Markdown идеально подходит для быстрых заметок, технических спецификаций и простых инструкций. Однако отсутствие встроенной поддержки сложных макетов и недостаточная автоматизация индексации ограничивают применение Markdown для масштабных проектных документов.

Typst представляет собой новое поколение инструментов, сочетающих удобство Markdown с возможностями LaTeX. Недостатком на сегодняшний день является относительная новизна продукта и меньшая популярность среди научного сообщества по сравнению с традиционными решениями вроде LaTeX.

Таким образом, выбор конкретного инструмента зависит от характера проекта, уровня требуемого контроля над оформлением и потребностей команды разработчиков. Для специализированных задач с большим объемом математики и статистики предпочтительным решением остается LaTeX. Если же речь идет о создании обычных документов общего назначения, оптимальным выбором станет Microsoft Word. Когда важна простота реализации и скорость прототипирования, целесообразно применять Markdown. Наконец, в ситуациях, когда важны простота и функциональность одновременно, перспективным вариантом представляется Typst.

УДК 004.059

Е.В. Обухова, маг.; Н.П. Шутько, доц., канд. техн. наук
(БГТУ, г. Минск)

ОЦЕНКА ЭФФЕКТИВНОСТИ ТРАНСКОДИРОВАНИЯ ВИДЕО НА БАЗЕ MOVIEPY И FFMPEG

Выбор оптимального программного решения является определяющим фактором для достижения требуемых характеристик видео при его преобразовании. Целью данной работы является сравнительная оценка производительности, ресурсоемкости качества видеопотока, формируемого в процессе транскодирования с использованием разработанных на языке Python программных модулей: на основе библиотеки MoviePy (далее – модуль А) и на базе фреймворка FFmpeg (далее – модуль Б). Анализ и сопоставление полученных данных позволят определить наиболее эффективную реализацию для включения в состав плагина транскодирования видеороликов в формат Theora для игрового движка Ren’Py.

Рассмотрим техническую часть реализованных модулей и сравним их по следующим аспектам: организация конвейера обработки,

управление выделяемой оперативной памятью и гибкость настройки параметров кодирования. Важно отметить, что для реализации модулей использовались библиотека MoviePy версии 2.2.1 и фреймворк FFmpeg версии 7.1.1

Первым пунктом сравнения является организация конвейера обработки получаемых данных. Модуль А манипулирует видеопотоком, представляющим собой коллекцию независимых кадров [1]. Для сборки видеопоследовательности используется встроенный класс ImageSequenceClip, который инкапсулирует логику управления кадрами (листинг 1).

```
def convert(video_frames: list[np.ndarray], ...):
video_clip = ImageSequenceClip(video_frames, fps=fps)
```

Листинг 1 – Обработка данных модулем А

В модуле Б полученные кадры передаются последовательно, исключая необходимость в предварительной загрузке всего видеопотока в оперативную память (листинг 2).

```
for frame in frame_generator:
clean_data = np.ascontiguousarray(frame[:, :, :3],
dtype=np.uint8).tobytes()
process.stdin.write(clean_data)
```

Листинг 2 – Обработка данных модулем Б

Следующим критерием анализа выступает управление выделяемой оперативной памятью. В модуле А библиотека MoviePy берет управление временными файлами на себя, однако, до начала кодирования передаваемая последовательность несжатых кадров должна быть помещена в структуру list[np.ndarray]. Вследствие этого объем потребляемой памяти растет линейно и зависит от общего количества полученных кадров и их пространственного разрешения. Подобный рост вычислительной сложности создает повышенную нагрузку на аппаратные ресурсы. Напротив, при использовании модуля Б в памяти в каждый момент времени удерживается только один несжатый видеокадр, что снижает нагрузку на ОЗУ.

Настройка параметров кодирования в модуле А осуществляется через метод write_videofile(), который предоставляет разработчику доступ только к базовым параметрам, таким как битрейт, частота кадров, тип кодека и иными параметрами (листинг 3). MoviePy автоматизирует вычисление иных настроек, что минимизирует вероятность возникновения ошибок конфигурации.

```
vid-
eo_clip.write_videofile(output_filepath,codec="libtheora",a
```

```
udio_codec="libvorbis", fps=fps, bitrate="8000k", au-
dio_bitrate="320k")
```

Листинг 3 – Настройка параметров кодирования в модуле А

В модуле Б формирование цепочки фильтров происходит вручную, что обеспечивает точный контроль над масштабированием, геометрией кадра и иными параметрами трансформации видеопотока. Управление параметрами в модуле Б представлено в листинге 4.

```
ffmpeg_cmd.extend(['-c:v', 'libtheora','-q:v','8','-
g','1','-vf','scale=1920:1080:force_original_aspect_ratio
=decrease,pad=1920:1080:(ow-iw)/2:(oh-ih)/2,format=yuv420p'
```

Листинг 4 – Настройка параметров кодирования в модуле Б

Для оценки эффективности разработанных модулей была проведена серия экспериментов по транскодированию набора данных, состоящего из 17 видеороликов с различными характеристиками: пространственным разрешением, размером файла, частотой кадров и типом контента (натурная съемка и 2D-анимация). В таблице приведен фрагмент полученных данных. Далее, на их основе был проведен сравнительный анализ показателей обоих модулей. Для графической интерпретации результатов использовалась библиотека Matplotlib (рис. 1).

Таблица – Фрагмент сводных данных транскодирования тестовых видеороликов

Файл	Метод	Время кодирования (сек)	Исходный размер (МБ)	Новый размер (МБ)	CPU (%)	RAM (МБ)	Битрейт (кбит/с)	FPS обработки
1387.mp4	FFmpeg	18.39	54.38	88.5	102.1	65.2	51733	18.27
	MoviePy	122.97	54.38	15.29	20.1	6223.1	8935	2.73
веска.mp4	FFmpeg	19.24	16.19	21.41	79.5	39.4	19490	11.69
	MoviePy	86.58	16.19	9.06	14.1	5549.9	8249	2.6

Анализ результатов показал, что общее время транскодирования модулем Б составило 340,64 сек. (25 FPS) – это быстрее в 4,4 раза, чем с использованием модуля А (1502,79 сек., 6.56 FPS). Низкая скорость последнего обусловлена использованием библиотеки MoviePy, которая вызывает фреймворк FFmpeg в качестве внешнего подпроцесса для обработки данных, что существенно увеличивает итоговое время. В вопросе потребления ресурсов модуль Б также эффективнее: в среднем 110,4 МБ против 5633,4 МБ у модуля А. Такая разница вызвана тем, что метод ImageSequenceClip в модуле А загружает массив кадров в ОЗУ целиком, тогда как модуль Б работает в поточном режиме, удерживая в памяти только один текущий кадр. Загрузка процессора модулем Б выше (68%), чем при использовании модуля А

(13,5%). Это объясняется способностью FFmpeg распределять задачи кодирования на все ядра устройства. Модуль А же ограничен однопоточностью, что препятствует полной загрузке вычислительных мощностей.

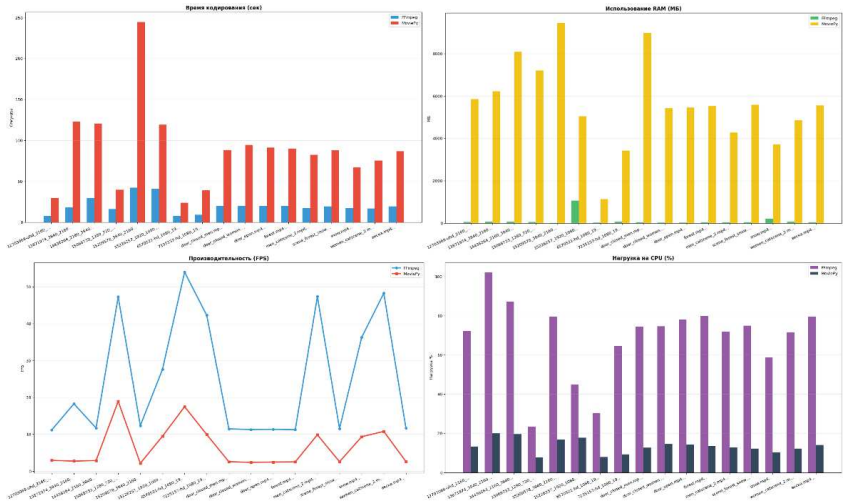

Рисунок 1 – Графическая интерпретация результатов

Дополнительно был проведен визуальный анализ видео, получившихся в ходе транскодирования. Для этого выбраны образцы натурной видеосъемки и 2D-анимации, характеризующиеся высоким качеством и отсутствием артефактов. На рисунке 2 приведены видеокадры после транскодирования модулем А (слева) и модулем Б (справа).

При анализе результатов транскодирования модулем А можно заметить явные изменения в кадрах видео, а именно: появление ступенчатых градиентов, наличие цифрового шума в тенях, общее искажение цветопередачи. В 2D-анимации с высокой долей статических кадров дефекты менее выражены, однако на светлых участках прослеживаются ступенчатые переходы между оттенками и черные пиксели. Видеопоток, полученный с помощью модуля Б, не содержит видимых артефактов, за исключением незначительного сдвига цветового тона.

Рисунок 2 – Кадры полученных видео

Таким образом, для получения удовлетворительного результата при транскодировании видео, следует использовать модуль Б. Он позволяет точно управлять параметрами, обеспечивает высокую скорость обработки и сохранение визуального качества. Однако при использовании данного модуля возможен рост объема выходного файла (до 5 раз). Модуль А же целесообразно применять исключительно для контента с большим количеством статичных кадров, где он демонстрирует четырехкратное сжатие при сохранении приемлемого качества.

ЛИТЕРАТУРА

1. Обухова, Е. В., Н. П. Шутько Разработка модуля конвертации видеороликов на основе библиотеки Moviepy и фреймворка ffmpeg // Передовые технологии и инновации в образовании и науке для улучшения качества жизни и стимулирования устойчивого экономического роста: Сборник статей VIII Международной научно-технической конференции. В 3-х томах, Минск, 03–05 декабря 2025 года. – Минск: БГТУ, 2025. – С. 454-460.

УДК 004.932.2

В.А. Ворошень, маг.;
Д.М. Романенко, зав. кафедрой ИиВД
(БГТУ, г. Минск)

ОПРЕДЕЛЕНИЕ АКЦЕНТНЫХ ЦВЕТОВ НА РАСТРОВЫХ ИЗОБРАЖЕНИЯХ С ИСПОЛЬЗОВАНИЕМ АЛГОРИТМОВ КЛАСТЕРИЗАЦИИ

Одним из композиционных средств, используемых при построении кадра, является акцент. Он представляет собой некоторую цветовую область на изображении, которая существенно отличается от прочих по цветовых характеристикам (например, по тону, яркости или насыщенности). В ряде задач компьютерного зрения может быть полезной автоматическая сегментация акцентных областей. Среди подобных наиболее интересной является задача анализа композиции растровых изображений с художественной точки зрения.

Какой же цвет считается акцентным? В рамках данного исследование такой цвет должен удовлетворять следующим признакам:

– далёк от медианного цвета, признанного фоновым;

– имеет достаточно крупную площадь, чтобы быть заметным (не менее 0,5% всей площади изображения);