

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5863

**Praćenje objekata u  
videozapisima**

Luka Mijić

Zagreb, lipanj 2018.

**SVEUČILIŠTE U ZAGREBU**  
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**  
**ODBOR ZA ZAVRŠNI RAD MODULA**

Zagreb, 20. ožujka 2018.

## **ZAVRŠNI ZADATAK br. 5863**

Pristupnik: **Luka Mijić (0036486718)**  
Studij: Računarstvo  
Modul: Računarska znanost

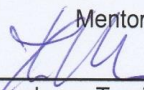
**Zadatak: Praćenje objekata u videozapisima**

Opis zadatka:

Praćenje objekata u videozapisima jedna je od važnih primjena računalnog vida. U okviru završnog rada potrebno je proučiti postupke praćenja objekata opisane u literaturi. Programski ostvariti sustav temeljen na prikladnim metodama praćenja koji će omogućiti prikaz originalnog videozapisa, odabir objekta za praćenje, te iscrtavati pozicije i trajektorije praćenih objekata u daljnjim okvirima videozapisa. Pripremiti bazu video zapisa za testiranje sustava, analizirati ponašanje implementiranog sustava te prikazati i ocijeniti ostvarene rezultate. Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne sekvence i rezultate, uz potrebna objašnjenja i dokumentaciju te navesti korištenu literaturu.

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


Mentor:

  
\_\_\_\_\_  
Doc. dr. sc. Tomislav Hrkać

Djelovođa:

  
\_\_\_\_\_  
Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za  
završni rad modula:

  
\_\_\_\_\_  
Prof. dr. sc. Siniša Srblić



# SADRŽAJ

1. Uvod.....	1
2. Algoritam Fragtrack.....	4
2.1. Osnovni Pojmovi.....	4
2.1.1. OpenCV.....	4
2.1.2. BGR prostor boja.....	4
2.1.3. HSV prostor boja.....	5
2.1.4. Histogram.....	7
2.1.5. Integralna slika.....	8
2.2. Fragtrack – implementacija.....	11
3. Rezultati.....	15
3.1. Test 1.....	15
3.2. Test 2.....	17
3.3. Test 3.....	19
3.4. Test 4.....	20
4. Moguće nadogradnje sustava.....	23
4.1. Višedretvenost.....	23
4.2. Skaliranje okvira.....	23
5. Zaključak.....	24
Literatura.....	25
Sažetak.....	26

# 1. Uvod

Računalni vid je znanstvena disciplina koja se bavi teorijom sustava koji pokušavaju saznati informacije iz slikovnih podataka. Izvori slikovnih podataka mogu biti razni, a neki od njih su videozapis, medicinski skener, zaštitna kamera u trgovačkom centru itd. Razvoj ove grane je počeo u kasnim 1960-tima na sveučilištima koja su bili pioniri u području umjetne inteligencije. Prvotna zamisao je bila oponašati ljudski vizualni sustav. Tako je profesor MIT-a Marvin Minsky u 1966. godini dao zadatak studentu da preko ljeta spoji kameru na računalo i da računalo opiše što vidi. Ispostavilo se da je to ipak prekompleksan zadatak za ljetni projekt jednog studenta, jer je to problem koji ni pedeset godina kasnije nije riješen. Istraživanja u 1970-tima postavila su rane temelje za mnoge algoritme u računalnom vidu koji se danas koriste, primjerice temelji algoritma za detekciju rubova. U sljedećim desetljećima dogodio se porast korištenja strogih matematičkih analiza za rješavanje problema računalnog vida. Tako je u drugom dijelu 1990-tih godina korištenjem statističkih tehnika za rješavanja problema pronalaska lica u slikama nastao takozvani Eigenface. U novije vrijeme pojavio se rast rješavanja problema računalnog vida pomoću strojnog učenja. Glavni razlog toga su dvije stvari. Prva je puno moćnije i jeftinije sklopovlje, a drugi je što su podaci korišteni za učenje dostupniji i više ih je nego ikada prije. Također sklopovlje je postalo puno pogodnije za duboko učenje.

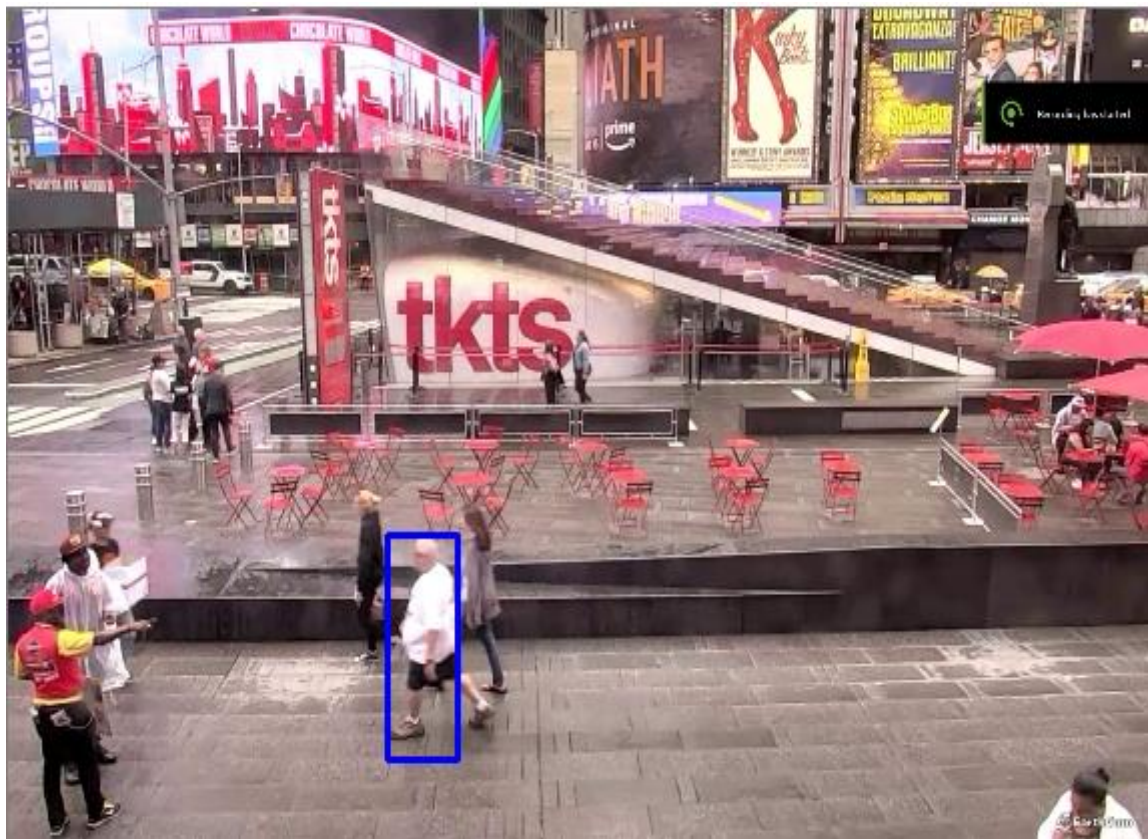


Slika 1. Primjer kvalitete klasifikacije koja ipak sadrži neke pogreške [1]



Učenje se na GPU-u može odvijati puno brže nego prije. Upravo skupljanje tih podataka je razlog zašto kompanije poput Google-a i Facebook-a nude besplatan skladišni prostor za korisničke slike i videozapise. Primjer ovoga je Google AI koji je za učenje pronalaska mačjih lica u videozapisima koristio 16 000 računala i 10 milijuna Youtube videozapisa. Računalni vid sadrži dosta poddisciplina kao što je praćenje objekata, detekcija objekata, restauracija slike, predviđanje kretnji itd.

Ljudi često miješaju područje praćenja i detekcije objekata. Detekcija traži definirani objekt na slici, dok praćenje prati već označeni objekt iz slike u sliku u videozapisu. Često te dvije grane rade zajedno, primjerice sustav za detekciju objekta traži objekt na slici, a onda sustav za praćenje prati njegovu lokaciju iz slike u sliku.



Slika 2. Sustav za praćenje objekata

Tema ovog rada je praćenje objekata iz slike u sliku. Ovo područje je veoma kompleksno te dobar algoritam mora bit sposoban savladati mnogo prepreka, kao što su objekti koji se kreću prebrzo u odnosu na broj slika snimljenih po sekundi, objekti koji mijenjaju oblik iz slike u sliku. Ne postoji definitivni algoritam za praćenje objekata koji će

se koristiti u svakoj situaciji. Svaki algoritam ima svoje prednosti i mane. Ako korisniku treba sustav koji može pratiti objekte u stvarnom vremenu, treba biti spreman na gubitak određene preciznosti, ako je preciznost najpotrebnija, korisnik treba biti spreman na to da algoritam neće biti u stanju raditi obradu u stvarnom vremenu. Ovaj rad će se fokusirati na algoritam Fragtrack koji je baziran na fragmentiranju odabranog predloška i uspoređivanju histograma tih fragmenata.

## 2. Algoritam Fragtrack

### 2.1. Osnovni pojmovi

Prije objašnjenja same implementacije algoritma, prvo treba objasniti neke osnovne pojmove bez kojih bi shvaćanje ovog algoritma bilo otežano.

#### 2.1.1. OpenCV

OpenCV (Open Source Computer Vision) je biblioteka koja nudi razne funkcionalnosti usko povezane s računalnim vidom. Originalno ju je razvio Intel. Biblioteka je pisana u programskom jeziku C++, a postoje i povezivanja biblioteke sa programskim jezikom Java, Python itd. Biblioteka je pod BSD licencom što znači da je besplatna za akademske i komercijalne potrebe. Biblioteka sadrži preko 2500 optimiziranih algoritama, među kojima su i algoritmi za praćenje. Implementacija rada ne sadrži gotove algoritme za praćenje objekata, ali koristi OpenCV za određene proračune.

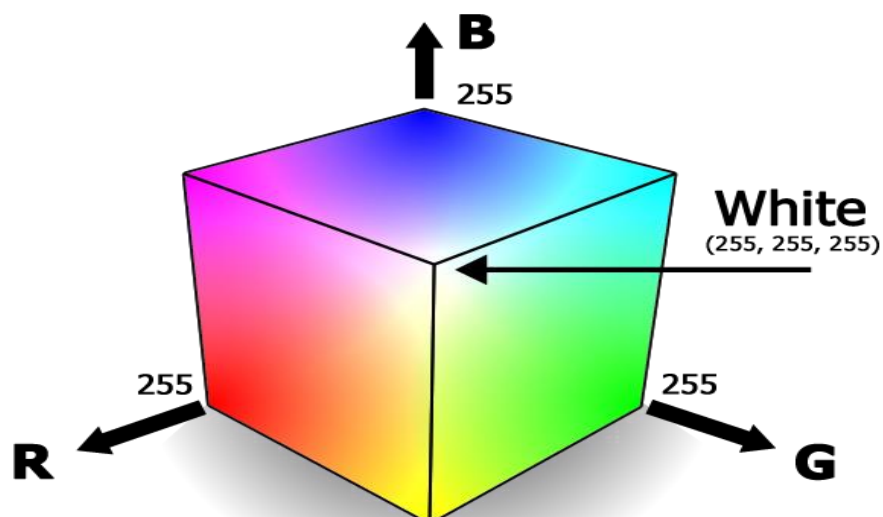
#### 2.1.2. BGR prostor boja

BGR je prostor boja koji za opisivanje boje koristi tri komponente, a to su: plava, zelena i crvena. Iako je puno popularniji i rasprostranjeniji RGB prostor boja u OpenCV-u je BGR zadan prostor boja. Razlog tome je što je BGR u prošlosti bio popularan format kod proizvođača kamera. Svaka od tri komponenti boje može poprimiti vrijednosti iz skupa  $[0, 255]$  ili iz skupa  $[0.0, 1.0]$ , ovisno o implementaciji.



Slika 3.  $[177, 10, 41]$  boja u BGR prostoru





Slika 4. Prikaz BGR prostora boja [8]

### 2.1.3. HSV prostor boja

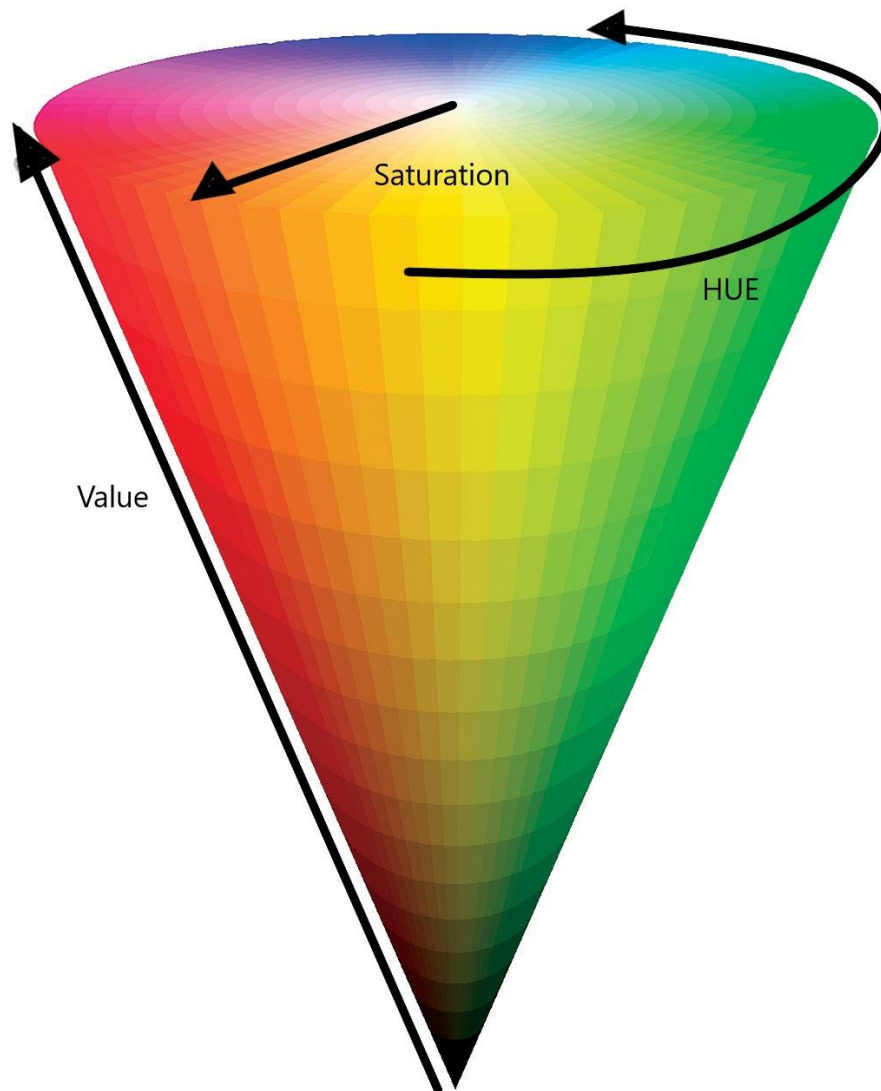
Za razliku od RGB i BGR prostora koji boju definiraju pomoću odnosa između primarnih boja HSV je puno sličniji ljudskom percipiranju boje. Također se sastoji od tri komponente, a to su:

1. Nijansa (eng. Hue)
  - Ova komponenta zapravo definira boju
  - Zamišljen je kao kružnica, a svaki kut predstavlja jednu nijansu
  - Poprima vrijednosti u skupu  $[0, 360]$
  - OpenCV implementacija za nijansu koristi brojeve iz skupa  $[0, 180]$ , razlog tome pohranjivanje informacije o nijansi u jednom bajtu

Tablica 1. Veza nijanse i kuta u OpenCV-u

Nijansa	Kut
Crvena	0-30
Žuta	30-60
Zelena	60-90
Cijan	90-120
Plava	120-150
Magenta	150-180

2. Zasićenost (eng. Saturation)
  - Količina sive boje
  - Uobičajno poprima vrijednosti u skupu  $[0.0, 1.0]$ , ali OpenCV koristi vrijednosti u skupu  $[0, 255]$
  - Efekt izbjeljivanja se može postići korištenjem vrijednosti koja je bliska 0
3. Vrijednost (eng. Value)
  - Ponekada se koristi izraz svjetlina (eng. Brightness) te se umjesto HSV piše HSB
  - Uobičajno poprima vrijednosti u skupu  $[0.0, 1.0]$ , ali OpenCV koristi vrijednosti u skupu  $[0, 255]$
  - Vrijednost 0 predstavlja crnu boju, a vrijednost 255 otkriva najviše boje



Slika 5. Prikaz HSV prostora boja.

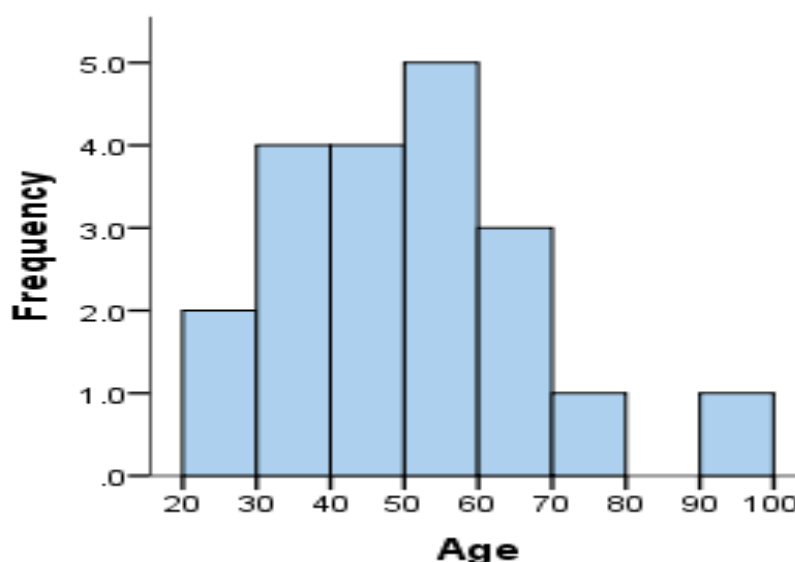
## 2.1.4. Histogram

Histogram je funkcija  $m_i$  koja broji zapažanja koja se mogu klasificirati u međusobno razdvojene kategorije. Ako je  $n$  ukupan broj zapažanja, a  $k$  ukupan broj kategorija, onda vrijedi:  $n = \sum_{i=1}^k m_i$  (2.1).

Postoji nekoliko načina prikazivanja histograma, a to su prikaz grafom, tablicom, matricom itd.

36	25	28	46	55	68	72	55	36	38
67	45	22	48	91	46	52	61	58	55

Slika 6. Primjer podataka koji se mogu pokazati histogramom

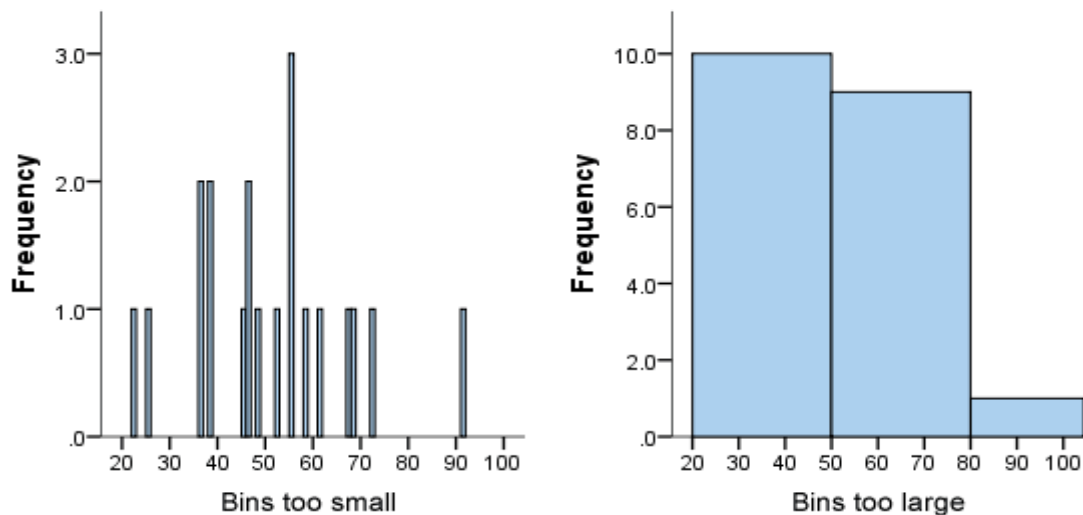


Slika 7. Prikaz grafom podataka iz slike broj 6 [6]

Pri stvaranju histograma jako je bitno odrediti raspon i broj kategorija. Ne postoji najbolja metoda za izračun broja kategorija. Primjerice u ovom radu će se histogram računati nad komponentom nijanse u HSV prostoru boja koja poprima vrijednosti 0-180, tako da je izabrano 18 kategorija. Postoje neke formalnije metode kao što su :

1. Korijen
  - $k = \sqrt[3]{n}$  (2.2)
2. Sturgesova metoda
  - $k = \lceil \log_2 n \rceil + 1$  (2.3)
  - Izvedena je iz binomne razdiobe i implicira približno normalnu razdiobu zapažanja
3. Riceovo pravilo
  - $k = \left\lceil 2 * n^{\frac{1}{3}} \right\rceil$  (2.4)

Lošim odabirom broja kategorija može se izgubiti informacija o trendovima.



Slika 8. Izgled histograma sa slike broj 7 s prevelikim i premalim brojem kategorija [6]

Normalizirani histogram je histogram kojemu su vrijednosti svih kategorija podijeljene sa  $n$ , tj. vrijednost svake kategorije je u rasponu  $[0, 1]$ , a vrijedi  $\sum_{i=1}^k m_i = 1$  (2.5). Jedan od razloga zbog kojega se koristi normaliziran histogram je usporedba histograma koji imaju različit broj ukupnih zapažanja.

## 2.1.5. Integralna slika

To je struktura podataka i algoritam za brzo i efikasno računanje sume vrijednosti pravokutnika koji je dio neke rešetke. Vrijedost na lokaciji  $(x, y)$  je zbroj svih vrijedosti lijevo i gore od trenutne lokacije. Postoje dva načina zbrajanja:

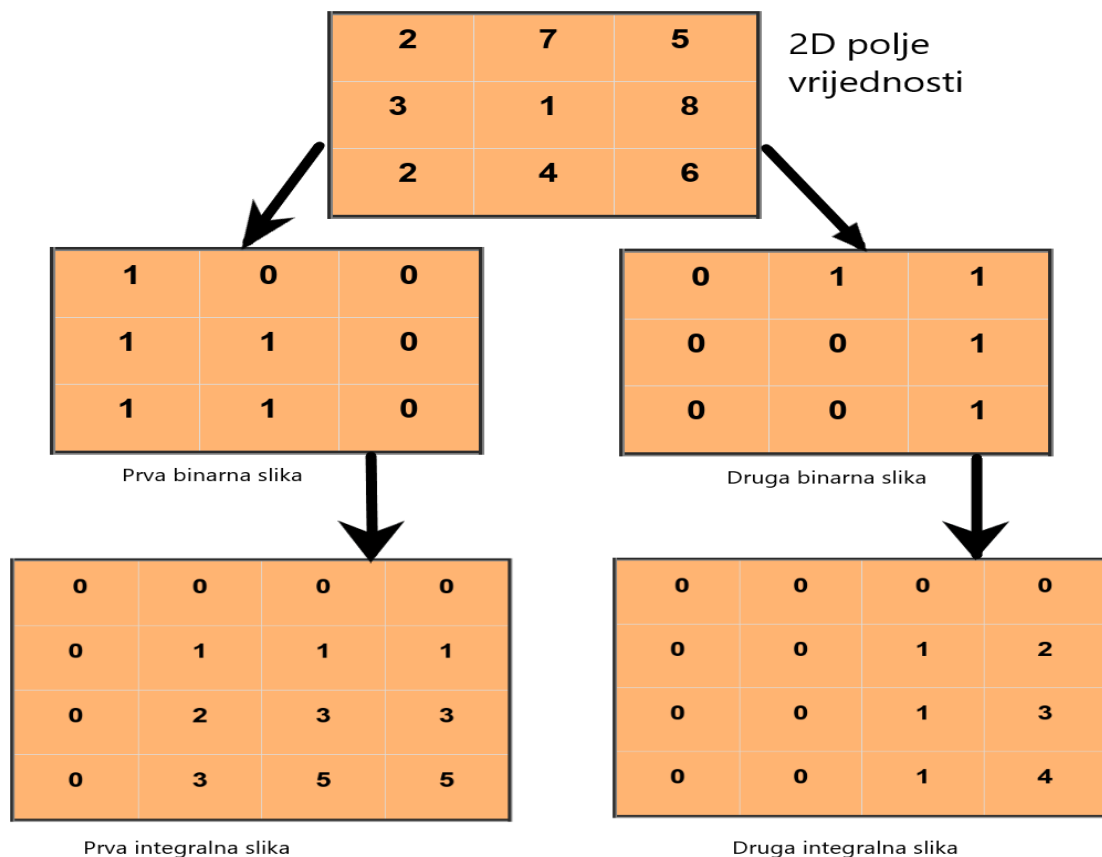
1. Uključujući
  - U zbrajanje je uključena vrijednost na  $(x, y)$  i vrijednosti na  $x$  stupcu i  $y$  retku
  - $I(x, y) = \sum_{x' \leq x} \sum_{y' \leq y} i(x', y')$  (2.6)
2. Isključujući
  - U zbrajanje se ne uključuje vrijednost na  $(x, y)$  niti vrijednosti koje dijele ili stupac ili redak s lokacijom  $(x, y)$
  - $I(x, y) = \sum_{x' < x} \sum_{y' < y} i(x', y')$  (2.7)
  - Konačni rezultat sadrži jedan redak i jedan stupac više

OpenCV sadrži implementaciju isključujućeg načina pa će fokus biti na toj verziji. Algoritam integriranja slike je koristan za optimizaciju. Primjerice ako trebamo izračunati histograme puno različitih dijelova iste slike. Integralnu sliku je dovoljno jednom izračunati, a do histograma bilo kojeg dijela slike se dolazi jednostavnim zbrajanjem i oduzimanjem nekoliko brojeva.

Algoritam računanja integralne slike za jednu komponentu iz prostora boja ( $k$  – broj kategorija;  $w$  – širina, tj. broj stupaca;  $h$  – visina, tj. broj redaka;  $n$  – broj zapažanja, tj.  $n = w * h$ ):

1. Priprema podataka
2. Stvaranje  $k$  binarnih slika (vrijednosti piksela mogu biti samo 0 ili 1) sa  $w * h$  piksela, inicijalno su vrijednosti svih piksela 0
3. Za svaki piksel na lokaciji  $(x, y)$  gdje je  $x$  u skupu  $[0, w - 1]$ , a  $y$  u skupu  $[0, h - 1]$  odrediti pripadnu kategoriju i pripadnoj binarnoj slici na lokaciji  $(x, y)$  postavi 1
4. Za svaku od  $k$  binarnih slika treba stvoriti jedno dvodimenzijско polje dimenzija  $(w + 1) * (h + 1)$ , te izračunati vrijednost svakog piksela koristeći formulu :
  - (2.7) ili
  - $I(x, y) = i(x - 1, y - 1) + I(x - 1, y) + I(x, y - 1) - I(x - 1, y - 1)$  (2.8)

Primjer rada algoritma s dvije kategorije, prva kategorija prima brojeve u skupu  $[1, 5]$ , a druga u skupu  $[6, 10]$  :



Slika 9. Postupak stvaranja integralne slike

Broj zapažanja koji pripada određenoj kategoriji može se saznati integralnom slikom koristeći vrijednost na lokaciji  $(w, h)$ . Primjerice na slici broj 9 se na  $(w, h)$  lokaciji u prvoj integralnoj slici nalazi broj 5, a u drugoj broj 4. Iz tih informacija se može saznati histogram cijelog dvodimenzionalnog polja. Tako je histogram ovog polja  $[5, 4]$ . Isto tako se može izračunati histogram za bilo koje dvodimenzionalno polje koje je podskup ovog polja. Za podpolje koje sadrži  $x$  elemente od  $x_1$  do  $x_2$  i  $y$  elemente od  $y_1$  do  $y_2$  vrijednost svake kategorije se računa formulom:

$$Count(k) = I_k(x_2 + 1, y_2 + 1) + I_k(x_1, y_1) - I_k(x_1, y_2 + 1) - I_k(x_2 + 1, y_1) \quad (2.9)$$

## 2.2. Fragtrack – implementacija

Fragtrack je algoritam za praćenje objekata u videozapisima koji se bazira na fragmentima, tj. odabrani predložak koji se prati se dijeli u fragmente i uspoređivanje se izvodi između histograma tih fragmenata. Broj fragmenata se zadaje proizvoljno, tj. broj fragmenata nije određen modelom objekta koji se prati. Zbog toga je algoritam otporan na djelomičnu prekrivenost.

Parametri algoritma:

- $n$  – broj horizontalnih fragmenata,  $m$  – broj vertikalnih fragmenata,
- $n * m$  – ukupan broj fragmenata
- $Q$  – faktor koji prima vrijednosti iz skupa  $[0, 1]$ , ako smatramo da će prekrivanje ostaviti uvijek barem 25% objekta vidljivo, vrijednost postavljamo na 0.25
- $w$  – širina predloška koji se prati
- $h$  – visina predloška koji se prati
- $c_x$  i  $c_y$  – centar oko kojega se stvara prostor pretraživanja
- $r$  – pola duljine kvadrata koji se formira oko centra predloška, definira prostor pretraživanja

Implementacijska logika:

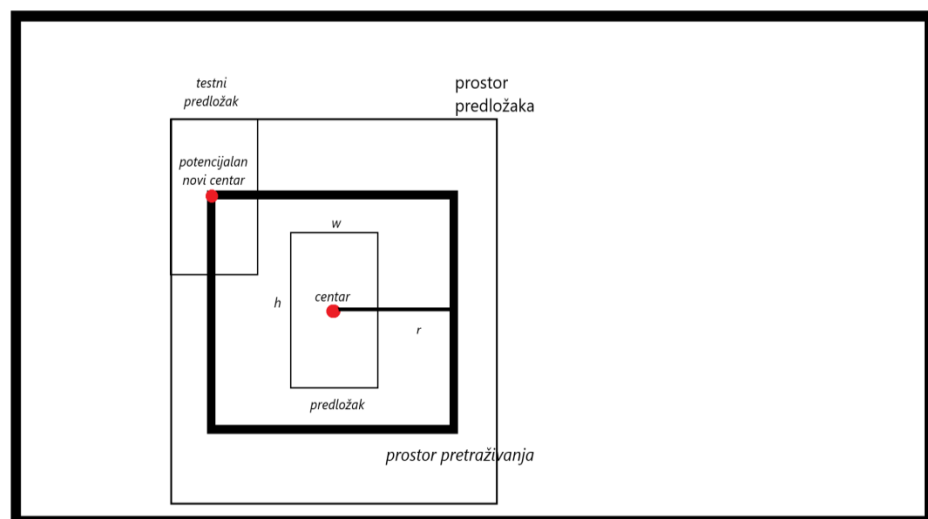
1. Odaberi videozapis
2. Otvori prvu sliku u videozapisu i ponudi korisniku odabir predloška koji će se pratiti



Slika 10. Biranje predloška



3. Pretvori predložak iz BGR prostora boja u HSV prostor boja
4. Izvuci komponentu nijanse (eng. Hue) i zanemari ostale dvije komponente iz predloška
5. Izračunaj integralnu sliku predloška
6. Podijeli predložak na  $n * m$  fragmenata i izračunaj normalizirani histogram za svaki fragment pojedinačno
7. Za svaku sliku u videozapisu napravi sljedeće:
  - a. Prostor koji se pretražuje počinje na pikselu  $(c_x - r, c_y - r)$ , a širina i visina prostora je  $2r$
  - b. Prostor pretraživanja nije sam po sebi dovoljan, već je potreban i prostor predložaka, tj. svi potencijalni centri su u prostoru pretraživanja, ali testni predlošci koji se stvaraju oko centra su u prostoru predložaka. Prostor predložaka počinje na lokaciji  $(c_x - r - \frac{w}{2}, c_y - r - \frac{h}{2})$ , a širina mu je  $w + 2r$ , a visina mu je  $h + 2r$ . Ako se bilo koja od lokacija u prostoru predložaka nalazi izvan originalne slike, tom pikselu se pridružuje vrijednost 0. Nad prostorom predložaka stvori integralnu sliku. Stvaranje integralne slike je skup postupak nad velikim slikama, pa se provodi samo na jednom prostoru predložaka, a ne na cijeloj slici.



Slika 11. Vizualizacija problema

- c. Za svaki  $(x, y)$  piksel koji se nalazi u prostoru pretraživanja napravi sljedeće:
  - i. Oko piksela stvori testni predložak, podijeli ga na  $n * m$  fragmenata i za svaki od fragmenata izračunaj normalizirani histogram
  - ii. Korištenjem neke proizvoljne mjere za udaljenost dva histograma usporedi fragmente testnog i originalnog predloška koji su na istim lokacijama, te ih pohrani u listu. Ima nekoliko načina za dobivanje mjere udaljenosti dva histograma. U ovoj implementaciji se koristi “algoritam preslagivanja zemlje” (eng. Earth’s Mover Algorithm).
  - iii. Sortiraj listu udaljenosti fragmenata uzlaznim redom
  - iv. Mjera  $d'$  koja predstavlja udaljenost dva predloška je udaljenost u listi koja se nalazi na  $[Q * n * m] - tom$  indeksu.
  - v. Ako je  $d'$  trenutno najmanja zabilježena udaljenost između predložaka pohrani je u  $d_{min}$  i spremi trenutnu  $(x, y)$  lokaciju kao potencijalni novi centar
- d. Postavi nove vrijednosti centra na lokaciju povezanu s  $d_{min}$
- e. Nacrtaj predložak povezan s  $d_{min}$  na trenutnoj slici

Ovaj algoritam je izveden koristeći nekoliko implementiranih Python modula i već postojećih biblioteka, a to su:

1. `integral_images.py`
  - ovaj modul sadrži neke korisne metode kao što su metoda za kreiranje integralne slike iz HSV slike, metode za računanje histograma koja prima područje interesa integralne slike i integralnu sliku
2. `patches.py`
  - ovaj modul u sebi sadrži jednu klasu, klasa se zove `Patches`
  - `Patches` je klasa koja modelira predložak, te u sebi čuva histograme svakog fragmenta, uz to sadrži metodu koja računa udaljenost između dva predloška
3. `frag_tracker.py`
  - ovaj modul također sadrži jednu klasu u sebi, a ta klasa se zove `FragTracker`
  - pri stvaranju primjerka klase `FragTracker` označava se predložak koji se

prati te se on fragmentira i pohrani

- klasa FragTracker sadrži metodu koja pokreće praćenje i zapravo glavnina ranije opisanog algoritma se odvija u toj metodi
- ostale metode su pomoćne metode koje pomažu čitljivosti i smanjivanju redundancije koda

#### 4. numpy

- to je biblioteka koja se koristi za modeliranje višedimenzionalnih polja
- u ovom radu se koristi za spremanje piksela slike, te se po potrebi koristi za proširenje prostora za predloške vrijednostima nula

#### 5. OpenCV

- ranije opisana biblioteka
- koristi se učitavanje videozapisa, čitanja videozapisa sliku po sliku, prebacivanje slike iz jednog prostora boja u drugi i za brzo integriranje binarnih slika

#### 6. PyEMD

- Python omotač za Ofir Peleovu i Michael Wermanovu C++ implementaciju “algoritma preslagivanja zemlje”
- Koristi se za uspoređivanje dva histograma

### 3. Rezultati

Tema ove sekcije je analiza uspješnosti implementacije ranije navedenog algoritma, vrijeme potrebno za obradu svake slike itd. Uz ranije navedene parametre dodan je još jedan parametar, a to je korak  $s$ . Parametar  $s$  je uveden radi ubrzavanja algoritma. Parametar  $s$  zadržava veličinu prostora pretraživanja, ali smanjuje broj piksela koji se provjeravaju. Pretražuju se samo pikseli u  $i * s$  retku i  $j * s$  stupcu. Vrijedi da su vrijednosti  $i, j$  prirodni brojevi i nula i da je  $i * s < h, j * s < w$ .

Nekoliko parametara određuje vrijeme izvođenja. Vrijeme usporedbe dva histograma ovisi samo o broju fragmenata. Npr. za  $n, m = 3$ , vrijeme usporedbe iznosi 0.0015 sekundu, a za  $n, m = 10$  vrijeme usporedbe iznosi 0.01 sekundu. Ti brojevi sami po sebi nisu preveliki, ali ako je  $r = 20$ , prostor pretraživanja je  $40 * 40 = 1600$  piksela. To znači da za izračunavanje novog centra u prvom slučaju treba 2.4 sekunde, a u drugom 16 sekundi.

Veličina predloška i odabrani radijus utječu na vrijeme računanja integralne slike te njezino vrijeme izvođenja može biti dugo za objekte koji imaju veliku širinu i visinu ili kada je  $r$  velik broj.

#### 3.1. Test 1

Ovaj test se odvija na Times Squareu u New Yorku. Osoba koja se prati je dosta različita od pozadine. Osoba koja se prati većim dijelom videozapisa zadržava isti oblik tijela. Pred kraj videozapisa osoba postane djelomično prekrivena.

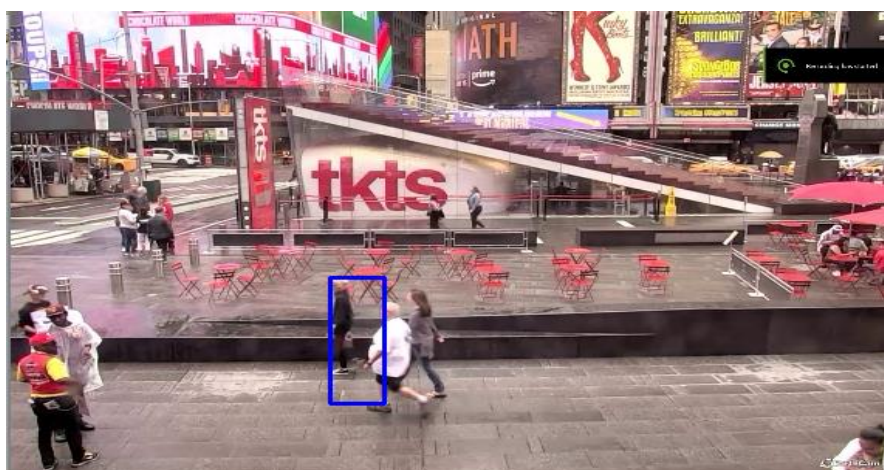
Tablica 2. Vrijednosti parametara prvog testa

Parametar	Vrijednost	Parametar	Vrijednost
$r$	20	$s$	3
$n$	10	$w$	35
$m$	10	$h$	97
$Q$	0.4		

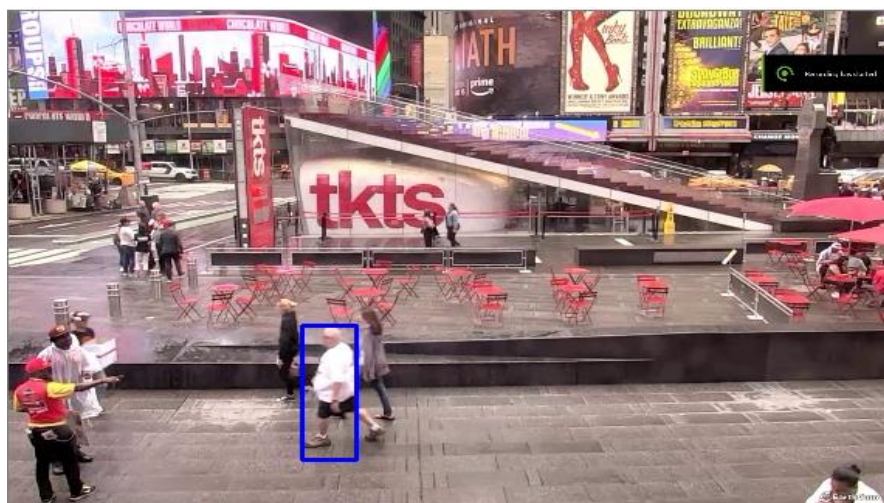


Slika 12. Početni odabir objekta koji se prati

Negdje na polovici videozapisa, okvir za praćenje napušta objekt. Ubrzo nakon toga okvir u prostoru predraživanja ponovo pronalazi objekt.



Slika 13. Rezultati na polovici videozapisa



Slika 14. Oporavak praćenja

Pred kraj videozapisa objekt koji se prati biva djelomično prekriven.



Slika 15. Djelomično prekrivanje objekta

Ubrzo nakon toga dolazi do potpunog prekrivanja i algoritam gubi objekt koji prati.

Tablica 3. Brzina Izvođenja

Proces	Vrijeme izvođenja
<b>Računanje integralne slike</b>	0.04s
<b>Usporedba 2 predloška</b>	0.01s
<b>Računanje novog centra</b>	2.2s

Zaključak testa:

- Zbog velikog broja fragmenata (100), računanje novog centra je bilo dosta sporo
- Iako je na polovici izvođenja okvir nije prekrivao objekt koji se prati, algoritam se uspio oporaviti, te je u konačnici rezultat bio zadovoljavajući
- Algoritam radi pri djelomičnom prekrivanju

## 3.2. Test 2

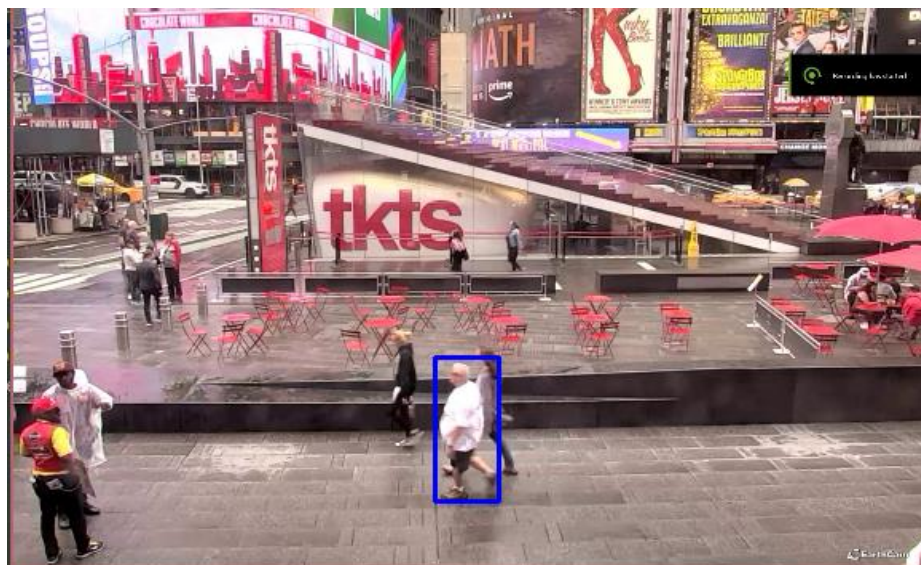
Drugi test će se odvijati na istom videozapisu kao i prvi test, samo sa drugačijim početnim parametrima.

Tablica 4. Vrijednosti parametara drugog testa

Parametar	Vrijednost	Parametar	Vrijednost
<b><i>r</i></b>	<b>20</b>	<b><i>s</i></b>	<b>3</b>
<b><i>n</i></b>	<b>4</b>	<b><i>w</i></b>	<b>35</b>
<b><i>m</i></b>	<b>4</b>	<b><i>h</i></b>	<b>97</b>
<b><i>Q</i></b>	<b>0.4</b>		

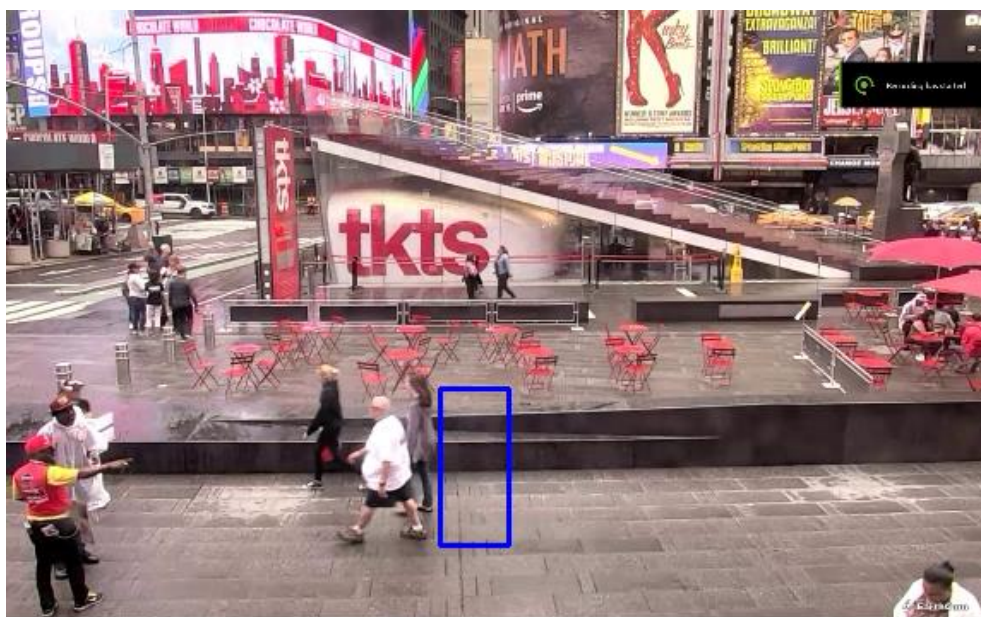


Glavna razlika u testu je bilo smanjenje broja fragmenata sa 100 na 16.



Slika 16. Objekt je unutar okvira

Ubrzo nakon slike broj 16 sustav gubi objekt te ga ne uspijeva ponovo pronaći u prostoru pretraživanja.



Slika 17. Okvir je izvan objekta koji se prati

Tablica 5. Brzina Izvođenja

Proces	Vrijeme izvođenja
Računanje integralne slike	0.04s
Usporedba 2 predložka	0.002s
Računanje novog centra	0.42s



Zaključak testa:

- Smanjenjem broja fragmenata izvođenje algoritma se znatno ubrzalo
- Iako se izvođenje ubrzalo učinkovitost algoritma je pala i algoritam nije zadovoljio kriterije

### 3.3. Test 3

Ovaj test se odvija na Times Squareu u New Yorku. Svrha testa je provjeriti ponašanje algoritama u situacijama kada se objekt stapa s okolinom.

Tablica 6. Vrijednosti parametara trećeg testa

Parametar	Vrijednost	Parametar	Vrijednost
$r$	20	$s$	3
$n$	10	$w$	17
$m$	10	$h$	49
$Q$	0.4		



Slika 18. Biranje objekta koji se prati

Algoritam relativno brzo gubi objekt koju prati, te se ne oporavlja.



Slika 17. Objekt koji se prati je izvan okvira

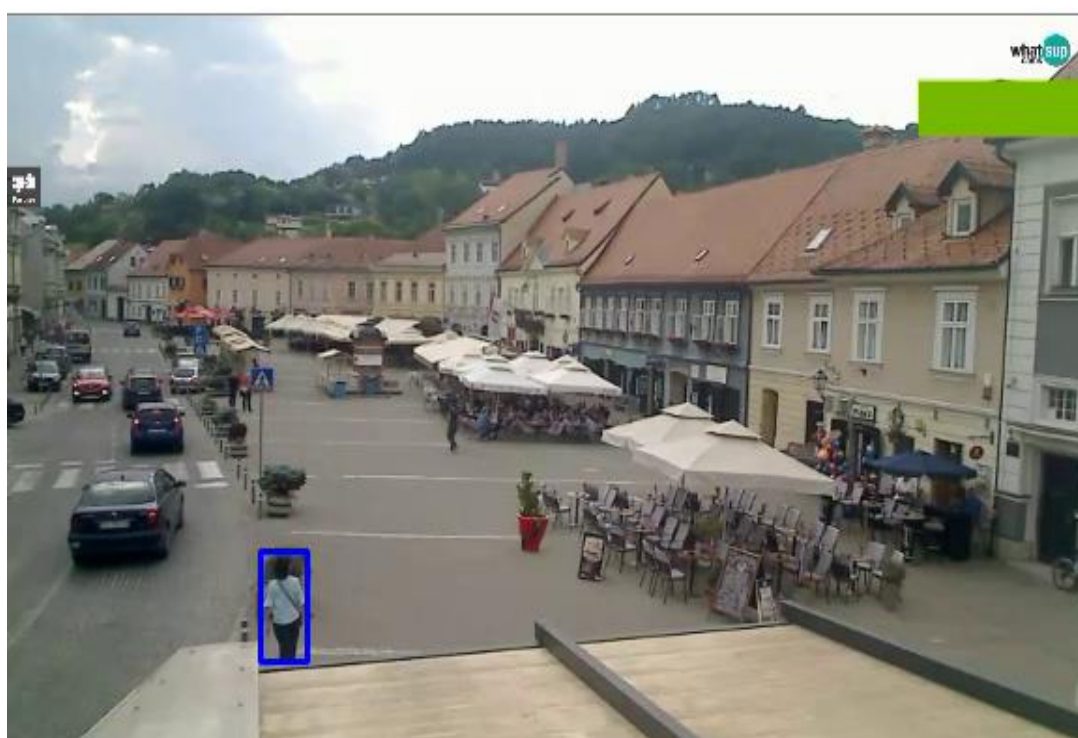
### 3.4. Test 4

Ovaj test se odvija na Trgu kralja Tomislava u Samoboru. Cilj ovog testa je provjeriti kako se algoritam nosi s promijenama veličine i oblika objekta koji se prati u videozapisu.

Tablica 7. Vrijednosti parametara četvrtog testa

Parametar	Vrijednost	Parametar	Vrijednost
$r$	2	$s$	1
$n$	10	$w$	26
$m$	10	$h$	54
$Q$	0.4		

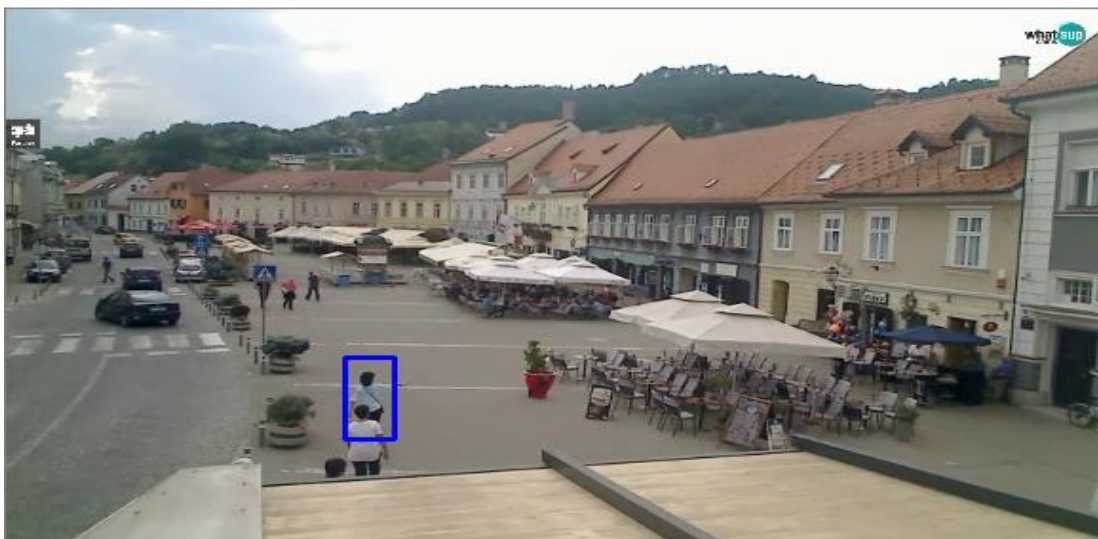
Parametar  $r$  je postavljen na vrijednost 2 (4 x 4 rešetka) iz razloga što je videozapis sniman u 30 sličica po sekundi, a osoba koja se prati se ne kreće brzo u odnosu na broj sličica po sekundi.



Slika 18. Početni odabir objekta



Slika 19. Objekt se vidi iz profila



Slika 20. Djelomično prekrivanje objekta



Slika 21. Objekt je u isto vrijeme prekriven te ga se vidi iz profila





Slika 22. Objekt pokriva puno manje dimenzije

Tablica 8. Brzina Izvođenja

Proces	Vrijeme izvođenja
<b>Računanje integralne slike</b>	0.008s
<b>Usporedba 2 predloška</b>	0.01s
<b>Računanje novog centra</b>	0.29s

Zaključak:

- Ako zadržimo veliki broj fragmenata, ukupno vrijeme izračuna novog centra se može drastično smanjiti smanjenjem vrijednosti varijable  $r$ . Naravno ovaj pristup se može koristiti samo u situacijama gdje se objekt kreće relativno sporo u odnosu na broj sličica po sekundi.
- Za dovoljan broj fragmenata algoritam se dobro nosio s mijenjanjem veličine objekta, mijenjanjem oblika objekta i prekrivanjem objekta
- Rezultati ovog testa su zadovoljavajući

## 4. Moguće nadogradnje sustava

### 4.1. Višedretvenost

Današnja računala često imaju procesore s nekoliko jezgri, najčešće četiri ili osam. To omogućava da više dretvi izvodi proračune paralelno.

Računanje udaljenosti za svaki predložak je neovisno o računanju udaljenosti drugog predloška. Primjerice  $n$  dretvi bi moglo istovremeno računat međusobno neovisno. Nakon što svaka dretva napravi svoje izračune trebalo bi usporediti minimume koji je svaka od njih izračunala i kao novi centar postaviti lokaciju povezanu s tom najmanjom udaljenosti.

### 4.2. Skaliranje okvira

Kao što je prikazano u četvrtom testu veličina objekta koji se prati ne mora biti stalna, te se može mijenjati iz slike u sliku.

Pristupi:

1. Izračunati najmanju udaljenost te provjeriti je li udaljenost manja za uvećanje i smanjenje predloška. Ako je manja udaljenost treba pomnožiti  $w, h$  proizvoljnim faktorom.
2. Pri provjeri svake lokacije se provjerava i udaljenost smanjenog i povećanog okvira. Ovaj pristup je jako skup jer bi vrijeme izvođenja naraslo približno tri puta.

## 5. Zaključak

Tema ovog rada je praćenje objekata u videozapisu s naglaskom na algoritam FragTrack. Praćenje objekata je process praćenja jednog ili više pokretnih objekata u videozapisu.

Kao što je i prikazano u testovima Fragtrack algoritam se dobro nosi s djelomičnim prekrivanjem objekata, mijenjanjem oblika objekta. Premda se algoritam temelji na histogramima fragmenata praćenje objekata koji se dobro stapa s okolinom nije uspješno u većini slučajeva. Naravno otpornost na pogreške nosi svoju cijenu te algoritam nije moguće koristiti za praćenje u stvarnom vremenu bez smanjivanja prostora pretraživanja, jakog računala i optimizacijskih preinaka.

# LITERATURA

- [1] Amit A., Ehud R., Ilan S.; *Robust Fragments-based Tracking using the Integral Histogram*; *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2006)
- [2] Ashken S.; *What is Computer Vision – Post 5: A Very Quick History*; 04.10.2016.; <https://blippar.com/en/resources/blog/2016/10/04/what-computer-vision-post-5-very-quick-history/> ; 06.06.2018.
- [3] Coldewey D.; *WTF is Computer Vision*; 14. 11. 2018.; <https://techcrunch.com/2016/11/13/wtf-is-computer-vision/> ; 08.06.2018.
- [4] Mallick S.; *Why does OpenCV use BGR color format?*; 27.09.2015.; <https://www.learnopencv.com/why-does-opencv-use-bgr-color-format/> ; 08.06.2018.
- [5] Howard Bear J.; *What is HSV color model?*; 16.03.2018.; <https://www.lifewire.com/what-is-hsv-in-design-1078068> ; 08.06.2018.
- [6] <https://statistics.laerd.com/statistical-guides/understanding-histograms.php> ; 09.06.2018.
- [7] Rubner Y., Tomasi C., Guibas L. J.; *The Earth Mover's Distance as a Metric for Image Retrieval*; *International Journal of Computer Vision*; 40/2000/2; str. 99-121
- [8] <https://old.medialooks.com/mformats/docs/CK%20Advanced.html> , *Advanced Chroma Key settings*, 13.06.2018.



## **Praćenje objekata u videozapisima**

### **Sažetak**

Tema rada je problematika praćenja objekata u videozapisima. Rad sadrži dijelove posvećene potrebnoj teorijskoj podlozi za shvaćanje temeljnih koncepata. Tako su ukratko opisani histogrami, integralne slike, prostori boja. Glavni fokus rada je na Fragtrack algoritmu za praćenje objekata. Algoritam se bazira na uspoređivanju histograma fragmenata predložka objekta. U radu su opisani svi koraci implementacije Fragtrack algoritma. Nakon toga su izvedeni testovi zbog provjere uspješnosti izvedbe algoritma. Testirani su neki složeniji slučajevi, primjerice slučajevi gdje objekt koji se prati biva prekriven ili mijenja svoj oblik ili veličinu. Implementacija je pokazala da se uspješno može nositi s tim problemima, ali ta se otpornost mora platiti vremenom izvođenja koje nažalost nije u stvarnom vremenu.

**Ključne riječi:** Fragtrack, praćenje objekata, praćenje bazirano na fragmentima, računalni vid, histogram

## **Tracking objects in image sequences**

### **Abstract**

Subject of this work is tracking objects in image sequences. Work contains section dedicated to theoretical groundwork required for fully understanding concepts that are discussed. Main focus of this work is Fragtrack object tracking algorithm. FragTrack is algorithm that is based on comparing histograms of fragments created from template image. All steps in implementation of algorithm are mentioned. After the implementation comes testing. Some more complex cases were tested. For example when partial occlusion occurs, when object that is being tracked changes it's shape or size. Implementation was able to handle those difficult cases, but because of that algorithm is not viable for running in real time.

**Keywords:** Fragtrack, object tracking, tracking based on fragments, computer vision, histogram