Применение методов распознавания образов для сжатия видеоинформации

Хашин С. И.
khash2@mail.ru
Иваново, ИвГУ

На сегодняшний день имеется большое количество алгоритмов сжатия видеоинформации [1, 2, 3, 4]. Но в последнее время появляется возможность добиться существенного прогресса в этой области. Основой этого являются возросшие вычислительные мощности, позволяющие использовать методы распознавания образов на изображении.

Математические методы

В отличие от других областей сжатия информации, при сжатии видео одной из основных проблем является требуемая вычислительная мощность. Используемые алгоритмы должны ориентироваться на процессоры, имеющиеся в наличии у пользователей, разрабатываемые — ориентироваться на перспективные процессоры. Учитывая довольно большой срок разработки новых алгоритмов и стандартов видеосжатия, можно ожидать, что вычислительная мощность не станет препятствием для их реализации. Поэтому для обработки видеоданных можно применить значительно более мощные и сложные методы, по сравнению с используемыми сегодня.

Во-первых, сегментация изображения. Применяемые на сегодняшний день алгориты разбивают изображение на квадраты или прямоугольники, и для каждого из них находят «движение» (точнее говоря, вектор сдвига) по отношнению к другим кадрам. Вместо этого можно разбивать изображение на сегменты произвольного вида.

Во-вторых, переход от междукадрового «движения» в виде сдвига к движениям более общего вида: сдвигам с поворотом и растяжением, произвольным аффинным и проективным преобразованиям плоскости.

В нынешних алгоритмах видеосжатия каждый рассматриваемый сегмент имеет совсем небольшой размер (обычно не более 16×16 точек), поэтому для него рассмотрение лишь простейших движений в виде сдвига вполне достаточно. Если же мы рассматриваем сегменты произвольного размера, то переход к более общим аффинным движениям неизбежен, а к проективным — возможен.

В третьих, для более эффективного нахождения движений сегементов и их кодирования требуется применить методы распознавания образов. Более конкретно, надо сконструировать «объекты» в виде объединения нескольких сегментов на изображении. Каждый объект не обязательно должен двигаться как единое целое, но движения всех его частей должны

быть связаны друг с другом. Целью данного шага является снижение количества запоминаемых движений, ориентировочно с нескольких сотен до нескольких десятков.

Вычислительно наиболее сложные алгоритмы, например, использующие методы искусственного интеллекта, будут применяться лишь на этапе кодирования, то есть когда нет жесткого ограничения по времени. Процесс же декодирования будет значительно более простым и вполне может уложиться в реальное время.

Для демонстрации эффективности предлагаемого метода видеосжатия надо сосредоточиться в основном на математических подходах и показать их работоспособность хотя бы в простейшей ситуации. Более точно, примем следующие ограничения:

— Не рассматриваем кодирование звука.
— Не реализуем полный кодер/декодер. Ограничиваемся минимально возможным вариантом: кодер сжимает заданную последовательность bmp или jpeg-файлов в один файл, декодер — распаковывает сжатый файл в цепочку bmp-файлов.
— Не ставим задачу построения быстрого кода. Для демонстрационной версии будет достаточно, если и кодер/декодер смогут обрабатывать один кадр за несколько минут (или даже десятков минут).
— Для сжатия «дифференциальных кадров», то есть разностей между кадром-прогнозом и точным кадром используем готовый алгоритм, например jpeg2000.
— Для сжатия «дискретных данных» (в основном, это будут вектора движения, коэффициенты аффинных и проективных преобразований) применяем готовый (свободный) алгоритм, например bzip2.

Схема алгоритма

В алгоритмах сжатия видеоданных принято делить все кодируемые кадры на 3 типа:

— I-кадры (Initial), кодируются независимо от остальных;
— P-кадры (Predicted), кодируются на основании прогноза с помощью «движений» от предыщущих I- и P-кадров;
— и, наконец, B-кадры (Bidirectional), кодируются на основании прогноза от соседних (предыщущих и последующих) I- и P-кадров;

I-кадры имеют наибольший объем, служат для начал декодирования видеоданных. При декодировании на основе I-кадра вычисляется цепочка последующих P-кадров, а затем вычисляются промежуточные B-кадры. Типичная последовательность кадров в видеопотоке такова:

$$\mathrm{IBBPBBPBB}\ldots\mathrm{IBBPBB}\ldots$$

P-кадры обычно имеют объем примерно на порядок меньше, чем I-кадры и B-кадры меньше еще в несколько раз. Использование B-кадров не обязательно.

В начальном варианте нашего алгоритма откажемся от использования B-кадров. Для кодирования начальных (I) кадров будем применять стандартный алгоритм Jpeg2000. Таким образом, остается описать лишь алгоритм кодирования P-кадров.

Кодирование P-кадров состоит из следующих шагов:

— разделение кадра на цветовые сегменты;
— поиск движения каждого найденного сегмента на следующих кадрах;
— выделение объектов, состоящих из сегментов, методами распознавания образов;
— построение «прогноза» следующего кадра по имеющимся сегментам и их движениям;
— добавление к кадру-прогнозу разности, сжатой (после некоторой предварительной обработки) алгоритмом jpeg-2000.

Нахождения цветовых сегментов. Задача сегментации изображений достаточно хорошо изучена. Для ее решения предложено несколько различных подходов. В настоящее время сегменты строятся на основе границ, полученных с помощью алгоритма «Canny edge detector». Ожидаемое количество сегментов — от нескольких десятков до нескольких сотен, типичное значение 200–500.

Нахождение движений. Классические методы нахождения движений (например, алгоритм Лукаса-Канады) дают в ответе лишь «вектор движения», т. е. под «движением» подразумевается лишь сдвиг. Такой подход полностью оправдан, если мы находим движение лишь небольших объектов (как квадрат 8×8). В нашем же случае объект может занимать большую часть кадра и для описания его движения требуются аффинные или даже проективные преобразования. Для этого были разработаны модификации метода Лукаса-Канады, позволяющие находить все требуемые типы движений плоскости.

Распознавание образов. Количество сегментов, а значит и их движений, слишком велико для эффективного их хранения в коде. Поэтому на изображении выделяем «объекты», состоящие из сегментов. Каждый объект либо двигается как единое целое, либо движения его частей зависят друг от друга по некоторому правилу. Поскольку все данные лишь приближенные, точного ответа быть не может, приходится применять некоторый эвристический алгоритм, методы распознавания образов.

Ключевым моментом в этой ситуации является то, что мы имеем численную характеристику качества распознавания. В полном виде это раз-

мер полученного файла. На практике, для достижения приемлемой скорости, этот размер приходится заменять некоторой аппроксимацией.

Другая особенность — дискретность задачи: каждый объект представляется в виде объединения небольшого (первые десятки) количества соседних сегментов. Она же дает возможность «наращивать» объекты, присоединяя на каждом шаге по одному сегменту к некоторому объекту.

Для реализации алгоритма используется стандартная двухуровневая нейронная сеть.

Кодирование движений. В текущей версии алгоритма используются аффинные преобразования плоскости:

$$x' = a_0 + a_1 x + a_2 y,$$
$$y' = a_3 + a_4 x + a_5 y.$$

Описание преобразования с помощью коэффициентов a_i не является оптимальным с точки зрения требуемого количества памяти. В частности, практически невозможно сформулировать требования к точности представления коэффициентов a_i. В разрабатываемом алгоритме аффинное преобразование задается с помощью других шести коэффициентов, зависящих еще и от рассматриваемого сегмента. Более точно, пусть A_0 — центр тяжести сегмента, r — среднеквадратичное расстояние его точек от центра, A_1, A_2 — точки, лежащие на расстоянии r над A_0 и вправо от неё. Тогда аффинное преобразование описывается тремя векторами: вектором сдвига для точки A_0; разницей векторов сдвига для точек A_1 и A_0; разницей вектора сдвига для точки A_2 и его прогноза по движению точек A_0 и A_1. При таком подходе требования к точности всех коэффициентов вполне определенные, и можно ограничиться четырьмя двоичными знаками после запятой.

Завершение кодирования. Результатом предыдущих этапов кодирования являются два потока данных: коэффициенты используемых аффинных преобразований и «дифференциальные кадры», содержащие разность между построенным кадром-прогнозом и точным кадром. В распространенных алгоритмах сжатия видео применяются специализированные алгоритмы, разработанные для сжатия именно таких потоков данных. Однако в разрабатываемом алгоритме, по крайнем мере, в начальной версии, для этого применяются стандартные универсальные алгоритмы. А именно, для сжатия дифференциальных кадров применяется Jpeg2000, для сжатия данных о движении — bzip2.

Важнейшим моментом является то, что данные о сегментации не требуется помещать в код видеофайла. Декодер может сам воспроизвести ту же сегментацию, что и кодер. Именно это обстоятельсто и позволяет получить преимущество перед другими методами кодирования видео.

Полученные результаты. Было проведено сравнение результатов сжатия видеоинформации предлагаемым методом с наиболее мощным алгоритмом из распространенных на сегодняшний день — H264 на искусственных кадрах. Кадры представляют собой движущийся фон, по которому в свою очередь двигаются 4–5 спрайтов, перекрывая друг друга. В этой исскуственной ситуации, наиболее выигрышной для нашего алгоритма, при высоком качестве изображения мы получаем размер дифференциальных кадров вместе с данными о движениях в 2–3 раза (в зависимости от заданного уровня шума) меньше, чем у алгоритма H264 при том же уровне шума.

Разумеется, такое соотношение не может сохраняться для реальных видеоданных. Однако оно показывает предел возможностей рассматриваемого метода сжатия, его потенциал.

Работа выполнена при поддержке РФФИ, проект № 07-07-00178.

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; version 1, Nov. 1995; version 2, Jan. 1998; version 3, Nov. 2000.

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

[4] 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.