# Обзор алгоритмов стереозрения[1]

*А. Т. Вахитов, Л. С. Гуревич, Д. В. Павленко*
*Санкт-Петербургский государственный университет*
av38@yandex.ru, gurevich.lev@gmail.com, dmit10@gmail.com

Задача стереозрения состоит в использовании двух или нескольких камер для получения данных о дальности до объекта. При этом входные данные могут представлять собой как просто отдельные изображения, так и видеоряд. Мы приводим обзор алгоритмов, связанных с частным случаем — сопоставлением двух изображений.

*Ключевые слова*: стереозрение.

## 1. Введение

Видео и фото тесно вплелись в нашу жизнь. Почти каждый мобильный телефон оснащен камерой. Почти каждая камера умеет записывать видео. Повсеместно распространилась 3D-графика. С развитием возможностей усиливается потребность в "дешевом" построении 3D-сцен. Самый очевидный из таких методов — стереозрение — получение трехмерной картины мира по видеоряду или нескольким изображениям.

Есть и другие возможные применения. В киноиндустрии — для создания спецэффектов и 3D-фильмов. В военном деле — для измерения расстояний. Интересное применение стереозрения предложено в [1] для проведения видеоконференцсвязи. Как правило, камера расположена сбоку от монитора, где отображается лицо собеседника. Таким образом, другой участник смотрит куда-то в сторону. Построение трехмерного изображения лица позволяет синтезировать новое изображение, на котором собеседник смотрит прямо в глаза, что усиливает эффект присутствия.

Совмещение изображений, позволяет человеку получить информацию о расстоянии до объектов по их расхождениям (disparity). Эта идея может быть использована в алгоритмах обработки изображений. Хотя, конечно же, зрение человека активное (то есть параметры оптической системы настраиваются под изображение) —

глаза вращаются в глазницах, меняется фокусное расстояние, как правило, построение системы активного зрения сложнее и дороже, чем зрения пассивного.

Система пассивного стереозрения, как правило, включает в себя 2 камеры. Хотя авторы [2] предлагают заменить их одной и системой из плоских или кривых зеркал, большинство алгоритмов решают общую задачу сопоставления 2-х изображений.

Существуют различные классификации алгоритмов сопоставления 2-х. Один из вариантов такой классификации представлен в [3]. Все алгоритмы делятся на локальные (в которых расхождение вычисляется в каждой точке на основе "похожести" окна вокруг этой точки и окна вокруг точек на другом изображении) и глобальные (основанные на минимизации функционала энергии — мы находим расхождение сразу для всех точек). Глобальные в свою очередь делятся по способу минимизации энергии. Как правило, это динамическое программирование [4–7] или нахождение минимального разреза графа [8, 9]. Алгоритмы разреза графа называют также двумерными. Они дают довольно точные результаты, но имеют меньшую производительность. Алгоритмы, обрабатывающие строки изображений независимо друг от друга [4, 5], называют одномерными. Они работают быстрее, но подвержены эффекту гребенки, с которым борются с помощью разных ухищрений. Нечто среднее по производительности и качеству из себя представляют алгоритмы оптимизации на поддереве графа, построенного на пикселях изображения.

Многие алгоритмы построены на модели случайных полей Маркова (MRF). Основное предположение в такой модели: расхождение в любой точке зависит только от расхождений соседних точек (как правило, считают, что их 4, хотя можно соседними считать и 8 точек, оставаясь в рамках модели). Минимизация энергии в них производится на основе разрезов графа или распространения доверия [10].

Кроме алгоритмов сопоставления 2-х изображений существуют и другие, выходящие за область нашего рассмотрения, например, алгоритм заметания плоскости [11].

## 2. Исходные предположения

Большинство алгоритмов подразумевают некоторые исходные предположения об окружающем мире. Перечислим наиболее распространенные.

- Самое типичное требование — ламбертовость поверхностей, то есть независимость освещения поверхности от угла зрения. Как правило, алгоритмы стереозрения вообще не учитывают законы распространения света.
- Поверхности в реальном мире кусочно-гладкие. Это требование чаще всего выражается в виде штрафов за различие расхождений или интенсивностей соседних пикселей.
- Предполагается, что камеры откалиброваны. Это значит, что известны как внутренние (оптические), так и внешние параметры (т. е. расположение в пространстве).
- Многие алгоритмы (например, [4]) подразумевают ограничение упорядоченности (order constraint). Это значит, что если на левом изображении точки идут слева направо в каком-то порядке, то в правом изображении они следуют в том же порядке (рис. 1).
- Большинство требуют, ректификации (rectification, встречается перевод — очистки) стереопары, то есть выполнения преобразования, при котором и правое, и левое изображение проецируются на плоскость, параллельную базовой линии (линии, соединяющей оптические центры камер). Тогда если $(x, y)$ — точка левой проекции, а $(x', y')$ — соответствующая ей точка правой проекции, то $y = y'$ (рис. 2).
- Распространенное требование — соответствие границ интенсивности (intencity edges) и резких перепадов функции расхождения (disparity edges). Это значит, что на границах объектов интенсивность меняется достаточно резко.
- Почти все алгоритмы предполагают, что каждому пикселю одного изображения соответствует пиксель другого, в противном случае он считается заслоненным. Впрочем некоторые алгоритмы игнорируют работу с заслоненными пикселями.

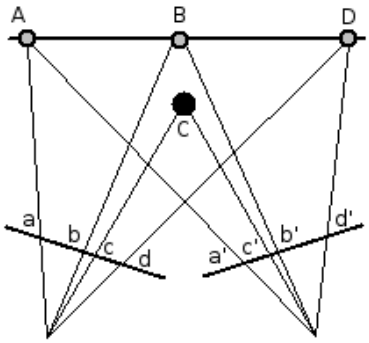


Рис. 1: Ограничение упорядоченности нарушено.

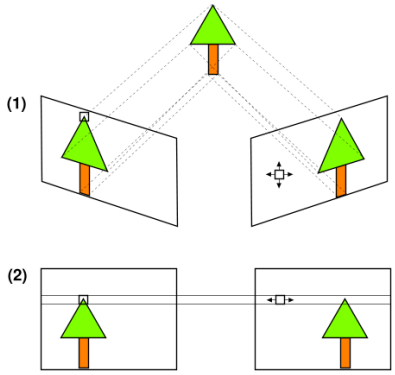


Рис. 2: Ректификация. Изображения с камер (1) проектируем на плоскость, параллельную базовой линии (2). Каждая строчка левого изображения соответствует той же строчке правого.

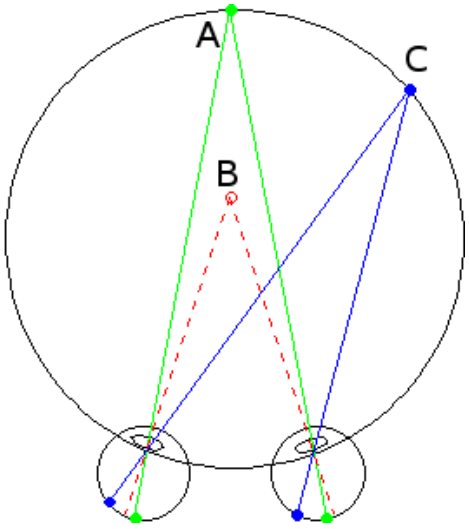


Рис. 3: Окружность Виета-Мюллера.

## 3. Представление результатов

Большинство алгоритмов как результат своей работы вычисляют функцию расхождения (disparity function) по отношению к какому-нибудь из изображений (или это может быть по отношению к некоторому циклопическому виду, как например, в [12]). Понятие расхождения (disparity, другой перевод — диспаритет) пришло из области исследования человеческого зрения. Расхождением в этом случае называется разность между углом фокусировки взгляда A, и углом, который образуют с объектом центры зрачков B [13]. Точки с нулевым расхождением образуют окружность. Она носит название окружности Виета-Мюллера (рис. 3).

В случае компьютерного зрения, как правило, функция расхождения вычисляется как $d(x, y) = x' - x$ (считаем, что стереопара ректифицирована). За $(x, y)$ обозначены координаты точки на изображении, принятом за основу (reference image), $(x', y')$ — соответствующая ей точка на другом изображении. Таким образом, $d(x, y)$ получает смысл величины обратной глубины точки (поскольку, $d(x, y) = \frac{|OO'|f}{z}$, где $O, O'$ — оптические центры камер,

соответственно, $|OO'|$ — длина базовой линии, а $f$ — фокусное расстояние камер (считаем, что оно совпадает), $z$ глубина "циклопического" вида). Точки $(x, y, d)$ образуют пространство расхождений.

## 4. Алгоритмы на основе динамического программирования

### 4.1. Динамическое программирование на DSI

В работе [4] вводится конструкция DSI (disparity space image). Предполагается, что стереопара ректифицирована. Пусть $s_i$ и $s'_i$ — соответствующие $i$-е строки левого и правого изображений, $I(x, y)$ и $I'(x, y)$ — соответствующие функции интенсивности. Тогда

$$DSI_i(x, d) = ||I(x, i) - I'(x - d, i)||,$$

где $N$ — ширина картинки, $d_{max}$ — максимальное допустимое расхождение на изображении, $-d_{max} \leq d \leq d_{max}$ и $0 \leq (x + d) \leq N$. Другой способ построения $DSI_i$ — вместо разности интенсивностей рассматривать величину, основанную на корреляции окон вокруг пикселей $(x, i)$ и $(x-d, i)$, при этом окна выбираются адаптивно (например, если точка на границе объектов, то окно с центром в этой точке будет плохим, лучше использовать окно, которое полностью лежит только в одном объекте). Отбрасываем строки $DSI_i$, которые соответствуют заведомо невозможным значениям $d$ (например, из условий, что пиксель на левом изображении должен быть правее соответствующего ему на правом, и из условия $|d| \leq d_{max}$). Задача сводится к поиску оптимального пути на полученной двумерной матрице (рис. 4). При этом за каждый тип движения назначается определенный штраф. Типы движения бывают: по горизонтали, по вертикали и по диагонали. Последние 2 типа соответствуют заслоненным областям (присутствующим только на одном из изображений стереопары). Здесь стоит отметить, что если некоторая поверхность на левом изображении имеет ширину 9 пикселей, а на правом — 3 пикселя, то в этой терминологии 3-м пикселям из изображения слева соответствуют 3 пикселя справа (это могут быть 3-й, 6-й и 9-й), остальные 6 в считаются заслоненными.

Задача решается методом динамического программирования.

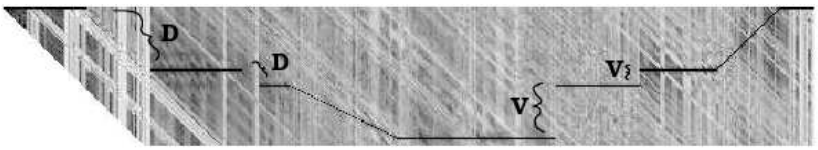


Рис. 4: $DSI_i$. Горизонтальная ось – x. Вертикальная ось – d. Черные пиксели соотвествуют близким к 0 значениям. Скачки V и D соответствуют заслоненным областям. Горизонтальные участки означают нахождение точного соответствия. Наклонные участки соответствуют наклонным поверхностям.

Если на изображении заранее известны GCP (ground control points – точки, положение и соответствие которых мы можем определить достаточно точно), количество возможных путей сокращается за счет использования ограничения: путь должен проходить так, чтобы он не противоречил расстановке GCP. Возможны также многозначные GCP — случай, когда точке одного изображения может соответствовать одна точка из небольшого набора вариантов на другом изображении.

Одна из главных особенностей метода состоит в том, что каждая строка обрабатывается независимо. Как отрицательное последствие: решение (функция глубины пикселя) может иметь вид гребенки. Положительным моментом является возможность распараллелить вычисления. Касательно производительности, авторы отмечают, что бо̀льшая часть времени тратится на построение DSI, в частности, на вычисление корреляций.

Чтобы уменьшить эффект гребенки, в [4] предлагается эвристика: заранее каким-нибудь образом выделить границы на изображениях (по резкому перепаду интенсивности, например). Границам объектов на DSI соответствуют вертикальные и диагональные линии. Тогда уменьшим стоимость пути, если он проходит по этим линиям.

В [5] предлагается другой способ борьбы с гребенкой: поиск оптимального пути про трехмерной конструкции, каждый слой которой — конструкция, похожая на DSI. Основная идея: если одна точка какой-то границы на левом изображении соответствует точке на правом, то точки на пересечении следующей строки и этой

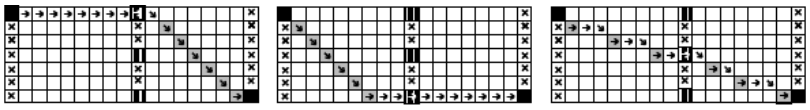

Рис. 5: $DSI_i$. Ограничение количества возможных путей с помощью GCP и многозначных GCP. Черные квадраты — GCP, штрихованные — многозначные. Путь должен проходить через GCP и один из вариантов многозначной GCP. Серые квадраты соответствуют заслоненным пикселям. Ясно, что минимально возможное количество заслоненных пикселей — 6 в данном случае — можно посчитать заранее. Таким образом, если уже найден путь, в котором ровно 6 заслоненных пикселей, он оптимален.

границы на левом и на правом изображениях тоже должны соответствовать (см. рис. 5).

### 4.2. Динамическое программирование по дереву

Большинство алгоритмов стереозрения строятся на идее минимизации функционала энергии

$$E(f) = E_{data}(f) + E_{smooth}(f) + E_{occ}(f),$$

где $f$ - конфигурация (разбиение точек на заслоненные и пары соответствующих друг другу); $E_{data}(f)$ — суммарный штраф за несоответствие свойств (как правило, интенсивностей) точек таких пар; $E_{smooth}(f)$ — суммарный штраф за непохожесть свойств соседних пикселей (опять же, как правило, интенсивностей); $E_{occ}(f)$ — суммарный штраф за объявление пикселей заслоненным (occluded). В некоторых алгоритмах $E_{occ}(f)$ не учитывается.

В [6] предлагается применение идеи динамического программирования для минимизации функционала

$$E(f) = E_{data}(f) + E_{smooth}(f), \;\; E_{data}(f) = \sum_{x \in \mathbf{P}} m_f(x),$$

где $m_f(x) = min\{|I(x) - I'(x')|, \tau\}$, $\mathbf{P}$ — множество пикселей в левом изображении, $x'$ соответствует $x$ согласно конфигурации $f$ (вариант с заслоненными пикселями не рассматриваем), $I, I'$ — функции интенсивности левого и правого изображений, $\tau$ — некоторая

константа. Пусть $G = (\mathbf{P}, N)$ — граф с вершинами-пикселям. Каждый пиксель (кроме крайних) соединен с 4-мя своими соседями (таким образом $N$ — множество соседних пикселей). Тогда

$$E_{smooth}(f) = \lambda \sum_{(x,\hat{x})\in N} s(d(x), d(\hat{x})),$$

где $s(d_1, d_2)$ — функция гладкости (функция, которая паре расхождений ставит в соответствие штраф за то, что они отличаются).

Если мы каким-либо образом выберем дерево-подграф $G$ (с выделенным корнем $root$) — $G' = (\mathbf{P}, N')$, то алгоритм минимизации энергии будет выражаться рекуррентной формулой

$$E_x(d(parent(x))) = \min_{d(x)\in D}(m_f(x) + s(d(x), d(parent(x)))+$$

$$\sum_{\hat{x}\in children(x)} (E_{\hat{x}}(d(x)))),$$

причем минимум берется по всем возможным присваиваниям пикселю-узлу $x$ расхождения $d(x)$, $D$ — множество возможных расхождений. В случае $x = root$ — корень, слагаемое $s(d(x), d(parent(x)))$ считаем равным 0. В процессе минимизации $E_{root}(d(root))$ мы для каждого пикселя $x$ выбрали расхождение $d(x) \in D$, таким образом определили (оптимальную) конфигурацию $f$.

В качестве $G'$ автор статьи предлагает два варианта MID Tree и MIDDT Tree. MID Tree — это наименьшее остовное дерево $G$ из того расчета, что вес ребра равен

$$v_{x\hat{x}} = |I(x) - I(\hat{x})|.$$

Такое дерево будет не уникально, очень много ребер будут иметь одинаковые веса. В качестве альтернативы строим дерево по немного другому принципу. Для каждой точки мы определим функцию $D(x)$, которая пикселю сопоставляет, насколько далеко от границ он содержится в некой однородной (например, по интенсивности) области. Тогда вес

$$u_{x\hat{x}} = d_{max} - \frac{1}{2}(D(x) + D(\hat{x})),$$

где $d_{max} = \max\limits_{x \in \mathbf{P}} d(x)$. Определим однородную область вокруг $x$ как

$$B = \{y \in \mathbf{P} \mid |I(y) - I(\hat{y})| \leq threshold, \forall (y, \hat{y}) \in N\}.$$

Тогда $D(x) = \max\limits_{\hat{x} \in B} dist(x, \hat{x})$, где расстояние, например, евклидово.

Практика, однако, показала, что лучше выбирать веса не $v_{x\hat{x}}$ или $u_{x\hat{x}}$, а их комбинацию

$$c_{x\hat{x}} = kv_{x\hat{x}} + u_{x\hat{x}},$$

где $k = \max\limits_{(x,\hat{x}) \in N'} u_{x\hat{x}}$. Минимальное остовое дерево, построенное с учетом этих весов — MIDDT.

# 5. Алгоритмы на основе случайных полей Маркова

## 5.1. Построение модели

Один из самых популярных способов моделирования задачи стереозрения — использования случайного поля Маркова (MRF). Обычно его представляют графом, где вершины (случайные переменные) соответствуют пикселям левого изображения, а ребра графически изображают зависимости между переменными. Такое моделирование наиболее популярно в задачах обработки изображений. Иногда, если мы сумели сегментировать изображение, используется модель: вершины — сегменты, ребра — отношение соседства. Впрочем, к нашей задаче это неприменимо. Важно, что зависимости между переменными симметричны (чего нет в байесовских сетях доверия). То есть, если случайным переменным $X$ и $Y$ соответствуют соседние вершины графа, то $P(X|Y) = P(Y|X)$.

Основное свойство случайных полей Маркова — каждая переменная зависит только своих соседей. Это свойство называется локальным свойством Маркова. По теореме Хаммерсли-Клиффорда локальное свойство Маркова эквивалентно тому, что случайный вектор из всех переменных-вершин имеет распределение:

$$P(X) = \frac{1}{Z} \prod_{C \in cl(G)} \varphi_C(X_C),$$

где $G$ — рассматриваемый граф, $cl(G)$ — множество всех клик (clique) $G$, то есть максимальных по включению полных подграфов $G$, $X_C$ — случайные переменные, соответствующие вершинам клики $C$, $\varphi_C$ — совместная вероятность переменных клики $C$, $Z$ — нормализующая константа. Часто вместо $\varphi_C$ удобнее рассматривать величину $V_C = -ln\ \varphi_C$. Она носит название потенциала клики.

Поскольку рассматриваемая нами модель — решетка, клики в таком графе будут иметь ровно 2 вершины. Таким образом, распределение $X$ выглядит так:

$$P(X) = \frac{1}{Z} \prod_{(p,q)\in N} e^{-V_{pq}(X_p, X_q)},$$

где $N$ — множество соседних ребер графа, $V_{pq}$ — все тот же потенциал клики из вершин $p$ и $q$.

Мы ищем такую конфигурацию $X$, при которой апостериорная вероятность $P(X)$ была бы максимальна. Таким образом, задачу стереозрения мы переформулировали полностью в терминах MRF. Для ее решения можно применять какой-либо из известных алгоритмов на случайных полях Маркова.

Иногда от формулировки задачи в виде максимизации вероятности удобнее перейти к формулировке в виде выражения для энергии. Энергией нашей модели в конфигурации $X = f$ назовем в этом случае

$$E(f) = -ln\ P(f) - ln\ Z = \sum_{(p,q)\in N} V_{pq}(f_p, f_q).$$

Тогда максимизация вероятности $P$ эквивалентна минимизации энергии $E$.

В нашей модели множество значений каждой случайной переменной $X_p$ — это множество расхождений (плюс, возможно, метка “заслонен”, обозначим ее *occ*). Отметим, что в этом случае можно не требовать ректификации изображений и работать с двумерными расхождениями.

Чаще всего наша конфигурация $X$ не наблюдается. Поэтому помимо скрытого вектора $X$ мы рассматриваем наблюдаемый случайный вектор $Y$, где $Y_p = I(p)$. Усложним $G$, добавив к вершинам, соответствующим $X$, вершины, соответствие $Y$. При этом $X_p$ и $Y_p$

соединим ребром для всех $p$ (это будет означать, что расхождение в данной точке зависит только от расхождений соседних пикселей и от его интенсивности). Зависимости между переменными $Y_p$ нам не понадобятся, поскольку это наблюдаемые переменные. Поэтому не соединяем их ребрами. Тогда клики разобьются на два типа: из соседних вершин $X_p$ и $X_q$; и вершин $X_p$ и $Y_p$. Тогда распределение будет равно

$$P(XY) = \frac{1}{Z} \prod_{C \in cl(G)} \varphi_C(X_C) = \frac{1}{Z} \prod_{(p,q) \in N} \varphi_{pq}(X_p, X_q) \prod_{p \in \mathbf{P}} \varphi_p(X_p, Y_p) =$$

$$= \frac{1}{Z} \prod_{(p,q) \in N} e^{-V_{pq}(X_p, X_q)} \prod_{p \in \mathbf{P}} e^{-W_p(X_p, Y_p)},$$

где $\mathbf{P}$ — множество всех пикселей левого изображения. Наша цель — максимизация $P(X|Y) = \frac{P(XY)}{P(Y)}$. Поскольку $P(Y)$ — константа, достаточно максимизировать $P(XY)$. В терминах энергии (без учета константы $ln\ P(Y)$, не влияющей на оптимизацию) это перепишется как

$$E(f) = \sum_{(p,q) \in N} V_{pq}(f_p, f_q) + \sum_{p \in \mathbf{P}} W_p(f_p, I(p)).$$

### 5.2. Алгоритм разреза графа

Выбор функций-потенциала $V_{pq}$ и $W_p$ (что равносильно назначению условной вероятности для связи между переменными) может быть относительно произвольным. Достаточно, чтобы он был неотрицательным и симметричным по аргументам. Один из самых встречаемых вариантов — модель Поттса — использован в [8,9]. Изначально она известна как модель, описывающая взаимодействие спинов на кристаллической решетке:

$$V_{pq}(f_p, f_q) = v_{pq} \cdot (1 - \delta(f_p - f_q)),$$

где $\delta$ — символ Кронекера, $v_{pq}$ — штраф за нарушении гладкости функции расхождения.

Для $W_q$ в [8] используется $W_p(f_p, I(p)) = (I(p) - I'(p + f_p))^2$, если $f_p \neq occ$, $W_p(f_p, I(p)) = S_p$ — штраф за заслоненный пиксель,

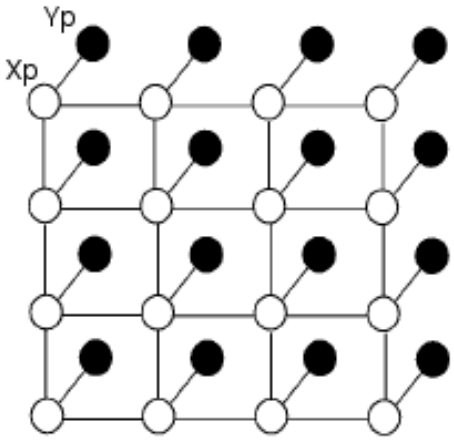


Рис. 6: Случайное поле Маркова со скрытыми (белые) и наблюдаемыми (черные) переменными.

в принципе может зависеть от $p$, иначе. В [9,14] используется модуль разности вместо квадрата разности.

В терминах общей задачи минимизации

$$E_{data}(f) = \sum_{p \in \mathbf{P}, f_p \neq occ} W_p(f_p, I(p)),$$

$$E_{occ}(f) = \sum_{p \in \mathbf{P}, f_p = occ} S_p,$$

$$E_{smooth}(f) = \sum_{(p,q) \in N} V_{pq}(f_p, f_q).$$

Таким образом задача часто записывается в виде минимизации

$$E(f) = E_{data}(f) + E_{smooth}(f) + E_{occ}(f).$$

Минимизация энергии с помощью разреза графа ( [8,9]) основывается на базовых понятиях: $\alpha - expansion$ и $\alpha\beta - swap$. Допустим есть некоторое конечное множество меток $L$. Есть некоторое множество пикселей $\mathbf{P}$. Каждому пикселю $p \in \mathbf{P}$ мы присваиваем некоторую метку $l_p \in L$. Обозначим $P_l = \{p \in \mathbf{P} | l_p = l\}, f = \{P_l | l \in L\}$

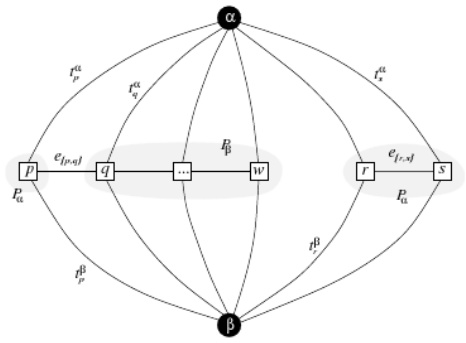


Рис. 7: Граф энергии для $\alpha\beta - swap$. Вершины — пиксели из $P_\alpha \cup P_\beta$. Ребра — слагаемые формулы энергии. Ребра между вершинами-пикселями — слагаемые, зависящие только от 2-х пикселей. Ребра между выделенными вершинами и вершинами-пикселями - слагаемые, зависящие от вершины и ее метки. Мы хотим потерять как можно меньше энергии вследствие разреза. Для $\alpha - expansion$ строится похожий граф с выделенными вершинами $\alpha$ и не-$\alpha$, но конструкция там чуть сложнее [9].

— текущая конфигурация (в нашем случае метки могут быть расхождениями [8]).

Пусть даны метки $\alpha, \beta \in L$. Переход к конфигурации $f'$ называется $\alpha\beta - swap$, если $P_l = P'_l$ для всех меток $l \neq \alpha, \beta$. Пусть дана метка $\alpha \in L$. Переход к конфигурации $f'$ называется $\alpha - expansion$, если $P_\alpha \subset P'_\alpha$ и $P'_l \subset P_l$ для любых $l \neq \alpha$.

Соответственно обозначим $\alpha\beta - swap(f)(\alpha - expansion(f))$ — множество конфигураций, в которые мы можем перейти за один $\alpha\beta - swap(\alpha - expansion)$. Алгоритм минимизации энергии с использованием $\alpha\beta - swap(f)$ состоит из последовательности уменьшений функционала энергии. На каждом шаге выполняется выбир одной операции $\alpha\beta - swap(f)$, которая уменьшает энергию больше, чем другие.

1. Начинаем с произвольной конфигурации f
2. success := false
3. Для каждой пары меток $\{\alpha, \beta\} \subset L$

- $\hat{f} := \underset{f' \in \alpha\beta - swap(f)}{\arg\max} E(f')$
- Если $E(\hat{f}) < E(f)$, присваиваем $f := \hat{f}$ и success := true

4. Если success = 1 goto 2
5. Возвращаем конфигурацию f

Алгоритм минимизации энергии с использованием $\alpha - expansion$ очень похож на предыдущий с той разницей, что конфигурация меняется другой операцией.

1. Начинаем с произвольной конфигурации f
2. success := 0
3. Для каждой метки $\alpha \in L$
   - $\hat{f} := \underset{f' \in \alpha - expansion(f)}{\arg\max} E(f')$
   - Если $E(\hat{f}) < E(f)$, set $f := \hat{f}$ и success := 1
4. Если success = 1 goto 2
5. Возвращаем конфигурацию f

Минимизация на каждом шаге происходит собственно с помощью минимального разреза графа (рис. 7). Разрезу соответствует новое присваивание меток $\alpha$ и $\beta$ ($\alpha$ и не-$\alpha$ – в случае $\alpha - expansion$), то есть новая конфигурация.

Алгоритм находит локальный минимум функции энергии. Существует алгоритм основанный на Multiway Graph Cut ( [14]), который ищет глобальный минимум, но задача Multiway Graph Cut относится к классу NP-полных, поэтому он не рассматривается как неэффективный.

В работе [8] отмечается, что алгоритм на основе $\alpha - expansion$ показывает лучшие результаты, нежели на основе $\alpha\beta - swap$.

### 5.3. Алгоритм распространения доверия

Алгоритм распространения доверия (belief propagation) — стандартный способ решения задачи максимизации апостериорной вероятности (в байесовских сетях доверия и MRF) при условии отсутствия циклов.

В модели, основанной на случайных полях Маркова, каждый пиксель левого изображения — это случайная переменная, и ему сопоставляется узел MRF. В качестве соседних узлов обычно рассматриваются 4 соседних пикселя.

Таким образом, в нашей модели огромное количество циклов. Поэтому в [10] было предложено использовать модификацию Loopy Belief Propagation, который отличается от простого алгоритма распространения доверия только тем, что игнорирует наличие циклов и выполняется не за один проход, а итертивно, до сходимости. В случае произвольных графов Loopy Belief Propagation может сходиться не к максимально вероятной конфигурации, хуже того, он может вообще не сходиться. В случае сходимости, он сходится к стационарной точке энергии Бете ( [15]), которая является аппроксимацией $E(f)$ и совпадает с ней в случае отсутствия циклов. Более детально условия сходимости этого алгоритма разобраны в [16].

Алгоритм распространяет по сети некие величины, называемые сообщениями. Например, $m_{pq}(f_p, f_q)$ — сообщение, отправляемое вершиной $X_p$ вершине $X_q$ — насколько переменная $X_p$ "думает", что $X_q$ в соответствующем состоянии.

- Инициируем все сообщения одинаковым распределением.
- Обновляем сообщения некоторое (большое) количество раз

$$m_{pq}^{i+1}(f_q) \leftarrow \kappa \max_{f_p} \varphi_{pq}(f_p, f_q)\varphi_p^i(f_p) \prod_{f_k \in N(f_p)-q} m_{kp}^i(f_p).$$

- Вычисляем свидетельства

$$b_p(f_p) \leftarrow \kappa\, \varphi_p(f_p) \prod_{f_k \in N(f_p)} m_{kp}(f_p).$$

- По свидетельствам вычисляем наиболее вероятную конфигурацию
$$f_p^{MAP} = \arg\max_{f_k} b_p(f_k).$$

За подробным описанием и объяснением принципа работы алгоритма следует обратиться к [15, 16].

## 6. Заключение

Мы изложили основные шаги некоторых типичных алгоритмов стереозрения. Несмотря на качество, их производительность оставляет желать лучшего (в [4] несколько секунд на строчку на изображение 512x512, около минуты на изображение 384x288 в [8]). Впрочем, одномерные алгоритмы обладают возможностью быть распараллеленными (в идеале: по строке на вычислительное устройство), что очень хорошо в условиях наиболее популярного метода наращивания мощностей вычислительной техники.

Качество работы алгоритмов можно улучшить, если известна информация о сегментации изображения. Так соседним пикселям из одного сегмента, скорее всего, должны быть присвоены похожие расхождения. Например, в методе [10] использование информации помогает сократить число ошибок до 40%.

Низкая производительность не позволяет использовать в лоб алгоритмы стереозрения для получения дальностных данных из видеоряда. Интересное направление исследования — изучение возможностей использования связки из алгоритма сопоставления изображений и детектора движущихся объектов (кстати, в качестве детектора движения можно использовать те же алгоритмы на разрезах графа, как отмечается в [8]). Также возможно использование какого-нибудь алгоритма предсказания нахождения объекта в следующий момент. Во-первых, это сократит область изображения для вычисления новых расхождений (вычисляем новые расхождения только на движущихся объектах). Во-вторых, алгоритмы на основе разреза графа и распространения доверия итеративны по своей природе, и улучшают текущую конфигурацию. Таким образом, возможно, им на вход можно подавать предыдущую конфигурацию или предсказание конфигурации, что должно положительно сказаться на производительности.

**Список литературы**


[1] *Ott M., Lewis J. P., Cox I.* Teleconferencing eye contact using a virtual camera // Proc. Conf. "Human Factors in Computing Systems". 1993. PP. 119–110.

[2] *Luping A., Yunde J., Mingtao P., Hongbin D.* Precise shape measurement of dynamic surface via single camera stereo // International Journal of Information Acquisition. 3(1). 2006. PP. 1–8.

[3] *Scharstein D., Szeliski R.* A taxonomy and evaluation of dense two-frame stereo correspondence algorithms // Int. Journal of Computer Vision 47. April-June 2002. PP. 7–42.

[4] *Bobick A. F., Intille S. S.* Large occlusion stereo // Int. Journal of Computer Vision. 33(3). 1999. PP. 181–200.

[5] *Ohta Y., Kanade T.* Stereo by intra- and inter- scanline search using dynamic programming // IEEE TPAMI. 7(2). 1985. PP. 139–154.

[6] *Veksler O.* Stereo correspondence by dynamic programming on a tree // Proc. CVPR. Vol. 2. 2005. PP. 384–390.

[7] *Naveed I. R., Huijun Di, GuangYou Xu* Refine stereo correspondence using bayesian network and dynamic programming on a color based minimal span tree // ACIVS. 2006. PP. 610–619.

[8] *Kolmogorov V., Zabih R.* Computing visual correspondence with occlusions using graph cuts // ICCV. Vol. 2. 2001. PP. 508–515.

[9] *Boykov Y., Veksler O., Zabih R.* Fast approximate energy minimization via graph cuts // IEEE TPAMI 23(11). 2001. PP. 1222–1239.

[10] *Sun J., Shum H., Zheng .N* Stereo matching using belief propagation // In ECCV. 2002. PP. 510–524.

[11] *Collins R. T.* A space-sweep approach to true multi-image matching // Proc. CVPR. 1996. PP. 358–363.

[12] *Халидов В., Forbes F., Hansard M., Arnaud E., Horaud R.* Обнаружение и локализация объектов в пространстве посредством самообучающейся модели кластеризации // Стохастическая оптимизация в информатике. СПб.: Изд-во СПбГУ. Вып. 4. 2008. С. 211–235.

[13] *Форсайт Д. А., Понс Ж.* Компьютерное зрение: современный подход. – М.: Вильямс. 2004. 928 с.

[14] *Boykov Y., Veksler O., Zabih R.* Markov random fields with efficient approximations // Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition. June 23-25 1998. PP. 648.

[15] *Yedidia J.S., Freeman W. T., Weiss Y.* Understanding belief propagation and its generalizations // Exploring artificial intelligence in the new millennium. Morgan Kaufmann Publishers Inc. San Francisco, CA. 2003. PP. 239–269.

[16] *Crick C., Pfeffer A.* Loopy belief propagation as a basis for communication in sensor networks // The Journal of Machine Learning Research. 6. Dec. 2005. PP. 905–936.