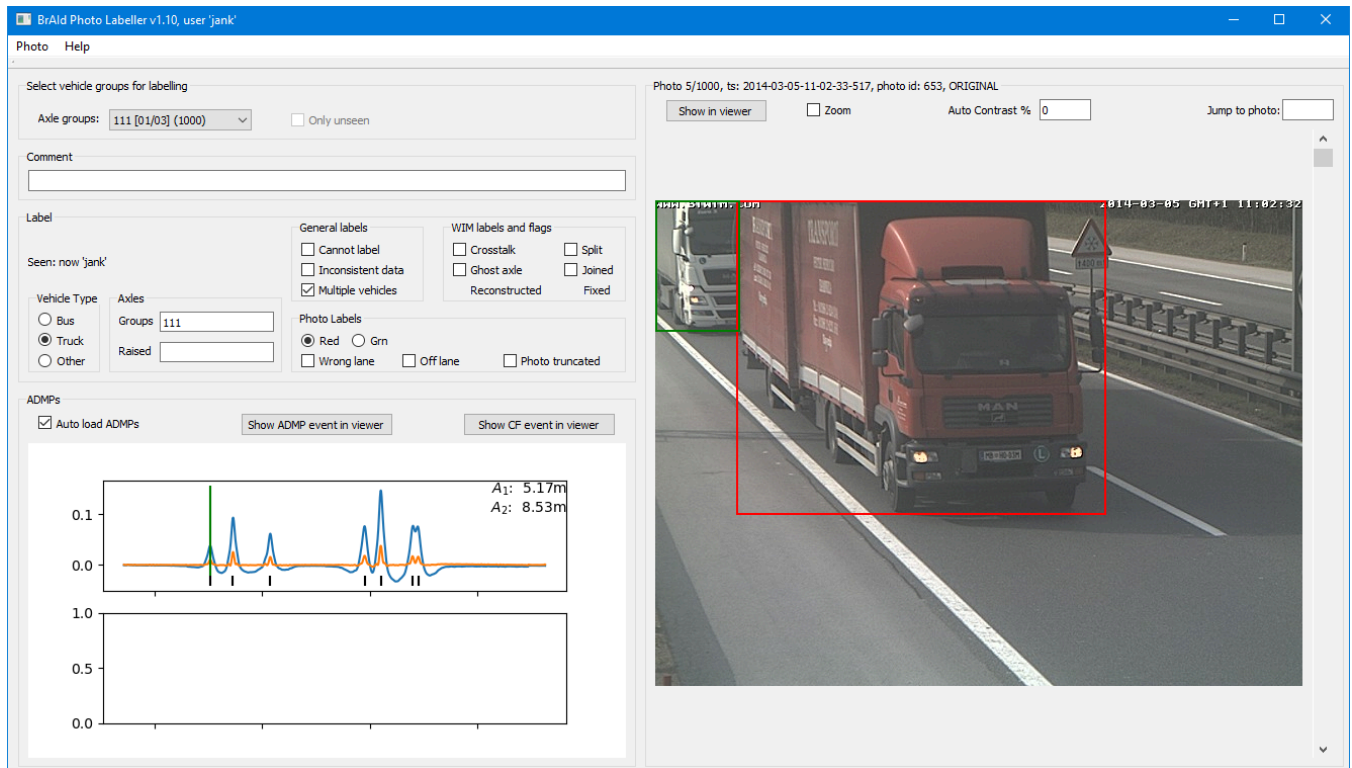
Z odkljukanim izbirnim poljem *Auto load ADMPs* se signali naložijo avtomatično, skupaj s sliko. Drugače je potrebno pritisniti **D**.

S tipkama *Show ADMP event in viewer* in *Show CF event in viewer* se lahko signale pregleda v zunanjem pregledovalniku SiWIM eventov. Dve možnosti sta zato, ker v *CF* event-ih ni diagnostike za detektorje osi, v *ADMP* event-ih pa ni diagnostike o tehtanju.

Pri pretehtavanju vozil so šli podatki nazadnje skozi modul **cf**, ki surove teže pomnoži s kalibracijskim faktorjem. Vendar je bilo v teh datotekah izklopljeno shranjevanje diagnostik za detekcijo osi. Za potrebe projekta sem spustil vse originalne event-a še skozi generiranje diagnostik za ADMPje, ni pa šlo skozi tehtanje.

## Photo

Ko se izbere skupine osi, se v razdelku *Photo* takoj pojavi prva fotografija znotraj te grupe. Primer je na naslednji sliki:



V imenu razdelka je napisana zaporedna številka vozila, število vseh vozil, timestamp vozila, ID fotografije ter *ORIGINAL*, če oznake slike niso bile spremenjene ali *CHANGED*, če so bile.

Ko je slika naložena, jo je možno s klikom na *Show photo in viewer* naložiti v eksterni pregledovalnik slik.

### Izbira fotografije

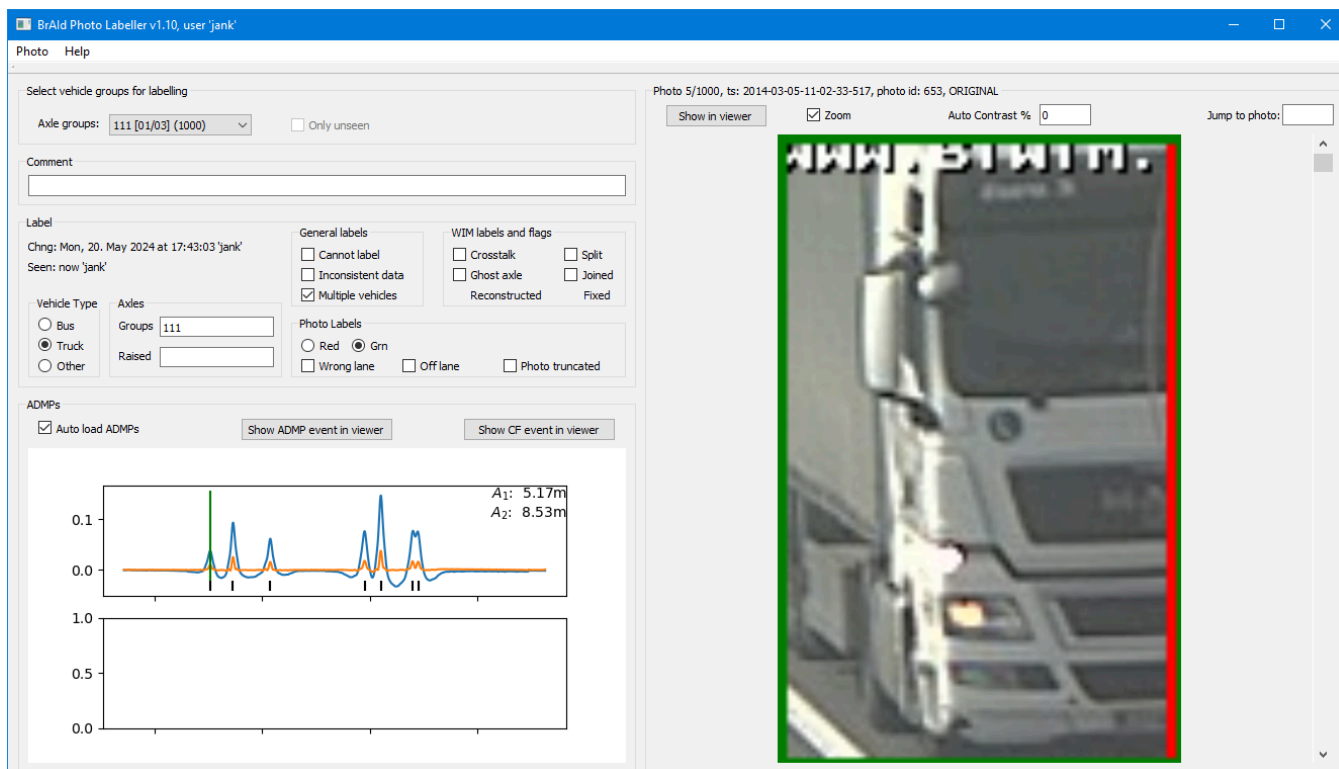
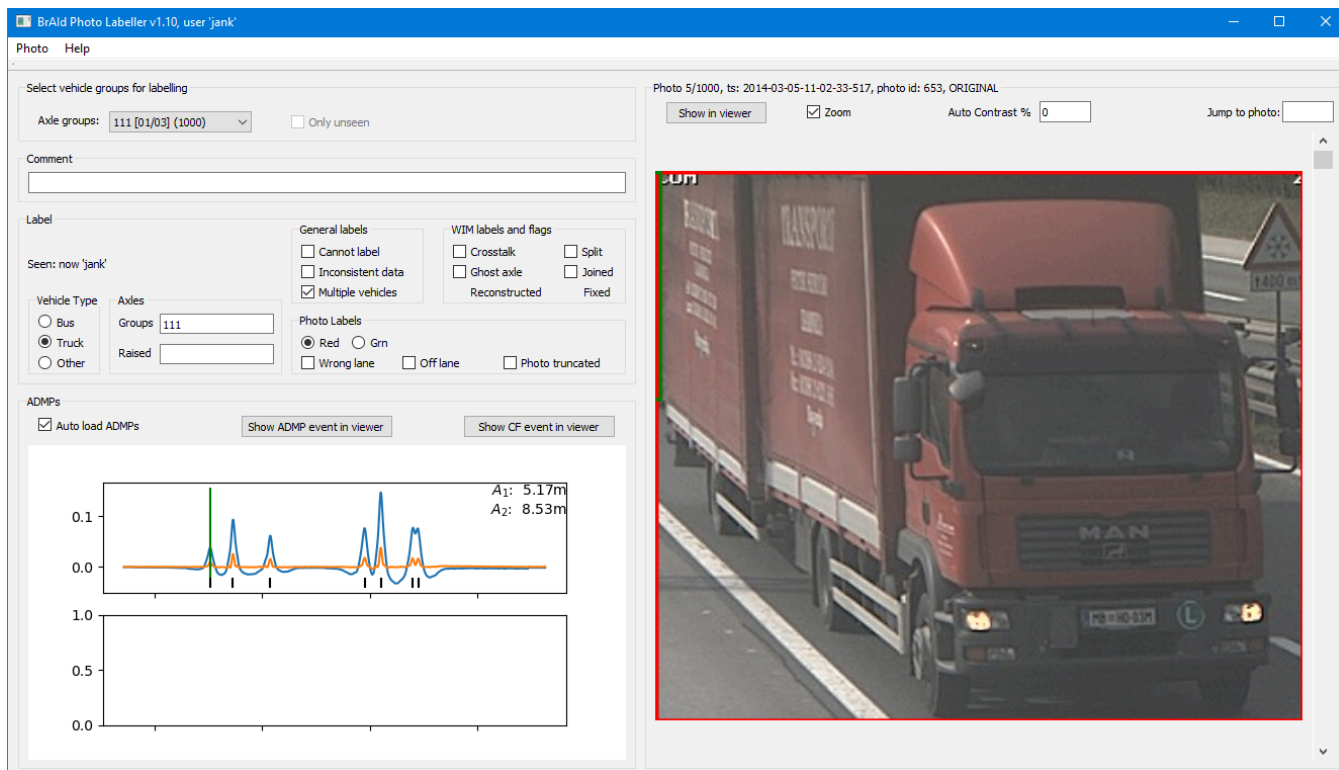
Slike se lahko izbira s puščico gor — *<Up>* ali dol — *<Down>*. Lahko pa tudi s klikanjem na drsni trak poleg slike.

Za lažjo izbiro slike je možno vpisati številko slike v vnosno polje zgoraj desno in s pritiskom na tipko *Jump to photo* neposredno izbrati to sliko.

### Zoženje pogleda (zoom)

Možno je tudi zožiti pogled (zoom) na okvirček ali ga razširiti nazaj na celo sliko. To se naredi s klikom na izbiro polje *Zoom* ali z bližnjico *<Z>*. Če se spremeni izbira barve oznake, se spremeni tudi pogled. Primera sta na naslednjih slikah:





## Popravljanje slik

Z klikom miške na sliko je možno izvesti preproste popravke slike. V naslednji tabeli so zbrane tipke in z njimi povezani popravki:

Tipka	Popravek
<Leva>	Povečanje kontrasta za 25%
dvoklik <Leva>	Povečanje kontrasta za 125%

Tipka	Popravek
<Shift>+<Leva>	Zmanjšanje kontrasta za 25%
<Desna>	Povečanje svetlosti za 25%
<b>dvoklik</b> <Desna>	Povečanje svetlosti za 125%
<Shift>+<Desna>	Zmanjšanje svetlosti za 25%
<Srednja>	Vrnitev originalne slike

Od verzije 1.10 naprej je možno določiti, da se pri nalaganju slike avtomatsko poveča kontrast, če je vrednosti v vnosnem polju *Auto Contrast %* različna od 0, od verzije 1.12 naprej pa tudi svetlost, za vrednost v vnosnem polju *Auto Brightness %*.

## Nastavljanje oznak

Na vrhu razdelka je izpisano uporabniško ime zadnjega, ki je fotografijo videl ter, če so bile oznake spremenjene, ime uporabnika, ki je zadnji spreminjal oznake.

V zgornjem delu razdelka *Label* so polja s katerimi lahko spreminjamo oznake. Skoraj vsa polja imajo asociirano bližnjico, ki je bila izbrana tako, da minimizira porabljen čas in premikanje prstov na tastaturi.

Takoj, ko se zabeleži sprememba katere izmed oznak, se ta sprememba napiše v datoteko **metadata.hdf5**. S tem je možnost, da bi stran vrgli delo, minimalna.

## Mnogotera vozila

AI včasih detektira več vozil. V tem primeru je najbolj verjetno vozilo označeno z rdečim kvadratom, potem pa si sledijo zelena, morda, cyan, rumena, meganta in bela. Prikazane so samo oznake, ki so tudi prisotne na sliki.

To je prva oznaka, ki jo je treba nastaviti, če je to potrebno.

## Tip vozila

Tip lahko določimo s tipkami **B** za **Bus**, **T** za **Truck** in **O** za **Other**. Nekateri tovornjaki, ki jih je AI napačno klasificiral kot avtobus, imajo že nastavljeno to izbirno polje (na osnovi skupin osi)

## Osi, grupe in dvignjene osi

V polju *Groups* se prikažejo trenutno detektirane skupine osi, npr., 113. Če se izkaže, da je SiWIM napačno detektiral osi, se tukaj popravi v pravilno vrednost. Sprememba se takoj zapiše med metapodatke. V polju obstaja tudi "undo", s klasično tipko **<Ctrl>-Z**.

V polju *Raised* se navede grupo v kateri je dvignjena os. Tipičen primer je, ko šleper dvigne prvo os v trojčku na polpriklopniku. Tedaj bi SiWIM detektiral skupine 112. V tem primeru v polje *Raised* vpišemo vrednost **3** (ker je manjkajoča os v tretji skupini osi). Lahko je dvignjenih več os, tedaj z vejico ločimo grupe z dvignjenimi osmi. V fiktivnem primeru, ko bi polpriklopnik dvignil dve osi v trojni osi, pa še vlekel bi priklopnik z eno dvignjeno osjo od dveh, bi v polje vnesel **3,3,4**.

Pri spreminjanju polja *Raised* aplikacija samodejno popravi vrednost v polju *Groups*, v tem primeru bi se skupine 112 spremenile v 113. Po tem je seveda možno še ročno popraviti polje *Groups*.

Za hitrejše nastavljanje dvignjenih osi so na voljo bližnjice **<Alt>+<N>**, kjer je **<N>** tipka od 2 do 5.

N.B.: Pri avtomatskem spreminjanju polja *Groups*, se za izhodišče vedno vzame originalno vrednost. Torej, če vozilu 122 ročno popravimo grupe na 123, potem pa še v polju *Raised* določimo dvignjeno osi v drugi grupi z vnosom vrednosti **2**, bo aplikacija zavrgla ročno spremembo skupin in končni rezultat bodo skupine 132.

### Oznake fotografije

- **Napačni pas:** Načeloma so med vozili izbrana samo tista, ki jih je SiWIM detektiral na prvem pasu. Če je AI našel vozilo na drugem pasu, se to označi tukaj. Bližnjica je **L** za *Wrong lane*.
- **S pasu:** Včasih se zgodi, da vozilo ne vozi po svojem pasu. Bližnjica je **F**, za *Off lane*.
- **Slika odrezana:** Če je slika vozila odrezana. Bližnjica je **U** za *Photo truncated*.

### WIM oznake

- **Presluh:** Včasih pride do presluha z enega pasu na drugega in vozilo se pojavi na obeh pasovih. Bližnjica je **R** za *Crosstalk*.
- **Navidezna os:** To je mišljeno predvsem za osi pred ali po legitimnem vozilu, ne odvečno osi znotraj vozila. Bližnjica = **G** za *Ghost axle*.
- **Vozilo razpolovljeno:** Če je medosna razdalja v kakšnem vozilu daljša od najdaljše v klasifikacijski tabeli, SiWIM razpolovi vozilo med tema osema v dve vozili. Bližnjica je **S** za *Split*.
- **Vozilo združeno:** Če si dve vozili sledita preblizu eno drugemu, jih SiWIM združi v eno vozilo. Bližnjica je **J** za *Joined*.
- **Več vozil:** Da bi se začetno raziskovanje FAMNITa omejilo samo na dogodke v katerih je prisotno samo eno vozilo, se označi vozilo, ki ni samo v dogodku. Načeloma je ta napaka že strojno nastavljena iz informacij iz NSWd, lahko pa se zgodi, da kakšen primer uide.

Tedaj se lahko uporabi bližnjico **M** za *Multiple vehicles*.

### Rekonstruirane in popravljene osi

Poleg teh štirih oznak, ki jih lahko uporabnik nastavi, sta še oznaki za rekonstruirana in popravljena vozila (*Reconstructed* in *Fixed*). Ti dve sta generirani iz NSWd datotek in shranjeni v datoteki `metadata.hdf5`. Obe lahko razložita diskrepance med detektiranimi grupami, sliko in grafom in s tem zmanjšata možnost, da je potrebno uporabiti oznako za nekonsistentne podatke.

Rekonstrukcija je funkcija, ki vozilom z določenimi grupami osi doda os po določenih pravilih, ponovno preračuna teže in, če pride do izboljšanja prilagajanja signalu, obdrži dodane osi. Te dodane osi se ne shranijo v diagnostični kanal, prikazan v grafu *ADMPs*.

Funkcija *fix* pa je bila implementirana v zunanji Python skripti in je delovala na NSWd datotekah

### Splošne oznake

- **Nekonsistentni podatki:** Včasih pride do razhajanj med detektiranimi osmi in osmi prikazanimi na grafu. To je zato, ker so bile osi za graf rekonstruirane z, kot kaže, malenkost drugačnimi parametri detekcije osi, lahko pa tudi zaradi rekonstrukcije in popravljanja osi. Načeloma je v tem primeru že strojno nastavljena oznaka za [rekonstruirane in/ali popravljene osi](#), lahko pa kak primer uide strojnemu označevanju.

Tedaj se lahko preveri stanje z ogledom originalnih podatkov (*Show CF event in viewer*) in označi napako. Bližnjica je **I**, za *Inconsistent data*.

- **YOLO napaka:** Včasih YOLO ne najde vozila ali pa označi samo napačno vozilo. Bližnjica je **Y** za **YOLO error**.
- Zadnjo možnost se uporabi, ko ni dovolj informacij, da bi sploh pregledal sliko in jo označil (ali pa ne). Lahko pa YOLO sploh ni našel pravega vozila. Tedaj se uporabi **Ne morem označiti**. Bližnjica je **N** za *Cannot label / not found*.

## Komentar

Za dodajanje splošnih komentarjev je polje *Comment*. Vnos teksta je potrebno potrditi s pritiskom na tipko **<Enter>**.

## Smernice za označevanje

Tukaj so zbrane smernice za označevanje, ki smo jih dorekli na sestanku 14. maja 2024:

- Za ne-avtobuse se v principu uporabi tip vozila *Truck* (tudi za kombije, gasilce,...). Tip vozila *Other* se uporablja le, če je res kaj zelo čudnega.
- Pri oznaki *Cannot label* se ne piše razloga — le malo verjetno je, da bo kdo to kdaj gledal.
- Komentar se uporabi le, če je kar res izjemnega. Ko/če se bomo odločili pregledovati komentarje, je koristno, da jih ni preveč.

## Opis podatkov v datoteki `metadata.hdf`

V tem razdelku je zbrana dokumentacija za datoteko in pot do te datoteke.

Osnovni SiWIM podatki so zbrani v datotekah `.nswd`, ki vsebujejo podatke o vozilih, ter pripadajoče slike v datotekah `.vehiclephotos`, iz katerih je FAMNIT s pomočjo `siwim-pi` knjižnice ekstrahirale slike, jih poslal skozi YOLO algoritem za prepoznavo vozil in ZAGu poslal dva kompleta podatkov:

- `recognized_vehicles.json` v kateri so za vsako sliko zbrani podatki, med drugim ID slike timestamp-a fotografije in vozila, en ali več detektiranih tipov vozila,...
- Slike na katerih so z barvnimi kvadrati označena detektirana vozila

## Struktura datoteke in branje

Datoteka je strukturirana v dva nivoja. Na prvem nivoju so grupe (ekvivalentno direktorijem), katerih imena ustrezajo skupinam osi vozila, npr., 113, 11,... Znotraj tega so posamezni dataset-i, katerih imena ustrezajo ID slik. Dataset vsebuje niz z JSON predstavitvijo metapodatkov za vsako sliko. Branje je možno z naslednjo funkcijo:

```
def load_metadata(rv, filename, axle_groups=None, photo_id=None, exists=False,
seen_by=False):
    """Loads metadata for recognised vehicle, `rv`
    If `axle_groups` and/or `photo_id` is None, the value is read from `rv`
    If both are not None, you can pass None for `rv`
    If `exists` is True, returns True if the entry exists, False otherwise
    If `seen_by` is True, returns True if it has been seen by anyone, False otherwise
    If there's no data, the default metadata `{'seen_by': None, 'changed_by': None}` is
    returned
    """
    if axle_groups is None:
```

```

    axle_groups = rv['axle_groups']
    if photo_id is None:
        photo_id = rv['photo_id']
    try:
        with h5py.File(filename, 'r') as f:
            result = json.loads(f[f"{axle_groups}/{photo_id}"].asstr()[()])
            if exists:
                return True
            elif seen_by:
                return result['seen_by'] is not None
            else:
                return result
    except:
        return False if exists or seen_by else {'seen_by': None, 'changed_by': None}

```

pisanje pa s funkcijo

```

def save_metadata(rv, metadata, filename, axle_groups=None, photo_id=None, timeout=None):
    """Saves metadata for recognised vehicle
    If `axle_groups` and/or `photo_id` is None, the value is read from `rv`
    If both are not None, you can pass None for `rv`
    Waits for up to `timeout` seconds before raising an exception, if the file cannot be
    written to
    """
    if axle_groups is None:
        axle_groups = rv['axle_groups']
    if photo_id is None:
        photo_id = rv['photo_id']

    f = None
    wait_until = datetime.datetime.now()
    if timeout:
        wait_until += datetime.timedelta(seconds=timeout)
    try:
        while True:
            try:
                f = h5py.File(filename, 'a')
                break
            except:
                if datetime.datetime.now() < wait_until:
                    time.sleep(0.1)
                    continue
                else:
                    raise RuntimeError(filename)
        try:
            grp = f.require_group(f"{axle_groups}")
        except TypeError:
            grp = f[f"{axle_groups}"]
        try:
            grp[str(photo_id)] = json.dumps(metadata)

```

```

except:
    data = grp[str(photo_id)]
    data[...] = json.dumps(metadata)
finally:
    if f is not None:
        f.close()

```

## Izbira časovnega intervala

Čeprav je referenčnih podatkov za skoraj dve leti, smo izmed teh izbrali samo obdobje med 5. marcem 2014 in 3. septembrom 2014. V ostalem času slik ni, ali pa so narejene z IR kamero, ki je nerodno postavljena tako, da slika tudi drugo stran, ali pa je nasploh slabo nastavljena.

## Filtriranje podatkov in generiranje pomožne datoteke

V originalnih podatkih je bilo nekaj podatkov odveč. Izkazalo se je, da se je nesmiselno ukvarjati z vozili na pasu 2, saj so zelo pogosto na sliki skriti za vozilom na pasu 1, še bolj pa je relevantno dejstvo, da so na pasu 2 redkokdaj vozila, ki so za obravnavo zanimiva, saj tovornjaki v ogromni večini vozijo po pasu 1.

Za hitrejšo obdelavo je bilo potrebno narediti preslikavo timestamp vozila → timestamp eventa.

Prva skripta za pred-procesiranje je prebrala vse `.nswd` datoteke ter originalno datoteko

`recognized_vehicles.json` in generirala novo datoteko `recognized_vehicles.json`, v kateri so samo vozila na pasu 1, ter `vehicle2event.json` z zahtevano preslikavo.

## Metapodatki

Poln spisek metapodatkov z razlagami:

- `seen_by` se nastavi takrat, ko se slika naloži v aplikacijo. S tem je potrjeno, da je nekdo videl sliko. Vrednost je tuple, prvi element je timestamp, drugi uporabniško ime.
- `changed_by` se nastavi takrat, ko se s pomočjo aplikacije spremeni metapodatek. Ta podatek ostane med metapodatki tudi v situaciji, ko se po pomoti nastavi neka oznaka (npr., *Cannot label*), potem pa se oznaka umakne. Tudi tu je vrednost tuple.
- `vehicle_type` vsebuje tip vozila, `bus`, `truck`, ali `other`.
- `axle_groups` vsebuje skupine osi vozila. Če podatka ni, ali pa se vrednost ujema z originalno vrednostjo, je originalna vrednost OK.
- `raised_axles` pove v kateri skupini osi je morebitna dvignjena os.
- `segment` pove kateri izmed barvnih okvirčkov na sliki označuje pravo vozilo. Možne vrednosti so `r`, `g`, `b`, `c`, `y`, `m`, in `w`. Če podatka ni, je YOLO pravilno detektiral vozilo, ki je označeno z rdečim okvirčkom.
- `errors` je `dict`, katerega ključi opisujejo napake, vrednosti pa so lahko `0`, če napaka ni prisotna ali `2`, če napaka je prisotna. Vrednost `0` je vpisana samo v primeru, da je uporabnik napako najprej označil, potem pa se je premislil, drugače pa vrednost ni vpisana. Možne napake so: `wrong_lane`, `off_lane`, `photo_truncated`, `vehicle_split`, `vehicle_joined`, `crosstalk`, `ghost_axle`, `multiple_vehicles`, `reconstructed`, `fixed`, `inconsistent_data`, `yolo_error` in `cannot_label`.
- `comment` je niz, v katerega so lahko napisani komentarji k posameznemu vozilu.

## Pred-nastavljeni metapodatki

Preden se je začelo izvajati ročno pregledovanje slik, so se v metapodatke strojno napisali določene vrednosti. V teh primerih se ni nastavljal polje `changed_by`, ki je rezervirano za ročne spremembe, prav tako se ni nastavljal polje `seen_by`.

- Za vsa vozila, katerih osne skupine *niso* v spisku [11, 12, 111, 121], YOLO pa jih je prepoznal kot avtobuse, se je nastavilo `vehicle_type` na `truck`.
- Vozila, ki so bila v event-u skupaj še s kakšnim drugim vozilom, se je nastavilo `error: multiple_vehicles`.
- Vozila, ki jim je SiWIM rekonstruiral osi (to se ne vidi na ADMP grafih), so dobila oznako `error: reconstructed`. Te oznake se v aplikaciji ne da spremeniti.
- Vozila, ki jih je algoritem v `fix.py` pri post-procesiranju popravil, ali pa ki so bila ročno popravljena z aplikacijo *SiWIM-D*, so dobila oznako `error: fixed`. Tudi te oznake se v aplikaciji ne da spreminjati.