ИНФОРМАТИКА

2006 **апрель-июнь** **№ 2**

УДК 528.2

Б.А. Залесский, А.И. Кравчонок

ОТСЛЕЖИВАНИЕ ДИНАМИЧЕСКИХ ОБЪЕКТОВ И ИХ РАСПОЗНАВАНИЕ С ПОМОЩЬЮ ГРАФОВЫХ АЛГОРИТМОВ

Предлагаются методы и алгоритмы отслеживания и распознавания объектов на цветных видеопоследовательностях, снятых стационарной видеокамерой. Разработанные алгоритмы дают возможность отслеживать и распознавать динамические объекты в режиме реального времени. Использование цветных изображений позволяет повысить качество решения задачи. Выделение, сегментация и отслеживание объектов осуществляются с помощью кластерных представлений объектов.

Для распознавания объектов описываются алгоритмы сравнения плоских укладок графов. В отличие от классической задачи поиска изоморфных подграфов заданного графа, задача сравнения укладок является полиномиально разрешимой. Сравнение укладок графов, описывающих динамические объекты, занимает незначительную часть времени, нужного для обработки каждого кадра видеопоследовательности.

Введение

Область обработки изображений, связанная с обнаружением, отслеживанием и распознаванием объектов на видеопоследовательностях, интенсивно развивается на протяжении последних двух десятилетий. Большое число известных в настоящее время алгоритмов может быть условно разделено на три основных класса: алгоритмы, основанные на представлении последовательности изображений видеопотоком; байесовские алгоритмы и комбинированные алгоритмы, в которых использованы одновременно модель видеопотока и байесовский подход. Наиболее известные алгоритмы первого типа – алгоритм Лукаса – Канаде [1] и алгоритм Ши – Томаши [2]. Ко второму типу относятся, например, алгоритмы, основанные на применении фильтра Калмана [3]. Популярные в настоящее время алгоритмы конденсации Изарда – Блейка [4] и другие [5, 6] используют одновременно представление последовательности изображений в виде видеопотока и байесовскую модель его возможных изменений.

В статье [7] были предложены алгоритмы обнаружения и отслеживания динамических объектов видеопоследовательностей полутоновых изображений. В настоящей работе разработаны *методы обнаружения и отслеживания динамических объектов на цветных видеопоследовательностях*, основанные на *кластерных представлениях* изображений. Кластеризация цветных изображений проводилась с помощью фонового кадра и методов наращивания областей с одновременным вычислением средних характеристик кластера на каждом шаге. Это позволило получить устойчивое обнаружение динамических объектов цветных видеопоследовательностей, а также их отслеживание в режиме реального времени так, что обработка каждого кадра заняла менее 0,04 с. Представлены также новые *алгоритмы распознавания полутоновых и цветных изображений*, основанные на сравнении плоских укладок графов, которые описывают кластеризованные оценки этих изображений (напомним, что графы, описывающие 2D-изображения, являются планарными). В отличие от классической NP-трудной задачи нахождения изоморфных подграфов двух графов *задача сравнения плоских укладок двух раскрашенных графов* может быть сформулирована как разрешимая за полиномиальное время. Приведенная ниже формулировка этой задачи ориентирована специально на распознавание объектов по их 2D-изображениям. Различные версии алгоритмов сравнения плоских укладок двух графов требуют в худших случаях от $O(n^3)$ до $O(n^4)$ операций, где n – число вершин большего графа, хотя для большинства практических задач распознавания число операций $O(n)$. В частности, линейное время необходимо для сравнения плоских укладок графов с индивидуальной раскраской вершин или графов, не содержащих большого числа повторяющихся подграфов. Построенные алгоритмы схожи с алгоритмом сравнения изображений объектов путем вычисления максимума коэффициента корреляции между одним из них и всевозможными сдвигами и поворотами второго.

Предложенные алгоритмы были реализованы на языке C++ и протестированы на видеопоследовательностях, полученных цифровой видеокамерой и камерами видеонаблюдения (видеопоследовательности размещены на сайте http://homepages.inf.ed.ac.uk/rbf/CAVIAR).

1. Кластерный алгоритм отслеживания объектов

Одно из основных требований к алгоритмам обнаружения, отслеживания и распознавания объектов на полутоновых и цветных видеопоследовательностях – высокое быстродействие, достаточное для обработки каждого кадра в режиме реального времени. Например, при стандартной скорости видеопоследовательности 24–25 кадр/с время работы с каждым кадром не должно превышать $\approx$ 0,04 с, а при скорости 30 кадр/с – 0,03 с.

Предложенные алгоритмы в совокупности представляют собой набор действий, достаточных для обнаружения, отслеживания и распознавания объектов цветных видеопоследовательностей за отведенное для этого время.

1.1. Обозначения и определения

Пусть $S=\{0,1,\ldots,N-1\}\times\{0,1,\ldots,N-1\}$ – множество 2D-точек $\mathbf{j}\in S$ с целочисленными координатами – пиксели $N\times N$ *цветного изображения* $\mathbf{I}$, представляющего собой $N\times N$ матрицу с элементами-векторами $I_j=\left(I_{r,j},I_{g,j},I_{b,j}\right)\in\{0,1,\ldots,255\}^3$ из RGB-пространства. Вместо RGB можно использовать другие цветовые пространства, хотя тестирование алгоритмов показало, что применение RGB позволяет получать решения задач обнаружения, отслеживания и распознавания динамических объектов удовлетворительного качества.

Предлагаемый метод не зависит от выбранной системы окрестностей, задающей соседство на множестве пикселей изображения, однако в дальнейшем для простоты будем предполагать, что система окрестностей на множестве S содержит либо 4, либо 8 ближайших пикселей. Термин *кластер* понимается как максимальное относительно включения связное в выбранной системе окрестностей множество пикселей $A\subset S$, обладающих определенным общим свойством, например значениями I_j из наперед заданного прямоугольника, или иным. Предполагается, что различные кластеры образуют разбиение S, т. е. $A_i\cap A_j=\varnothing,\ i\neq j$ и $S=\cup A_i$.

Теоретически два кластера *имеют общую границу*, если у них есть хотя бы два соседних пикселя, лежащих в разных кластерах. На практике могут быть использованы другие определения соседства. Например, в задачах отслеживания и распознавания кластеры A_i, A_j могут рассматриваться как соседи, если они имеют больше наперед заданного числа пар соседних пикселей $\{\mathbf{k}_l,\mathbf{m}_l\},\ \mathbf{k}_l\in A_i,\mathbf{m}_l\in A_j$. В других случаях соседними считаются A_i, A_j, лежащие на расстоянии друг от друга меньше наперед заданного.

Последовательность изображений обозначим $\mathbf{I}_t$, $t=1,2,\ldots$. Зависимость кластеров текущего изображения от t будет опускаться.

1.2. Алгоритм отслеживания динамических объектов

Задача отслеживания объектов на последовательности изображений решается путем выполнения предлагаемого алгоритма (рис. 1). Результаты выполнения каждого этапа такого алгоритма показаны на рис. 2.

В данной работе не рассматривается проблема покадрового ввода цветных изображений. Отметим лишь, что имеющиеся в настоящее время технические средства (обычно это цифровая видеокамера либо аналоговая видеокамера и устройство видеозахвата, например специализированная плата, преобразующая аналоговый сигнал в цифровую форму или TV-тюнер) позволяют поставлять программе кадры цветных изображений за приемлемое время.

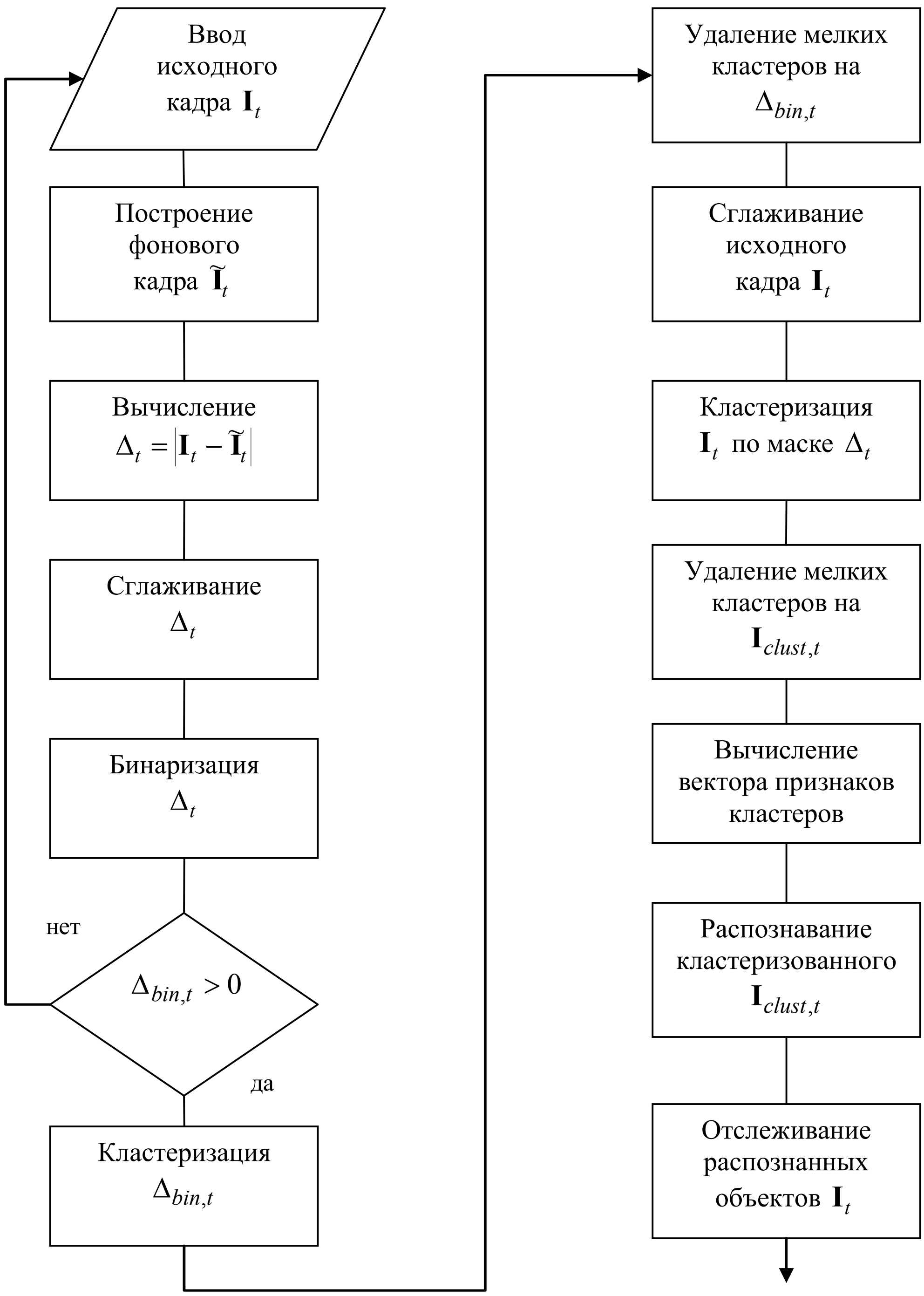


Рис. 1. Алгоритм отслеживания и распознавания объектов

Для *построения фонового изображения* были изучены два способа, которые с практической точки зрения дали приблизительно одинаковые результаты за одинаковое время.

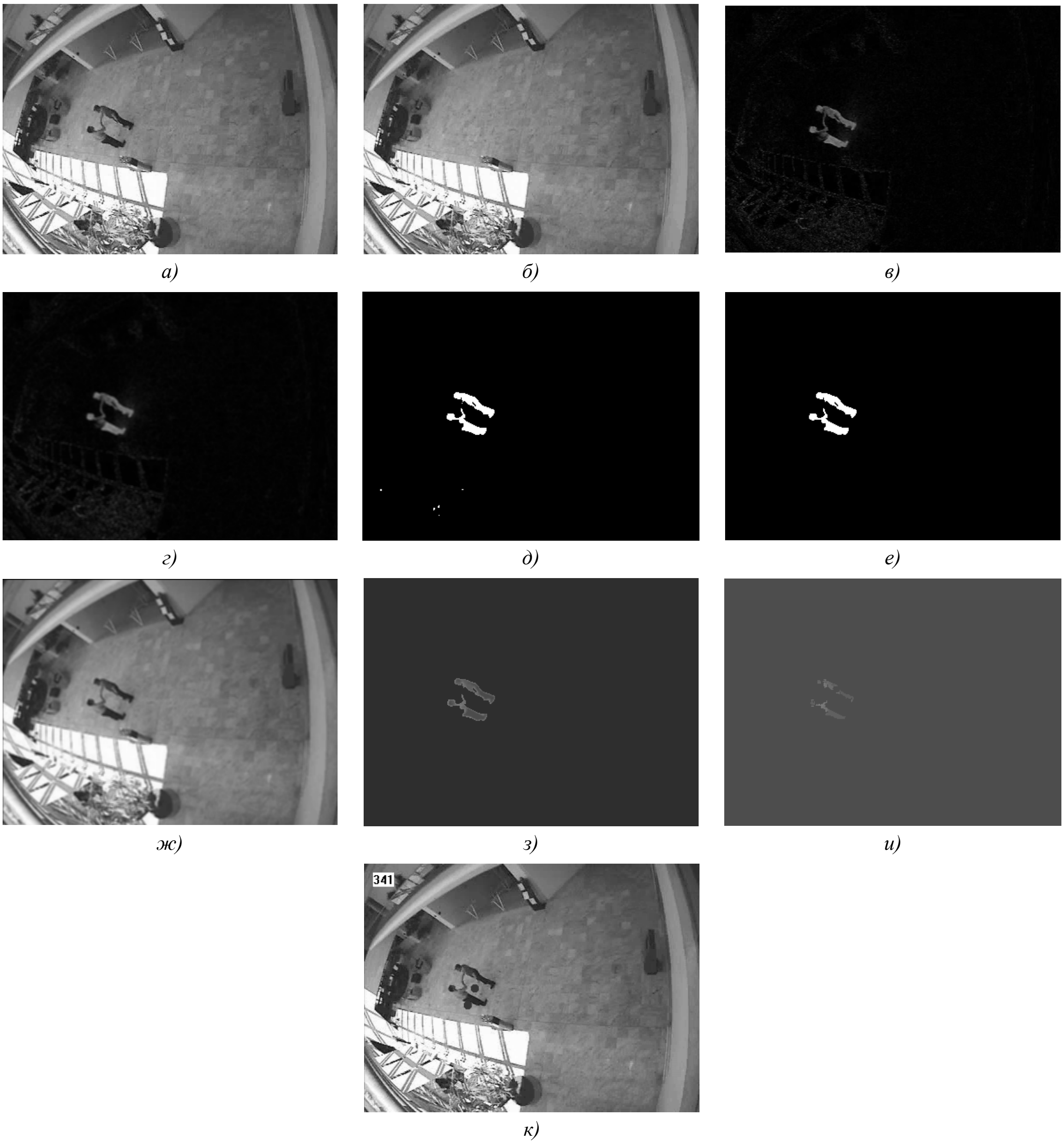


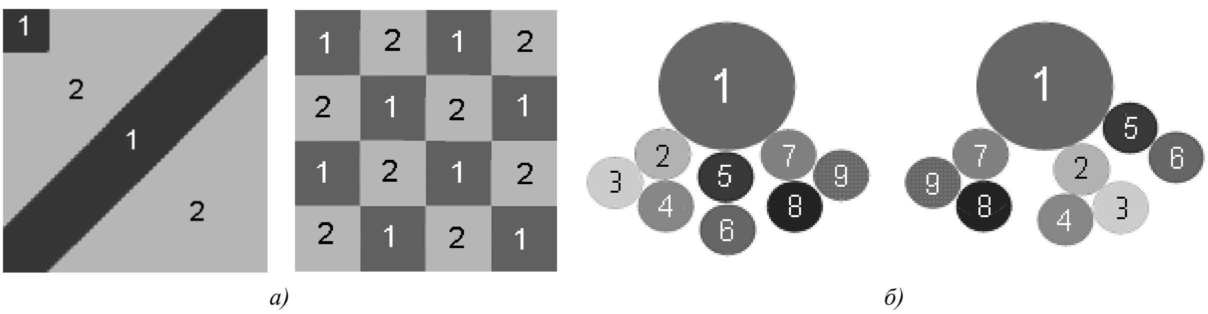
Рис. 2. Этапы алгоритма отслеживания объектов: *а)* исходный кадр $\mathbf{I}_t$; *б)* фоновый кадр $\tilde{\mathbf{I}}_t$; *в)* разность между текущим и фоновым кадрами Δ_t ; *г)* сглаженная разность Δ_t ; *д)* бинаризированная сглаженная разность $\Delta_{bin,t}$; *е)* $\Delta_{bin,t}$ с удаленными мелкими кластерами; *ж)* сглаженный кадр $\mathbf{I}_t$; *з)* кластеризованный по маске $\Delta_{bin,t}$ кадр $\mathbf{I}_t$; *и)* кластеризованный по маске $\Delta_{bin,t}$ кадр $\mathbf{I}_t$ с удаленными мелкими кластерами; *к)* распознавание объектов

Первый способ, реализованный в пакете OpenCV, использует обычное среднее арифметическое последовательности изображений $\tilde{\mathbf{I}}_t = \frac{1}{k+1}\sum_{l=0}^{k}\mathbf{I}_{t-l}$, второй – более сложный – строит фоновый кадр, присоединяя к фоновому изображению пиксели, которые не меняют цвет на нескольких $\mathbf{I}_t, \mathbf{I}_{t-1}, \ldots, \mathbf{I}_{t-k}$.

Затем вычисляется *модуль разности текущего и фонового изображений* $\Delta_t = \left|\mathbf{I}_t - \widetilde{\mathbf{I}}_t\right|$ – полутоновое изображение, которое на следующем шаге сглаживается с помощью обычного оконного осредняющего фильтра. Лучшее качество обработки дало бы применение медианного фильтра вместо осредняющего, однако в этом случае время обработки каждого кадра возрастает до недопустимых пределов. Сглаженное изображение Δ_t бинаризуется с помощью порогового алгоритма. Порог выбирается эмпирически. Точность его задания не имеет существенного значения, так как бинарное изображение используется в качестве маски для сегментации динамических объектов цветной видеопоследовательности. После бинаризации сглаженной разности Δ_t и проверки условия $\Delta_{bin,t} > 0$ наличия на $\mathbf{I}_t$ динамического изображения вычисляются характеристики кластеров на бинарном изображении $\Delta_{bin,t}$. Для этого используется вариант алгоритма наращивания областей, в котором на каждом шаге к уже найденной части кластера присоединяются не просмотренные ранее соседние пиксели белого цвета. Просмотренные пиксели помечаются в специальном массиве меткой соответствующего кластера, показывающей, что пиксель уже просмотрен. Одновременно составляется список кластеров. Если на $\Delta_{bin,t}$ присутствуют мелкие кластеры, они удаляются на основе списка, полученного на предыдущем шаге. Обработанное таким способом бинарное изображение $\Delta_{bin,t}$ используется как маска для *сглаживания* и *кластеризации* исходного цветного изображения $\mathbf{I}_t$. Применение маски позволяет сократить время, необходимое для выполнения двух вышеуказанных операций. На исходном изображении $\mathbf{I}_t$ всегда присутствует шум, поэтому сглаживание существенно улучшает качество выполнения последующих операций. Как и при сглаживании полутонового изображения Δ_t, для сглаживания $\mathbf{I}_t$ применяется осредняющий оконный фильтр.

Кластеризация цветного изображения $\mathbf{I}_t$ производится только в той его части, где $\Delta_{bin,t} > 0$. Для ее выполнения используется вариант алгоритма наращивания областей, который на каждом шаге добавляет к уже сформированной части кластера соседние с ним точки при условии, что разность цвета кластера и цвета точки меньше установленного порога. После завершения кластеризации $\mathbf{I}_t$ с него удаляются мелкие кластеры. *Признаки кластеров* (такие, как размер, граница и ее длина, различные моменты, соседство с другими кластерами и т. д.) находятся на этапе выполнения алгоритма наращивания областей.

Распознавание изображений динамических объектов – в общем случае наиболее трудная с теоретической точки зрения часть алгоритма. Класс цветных изображений очень велик. Разные его представители имеют разную структуру. К тому же объекты на изображениях могут менять свою форму или/и размеры, быть частично закрыты другими объектами.



а) *б)*

Рис. 3. Примеры разных изображений, имеющих одинаковые гистограммы

Применение простых алгоритмов, таких, например, как сравнение гистограмм изображений объектов или вычисление их корреляции может приводить либо к существенным ошибкам, либо к большим вычислительным затратам. Различные изображения могут иметь одинаковые гистограммы цвета (рис. 3, *а*).

В предложенном алгоритме распознавание объектов осуществляется на основе сравнения графов, описывающих кластеризованные изображения $\mathbf{I}_{clust,t}$. Это делается путем подсчета совпадающих по цвету вершин на графах двух изображений или ребер одинакового цвета либо путем подсчета вершин, совпадающих вместе со своими окрестностями.

Для решения более сложных задач распознавания изображений приведем ниже алгоритм распознавания цветных изображений, основанный на сравнении плоских укладок раскрашенных графов.

В силу того что обработка изображений производится покадрово в режиме реального времени (25–30 кадр/с), как правило, изображения объекта на соседних кадрах изменяются незначительно. Графы, описывающие эти изображения объекта, также отличаются незначительно. Это дает возможность сравнить их и модифицировать эталонный граф, описывающий объект в данный момент времени.

2. Алгоритм сравнения плоских укладок графов для распознавания изображений

Известно, что в общем случае задача поиска в графе подграфа, изоморфного заданному, является NP-трудной [8], требующей выполнения экспоненциального числа операций от числа вершин. Более того, она является одной из наиболее трудных алгоритмических задач теории графов. Это означает, что в общем случае практически невозможно решить задачу распознавания изображений за малое время путем нахождения изоморфных подграфов в графах кластеризованных изображений. Однако специфика задачи распознавания изображений позволяет сформулировать ее таким образом, что существует эффективное решение, требующее выполнения от $O(n^3)$ до $O(n^4)$ операций от числа вершин n большего графа. При этом оказывается, что для большинства задач распознавания изображений число операций, нужных для сравнения описывающих их графов, практически равно $O(n)$. Таковыми являются, например, задачи сравнения графов с индивидуальной раскраской вершин или ребер графов, не содержащих большого числа изоморфных подграфов.

2.1. Постановка задачи сравнения графов изображений

Пусть $G=(E,V)$ – граф, представляющий кластеризованное изображение, который в дальнейшем будет называться *графом изображения*. Помеченные вершины графа соответствуют кластерам $\mathbf{I}_{clust}$, а ребра графа обозначают соседство кластеров на $\mathbf{I}_{clust}$ (напомним, что разные варианты определения соседства кластеров приведены во введении). Метки вершин будем обозначать λ_i. В общем случае метка λ_i вершины $e_i \in E$ – это вектор признаков кластера. Так как метки λ_i вершин представляют собой векторы (и к тому же их конечное число), при необходимости они могут быть упорядочены относительно какого-либо порядка $\leq$. Это может быть лексикографический или любой иной порядок. В задачах распознавания изображений ребра графа тоже могут быть помечены, например, длиной общей границы соседних кластеров или числом, зависящим от расстояния между ними. Однако далее для простоты изложения будем рассматривать только графы с непомеченными ребрами и с вершинами, помеченными числами из конечного фиксированного множества. Способы построения графов, описывающих изображения, и задания меток вершин этих графов приведены в работе [7], поэтому в настоящей статье этот вопрос не рассматривается. Заметим лишь, что множество меток должно выбираться достаточно аккуратно, чтобы соответствовать требованиям решаемой задачи. Для наглядности можно представить себе, что вершины графов помечены цветом. Это тем более имеет смысл в задачах распознавания цветных изображений.

Рассмотрим теперь особенности задачи сравнения графов, описывающих 2D-изображения. Во-первых, такие графы планарны. Во-вторых, каждая пара вершин графа, соединенных ребром, обязательно имеет разные метки. В-третьих, изоморфные в традиционном смысле графы в общем случае представляют несовпадающие или непохожие изображения, даже если кластеры, соответствующие совпадающим вершинам графов, тоже совпадают. На рис 3, *б* приведены два несовпадающих изображения, графы которых изоморфны. Напомним также о трудностях, которые возникают при сравнении оригинального и зеркально отраженного изображений. Часто человек не распознает их как изображения одного объекта, хотя такие изображения имеют изоморфные графы, причем с не самым сложным типом изоморфизма. Помимо этого, вспомним, что при распознавании на самом деле сравниваются проективные преобразования объектов путем их мысленного совмещения или мысленного совмещения деталей объектов. В силу вышесказанного сформулируем *задачу сравнения графов изображений как задачу сравнения их плоских укладок*. Напомним, что плоской укладкой графа называется изображение графа на плоскости, в котором его вершины изображены точками плоскости, а ребра – плоскими жордановыми кривыми, пересекающимися только в вершинах [8]. По построению графы 2D-изображений являются плоскими укладками. На рис. 4, *а*, *б* показаны две разные плоские укладки одного и того же графа. Нетрудно убедиться, что соответствующие им изображения воспринимаются как различные (рис. 4, *е*, *ж*). Поэтому определение совпадающих укладок формулируется следующим образом.

О п р е д е л е н и е 1. *Будем говорить, что две плоские укладки совпадают, если одну из них можно преобразовать во вторую, передвигая по плоскости вершины первой, не пересекая при этом ни одного ее ребра.*

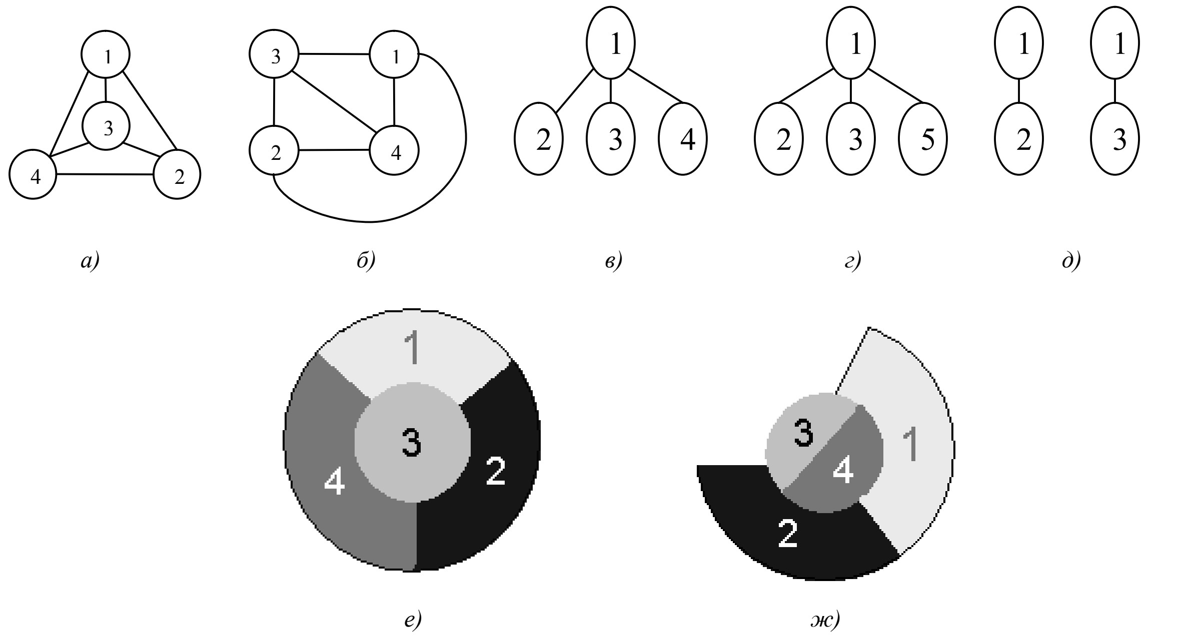


Рис. 4 Примеры различных укладок графа и их изображений

Заметим, что при таком определении графы, представляющие изображение и его зеркальное отражение, в общем случае не совпадают. Однако распознавание зеркально отраженных изображений можно проводить на основе определения 1, сравнивая укладку первого графа с зеркально отраженной укладкой второго.

С точки зрения распознавания изображений наиболее подходящими являются следующие алгоритмы сравнения плоских укладок: нахождения максимальной по числу вершин связной общей укладки, содержащейся во всех сравниваемых укладках, и нахождения всех связных укладок, содержащихся во всех сравниваемых укладках. Первый алгоритм больше подходит для сравнения динамических объектов, а второй – для сравнения протяженных составных изображений, содержащих несколько совпадающих объектов. Примерами составных изображений являются карты и 3D-рельефы. В данной работе рассмотрим алгоритм, который находит так называемую точную максимальную общую укладку двух укладок.

О п р е д е л е н и е 2. *Точной общей укладкой назовем такую укладку, у которой метка каждой вершины совпадает с меткой соответствующих ей вершин исходных укладок и, кроме того, метки всех соседних с ней вершин совпадают с метками соответствующих соседних вершин исходных укладок.*

Например, укладки на рис. 4, *в*, *г* имеют две общие точные подукладки, изображенные на рис. 4, *д*.

2.2. Алгоритм нахождения максимальной точной подукладки

Алгоритм нахождения максимальной точной подукладки двух укладок сравнительно прост. При его описании будем считать, что число вершин первой укладки $n_1 = |E_1|$ не превосходит числа вершин $n_2 = |E_2|$ второй укладки и что число разных меток G_1 равно K. Для произвольной укладки Q будем обозначать ее вершины $E(Q)$, а ребра $V(Q)$. Алгоритм состоит из следующих шагов.

Шаг 1 (предварительный). Составить списки вершин $E_{1,i}$ $(i = 1 \div K)$ первой (меньшей) укладки G_1, которые имеют одинаковые метки λ_i. Одновременно составить списки вершин второй укладки $E_{2,i}$ $(i = 1 \div K)$, которые имеют одинаковые метки λ_i, принадлежащие множеству меток первой укладки.

Шаг 2 (предварительный). Присвоить значение переменной текущего номера списка $\ell := 0$ счетчику числа вершин текущей общей укладки $J := 0$ и максимальной общей укладки $J_{\max} := 0$.

Шаг 3 (инициализация). Присвоить номер списка $\ell := \ell + 1$. Если $\ell > K$, идти на шаг 12, иначе пометить списки $E_{1,\ell}$ и $E_{2,\ell}$ как целиком не просмотренные. Выбрать начальный элемент для сравнения укладок $e_{beg,1}$ из списка $E_{1,\ell}$ и пометить его в списках $E_{1,\ell}$ как просмотренный.

Шаг 4 (инициализация). Если все элементы списков $E_{1,\ell}$ и $E_{2,\ell}$ просмотрены, идти на шаг 3.

Шаг 5 (инициализация). Если в списке $E_{2,\ell}$ есть непросмотренные элементы, выбрать начальный элемент для сравнения укладок $e_{beg,2}$ и пометить его в $E_{2,\ell}$ как просмотренный.

Шаг 6 (инициализация). Если в списке $E_{2,\ell}$ нет непросмотренных элементов, а в $E_{1,\ell}$ есть, выбрать начальный элемент списка $e_{beg,1}$ в $E_{1,\ell}$ из числа непросмотренных. Пометить его как просмотренный, а все элементы списка $E_{2,\ell}$ как непросмотренные. Выбрать начальный элемент для сравнения укладок $e_{beg,2}$ и пометить его в $E_{2,\ell}$ как просмотренный.

Шаг 7 (сравнение начальных вершин). Сравнить окрестности $O_{e_{beg,1}}$ и $O_{e_{beg,2}}$. Если $O_{e_{beg,1}} \neq O_{e_{beg,2}}$, повернуть $O_{e_{beg,2}}$ до первого совпадения с $O_{e_{beg,1}}$. Если для всех поворотов $O_{e_{beg,2}}$ относительно $O_{e_{beg,1}}$ совпадения нет, присвоить значение счетчику числа вершин максимальной общей укладки $J := 1$, $J_{\max} := \max\{J, J_{\max}\}$ и идти на шаг 4.

Шаг 8 (начало построения общей укладки). Положить общую укладку

$$G_{com} := \left(E(O_{e_{beg,1}}), V(O_{e_{beg,1}})\right),$$

а также список активных вершин для просмотра $A_1 := O_{e_{beg,1}} \setminus e_{beg,1}$, $A_2 := O_{e_{beg,2}} \setminus e_{beg,2}$ (здесь и далее считаем, что окрестность O_e содержит e).

Шаг 9 (построение общей укладки). Если A_1 и A_2 пусты, присвоить $J := |G_{com}|$, $J_{\max} := \max\{J, J_{\max}\}$ и идти на шаг 7. Если списки активных вершин A_1 и A_2 не пусты, выбрать произвольную вершину $e_{i,1} \in A_1$ и соответствующую ей вершину $e_{i,2} \in A_2$. Удалить их из списка активных вершин $A_1 := A_1 \setminus e_{i,1}$ и $A_2 := A_2 \setminus e_{i,2}$.

Шаг 10 (построение общей укладки). Сравнить окрестности $O_{e_{i,1}}$ и, если они совпадают, положить

$$G_{com} := \left(E(G_{com}) \bigcup E(O_{e_{i,1}}), V(G_{com}) \bigcup V(O_{e_{i,1}}) \right),$$

а также $A_1 := A_1 \bigcup O_{e_{i,1}}$, $A_2 := A_2 \bigcup O_{e_{i,2}}$.

Шаг 11 (построение общей укладки). Идти на шаг 9.

Шаг 12. STOP.

Заметим, что при выполнении шага 10 нужно сравнивать вершины окрестностей с учетом их упорядоченности относительно родительских вершин $e_{i,1}, e_{i,2}$. Это позволит провести сравнение всех окрестностей $O_{e_{j,1}}$ и $O_{e_{j,2}}$ в худшем случае за $O(n_1)$ операций.

Оценим трудоемкость алгоритма нахождения максимальной точной подукладки исходя из оценок трудоемкости каждого шага. Шаг 1 может быть выполнен, по крайней мере, двумя способами. Если составлять списки $E_{1,i}, E_{2,i}$ $(i = 1 \div K)$ одинаково помеченных вершин укладок, непосредственно просматривая множества всех вершин E_1, E_2 двух укладок, потребуется в общем случае $O(n_1^2)$ операций для первой укладки и $O(n_1 n_2)$ для второй. Если множества вершин укладок подчинены какому-либо порядку, их можно предварительно упорядочить, потратив соответственно $O(n_1 \ln(n_1))$ и $O(n_2 \ln(n_2))$ операций. При этом число операций, нужных для составления списков $E_{1,i}, E_{2,i}$ $(i = 1 \div K)$, составит соответственно $O(n_1)$ и $O(n_2)$. Таким образом, в случае большого числа вершин, которые подчиняются какому-либо порядку, эффективнее будет работать с упорядоченными списками. Шаг 3 дает множитель $O\left(|E_{1,\ell}| |E_{2,\ell}| \right)$ для числа операций, нужных для сравнения двух укладок, начиная с вершин, принадлежащих этим классам. Для выполнения шага 7 нужно не более $O(n_1 n_2)$ операций, а для шагов 8–10 – не более $O(n_1)$. Следовательно, выполнение алгоритма в худшем случае потребует $O(n_1 n_2^2)$ операций для вполне упорядоченных множеств вершин укладок и $O(n_1^2 n_2^2)$ операций, если множества вершин укладок неупорядочены.

Как уже отмечалось выше, во многих задачах практического распознавания изображений с помощью предложенного алгоритма число требуемых операций оказывается равным $O(n_1)$ в силу того, что большинство вершин имеет индивидуальные метки. Наибольшее число операций $O(n_1^2 n_2^2)$ требуется при распознавании объектов, которые обладают периодической структурой, таких, например, как «шахматная доска» на рис. 3, *а*. Однако даже в этом случае сравнение изображений занимает малое время, так как обычно число вершин в графах, описывающих изображения, невелико. Оно значительно меньше, чем число операций, нужных для вычисления корреляционных матриц изображений.

Дальнейшим этапом совершенствования алгоритма является построение «неточного» сравнения двух укладок, в котором общими считаются одинаково помеченные вершины с частично совпадающими окрестностями.

Заключение

Предложены новые алгоритмы отслеживания и распознавания 2D-изображений объектов цветных видеопоследовательностей, использующие их кластерные представления. Разработанные алгоритмы дают возможность отслеживать и распознавать динамические объекты в режиме реального времени. Применение цветных изображений позволяет повысить качество решения задачи.

Для распознавания изображений предложены эффективные алгоритмы сравнения плоских укладок графов, требующие выполнения $O(n_1^2 n_2^2)$ операций для двух укладок с числом вершин n_1 и n_2 соответственно. Сравнение укладок графов, которые описывают динамические объекты, занимает незначительную часть времени, нужного для обработки каждого кадра видеопоследовательности.

Список литературы

1. Lucas, B.D. An iterative image registration technique with an application to stereo vision / B.D. Lucas, T. Kanade // Proc. International Joint Conference on Artificial Intelligence. – Vancouver, Canada, 1981. – P. 674–679.

2. Shi, J. Good features to track / J. Shi, C. Tomasi // IEEE Conference on Compurer Vision and Pattern Recognition (CVPR94). – Seattle, 1994. – P. 593–600.

3. Matthies, L.H. Kalman filter-based algorithms for estimating depth from image sequences / L.H. Matthies, T. Kanade, R. Szeliski // Int. Journal of Computer Vision. – Vol. 3. – 1989. – P. 209–236.

4. Isard, M. Condensation – conditional density propagation for visual tracking / M. Isard, A. Blake // Int. Journal of Computer Vision. – 1998. – № 29(1). – P. 5–28.

5. Isard, M. BraMBLe: A Bayesian Multiple-Blob Tracker / M. Isard, J. MacCormick // International Conference on Computer Vision. – Vancouver, Canada, 2001. – Vol. 2. – P. 34–41.

6. A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking / S. Arulampalam [et al.] // IEEE Transactions on signal processing. – V. 50. – № 2. – 2001. – P. 174–188.

7. Залесский, Б.А. Отслеживание и распознавание движущихся объектов на основе их кластерного представления / Б.А. Залесский, А.И. Кравчонок // Информатика. – № 2. – 2004. – С. 68–78.

8. Лекции по теории графов / В. А. Емеличев [и др.] – М.: Наука, 1990.

Поступила 10.01.06

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: alpha_storm@mail.ru

B.A. Zalesky, A.I. Kravchonok

TRACKING DYNAMICAL OBJECTS AND THEIR RECOGNITION BY GRAPH ALGORITHMS

New algorithms to track and recognize 2D images of objects in color videosequences were developed. These algorithms are based on cluster representations of images. Also, efficient algorithms to compare planar embeddings of graphs are proposed. They allow tracking and recognition of dynamical objects in real-time mode.