FishTracker

*Universitatea "Dunărea de Jos" Galați*

*Facultatea de Automatică, Calculatoare, Inginerie Electrică și Electronică*

*Autor: Iordăchescu Cristian Ștefan*

*Grupa: 22C23*

*Profesor coordonator: Lector Simona Moldovanu*

FishTracker este o simplă aplicație de detectare a unor pești într-un acvariu pe baza unui clip video

## Instalare

Pentru a instala FishTracker, trebuie să aveți instalat Python 3.7 sau mai nou. Apoi, puteți instala dependențele printr-un nou Python virtual environment:

```powershell

python3 -m venv .venv

.\.venv\Scripts\activate.ps1

pip install -r requirements.txt

```

## Utilizare

Pentru a rula FishTracker, trebuie să aveți un clip video în formatul MP4.

Calea clipului trebuie specifica in main.py

```python

video\_path = "sample/fish.mp4"

```

Puteți rula aplicația cu comanda:

```powershell

python main.py

```

## Descriere

1. Se importa librăriile necesare

```python

import cv2

```

1. Se importa clasa ajutătoare pentru urmărirea de obiecte și se inițializează obiectul

```python

from tracker import \*

tracker = EuclideanDistTracker()

```

1. Se importă și citește clipul video

```python

video\_path = "sample/fish.mp4"

cap = cv2.VideoCapture(video\_path)

```

1. Se inițializează detectarea fundalului cu algoritmul *K-nearest neighbours* și se setează parametrii

```python

object\_detector = cv2.createBackgroundSubtractorKNN(

history=100, dist2Threshold=500.0)

```

* 1. history: numărul de cadre pe care le ia în considerare algoritmul
  2. dist2Threshold: distanța maximă până la care se consideră ca un obiect este parte din background sau nu

1. Se parcurg cadrele video și se alege o arie de interes

```python

while cap.isOpened():

ret, frame = cap.read()

roi = frame[5: 715, 50: 1200]

```

1. Se aplică separarea de fundal și se curăță fundalul prin binarizare

```python

mask = object\_detector.apply(roi)

\_, mask = cv2.threshold(mask, 254, 255, cv2.THRESH\_BINARY)

```

1. Se detectează contururile obiectelor

```python

contours, \_ = cv2.findContours(

mask, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

```

1. Se parcurg contururile și se desenează dreptunghiul de interes daca aria conturului este mai mare decat 400px

```python

detections = []

for cnt in contours:

area = cv2.contourArea(cnt)

if area > 400:

x, y, w, h = cv2.boundingRect(cnt)

detections.append([x, y, w, h])

```

1. Se aplica urmărirea de obiecte pe baza distanței euclidiene

```python

boxes\_ids = tracker.update(detections)

```

1. Se desenează dreptunghiul de interes și id-ul obiectului

```python

for box\_id in boxes\_ids:

x, y, w, h, id = box\_id

cv2.putText(roi, str(id), (x, y - 15),

cv2.FONT\_HERSHEY\_PLAIN, 2, (255, 0, 0), 2)

cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 3)

```

1. Se afișează rezultatul in frame dar și masca

```python

cv2.imshow("Mask", mask)

cv2.imshow("Frame", frame)

```

1. Se oprește execuția la apăsarea tastei `ESC`

```python

key = cv2.waitKey(30)

if key == 27:

break

```

1. Se eliberează resursele

```python

cap.release()

cv2.destroyAllWindows()

```

## Referințe

* [Object Tracking with Opencv and Python](https://pysource.com/2021/01/28/object-tracking-with-opencv-and-python/)