Aplikacija za označavanje objekata u digitalnim slikama

Prakljačić, Stjepan

Master's thesis / Diplomski rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:705999

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-15**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

Aplikacija za označavanje objekata u digitalnim slikama

Diplomski rad

Stjepan Prakljačić

Osijek, 2023.

SADRŽAJ

1. UVOD 1

2. NAČINI OZNAČAVANJA OBJEKATA U SLIKAMA 3

2.1. Postupak označavanja objekata u digitalnim slikama 4

2.2. Način pohrane oznaka objekata u strukturiranim računalnim formatima 5

2.2.1. COCO JSON 6

2.2.2. PASCAL VOC XML 9

2.2.3. YOLO TXT 11

2.3. Postojeći programski alati za označavanje objekata u slikama 12

3. IZRADA APLIKACIJE ZA OZNAČAVANJE PODATAKA ZA POTREBE IZGRADNJE MODELA ZASNOVANIH NA STROJNOM UČENJU 15

3.1. Analiza zahtjeva na aplikaciju 15

3.2. Dizajn aplikacije 16

3.3. *Qt* i *Qt Designer* alati 20

3.4. Implementacija sučelja za upravljanje projektima 22

3.5. Implementacija sučelja za označavanje objekata 25

3.6. Implementacija automatske detekcije objekata u slikama 34

4. VERIFIKACIJA APLIKACIJE ZA OZNAČAVANJE PODATAKA ZA POTREBE IZGRADNJE MODELA ZASNOVANIH NA STROJNOM UČENJU 38

4.1. Verifikacija sučelja za upravljanje projektima 39

4.2. Verifikacija sučelja za označavanje objekata 43

4.3. Verifikacija automatske detekcije objekata u slikama 57

5. ZAKLJUČAK 59

LITERATURA 60

SAŽETAK 62

ABSTRACT 63

ŽIVOTOPIS 64

1. UVOD

Posljednjih su godina tehnike strojnog učenja (engl. *Machine Learning* – ML) revolucionirale razna područja, od računalnog vida do obrade prirodnog jezika. Pristupi koje se koriste su najčešće iz područja nadziranog učenja te zahtijevaju dostupnost označenih skupova podataka. Svaki podatak u takvom podatkovnom skupu ima odgovarajuću oznaku (engl. *annotation*). Na primjer, kod problema klasifikacije slika, svaka slika ima oznaku kojoj kategoriji pripada. Međutim, stvaranje takvih skupova podataka je često izazovan i dugotrajan proces, pogotovo kada je potrebno označiti veliku količinu podataka.

Kod ručnog označavanja, osoba koristi svoje kognitivne sposobnosti, znanje o domeni i razumijevanje konteksta kako bi točno označila podatkovni skup. Kako bi se proces označavanja olakšao i pojednostavio, razvijeni su različiti programski alati za označavanje koji se koriste različitim domenama i tipovima podataka.

Detekcija objekata ključni je problem računalnog vida, koji uključuje lokalizaciju i klasifikaciju objekata u digitalnim slikama. Moderni detektori objekata, temelje se na dubokim neuronskim mrežama. Za efikasno treniranje takvih mreža najčešće je potrebna velika količina označenih slika na razini prisutnih objekata što podrazumijeva oznaku lokalizacije objekta na slici, kao na primjer, pomoću graničnog okvira (engl. *bounding box*) i klase ili kategorije kojoj objekt pripada. Takav način označavanja slika značajno je složeniji nego kod jednostavne klasifikacije slika (pogotovo kada postoji mnogo objekata na slici) stoga su neophodni programski alati kako bi se postupak označavanja olakšao i ubrzao.

U ovom radu predložena je aplikacija koja omogućuje označavanje objekata od interesa u digitalnim slikama. Aplikacija koristi intuitivan način označavanja objekata uz pomoć miša, omogućujući korisnicima precizno i lako označavanje objekata od interesa. Također, korištenjem korisničkih sučelja za upravljanje projektima te sučelja za označavanje objekata, aplikacija omogućuje kreiranje i upravljanje korisničkim projektima, vizualizaciju, uređivanje i brisanje kreiranih oznaka te izvoz datoteka s oznakama u različitim formatima na disk. Dodatno, predložena aplikacija nudi mogućnost inicijalnog kreiranja oznaka nad objektima od interesa, korištenjem detektora temeljenog na dubokoj neuronskoj mreži.

Ovaj rad je strukturiran na sljedeći način. U drugom poglavlju opisani su načini označavanja, tipovi oznaka te postojeća programska rješenja koja omogućuju označavanje objekata od interesa u digitalnim slikama. Poglavljem tri detaljno je prikazan proces razvoja i implementacije aplikacije, korištenjem popularnih alata kao što su *Qt* i *Qt Designer*, osiguravajući jednostavna i vizualno privlačna sučelja. Za procjenu učinkovitosti aplikacije provode se sveobuhvatni procesi verifikacije, opisani u četvrtom poglavlju. Ovi procesi uključuju verifikaciju funkcionalnosti kreiranja projekta kroz sučelje za upravljanje projektima, funkcionalnosti kreiranja, pohrane i vizualizacije oznaka kroz sučelje za označavanje objekata te verifikaciju funkcionalnosti detekcije, pohrane i vizualizacije oznaka kroz proces automatske detekcije objekata u slikama, korištenjem detektora temeljenog na dubokoj neuronskoj mreži.

2. NAČINI OZNAČAVANJA OBJEKATA U SLIKAMA

Računalni vid je područje umjetne inteligencije (engl. *Artificial Intelligence* - AI) koje omogućuje računalima i sustavima prepoznavanje i izvlačenje značajnih informacija iz digitalnih slika. Osnovi problem u okviru računalnog vida je detekcija objekata gdje računalni algoritam lokalizira i klasificira objekte od interesa u digitalnim slikama.

Označavanje podataka u slikama ima ključnu ulogu u razvoju modela detekcije objekata koji se temelje na strojnom učenju. To je proces identificiranja neobrađenih podataka i davanja smislenih i informativnih oznaka tim podacima osiguravajući kontekst za modele strojnog učenja kako bi iz njih učili. Ručna obrada podataka dodavanjem oznaka nestrukturiranim podacima omogućuje algoritmima strojnog učenja da automatski prepoznaju koncepte koje te oznake opisuju [1].

Ručno označavanje podataka je vremenski zahtjevan proces jer zahtijeva da osoba ili tim koji provodi označavanje, analizira svaki podatkovni primjer i dodijeli mu jednu ili više odgovarajućih oznaka. Ovisno o složenosti zadatka i veličini skupa podataka, ovaj proces može potrajati, posebno kada se radi o velikom skupu podataka. U okviru računalnog vida, postupak označavanja svodi se na pregledavanje skupa slika od strane osobe ili tima koji provodi označavanje te se svakoj slici dodjeljuju oznake ovisno o problemu koji se razmatra [2], [3], [4]. Pri tome razlikujemo tri glavna problema:

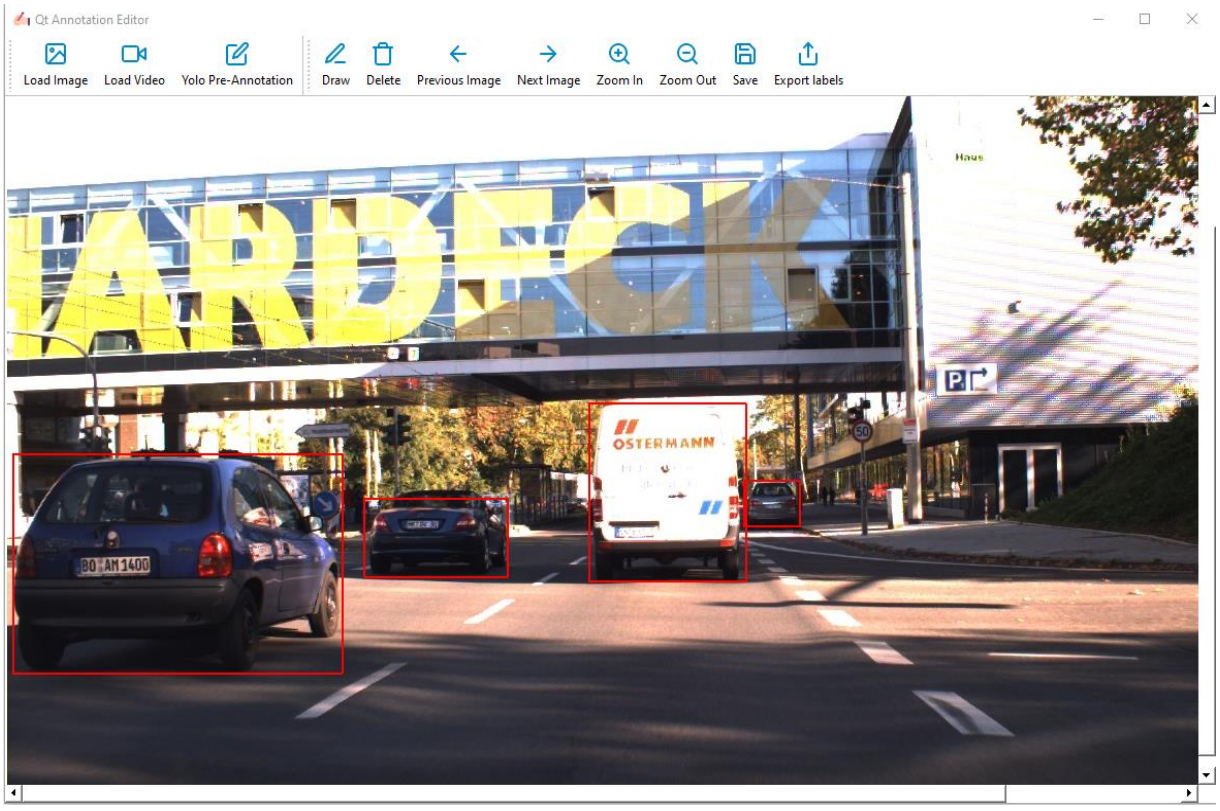
1. klasifikacija slika - postupak je kategorizacije slike u jednu ili više unaprijed definiranih klasa ili kategorija [5],
2. detekcija objekata - uključuje otkrivanje i lokaliziranje objekata od interesa unutar slike [6], i
3. segmentacija slike - postupak je dijeljenja slike u više segmenata ili regija, od kojih svaki predstavlja drugačiji objekt ili pozadinu na slici. Semantičkom segmentacijom dodjeljuju se semantičke oznake svakom pikselu na slici s ciljem pojednostavljenja prikaza slike [7].

U ovom radu, fokus je na problemu detekcije objekata u digitalnim slikama. Razvijena aplikacija pojednostavljuje izgradnju detektora objekata intuitivnim načinom označavanja objekata na računalu uz pomoć miša.

2.1. Postupak označavanja objekata u digitalnim slikama

Označavanjem objekata od interesa povezujemo svaki objekt s određenom klasom ili kategorijom, pružajući semantičko razumijevanje objektnog identiteta ili karakteristika. Na primjer, objekti unutar slike mogu biti označeni oznakama klase kao što su: „automobil“, „osoba“, „drvo“ itd., pokazujući vrstu prisutnog objekta [8].

Najčešći način označavanja objekata u digitalnoj slici je crtanjem graničnog okvira, na način da se definira početna točka te visina i širina pravokutnika oko objekta, kao što je prikazano slikom 2.1. Pomoću graničnog okvira moguće je nacrtati pravokutne okvire oko bilo kojeg objekta, a zatim toj površini dodijeliti oznaku. Svrha graničnog okvira je definiranje prostornog opsega objekta i pružanje vizualne reference za modele strojnog učenja koji su osposobljeni za detekciju objekata na slikama.



Slika 2.1. Primjer označavanje objekata graničnim okvirima u digitalnoj slici

Označavanje crtanjem poligona oko objekta na slici još je jedan način označavanja objekata. Na slikama se ove oznake mogu koristiti za ocrtavanje statičnih objekata.

Kada su ceste i staze objekti od interesa, za označavanje se koriste polilinije. Polilinija je linija koja je definirana kao niz povezanih točaka u dvodimenzionalnom prostoru koje se povezuju linearno kako bi se pratila struktura cesta i staza.

Označavanje ključnim točkama uključuje prepoznavanje i označavanje specifičnih točaka na objektu unutar slike u dvodimenzionalnom prostoru. Te točke, obično su važne značajke ili orijentiri, kao što su uglovi zgrade ili zglobovi ljudskog tijela. Označavanje ključnim točkama obično se koristi u aplikacijama kao što su procjena položaja, prepoznavanje radnji i praćenje objekata [9].

Odabirom odgovarajućeg programskog alata za označavanje olakšava se postupak označavanja i pruža potrebna funkcionalnost za vizualizaciju kreiranih oznaka. Alati koji se koriste za označavanje trebaju biti intuitivni, jednostavni za korištenje i kompatibilni sa željenim tipovima oznaka. Dodatno, alati moraju omogućiti stvaranje oznake neovisno o načinu označavanja te moraju pružiti mogućnost korištenja pomoćnih alata automatizacije procesa [10], [11].

2.2. Način pohrane oznaka objekata u strukturiranim računalnim formatima

Nakon procesa označavanja objekata na slikama, oznake kreirane graničnim okvirima se pohranjuju u jednu ili više datoteka, uspostavljajući nedvosmislenu vezu između svake pojedinačne slike u skupu podataka i odgovarajućih kreiranih oznaka. Iz tog razloga oznake se pohranjuju u strukturirane računalne formate kako bi se olakšala analiza i daljnja obrada [12]. Unatoč mogućnosti pretvaranja oznake iz jedne u drugu, posjedovanje alata koji može izravno napraviti oznaku u željenom formatu, pristup je koji štedi mnogo vremena. Formati pohrane datoteka s oznakama korišteni u ovom radu, a o kojima će biti detaljnije govora u sljedećim poglavljima su: PASCAL VOC XML (engl. *Extensible Markup Language*), COCO JSON (engl. *JavaScript Object Notation)*, YOLO TXT (engl. *Text Document File*) [13], [14].

2.2.1. COCO JSON

MS COCO (engl. *Microsoft Common Objects in Context*) podatkovni je skup, korišten u okviru računalnog vida za detekciju, segmentaciju i klasifikaciju objekata. COCO format za pohranjivanje oznaka je strukturirani način pohranjivanja oznaka u jednoj datoteci, korištenjem JSON formata koji se temelji na dvije strukture: kolekcija ključ-vrijednost i uređeni popisi vrijednosti. Kolekcije ključ-vrijednost analogne su *Python* rječnicima, gdje je svaki ključ povezan s odgovarajućom vrijednošću. Uređeni popis vrijednosti realizira se kao *Python* lista. Datoteka za pohranu slike je standardna datoteka u bilo kojem formatu, poput JPG (engl. *Joint Photographic Expert Group*) ili PNG (engl. *Portable Network Graphics*) [15].

COCO JSON se sastoji od:

- *info* - kolekcija općih informacija o skupu podataka,
- *images* - kolekcija slika u skupu podataka,
- *annotations* - kolekcija oznaka (uključujući granične okvire) koje su prisutne na svim slikama u skupu podataka,
- *licenses* - kolekcija informacija o licenci za slike u skupu podataka,
- *categories* - uređeni popis vrijednost klasa (kategorija) objekata od interesa.

Kako bi se izradio COCO JSON primijenjenih oznaka, obavezno je koristiti kolekcije: *images*, *annotations* te uređeni popis *categories*, dok ostale kolekcije (*info*, *licenses*) nisu obavezne [16]. Slika 2.2. prikazuje opći oblik COCO JSON formata za pohranu oznaka.

Linija ***Kod***

```
1:     {
2:       “info“                                :info,
3:       “images“                              :[image],
4:       “annotations“                         :[annotation],
5:       “licenses“                            :[license],
6:     }
7:     info{
8:        “year“                               :int,
9:       “version“                             :str,
10:      “description“                         :str,
11:      “contributor“                         :str,
12:      “url“                                 :str,
13:      “date_created“                        :datetime,
14:    }
15:    image{
16:      “id“                                  :int,
17:      “width“                               :int,
18:      “height“                              :int,
19:      “file_name“                           :str,
20:      “license“                             :int,
21:      “flickr_url“                          :str,
22:      “coco_url“                            :str,
23:      “date_created“                        :datetime,
24:    }
25:    license{
26:      “id“                                  :int,
27:      “name“                                :str,
28:      “url“                                 :str,
29:    }
```

Slika 2.2. Opći oblik COCO manifesta

Metapodaci svake slike predstavljene COCO JSON formatom strukturirani su kao kolekcije ključ-vrijednost s atributima:

- *id* - jedinstveni identifikator za sliku,
- *width* - širina slike u pikselima,
- *height* - visina slike u pikselima,
- *file_name* - naziv datoteke za pohranu slike,
- *license* - niz licenci,

- *flickr_url* - lokacija slike na Flickru,
- *coco_url* - lokacija slike,
- *date_captured* - datum i vrijeme kada je slika snimljena.

Slikom 2.3. prikazan je opći oblik oznake u COCO JSON formatu strukturirane kao kolekcija ključ-vrijednost s atributima:

- *id* - jedinstveni identifikator za oznaku,
- *image_id* - odgovara id-u slike,
- *category_id* - identifikator za oznaku koja identificira objekt unutar graničnog okvira. (Preslikava se u polje *id-a* kategorija),
- *segmentation* - informacije o segmentaciji u RLE ili *polygon* formatu za objekte na slici,
- *area* - označava površinu označenog objekta na slici u pikselima,
- *bbox* - koordinate gornje lijeve točke te visina i širina graničnog okvira u pikselima,
- *iscrowd* - označava radi li se o RLE ili *polygon* zapisu [17].

Linija ***Kod***

```
1:    annotation{
2:       “id“                          :int,
3:       “image_id“                    :int,
4:       “category_id“                 :int,
5:       “segmentation“                :RLE or[polygon],
6:       “area“                        :float,
7:       “bbox“                        :[x,y,width,height],
8:       “iscrowd“                     :0 or 1,
9:    }
10:   categories[{
11:      “id“                          :int,
12:      “name“                        :str,
13:      “supercategory“               :str,
14:   }]
```

Slika 2.3. Opći oblike oznake COCO manifesta

2.2.2. PASCAL VOC XML

PASCAL VOC (engl. *Visual Object Classes*) podatkovni je skup, korišten u okviru računalnog vida za detekciju, segmentaciju i klasifikaciju objekata. Sastoji se od tisuća slika označenih graničnim okvirima, segmentacijskim maskama i oznakama klasa [18].

Pascal VOC format za pohranu oznaka je strukturirani način označavanja slika u podatkovnom skupu, koji uparuje svaku sliku s pripadnom XML datotekom oznaka. Pascal VOC format uključuje tri primarne komponente: datoteke za pohranu slike, datoteke s oznakama te datoteke klasa. Datoteka za pohranu slike je standardna datoteka u bilo kojem formatu, poput JPG ili PNG. Datoteke s oznakama su strukturirane su XML formatom te sadrže granične okvire objekata i segmentacijske maske za svaki objekt na slici [19]. Slika 2.4. prikazuje opći oblik datoteke s oznakama. Svaka XML datoteka strukturirana je sa sljedećim atributima:

- *folder* - određuje naziv mape u kojoj se nalazi slika,
- *filename* - određuje naziv datoteke za pohranu slike, uključujući njenu ekstenziju,
- *path* - označava putanju do datoteke za pohranu slike,
- *source* - sadrži informacije o izvoru slike,
- *size* - sadrži informacije o dimenzijama slike,
- *width* - označava širinu slike u pikselima,
- *height* - označava visinu slike u pikselima,
- *depth* - predstavlja dubinu boje slike (obično 3 za RGB slike),
- *segmented* - označava je li slika segmentirana (0 ako nema segmentacije, 1 ako je segmentirana),
- *object* - predstavlja instancu objekta prisutnu na slici,
- *name* - određuje klasu objekta,
- *bndbox* - sadrži informacije o graničnom okviru objekta,
- *xmin* - određuje x-koordinatu graničnog okvira gornjeg lijevog vrha u pikselima,
- *ymin* - određuje y-koordinatu graničnog okvira gornjeg lijevog vrha u pikselima,
- *xmax* - određuje x-koordinatu graničnog okvira donjeg desnog vrha u pikselima,
- *ymax* - određuje y-koordinatu graničnog okvira donjeg desnog vrha u pikselima,
- *segmentation* - definira segmentaciju objekta u obliku poligon zapisa.

Linija ***Kod***

```
1:     <annotation>
2:       <folder>folder_name</folder>
3:       <filename>image_name.jpg</filename>
4:       <path>/path/to/image/image_name.jpg</path>
5:       <source>
6:         <database>PASCAL VOC2007</database>
7:         <annotation>PASCAL VOC2007</annotation>
8:         <image>flickr</image>
7:       </source>
8:       <size>
9:         <width>width_of_image</width>
10:        <height>height_of_image</height>
11:        <depth>3</depth>
12:      </size>
13:      <segmented>0</segmented>
14:      <object>
15:        <name>object_class_name</name>
16:        <pose>Unspecified</pose>
17:        <truncated>0</truncated>
18:        <difficult>0</difficult>
19:        <bndbox>
20:          <xmin>xmin_of_object</xmin>
21:          <ymin>ymin_of_object</ymin>
22:          <xmax>xmax_of_object</xmax>
23:          <ymax>ymax_of_object</ymax>
24:        </bndbox>
25:        <segmentation>
26:          <polygon>
27:            <x1>,<y1>,<x2>,<y2>,<x3>,<y3>,...,<xn>,<yn>
28:          </polygon>
29:        </segmentation>
30:      </object>
31:    </annotation>
```

Slika 2.4. Opći oblik datoteke s oznakama Pascal VOC formata

2.2.3. YOLO TXT

YOLO TXT format za pohranu oznaka je strukturirani način označavanja slika u podatkovnom skupu, koji uparuje svaku sliku s pripadnom tekstualnom datotekom oznaka. Slika 2.5. prikazuje opći oblik datoteke s oznakama u YOLO TXT formatu [20] .

Svaka tekstualna datoteka s oznakama opisana je sljedećim atributima:

- *object-class* - određuje klasu objekta,
- *xmin* - određuje x-koordinatu graničnog okvira gornjeg lijevog vrha u pikselima, skaliranu s obzirom na širinu slike,
- *ymin* - određuje y-koordinatu graničnog okvira gornjeg lijevog vrha u pikselima, skaliranu s obzirom na visinu slike,
- *width* - određuje širinu graničnog okvira u pikselima,
- *height* - određuje visinu graničnog okvira u pikselima.

Linija ***Kod***

```
1:      <object-class> <xmin> <ymin> <width> <height>
```

Slika 2.5. Opći oblik datoteke s oznakama u YOLO TXT formatu

Svaki redak u datoteci s oznakama odgovara jednom označenom objektu na slici, a prilikom pohrane na disk datoteci se dodjeljuje ime identično imenu slike [21]. Slika 2.6. prikazuje primjer zapisa oznaka u YOLO TXT formatu za sliku na kojoj su prisutna dva objekta.

Linija ***Kod***

```
1:      0 0.374269 0.590625 0.220904 0.222005
2:      1 0.376667 0.587905 0.193333 0.210723
```

Slika 2.6. Primjer zapisa oznaka u YOLO TXT formatu za sliku na kojoj su prisutna dva objekta

2.3. Postojeći programski alati za označavanje objekata u slikama

U radu [8] autori su predstavili LabelMe, alat za označavanje slika temeljen na web pregledniku koji je razvio Laboratorij za računalnu znanost i umjetnu inteligenciju na MIT-u. Ovaj je alat osmišljen kako bi olakšao ručno označavanje objekata u slikama, omogućujući stvaranje označenih skupova podataka za modele računalnog vida.

U radu [9], predstavljen je VGG Image Annotator (VIA), alat za označavanje slika otvorenog koda koji je razvila Visual Geometry Group (VGG) na Sveučilištu u Oxfordu. Primarni cilj VIA-e je ponuditi korisnički pristupačnu platformu za označavanje slika, olakšavajući stvaranje označenih skupova podataka za različite zadatke računalnog vida.

Labelbox, predstavljen u radu [10], platforma je za označavanje podataka temeljena na oblaku. Cilj platforme je pojednostaviti stvaranje i upravljanje označenim podacima za projekte umjetne inteligencije i strojnog učenja. Pomoću Labelbox-a istraživači i znanstvenici koji se bave podacima mogu pojednostaviti proces generiranja visokokvalitetnih označenih skupova podataka za modele strojnog učenja.

Tablicom 2.1. prikazane su neke od zajedničkih, ali i specifičnih funkcionalnosti postojećih programskih alata, kako slijedi:

- Vrste oznake - odnosi se na vrstu oznake koje se mogu primijeniti,
- Platforma temeljena na *web-u* - odnosi se na pristup alatu putem internet preglednika bez potrebe instalacije alata na računalo,
- Dostupna verzija za stolno računalo - odnosi se na dostupnost samostalne verzije alata za računalo, koja može pružiti određene prednosti poput poboljšane izvedbe ili izvanmrežnog pristupa,
- Prilagođeni atributi - odnosi se na mogućnost korisničkog dodavanja dodatnih prilagođenih informacija u postojeće oznake, kao što su rezultati pouzdanosti, *ID* instanci ili bilo koji drugi relevantni podaci,
- Organizacija slika - odnosi se na način na koji alat pomaže korisnicima u upravljanju njihovim slikama, poput grupiranja u projekte ili mape,
- Podrška za suradnju - odnosi se na mogućnost označavanje od strane više korisnika na istom projektu, bilo istovremeno ili asinkrono,
- Automatizacija/aktivno učenje - odnosi se na mogućnost integracija s modelima strojnog učenja za automatsko generiranje oznaka,

- Formati za izvoz podataka - različiti formati datoteka koje alat podržava prilikom izvoza komentara i označenih podataka,
- Integracija s okvirima strojnog učenja - odnosi se na mogućnost integracija s popularnim okvirima strojnog učenja kao što su *TensorFlow* i *PyTorch*,
- Programsko aplikativno sučelje (engl. *Application Programming Interface* – API) za upravljanje podacima - označava sadrži li alat API koje omogućava korisnicima programsku interakciju sa označenim skupovima podataka, omogućujući prilagođene tijekove rada i automatizaciju,
- Verzija i kontrola kvalitete:
    - Određivanje verzija - odnosi se na praćenje promjene u skupu podataka tijekom vremena, omogućujući usporedbu između različitih verzija,
    - Kontrola kvalitete (engl. *Quality control* - QC) - odnosi se na značajke koje osiguravaju točnost i dosljednost označavanja u skupu podataka.

Tablica 2.1. Prikaz usporedbe funkcionalnosti postojećih programskih alata

Funkcionalnost	***LabelMe***	***VIA***	***Labelbox***
Vrste oznake	Granični okviri, poligoni, semantička segmentacija, ključne točke, linije	Granični okviri, poligoni, ključne točke, regije, linije	Granični okviri, poligoni, ključne točke, linije
Platforma temeljena na *web-u*	Da	Da	Da
Dostupna verzija za stolna računala	Ne	Da	Ne
Prilagođeni atributi	Ne	Da	Da
Organizacija slike	Osnovna struktura mape	Projekti s prilagodljivim grupama	Projekti s mapama i skupovima podataka
Podrška za suradnju	Ne	Da	Da
Automatizacija/Aktivno učenje	Ne	Integracija s modelima strojnog učenja, podržava aktivno učenje	Integracija s modelima strojnog učenja, podržava aktivno učenje
Formati za pohranu podataka	JSON, XML i drugi	JSON, CSV i drugi	JSON, CSV i drugi
Integracija s okvirima strojnog učenja	Ne	*TensorFlow*, *PyTorch*	*TensorFlow*, *PyTorch*
API za upravljanje podacima	Ne	Da	Da
Verzija i kontrola kvalitete	Ne	Da	Da

3. IZRADA APLIKACIJE ZA OZNAČAVANJE PODATAKA ZA POTREBE IZGRADNJE MODELA ZASNOVANIH NA STROJNOM UČENJU

U ovom poglavlju dani su zahtjevi na aplikaciju te je opisan proces implementacije aplikacije kroz implementaciju sučelja za upravljanje projektima te implementaciju sučelja za označavanje objekata. Također, dan je opis implementacije automatske detekcije objekata kao i programski alati korišteni u procesu razvoja aplikacije.

3.1. Analiza zahtjeva na aplikaciju

Ovo poglavlje bavi se temeljnom analizom zahtjeva na aplikaciju. Ključne funkcije i značajke koje aplikacija treba sadržavati navedene su u nastavku.

- Izrada, konfiguracija i pohrana projekta

Osnovni zahtjev na aplikaciju je omogućiti kreiranje projekata od strane korisnika. Korisnik bi trebao imati mogućnost kreiranja različitih projekata, prilagođenih specifičnim zahtjevima. To uključuje definiranje naziva projekta i odabir lokacije na disku kao i mogućnost definiranja klasa objekata od interesa. Također, aplikacija bi trebala korisniku omogućiti odabir galerije slika koja je relevantna za projekt. Nakon kreiranja projekta, aplikacija treba generirati odgovarajuću mapu s nazivom projekta na odabranoj lokaciji na disku. Ova bi mapa trebala sadržavati odabranu galeriju slika i pridruženu konfiguracijsku datoteku koja sadrži sve relevantne parametre projekta koji će biti korišteni prilikom konfiguracije inicijalnih parametara sučelja za označavanje objekata.

- Vizualizacija slika podatkovnog skupa i rukovanje slikama

Aplikacija za označavanje objekata treba pružiti korisničko sučelje koje omogućuje vizualizaciju slika, uključujući označene objekte unutar svake prikazane slike. Korisnik bi trebao imati sposobnost rukovanja slikama unutar projektne galerije, odnosno, aplikacija treba omogućiti korisniku uvećanje prikaza slika radi preciznije vizualizacije objekta od interesa, kao i pregledavanje slika unutar galerije. Dodatno, aplikacija treba omogućiti korisniku proširenje galerije sa novima slikama. Ova ključna značajka omogućuje korisniku da pohrani nove slike u galeriju projekta, čime se osigurava prilagodljivost aplikacije tijekom vremena.

- Označavanje objekata i rukovanje oznakama

Aplikacija treba omogućiti korisniku crtanje graničnih okvira oko objekata od interesa, s mogućnošću podešavanja oznaka kroz prilagodbe dimenzija ili položaja. Fleksibilnost brisanja

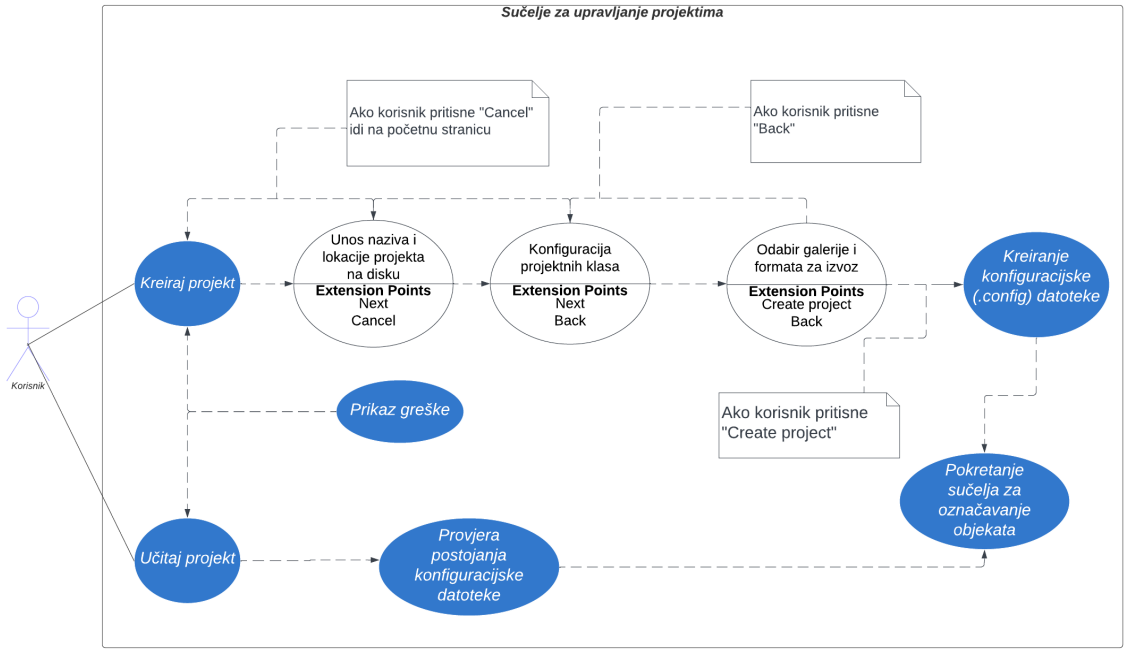
graničnih okvira jednako je važna za održavanje točnosti u procesu označavanja. U scenarijima koji uključuju automatsku detekciju objekata, aplikacija treba provjeriti svaku kreiranu oznaku. Ako se prilikom automatske detekcije pojave klase objekata ili granični okviri koji do tada nisu uključeni u proces označavanja, aplikacija mora osigurati njihovo daljnje korištenje.

- Izvoz i pohrana oznaka

Prilikom procesa označavanja, korisnik bi trebao imati mogućnost pohrane pojedinačnih oznaka u za to predviđenu bazu podataka. Po završetku označavanja cijelog ili određenog dijela skupa podataka, aplikacija mora omogućiti korisniku izvoz datoteka sa oznakama na disk u strukturiranom formatu.

3.2. Dizajn aplikacije

Ovo potpoglavlje prikazuje fazu dizajna aplikacije kroz *wireframe-ove* i *Use Case* dijagrame. Slika 3.1. prikazuje *Use Case* dijagram sučelja za upravljanje projektima. Veze prikazane punim linijama označavaju akcije koje korisnik može odabrati prilikom pokretanja sučelje. Odabirom „Kreiraj projekt“, korisniku se omogućuje konfiguracije projektnih parametara. Pritiskom na „*Next*“ i „*Back*“, korisnika se vodi kroz korake konfiguracije. Pritiskom na „Učitaj projekt“, korisnik odabire postojeći projekt te pokreće učitavanje sučelja za označavanje objekata.

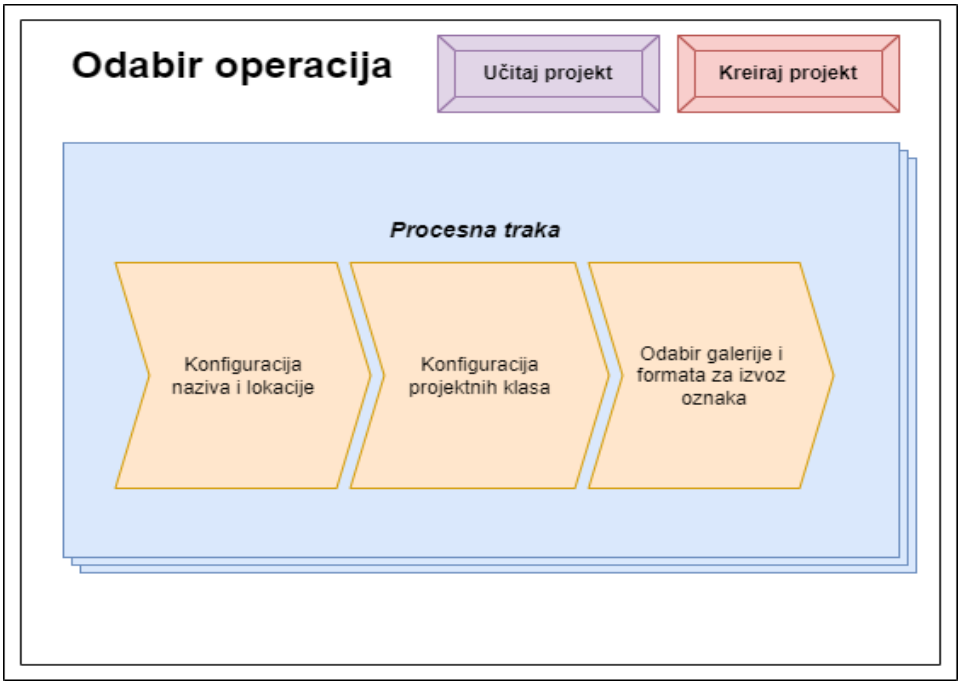


Slika 3.1. Prikaz Use Case dijagrama sučelja za upravljanje projektima

Slika 3.2. prikazuje *Use Case* dijagram sučelja za označavanje objekata. Veze prikazane punim linijama predstavljaju akcije alatne trake koje korisnik može odabrati prilikom procesa označavanje objekata u digitalnim slikama.

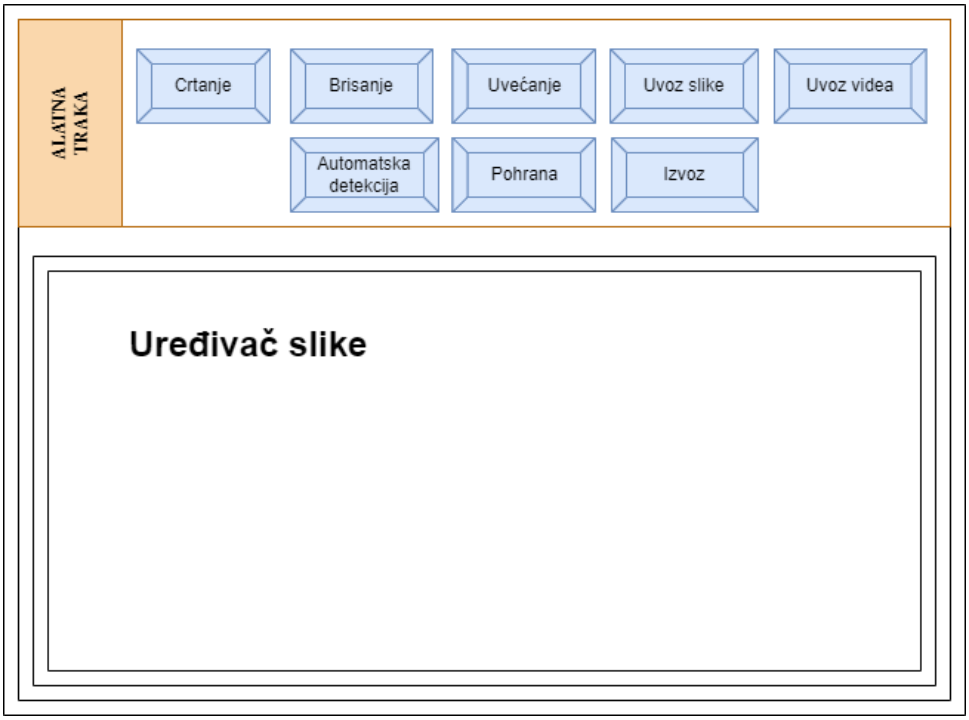
Slika 3.2. Prikaz Use Case dijagrama sučelja za označavanje objekata

Slika 3.3. prikazuje *wireframe* predloženog sučelja za upravljanje projektima.



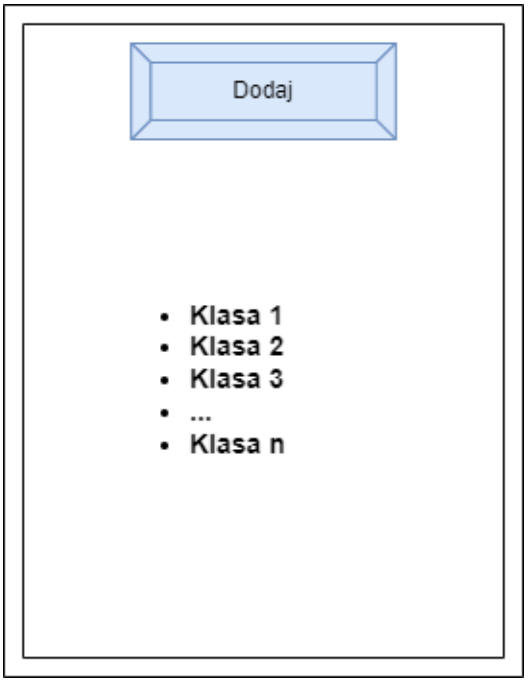
Slika 3.3. Prikaz wireframe-a sučelja za upravljanje projektima

Slika 3.4. prikazuje *wireframe* predloženog sučelja za označavanje objekata s dostupnim akcijama alatne trake.



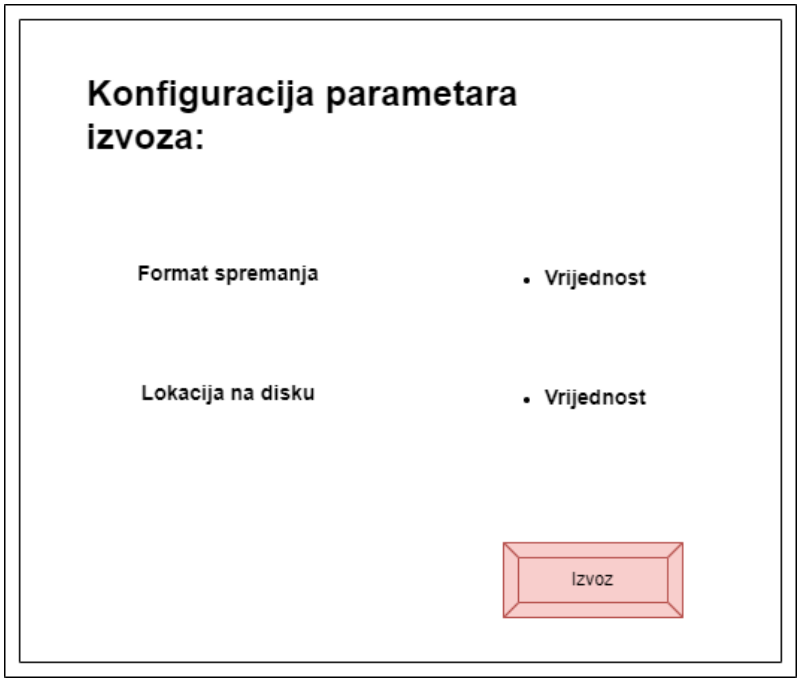
Slika 3.4. Prikaz wireframe-a sučelja za označavanje objekata

Slika 3.5. prikazuje *wireframe* modula za odabir klase označenog objekta.



Slika 3.5. Prikaz wireframe-a modula za odabir klase označenog objekta

Slika 3.6. prikazuje *wireframe* modula za izvoz datoteka s oznaka na disk.



Slika 3.6. Prikaz wireframe-a modula za konfiguraciju parametara izvoza datoteka s oznakama na disk

Kako bi se ostvarili postavljeni ciljevi na aplikaciju, za razvoj i implementaciju aplikacije odabran je *Qt* aplikacijski okvir i *Python* programski jezik. Odabir *Qt-a,* kao alata za razvoj aplikacije proizlazi iz mogućnosti stvaranja složenih i intuitivnih grafičkih sučelja, osiguravajući jednostavnost prilikom korištenja. Također, *Qt-ova* opsežna dokumentacija i podrška zajednice olakšavaju učinkovit razvoj i rješavanje problema dok korištenje *Python* biblioteka, kao što je *OpenCV* za manipulaciju slikama i *PyQt* za izgradnju grafičkih sučelja, značajno ubrzava proces razvoja.

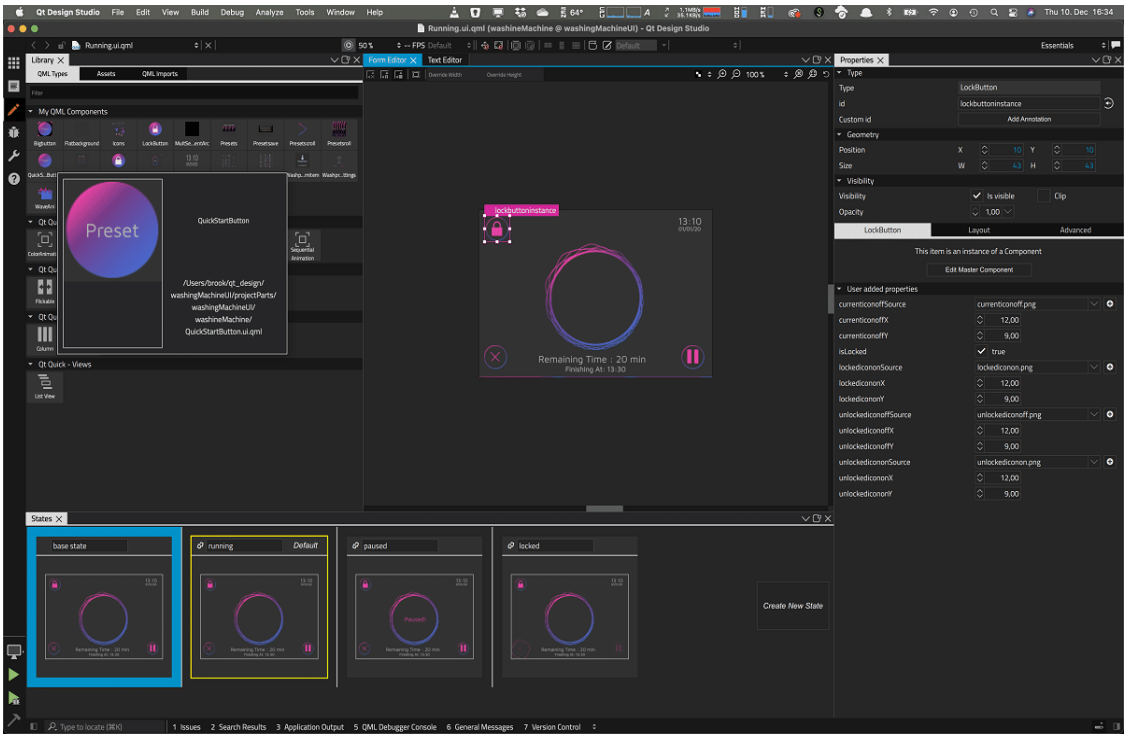
3.3. *Qt* i *Qt Designer* alati

Qt je aplikacijski okvir otvorenog koda, temeljen na C++ programskom jeziku, koji omogućuje razvoj programa visoke skalabilnosti i performansi. Također, *Qt* pruža širok raspon biblioteka i alata, što ga čini idealnim za izradu složenih aplikacija koje zahtijevaju odziv u stvarnom vremenu. Svestranost *Qt-a* dovela je do njegove popularnosti u raznim industrijama, uključujući strojno učenje, gdje pomaže u razvoju sučelja prilagođenih korisniku [22], [23]. *Qt Designer,* prikazan slikom 3.7. grafički je alat koji dolazi u paketu s *Qt* okvirom. Primarna svrha *Qt Designer-a* je olakšati dizajn grafičkih sučelja bez potrebe za detaljnim poznavanjem razvojnih programskih jezika. Korištenjem sučelja za povlačenje i ispuštanje, programeri mogu jednostavno organizirati komponente sučelja. Na taj način, značajno se ubrzava proces razvoja aplikacije i minimiziraju se pogreške programiranja.

Prednosti korištenja *Qt-a* i *Qt Designera* za razvoj aplikacije su razne, a neke od njih su:

1. Implementacija na više platformi - aplikacije kreirane korištenjem *Qt* okvira, kompatibilne su s više platformi, čime se eliminira programiranje specifično za pojedinu platformu,
2. Skalabilnost - *Qt* okvir lako se prilagođava potrebama projekta, koji mogu varirati od malih aplikacija do velikih i složenih programa,
3. opsežna zajednica i podrška,
4. *Qt Quick* za brzu izradu prototipa - *Qt Quick*, deklarativni jezik i okvir, olakšava brzu izradu prototipova modernih i vizualno privlačnih korisničkih sučelja. Uz animacije, prijelaze i vizualne efekte, programeri mogu brzo napraviti prototip sučelja, što dovodi do bržih iteracija i poboljšanog korisničkog iskustva,
5. Integracija sa značajkama izvorne platforme - *Qt* omogućuje integraciju sa značajkama izvorne platforme, omogućujući programerima pristup hardveru uređaja sistemskim API-jima i mogućnostima specifičnim za platformu,

6. Opsežna podrška za platformu - *Qt-ove* mogućnosti proširuju se izvan aplikacija za stolna računala i također uključuju podršku za mobilne aplikacije te ugrađene sustave,
7. Komercijalne i opcije otvorenog koda - *Qt* nudi komercijalne i opcije otvorenog koda licenciranja, pružajući razvojnim stručnjacima i organizacijama fleksibilnost odabira.



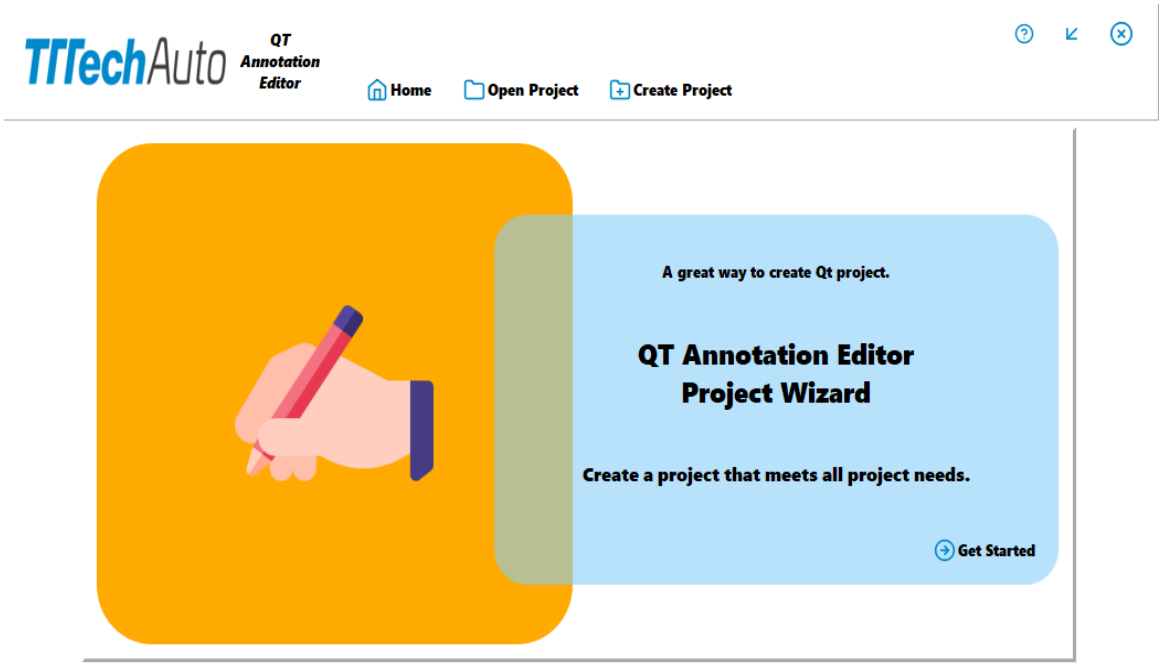
Slika 3.7. Sučelje Qt Designer grafičkog alata

PySide6 jedna je od poveznica između *Qt* aplikacijskog okvira i *Python-a* te se koristi kao alat koji omogućuje programerima da iskoriste *Qt-ove* mogućnosti unutar *Python* aplikacija. Jedna od najznačajnijih prednosti *PySide6* je njegova sposobnost stvaranja aplikacija kompatibilnih s više platformi [24]. Također, *PySide6* omogućuje primjenu biblioteka *Qt* modula, pružajući programerima sveobuhvatan skup alata i značajki, a neke od mogućnosti su:

- stvaranje prilagođenih *widgeta*,
- korištenje uređivača svojstava radi modifikacije vizualnih aspekata aplikacije,
- korištenje signala i utora (engl. *signal-slot functions)* radi olakšanja komunikacije između korisničkog sučelja aplikacije i osnovnog koda,
- dinamično prilagođavanje dimenzija korištenjem značajki upravljanja izgledom (engl. *layout management features*),
- mogućnost izravnog testiranja,
- mogućnost rukovanja resursima.

3.4. Implementacija sučelja za upravljanje projektima

Sučelje za upravljanje projektima implementirano je u skladu sa zahtjevima aplikacije, pružajući korisniku mogućnost definiranja projektnih parametara sukladno specifičnim korisničkim zahtjevima. Sučelje za upravljanje projektima omogućuje korisniku unos naziva projekta, odabir lokacije projekta na disku, konfiguriranje klasa objekata od interesa, odabir i pohranu galerije sa slikama i određivanje formata za izvoz datoteka s oznakama na disk. Slika 3.8. prikazuje konačan dizajn sučelja za upravljanje projektima.



Slika 3.8. Dizajn sučelja za upravljanje projektima

Sučelje se sastoji od *Qt* komponenti opisanih u nastavku.

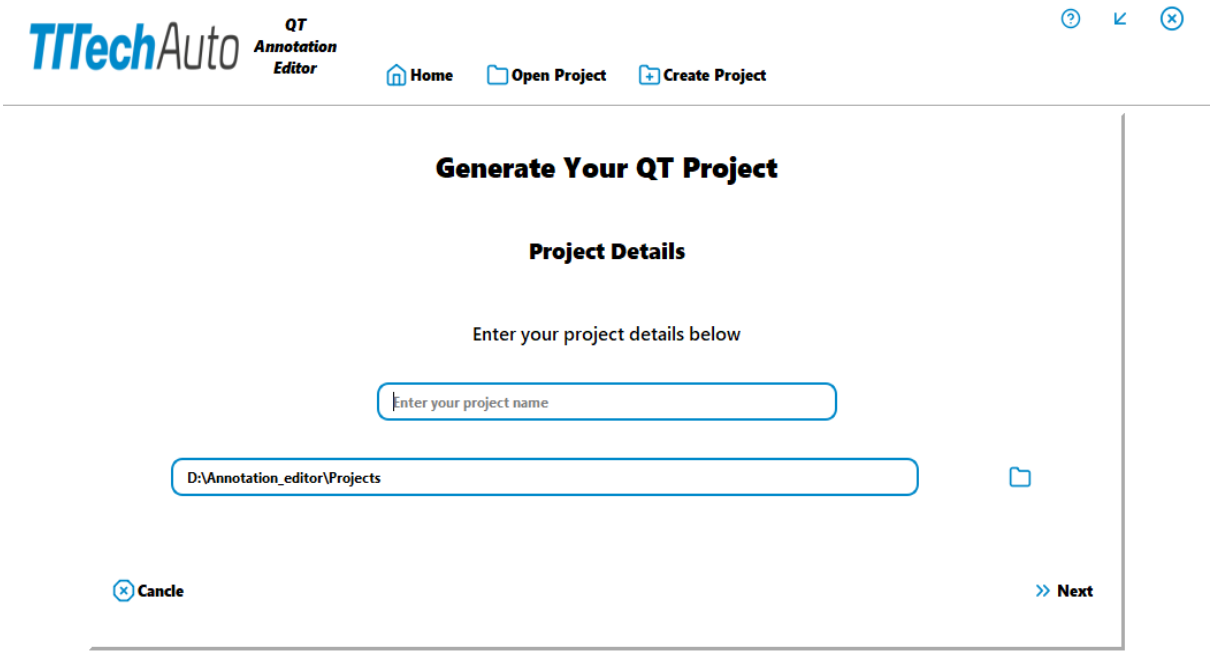
- *QFrame* - temeljna klasa u *Qt* okviru koja pruža jednostavan spremnik za druge *widgete*. Služi kao osnovna klasa za razne *widgete* nalik okvirima, nudeći značajke poput obruba, pozadine i ispune. Može se koristiti za grupiranje *widgeta* zajedno i pružanja vizualnog odvajanja unutar korisničkog sučelja.
- *QWidget* - osnovna klasa za sve UI elemente u *Qt-u*. Predstavlja pravokutno područje na zaslonu i djeluje kao spremnik za druge *widgete*. Omogućuje osnovne funkcije kao što su rukovanje događajima, upravljanje fokusom i renderiranje.

- *QStackedWidget* - spremnik je koji prikazuje samo jedan od svojih podređenih *widgeta* u isto vrijeme, slažući ih jedan na drugi. Često se koristi za implementaciju sučelja s više stranica, gdje se korisnici kreću između različitih stranica prikazivanjem/skrivanjem pojedinačnih podređenih *widgeta*.
- *QLabel* – koristi se za prikaz statičnog teksta ili slika. Obično se koristi za pružanje opisnog teksta ili naslova unutar korisničkog sučelja.
- *QLineEdit* – korisnicima omogućuje polje za unos i uređivanje teksta u jednom retku. *QLineEdit* podržava razne vrste unosa, kao što je običan tekst, lozinke (za siguran unos) i maskirani unos (npr. za brojeve kreditnih kartica).
- *QPushButton* – implementacija je gumb na kojeg korisnik može kliknuti u korisničkom sučelju. Široko se koristi za pokretanje radnji ili funkcija, što ga čini bitnom komponentom za interakciju.
- *QRadioButton* – predstavlja radio gumb koji se može odabrati. Radio gumbi se često koriste u grupama, a korisnici mogu odabrati samo jednu opciju iz skupa međusobno isključivih izbora. Obično se koriste za odabir preferencija ili scenarije višestrukog izbora.
- *QListView* – služi za prikaz stavki u okomitom ili vodoravnom rasporedu. Uobičajeno se koristi za predstavljanje podataka u obliku popisa, gdje se svaka stavka može odabrati ili s njom komunicirati.
- *QProgressBar* – predstavlja indikator napretka. Vizualno prikazuje napredak zadatka ili operacije, pružajući korisnicima povratne informacije o trenutnom stanju dovršenosti [25].

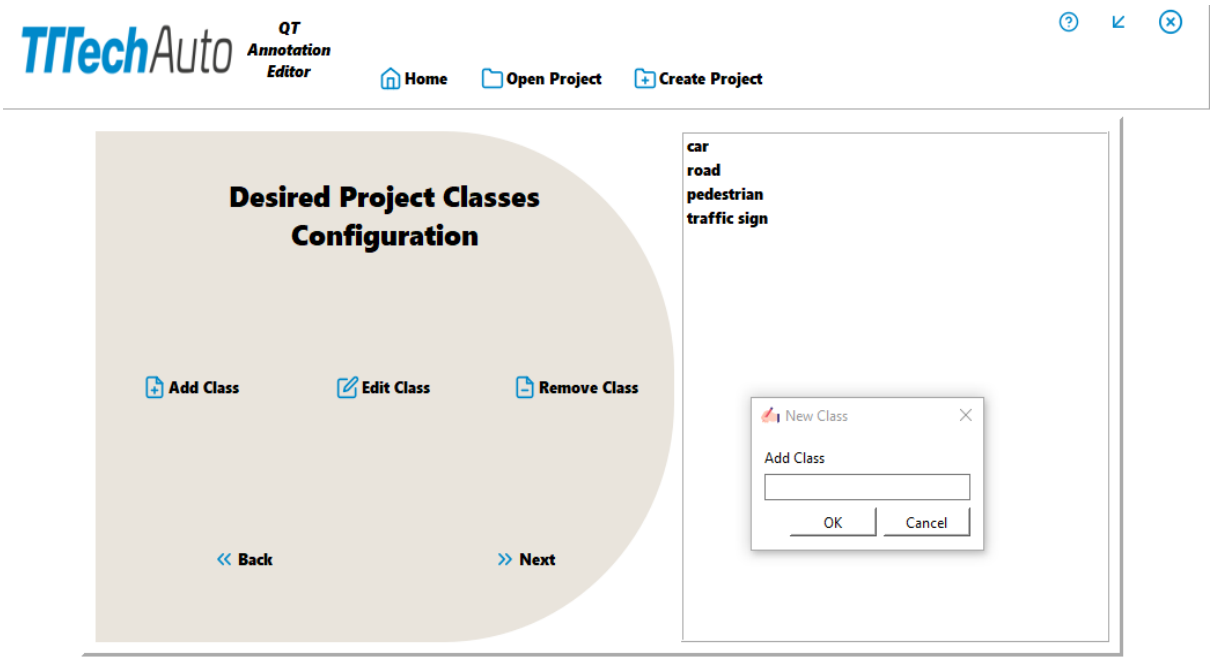
Jedna od temeljnih funkcionalnosti kreiranog sučelja je pružiti mogućnost odabira naziva projekta i odabira željene projektne lokacije na disku od strane korisnika. Koristeći komponentu *QStackedWidget*, sučelje za upravljanje projektima, dijeli se na nekoliko intuitivnih *QWidget* komponenti. Slika 3.9. prikazuje *QWidget* koji integracijom komponenti *QLabel* i *QLineEdit*, nudi korisniku mogućnost unosa naziva projekta kao i odabira lokacije projekta na disku. Glavna zadaća aplikacije prilikom ovog procesa je osigurati jedinstvenost projektnog naziva te ispravnu projektnu lokaciju.

Kako bi se zadovoljile različite potrebe projekta, sučelje nudi korisniku mogućnost konfiguriranja projektnih klasa. Slika 3.10. prikazuje *QWidget* koji integracijom komponenti *QLineEdit, QPushButton* i *QListView* nudi korisniku mogućnost unosa i prilagodbe proizvoljnog broja klasa objekata u skladu s potrebama korisničkog projekta. Na ovaj način korisniku se nudi mogućnost

prilagodbe projekta tako da odgovara individualnim preferencijama i ciljevima. Glavna zadaća aplikacije prilikom ovog procesa je osigurati barem jednu klasu objekta od interesa.

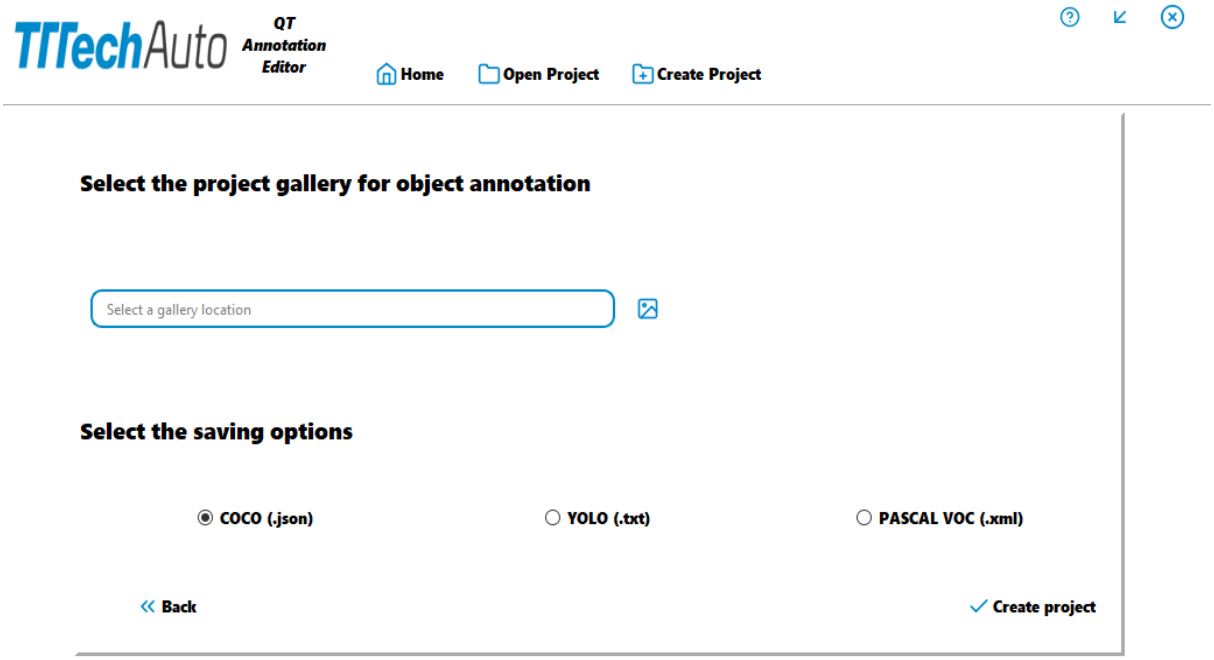


Slika 3.9. Primjer konfiguracije projektnih parametara, unos projektnog naziva te odabir lokacije projekta na disku



Slika 3.10. Primjer konfiguracije projektnih klasa uz prikaz korisničkog dijaloga za dodavanje nove instance projektne klase.

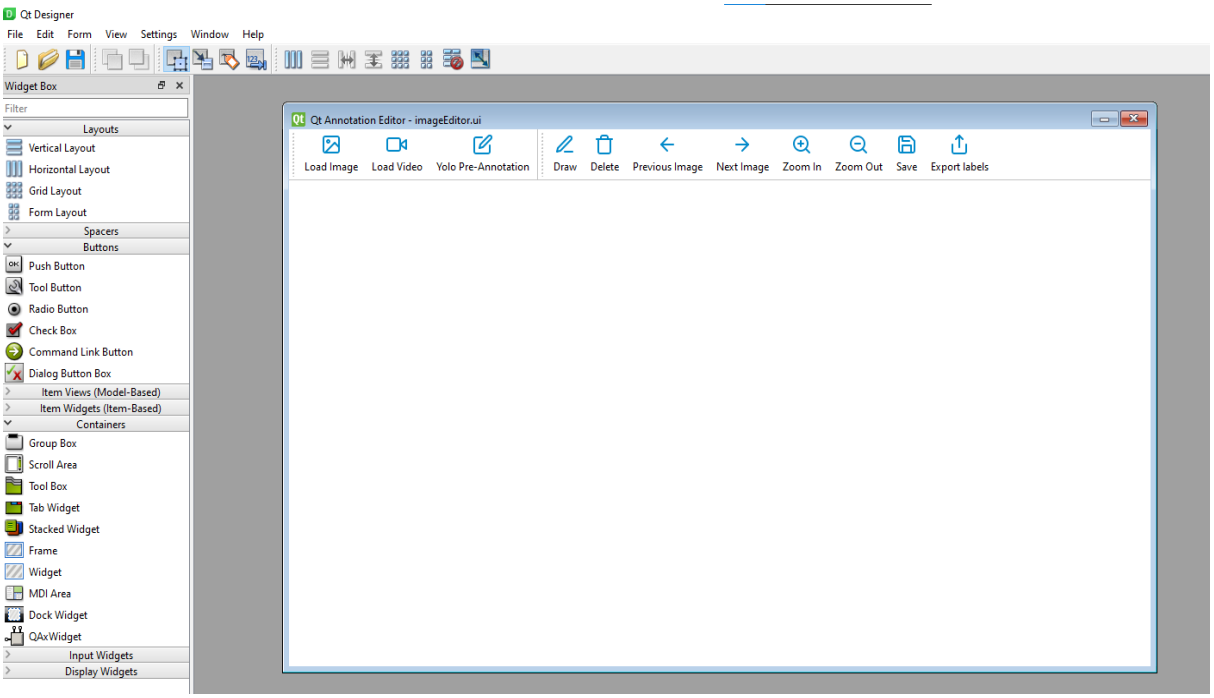
Osim toga, sučelje za upravljanje projektima omogućuje korisniku odabir galerije koja će prilikom procesa označavanja služiti kao repozitorij za slike i pohranu datoteka s kreiranim oznakama. Slika 3.11. prikazuje *QWidget* koji integracijom komponenti *QLineEdit*, *QPushButton* i *QRadioButton*, korisnicima omogućuje odabir projektne galerije te formata za izvoz datoteka s oznakama na disk. Glavna zadaća aplikacije prilikom ovog procesa je osigurati ispravnost formata slika unutar galerije projekta te odabir formata za izvoz datoteka s oznakama na disk. Uspješnim kreiranjem projekta, aplikacija izrađuje mapu projekta koja pohranjuje mapu galerije i konfiguracijsku datoteku koja sadrži sve definirane parametre projekta.



Slika 3.11. Primjer konfiguracije projektnih parametara, odabir projektne galerije te formata za izvoz datoteke s oznakama na disk

3.5. Implementacija sučelja za označavanje objekata

Sučelje za označavanje objekata implementirano je u skladu sa zahtjevima aplikacije, nudeći korisniku skupinu alata za precizno označavanje objekata unutar digitalnih slika. Sučelje omogućuje korisniku prikaz slika iz projektne galerije, kreiranje, uređivanje, brisanje te vizualizaciju kreiranih oznaka, kao i mogućnost za izvoz datoteka s oznakama na disk u različitim formatima. Slika 3.12. prikazuje dizajn sučelja za označivanje objekata u digitalnim slikama.

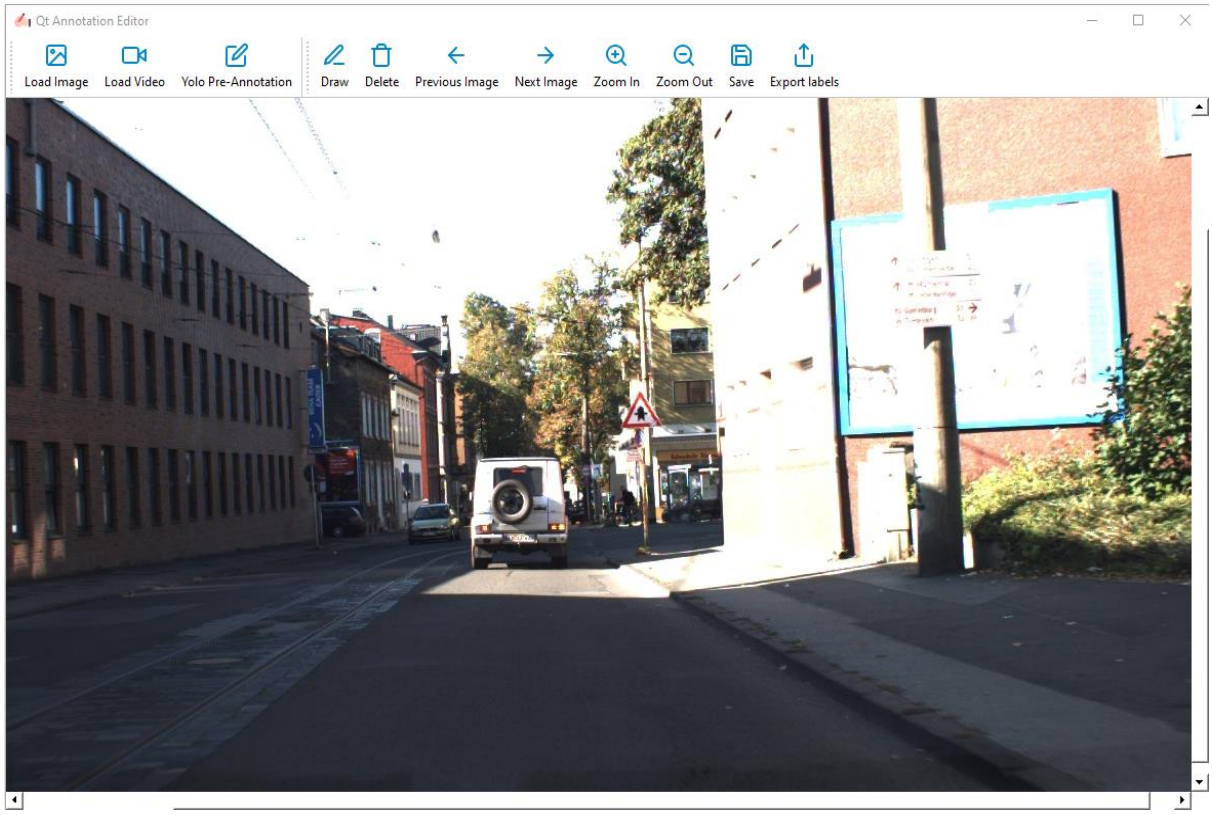


Slika 3.12. Dizajn sučelja za označivanje objekata u digitalnim slikama

Temeljna funkcionalnost sučelja za označavanje objekata u slikama, odnosi se na mogućnost korisničke kontrole nad galerijom slika unutar projekta. Sučelje treba podržavati izmjenu slika, omogućujući korisnicima jednostavno kretanje kroz galeriju. Osim toga, sučelje treba omogućiti povećavanje i smanjivanje dimenzija slika kako bi se povećala preciznost označavanja na mjestima gdje je to potrebno. Realizacija zadanih funkcionalnosti ostvaruje se integracijom *Qt* komponenti kao što su:

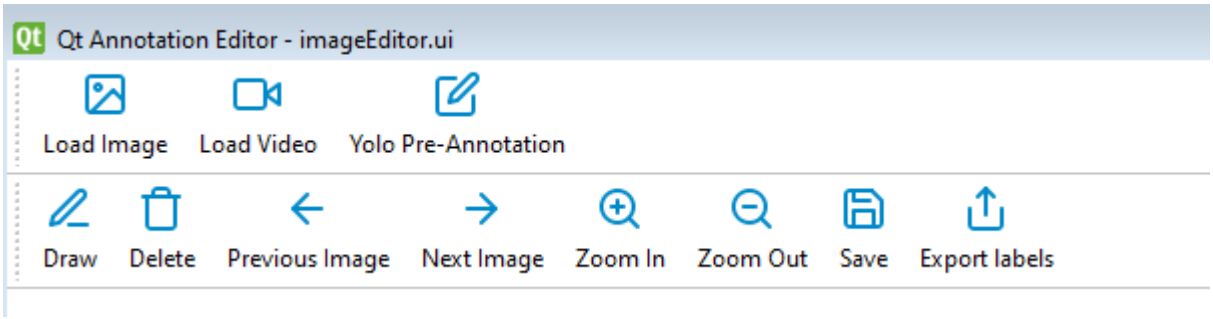
- *QGraphicsView* - pruža pogled na 2D scenu, djelujući kao okvir za prikaz gdje se grafičke stavke prikazuju i njima se upravlja. *QGraphicsView* se može smatrati platnom na kojem se crtaju grafički elementi. *QGraphicsView* podržava značajke kao što su pomicanje, uvećanje i rotacija, što ga čini izvrsnim izborom za prikaz velike ili složene grafike uz mogućnost jednostavne navigacije i interakcije.
- *GraphicsScene* - klasa spremnika u *Qt* okviru koja drži i upravlja grafičkim stavkama prikazanim u *QGraphicsViewu*. Djeluje kao grafikon scene, organizirajući i koordinirajući pozicioniranje i interakciju različitih grafičkih stavki. Klasa nudi mogućnosti upravljanja geometrijom te rukovanje događajima za grafičke elemente unutar granica [25].

Slika 3.13. prikazuje vizualizaciju slike korištenjem *GraphicsScene* u komponenti *QGraphicsView*.



Slika 3.13. Primjer vizualizacije slike projektne galerije korištenjem komponente QGraphicsView

Pojedine funkcionalnosti sučelja, realizirane su korištenjem mogućnosti *Qt* okvira, poznatog kao alatna traka koja korisniku omogućuje skup alata za označavanje i rukovanje slikama. Alatna traka omogućuje korištenje *QAction-a* (akcije koje korisnici mogu pokrenuti). Svaka *QAction* odgovara određenom alatu ili funkciji, kao što je crtanje graničnih okvira, pohrana oznaka ili uvećanje. Povezivanjem *QAction-a* s odgovarajućim elementima korisničkog sučelja kao što su gumbi, korisniku je omogućen intuitivan pristup alatima prilikom realizacije projektnih ciljeva. Slika 3.14. prikazuje implementirane funkcionalnosti alatne trake.



Slika 3.14. Realizacija funkcionalnosti alatne trake

Korištenjem opcija „*Load Image*“ i „*Load Video*“ alatne trake, korisniku je omogućeno dodati pojedinačne slike i video u galeriju projekta. Navedene funkcionalnosti osiguravaju kontinuirano proširivanje projektne galerije sa slikama. Glavna zadaća aplikacije prilikom ovog procesa je osigurati konzistentnost prilikom promjene naziva dodanih slika, prateći praksu označavanja slika unutar galerije u kojoj se slike pohranjuju.

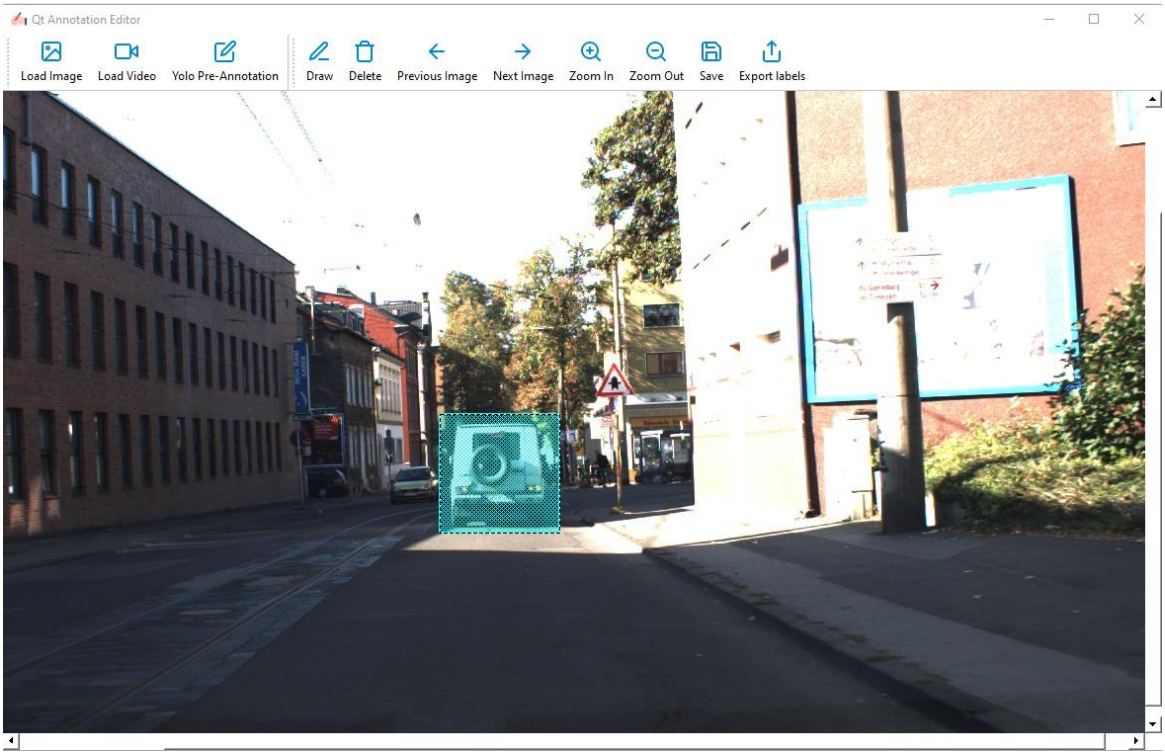
Korištenjem opcije „*Draw*“ alatne trake, korisniku je omogućeno kreiranje oznaka crtanjem graničnog okvira oko objekta od interesa. Slika 3.15. prikazuje dio klase oznake. Granični okvir oznake, nastaje pritiskom miša u komponenti *QGraphicsView* što označava početnu koordinatu graničnog okvira. Akcija povlačenja miša, mijenja dimenzije (visinu i širinu) dok otpuštanje miša, označava kraj procesa kreiranja graničnog okvira. Kreirane koordinate graničnog okvira pohranjene su u atributu oznake *rectangle*, koji se može ažurirati korištenjem funkcije *itemChange()*. Također, ovako definirana oznaka, nudi mogućnost dodjele klase označenog objekta od interesa, korištenjem funkcije *setClass()* i atributa *className*.

Korištenjem opcije „*Delete*“ alatne trake ili pritiskom *Delete* na tipkovnici, korisniku je omogućeno brisanje prethodno kreiranog graničnog okvira oznake, a time i brisanje same oznake. Glavna zadaća aplikacije prilikom ovog procesa je vizualizirati sliku, odnosno, prethodno označeni objekt nakon brisanja oznake unutar *QGraphicsView-a* te informirati korisnika o uspješnosti postupka brisanja. Slika 3.16. prikazuje odabir kreirane oznake za postupak brisanja.

Linija ***Kod***

```
1:     class RectItem(QGraphicsRectItem):
2:       def __init__(self, rect, parent=None):
3:         super().__init__(rect, parent)
4:         self.setPen(QPen(Qt.red, 2, Qt.SolidLine))
5:         self.setFlags(QGraphicsItem.ItemIsSelectable
6:               | QGraphicsItem.ItemIsMovable
7:               | QGraphicsItem.ItemIsFocusable
8:               | QGraphicsItem.ItemSendsGeometryChanges
7:               | QGraphicsItem.ItemSendsScenePositionChanges)
8:         self.setAcceptHoverEvents(True)
9:         self.className = None
10:        self.rectangle = [0, 0, 0, 0]
11:        self.hovering = False
12:      def setClass(self, className):
13:        self.className = className
14:      def printRectangle(self):
15:                print(self.className, self.rectangle)
16:      def itemChange(self, change, value):
17:        ...
           if change == QGraphicsItem.ItemPositionChange and
18:        self.scene():
19:        ...
20:        return super().itemChange(change, value)
21:      def mouseMoveEvent(self, event):
22:          ...
23:        # Update the dimensions of the rectangle
24:        self.rectangle = [x, y, newWidth, newHeight]
25:        self.setRect(newRect)
```

Slika 3.15. Prikaz dijela klase oznake



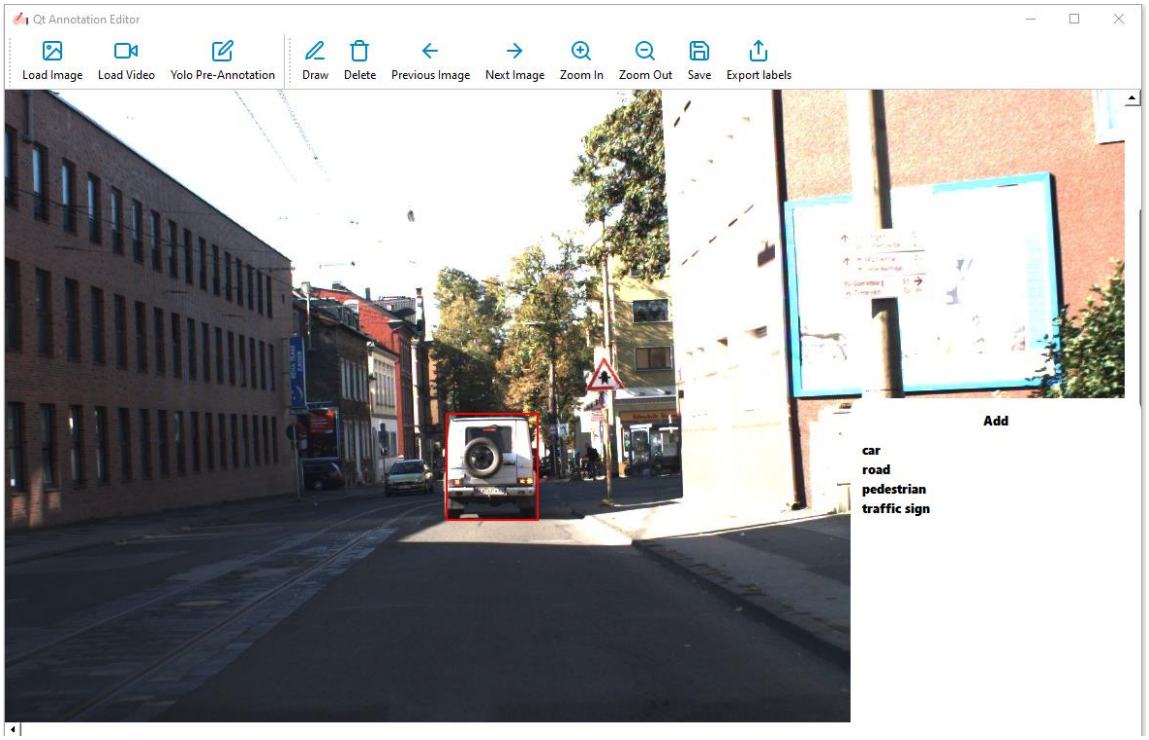
Slika 3.16. Realizacija funkcionalnosti odabira kreirane oznake

Kako bi se osiguralo jednostavno rukovanje prilikom odabira klase označenog objekta za vrijeme procesa označavanja, kreiran je aplikacijski modul *ClassDialog*. Slika 3.17. prikazuje dio klase aplikacijskog modula za odabir klase označenog objekta. Po završetku procesa označavanja, korisnik odabire klasu označenog objekta od interesa koristeći dijaloški okvir prikazan slikom 3.18. Odabirom odgovarajuće klase, prethodno definirane prilikom kreiranja projekta te pritiskom na gumb „*Add*“, službeno se završava proces označavanja objekta. Aplikacija, korištenjem funkcije *setLabel()* i funkcije *setClass()*, dodjeljuje vrijednost atributu *className* te pohranjuje vrijednost koordinata kreiranog graničnog okvira oznake.

Linija ***Kod***

```
1:      class ClassDialog(QWidget):
2:        def __init__(self, class_items, rectangle):
3:            self.class_items = class_items
4:            self.rectangle = rectangle
5:            QWidget.__init__(self)
6:            self.ui = Ui_ClassDialog()
7:            self.ui.setupUi(self)
8:            for classItem in self.class_items:
7:              self.ui.class_config_view.addItem(classItem)
8:            self.ui.add_btn.clicked.connect(self.setLabel)
9:        def setLabel(self):
10:           ...
11:           self.rectangle.setClass(className.text())
12:           ...
13:           self.close()
```

Slika 3.17. Prikaz dijela klase aplikacijskog modula za odabir klase (kategorije) označenog objekta



Slika 3.18. Primjer korištenja aplikacijskog modula za odabir klase označenog objekta s prethodno definiranim klasama: „car“, „road“, „pedestrian“, „traffic sign“

Kako bi se osiguralo jednostavno rukovanje kreiranim oznakama bez obzira na format izvoza oznaka, kreiran je aplikacijski modul za rukovanje oznakama *AnnotationMenager*. Slika 3.19. prikazuje dio klase aplikacijskog modula za rukovanje oznakama.

Linija ***Kod***

```
1:    class AnnotationMenager:
        def __init__(self, projectPath, savingFormat,
2:      categories):
3:        self.projectPath = projectPath
4:        self.savingFormat = savingFormat
5:        self.categories = categories
6:        self.annotations = {}
          self.annotationsExportPath =
7:        os.path.join(self.projectPath, "annotations.config")
8:        self.msg = MessageHandler()
7:      def annotationInit(self):
8:        ...
9:      def createAnnotationFile(self):
10:       ...
        def addAnnotations(self, imageID, imageName,
11:     imageHeight, imageWidth, rectItems):
12:       ...
13:     def addNewCategory(self, newValue):
14:       ...
15:     def deleteAnnotation(self, deleteIndex, imageID):
16:       ...
        def normalizeBoundingBox(self, boundingBox,
17:     imageHeight, imageWidth):
18:       ...
19:       return annotation
20:     def annotationCheck(self, imageID):
21:       ...
22:     def getAnnotations(self):
23:       return self.annotations
24:     def export(self, path):
25:       if self.savingFormat == ".txt":
26:       ...
27:       elif self.savingFormat == ".json":
28:       ...
29:       else:
30:       ...
31:       # Storing the annotations dictionary
```

Slika 3.19. Prikaz dijela klase aplikacijskog modula za rukovanje oznakama

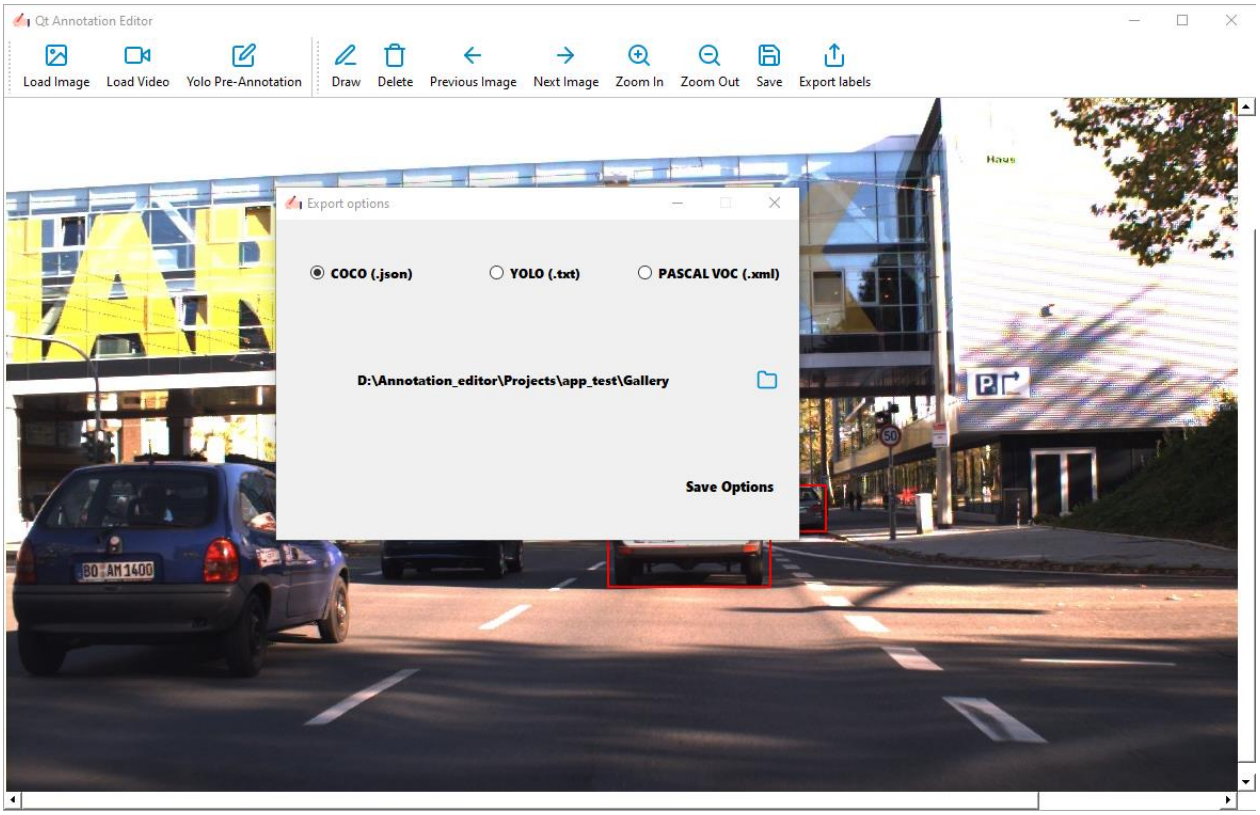
Korištenjem opcije „*Save*“ alatne trake, korisniku je omogućena pohrana kreiranih oznaka u radni rječnik oznaka *annotations* za svaku pojedinu sliku u skupu slika koje pripadaju projektu. Aplikacija, korištenjem modula za rukovanje oznakama, vodi brigu o pohrani svake oznake na prikazanoj slici na način da vrši provjeru postojanosti oznake u radnom rječniku oznaka, korištenjem funkcije *annotationCheck()*. U slučaju da oznaka ne postoji, korištenjem funkcije *addAnnotations()* aplikacija pohranjuje kreiranu oznaku. U slučaju promjene dimenzija oznaka, aplikacija vrši ažuriranje vrijednosti graničnih okvira promijenjenih oznaka te obavještava korisnika o uspješnosti pohrane. Kako bi se osiguralo jednostavna konfiguracija parametara za izvoz datoteka s oznakama na disk, kreiran je aplikacijski modul *ExportDialog*. Slika 3.20. prikazuje dio klase aplikacijskog modula za konfiguraciju parametara izvoza datoteka s oznakama na disk.

Linija ***Kod***

```
1:    class ExportDialog(QWidget):
2:      def __init__(self, annotationMenager, editor):
3:        self.annotationMenager = annotationMenager
4:        self.editor = editor
5:        QWidget.__init__(self)
6:        self.ui = Ui_ExportDialog()
7:        self.msg = MessageHandler()
8:        self.ui.setupUi(self)
7:        self.exportInit()
8:        self.ui.save_btn.clicked.connect(self.saveOptions)
          self.ui.change_location_btn.clicked.connect
9:          (self.changeSavingLocation)
10:     def exportInit(self):
11:       ...
12:     def saveOptions(self):
13:       self.checkSavingFormat()
          self.editor.exportPath = self.ui.saving_location.text
14:       ()
15:       self.annotationMenager.export(self.editor.exportPath)
          self.msg.raiseInfo("The labels have been successfully
16:       stored!")
17:     def checkSavingFormat(self):
18:       ...
19:     def changeSavingLocation(self):
20:       ...
```

Slika 3.20. Prikaz dijela klase aplikacijskog modula za konfiguraciju parametara izvoza datoteka s oznakama na disk

Korištenjem opcije „*Export labels*“ alatne trake, sučelje prikazuje dijaloški okvir, prikazan slikom 3.21. koji korisniku omogućuje promjene prethodno definiranog formata te promjenu odredišne lokacija za izvoz datoteka s oznakama na disk. Nakon odabira željenih parametara pohrane, aplikacija uz pomoć modula za rukovanje oznakama (korištenjem funkcije *export()*) obrađuje oznake i izvozi datoteke s oznakama u željenom formatu na odabranu lokaciju.



Slika 3.21. Primjer korištenja dijaloškog okvira za odabir parametara izvoza datoteka s oznakama na disk

3.6. Implementacija automatske detekcije objekata u slikama

Sa stajališta aplikacije, proces automatske detekcije podrazumijeva inicijalno postavljanje oznaka oko objekata od interesa s ciljem pojednostavljivanja procesa označavanja. YOLOv5 ((engl. *You Only Look Once*) verzija 5) je detektor objekata korišten prilikom implementacije funkcionalnosti automatske detekcije objekata s klasama kao što su: automobil, pješak, znak stop, semafor itd. [26]

Ključne karakteristike i značajke YOLOv5 su:

- Arhitektura - YOLOv5 se sastoji od tri glavna dijela:

1. Okosnica (engl. *Backbone*) - odnosi se na temeljne slojeve konvolucijske neuronske mreže (engl. *Convolutional Neural Network* - CNN) koji su odgovorni za izdvajanje značajki ulazne slike.
2. Vrat (engl. *Neck*) - odnosi se skup slojeva za kombiniranje značajki slike i njihovo prosljeđivanje u slojeve za detekciju.
3. Glava (engl. *Head*) - odnosi se na skup slojeva odgovornih za detekciju na temelju značajki izdvojenih od slojeva *Backbone* i, ako je primjenjivo, dodatno pročišćenih kroz među slojeve *Neck*.

Zajednička točka svih arhitektura detekcije objekata je da će značajke ulazne slike bit komprimirane kroz ekstraktor značajki *Backbone* i zatim proslijeđen u detektor objekata (*Detection Neck* i *Detection Head*) [27], [28]. *Neck* u ovom slučaju radi kao agregator značajki, koji ima zadatak kombinirati, miješati i pripremiti značajke nastale u slojevima *Backbone* za nadolazeći korak detekcije koji se provodim u slojevima *Head*. Razlika je ovdje što je *Head* odgovoran za detekciju zajedno s klasifikacijom za svaki granični okvir.

- Varijante modela - YOLOv5 dolazi u različitim varijantama modela, označenim slovima (YOLO5s, YOLO5m, YOLO5l i YOLO5x), s različitim razinama veličine i složenosti. Varijanta YOLO5s je strukturno najjednostavnija i najbrža, dok je YOLO5x strukturno najsloženija i najtočnija, ali sporija u smislu brzine zaključivanja.
- Unaprijed istrenirane težine - YOLOv5 često je unaprijed istreniran na skupovima podataka velikih razmjera, kao što je COCO, kako bi naučio opće značajke otkrivanja objekata prije finog podešavanja na prilagođene skupove podataka.
- Jednostavnost korištenja - YOLOv5 dizajniran je da bude jednostavan za korištenje, s jednostavnim i intuitivnim API za laku integraciju u aplikacije.
- Javno dostupan - YOLOv5 dostupan je na GitHubu, a održava ga Ultralytics, osiguravajući rastuću zajednicu suradnika i korisnika.

Detektor objekata YOLOv5 se u aplikaciju učitava koristeći biblioteku *PyTorch* te pritiskom na opciju „*Yolo Pre-Annotate*“ alatne trake [29]. Nakon učitavanja modela, kursor mijenja simbol u čekanje, označavajući početak automatiziranog procesa detekcije opisanog koracima:

1. Priprema unosa - prije pokretanja postupka detekcije, ulazna slika se priprema promjenom veličine na fiksnu ulaznu veličinu i normaliziranjem vrijednosti piksela kako bi se osigurala kompatibilnost sa zahtjevima YOLOv5 detektora.

2. Postupak zaključivanja - tijekom postupka zaključivanja, pripremljena ulazna slika prolazi kroz učitani YOLOv5 detektor. Detektor obrađuje sliku te se dobiva rezultat detekcije objekata.
3. Pružanje relevantnih informacija o otkrivenim objektima - YOLOv5 pruža informacije o otkrivenim objektima na ulaznoj slici. Ove informacije uključuju:
    - Koordinate graničnog okvira - za svaki otkriveni objekt, model predviđa koordinate graničnog okvira (x, y, širina, visina) u odnosu na ćeliju mreže u kojoj je objekt lokaliziran. Ove koordinate označavaju položaj i veličinu otkrivenog objekta na slici.
    - Predviđanja klase - uz koordinate graničnog okvira, model predviđa vjerojatnosti klase za svaki granični okvir. Ove vjerojatnosti predstavljaju vjerojatnost da objekt pripada određenoj klasi (npr. osoba, automobil, pas itd.).
    - Ocjene pouzdanosti - model dodjeljuje vjerojatnost pojavljivanja objekta u svakom predviđenom graničnom okviru, predstavljajući uvjerenje modela u točnost detekcije.
4. Ne-maksimalno potiskivanje (engl. *Non-maximum Suppression* - NMS) - nakon predviđanja graničnog okvira, vjerojatnosti klase i rezultata pouzdanosti, primjenjuje se NMS kako bi suzbile višestruke detekcije i detekcije niske pouzdanosti. NMS osigurava da je svaki objekt predstavljen samo jednim graničnim okvirom s najvećom ocjenom pouzdanosti, smanjujući broj suvišnih detekcija.
5. Izlazni rezultati - konačni izlaz detektora je popis otkrivenih objekata, od kojih je svaki predstavljen graničnim okvirom s pripadajućim predviđanjima klase i rezultatima pouzdanosti [30].

Slika 3.22. prikazuje realizaciju automatske detekcije korištenjem funkcije *preAnnotate(),* definirane unutar izvršnog modula (*Editor*) sučelja za označavanje objekata.

Linija ***Kod***

```
1:      class Editor(QMainWindow):
2:        def __init__(...):
3:         ...
4:        def preAnnotate(self):
5:          self.setCursor(Qt.WaitCursor)
6:          annotations = []
            model = torch.hub.load("ultralytics/yolov5",
7:          "yolov5s")
            device = torch.device("cuda" if
8:          torch.cuda.is_available() else "cpu")
9:          model.to(device)
10:         results = model(self.currentImage)
11:         objects = results.pandas().xyxy[0]
12:         for _, obj in objects.iterrows():
13:           className = obj["name"]
              xmin, ymin, xmax, ymax = obj[["xmin", "ymin",
14:           "xmax", "ymax"]]
15:           boundingBox = [xmin, ymin, width, height]
16:           rectItem = self.getRectItem(boundingBox, className)
17:           self.annotationMenager.addAnnotations(...)
18:         self.imageDisplay()
```

Slika 3.22. Prikaz dijela funkcije za automatsko označavanje objekata

Korištena funkcija *preAnnotate()* u sklopu prikazane strukture obavlja niz zadataka: postavlja kursor u stanje čekanja, inicijalizira listu za detektirane objekte, učitava YOLOv5 detektor, konfigurira uređaj da koristi programski okvir za proširenje mogućnosti ubrzanja grafičke procesorske jedinice (engl. *Compute Unified Device Architecture* - CUDA), obrađuje trenutnu sliku, dohvaća podatke o detektiranim objektima, a zatim iterira kroz te objekte. Za svaki objekt izdvajaju se naziv njegove klase i koordinate graničnog okvira, te se stvara oznaka *RectItem* kako je opisano poglavljem 3.5. Time se proces automatske detekcije završava, a korisniku se omogućuje rukovanje oznakama i vizualizacija rezultata detekcije na slici koja se obrađuje.

4. VERIFIKACIJA APLIKACIJE ZA OZNAČAVANJE PODATAKA ZA POTREBE IZGRADNJE MODELA ZASNOVANIH NA STROJNOM UČENJU

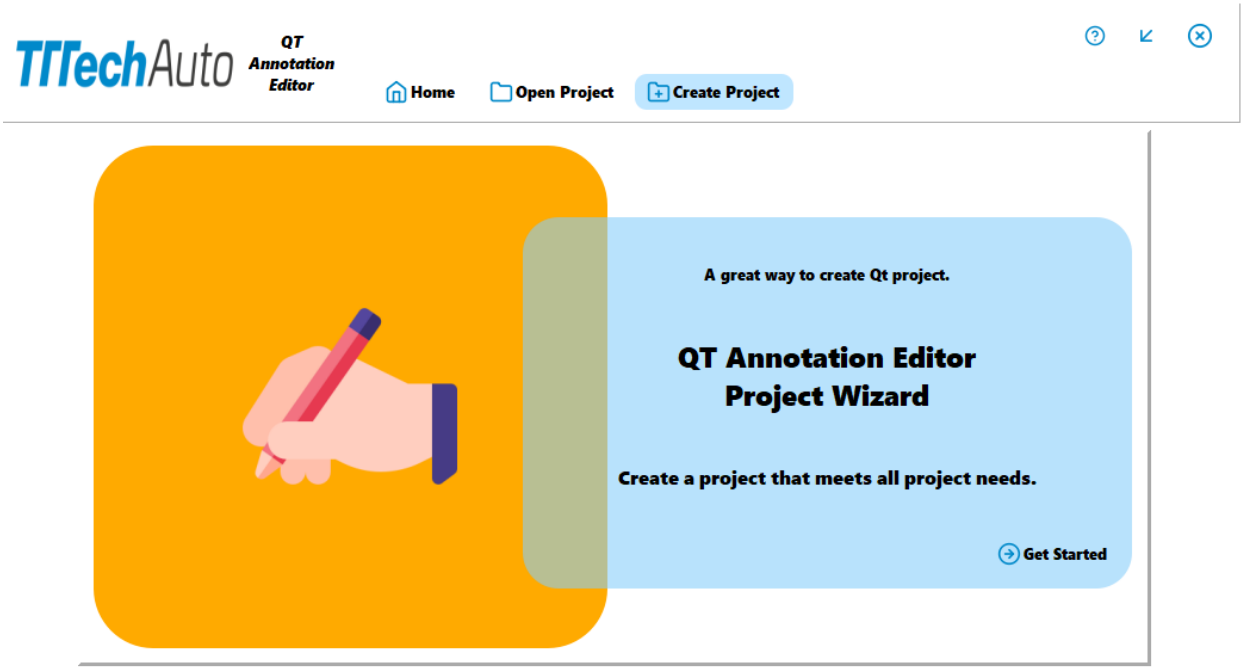
Verifikacija sučelja za upravljanje projektima, provedena je kreiranjem novog projekta uz testne parametre naziva projekta, proizvoljan odabir projektne lokacije na disku te uz konfiguraciju proizvoljnog broja projektnih klasa. Galerija odabrana prilikom kreiranja, nazvana je „*Gallery_1*“ i sastoji se od skupa koji broji trideset slika u PNG formatu. Format za izvoz datoteke s oznakama prilikom kreiranja projekta, postavljen je na vrijednost COCO JSON. Cilj procesa verifikacije je provjeriti ispravnost implementacije funkcionalnosti sučelje za upravljanje projektima. Uspjeh procesa verifikacije sučelja za upravljanje projektima, definiran je kreiranjem mape projekta na odabranoj projektnoj lokaciji na disku, kao i pohranom odabrane galerije slike uz izgradnju konfiguracijske datoteke s definiranim projektnim parametrima.

Verifikacija sučelja za označavanje objekata, provedena je na projektu kreiranom od strane korisnika uz korištenje sučelja za upravljanje projektima. Cilj procesa verifikacije je provjeriti ispravnost implementacije funkcionalnosti sučelje za označavanje objekata. Proces verifikacije sastoji se od verifikacije funkcionalnosti implementiranih uz pomoć alatne trake kako je opisano poglavljem 3.5. Uspjeh procesa verifikacije sučelja za označavanje objekata, definiran je vizualizacijom kreiranih oznaka, rukovanjem pojedinim oznakama te izvozom datoteka s oznakama u odabranom formatu na disk.

Verifikacija automatske detekcije objekata za cilj ima osigurati ispravnost vizualizacije i pohrane oznaka detektiranih objekata, korištenjem detektora YOLOv5. Uspjeh procesa verifikacije automatske detekcije definiran je kreiranjem oznaka detektiranih objekata, njihovom vizualizacijom te uspješnom pohranom u radni rječnik oznaka.

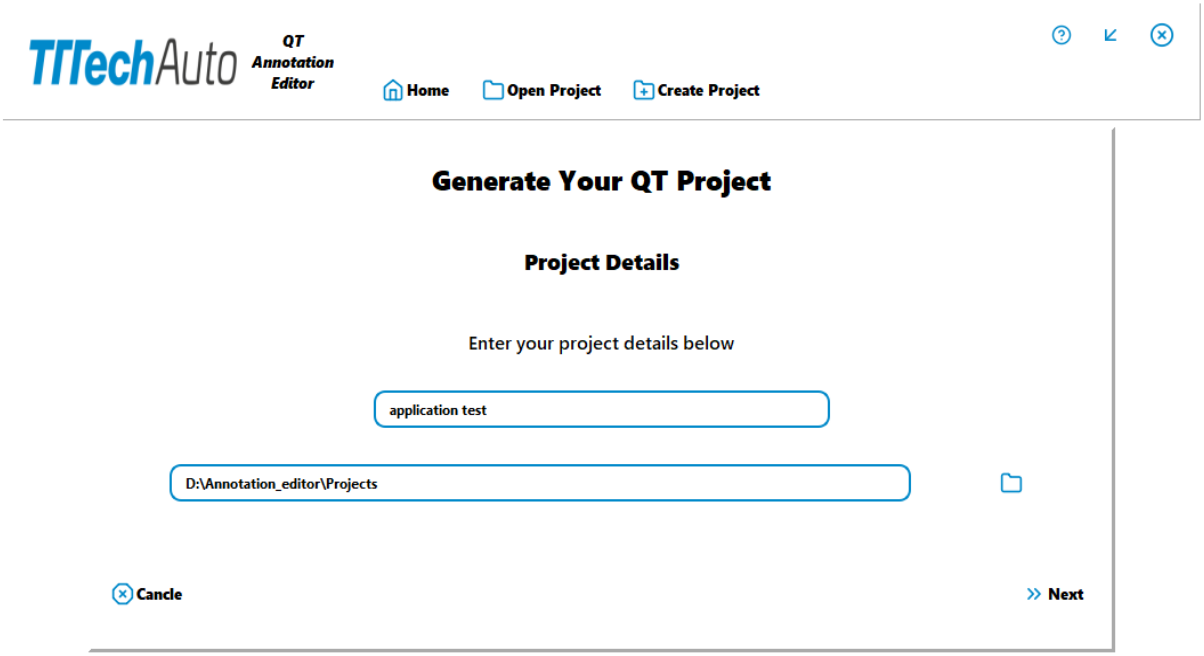
4.1. Verifikacija sučelja za upravljanje projektima

Prilikom pristupa sučelju za upravljanje projektima, korisniku se nude mogućnosti kreiranja novog projekta ili učitavanja već postojećeg projekta. Slika 4.1. prikazuje mogućnost odabira projektnih radnji.



Slika 4.1. Verifikacija odabira projektnih radnji

Prilikom kreiranja novog projekta, od korisnika se traži unos naziva projekta te odabir željene projektne lokacije. Za odabir lokacije, sučelje korisniku pruža intuitivan pristup navigaciji, a prilagođeno tekstualno polje, osigurava unos naziva i jedinstveni identitet projekta. Slika 4.2. prikazuje verifikaciju unosa projektnog naziva „*application test*“. Projektna lokacija predefinirano je postavljena unutar aplikacijske mape „*Projects*“. Upotrebom *Qt* alata prilikom promjene predefinirane vrijednosti projektne lokacije, onemogućen je neispravan odabir lokacije na disku.

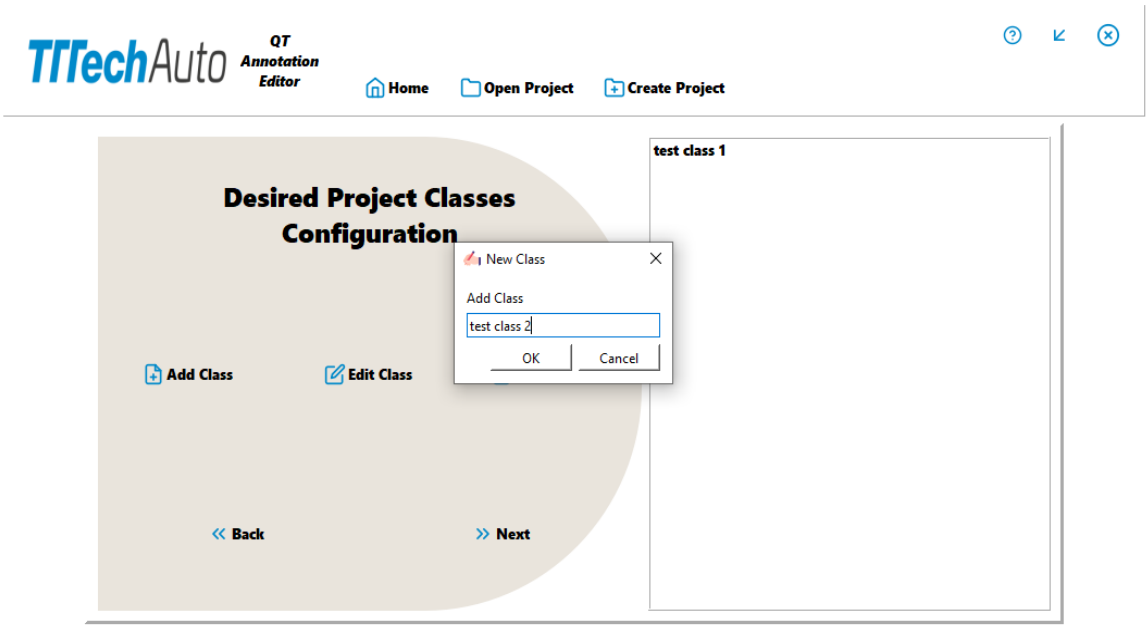


Slika 4.2. Verifikacija unosa naziva projekta

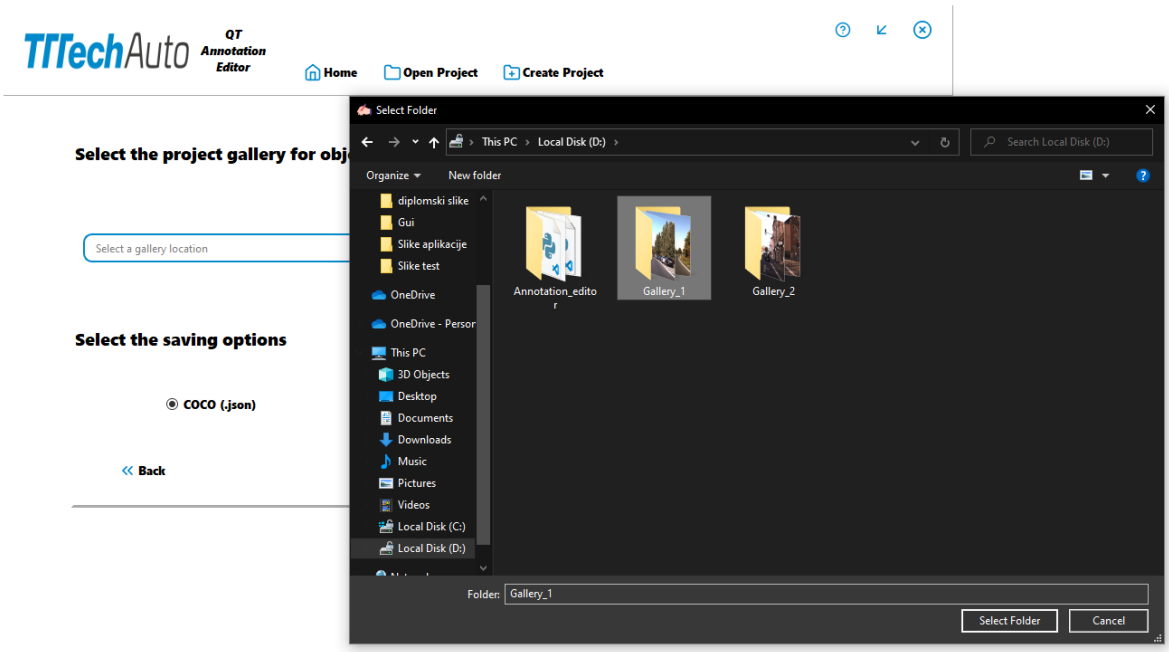
Kako bi projekt prilagodili specifičnim potrebama, korisnik može kreirati i urediti proizvoljan broj klasa koje definiraju objekte od interesa unutar projekta, kako je opisano poglavljem 3.4. Slika 4.3. prikazuje dijaloški okvir za unos projektne klase „*test class 1*“ i „*test class 2*“.

Korisnik je ovlašten dodatno prilagoditi svoj projekt odabirom projektne galerije te formata za izvoz datoteke s oznakama na disk. U slučaju odabira mape koja ne predstavlja galeriju, aplikacija kreira praznu mapu u koju korisnik naknadno (korištenjem alata unutar sučelja za označavanje objekata) može dodati sadržaj. Slika 4.4. prikazuje dijaloški okvir za odabir galerije projekta.

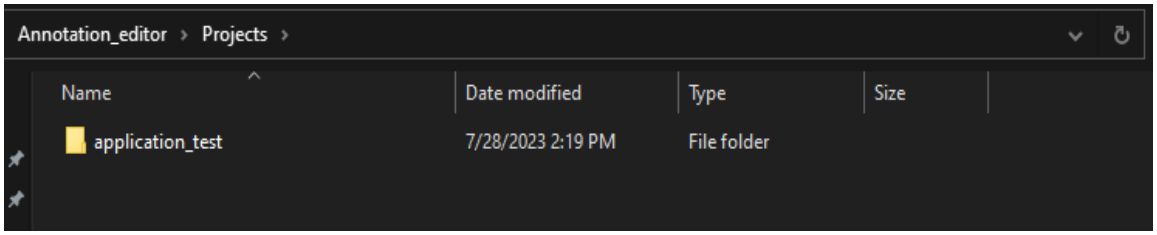
Uz sve osigurane unose, aplikacija će stvoriti namjensku mapu projekta. Unutar ove mape, pohranjuje se mapa galerije sa sadržajem i generira se konfiguracijska datoteka. Također, u slučaju izostanka konfiguracije bilo kojeg parametra projekta, korisnika se obavještava o pogrešci. Slike 4.5, 4.6 i 4.7. prikazuju uspješnu provedbu postupka verifikacije kreiranja projekta kroz kreiranje mape projekta, pohranu mape galerije te generiranja konfiguracijske datoteke.



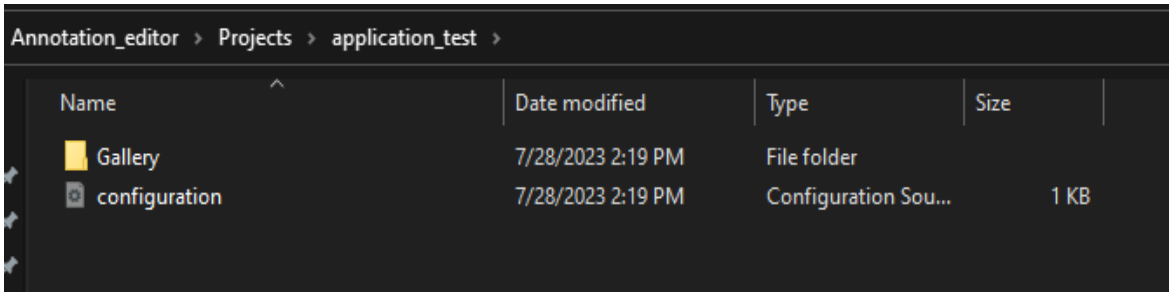
Slika 4.3. Verifikacija prikaza dijaloškog okvira za unos projektne klase



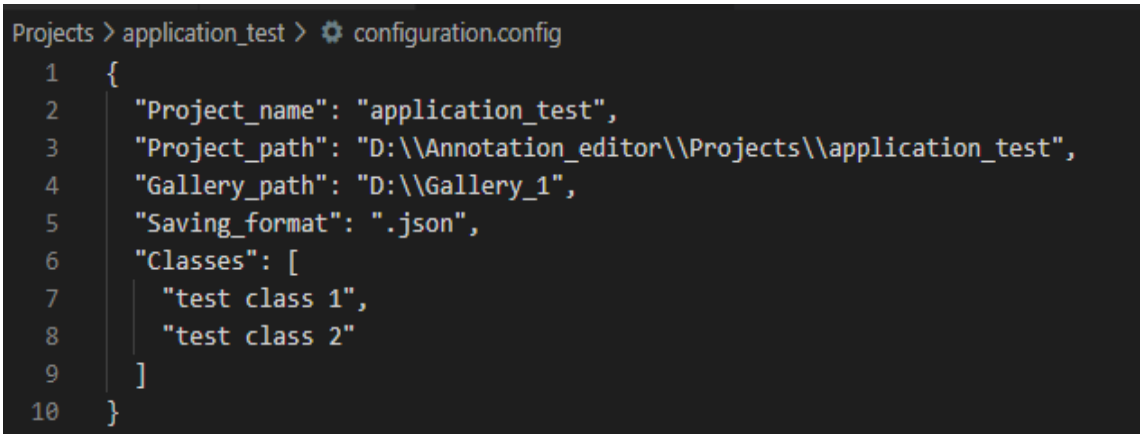
Slika 4.4. Verifikacija prikaza dijaloškog okvira za odabir galerije projekta



Slika 4.5. Verifikacija uspješnog kreiranja mape projekta na odabranoj projektnoj lokaciji



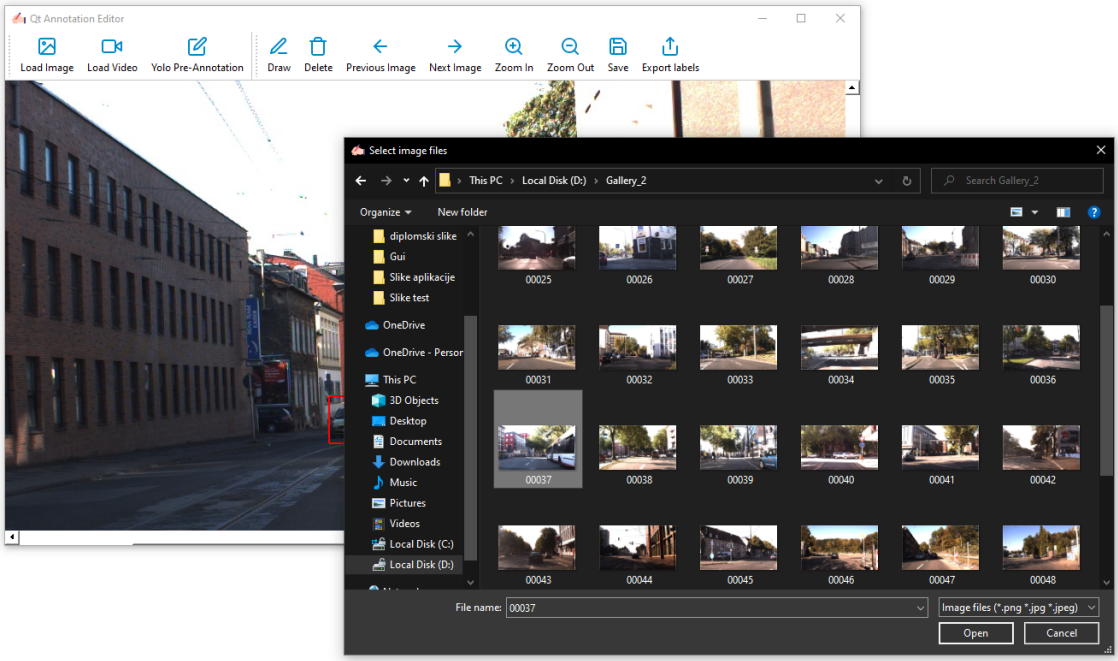
Slika 4.6. Verifikacija uspješne pohrane galerije „Gallery_1“ unutar mape projekta



Slika 4.7. Prikaz sadržaja generirane konfiguracijske datoteke

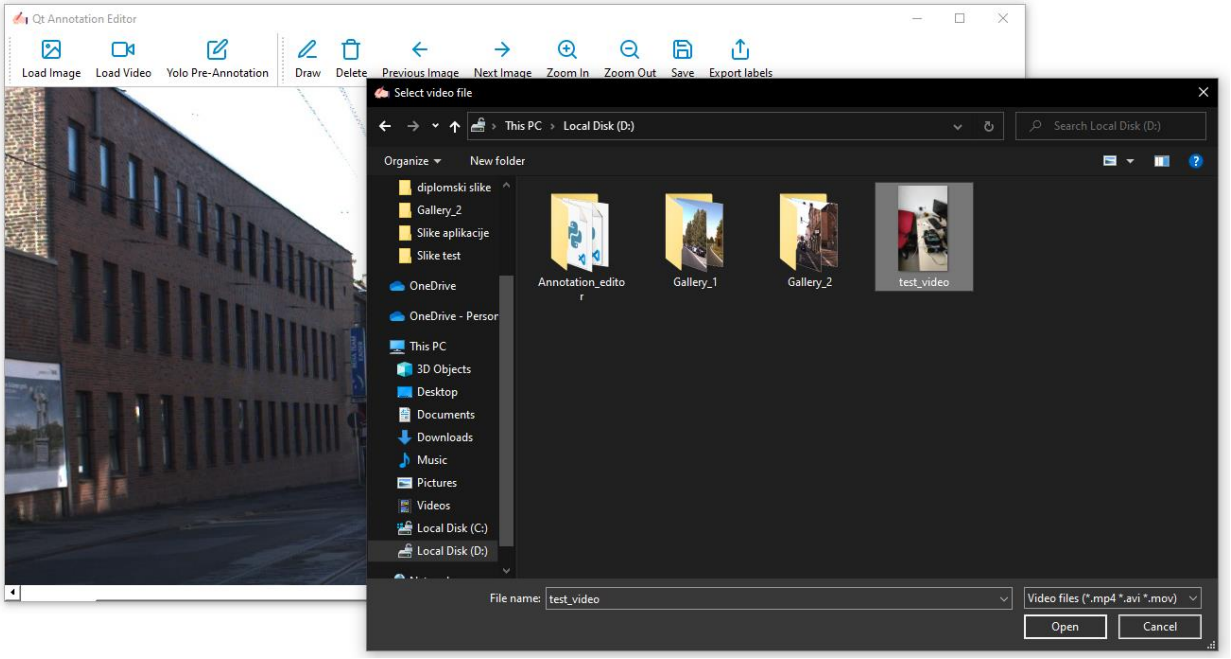
4.2. Verifikacija sučelja za označavanje objekata

Pokretanjem sučelja za označavanje objekata, korisniku se prikazuje početna slika u galeriji. U slučaju da je galerija prazna, korisnika se obavještava o nemogućnosti prikaza slike. Aplikacija za svaku prikazanu sliku vrši provjeru postoje li već kreirane oznake na nekim od objekata? U slučaju da oznake postoje, iste se vizualiziraju korisniku s mogućnošću uređivanja. Alatna traka služi korisniku za odabir radnji nad prikazanom slikom. Pritiskom na „*Load Image*“, korisniku se prikazuje dijaloški okvir s navigacijom i filtrom formata (PNG ili JPG) za dodavanje slika u postojeću projektnu galeriju na način opisan poglavljem 3.5. Slika 4.8. prikazuje dijaloški okvir s navigacijom i filtrom formata za dodavanje slika u postojeću projektnu galeriju.



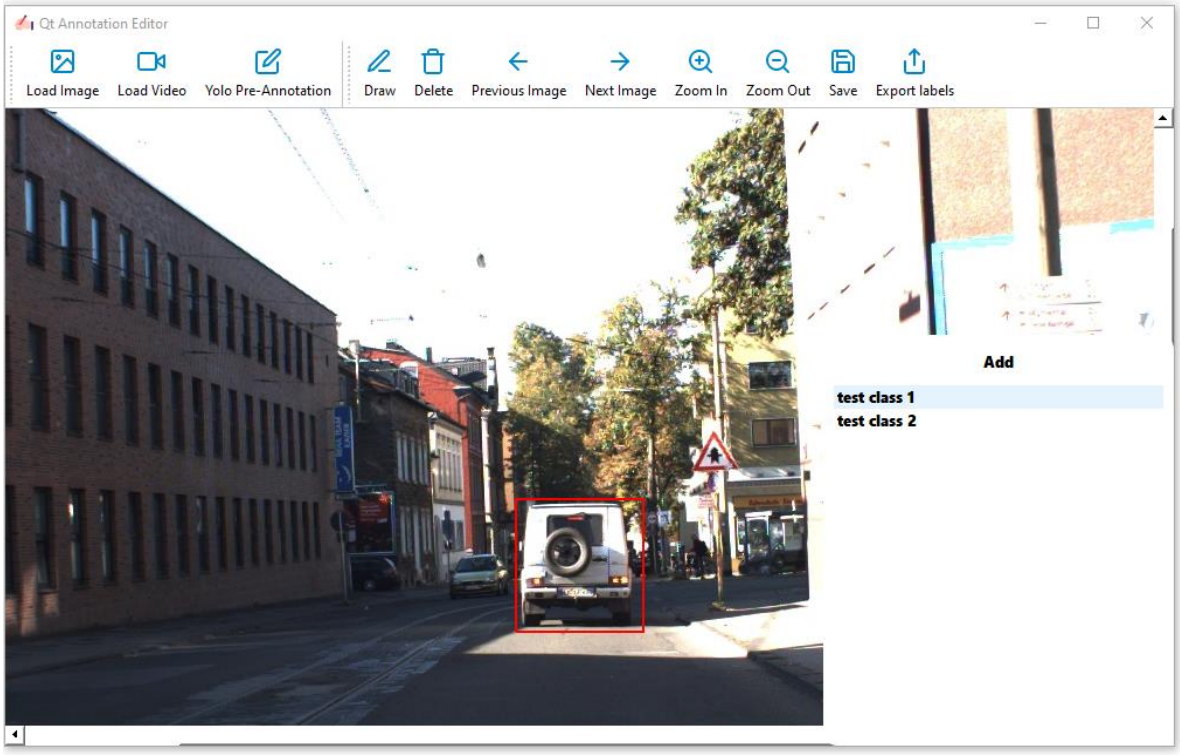
Slika 4.8. Verifikacija prikaza dijaloškog okvira s navigacijom i filtrom formata za dodavanje slike u postojeću projektnu galeriju

Pritiskom na „*Load Video*“, korisniku se prikazuje dijaloški okvir s navigacijom i filtrom formata (engl. *MPEG-4 Part 14* - MP4, *Audio Video Interleave* - AVI, *QuickTime multimedia file format* - MOV) za dodavanje videa zapisa. Aplikacija će za odabrani video provesti postupak rastavljanja video na pojedinačne slike te će za svaku dodanu sliku izvršiti promjenu naziva na način opisan poglavljem 3.5. U slučaju dodavanja nečitljivog videa, aplikacija korisnika obavještava o pogrešci. Slika 4.9. prikazuje dijaloški okvir s navigacijom i filtrom formata za dodavanje video zapisa.

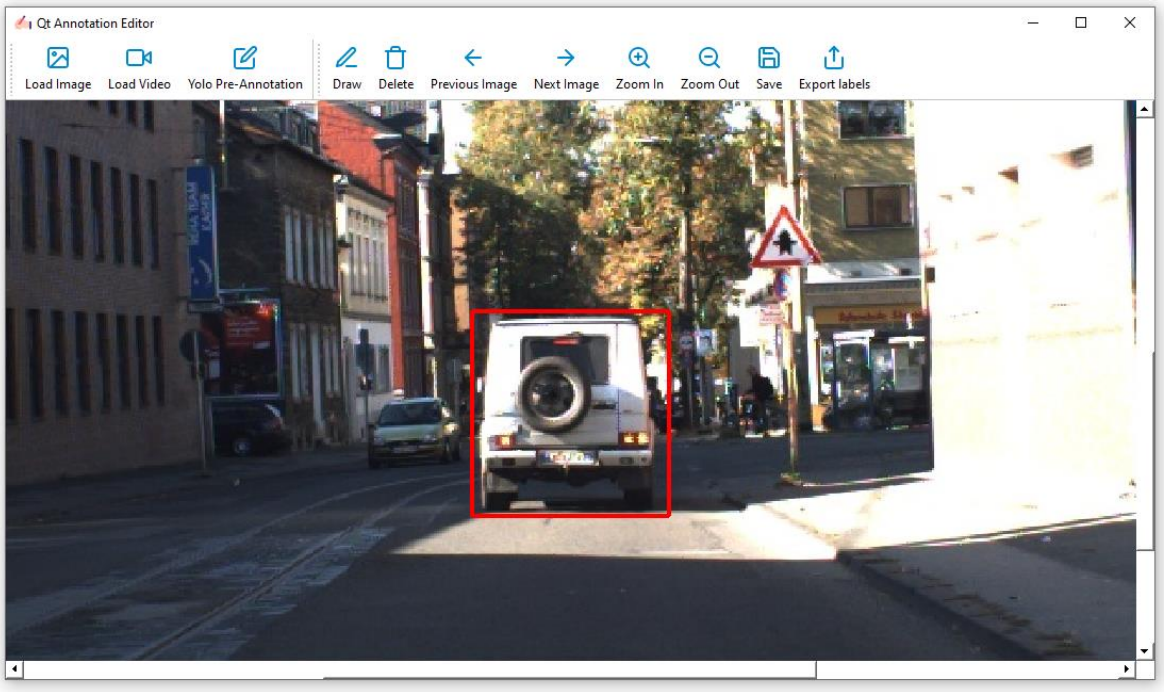


Slika 4.9. Verifikacija prikaza dijaloškog okvira s navigacijom i filtrom formata za dodavanje video zapisa

Pritiskom opcija „*Next Image*“ ili „*Previous Image*“ na alatnoj traci, korisniku je omogućena promjena slike koja se prikazuje u grafičkoj sceni. Pritiskom na opciju „*Draw*“, korisniku se nudi mogućnost kreiranja oznake nad objektom od interesa, na način opisan poglavljem 3.5. Slika 4.10. prikazuje verifikaciju funkcionalnost kreiranja oznake. Također, kako bi se omogućila bolja vizualizacija objekata od interesa prilikom procesa označavanja, korisniku je omogućeno uvećanje ili smanjenje prikaza slike, pritiskom na opcije „*Zoom In*“ ili „*Zoom Out*“ na alatnoj traci. Slika 4.11. prikazuje objekt od interesa prikazanog korištenjem opcije uvećanja.



Slika 4.10. Verifikacija funkcionalnosti kreiranja oznake

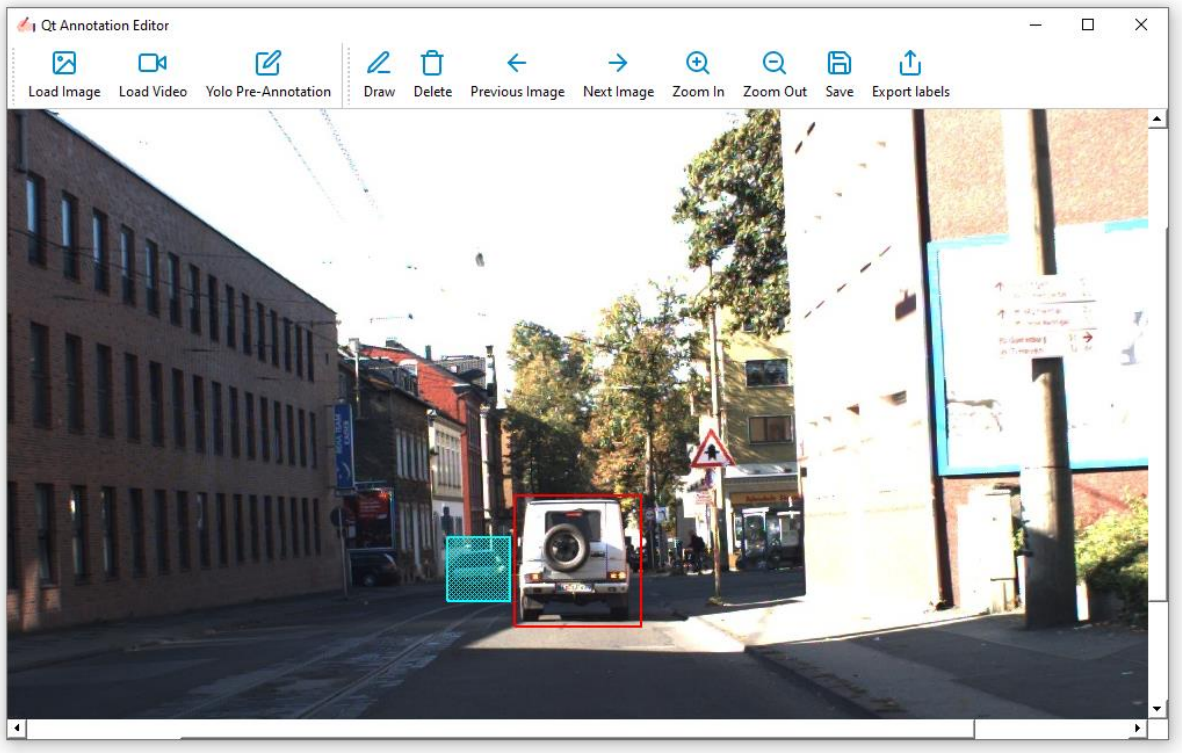


Slika 4.11. Verifikacija funkcionalnosti uvećanja prikaza

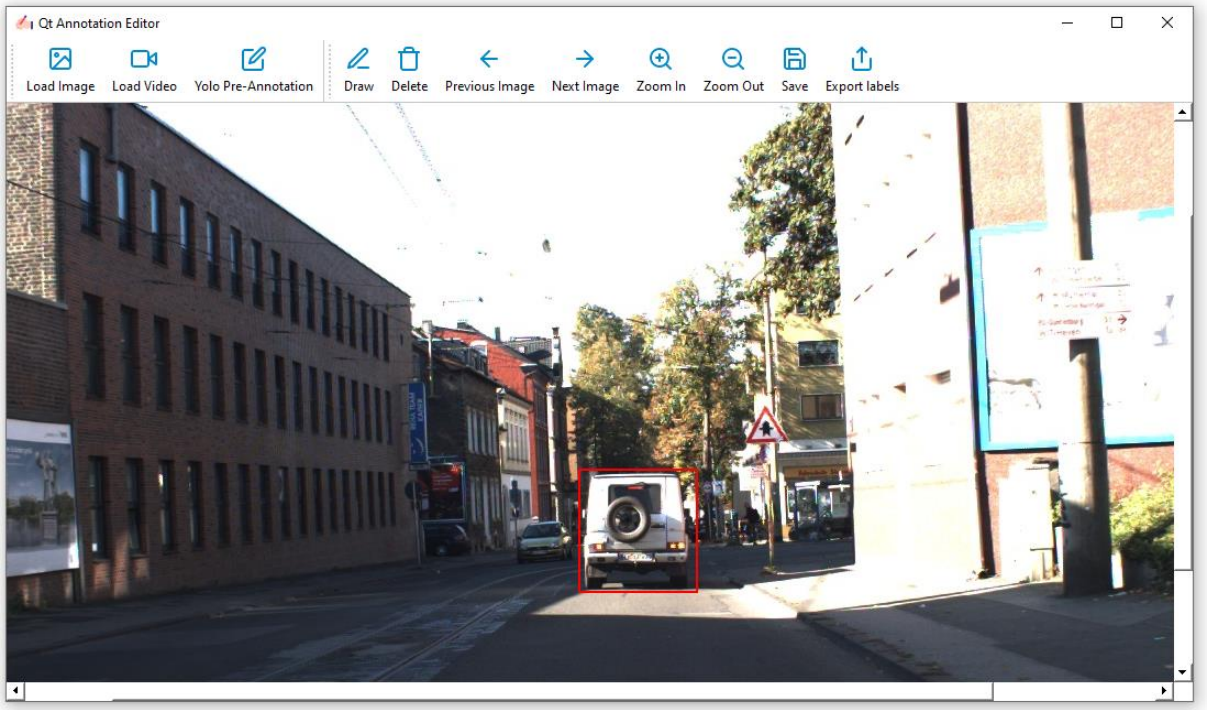
Pritiskom na „*Delete*“, korisnik može obrisati prethodno kreiranu oznaku na način opisan istim poglavljem. Pomicanje miša kao i opcija lijevog klika te prihvaćanja i otpuštanja, implementiraju svojstva rukovanja graničnim okvirom, odnosno, omogućuju promjenu prethodno kreiranih koordinata graničnog okvira. Slika 4.12. prikazuje verifikaciju promjene dimenzije graničnog okvira. Slika 4.13. prikazuje verifikaciju funkcionalnost brisanja prethodno kreirane oznake.

Provedbom postupka označavanja objekata na prikazanoj slici, korisniku se nudi mogućnost pohrane oznaka u radni rječnik oznaka, pritiskom na „*Save* “, na način opisan poglavljem 3.5. O uspješnosti pohrane oznake, aplikacija obavještava korisnika porukom o uspješnosti. Slika 4.14. prikazuje verifikaciju uspješnosti pohrane oznake u rječnik oznaka.

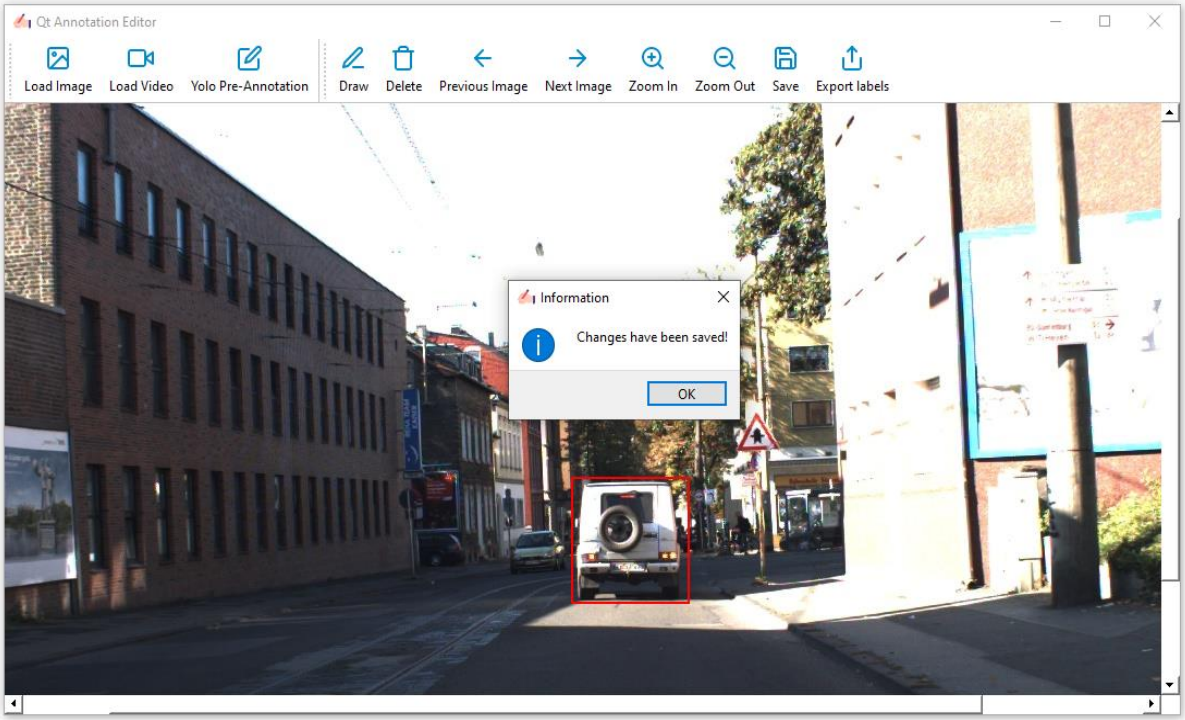
Po završetku cjelokupnog procesa označavanja, pritiskom na „*Export labels*“, korisniku se prikazuje dijaloški okvir koji nudi mogućnost promjene prethodno definiranog formata, kao i promjenu lokacije za izvoz datoteka s oznakama na disk. Slika 4.15. prikazuje dijaloški okvir konfiguracije parametara za izvoz datoteka s oznakama na disk.



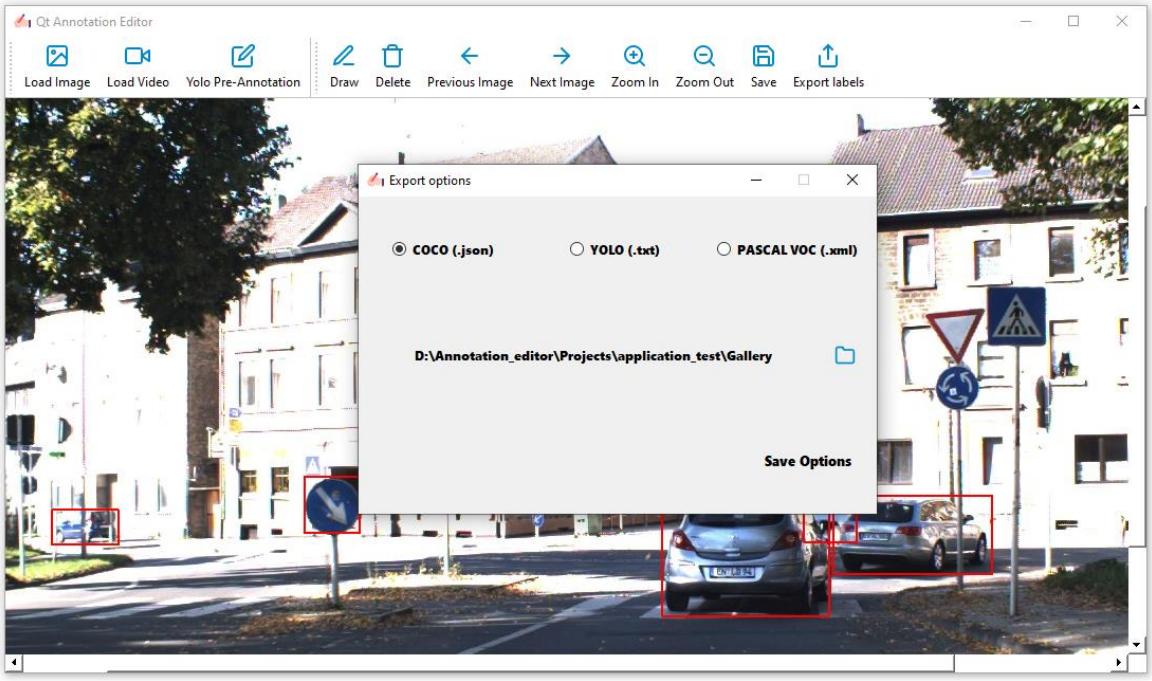
Slika 4.12. Verifikacija promjene dimenzije graničnog okvira



Slika 4.13. Verifikacija funkcionalnosti brisanja prethodno kreirane oznake

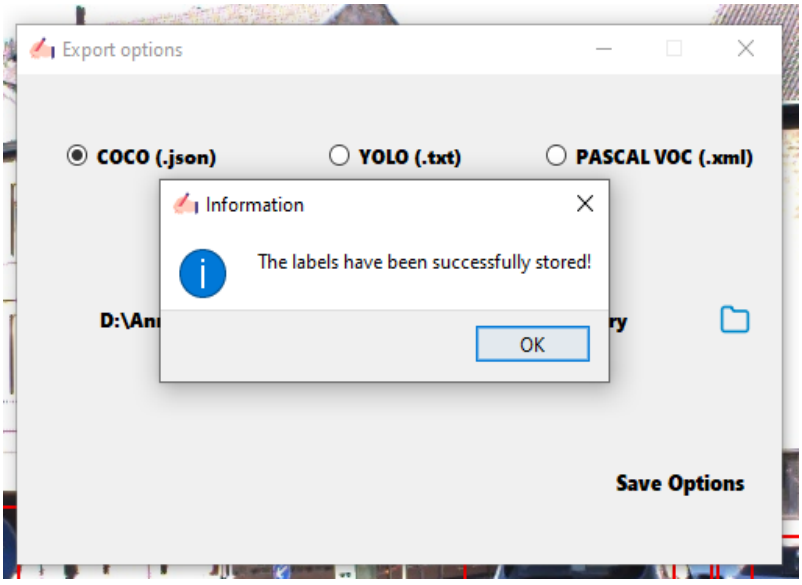


Slika 4.14. Verifikacija uspješnosti pohrane oznake u rječnik oznaka



Slika 4.15. Verifikacija funkcionalnosti prikaza dijaloškog okvira konfiguracije parametara za izvoz datoteka s oznakama na disk

Odabirom željenih vrijednosti korisnik završava postupak označavanja objekata u slikama. Aplikacija na odredišnoj lokaciji kreira datoteke s oznakama, a korisnika se obavještava o uspješnosti postupka, na način opisan poglavljem 3.5. Slika 4.16. prikazuje dijaloški okvir prikaza informacije o uspješnosti provedenog postupka izvoza datoteka s oznakama na disk.



Slika 4.16. Verifikacija prikaza dijaloškog okvira informacije o uspješnosti provedenog postupka izvoza datoteka s oznakama na disk

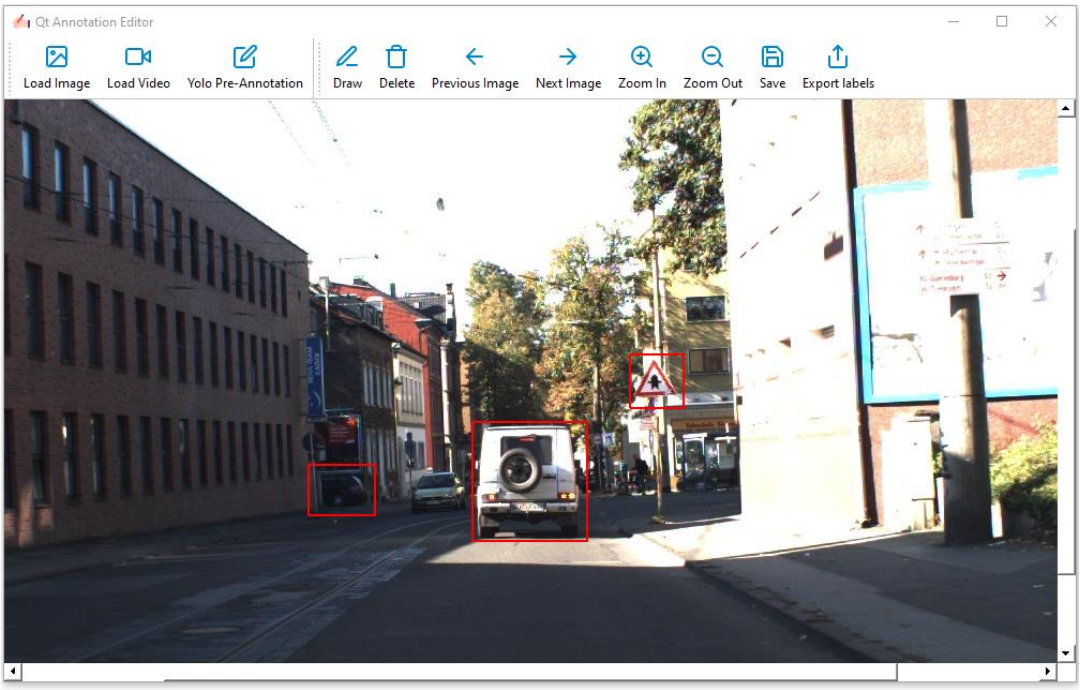
Slike 4.17, 4.18 i 4.19. prikazuju tri slike iz skupa neoznačenih slika u sklopu galerije projekta „*application_test*“ nad kojima će se izvršiti proces verifikacije izvoza datoteka s oznakama u različitim formatima korištenjem aplikacijskog modula za konfiguraciju parametara izvoza datoteka s oznakama na disk, kako je opisano poglavljem 3.5.

Slika 4.17. Prikaz neoznačene slike „000000“

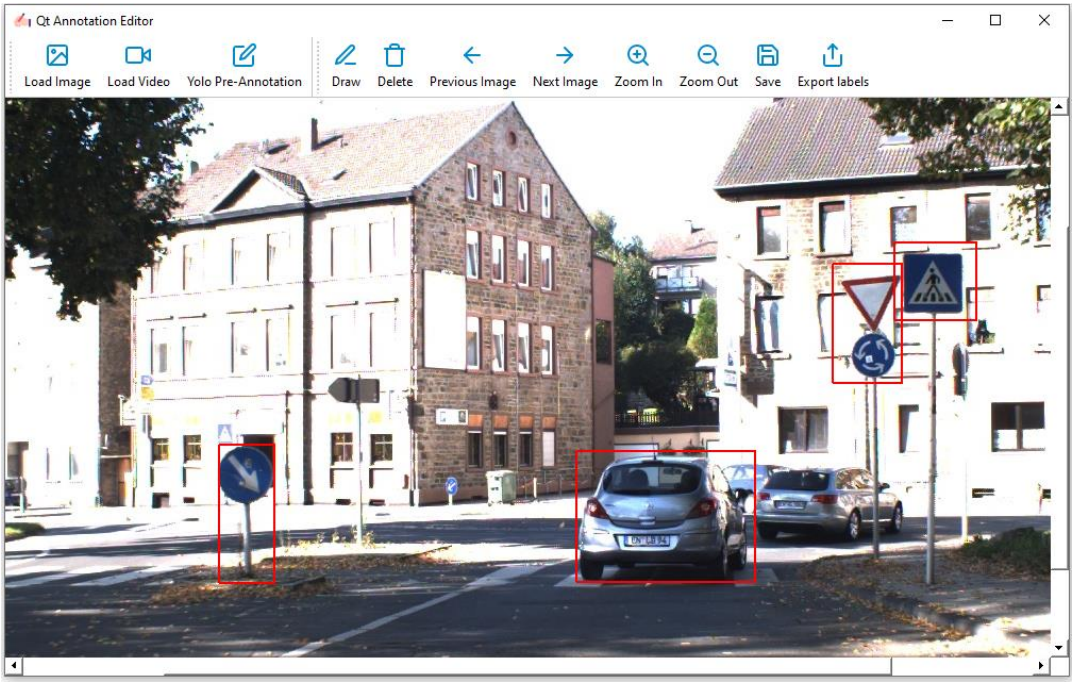
Slika 4.18. Prikaz neoznačene slike „000001“

Slika 4.19. Prikaz neoznačene slike „000002“

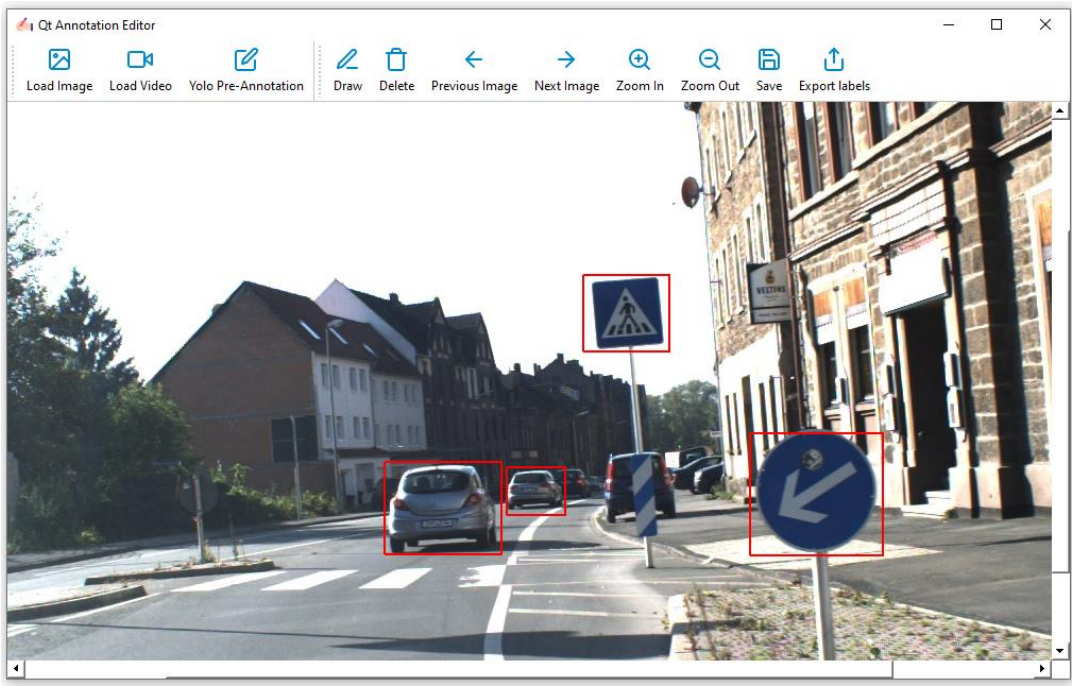
Provedbom postupka označavanja, kako je opisano poglavljem 3.5, nad objektima od interesa kreiraju se oznake. Slike 4.20, 4.21 i 4.22. prikazuju označene objekte od interesa.



Slika 4.20. Prikaz označenih objekata na slici „000000“

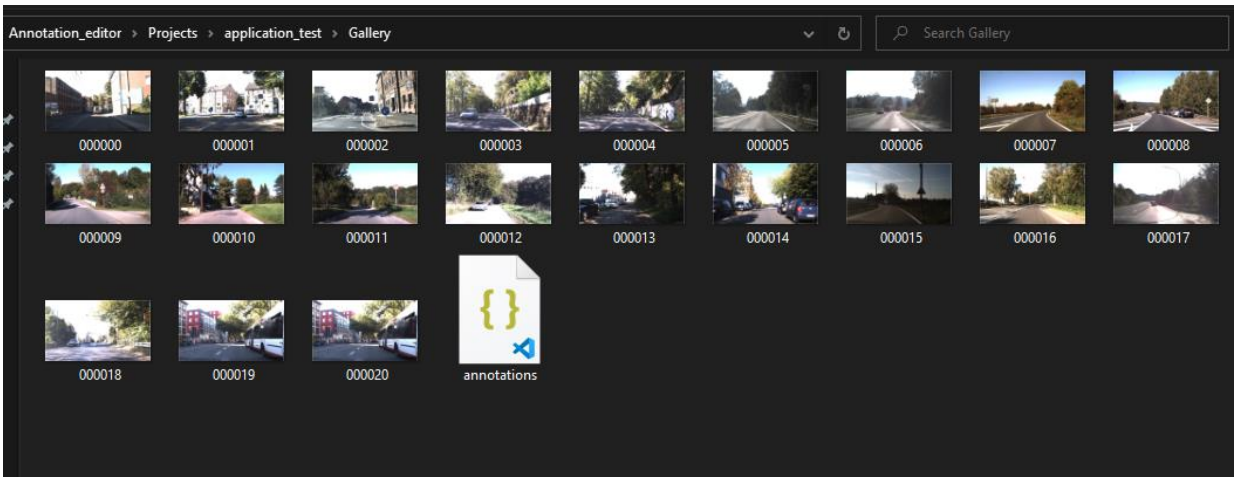


Slika 4.21. Prikaz označenih objekata na slici „000001“



Slika 4.22. Prikaz označenih objekata na slici „000002“

Izvozom oznaka na željenu lokaciju, dobiva se jedna ili više datoteka s oznakama u odabranom formatu. Slika 4.23. prikazuje izvoz datoteke s oznakama na lokaciji galerije projekta u COCO JSON formatu pod nazivom „*annotations.json*“.



Slika 4.23. Prikaz datoteke s oznakama na lokaciji galerije projekta u COCO JSON formatu

Sadržaj dijela datoteke „*annotations.json*“, prikazan je slikama 4.24, 4.25 i 4.26.


```
"images": [
  {
   "id": 1,
   "license": null,
   "file_name": "000001",
   "height": 800,
   "width": 1360,
   "date_captured": null
  },
  {
   "id": 0,
   "license": null,
   "file_name": "000000",
   "height": 800,
   "width": 1360,
   "date_captured": null
  },
  {
   "id": 2,
   "license": null,
   "file_name": "000002",
   "height": 800,
   "width": 1360,
   "date_captured": null
  }
 ],
```


Slika 4.24. Prikaz sadržaja datoteke s oznakama u COCO JSON formatu

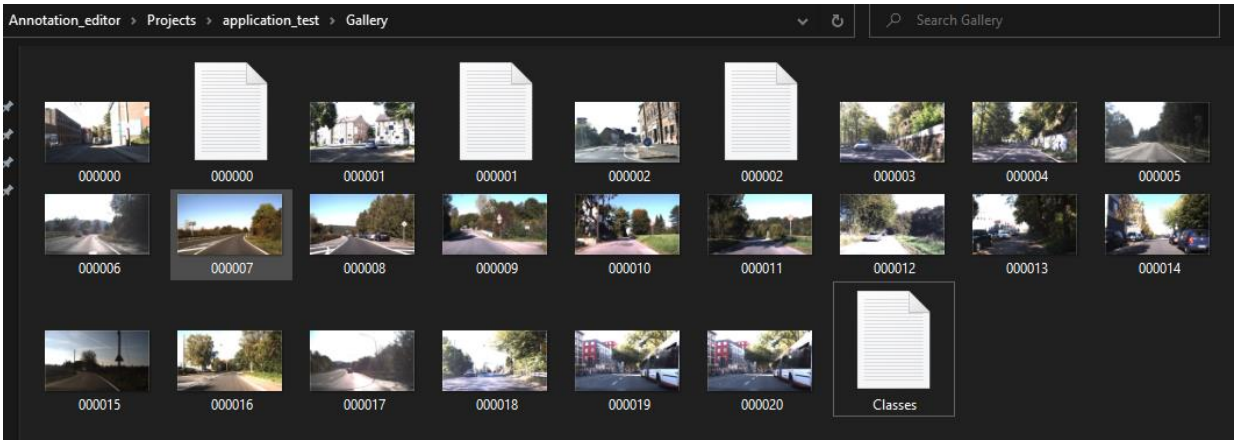
```
"annotation": [
  {
   "id": 0,
   "image_id": 0,
   "category_id": 0,
   "bbox": [
    0.45955882352941174,
    0.5875,
    0.07867647058823529,
    0.13875
   ],
   "segmentation": null,
   "area": null,
   "iscrowd": 0
  },
  {
   "id": 1,
   "image_id": 0,
   "category_id": 0,
   "bbox": [
    0.34705882352941175,
    0.6375,
    0.045588235294117645,
    0.05875
   ],
   "segmentation": null,
   "area": null,
   "iscrowd": 0
  },
  {
   "id": 2,
   "image_id": 0,
   "category_id": 1,
   "bbox": [
    0.5676470588235294,
    0.50875,
    0.03676470588235294,
    0.0625
   ],
   "segmentation": null,
   "area": null,
   "iscrowd": 0
  },
```


Slika 4.25. Prikaz sadržaja datoteke za sliku „000000“ u COCO JSON formatu

```
"categories": [
  {
   "id": 0,
   "name": "test class 1",
   "supercategory": null
  },
  {
   "id": 1,
   "name": "test class 2",
   "supercategory": null
  },
```

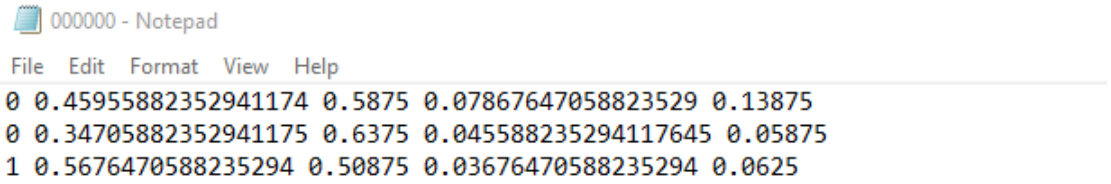

Slika 4.26. Prikaz sadržaja datoteke s oznakama u COCO JSON formatu

Slika 4.27. prikazuje izvoz datoteka s oznakama na lokaciji galerije projekta u YOLO TXT formatu.

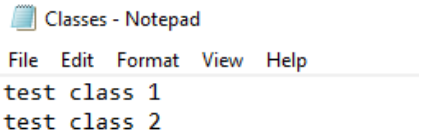


Slika 4.27. Prikaz datoteka s oznakama na lokaciji galerije projekta u YOLO TXT formatu

Slike 4.28 i 4.29. prikazuju sadržaj datoteka s oznakama u YOLO TXT formatu.

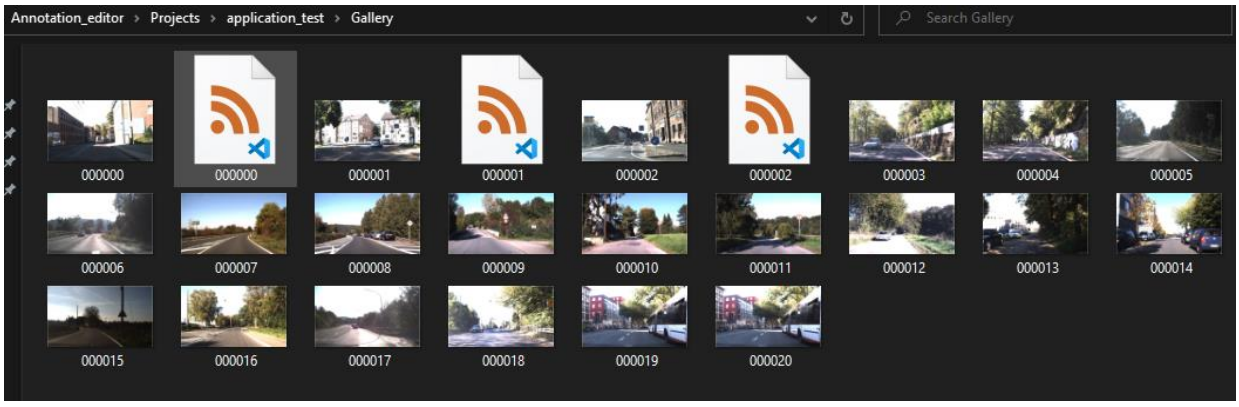


Slika 4.28. Prikaz sadržaja datoteke „000000“ u YOLO TXT formatu



Slika 4.29. Prikaz sadržaja datoteke „Classes.txt“ u YOLO TXT formatu

Slika 4.30. prikazuje izvoz datoteka s oznakama na lokaciji galerije projekta u PASCAL VOC XML formatu.



Slika 4.30. Prikaz datoteka s oznakama na lokaciji galerije projekta u PASCAL VOC XML formatu

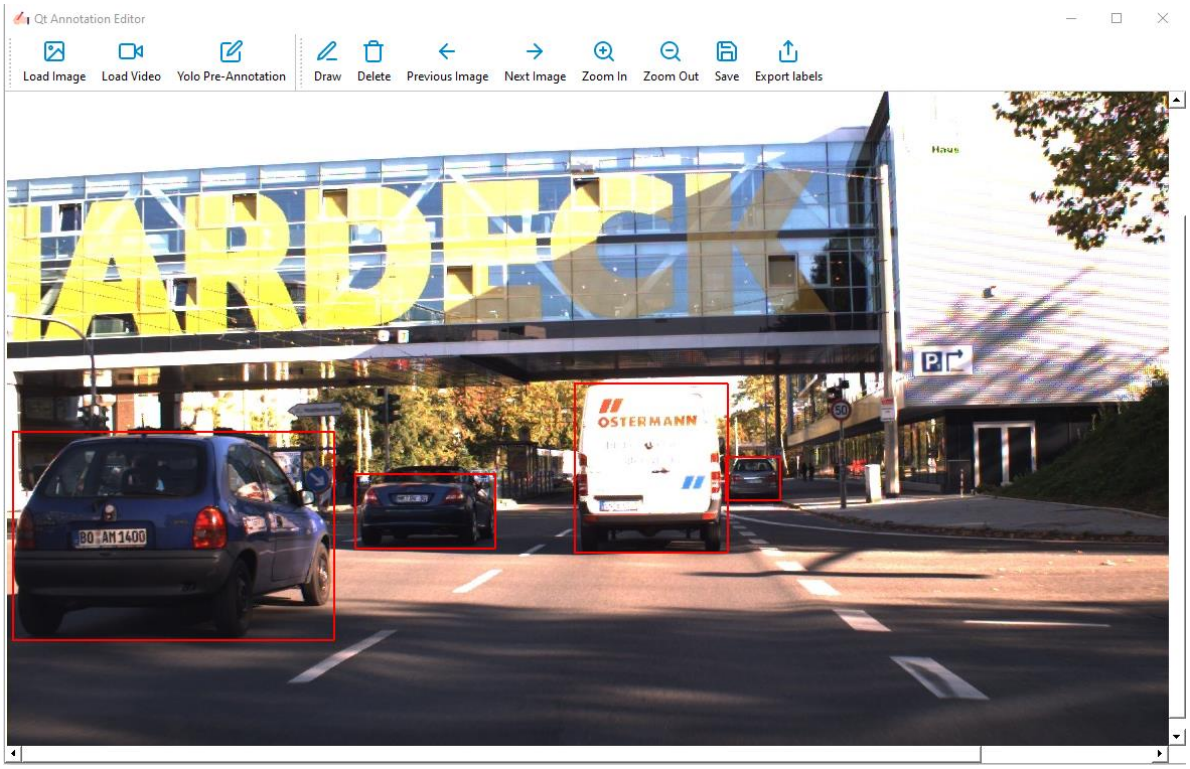
Slika 4.31. prikazuje sadržaj datoteke s oznakama u PASCAL VOC XML formatu.

```
<annotation>
    <filename>000000</filename>
    <size>
        <width>1360</width>
        <height>800</height>
        <depth>3</depth>
    </size>
    <object>
        <name>test class 1</name>
        <bndbox>
            <xmin>0.45955882352941174</xmin>
            <ymin>0.5875</ymin>
            <xmax>0.538235294117647</xmax>
            <ymax>0.7262500000000001</ymax>
        </bndbox>
    </object>
    <object>
        <name>test class 1</name>
        <bndbox>
            <xmin>0.34705882352941175</xmin>
            <ymin>0.6375</ymin>
            <xmax>0.3926470588235294</xmax>
            <ymax>0.6962499999999999</ymax>
        </bndbox>
    </object>
    <object>
        <name>test class 2</name>
        <bndbox>
            <xmin>0.5676470588235294</xmin>
            <ymin>0.50875</ymin>
            <xmax>0.6044117647058823</xmax>
            <ymax>0.57125</ymax>
        </bndbox>
    </object>
</annotation>
```

Slika 4.31. Prikaz sadržaja datoteke „000000“ u PASCAL VOC XML formatu

4.3. Verifikacija automatske detekcije objekata u slikama

Verifikacija implementacije automatske detekcije objekata korištenjem detektora YOLOv5, opisanog poglavljem 3.6, označava proces kreiranja oznaka detektiranih objekata, njihove vizualizacija te pohrane u radni rječnik oznaka, kako je opisano u poglavlju 3.5. Korištenjem opcije „*Yolo Pre-Annotate*“ alatne trake, korisniku je omogućeno inicijalno označavanje objekata korištenjem detektora. Pri tome aplikacija vodi računa o vizualizaciji i pohrani detektiranih oznaka u rječnik oznaka na način opisan poglavljem 3.5. Slika 4.32. prikazuje verifikaciju procesa kreiranja oznaka detektiranih objekata te njihovu vizualizaciju.



Slika 4.32. Verifikacija procesa kreiranja i vizualizacije oznaka detektiranih objekata, korištenjem YOLOv5 detektora

Korištenjem opcije „*Export labels“* alatne trake, prethodno pohranjene oznake iz rječnika oznaka, izvoze se na odabranu lokaciju na disku, kao što je opisano poglavljem 3.5. Slika 4.33. prikazuje verifikaciju dodavanja novo detektiranih klasa objekata u COCO JSON formatu u datoteku „*annotations.json“*, opisanu slikama 4.24 i 4.26. Na slici 4.33. vidljivo je kako je aplikacija uspješno dodala klase: „*car*“, „*stop sign*“ i „*traffic light*“, koje prethodno nisu definirane projektom, na način opisan poglavljem 4.1.

```
"categories": [
  {
   "id": 0,
   "name": "test class 1",
   "supercategory": null
  },
  {
   "id": 1,
   "name": "test class 2",
   "supercategory": null
  },
  {
   "id": 2,
   "name": "car",
   "supercategory": null
  },
  {
   "id": 3,
   "name": "stop sign",
   "supercategory": null
  },
  {
   "id": 4,
   "name": "traffic light",
   "supercategory": null
  }
 ],
```


Slika 4.33. Verifikacija dodavanja novo detektiranih klasa objekata

5. ZAKLJUČAK

Ovim radom uspješno je razvijena i implementirana aplikacija za označavanje objekata prisutnih u digitalnim slikama. Aplikacija implementira dva sučelja, sučelje za upravljanje projektima i sučelje za označavanje objekata, od kojih svako doprinosi sveobuhvatnoj i korisnički orijentiranoj platformi za označavanje podataka. Sučelje za upravljanje projektima služi kao početna točka, omogućujući korisnicima da s lakoćom definiraju važne parametre projekta. Integracijom *Qt* i *Qt designer-a*, sučelje postiže intuitivan i korisnički orijentiran dizajn, pojednostavljujući proces postavljanja i konfiguracije projekta. Kroz implementaciju sučelja korisniku je omogućen unos naziva projekta, odabir projektne lokacije na disku, konfiguracija projektnih klasa, odabir galerije i odabir formata za izvoz datoteke s oznakama na disk, olakšavajući na taj način personalizirane tijekove rada za označavanje podataka.

Sučelje za označavanje objekata pruža krajnjem korisniku jednostavan način označavanja podataka. Iskorištavanjem funkcionalnosti crtanja aplikacije, korisnici mogu precizno označiti objekte od interesa crtanjem graničnih okvira i odabirom prethodno definiranih klasa. Mogućnost vizualizacije te rukovanja oznakama prilikom procesa označavanja objekata od interesa osigurava kreiranje preciznih oznaka. Sučelje dodatno podržava proširenje galerije, omogućujući dodavanje novih slika i videozapisa u postojeći skup podataka. Štoviše, mogućnost izvoza označenih podataka u različitim formatima kao što su JSON, TXT ili XML, poboljšava kompatibilnost s različitim modelima strojnog učenja te omogućuje treniranje modernih detektora objekata.

Verifikacija sučelja za upravljanje projektima, provedena je kreiranjem novog projekta uz testne parametre naziva projekta, proizvoljan odabir projektne lokacije na disku te uz konfiguraciju proizvoljnog broja projektnih klasa. Verifikacija sučelja za označavanje objekata, provedena je na projektu kreiranom od strane korisnika uz korištenje sučelja za upravljanje projektima. Verifikacija automatske detekcije objekata za cilj ima osigurati ispravnost vizualizacije i pohrane oznaka detektiranih objekata, korištenjem detektora YOLOv5.

Aplikacija za označavanje objekata u digitalnim slikama, dodatno nudi razne mogućnosti unaprjeđenja. Jedna od mogućnosti bila bi omogućavanje označavanja objekata uz pomoć poligona. Također, granični okviri svake kreirane oznake, mogu biti predstavljeni bojom odabrane klase. Time bi se postigla bolja vizualizacije raznolikosti označenih objekata od interesa. Također, kako bi se dodatno poboljšao proces automatske detekcije, moguće je korisniku ponuditi odabir detektora različite složenosti.

LITERATURA

[1] „E-Book-Deeplobe-Data-Labeling.pdf“. Pristupljeno: 03. srpanj 2023. [Na internetu]. Dostupno na: https://deeplobe.ai/wp-content/uploads/2021/11/E-Book-Deeplobe-Data-Labeling.pdf

[2] R. Munro i R. Monarch, *Human-in-the-Loop Machine Learning: Active Learning and Annotation for Human-centered AI*. Simon and Schuster, 2021.

[3] „Automatic vs. Manual Data Labeling“. Pristupljeno: 03. srpanj 2023. [Na internetu]. Dostupno na: https://www.diva-portal.org/smash/get/diva2:1460858/FULLTEXT01.pdf

[4] „The Complete Guide to Image Annotation for Computer Vision“. https://encord.com/blog/guide-image-annotation-computer-vision/ (pristupljeno 06. srpanj 2023.).

[5] B. Javidi, *Image Recognition and Classification: Algorithms, Systems, and Applications*. CRC Press, 2002.

[6] B. Cyganek, *Object Detection and Recognition in Digital Images: Theory and Practice*. John Wiley & Sons, 2013.

[7] T. Lei i A. K. Nandi, *Image Segmentation: Principles, Techniques, and Applications*. John Wiley & Sons, 2022.

[8] A. Hanbury, „A survey of methods for image annotation“, *J. Vis. Lang. Comput.*, sv. 19, izd. 5, str. 617–627, lis. 2008, doi: 10.1016/j.jvlc.2008.01.002.

[9] „The Ultimate Guide to Image Annotation for Computer Vision: A Comprehensive Resource for Optimal Results“. https://encord.com/blog/image-annotation-guide/ (pristupljeno 30. kolovoz 2023.).

[10] G. Lin, *Building Simple Annotation Tools*. University of California, San Diego, 2016.

[11] S. Bianco, G. Ciocca, P. Napoletano, i R. Schettini, „An interactive tool for manual, semi-automatic and automatic video annotation“, *Comput. Vis. Image Underst.*, sv. 131, str. 88–99, velj. 2015, doi: 10.1016/j.cviu.2014.06.015.

[12] „How to Organize Data Labeling for Machine Learning: Approaches and Tools“, *AltexSoft*, 03. srpanj 2023. https://www.altexsoft.com/blog/datascience/how-to-organize-data-labeling-for-machine-learning-approaches-and-tools/ (pristupljeno 03. srpanj 2023.).

[13] W. John, *Encyclopedia of Data Science and Machine Learning*. IGI Global, 2023.

[14] D. Gupta, S. Bhattacharyya, A. Khanna, i K. Sagar, *Intelligent Data Analysis: From Data Gathering to Data Comprehension*. John Wiley & Sons, 2020.

[15] T.-Y. Lin *i ostali*, „Microsoft COCO: Common Objects in Context“, izd. arXiv:1405.0312. arXiv, 20. veljača 2015. Pristupljeno: 03. srpanj 2023. [Na internetu]. Dostupno na: http://arxiv.org/abs/1405.0312

[16] „COCO - Common Objects in Context“. https://cocodataset.org/#home (pristupljeno 30. kolovoz 2023.).

[17] „COCO format - Rekognition“. https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-coco-overview.html (pristupljeno 13. srpanj 2023.).

[18] „Understanding PASCAL VOC Dataset“, *Engineering Education (EngEd) Program | Section*. https://www.section.io/engineering-education/understanding-pascal-voc-dataset/ (pristupljeno 31. srpanj 2023.).

[19] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, i A. Zisserman, „The Pascal Visual Object Classes (VOC) Challenge“, *Int. J. Comput. Vis.*, sv. 88, izd. 2, str. 303–338, lip. 2010, doi: 10.1007/s11263-009-0275-4.

[20] „YOLO - Plainsight“. https://docs.plainsight.ai/labels/exporting-labels/yolo (pristupljeno 30. kolovoz 2023.).

[21] Ultralytics, „Object Detection Datasets Overview“. https://docs.ultralytics.com/datasets/detect (pristupljeno 30. kolovoz 2023.).
[22] J. Blanchette i M. Summerfield, *C++ GUI programming with Qt 4*, 2nd ed., Extensively rev. and Expanded. Upper Saddle River, NJ: Prentice Hall in association with Trolltech Press, 2008.
[23] L. Z. Eng i R. Rischpater, *Application Development with Qt Creator: Build cross-platform applications and GUIs using Qt 5 and C++, 3rd Edition*. Packt Publishing Ltd, 2020.
[24] M. Fitzpatrick, *Create GUI Applications with Python & Qt6 (PyQt6 Edition): The hands-on guide to making apps with Python*. Martin Fitzpatrick, 2022.
[25] „Qt Documentation | Home“. https://doc.qt.io/ (pristupljeno 30. kolovoz 2023.).
[26] G. Jocher, „YOLOv5 by Ultralytics“. svibanj 2020. doi: 10.5281/zenodo.3908559.
[27] A. Bochkovskiy, C.-Y. Wang, i H. Liao, *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020.
[28] R. Iyer, K. Bhensdadiya, i P. Ringe, „Comparison of YOLOv3, YOLOv5s and MobileNet-SSD V2 for Real-Time Mask Detection“, *Int. J. Res. Eng. Technol.*, str. 2395–0056, srp. 2021.
[29] „PyTorch-Brand-Guidelines.pdf“. Pristupljeno: 30. kolovoz 2023. [Na internetu]. Dostupno na: https://pytorch.org/assets/brand-guidelines/PyTorch-Brand Guidelines.pdf?ref=blog.roboflow.com
[30] M. Tan, R. Pang, i Q. V. Le, „EfficientDet: Scalable and Efficient Object Detection“, u *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA: IEEE, lip. 2020, str. 10778–10787. doi: 10.1109/CVPR42600.2020.01079.

SAŽETAK

Osnovni problem u okviru računalnog vida je detekcija objekata. To je proces identificiranja neobrađenih podataka i davanja smislenih i informativnih oznaka tim podacima. Označavanjem objekata od interesa povezujemo svaki objekt s određenom klasom ili kategorijom, pružajući semantičko razumijevanje objektnog identiteta ili karakteristika. Najčešći način označavanja objekata u digitalnoj slici je crtanjem graničnog okvira oko objekta. Alat korišten za označavanje, treba biti intuitivan, jednostavan za korištenje i kompatibilan sa željenim tipovima oznaka. U ovom radu predstavljeno je i uspoređeno nekoliko programskih alata te su jasno naznačene njihove prednosti, zajedničke funkcionalnosti kao i njihove razlike. Nakon procesa označavanja, oznake se pohranjuju u jednu ili više datoteka, uspostavljajući nedvosmislenu vezu između svake pojedinačne slike u skupu podataka i odgovarajućih kreiranih oznaka. Iz tog razloga datoteke s oznakama se pohranjuju u strukturirane računalne formate (XML, JSON, TXT). Aplikacija za označavanje objekata u digitalnim slikama predstavljena ovim radom, implementira dva sučelja korištenjem *Qt* okvira: sučelje za upravljanje projektima i sučelje za označavanje objekata. Sučelje za upravljanje projektima implementirano je na način da omogućuje korisniku kreiranje projekata. To uključuje definiranje naziva projekta, odabir lokacije na disku, mogućnost definiranja klasa objekata od interesa, odabir projektne galerije te formata izvoza datoteka s oznakama. Sučelje za označavanje objekata, korisniku omogućuje kreiranje oznaka nad objektima od interesa, njihovu vizualizaciju, rukovanje kreiranim oznakama te pohranu u radni rječnik oznaka, uz mogućnost dodatne konfiguracije parametara izvoza datoteke s oznakama na disk. Implementirani proces automatske detekcije korištenjem YOLOv5 detektora, dodatno olakšava proces označavanja, nudeći korisniku mogućnost inicijalne detekcije, kreiranja te vizualizacije oznaka, detektiranih objekata od interesa. Svaka funkcionalnost verificirana je kroz procese kreiranja testnog projekta i označavanja objekata od interesa prisutnih u skupu neoznačenih slika galerije projekta.

Ključne riječi: Aplikacije za označavanje objekata, Strojno učenje, YOLOv5, *Qt*, Granični okviri, Sučelje za upravljanje projektima, Sučelje za označavanje objekata

ABSTRACT

A fundamental problem in computer vision is object detection. That is the process of identifying raw data and assigning informative annotations to that data. By annotating objects of interest, we associate each object with a specific class or category, offering semantic insights into the identity or characteristics of objects. The most prevalent method to annotate objects in digital images is by drawing bounding boxes around them. The chosen annotation tool should be intuitive, user-friendly, and compatible with the desired annotation formats. In this paper, several software tools are compared in such a way that their advantages, common functionalities and their differences are indicated. After the annotation process, the annotations are stored in one or multiple files, establishing an unequivocal link between each individual image in the dataset and its corresponding annotation file. For this reason, annotation files are stored in structured computer formats (such as XML, JSON, or TXT). The application for annotating objects in digital images presented in this paper, employs two interfaces built using the Qt framework: one for project management and another for object annotation. The project management interface is designed to enable users to initiate projects. This encompasses tasks like defining project names, selecting disk locations, specifying object classes, choosing a project gallery, and setting annotation file formats. The object annotation interface empowers users to create annotations on objects of interest, visualize, manage, and save them, with options to configure storage parameters. The integrated automated detection process uses the YOLOv5 detector to expedite annotation. It provides users with preliminary detection, annotation creation, and visualization of identified objects of interest. Each functionality has been verified by creating a test project and annotating objects of interest within the set of non-annotated images in the project gallery.

Keywords: Application for Objects Annotation, Machine Learning, YOLOv5, Qt, Bounding Boxes, Project Management Interface, Object Annotation Interface

ŽIVOTOPIS

Stjepan Prakljačić je rođen 6. 7. 1999. godine u Osijeku. Godine 2001. preselio se s roditeljima u mjesto Viškovce gdje je pohađao Osnovu školu Luke Botića. Godine 2013. upisao se u prvi razred Srednje strukovne škole Antuna Horvata u Đakovu, smjer tehničar za mehatroniku. Tijekom srednjoškolskog obrazovanja aktivno je sudjelovao u školskim projektima te mu je dodijeljena nagrada Hrvatske udruge poslodavaca pod nazivom „Poduzetnici budućnosti“ za projekt i poduzetničku ideju pod nazivom „Solarni, prijenosni hladnjak“. Godine 2017. dodijeljeno mu je priznanje kao najboljem učeniku u tehničkim zanimanjima, u četverogodišnjem trajanju. Nakon završetka srednjoškolskog obrazovanja, upisao se na preddiplomski sveučilišni studij Računarstvo na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. Godine 2021. upisao se na Sveučilišni diplomski studij, smjer Automobilsko računarstvo i komunikacije na istom fakultetu te postaje stipendist kompanije TTTech Auto d.o.o. u kojoj odrađuje studentsku praksu.

_____

Potpis autora