Институт прикладной математики имени М. В. Келдыша
Российской академии наук

На правах рукописи

Синдеев Михаил Сергеевич

Исследование и разработка алгоритмов матирования видеопоследовательности

Специальность 05.13.11 – математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Диссертация на соискание учёной степени
кандидата физико-математических наук

Москва – 2013

СОДЕРЖАНИЕ

Данная работа посвящена задаче матирования – выделения объектов в изображении или видеопоследовательности с целью монтажа, т.е. последующего наложения объекта на новый фон. Решение задачи матирования заключается в вычислении маски прозрачности, называемой «альфа-каналом», и цвета каждого пиксела объекта. Критерием качества матирования является незаметность монтажа для зрителя.

В настоящее время для выделения объектов в видеопоследовательности используется процедура «ротоскопирования» ([22], [61], [83], [84]), требующая ручного построения контура объекта. Предлагается новый метод вычисления оптического потока в альфа-канале для автоматизации задачи матирования. Метод основан на совместном нахождении оптического потока и маски прозрачности итерационной оптимизацией. Предложен алгоритм быстрого вычисления оптического потока и алгоритм матирования видео при фиксированном потоке с учетом перекрытий. Для сегментации ключевых кадров предложен алгоритм вычисления прозрачности на основе байесовского подхода.

ВВЕДЕНИЕ

Матированием называется отделение объекта переднего плана от фона на изображении. При этом нужно для каждого пикселя получить цвет и значение прозрачности. Затем извлеченный объект можно наложить на другой фон или применить к нему какую-либо обработку, например, цветокоррекцию.

Предположим, что исходное изображение I является смесью двух изображений F и B (объект переднего плана и фон) с каналом прозрачности α. В каждом пикселе должно удовлетворяться следующее уравнение:

$$I = \alpha F + (1-\alpha)B, \tag{1}$$

где I, F и B – трехмерные векторы в цветовом пространстве RGB, $0 \le \alpha \le 1$. Задача состоит в получении α, F и иногда B по заданному исходному изображению I с использованием какого-либо дополнительного ввода со стороны пользователя (без этого невозможно определить, какой объект требуется выделить). На рис. 1 показан пример такого разложения.

Следует заметить, что если два из трех значений (F, B, α) известны, то третье может быть однозначно посчитано. Задача является сильно недоопределенной, т.к. для каждого цвета исходного изображения I существует бесконечное число комбинаций цветов объекта и фона. Для того, чтобы сделать её решаемой, требуется некоторая регуляризация.

Рисунок 1. Уравнение смешивания

Задача матирования возникла в художественной фотографии довольно давно, и долгое время решалась трудоемкими аналоговыми методами. Широкое применение матирование нашло в кино: неподвижные или подвижные (покадровые) маски («маты») объекта переднего плана рисовались на стеклянных панелях и предотвращали экспонирование фона на пленку, куда затем отдельным проходом экспонировался новый фон на основе инвертированной маски. Сведение слоев могло также осуществляться оптическими способами [25], например, проецированием через полупрозрачное зеркало.

Из-за сложностей в создании покадровых масок кинематографисты часто ограничивались неподвижными масками, заведомо захватывающими область перемещения объектов переднего плана (чаще всего – актеров), при этом фон за ними брался из реальной сцены, а дорисовка фона был ограничена областью, в которую объекты переднего плана заведомо не попадали. Появление цифрового видео сделало возможным произвольные манипуляции с изображением, ранее недоступные при работе с аналоговым представлением видеосигнала.

Матирование занимает важное место в профессиональной обработке видео и кинопроизводстве и применяется для замены/модификации фона (см. рис. 2), цветокоррекции отдельных объектов, а также для преобразования видео в стереоскопический (3D) формат.

Помимо визуальных эффектов в видео, потенциальной областью применения является дополненная реальность. Существующие технологии позволяют дополнять видеопоток синтетическими объектами в реальном времени, однако возможности по бесшовному совмещению этих объектов с реальными ограничены – они, как правило, просто накладываются на входное изображение, в то время как желательно обрабатывать перекрытия искусственных объектов реальными.

Задача матирования видео заключается в выделении объекта переднего плана из видеопоследовательности с построением карты прозрачности, которая позволяет учитывать такие эффекты, как

Рисунок 2. Пример матирования кадра из видеопоследовательности

- естественная прозрачность объекта (стекло, тонкая ткань)
- особенности пространственной дискретизации (размытие краев объектов)
- особенности временно́й дискретизации (размытие движения).

В настоящее время задача матирования частично решена для некоторых случаев: выделение объекта на фоне константного цвета, выделение объектов с размытыми краями при известной функции рассеяния точки [42], построение мягкой границы для объекта, заданного бинарной маской и тернарной разметкой (для выделения мелких деталей, волос и т.д.) [33], [44].

В данной работе рассматривается относительно общая постановка задачи: видео может быть произвольным (особых способов съемки не требуется). В качестве входной разметки предлагается использовать два ключевых кадра (первый и последний кадр видео) с полностью заданным альфа-каналом, что образует граничное условие в видеообъеме. При этом сами ключевые кадры можно размечать с помощью существующих алгоритмов матирования изображений. Такой подход позволяет точно выделить желаемый объект, при условии, что он не сильно изменяет форму между ключевыми кадрами. В противном случае, требуется увеличение ключевых кадров – либо добавление частичных ключевых кадров, либо создание новых полных ключевых кадров. Во втором случае видео разбивается ключевыми кадрами на набор фрагментов, каждый из которых имеет два ключевых кадра (в нача-

ле и в конце), что совпадает с изначальной постановкой задачи, т.е. такой случай не надо обрабатывать отдельно.

Чтобы сделать задачу решаемой и ввести объективные критерии качества, сузим класс рассматриваемых видеопоследовательностей. Пусть исходное видео представляет собой непрерывную сцену длиной до 10 секунд с разрешением от 320x240 до 1280x720[1]. И хотя длительность не является абсолютной величиной (т.к. зависит от частоты кадров), уменьшать частоту кадров исходного видеоматериала не рекомендуется, т.к. это понизит точность вычисления оптического потока. Предпочтительнее уменьшать разрешение, а затем переносить результат (канал прозрачности) на исходное разрешение покадрово с помощью алгоритма [33]. Предполагается, что частота кадров находится в диапазоне 15-25 кадров в секунду (что соответствует вебкамере и телевизионному стандарту PAL соответственно).

Освещение в сцене должно быть стабильным, как и другие параметры, которые могут резко изменить изображение (выдержка, фокус). Должно отсутствовать резкое движение объекта/камеры[2]. Объект не должен сильно изменять форму в процессе движения. Последнее требование позволяет работать строго с двумя полными ключевыми кадрами.

Данные условия не являются необходимыми, однако позволяют формализовать задачу и продемонстрировать преимущества предложенного алгоритма по сравнению другими существующими методами. При практическом применении алгоритма несоблюдение требований на входные данные можно компенсировать добавлением ключевых кадров и более тонкой настройкой параметров алгоритма.

Целью работы является исследование и разработка методов и алгоритмов выделения объекта переднего плана от фона в видеопоследовательности,

---

[1] Ограничения на длительность и разрешение видео обусловлены в основном объемом памяти компьютера и приведены для типичной конфигурации (2-4 ГБ ОЗУ).

[2] Ограничения на скорость движения обусловлены алгоритмом вычисления оптического потока и описаны в разделе 3.2.1

а также создание на основе разработанных методов программных модулей для матирования видео. Полученная система должна превосходить существующие по соотношению качество результата / объем пользовательского ввода. Основной критерий качества результата – при наложении извлеченного слоя на новый фон монтаж должен быть не заметен.

Существующие подходы

В этом подразделе описаны два наиболее распространенных подхода к решению задачи матирования, не являющихся автоматическими (требуют специальных условий съемки или ручной работы), но активно применяемых на практике. Во второй главе будут описаны существующие автоматические методы.

Выделение по цвету (chroma-keying) – довольно старый метод, активно применяющийся в кинематографии. Он требует съёмки объекта на однородном фоне («заднике») определённого цвета (называемого ключевым – key color), при этом данный цвет должен отсутствовать в изображении объекта. Изначально chroma-keying осуществлялся без помощи компьютера путём многократной пересъёмки киноплёнки с различными светофильтрами. Наиболее часто используется синий или зелёный фон.

Существует несколько различных алгоритмов выделения по цвету. Разметка в таких алгоритмах не требуется – они работают только с изображением и заданным цветом фона. Они обрабатывают пиксели изображения независимо и основываются на эмпирических формулах, выражающих α через C. Простейшим примером такой зависимости является линейная комбинация цветовых каналов (для синего фона):

$$\alpha = (r + g) / 2 - b \qquad (2)$$

Результат отсекается по отрезку [0, 1]. Предполагается, что значения r, g, b также задаются на отрезке [0, 1]. Подобные методы накладывают ограничение не только на цвет фона, но и на цвет объекта: он не должен совпадать с цветом фона и, возможно, с каким-либо цветом, для которого исполь-

зуемый метод определяет значение α неверно (например, приведенная формула классифицирует все оттенки серого как фон).

Другой вариант, предложенный в [65] (для фона, являющегося линейной комбинацией синего и зелёного):

$$\alpha = 1 - a_1(b - a_2 g)$$
$$F_b = \min(b, a_2 g) \qquad (3)$$

Здесь параметры a_1, a_2 настраиваются вручную. Синяя компонента цвета объекта (b_F) отсекается, остальные без изменения берутся из исходного изображения, т.е. $F_r = r$, $F_g = g$. Наиболее распространенные формулы, применяемые в коммерческих программах выделения по цвету, приведены в [66], [19]. Некоторые алгоритмы, в отличие от приведенных, используют цветовое пространство HSV, как, например, [7].

Подобные эмпирические формулы используются в программах Ultimatte [82], Keylight [79]. Алгоритм Primatte [81] использует образцы цвета из исходного изображения и строит по ним цветовую модель фона. Примеры практического применения выделения по цвету показаны на рис. 3.

Преимущество данного метода заключается в возможности его работы в реальном времени, а также в возможности качественного извлечения теней, бликов и полупрозрачных объектов. Существуют аппаратные реализации алгоритмов выделения по цвету, работающие в реальном времени с видеосигналом (например, [79]).

Следует заметить, что если есть возможность сфотографировать объект два раза на фоне различного, но известного цвета, то задача перестает быть недоопределенной и её можно решить методом триангуляции [47]. На практике такой вариант применяется редко, но его используют для получения эталонных изображений альфа-канала для оценки качества результатов других методов.

Рисунок 3. Пример использования выделения по цвету в кино и на телевидении

Также существуют методы, полагающиеся на специальные условия съемки [57], [58], [59], [39], [40], [72] в том числе с использованием нескольких синхронизированных камер [29].

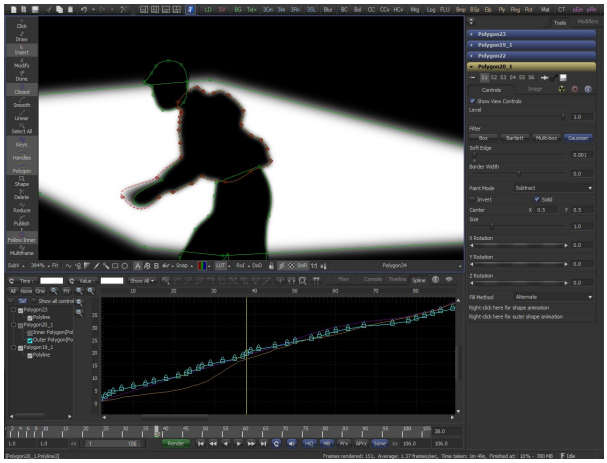
Ротоскопирование

В случае, когда возможность снять объект на фоне константного цвета отсутствует, применяется ротоскопирование ([83], [84]) – ручное создание маски объекта. Невозможность обеспечить одноцветный фон для автоматической обработки может быть вызвана разными причинами, например:

- архивные и другие видеоматериалы, при съемке которых матирование не планировалось
- слишком большая сцена, которую проблематично покрыть одноцветным задником
- невозможность добиться равномерного освещения задника

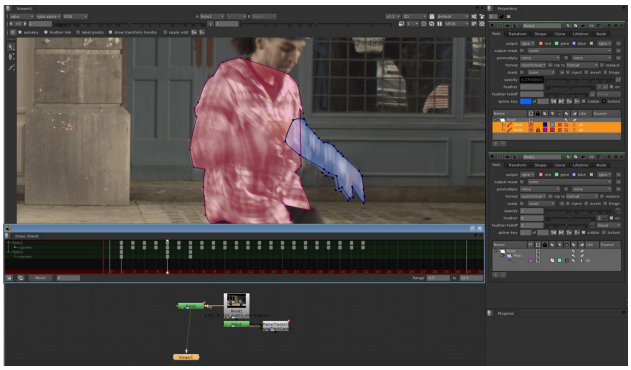
- необходимость взаимодействия переднего и заднего плана (например, актер берет какой-то предмет, который до этого был задним планом) – в частности, когда стоит задача не заменить фон, а лишь дополнить/видоизменить имеющийся фон (например, сделать раздельную цветокоррекцию переднего и заднего плана)

Особо следует отметить задачу преобразования монокулярного видеоматериала в бинокулярный (стереоскопическое 3D) или даже в многоракурсный. Этот случай попадает и в первую, и в последнюю вышеописанные категории.

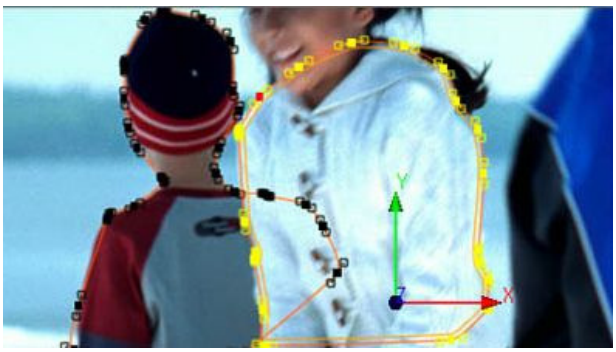


(а)

(б)



(в)



(г)

Рисунок 4. Процесс сплайнового ротоскопирования в разных видеоредакторах. Иллюстрация из [83]. **(а)** Digital Fusion (Eyeon Software Inc.), **(б)** Commotion (Pinnacle Systems), **(в)** Nuke (The Foundry), **(г)** Flame (Autodesk)

В некоторых случаях ротоскопирование можно частично автоматизировать, например, если движение объекта является аффинным/плоским, т.е. видимый силуэт объекта деформируется в процессе движения лишь линейным образом.

Обычно ротоскопирование осуществляется с помощью сплайнов Безье, задаваемых в ключевых кадрах (рис. 4). Контрольные точки интерполируются между ключевыми кадрами (линейно, либо тоже с помощью сплайнов). Дополнительно можно без особых усилий добиться следующих эффектов:

- размытие движения – получается размытием созданной маски, т.к. скорость сплайна в каждой точке известна (может быть вычислена из положения сплайна в ключевых кадрах)
- константное размытие границ (фокусировка) – получается применением гауссова фильтра к маске
- размытие границ переменной толщины – достигается созданием двух вложенных сплайнов, один из которых соответствует прозрачности $\alpha = 0$, а другой – $\alpha = 1$. Прозрачность интерполируется между сплайнами. Данный случай более трудозатратный, т.к. сплайнов становится вдвое больше, но обычно внешний и внутренний сплайны движется согласованно.

Однако мелкие детали (например, волосы) очень трудно ротоскопировать, т.к. приходится создавать много сплайнов и двигать их независимо. В сложных случаях количество сплайнов может достигать нескольких десятков (рис. 5). В задаче преобразования 2D в стереоскопическое 3D может требоваться разбиение кадра на 30-120 слоев [83].

Цель работы

Целью данной диссертации является разработка автоматизированного алгоритма матирования видео (без особых требований к съемке). Критерий качества: при наложении на новый фон монтаж должен быть незаметен.

(а) (б)

Рисунок 5. Применение сплайнового ротоскопирования для коррекции изображения (а) и преобразования 2D в стереоскопическое 3D (б). Иллюстрация из [83].

СОДЕРЖАНИЕ РАБОТЫ ПО ГЛАВАМ

Работа состоит из четырех глав. В первой главе рассматривается задача матирования изображений и предлагается алгоритм матирования на основе байесовского подхода. Во второй главе формулируется задача матирования видео по ключевым кадрам и предлагается алгоритм ее решения, основывающийся на двух других алгоритмах: вычисления оптического потока и матирования видеообъема. В третьей главе рассматривается задача вычисления оптического потока. В четвертой главе описывается алгоритм матирования видеообъема.

1. АЛГОРИТМ МАТИРОВАНИЯ ИЗОБРАЖЕНИЙ

Алгоритмы матирования изображений получают на вход исходное изображение с разметкой. Как описано во введении, разметка требуется для устранения неоднозначности в разложении на передний план, фон и канал прозрачности (1). Обычно используется тернарная разметка – заданная пользователем сегментация изображения на 3 области: объект переднего плана, фон и неизвестную (переходную) область. Первые две дают некоторую информацию об объекте, который необходимо извлечь, а последняя определяет регион, к которому нужно применить алгоритм. Результатом алгоритма является слой переднего плана с известным цветом и прозрачностью для каждого пикселя, а также слой фона. При смешивании, эти слои должны давать в точности исходное изображение.

Таким образом, разметка задает области:

- с $\alpha = 0$, $B = I$
- с $\alpha = 1$, $F = I$
- с неизвестными F, B, α.

В пикселах неизвестной области задача является сильно недоопределенной, т.к. для каждого цвета исходного изображения I существует бесконечное число комбинаций цветов объекта и фона. Для того, чтобы сделать её решаемой, требуется некоторая дополнительная регуляризация. В предлагаемом алгоритме делается предположение о когерентности цветов близко расположенных пикселов. Распределение цвета моделируется набором нормальных распределений, а для оценки правдоподобия разложения (1) используется формула Байеса.

1.1. Байесовский подход

Одним из основных методов регуляризации является байесовский вывод. Применительно к задаче матирования он был впервые предложен в статье [17]. Пиксели обрабатываются, начиная с границ регионов объекта/фона, сужая неизвестную область шаг за шагом. Пиксели, обработанные на предыдущих шагах, также учитываются в выборках объекта и фона в дополнение к пикселям из известных областей. В качестве цветовой модели используется множество ориентированных гауссиан (см. рис. 6). Алгоритм использует схему Байеса для максимизации правдоподобия значений F, B и α. Условная вероятность для F, B и α по наблюдаемому цвету C может быть переписана с помощью формулы Байеса:

$$\mathrm{P}(\mathrm{F},\mathrm{B},\alpha \mid \mathrm{C}) = \frac{\mathrm{P}(\mathrm{C} \mid \mathrm{F},\mathrm{B},\alpha)\mathrm{P}(\mathrm{F})\mathrm{P}(\mathrm{B})\mathrm{P}(\alpha)}{\mathrm{P}(\mathrm{C})}, \tag{4}$$

где P(C|F,B,α) оценивается через расстояние между C и смесью F и B (т.е. через норму разности левой и правой части в уравнении (1)),

P(F) и P(B) оцениваются с помощью плотности гауссиан объекта и фона,

P(α) игнорируется – предполагая все значения α равновероятны, т.е. распределение P(α) – равномерное на отрезке[0, 1],

P(C) является константой по отношению к максимизируемым параметрам.

Взяв логарифм от (4) и опуская члены, не влияющие на параметры, которые нужно найти, получаем:

$$\mathrm{L}(\mathrm{F},\mathrm{B},\alpha \mid \mathrm{C}) = \mathrm{L}(\mathrm{C} \mid \mathrm{F},\mathrm{B},\alpha) + \mathrm{L}(\mathrm{F}) + \mathrm{L}(\mathrm{B}), \tag{5}$$

где L(·) = log P(·).

Авторы [17] используют следующие оценки для L(C|F,B,α), L(F), L(B):

$$\mathrm{L}(\mathrm{C} \mid \mathrm{F},\mathrm{B},\alpha) = -\left\|\mathrm{C} - \alpha\mathrm{F} - (1-\alpha)\mathrm{B}\right\|^2 / \sigma_{\mathrm{C}}^2, \tag{6}$$

(σ_{C} настраивается пользователем),

$$\mathrm{L}(\mathrm{F}) = -(\mathrm{F} - \overline{\mathrm{F}})^{\mathrm{T}} \Sigma_{\mathrm{F}}^{-1} (\mathrm{F} - \overline{\mathrm{F}}) / 2, \tag{7}$$

где $\overline{F}$ и Σ_F являются математически ожиданием и ковариационной матрицей гауссианы объекта переднего плана, L(B) – аналогично L(F).

В случае нескольких пар кластеров объекта/фона, оптимальные *F*, *B* и α вычисляются для каждой пары, после чего выбирается пара, дающая наибольшую величину правдоподобия $L(F,B,\alpha \mid C)$.

Для максимизации неквадратичной функции (5) используется итеративная процедура, поочередно принимающая α и F, B за константы, что дает две квадратичные задачи. В качестве начального приближения для α используется среднее значение по окрестности обрабатываемого пикселя.

Для константной α выводится следующая линейная относительно F и B система линейных алгебраических уравнений размером 6 на 6:

$$\begin{bmatrix} \Sigma_F^{-1} + I\alpha^2/\sigma_C^2 & I\alpha(1-\alpha)/\sigma_C^2 \\ I\alpha(1-\alpha)/\sigma_C^2 & \Sigma_B^{-1} + I(1-\alpha)^2/\sigma_C^2 \end{bmatrix} \begin{bmatrix} F \\ B \end{bmatrix} = \begin{bmatrix} \Sigma_F^{-1}\overline{F} + C\alpha/\sigma_C^2 \\ \Sigma_B^{-1}\overline{B} + C(1-\alpha)/\sigma_C^2 \end{bmatrix} \tag{8}$$

Для константных F и B результат для α является проекцией C на отрезок FB:

$$\alpha = \frac{(C-B)\cdot(F-B)}{\|F-B\|^2} \tag{9}$$

Вычисление F, B и α поочередным применением формул (8) и (9) продолжается, до наступления сходимости.

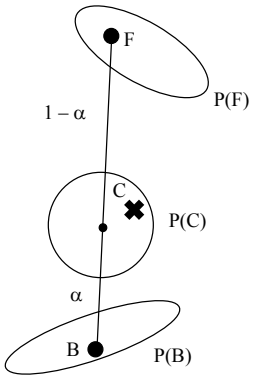
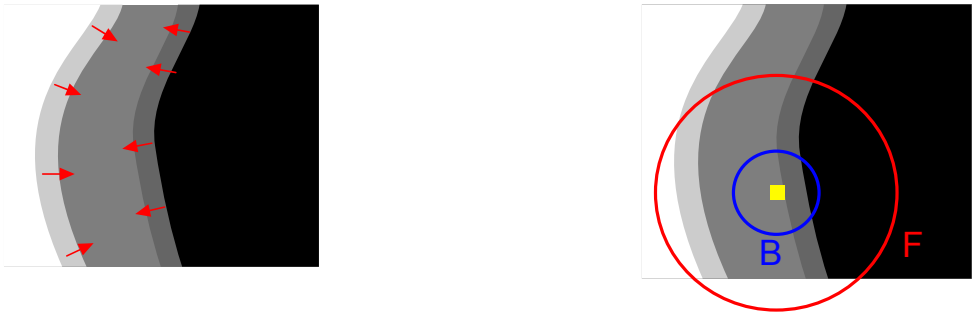


Рисунок 6. Иллюстрация цветовых распределений объекта/фона в цветовом пространстве RGB



(а) (б)

Рисунок 7. Процесс байесова матирования [17]. **(а)** Пикселы обрабатываются равномерно от границ неизвестной области. **(б)** образцы цветов F, B берутся из окрестности обрабатываемого пиксела.

Распределения P(F) и P(B) вычисляются на основе как размеченных пикселов объекта/фона, так и ранее обработанных. Это показано на рис. 7. образцы цветов F, B для вычисления распределений P(F), P(B) берутся из окрестности обрабатываемого пиксела. Радиус окрестности может быть адаптивным и увеличиваться в случае, если образцов цвета не найдено (т.е. если окрестность целиком лежит в неизвестной области), либо найдено недостаточно для надежной оценки параметров распределения.

1.2. Алгоритм аналитического матирования

Данный алгоритм был предложен в статье [33]. Он может работать с разметкой низкой точности. Цветовая статистика не используется – альфа-канал напрямую ищется из изображения. Алгоритм называется аналитическим, т.к. описывает аналитическое решение задачи, в отличие от итерационных алгоритмов.

Идея алгоритма основана на предположении, что цвета пикселей в малых фрагментах изображений F и B лежат примерно на одной прямой в пространстве RGB (являются линейной комбинацией двух цветов). Тогда α можно считать линейно зависящей от цвета в малом фрагменте изображения:

$$\alpha_p \approx aC_p + b \tag{10}$$

Здесь p – любой пиксель рассматриваемого фрагмента, а значения a и b фиксированы для всего фрагмента. Для цветного изображения a – вектор, для изображения в градациях серого – число. Во втором случае

$$\begin{aligned} a &= 1 / (F - B), \\ b &= -B / (F - B). \end{aligned} \tag{11}$$

В случае цветного изображения коэффициенты a, b также можно выразить через F и B. Модель цветовой линии проиллюстрирована на рис. 8.

В качестве фрагментов изображения в алгоритме рассматриваются все окна размером 3x3 пикселя, принадлежащие изображению. Для нахождения α минимизируется функционал:

$$J(\alpha, a, b) = \sum_{j \in \Omega} \left(\sum_{i \in w_j} (\alpha_i - a_j C_i - b_j)^2 + \varepsilon a_j^2 \right), \tag{12}$$

где ε – параметр регуляризации, задающий дополнительную гладкость, w_j – окно 3x3 с центром в пикселе j.

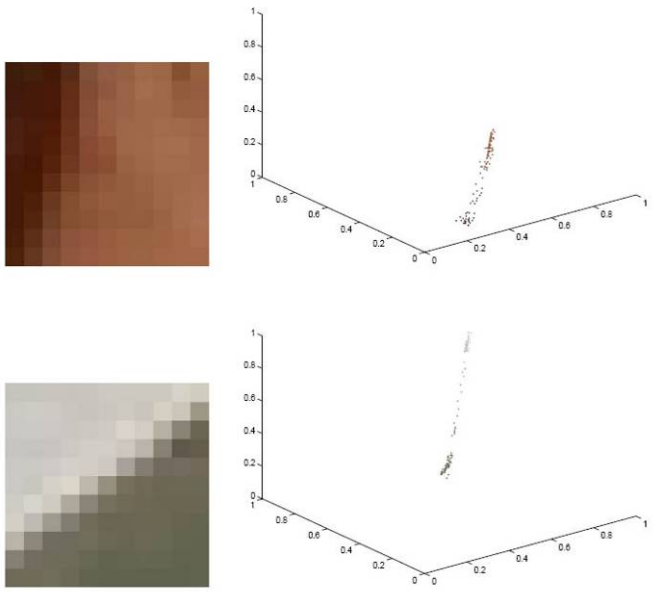


Рисунок 8. Иллюстрация модели цветовой линии: на небольших фрагментах естественных изображений цвета пикселей (в трехмерном цветовом пространстве RGB) могут быть аппроксимированы отрезком

В [33] показано, как можно при минимизации J(α, a, b) исключить a, b (выразив их через α) и свести задачу к минимизации квадратичной формы (при наборе ограничений $\alpha_i = 0, 1$ в пикселях фона и объекта соответственно). Функционал в таком случае будет иметь вид:

$$J(\alpha) = \alpha^T L \alpha, \qquad (13)$$

где L – матрица NxN (N – общее число пикселей в изображении), называемая лапласианом, т.к. она является дискретным аналогом оператора Лапласа на произвольном графе.

Для цветных изображений задача решается аналогичным образом. После нахождения α требуется найти F и B. На основе подобных рассуждений эта задача также сводится к решению линейной системы.

Для выполнения соотношения (10) для изображения в градациях серого требуется гладкость изображений F и B. В случае RGB-изображения это условие ослабляется – достаточно, чтобы в малых окнах цвета F лежали примерно на одной прямой, и цвета B лежали примерно на одной прямой.

Данный алгоритм дает точное решение в случаях, когда во всех окнах 3x3 на изображении каждый из цветов F и B является смесью не более, чем двух цветов (кроме вырожденных случаев).

Примеры работы данного алгоритма приведены на рис. 9. Недостатком данного алгоритма является медленная скорость работы и локальность цветовой модели (даже если объект и фон сильно различаются по цвету, алгоритм это не учитывает, т.к. опирается лишь на модель цветовой линии в окнах малого размера). Поэтому данный алгоритм в данной работе не был напрямую использован для матирования ключевых кадров, но идея модели цветовой линии использована в иерархическом подходе (см. раздел 1.6).

Однако у этого алгоритма есть преимущество, которое позволяет использовать его для матирования видео (см. главы 2 и 4): достаточно вычислить лапласиан один раз, после чего его можно многократно применять для вычисления функционала невязки (13) для разных значений канала прозрачности α. Также в статье [33] был предложен метод вычисления каналов F, B по фиксированным I, α, который в данной работе использован применительно к видео.

(а) (б) (в) (г)

Рисунок 9. Примеры работы алгоритма аналитического матирования: **(а, в)** изображение с наложенной разметкой, **(б, г)** результат (канал прозрачности)

1.3. Предлагаемый алгоритм

Для матирования ключевых кадров в данной работе предлагается алгоритм, основанный на байесовском подходе. В следующих трех разделах описаны отличия предложенного метода от оригинального алгоритма [17]. Затем в разделе 1.7 описан интерактивный подход к уточнению разметки и результата, позволяющий пользователю эффективно исправлять ошибки алгоритма.

Итоговый алгоритм предлагается в качестве основного для создания ключевых кадров, хотя не является единственным возможным вариантом. На практике можно применить любой алгоритм, выдающий в качестве результата канал прозрачности.

1.4. Гладкость канала прозрачности

Байесов алгоритм матирования очень чувствителен к перекрытиям цветовых моделей объекта и фона. В исходном алгоритме это делает вычисление α нестабильным и часто приводит к существенному шуму в получаемом канале прозрачности. Использование простого размытия или медианного фильтра может улучшить канал прозрачности, но при этом маленькие детали объекта могут быть потеряны. Поэтому в данной работе предлагается добавить член, отвечающий за гладкость канала прозрачности в байесову схему. Смоделируем гладкость одномерной гауссианой с математическим ожиданием α_0 равным среднему значению прозрачности по ранее обработанным пикселям, т.е. тем же значением, что используется в качестве начального приближения для α при решении системы (8). Этот член, отвечающий за гладкость, вводится в (4) как P(α).

Для L(α) используется следующее выражение [50]:

$$\mathrm{L}(\alpha) = -\left\|\alpha - \alpha_0\right\|^2 / \sigma_\alpha^2, \tag{14}$$

Таким образом, максимизируется следующее логарифмическое правдоподобие:

$$\mathrm{L}(\mathrm{F},\mathrm{B},\alpha \mid \mathrm{C}) = \mathrm{L}(\mathrm{C} \mid \mathrm{F},\mathrm{B},\alpha) + \mathrm{L}(\mathrm{F}) + \mathrm{L}(\mathrm{B}) + \mathrm{L}(\alpha), \tag{15}$$

Приравнивая частные производные (15) по α нулю, получаем следующее выражение для α:

$$\alpha = \frac{\alpha_0 / \sigma_\alpha^2 + (C-B)\cdot(F-B)/\sigma_C^2}{1/\sigma_\alpha^2 + \left\|F-B\right\|^2/\sigma_C^2} \tag{16}$$

Формула (16) заменяет формулу (9) в процессе минимизации.

σ_α можно определить несколькими способами. Во-первых, можно использовать значение, настраиваемое пользователем. Во-вторых, можно поставить значение σ_α в зависимость от расстояния между гауссианами объекта переднего плана и фона, чтобы сглаживать в первую очередь места наи-

большей неуверенности, в то же время избегая слишком большого сглаживания в целом:

$$\sigma_\alpha = \sigma_\alpha^0 + \lambda \cdot \rho(P(F), P(B)), \tag{17}$$

где σ_α^0 и λ настраиваются пользователем (в экспериментах использовалось $\sigma_\alpha^0 = \lambda = 0{,}1$), а $\rho(\cdot,\cdot)$ является расстоянием между двумя распределениями (расстояние между центрами гауссиан или, например, взаимная информация).

В-третьих, можно использовать двухпроходное байесово матирование, используя значения правдоподобия, вычисленные на первом проходе, для оценки σ_α на втором проходе.

Чтобы сделать сглаживание менее равномерным и более согласованным с цветовыми изменениями, используется взвешенное среднее для α_0 в пикселе q:

$$\alpha_0 = \frac{1}{W}\sum_p \alpha_p \cdot w_p, \tag{18}$$

где суммирование происходит по уже обработанным (или известным) пикселям в окрестности пикселя q со следующими весами (W является суммой всех w_p для пикселя q):

$$w_p = \exp\left(-\frac{\left\|C_p - C_q\right\|^2}{2\sigma_w^2}\right) \tag{19}$$

(используется эмпирически подобранное значение $\sigma_w = 0{,}2$). Это же значение α_0 используется в качестве начального приближения для α в уравнении (8). Использование члена, отвечающего за гладкость, почти не влияет на время работы алгоритма. Пример работы алгоритма показан на рис. 10.

(а) (б) (в) (г)

Рисунок 10. (а) исходное изображение, (б) тернарная разметка, (в) вычисленный канал прозрачности, (г) результат наложения на новый фон.

1.5. Сортировка пикселов по цветовой близости

В предлагаемом подходе вместо равномерной обработки пикселов (как на рис. 7) пикселы сортируются по цветовой близости к соседним пикселам. Первым обрабатывается какой-либо пиксел у границы неизвестной области. Затем может быть обработан соседний с ним пиксел, либо какой-либо другой пиксел у границы, в зависимости от того, какой пиксел из необработанных ближе всего по цвету к соседнему обработанному пикселу. Такой подход позволяет улучшить качество результата. В исходном алгоритме неизвестная область постепенно уменьшается (равномерно от границ), сходясь к своей средней линии. В предлагаемом алгоритме область уменьшается неравномерно, сходясь к некоторой границе внутри изображения (т.к. при сортировке по цветовой близости в первую очередь будут обрабатываться пикселы по каждую из сторон этой границы, но сама граница скорее всего не будет пересечена). Это можно гарантировать при наличии ровно одной границы внутри неизвестной области. В противном случае, результат зависит от интенсивности границ: если в неизвестной области присутствует сильная, но некорректная (т.е. не разделяющая объект и фон) граница, то она может исказить результат.

В сочетании с условием гладкости, данный подход позволяет притянуть границу в канале прозрачности к найденной границе на изображении.

1.6. Иерархический подход

Другим улучшением является иерархическое байесово матирование. Главной задачей этого улучшения было уменьшение времени работы алгоритма без потери качества матирования. Самым простым способом было бы применить байесово матирование к уменьшенному изображению, после чего увеличить изображение до исходного разрешения и выполнить байесово матирование ещё раз, но со значительно меньшим радиусом выборки. Но есть более эффективный способ: применяя иерархический подход [33], описанный в разделе 1.2, к результату байесова матирования, можно вычислить коэффициенты линейного приближения прозрачности значениями цвета для уменьшенного изображения, и использовать их, чтобы увеличить канал прозрачности до исходного разрешения. Линейное приближение определяется коэффициентами *a* и *b*, которые определяются из соотношения

$$\alpha_p \approx aC_p + b \tag{20}$$

где p – пиксель в окне (размером, например, 3x3), *a* и *b* являются фиксированными по окну коэффициентами. Для изображений в градациях серого коэффициенты *a* и *b* могут быть вычислены по следующим формулам:

$$\begin{aligned} a &= 1 / (\mathrm{F} - \mathrm{B}), \\ b &= -\mathrm{B} / (\mathrm{F} - \mathrm{B}). \end{aligned} \tag{21}$$

Для цветных изображений, они также могут быть выражены через *F* и *B* (в данном случае *a* является 3-мерным вектором):

$$\alpha_p \approx \sum_k a^k C_p^k + b, \tag{22}$$

где k является индексом цветовой компоненты.

Это позволяет полностью избавиться от второго прохода алгоритма, т.к. обычно получающийся канал прозрачности достаточно точен. Для восстановления изображений *F* и *B* можно предположить, что их RGB-каналы являются линейными комбинациями каналов исходного изображения *C* (хотя

также можно провести второй проход байесова матирования для константной α).

Уменьшенное изображение получается билинейной интерполяцией. При вычислении уменьшенной разметки, пиксель считается принадлежащим объекту/фону, только если все соответствующие пиксели разметки исходного разрешения принадлежат объекту/фону, в противном случае пиксель помечается, как неизвестный. Значения параметров байесова матирования, такие как сигма, используемая для пространственного взвешивания выборки, тоже уменьшаются. Байесово матирование выполняется на уменьшенном изображении/разметке и выдаёт изображения α, F и B. Эти изображения нужно увеличить до исходного разрешения. Для канала прозрачности α и для каждого канала изображений F и B применяется следующая процедура:

1. Вычислить коэффициенты a и b для каждого пикселя уменьшенного изображения, с помощью метода наименьших квадратов по окну 3 на 3, как в уравнении (20).
2. Увеличить карты коэффициентов, используя билинейную интерполяцию.
3. Вычислить увеличенное изображение (исходного разрешения), применяя уравнение (20) к исходному изображению C и коэффициентам a и b из увеличенных изображений коэффициентов, полученных на шаге 2.

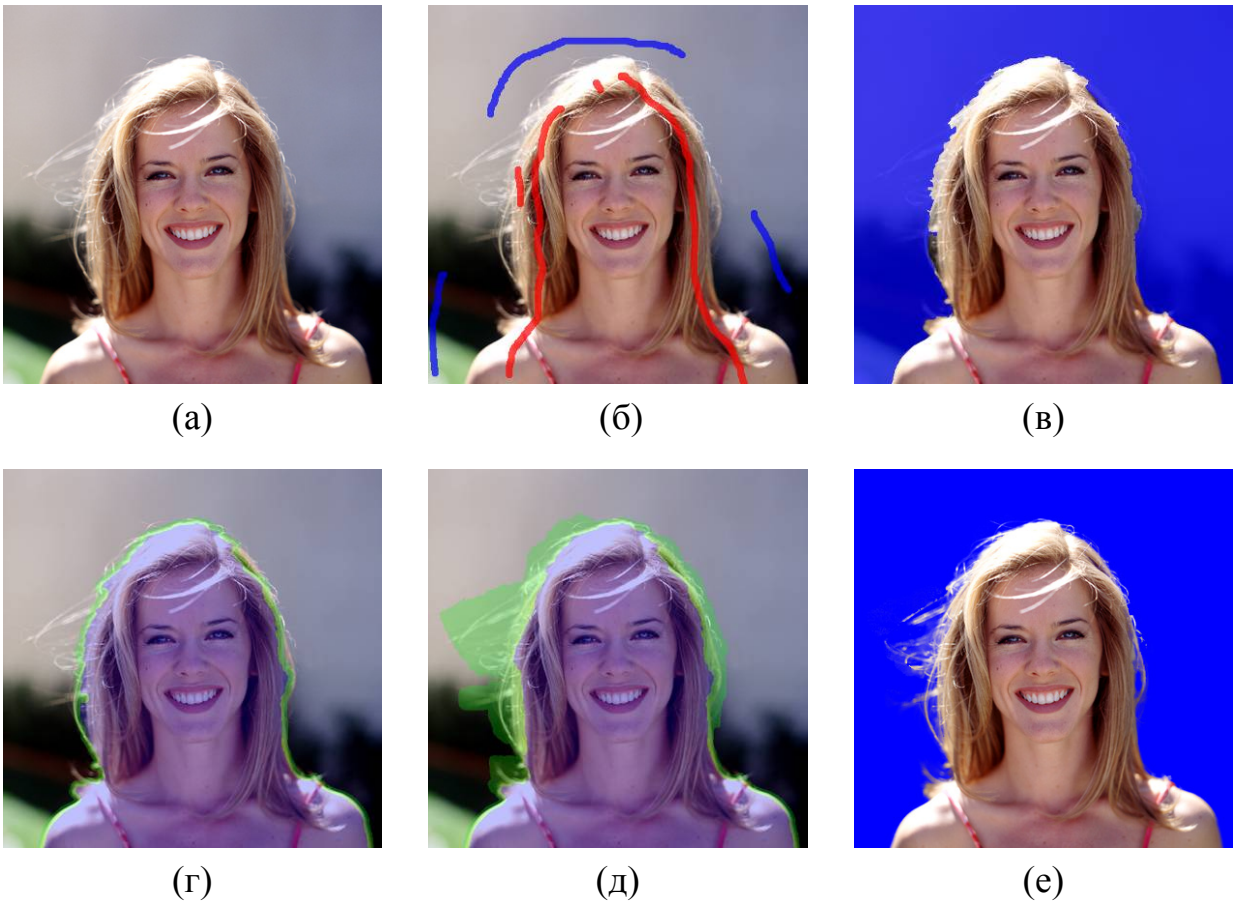
Можно заметить, что применение уравнения (20) к уменьшенному изображению размывает альфа-канал. Чтобы предотвратить это, на шаге 1 можно приравнять значение альфы в обрабатываемом пикселе (т.е. в центральном пикселе окна 3 на 3) правой части уравнения.

Применение байесова матирования к изображениям меньшего разрешения дает нелинейное ускорение, т.к. уменьшается не только число обрабатываемых пикселей, но и области поиска образцов цветов F, B. При небольших коэффициентах масштабирования (≤ 4) качество матирования практически не изменяется.

1.7. *Интерактивное матирование изображений*

Выше был описан алгоритм вычисления канала прозрачности на основе исходного изображения и тернарной разметки. Теперь рассмотрим процедуру создания такой разметки пользователем.

Вначале пользователю предлагается построить жесткую (бинарную) разметку, используя мазки кистями объекта и фона. Для этого используется алгоритм GrowCut [64]. Целью данного шага является построение точной сегментации четких контуров и приближенной сегментации мягких областей.



(а) (б) (в)

(г) (д) (е)

Рисунок 11. Основные шаги интерактивного алгоритма: **(а)** исходное изображение, **(б)** разреженная разметка, **(в)** бинарная разметка, полученная алгоритмом GrowCut, **(г)** сгенерированная на основе нее тернарная разметка, **(д)** доработанная пользователем тернарная разметка, **(е)** результат матирования по разметке (д)

Затем генерируется неизвестная область тернарной разметки морфологическим сжатием областей объекта и фона. Радиус сжатия выбирается пользователем и может быть изменен в реальном времени. Задача пользователя – настроить ширину переходной области так, чтобы были покрыты все четкие края с учетом их средней размытости. Эти шаги показаны на рис. 11.

Для нечетких и полупрозрачных областей, которые невозможно обработать бинарной сегментацией и морфологией, предложен инструмент расширения разметки [51]. Он позволяет быстро добавлять большие переходные области, соединенные с уже существующими. Инструмент работает следующим образом:

1. Пользователь рисует фрагмент границы нечеткой области внутри объекта или фона.
2. Концы границы соединяются с существующей переходной областью. Для поиска кратчайшего соединяющего пути в изображении используется алгоритм Дейкстры [20].
3. Концы найденных путей соединяются между собой, чтобы образовать замкнутую область. Для этого ищется путь от одного конца до другого внутри неизвестной области. Если такого пути нет (такое возможно, если неизвестная область не является односвязной), действие отменяется.
4. Полученная область, состоящая из пользовательской траектории, двух соединений и внутреннего соединения, помечается как переходная.

Граница на первом шаге может быть получена одним из двух способов:

- явное рисование границы пользователем
- указание пользователем двух точек, которые затем соединяются с помощью алгоритма Дейкстры.

Предложенный метод взаимодействия с пользователем является интуитивным и удобным для быстрого расширения существующей переходной области в зоны полупрозрачных элементов. Такой подход быстрее ручного ри-

сования переходной области и в то же время получает более точную область (она имеет меньшую площадь). Данный способ разметки проиллюстрирован на рис. 12.

Когда пользователь завершил редактирование разметки, применяется алгоритм байесовского матирования со сглаживанием, описанный ранее. Пользователь может выбирать степень гладкости и коэффициент уменьшения изображения при иерархической обработке.

На последнем этапе пользователь может редактировать канал прозрачности следующими инструментами: мягкие кисти, контрастность, размытие, смазывание. После мазка одним из этих инструментов вызывается байесовский алгоритм для обновления значений F, B при фиксированном α. Также пользователь может изменять разметку и вызывать пересчет значений прозрачности в выбранной прямоугольной области. Пересчет можно вызывать и без изменения разметки при изменении степени гладкости. Таким образом, можно использовать разные параметры для разных частей изображения.

(а) (б) (в)

Рисунок 12. Процесс расширения разметки. **(а)** Точки A, B заданы пользователем, точки C и D – найденные к ним ближайшие точки на разметке. Путь A–B рисуется пользователем или ищется автоматически. **(б)** Добавление одиночного волоса (или тонкой пряди волос) предложенным методом становится простым – достаточно выделить две точки вблизи кончика волоса. **(в)** Волос добавлен в разметку.

Весь цикл работы пользователя над изображением показан на рис. 13. Пользователь начинает с разреженной разметки. Он может завершить процесс на шаге «Просмотр результата», либо выбрать один из двух способов уточнения, доступных на данном шаге. Возврат с шага «Матирование» назад возможен только с потерей текущего результата.

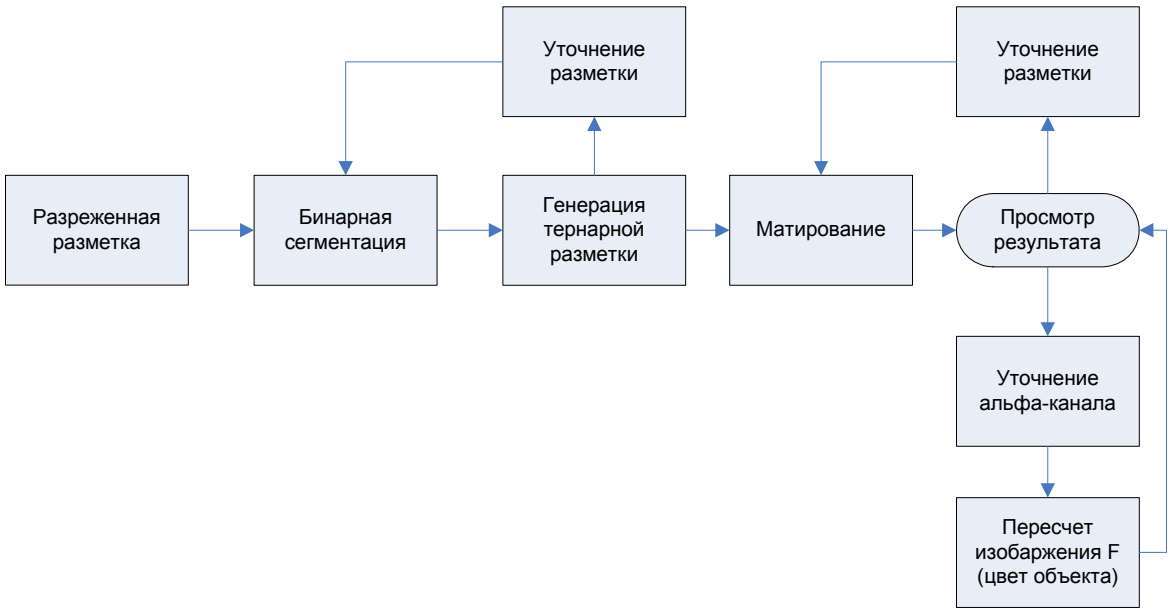


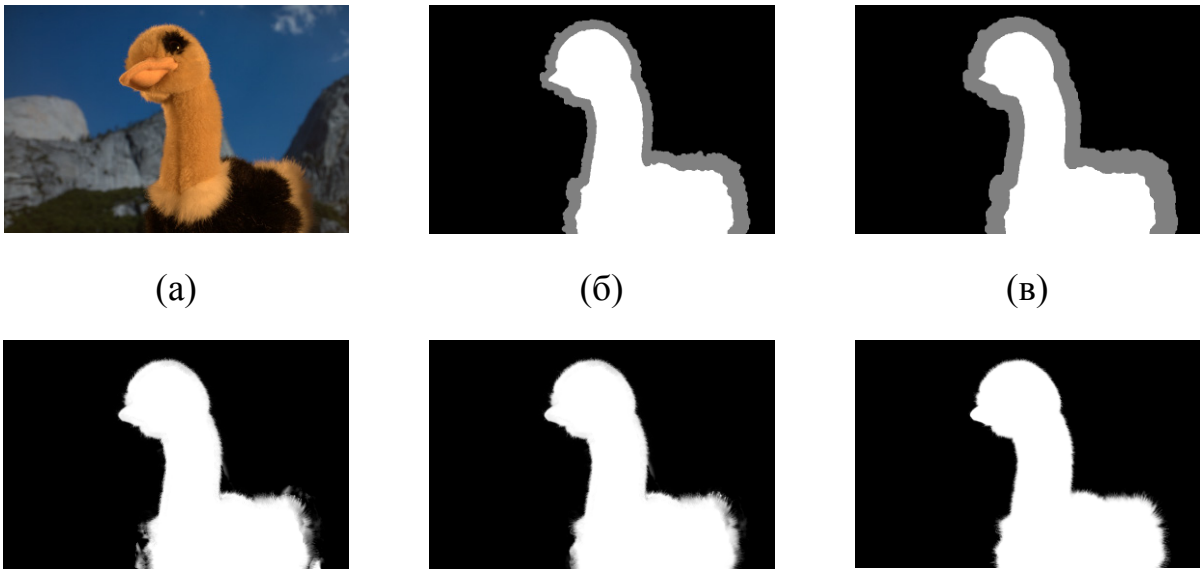
Рисунок 13. Схема интерактивного процесса матирования изображения.

1.8. Численное сравнение

Численное сравнение проводилось на 27 изображениях из тестовой базы [43]. В тестовой базе для каждого изображения есть две тернарные разметки (Trimap1 и Trimap2) – более точная и менее точная (с большей толщиной неизвестной области). Улучшенный алгоритм сравнивался с исходным алгоритмом байесового матирования [17]. Результаты приведены в таблице 1.

Видно, что наибольшее улучшение достигается на наборе с менее точной разметкой [78]. В этом случае сложнее найти границу объекта пользуясь только цветовой информацией, поэтому условие гладкости альфа-канала улучшает результат.

На рис. 14 показан пример разметок, эталонный канал прозрачности и результат на разметке Trimap2.



(а) (б) (в)

(г) (д) (е)

Рисунок 14. (а) изображение №14 из тестовой базы [43], (б) разметка Trimap1, (в) разметка Trimap2, (г) результат алгоритма [17], (д) результат предложенного алгоритма для разметки Trimap2, (е) эталонный результат из [43[)

Таблица 1. Численная оценка предложенного алгоритма.

	Среднеквадратичная ошибка		Среднеабсолютная ошибка	
Тестовый набор	Байесов алгоритм	Предложенный алгоритм, улучшение (%)	Байесов алгоритм	Предложенный алгоритм, улучшение (%)
Trimap1	0,0057	0,0054 (6%)	0,0146	0,0136 (7%)
Trimap2	0,0099	0,0074 (25%)	0,0227	0,0192 (16%)

1.9. Программная реализация

Алгоритм матирования и интерактивный процесс (включающий создание разметки и редактирование результата) реализованы авторами в программе GrowCut – подключаемому модулю к программе Adobe Photoshop, как показано на рис. 15. В реализации использованы библиотеки Photoshop SDK и OpenCV. Язык реализации – C++.

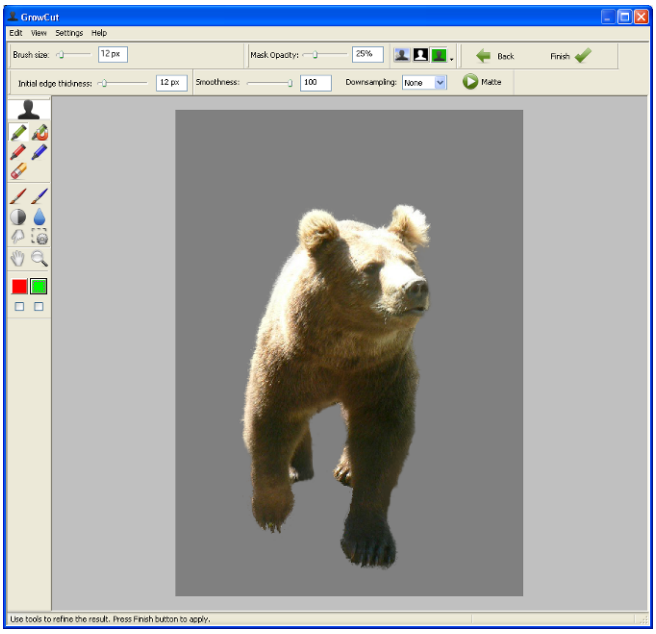


Рисунок 15. Снимок экрана программной реализации.

1.10. Заключение

В данной главе был предложен новый алгоритм интерактивного матирования изображений. Алгоритм основан на байесовом методе матирования, но с добавлением условия гладкости результирующего канала прозрачности α. Интерактивная процедура матирования изображений упрощает и ускоряет процесс извлечения объекта переднего плана на изображении, комбинируя бинарную сегментацию (для более точного поиска жестких границ) и мягкое матирование со сглаживанием.

2. МАТИРОВАНИЕ ВИДЕО ПО КЛЮЧЕВЫМ КАДРАМ

В данной главе рассматриваются подходы к матированию видео. По сравнению с матированием изображений, данная задача является более сложной, так как требует согласованности масок в соседних кадрах, чтобы избежать эффекта мерцания. Естественным подходом к обеспечению такой согласованности является использование оптического потока, представляющего собой карту межкадрового движения (векторное поле скоростей для каждого пиксела). В работе предложена идея «альфа-потока», которая дополняет оптический поток и делает его пригодным для матирования, устраняя некоторые недостатки обычного оптического потока.

2.1. Существующие подходы

Существующие алгоритмы матирования видео можно разделить на три группы по способу анализа движения:

- методы, не использующие движение (рассматривают видео как изотропный 3D-объем)
- методы, использующие точечные траектории
- методы, использующие оптический поток

Алгоритмы, использующие движение, также можно разбить на две группы:

- Методы, вычисляющие движение тернарной разметки, а затем применяющие какой-либо алгоритм матирования изображений покадрово
- Методы, вычисляющие движение канала прозрачности

К методам, не использующим оптический поток, относятся алгоритмы [35] и [68]. Эти методы осуществляют пересегментацию исходных кадров и строят трехмерный граф из полученных сегментов. Временны́е связи между сегментами основываются только на их цвете, что часто приводит к следующей проблеме: граница результирующей маски часто проходит по разным краям в последовательных кадрах (особенно в случае перекрытия). Это сложно исправить добавляя разметку объекта/фона, т.к. пользователь должен будет покрыть значительное количество двухмерных сегментов. Такие артефакт «мерцания» затрудняют просмотр видео и выдают факт наложения объекта на другой фон.

Другим вариантом трехмерного подхода является замена жесткой сегментации на мягкую. Например, алгоритмы [8], [21] используют 6-связный трехмерный объем для вычисления прозрачности. Метод [16] использует kd-дерево для сегментации трехмерного объема. Обычно трехмерный подход приводит к противоположной мерцанию проблеме – «растеканию». Вместо четкого следования контуру объекта, маска прозрачности оставляет за собой следы, которые повторяют предыдущее положение объекта и медленно растворяются со временем. Пример такой проблемы показан на рис. 17, где ме-

тод [33] был применен к двухмерным срезам видеообъема по плоскостям y = const, что аппроксимирует работу трехмерного метода (т.к. движение в кадре преимущественно горизонтальное).

Чтобы избежать проблем мерцания/растекания имеет смысл отслеживать позицию объекта вместо того, чтобы накладывать мягкое условие межкадровой согласованности прозрачности. Обычно в этом случае делается предположение о том, что маска прозрачности подчиняется той же модели движения, что и само изображение. Существует несколько методов, применяющих эту стратегию. Все они предварительно вычисляют оптический поток для всей видеопоследовательности. Оптический поток определяется как карта видимого движения пикселов между кадрами ([27]). Для вычисления оптического потока для данной пары изображений I_1, I_2 требуется найти векторное поле V = (u, v), совмещающее эти изображения:

$$I_1(x, y) \approx I_2(x + u(x, y), y + v(x, y)) \tag{23}$$

На рис. 16 показан пример такого векторного поля для пары изображений.

Рисунок 16. Пример оптического потока

Методы, использующие оптический поток, различаются способом связывания маски и потока. Алгоритм распространения прозрачности [48] использует трехмерное окно в виде параллелепипеда, наклоненного вдоль оптического потока по оси времени. Алгоритм [31] использует трехмерное окно фиксированной формы, но перевзвешивает его элементы на основе вектора потока (образуя трехмерное ядро). Однако такой подход применим лишь при

малой скорости движения объекта, т.е. когда модуль вектора потока не превосходит размер ядра; в противном случае перекрытия приводят к растеканию. Иногда вместо оптического потока на уровне пикселов применяют поток на уровне цветовых распределений ([9]), однако такой подход может выдавать только бинарную сегментацию и имеет артефакты в случае перекрывающихся цветовых распределениях объекта/фона.

Другой вариант – распространение бинарной сегментации или тернарной разметки от ключевого кадра ([18]) или использование результата сегментации в качестве тернарной разметки для следующего кадра, как в методе [10]. Также может быть использована вспомогательная последовательность, содержащая только фон ([18], [49]), снятая отдельно или восстановленная путем аффинной/гомографической аппроксимации движения фона, как реализовано в семействе программ Imagineer Systems [80].

Рисунок 17. Проблема алгоритмов изотропного видеообъема – «растекание» альфа-канала

Далее рассматриваются два метода, наиболее близкие к предлагаемому: метод автоматизированного ротоскопирования [1] и алгоритм [62]. Эти методы вычисляют только бинарную сегментацию. Оба метода ослабляют условие совпадения моделей движения карты прозрачности и изображения, используя вместо него некоторую модель движения маски. Алгоритм автоматизированного ротоскопирования анализирует движение в узкой полоске вдоль внутренней стороны контура объекта. Это позволяет избежать перекрытий.

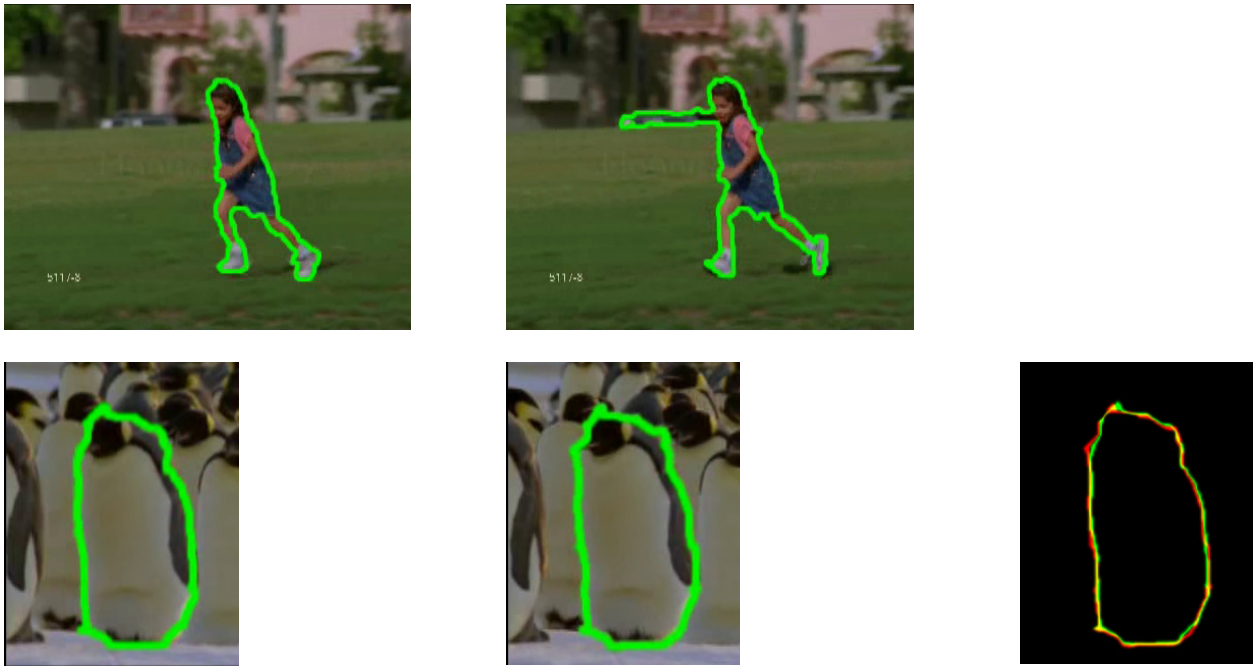


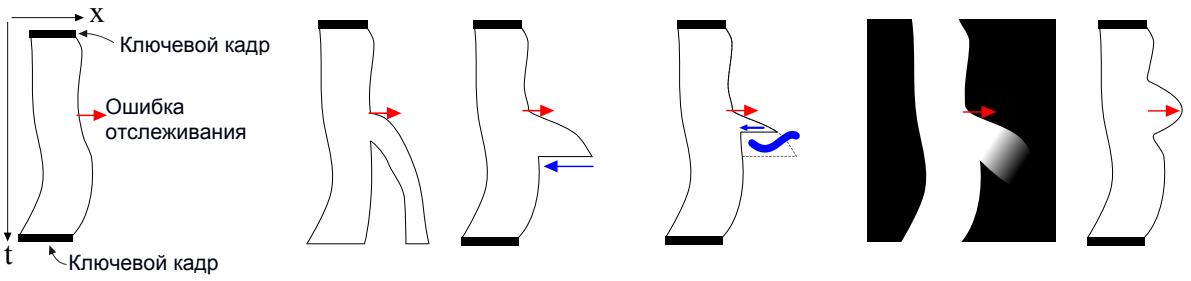
Рисунок 18. Артефакты мерцания в алгоритме [62] – два последовательных кадра имеют сильно различающиеся маски (по сравнению с реальной скоростью объекта).

Алгоритм отслеживания [62] ищет совместное решение бинарной маски и оптического потока. Задача формулируется в виде многометочной оптимизации на графе, что накладывает ограничение на время работы и память. Из-за этого алгоритм работает лишь с одним ключевым кадром и осуществляет последовательное отслеживание в скользящем временно́м окне длиной в несколько кадров. Из-за этого, маска со временем может съезжать, как и в более простых методах последовательного отслеживания. На рис. 18 показан

артефакт мерцания в алгоритме [62] – два последовательных кадра имеют сильно различающиеся маски. При этом межкадровое дрожание границ маски может наблюдаться как в случае ошибки отслеживания (вверху), так и для корректной маски.

2.2. Основные проблемы существующих методов

В дополнение к классификации методов, предложенных в предыдущем разделе, возникает желание классифицировать методы по их недостаткам, чтобы предложить новый метод, лишенный большинства из них. Некоторые из недостатков уже были описаны на примерах конкретных алгоритмов. На рис. 19 проиллюстрированы ошибки методов на примере одномерной видеопоследовательности при наличии ошибки отслеживания движения (красная стрелка).



(а) (б) (в) (г) (д) (е)

Рисунок 19. Иллюстрация ошибок существующих методов.

(а) одномерное видео и истинная сегментация (черный контур)

(б) методы последовательного отслеживания: при ошибке часть фона начинает отслеживаться как отдельный объект

(в) методы изотропного видеообъема: объект отслеживается неправильно, но в какой-то момент возникает резкое изменение контура (синяя стрелка) – мерцание

(г) сложность исправления результата (в) – синий мазок исправляет лишь часть кадров, и мерцание сохраняется (синяя стрелка); для полного исправления пользователь вынужден разметить ошибочную часть маски во всех кадрах

(д) матирование изотропного видеообъема: «растекание» прозрачности

(е) симметричные методы

Основная проблема – временна́я стабильность результата. Лучше всего с ней справляются симметричные методы, основным из которых является ал-

горитм автоматического ротоскопирования [1]. В случае ошибки отслеживания такие алгоритмы образуют артефакт «выступа» (рис. 19 (е)). В предположении, что ошибки отслеживания неизбежны, такой артефакт можно считать наименее проблематичным и легко исправимым:

- Выступ является гладким (плавным во времени). Таким образом, он не привлекает внимание зрителя и скрывает наличие монтажа.
- Выступ является максимальным в том кадре, где произошла ошибка отслеживания. Поэтому его легче исправить, т.к. пользователь интуитивно исправляет сначала наибольшие по площади дефекты маски, что не получится в случае (в)-(г) на рис. 19, т.к. там максимальное отклонение маски не совпадает с местом возникновения ошибки.
- Выступ также имеет наименьший размер по сравнению с остальными методами, т.к. возникает локализовано в месте ошибки и не влияет на дальнейшее отслеживание.

Таким образом, возникает желание взять за основу метод автоматизированного ротоскопирования и обобщить его таким образом, чтобы он корректно работал с произвольной маской прозрачности, а не только с жесткой сегментацией на основе сплайнового контура.

2.3. Общая идея предлагаемого алгоритма

Будем рассматривать видеообъем, ограниченный двумя полными ключевыми кадрами (рис. 20). В случае длинной видеопоследовательности полных ключевых кадров может быть больше. Тогда будем рассматривать фрагменты видеообъема между всеми парами последовательных ключевых кадров. Т.к. в каждом фрагменте ключевые кадры являются граничными условиями, все фрагменты будут согласованы между собой.

Основной мотивацией предлагаемого метода является подход, который применяют ротоскописты при ручном матировании видео. Они анимируют сплайновый контур, интерполируя его между ключевыми кадрами. На практике такой подход дает хороший результат и скрывает артефакты, т.к. плавное движение контура часто совпадает с реальным движением границ объекта, или по крайней мере аппроксимирует его с достаточной точностью, при которой мелкие дефекты становятся незаметны зрителю.

Этот подход можно расширить, добавив информацию о движении, взятую из видео. Заменив сплайновые контуры маской прозрачности, можно рассматривать задачу матирования видео как задачу интерполяции маски между ключевыми кадрами на основе движения из видео.

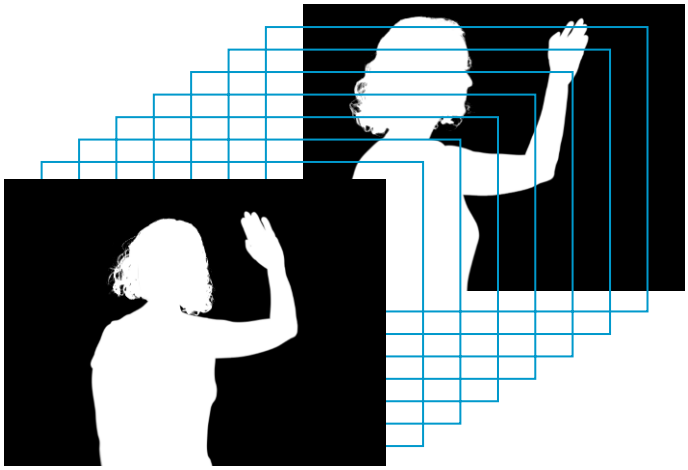


Рисунок 20. Схематичное изображение видеообъема с двумя ключевыми кадрами

Основным наблюдением является тот факт, что оптический поток в канале прозрачности α является гораздо более гладким, чем оптический поток в исходном видео (рис. 21). Кроме того, он определен на границе объекта (при перекрытии объект-фон), в то время как оптический поток изображения не определен в данном случае. Поток в канале прозрачности не определен только при самоперекрытиях, т.е. при перекрытиях объект-объект и фон-фон. Однако в этих случаях прозрачность равна 0 или 1, т.е. невозможность определить поток не влияет на результат (карту прозрачности).

(а) (б)

Рисунок 21. Сравнение оптического потока с потоком в канале прозрачности. Предположим, что ваза вращается вокруг оси симметрии. **(а)** поток соответствует линейной скорости движения точек на поверхности (желтые стрелки); кроме того, он не определен на границе – точки уходят из области видимости на заднюю (скрытую) часть поверхности или приходят из нее. **(б)** Поток маски всюду нулевой, в том числе на границах. Видимый силуэт никак не меняется при вращении вазы.

(а) (б)

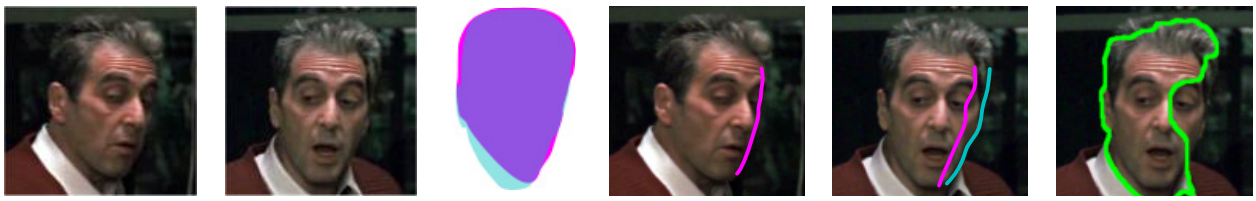
Рисунок 22. Форма маски меняется не сильно, в то время как точки реальной трехмерной поверхности значительно переместились.

Вращение объектов или их частей в плоскости, не совпадающей с экранной, является очень распространенным видом движения в видео. На рис. 22 показан более естественный пример. Форма маски меняется не сильно, в то время как точки реальной трехмерной поверхности значительно переместились. Этот факт активно используется при ручном ротоскопировании, когда маска объекта описывается сплайном, а деформация осуществляется перемещением контрольных точек.

Это приводит нас к идее ввести поток в канале прозрачности («альфа-поток» [53]) аналогично оптическому потоку (23):

$$\alpha_1(x,y) \approx \alpha_2(x+u(x,y), y+v(x,y)). \qquad (24)$$

Таким образом, вместо сплайновой деформации контура, можно моделировать деформацию всей маски с учетом значений прозрачности. Как видно на рис. 23, маска не сильно меняется между кадрами, в отличие от физической трехмерной поверхности, которой она соответствует. Алгоритмы, использующие оптический поток для вычисления движения маски, чувствительны к изменениям ориентации объекта в пространстве и могут выдавать неправильную сегментацию (рис. 23 (е)). Использование альфа-потока позволит избежать этой проблемы, т.к. отслеживаться будет лишь видимый силуэт объекта, а не его трехмерная поверхность.



(а) (б) (в) (г) (д) (е)

Рисунок 23. Иллюстрация принципа альфа-потока: **(а, б)** первый и последний кадры фрагмента, **(в)** маски, соответствующие этим кадрам, **(г)** контур в первом кадре, **(д)** два способа переноса контура на последний кадр – в предположении, что контур лежит на поверхности, и в предположении, что контур является видимой границей объекта, **(е)** результат алгоритма SnapCut [10] – видно, что данный алгоритм предполагает принадлежность контура поверхности, что неверно. Результат предложенного алгоритма см. в разделе 4.6.

Однако, найти альфа-поток напрямую нельзя, т.к. альфа-канал изначально неизвестен, поэтому искать поток и прозрачность надо совместно, либо итерационно. Если формулировать задачу с использованием только одного ключевого кадра, можно использовать алгоритм последовательного отслеживания ([52], [76]) и искать только поток, деформируя им канал прозрачности из ключевого кадра.

В данной работе для вычисления альфа-потока и результирующего канала прозрачности предлагается следующий базовый алгоритм [53]:

Алгоритм 1

1. Вычисление начального приближения для альфа-потока
2. Матирование (вычисление канала прозрачности при фиксированном потоке)
3. Вычисление альфа-потока при фиксированном канале прозрачности
4. Повторение с шага 2

В качестве начального приближения для альфа-потока (шаг 1) можно использовать обычный оптический поток. В данной общей формулировке алгоритма

можно использовать различные алгоритмы вычисления оптического/альфа-потока и матирования, т.е. алгоритм обладает расширяемостью при появлении новых алгоритмов вычисления потока и матирования изображений.

Альфа-поток поток является регуляризацией для α, а не условием связи изображений и прозрачности, как сделано в существующих методах. Такая регуляризация позволяет использовать произвольное число полных и частичных ключевых кадров. Связь прозрачности с изображениями при этом обеспечивается двумя способами:

- использованием обычного оптического потока с небольшим весом
- использованием алгоритма матирования изображений, обобщенного на видеообъем с учетом перекрытий (этот подход изложен в четвертой главе)

На рис. 24 приведена уточненная схема алгоритма с указанием конкретных алгоритмов, применяемых в данной работе (как было сказано выше, общий алгоритм допускает свободный выбор вспомогательных алгоритмов вычисления потока и матирования). Алгоритм использует обычный оптический поток в цветовых каналах RGB в качестве начального приближения. Затем после матирования он уточняется за счет использования альфа-канала. RGB каналы также используются в качестве регуляризации, но с меньшим весом.

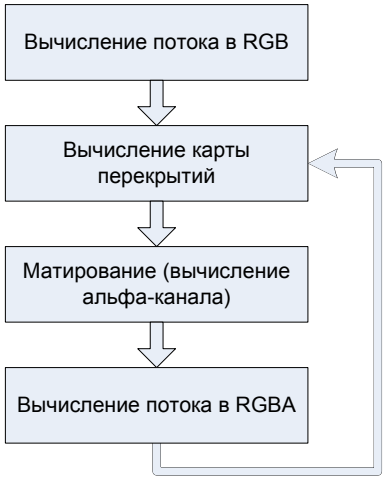


Рисунок 24. Схема алгоритма

2.4. Функционал энергии

Совместное вычисление канала прозрачности и оптического потока в нем может быть сформулировано как задача минимизации следующего функционала энергии:

$$E(V,\alpha,M)=\iiint\left[\left(\frac{\partial\alpha}{\partial V}\cdot M\right)^2+\nu J(I,\alpha)+\mu\left(\frac{\partial I}{\partial V}\cdot M\right)^2+\lambda\|\nabla V\|^2+\rho(M,I,V)\right]dx\,dy\,dt, \qquad (25)$$

где M – бинарная маска видимости,

ρ – условие разреженности маски перекрытий,

J – целевой функционал матирования для каждого кадра,

$\frac{\partial}{\partial V}$ – производная вдоль вектора потока V.

Слагаемые данных $\left(\frac{\partial\alpha}{\partial V}\right)^2$, $\left(\frac{\partial I}{\partial V}\right)^2$ и гладкости потока $\lambda\|\nabla V\|^2$ являются стандартными в задачах вычисления оптического потока ([27]). Коэффициент λ определяет силу сглаживания. Слагаемые данных умножаются на маску перекрытий $M \in \{0, 1\}$, которая позволяет игнорировать векторы потока, относящиеся к перекрытиям, устанавливая значения слагаемых данных в ноль в соответствующих пикселах. Более подробно слагаемые, отвечающие за оптический/альфа-поток будут описаны в следующей главе.

Коэффициент μ отвечает за вклад оптического (RGB-) потока. При $\mu = 0$ имеем чистый альфа-поток. Кроме того, этот коэффициент можно менять в процессе минимизации, установив его в некоторое начальное значение $\mu_0 > 0$ и постепенно уменьшая до нуля. Так можно сгладить переход от начального приближения (оптического потока) к альфа-потоку, что иногда улучшает результат.

J – функционал матирования изображений, применяемый покадрово. Он отвечает за связь между прозрачностью и изображением. Весовой коэффициент ν регулирует вклад покадрового матирования в общий функционал. При $\nu = 0$ на результат влияют лишь ключевые кадры, а информация о прозрачности переносится с них альфа-потоком. При $\nu > 0$ прозрачность может

локально адаптироваться под конкретные кадры. Это может быть полезно при изменении формы/топологии силуэта объекта, а также при появлении размытия движения в некоторых кадрах, т.к. его невозможно учесть при создании ключевых кадров. В качестве функционала покадрового матирования в данной работе взят функционал аналитического матирования [33]. Данный алгоритм уже был описан в разделе 1.2, в частности, выражение для J приведено в формуле (13).

Функция ρ будет описана в четвертой главе. Она накладывает ограничение разреженности на маску M (требует, чтобы в ней не было большого числа нулей, особенно в последовательных кадрах, вдоль потока V), в противном случае существует тривиальное решение $M = 0$, $V = 0$, $\alpha = const$[3].

Решение задачи матирования видеопоследовательности сводится к минимизации функционала (25) с граничным условием на значения прозрачности в ключевых кадрах. Минимизация производится поочередно по переменным V, M, α. Слагаемые функционала энергии описаны подробнее в следующих главах.

Поток V полагается двунаправленным (прямой поток и обратный поток), что обеспечивает симметричность алгоритма по оси времени.

[3] Константная α минимизирует функционал $J(\alpha)$ (13), при этом константа может быть разной для разных кадров. При $M = 0$ связь между кадрами теряется и ключевые кадры никак не повлияют на результат.

2.5. Заключение

В данной главе был предложен алгоритм матирования видеопоследовательности по ключевым кадрам. Преимуществом предложенного алгоритма является временна́я согласованность результата (альфа-канала), достигаемая за счет условия гладкости потока в этом канале, а не в оптическом потоке (который является промежуточным представлением и влияет на результат косвенно).

Алгоритм альфа-потока является итерационным и многократно вызывает алгоритмы вычисления потока и матирования для всех кадров. Отсюда вытекает необходимость сделать данные алгоритмы быстрыми, чему посвящены две следующие главы.

3. ВЫЧИСЛЕНИЕ ОПТИЧЕСКОГО ПОТОКА

В данной главе рассматривается задача вычисления оптического потока как составная часть алгоритма матирования видео.

Оптическим потоком называется карта видимого движения пикселей между кадрами. Задача его поиска формулируется так: по данной паре изображений построить оптический поток $V = (V_x, V_y)$, совмещающий эти изображения.

Оптический поток является естественным способом моделирования движения объектов на изображении между двумя кадрами. При этом неоднозначность в сопоставлении пикселей, связанная с недостатком цветовой информации, устраняется введением регуляризации – условия гладкости оптического потока.

Каждый пиксель изображения можно считать вектором, состоящим из компонент цветовой модели RGB, либо значением прозрачности, либо вектором из компонент RGBA (RGB и значения прозрачности α). Таким образом, любой алгоритм вычисления потока, рассматривающий пиксели как векторы, может работать как с оптическим потоком, так и с альфа-потоком. Для работы с RGBA-потоком, в котором компоненты RGB имеют вес, отличный от веса прозрачности, обычно достаточно умножить эти компоненты на некоторый коэффициент. Например, слагаемое данных в функционале оптического потока является квадратичным, этот коэффициент равен квадратному корню из веса.

В данной работе предложено два новых алгоритма вычисления оптического потока: более быстрый двухкадровый алгоритм (раздел 3.2) и более точный алгоритм на основе траекторий (раздел 3.3).

3.1. Основные подходы к вычислению оптического потока

Пусть требуется построить карту соответствия между двумя изображениями, являющимися соседними кадрами видеопоследовательности I_0 и I_1. С учетом погрешностей регистрации изображений (шум CCD-матрицы, перекрытия, артефакты дискретизации), соответствие между изображениями можно записать как

$$I_0(x,y) \approx I_1\big(x + V_x(x,y), y + V_y(x,y)\big), \tag{26}$$

где V_x, V_y – вертикальная и горизонтальная компоненты видимого движения пикселов между кадрами, называемого оптическим потоком V. Для нецелых значений сдвига может быть применена билинейная или бикубическая интерполяция, либо более простая интерполяция на основе производной:

$$I(x,y) = I(x_0,y_0) + x_\Delta \cdot \frac{\partial I}{\partial x}(x_0,y_0) + y_\Delta \cdot \frac{\partial I}{\partial y}(x_0,y_0), \tag{27}$$

где x_0, y_0 – целые части координат, а x_Δ, y_Δ – дробные:

$$\begin{aligned} x_0 &= \lfloor x \rfloor, \\ y_0 &= \lfloor y \rfloor, \\ x_\Delta &= x - x_0, \\ y_\Delta &= y - y_0. \end{aligned} \tag{28}$$

Частные производные $\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}$ вычисляются как конечные разности между соседними пикселами. Иногда перед вычислением производных изображение размывается, чтобы устранить шум. Однако в приведенном варианте полученное изображение I является разрывным – односторонний предел I в точке (x–0, y–0) равен $I(x,y) + \frac{\partial I}{\partial x}(x,y) + \frac{\partial I}{\partial y}(x,y)$ и может не совпадать со значением в точке (2, 2). Поэтому предпочтительнее использовать билинейную интерполяцию:

$$\begin{aligned} I(x,y) = {} & I(\lceil x \rceil, \lceil y \rceil) \cdot x_\Delta \cdot y_\Delta + I(\lfloor x \rfloor, \lceil y \rceil)(1 - x_\Delta) \cdot y_\Delta + \\ & + I(\lceil x \rceil, \lfloor y \rfloor) \cdot x_\Delta \cdot (1 - y_\Delta) + I(\lfloor x \rfloor, \lfloor y \rfloor) \cdot (1 - x_\Delta) \cdot (1 - y_\Delta) \end{aligned} \tag{29}$$

или интерполяцию более высокого порядка.

Поскольку для каждого пиксела *(x, y)* изображения I_0 можно найти много пикселов в изображении I_1 с совпадающим или близким цветом, решение для V не единственно. Поэтому задачу нахождения оптического потока обычно формулируют как минимизацию функционала, состоящего из члена данных E_D и члена гладкости E_S, умноженного на коэффициент регуляризации $\lambda > 0$ (см. [27]):

$$E(V) = E_D(V) + \lambda E_S(V), \tag{30}$$

$$E_D(V) = \iint \left(I_0(x,y) - I_1\left(x + V_x, y + V_y\right)\right)^2 dxdy, \tag{31}$$

$$E_S(V) = \iint \left\|\nabla V_x\right\|^2 + \left\|\nabla V_y\right\|^2 dxdy, \tag{32}$$

$$V = \arg\min_V E(V) \tag{33}$$

Возможны и другие способы регуляризации, использующие неквадратичные нормы для данных слагаемых [55], например:

$$E_D(V) = \iint \psi\left(I_0(x,y) - I_1\left(x + V_x, y + V_y\right)\right) dxdy, \tag{34}$$

$$E_S(V) = \iint \psi(\nabla V) dxdy, \tag{35}$$

$$\psi(z) = \sqrt{z^2 + \sigma^2} \text{ для малого } \sigma. \tag{36}$$

Основная цель такой регуляризации – учесть перекрытия и разрывы в потоке.

Однако робастные нормы лишь примерно интерполируют поток в области перекрытий, используя значения из соседних пикселов. Выбор подходящей нормы в слагаемом гладкости позволяет добиться правдоподобного профиля интерполяции для определенных классов объектов, например, для объектов с гладким контуром или объектов, выделяющихся по цвету – в этом случае норма ставится в зависимость от изображения I_0 ([41]).

Некоторые методы поиска перекрытий помимо прямого потока вычисляют обратный поток V' от I_1 к I_0 и сравнивают их, помечая пикселы $|V + V'| > \delta$ как перекрытия для некоторого фиксированного порога δ (обратный поток берется со знаком «+», т.к. он в сумме с прямым потоком в идеале должен давать 0), либо используют фиксированный штраф для перекрытых

пикселов [70]. Иногда принимается во внимание пространственная разреженность карты перекрытий, например, путем фильтрации полученной бинарной карты [70].

Однако в силу неоднозначности потока возможны случаи, когда при похожих цветах объектов в изображениях слагаемое данных получается слабым и поток V, V' различаются лишь из-за регуляризации, которая оптимизируется независимо для прямого и обратного потока. Возможна формулировка совместной минимизации с общей регуляризацией, но она не улучшает ситуацию, т.к. результирующий функционал имеет много локальных минимумов.

Другая проблема – выбор порога отсечения перекрытий, т.к. количество перекрытых пикселов может существенно различаться в разных кадрах. Одним из решений является добавление в задачу неизвестной бинарной маски перекрытий и попиксельное умножение слагаемого данных E_D на эту маску. Такой подход приводит к тому, что глобальный минимум E достигается при отметке всех пикселов как перекрытых. Попытка выбора оптимального штрафа за избыток перекрытий равносильна проблеме поиска порога δ, т.е. не может быть решена сразу для всех кадров.

Непрерывные методы

В непрерывных методах поиска оптического потока задача формулируется как минимизация следующего функционала энергии ([27], [5]):

$$E(V) = \iint \left[\left(\frac{\partial I}{\partial V} \right)^2 + \lambda \|\nabla V\|^2 \right] dx\, dy, \qquad (37)$$

где $\frac{\partial}{\partial V}$ – производная вдоль вектора потока V,

λ – коэффициент гладкости потока.

При этом производная I по времени может быть записана только дискретным образом как конечная разность для двух изображений I_0, I_1:

$$\frac{\partial I}{\partial V}(x,y) \approx I_0(x,y) - I_1\big(x + V_x(x,y),\ y + V_y(x,y)\big). \qquad (38)$$

Сами значения компонент вектора сдвига V не обязаны быть целочисленными, т.к. используется интерполяционная формула (27) или (29).

Дискретные методы

В дискретных методах компоненты векторов оптического потока могут быть только целочисленными. Для достижения субпиксельной точности можно увеличить исходное изображение в нужное число раз каким-либо методом интерполяции (но при этом возрастает время работы алгоритма).

Диапазон значений (V_x, V_y) ограничивается множеством $\{-M, \ldots, M\}^2$, где M – максимальный сдвиг в пикселах.

Проблема перекрытий

Основной сложностью при поиске оптического потока между кадрами является обработка перекрытий – некоторые области (i+1)-го кадра не видны на i-ом, т.к. закрыты движущимися объектами, однако локализация таких областей является недоопределенной задачей. Таким образом вычисленный оптический поток в окрестностях границ объектов является некорректным.

Один из вариантов решения проблемы перекрытий предложен в алгоритме траекторного потока (раздел 3.3). В двухкадровом алгоритме (раздел 3.2) данная проблема напрямую не решается, вместо этого предлагается отсеивать перекрытия на стадии матирования (глава 4).

Многокадровые алгоритмы

Некоторые алгоритмы вычисления потока анализируют сразу несколько последовательных кадров. В алгоритме [28] предполагается, что траектории всех точек образуют линейное пространство малой размерности. Такой подход работает только для статичных сцен с движущейся камерой. Обобщение этого метода [23] допускает присутствие деформирующихся объектов, но требует, чтобы пространство деформаций имело малую размерность.

Метод [67] использует траектории, но предполагает, что все их точки видимы во всех кадрах, т.е. не решает проблему перекрытий. В статьях [3], [4] предложено двойственное представление пространства траекторий и доказана его эквивалентность пространству форм деформируемого объекта. Однако и там не решается проблема перекрытий, и по-прежнему действует ограничение на размерность пространства деформаций, поэтому данный метод применяется в основном для многоракурсной 3D-реконструкции [75].

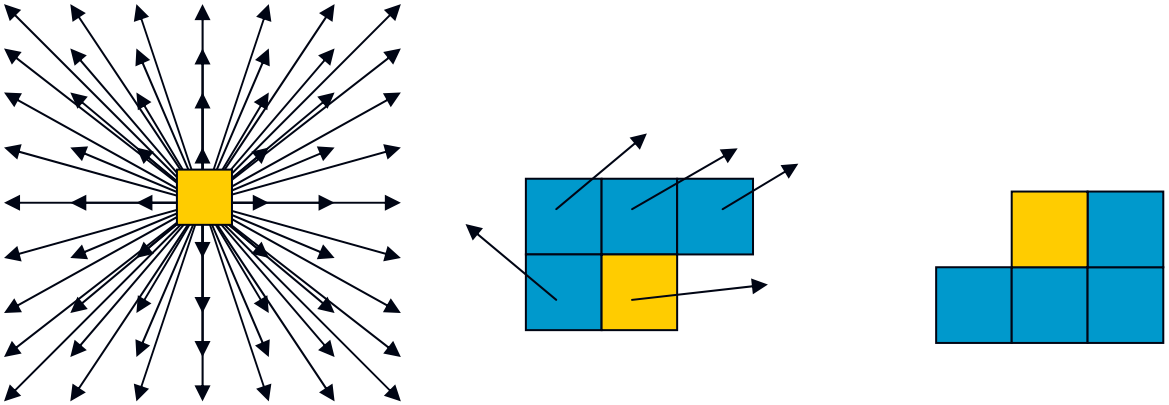
3.2. Предлагаемый двухкадровый алгоритм

Предложенный двухкадровый метод является комбинацией непрерывного и дискретного подходов и использует метод раздельной оптимизации для поиска минимума функционала энергии [54]. В данном методе неизвестное поле V дополняется аналогичным полем U и слагаемые энергии применяются к этим полям независимо. Также добавляется мягкое условие равенства U и V:

$$E(U,V) = \iint \left[\left(\frac{\partial I}{\partial V} \right)^2 + \theta (U-V)^2 + \lambda \left\| \nabla U \right\|^2 \right] dx\, dy\,. \tag{39}$$

Устремляя параметр θ к бесконечности, получим минимум данной энергии, в котором U = V, и достигается совместный минимум слагаемых данных и гладкости. Минимизация производится итерационно с увеличением параметра θ. Каждое из слагаемых само по себе эффективно минимизируется предназначенным для него алгоритмом.

Слагаемое данных минимизируется переборным алгоритмом на основе алгоритма PatchMatch [13]. Вместо патчей используются единичные пикселы, т.к. за пространственную когерентность отвечает слагаемое гладкости потока. Основная идея метода – оптимизация перебора. Пусть область поиска имеет размер M×M. Тогда вместо перебора M^2 значений в каждом пикселе осуществляется выбор одного случайного значения в каждом пикселе, а затем используется двухпроходное распространение решения на соседние пикселы. Ввиду пространственной когерентности (обусловленной как структурой естественных изображений, так и непосредственно слагаемым гладкости) многие связные области пикселов имеют почти одинаковые значения потока, поэтому если оптимальный вектор найден хотя бы в одном пикселе области, он распространится на всю область. Шаблон обработки пикселей показан на рис. 25.



(а) (б) (в)

Рисунок 25. Шаблоны перебора вектора сдвига: **(а)** полный перебор (M^2 возможных векторов в каждом пикселе), **(б)** Перебор методом PatchMatch – вектор потока выбирается из 5 кандидатов при прямом проходе (текущий пиксел и 4 ранее обработанных пиксела), **(в)** шаблон обратного прохода PatchMatch.

Альфа-поток получается заменой I на α:

$$E_{\alpha}(V) = \iint \left[\left(\frac{\partial \alpha}{\partial V} \right)^2 + \lambda \left\| \nabla V \right\|^2 \right] dx\, dy\,. \tag{40}$$

В качестве дополнительной регуляризации, которая важна на первых итерациях, когда значения α не достаточно точны, используется поток для каналов RGBA, т.е. энергия является суммой энергий оптического и альфа- потоков:

$$E_{RGB\alpha}(V) = \iint \left[w_{RGB} \left(\frac{\partial I}{\partial V} \right)^2 + \left(\frac{\partial \alpha}{\partial V} \right)^2 + \lambda \left\| \nabla V \right\|^2 \right] dx\, dy\,, \tag{41}$$

где w_{RGB} – вес цветовых каналов.

3.2.1 Ограничения на входные данные

Для корректной работы алгоритма требуется, чтобы на этапе стохастического поиска нашелся корректный вектор потока хотя бы в одном пикселе каждой области, в которой поток является константным. Тогда двухпроходной алгоритм распространения скопирует этот вектор во все пикселы области. Этап сглаживания ослабляет требование константности потока: достаточно, чтобы поток приблизительно линейно или аффинно зависел от координаты, однако при этом желательно найти два корректных вектора потока в каждой такой области. Разбив область с линейным потоком на две части, в каждой из которых корректно найден средний константный поток, получим корректный линейный поток при сглаживании всей этой области. При этом явно искать какое-либо разбиение не требуется, достаточно работать на уровне пикселов.

Пусть максимальная скорость движения равна K пикселов на кадр. Тогда окно поиска имеет размер $M = 2K + 1$ и содержит M^2 возможных векторов потока. Вероятность «угадывания» корректного вектора потока в одном пикселе равна $1 / M^2$. Среднее количество необходимых пикселов для одного успеха вычисляется как математическое ожидание геометрического распределения и равно M^2. Тогда область, соответствующая недеформируемой части объекта (т.е. движущейся аффинно), должна иметь площадь $\approx 2M^2$ пикселов. Если наименьшая недеформируемая область объекта в кадре имеет площадь A пикселов, то максимальную скорость движения, при которой алгоритм сработает, можно оценить как $K = \frac{\sqrt{A/2}-1}{2} \approx \frac{\sqrt{A}}{8}$, т.е. объект должен сдвигаться за кадр не более, чем на одну восьмую своего линейного размера. Например, в случае человека размер областей, движущихся аффинно (туловище, руки) можно взять за 40 см. Тогда скорость равна 5 см/кадр, что при частоте 25 кадров/с дает скорость 1,25 м/с = 4,5 км/ч, что соответствует скорости ходьбы человека. Эта оценка не зависит от расстояния до камеры и

разрешения изображения, однако верна лишь при правильно выставленном размере окна поиска K.

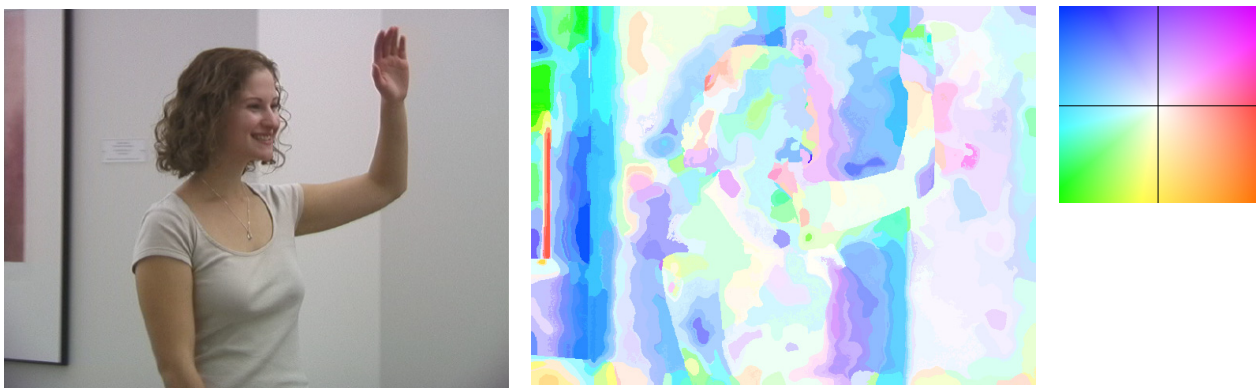
3.2.2 Экспериментальная оценка

На данный момент отсутствует хороший способ получить эталонный оптический поток для реальных данных. Основной тестовой базой является [12], но эталонные данные для потока доступны лишь для простых сцен с линейным/аффинным движением. Эталонные данные доступны только для синтетических сцен, но они недостаточно реалистичны и движение в большинстве них также линейно.

Поэтому было решено оценивать оптический поток по результату матирования, т.е. в составе всего алгоритма. Пример результата алгоритма показан на рис. 26. На рисунке для визуализации векторного поля потока используется цветовой ключ, предложенный в статье [12] – насыщенность цвета кодирует длину вектора потока (скорость в точке), а цветовой тон – направление. При этом для нулевого вектора направление не определено, что также отражено при таком кодировании: для цвета нулевой насыщенности (белого) цветовой тон не определен. Еще одно полезное свойство такого кодирования: при печати в градациях серого теряется только информация о направлении, но сохраняется модуль векторов.

3.2.3 Время работы

Предложенный алгоритм требует примерно 2 секунды на кадр размером 640x480. Для сравнения в таблице 2 приведено время работы алгоритмов из сравнения Middlebury [12]. Видно, что оно существенно выше. Время работы является критическим, т.к. в предложенном алгоритме матирования видео оптический поток нужно вычислять для всех кадров по нескольку раз (см. раздел 2.3).



(а) (б) (в)

Рисунок 26. Результат предложенного алгоритма. **(а)** кадр I_0, **(б)** посчитанный оптический поток, **(в)** цветовой ключ для визуализации оптического потока.

Таблица 2. Время работы алгоритмов из сравнения [12]. Алгоритмы упорядочены по качеству результата. Предложенный алгоритм работает ~2 секунды.

Алгоритм	Время работы, с
MDP-Flow2	342
NN-field	1535
ADF	187
Layers++	18206
nLayers	36150
Sparse-NonSparse	713
ALD-Flow	61
COFM	600
Efficient-NL	400
TC-Flow	2500

3.3. *Предлагаемый траекторный алгоритм*

Основная идея алгоритма заключается в использовании нескольких кадров сразу для нахождения перекрытий и уточнения слагаемого данных [77]. В данном алгоритме рассматривается T = 2K + 1 кадров с номерами –K, …, K. Финальным результатом является оптический поток из кадра I_0 в I_1 и обратный поток из I_0 в I_{-1}, но при этом в качестве промежуточного результата находятся 2K потоков из I_0 в остальные кадры. В каждом пикселе кадра I_0 эти потоки образуют траекторию.

Вместо пространственной разреженности перекрытий, применяемой в двухкадровых алгоритмах, рассматривается их временна́я разреженность. Предполагается, что каждая точка кадра I_0 видна по меньшей мере в K+1 последовательном кадре, включая I_0. Тогда можно ввести карту видимости $\rho \in \{0, ..., K\}$, значения которой означают, что пиксел является видимым в кадрах $\rho - K, \ldots, K$. Таким образом, слагаемое данных будет применяться только к этим кадрам, что соответствует идее сортирующего суммирования [30], применяемой в алгоритмах стереосопоставления.

Т.к. потоки вычисляются относительно кадра I_0, их можно считать одним «траекторным» потоком, где каждому пикселу сопоставлен 4K-мерный (или, что то же самое, 2(T – 1)-мерный) вектор потока

$$\vec{V} = \left(V_{-K,x}, V_{-K,y} \ldots, V_{-1,x}, V_{-1,y}, V_{1,x}, V_{1,y}, \ldots, , V_{K,x}, V_{K,y}\right). \quad (42)$$

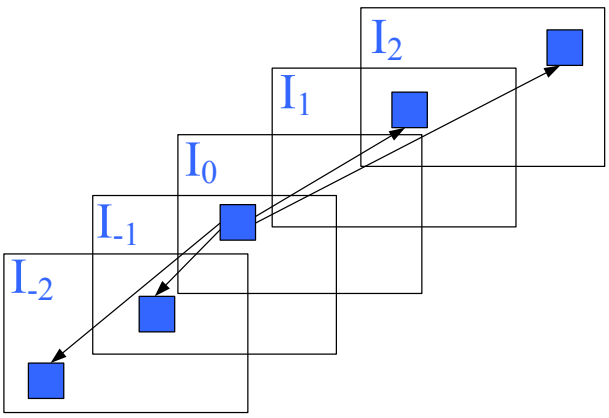


Рисунок 27. Иллюстрация траекторного потока.

Такое векторное представление (проиллюстрированное на рис. 27) позволяет использовать любые известные функционалы гладкости, определенные для двухмерного потока, которые могут быть сформулированы в терминах векторной алгебры, а также многие алгоритмы минимизации данных функционалов (например, [55], [60]). Исключением будут алгоритмы, явно использующие факт двухмерности, а также алгоритмы, несовместимые с членом данных E_D (однако их часто можно адаптировать для данной задачи, либо использовать методы раздельной оптимизации, используя другой алгоритм для члена данных). Кроме того, стохастические и переборные алгоритмы могут потерять свою эффективность из-за увеличения пространства поиска.

Метод траекторного потока использует следующий функционал энергии:

$$E(V,\rho) = E_D(V,\rho) + \lambda E_S(V) + \mu E_T(V) + \nu E_\rho(\rho), \tag{43}$$

в котором слагаемое данных определено с учетом видимости:

$$E_D(V,\rho) = \frac{1}{K}\sum_x \sum_y \sum_{\substack{t=\rho(x,y)-K, \\ t\neq 0}}^{\rho(x,y)} \left(I_0(x,y) - I_t\left(x+V_{t,x}, y+V_{t,y}\right)\right)^2 \quad , \tag{44}$$

слагаемое пространственной гладкости уравнивает разброс значений по разным кадрам, путем деления на |t|:

$$E_S(V) = \frac{1}{T-1}\sum_x \sum_y \left[\sum_{\substack{t=-K, \\ t\neq 0}}^{K} \left(\frac{\nabla V}{|t|} \right)^2 \right]^\gamma , \tag{45}$$

при этом норма градиента является инвариантной к повороту при $\gamma = 0{,}5$, что в двухмерном случае записывается как

$$\psi(V) = \sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2} , \tag{46}$$

причем суммирование по $\underline{t}$ также производится под знаком радикала. Это означает, что в случае разрывности решения разрыв потока в некоторой точке происходит одновременно в горизонтальной и вертикальной компонентах

потока и сразу во всех кадрах. Таким образом, траектории в двух соседних пикселах не могут разойтись, а через несколько кадров слиться в единый объект.

На практике используется значение $\gamma = 0{,}45$, как рекомендовано в статье [55], порождающее невыпуклую норму, которая «поощряет» более четкие края в случае разрывного потока.

Деление на $|t|$ в формуле (45) уравнивает разброс значений по разным кадрам, т.к. он возрастает при удалении от центрального кадра.

Энергия временно́й гладкости использует вторую производную потока по времени, т.е. поощряет траектории, близкие к линейным:

$$E_T(V) = \frac{1}{T-2} \sum_x \sum_y \sum_{t=-K+1}^{K-1} (V_{t-1} - 2V_t + V_{t+1})^2 , \tag{47}$$

при этом полагается $V_0 = 0$.

Коэффициенты $1 / K$, $1 / (T - 1)$, $1 / (T - 2)$ в формулах (44), (45), (47) соответствуют количеству слагаемых на пиксел, зависящему от числа рассматриваемых кадров T, что упрощает подгонку коэффициентов λ, μ, ν при изменении числа кадров.

Функционал гладкости карты видимости также является квадратичным:

$$E_\rho(\rho) = \sum_x \sum_y (\nabla \rho(x, y))^2 , \tag{48}$$

При этом градиент означает конечную разностью ввиду дискретности значений ρ. Выбор квадратичной функции штрафа обусловлен структурой перекрытий в видео: движущийся объект оставляет за собой «след» из перекрытий, каждое из которых сдвинуто на 1 кадр относительно предыдущего, поэтому скачки в карте видимости, превышающие 1 кадр, не желательны – они скорее всего соответствуют неправильно найденным перекрытиям. Надежными перекрытиями являются такие, которые последовательно возникают по пути движения объекта, поэтому имеет смысл искать перекрытия именно с такой структурой.

Также можно ввести два дополнительных члена, связанных с картой видимости.

Один из них запрещает выход видимой части траектории за края изображения, т.к. обращает энергию в бесконечность. Таким образом, данная ситуация рассматривается как обычное перекрытие, в отличие от алгоритмов двухкадрового потока, где такие случаи приходится обрабатывать отдельно, чаще всего обнулением производных для векторов потока, выходящих за края изображения.

Второй дополнительный член поощряет, с очень маленьким весом, близость ρ к значению $K / 2$, т.е. симметрию диапазона видимости. Такая регуляризация улучшает маску видимости для выходного потока V_1, т.к. иначе в простых случаях при отсутствии перекрытий может быть выбран диапазон видимости $-K, \ldots, 0$ и результирующий поток будет менее информативным.

На рис. 28 показан пример траекторий для одномерного случая. Видимые части траекторий не перекрываются с соседними (хотя явного условия, запрещающего такие перекрытия нет), что говорит о корректности работы алгоритма (т.е. слагаемые гладкости и карта видимости правильно распознают траектории, относящиеся к разным движущимся слоям).

3.3.1 Минимизация

Для минимизации функционала энергии используется метод глобальной оптимизации QPBO [45]. Данный метод является бинарным и «склеивает» решения-кандидаты. В работе предложено несколько эвристик для генерации такого решения. Часть из них заключается в небольшом возмущении текущего решения (например, сдвиг на 1 пиксел по оси x), другая часть осуществляет локальную оптимизацию неполного функционала энергии (включающего не все слагаемые).

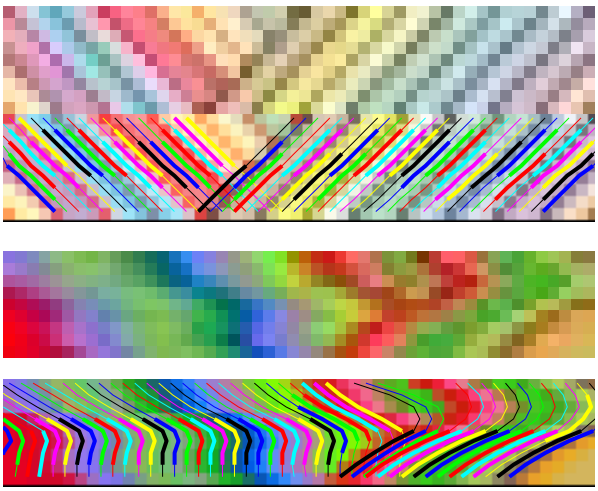


Рисунок 28. Искусственные одномерные примеры траекторного потока. Показаны исходные одномерные видеопоследовательности (T = 9 кадров, по вертикали – ось времени) и найденные траектории. Толстые части траекторий – диапазоны видимости.

Будем осуществлять минимизацию на двухмерной сетке, соответствующей изображению I_0. Каждому узлу сетки соответствует вектор потока $\vec{V}$ (см. обозначение (42)) и значение ρ. Можно строить решения-кандидаты, состоящие из пар $(\vec{V}, \rho)$, но можно сделать их независимыми, разместив их на двух сетках (рис. 29).

Первая сетка содержит пространственные ребра, отвечающие за слагаемое E_S. Узлы содержат унарный потенциал E_T. Ребра второй сетки отвечают за слагаемое E_ρ. Ребра, соединяющие соответственные узлы первой и второй сеток, отвечают за слагаемое E_D.

Инвариантные к повороту нормы не могут быть заданы с помощью ребер двухмерной сетки, поэтому используется аппроксимация – каждая вершина соединяется не с 4, а с 16 ближайшими [14].

3.3.2 Начальное приближение

В качестве начального приближения вычислим двухкадровые потоки между последовательными кадрами в прямом и обратном направлении от центрального кадра: $I_0 \to I_1$, $I_1 \to I_2$, ...; $I_0 \to I_{-1}$, $I_{-1} \to I_{-2}$, ... Затем выполним

рекурсивную трансформацию этих потоков, чтобы получить потоки между центральным кадром и остальными кадрами. Для вычисления двухкадровых потоков использовалась библиотека mexOpticalFlow [36].

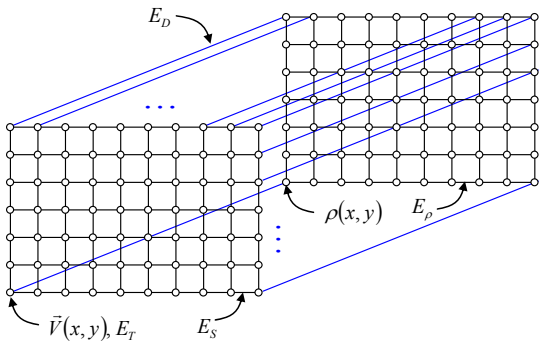


Рисунок 29. Структура графа для совместной минимизации V и ρ методом QPBO

3.3.3 Решения-кандидаты

Будем генерировать решения-кандидаты, используя несколько различных эвристик, и сливать их с текущим решением методом QPBO. Алгоритм слияния [32] гарантирует невозрастание энергии, что снижает требования к решениям-кандидатам.

Решения-кандидаты для траекторного потока $\vec{V}$:

1. Всевозможные сдвиги всей сетки на ±1 пиксел по каждой оси (т.е. значение потока $\vec{V}$ копируется из соседнего пиксела, компоненты вектора при этом не меняются)
2. Выбор траектории $\vec{V}$ в случайном пикселе *(x, y)* и использование его в качестве константного кандидата. Т.к. сцены часто состоят из нескольких объектов, вместо равномерного распределения для *(x, y)* сначала осуществляется сегментация потока методом «k средних», затем случайно выбирается сегмент, а в нем – случайный пиксел. Это нужно, чтобы объекты большей площади не имели преимущества.
3. Двухмерное гауссово размытие потока $\vec{V}$

4. Один шаг градиентного спуска для слагаемых, отвечающих за данные на двухмерной сетке (т.е. без учета пространственной гладкости): $E_D(V, \rho) + \mu E_T(V)$
5. Попиксельная минимизация $E_D(V, \rho) + \mu E_T(V)$ по $\vec{V}$ методом динамического программирования. Используется ускоренный приближенный вариант, основанный на идее алгоритма PatchMatch [13]. Вместо выбора наилучшего кандидата $\vec{V}$ по шаблону из 5 пикселов предлагается строить новое решение, вообще говоря не совпадающее с этими 5 кандидатами (но совпадающее с одним из них в каждом кадре). Для этого используется алгоритм динамического программирования. Метод проиллюстрирован на рис. 30.

Как видно, часть решений-кандидатов соответствует копированию «удачных» траекторий $\vec{V}$ из одних пикселей в другие, другая часть генерирует новые решения в окрестности текущего решения.

Кандидаты для карты видимости являются константами $\rho = 0, \ldots, K$ (также были опробованы сдвиги и увеличение/уменьшение ρ на 1, но эти варианты не дают существенного улучшения по сравнению с использованием констант).

Кандидаты для всех переменных $(\vec{V}, \rho)$ строятся в виде декартова произведения описанных кандидатов [62], [56]. Для каждого кандидата $\vec{V}$ пробуем каждый кандидат ρ), за исключением 1-го типа кандидатов ρ (сдвиг) – вместо всевозможных сдвигов выбирается тот же самый сдвиг, что и для $\vec{V}$, либо нулевой сдвиг, если кандидат $\vec{V}$ получен не сдвигом/копированием (т.е. тогда оптимизируется только $\vec{V}$ при фиксированном ρ), либо локально-оптимальное значение ρ в каждом пикселе для кандидата 5 (динамическое программирование).

Т.к. кандидаты зависят от текущего решения, цикл оптимизации надо повторять до тех пор, пока полная энергия не перестанет уменьшаться.

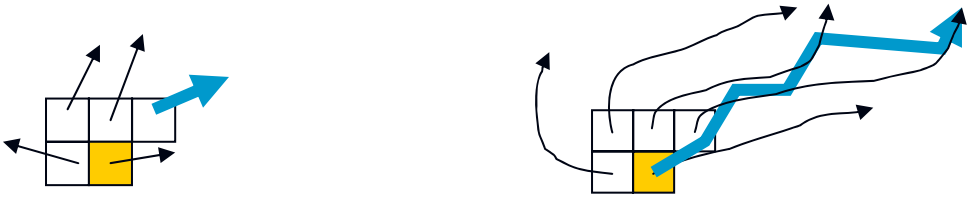


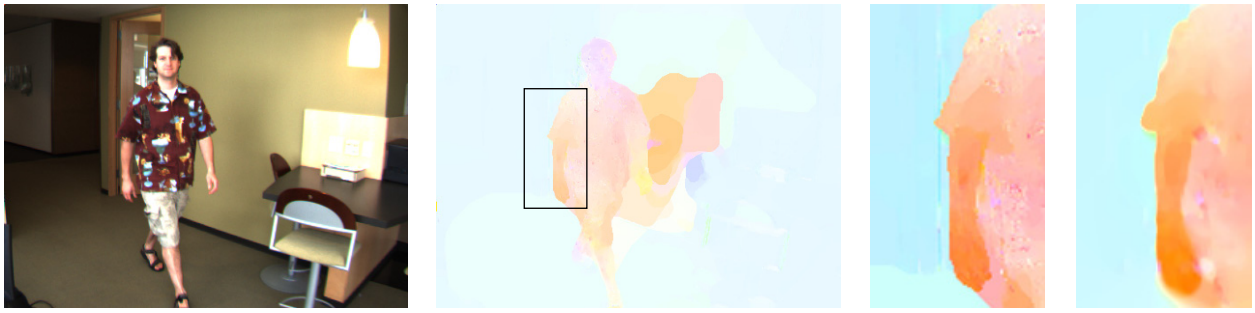
Рисунок 30. Обобщение алгоритма PatchMatch для траекторий: (а) оптический поток – выбор наилучшего вектора из 5 кандидатов; (б) траекторный поток – синтез новой траектории на основе 5 кандидатов.

3.3.4 Иерархический подход

Аналогично двухмерным методам оптического потока, возможно применение иерархического подхода с уменьшением размера всех кадров до небольшого размера (например, 40x30 пикселей), а затем поэтапным увеличением до исходного размера ([62], [36]). При этом, после каждого уровня иерархии можно выполнить дополнительное слияние: решение с предыдущего уровня плюс двухмерный поток для данного разрешения, преобразованный в траектории (как описано в подразделе «начальное приближение»).

(а) (б) (в) (г) (д)

Рисунок 31. Результаты на тестовой базе Middlebury. (а-г) изображения и полученный оптический поток, (д) цветовой ключ визуализации потока.



(а) (б) (в) (г)

Рисунок 32. Сравнение с алгоритмом [36]. **(а)** изображение I_0 из тестовой базы [12], T = 7 кадров, **(б)** поток V_1 , **(в)** увеличенный фрагмент, выделенный рамкой в (б), **(г)** результат алгоритма [36]. Предложенный алгоритм выдает более точный контур в случае перекрытия (разрывного потока), в то время как [36] сглаживает поток, что некорректно.

3.3.5 Результаты

Экспериментальная оценка метода проводилась на тестовой базе Middlebury [12]. Примеры приведены на рис. 31, 32. По сравнению с результатом алгоритма mexOpticalFlow [36] контур объекта более точный и не содержит гладких переходов между объектом и фоном, которые видны в виде белой окантовки на рис. 32 (г). Это достигается за счет построения карты видимости и совместной оптимизации видимости и потока.

На рис. 33 сравнивается результат матирования с использованием двухкадрового оптического потока (раздел 3.2) и траекторного потока. Траекторный поток требует значительно больше времени на вычисление, но обеспечивает бóльшую межкадровую согласованность карты прозрачности, особенно вблизи перекрытий. Артефакты вблизи краев кадра также устраняются, благодаря тому, что края кадра рассматриваются как обычные перекрытия (что невозможно в двухкадровом методе).

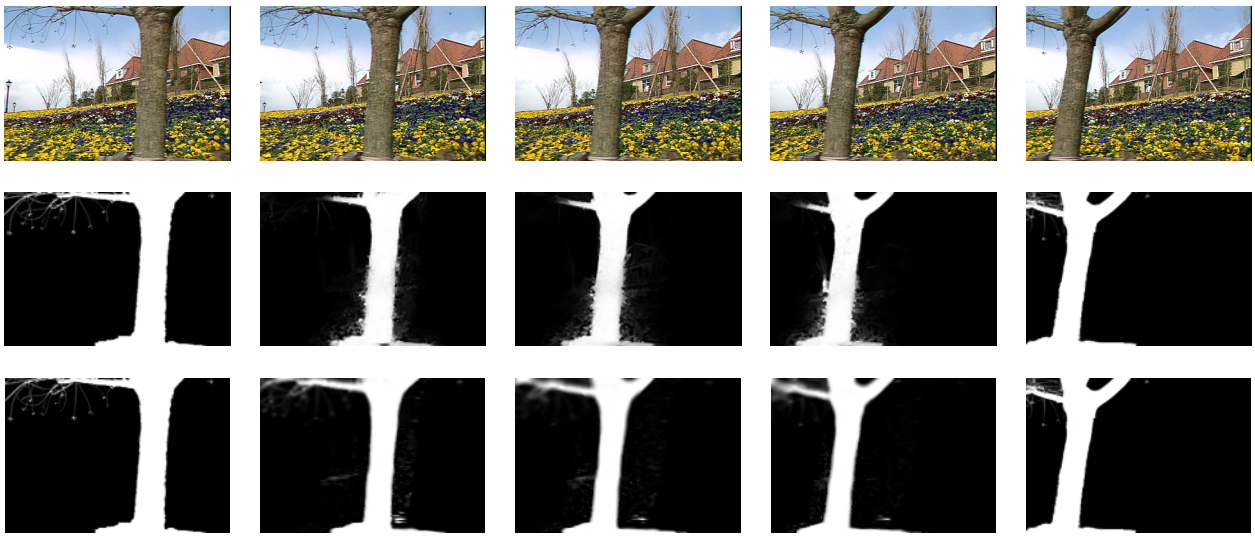


Рисунок 33. Сравнение двухкадрового и траекторного потока применительно к матированию видео. Сверху: исходная видеопоследовательность – статичная сцена, снятая движущейся камерой (объект переднего плана порождает перекрытие в каждом кадре). В середине: результат матирования на основе двухкадрового потока (первый и последний кадры – ключевые). Снизу: результат матирования на основе траеткорного потока.

Также улучшается четкость границ без применения фильтра разреженности, который применялся в двухкадровом методе (см. раздел 4.4). Однако ввиду высокой вычислительной сложности алгоритм применялся к уменьшенному изображению, поэтому мелкие детали получились размытыми.

3.3.6 Возможное упрощение

Рассмотрим упрощенную формулировку. Уберем гладкость карты видимости из полной энергии (43):

$$E(V) = E_D(V) + \lambda E_S(V) + \mu E_T(V), \tag{49}$$

и будем минимизировать видимость независимо в каждом пикселе в слагаемом данных:

$$E_D(V) = \frac{1}{K} \sum_x \sum_y \min_{\rho=0,K} \sum_{\substack{t=\rho-K, \\ t\neq 0}}^{\rho} \left(I_0(x,y) - I_t\left(x + V_{t,x}, y + V_{t,y}\right)\right)^2 . \tag{50}$$

В данном случае теряется гладкость карты видимости. Такой вариант соответствует сортирующему суммированию в задаче многоракурсного стерео, когда в качестве слагаемого данных в каждом пикселе выбирается сумма по K кадрам, имеющим наименьшую рассогласованность по цвету в данном пикселе (что соответствует вероятному отсутствию перекрытия в данных кадрах). В отличие от задачи стерео, изображения в видео упорядочены, поэтому суммирование всегда осуществляется по последовательному диапазону кадров. Однако значение потока разное между парами кадров, в то время как в стерео достаточно одного значения диспаритета, определяющего сдвиги между любой парой кадров.

Формулировка (49), (50) упрощает минимизацию и позволяет использовать существующие алгоритмы оптического потока. Для дискретных алгоритмов минимизацию по ρ в (50) можно выполнять на лету. Для непрерывных алгоритмов (требующих дифференцируемость слагаемого данных) можно либо чередовать минимизацию по ρ и по V, либо использовать дифференцируемые аналоги функции «min», основанные на сигма-функции или аппроксимации усредненного поля [73], [74]:

$$\min(z_1, z_2, ...) \approx -\frac{1}{\beta}\ln\left(e^{-\beta z_1} + e^{-\beta z_2} + ...\right) \tag{51}$$

для достаточно большого β.

4. МАТИРОВАНИЕ ВИДЕООБЪЕМА С УЧЕТОМ ПЕРЕКРЫТИЙ

В данной главе описывается алгоритм матирования видеообъема. Он является частью алгоритма матирования видео, описанного в главе 2. Задача алгоритма – вычисление канала прозрачности для видео, используя ключевые кадры (глава 1) и оптический поток (глава 3). Для этого требуется минимизировать функционал (25) при фиксированном оптическом потоке V.

Если предположить, что в видео нет перекрытий и оптический поток всюду корректен (M = 1), задача сведется к минимизации квадратичной формы

$$\sum_{p,q} A_{p,q}\left(\alpha_p - \alpha_q\right)^2 , \tag{52}$$

где p, q – индексы пикселов (однозначно определяющие номер кадра и координату пиксела в видеообъеме), а A – матрица весовых коэффициентов. При этом $A_{p,q} \neq 0$ для соседних пикселов одного кадра (весовой коэффициент определяется функционалом J, отвечающим за матирование внутри кадра) и для пикселов двух последовательных кадров, соединенных вектором оптического потока. Предполагается, что поток целочисленный (он может быть получен округлением вещественного потока).

Условная минимизация квадратичной формы осуществляется решением системы линейных алгебраических уравнений

$$A\vec{\alpha} = b , \tag{53}$$

где правая часть b содержит значения прозрачности из ключевых кадров. Вектор $\vec{\alpha}$ имеет размер W×H×T, где W×H – размер кадра в пикселах, T – число кадров (т.е. полный размер видеообъема в пикселах). Данная система уравнений большая, но разреженная, и может быть решена методом сопряженных градиентов.

Коэффициенты для внутрикадровых слагаемых квадратичной формы подробно описаны в статье [33] применительно к матированию изображений

(52) они войдут с коэффициентом ν из формулы (25)), поэтому остановимся на межкадровых слагаемых.

Они имеют вид

$$M \cdot \left(\alpha(x,y,t) - \alpha(x+u, y+v, t+1)\right)^2, \tag{54}$$

где M – бинарная маска видимости, u, v – компоненты вектора оптического потока. Аналогично, обратный оптический поток V' устанавливает связь пиксела с пикселом предыдущего кадра

$$M' \cdot \left(\alpha(x,y,t) - \alpha(x+u', y+v', t-1)\right)^2, \tag{55}$$

причем маска перекрытий у обратного потока своя (M').

При M = 1 все векторы потока влияют на результирующую прозрачность. Однако при наличии перекрытий получится, что векторы потока связывают пикселы объекта и фона, что приведет к «растеканию» карты прозрачности. Чтобы этого избежать, надо высилить корректную маску перекрытий M и исключить такие векторы из слагаемых квадратичной формы. В следующем разделе предложен метод нахождения перекрытий путем сегментации видеообъема на цепочки пикселов – временны́е суперпикселы.

Идея исключать ограничения приблизительного равенства возникает, например, в задаче трехмерной реконструкции по освещению [2]. Авторы [2] утверждают, что прореживание ограничений лучше делать детерминированным алгоритмом (для своей задачи авторы применяют минимальное покрывающее дерево для графа ограничений), чем стохастическим алгоритмом, таким как RANSAC.

После сегментации видеообъема на суперпикселы нужно отбросить векторы потока, соединяющие два разных суперпиксела, оставив только те, которые проходят внутри одной цепочки. Далее будет показано, что можно упростить систему линейных уравнений (53), предположив, что каждый пиксел в цепочке имеет одну и туже прозрачность, т.е. приблизительное равенство прозрачности в формуле (24) заменяется на строгое равенство всюду, где M = 1, и полностью игнорируется при M = 0.

Предложенный подход учитывает возможные перекрытия, т.е. неявным образом задает функционал $\rho(M, I, V)$ в энергии $E(V, \alpha, M)$ (см. формулу (25)).

4.1. Принцип минимальной длины описания

Задача поиска перекрытий заключается в том, что для некоторых векторов оптического потока не выполняется принцип постоянства цвета (23). Для решения этой задачи можно сначала вычислить оптический поток, игнорируя перекрытия, а затем исключить некоторые из них. При этом важно избежать тривиального решения, при котором все векторы будут исключены, т.е. избежать тривиального минимума функционала энергии (25) при M = 0. Возникает необходимость как-то ограничить снизу количество отбрасываемых векторов, а затем решить задачу отбрасывая векторы, пока не будет достигнут этот минимум. При этом ограничение на выброс векторов не обязано быть константой – оно может зависеть от карты видимости, т.е. могут существовать различные конфигурации, которые уже нельзя уменьшить, отбрасывая векторы.

В качестве такого критерия в данной работе предлагается использовать принцип минимальной длины описания. Будем кодировать цвета пикселов видеообъема, предполагая, что цвета вдоль корректных векторов потока меняются не сильно и подчиняются нормальному распределению (т.е. при отсутствии перекрытия изменение цвета может быть вызвано только шумом). При этом цепочка последовательных пикселов, соединенных не отброшенными векторами потока, кодируется единым распределением. Тогда выражение для длины такого описания может быть записано с помощью энтропии нормального распределения.

Если пытаться кодировать все пикселы с помощью одного нормального распределения, дисперсия будет высокой и следовательно суммарная энтропия тоже будет высокой. Исключение некоторых векторов потока разобьет видеообъем на цепочки пикселов, соответствующих одной движущейся точке физической сцены и имеющий постоянный цвет, поэтому суммарная энтропия будет уменьшаться. Однако, если продолжить исключать векторы потока и исключить их все, останется набор несвязных пикселов и кодировать каждый из них отдельным распределением будет накладно (суммарная эн-

тропия будет высокой). Из этих соображений получается, что минимальная длина описания достигается не на крайних случаях (все векторы используются/все векторы отброшены), а на какой-то промежуточной конфигурации, как и требуется в данной задаче.

Кроме того, полученное разбиение имеет физический смысл – каждая цепочка пикселов соответствует траектории движения некоторой точки сцены без перекрытий. Начало и конец траектории обычно соответствуют перекрытиям, хотя они также могут соответствовать изменениям освещенности или другим случаям явлениям, приводящим к изменению видимого цвета точки. Данная проблема возникает из-за невозможности достоверно определить наличие перекрытия в кадре, т.к. он содержит недостаточно информации о трехмерной сцене. Однако появление лишних перекрытий не так страшно: помимо временны́х связей алгоритм матирования использует пространственные, т.е. обеспечивает избыточное покрытие видеообъема межпиксельными связями. В случае же необнаружения перекрытия (т.е. ошибочного включения лишнего вектора) результирующая карта прозрачности будет содержать артефакт растекания, если одна из точек реальной сцены, связанных этим вектором, принадлежат объекту, а другая – фону.

Таким образом, данный подход обладает некоторой устойчивостью, позволяющей исключать сомнительные векторы, считая их перекрытиями.

4.2. Временны́е суперпикселы

Назовем цепочки пикселов в последовательных кадрах, связанных не отброшенными векторами оптического потока, временны́ми суперпикселами[4].

В используемой модели представления видеообъема как набора суперпикселов, каждый суперпиксел хранит

- средний цвет μ, дисперсию σ^2
- начало t_1, длительность τ.

Каждый пиксел суперпиксела хранит:

- цвет относительно μ
- координату x(t), y(t).

Тогда, используя принцип минимальной длины описания, можно потребовать от разбиения видеообъема на суперпикселы минимальность их общей энтропии

$$H = H_\mu + \tau \cdot H_p + \tau \cdot H_c, \tag{56}$$

где H_μ – число битов для хранения среднего цвета и дисперсии (константа),

H_p – число битов для кодирования координат одного пиксела (x(t), y(t)) – также константа,

H_c – число битов для хранения отклонения цвета пиксела от среднего цвета μ в предположении, что это отклонение подчиняется нормальному распределению с дисперсией σ^2. Энтропия нормального распределения определяется как

$$H_c(\sigma) = \frac{1}{2}\log\left(2\pi e\sigma^2\right). \tag{57}$$

При этом H_μ и H_p являются константами. В связи с тем, что энтропии разных распределений несоизмеримы, нет возможности вывести эти значения из битовой длины кодируемых значений цвета и координат. Поэтому

[4] По аналогии с пространственными суперпикселами, применяющимися в некоторых алгоритмах сегментации изображений [63], [71]

предлагается подобрать значения этих констант вручную и зафиксировать их (см. листинг 2). Таким образом, алгоритм будет вычислять только энтропию для цветов отдельных пикселов суперпиксела (57). Также добавляется константа σ_0, задающая масштаб для цветовой гауссианы:

$$H_c(\sigma) = \frac{1}{2}\ln\left(2\pi e\left(\frac{\sigma}{\sigma_0}\right)^2\right). \tag{58}$$

Суперпиксел хранится как связный список писклов. Псевдокод структуры данных, описывающий пиксел, приведен в листингах 1-2. Для вычисления энтропии пиксел должен хранить обновляемые значения μ, σ для суперпиксела, которому он принадленжит. Это достигается путем хранения суммы цветов, суммы квадратов цветовых компонент и длины суперпиксела. Эти значения легко обновлять при слиянии суперпикселов, и по ним всегда можно вычислить параметры μ, σ распределения цветов суперпиксела.

Минимизировать энтропию предлагается жадным алгоритмом. Он начинает с нулевой разметки (все векторы считаются перекрытиями) и постепенно добавляет векторы потока, пока суммарная энтропия увеличивается. На каждом шаге выбирается для включения вектор, уменьшающий энтропию на максимальное значение.

Жадный алгоритм использует двоичную кучу для хранения векторов потока и соответствующих им значений уменьшения энтропии. После выбора вектора, дающего наибольшее уменьшение, суперпикселы, соединяемые этим вектором объединяются (обновляются значения μ, σ) и пересчитываются значения уменьшения энтропии для векторов, инцидентных новому (объединенному) суперпикселу. Этот процесс проиллюстрирован на рис. 34. В листинге 3 приведен псевдокод структуры данных для хранения неориентированного вектора потока, соединяющего два пиксела (до их слияния). Как видно из рис. 34 (д), векторы потока не используются после слияния суперпикселов, поэтому обновлять значения μ, σ требуется не во всех пикселах суперпиксела, а только на его концах.

Листинг 1. Структура данных для описания пиксела внутри суперпиксела (псевдокод)

```
struct Pixel
{
	// Указатели на предыдущий/следующий пиксел суперпиксела
	Pixel *next, *prev;

	// Длина (в кадрах)
	int length;

	// Сумма цветовых компонент суперпиксела и их квадратов
	// для последующего вычисления среднего и дисперсии
	Color sum, sum2;

	Pixel ()
	{
		next = prev = NULL;
		length = 0;
		sum = sum2 = 0;
	}

	double GetCost() const
	{
		Color mu = sum / length;
		double sigma2 = (sum2 / length - mu^2).sum();

		return CalcCost(len, sigma2);
	}
};
```

Листинг 2. Константы и основные операции над структурой Pixel

```
// Энтропия Hμ
const double COST_0 = 2;

// Энтропия Hp
const double COST_1 = 0.07;

// Параметр σ0
const double SIGMA_SCALE = 0.008;

double CalcCost(int length, double sigma2)
{
	double frame_cost = COST_1 +
				log(2 * pi * e * sigma2 / SIGMA_SCALE^2) * 0.5;

	return COST_0 + length * max(0.0, frame_cost);
}

double GetMergedCost(const Pixel &p1, const Pixel &p2)
{
	int length = p1.length + p2.length;
	Color mu = (p1.sum + p2.sum) / length;
	double sigma2 = ((p1.sum2 + p2.sum2) / length - mu^2).sum();

	return CalcCost(length, sigma2);
}
```

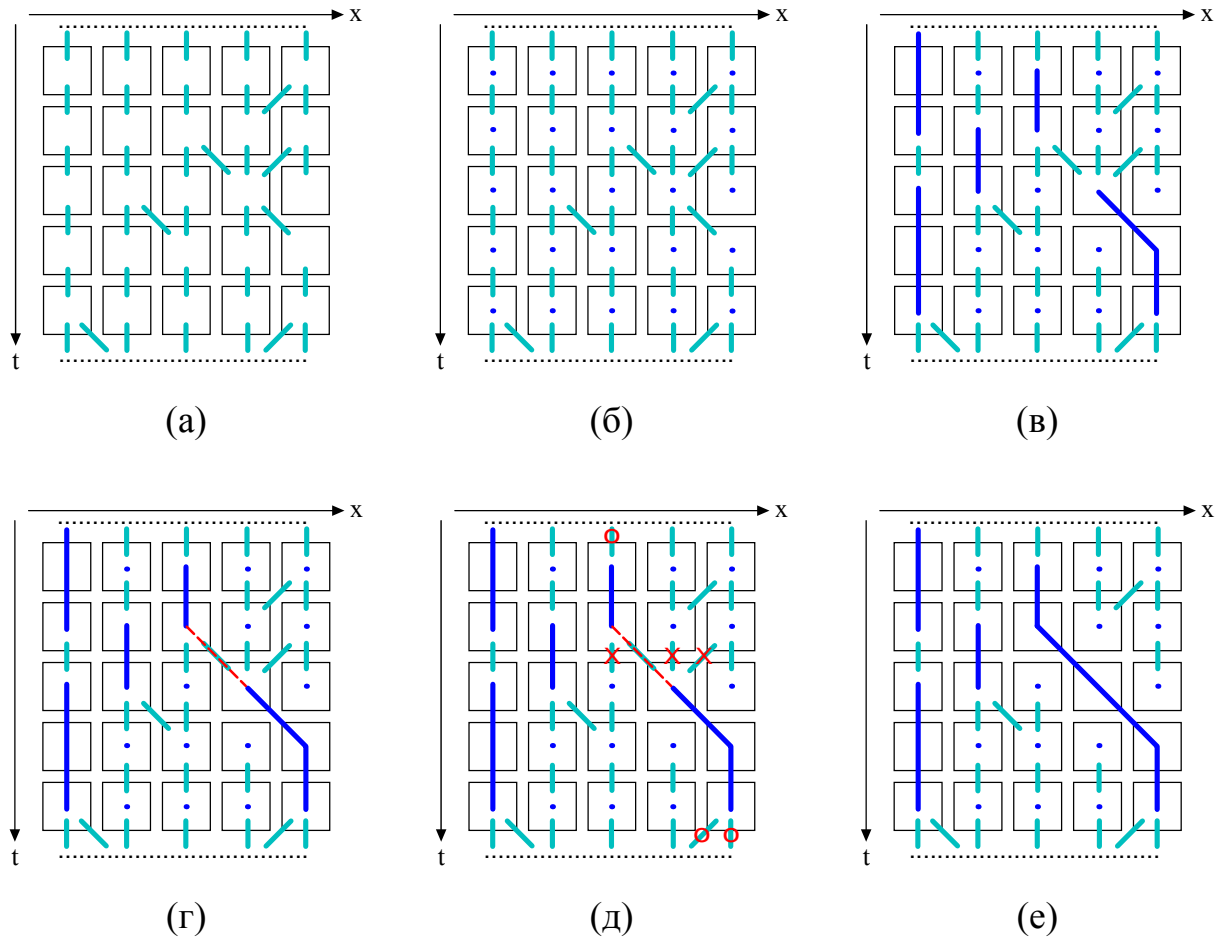


Рисунок 34. Процесс построения временны́х суперпикселов. **(а)** Исходное одномерное видео и векторы оптического потока (векторы прямого и обратного потока объединяются и рассматриваются как неориентированные ребра). **(б)** Инициализация суперпикселов: для каждого пиксела создается вырожденный суперпиксел (показаны синими точками). **(в)** Результат после нескольких шагов объединения суперпикселов. Далее будет подробно рассмотрен следующий шаг для данного состояния. **(г)** Выбран вектор оптического потока, дающий наибольшее уменьшение суммарной энтропии. Значения стоимости слияния посчитаны заранее для каждого ребра, поэтому такой выбор может быть сделан быстро. Два суперпиксела, соединяемые этим вектором, будут объединены (красный пунктир). **(д)** При объединении будут удалены ребра, помеченные красными крестиками, т.к. они ответвляются от полученного суперпиксела (что нарушает требование последовательности кадров, через которые проходит суперпиксел). В ребрах, помеченных красными кружочками, требуется пересчитать стоимость слияния (эти ребра находятся на концах объединенного суперпиксела). **(е)** Результат слияния суперпикселов.

Листинг 3. Структура данных для описания вектора потока (псевдокод)

```
struct FlowVector
{
	// Пара пикселов, соединяемых вектором потока
	// (должны принадлежать двум соседним кадрам)
	Pixel *p1, *p2;
	double merging_gain;

	FlowVector(Pixel *pixel1, Pixel *pixel2)
		: p1(pixel1), p2(pixel2)
	{
		// Проверить, что объединение пикселов допустимо
		assert(p1->next == NULL && p2->prev == NULL);

		double cost_old = p1->GetCost() + p2->GetCost();
		double cost_new = GetMergedCost(*p1, *p2);
		merging_gain = cost_old - cost_new;
	}

	// Оператор сравнения, позволяющий хранить векторы в бинарной куче:
	// векторы с большей выгодой слияния (уменьшением энтропии)
	// должны идти раньше в упорядоченном множестве
	bool operator<(const FlowVector &v) const
	{
		return merging_gain > v.merging_gain;
	}
};
```

Если предполагается, что суперпикселы будут достаточно длинными, можно помимо `next` и `prev` хранить указатели `head` и `tail` на концы суперпиксела, чтобы не находить их каждый раз последовательным обходом.

Псевдокод ининициализации, соответствующий рис. 34, приведен в листинге 4. Псевдокод алгоритма построения суперпикселов приведен в листинге 5. После работы алгоритма суперпикселы можно извлечь, обходя цепочки пикселов (по указателям `next` и `prev`).

На рис. 35 показан результат работы алгоритма на искусственном одномерном видео. На рис. 36 показан пример разбиения видеообъема на суперпикселы в виде среза полученного видеообъема одним кадром (показаны длины суперпикселов). Видно, что на границе движущегося объекта, т.е. в наиболее вероятных областях перекрытий, суперпикселы преимущественно вырожденные (состоят из одиночных пикселов), а в областях без перекрытий более длинные, что обеспечивает связность видеообъема. За счет этого финальный результат матирования содержит меньше артефактов.

Листинг 4. Псевдокод инициализации алгоритма построения суперпикселов

```
// На входе:
// video – видеообъем размера W×H×T
// flow_{fwd,bkw}_{x,y} – прямой и обратный оптический потоки,
// заданные своими компонентами x, y
Pixel *pixels = new Pixel[H, W, T];
for (int t = 0; t < T; t++)
      for (int y = 0; y < H; y++)
            for (int x = 0; x < W; x++)
            {
                  Pixel &p = pixels[y, x, t];
                  p.length = 1;
                  p.sum = video[y, x, t];
                  p.sum2 = p.sum^2;
            }

multiset<FlowVector> flow_vectors;
typedef multiset<FlowVector>::iterator FlowVectorsIterator;

multimap<Pixel *, FlowVectorsIterator> pixel_vectors;
typedef multimap<Pixel *, FlowVectorsIterator>::iterator PixelVectorsIterator;

// Прямой поток
for (int t = T - 2; t >= 0; t--)
      for (int y = 0; y < H; y++)
            for (int x = 0; x < W; x++)
            {
                  int x1 = x + flow_fwd_x[y, x, t];
                  if (x1 < 0 || x1 >= W)
                        continue;
                  int y1 = y + flow_fwd_y[y, x, t];
                  if (y1 < 0 || y1 >= H)
                        continue;
                  Pixel *p1 = &pixels[y, x, t], *p2 = &pixels[y1, x1, t + 1];
                  FlowVectorsIterator it =
                                    flow_vectors.insert(FlowVector(p1, p2));
                  pixel_vectors.insert(make_pair(p1, it));
                  pixel_vectors.insert(make_pair(p2, it));
            }

// Обратный поток
for (int t = 1; t < T; t++)
      for (int y = 0; y < H; y++)
            for (int x = 0; x < W; x++)
            {
                  int x1 = x + flow_fwd_x[y, x, t];
                  if (x1 < 0 || x1 >= W)
                        continue;
                  int y1 = y + flow_fwd_y[y, x, t];
                  if (y1 < 0 || y1 >= H)
                        continue;

                  // Избегаем дублирования векторов
                  if (flow_fwd_x[y1, x1, t - 1] == -flow_bkw_x[y, x, t]
                     && flow_fwd_y[y, x, t - 1] == -flow_bkw_y[y, x, t])
                        continue;

                  Pixel *p1 = &pixels[y, x, t], *p2 = &pixels[y1, x1, t - 1];
                  FlowVectorsIterator it =
                                    flow_vectors.insert(FlowVector(p1, p2));
                  pixel_vectors.insert(make_pair(p1, it));
                  pixel_vectors.insert(make_pair(p2, it));
            }
```

Листинг 5. Псевдокод алгоритма построения суперпикселов

```
// Повторяем пока не останется больше векторов или пока суммарная энтропия
// не перестанет уменьшаться
for (;;)
{
	if (flow_vectors.empty())
		break;

	// Выбираем вектор с наибольшим уменьшением энтропии
	FlowVector v = *flow_vectors.begin();

	// Останавливаемся, если уменьшить энтропию уже нельзя
	if (v.merging_gain < 0)
		break;

	// Удаляем векторы прямого потока, инцидентные p1
	foreach (PixelVectorsIterator it : pixel_vectors.equal_range(v.p1))
		flow_vectors.erase(it->second);

	// Удаляем векторы обратного потока, инцидентные p2
	foreach (PixelVectorsIterator it : pixel_vectors.equal_range(v.p2))
		if (it->second->p1 != v.p2)
			flow_vectors.erase(it->second);

	// Объединяем суперпикселы
	v.p1->next = v.p2;
	v.p2->prev = v.p1;

	// Находим концы цепочек пикселов
	Pixel *tail = v.p1, *head = v.p2;
	while (tail->prev)
		tail = tail->prev;
	while (head->next)
		head = head->next;

	// Обновляем параметры на концах цепочки
	tail->len = head->len = v.p1->len + v.p2->len;
	tail->sum = head->sum = v.p1->sum + v.p2->sum;
	tail->sum2 = head->sum2 = v.p1->sum2 + v.p2->sum2;

	// Обновляем векторы на концах цепочки
	foreach (PixelVectorsIterator it : pixel_vectors.equal_range(tail))
	{
		FlowVectorsIterator old_vec_it = it->second;
		FlowVector v1 = *old_vec_it;
		FlowVectorsIterator new_vec_it =
				flow_vectors.insert(FlowVector(v1.p1, v1.p2));
		// Обновляем обратный вектор
		foreach (PixelVectorsIterator it_rev :
					pixel_vectors.equal_range(v1.p1))
		{
			if (it_rev->second->p1 == v1.p1
					&& it_rev->second->p2 == v1.p2)
			{
				it_rev->second = new_vec_it;
				break;
			}
		}
		flow_vectors.erase(old_vec_it);
		it->second = new_vec_it;
	}
	// Аналогично для head
	foreach (PixelVectorsIterator it : pixel_vectors.equal_range(head))
	...
}
```

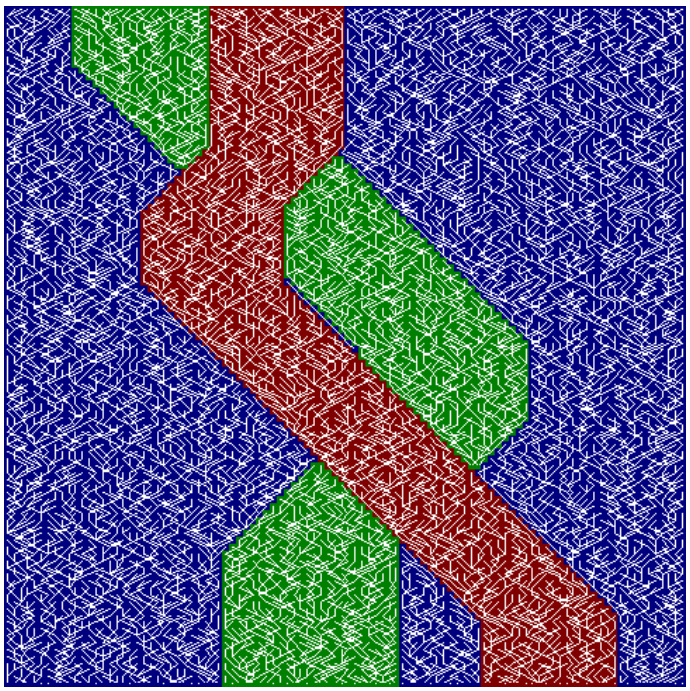
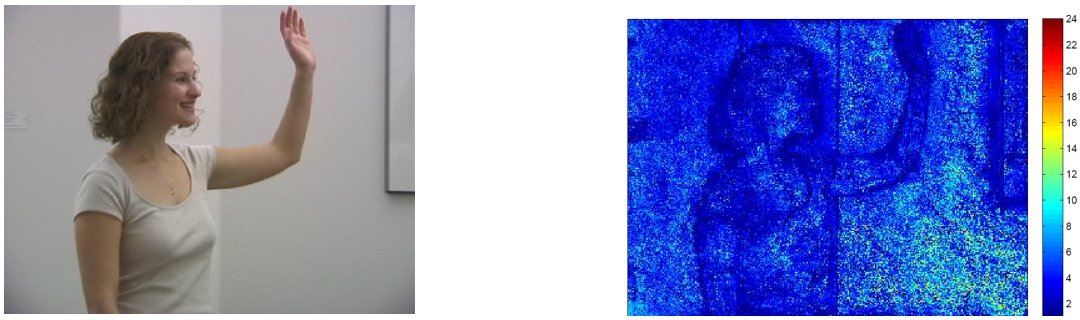


Рисунок 35. Результат работы алгоритма построения суперпикселов на зашумленных одномерных данных (по вертикали – ось времени). Оптический поток вычислялся эвристическим алгоритм без условия гладкости (поэтому многие векторы потока пересекаются). Как видно, суперпикселы не пересекают границы объектов, что гарантирует корректное матирование (отсутствие артефакта растекания прозрачности).



(а) (б)

Рисунок 36. Пример разбиения на суперпикселы: (а) кадр видеопоследовательности, (б) длины траекторий (в кадрах), проходящих через пикселы этого кадра.

Большинство траекторий имеют длину 2-4 кадра. Многие состоят из одного кадра (в областях высокой вероятности перекрытий). Однако это компенсируется небольшим количеством достаточно длинных траекторий, проходящих через всю видеопоследовательность. В основном, длинные суперпикселы получаются для областей, где $\alpha \in \{0, 1\}$, в то время как суперпикселы в областях с $0 < \alpha < 1$ получаются состоящими из одного пиксела. Такие суперпикселы не создают ограничений на канал прозрачности (см. следующий раздел), поэтому можно рассматривать такую сегментацию как ослабленную тернарную разметку для всего видеообъема.

4.3. Матирование на основе суперпикселов

Получив карту видимости M с помощью сегментации видеообъема на суперпикселы, мы имеем все необходимые данные для минимизации функционала (25) по α. При фиксировании прочих неизвестных, получится квадратичная форма (52) – (53), т.е. результирующий функционал будет штрафовать сильно различающиеся значения прозрачности в определенных парах пикселов. Эти пары определяются пространственными и временны́ми связями, которые образуют неориентированный граф. Вершинами графа являются все пикселы видеообъема.

Временны́е (межкадровые) связи являются неориентированными векторами потока (при M = 1, т.к. векторы для M = 0 на данном этапе уже отброшены), как показано на рис. 37. Пространственные связи образуются на основе лапласиана каждого кадра (13). Лапласиан [33] имеет фиксированную структуру (топологию), связывая каждый пиксел с 24 соседними в данном кадре (при минимальном размере окрестности – 3×3 пиксела – и порождаемом ею ядре 5×5 пикселов).

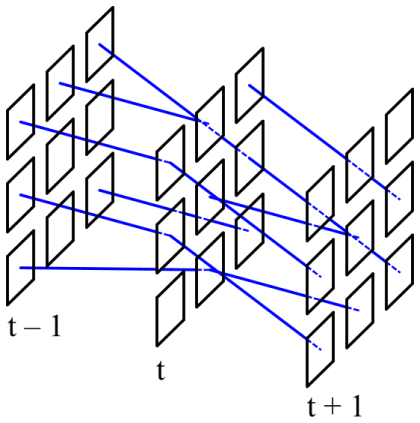


Рисунок 37. Межкадровые связи, определяемые векторами оптического потока (после отбрасывания перекрытий с помощью суперпикселов). Внутрикадровые связи (на рисунке не показаны) строятся на основе лапласиана каждого кадра.

Таким образом, видеообъем будет связным, если для каждой пары соседних кадров будет хотя бы один неотброшенный вектор оптического потока. На практике это достигается почти всегда, особенно при соблюдении требований к исходным данным, перечисленных во введении (отсутствие перепадов освещенности и т.д.). При нарушении связности решение по α будет неоднозначным (пикселы, не достигаемые из ключевых кадров, получат произвольные значения прозрачности).

Для ускорения алгоритма предлагается рассматривать межкадровые связи как жесткие. Приблизительное равенство прозрачности в связанных пикселах заменяется точным равенством:

$$\alpha_1(x,y) = \alpha_2(x+u(x,y), y+v(x,y)). \quad (59)$$

Тогда в каждом суперпикселе значения прозрачности всех пикселов будут одинаковыми и можно обозначить их новой переменной z_k:

$$z_k = \alpha_{i_{k,1}} = \alpha_{i_{k,2}} = \ldots = \alpha_{i_{k,n_k}}, \quad (60)$$

где k – индекс суперпиксела,

$i_{k,j}$ – индекс j-го пиксела в k-ом суперпикселе,

n_k – количество пикселов в k-ом суперпикселе.

Тогда вектор значений прозрачности пикселов можно выразить через вектор прозрачностей суперпикселов с помощью матрицы S как

$$\vec{\alpha} = S\vec{z}. \quad (61)$$

Матрица S является разреженной, состоит только из значений 0 и 1 и отображает суперпикселы на пикселы, т.е.

$$S_{k,i_{k,j}} = 1 \quad (62)$$

для всех k и j ($j = 1..n_k$). Остальные элементы S равны нулю.

В итоге нам требуется минимизировать функционал (25) по z. Число неизвестных z в 2-4 раза меньше, чем α (т.к. суперпикселы имеют среднюю

длину 2-4 кадра). Пространственно-временной аналог лапласиана (13) тогда можно записать в терминах суперпикселов [53]:

$$\begin{aligned} \hat{L} &= S^T \{L_t\} S, \\ J(z) &= z^T \hat{L} z, \end{aligned} \tag{63}$$

где $\{L_t\}$ означает блочно-диагональную матрицу всех двухмерных лапласианов для кадров t=1..T (полагая, что линейные индексы компонент вектора $\vec{\alpha}$ упорядочены по t).

4.3.1 Граничные условия

Граничные условия (ключевые кадры) применяются к суперпикселам. В некоторых случаях длинные траектории могут получить две или более различных метки из разных ключевых кадров. Эта проблема решается разбиением суперпиксела на несколько в точке максимального различия цвета вдоль него.

4.3.2 Сравнение с другими методами сегментации

Предложенные временны́е суперпикселы отличаются от других представлений. Они обеспечивают плотное разбиение, в отличие от разреженного разбиения [46], которое может не покрывать некоторые части видеообъема (или покрывать недостаточно детально). В отличие от двухмерных и трехмерных суперпикселов ([24], [68]) предложенные траектории покрывают ровно один пиксел в каждом кадре, поэтому они не объединяют пикселы пространственно, что могло бы привести к некорректной сегментации при похожих цветах объекта/фона. Поэтому решение о пространственной связности пикселов откладывается до этапа матирования.

Более подробное сравнение результатов будет приведено в разделе 4.7.

4.4. Фильтр разреженности

Часто суперпикселы получают конфликтующую информацию о метках от своих соседей из-за значительных перекрытий или изменений внешнего вида пикселов (изменения освещенности и т.д.). Квадратичная природа лапласиана в таких случаях приводит к присвоению промежуточных значений $\alpha \approx 0{,}5$ большим областям на изображениях. Стандартным подходом к предотвращению такой ситуации является использование робастных норм (например, L_1) или дополнительного слагаемого энергии, которое притягивает значение α к нулю или единице, например,

$$\alpha^{0,9} + (1-\alpha)^{0,9}, \tag{64}$$

и таким образом обеспечивает разреженность решения. Однако неквадратичность таких слагаемых сильно усложняет оптимизацию, обычно требуя несколько итераций алгоритма IRLS поверх текущего алгоритма.

Авторы статьи [26] заметили, что аналогичный результат можно получить значительно более быстрым методом, который просто фильтрует изображение сверткой определенным ядром. В статье [26] приведен способ построения ядра фильтра по матрице Лапласа (13).

В данной работе предлагается применять фильтр разреженности (guided filter) к значениям прозрачности суперпикселов, используя полный трехмерный лапласиан $\hat{L}$ (63). В результате применения фильтра большинство артефактов в канале прозрачности устраняется. Фильтрация осуществляется после каждой итерации алгоритма 1 (см. раздел 2.3) между шагами 2 и 3. Альфа-поток вычисляется с использованием отфильтрованного канала прозрачности.

При работе с объектами, обладающими естественной прозрачностью (например, дым), целесообразно отключить фильтрацию либо совсем, либо на последней итерации алгоритма, чтобы получить более точную прозрачность.

4.5. Программная реализация

Алгоритм был реализован в среде Matlab (рис. 38). Некоторые части алгоритма были реализованы на C++ в виде .mex-модулей. Для дискретной минимизации (при использовании траекторного потока) использовалась библиотека QPBO [45].

Архитектура предложенной системы матирования видео показана на рис. 39. Система состоит из 5 основных модулей. На рисунке также показан поток данных между ними и язык программной реализации каждого из модулей. Связка модулей «Вычисление оптического потока» + «Построение суперпикселов» и модуль «Вычисление траекторного потока» являются альтернативами, то есть используются либо первые два модуля, либо третий. Обе комбинации имеют на выходе данные одного и того же формата – оптический поток + маска перекрытий.

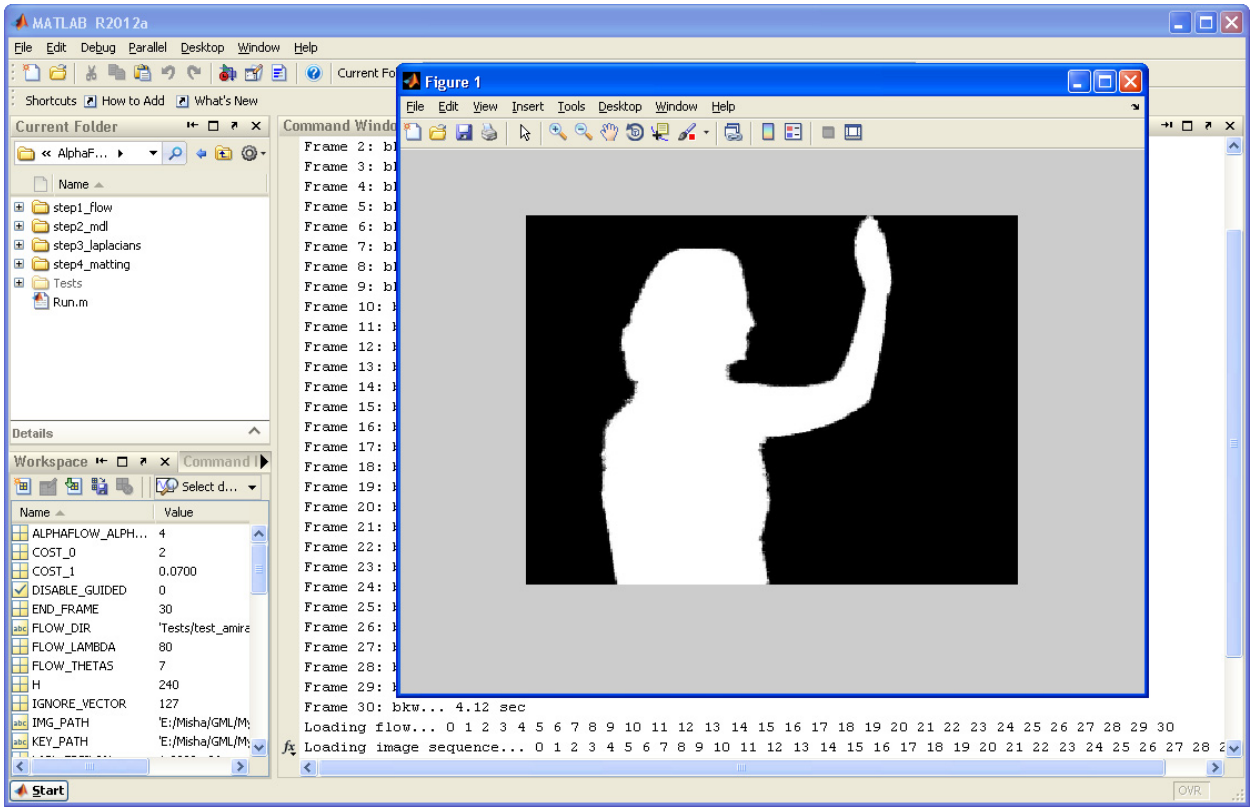


Рисунок 38. Снимок экрана Matlab-реализации алгоритма матирования видео.

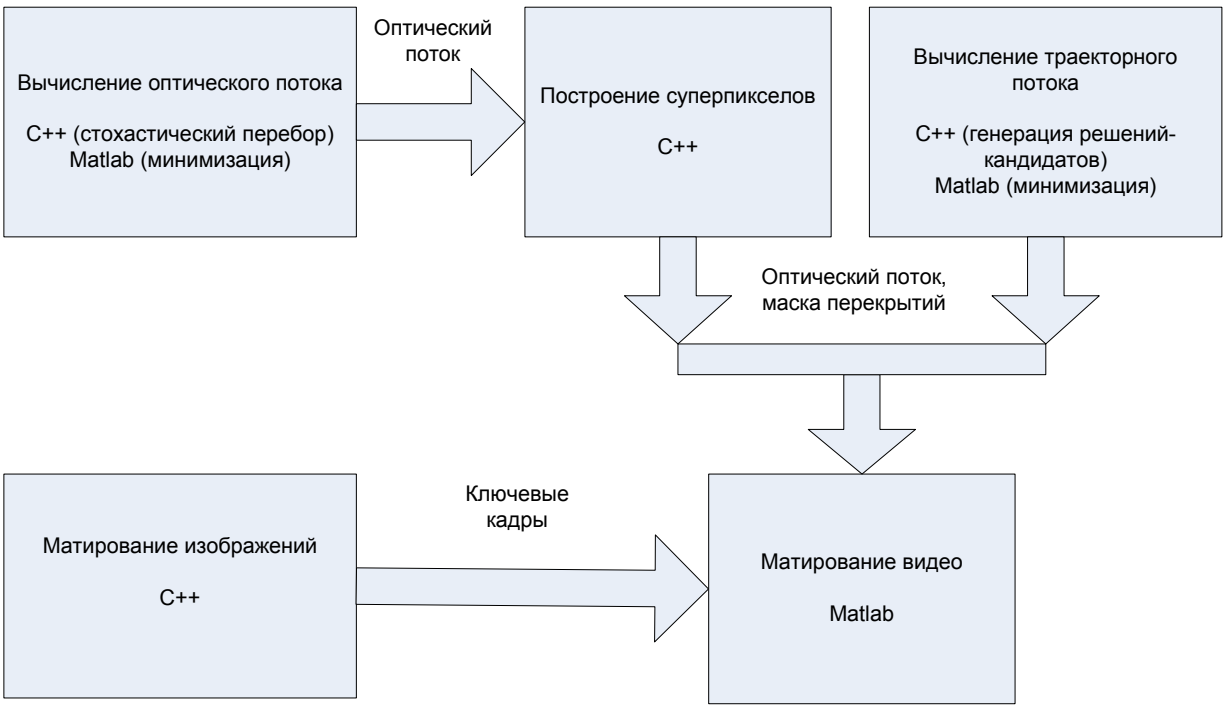


Рисунок 39. Архитектура системы и поток данных в ней.

4.6. Результаты

Примеры результатов работы алгоритма приведены на рис. 40.

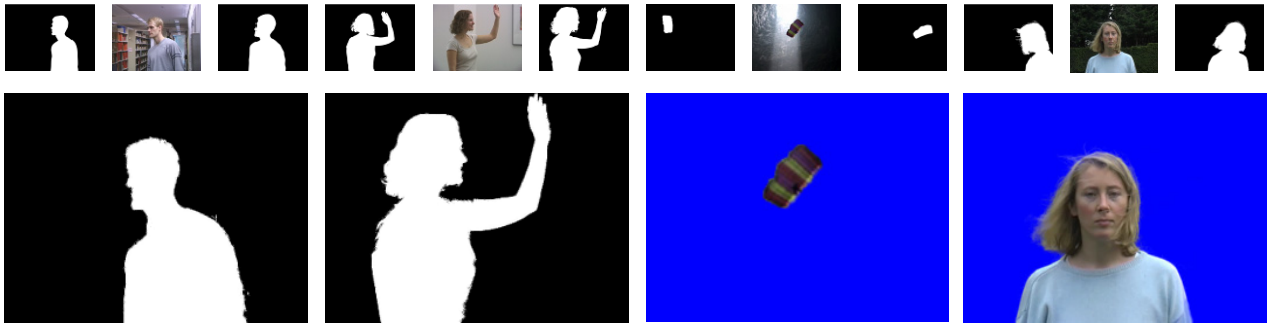


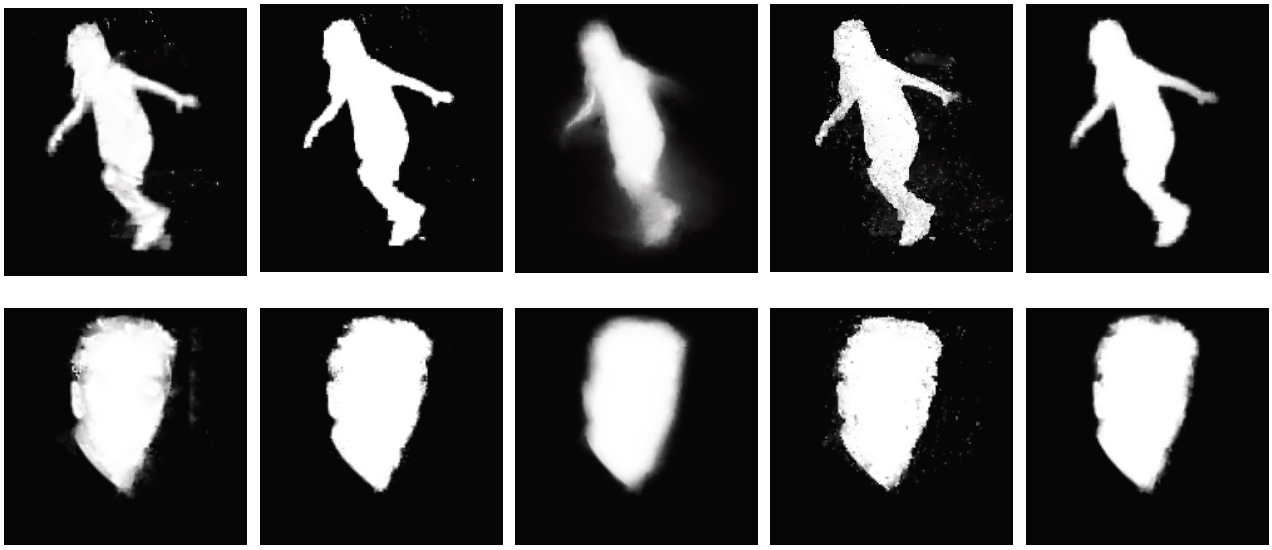
Рисунок 40. Результаты работы алгоритма. Вверху: ключевые кадры и промежуточный кадр. Внизу: результат для промежуточного кадра (показан либо альфа-канал, либо наложение на фон константного цвета).

За счет совместного использования предложенного метода поиска оптического потока, суперпикселов для поиска перекрытий, а также ключевых кадров, можно обрабатывать видеопоследовательности с межкадровым движением около 15-20 пикселов (при разрешении 640x480), что в 3-4 раза превосходит соответствующую оценку для метода Лукаса-Канаде [37], равную 5 пикселов на кадр.

4.6.1 Вклад отдельных слагаемых

Для обоснования предложенного алгоритма и необходимости всех его шагов были проанализированы результаты, получаемые при отключении отдельных шагов и слагаемых функционала энергии (25). Результаты эксперимента показаны на рис. 41. Простейший вариант (а) использует только оптический поток. Видны некорректные значения прозрачности внутри объекта, а также посторонние точки/пятна снаружи. Они могут быть частично скомпенсированы фильтром разреженности (б), но артефакты все равно сохраняются и частично искажается форма объекта (граница перестает быть гладкой). Однако простое применение альфа-потока (в) чрезмерно сглаживает результат, особенно при быстром движении. Это связано с перекрытиями, некорректная обработка которых приводит к «растеканию» прозрачности. Также была рас-

смотрена возможность вычислять альфа-поток по бинаризованной (по порогу 0,5) маске прозрачности (г). В этом случае результат получается близкий к корректному в плане целостности, но сильно зашумленный. Этот эксперимент мотивирует вычислять поток именно для карты прозрачности, а не простой маски объекта, как делается в алгоритмах жесткой сегментации [35], [68]. Корректный результат получается только при использовании всех шагов алгоритма (д).



(а) (б) (в) (г) (д)

Рисунок 41. Результат работы алгоритма при отлючении различных слагаемых функционала (25) и шагов алгоритма на примере двух видеопоследовательностей.

(а) без альфа-потока и фильтра разреженности

(б) без альфа-потока

(в) без обработки перекрытий

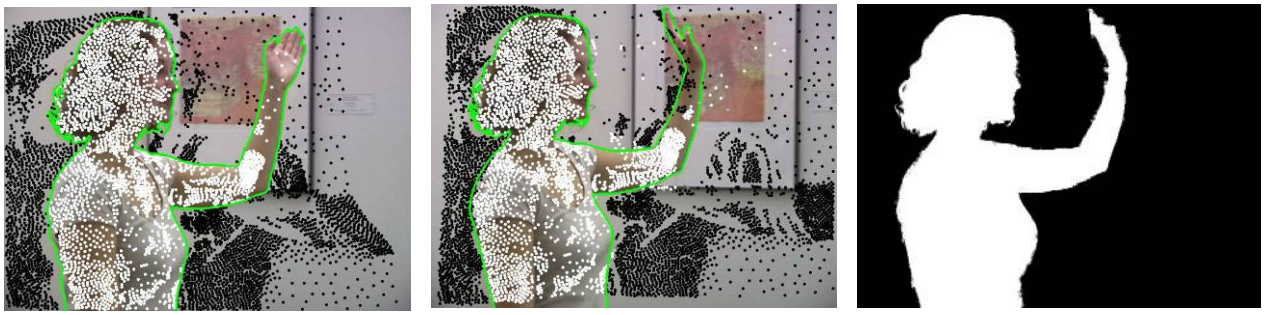
(г) использование бинарной маски при вычислении альфа-потока

(д) финальный результат

4.7. Сравнение

Сравнение проводилось на наборе из 25 тестовых роликов. Сравнивались различные подходы к задачам сегментации и матирования видео, такие как

- Video SnapCut [10]
- Particle Video [46]
- Motion Coherent Tracking with Multi-label MRF optimization [62]
- Video Object Cut and Paste [35]
- и т.д.



(а) (б) (в)

Рисунок 42. Сравнение с алгоритмом Particle Video [46]. Т.к. данный алгоритм не является алгоритмом сегментации (но является алгоритмом отслеживания точек с помощью оптического потока, поэтому сравнение с ним важно), был использован следующий подход: **(а)** в ключевом кадре траектории (точки) помечаются значениями прозрачности (зелёным показан эталонный контур), **(б)** значения прозрачности переносятся на другой кадр и анализируются (сравниваются с эталонным контуром) – видны ошибки I-го и II-го рода (если рассматривать задачу как поиск точек объекта), искажена структура объекта. **(в)** Результат предложенного алгоритма – ошибок значительно меньше, структура объекта сохранена.

4.7.1 Численное сравнение

Для объективного (численного) сравнения алгоритмов требуется набор эталонных данных, т.е. видеопоследовательностей, для которых известен канал прозрачности, и метрика сравнения.

Проблема получения эталонных данных не решена для видео. Для изображений обычно применяется фотографирование объекта на нескольких сменных фонах [47], обычно на фоне ЖК-дисплея [43]. Для видео в общем случае снять каждый кадр на нескольких фонах проблематично. Поэтому обычно используется ручная разметка данных ротоскопированием. В данной работе использовались размеченные данные из тестовой базы [62] и несколько роликов, размеченных автором.

4.7.2 Метрика сравнения

Простейшей метрикой сравнения является среднеквадратичное отклонение от эталонного результата:

$$\frac{\sum_{x,y,t}\left(\alpha_{x,y,t}-\alpha^{*}_{x,y,t}\right)^2}{W\cdot H\cdot T}, \tag{65}$$

где α^{*} – эталонный канал прозрачности, W, H, T – размер видеообъема. Также можно ввести среднеабсолютное отклонение, устойчивое к шуму:

$$\frac{\sum_{x,y,t}\left|\alpha_{x,y,t}-\alpha^{*}_{x,y,t}\right|}{W\cdot H\cdot T}, \tag{66}$$

однако канал прозрачности α^{*}, в отличие от изображения, обычно не подвержен зашумлению и выбросам (и даже наоборот желательно штрафовать шум и выбросы в α), поэтому у метрики (66) нет особых преимуществ перед (65).

Тем не менее, метрики, сравнивающие α и α^{*} подвержены некоторой проблеме, особенно если эталонная разметка α^{*} сделана вручную. Дело в том, что граница объекта может быть сдвинута внутрь/наружу на несколько пикселов, что практически не влияет на восприятие результата, однако срав-

нение методов по такой метрике приводит к тому, что поощряются не более точные методы, а такие, в которых граница сдвинута на ту же величину. Эта проблема усиливается, если в сравнении участвуют методы, не использующие ключевые кадры (которые можно взять напрямую из α^*), а использующие, например, тернарную разметку.

В связи с этим в данной работе предлагается метрика, инвариантная к подобным искажениям канала прозрачности. Метрика оценивает плавность движения результирующей маски (отсутствие дрожания). Она допускает некоторые неточности в маске при условии, что они сохраняются от кадра к кадру (не мерцают).

Вычислим площадь объекта в каждом кадре, определив ее как сумму значений прозрачности:

$$A(t) = \sum_{x,y} \alpha(x, y, t) \tag{67}$$

и аналогично для эталонной прозрачности:

$$A^*(t) = \sum_{x,y} \alpha^*(x, y, t). \tag{68}$$

Нормализуем площадь A(t) с помощью линейного преобразования и прибавления константы:

$$\widetilde{A}(t) = kA(t) + b\,. \tag{69}$$

Сделаем параметры k и b свободными и при вычислении метрики будем минимизировать ее по этим параметрам. Параметры не зависят от t, т.е. минимизация проводится для всего видео:

$$\sum_t \left(\widetilde{A}(t) - A^*(t)\right)^2 \to \min\,. \tag{70}$$

Вычислим оценку межкадрового дрожания как разницу производных нормализованной площади результата и эталонной площади и усредним по всем кадрам

$$\frac{1}{T}\int_0^T \left|\frac{d\widetilde{A}}{dt} - \frac{dA^*}{dt}\right| dt\,, \tag{71}$$

где T – длительность видео. В дискретной форме это запишется как

$$\frac{1}{T-1}\sum_{t=1}^{T-1}\left|\left(\tilde{A}(t+1)-\tilde{A}(t)\right)-\left(A^{*}(t+1)-A^{*}(t)\right)\right|. \qquad (72)$$

При таком подходе можно сравнивать каналы прозрачности разных разрешений (изменится только значение k), а также фрагментов канала прозрачности, полученных обрезкой области фона (при такой обрезке площадь объекта вообще не изменится). Последнее актуально, если алгоритм матирования применялся к части кадра для экономии времени (при условии, что область интереса была выбрана корректно, т.е. она не отсекла часть объекта).

Результаты сравнения с использованием предложенной метрики, а также суммарной квадратичной ошибки приведены в таблице 3. Ключевые кадры брались из эталонной разметки. В методах, где это невозможно, создавалась как можно более похожая разметка. Как видно, предложенный алгоритм имеет наименьшую результирующую ошибку по обоим критериям. Кроме того, упорядочение алгоритмов по каждой из метрик почти совпадает, кроме последних двух алгоритмов. Метод матирования изображений [33], примененный к видеообъему дает бóльшую ошибку, но меньше дрожания, в то время как алгоритм бинарной сегментации [35] дает противоположный эффект. Это объяснимо, т.к. [33] дает очень гладкий, но обычно некорректный результат, а [35] – более точный результат, но с ошибками мерцания (см. раздел 2.1).

Таблица 3. Численное сравнение алгоритмов

Алгоритм	**Оценка дрожания**	**Квадратичная ошибка**
Предложенный алгоритм	232,9	1418,7
Алгоритм отслеживания [62]	481,5	2068,1
Video SnapCut [10]	614,9	8201,1
Объемный алгоритм CFS [33]	1051,7	25612,7
Video Object Cut&Paste [35]	1719,4	14060,3

4.7.3 Устойчивость к ошибкам в ключевых кадрах

До сих пор предполагалось, что ключевые кадры берутся из эталонных данных. На практике такие данные недоступны, а ключевые кадры создаются вручную, поэтому от алгоритма требуется устойчивость к ошибкам в них. Исследовать устойчивость можно на синтетических данных, внося ошибки искусственно.

Результаты такого эксперимента приведены на рис. 43. Исходная последовательность – трехмерная вращающаяся и сдвигающаяся ваза поверх сдвигающегося двухмерного фона (на основе изображений из тестовой базы Беркли [38]). Разрешение 160x120, длительность 16 кадров. Искаженные маски ключевых кадров были получены морфологическим расширением/сжатием с радиусом 2 пиксела. Первые три результата –альфа-поток с использованием одного ключевого кадра. Затем – результат алгоритма SnapCut [10]. Последние три результата – альфа-поток с использованием двух ключевых кадров.

Кадр 1 является ключевым во всех тестах. Кадр 16 является ключевым в последних трех. Алгоритм SnapCut может работать только с одним ключевым кадром.

Предложенный алгоритм демонстрирует бóльшую устойчивость к неточностям в ключевых кадрах во всех случаях, хотя наилучший результат достигается при использовании двух ключевых кадров.

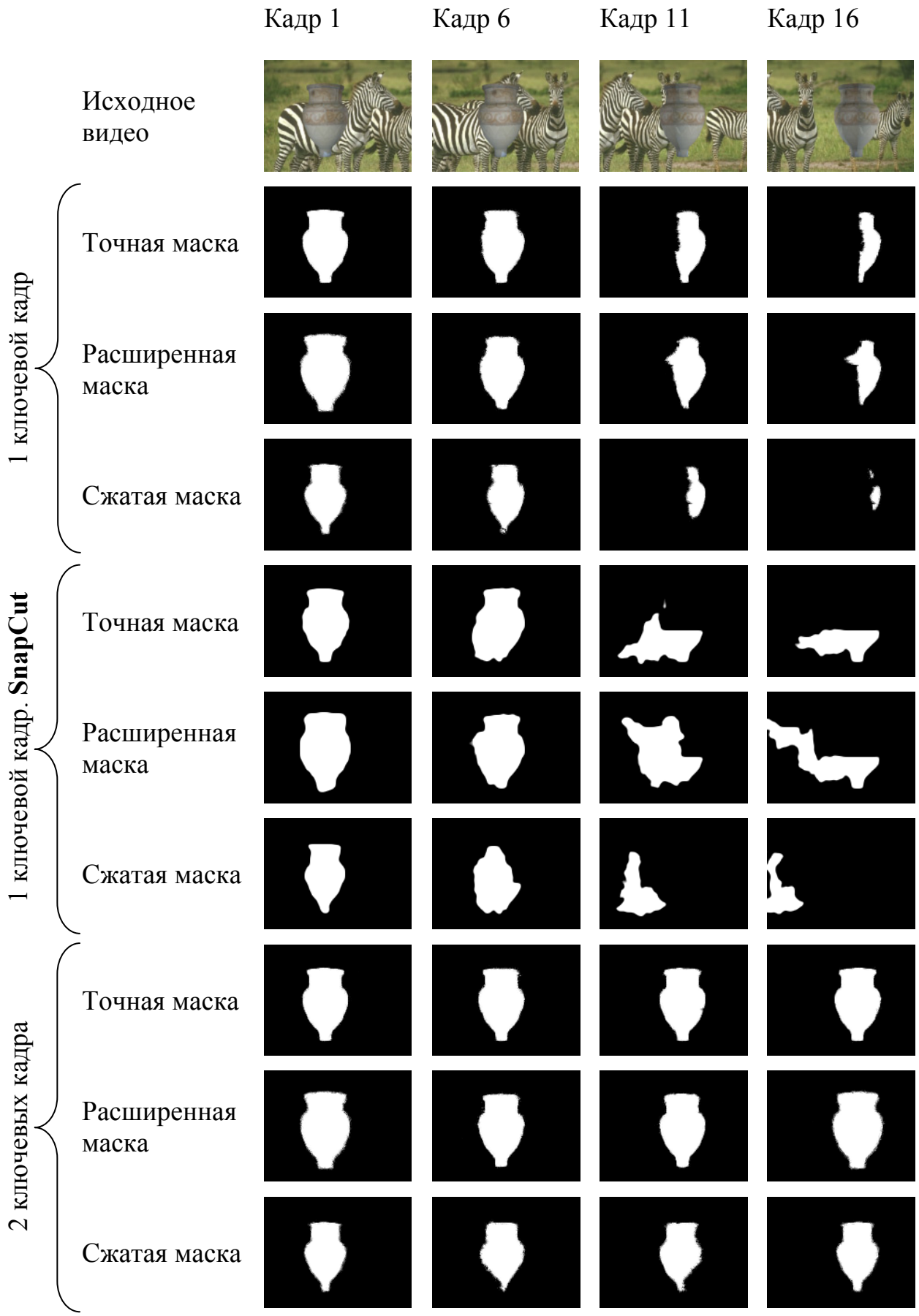


Рисунок 43. Искусственный пример (см. текст).

ЗАКЛЮЧЕНИЕ

В данной работе рассматривалась задача матирования видео – выделения объектов переднего плана из видеопоследовательности с построением маски (канала прозрачности). В работе предложен метод совместного нахождения канала прозрачности и оптического потока в нем с использованием ключевых кадров.

В первой главе рассмотрена задача матирования изображений и предложен алгоритм матирования на основе байесовского подхода. Во второй главе сформулирована задача матирования видео по ключевым кадрам и предложен алгоритм ее решения, основывающийся на двух других алгоритмах: вычислении оптического потока и матирования видеообъема. В третьей главе рассмотрена задача вычислении оптического потока. Предложены два алгоритма – более быстрый двухкадровый и более точный траекторный. В четвертой главе описан алгоритм матирования видеообъема. Предложено разбиение видеообъема на временны́е суперпикселы для обработки перекрытий.

Результаты алгоритма продемонстрированы на тестовой выборке видеопоследовательностей. Для части из них была создана эталонная разметка и проведено численное сравнение, показавшее превосходство предложенного алгоритма над существующими методами при аналогичных входных данных (ключевых кадрах).

СПИСОК РИСУНКОВ

ЛИТЕРАТУРА

1. Agarwala A., Hertzmann A., Salesin D., Seitz S. Keyframe-based tracking for rotoscoping and animation // In proc. of SIGGRAPH. 2004. P. 584–591
2. Agrawal A., Raskar R., Chellappa R. What is the range of surface reconstructions from a gradient field? // In proc. of ECCV. 2006. P. 578–591
3. Akhter I., Sheikh Y., Khan S., Kanade T. Nonrigid Structure from Motion in Trajectory Space // In proc. of NIPS. 2008.
4. Akhter I., Sheikh Y., Khan S., Kanade T. Trajectory Space: A Dual Representation for Nonrigid Structure from Motion // In proc. of PAMI. 2011. V 33, N 7. P. 1142–1456
5. Alvarez L., Esclarin J., Lefebure M., Sanchez J., A PDE model for computing the optical flow // In proc. of XVI Congreso de Ecuaciones Diferenciales y Aplicaciones. 1999. P. 1349–1356
6. Apostoloff N., Fitzgibbon A. Automatic video segmentation using spatiotemporal T-junctions // In proc. of BMVC. 2006. P. 1089–1098
7. High Quality Chroma Key: CS 294 Project Final Report / Ashihkmin M.
8. Bai X., Sapiro G. Geodesic matting: A framework for fast interactive image and video segmentation and matting // International Journal of Computer Vision. 2009. V 82, N 2. P. 113–132
9. Bai X., Wang J., Sapiro G. Dynamic Color Flow: A Motion-Adaptive Color Model for Object Segmentation in Video // In proc. of ECCV. 2010. V 5. P. 617–630
10. Bai X., Wang J., Simons D., Sapiro G. Video SnapCut: robust video object cutout using localized classifiers // In proc. of SIGGRAPH. 2009. P. 1–11
11. Bai X., Wang J., Simons D. Towards temporally-coherent video matting // In proc. of Mirage. 2011. P. 63–74
12. Baker S., Scharstein D., Lewis J., Roth S., Black M., Szeliski R.. A Database and Evaluation Methodology for Optical Flow // International Journal of Computer Vision. 2011. V 92, N 1. P. 1–31

13. Barnes C., Shechtman E., Finkelstein A., Goldman D. PatchMatch: A randomized correspondence algorithm for structural image editing // Communications of the ACM. 2011. V 54, N 11. P. 103–110
14. Boykov Y., Kolmogorov V. Computing geodesics and minimal surfaces via graph cuts // In proc. of ICCV. 2003. P. 26–33
15. Chen J., Paris S., Wang J., Cohen M., Durand F. The video mesh: A data structure for image-based video editing // In proc. of ICCP. 2011. P. 1–8
16. Choi I., Lee M., Tai Y. Video Matting Using Multi-frame Nonlocal Matting Laplacian // In proc. of ECCV. 2012. V 4. P. 540–553
17. Chuang Y., Curless B., Salesin D., Szeliski R. A Bayesian Approach to Digital Matting // In proc. of CVPR. 2001. P. 264–271
18. Chuang Y., Agarwala A., Curless B., Salesin D., Szeliski R. Video matting of complex scenes // In proc. of SIGGRAPH. 2002. P. 243–248
19. Пат. N 5,343,252 США / Dadourian A. Method and Apparatus for Compositing Video Images. 30 августа 1994
20. Dijkstra E. A note on two problems in connexion with graphs // Numerische Mathematik. 1959. N 1. P. 269–271
21. Ding Z., Chen H., Guan Y., Chen W., Peng Q. GPU accelerated interactive space-time video matting // Computer Graphics International. 2010
22. Пат. N 1242674 США / Fleischer M. Method of Producing Moving-Picture Cartoons. 9 октября 1917
23. Garg R., Pizarro L., Rueckert D., de Agapito L. Dense Multi-frame Optic Flow for Non-rigid Objects Using Subspace Constraints // In proc. of ACCV. 2010. P. 460–473
24. Grundmann M., Kwatra V., Han M., Essa I. Efficient hierarchical graph based video segmentation // In proc. of CVPR. 2010. P. 2141–2148
25. Hanson B. Matte Painting: Art in Film Special Effects [PPT] (http://www-graphics.stanford.edu/courses/cs99d-00/projects/BrookeHanson-specialfx.ppt)
26. He K., Sun J., Tang X. Guided image filtering // In proc. of ECCV. 2010. P. 1–14

27. Horn B., Schunck B. Determining optical flow // Artificial Intelligence. 1981. V 17. P. 185–203
28. Irani M. Multi-Frame Optical Flow Estimation Using Subspace Constraints // International Journal of Computer Vision. 2002. V 48, N 3. P. 173–194
29. Joshi N., Matusik W., Avidan S. Natural Video Matting using Camera Arrays // ACM Transactions on Graphics. 2006. V 25 N 3. P. 779–786
30. Kang S., Szeliski R., Extracting view-dependent depth maps from a collection of images // International Journal of Computer Vision. V 58, N 2. 2004. P. 139–163
31. Lee S., Yoon J., Lee I. Temporally coherent video matting // Graphical Models. 2010. N 72. P. 25–33
32. Lempitsky V., Roth S., Rother C.. FusionFlow: discrete-continuous optimization for optical flow estimation // In proc. of CVPR. 2008. P. 1–8
33. Levin A., Lischinski D., Weiss Y. A Closed Form Solution to Natural Image Matting // In proc. of CVPR. 2006. P. 61–68
34. Lezama J., Alahari K., Sivic J., Laptev I. Track to the future: Spatio-temporal video segmentation with long-range motion cues // In proc. of CVPR. 2011. P. 3369–3376
35. Li Y., Sun J., Shum H. Video object cut and paste // In proc. of SIGGRAPH. 2005. P. 595–600
36. Beyond pixels: exploring new representations and applications for motion analysis: Doctoral Thesis. Massachusetts Institute of Technology. 2009 / Liu C.
37. Lucas B., Kanade T. An iterative image registration technique with an application to stereo vision // Proceedings of Imaging Understanding Workshop. 1981. P. 121–130
38. Martin D., Fowlkes C., Tal D., Malik J.. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics // In proc. of ICCV. 2001. P. 416–423

39. Пат. N 7,609,327 США / Matusik W. Polarization difference matting using a screen configured to reflect polarized light. 27 октября 2009
40. McGuire M., Matusik W., Pfister H., Hughes J., Durand F. Defocus video matting // ACM Transactions on Graphics. 2005. V 24, N 3. P. 567–676
41. Rhemann C., Hosni A., Bleyer M., Rother C., Gelautz M. Fast Cost-Volume Filtering for Visual Correspondence and Beyond // In proc. of CVPR. 2011. P. 3017–3024
42. Rhemann C., Rother C., Kohli P., Gelautz M. A Spatially Varying PSF-based Prior for Alpha Matting // In proc. of CVPR. 2010. P. 2149–2156
43. Rhemann C., Rother C., Wang J., Gelautz M., Kohli P., Rott P. A Perceptually Motivated Online Benchmark for Image Matting // In proc. of CVPR. 2009. P. 1826–1833
44. Rother C., Kolmogorov V., Blake A. GrabCut – Interactive Foreground Extraction using Iterated Graph Cuts // ACM Transactions on Graphics (SIGGRAPH). 2004. V 23. P. 309–314
45. Rother C., Kolmogorov V., Lempitsky V., Szummer M. Optimizing binary MRFs via extended roof duality // In proc. of CVPR. 2007. P. 1–8
46. Sand P., Teller S. Particle video: Long-range motion estimation using point trajectories // International Journal of Computer Vision. 2008. P. 72–91
47. Smith A., Blinn J. Blue Screen Matting // In proc. of SIGGRAPH. 1996. P. 259–268
48. Tang Z., Miao Z., Wan Y., Zhang D. Video matting via opacity propagation // The Visual Computer. 2011. P. 1–15
49. Sarim M., Hilton A., Guillemaut J., Kim H. Non-parametric natural image matting // In proc. of ICIP. 2009. P. 3213–3216
50. Sindeyev M., Konushin V., Vezhnevets V. Improvements of Bayesian Matting // In proc. of Graphicon. 2007. P. 88–95
51. Sindeyev M., Konushin V. A Novel Interactive Image Matting Framework // In proc. of Graphicon. 2008. P. 41–44

52. Sindeyev M.., Konushin V.. A Novel Approach to Video Matting using Optical Flow // In proc. of Graphicon. 2009. P. 340–343
53. Sindeev M., Rother C., Konushin A. Alpha-Flow for Video Matting // In proc. of ACCV. 2012
54. Steinbrücker F., Pock T., Cremers D. Large Displacement Optical Flow Computation without Warping // In proc. of ICCV. 2009. P. 1609–1614
55. Sun D., Roth S., Black M. Secrets of optical flow estimation and their principles // In proc. of CVPR. 2010. P. 2432–2439
56. Sun D., Sudderth E., Black M. Layered Segmentation and Optical Flow Estimation Over Time // In proc. of CVPR. 2012. P. 1768–1775
57. Sun J., Li Y., Kang S., Shum H. Flash Matting // In proc. of SIGGRAPH. 2006. V 25, N 3. P. 599–604
58. Sun J., Sun J., Kang S., Xu Z., Tang X., Shum H. Flash Cut: Foreground Extraction with Flash/No-Flash Image Pairs // In proc. of CVPR. 2007.
59. Пат. N 7,724,952 США / Shum H., Sun J., S. Kang S., Li Y. Object matting using flash and no-flash images. 25 мая 2010
60. Tao M., Bai J., Kohli P., Paris S. SimpleFlow: A Non-iterative, Sublinear Optical Flow Algorithm // Computer Graphics Forum (Eurographics), V 2 N 31. 2012.
61. Пат. N 6,061,462 США / Tostevin N., Moran M., Gardner J., Marrero N., Cook R. Digital Cartoon and Animation Process. 9 мая 2000
62. Tsai D., Flagg M., Rehg J. Motion coherent tracking with multi-label MRF optimization // In proc. of BMVC. 2010. V 56. P. 1–11
63. Veksler O., Boykov Y., Mehrani P. Superpixels and Supervoxels in an Energy Optimization Framework // In proc. of ECCV. 2010. P. 211–224
64. Vezhnevets V., Konouchine V. Grow-Cut – Interactive Multi-Label N-D Image Segmentation by Cellular Automata // In proc. of Graphicon. 2005. P. 150–156
65. Пат. N 3,595,987 США / Vlahos P. Electronic Composite Photography. 27 июля 1971

66. Пат. N 4,100,569 США / Vlahos P. Comprehensive Electronic Compositing System. 11 июля 1978
67. Volz S., Bruhn A., Valgaerts L., Zimmer H. Modeling temporal coherence for optical flow // In proc. of ICCV. 2011. P. 1116–1123
68. Wang J., Bhat P., Colburn R., Agrawala M., Cohen M. Interactive video cut-out // In proc. of SIGGRAPH. 2005. P. 585–594
69. Wang J., Cohen M. Optimized Color Sampling for Robust Matting // In proc. of CVPR. 2007. P. 1–8
70. Xiao J., Cheng H., Sawhney H., Rao C., Isnardi M. Bilateral filtering-based optical flow estimation with occlusion detection // In proc. of ECCV. 2006. P. 211–224
71. Ren X., Malik J. Learning a classification model for segmentation // In proc. of ICCV. 2003. V 1. P. 10–17
72. Wang O., Finger J., Yang Q., Davis J., Yang R. Automatic Natural Video Matting with Depth // In proc. of PCCGA. P. 469–472
73. A unified framework for large- and small-displacement optical flow estimation: Technical report. The Chinese University of Hong Kong. 2010 / Xu L., Jia J., Matsushita Y.
74. Xu L., Jia J., Matsushita Y. Motion Detail Preserving Optical Flow Estimation // IEEE Transactions on Pattern Analysis and Machine Intelligence. V 34, N 9. 2012. P. 1744–1757
75. Zaheer A., Akhter I., Baig M., Marzban S., Khan S. Multiview structure from motion in trajectory space // In proc. of ICCV. 2011. P. 2447–2453
76. Синдеев М., Конушин В. Матирование видео на основе оптического потока // Сборник тезисов XVI Международной научной конференции студентов, аспирантов и молодых учёных «Ломоносов-2009», секция «Вычислительная математика и кибернетика». 2009. С. 75.
77. Синдеев М., Конушин А., Ротер К. Многокадровый оптический поток на основе траекторий // Труды конференции Графикон. 2012. С. 288–291

78. Синдеев М., Конушин В. Интерактивное байесовское матирование изображений // Программные продукты и системы. 2012. N 4. С. 167–171
79. Keylight – Advanced Blue Screen and Green Screen Keying [HTML] (http://www.thefoundry.co.uk/products/keylight/)
80. Planar tracking tools for visual effects and post-production – Imagineer Systems [HTML] (http://www.imagineersystems.com/)
81. Primatte [HTML] (http://www.primatte.com/)
82. Ultimatte – Hardware for processing bluescreen and greenscreen compositing [HTML] (http://ultimatte.com/)
83. The Art of Roto: 2011 [HTML] (http://www.fxguide.com/featured/the-art-of-roto-2011/)
84. Rotoscoping Techniques in NUKE 5.2 Tutorial [HTML] (http://www.digitaltutors.com/11/training.php?pid=270)