Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

FERI

Dinamična analiza kode

Simon Plazar

Maribor, marec 2024

KAZALO VSEBINE

KAZALO SLIK

PROGRAMSKI JEZIK, V KATEREM JE IZBRANA NALOGA, TER UPORABLJENO ORODJE ZA PROFILIRANJE IN ANALIZO POKRITOSTI

Programski jezik: Python

Orodje za profiliranje: Python paket **line_profiler**

Orodje za analizo pokoristi: Python paket **coverage**

REZULTAT ORODJA ZA ANALIZO POKRITOSTI

```
(venv) PS D:\Coding\Current\prepoznava_pesca\System\Detection> coverage report -m .\MUL_Stream.py
Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
.\MUL_Stream.py     154     56    64%   12-13, 23, 29, 49-58, 73-74, 82-83, 89-90, 96, 102-130, 154-165, 176, 190, 198, 216, 221-224, 241-246, 273, 280
-----------------------------------------------
TOTAL               154     56    64%
```

Slika 1: Coverage-Stream

```
(venv) PS D:\Coding\Current\prepoznava_pesca\System\Detection> coverage report -m .\MUL_Detect.py
Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
.\MUL_Detect.py     207     64    69%   20-21, 31, 37, 81-82, 90-91, 97-98, 104, 113-114, 123-124, 130-132, 168-170, 177, 194, 213-234, 250-251, 279, 286, 303, 320-336, 339-355
-----------------------------------------------
TOTAL               207     64    69%
```

Slika 2: Coverage-Display

REZULTAT ORODJA ZA PROFILIRANJE PRED IMPLEMENTIRANIMI IZBOLJŠAVAMI

ProfilerDisplaySlow.txt

ProfilerStreamSlow.txt

IDENTIFICIRANE NAJPOČASNEJŠE DELE KODE

Najpočasnejši odseki kode so naslednji:

Stream:

Ustvarjanje video toka: video = cv2.VideoCapture(video_path)

Prejemanje slike iz toka: ret, frame = video.read()

Prikazovanje slike za razhroščevanje: cv2.imshow('Video Producer', …

Display:

Preverjanje možnih cuda naprav: if torch.cuda.is_available():

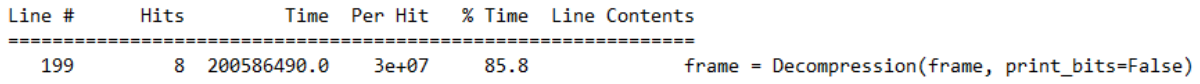
Nalaganje uteži za prepoznavanje: model = YOLO(weights_path)

Ustvarjanje kafka prodcera: producer = kafka.KafkaProducer…

Prejemanje zaznav od modela: results = model.predict *

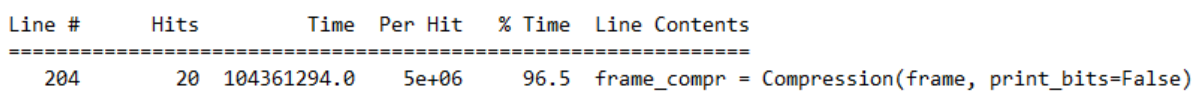
Prikazovanje slike za razhroščevanje: cv2.imshow('Video Consumer', … *

Zadnjih dveh odsekov kode, ki sta časovno potratna, se nemoremo izognit in se izvedeta vsako iteracijo zanke. Naslednji najbolj potratni odsek kode, ki se izvaja pa je kompresija in dekompresija slik videoposnetka.



Line #	Hits	Time	Per Hit	% Time	Line Contents
199	8	200586490.0	3e+07	85.8	frame = Decompression(frame, print_bits=False)

Slika 3: Decompression Slow



Line #	Hits	Time	Per Hit	% Time	Line Contents
204	20	104361294.0	5e+06	96.5	frame_compr = Compression(frame, print_bits=False)

Slika 4: Compression Slow

OPIS VZROKOV POČASNOSTI

Natančnejše analiziranje kompresije in dekompresije:


ProfileCompression Slow.txt


ProfileColorConversionSlow.txt


ProfileResize.txt

Iz analize je razvidno, da so najbolj časovno potratni odseki kode tam, kjer sliko prevzorčimo v nižjo ločljivost (97%) in spremenimo barvni prostor (~3%). Ti odseki so počasne, saj so to lastne implementacije obsežnih algoritmov, zapisani v jeziku Python, ki je sam po sebi že počasen jezik.

Vsaka operacija kompresije traja 0.499104 s, dekompresije pa 2.55023 s.

OPIS POPRAVKOV

Lastne implementacije sem zamenjal z optimiziranimi različicami, napisani v hitrejših programskih jezikih, kot so C ali C++. Algoritmi paketa cv2 uporabljajo napredne tehnike za izvajanje teh algoritmov in so posledično dosti hitrejši. Zamenjal sem Vrstice kode s svojo implementacije s funkcijami paketa cv2. Moral sem paziti na obliko podatkov in vrstni red argumentov, saj ima paket cv2 obratni vrstni red argumentov (višina in širina, RGB->BGR…).

REZULTAT ORODJA ZA PROFILIRANJE PO IMPLEMENTIRANIH IZBOLJŠAVAH

Rezultati izboljšave kompresije in dekompresije:


ProfileCompression Fast.txt

Izboljševanje teh dveh funkcij znotraj funkcij za kompresijo in dekompresijo je algoritem zelo pohitrilo. Čas kompresije je padel kar na 0.0037059 s na operacijo in čas dekompresije je padel kar na 0.0027582 s na operacijo.


ProfileStreamFast.txt


ProfileDisplayFast.txt

```
Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
   200       240    5271141.0  21963.1      5.5   frame_compr = Compression(frame, print_bits=False)
```

Slika 5: Compression Fast

```
Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
   195       159    6516687.0  40985.5      8.1  frame = Decompression(frame, print_bits=False)
```

Slika 6: Decompression Fast

Z izboljšavo algoritma za kompresijo, pretvarjanje barvnega prostora in prevzorčenja slike so izboljšave jasno vidne. Algoritem sedaj teče gladko, hitro in brez kolcanja. Izboljšala se je konsistentnost toka in s tem zanesljivost aplikacije.