Дигитално процесирање на слика

Тема:

Алгоритам за пресметување на Optical Flow во OpenCV

Стефан Тагарски 221184
Давид Христов 221085

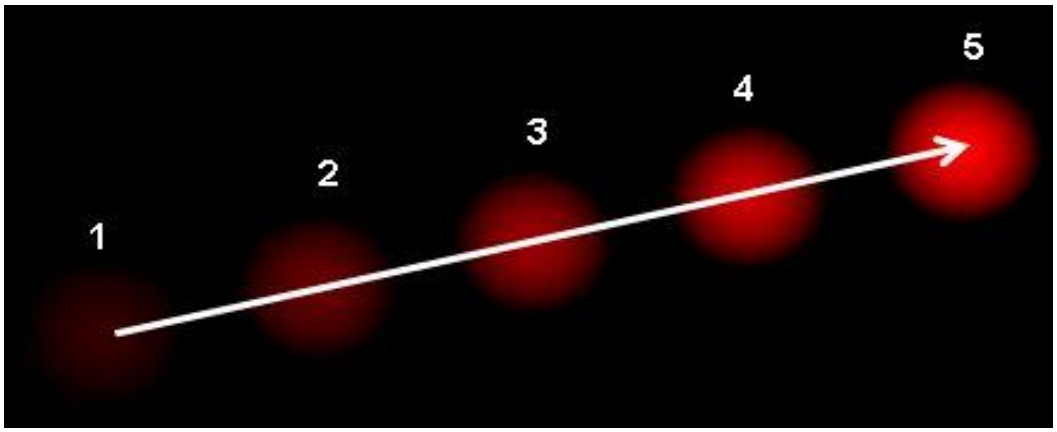
• Вовед
• Преглед на Optical Flow
• Примени на Optical Flow
• Optical Flow Algorithms
• Lucas-Kanade Optical Flow
• Farneback Optical Flow
• Deep Learning-based Optical Flow
• Споредба на методите на Optical Flow
• Избор на метод на Optical Flow
• Имплементација на Lucas-Kanade Optical Flow во OpenCV • Имплементација на Farneback Optical Flow in OpenCV
•
• Заклучок

Вовед

Оптичкиот тек(анг. Flow) е клучен концепт во компјутерската визија што вклучува проценка на движењето помеѓу два последователни кадри во видео секвенца. Ова движење обично е претставено како векторско поле, каде што секој вектор претставува поместување на пиксел од првата рамка до втората. Разбирањето и имплементирањето на Оптичкиот тек може да обезбеди значителен увид во откривањето на движење, следењето на објекти и разни други апликации.

Концептот на оптички проток беше воведен од американскиот психолог Џејмс Џ. Гибсон во 1940-тите за да го опише визуелниот стимул што им се дава на животните кои се движат низ светот. Оптичкиот тек, или оптичкиот проток, е модел на привидно движење на предмети, површини и рабови во визуелна сцена предизвикано од релативното движење помеѓу набљудувачот и сцената. Може да се дефинира и како дистрибуција на привидните брзини на движење на обрасците на осветленоста во сликата. Гибсон ја истакна важноста на оптичкиот тек за перцепција на способноста да се согледаат можностите за дејствување во околината.

Во роботиката и компјутерската визија, терминот оптички проток опфаќа техники поврзани со обработка на слики и контрола на навигацијата. Овие техники вклучуваат детекција на движење, сегментација на објекти, информации за време до контакт, фокус на пресметки за проширување, осветленост, кодирање компензирана со движење и мерење на стерео диспаритет. Оптичкиот тек во суштина е 2D векторско поле каде секој вектор е вектор на поместување што го покажува движењето на точките од првата рамка до втората, предизвикано од движењето на објектот или камерата.



(Прикажува топка која се движи во 5 последователни рамки. Стрелката го прикажува неговиот вектор на поместување)

Преглед на Optical Flow

Техниките на оптички проток го анализираат привидното движење на обрасците на осветленоста во низа на слики. Овие техники претпоставуваат дека очигледната осветленост на која било точка на сликата останува константна со текот на времето. Примарната цел е да се пресмета векторот на движење за секој пиксел на сликата.

Основни принципи

Константност на осветленоста: Светлината на пикселот останува константна додека се движи од една рамка во друга.

Просторна кохерентност: соседните пиксели во рамнината на сликата обично имаат слично движење.

Временска кохерентност: полето за движење непречено се менува со текот на времето.

Оптичкиот тек работи на неколку претпоставки:

1. Интензитетот на пикселите на објектот не се менува помеѓу последователни рамки.
2. Соседните пиксели имаат слични движења.

Размислете за пиксел $\boldsymbol{I(x, y, t)}$ во првата рамка, се движи по растојание (dx, dy) во следната рамка преземена по dt време. Значи, бидејќи тие пиксели се исти и интензитетот не се менува, можеме да кажеме, $\boldsymbol{I(x, y, t) = I(x + dx,\ y + dy,\ t + dt)}$

Потоа земете ја приближувањето на Тејлоровата серија на десната страна, отстранете ги вообичаените поими и поделете го со $\boldsymbol{dt}$ за да се добие следнава равенка: $f_x u + f_y v + f_t = 0$

каде:

$$f_x = \frac{\partial f}{\partial x}\ ;\ f_y = \frac{\partial f}{\partial y}$$

$$u = \frac{dx}{dt}\ ;\ v = \frac{dy}{dt}$$

Горенаведената равенка се нарекува равенка на оптички проток, Каде ***(u,v)*** се координатите на пиксел на сликата.

Примени на Optical Flow

Оптичкиот тек има широк опсег на примени во компјутерската визија, вклучувајќи, но не ограничувајќи се на:

Следење на објекти: Идентификување и следење објекти преку низа рамки.

Компресија на видео: Намалување на вишокот со кодирање на информации за движење.

Откривање на движење: Откривање и анализа на движење во видео секвенци.

Стабилизација: компензира за движењето на камерата за да се создадат стабилни видео секвенци.

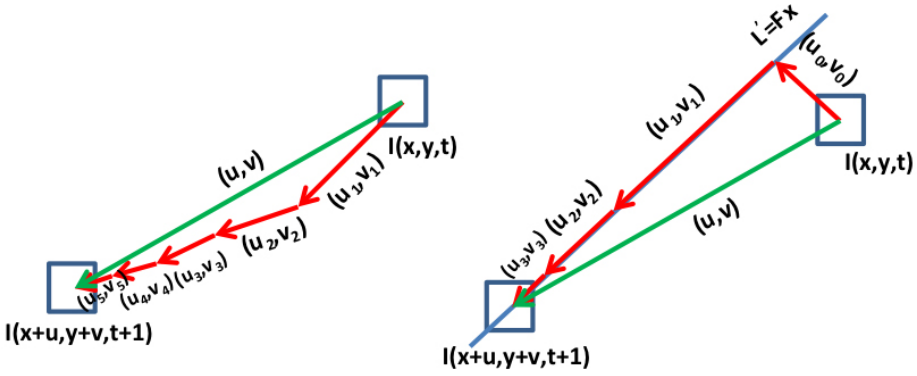
3D реконструкција: Заклучување на информации за длабочина од движење.

Optical Flow Algorithms

Алгоритмите за оптички тек имаат за цел да го проценат движењето помеѓу последователните кадри во видео секвенца. Овие алгоритми може да се категоризираат на ретки и густи методи. Ретки методи го пресметуваат протокот на подмножество пиксели, често избрани од аголни детектори, додека густите методи го пресметуваат протокот на секој пиксел во рамката. Овој дел ги опфаќа најшироко користените алгоритми, нивните принципи, предности и ограничувања.

Lucas-Kanade Optical Flow

Методот Лукас-Канаде е техника на редок оптички проток што претпоставува мало и речиси постојано движење во локално соседство на пиксели. Овој метод ги решава векторите на поместување со користење на просторни и временски градиенти на сликата.



Проценка на оптички проток: Лево: Лукас-Канаде; Десно: Лукас-Канаде потпомогнат од епиполарната геометрија

Клучни карактеристики:

Sparse метод: Пресметува проток за збир од избрани точки (обично агли).

Локално соседство: користи мал прозорец околу секоја точка за да го процени движењето.

Линеарни равенки: Решава систем на линеарни равенки добиени од претпоставката за постојаност на осветленоста.

Математичка формулација:

Со оглед на две последователни рамки It и It+1, методот Лукас-Канаде го проценува поместувањето(u,v) со решавање на следнава равенка за секој пиксел во локално соседство:

$$\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_x I_t \\ -I_y I_t \end{bmatrix}$$

Каде:

• Ix и Iy се градиенти на сликата во x и y насоките, соодветно.

• Toa е временски градиент (разлика помеѓу последователни рамки).

Предности:

1. Ефикасен и погоден за апликации во реално време.
2. Едноставен за спроведување.

Ограничувања:

1. Претпоставува мало движење, кое може да не важи за објекти кои брзо се движат.
2. Перформансите се намалуваат во региони со ниска текстура (на пр., униформни области).

Farneback Optical Flow

Методот Farneback е dense алгоритам за оптички проток кој ги приближува соседствата на пикселите користејќи полиномни проширувања. Овој метод обезбедува густо поле на проток, проценувајќи го движењето за секој пиксел во рамката.

Клучни карактеристики:

Dense метод: Пресметува проток за секој пиксел во рамката.

Проширување на полином: Моделира соседства на слики со полиномни функции.

Цврсти до големи поместувања: Може да се справи со поголеми движења во споредба со Лукас-Канаде.

Математичка формулација:

Методот Farneback одговара на полином на соседството на секој пиксел за приближно да ја приближи функцијата на сликата. Потоа ги користи овие приближувања за да ги процени векторите на поместување. Алгоритмот итеративно го усовршува полето на проток за да ја подобри точноста.

Предности:

1. Обезбедува густи информации за протокот.
2. Попрецизно за поголеми движења.

Ограничувања:

Компјутерски интензивни во споредба со ретки методи.

Потребна е поголема меморија и процесорска моќ.

Deep Learning-based Optical Flow

Неодамнешните достигнувања во длабокото учење доведоа до развој на методи за проценка на оптички проток користејќи конволутивни невронски мрежи (CNN). Овие методи, како што се FlowNet и PWC-Net, користат големи збирки на податоци и техники за учење за да ги предвидат полињата на проток.

Клучни карактеристики:

Управување со податоци: обучени за големи колекции на податоци на видеа и соодветни полиња на проток.

Учење од крај до крај: мрежата учи да го проценува оптичкиот проток директно од необработените вредности на пиксели.

Висока прецизност: Може да фати сложени модели на движење што традиционалните методи може да ги пропуштат.

FlowNet:

FlowNet е еден од првите модели на оптички проток базиран на CNN. Се состои од архитектура на енкодер-декодер, каде што енкодерот извлекува карактеристики од влезните рамки, а декодерот го предвидува полето на проток.

Предности:

1. Учи од податоците, кои можат добро да се генерализираат на различни сценарија.
2. Може да се справи со големи и сложени движења.

Ограничувања:

1. Потребна е голема количина на означени податоци за обука.
2. Пресметано скапи за време на обуката и заклучоците.

PWC-Net:

PWC-Net (Pyramid, Warping и Cost Volume Network) го подобрува FlowNet со инкорпорирање на пристап базиран на пирамида и обем на трошоци за подобро снимање на деталите за движењето.

Предности:

1. Попрецизен од FlowNet, особено за мали и брзи движења.
2. Ефикасно во однос на пресметковните ресурси во споредба со FlowNet.

Ограничувања:

1. Сè уште бара значителна пресметковна моќ за обука.
2. Зависност од квалитетот и квантитетот на податоците за обуката.
3. Споредба на методите на оптички проток

Споредба на методите на Optical Flow

Ретки(sparse) наспроти густи(dense) методи:

Sparse методи (на пр. Лукас-Канаде):

1. Побрзо и помалку пресметковно интензивно.
2. Погоден за апликации каде што движењето треба да се следи само во клучните точки (на пр., следење на објекти).

Густи методи (на пр., Farneback, засновано на длабоко учење):

1. Обезбедете детални информации за движење низ целата слика.
2. Пресметковно повеќе бара, но нуди побогати информации за апликации како видео стабилизација и анализа на движење.

Традиционални наспроти методи засновани на длабоко учење:

Традиционални методи (на пр. Лукас-Канаде, Фарнебек):

1. Врз основа на математички модели и претпоставки.
2. Потребно е рачно подесување на параметрите.

Методи засновани на длабоко учење (на пр., FlowNet, PWC-Net):

1. Научете шеми на движење директно од податоци.
2. Генерално попрецизни, но бараат големи збирки на податоци и значителни пресметковни ресурси за обука.

Избор на метод на оптички проток

Изборот на методот на оптички проток зависи од специфичните барања на апликацијата:

Апликации во реално време: Лукас-Канаде се претпочита поради неговата ефикасност.

Детална анализа на движење: Далечните или методите засновани на длабоко учење обезбедуваат подетални полиња на проток.

Комплексни модели на движење: Методите засновани на длабоко учење се одлични во снимањето сложени и големи движења.

На следните страници ќе разгледаме in-depth Имплементација на Lucas-Kanade и Farneback Optical Flow во OpenCV.

Имплементација на Lucas-Kanade Optical Flow in OpenCV

Во овој дел, ќе разговараме за имплементацијата на алгоритмот за оптички тек Лукас-Канаде користејќи OpenCV во Python. Ќе го поминеме секој чекор од кодот и ќе дадеме објаснувања за подобро разбирање.

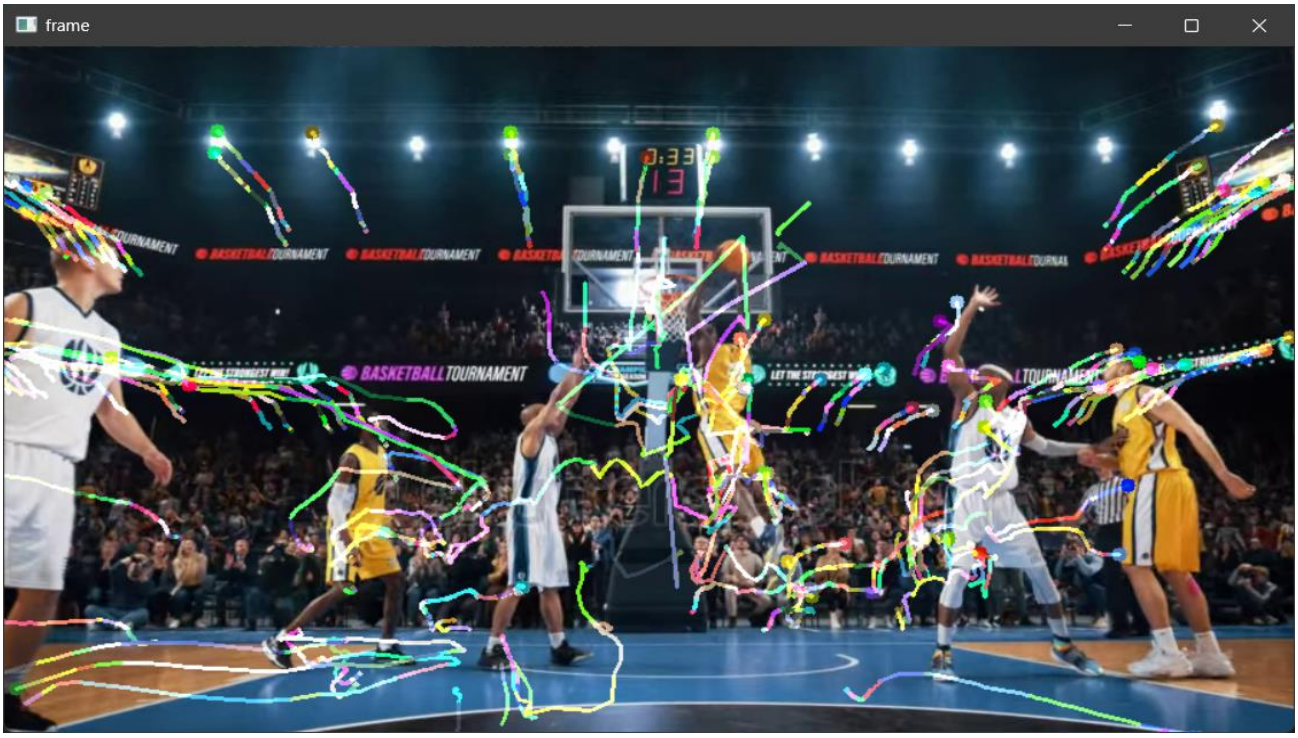
```
import numpy as np
import cv2

# Load the video

# cap = cv2.VideoCapture('video1.webm')
cap = cv2.VideoCapture('video2.webm')

# Parameters for Lucas-Kanade optical flow
lk_params = dict(winSize=(15, 15),
                 maxLevel=2,
                 criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10,
0.03))

# Parameters for Shi-Tomasi corner detection
feature_params = dict(maxCorners=100,
                      qualityLevel=0.3,
                      minDistance=7,
                      blockSize=7)

# Read the first frame and find initial points
ret, old_frame = cap.read()
old_gray = cv2.cvtColor(old_frame, cv2.COLOR_BGR2GRAY)
p0 = cv2.goodFeaturesToTrack(old_gray, mask=None, **feature_params)

# Create a mask image for drawing
mask = np.zeros_like(old_frame)

# Colors for drawing different tracks
colors = np.random.randint(0, 255, (100, 3))

# Process each frame in the video
try:
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Calculate optical flow
        p1, st, err = cv2.calcOpticalFlowPyrLK(old_gray, frame_gray, p0, None, **lk_params)
```

```
        if p1 is not None:
            good_new = p1[st == 1]
            good_old = p0[st == 1]

            # Draw the tracks
            for i, (new, old) in enumerate(zip(good_new, good_old)):
                a, b = np.int32(new.ravel())
                c, d = np.int32(old.ravel())
                color = colors[i % 100].tolist()
                print("a, b:", a, b)
                print("c, d:", c, d)
                mask = cv2.line(mask, (a, b), (c, d), color, 2)
                frame = cv2.circle(frame, (a, b), 5, color, -1)

            img = cv2.add(frame, mask)
            cv2.imshow('frame', img)
            k = cv2.waitKey(30) & 0xff
            if k == 27:
                break

            # Update the previous frame and previous points
            old_gray = frame_gray.copy()
            p0 = good_new.reshape(-1, 1, 2)
        else:
            # Re-detect points if no points are tracked
            p0 = cv2.goodFeaturesToTrack(old_gray, mask=None, **feature_params)

except Exception as e:
    print(f"Error: {e}")

cap.release()
cv2.destroyAllWindows()
```

Објаснување:

1. **Внесување на библиотеки (Importing Libraries)**:
   - Импортираме потребните библиотеки: numpy за нумерички операции и cv2 за задачи на компјутерската визија.
2. **Вчитување на видео (Loading the Video)**:
   - Го вчитуваме видеото користејќи cv2.VideoCapture('video2.webm').
3. **Параметри за Лукас-Канаде Оптички поток (Parameters for Lucas-Kanade Optical Flow)**:
   - Поставуваме параметри за алгоритмот на Лукас-Канаде за оптички поток користејќи речник lk_params. Овие параметри вклучуваат големина на прозорец, максимално ниво на пирамида и критериум за завршување.

4. **Параметри за детекција на агли од Ши-Томаси (Parameters for Shi-Tomasi Corner Detection)**:
    - Поставуваме параметри за алгоритмот за детекција на агли од Ши-Томаси користејќи речник feature_params. Овие параметри вклучуваат максимален број на агли, ниво на квалитет, минимална дистанца и големина на блок.
5. **Почетен кадар и точки (Initial Frame and Points)**:
    - Читаме првиот кадар од видеото и го претвараме во сива слика (old_gray). Потоа, користиме функцијата cv2.goodFeaturesToTrack за да детектираме почетни точки (p0) во првиот кадар.
6. **Маска и бои (Mask and Colors)**:
    - Креираме маска слика за цртање на следови од оптичкиот поток и генерираме случајни бои за секој след.
7. **Обработка на кадри (Processing Frames)**:
    - Влегуваме во циклус за обработка на секој кадар во видеото.
    - Читаме секој кадар (frame) и го претвараме во сива слика (frame_gray).
    - Пресметуваме оптички поток користејќи cv2.calcOpticalFlowPyrLK меѓу претходниот кадар (old_gray) и тековниот кадар (frame_gray), добивајќи нови точки (p1).
    - Филтрираме точки (good_new, good_old) врз основа на статусот (st) вратен од пресметката на оптичкиот поток.
    - Цртаме следови и ја ажурираме маската за секоја точка користејќи cv2.line и cv2.circle.
    - Прикажуваме го кадарот со следовите (img) и го управуваме внесот на тастатурата за излез од циклусот.
    - Го ажурираме претходниот кадар и точките за следната итерација.
8. **Работа со грешки (Error Handling)**:
    - Ги фатаме исклучоците кои можат да се појават при обработка на видеото и ги прикажуваме пораките за грешка.
9. **Ослободување на ресурси (Release Resources)**:
    - На крај, го ослободуваме захватот на видео (cap) и ги затвараме сите прозорци на OpenCV за чистење на ресурсите.



Пример од еден frame како изгледа имплементацијата на овој алгоритам

Имплементација на Farneback Optical Flow in OpenCV

Во оваа секција, ќе го објасниме кодот за имплементација на методот на Farneback за денсен оптички поток користејќи OpenCV во Python. Ќе обидеме да ги разјасниме секој дел од кодот за подобро разбирање.

```
import cv2
import numpy as np

cap = cv2.VideoCapture('video2.webm')
# cap = cv2.VideoCapture('video1.webm')

ret, frame1 = cap.read()
prvs = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)

hsv_mask = np.zeros_like(frame1)
hsv_mask[..., 1] = 255

while True:
    ret, frame2 = cap.read()
    if not ret:
        break
    next = cv2.cvtColor(frame2, cv2.COLOR_BGR2GRAY)

    flow = cv2.calcOpticalFlowFarneback(prvs, next, None, 0.5, 3, 15, 3, 5, 1.2, 0)

    mag, ang = cv2.cartToPolar(flow[..., 0], flow[..., 1])
    hsv_mask[..., 0] = ang * 180 / np.pi / 2
    hsv_mask[..., 2] = cv2.normalize(mag, None, 0, 255, cv2.NORM_MINMAX)

    bgr = cv2.cvtColor(hsv_mask, cv2.COLOR_HSV2BGR)
    cv2.imshow('frame', bgr)

    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

    prvs = next

cap.release()
cv2.destroyAllWindows()
```

Објаснување:

1. **Вчитување на библиотеките (Importing Libraries)**:
   - Импортираме потребните библиотеки: cv2 за функциите на OpenCV и numpy за нумерички операции.

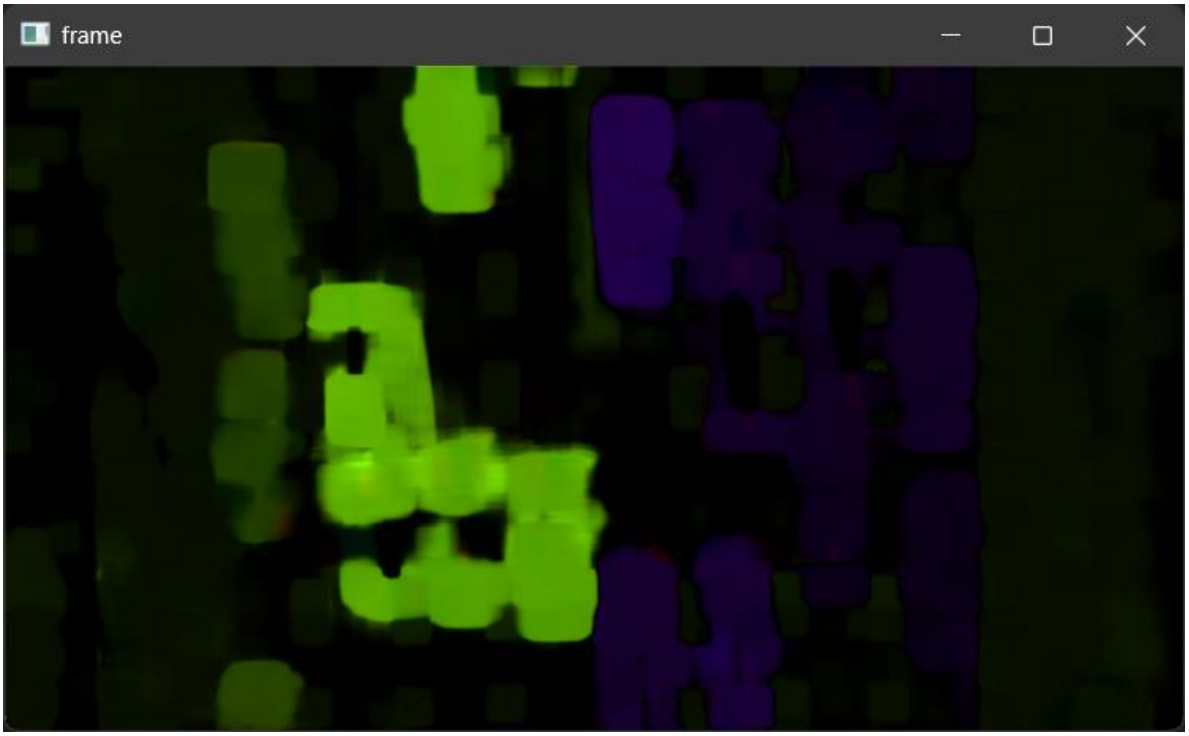
2. **Вчитување на видеото (Loading the Video)**:
   - Го вчитуваме видеото користејќи cv2.VideoCapture('video2.webm').

3. **Прва слика и претходна слика (First Frame and Previous Frame)**:
   - Читаме го првиот кадар од видеото и го конвертираме во сива слика (prvs).

4. **Креирање на маска и поставување на вредности (Mask Creation and Value Initialization)**:
   - Креираме маска за просторно-фреквентен простор (hsv_mask) и поставуваме вредности за степенот на заситеност на бојата.

5. **Обработка на кадри (Frame Processing)**:
   - Влегуваме во циклус за обработка на секој кадар во видеото.
   - Пресметуваме оптички поток со методот на Farneback помеѓу претходниот кадар (prvs) и тековниот кадар (next).
   - Пресметуваме магнитуда и агол на потокот и ги вметнуваме во маската.
   - Преобразуваме маската од HSV во BGR формат за приказ на екран.
   - Чекаме за притискање на ESC за прекин на циклусот.

6. **Ослободување на ресурсите (Releasing Resources)**:
   - Го ослободуваме видеото (cap) и затвараме сите прозорци на OpenCV за чистење на ресурсите.



Пример од еден frame како изгледа имплементацијата на овој алгоритам

Заклучок

Оптичкиот тек е моќна алатка во компјутерската визија со различни апликации. Разбирањето на принципите и имплементацијата на алгоритмите во OpenCV може да обезбеди значителни сознанија и практични решенија за анализа на движење, следење на објекти и многу повеќе. Со внимателно избирање на соодветниот метод и оптимизирање на перформансите, техниките на оптички проток може ефективно да се користат во различни сценарија од реалниот свет.

Користена Литература

- https://docs.opencv.org/4.x/d4/dee/tutorial_optical_flow.html

- https://www.researchgate.net/figure/Optical-flow-estimation-Left-the-Lucas-Kanade-Right-the-Lucas-Kanade-aided-by_fig1_280567385

- https://github.com/opencv

- https://www.geeksforgeeks.org/python-opencv-dense-optical-flow/

- https://learnopencv.com/optical-flow-in-opencv/