Аффинная версия алгоритма Лукаса-Канады*

Хашин С. И.
khash2@mail.ru
Иваново, Ивановский Университет

Предлагается алгоритм поиска межкадрового движения в виде аффинного преобразования общего вида при сжатии видеоинформации. Он является обобщением классического алгоритма Лукаса-Канады [1, 6], применяемого для нахождения междкадрового движения в виде сдвига. Предложен эффективный метод кодирования построенных преобразований.
Все рассмотренные алгоритмы реализованы и протестированы, язык реализиации — $C++$.

Одним из основных инструментов, используемых при сжатии видеоинформации, является приближенное построение области на следующем кадре в виде сдвига соответствующей области с предыдущего кадра. Эту схему используют все работающие в настоящее время алгоритмы — от $MPEG-2$ до $H.264$ [2, 3, 4, 5]. Они различаются в этом пункте лишь размером обрабатываемой области (от 16∗16 до 2∗2), и методом интерполяции яркости точки между пикселами (билинейная, бикубическая и до би-пятой степени в $H.264$). Для нахождения таких сдвигов используется классический алгоритм Лукаса-Канады [1, 6]. Он отличается высокой скоростью работы и большой устойчивостью.

Однако, в разрабатываемых, перспективных алгоритмах видеосжатия (например [8]) только сдвигов оказывается недостаточно. В основном это связано с тем, что рассматриваемая область на кадре (сегмент) может оказаться достаточно большой и ее движение не может быть удовлетворительно описано с помощью сдвига. Поэтому приходится переходить к более сложным движениям на плоскости:

— сдвиг;
— сдвиг и поворот;
— сдвиг, поворот и растяжение;
— общее аффинное преобразование плоскости;
— общее проективное преобразование плоскости.

Алгоритму нахождения таких преобразований и посвящена настоящая статья.

Метод Лукаса-Канады

Опишем вначале, вкратце, классический метод Лукаса-Канады (см. например, [1]).

Пусть яркость первого кадра из пары задается в целочисленных точках таблицей $f[x, y]$. Ее продолжение с помощью некоторого интерполяционного метода на всю плоскость и будем обозначать ее $f(x, y)$. Для второго кадра из пары значения в целочисленных точках — $g[x, y]$, на всей плоскости — $g(x, y)$.

Пусть мы хотим аппроксимировать значения функции $g[x, y]$ в целочисленных точках некоторой области U, обычно это прямоугольник сравнительно небольшого размера, от 2∗2 до 16∗16. Для этого положим $g[x, y] \approx f(x - v_x, y - v_y)$, где (v_x, v_y) — некоторый вектор (*вектор скорости, вектор сдвига*), не обязательно целочисленный. Для оценки погрешности этого приближения рассмотрим сумму

$$S(v_x, v_y) = \sum_{(x,y)\in U} (g[x, y] - f(x - v_x, y - v_y))^2. \quad (1)$$

Определение 1. *Вектором движения для пары кадров (f, g) в области U будем называть вектор $v = (v_x, v_y)$, для которого $S(v_x, v_y)$ — минимальна.*

Замечание 1. Значения функции $g[x, y]$ берутся только в целочисленных точках, а функции $f(x, y)$ — в любых действительных. Для этого требуется некоторая интерполяционная формула (см. например, [7, 9]), в алгоритме Лукаса-Канады предполагается — билинейная.

Замечание 2. Нахождение векторов скорости требуется не во время просмотра фильма, а во время кодирования. Поэтому временные требования к алгоритму не слишком жесткие — здесь не требуется обрабатывать по 24 кадра в секунду на стандартном процессоре.

В точке минимума должны выполняться соотношения:

$$\begin{aligned} \partial S/\partial v_x &= 2\sum(g[x, y] - f(P_v))f'_x(P_v) = 0\ , \\ \partial S/\partial v_y &= 2\sum(g[x, y] - f(P_v))f'_y(P_v) = 0\ , \end{aligned} \quad (2)$$

где через P_v обозначена точка с координатами $(x - v_x, y - v_y)$.

Будем предполагать, что (v_x, v_y) — некоторое, достаточно хорошее приближение к точному решению системы уравнений (2), которое будем искать в виде $(v_x + \Delta_x, v_y + \Delta_y)$. *Достаточно хорошее* означает, что для $f(x - (v_x + \Delta_x), y - (v_y + \Delta_y))$ можно использовать линейную аппроксимацию:

$$\begin{aligned} &f(x - (v_x + \Delta_x), y - (v_y + \Delta_y)) \approx \\ &\approx f(x - v_x, y - v_y) - \Delta_x f'_x - \Delta_y f'_y\ . \end{aligned} \quad (3)$$

С учетом этого, система уравнений (2) будет линейной системой размера 2×2 от переменных (Δ_x, Δ_y).

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

Один шаг метода Лукаса-Канады состоит в решении этой системы уравнений и замене вектора скорости (v_x, v_y) на $(v_x + \Delta_x, v_y + \Delta_y)$.

Так как процедуру приходится выполнять много раз, то следует заранее, наряду с матрицами $f[x,y]$, $g[x,y]$, приготовить матрицы частных производных $f'_x[x,y]$ и $f'_y[x,y]$.

Пирамидальная версия. В данном алгоритме важным является нахождение хорошего начального приближения для вектора скорости. Для этого обычно применяют *пирамидальную* версию алгоритма. Ее идея заключается в том, что наряду с исходной парой изображений (f, g) рассматривают эти же изображения сжатые в два раза (f_2, g_2), в четыре (f_4, g_4) и т.д. (*пирамида*). Вектора скорости находят сначала на самом верхнем уровне пирамиды и затем спускаются вниз этаж за этажом. На самом верхнем уровне в качестве начального приближения берут нулевой вектор. На нижних уровнях за начальное приближение берут удвоенную скорость, полученную на предыдущем шаге.

Все это вместе взятое обеспечивает хорошее сочетание скорости, точности и устойчивости алгоритма нахождения межкадрового движения в виде сдвигов.

Обобщение метода Лукаса-Канады

В используемых алгоритмах видеосжатия не используются движения плоскости более сложные, чем сдвиг. Это связано с тем, что обрабатываемые области малы и их движения достаточно хорошо описываются сдвигами. Но если переходить к алгоритмам видеосжатия следующего поколения ([8]), то придется обрабатывать и довольно большие сегменты. Например, межкадровое движение фона, занимающего большую часть кадра, часто можно описать одной-единственной формулой. В этом случае придется иметь дело с более сложными движениями:

— сдвиг (2 параметра);

$$\begin{aligned} x' &= a_1 + x \ , \\ y' &= a_2 + y \ , \end{aligned} \tag{4}$$

— сдвиг и поворот (3 параметра);

$$\begin{aligned} x' &= a_1 + \cos a_3 \cdot x + \sin a_3 \cdot y \ , \\ y' &= a_2 - \sin a_3 \cdot x + \cos a_3 \cdot y \ , \end{aligned} \tag{5}$$

— сдвиг, поворот и растяжение (4 параметра);

$$\begin{aligned} x' &= a_1 + a_4(\cos a_3 \cdot x + \sin a_3 \cdot y) \ , \\ y' &= a_2 + a_4(-\sin a_3 \cdot x + \cos a_3 \cdot y) \ , \end{aligned} \tag{6}$$

— общее аффинное преобразование плоскости (6 параметров);

$$\begin{aligned} x' &= a_1 + a_2 \cdot x + a_3 \cdot y \ , \\ y' &= a_4 + a_5 \cdot x + a_6 \cdot y \ , \end{aligned} \tag{7}$$

— общее проективное преобразование плоскости (8 параметров).

$$\begin{aligned} x' &= \frac{a_1 + a_2 \cdot x + a_3 \cdot y}{1 + a_7 \cdot x + a_8 \cdot y} \ , \\ y' &= \frac{a_4 + a_5 \cdot x + a_6 \cdot y}{1 + a_7 \cdot x + a_8 \cdot y} \ , \end{aligned} \tag{8}$$

В общем виде будем это записывать так:

$$\begin{aligned} x' &= A_x(x,y) \ , \\ y' &= A_y(x,y) \ , \end{aligned} \tag{9}$$

где функции A_x, A_y зависят, помимо x, y, еще от некоторого набора параметров $(a_1, \ldots, a_k)$.

Каждое движение $A = (A_x, A_y)$ можно рассматривать и как более сложное: сдвиг — как сдвиг и поворот на нулевой угол, сдвиг и поворот — как сдвиг, поворот и растяжение с коэффициентом 1 и так далее.

Можно рассмотреть и обратную редукцию — более сложного движения к более простому, при заданной области на плоскости.

Определение 2. *Пусть A_1 — движение плоскости одного из типов от (5) до (8). Преобразование A_2 более простого типа будем называть редукцией A_1 на множестве U, если оно менее всего отклоняется от исходного движения A_1 на точках области U среди всех преобразований данного типа (в среднеквадратичном смысле).*

Задача нахождения редукции решается аналитически и мы будем спускаться от сложных движений к более простым, если это не ведет к потере точности.

По аналогии с обычным алгоритмом Лукаса-Канады, будем искать приближения для функции $g[x,y]$ в виде $g[x,y] \approx f(A_x(x,y), A_y(x,y))$. Рассмотрим сумму по точкам области U:

$$S(a_i) = \sum (g[x,y] - f(A_x(x,y), A_y(x,y)))^2. \tag{10}$$

Определение 3. *Оптимальным движением A для пары кадров (f, g) в области U будем называть движение плоскости одного из перечисленных выше типов, для которого величина $S(A) = S(a_i)$ — минимальна среди всех движений данного типа.*

Для того, чтобы преобразование A задавало оптимальное движение области U должны выполняться соотношения (аналог уравнений (2)):

$$\frac{\partial S}{\partial a_i} = 0 \ , \quad i = 1, \ldots, k \ , \tag{11}$$

После линеаризации, аналогичной (3), система (11) окажется линейной системой уравнений размера $(k \times k)$ относительно приращений $(\Delta_1, \ldots, \Delta_k)$ параметров преобразования $(a_1, \ldots, a_k)$.

Определение 4. *Базовым шагом обобщенного метода Лукаса-Канады будем называть решение системы уравнений (11) и замену параметров преобразования $(a_1, \ldots, a_k)$ на $(a_1 + \Delta_1, \ldots, a_k + \Delta_k)$.*

Замечание 3. Если движение плоскости ищется в виде (4), то обобщенный метод Лукаса-Канады в точности совпадает с классическим.

Замечание 4. На сегодняшний день движения в виде общего проективного преобразования (8) не реализовано. Возможно, в этом и нет необходимости: переход от общего аффинного преобразования к нему, скорее всего не даст заметного выигрыша. Но это пока лишь предположение, хотя и весьма вероятное.

Замечание 5. Рассмотренные алгоритмы довольно сложны с чисто математической точки зрения, являются итерационными, многошаговыми процессами. Поэтому для отладки были взяты искусственную пары кадров, в который первый получается из второго путем заранее заданного аффинного преобразования, — искомое движение в этом случае заранее точно известно.

Редукция исходной области U

Для решения задачи минимизации мы должны задаться некоторой областью U на втором кадре из пары. На самом деле для работы алгоритма Лукаса-Канады вовсе не требуется, чтобы U было областью в геометрическом понимании — это просто произвольное множество точек со второго кадра.

Исходная область U может состоять из очень большого количества точек, вплоть до почти всего кадра, а это — до двух миллионов точек в случае `FULL-HD`-кадров (1920×1080). Поэтому для эффективного применения алгоритма количество точек в области надо сократить.

Определение 5. *Подмножество точек U' из U будем называть его* редукцией, *если оно содержит все те точки из U, координаты которых делятся на k для некоторого натурального k.*

Определение 6. *Подмножество точек U'' из U будем называть его* равномерной редукцией, *если оно получается следующим образом. В начале в U'' кладем одну, случайно выбранную точку из U. Затем последовательно добавляем точку из U, находящуюся на расстоянии не менее k от всех уже выбранных точек для некоторого k.*

Оба множества, U' и U'' содержат порядка $|U|/k^2$ точек, поэтому выбрав подходящее k мы можем получать множества, мощность которых примерно равна любой наперед заданной.

На первый взгляд равномерная редукция кажется более надежным способом, хотя и существенно более сложным вычислительно. Однако на практике оказалось, что простая редукция, гораздо более быстрая, дает вполне удовлетворительные результаты, мало отличающиеся от равномерной (в случае наших областей и с точки зрения наших целей). Поэтому в дальнейшем мы будем рассматривать только простую редукцию.

В разрабатываемом алгоритме используются две (простых) редукции U_1 и U_2 исходной области U. U_1 — это редукция до примерно 2000 точек. Если мощность исходной области U меньше 5000 точек, то просто полагаем $U_1 = U$. U_2 — это редукция до примерно 200 точек. Если мощность исходной области U меньше 500 точек, то полагаем $U_1 = U_2 = U$.

Область U_2 используется для нахождения начального приближения искомого движения плоскости (A_x, A_y), область U_1 — для последующего уточнения. Если оказывается, что $U_1 = U_2$, то мы получает только один этап, без последующего уточнения.

Общая схема алгоритма

Сначала рассмотрим работу алгоритма для фиксированной области $\tilde{U}$ на втором кадре из пары (f, g) (это может быть как U_1, так и U_2). Пусть A_0 — начальное приближение к оптимальному движению плоскости на этой области. A_0 может быть любого типа — от сдвига до общего аффинного преобразования.

Для каждого преобразования A у нас есть его численная оценка (погрешность): величина $S = S(A)$ из формулы (10), чем она меньше, тем лучше. В идеале она должна равняться нулю.

Рассматриваемый этап алгоритма состоит их следующих шагов.

1. Упрощение начального приближения. Исходное преобразование A_0 редуцируем к более простому типу (в смысле определения 2), до тех пор, пока его погрешность уменьшится не более, чем на C_1 процентов. Полученное преобразование обозначим A_1 (оно может и совпадать с A_0).

2. Базовые шаги. Начиная с преобразования A_1 выполняем базовые шаги (опр. 4) обобщенного метода Лукаса-Канады пока не исчерпаем количество шагов (не более C_2) и уменьшение погрешности на одном базовом шаге составляет не менее C_3%. Найденное преобразование обозначим A_2, оно того же типа, что и A_1.

3. Усложнение типа преобразования. Если тип преобразования A_2 не самый сложный (не полное аффинное преобразование, проективные мы пока не рассматриваем вообще), то повышаем тип (от сдвига — к сдвигу и повороту, и т.д.) и с новым преобразованием повторяем базовые шаге оптимизации. Если погрешность уменьшилась менее, чем на C_1 процентов, то от такого усложнения отказываемся и заканчиваем алгоритм.

Замечание 6. В программе, константы взяты следующие: $C_1 = 5\%$, $C_2 = 10$, $C_3 = 0.1\%$.

На основе большого количества численных экспериментов был сделан вывод, что пирамидальная схема является в данном случае неэффективной. Вместо нее был предложена редукция от заданной области U к паре областей (U_1, U_2), таких, что мощность U_1 порядка 2000 точек, а у U_2 — порядка 200 (см. выше).

Сначала, начиная с тождественного преобразования (сдвиг на вектор $(0,0)$), находим оптимальное преобразование для области U_2. Затем, используя полученное преобразование в качестве начального приближения, находим оптимальное преобразование для области U_1.

Кодирование преобразований

Запоминать построенные аффинные преобразования путем хранения всех их коэффициентов, конечно же неэффективно. Более эффективный способ можно предложить вспомнив, что преобразование A применяется не на всей плоскости, а лишь в области U. Рассмотрим на плоскости точку P_0 — центр тяжести области U (округленный до целых) и обозначим через r среднеквадратичное расстояние от точек U до P_0, округленное до целых. Построим точку P_1 на расстоянии r от P_0 вправо и P_2 на расстоянии r от P_0 вниз. Любое аффинное преобразование A полностью определяется образами этих трех точек $A(P_0)$, $A(P_1)$, $A(P_1)$. Введем три пары чисел:

1. Пара (w_0, w_1) — вектор $A(P_0) - P_0$, т.е. вектор сдвига центра тяжести области U.

2. Пара (w_2, w_3) — разность векторов сдвига точек P_1 и P_0.

3. Третья пара. Пусть A' — преобразование типа (6): <<*сдвиг + поворот + растяжение*>>, переводящее P_0 в $A(P_0)$ и P_1 в $A(P_1)$. Такое всегда существует и единственно. В качестве третьей пары чисел (w_4, w_5) возьмем координаты вектора $A(P_2) - A'(P_2)$.

Теорема 1. *Числа $w_0, \ldots, w_5$ однозначно определяют преобразование A.*

Замечание 7. Если преобразование является сдвигом, то $w_2 = w_3 = w_4 = w_5 = 0$. Если преобразование близко к сдвигу, то $w_2, w_3, w_4, w_5 \approx 0$.

Замечание 8. Если преобразование имеет тип <<*сдвиг + поворот + растяжение*>>, то $w_4 = w_5 = 0$. Если преобразование близко к таком типу, то $w_4, w_5 \approx 0$.

Построенное преобразование для области U будем запоминать не через его коэффициенты, а через шесть чисел $(w_0, \ldots, w_5)$, которые можно округлить до заданной точности (1/16).

Полученные результаты

Построено обобщение классического алгоритма Лукаса-Канады для нахождения межкадрового движения не только в виде сдвигов, но и в виде более сложных движений плоскости, вплоть до аффинного преобразования общего вида.

Разработанные алгоритмы реализованы на языке C++. Разработка велась так, чтобы можно было использовать не менее трех различных компиляторов, чтобы не быть привязанным в будущем к конкретной системе программирования.

На основе большого количества численных экспериментов построена схема работы, обеспечивающая баланс между точностью, устойчивостью и объемом вычислений.

Предложен эффективный метод хранения построенных преобразований.

Литература

[1] *S. Baker, R. Gross, I. Matthews* Lucas-Kanade 20 Years On: A Unifying Framework // Int.J.of Computer Vision, 2002,— Vol. 56, Pp. 111–122.

[2] *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.

[3] *ITU-T* Video coding for low bit rate communication // ITU-T Recommendation H.263; version 1, Nov. 1995; version 2, Jan. 1998; version 3, Nov. 2000.

[4] *ITU-T Rec. H.264 / ISO/IEC 11496-10.* Advanced Video Coding // Final Committee Draft, Document JVT-E022, September 2002.

[5] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification // (ITU-TRec.H.264.ISO/IEC14496-10AVC) Joint Video Team (JVT), Mar. 2003, Doc. JVT-G050.

[6] *B. D. Lucas, T. Kanade* An iterative image registration technique with an application to stereo vision // Proc. of Imaging Understanding Workshop. 1981. — Pp. 121–130

[7] *Гонсалес Р., Вудс Р.* Цифровая обработка изображений // М., Техносфера, 2006, 1072 с.

[8] *Хашин С. И.* Применение методов распознавания образов для сжатия видеоинформации // Докл. всеросс. конф. ММРО-13. — М.: МАКС Пресс, 2007. — С. 420-424.

[9] *Яне Б.* Цифровая обработка изображений // М., Техносфера, 2007, 583 с.