ГУАП

КАФЕДРА № 42

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| старший преподаватель |  |  |  | Т.А. Суетина |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №8 |
| РАБОТА С ДАННЫМИ ВИДЕОФАЙЛА ФОРМАТА MP4 |
| по курсу: Техника аудиовизуальных средств информации |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4329 |  |  |  | Д.С. Шаповалова |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2025

# **1. Цель работы:**

получить теоретические знания о структуре видеофайла, типах кадров и принципов их обработки. На практике реализовать полученные знания.

# **2. Задание:**

Для файла в формате mp4 выполнить:

1. Считать параметры заголовка и вывести на экран полученные данные.
2. Создать файл, в который загрузить до 100 последовательных кадров и сохранить их в формате JPG.
3. Вывести время и I-кадры из сохраненного файла.
4. Вывести время и P-кадры из сохраненного файла.
5. Вычислить максимальное значение межкадровой разности сохраненной последовательности кадров.
6. С помощью ffmpeg создать видео с внешним представлением для отображения векторов движения.

Для всех заданий указать используемые функции из библиотеки, описать их назначение, принцип работы.

# **3. Ход работы:**

**Чтение заголовка файла**

С помощью библиотеки python-ffmpeg и Python программы был считан заголовок mp4 файла, заголовок видеопотока и заголовок аудиопотока (рисунок 1).

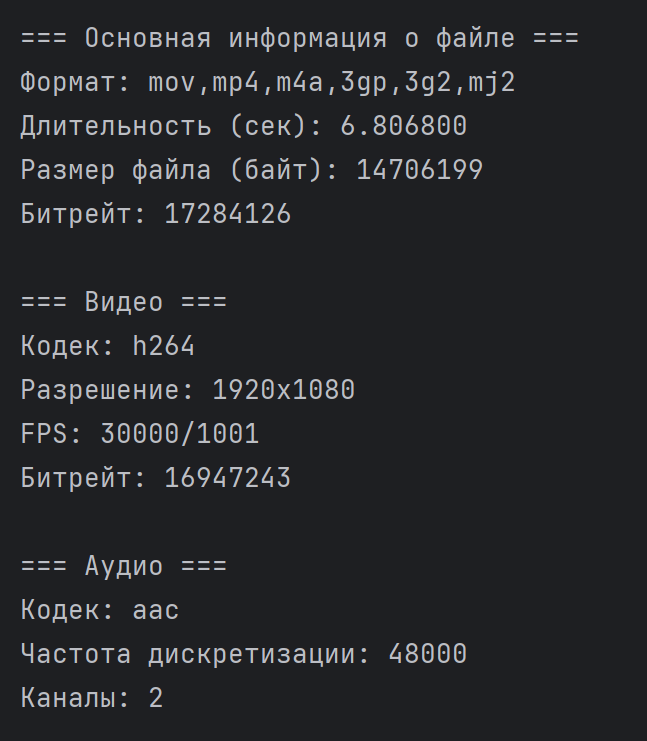


Рисунок 1.1 – Заголовок mp4 файла

С помощью ffprobe, путём ввода команды –

*ffprobe -v error -select\_streams v:0 -show\_entries stream=pix\_fmt -of default=noprint\_wrappers=1:nokey=1 "C:\Users\etoki\Documents\SUAI\_09.03.02\5\_semester\ТАСИ\8 лр\Футаж\_PIXEL\_light\_and\_sun.mp4"*

– в консоль, было получено значение битовой глубины (рисунок 1.2)

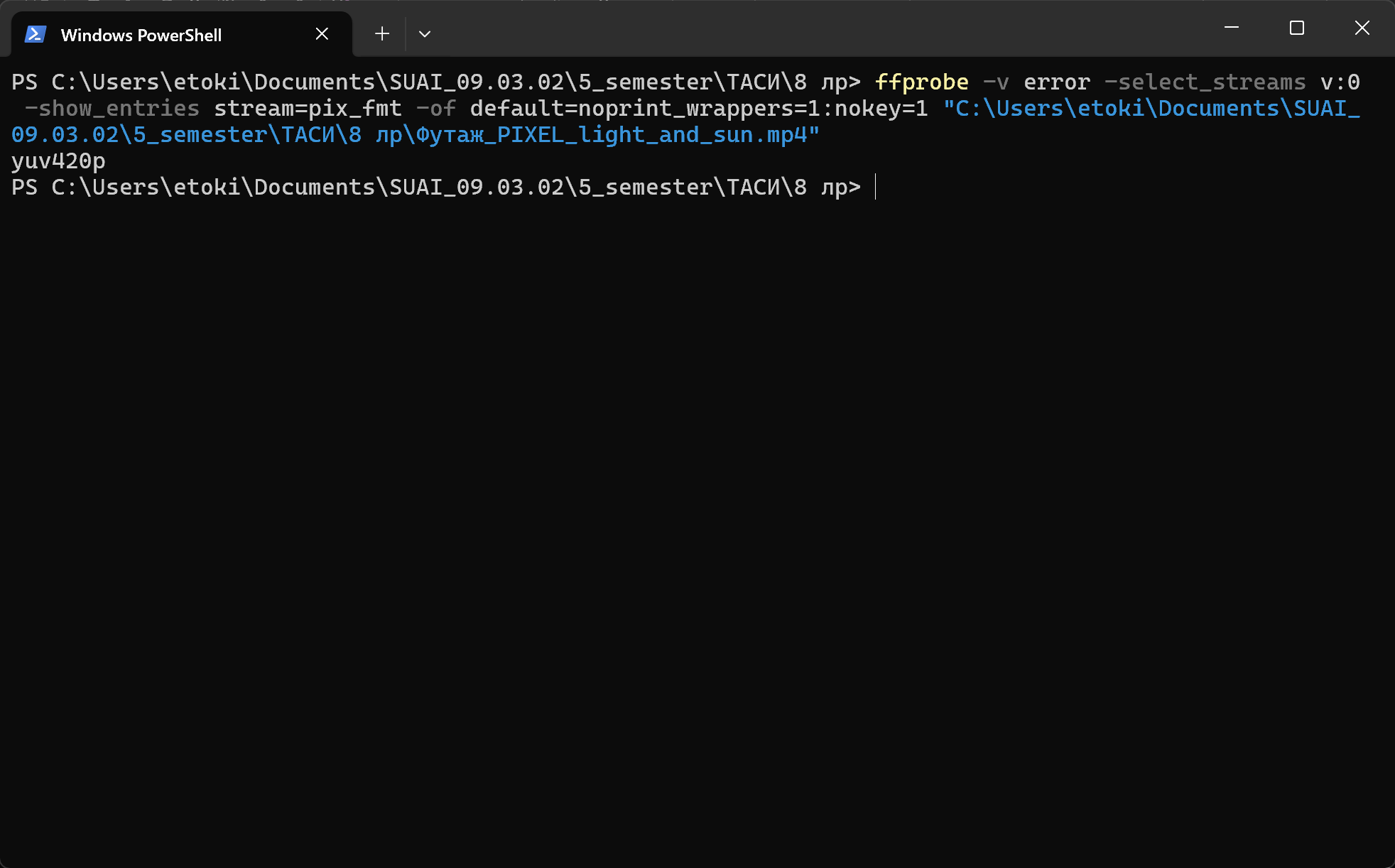


Рисунок 1.2 – Битовая глубина видео

Используемая функция:

* ffmpeg.probe(video\_path) — из библиотеки ffmpeg-python

Назначение:

Получение метаданных видеофайла (формат, потоки, длительность, битрейт и пр.) без декодирования содержимого.

Принцип работы:

* Вызывает утилиту ffprobe.
* Возвращает структуру в формате JSON, содержащую:
  + format — общая информация о контейнере (MP4 и т.д.);
  + streams — список потоков (видео, аудио, субтитры), с деталями: кодек, разрешение, частота кадров, битрейт и др.
* Не требует распаковки видео — работает быстро и эффективно.

**Кадры**

С помощью Python были извлечены первые 100 кадров и сохранены в формате JPG.

В качестве примера представлены первые 3 кадра на рисунках 2.1-2.3:



Рисунок 2.1 – Кадр 1



Рисунок 2.2 – Кадр 2



Рисунок 2.3 – Кадр 3

Используемые средства:

* Через ffmpeg-python:

python

1

* Или через утилиту ffmpeg:

bash

1

Назначение:

Извлечение фиксированного числа кадров из видео и сохранение в виде отдельных изображений.

Принцип работы:

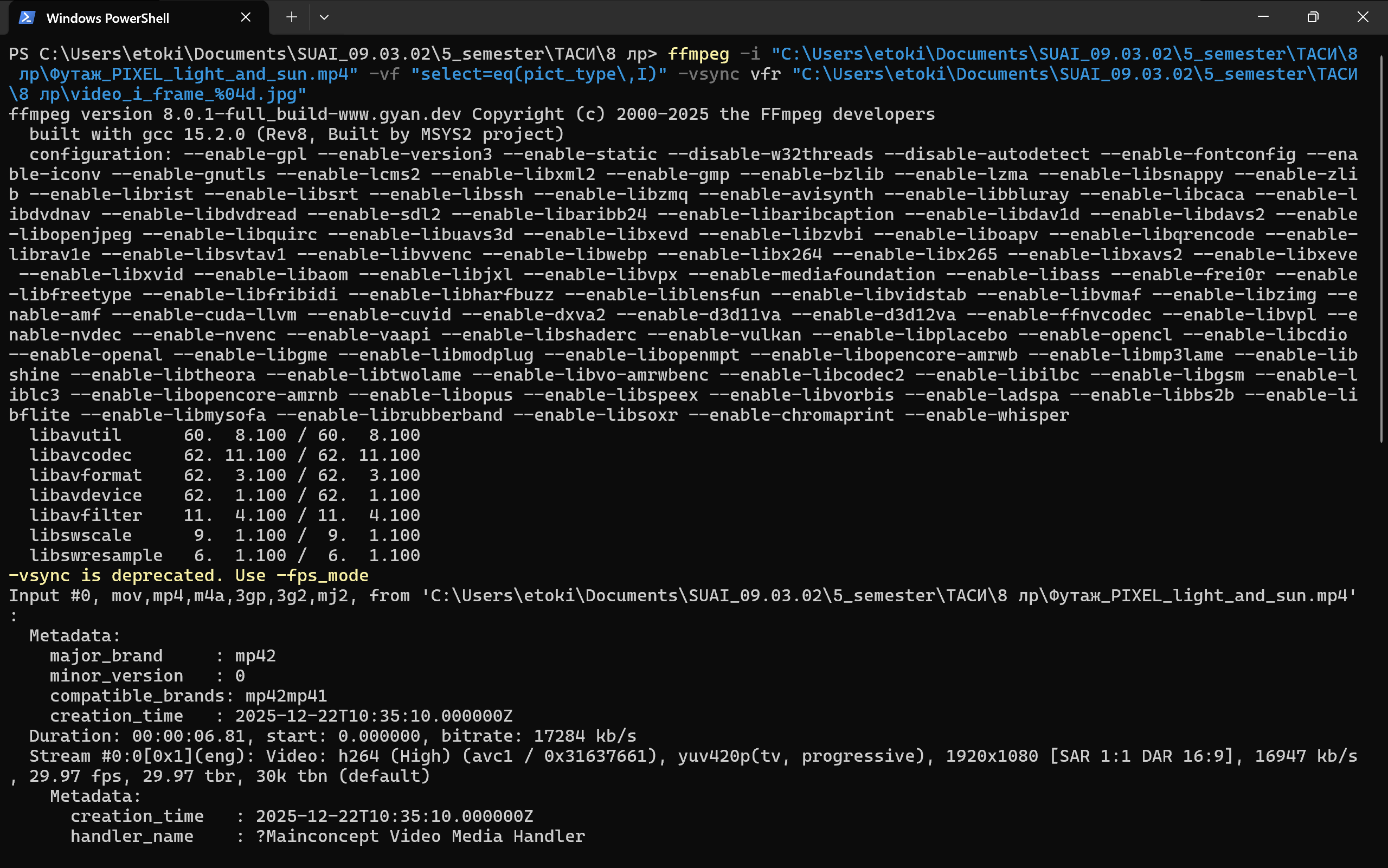
* -vframes N — указывает, сколько кадров декодировать с начала видео.
* FFmpeg декодирует видео и сохраняет каждый кадр как отдельный файл в формате JPG.
* Имена файлов генерируются по шаблону (например, frame\_001.jpg, frame\_002.jpg, …).

**I-кадры**

С помощью команды –

*ffmpeg -i “C:\Users\etoki\Documents\SUAI\_09.03.02\5\_semester\ТАСИ\8 лр\Футаж\_PIXEL\_light\_and\_sun.mp4” -vf “select=’eq(pict\_type\,i)’” -vsync vfr “C:\Users\etoki\Documents\SUAI\_09.03.02\5\_semester\ТАСИ\8 лр\video\_i\_frame\_%04d.jpg”*

– были извлечены из видеопотока все кадры и сохранены только те, у которых параметр *pict\_type* равен I.



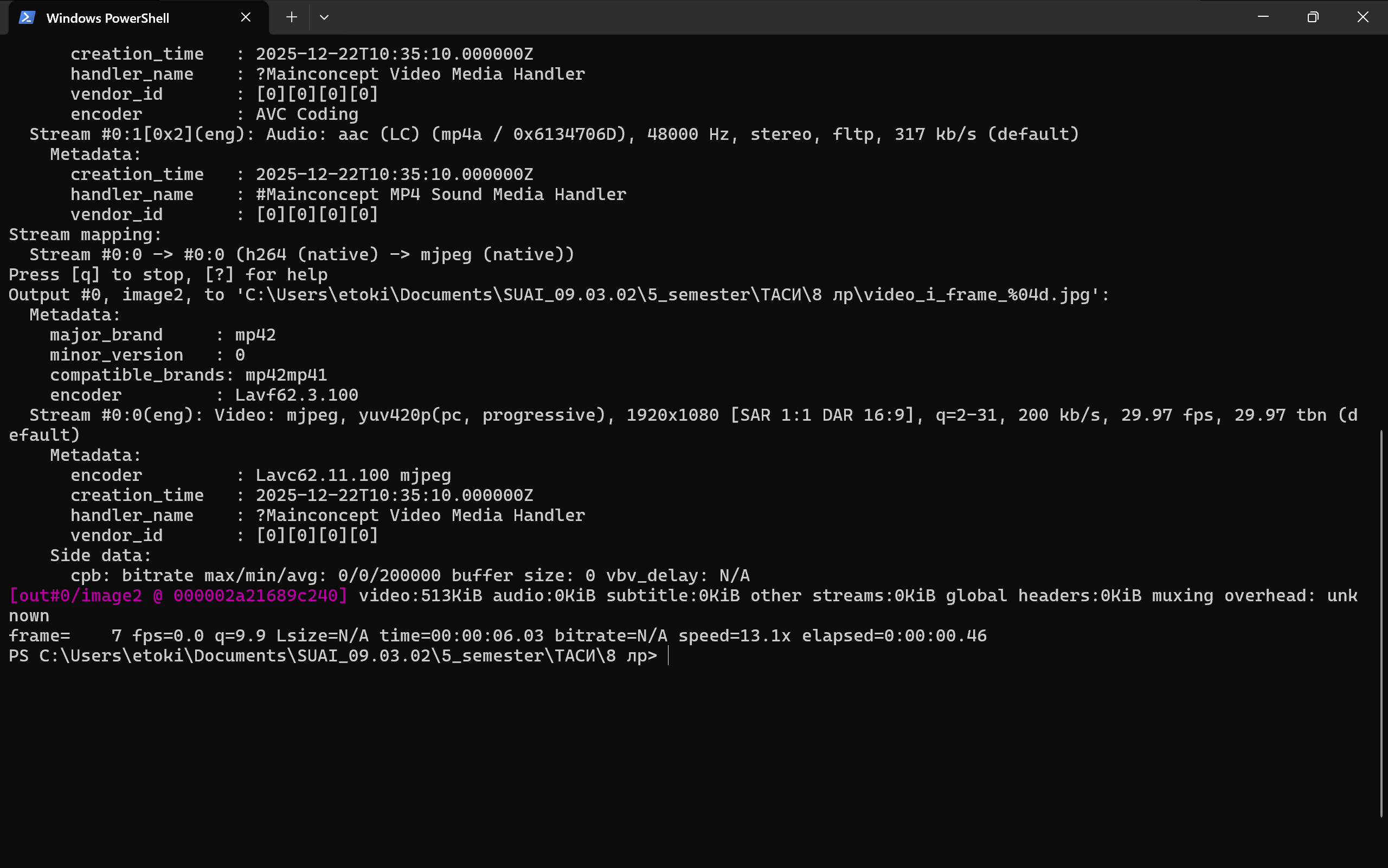


Рисунок 3.1.1 – Выполнение команды

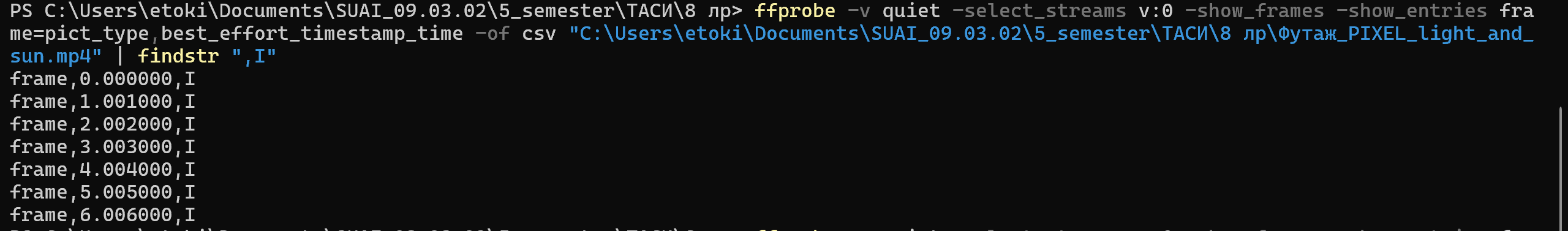


Рисунок 3.1.2 – Время появления I кадров, вывод в консоль

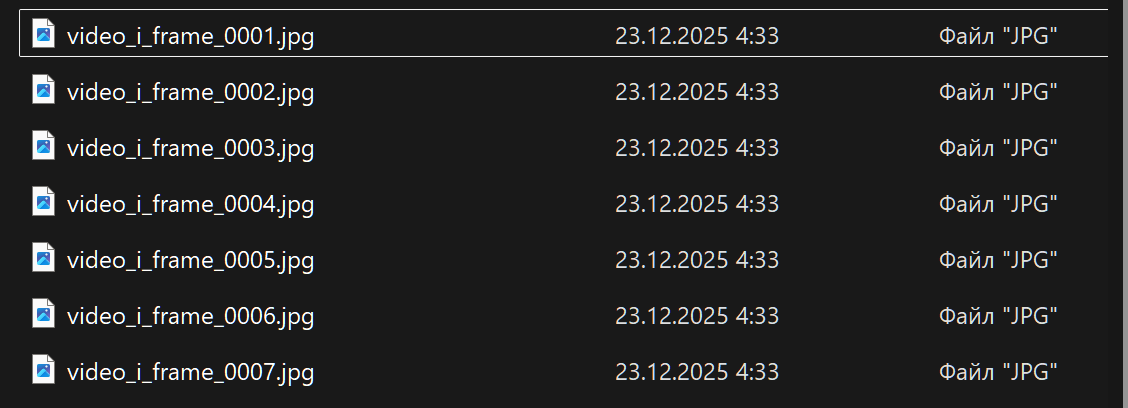


Рисунок 3.2 – Полученные I-кадры в проводнике



Рисунок 3.3.1 – Первый I-кадр

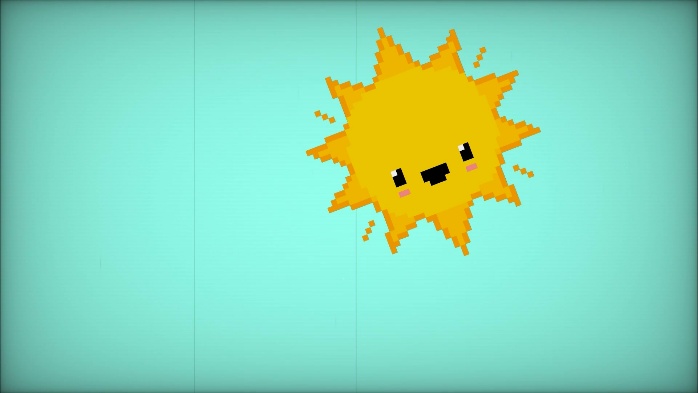


Рисунок 3.3.1 – Второй I-кадр

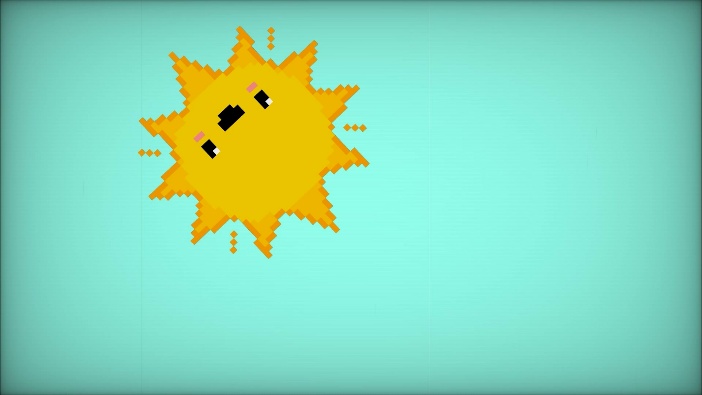


Рисунок 3.3.1 – Третий I-кадр

С помощью команды –

*ffprobe -v quiet -select\_streams v:0 -show\_frames -print\_format compact "C:\Users\etoki\Documents\SUAI\_09.03.02\5\_semester\ТАСИ\8 лр\Футаж\_PIXEL\_light\_and\_sun.mp4" | Out-File -FilePath "C:\Users\etoki\Documents\SUAI\_09.03.02\5\_semester\ТАСИ\8 лр\data1\frames\_info.txt" -Encoding UTF8*

– были извлечены данные о временных метках всех кадров.

Далее, с помощью Python были отобраны I-кадры и сохранены их временные метки.

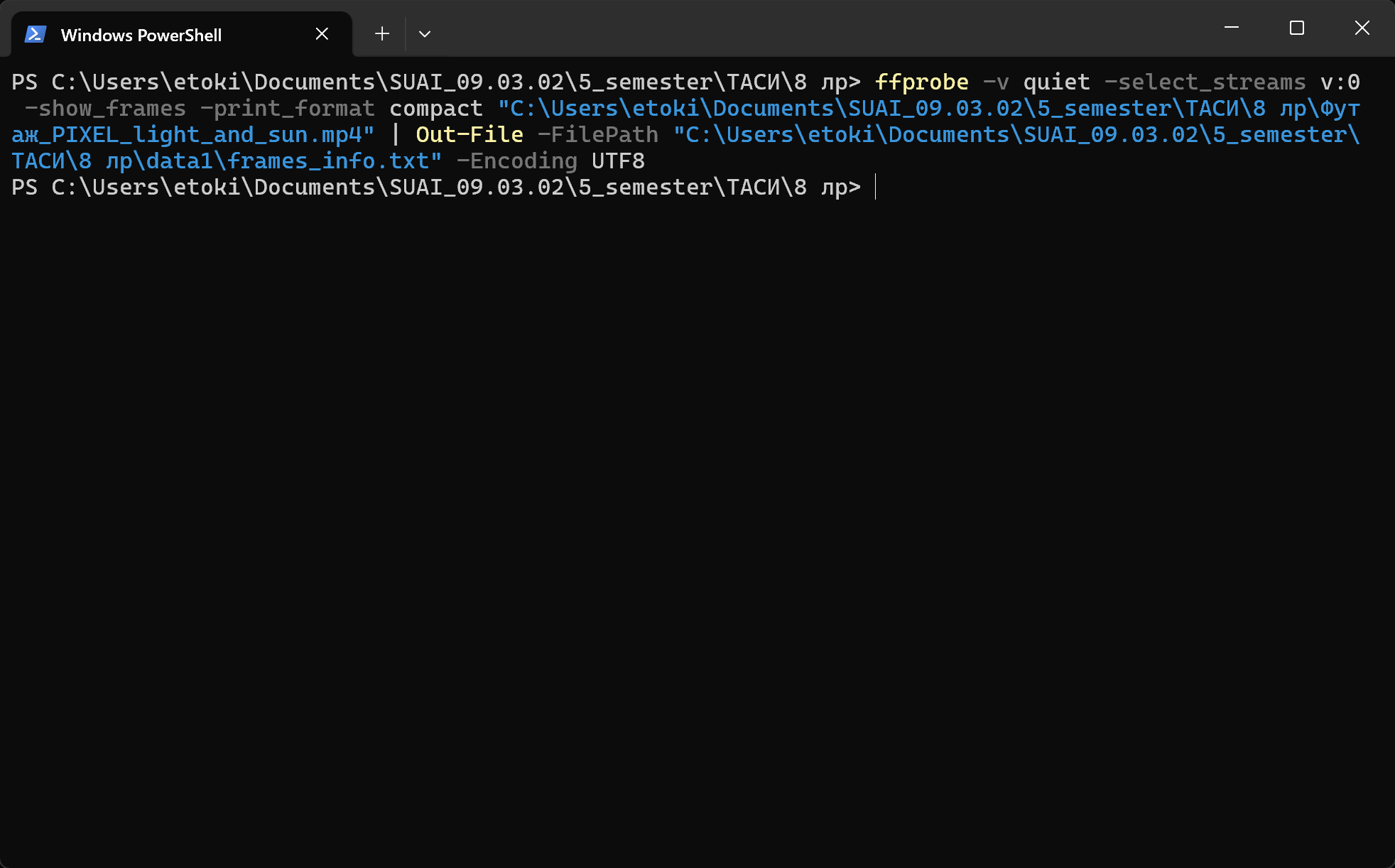


Рисунок 4.1 – Выполненная команда

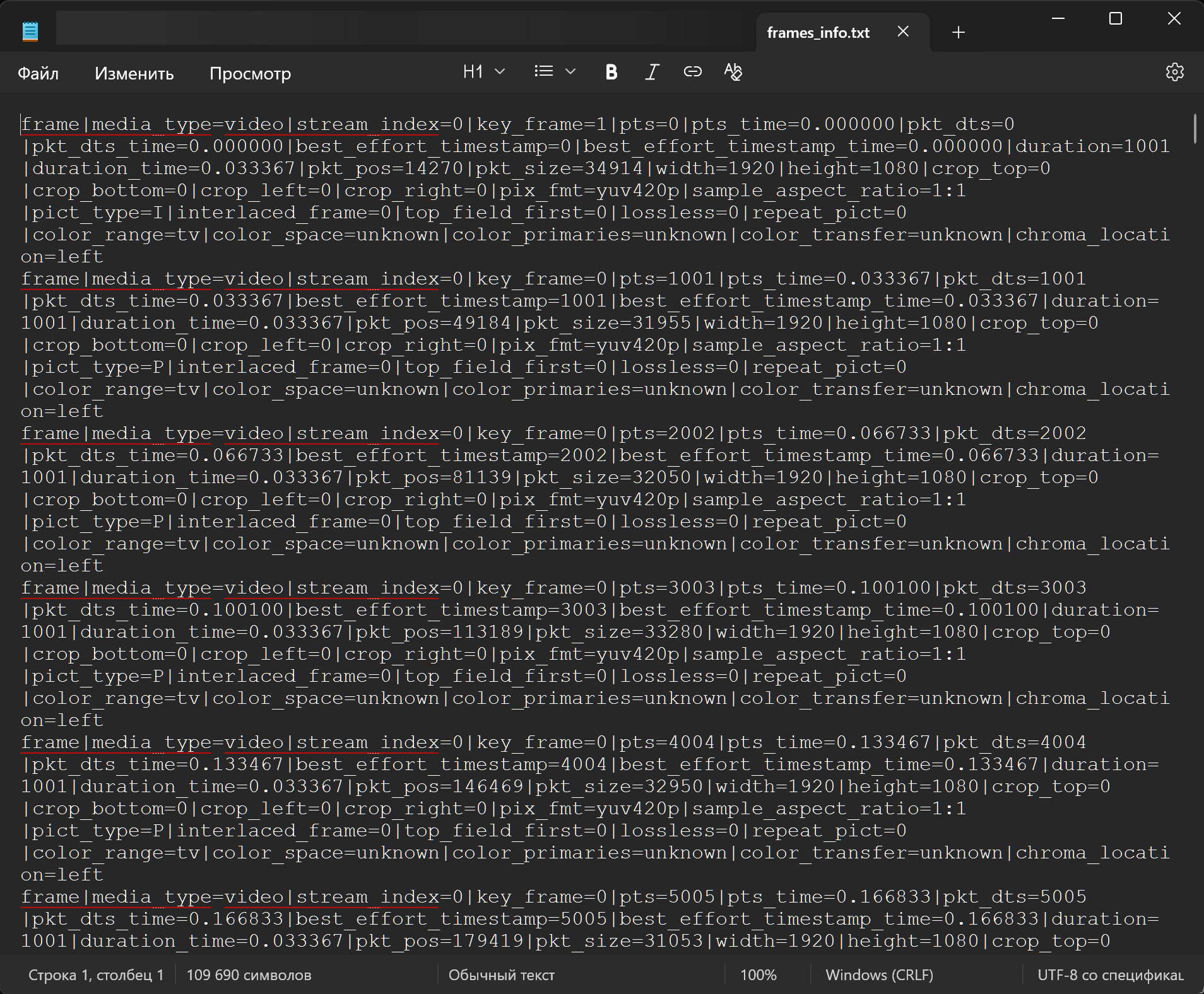


Рисунок 4.2 – Временные метки всех кадров (первые 5 кадров)

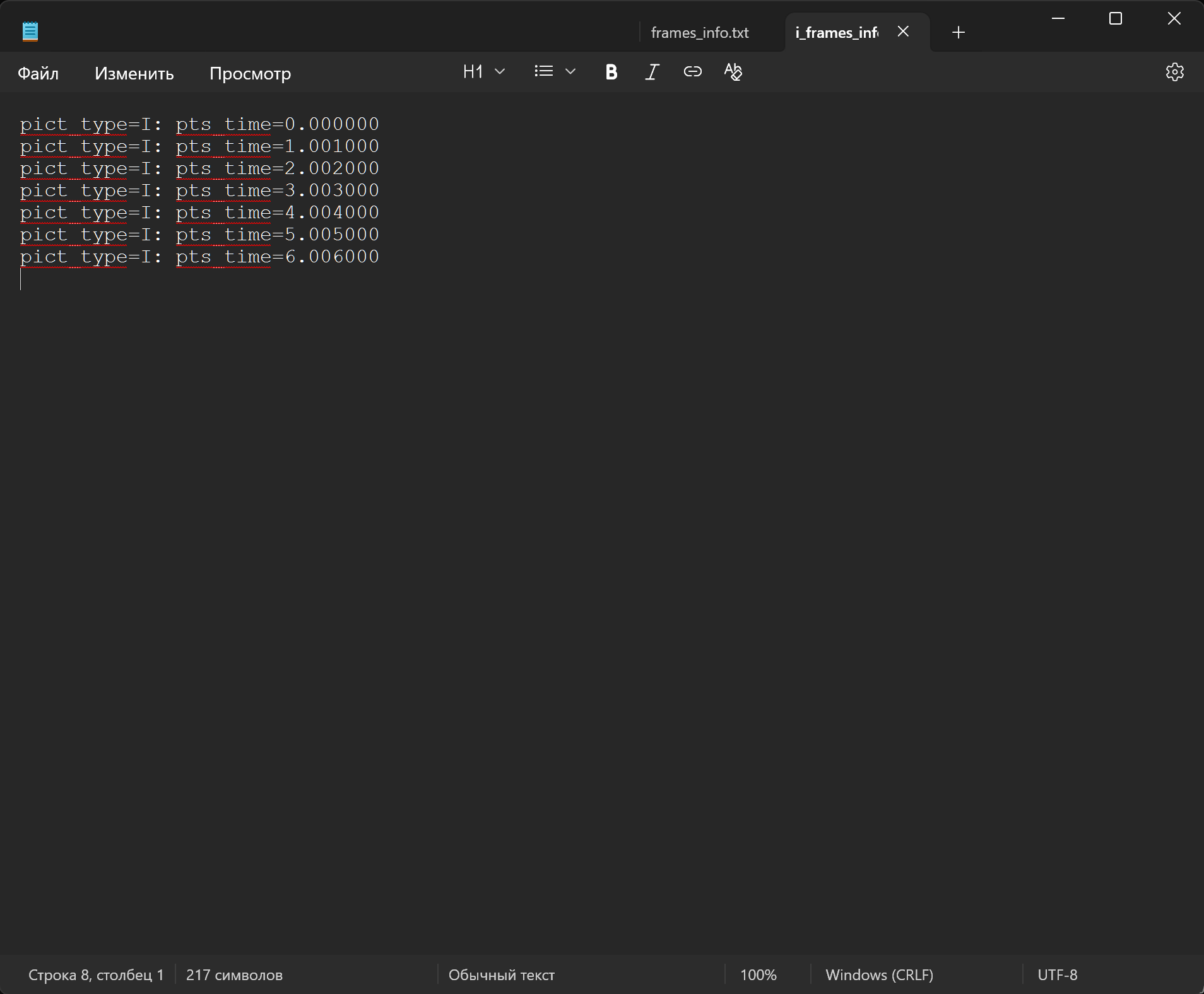


Рисунок 4.3 – Временные метки I-кадров

Используемые средства:

* Утилита ffprobe.
* Обработка вывода в Python (через subprocess.run() или ffmpeg.probe() с JSON).

Назначение:

Получение временных меток и типов всех кадров в видеопотоке.

Принцип работы:

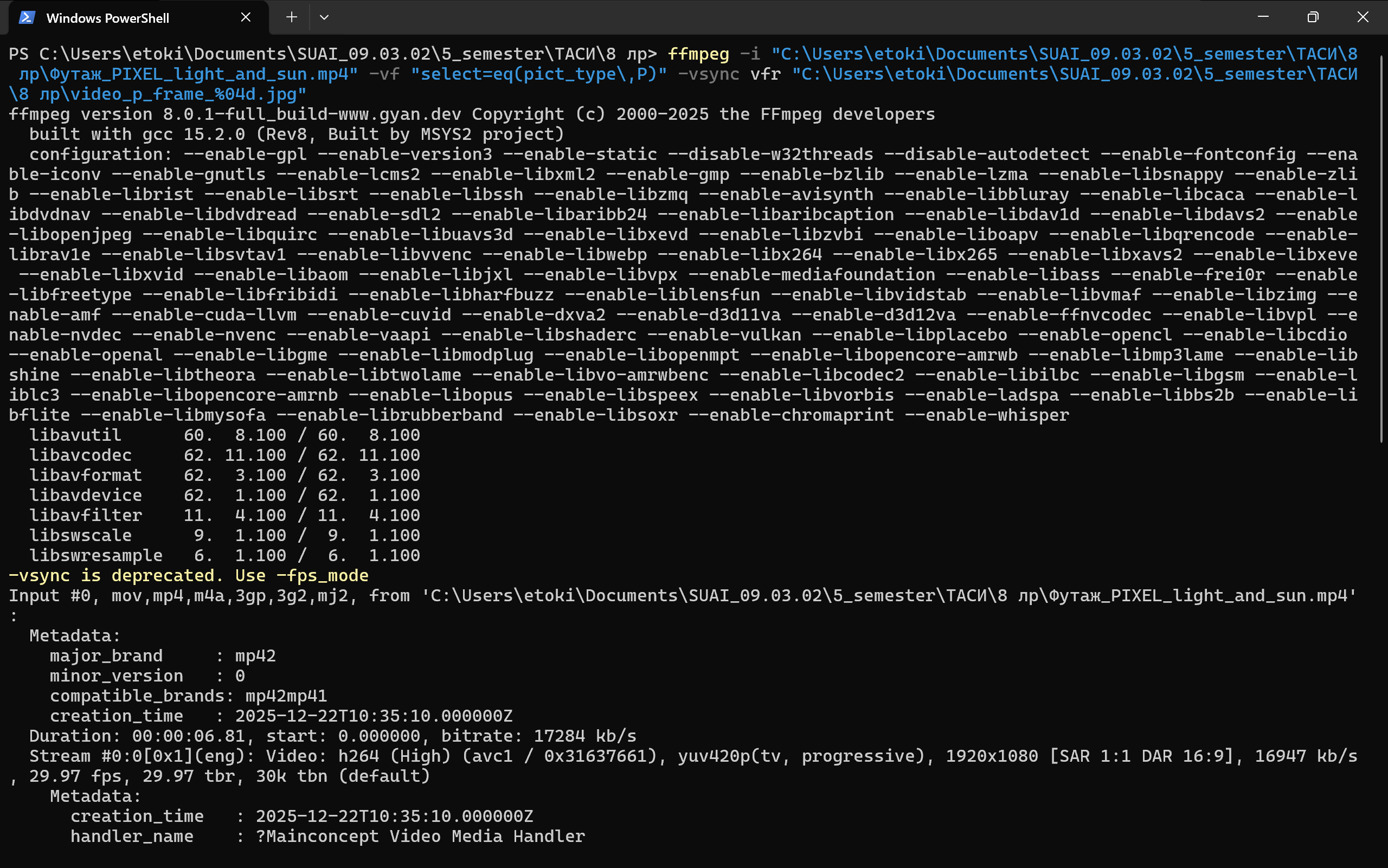
* ffprobe проходит по каждому закодированному кадру и извлекает:
  + pict\_type — тип кадра (I, P, B);
  + best\_effort\_timestamp\_time — время в секундах.
* В Python фильтруются строки с pict\_type=I, и из них сохраняются временные метки.

**P-Кадры**

Команда –

*ffmpeg -i "C:\Users\etoki\Documents\SUAI\_09.03.02\5\_semester\ТАСИ\8 лр\Футаж\_PIXEL\_light\_and\_sun.mp4" -vf "select=eq(pict\_type\,P)" -vsync vfr "C:\Users\etoki\Documents\SUAI\_09.03.02\5\_semester\ТАСИ\8 лр\video\_p\_frames\video\_p\_frame\_%04d.jpg"*

– были извлечены из видеопотока все кадры и сохранены только те, у которых параметр *pict\_type* равен P.



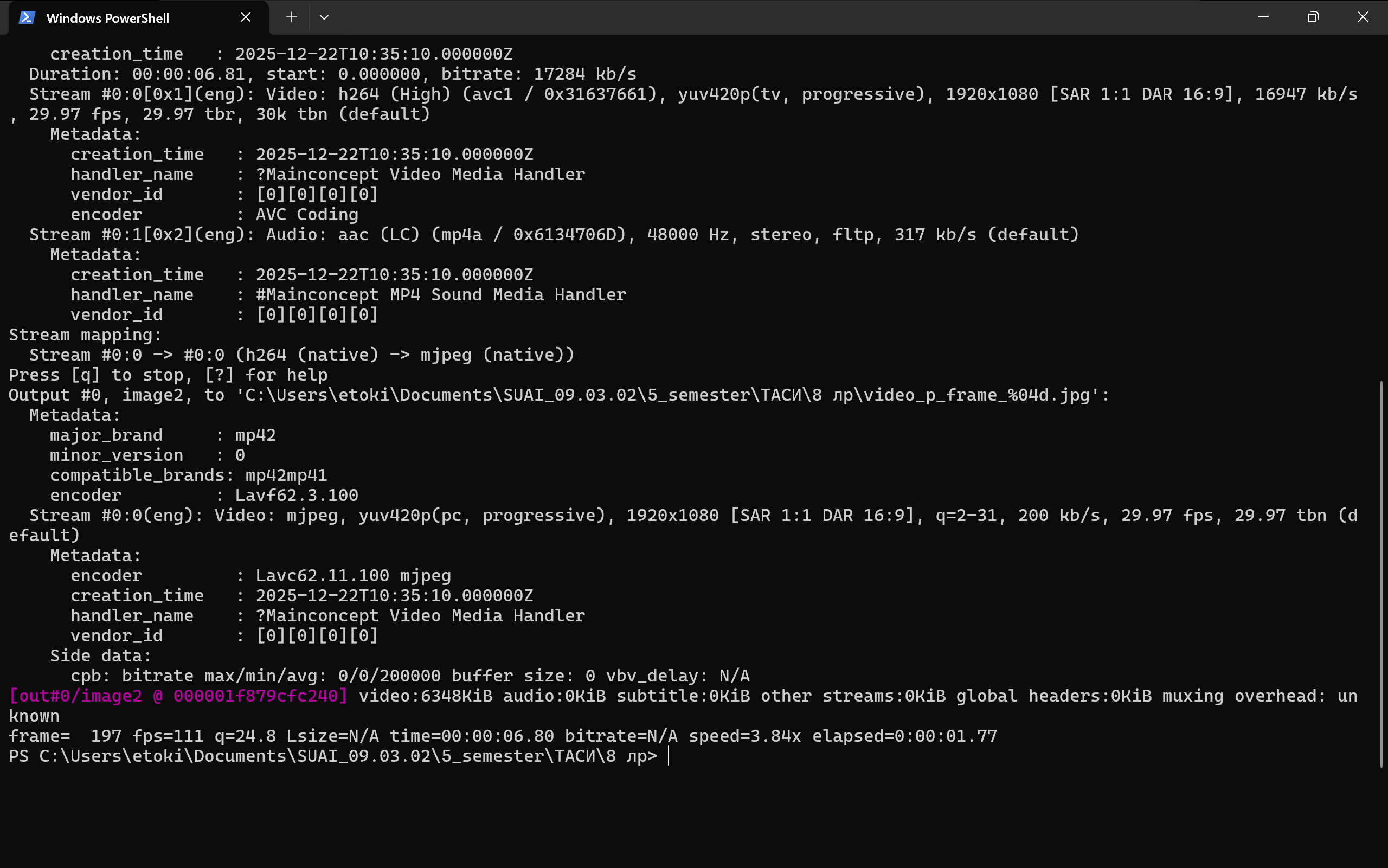


Рисунок 5.1.1 – Выполнение команды

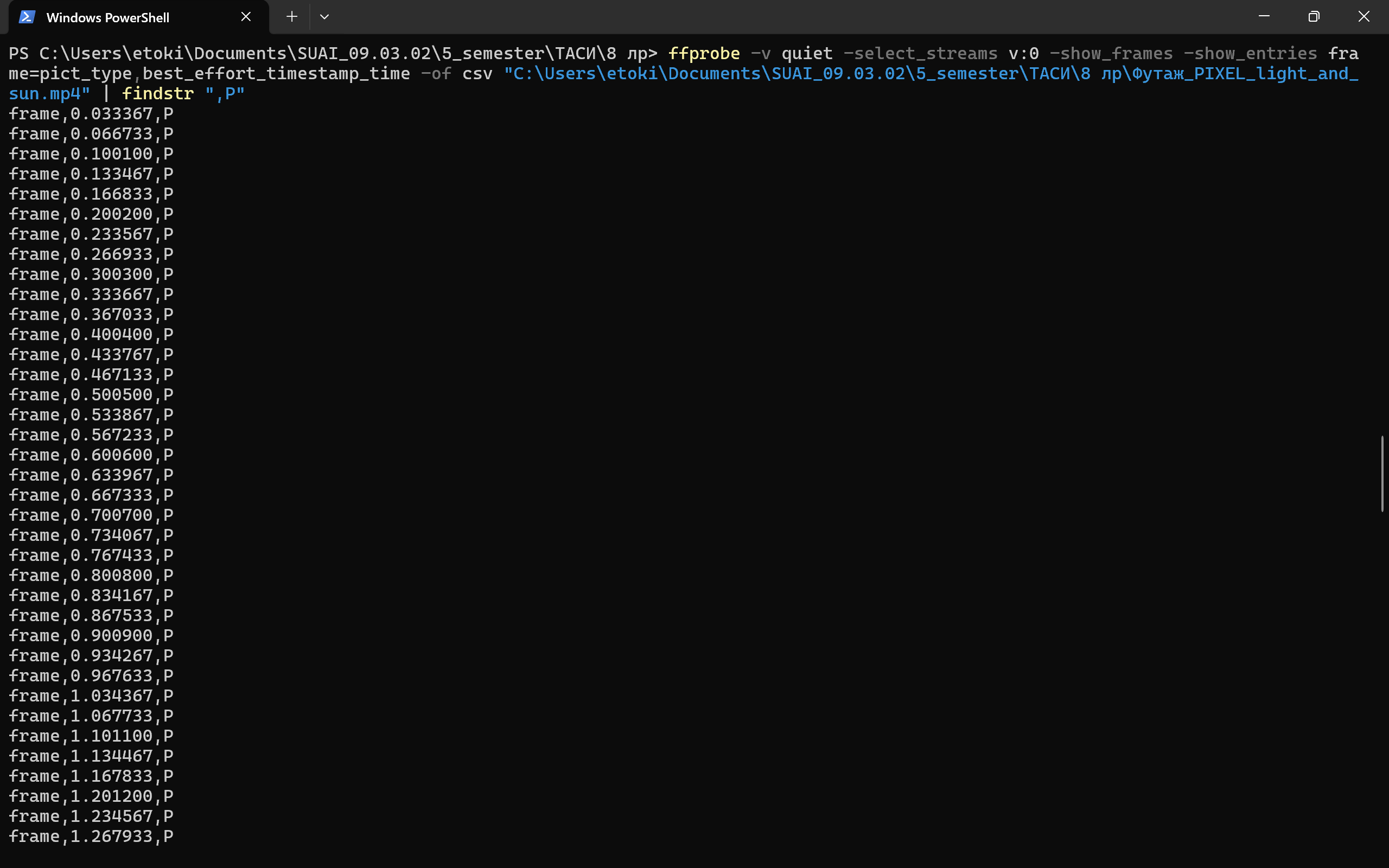


Рисунок 5.1.2 - Время появления P-кадров, вывод в консоль

Общее число P-кадров – 197, частично представлены на рисунках 5.2 – 5.3.3:



Рисунок 5.2 – Полученные P-кадры в проводнике



Рисунок 5.3.1 – Первый P-кадр



Рисунок 5.3.2 – Второй P-кадр



Рисунок 5.3.3 – Третий P-кадр

Из ранее извлечённых данных о временных метках всех кадров были отобраны с помощью Python P-кадры и сохранены их временные метки отдельно.

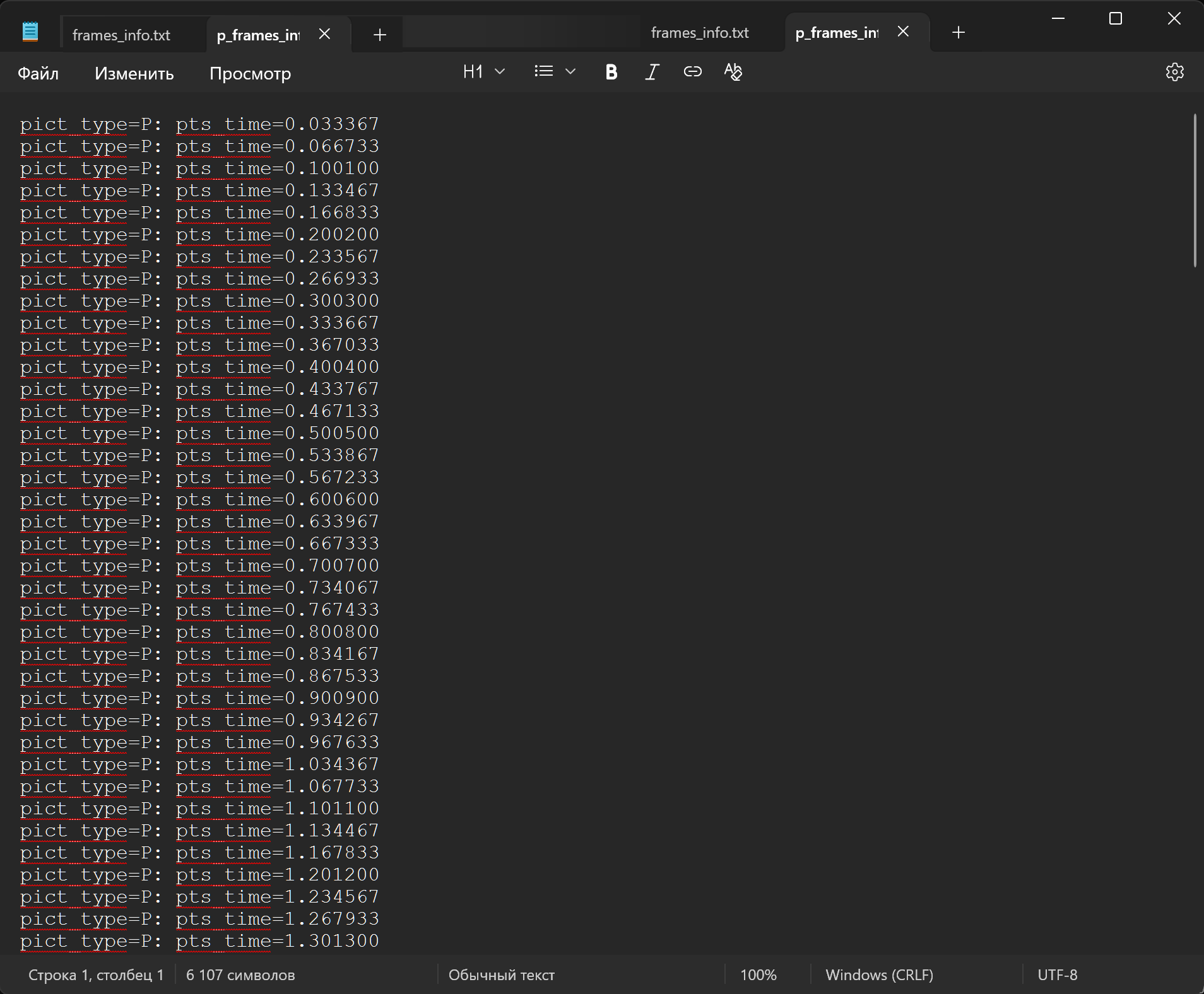


Рисунок 6 – Временные метки P-кадров (первые 38 кадров)

Используемые средства:

* Те же, что и в п. 3: ffprobe + фильтрация по pict\_type=P.

Назначение:

Аналогично I-кадрам, но для P-кадров.

Принцип работы:

* В том же выводе ffprobe ищутся строки с pict\_type=P.
* Сохраняются соответствующие временные метки.

**Максимальное значение межкадровой разности**

С помощью Python и библиотек opencv, numpy, os были найдены и сохранены 2 кадра с максимальным значением межкадровой разности, а также представлена их межкадровая разность.



Рисунок 7.1 – 166 кадр



Рисунок 7.2 – 167 кадр

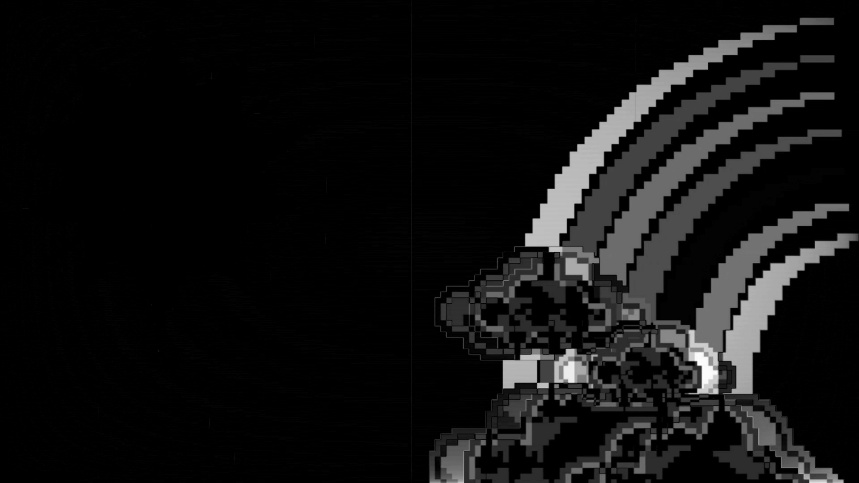


Рисунок 7.3 – Межкадровая разность

Межкадровая разность в 11,9 (условных единиц яркости).

Используемые функции из OpenCV (cv2):

* cv2.VideoCapture(video\_path) — чтение видео по кадрам.
* cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) — преобразование в оттенки серого.
* cv2.absdiff(img1, img2) — поэлементная абсолютная разность двух изображений.
* np.mean(diff) или np.sum(diff) — усреднение или суммирование разности для получения скалярной меры.

Назначение:

Оценка степени изменения сцены между соседними кадрами.

Принцип работы:

* Видео читается кадр за кадром.
* Каждый кадр преобразуется в градации серого.
* Для каждой пары соседних кадров вычисляется матрица разности пикселей.
* Из этой матрицы вычисляется скаляр — среднее/сумма значений.
* Отслеживается максимальное значение и сохраняются соответствующие кадры.

Опционально: визуализация разности через cv2.normalize() и cv2.imwrite().

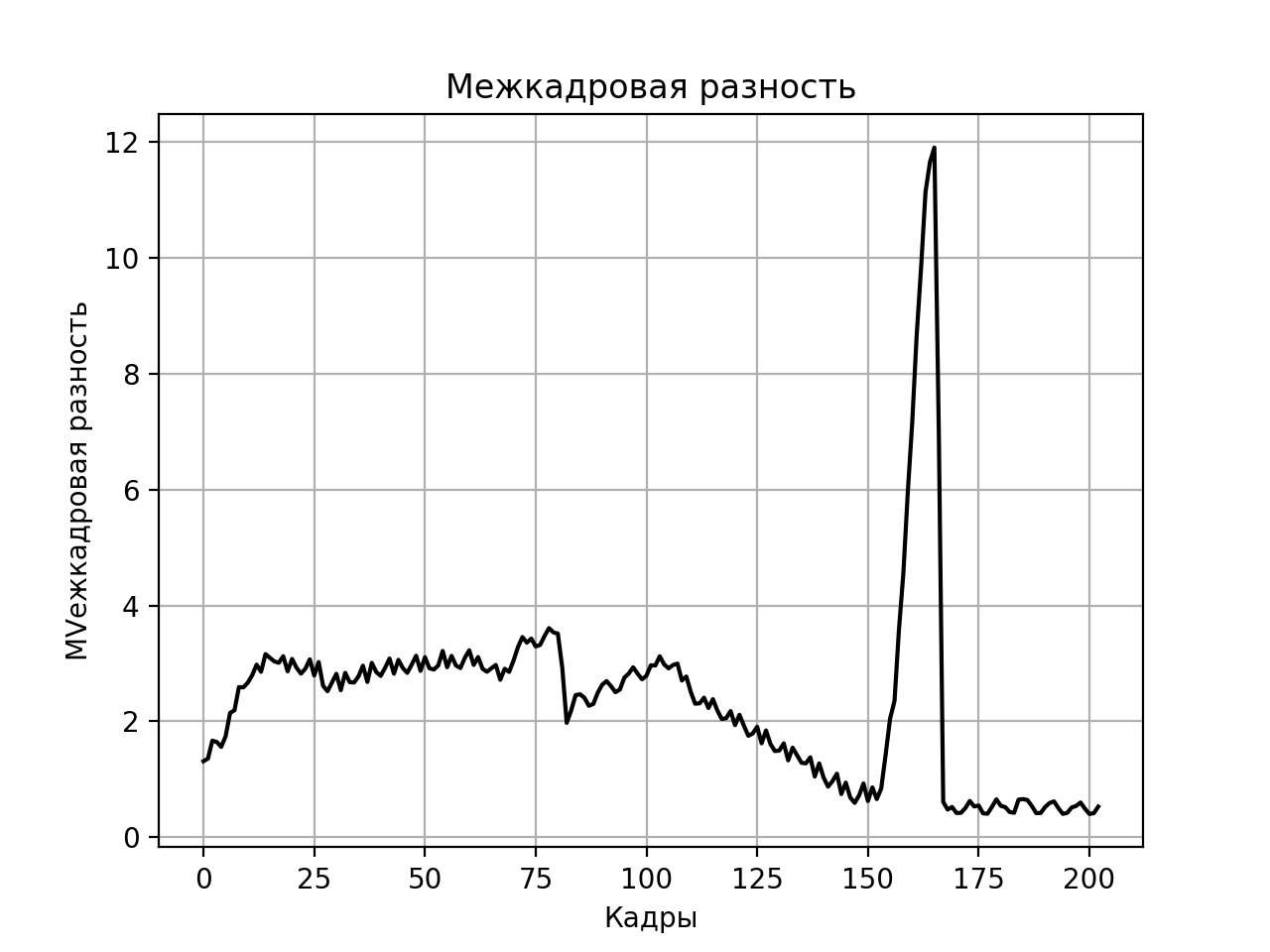


Рисунок 7.4 – График изменения межкадровой разности

**Векторы движения**

С помощью команды –

*ffmpeg -flags2 +export\_mvs -i "C:\Users\etoki\Documents\SUAI\_09.03.02\5\_semester\ТАСИ\8 лр\Футаж\_PIXEL\_light\_and\_sun.mp4" -vf "codecview=mv=pf+bf+bb" -c:v libx264 -crf 18 -preset veryfast "C:\Users\etoki\Documents\SUAI\_09.03.02\5\_semester\ТАСИ\8 лр\new\_video.mp4"*

– было сделано видео, демонстрирующее перемещение элементов в видео, векторы движения.

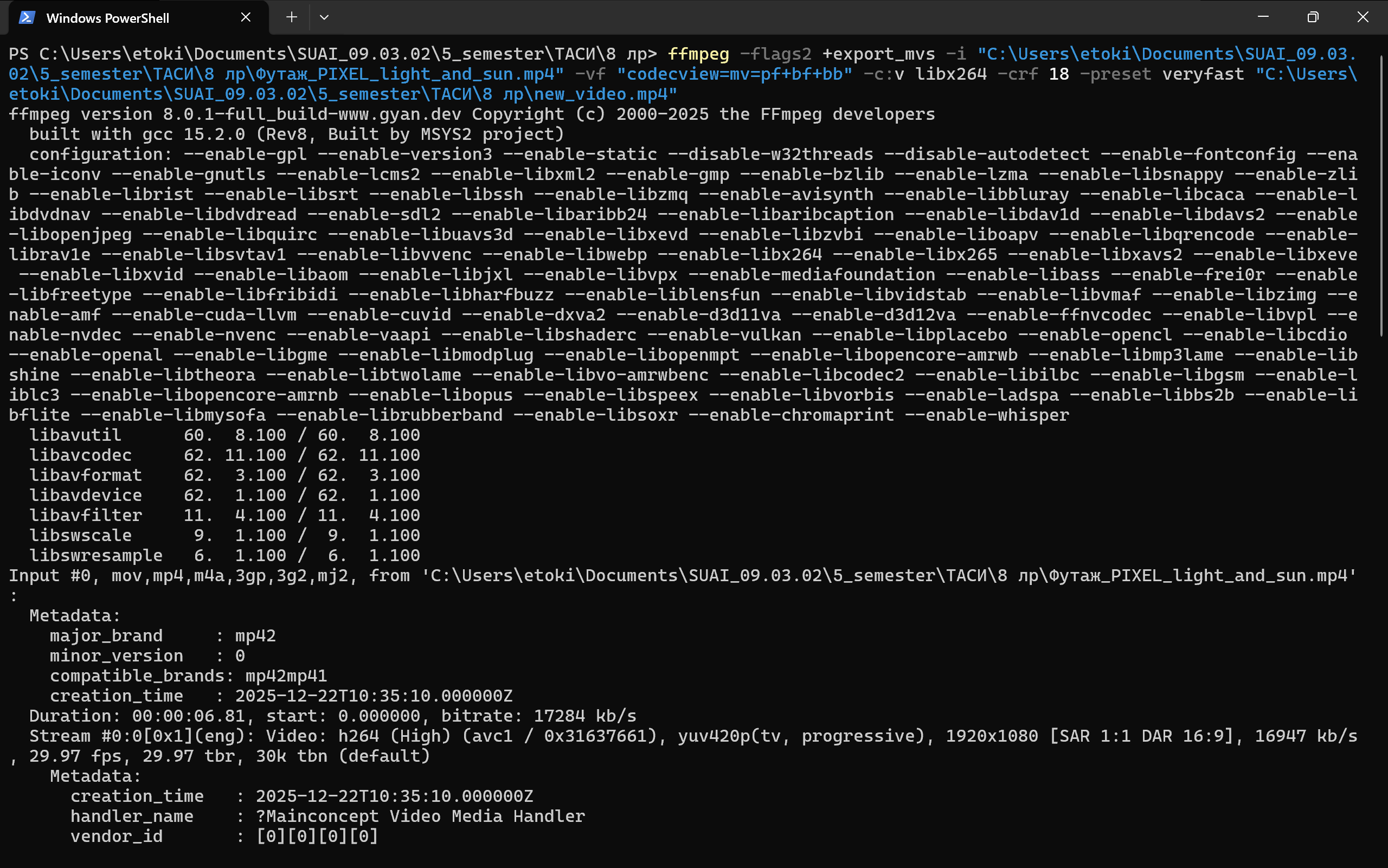


Рисунок 8.1.1 – Выполнение команды

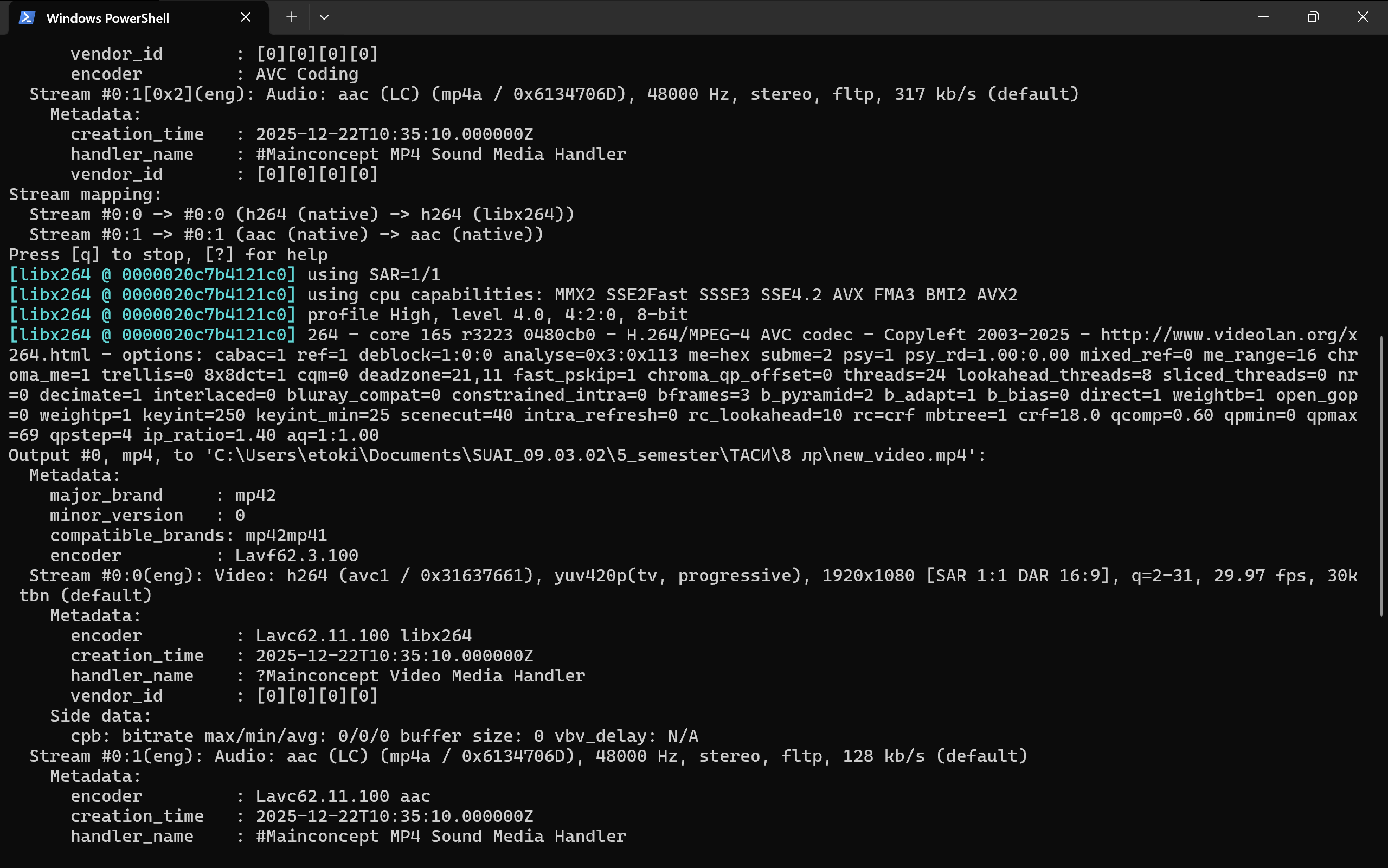


Рисунок 8.1.2 – Выполнение команды

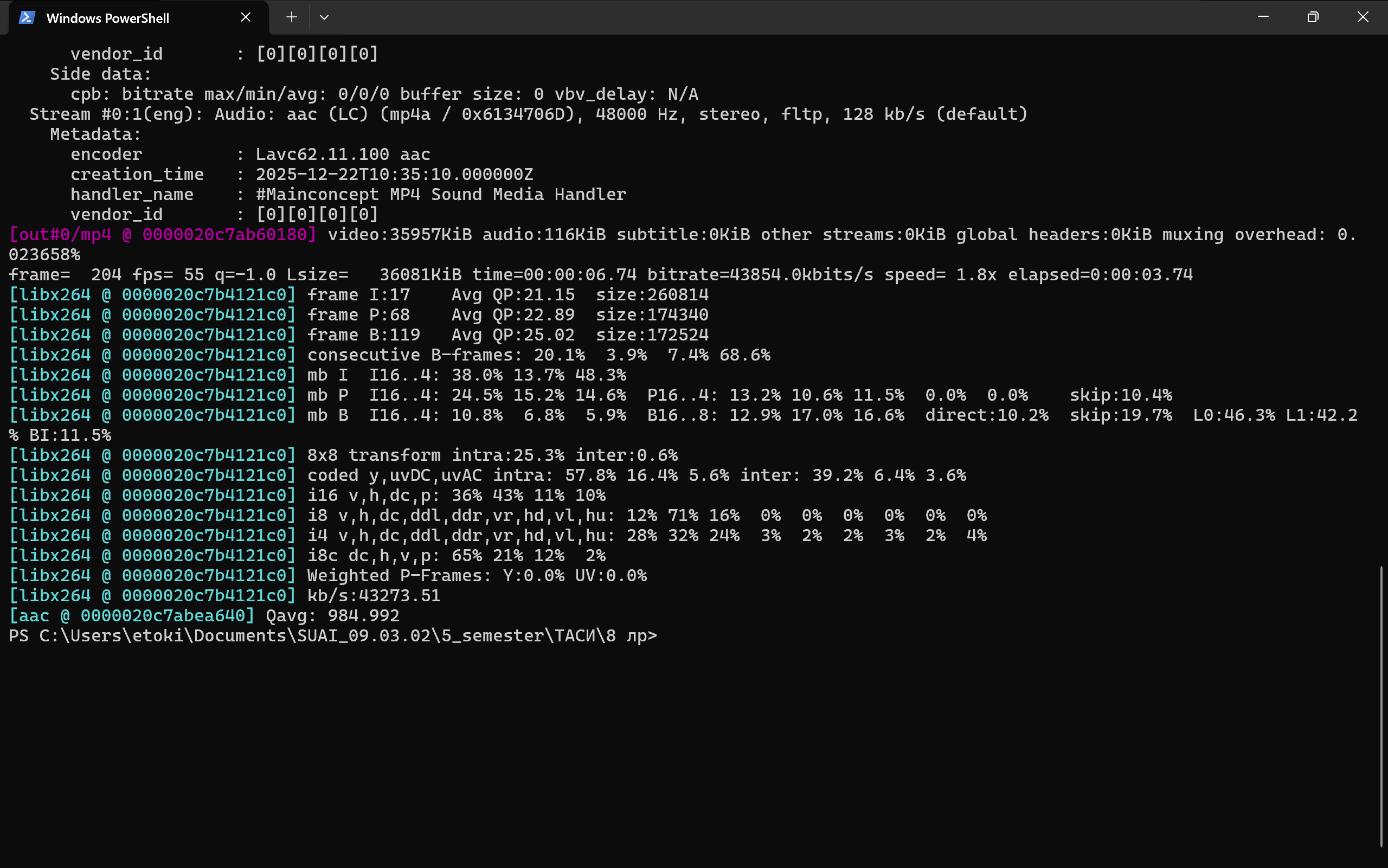


Рисунок 8.1.3 – Выполнение команды

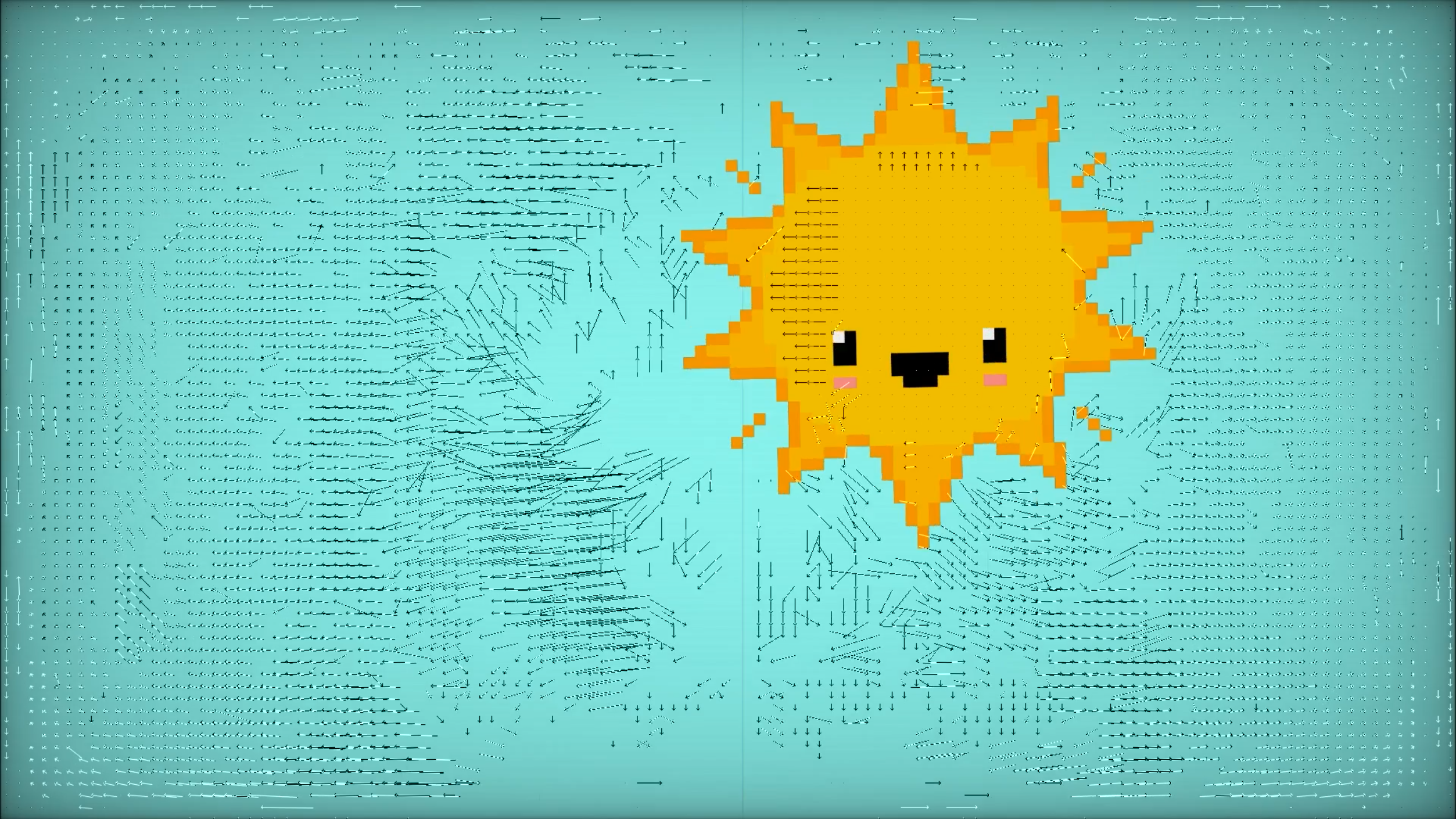


Рисунок 8.2 – Кадр из полученного видео

Назначение используемой команды:

Визуализация векторов движения, используемых кодеком при межкадровом сжатии.

Принцип работы:

* -flags2 +export\_mvs — заставляет декодер экспортировать векторы движения (motion vectors).
* Фильтр codecview=mv=pf+bf+bb отображает:
  + pf — векторы из P-кадров (предсказание от предыдущего кадра),
  + bf — векторы из B-кадров (вперёд),
  + bb — векторы из B-кадров (назад).
* На выходе — видео, где поверх кадров нарисованы стрелки, показывающие направление и величину движения участков сцены.

Команда требует, чтобы исходное видео было закодировано с motion vectors (например, H.264/H.265).

# **ВЫВОД**

В ходе выполнения лабораторной работы были получены теоретические и практические знания о структуре современного видеофайла, принципах межкадрового сжатия и типах кадров (I-, P- и B-кадры). Было продемонстрировано, что видеопоток строится на чередовании опорных (I) и предсказанных (P, B) кадров, что позволяет значительно сократить объём данных без существенной потери визуального качества.

Практическая часть работы позволила:

* извлечь и проанализировать метаданные видеофайла (формат, длительность, битрейт, разрешение, частота кадров);
* сохранить первые 100 кадров в формате JPG для визуального анализа;
* выделить I- и P-кадры как с помощью утилиты ffmpeg, так и путём программной обработки данных, полученных через ffprobe;
* рассчитать межкадровую разность и определить кадры с максимальным изменением содержимого, что подтверждает наличие динамических сцен в видео;
* визуализировать векторы движения в отдельном видео с использованием фильтра codecview, что наглядно демонстрирует направление и интенсивность перемещения объектов между кадрами.

Все задачи реализованы с применением библиотек ffmpeg-python, OpenCV, NumPy и matplotlib, что подтверждает их эффективность для анализа и обработки видеоданных. Работа показала, как теоретические аспекты видеокодеков (например, предсказание движения, опорные кадры) находят своё отражение в реальных инструментах обработки видео.

Таким образом, цель лабораторной работы — освоить структуру видеофайла и методы его анализа на основе типов кадров и межкадровых характеристик — была успешно достигнута.

# **ПРИЛОЖЕНИЕ А**

**Листинг Программы**

**Считывание заголовка:**

import ffmpeg  
  
  
def get\_mp4\_info(path: str) -> None:  
 try:  
 probe = ffmpeg.probe(path)  
  
 # Достаём данные видео-потока  
 video\_streams = [s for s in probe["streams"] if s["codec\_type"] == "video"]  
 audio\_streams = [s for s in probe["streams"] if s["codec\_type"] == "audio"]  
  
 print("=== Основная информация о файле ===")  
 print("Формат:", probe["format"]["format\_name"])  
 print("Длительность (сек):", probe["format"]["duration"])  
 print("Размер файла (байт):", probe["format"]["size"])  
 print("Битрейт:", probe["format"]["bit\_rate"])  
  
 if video\_streams:  
 v = video\_streams[0]  
 print("\n=== Видео ===")  
 print("Кодек:", v.get("codec\_name"))  
 print("Разрешение:", f"{v.get('width')}x{v.get('height')}")  
 print("FPS:", v.get("avg\_frame\_rate"))  
 print("Битрейт:", v.get("bit\_rate"))  
  
 if audio\_streams:  
 a = audio\_streams[0]  
 print("\n=== Аудио ===")  
 print("Кодек:", a.get("codec\_name"))  
 print("Частота дискретизации:", a.get("sample\_rate"))  
 print("Каналы:", a.get("channels"))  
  
 except ffmpeg.Error as e:  
 print("Ошибка FFmpeg:", e)  
  
  
# Пример вызова  
video\_path = r"C:\Users\etoki\Documents\SUAI\_09.03.02\5\_semester\ТАСИ\8 лр\Футаж\_PIXEL\_light\_and\_sun.mp4"  
print()  
get\_mp4\_info(video\_path)

**Сохранение первых 100 кадров видео:**

def extract\_100\_frames(video\_path: str, output\_dir: str):  
 os.makedirs(output\_dir, exist\_ok=True)  
 output\_pattern = os.path.join(output\_dir, "frame\_%03d.jpg")  
 try:  
 (  
 ffmpeg  
 .input(video\_path)  
 .output(output\_pattern, vframes=100, loglevel="quiet")  
 .overwrite\_output()  
 .run(capture\_stdout=True, capture\_stderr=True)  
 )  
 print(f"Сохранено до 100 кадров в: {output\_dir}")  
 except ffmpeg.Error as e:  
 print("Ошибка при извлечении кадров:", e.stderr.decode())  
 raise

**Нахождение временных меток I-кадров и P-кадров:**

def write\_frames\_info(all\_frames\_path: str, i\_frames\_info\_path: str, p\_frames\_info\_path: str) -> None:  
 with (  
 open(all\_frames\_path, "r", encoding="utf") as all\_frames\_file,  
 open(i\_frames\_info\_path, "w", encoding="utf8") as i\_file,  
 open(p\_frames\_info\_path, "w", encoding="utf8") as p\_file,  
 ):  
 all\_frames\_file.read(1)  
  
 # 5 - Временная метка в секундах  
 # 22 - тип кадра  
  
 for line in all\_frames\_file:  
 line\_info = line.split("|")  
 match line\_info[22]:  
 case "pict\_type=I":  
 print(f"{line\_info[22]}: {line\_info[5]}", file=i\_file)  
 case "pict\_type=P":  
 print(f"{line\_info[22]}: {line\_info[5]}", file=p\_file)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 all\_frames\_path = r"C:\Users\etoki\Documents\SUAI\_09.03.02\5\_semester\ТАСИ\8 лр\data1\frames\_info.txt"  
 i\_frames\_info\_path = r"C:\Users\etoki\Documents\SUAI\_09.03.02\5\_semester\ТАСИ\8 лр\data1\i\_frames\_info.txt"  
 p\_frames\_info\_path = r"C:\Users\etoki\Documents\SUAI\_09.03.02\5\_semester\ТАСИ\8 лр\data1\p\_frames\_info.txt"  
  
 write\_frames\_info(all\_frames\_path, i\_frames\_info\_path, p\_frames\_info\_path)

**Нахождение максимальной межкадровой разницы:**

import cv2  
import numpy as np  
import os  
  
  
def extract\_max\_diff\_frames(video\_path, output\_dir="output\_frames"):  
 os.makedirs(output\_dir, exist\_ok=True)  
  
 cap = cv2.VideoCapture(video\_path)  
 if not cap.isOpened():  
 raise IOError(f"Не удалось открыть видео: {video\_path}")  
  
 prev\_frame\_gray = None  
 max\_diff\_value = 0  
 max\_diff\_frames = (None, None)  
 frame\_index = 0  
 best\_index = -1  
  
 while True:  
 ret, frame = cap.read()  
 if not ret:  
 break  
  
 gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
  
 if prev\_frame\_gray is not None:  
 diff = cv2.absdiff(gray, prev\_frame\_gray)  
 diff\_value = np.sum(diff)  
  
 if diff\_value > max\_diff\_value:  
 max\_diff\_value = diff\_value  
 max\_diff\_frames = (prev\_frame.copy(), frame.copy())  
 best\_index = frame\_index  
  
 prev\_frame\_gray = gray  
 prev\_frame = frame  
 frame\_index += 1  
  
 cap.release()  
  
 if max\_diff\_frames[0] is None:  
 raise RuntimeError("Недостаточно кадров для анализа")  
  
 frame1\_path = os.path.join(output\_dir, "frame\_max\_diff\_1.jpg")  
 frame2\_path = os.path.join(output\_dir, "frame\_max\_diff\_2.jpg")  
  
 cv2.imwrite(frame1\_path, max\_diff\_frames[0])  
 cv2.imwrite(frame2\_path, max\_diff\_frames[1])  
  
 diff\_image = cv2.absdiff(  
 cv2.cvtColor(max\_diff\_frames[0], cv2.COLOR\_BGR2GRAY), cv2.cvtColor(max\_diff\_frames[1], cv2.COLOR\_BGR2GRAY)  
 )  
  
 # Нормализация для лучшей визуализации  
 diff\_image\_norm = cv2.normalize(diff\_image, None, 0, 255, cv2.NORM\_MINMAX)  
  
 diff\_path = os.path.join(output\_dir, "frame\_difference.jpg")  
 cv2.imwrite(diff\_path, diff\_image\_norm)  
  
 return {  
 "frame1": frame1\_path,  
 "frame2": frame2\_path,  
 "difference\_image": diff\_path,  
 "difference\_value": int(max\_diff\_value),  
 "second\_frame\_index": best\_index,  
 }  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 video\_file = r"C:\Users\etoki\Documents\SUAI\_09.03.02\5\_semester\ТАСИ\8 лр\Футаж\_PIXEL\_light\_and\_sun.mp4" # путь к mp4 файлу  
 result = extract\_max\_diff\_frames(video\_file)  
 print("Сохранены кадры с максимальной разностью:")  
 for k, v in result.items():  
 print(f"{k}: {v}")

**Визуализация межкадровой разницы:**

import cv2  
  
  
def write\_frame\_diff(video\_path: str, output\_path: str) -> None:  
 cap = cv2.VideoCapture(video\_path)  
  
 if not cap.isOpened():  
 raise Exception("Не удалось открыть видео: " + video\_path)  
  
 ret, prev\_frame = cap.read()  
 if not ret:  
 raise Exception("Не удалось прочитать первый кадр в видео.")  
  
 # Переводим в grayscale  
 prev\_gray = cv2.cvtColor(prev\_frame, cv2.COLOR\_BGR2GRAY)  
  
 diff\_values = []  
  
 while True:  
 ret, frame = cap.read()  
 if not ret:  
 break  
  
 gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
  
 # абсолютная разница между кадрами  
 diff = cv2.absdiff(gray, prev\_gray)  
  
 # средняя разница кадра (одно число)  
 mean\_diff = diff.mean()  
  
 diff\_values.append(mean\_diff)  
  
 prev\_gray = gray  
  
 cap.release()  
  
 # запись результата  
 with open(output\_path, "w", encoding="utf-8") as f:  
 for value in diff\_values:  
 f.write(f"{value}\n")  
  
 print(f"Готово! Межкадровая разница записана в:\n{output\_path}")  
  
  
def show\_graph():  
 import matplotlib.pyplot as plt  
  
 with open(  
 r"C:\Users\etoki\Documents\SUAI\_09.03.02\5\_semester\ТАСИ\8 лр\data1\frame\_diff.txt", "r", encoding="utf8"  
 ) as txt\_file:  
 data = list(map(float, txt\_file))  
 t = range(len(data))  
  
 plt.plot(t, data, color="black")  
 plt.title("Межкадровая разность")  
 plt.xlabel("Кадры")  
 plt.ylabel("МVежкадровая разность")  
 plt.grid()  
 plt.show()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 video\_path = r"C:\Users\etoki\Documents\SUAI\_09.03.02\5\_semester\ТАСИ\8 лр\Футаж\_PIXEL\_light\_and\_sun.mp4"  
 output\_path = r"C:\Users\etoki\Documents\SUAI\_09.03.02\5\_semester\ТАСИ\8 лр\data1\frame\_diff.txt"  
  
 # print()  
 # with open(output\_path, "r", encoding="utf8") as txt\_file:  
 # print(f"Максимальное значение межкадровой разности: {max(map(float, txt\_file))}")  
  
 show\_graph()