

Динамическая сегментация последовательности кадров*

Хашин С. И.

khash2@mail.ru, <http://math.ivanovo.ac.ru/dalgebra/Khashin/>

Ивановский государственный университет

Описан алгоритм сегментации, основанный не на одном кадре, а на паре соседних кадров из видеопоследовательности. По сравнению с обычной, статической сегментацией каждого кадра по отдельности качество значительно повышается. При сохранении той же погрешности количество сегментов удается сократить в несколько десятков раз. Типичное количество сегментов, нужных для получения приемлемой погрешности, уменьшается с 200 – 500 до 5 – 15.

Ключевые слова: *сегментация, последовательности кадров, клеточный автомат.*

Dynamic segmentation of frames sequences*

Khashin S. I.

Ivanovo State University

The image segmentation algorithm, based not on a single frame, but on the pair of frames from a video sequence is described. Compared to usual, static segmentation of each frame individually, the quality is improved drastically. While maintaining the same error, the number of segments can be reduced in tens times. Typical number of segments needed to produce an acceptable error is reduced from 200–500 to 5–15.

Keywords: *segmentation, frames sequences, cellular automaton.*

Введение

В основе алгоритмов сжатия видеоинформации лежит идея построения прогноза текущего кадра на основе предыдущего (или предыдущих). Во всех применяемых сегодня алгоритмах [1, 2, 3] для построения прогноза текущего кадра на основе предыдущего он разбивается на квадраты или прямоугольники небольшого размера (от 32×32 до 4×4) и для каждого из них находится вектор движения, с помощью этих векторов строится прогноз текущего кадра, находится дифференциальный кадр, *displaced frame difference (DFD)*, который тем или иным образом кодируется. Эти методы позволяют в несколько раз увеличить коэффициент сжатия без потери качества по сравнению со сжатием отдельных статических изображений.

Недостатком такого подхода является то, что движущиеся объекты плохо аппроксируются прямоугольниками и для получения приемлемого результата приходится разбивать кадр на очень мелкие блоки.

В работах [4, 5, 6] предложен новый, объектно-ориентированный алгоритм сжатия. Его основа — сегментация предыдущего кадра, т. е. разбиение его не на прямоугольники, а на сегменты произвольной формы.

Различных алгоритмов сегментации существует достаточно много, см., например, [7, 8]. Но экспериментальная проверка показывает, что, с нашей точки зрения, различия меж-

Работа выполнена при финансовой поддержке РФФИ, проект № 11-07-00653.

ду ними не так велики: погрешность сегментации (см. ниже определение 9) различается на 10–15% при ожидать появления новых методов сегментации, существенно уменьшающих эту погрешность.

Для наших целей алгоритм сегментации должен быть управляемым, т. е. должна быть возможность строить сегментацию с заранее заданным (ориентировочным) количеством сегментов. Практика показала, что для кадров размера 720×480 для построения кадра-прогноза с той же погрешностью, какую дает стандарт $h264$ при обычной величине шума $\text{PSNR} = 40$ (Peak Signal-to-Noise Ratio) обычно требуется сегментация на 100–500 сегментов. Причем это количество слабо зависит от размера кадра. При переходе от размера 320×240 к 1920×1080 требуемое количество сегментов увеличивается лишь в 2–4 раза.

Оказывается алгоритм можно существенно улучшить, если сегментировать предыдущий кадр не сам по себе, а на основе целой цепочки из нескольких предыдущих кадров. В простейшем случае можно сегментировать пару предыдущих кадров. В этом случае многие визуально различные сегменты обладающие одинаковым (аффинным) движением будут объединены в один сегмент. В результате, для получения сегментации того же качества оказывается достаточно строить не несколько сотен сегментов, а не более одного десятка. Отметим, что *качество* сегментации в нашем случае — это не субъективная оценка, а вполне конкретное число (см. Определение 9).

Построению такого алгоритма сегментации и посвящена настоящая статья.

Основные определения

В настоящей работе мы будем рассматривать прямоугольные полноцветные (TrueColor) изображения.

Определение 1. Изображением (кадром) размера $tx \times ty$ будем называть набор из трех матриц (*RGB*) размера $tx \times ty$. Обычно, но не обязательно, элементы этих матриц являются целыми числами из отрезка $[0..255]$. Эти три матрицы мы будем рассматривать как одно отображение $F : U \rightarrow \mathbb{R}^3$, где U — подмножество в \mathbb{Z}^2 , состоящее из точек $(x, y) : 0 \leq x < tx, 0 \leq y < ty$. Точки из U будем называть пикселями для отличия от остальных точек \mathbb{R}^2 . Отображение F естественным образом продолжается до отображения $\mathbb{Z}^2 \rightarrow \mathbb{R}^3$. Кроме того, используя некоторую интерполяционную формулу (билинейную, бикубическую, сплайны и т. д.), функцию F можно продолжить до непрерывной кусочно-полиномиальной функции $\mathbb{R}^2 \rightarrow \mathbb{R}^3$, которую мы также будем обозначать буквой F .

Определение 2. Сегментацией $U = \cup U_i$ изображения будем называть разбиение области U на произвольные части U_1, \dots, U_N , сегменты.

Определение 3. Размером $k(U_i)$ сегмента U_i будем называть количество пикселов в нем.

Определение 4. Пиксель $(x, y) \in U$ будем называть граничным для данной сегментации, если он принадлежит одному сегменту, а один из его четырех соседних — другому.

Определение 5. Два сегмента называются соседними, если в первом сегменте существует пикセル, один из соседних к которому принадлежит второму сегменту.

Определение 6. Аффинной сегментацией $S = \{U_i, A_i, i = 1, \dots, N\}$ будем называть обычную сегментацию $U = \cup U_i$ снабженную набором аффинных преобразований $A_i : U_i \rightarrow \mathbb{R}^2$:

$$A_i : \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} a_0 + a_1x + a_2y \\ a_3 + a_4x + a_5y \end{pmatrix}. \quad (1)$$

Для каждого пикселя (x, y) из области U обозначим через $S^*(x, y) \in \mathbb{R}^2$ точку $A_i(x, y)$, где i — номер сегмента, к которому принадлежит пикセル (x, y) .

Определение 7. а) Пусть f — функция $\mathbb{R}^2 \rightarrow \mathbb{R}^3$ и A — аффинное преобразование плоскости вида (1). Через $A^*(f)$ обозначим функцию $\mathbb{R}^2 \rightarrow \mathbb{R}^3$

$$A^*(f)(x, y) = f(a_0 + a_1 x + a_2 y, a_3 + a_4 x + a_5 y).$$

б) Пусть, дополнительно, (F_1, F_2) — два изображения (кадра) одинакового размера (то есть два отображения $U \rightarrow \mathbb{R}^3$). Для каждого пикселя (x, y) длину вектора $F_1(x, y) - A^*(F_2)(x, y)$ будем называть погрешностью преобразования A на паре кадров (F_1, F_2) в этом пикселе.

Определение 8. Пусть S — аффинная сегментация области U . Тогда для произвольного изображения F через $S^*(F)$ обозначим новое изображение, строящееся по формулам $S^*(F)(x, y) = F(A_i^*(x, y))$, где i — номер сегмента, к которому принадлежит пикSEL (x, y) .

Оценка качества сегментации в общем виде — сложная и, вообще говоря, очень субъективная задача. Если нас интересуют не отдельные изображения, а видеопоследовательности, цепочки изображений, мы можем предложить вполне естественное для данной ситуации понятие качества сегментации.

Определение 9. Пусть (F_1, F_2) — два изображения одинакового размера и S — аффинная сегментация области U . Изображение $F_1 - S^*(F_2)$ будем называть разностным изображением. Среднеквадратичное значение длины вектора $|F_1 - S^*(F_2)|$ по всем пикселам области U будем называть погрешностью сегментации.

Всюду в дальнейшем мы будем предполагать фиксированной пары кадров (F_1, F_2) одинакового размера $tx \times ty$.

Алгоритм «A»

Обычно при обработке видеоданных [1, 2, 3, 9] используются те или иные методы нахождения движения. Все они являются различными вариантами одного и того же алгоритма Лукаса–Канады [10, 11] и находят движения лишь в виде сдвигов. Эти методы очень эффективны, но лишь для случая малых областей, сегментов. Во всех реализованных на сегодняшний день способах видеообработки это так и есть. Но в случае больших сегментов, занимающих значительную долю всего кадра, мы должны искать движения более общего вида, например, аффинных преобразований вида (1).

В работе [6] приведен алгоритм поиска таких преобразований, минимизирующих погрешность для данной пары кадров при фиксированной сегментации. Этот алгоритм заключается в следующем. Для каждого сегмента U_i рассмотрим аффинное преобразование A_i вида (1). Для него на сегменте U_i вычисляем величину

$$S = \sum_{(x,y) \in U_i} |F_1(x, y) - A_i^*(F_2)(x, y)|^2$$

и находим значения (a_0, \dots, a_5) , при которых S достигает минимума. Описанный алгоритм является естественным обобщением стандартного алгоритма Лукаса–Канады на случай аффинных преобразований и практически так же эффективен и надежен. Разумеется, так как нам надо определить не две координаты вектора сдвига, а все шесть параметров

аффинного преобразования, он будет работать медленнее обычного алгоритма Лукаса–Канады. Однако это будет замедление в разы, а не на порядки. Будем называть этот процесс алгоритмом «А».

В задаче, исследуемой в настоящей работе, мы имеем возможность менять не только аффинные преобразования, но и саму сегментацию.

Клеточный автомат

Сегменты, получающиеся в процессе построения, иногда могут иметь очень хаотичную структуру. Для их сглаживания будем использовать клеточный автомат. При его описании «соседними» для данного пикселя считаются его восемь соседей. Таким образом, у углового пикселя области U — три соседних, у пикселя на границе кроме угловых — 5 соседних, у внутренних — по 8 соседей.

Один шаг работы клеточного автомата заключается в следующем. Перебираем все пиксели области U в псевдослучайном порядке. Если очередной пикセル принадлежит сегменту i , а более половины его соседей — к сегменту $j \neq i$, то относим этот пикセル к сегменту j , в противном случае ничего не делаем. Таким образом, за один шаг мы перебираем все пиксели кадра в псевдослучайном порядке.

Работа алгоритма зависит от выбранного способа псевдослучайного обхода всех $mx \times my$ пикселов области U . Метод рандомизации, применяемый в нашей работе, основан на выборе простого числа p , несколько большего, чем $mx * my$ и первообразного корня a по модулю p . Как известно, в этом случае последовательность $1, a, a^2, \dots, a^{p-1}$ является перестановкой чисел $1, \dots, p - 1$.

Применением клеточного автомата к данной сегментации будем называть результат последовательного выполнения описанных выше шагов клеточного автомата. Работа алгоритма прекращается, если на очередном шаге автомата сегментация не изменилась.

Хорошо известно, что описанный выше клеточный автомат закончит свою работу за конечное количество шагов.

Практика показывает, что описанный алгоритм дает хорошие результаты по сглаживанию сегментов.

Разделение сегмента

Пусть (F_1, F_2) — два изображения одинакового размера и U_1 — некоторый сегмент на U . Выделим из него подсегмент U_2 по следующему алгоритму. Вначале зафиксируем некоторую величину ожидаемого шума e , для типичных последовательностей кадров удобно будет взять $e = 2$.

1. Строим матрицу R , принимающую значения 1 в пикселях сегмента и 0 в остальных пикселях.
2. Сглаживаем матрицу R (см. ниже) с радиусом $(mx + my)/10$.
3. Находим пикセル (x_S, y_S) , в котором эта сглаженная матрица принимает наибольшее значение. Если таких пикселов несколько, берем первый из них в лексикографическом порядке.
4. Обходим по спирали пиксели вокруг (x_S, y_S) . Пиксели, принадлежащие сегменту U_1 , добавляем в сегмент U_2 до тех пор, пока не получим 100 пикселов. Если во всем сегменте не более 100 пикселов, то такой сегмент не будем разбивать на части.
5. С помощью алгоритма «А» находим аффинное преобразование A_2 , оптимальное для сегмента U_2 . Величину погрешности преобразования A_2 на сегменте U_2 обозначим e_2 . Если $e_2 < e$, положим $e_2 = e$.

6. Находим E — матрицу погрешностей преобразования A_2 на всей области U .
7. Сглаживаем матрицу E с радиусом 10.
8. В сегмент U_2 теперь отбираем те пиксели из исходного сегмента U_1 , для которых величина сглаженной погрешности не превышает e_2 .
9. Сглаживаем построенную область с помощью клеточного автомата.
10. Повторяем шаги (5, 6, 7, 8, 9) до тех пор, пока сегмент U_2 не стабилизируется. Численные эксперименты показывают, что требуемое количество итераций от 3 до 6, причем более четырех повторений требуется крайне редко.

При разбиении сегмента используется некоторая версия алгоритма «сглаживания» матрицы. Опишем его более подробно.

Определение 10. Сглаживанием матрицы R с радиусом k назовем ее свертку с ядром $Q(x, y) = q(|x|) \cdot q(|y|)$, где

$$q(x) = \begin{cases} (k+1-x)/(k+1)^2 & x \leq k+1 \\ 0 & x > k+1 \end{cases},$$

т. е.

$$R'(x, y) = \sum_{dx, dy=-k}^k Q(dx, dy) \cdot R(x + dx, y + dy)$$

В частности, при сглаживании с радиусом 0 мы получаем исходную матрицу, при сглаживании с радиусом 1 — свертку с ядром:

$$\frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Можно использовать и более сложные методы сглаживания, например гауссово [7, 8], но это вряд ли существенно изменит результаты.

Улучшение сегментации

Пусть (F_1, F_2) — два изображения одинакового размера и $S = \{U_i, A_i, i = 1, \dots, N\}$ — аффинная сегментация области U . Для ее улучшения применяем следующий алгоритм.

1. Для всех аффинных преобразований A_i строим E_i — матрицы погрешностей преобразований A_i на всей области U .
2. Сглаживаем матрицы E_i с радиусом 10.
3. В сегмент U_i отбираем те пиксели из U , для которых величина сглаженной погрешности наименьшая среди всех E_i .
4. Сглаживаем построенные сегменты с помощью клеточного автомата.
5. Удаляем пустые сегменты.
6. С помощью алгоритма «А» находим оптимальные аффинные преобразования для каждого вновь полученного сегмента U_i .
7. Повторяем шаги (1 — 6) до тех пор, пока сегменты U_i почти не стабилизируются. Более точно — пока количество пикселов, перешедших в другой сегмент, в течение шага не станет меньше небольшой выбранной константы. В проведенных экспериментах константа выбиралась равной 10. Практика показала, что после четырех повторений практически всегда сегменты полностью перестают изменяться.

Динамическая сегментация

Пусть дана пара последовательных кадров (F_1, F_2) и пусть мы хотим разбить второй кадр F_2 на N сегментов.

Алгоритм динамической сегментации состоит из следующих шагов.

1. Начинаем с одного сегмента U_0 , состоящего из всей области U и тождественного аффинному преобразованию A_0 .
2. На каждом шаге выделяем из U_0 один сегмент с помощью алгоритма, описанного выше, до тех пор, пока не получим требуемое количество сегментов.
3. Улучшаем полученную сегментацию.

Результаты экспериментов

В данной задаче для оценки качества получающейся сегментации имеется уже готовый числовой параметр — погрешность сегментации (см. Определение 9).

Один сегмент, сдвиг

Для проверки работоспособности алгоритма берем следующую пару кадров: $f1$ — левые 500 столбцов стандартного изображения `lena.bmp` размера 512×512 , $f2$ — правые 500 столбцов. Таким образом, $f2$ получается из $f1$ со сдвигом на 12 точек по горизонтали. В результате применения описанного алгоритма получаются два сегмента. Один содержит почти весь кадр, точнее говоря 99,66% от площади кадра с аффинным преобразованием:

$$\begin{aligned}x' &= 11,998994 + 1,000002x + 0,000003y; \\y' &= 0,000539 - 0,000001x + 0,999998y.\end{aligned}$$

Фактически оно совпадает со сдвигом на 12 точек влево. Второй сегмент появляется только из-за краевых эффектов. Общая погрешность преобразования оказывается равной 2,7. На примере этой практически идеальной ситуации мы видим, каких показателей стоит ожидать от «хорошей» аффинной сегментации.

Несколько сегментов со спрайтами

Здесь мы опять строим пару искусственных кадров. Каждый кадр размера 640×480 состоит из фона и четырех небольших спрайтов (эллипсы с полуосами 50..70). И для фона, и для каждого спрайта задано свое аффинное преобразование.

Алгоритм построил 8 сегментов, на рис. 2(а) разными цветами показаны разные сегменты, (б) — прорисованы их границы на исходном изображении. Погрешность сегментации оказывается равной 8,6. Более подробное изучение дифференциального кадра показывает, что, как и следовало ожидать, наибольшая погрешность — в областях, которым нет соответствия на втором кадре, «uncovered background». Это случай тоже можно рассматривать как близкий к идеальному, все сегменты и аффинные преобразования оказались найдены верно.

Реальные видеопоследовательности

Описанные алгоритмы были применены к некоторым реальным видеопоследовательностям. Среди них стандартная видеопоследовательность `Foreman` размерности 352×288 и мультифильм `Factory` размерности 1920×1080 .

В последовательности `Foreman` получается от 2 до 11 сегментов (на рис. 4, кадр № 173 — 10 сегментов. Слева разные сегменты показаны разным цветом, справа — на исходном изображении прорисованы границы сегментов), погрешность сегментации — от 5,6 до 9,8.



Рис. 1: Два кадра со спрайтами

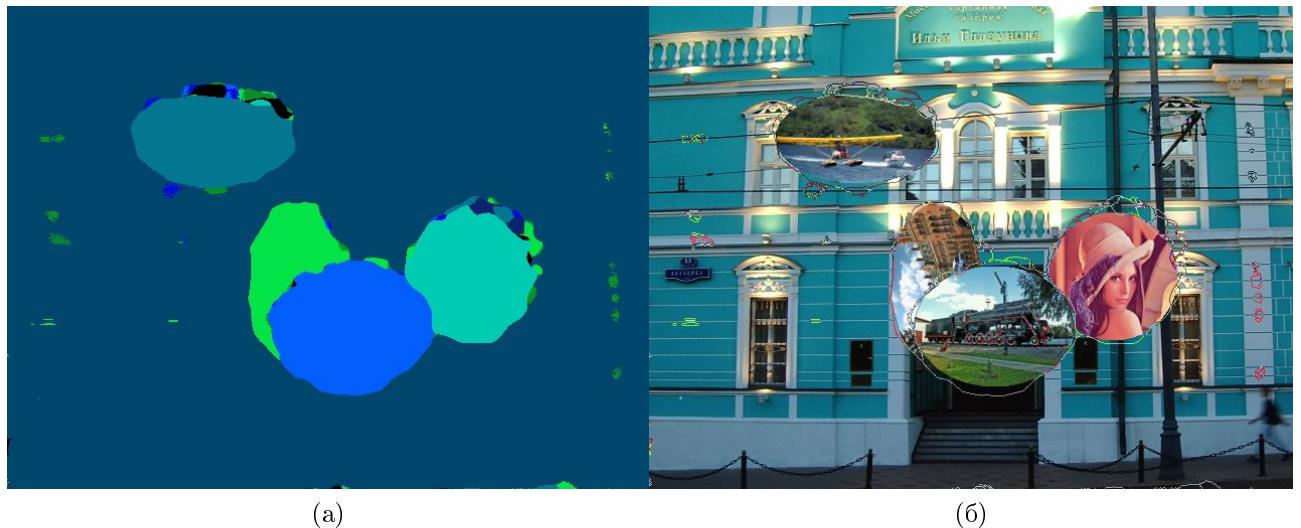


Рис. 2: Сегментация, 8 сегментов

Видно, что сегментация связана именно с различными движениями частей кадра. Например, если глаза двигаются синхронно со всем лицом, то они не выделяются в отдельные сегменты.

В последовательности *Factory* получается от 2 до 15 сегментов, погрешность сегментации — от 2,6 до 11,2. То есть при огромном увеличении площади кадра количество сегментов увеличивается незначительно.

Заключение

В статье приводятся лишь первые экспериментальные результаты, полученные на сравнительно небольшом количестве примеров. Более полное экспериментальное исследование алгоритма, а также его оптимизация будут произведены позднее.

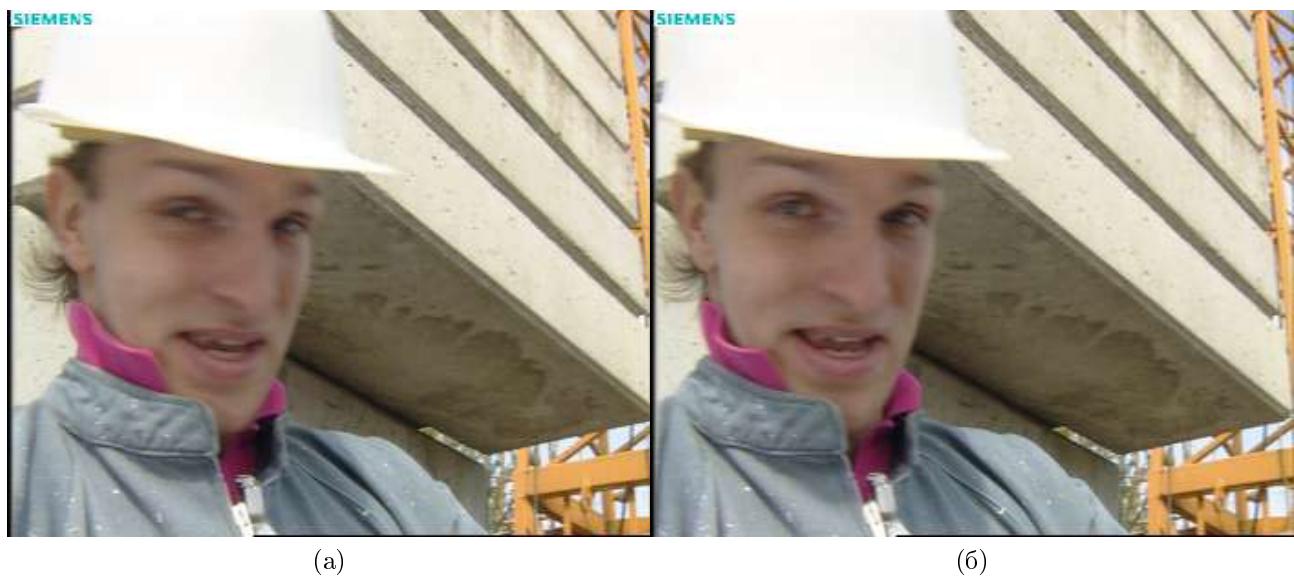


Рис. 3: Исходные кадры, номера (172, 173)

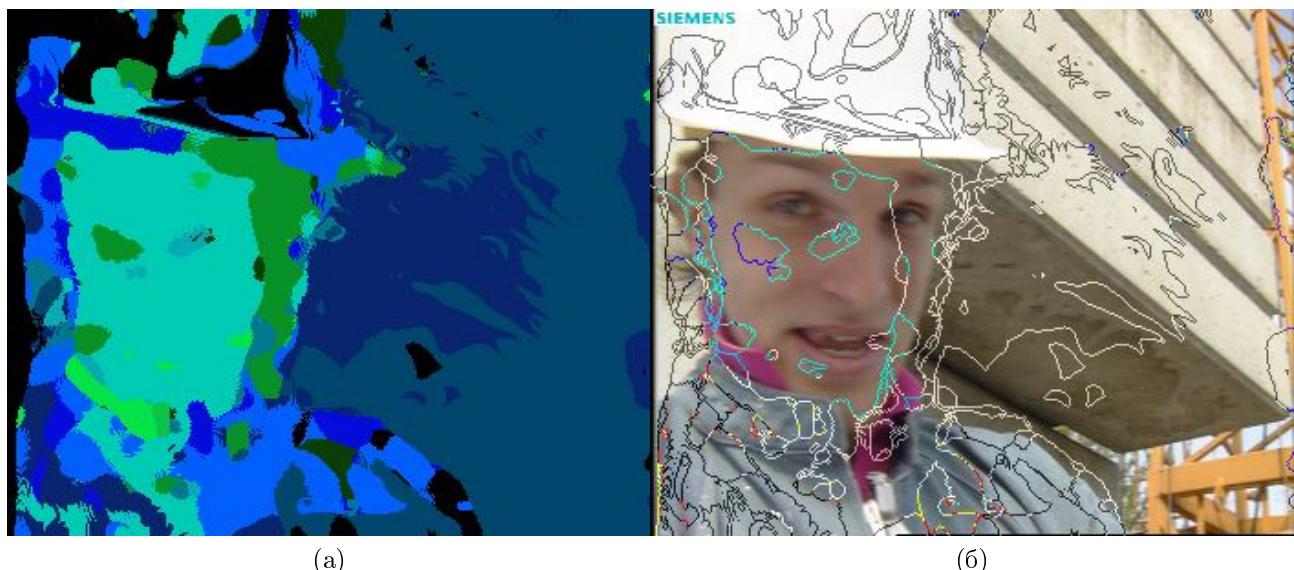


Рис. 4: Сегментация на 10 сегментов

1. Предложенный метод дает весьма эффективный алгоритм сегментации: по сравнению со статической сегментацией при той же погрешности количество сегментов можно уменьшить в несколько десятков раз. Здесь требуется более детальное сравнение.
2. Алгоритм надежен: визуальное изучение получающихся сегментов показывает, что нет практически ни одной существенной ошибки.
3. Количество сегментов меняется в пределах от 2 до 20, причем оно мало меняется с ростом размера кадра.
4. Погрешность сегментации — от 1 до 14, причем она не изменяется с ростом размера кадра.
5. В реальных кадрах основной составляющей погрешности является не ошибки сегментации и нахождения движения, а «uncovered background» и цветовое различие между

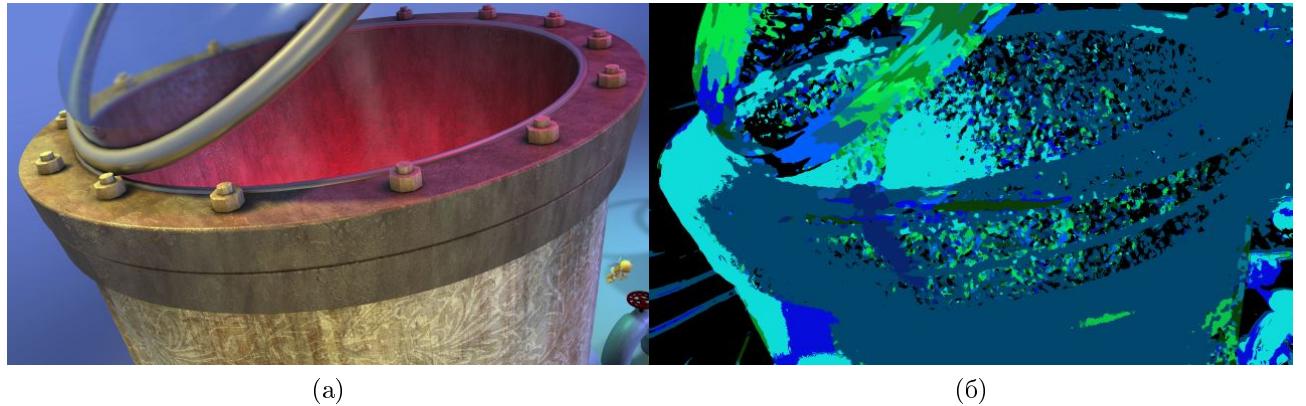


Рис. 5: Исходный кадр размера 1920×1080 и его сегментация на 15 сегментов

кадрами, т. е. причины, которые невозможно устранить выбором сегментации и аффинных преобразований.

Литература

- [1] *ITU-T and ISO/IEC JTC 1 Generic coding of moving pictures and associated audio information. Part 2: Video* // ITU-T Recommendation H.262 – ISO/IEC 13818-2 (MPEG-2), Nov. 1994.
- [2] *ITU-T Video coding for low bit rate communication* // ITU-T Recommendation H.263; ver. 1, Nov. 1995; ver. 2, Jan. 1998; ver. 3, Nov. 2000.
- [3] *ITU-T Rec. H.264 / ISO/IEC 11496-10. Advanced video coding* // Final Committee Draft, Document JVT-E022, Sept. 2002.
- [4] *Хашин С. И. Применение методов распознавания образов для сжатия видеоинформации* // Докл. всеросс. конф. ММРО-13. — М.: МАКС Пресс, 2007. С. 420–424.
- [5] *Хашин С. И. Оценка качества сегментации изображения* // Вестник ИвГУ. Вып. 2. — Иваново, 2010. "С. 112–118.
- [6] *Хашин С. И. Аффинная версия алгоритма Лукаса-Канады* // Докл. всеросс. конф. ММРО-13. — М.: МАКС Пресс, 2011. "С. 459–462.
- [7] *Гонсалес Р., Будс Р. Цифровая обработка изображений.* — М.: Техносфера, 2006. — 1072 с.
- [8] *Яне Б. Цифровая обработка изображений.* — М.: Техносфера, 2007. 583 с.
- [9] *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification* // (ITU-T Rec.H.264.ISO/IEC14496-10AVC) Joint Video Team (JVT), Mar. 2003. Doc. JVT-G050.
- [10] *Lucas B. D., Kanade T. An iterative image registration technique with an application to stereo vision* // Imaging Understanding Workshop Proceedings. 1981. Pp. 121–130
- [11] *Baker S., Gross R., Matthews I. Lucas-Kanade 20 years on: A unifying framework* // Int. J. Computer Vision, 2002. Vol.56. Pp. 111–122.