САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра системного программирования

СОЗДАНИЕ СИСТЕМЫ ОБРАБОТКИ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ АЛГОРИТМОВ ПОИСКА МИНИМАЛЬНОГО РАЗРЕЗА ГРАФА

Дипломная работа студентки 545 группы

Кочановой Татьяны Александровны

Научный руководитель	..... / подпись /	д.ф.-м.н., проф. Граничин О. Н.
Рецензент	..... / подпись /	Морозков М.А.
"Допустить к защите" заведующий кафедрой	..... / подпись /	д.ф.-м.н., проф. Терехов А.Н.

Санкт-Петербург

2012

SAINT-PETERSBURG STATE UNIVERSITY
Mathematics & Mechanics Faculty

Software Engineering Chair

IMAGE PROCESSING TOOL DEVELOPMENT BASED ON GRAPH-CUT ALGORITHMS

Graduate paper

Kochanova Tatiana

Supervisor	.....	Dr. of Phys. and Math. Sci., professor Granichin O.N.
Reviewer	.....	Morozkov M. A.
"Approved by" Head of department	.....	Dr. of Phys. and Math. Sci., professor Terekhov A.N.

Saint-Petersburg
2012

Содержание

1. Введение

Широкий спектр проблем анализа изображений и распознавания образов сводится к задаче расстановки меток для пикселей изображения, для оптимального решения которой используются алгоритмы, минимизирующие некоторый функционал энергии, являющейся качественной характеристикой выбранной расстановки меток. Подобные алгоритмы применяются в самых различных областях, таких как стереозрение, восстановление или сегментация изображений и т. п.

Одна из главных целей дипломной работы — исследовать задачу минимизации функционала энергии для марковских случайных полей, сделать обзор существующих методов ее решения, заостряя внимание на их сильных и слабых сторонах, что позволяет выделить необходимые составляющие для разработки улучшенного алгоритма.

В работе представлен алгоритм, основанный на методах расстановки меток с помощью поиска минимального разреза графа. Гарантируется, что полученное решение будет отличаться от точного не более чем в известных пределах. Представленный алгоритм, в отличие от классических методов, требует меньших вычислительных затрат, что дает возможность обрабатывать изображения с большим расширением. Кроме того, особое внимание уделяется различным методам ускорения алгоритма и дальнейшего уменьшения его вычислительной сложности.

Работа организована следующим образом. После введения приводится формализованная постановка задачи и доказывается ее актуальность. В следующем разделе описываются возможные методы решения, их сильные и слабые стороны, анализируются возникающие проблемы. Далее представлен решающий рассматриваемую задачу алгоритм и его реализация, а также приведены примеры его работы и даны сравнительные характеристики. В заключении перечислены результаты исследования и перспективы дальнейшего развития описанного в дипломной работе алгоритма.

2. Постановка задачи

Множество проблем компьютерного зрения могут быть сформулированы в терминах задачи расстановки меток, целью которой является найти приближение некоторой характеристики изображения.

Присваивание пикселям меток из некоторого заранее заданного множества фактически разбивает изображение на набор областей. Для примера возьмем пару фотографий одного и того же объекта, сделанных с разных ракурсов. Рассмотрим ситуацию, когда метки, присваиваемые пикселям изображения, будут соответствовать смещению (на сколько сдвинулся пиксель одного изображения относительно второго). На Рис. 1 и Рис. 2 изображена пара исходных фотографий одного и того же объекта, сделанная с разных ракурсов. Данные фотографии, взятые из набора 'Tsukuba Dataset', часто используются для демонстрации работы алгоритмов стереозрения. Для наглядности каждой из меток, которые мы будем присваивать пикселям, сопоставим некоторый оттенок. Тогда полученная в результате функция расстановки меток естественным образом может быть представлена в виде изображения, раскрашенного с помощью цветов - меток. Результат такого разбиения изображен на Рис. 3.

Рис. 1: Первое исходное изображение.

Рис. 2: Второе исходное изображение.

Действительно, для многих задач обработки изображений требуется найти приближение некоторой величины, которая ведет себя гладко на поверхности объекта, однако на границах меняется очень сильно. Для решения такого рода задач будем строить функцию $\mathbf{f}$, которая каждому пикселю $\mathbf{p} \in \mathcal{P}$ присваивает метку $\mathbf{f_p}$ из конечного множества меток $\mathcal{L}$, оставаясь при этом кусочно-гладкой и соответствующей рассматриваемым данным.

Рис. 3: Полученное изображение.

2.1. Актуальность

Создание искусственных систем обработки изображений и распознавания образов остается сложной теоретической и технической проблемой. Необходимость в таких системах возникает в самых разных областях. Они могут использоваться в системах видеонаблюдения, в медицине, в технической диагностике и в других областях. Важность и актуальность рассматриваемой задачи обусловлена, в первую очередь, ростом объемов получаемой информации, необходимостью в качественной, максимально быстрой ее обработке.

Многие задачи обработки изображений основываются на задаче расстановки меток. Поэтому, научившись решать ее легко и быстро, можно значительно улучшить многие другие алгоритмы компьютерного зрения.

Сейчас основные сложности в решении подобных задач возникают из-за того, что они требуют слишком больших вычислительных затрат. В приложении к работе [5] показано, что задача минимизации энергии в одном из простейших случаев (для приближения, сохраняющего разрывы на границах объектов) является NP сложной. Следовательно, невозможно решить ее быстро, только если не P=NP.

NP-сложность поставленной задачи вынуждает нас искать не точное, а приближенное решение. Алгоритмы, которые будут рассмотрены в работе, гарантируют, что полученное приближенное решение будет отличаться от глобального минимума не более, чем в извест-

ных пределах.

Однако даже нахождение приближенного решения требует перебора значительного количества вариантов, что приводит к очень большим временным затратам и не позволяет обрабатывать изображения больших размеров. Поэтому важно разработать методы ускорения работы алгоритма, понять, за счет чего можно сократить количество вариантов для перебора, и постараться тем самым уменьшить вычислительные затраты, не теряя при этом точности, которую гарантируют классические подходы к решению поставленной задачи.

2.2. Формальная постановка задачи

Будем искать такую функцию $\mathbf{f}$, чтобы минимизировать

$$\mathbb{E}\left(\mathbf{f}\right) = \mathbf{E}_{\mathbf{s}mooth}\left(\mathbf{f}\right) + \mathbf{E}_{\mathbf{d}ata}(\mathbf{f}),$$

где $\mathbf{E}_{\mathbf{s}mooth}$ показывает, насколько $\mathbf{f}$ не кусочно-гладкая, а $\mathbf{E}_{\mathbf{d}ata}$ - насколько $\mathbf{f}$ соответствует рассматриваемым данным.

Для решения этой задачи мы будем рассматривать взвешенный неориентированный граф $\mathcal{G}$=($\mathcal{V}$, $\mathcal{E}$), где $\mathcal{V}$ - множество вершин графа, а $\mathcal{E}$ - множество ребер. Вес ребра из вершины p в вершину q задается значением $\mathbf{w}_{\mathbf{p}q}$. Каждый способ расстановки меток представляет собой некоторую функцию $\mathbf{f} : \mathcal{V} \to \mathcal{L}$ и соответствует некоторой стоимости, или энергии, которую мы стремимся минимизировать.

Для каждого $\mathbf{p} \in \mathcal{V}$ требуется выбрать такое $\mathbf{x_p} \in \mathcal{L}$ (где $\mathcal{L}$ - это множество меток), чтобы минимизировать энергию $\mathbb{E}$:

$$\mathbb{E}= \sum_{\mathbf{p}\in\mathcal{V}} \mathbf{c_p}(\mathbf{x_p}) + \sum_{(\mathbf{p},\mathbf{q})\in\mathcal{E}} \mathbf{w}_{\mathbf{p}q}\mathbf{d}(\mathbf{x_p}, \mathbf{x_q}),$$

где $\mathbf{c_p}\left(\cdot\right)$, $\mathbf{d}(\,\cdot\,,\ \cdot\,)$ являются унарными и бинарными потенциалами соответственно. При этом потенциалы $\mathbf{c_p}\left(\cdot\right)$ фактически соответствуют стоимости меток, иными словами тому, наскольно сильно они расходятся с рассматриваемыми данными, в то время как потенциалы $\mathbf{d}(\,\cdot\,,\ \cdot\,)$ для меток присвоенных соседним вершинам показывают, насколько сильно они отличаются.

3. Цель дипломной работы

В рамках дипломной работы были поставлены следующие задачи:

- анализ существующих подходов к решению задачи минимизации энергии,
- адаптация существующих алгоритмов для обработки изображений с высоким расширением,
- создание, на основе оптимизации метода FastPD, нового алгоритма, обладающего большей вычислительной эффективностью,
- экспериментальное сравнение созданного метода с существующими подходами,
- анализ результатов.

4. Обзор существующих алгоритмов

Для решения задачи минимизации энергии широко используются методы, основанные на поиске минимального разреза графа. С их помощью можно добиться очень хорошей точности при нахождении приближенного решения. Не стоит, однако, забывать про еще одно важное свойство, которое требуется для реального применения подобных алгоритмов – вычислительная эффективность.

Рассматриваемые далее методы работают, итеративно решая задачи по нахождению максимального потока для некоторого перестраиваемого от итерации к итерации графа. Соответственно, их эффективность непосредственно зависит от того, насколько быстро решается каждая из серии этих задач, что в свою очередь зависит от графа, для которого эта задача решается на каждой из итераций. Кроме того, безусловно, на скорость работы влияет также скорость сходимости приближений к итоговому решению, или, другими словами, количество необходимых итераций.

Обзор существующих решений начнем с предложенного в 2001 году подхода, послужившего впоследствии основой для многих других алгоритмов. Далее рассмотрим несколько более современных методов, достигающих столь же оптимального решения, но требующие уже гораздо меньших вычислительных затрат, и, следовательно, работающих за существенно меньшее время.

4.1. α – expansion

NP – сложность задачи нахождения минимума энергии вынуждает жертвовать точностью решения, ради уменьшения вычислительных затрат на его поиск. Таким образом, в результате работы алгоритма, мы будет получать только лишь приближенное решение, однако его точность на деле оказывается вполне достаточной. Свое название рассматриваемый алгоритм получил в честь названия типов движения (изменения текущего приближения функции), по отношению к которым он вычисляет локальный минимум. Рассмотрим подробнее понятие движения и введем строгое определение локального минимума энергии.

В функция расстановки меток $\mathbf{f}$ достигается локальный минимум энергии $\mathbb{E}$, если $\mathbb{E}(\mathbf{f}) \leq \mathbb{E}(\mathbf{f}')$ для любой $\mathbf{f}'$ близкой к $\mathbf{f}$ (т.е. $\mathbf{f}'$ лежит на расстоянии одного движения от $\mathbf{f}$). Ранее, во многих методах применялись так называемые стандартные движения, при которых только один пиксель может изменить свою метку. Однако в рассматриваемом нами методе

используется тип движений под названием α – *expansion*, который позволяет существенно ускорить процесс нахождения приближенного решения. Дадим им более точное определение.

Для каждой функции расстановки меток $\mathbf{f}$ можно единственным образом определить разбиение пикселей изображения $\mathbf{P}=\{\mathcal{P}_\ell \mid \ell \in \mathcal{L}\}$, где $\mathcal{P}_\ell=\{\mathbf{p} \in \mathcal{P} \mid \mathbf{f_p}=\ell\}$ - это подмножество пикселей изображения, которым присвоена метка ℓ. Для данной метки α, движение от разбиения $\mathcal{P}$ (функция расстановки меток $\mathbf{f}$) к разбиению $\mathcal{P}'$ (функция $\mathbf{f}'$) называется α – *expansion*, если $\mathcal{P}_\alpha \subset \mathcal{P'}_\alpha$ и $\mathcal{P'}_\ell \subset \mathcal{P}_\ell$ для любой метки $\ell \neq \alpha$. Другими словами движение типа α – *expansion* допускает любому множеству пикселей изображения менять присваиваемую метку на метку α.

Алгоритм, основывающийся на движениях типа α – *expansion*, начинает работу с некоторого начального приближения $\mathbf{f}$. Далее, для текущего приближения $\mathbf{f}$ и метки α, ищется такое приближение $\widehat{\mathbf{f}}$, что в нем достигается локальный минимум для энергии $\mathbb{E}$ относительно движений типа α – *expansion*.

Принцип работы данного алгоритма может быть представлен следующим псевдокодом:

1. Start with an arbitrary labeling f
2. Set success := 0
3. For each label $\alpha \in \mathcal{L}$

   3.1 Find $\widehat{\mathbf{f}}=\mathbf{a}rgmin\ \mathbb{E}(\mathbf{f}')$ among $\mathbf{f}'$ within one α – *expansion* of $\mathbf{f}$

   3.2 If $\mathbb{E}(\widehat{\mathbf{f}})< \mathbf{E}(\mathbf{f})$, set $\mathbf{f} := \widehat{\mathbf{f}}$ and success := 1
4. If success = 1 goto 2
5. Return $\mathbf{f}$

На каждом цикле своей работы (шаги 2-3-4) алгоритм выполняет итерацию для каждой метки $\alpha \in \mathcal{L}$. Цикл считается успешно завершенным, если в результате его работы было найдено более хорошее приближение $\mathbf{f}$. Алгоритм останавливается после первого неуспешного цикла. Заметим, что каждый цикл выполняется за $|\mathcal{L}|$ итераций, что может крайне негативно сказываться на скорости работы алгоритма, в случае, если количество возможных меток достаточно велико. Гарантируется, что алгоритм завершит работу за конечное число циклов.

Самой содержательной частью данного алгоритма является шаг 3, где ищется более хорошее приближение. Собственно на этом этапе и применяется алгоритм поиска минимального разреза графа. Более подробную информацию о структуре графа можно почерпнуть в статье [5].

Утверждается, что в решение $\widehat{\mathbf{f}}$, полученное в результате работы данного алгоритма, будет удовлетворять следующему неравенству:

$$\mathbb{E}(\widehat{\mathbf{f}}) \leq \mathbf{2c}\mathbb{E}(\mathbf{f}^*),$$

где $\mathbf{f}^*$ - глобальный минимум. (Доказательство см. в [5]).

4.2. PD3

В статье [7] N. Komodakis и G. Tziritas предлагают несколько новых подходов к решению задачи минимизации энергии.

Основной идеей предложенных ими методов является предложение решать одновременно прямую и двойственную задачи линейного программирования для нахождения наилучшего приближения. Для начала, остановимся более подробно на формулировке рассматриваемой нами проблемы в терминах задачи линейного программирования.

Рассмотрим пару двойственных задач:

Прямая: $\mathbf{m}in\ \mathbf{c^T}\ \mathbf{x},\ \ \mathbf{A}x{=}\mathbf{b},\ \mathbf{x} \geq \mathbf{0}$

Двойственная: $\mathbf{m}ax\ \mathbf{b^T}\ \mathbf{y},\ \ \mathbf{A^T y} \leq \mathbf{c}$

Целью является найти приближенное решение этих задач. Для этого может быть использована следующая схема: продолжаем подбирать пары прямых и двойственных решений ($\mathbf{x^k}$, $\mathbf{y^k}$), пока элементы последней пары $(\mathbf{x}, \mathbf{y})$ не будут оба удовлетворять неравенству: $\mathbf{c^T x} \leq \mathbf{f_{a}}_{pp} \cdot \mathbf{b^T y}$. В этом случае гарантируется, что $\mathbf{c^T x} \leq \mathbf{f_{a}}_{pp} \cdot \mathbf{c^T x^*}$, где $\mathbf{x}^*$-точное решение. Авторы [7] показывают, что данная схема верна при $\mathbf{f_{a}}_{pp}{=}\mathbf{2}\frac{\mathbf{d}_{\mathbf{m}ax}}{\mathbf{d}_{\mathbf{m}in}}$, где $\mathbf{d}_{min} = min_{a \neq b} d(a, b)$ и $\mathbf{d}_{max} = max_{a \neq b} d(a, b), a, b \in L$. То есть гарантируется, что полученный локальный минимум, будет отличаться от глобального не более чем в известных пределах. Для этого достаточно выполнения следующих условий:

$$\mathbf{h_p}\left(\mathbf{x_p}\right) = \mathbf{m}in_{\alpha \in \mathcal{L}} \mathbf{h_p}\left(\alpha\right), \forall \mathbf{p} \in \mathcal{V} \qquad (\mathbf{1})$$

$$\mathbf{y}_{\mathbf{p}q}\left(\mathbf{x_p}\right) + \mathbf{y}_{\mathbf{q}p}\left(\mathbf{x_q}\right) = \mathbf{w}_{\mathbf{p}q}\mathbf{d}\left(\mathbf{x_p}, \mathbf{x_q}\right), \forall \mathbf{p}, \mathbf{q} \in \mathcal{E} \qquad (\mathbf{2})$$

$$\mathbf{y}_{\mathbf{p}q}\left(\mathbf{a}\right) + \mathbf{y}_{\mathbf{q}p}\left(\mathbf{b}\right) \leq \mathbf{2w}_{\mathbf{p}q}\mathbf{d}_{\mathbf{m}ax}, \forall \mathbf{p}q \in \mathcal{E}, \mathbf{a} \in \mathcal{L}, \mathbf{b} \in \mathcal{L} \qquad (\mathbf{3})$$

В этих формулах $\mathbf{x}=\{\mathbf{x_p}\}_{\mathbf{p}\in\mathcal{V}}$ - прямые переменные, характеризуют метки, которые присвоены вершинам $\mathbf{p} \in \mathcal{V}$. $\mathbf{x_p}$ назовем активной меткой для вершины $\mathbf{p}$. Двойственные переменные представлены переменными $\mathbf{y}$(баланс) и $\mathbf{h}$(высота). Для каждого ребра $(\mathbf{p}, \mathbf{q})$ и метки $\mathbf{a}$ существует две переменные баланса $\mathbf{y}_{\mathbf{p}q}\left(\mathbf{a}\right), \mathbf{y}_{\mathbf{q}p}(\mathbf{a})$, на них накладывается требование $\mathbf{y}_{\mathbf{p}q}\left(\cdot\right) \equiv -\mathbf{y}_{\mathbf{q}p}(\,\cdot\,)$. В то время как для каждой вершины $\mathbf{p}$ и метки $\mathbf{a}$ существует одна переменная высоты $\mathbf{h_p}(\mathbf{a})$, причем $\mathbf{h_p}\left(\cdot\right) \equiv \mathbf{c_p}\left(\cdot\right) + \sum_{\mathbf{q}:\mathbf{q}p\in\mathcal{E}} \mathbf{y}_{\mathbf{p}q}(\,\cdot\,)$.

Данные прямые и двойственные переменные итеративно меняются, пока не будет достигнуто выполнение всех требуемых условий.

Рассмотрим теперь интуитивную интерпретацию прямых и двойственных переменных. Будем рассматривать для каждой вершины $\mathbf{p}$ копию всех существующих меток из $\mathcal{L}$. Представим их в виде мячиков, каждый из которых парит на некоторой высоте. Таким образом, в вершине $\mathbf{p}$ мячик, соответствующий метке $\mathbf{a}$, будет висеть на высоте $\mathbf{h_p}(\mathbf{a})$. Кроме того, будем считать, что мячики парами могут менять свою высоту. То есть если мячик, соответствующий метке $\mathbf{c}$ в вершине $\mathbf{q}$ поднялся на высоту δ, то мяч, соответствующий метке $\mathbf{c}$ в вершине $\mathbf{q}$ должен опуститься на δ, чтобы сохранить выполнение условия $\mathbf{y}_{\mathbf{p}q}\left(\mathbf{c}\right) \equiv -\mathbf{y}_{\mathbf{q}p}(\mathbf{c})$. Таким образом, переменные баланса отвечают на поднятие или опускание меток в вершинах.

Так называемый PD3 алгоритм итеративно двигает (поднимает и опускает) метки, пока не будет достигнуто выполнение требуемых условий (1-2-3). Для этого применяется следующая стратегия: гарантируется постоянное выполнение условий (2-3) и задача сводится к нахождению такой функции расстановки меток, которая удовлетворяла бы условию (1). Метки двигаются группами. Более точно – алгоритм выполняется для всех меток $\alpha \in \mathcal{L}$, и на каждой итерации возможно движение только одинаковых меток. На каждой итерации алгоритма для изменения двойственных переменных решается задача нахождения максимального потока на графе. Более подробную информацию о структуре графа можно найти в статье [7].

4.3. FastPD

В статье [9] N. Komodakis, G. Tziritas и N. Paragios предлагают новый подход к решению задачи минимизации энергии, основывающийся на предыдущем рассмотренном методе, однако во много раз превосходящий его по скорости вычислений. Так как в дипломной работе оптимизируется именно этот алгоритм, остановимся на описании принципа его работы более подробно.

Общий принцип его работы очень похож на принцип работы алгоритма PD3 и может быть представлен следующей схемой:

1. $[\mathbf{x}, \mathbf{y}] <$ Init_Duals_Primals(); $\mathbf{x}_{\mathbf{o}ld} \leftarrow \mathbf{x}$
2. For each label $\mathbf{c}$ in $\mathcal{L}$ do
3. $\mathbf{y} \leftarrow$ Preedit_Duals ($\mathbf{c}$, $\mathbf{x}$, $\mathbf{y}$);
4. $\left[\mathbf{x}^{'}, \mathbf{y}^{'}\right] \leftarrow$ Update_Duals_Primals ($\mathbf{c}$, $\mathbf{x}$, $\mathbf{y}$);
5. $\mathbf{y}^{'} \leftarrow$ Postedit_Duals ($\mathbf{c}$, $\mathbf{x}^{'}$, $\mathbf{y}^{'}$);
6. $\mathbf{x} \leftarrow \mathbf{x}^{'};; \mathbf{y} \leftarrow \mathbf{y}^{'};;$
7. End for
8. If $\mathbf{x} \neq \mathbf{x}_{\mathbf{o}ld}$ then
9. $\mathbf{x}_{\mathbf{o}ld} \leftarrow \mathbf{x};;$ goto 2;
10. End if

На 4 шаге схемы происходит изменение высоты шариков соответствующих рассматриваемой метке $\mathbf{c}$. Пусть например мы рассматриваем вершину $\mathbf{p}$, активной меткой для которой является $\mathbf{x}_{\mathbf{p}}$. Пусть кроме того, метка $\mathbf{c}$ в вершине $\mathbf{p}$ находится ниже, чем $\mathbf{x}_{\mathbf{p}}$. Тогда, для того, чтобы достичь выполнения условия (1), будем увеличивать высоту метки $\mathbf{c}$. Однако, изменяя $\mathbf{h}_{\mathbf{p}}(\mathbf{c})$, мы рискуем нарушить остальные 2 условия, выполняемость которых требуется сохранять в течении всего времени работы алгоритма. Поэтому возникает необходимость в шаге 5 и операции Postedit_Duals, которая обеспечивает необходимую корректировку обновленных значений двойственных переменных для того, чтобы они удовлетворяли неравенствам 2-3.

Главное отличие алгоритма FastPD от ранее рассмотренного PD3 заключается именно в принципе работы этапа корректировки улучшенных значений переменных (Postedit_Duals). Дело в том, что алгоритм FastPD корректирует полученные значения только в случае, если это крайне необходимо (что случается довольно редко), в то время как PD3 рискует испортить улучшенные на предыдущем шаге значения, меняя их всякий раз, как только $\mathbf{c} \neq \mathbf{x_p}$, $\mathbf{c} \neq \mathbf{x_q}$(что случается невероятно часто).

Подобное отличие дает алгоритму FastPD огромное преимущество и существенный выигрыш по времени. Более подробную информацию и экспериментальные данные можно посмотреть в работе [9].

5. Анализ существующих алгоритмов

Рассмотренные выше алгоритмы, основывающиеся на поиске минимального разреза графа, широко применяются в различных задачах компьютерного зрения. Они позволяют находить приближенное решение с достаточно хорошей точностью и используются в задачах, где применяется расстановка меток для пикселей изображения – сегментации изображений, выделении объекта, удалении шума, стереозрении и др.

Базовым классическим методом, который был предложен в 2001 году, можно считать α – *expansion*. На нем основывается множество впоследствии предложенных улучшений способов решения поставленной задачи. Метод α – *expansion* не обладает высокой вычислительной эффективностью, однако в результате его работы мы получаем решение, отличающиеся от точного не более чем в известных пределах (см. [5]).

За последние пару лет было предложено довольно много методов, улучшающих эффективность алгоритмов, подобных α – *expansion*. Одним из самых быстрых методов, существующих на данный момент, является FastPD. Он обладает значительным выигрышем по времени, не теряя при этом точности получаемых результатов, по сравнению с PD3, α – *expansion* и др. Его эффективность достигается за счет того, что в процессе работы он использует информацию не только от прямой задачи, но и от двойственной.

Однако, и FastPD и большинство других подходов не используют в своей работе знания о функционале энергии для уменьшения времени вычислений. Все эти методы вынуждены на каждой своей итерации проходить по всем существующим меткам, что крайне негативно сказывается на их вычислительной эффективности. Если учесть, что общее количество меток может быть довольно велико, в то время как реально значимыми из них чаще всего оказываются совсем немногие, кажется естественной идея присваивать меткам приоритеты, а не перебирать на каждом шаге их все.

6. Реализация

Для написания системы был выбран язык C++.

На вход программе поступает пара изображений в формате PPM. Этот формат достаточно прост и удобен в использовании и позволяет хранить информацию о цветных изображениях.

Сначала, производится предварительная обработка изображений для инициаллизации данных. На основе данных о каждом из пикселей изображения вычисляются значения для задания массивов данных label_costs, label_distances и weights, само изображение далее рассматривается как граф $\mathcal{G}$. Информация о нем хранится при помощи списка ребер и вышеперечисленных массивов данных. Вершины $\mathcal{V}$ графа - это пиксели изображения. Ребрами $\mathcal{E}$ вершина соединена с пикселями соседями. То есть, не считая крайних случаев, из каждой вершины выходит по четыре ребра. Label_costs – это массив размером (количество меток * количество вершин). Он содержит для каждой возможой пары метка-вершина стоимость метки в вершине, то есть насколько метка соответствует хранящимся в вершине данным. Элемент массив label_costs [l*количество вершин + p] – это $\mathbf{c_p}(l)$ см. "формальная постановка задачи". Label_distances – это массив размером (количество меток * количество меток), содержащий информацию о "расстоянии"между метками. Другими словами о том, насколько сильно метки в каждой возможной паре меток отличаются. Элемент массива label_distances [l1*количество меток + l2] – это $\mathbf{d}(l1, l2)$ см. "формальная постановка задачи". Weights – это массив весов ребер графа. Если i –порядковый номер ребра (p, q), то элемент массива weights [i] – это $\mathbf{w_{pq}}$ см. "формальная постановка задачи".

Стоит обратить особое внимание на этап задания входных параметров по данным изображения - значений массивов label_costs, weights, выбор общего количества меток в $\mathcal{L}$. Как видно из приведенных далее примеров, даже незначительное изменение пороговых значений, или использование различных метрик для пересчета, приводит к существенным отличиям в результатах.

Например для Рис. 5 значение label_costs ($\mathbf{c}_p(l)$) задаются по следующему принципу: $\mathbf{c_p}(l)$=$|I_r(p+l) - I_l(p)|^2$. В то время как для Рис. 6 и Рис.7 $\mathbf{c_p}(l)$=$min(t1, |I_r(p+l) - I_l(p)|)$, где $\mathbf{t}$ – некоторое пороговое значение, а $\mathbf{I}_r, I_l$ - правое и левое входные изображения (см. Рис. 1 и Рис. 2). То есть стоимость метки $\mathbf{l}$ в вершине $\mathbf{p}$ означает, насколько отличются пиксели правого изображения от сдвинутых на $\mathbf{l}$ пикселей левого. Для задания массива

label_distances использовалась так называемая метрика Потта. Для Рис. 5 – $\mathbf{d}(l1, l2) = 1$, если $\mathbf{l}1 \neq l2$. Для Рис. 6 – $\mathbf{d}(l1, l2) = min(t, |l1 - l2|)$, где $\mathbf{t}$ – некоторое пороговое значение. Массив weights для большинства тестов задавался просто константой. Заметим, что это практически не сказывалось на качестве получаемых результатов, поскольку за сччт массива label_distances, при расставлении меток учитывалось то, что рядом на изображении метки должны быть более ли менее похожими.

Для Рис. 4 выбрано N = 7 – небольшое (по сравнению с остальными тестами) количество возможных меток. Поэтому получившийся результат заметно проигрывает по качеству, однако он был получен за гораздо меньшее время, что для целей некоторых задач, может оказаться приемлимым. Ведь иногда лучше получить грубое разбиение очень быстро, чем гнаться за качеством результатов, тратя при этом на их подсчет очень много времени.

Для Рис. 5, Рис. 6 и Рис. 7 возможное количество меток N = 16.

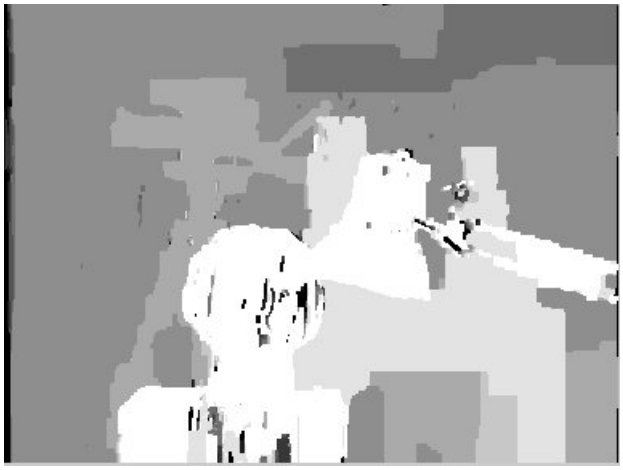

Рис. 4: Входные параметры 1.

Сначала над полученными данными выполняется процедура, которая создает граф, который далее будет обрабатываться в алгоритме. Кроме того, инициализируются некоторые начальные приближения к решению, которые далее будут итеративно улучшаться. Фактически, начальное приближеное строится очень просто – всем вершинам сначала присваивается одна и та же метка. Это практически не сказывается на времени работы алгоритма, поскольку за первую же итерацию, текущее приближение будет достаточно хорошо улучшено. После инициаллизации начинает работу процедура, в которой выполняется вся основная часть алгоритма. Данная процедура представляет собой цикл, который выполняется, пока

Рис. 5: Входные параметры 2.

Рис. 6: Входные параметры 3.

не будет достигнуто достаточно хорошее приближение к точному решению, свершая при этом не более чем max_iters количество итераций. Рассмотрим теперь, что собой представляет каждая из итераций.

Во время выполнения каждой из итераций происходит самая содержательная часть работы алгоритма, а именно, последовательно перебираются все возможные метки $\mathbf{l} \in \mathcal{L}$ и для каждой из меток выполняется процедура, которая пытается улучшить текущее приближение к решению. Рассматриваемый в работе метод решения задачи минимизации функцио-

Рис. 7: Входные параметры 4.

нала энергии базируется на алгоритме нахождения минимального разреза графа. Поэтому остановимся на нем более подробно.

6.1. Graph-Cut

Пусть граф $\mathcal{G}$=($\mathcal{V}$, $\mathcal{E}$) – взвешенный неориентированный граф с двумя выделенными вершинами (терминалами). Тогда назовем разрезом графа $\mathcal{G}$ такое множество ребер $\mathcal{C} \subset \mathcal{E}$, что не существует пути между терминалами в графе $\mathcal{G}\,(\mathcal{C})$ =($\mathcal{V}$, $\mathcal{E}$–$\mathcal{C}$). Кроме того, никакое подмножество $\mathcal{C}$ не должно обладать этим свойством. Стоимостью разреза, назовем сумму весов всех принадлежащих ему ребер. Тогда минимальный разрез – это разрез, стоимость которого минимальна.

Для того, чтобы найти функцию $\mathbf{f}$, которая бы для данного графа $\mathcal{G}$ минимизировала бы энергию $\mathbf{E}(f)$ предлагается действовать следующим образом: выбираем за основу некоторое начальное приближение $\mathbf{f_0}$ и далее постепенно улучшаем его. За очередной шаг алгоритма итеративно перебираются все возможные метки $\mathbf{l} \in \mathcal{L}$ и строится новое приближение $\mathbf{f_n}$, такое что $\mathbf{E}(f_n) < \mathbf{E}(f_{n-1})$.

В работе [7] показано, что на каждой итерации алгоритма для каждой из меток, искать, на сколько нужно изменить текущее приближение, можно при помощи алгоритма поиска минимального разреза графа. Вершинами этого графа $\mathcal{G}^{\mathbf{c}}$ будут являются как все верши-

ны исходного графа $\mathcal{G}$ (будем называть их внутренними), так и еще две дополнительные вершины (внешние). Кроме того, ребра графа $\mathcal{G}^{\mathbf{c}}$ тоже поделим на два типа - внутренние ребра и внешние ребра. Внутренними будем считать парные ребра pq и qp, где p и q - соседние версины в исходном графе $\mathcal{G}$. Внешними ребрами назовем те, которые соединяют внутреннюю вершину графа с одной их внешних. Таким образом из каждой внутренней вершины графа будет выходить одно внешнее ребро и несколько внутренних – по числу ее соседей. В то время как из внешних вершин будет выходить множество внешних ребер – к внутренним вершинам графа. Веса ребер и значения вершин графа $\mathcal{G}^{\mathbf{c}}$ вычисляются исходя из соответствующих значений для исходного графа $\mathcal{G}$, однако способ вычислений может несколько варьироваться в зависимости от рассматриваемого алгоритма.

Стоит отметить, что вес внутренних ребер показывает, насколько сильно могут отличаться метки, присвоенные соединенным ребром вершинам. Другими словами, с помощью внутренних ребер поддерживается одно из требуемых условий, а именно то, что функция в основном должна быть гладкой, но при этом сохранять разрывы на границах объектов. Что касается внешних вершин и внешних ребер - они отвечают за улучшение приближенного решения. То есть пусть на очередной итерации алгоритма $\mathbf{f_n}$ есть текущее приближение к решению и, перебирая все метки из $\mathcal{L}$, мы рассматриваем некоторую метку $\mathbf{l} \in \mathcal{L}$ и пытаемся посмотреть, не лучше ли метку какой-нибудь из вершин заменить на $\mathbf{l}$. Тогда для каждой из вершин именно с помощью внешних ребер определяется, насколько мы, меняя текущую метку вершины на метку $\mathbf{l}$, можем улучшить рассматриваемую функцию. При этом фактически каждая из внутренних вершин должна быть соединена с одной из двух внешних, но не с двумя одновременно. Если текущая метка, присвоенная внутренней вершине не удовлетворяет требуемым условиям и требуется что-то изменить, то соединяем данную вершину с первой из внешних. Другими словами, если рассматривать текущее приближение – функцию $\mathbf{f_n}$ и функцию $\mathbf{f_{n+1}}$, которая отличается от $\mathbf{f_n}$ только в данной вершине (поскольку присваивает ей метку $\mathbf{l}$), то если $\mathbf{E}(f_n) > \mathbf{E}(f_{n+1})$, то соединяем ее с первой внешней вершиной, иначе – со второй. То есть если условия для данной вершины можно считать выполненными: $\mathbf{E}(f_n) < \mathbf{E}(f_{n+1})$, что означает, что текущая метка для данной вершины лучше, чем метка $\mathbf{l}$, то она будет соединена внешним ребром со второй внешней вершиной.

Таким образом, алгоритм заканчивает работу, когда достигается выполнение требуемых условий для каждой из вершин, то есть в терминах графов – когда все внутренние вершины оказываются присоединенными ко второй внешней вершине. Для достижения этой цели на

каждой своей итерации алгоритм стремится уменьшить колическтво вершин связанных с первой внешней вершиной наиболее эффективным способов, учитывая при этом ограничения, которые накладывают на вершину ее связи с соседями.

6.2. Дополнительные улучшения

Таким образом, перебирая последовательно от итерации к итерации все возможные метки, и решая для каждой из них задачу поиска минимального разреза графа, мы приходим от элементарного начального приближение, которое всем пикселям изображения присваивает одну и ту же метку, к достаточно точному приближению к решению. Однако, естественным образом возникает вопрос, нельзя ли как-то сократить перебор? Ведь кажется, что перебирать все возможные метки, выполняя для каждой из них достаточно вычислительносложные действия несколько неоправданно. Действительно, что если набор $\mathcal{L}$ состоит из очень большого количества меток? Все ли они столь важны для достижения хорошего результата?

Ответ на эти вопросы приводит нас к возможности существенно улучшить существующий алгоритм за счет сокращения перебора и рассмотрения меток в отсортированном, а не в произвольном порядке. Действительно размер набора меток $\mathcal{L}$ и порядок меток в нем может очень сильно сказываться на вычислительной сложности и соответственно на скорости работы алгоритма. Было проведено множество исследований с целью научиться выбирать наиболее подходящее $\mathcal{L}$ и далее наилучшим образом с ним работать, что на каждом шаге алгоритма позволило бы находить оптимальное изменение текущего решения (такое, которое сильнее всего уменьшало бы $\mathbf{E}(f)$) за полиномиальное время [6, 10–12]. Однако, оказывается, что если использовать данные для текущего приближения, то можно на каждой итерации расставлять меткам из $\mathcal{L}$ приоритеты и сортировать их в соответствии с ними.

Стоит отметить, что для большинства задач важными являются всего несколько меток, а остальные существенной роли не играют. Таким образом и возможные улучшения текущего приближения при рассмотрении наиболее приоритетных меток оказываются несравнимо существенней, чем улучшения, которых можно достичь, рассматривая низкоприоритетные метки. Это становится особенно актуальным, если говорить об использовании рассматриваемых нами алгоритмов в задачах сегментации изображений и стереозрения. Действительно, чаще всего достаточно четко выделить 2–3 основных объекта на изображении, не затрачи-

вая на это слишком много времени, чем считать очень долго и получить в итоге разбиение на 20–30 областей, большая часть которых существенной роли не играет.

Итак, мы приходим к тому, что на каждой итерации, основываясь на обрабатываемых данных и текущем приближении выгодным оказывается сначала найти метку, которая по нашим прогнозам даст наибольший выйгрыш, а только после этого запускать процедуру поиска улучшений, но уже основываясь именно на данной метке.

В системе расстановка приоритетов меткам реализована следующим образом. Перед выполнением очередной итерации, направленной на улучшение текущего приближения путем расстановки некоторой метки вершинам, производится поиск этой метки, среди всего набора меток $\mathcal{L}$. Для этого заводится массив, в котором для каждой метки хранится число вершин, присваивание данной метки которым улучшит текущее приближение.

Стоит отметить, что помимо использованных в работе, существует еще множество различных подходов к решению задачи оптимизации и ускорения рассматриваемых алгоритмов. Это открывает нам широкие возможности, для улучшений существующей системы. В дальнейшем планируется применить рандомизированные методы на основе "сценарного подхода" [2], а также такие методы как вычисление псевдоградиента многомерной функции (потенциала) вдоль случайного направления [1].

7. Заключение

В работе описана задача поиска минимума функционала энергии для марковских случайных полей, демонстрируется ее актуальность и практическая применимость в различных задачах компьютерного зрения, таких как стереозрение и сегментация изображений. Производится обзор основных подходов и алгоритмов, применяющихся для ее решения и на основе их анализа выделяются их слабые места и возможные пути улучшения. Подробно описывается алгоритм, созданный на основе таких методов, как Graph-Cut и FastPD, а также приводятся наглядные примеры его работы.

Список литературы

[1] *Граничин О.Н.* Рандомизированные алгоритмы стохастической аппроксимации при произвольных помехах // Автоматика и Телемеханика, 2002, 2. С. 44-55.

[2] *Граничин О.Н., Шалымов Д.С., Аврос Р., Волкович З.* Рандомизированный алгоритм нахождения количества кластеров // Автоматика и Телемеханика, 2011, 4. С. 86-98.

[3] *Alahari K., Kohli P., Torr P. H. S.* Dynamic Hybrid Algorithms for MAP Inference in Discrete MRFs // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2010. Vol. 32 No. 10. P. 1846-1857.

[4] *Batra D., Kohli P.* Making the Right Moves: Guiding Alpha-Expansion using Local Primal-Dual Gaps // IEEE Conf. On Computer Vision and Pattern Recognition. Providence, RI. 2011. P. 1865-1872.

[5] *Boykov Y., Veksler O., Zabih R.* Fast Approximate Energy Minimization via Graph Cuts // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2001. Vol. 23 No. 11. P. 1222-1239.

[6] *Gould S., Amat F., Koller D.* Alphabet soup: A Framework for Approximate Energy Minimization // IEEE Conf. On Computer Vision and Pattern Recognition. 2009. P. 903-910.

[7] *Komodakis N., Tziritas G.* A New Framework for Approximate Labeling via Graph Cuts // Proc. 10th IEEE International Conference on Computer Vision. Beijing. 2005. Vol. 2. P 1018-1025.

[8] *Komodakis N., Tziritas G.* Approximate Labeling via Graph Cuts Based on Linear Programming // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2007. Vol. 29 No. 8. P. 1436-1453.

[9] *Komodakis N., Tziritas G., Paragios N.* Fast, Approximately Optimal Solutions for Single and Dynamic MRFs // IEEE Conference on Computer Vision and Pattern Recognition. Minneapolis, MN. 2007. P. 1-8.

[10] *Komodakis N., Tziritas G., Paragios N.* Perfomance vs Computational Efficiency for Optimizing Single and Dynamic MRFs: Setting the State of the Art with Primal-Dual Strategies. // Computer Vision and Image Understading. 2008.

[11] *Lempitsky V.S., Rother C., Roth S., Blake A.* Fushion Moves for Markov Random Field Optimization// IEEE Transactions on Pattern Analysis and Machine Intelligence. 2010.

[12] *Veksler O.* Graph cut based optimization for mrfs with truncated convex priors// IEEE Conf. On Computer Vision and Pattern Recognition. 2007.