

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5574

**Aplikacija za pridjeljivanje bilješki i
tekstualnih oznaka video isječcima**

DORA MLINARIĆ

Zagreb, srpanj 2018.

Zagreb, 16. ožujka 2018.

ZAVRŠNI ZADATAK br. 5574

Pristupnik: **Dora Mlinarić (0036480184)**
Studij: Računarstvo
Modul: Računalno inženjerstvo

Zadatak: **Aplikacija za pridjeljivanje bilješki i tekstualnih oznaka video isječcima**

Opis zadatka:

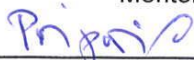
Vaš zadatak je osmisлити, izvesti u programskom jeziku Java te testirati aplikaciju kojom će se dodavati bilješke te dodjeljivati tekstualne oznake video isječcima. Omogućite dodavanja bilješki oblika pravokutnika ili poligona u video isječak. Pri tome pretpostavite da postoji konačan skup mogućih bilješki koji je hijerarhijski organiziran. Osim toga, aplikacija treba omogućiti dodavanje tekstualnih oznaka video isječcima te grupiranje video isječaka u različite tematske skupove kojima se zatim mogu dodavati zajedničke tekstualne oznake.

Proučite i opišite trenutno dostupne standarde za pridjeljivanje bilješki i tekstualnih oznaka video isječcima. Pretpostavite da će se aplikacija koristiti za rad s velikim brojem video isječaka pa pri njenom razvoju posebnu pozornost obratite na performanse.

Svu potrebnu literaturu i uvjete za rad osigurat će Vam Zavod za telekomunikacije.


Zadatak uručen pristupniku: 16. ožujka 2018.
Rok za predaju rada: 15. lipnja 2018.

Mentor:



Izv. prof. dr. sc. Krešimir Pripužić

Djelovođa:



Prof. dr. sc. Danko Basch

Predsjednik odbora za
završni rad modula:



Prof. dr. sc. Mario Kovač

Zahvala

Ovaj rad napravljen je u suradnji s Telegra Solutions d.o.o gdje radim kao student. Ovim putem zahvaljujem cijelom X-AID timu na motivaciji i ustupljenom znanju potrebnom za razvoj ove aplikacije.

Sadržaj

Uvod	1
1. Anotacije.....	2
1.1. Upotreba anotacija.....	2
1.2. Postojeće aplikacije za pridavanje bilješki	3
1.2.1. LabelImg.....	3
1.2.2. ImageAnnotation	4
1.2.3. LabelMe.....	6
1.2.4. Ostale aplikacije	6
1.2.5. Nedostaci postojećih aplikacija	7
1.3. Anotacije potrebne za izradu trening seta.....	7
2. Tehnologije korištene u razvoju aplikacije.....	8
2.1. JavaFX.....	8
2.1.1. Glavne značajke.....	9
2.1.2. Struktura JavaFX aplikacije.....	11
2.2. Brownies Collections.....	13
2.3. OpenCV	14
2.4. Pascal VOC	15
2.4.1. Pascal VOC format.....	16
3. Programsko rješenje aplikacije.....	17
3.1. Zahtjevi koje aplikacija ispunjava	17
3.2. Struktura i modeliranje aplikacije.....	18
3.2.1. Model aplikacije	19
3.2.2. Izgled i funkcionalnost aplikacije.....	22
Zaključak	31
Literatura	32

Sažetak.....	33
Summary.....	34
Popis kratica	35
Popis slika.....	36
Popis kodova	37
Privitak	38

Uvod

Današnjim razvojem tehnologije teži se k tome da računala zamijene ljude u mnogim poslovima. Pokušava se doći do točke kada će računalo moći obavljati određene radnje bolje nego čovjek. Za velik broj radnji i poslova to je već ostvareno. Neke radnje su čovjeku vrlo lagane i prirodne, no računalu predstavljaju veliki izazov. Jedna od takvih radnji je računalni vid. Računalni vid je područje umjetne inteligencije koje se bavi prepoznavanjem dvodimenzionalnih i/ili trodimenzionalnih predmeta. Jedan od primjera je prepoznavanje i klasifikacija objekata na slici. Čovjek će bez većih poteškoća na slici prepoznati vozilo, čovjeka, životinju i sl. Da bi računalno to moglo izvesti koriste se tehnologije kao što su strojno učenje (engl. *Machine learning*) ili duboko učenje (engl. *Deep learning*). Predmet ovog rada nisu navedene tehnologije nego su navedene kako bi se objasnila motivacija razvoja same aplikacije. Strojno učenje je grana računarske znanosti koja daje računalu sposobnost učenja bez da je ono eksplicitno programirano. Da bi računalo naučilo treba ga trenirati. U postupku treniranja računalu se daje skup podataka i govori mu se što taj podatak predstavlja. Npr. da bi računalo znalo prepoznati na slici vozila, treba mu dati slike na kojima su vozila i reći „to je vozilo“, odnosno slike na kojima nisu vozila i reći „to nije vozilo“. Pa upravo kako bi se olakšalo pribavljanje takvih skupova podataka odnosno *setova za treniranje* (engl. *Training set*) i organizacija istih napravljena je aplikacija za pridjeljivanje bilješki i tekstualnih oznaka, nazvana *Xtag*.

1. Anotacije

Anotacijom (engl. *Annotation*) se određenoj slici daju metapodaci (engl. *metadata*), tj. komentar ili opis koji se odnosi na određena područja ili objekte na slici (Wikipedia).



Slika 1.1 Primjer anotacija na slici

1.1. Upotreba anotacija

Anotacije se mogu koristiti u različite svrhe kao što su:

- **Textual scholarship** – disciplina koja često koristi tehnike anotacija za opisivanje ili dodavanje dodatnog povijesnog konteksta tekstu i fizičkim dokumentima
- **Učenje** – mnogi studenti i učenici koriste tehnike pridjeljivanja bilješki tekstu dok uče, npr. pravokutnikom označe tekst i dopišu potrebne informacije, asocijaciju koja će ih podsjetiti što je to točno dok će ponavljati kasnije i sl.
- **Web** – jezici za označavanje podataka kao što su XML ili HTML označavaju tekst na način koji je sintaktički različit od samog teksta. Može se upotrijebiti za dodavanje informacija o željenoj vizualnoj prezentaciji ili semantičkim informacijama koje su razumljive računalu
- **Kontrola izvora** (engl. *Source control*) – postoje funkcije bilješki kao što su „krivanja“ ili „pohvala“ koje se koriste u sustavima kontrole izvora npr. *Git*.

- **Java anotacije** – anotacije se koriste kao poseban oblik sintaktičkih meta podataka u izvornom kodu; klase, metode, varijable, parametri i paketi mogu biti anotirani
- **Računalna biologija** – DNA anotacija ili anotacija genoma je proces identificiranja lokacija gena i svih regija u genomu i određivanje onoga što ti geni rade (Wikipedia)
- **Slike** – označavanje se obično koristi za vidljive meta podatke postavljene na sliku bez mijenjanja glavne slike, kao što su ljepljive bilješke (engl. *Sticky notes*), virtualni laserski pokazivači, krugovi, **pravokutnici**, strelice i sl.

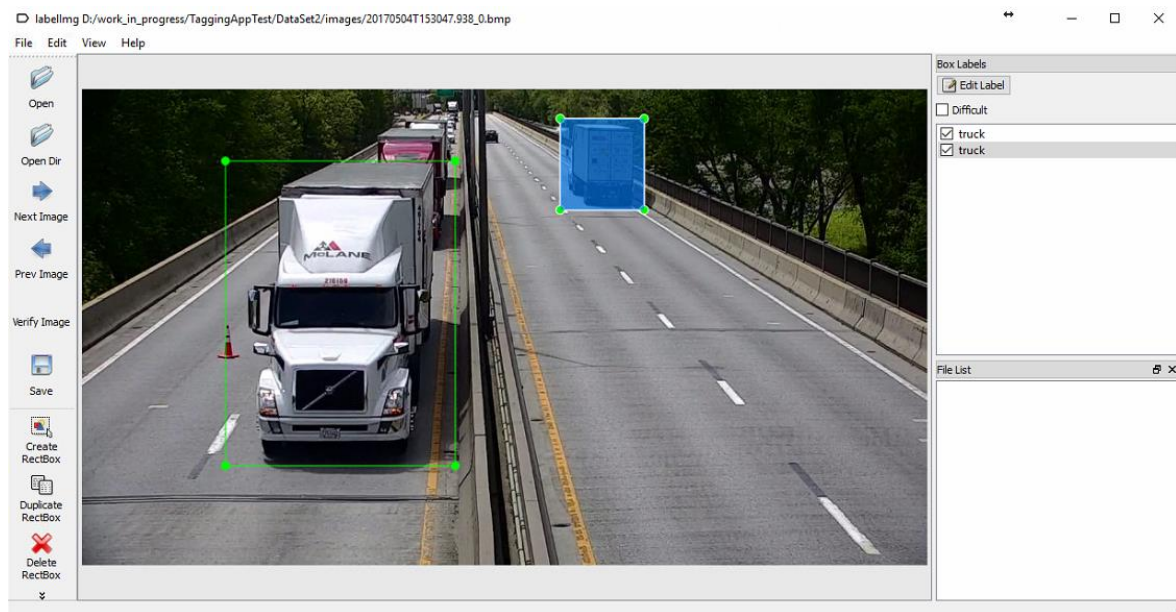
I razne druge. Za potrebe ovog rada na slici ćemo označavati vozila i pješake i dodavati im određene klase i još neke podatke o kojima ćemo reći nešto više u nastavku, iako će sama aplikacija podržavati unos klasa po želji. Klase će korisnik moći sam definirati pomoću konfiguracijskih parametara.

1.2. Postojeće aplikacije za pridavanje bilješki

Pretraživanjem interneta nema previše rezultata za takve aplikacije. U nastavku ćemo opisati nekoliko njih dostupnih na internetu. Većinom su to jednostavnije aplikacije s kojima se može postići jedna određena stvar, a ako želimo još nešto dodatno, mora se koristiti neka druga dodatna aplikacija.

1.2.1. LabelImg

LabelImg je grafički alat za bilježenje anotacija i označavanje objekata pravokutnicima na slikama (Tzutalin, 2015). Razvijen je u programskom jeziku *Python*. Anotacije sprema kao XML datoteke u *PASCAL VOC* formatu s kojim ćemo se upoznati kasnije.



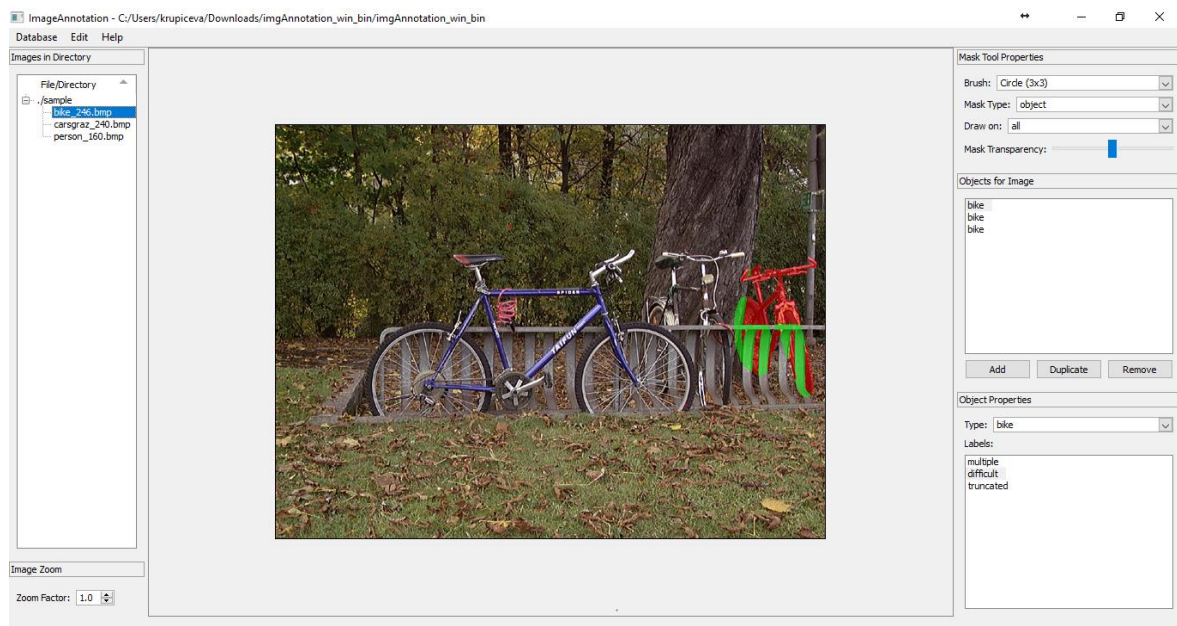
Slika 1.2 Izgled aplikacije LabelImg

Aplikacija se može preuzeti sa slijedećeg linka: <https://github.com/tzutalin/labelImg>

LabelImg je dosta intuitivan za korištenje, omogućava označavanje objekata pravokutnicima i dodavanje oznake tom pravokutniku. Također anotaciji se može pridodati labela *Difficult* koja označava da je objekt na slici teško prepoznatljiv. Veliki nedostatak je što klase moraš ručno upisivati, automatska lista klasa i odabiranje jedne od njih bi znatno ubrzalo posao korisniku. Također ako npr. učitamo mapu sa slikama, na jednoj od slika napravimo anotacije i spremimo je, pređemo na slijedeću sliku i ukoliko se vratimo na tu prijašnju sliku, aplikacija neće pokazivati koji su sve objekti označeni na slici. Takav scenarij povlači za sobom to da korisnik mora sam otvoriti XML datoteku u nekom naprednijem tekst editoru i ručno pogledati koje klase su dodane, pročitati koordinate pravokutnika i zatim u nekoj aplikaciji koja može prikazivati pravokutnike na slici na temelju koordinata provjeriti gdje se točno nalazi anotirani objekt. Dakle, *LabelImg* je jednostavan i koristan alat koji radi ono osnovno za što je namijenjen, no mogao bi se znatno unaprijediti kako bi korisniku ubrzao i automatizirao proces pridjeljivanja bilježaka i označavanja objekata na slici.

1.2.2. ImageAnnotation

Ova aplikacija dolazi u dvije verzije, jedna podržava dodavanje pravokutnika (Slika 1.4), a druga maski (Slika 1.3). Obje su pisane u Qt-u (Klaser, 2010).



Slika 1.3 Izgled aplikacije ImageAnnotation s dodavanjem maski



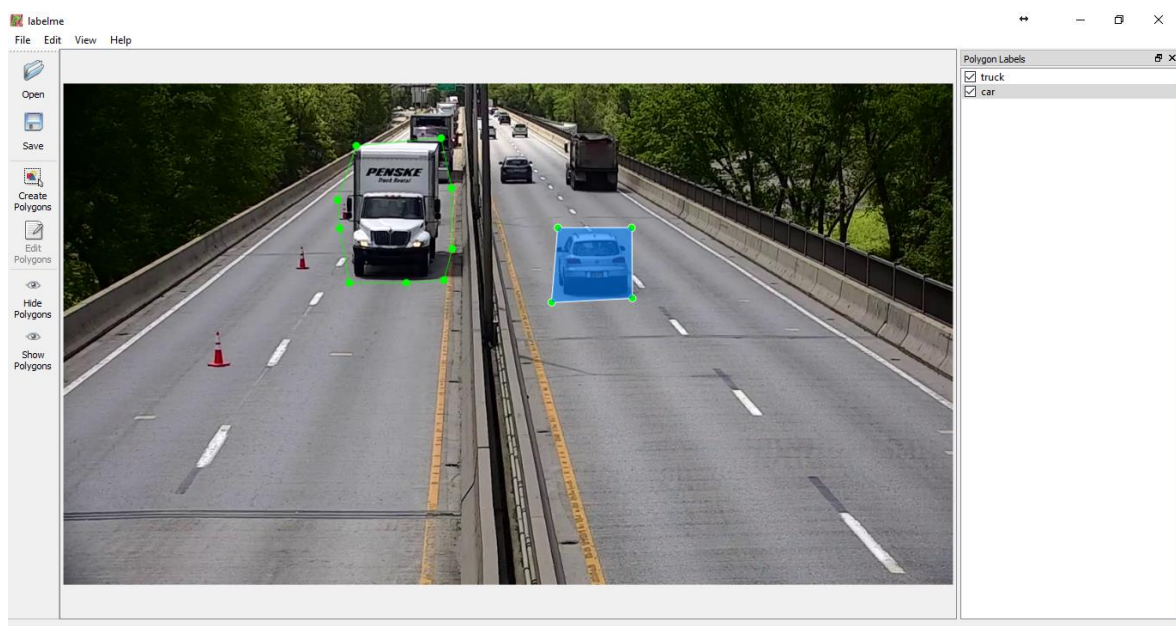
Slika 1.4 Izgled aplikacije ImageAnnotation s dodavanjem pravokutnika

Inačica koja radi s maskama dostupna je i testirana na operacijskom sustavu *Windows*, dok inačica koja radi s pravokutnicima nije testirana na *Windowsu* već samo na *Linuxu*. Inačica s pravokutnicima može se pokrenuti na *Windowsima*, no moramo je sami *buildat*. Za to su nam potrebne razne dodatne biblioteke, pa je u sklopu ovog rada nećemo testirati. Što se tiče druge inačice, također je dosta jednostavna kao i *LabelImg*. Na slici označimo objekt i

odmah nam se u istoj mapi gdje se nalazi i slika, stvori nova datoteka, ekstenzije .png koja predstavlja masku za označeni objekt. Aplikacija je dosta *bugovita*, npr. ako smanjimo prozor aplikacije dio slike postane crne boje i kako mičemo pokazivač miša izvan samog prozora, tako se na crnom dijelu slike iscrtavanju linije. Ideja same aplikacije je dobra, no potrebno je još rada na njoj da bi ju korisnik neometano mogao koristiti i kako bi korisniku koji priprema trening set ubrzala posao.

1.2.3. LabelMe

Labelme aplikacija vrlo je slična *LabelImg* aplikaciji po izgledu. No za razliku od već opisane *LabelImg*, ova podržava označavanje objekata poligonima, a ne pravokutnicima. Output ne dobijemo u XML datoteci već u JSON datoteci. Također takvu JSON datoteku ne možemo prikazati pomoću same aplikacije, što bi nam bilo vrlo korisno. Nema podržane prečace pa je rad s njom vrlo spor, pogotovo kada treba označiti nekoliko tisuća slika sa nekoliko stotina tisuća objekata na njima.



Slika 1.5 Izgled aplikacije LabelMe

1.2.4. Ostale aplikacije

Naveden je samo dio aplikacija, osim njih postoje još razne, neke od njih su ostvarene kao web aplikacije, a neke kao desktop. Popis trenutno dostupnih aplikacija može se vidjeti na slijedećem linku: https://en.wikipedia.org/wiki/List_of_manual_image_annotation_tools.

1.2.5. Nedostaci postojećih aplikacija

Većinom opisane aplikacije ne podržavaju rad samo s tipkovnicom što znatno ubrzava rad korisniku, već samo rad s mišem. Također prilikom učitavanja foldera s više od cca. 50000 slika, većina aplikacija se smrzne jer ima problema s obradom velikih količina podataka. Još jedan od nedostataka je što se mogu označiti objekti i stvoriti novi trening setovi, no ne mogu se učitati već postojeći setovi koji imaju anotacije u određenom obliku uz sebe. U skoro sve aplikacije korisnik mora sam upisati klasu objekta, što kod velike količine podataka postaje zamorno i nije efektivno. Nijedna od aplikacija ne podržava istovremeno i organizaciju odnosno manipulaciju setovima.

1.3. Anotacije potrebne za izradu trening seta

Aplikacija mora moći dodijeliti objektu:

- Neku od predefiniranih klasa, npr. automobil, kamion, kamionet, kombi, motor i sl. odnosno mora moći učitati popis klasa kroz konfiguracijsku datoteku
- Boju objekta
- Zastavicu je li objekt teško prepoznatljiv
- Zastavicu preklapa li se objekt s nekim drugim objektom na slici pa nije cijeli vidljiv
- Zastavicu je li objekt cijeli na slici ili se vidi djelomično

Osim anotacija pridijeljenih objektu neke slike, i samom datasetu bi se trebale moći dodati anotacije:

- Kakav je kut kamere kojim su snimljene slike u datasetu
- Visina kamere
- Doba dana kada su slike u setu snimane
- Vremenski uvjeti koji prevladavaju u setu
- Kvaliteta slike, jesu li slike oštre ili mutne i sl.

2. Tehnologije korištene u razvoju aplikacije

Aplikacija za pridjeljivanje bilješki i tekstualnih oznaka odnosno *XTag* razvijena je u programskom jeziku Java. U nastavku slijedi opis tehnologija korištenih u njenom razvoju.

2.1. JavaFX

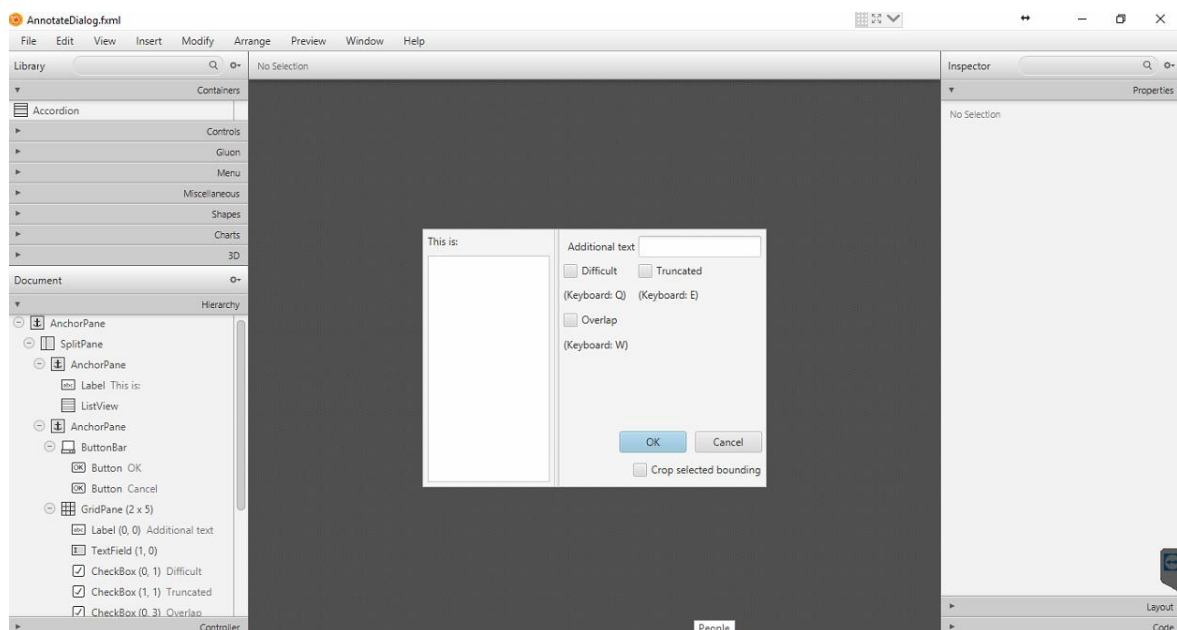
JavaFx je skup grafičkih i medijskih paketa koji razvojnim programerima omogućuje dizajniranje, stvaranje, testiranje, ispravljanje i implementaciju bogatih korisničkih aplikacija koje dosljedno rade na različitim platformama (Pawlan, 2013). Napisan je kao Java API. JavaFx je trebala zamijeniti Swing kao standardnu GUI biblioteku za Java SE, ali obje će biti uključene u doglednoj budućnosti. Budući da je JavaFX napisana kao Java API, JavaFX kod može referencirati bilo koju drugu Java biblioteku. Izgled aplikacija pisanih u ovom aplikacijskom programskom sučelju može se prilagoditi i urediti CSS-om. CSS razdvaja izgled i stil od implementacije pa se razvojni programeri mogu usredotočiti na kodiranje logike dok grafički dizajneri mogu lako prilagoditi izgled pomoću CSS-a. Kako bi se odvojili UI i *back-end* logika, prezentacijski aspekt korisničkog sučelja može se razviti kroz FXML skriptni jezik, a Java kod koristimo za aplikacijsku logiku. FXML je XML orijentirani skriptni jezik koji je razvila Oracle korporacija za dizajn JavaFX aplikacija. Korisničko sučelje može se razvijati i tradicionalno unutar Java koda ako neko tako preferira, ali dizajn odvojen u FXML-u je čišći i pregledniji način pisanja. Ako, pak, osoba koja razvija aplikaciju preferira razvoj korisničkog sučelja bez pisanja koda, može koristiti *Scene Builder*. Kroz njega osoba može dizajnirati izgled, i kako ga dizajnira tako *Scene Builder* stvara FXML kod koji se može prenijeti u IDE, a onda razvojni programer dalje gradi aplikacijsku logiku neometano. JavaFX API-ji su potpuno integrirani u JRE i JDK. Zbog toga što je JDK dostupan za sve veće *desktop* platforme (*Windows, MAC OS, Linux*), JavaFX aplikacije kompajlirane na JDK 7 i kasnije također se pokreću na svim glavnim desktop platformama. Podrška za ARM platforme također je dostupna od verzije 8 na dalje. JDK za ARM uključuje osnovu, grafičke i kontrolne komponente JavaFX-a. Kompatibilnost s više platformi omogućuje dosljednost i razvojnim programerima i korisnicima. Oracle osigurava sinkronizirana izdanja i ažuriranje za sve platforme te bogatu korisničku podršku. Primjer kako izgleda FXML (Slika 2.1) kojeg je generirao *Scene Builder* te izgled samog *Scene Buildera* (Slika 2.2) nalazi se u nastavku.


```

<?xml version="1.0" encoding="UTF-8"?>
<import javafx.scene.control.Button>
<import javafx.scene.control.ButtonBar>
<import javafx.scene.control.CheckBox>
<import javafx.scene.control.Label>
<import javafx.scene.control.ListView>
<import javafx.scene.control.SplitPane>
<import javafx.scene.control.TextField>
<import javafx.scene.layout.AnchorPane>
<import javafx.scene.layout.ColumnConstraints>
<import javafx.scene.layout.GridPane>
<import javafx.scene.layout.RowConstraints>
</import>
<AnchorPane prefHeight="300.0" prefWidth="400.0" styleSheets="@darkTheme.css" xmlns="http://javafx.com/javafx/8.0.141" xmlns:fx="http://javafx.com/fxml/1" fx:controller="fer.hr.telegra.view.AnnotateDialogController">
  <children>
    <SplitPane dividerPositions="0.38944723618090454" layoutX="87.0" layoutY="164.0" prefHeight="400.0" prefWidth="600.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
      <items>
        <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="160.0" prefWidth="180.0">
          <children>
            <Label layoutX="14.0" layoutY="14.0" text="This is:" AnchorPane.leftAnchor="5.0" AnchorPane.topAnchor="5.0" />
            <ListView fx:id="listOfAnnotations" layoutX="39.0" layoutY="111.0" prefHeight="200.0" prefWidth="200.0" AnchorPane.bottomAnchor="5.0" AnchorPane.leftAnchor="5.0" AnchorPane.rightAnchor="5.0" AnchorPane.topAnchor="30.0" />
          </children>
        </AnchorPane>
        <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="160.0" prefWidth="180.0">
          <children>
            <ButtonBar layoutX="289.0" layoutY="344.0" prefHeight="43.0" prefWidth="237.0" AnchorPane.bottomAnchor="30.0" AnchorPane.leftAnchor="50.0" AnchorPane.rightAnchor="5.0">
              <buttons>
                <Button defaultButton="true" mnemonicParsing="false" onAction="#handleOK" prefWidth="100.0" text="OK" />
                <Button mnemonicParsing="false" onAction="#handleCancel" prefWidth="100.0" text="Cancel" />
              </buttons>
            </ButtonBar>
            <GridPane layoutX="20.0" layoutY="72.0" AnchorPane.bottomAnchor="150.0" AnchorPane.leftAnchor="5.0" AnchorPane.rightAnchor="5.0" AnchorPane.topAnchor="5.0">
              <columnConstraints>
                <ColumnConstraints hgrow="SOMETIMES" maxWidth="110.0" minWidth="10.0" prefWidth="87.0" />
                <ColumnConstraints hgrow="SOMETIMES" maxWidth="147.0" minWidth="10.0" prefWidth="143.0" />
              </columnConstraints>
              <rowConstraints>
                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
              </rowConstraints>
              <children>
                <Label text="Additional text" GridPane.halignment="CENTER" GridPane.valignment="CENTER" />
                <TextField fx:id="additionalText" GridPane.columnIndex="1" />
                <CheckBox fx:id="difficult" mnemonicParsing="false" text="Difficult" GridPane.rowIndex="1" />
                <CheckBox fx:id="truncated" mnemonicParsing="false" text="Truncated" GridPane.columnIndex="1" GridPane.rowIndex="2" />
                <CheckBox fx:id="overlap" mnemonicParsing="false" text="Overlap" GridPane.rowIndex="3" />
                <Label text="(Keyboard: Q)" GridPane.rowIndex="2" />
                <Label text="(Keyboard: E)" GridPane.columnIndex="1" GridPane.rowIndex="3" />
                <Label text="(Keyboard: W)" GridPane.rowIndex="4" />
              </children>
            </GridPane>
            <GridPane layoutX="20.0" layoutY="422.0" AnchorPane.bottomAnchor="5.0" AnchorPane.leftAnchor="50.0" AnchorPane.rightAnchor="5.0">
              <columnConstraints>
                <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
              </columnConstraints>
            </GridPane>
          </children>
        </AnchorPane>
      </items>
    </SplitPane>
  </children>
</AnchorPane>

```

Slika 2.1 Primjer FXML dokumenta



Slika 2.2 Primjer dizajniranja u Scene Builderu

2.1.1. Glavne značajke

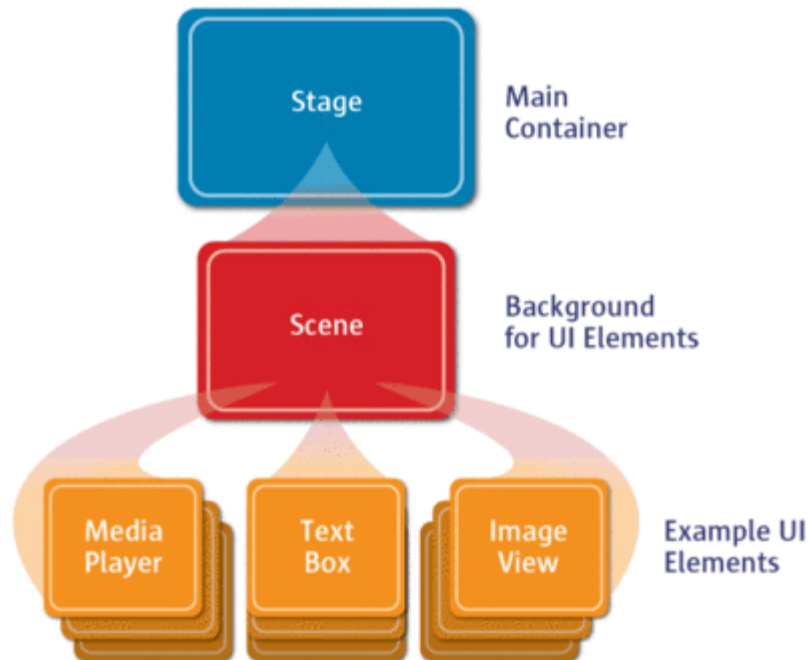
Osim spomenutih osnovnih značajki kao što su Java API-ji, FXML, *Scene Builder* i CSS u nastavku slijedi opis ostalih glavnih značajki.

- **Webview** – web komponenta koja koristi WebKitHTML kako bi omogućila ugrađivanje web stranica u JavaFX aplikaciju. JavaScript koji se vrti u *webviewu* može pozivati Java API-je i obrnuto. U verziji 8 dodana je podrška za HTML5
- **Swing interoperabilnost** – postojeće swing aplikacije mogu biti nadograđene JavaFX značajkama, kao što su bogata reprodukcija grafičkih medija i ugrađeni web sadržaj. Također klasa `SwingNode` koja omogućuje ugrađivanje swing sadržaja u JavaFX aplikacije, dodana je u verziji 8
- **Tema *Modena*** – ova nova tema zamjenjuje *Caspian* temu kao zadanu za JavaFX aplikacije. Stara *Caspian* tema je i dalje dostupna za uporabu dodavanjem sljedeće linije koda: `setUserAgentStylesheet(STYLESHEET_CASPIAN)` u `start()` metodu aplikacije. Temu možemo raspakirati iz JavaFX jar datoteke (*jfxrt.jar*) koja se nalazi u našoj Java mapi na računalu te na taj način lako pogledati koje komponente želimo prepraviti kako bi promijenili izgled
- **3D grafičke značajke** – u verziji 8 su dodane nove API klase za 3D grafičku biblioteku: za `Shape3D` (`Box`, `Cylinder`, `MeshView`, i `Sphere` podklase), `SubScene`, `Material`, `PickResult`, `LightBase` (`AmbientLight` i `PointLight` podklase), i `SceneAntialiasing`
- **Canvas API** – Canvas API omogućuje izravno crtanje unutar područja JavaFX scene koja se sastoji od jednog grafičkog elementa
- **API za ispisivanje** – `javafx.print` paket dodan je u verziji 8 i daje javne klase za JavaFX API za ispisivanje
- ***Rich Text Support*** – JavaFX 8 donosi poboljšanu tekstualnu podršku uključujući dvosmjerne tekstove i složene tekstualne skripte kao što su tajlandski i hindski u kontrolama i više linijski te više stilski tekst u tekst čvorovima (engl. *Text node*)
- ***Multitouch* podrška** – pruža podršku za *multitouch* operacije
- **Hi-DPI podrška** – od verzije 8 podržani su Hi-DPI zasloni
- ***Hardware-accelerated graphics pipeline*** – grafika se temelji na cjevovodu (engl. *Pipeline*) te nudi glatku grafiku koja se brzo prenosi kada se koristi s podržanim grafičkim karticama ili GPU

- **Medijski motor visokih performansi** (engl. *High-performance media engine*) – medijski cjevovod podržava reprodukciju web multimedijskog sadržaja. Pruža stabilan medijski okvir (engl. *Framework*) s malim latencijama koji se temelji na *GStreamer* multimedijalnom okviru

2.1.2. Struktura JavaFX aplikacije

Glavna klasa u JavaFX aplikaciji mora naslijediti `javafx.application.Application` i mora imati između ostalog dvije glavne metode: `public void start(Stage stage)` i `public static void main(String[] args).main()` metodu nije nužno implementirati jer će se aplikacija odnosno jar datoteka normalno pokretati i bez nje, no u praksi se preporuča kako bi se mogle pokrenuti jar datoteke koje su stvorene bez *JavaFX Launchera* kao npr. kada se koristi IDE u kojem JavaFX nije u potpunosti integrirana ili kod Swing aplikacija koje imaju ugrađene JavaFX funkcionalnosti jer bez te metode neće raditi. Najvažniji dio je `start()` metoda koja se automatski poziva nakon pokretanja aplikacije. Ta metoda kao parametar prima `Stage`. Slijedeća slika ilustrira strukturu svake JavaFX aplikacije (Slika 2.3).



Slika 2.3 Struktura svake JavaFX aplikacije

Ovu strukturu možemo zamisliti najjednostavnije kao kazališnu scenu. `Stage` je glavni spremnik odnosno obično je to prozor s obrubom i tipičnim gumbima za minimiziranje, maksimiziranje i zatvaranje aplikacije. Unutar njega dodajemo `Scene` koja se može

naravno zamijeniti nekom drugom scenom, a unutar scene dodajemo stvarne JavaFX elemente kao što su `AnchorPane`, `TextBox`, `ImageView` itd.

```
Public class MainApp extends Application{
    Public static void main(String[] args){
        launch(args);
    }
    @Override
    Public void start(Stage stage){
        stage.setTitle(„Prva aplikacija“);
        Button button = new Button();
        button.setText(„Pozdravi“);
        button.setOnAction(new
EventHandler<ActionEvent>() {
            @Override
            Public void handle(ActionEvent e){
                System.out.println(„Bok!“);
            }
        });
        StackPane root = new StackPane();
        root.getChildren().add(button);
        stage.setScene(new Scene(root, 300, 250));
        stage.showAndWait();
    }
}
```

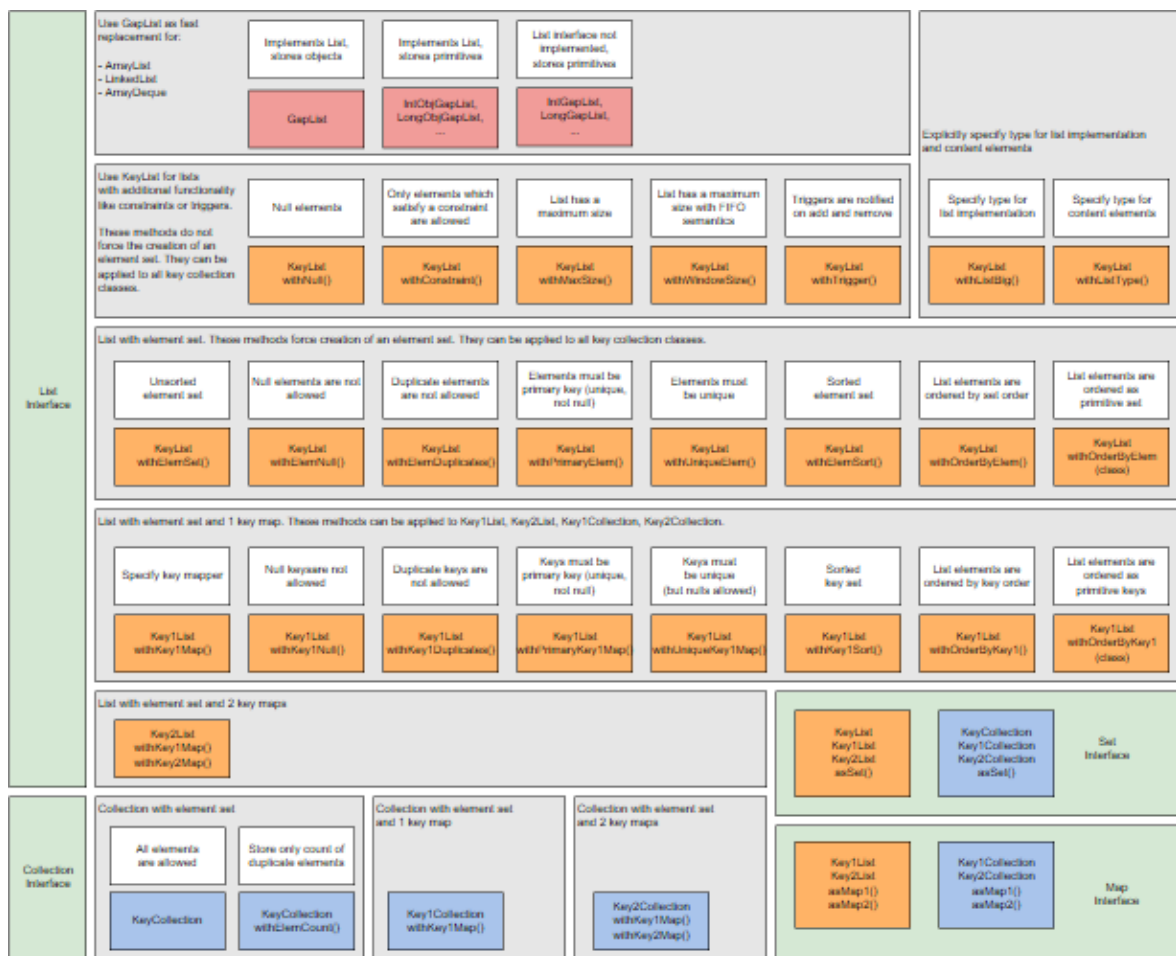
Kod 2.1 Primjer osnovne JavaFX aplikacije

U prikazanom kodu (Kod 2.1) pokazan je primjer jednostavne JavaFX aplikacije. Stvorili smo pozornicu kojoj smo dali naslov, gumb kojim ćemo nešto ispisati te smo tom gumbu dodali `EventHandler` kada ga pritisnemo da ispiše poruku „Bok!“. Kao spremnik za naš gumb odabrali smo `StackPane` te mu dodali gumb, zatim smo stvorili novu scenu dimenzija 300x250 i tu scenu stavili u pozornicu koju smo naposljetku i prikazali. Ovaj vrlo jednostavan primjer ilustrira strukturu koda za stvaranje JavaFX aplikacija.

2.2. Brownies Collections

Brownies Collections je biblioteka visokih performansi za Javu. Ona pruža svojevršno poboljšanje standardnih java *Collections* okvira (Mauch). U njoj je naglasak na poboljšanju iskoristivosti memorije, brzim manipulacijama nad listama te podrškom za baratanje velikom količinom podataka.

`GapList` kombinira snagu `ArrayList` i `LinkedList`. Implementirana je kako bi ponudila djelotvoran slučajni pristup elementima po indeksu kao što to `ArrayList` čini, a istodobno i učinkovito dodavanje i uklanjanje elemenata na početak i kraj kao što to nudi `LinkedList`. Isto tako, iskorištava lokalitet reference koja se često vidi u aplikacijama radi daljnjeg poboljšanja performansi, npr. za iteriranje kroz listu. Druga bitna implementacija liste je `BigList`. To je lista optimizirana za pohranu velikog broja elemenata. Pohranjuje elemente u blokove fiksne veličine tako da za dodavanje ili uklanjanje samo nekoliko elemenata treba biti presloženo. Blokovi se dijele ili spajaju po potrebi i spremljeni su u obliku strukture stabla (engl. *Tree*) radi bržeg pristupa. Kopiranje takve liste vrlo je učinkovito jer je implementirano na *copy-on-write* pristupu. *Copy on write* (CoW) je tehnika upravljanja resursima koja se koristi u računalnom programiranju kako bi se učinkovito implementirale operacije „duplicirati“ i „kopirati“ na promjenjiva sredstva (Bovet, 2002). Obje liste su dizajnirane da se koriste kao zamjena za `ArrayList`, `LinkedList` i `ArrayDeque` implementirajući sve njihove metode. Osim toga, dostupne su i mnoge druge korisne metode koje nudi zajednička apstraktna klasa `IList`. Postoje specijalizirane implementacije lista za sve primitivne tipove podataka. Kako se skladištenje ostvaruje pomoću polja primitivnih tipova podataka, štedi se memorija i vrijeme izvršavanja se ubrzava. Te se klase nazivaju `IntGapList` / `IntBigList`, `LongGapList` / `LongBigList` itd. Za svaku listu primitivnih tipova podataka postoji i omotač (engl. *Wrapper*) koji omogućuje pristup primitivnim podacima kroz standardno sučelje `List`. Na taj način štedi se memorija. Te se klase nazivaju `IntObjGapList` / `IntObjBigList`, `LongObjGapList` / `LongObjBigList`, itd. Da bi se povećala produktivnost ključevi i ograničenja dodani su kolekcijama na ortogonalan i deklarativan način. Te značajke nude se u klasama koje implementiraju *Collection* (Klase `KeyCollection`, `Key1Collection`, `Key2Collection`) i sučelje `List` (klase `KeyList`, `Key1List`, `Key2List`).



Slika 2.4 Dijagram klasa Brownies Kolekcije (<http://www.magicwerk.org/page-collections-classes.html>)

2.3. OpenCV

OpenCV (*Open Source Computer Vision*) je biblioteka programskih funkcija uglavnom usmjerenih na računalni vid (engl. *Computer Vision*). Izvorno ju je razvio Intel, a sada ga održava tvrtka Itseez (Wikipedia). Biblioteka je višeplatformska i besplatna za korištenje pod licencom otvorenog kosa BSD. OpenCV je napisan u C++ i njegovo primarno sučelje je u C++, ali ipak zadržava manje sveobuhvatno iako opsežno starije C sučelje. Postoje verzije biblioteke u Pythonu, Javi i MATLAB-u odnosno OCTAVI. API za ta sučelja postoji opisan u online dokumentaciji. Omotači (engl. *Wrappers*) u drugim jezicima kao što su C#, Ch, Perl, Haskell i Ruby razvijeni su kako bi potaknuli usvajanje šire publike. Svi novi algoritmi razvijaju se u C++ sučelju. S obzirom da je aplikacija za pridjeljivanje bilješki i tekstualnih oznaka razvijena u Javi, zanima nas OpenCV Java sučelje.

OpenCV Java biblioteku možemo dodati u svom IDE-u kao korisničku biblioteku te je koristiti u svim svojim projektima.

```
public class HelloCV{  
    public static void main(String[] args){  
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);  
        Mat mat = Mat.eye(3, 3, CvType.CV_8UC1);  
        System.out.println("mat = " + mat.dump());  
    }  
}
```

Kod 2.2 Primjer korištenja OpenCV biblioteke

Kako bi koristili biblioteku nakon što smo je dodali u projekt trebamo je učitati, to činimo dodavanjem slijedeće linije na početak `main()` metode kako je prikazano (Kod 2.2): `System.loadLibrary(Core.NATIVE_LIBRARY_NAME)`. Biblioteka podržava rad s videom što je primarni razlog njenog korištenja u razvoju ove aplikacije.

2.4. Pascal VOC

Pascal VOC je projekt koji pruža standardizirane skupove slikovnih podataka za prepoznavanje klase objekata (engl. *object class recognition*), također pruža zajednički set alata za pristup skupovima podataka i anotacijama. Omogućuje procjenu i usporedbu različitih metoda. Vodi izazove koji vrednuju performanse pri prepoznavanju klase objekta. Izazovi su se vodili u razdoblju od 2005. godine do 2012. godine, no setovi podataka, iako je samo natjecanje završeno, dostupni su na njihovim stranicama. 2005. godine kada je izazov pokrenut prvi put postojale su samo 4 klase u anotacijama (bicikl, automobil, motocikl i čovjek) a skup podataka sadržavao je 1578 slika s 2209 anotiranih objekata. 2006 dodano je novih 6 klasa, a 2007 je broj klasa narastao na 20 i od tad ostao na tom broju. Klase su slijedeće:

- Osoba: osoba
- Životinje: ptica, mačka, krava, pas, konj, ovca
- Vozila: avion, bicikl, brod, bus, automobil, motocikl, vlak
- Unutrašnjost: boca, stolac, stol za objed, posuda, kauč, tv/monitor

Klase su prevedene na hrvatski jezik, u praksi se koriste engleski termini kako bi format bio standardiziran. 2007 godine skup podataka sadržavao je 9963 slike s 26 640 anotiranih

objekata, dok je zadnje 2012 godine broj slika narastao na 11530 s 27450 ROI anotiranih objekata i 6929 segmentacije.

2.4.1. Pascal VOC format

Kako bi svi sudionici u izazovu mogli rezultate svojih prepoznavanja i klasifikacije objekata evaluirati stvoren je standardizirani format ispisa anotacija pomoću XML-a.

```
<annotation verified="no">
  <folder>images</folder>
  <filename>20171215T052023.802_0</filename>
  <path>D:/work_in_progress/TaggingAppTest/DataSet2/images/20171215T052023.802_0.bmp</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>720</width>
    <height>576</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>car</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <Difficult>0</Difficult>
    <bndbox>
      <xmin>386</xmin>
      <ymin>50</ymin>
      <xmax>507</xmax>
      <ymax>188</ymax>
    </bndbox>
  </object>
</annotation>
```

Slika 2.5 Struktura Pascal VOC XML datoteke

Izgled i struktura Pascal VOC formata dana je slikom iznad (Slika 2.5). Korijenski element je `<annotation>`. Potrebno je da XML sadrži elemente `<folder>` u kojem će biti ime mape u kojoj se nalazi slika, `<path>` odnosno puna putanja do slike, `<source>` unutar kojeg je `<database>` koji označava bazu u kojoj se slika nalazi, ukoliko nema baze piše se *Unknown*, `<size>` s veličinom slike, zastavicu `<segmented>` koja označava je li slika i segmentirana, a ne samo anotirana s pravokutnicima te jedan ili više elemenata `<object>` koji predstavljaju anotacije odnosno pravokutnike na slici. Svaka anotacija mora sadržavati element `<name>` koja predstavlja klasu, zastavicu `<truncated>` koja označava je li vozilo djelomično vidljivo na slici, zastavicu `<difficult>` koja nam govori je li objekt teško prepoznatljiv na slici, npr. zbog loše kvalitete slike i sl. te element `<bndbox>` koji predstavlja granični okvir (engl. *Bounding box*) koji u sebi sadrži koordinate piksela gdje se pravokutnik na slici nalazi.

3. Programsko rješenje aplikacije

Aplikacija za pridjeljivanje bilješki i dodavanje tekstualnih oznaka razvijena je u programskom jeziku Java, kako je već i spomenuto. U nastavku slijedi opis i struktura programskog rješenja aplikacije.

3.1. Zahtjevi koje aplikacija ispunjava

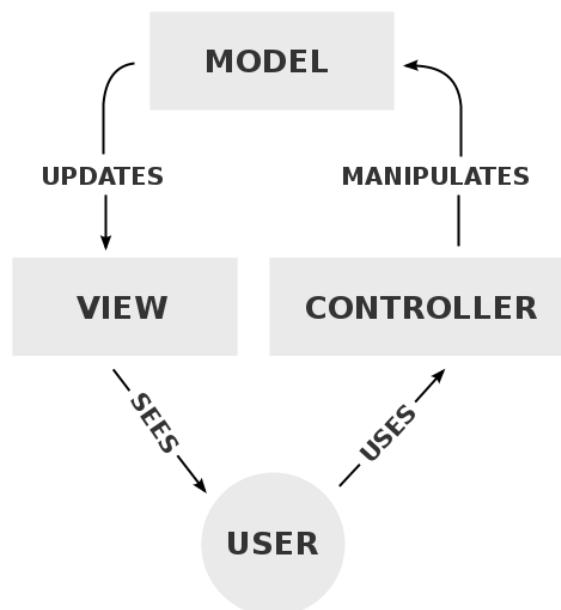
U poglavlju 1.3 Anotacije potrebne za izradu trening seta navedene su anotacije na razini objekta i na razini skupa podataka koje izlaz iz aplikacije mora podržavati. Osim toga aplikacija mora omogućiti slijedeće funkcionalnosti:

- Ulaz u aplikaciju su skupovi slikovnih podataka, sukladno tome mora omogućavati manipulaciju istima, dodavanje, uređivanje, brisanje skupova podataka sadržanih u nekoj bazi
- Slike koje su ulaz često su zapravo dio videa te aplikacija mora imati funkcionalnost za izvoz svakog n-tog okvira odnosno slike iz videa te time stvaranje novog skupa slikovnih podataka
- Skupovi podataka često su jako veliki (velik broj slika s višestrukim anotacijama objekata po slici) te mora moći baratati skupom velikih podataka (engl. *Big Data*), odnosno bitno je zadržavanje optimalnih performansi prilikom učitavanja i manipulacijom takvih skupova podataka
- Označavanje objekata pravokutnicima te manipulacija samim pravokutnicima kao i dodavanje bilješki tim pravokutnicima i manipulacija dodanim bilješkama
- Aplikacija mora imati automatsko spremanje podataka, odnosno korisnik ne treba nakon svake promjene pritiskati gumb pohrani (engl. *Save*)
- Izvoz označenih objekata i pridijeljenih bilješki u PASCAL VOC XML formatu
- Osim izvoza podataka aplikacija mora moći i učitati XML datoteke za neki skup podataka ako takve datoteke odnosno anotacije postoje. Drugim riječima ukoliko imamo skup podataka sa slikama i pridruženim XML datotekama i aplikaciji damo putanju do obje mape, ona će prikazati na ekranu sliku i odgovarajuće pravokutnike te omogućiti daljnju manipulaciju kao da su oni stvoreni kroz samu aplikaciju
- Treba podržavati prečace za rad samo s tipkovnicom kako bi korisniku ubrzali proces stvaranja skupa podataka

- Korisnik ne upisuje klase ručno (kao što je to slučaj kod većine postojećih aplikacija za anotiranje) već bira klasu iz ponuđene liste i na taj način skraćuje vrijeme samog procesa pridjeljivanja bilješki.
- Moguća je konfiguracija parametara kroz aplikaciju
- Pregled statistike koliko kojih klasa anotacija ima u pojedinom setu

3.2. Struktura i modeliranje aplikacije

Aplikacija je razvijena na principu Model – izgled – kontrolor (engl. *Model – View – Controller*, skraćeno MVC). MVC je programski uzorak koji se obično koristi za dizajniranje, modeliranje i razvoj korisničkih aplikacija. On odvaja aplikaciju u 3 povezana dijela: Model koji sadrži klase koje predstavljaju model objekata podataka korištenih u aplikaciji, izgled u kojem su klase koje su zadužene za samo grafičko korisničko sučelje te kontrolor dio u kojem su klase koje upravljaju modelom podataka. Odnos ta 3 dijela dan je slikom ispod (Slika 3.1).



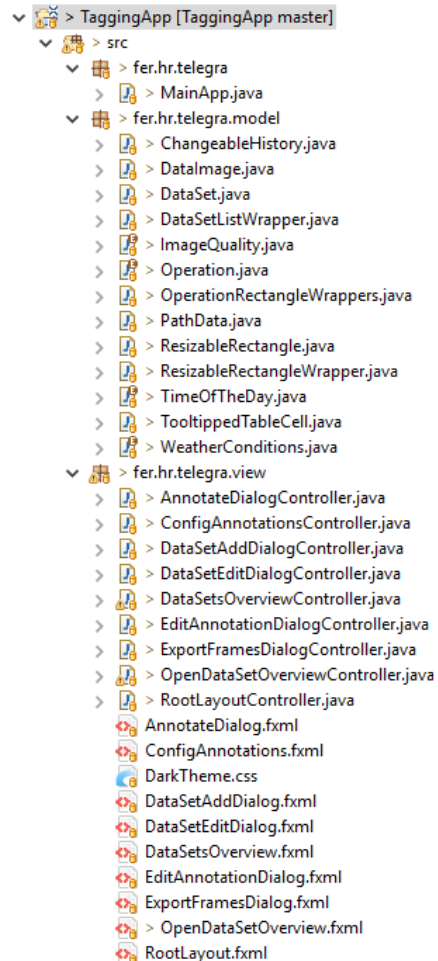
Slika 3.1 Odnos MVC-a

Paketi koji odgovaraju MVC-u ove aplikacije su redom kako slijedi:

- fer.hr.telegra.model
- fer.hr.telegra.view

- fer.hr.telegra (kontrolor paket)

Paket izgled (engl. *View*) osim klasa koje sadrže grafičko korisničko sučelje u obliku FXML-a, sadrži i klase koje upravljaju tim izgledom i direktno je svaka povezana s jednom FXML datotekom te iz tog razloga mora biti u istom paketu kao i FXML. Struktura aplikacije dana je slijedećom slikom (Slika 3.2).



Slika 3.2 Struktura paketa i klasa u aplikaciji

3.2.1. Model aplikacije

Četiri najvažnije klase koje predstavljaju model podataka su `DataSet`, `DataImage`, `ResizableRectangle` i `ResizableRectangleWrapper`.

Klasom `DataSet` kako i samo ime govori modeliran je skup podataka kojeg imamo u aplikaciji, odnosno u aplikaciji imamo skup takvih skupova podataka. Klasa sadrži slijedeće članske varijable:


```

private final StringProperty dataSetName;
private StringProperty dataSetImagesLocation;
private StringProperty dataSetAnnotationsLocation;
private DoubleProperty cameraAngle;
private DoubleProperty cameraHigh;
private ObjectProperty<WeatherConditions> weatherCondition;
private ObjectProperty<ImageQuality> imageQuality;
private ObjectProperty<TimeOfDay> timeOfDay;
private ObservableList<DataImage> dataSetImages =
FXCollections.observableArrayList();
private ObservableList<DataImage>
dataSetImagesWithAnnotations =
FXCollections.observableArrayList();
private ObservableList<DataImage> dataSetVerifiedImages =
FXCollections.observableArrayList();
private ObservableMap<String, Integer> annotations =
FXCollections.observableHashMap();

```

Kod 3.1 Prikaz članskih varijabla klase DataSet

Kao što se vidi iznad (Kod 3.1) za sve varijable korišteni su `Properties`. Njihovo korištenje je uobičajeno za modeliranje u JavaFX aplikacijama. Ono nam omogućuje da budemo automatski obaviješteni kada je neka od varijabli promijenjena što nam pomaže održati izgled i podatke sinkroniziranima. Klase unutar `JavaFX.beans.property` osim podrške za automatsko obavješćavanje promjena sadrže i ugrađenu podršku za povezivanje (engl. *Binding*). Povezivanje je moćan mehanizam za izražavanje direktnih odnosa između varijabli. Kada objekti sudjeluju u povezivanju, promjene koje se dogode na jednom objektu automatski će se reflektirati i na drugi objekt. Npr. povežemo li `textProperty` elemenata `TextField` i `Label`, unosom/promjenom teksta u `TextField` on će se pokazati i na `Label`. `DataSet` klasa osim uobičajenih *gettera* i *settera* ima i *property getter* za svaku člansku varijablu, što je također praksa kod korištenja JavaFXa i `Propertiesa`. U glavnoj kontrolor klasi odnosno `MainApp` glavni skup podataka nam je lista `DataSetova` ostvarena kao `ObservableList`. Ta lista je dio JavaFX kolekcija koja omogućuje praćenje promjena nad listom i na taj način osigurava sinkroniziranost izgleda i podataka. Također većina ostalih lista podataka u aplikaciji implementirana je pomoću te JavaFX kolekcije. Ova klasa sadrži 3 `ObservableListe` koje predstavljaju listu slika koje još nemaju anotacije, listu slika s anotacijama te listu verificiranih slika.

Klasa `ImageData` predstavlja jednu sliku u setu podataka.

Klasa `ResizableRectangle` nasljeđuje standardnu Java klasu `Rectangle`. Ta klasa predstavlja pravokutnik kojim ćemo označiti neki objekt na slici. Ona omogućuje korisniku crtanje, pomicanje, označavanje i mijenjanje veličine pravokutnika. Pomicanje i označavanje implementirano je tako što je unutar modela klase stvoren još jedan pravokutnik kojem su povezane koordinate i širina i visina s izvornim pravokutnikom kako slijedi u nastavku (Kod 3.2).

```
moveRect.xProperty().bind(super.xProperty());  
moveRect.yProperty().bind(super.yProperty());  
moveRect.widthProperty().bind(super.widthProperty());  
moveRect.heightProperty().bind(super.heightProperty());
```

Kod 3.2 Povezivanje propertiesa dodatnog i izvornog pravokutnika

Novom dodatnom pravokutniku dodani su `EventHandler`i koji određuju ponašanje i funkciju pomicanja i označavanja. Mijenjanje veličine ostvareno je pomoću još 8 kvadratića dodanih na rubove kao što je prikazano na slici (Slika 3.3). Ti kvadratići imaju svoje `EventHandler`e koji reagiraju na korisnikovo micanje kursorom miša te sukladno potezu mijenjanju veličinu izvornog pravokutnika. U klasi je također implementirana mogućnost mijenjanja veličine pravokutnika prema zadanom *Aspect Ratio*-u. Ako je pravokutnik crtan sa njime npr. u odnosu 1:1 što je zapravo kvadrat, klasa pamti podatak o tom odnosu kako bi se mijenjanje veličine moglo odraditi samo u zadanim okvirima.



Slika 3.3 Izgled pravokutnika za označavanje u aplikaciji

Samo crtanje pravokutnika nije implementirano kroz tu klasu, već u jednoj od kontrolor klasa te se u njoj stvara nova instanca `ResizableRectangle` i kroz konstruktor joj se predaju koordinate gdje je korisnik kliknuo za početak i kraj pravokutnika.

Klasa `ResizableRectangleWrapper` predstavlja anotacije koje su pridodane pravokutniku. Ona sadrži sve podatke koje korisnik unese. U poglavljima 3.1 Zahtjevi koje aplikacija ispunjava i 2.4.1 Pascal VOC format navedeni su podaci koje korisnik treba moći unijeti i u kojem obliku trebaju na kraju biti pohranjeni. Postoji razlika u odnosu na Pascal VOC format u smislu da je nadodano pet novih elemenata, no nedostatak istih u XML-u ne smeta aplikaciji da pročita i prikaže označne objekte i anotacije. Elementi koji su nadodani je zastavica `<overlap>` koja označava je li vozilo tj. objekt djelomično vidljiv jer je zaklonjen nekim drugim objektom (ne treba miješati s `<truncated>` koji označava da se objekt djelomično vidi jer nije stao u kadar), element `<text>` gdje korisnik po želji može i ne mora unijeti neki dodatni tekst osim imena klase koji bira iz popisa, element `<color>` koji predstavlja boju objekta koju je korisnik označio te element `<aspect_ratio>` koji čuva informaciju o gore spomenutom omjeru slike.

`ImageQuality`, `WeatherConditions` i `TimeOfTheDay` su enum klase koje sadrže vrijednosti koje se mogu pridružiti kao anotacije na razini skupa podataka.

`ToolTippedTableCell` je klasa kao i normalna klasa koja predstavlja ćeliju tablice, samo što za razliku od normalne ima alat koji prikazuje sadržaj ćelije (engl. *Tooltip*). Nasljeđuje klasu `TableCell` koja je dio JavaFX scene. Pomoću nje korisniku je omogućeno čitanje sadržaja samim prelaskom miša preko ćelije.

`DataSetListWrapper` predstavlja omotač oko liste skupova podataka kako bi se spremila u bazu podataka koja je organizirana kroz XML strukturu. Aplikacija bi se dala nadograditi na način da koristi neku pravu bazu podataka, za sad koristi XML kao jednostavnu verziju.

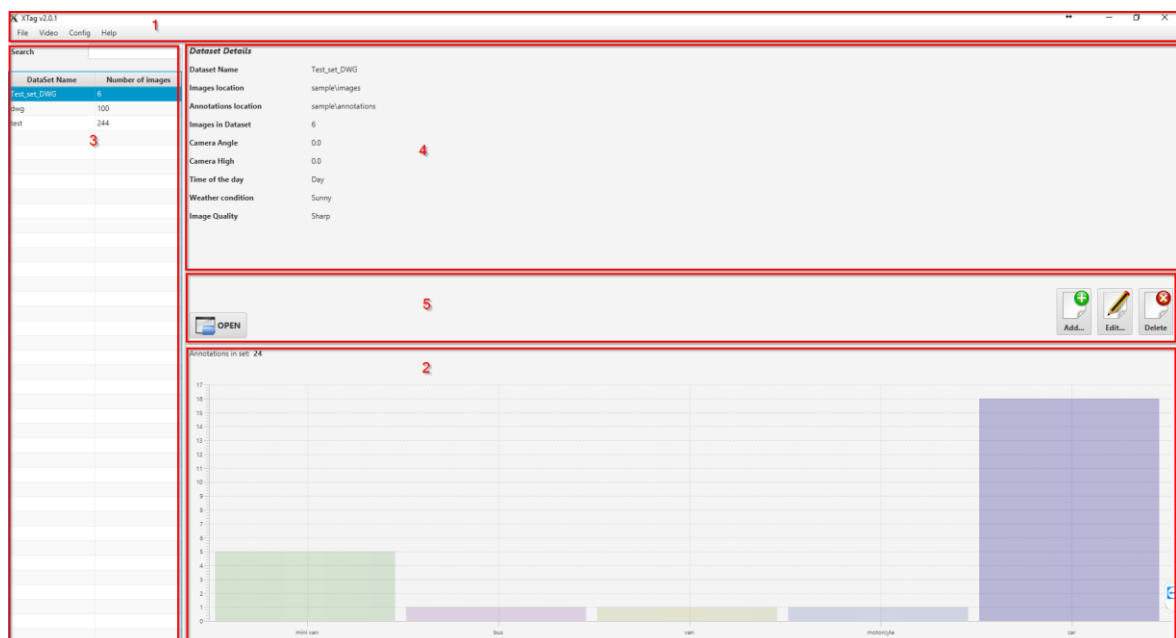
`PathData` je *Singleton* klasa koja sadržava informaciju o zadnje korištenom putu u aplikaciji. *Singleton* klasa je oblikovni obrazac koji ograničava instanciranje klase na samo jedan objekt.

3.2.2. Izgled i funkcionalnost aplikacije

Glavna kontrolor i glavna klasa općenito je `MainApp` klasa. Unutar nje se inicijalizira pozornica i scena kako je to pokazano u 2.1.2 Struktura JavaFX aplikacije. Svaka druga kontrolor klasa prima referencu na glavnu klasu kako bi mogla pozivati statičke metode koje prikazuju različite scene i dijaloge odnosno dodatne prozore osim onog glavnog u aplikaciji. Dakle ona sadrži metode za pozivanje prikaza svih prozora koji se u aplikaciji mogu pojaviti. Osim toga u njoj se vrše i sve konfiguracijske inicijalizacije.

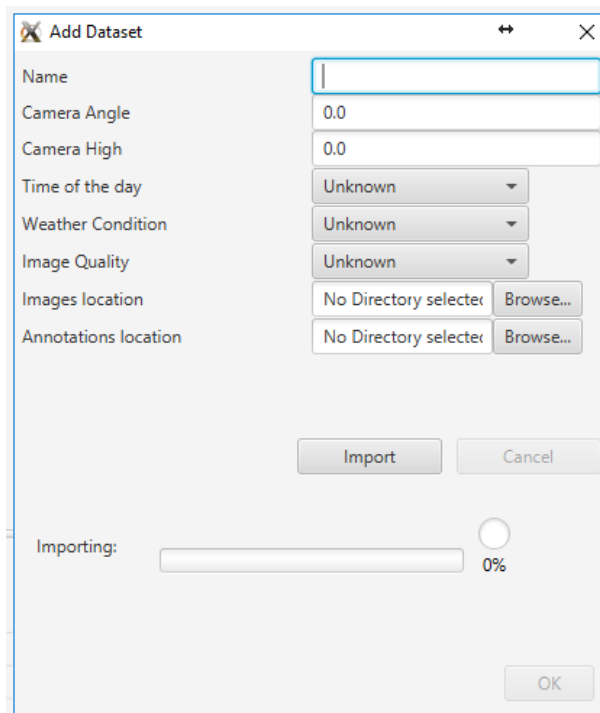
Kako je već navedeno klase koje kontroliraju ponašanje određenog izgleda definiranog u pojedinom FXML-u moraju biti u istom paketu kao i FXML jer ih u suprotnom *Scene Builder* neće moći pronaći. Zato se u ovom poglavlju opisuju izgled i funkcionalnost zajedno, a ne odvojeno.

Pokretanjem aplikacije prikazuje se `RootLayout.fxml` koji sadrži glavni izbornik, standardne gumbe za minimiziranje, maksimiziranje i zatvaranje aplikacije te se unutar njega stavljaju ostale scene (Označeno brojem 1 na slici Slika 3.4). Prva scena koja se poziva čim aplikacija starta je opisana `DataSetsOverview.fxml` datotekom i kontrolirana pomoću pripadajuće kontrolor klase `DataSetsOverviewController` (Označeno brojem 2, 3, 4 i 5 na slici Slika 3.4). Korisnik to vidi kako je prikazano sljedećom slikom (Slika 3.4).



Slika 3.4 Izgled prozora aplikacije

S lijeve strane nalazi se tablica sa svim skupovima skupova podataka koju imamo u bazi podataka (označena brojem 3). Klikom na neki od skupova podataka, njegovi podaci i detalji prikazuju se u dijelu označenim s brojem 4 i brojem 2. Broj 2 prikazuje statistiku koliko kojih klasa anotacija ima u tom setu pomoću `barChart`-a. Brojem 5 označen je dio gdje se nalaze gumbi za stvaranje novog skupa podataka, uređivanje i brisanje postojećeg te otvaranje odabranog skupa podataka. Klikom na jedan od gumba dodaj ili uredi skup podataka otvara se sličan prozor prikazan na slici ispod (Slika 3.5).



Slika 3.5 Izgled prozora za dodavanje/uređivanje skupa podataka

U njemu korisnik može unijeti podatke i anotacije na razini skupa podataka za taj skup. Glavna razlika između dodaj i uredi funkcije je ta što se prilikom dodavanja novog seta automatski uvoze sve slike (što je napad na performanse izvođenja jer postoji mogućnost velikog skupa podataka), a prilikom uređivanja se to događa samo ukoliko se mijenja lokacija slika i anotacija. Ako se ne mijenja putanja do lokacija nema smisla svaki put ispočetka uvoziti jednake slike. Prilikom učitavanja slika koristimo se već opisanom kolekcijom `BigList`.

```
BigList<File> images =
BigList.create(dir.listFiles(IMAGE_FILTER));
Iterator<File> itr = images.iterator();
```

Kod 3.3 Stvaranje `BigList` koja sadrži slike

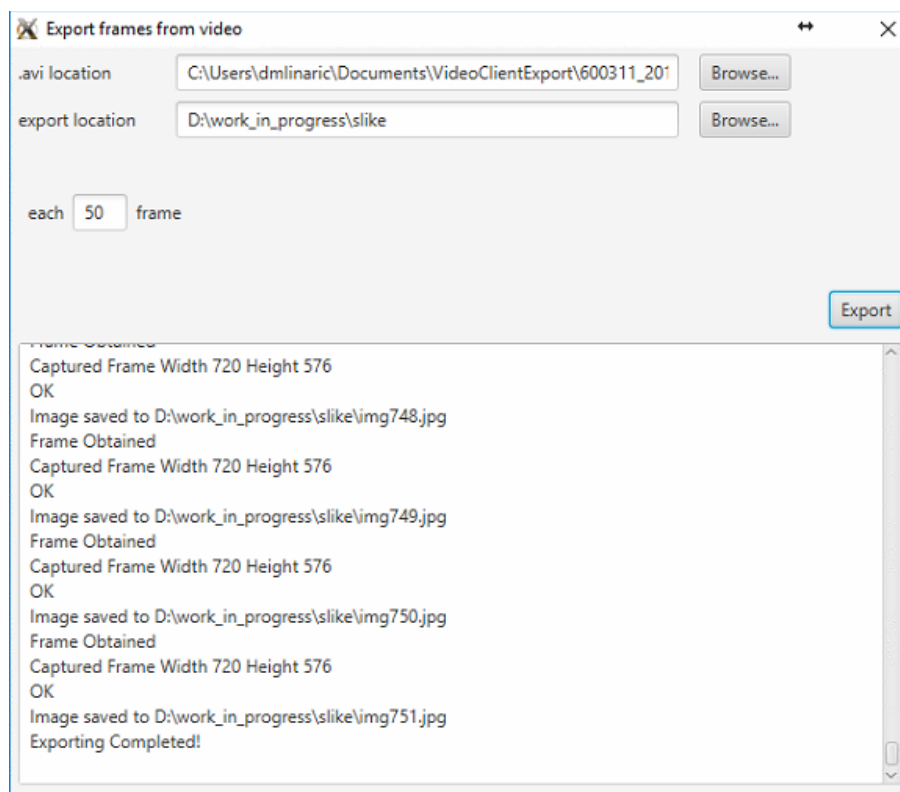
U kodu iznad (Kod 3.3) prikazano je kako se stvara lista u koju se spremaju sve slike te potom nad tom listom stvara iterator pomoću kojeg prolazimo po svim slikama i dodajemo ih u naš skup podataka. Prilikom dodavanja slika u slučaju kada smo koristili običnu listu iz Java kolekcija, a imali smo više tisuća slika, aplikacija bi se zablokirala. Koristeći `BigList` kolekciju taj problem je otklonjen. Sam proces uvoza slika i parsiranje xml datoteka izvodi se u posebnoj dretvi. Unutar kontrolor klase napravljena je dodatna unutarnja klasa koja nasljeđuje `javafx.concurrent.Task<V>` te predstavlja logiku uvoza i parsiranja koja je zatim dana novoj dretvi na izvođenje. Razlog kreiranja jednog takvog *Taska*-a je taj što

nas zanima koliko posto posla je odrađeno u određenom trenutku kako bi to mogli prikazati pomoću ProgressBara da korisnik ima uvid ukoliko ta akcija traje neko vrijeme. A ta klasa nam upravo to omogućuje. Kod u nastavku:

```
importTask.addEventHandler(WorkerStateEvent.WORKER_STATE_SUCCEEDED,
    new EventHandler<WorkerStateEvent>() {
        @Override
        public void handle(WorkerStateEvent t) {
            statusLabel.textProperty().unbind();
            statusLabel.setText("Imported!");
            ok.setDisable(false);
            okClicked = true;
        }
    });
new Thread(importTask).start();
```

Kod 3.4 Dodavanje Taska dretvi

U glavnom izborniku imamo Video padajući izbornik i u njemu *Export frames from avi...*



Slika 3.6 Izgled prozora za izvoz slika iz videa

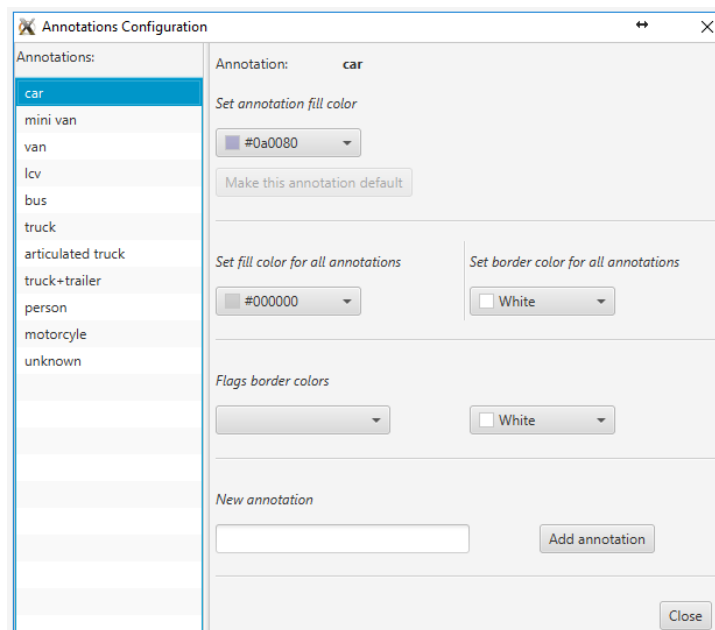
Klikom na njega otvara se prozor kao na slici iznad (Slika 3.6) modeliran ExportFramesDialog.fxml i kontrolor klasom

ExportFramesDialogController. U njemu korisnik navodi putanju do videa iz kojeg hoće izvesti slike, putanju do mape u koju će spremi slike, te broj svakog n-tog okvira koji će biti izvezen. Za manipulaciju videom korištena je VideoCapture klasa iz OpenCV biblioteke, a za konkretan okvir slike klasa Mat iz iste kolekcije koja je potom prebačena u BufferedImage objekt. S obzirom da je to proces koji traje korisnik vidi ispis koja se slika trenutno obrađuje. Da bi to bilo moguće sama radnja izvoza slika mora se izvoditi u posebnoj dretvi kako bi se izgled odnosno u ovom slučaju TextArea s logom ispisa paralelno sinkronizirao. Zato cijeli taj odsječak koda vrtimo u posebnoj dretvi i unutar nje pozivamo funkciju za ispis u TextArea element. A unutar te funkcije određujemo kada ćemo dodati tekst u ispis kako je prikazano kodom (Kod 3.5).

```
private void writeLog(String message) {
    if (Platform.isFxApplicationThread()) {
        textArea.appendText(message + "\n");
    } else {
        Platform.runLater(() ->
            textArea.appendText(message + "\n"));
    }
}
```

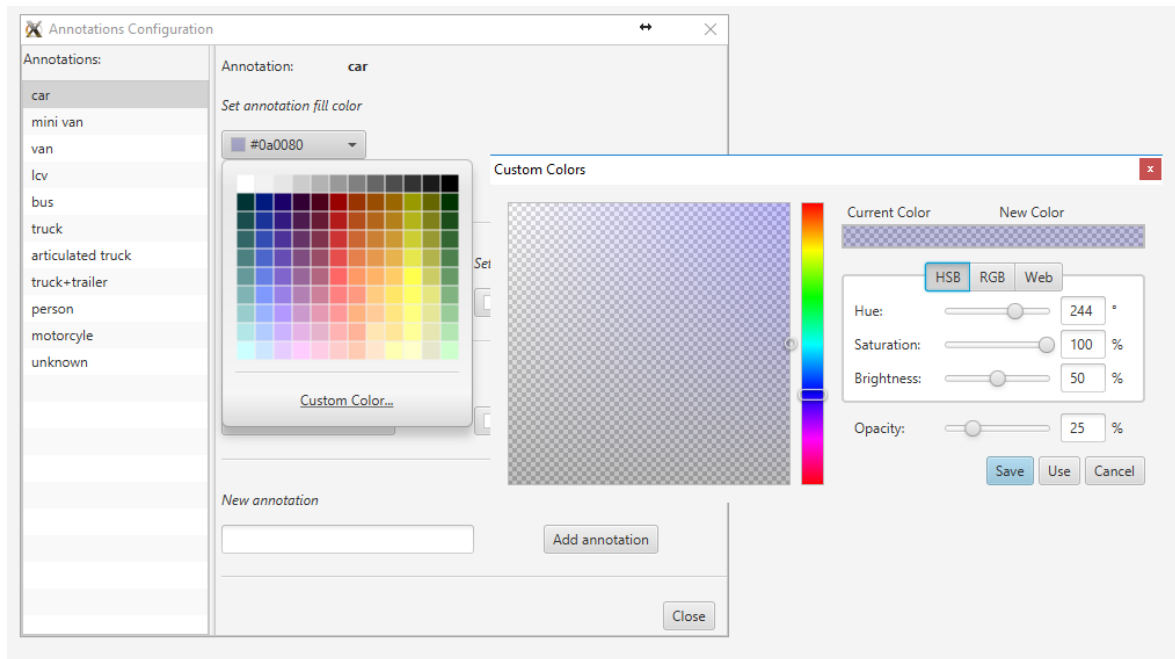
Kod 3.5 kod za osvježavanje izgleda u posebnoj dretvi

Drugi izbornik koji imamo u glavnom izborniku je *Config* i u njemu u padajućem izborniku *Annotations*. Klikom na njega otvara nam se slijedeći prozor (Slika 3.7).



Slika 3.7 Izgled prozora za konfiguriranje

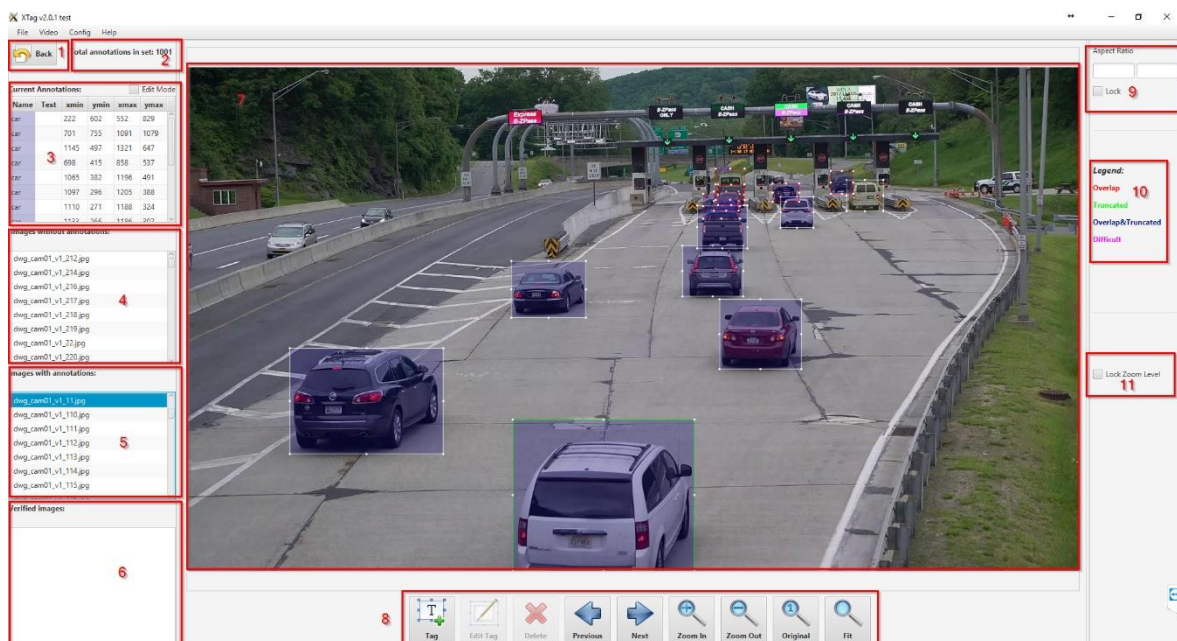
U njemu možemo konfigurirati opcije. Svakoj klasi anotacija možemo promijeniti boju ispunje pravokutnika i neprozirnost boje. Možemo svim klasama odrediti istu boju. Svim klasama također možemo odrediti boju obruba pravokutnika te za svaku od zastavica možemo definirati posebnu boju obruba pravokutnika. Takva vizualizacija je odabrana zbog lakšeg pregleda korisniku nakon što obavi anotiranje da vidi potencijalne greške. Kroz ovaj prozor korisnik također može dodati neku novu klasu i promijeniti redoslijed klasa u listi *Drag and drop* tehnikom. Primjer biranja boje dan je slijedećom slikom (Slika 3.8).



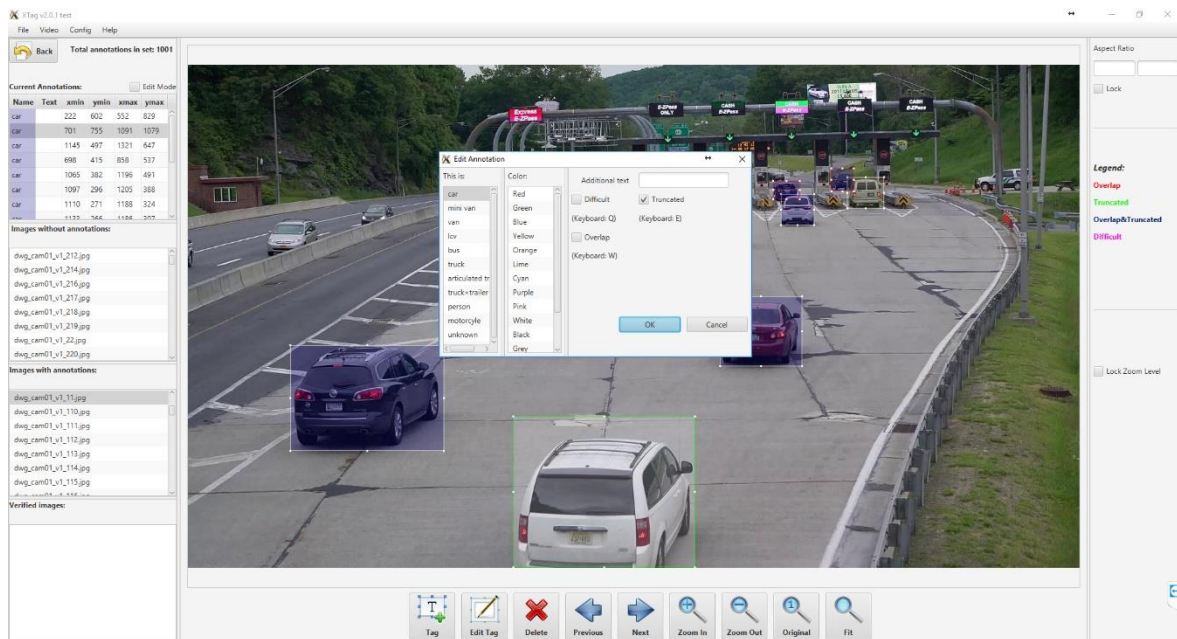
Slika 3.8 Biranje boje za određenu klasu

Ako korisnik odabere određeni skup podataka i pritisne gumb otvori (engl. *Open*) prikazuje mu se izgled kao na slici (Slika 3.9). U tom slučaju u `RootLayout` smješta se posve nova scena modelirana `OpenDataSetOverview.fxml` datotekom i `OpenDataSetOverviewController` klasom. Na dijelu označenim brojem 1 nalazi se gumb povratak (engl. *Back*) kojim se korisnik vraća natrag na pregled svih skupova podataka. Pod brojem 2 korisnik vidi ukupni broj anotacija u tom trenutku u otvorenom skupu podataka. Trenutni objekti prikazani su u tablici označenoj brojem 3. U toj tablici također je i svaka klasa obojana svojom bojom radi boljeg pregleda. U dijelu 4 nalazi se lista slika koje još nemaju anotacije, pod brojem 5 lista slika s anotacijama te pod brojem 6 lista verificiranih slika. Kada se na sliku u listi slika bez anotacija doda prva anotacija, ona se automatski prebacuje u listu slika s anotacijama, ali ostaje u pregledu slike pod brojem 7. Listanje po slikama napravljeno je na način da su prve dvije liste povezane i to tako da ako se nalazimo na prvoj slici u listi slika bez anotacija i odaberemo prethodnu sliku prebacuje

nas na zadnju sliku u listi slika s anotacijama. Dok kad se nalazimo na zadnjoj slici u listi slika s anotacijama i stisnemo sljedeću sliku, prebaciti će nas na prvu sliku u listi slika bez anotacija. Pregled slike vidi se u dijelu 7, dok su u dijelu 8 redom gumbi za označavanje objekata pravokutnikom, uređivanje odabranog pravokutnika, brisanje odabranog pravokutnika, gumbi za prikaz prethodne i sljedeće slike te gumbi za približavanje i udaljšavanje odnosno za prikaz slike u izvornoj veličini i prikaz slike da stane u scenu prozora. Stvaranjem novog ili uređivanjem postojećeg odabranog pravokutnika otvara se prozor prikazan slikom (Slika 3.10). U njemu korisnik bira klasu iz ponuđene liste (koja je zadana konfiguracijom), boju objekta, te potrebne zastavice. Postoje prečaci na tipkovnici koji korisniku ubrzavaju postupak označavanja i pridjeljivanja bilješki. Prvih 9 klasa u popisu moguće je odabrati redom tipkama 1-9, SHIFT + tipke 1-9 prvih 9 boja, zastavicama *Difficult*, *Overlap* i *Truncated* odabrati odnosno poništiti odabir redom tipkama Q, W, E, a klikom na enter ili slovo R na tipkovnici prihvaća se odabir i zatvara prozor.



Slika 3.9 Izgled koji korisnik vidi kada otvori određeni skup podataka

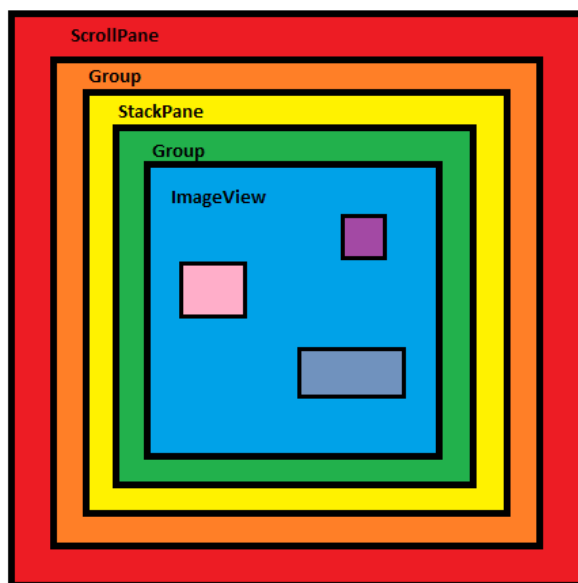


Slika 3.10 Izgled prozora za anotiranje i označenog objekta

Sve promjene koje korisnik radi uključujući stvaranje novih anotacija, uređivanje postojećih, mijenjanje veličine pravokutnika te micanje pravokutnika po sceni se automatski pohranjuju i korisnik ne mora brinuti o spremanju podataka. Koordinate piksela u tablici anotacija povezane su s trenutnim koordinatama određenog pravokutnika što znači kako mi ćemo pravokutnik po slici tako se mijenja iznos piksela u tablici. Brisanje odabranog pravokutnika osim gumbom predviđenim za to moguće je pritiskom tipke *Delete* na tipkovnici. Uređivanje pravokutnika osim odabirom gumba za to moguće je dvostrukim klikom na željeni pravokutnik. U svrhu uređivanja postoji i *Edit Mode* u koji ulazimo označimo li kvačicu pod brojem 3. Kada smo u tom načinu rada klikom na željenu anotaciju automatski nam se otvara prozor za uređivanje, te su sve ostale anotacije na slici osim odabrane, nevidljive. Također korisnik može strelicama navigirati kroz tablicu s anotacijama i za svaku anotaciju se otvara prozor za uređivanje i sakrivaju druge anotacije na slici. Ukoliko ponovno želi prikazati sve anotacije, treba kliknuti desnom tipkom miša na sliku. Razlog uvođenja ovog načina rada je taj što se nakon testiranja ispostavilo kako neki korisnici ne mogu razmišljati istovremeno o klasi objekta i o boji objekta pa je zamišljeno da prvo označe sve klase, a zatim prođu ponovno kroz cijeli set i na brzi način označe boje objekata. U dijelu 10 korisnik ima brzi pregled koja boja obruba pravokutnika odgovara kojoj zastavici tako da može netom nakon anotiranja vidjeti je li odabrao dobru zastavicu bez da otvara prozor za uređivanje. Kada se u konfiguraciji promijene te boje one se automatski mijenjaju i u toj legendi te na svim anotacijama na slici. U dijelu 9 može se postaviti zadani omjer slike, npr. ako objekti trebaju

biti kvadrati postaviti ćemo omjer na 1:1. Kada je postavljen omjer mijenjanje veličine pravokutnika moguće je samo u tom omjeru. Iz tog razloga u XML-u čuvamo informaciju o omjeru kako bi kada ponovno učitamo set podataka mogli očuvati taj omjer. Približavanje i udaljavanje na slici moguće je s gumbima predviđenim za to ili pomoću CTRL i pomicanjem kotačića na srednjem gumbu miša. Jednom kada smo u približenoj slici, po slici se mićemo pritiskom na srednji gumb miša i micanjem miša. U dijelu 11 imamo mogućnost da zaključamo trenutni postotak približavanja što je korisno ako želimo anotirati kroz set podataka samo na jednom dijelu slike. Kako bi micanje i zumiranje radilo koriste se sljedeći JavaFX elementi: `javafx.scene.image.ImageView`, `javafx.scene.Group`, `javafx.scene.layout.StackPane` i `javafx.scene.control.ScrollPane`

U prvi Group element dodamo ImageView na prvo mjesto koji će sadržavati sliku te svakom anotacijom dodaje se novi ResizableRectangle. Taj Group nam služi kako bi sadržavao izvornu sliku i sve nacrtane objekte i predstavlja drugim riječima platno (engl. *Canvas*). Da bi svaka slika bila centrirana na ekranu taj Group stavlja se unutar StackPane-a. Zatim kako bi se omogućilo micanje po slici jednom kad je zumirana, sve to zajedno omotavamo u još jedan novi Group element te na kraju sve to zajedno stavljamo u ScrollPane. Takva organizacija omogućuje laku manipulaciju nad slikom kakva je uobičajena u aplikacijama. Organizacija dana je sljedećom slikom (Slika 3.11):



Slika 3.11 Organizacija elemenata za prikaz slike i anotacija

Zaključak

Cilj ovog rada bila je izrada aplikacije za pridjeljivanje bilješki i tekstualnih oznaka video isječcima. Aplikacija je razvijena u programskom jeziku Java pomoću JavaFX API-ja. Aplikacija se može koristiti u stvarnim projektima gdje korisnici trebaju stvoriti i održavati skupove podataka za treniranje / testiranje algoritama strojnog učenja. Za mene je izrada ove aplikacije bila jako intenzivno i korisno iskustvo. Korištenje JavaFX API-ja odnosno učenje istog bitno se razlikuje od klasičnog Swing API-ja za grafičko korisničko sučelje. Aplikacija je izrađena od početka te kao takav projekt dala mi je vrijedno iskustvo u mom budućem poslovnom životu zajedno sa samom motivacijom izrade aplikacije koju sam stekla radeći kao student u razvojnom timu Telegra Solutions d.o.o. u čijoj suradnji je ovaj rad i napravljen.

Moguće su dorade aplikacije u vidu korištenja neke od verzija stvarnih baza podataka, dodavanja novih funkcionalnosti, pregledom raznih statistika nad skupovima podataka i sl.



Literatura

- [1] WIKIPEDIA, Annotation, <https://en.wikipedia.org/wiki/Annotation>
- [2] WIKIPEDIA, DNA annotation, https://en.wikipedia.org/wiki/DNA_annotation
- [3] TZUTALIN, LabelImg. Git code (2015). <https://github.com/tzutalin/labelImg>
- [4] KLASER, A. Image annotation tool with bounding boxes, https://lear.inrialpes.fr/people/klaeser/software_bbox_image_annotation, 2010.
- [5] KLASER, A. Image annotation tool with image masks, https://lear.inrialpes.fr/people/klaeser/software_image_annotation, 2010.
- [6] PAWLAN, M. What is JavaFX?, <https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm>, 2013.
- [7] MAUCH, T. Brownies Collections, <http://www.magicwerk.org/page-collections-overview.html>
- [8] BOVET, D.P. *Understanding the Linux Kernel*, 2002.
- [9] WIKIPEDIA, OpenCV, <https://en.wikipedia.org/wiki/OpenCV>

Sažetak

Kako bi se korisnicima koji stvaraju i održavaju skupove podataka za treniranje / testiranje algoritama strojnog učenja olakšao sam proces istog, razvijena je aplikacija za pridjeljivanje bilješki i tekstualnih oznaka video isječcima opisana ovim radom. U prvom dijelu dan je uvod u pojam anotacija i pregled postojećih aplikacija te navedeni nedostaci istih. Zatim su opisane tehnologije korištene prilikom izrade aplikacije, a to su: Java, JavaFX, FXML, Brownies kolekcija, OpenCV, Pascal VOC. Dok je u posljednjem poglavlju opisano konkretno programsko rješenje i njegov izgled, funkcionalnost i implementacija.

Ključne riječi:

Anotacija, Java, JavaFX, FXML, CSS, OpenCV, Pascal VOC

Summary

To help users create and maintain data sets for training / testing machine learning algorithms, an Application for Annotating and Tagging of Video Clips has been developed which is described in this final work. The first part introduces the concept of annotating and review of existing applications and their disadvantages. Then, the technologies used in creating the application are described: Java, JavaFX, FXML, Brownies Collection, OpenCV, Pascal VOC. While in the last chapter, a specific program solution and its design, functionality and implementation are described.

Keywords:

Annotation, Java, JavaFX, FXML, CSS, OpenCV, Pascal VOC

Popis kratica

XML	<i>Extensible Markup Language</i>	jezik za označavanje podataka
HTML	<i>HyperText Markup Language</i> stranica	prezentacijski jezik za izradu web
JSON	<i>JavaScript Object Notation</i>	JavaScript format za notaciju objekata
DNA	<i>Deoxyribonucleic acid</i>	Deoksiribonukleinska kiselina
API	<i>application programming interface</i>	Aplikacijsko programsko sučelje
GUI	<i>graphical user interface</i>	Grafičko korisničko sučelje
CSS	<i>Cascading Style Sheets</i>	Stilski jezik
UI	<i>User interface</i>	Korisničko sučelje
IDE	<i>Integrated Development Environment</i>	Integrirano razvojno okruženje
JRE	<i>Java SE Runtime Environment</i>	Java SE virtualno okruženje
JDK	<i>Java Development Kit</i>	Java razvojni paket
GPU	<i>graphics processing unit</i>	grafička procesorska jedinica
BSD	<i>Berkeley Software Distribution</i> podršku	Berkeley distribucija za programsku
MVC	<i>Model-View-Controller</i>	Model-izgled-kontrolor

Popis slika

Slika 1.1 Primjer anotacija na slici	2
Slika 1.2 Izgled aplikacije LabelImg	4
Slika 1.3 Izgled aplikacije ImageAnnotation s dodavanjem maski	5
Slika 1.4 Izgled aplikacije ImageAnnotation s dodavanjem pravokutnika	5
Slika 1.5 Izgled aplikacije LabelMe	6
Slika 2.1 Primjer FXML dokumenta	9
Slika 2.2 Primjer dizajniranja u Scene Builderu	9
Slika 2.3 Struktura svake JavaFX aplikacije	11
Slika 2.4 Dijagram klasa Brownies Kolekcije (http://www.magicwerk.org/page-collections-classes.html)	14
Slika 2.5 Struktura Pascal VOC XML datoteke	16
Slika 3.1 Odnos MVC-a	18
Slika 3.2 Struktura paketa i klasa u aplikaciji	19
Slika 3.3 Izgled pravokutnika za označavanje u aplikaciji	21
Slika 3.4 Izgled prozora aplikacije	23
Slika 3.5 Izgled prozora za dodavanje/uređivanje skupa podataka	24
Slika 3.6 Izgled prozora za izvoz slika iz videa	25
Slika 3.7 Izgled prozora za konfiguriranje	26
Slika 3.8 Biranje boje za određenu klasu	27
Slika 3.9 Izgled koji korisnik vidi kada otvori određeni skup podataka	28
Slika 3.10 Izgled prozora za anotiranje i označenog objekta	29
Slika 3.11 Organizacija elemenata za prikaz slike i anotacija	30

Popis kodova

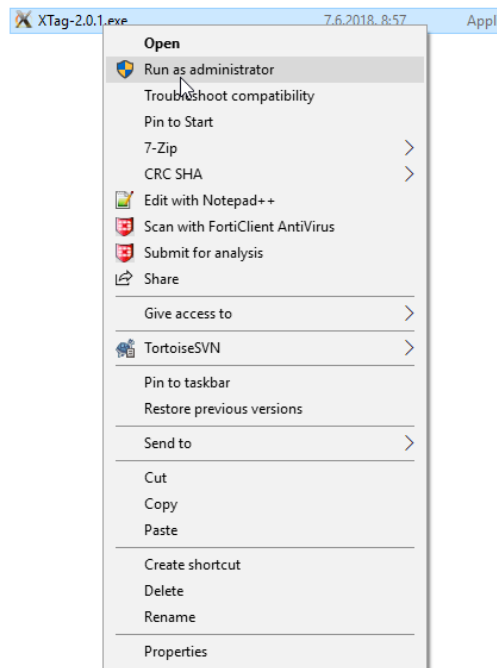
Kod 2.1 Primjer osnovne JavaFX aplikacije	12
Kod 2.2 Primjer korištenja OpenCV biblioteke	15
Kod 3.1 Prikaz članskih varijabla klase DataSet	20
Kod 3.2 Povezivanje propertiesa dodatnog i izvornog pravokutnika.....	21
Kod 3.3 Stvaranje BigListe koja sadrži slike	24
Kod 3.4 Dodavanje Taska dretvi	25
Kod 3.5 kod za osvježavanje izgleda u posebnoj dretvi.....	26

Privitak

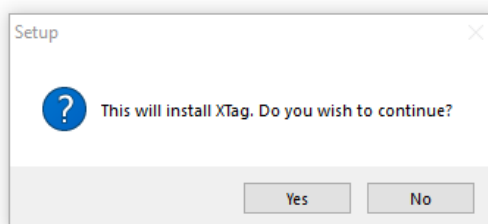
Upute za korištenje

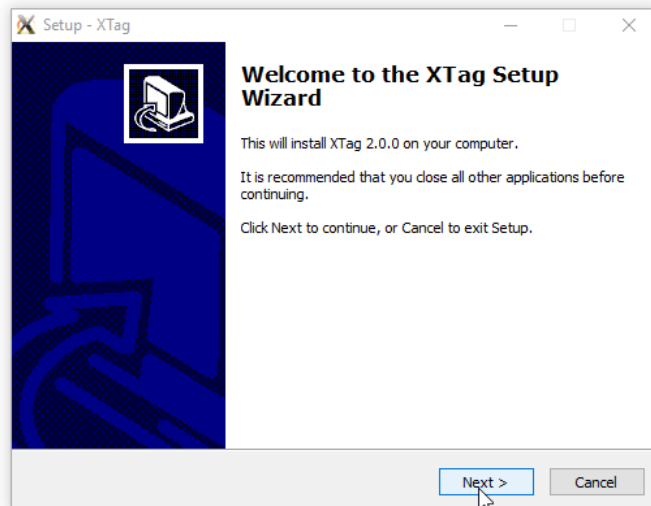
INSTALACIJA

Kliknite desnim klikom miša na instalacijsku datoteku nastavka .exe i odaberite „Run as administrator“

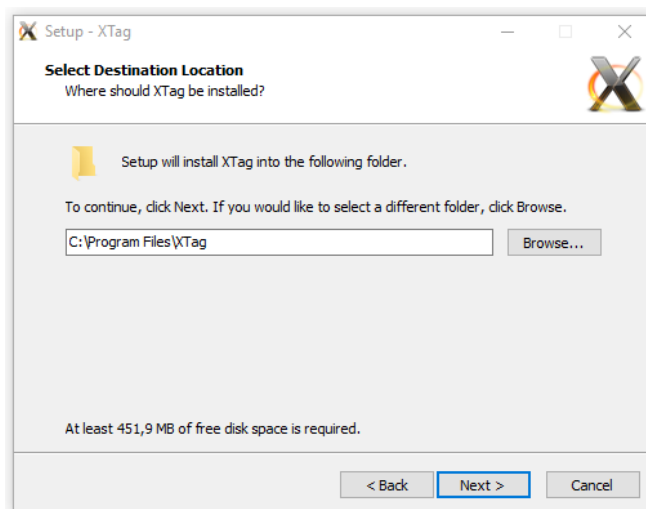


Nakon toga će vas pitati da potvrdite želite li instalirati XTag na vaše računalo, odaberite „Yes“ i potom „Next“.

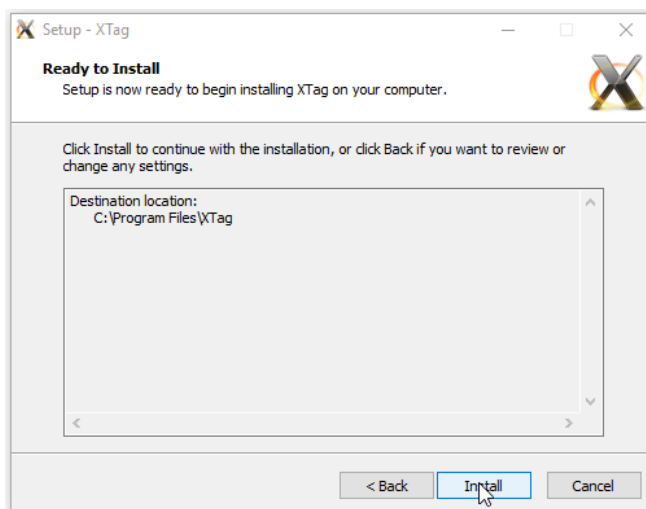




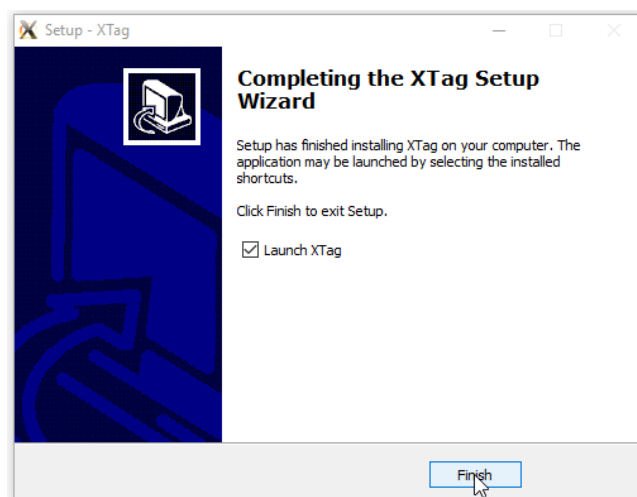
Odaberite lokaciju gdje želite instalirati aplikaciju.



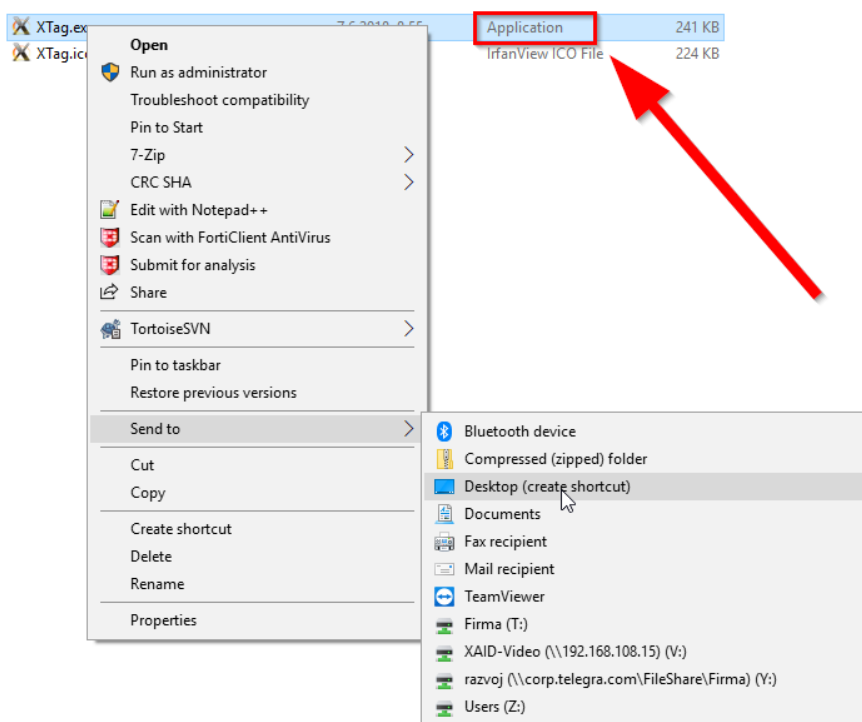
I na kraju odaberite „Install“.



Nakon uspješne instalacije odaberite „Finsih“.

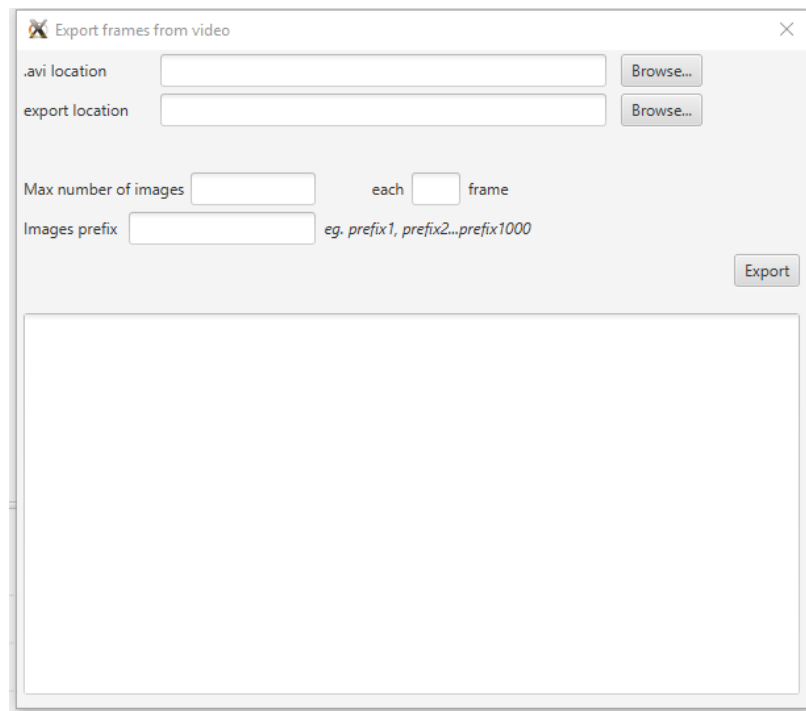


Ukoliko vam na Radnoj površini nije kreiralo ikonu za pokretanje aplikacije, navigirajte do mape gdje ste instalirali aplikaciju (Ako niste mijenjali put onda je to C:\Program Files\XTag). Kliknite desni klik na XTag.exe odaberite „Send to“ i „Desktop (Create Shortcut)“. Pazite da ukoliko nemate upaljen prikaz ekstenzija na operacijskom sustavu, odaberete onaj XTag kod kojeg pod *Type* piše *Application* kao na slici.



IZVOZ OKVIRA IZ VIDEO

Kada uđete u aplikaciju odaberite meni *Video>Export frames from avi...* i otvoriti će vam se prozor kao na slici



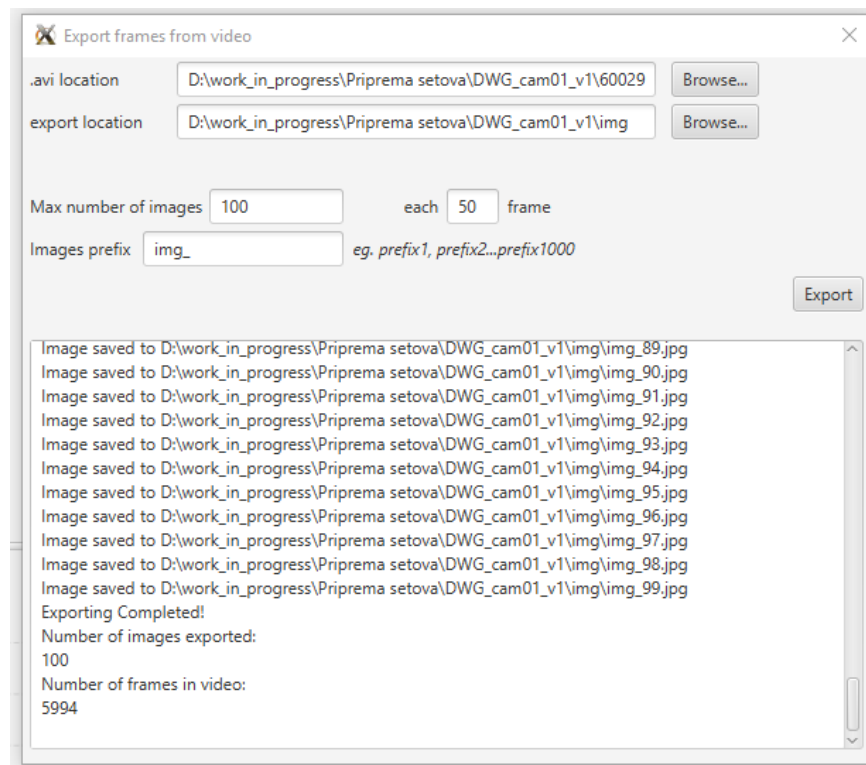
U polje *.avi location* unesite put do vašeg videa koji za sada mora biti u *avi* formatu, a u polje *export location* put do mape u koju želite da vam se izvezu okviri (slike) iz zadanog videa.

U polje *Max number of images* upišite maksimalni broj okvira koji želite izvest, ukoliko je broj okvira koje je moguće izvesti manji od tog broja, izvest će se taj manji broj, a u suprotnom broj koji ste zadali.

U polje *each frame* upišite svaki koji okvir iz videa želite izvest (preporuča se svaki 50ti zbog raznolikosti setova podataka).

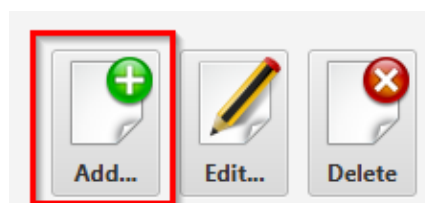
U polje *Images prefix* upišite kako želite da bude prefiks imena okvira odnosno slika. Npr. ukoliko upišete „img_“ onda će slike biti nazvane redom „img_0“, „img_1“ itd...

Ako ste sve dobro popunili i stisnuli *Export* krenuti će izvoz okvira iz zadanog videa i u prostoru ispod moći ćete pratiti koji okvir se trenutno obrađuje te ako je sve prošlo u redu dobiti poruku sličnu kao na slijedećoj slici:



DODAVANJE NOVOG SETA PODATAKA

Na početnom zaslonu aplikacije odaberite gumb *Add...* i otvoriti će vam se prozor kao na slici:



U polje *Name** upišite željeno ime seta podataka.

U polje *Camera Angle* ukoliko znate upišete kut pod kojim je kamera s kojim je video sniman.

U polje *Camera High* ukoliko znate upišete visinu na kojoj se kamera s kojom je sniman video nalazi.

Iz izbornika *Time of the day* odaberite doba dana kada je taj video sniman.

Iz izbornika *Weather Condition* odaberite kakvi su vremenski uvjeti u videu.

Iz izbornika *Image quality* odaberite kvalitetu slike na videu.

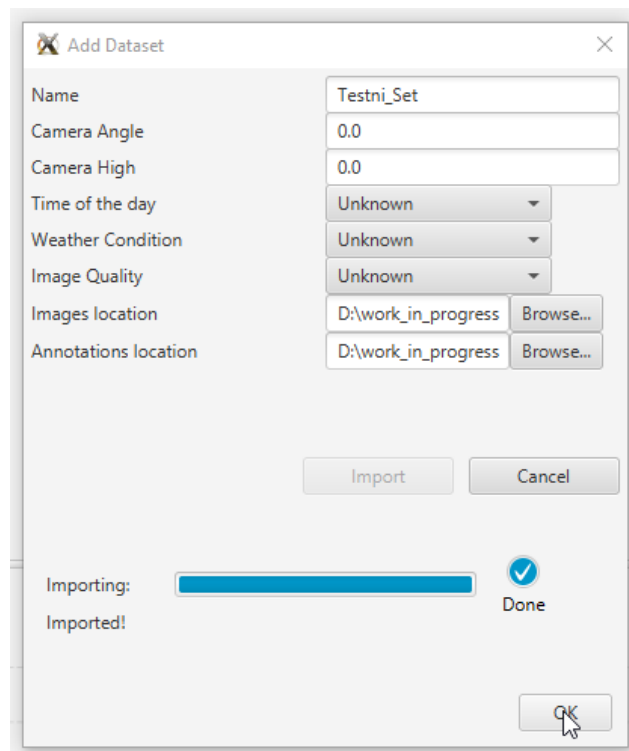
U polje *Images location** upišite put do mape gdje se nalaze slike odnosno okviri.

U polje *Annotations location** upišite put do mape gdje će aplikacija spremati anotacije koje ćete označavati odnosno ako već postoje anotacije. Neka to budu dva različita foldera.

**Stvari označene sa zvjezdicom su obavezne za popuniti*

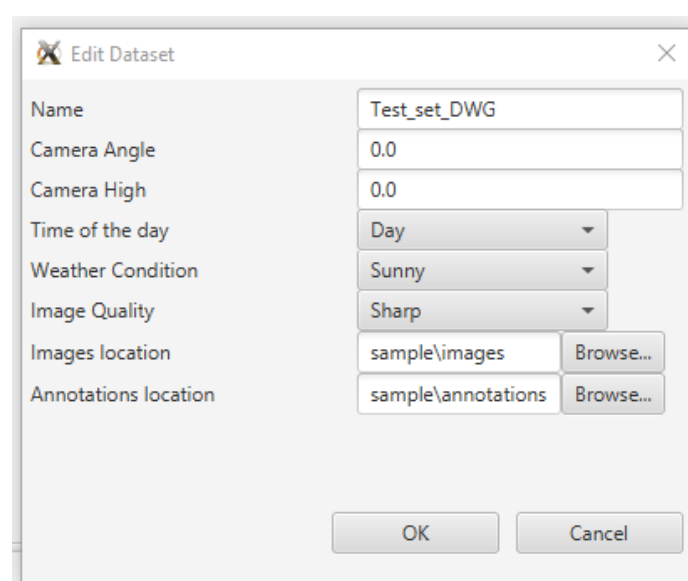
Kada ste sve popunili stisnite *Import* i uvoz novog seta će početi. Ovisno koliko slika ima i postoje li već anotacije uvoz može potrajati.

Na kraju odaberite *OK* kako bi završili dodavanje novog seta.



UREĐIVANJE SETA PODATAKA

Na početnom ekranu aplikacije odaberite gumb *Edit...* i otvoriti će vam se prozor kao na slici

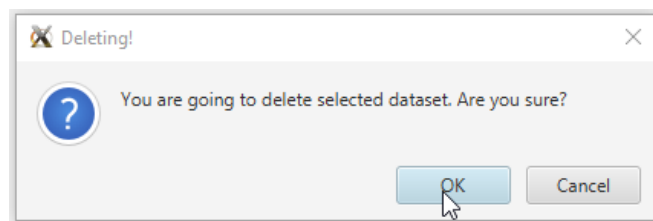
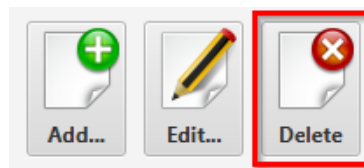


Nakon što ste završili s mijenjanjem podataka odaberite *OK*, odnosno *Cancel* ukoliko ste odustali od mijenjanja podataka.

BRISANJE SETA PODATAKA

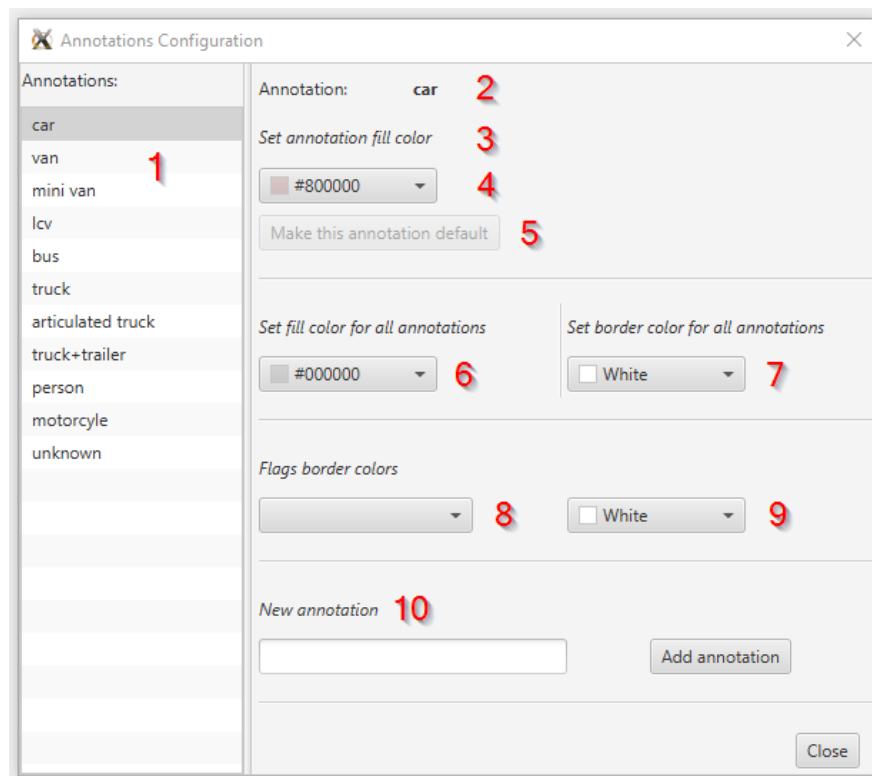
Na početnom ekranu aplikacije odaberite set koji želite izbrisati te potom *Delete*. Pojaviti će se prozor sa upozorenjem želite li zbilja trajno obrisati taj set podataka iz aplikacije.

Napomena: NE brišu se stvari podaci sa diska te kasnije možete ponovno dodati set u aplikaciju.



KONFIGURACIJA

Odaberite izbornik *Config>Annotations* i otvoriti će vam se prozor kao na slici:



1. Popis postojećih klasa anotacija u aplikaciji
2. Trenutno odabrana klasa
3. Odabir boje ispune pravokutnika za odabranu klasu
4. Odabir boje ispune pravokutnika za odabranu klasu
5. Gumb za postavljanje zadane klase
6. Postavljanje zadane boje ispune pravokutnika za sve klase
7. Postavljanje boje obruba pravokutnika za sve klase
8. Biranje zastavice za postavljanje boje obruba pravokutnika
9. Odabir boje obruba pravokutnika za odabranu zastavicu
10. Dodavanje nove klase anotacije

DODAVANJE NOVE KLASA ANOTACIJA

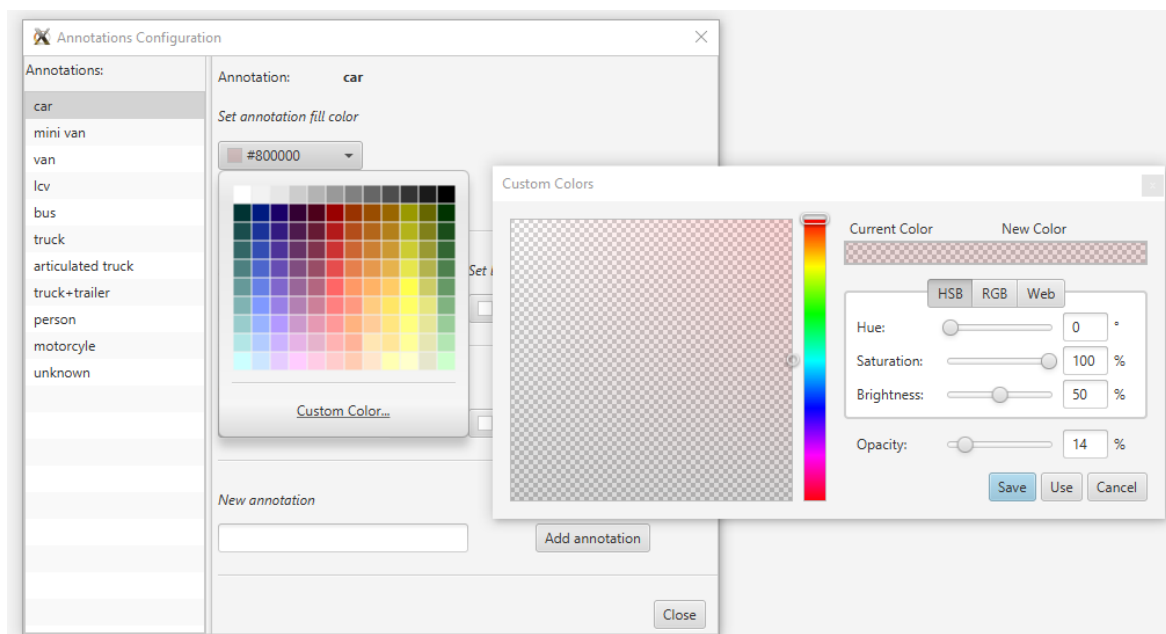
Pod dijelom 10. na slici u za to predviđeni prostor upišite ime nove klase te odaberite *Add annotation*. Nakon toga klasa se pojavljuje na kraju liste pod dijelom 1.

PROMJENA REDOSLIJEDA KLASA ANOTACIJA

Ukoliko korisnik želi promijeniti redoslijed klasa koje vidi, može to učiniti na način da odabere klasu koju želi i povuče je na mjestu gdje želi da bude (*Drag and drop*). Nakon toga i kada će anotirati pri otvaranju prozora za anotiranje imati će taj redoslijed.

MIJENJANJE BOJE ISPUNE PRAVOKUTNIKA ODREĐENE KLASA ANOTACIJE

Dijelovi 2., 3. i 4. odnose se na ovo poglavlje. Kada ste odabrali klasu iz liste kliknite na 3. i otvara vam se standardni prozor za izbor boja kao na slici:



Ovdje možete promijeniti boju, smanjiti ili povećati prozirnost (*opacity*) po želji. Nakon odabira *Save* sve promjene biti će automatski vidljive na slikama.

MIJENJANJE BOJE ISPUNE PRAVOKUTNIKA ZA SVE KLASSE

U dijelu 6. izaberite boju kao i u poglavlju MIJENJANJE BOJE ISPUNE PRAVOKUTNIKA ODREĐENE KLASSE ANOTACIJA.

MIJENJANJE BOJE OBRUBA PRAVOKUTNIKA ZA SVE KLASSE

U dijelu 7. izaberite boju kao i u poglavlju MIJENJANJE BOJE ISPUNE PRAVOKUTNIKA ODREĐENE KLASSE ANOTACIJA.

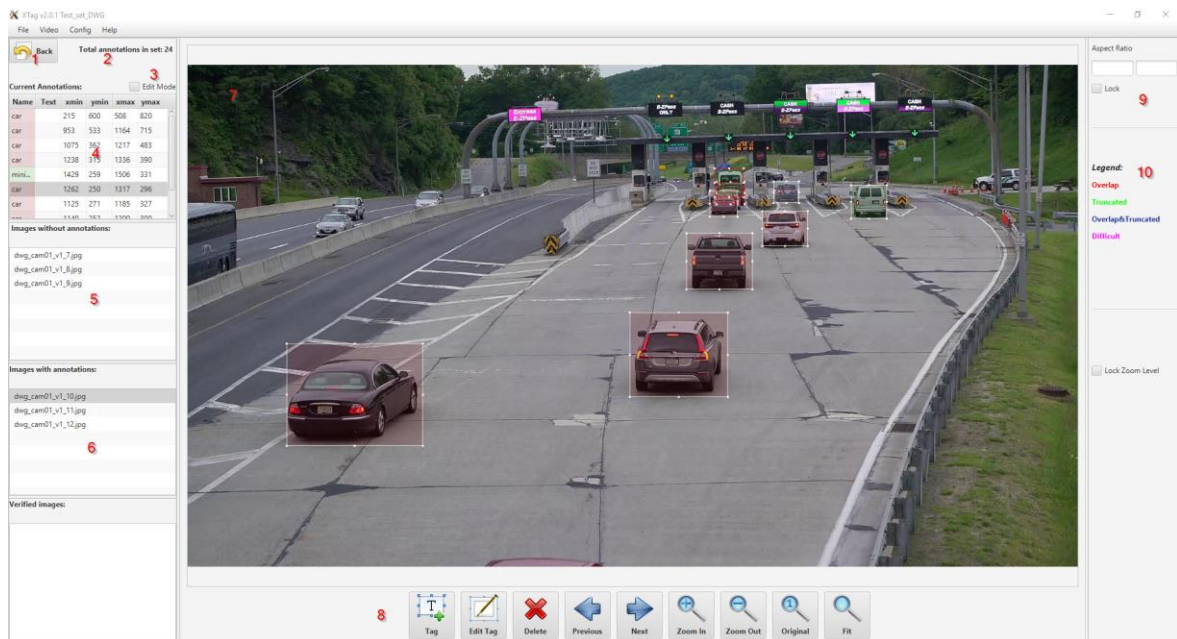
MIJENJANJE BOJE OBRUBA PRAVOKUTNIKA ZA ZASTAVICE

Dijelovi 8. i 9. odnose se na ovo poglavlje. U dijelu 8. odaberite koju zastavicu želite urediti te potom u dijelu 9. izaberite boju kao i u poglavlju MIJENJANJE BOJE ISPUNE PRAVOKUTNIKA ODREĐENE KLASSE ANOTACIJA.

POSTAVLJANJE ZADANE KLASSE ANOTACIJA

Odaberite klasu koju želite da bude zadana (*default*) i kliknite na gumb pod brojem 5. *Make this annotation default*. Zadana klasa se automatski pridjeljuje pri anotiranju odnosno nije ju potrebno eksplicitno označiti u listi prilikom anotiranja.

ANOTIRANJE – OZNAČAVANJE OBJEKATA NA SLICI

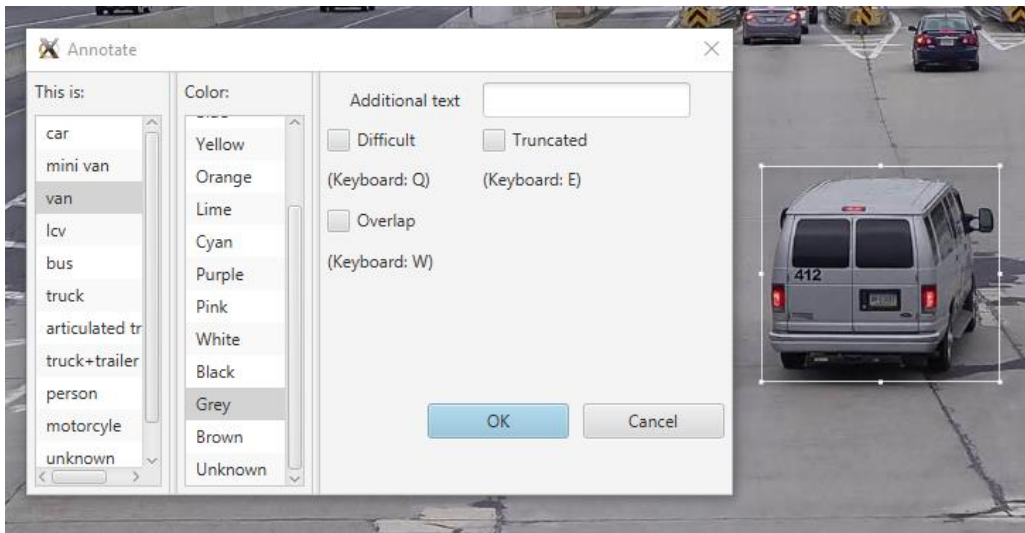


Nakon što odaberete željeni set podataka s početnog ekrana aplikacije otvara vam se ekran kao na slici iznad.

1. Gumb za vraćanje na prethodni ekran (početni)
2. Trenutačni ukupni broj anotacija u setu
3. Checkbox za *Edit mode*
4. Tablica s anotacijama na trenutnoj slici
5. Popis slika bez anotacija
6. Popis slika s anotacijama
7. Prikaz trenutno odabrane slike
8. Gumbi za anotiranje, kretanje i zumiranje
9. Dio za zaključavanje omjera slika
10. Legenda s bojama obruba zastavica

DODAVANJE NOVE ANOTACIJE

U dijelu 8. odabrite gumb *Tag* i možete početi anotirati, nakon toga mišem povučete po slici gdje želite označiti objekt i otvara vam se prozor kao na slici:



U prvoj listi izaberete klasu označenog objekta, a u drugoj listi boju objekta.

Zastavice:

Difficult – ukoliko je objekt teško prepoznatljiv na slici

Overlap – ukoliko je objekt zaklonjen bilo kojim drugim objektom na slici

Truncated – ukoliko objekt nije cijelu u kadru odnosno slici

Kada ste gotovi odaberete *OK* i vaša anotacija je spremljena i prikazana je u odgovarajućim bojama na slici.

PREČACI:

- Prvih 9 klasa – tipke redom 1,2,3,4,5,6,7,8,9
- Prvih 9 boja – SHIFT + 1,2,3,4,5,6,7,8,9
- Difficult – Q
- Overlap – W
- Truncated – E
- Enter odnosno završetak anotiranja – R

UREĐIVANJE ANOTACIJE

Svaku anotaciju možete micati po želji po slici radi bolje preciznosti. Također svakoj anotaciji možete mijenjati veličinu pomoću 8 malih kvadratića na rubovima.

Ukoliko želite urediti podatke o anotaciji, kliknite na nju te odaberite gumb *Edit* iz dijela 8. ili dva puta kliknite na željenu anotaciju te vam se otvara ponovno prozor kao i kod

dodavanja nove anotacije s razlikom da su vam označene informacije koje trenutno anotacija sadrži

PREČAC:

- Edit – dvostruki klik miša na anotaciju

BRISANJE ANOTACIJE

Odaberite anotaciju koju želite izbrisati te kliknite na gumb *Delete* iz dijela 8. ili na tipkovnici pritisnite tipku delete.

EDIT MODE

Imate mogućnost naknadnog brzo uređivanja anotacija. Stavite kvačicu u dijelu 3. te kako odabirete u dijelu 4. anotacije iz tablice (možete se kretati strelicama po tablici) tako vam se za svaku otvara prozor za uređivanje te je jedino taj objekt vidljiv na slici, a ostali su trenutno sakriveni. Ako želite prikazati ponovno sve objekte desnim klikom miša stisnite na samu sliku.

KRETANJE PO SLIKAMA

Za kretanje po slikama koristite gumb *Previous* i *Next* iz dijela 8. ili prečace na tipkovnici A i D.

PREČACI:

- Prethodna slika – A
- Slijedeća slika – D

Kada označite prvu anotaciju na slici koja do sad nije imala anotacija ona se automatski seli u listu slika s anotacijama (dio 6.).

Ako se trenutno nalazite na zadnjoj slici u listi slika s anotacijama (dio 6.) i pritisnete slijedeća slika, prebaciti će vas na prvu sliku u listi slika bez anotacija (dio 5.).

Ako se trenutno nalazite na prvoj slici u listi slika bez anotacija (dio 5.) i pritisnete prethodna slika, prebaciti će vas na zadnju sliku u listi slika s anotacijama (dio 6.).

PRIBLIŽAVNJE / UDALJAVANJE SLIKE

U dijelu 8. koristite gumb *Zoom in*, *Zoom out*, *Original* i *Fit*. Približiti i udaljiti možete se i pomoću CTRL + scroll srednjeg gumba miša. Na približenoj slici možete se kretati pritiskom na srednji gumb miša i micanjem.