Математические структуры и моделирование
2015. № 3(35). С. 81–90

УДК 004.93'11

ОБОСНОВАНИЕ СТРУКТУРЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ «УМНОЙ» КАМЕРЫ ВИДЕОНАБЛЮДЕНИЯ

П.В. Прохоров
к.т.н., e-mail: pavelvpster@gmail.com

Омский государственный университет им. Ф.М. Достоевского

Аннотация. Возможности современных микрокомпьютеров в сочетании с веб-камерами и доступными облачными хранилищами файлов позволяют создать эффективную систему видеонаблюдения с расширенными функциональными возможностями, в том числе, обнаружением движения и лиц и даже идентификацией пользователей.

Ключевые слова: видеонаблюдение, видеокамера, обнаружение движения, сжатие.

Введение

Традиционные (аналоговые) системы видеонаблюдения обеспечивали лишь две функции: отображение видео пользователю в реальном масштабе времени, возможно, с нескольких камер и запись видео также, возможно, с нескольких камер. Профессиональные системы использовали цифровые форматы записи и автоматизацию ротации носителей. Традиционные системы обладают высокой чувствительностью в темноте, однако, их разрешение не соответствует современным требованиям, а картинка, как правило, черно-белая.

Широкое распространение у обычных пользователей получили автомобильные видеорегистраторы. Эти устройства предложили новый принцип записи: циклическая запись фрагментами (chunk). Записывается фрагмент фиксированного размера (размер может выбираться в настройках). По его завершении начинается запись нового фрагмента. Количество фрагментов может задаваться в настройках или определяться размером карты памяти. По мере необходимости самые старые фрагменты удаляются.

С технической точки зрения запись фрагментами обусловлена особенностями большинства файловых систем: при аварийном завершении работы не закрытые файлы могут не сохраниться или оказаться повреждёнными. Под аварийным завершением также понимается неожиданное выключение питания устройства. В последние годы начали появляться «умные» камеры. Они отличаются расширенным набором функций. Наиболее распространено обнаружение движения. Вероятно, в ближайшее время стоит ожидать реализацию обнаружения лиц и идентификации пользователей. Рассмотрим возможные функции «умной» камеры.

1. Функции «умной» камеры

1. Циклическая запись фрагментами видео или отдельными кадрами. Последний вариант обеспечивает дополнительную эффективность, поскольку позволяет использовать сжатие в формате JPEG на стороне камеры. Такую возможность обеспечивает большинство веб-камер. Отдельные модели обеспечивают сжатие видео в формате MPEG, но пока это редкость.

2. Сохранение записанной информации на локальном носителе информации (карте памяти) и «в облаке» (NFS, FTP, WebDAV).

3. Обнаружение движения с формированием интегрального индекса движения (0-100).

4. Прореживание кадров (децимация, decimation) в зависимости от интегрального индекса движения.

5. Обнаружение лиц (face detection). Сигнализация системе следующего уровня об обнаружении лиц, передача изображения лица, метаданных, например, для идентификации пользователей.

6. Доступ к изображению и настройкам камеры через веб-интерфейс.

7. Дистанционное управление через RPC, например, ZeroC Ice.

Рассмотрим варианты реализации перечисленных функций.

2. Варианты реализации

Сразу оговоримся, что мы не рассматриваем готовые решения (программы), а только технологии (библиотеки). Распространение получили два решения для обработки видео: программа VLC Media Player [1] и библиотека GStreamer [2].

Программа VLC Media Player распространяется свободно (open source) и является не только полноценным аудио-видео плеером для множества операционных систем, но и обеспечивает возможность запуска своих «цепочек обработки видео» посредством интерфейса командной строки.

Далее приведен пример командной строки для трансляции видео по протоколу HTTP:

```
cvlc --no-audio v4l2:///dev/video0 :width=640 :height=480
:live-caching='300' :sout='#transcode{vcodec=MJPG,vb=300,
width=640,height=480}:std{access=http{mime=multipart/
x-mixed-replace;boundary=--7b3cc56e5f51db803f790dad720ed50a}
,mux=mpjpeg,dst=:64200/webcam}'
```

К недостаткам следует отнести повышенные требования к ресурсам процессора. Так эксперименты показали, что на Raspberry Pi загрузка процессора стремится к 100% даже при тривиальной трансляции видео по сети.

Библиотека GStreamer также распространяется свободно и может быть откомпилирована под большинство операционных систем. Более того, она часто используется для реализации работы с мультимедиа в других библиотеках (backend), например, в библиотеке Qt.

Пример трансляции видео по протоколу TCP (для просмотра необходимо также использовать GStreamer):

```
gst-launch v4l2src device=/dev/video0 ! 'image/jpeg,
width=640,height=480,framerate=15/1' ! multipartmux !
tcpserversink port=64300 sync=false
```

Обе рассмотренные технологии обеспечивают получение видео с веб-камер, его простейшие преобразования (размер, яркость, прореживание кадров), сжатие в различные форматы, сохранение и передачу по сети. Функции распознавания придется реализовать самому.

Разумеется, остается возможность разработать собственную программу. Учитывая наработки в области компьютерного зрения (библиотека CV [3]), обработке изображений (библиотека Image [4]) и получения видео (библиотека Video [5]), эта задача представляется не значительно сложнее использования GStreamer. Однако при должном старании можно обеспечить более высокую эффективность использования ресурсов процессора за счет исключения лишних операций.

Рассмотрим эффективную структуру программы функциональных возможностей.

3. Структура программного обеспечения

Эффективной структура программы становится только в конкретных условиях.

4. Вариант 1. Запись кадров с фиксированным прореживанием

Возможные структуры программы с использованием сжатия на стороне вебкамеры (а) и без него (б) показаны на следующем рисунке.

Пунктиром обозначены действия, выполняемые на стороне камеры.

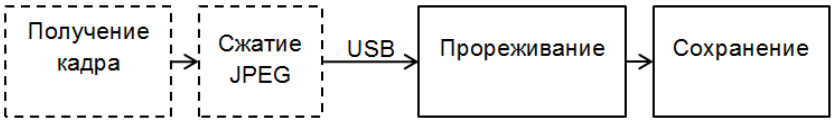

Рис. 1а. С использованием сжатия на стороне камеры

Рис. 1б. Без использования сжатия на стороне камеры

Очевидно, что операция кодирования JPEG в программе лишняя — лучше использовать сжатие на стороне камеры.

5. Вариант 2. Запись кадров с меткой времени с фиксированным прореживанием

Добавить метку времени можно только на несжатое изображение, поэтому использование сжатия на стороне камеры неэффективно, поскольку потребует дополнительную операцию декодирования JPEG.

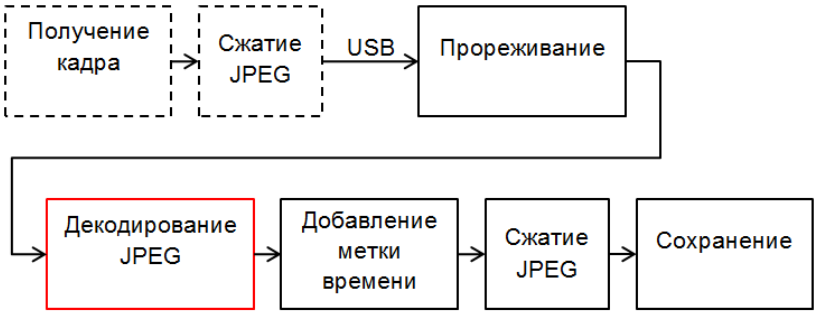

Рис. 2а. С использованием сжатия на стороне камеры

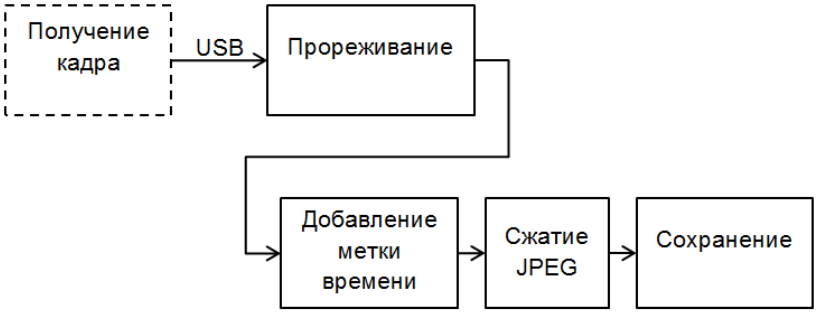

Рис. 2б. Без использования сжатия на стороне камеры

6. Вариант 3. Обнаружение движения и управляемое прореживание

Для обнаружения движения также используется несжатое изображение.

На рисунках K — коэффициент децимации. Он зависит от интегрального индекса движения. Для определения коэффициента используется пороговый детектор и две константы: для случая, когда движение есть, и для его отсутствия.

Алгоритм обнаружения движения реализован в библиотеке CV.

Количество операций на рис. 3а и 3б одинаково.

7. Вариант 4. Обнаружение движения и управляемое прореживание с меткой времени

Операция декодирования JPEG в программе лишняя.

Рис. 3а. С использованием сжатия на стороне камеры

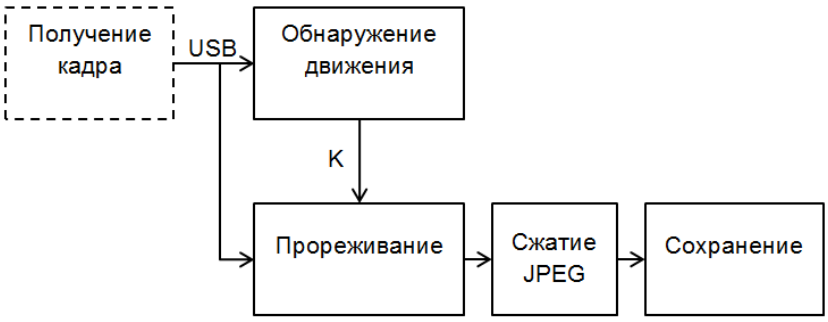

Рис. 3б. Без использования сжатия на стороне камеры

Рис. 4а. С использованием сжатия на стороне камеры

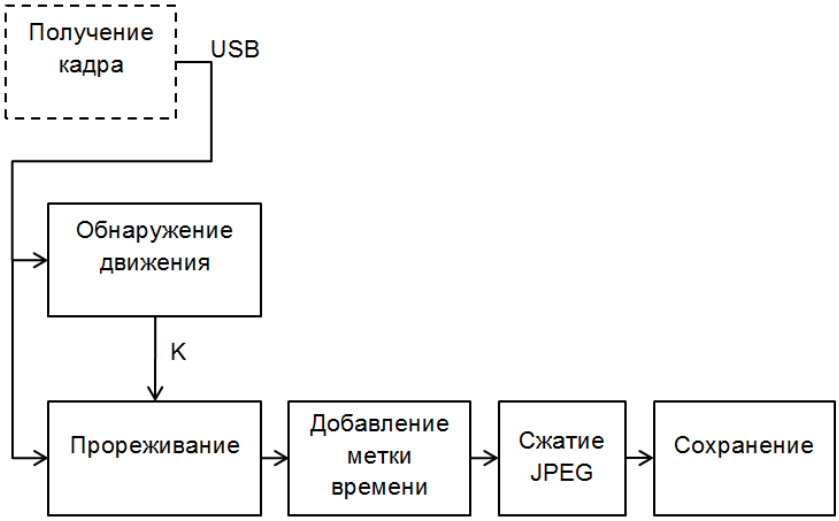

Рис. 4б. Без использования сжатия на стороне камеры

Таблица 1. Выигрыш от исключения лишнего декодирования JPEG

ЭВМ / Реализация	Чтение кадров в JPEG	Чтение кадров без сжатия (Raw RGB)
ПЭВМ, Ubuntu 14.04	29%	13,2%
ЭВМ Raspberry Pi 2 model B	100%	38%

8. Вариант 5. Максимальный набор функций

С использованием библиотеки CV обнаружение лиц может выполняться параллельно с обнаружением движения асинхронно вне реального времени. Предоставление изображения через веб-интерфейс должно осуществляться в масштабе времени, близком к реальному.

Через веб-интерфейс можно транслировать «бесконечный» видеофайл. Однако эффективнее предоставить страницу, которая при помощи сценария будет обновлять изображение. При этом сценарий может оценить время запроса одного изображения и отрегулировать интервал обновления так, чтобы не создавать избыточную нагрузку на сеть.

Количество операций на рис. 5а и 5б одинаково, однако, схема без использования сжатия на стороне камеры более гибкая, поскольку можно изменить формат изображения, предоставляемого через веб-интерфейс, например, на PNG.

Также отметим, что большинство веб-камер позволяют выбрать формат представления цвета несжатых изображений: RGB, IUV. Для обнаружения движения и обнаружения лиц лучше использовать формат IUV, поскольку его канал I представляет собой интенсивность, т.е. исключается преобразование изображения в «оттенки серого» перед обработкой, разумеется, при условии, что такой алгоритм работы поддерживается программой.

9. Вычислительный эксперимент

В качестве вычислительного эксперимента:

1. Оценим выигрыш от исключения лишнего декодирования JPEG.

2. Оценим загрузку процессоров различных ЭВМ вариантами нашей программы с разным набором функций.

3. Сравним загрузку процессоров нашей программой и GStreamer.

Во всех экспериментах будем использовать ПЭВМ с процессором Intel Core i7 4770K под управлением операционной системы Ubuntu 14.04 и микро-ЭВМ Raspberry Pi 2 model B под управлением операционной системы Raspbian.

1. Для оценки выигрыша от исключения лишнего декодирования JPEG рассмотрим загрузку процессоров ЭВМ для варианта 2 (запись кадров с меткой времени с фиксированным прореживанием до 1 кадра в секунду). Загрузка одного ядра процессоров приведена в таблице 1.

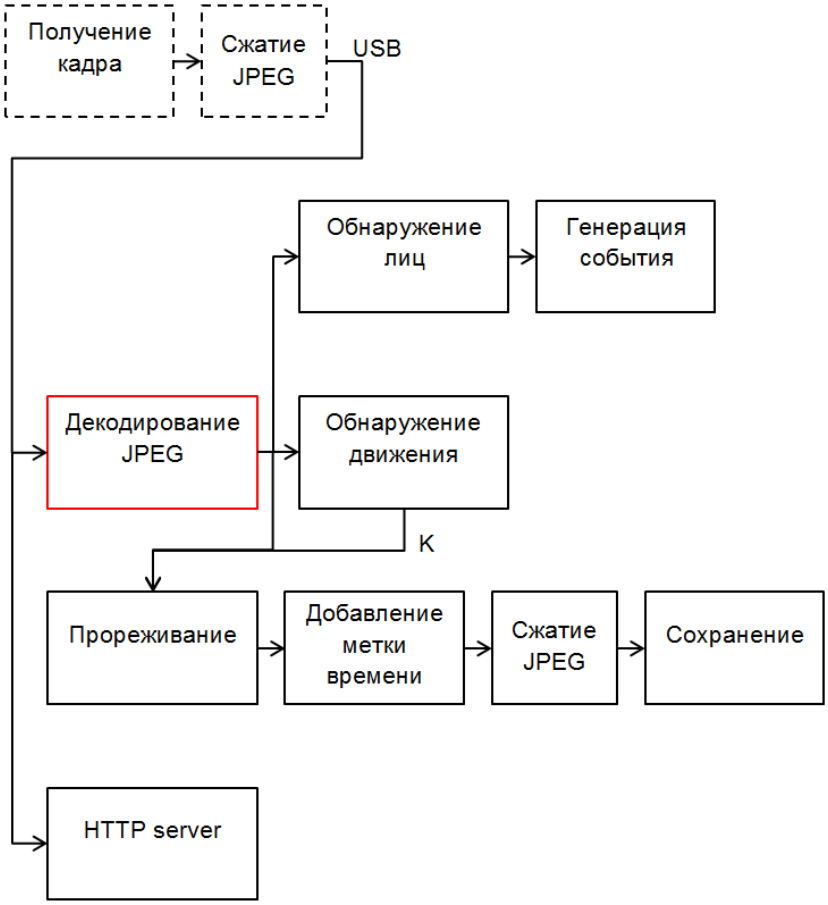

Рис. 5а. С использованием сжатия на стороне камеры

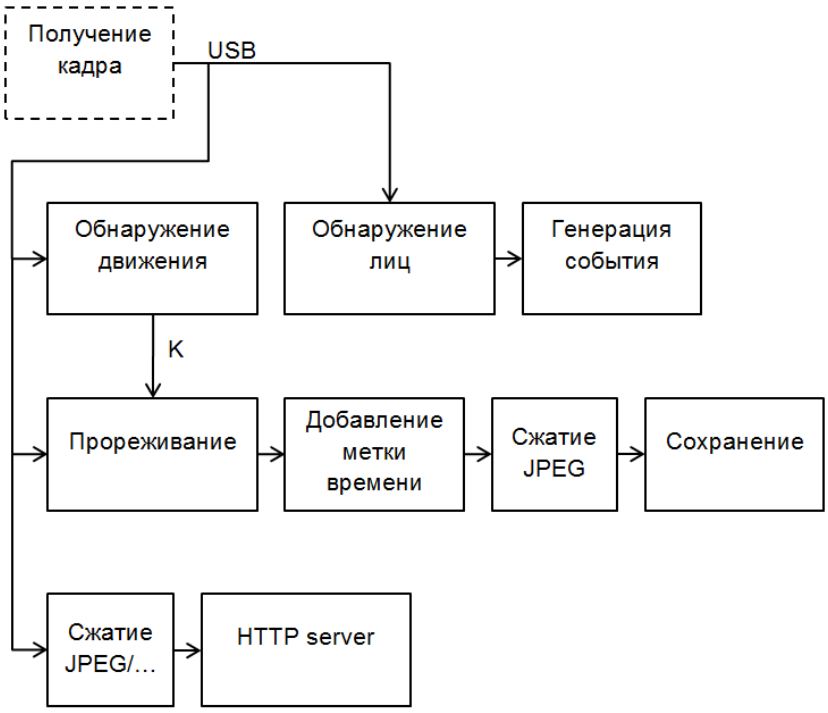

Рис. 5б. Без использования сжатия на стороне камеры

Таблица 2. Загрузка процессора различными вариантами программы

Вариант	ПЭВМ, Ubuntu 14.04		ЭВМ Raspberry Pi 2 model B	
	Загрузка	Время	Загрузка	Время
2	13%	-	38%	-
4	52%	25 мс	165%	430 мс
5	140%		250%	
Движение		22 мс		500 мс
Лица		240 мс		9 с

Значения получены при помощи команды top с усреднением за 20 секунд с фильтрацией по идентификатору процесса PID. Отметим, что с ростом набора функций выигрыш от оптимального чтения кадров будет не столь заметным на фоне загрузки процессора функциями распознавания и т.д.

2. Для оценки загрузки процессора выберем варианты 2, 4 и 5.

Загрузка одного ядра процессора и время, затраченное на обработку одного кадра, приведены в таблице 2 (значения загрузки больше 100% свидетельствуют о загрузке более чем одного ядра).

3. Для сравнения с GStreamer опять будем использовать вариант 2.

Загрузка одного ядра процессора нашей программой составляет 38%. Загрузка одного ядра программой, использующей GStreamer, составляет 8%.

10. Заключение

В статье приводится обоснование оптимальной по критерию количества преобразований изображения структуры программного обеспечения «умной» камеры для различных наборов функций.

В зависимости от дальнейшей обработки эффективным оказывается использование сжатия на стороне камеры или получение несжатого изображения. И та и другая возможности предоставляются большинством веб-камер.

Полученные результаты показывают, что реализация «умной» камеры видеонаблюдения без функции обнаружения лиц на ЭВМ Raspberry Pi 2 model B вполне возможна.

Исходный код разработанной программы размещён в сети Интернет [6]. Исходный код программы по варианту 2, использующей библиотеку GStreamer, также размещён в сети Интернет [7].

Рассмотренные структуры не привязаны к конкретной реализации или библиотеке компьютерного зрения, хотя автором предлагается своя библиотека компьютерного зрения на языке Java.

Литература

1. VLC Media Player. URL: `http://www.videolan.org/vlc/` (дата обращения: 01.09.2015)
2. Библиотека GStreamer. URL: `http://gstreamer.freedesktop.org/` (дата обращения: 01.09.2015)
3. Проект CV. URL: `https://github.com/pavelvpster/CV` (дата обращения: 01.09.2015)
4. Проект Image. URL: `https://github.com/pavelvpster/Image` (дата обращения: 01.09.2015)
5. Проект Video. URL: `https://github.com/pavelvpster/Video` (дата обращения: 01.09.2015)
6. Проект SurveillanceCamera. URL: `https://github.com/pavelvpster/SurveillanceCamera` (дата обращения: 01.09.2015)
7. Проект Surveillance. URL: `https://github.com/pavelvpster/Surveillance` (дата обращения: 01.09.2015)

JUSTIFICATION OF THE STRUCTURE OF THE "SMART" SURVEILLANCE CAMERA'S PROGRAM SOFTWARE

P.V. Prohorov
Ph.D. (Eng.), e-mail: pavelvpster@gmail.com

Omsk State University n.a. F.M. Dostoevskiy

Abstract. The capabilities of modern microcomputers along with webcams and affordable cloud storage allow us to establish an effective surveillance system with advanced functional capabilities, including motion and facial detection and even the identity of users.

Keywords: surveillance system, surveillance camera, motion detection, compression.