Параллельная реализация алгоритма итеративного распространения доверия в задаче сопоставления изображений

Д.А. Дворников[a], Е.В. Гошин[a]

[a] *Самарский национальный исследовательский университет имени академика С.П. Королёва, 443086, Московское шоссе, 34, Самара, Россия*

Аннотация

Важным этапом в обработке изображений является их сопоставление. В ходе сопоставления производится поиск объектов, которые присутствуют на обоих изображениях. Из-за различного положения камеры при фиксации изображений, соответствующие фрагменты изображений могут отличаться, вследствие чего результат сопоставления может содержать ошибки. Данная статья посвящена алгоритму сопоставления, основанному на использовании случайных Марковских полей. Предложены два подхода к увеличению производительности итеративного алгоритма распространения доверия: параллельная реализация и использование пирамиды изображений. Приведено сравнение результатов сопоставления по точности и показатели ускорения.

Ключевые слова: сопоставление изображений; компьютерное зрение; параллельные вычисления; пирамида изображений; случайные поля Маркова; диспаратность

1. Введение

Одним из направлений машинного зрения [1] является стереозрение [2]. Его применение позволяет получить представление о глубине изображения, расстоянии до объектов и составлении трехмерной картины окружающего мира. Для этого требуется как минимум два изображения, взятые с разных точек обзора.

Из эффекта параллакса, известно, что объекты, расположенные близко к точке обзора, будут двигаться быстрее при движении, чем объекты, расположенные далеко от точки обзора. Так же и на изображениях, предмет расположенный ближе к нам, будет смещён в пикселях сильнее. По смещению в пикселах одного и того же объекта на двух изображениях, можно рассчитать расстояние от точки обзора до этого объекта и получить трёхмерную координату. Смещения часто называют маркировками/метками. При визуализации карты сдвигов получается чёрно-белое изображение, где каждый сдвиг является меткой, которая определяет яркость соответствующего пикселя на изображении. Пример представлен на рис. 1.

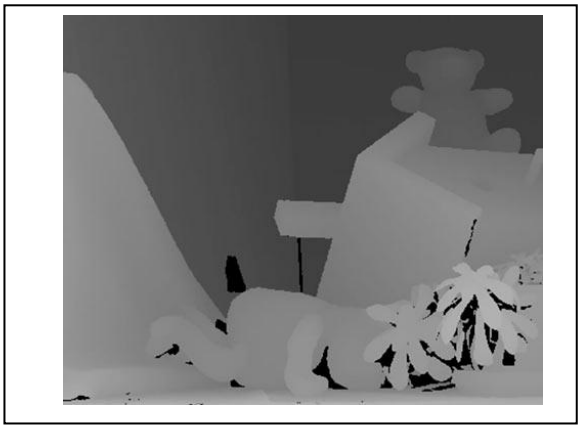

Рис. 1. Визуализация сдвигов пикселей.

Здесь видно, объекты находящиеся ближе к нам – более светлые, так как сдвиг пикселов у ближних объектов сильнее. Основная задача стереозрения состоит в том, чтобы правильно детектировать соответствующие пикселы на обоих изображениях.

2. Случайные поля Маркова

Один из самых популярных способов моделирования задачи стереозрения — использование случайного поля Маркова (MRF) [3,4]. MRF является ненаправленной графической моделью, которая может быть определена пространственными зависимостями. Как и все графические модели, она состоит из узлов и связей. Пример модели MRF размерностью 4x4 представлен на рис. 2.

Здесь, чёрные узлы являются наблюдаемыми узлами, которые в нашем случае являются значениями интенсивности пикселей. Белые узлы являются скрытыми узлами, которые представляют собой значения сдвигов пикселей, которые требуется найти. Связи между узлами являются зависимостями между пикселями. Каждая переменная зависит только

от своих соседей, это является основным свойством случайных полей Маркова и называется локальным свойством Маркова.

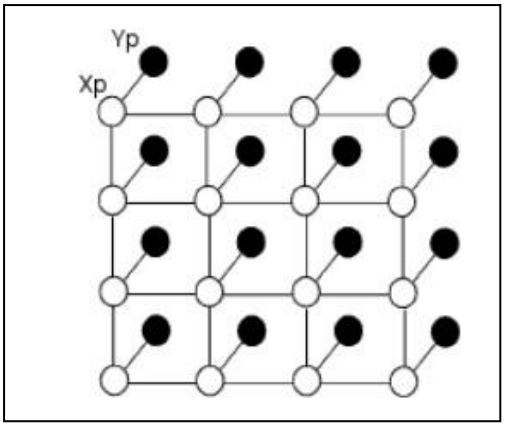

Рис. 2. Модель MRF изображения состоящего из 4x4 пикселя.

Задача стереозрения с точки зрения MRF формулируется в виде следующей функции энергии:

$$E(Y,X) = \sum_{i} DataCost(y_i, x_i) + \sum_{j=neighbours\ of\ i} SmothnessCost(x_i, x_j),$$

где X и Y являются положением наблюдаемого пикселя и скрытого узла. Функция используется для подведения итогов сопоставления и вычисляет энергию в данном узле. Суть состоит в том, чтобы получить минимальную сумму энергий всех пикселей.

Функция DataCost, возвращает вес присвоения значения метки. Рассчитывается вес следующим образом: берётся блок пикселей определённого размера вокруг обозреваемого пикселя на одном изображении и блок пикселей со сдвигом равным значению метки на другом и вычисляется среднее разности энергий блоков пикселей. Таким образом, чем выше схожесть блоков, тем значение энергии меньше.

Функция SmoothnessCost, обеспечивает соблюдение гладкой маркировки соседних скрытых узлов. Для этого берётся функция, которая “штрафует” соседние, сильно отличающиеся друг от друга метки. Обычно, используется одна из трёх функций сглаживания, представленных на рис. 3.

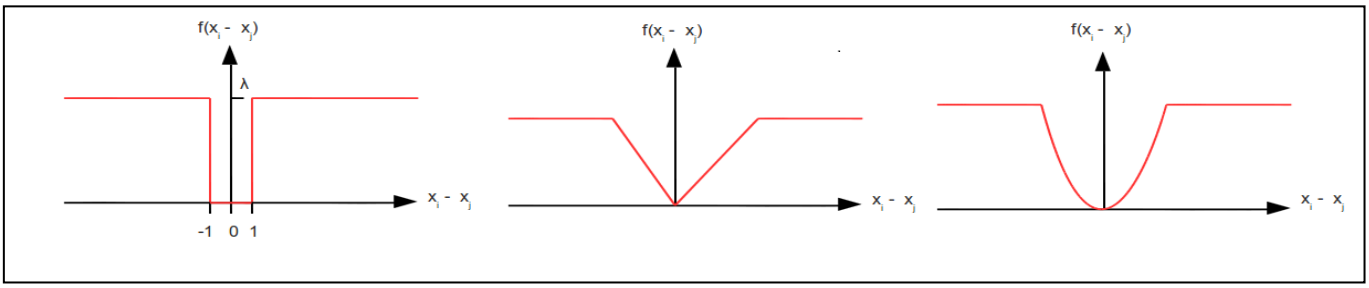

Рис. 3. а) бинарная функция; б) линейная функция; в) квадратичная функция.

Выше представлены бинарное, линейное и квадратичное сглаживание. В первом случае если разность соседних маркировок узлов выше определённого порога, то возвращается “штраф” в размере α, иначе 0.

Во втором случае α домножается на значение разности маркировок, а в третьем на квадрат разности.

Функция расчёта энергии позволяет узнать количество энергии изображения при определённых сдвигах каждого пикселя, но для нахождения таких сдвигов, при которых энергия будет минимальна, требуется алгоритм, который будет сходиться. Стандартным способом решения данной задачи является применение алгоритма распространения доверия [3].

3. Алгоритм распространения доверия

Алгоритм распространения доверия позволяет найти приближённое решение MRF. В процессе выполнения, алгоритм распространяет по сети наборы значений, называемые сообщениями. Например, сообщение, отправляемое вершиной A вершине B — содержит информацию насколько переменная A “думает”, что B в соответствующем состоянии. Если переменная может находиться в одном из n состояний, то в сообщении будет находиться n значений, первое значение говорит с какой вероятностью у B метка 1, второе с какой вероятностью метка 2 и т.д.

Итак, у каждого скрытого узла имеется 5 сообщений, первые 4 от соседних узлов, которые предполагают с какими вероятностями обозреваемый узел может иметь ту или иную метку и пятое сообщение, в котором хранится рассчитанные значения DataCost.

Первым шагом алгоритма является инициализация всех сообщений одинаковыми сообщениями. Далее запускается итеративная функция, которая заканчивает свою работу, когда суммарная энергия пикселей достигнет нижнего определённого порога. За одну итерацию производится рассылка сообщений во всех направлениях соседним пикселям, пересчёт собственных сообщений (вследствие чего изменяется состояние системы) и производится расчёт суммы энергий всех пикселей. Пересылки сообщений могут идти в любом порядке, например, сначала можно пересылать сообщения слева-направо, потом справа-налево, потом сверху-вниз и наконец снизу-вверх. После завершения выполнения итеративной функции с пересылками сообщений, на выходе формируется результат сопоставления. Данный алгоритм обладает хорошей сходимостью к рассчитываемому результату, однако даже одна итерация занимает довольно продолжительное время. Для увеличения производимости исходного алгоритма рассмотрим два способа его модернизации.

4. Пирамида изображений

Во время работы алгоритма сопоставления, выполняется довольно много операций даже на одной итерации. Для каждого пикселя вычисляется по 4 сообщения, каждое из них хранит в себе по n значений, которое равно количеству возможных меток, а в конце итерации рассчитываются значения всех меток пикселей. Если взять изображение размерностью 100x100 пикселей, количество возможных меток у каждого пикселя равным 16, то получим $4 \cdot 16 \cdot 100 \cdot 100 + 100 \cdot 100 = 650000$ операций на одной итерации. На практике изображения на порядок больше.

Для уменьшения числа выполняемых операций в алгоритме, было предложено использовать изображения с различными масштабами, или другими словами пирамиду изображений [5]. Пирамидой изображений называется изображение, представленное в нескольких масштабах. Пример такого изображения представлен на рис. 4.

Рис. 4. Пирамида изображений, состоящая из трёх уровней.

Суть данного подхода состоит в том, чтобы сначала сопоставить несколько изображений уменьшенных в несколько раз, а в конце произвести сопоставление исходных изображений, причем на каждом следующем масштабе учитывать метки пикселей предыдущего масштаба.

В стандартном алгоритме каждый пиксел характеризовался четырьмя сообщениями с вероятностями выставления меток, набором DataCost для каждой метки и самой меткой. В модернизированном алгоритме каждый пиксел изображения будет характеризоваться ещё одним параметром, который будет являться значением метки пикселя на предыдущем масштабе и умноженным вдвое. Метка характеризует сдвиг между соответствующими пикселями и при увеличении размерности изображения вдвое – сдвиг тоже будет увеличиваться вдвое.

Как известно, при увеличении изображения в 2 раза, один пиксел исходного изображения будет представлен четырьмя пикселями увеличенного изображения, поэтому метка каждого пикселя исходного изображения будет распределяться по четырём пикселям увеличенного изображения. Пример такого соответствия представлен на рис. 5.

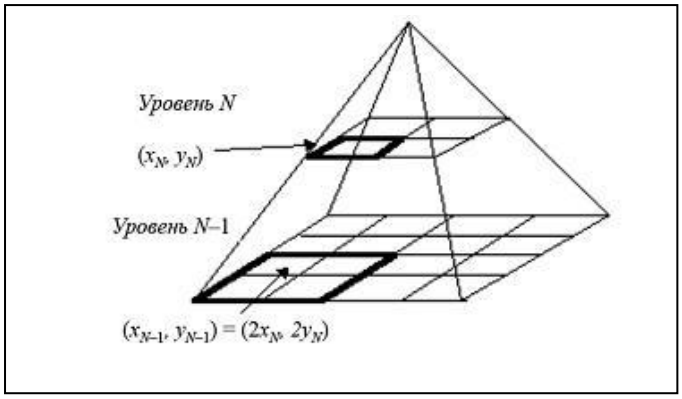

Рис. 5. Соответствие пикселей на соседних уровнях масштабирования.

Ещё одним важным моментом является то, что теперь метка каждого пикселя изображения будет характеризоваться значением суммы меток текущего уровня и предыдущего. Это означает то, что на верхнем уровне пирамиды (когда размеры изображений минимальны) будет рассчитываться лучшая метка каждого пикселя, а на всех последующих уточняться.

Например, возьмём количество меток равным 8 и рассмотрим значение метки одного пикселя на всех уровнях. На минимальном уровне будет рассчитана лучшая метка, далее при переходе на следующий масштаб значение метки запишется в поле, предназначенное для хранения значения предыдущей метки и умножится на 2, так как размерность пиксельной сетки увеличилось в два раза. Если метка была равна 6, то она станет равна 12. Теперь на следующем масштабе будет уже рассчитываться не лучшее значение метки, а значение сдвига относительно предыдущей метки, то есть будет производиться уточнение предыдущей метки. И так далее на последующих уровнях. Таким образом при количестве возможных меток равном 8 у пикселя, мы можем найти сдвиг больше 8, при этом не увеличивая количество рассчитываемых вероятностей меток в сообщениях.

Прирост производительности выражается в уменьшении числа рассчитываемых вероятностей в сообщениях и уменьшении числа обрабатываемых пикселей на изображениях с меньшим масштабом.

5. Параллельность

Вторым подходом ускорения алгоритма, является его распараллеливание. При выполнении прогонки алгоритма по какому-либо из направлений (влево, вправо, вверх, вниз), операции, производимые в разных строках(столбцах) изображений, являются независимыми. Поэтому прогонку в какое-либо направление можно вычислять параллельно в n потоков.

Возьмём следующую последовательность пересылок: влево, вправо, вверх и вниз. Визуализация таких пересылок представлена на рис. 6.

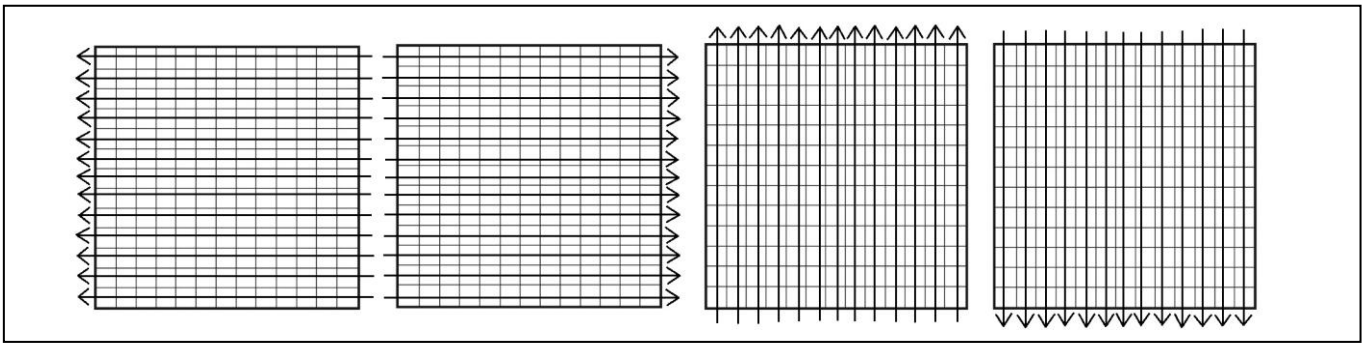

Рис. 6. Визуальное представление прогонок по направлениям (влево, вправо, вверх, вниз).

Так как алгоритм можно распараллелить на n потоков, то при вычислении сообщений, строки(столбцы) изображений будем мысленно делить на n примерно равных частей. Пример разделения на 3 потока представлен на рис. 7.

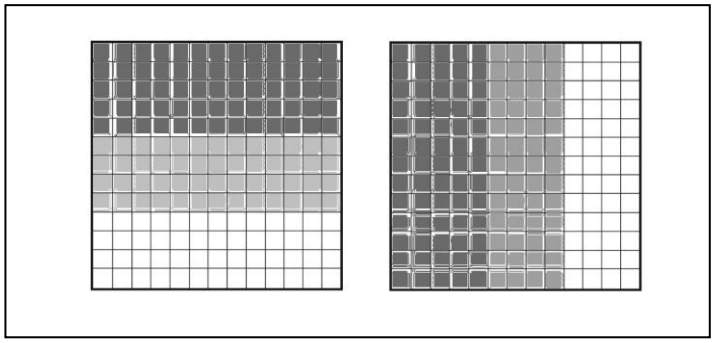

Рис. 7. Распределение данных пикселей по процессорам.

Сообщения, соответствующие тёмно-серым ячейкам, будут рассчитываться на первом процессоре, соответствующие светло-серым ячейкам на втором процессоре, а остальные на третьем.

Сначала прогонка производится по строкам влево. Так как у каждого процессора имеется своя локальная память, то в неё необходимо загрузить начальные данные сообщений, которые уже имеются в сообщениях каждого пикселя. То-есть если процессор выполняет прогонку по 10 пикселям, то в память необходимо загрузить состояния сообщений этих 10 пикселей.

После инициализации сообщений каждого процессора и выполнении прогонки влево, сразу выполняем прогонку вправо, так как, как уже было сказано ранее, вычисления в прогонках по строкам (столбцам) не зависят друг от друга, и все необходимая информация уже имеется в сообщениях.

После прогонки вправо, необходимо собрать все рассчитанные данные на главном процессоре и разослать снова таким образом, чтобы процессоры были готовы выполнить прогонки вверх и вниз.

После выполнения такой перегруппировки данных выполняются прогонки вверх и вниз, причём после прогонки вверх, коммуникаций между процессорами не требуется. В конце снова собираем все рассчитанные данные на главном процессоре, прогонка по всем направлениям завершена.

Таким образом, в рассмотренной i-ой итерации было произведено 4 рассылки данных при прогонках по всем направлениям.

6. Экспериментальная часть

В ходе модернизирования алгоритма сопоставления изображений путём внедрения пирамиды изображений, была получена его программная реализация.

В данном разделе рассмотрим результаты сопоставления изображений, полученные из модернизированного алгоритма и сравним их с результатами исходного.

Высота пирамиды при проведении экспериментов была равна 2, то есть сопоставление производилось на основе исходных изображений и их копий, уменьшенных вдвое. В стандартном алгоритме выполнялось n итераций для вычисления карты сдвигов. В модернизированном n итераций по изображениям, уменьшенным в двое и n итераций по исходным изображениям. В качестве оценки точности сопоставления были взяты энергии сопоставленных изображений в конце выполнения алгоритма. Результаты представлены в таблице 1.

Таблица 1. Результаты выполнения, исходного и модернизированного алгоритмов

Алгоритм	Количество итераций	Время выполнения (мс)	Энергия
Стандартный	5	29888	432170
Стандартный	10	55573	424086
Стандартный	15	71399	419337
Модернизированный	5 и 5	9583	423853
Модернизированный	10 и 10	17903	395708
Модернизированный	15 и 15	26008	389556

Изображения, являющиеся результатами сопоставлений представлены на рис. 8, 9 и 10.

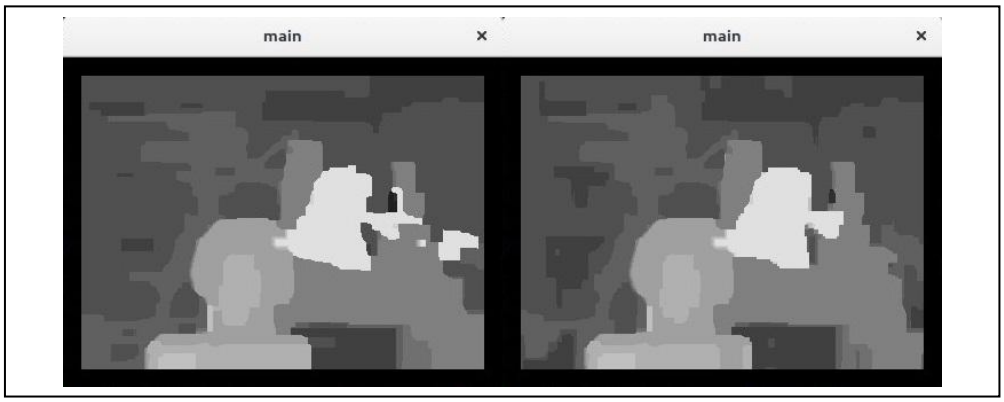

Рис. 8. Результаты сопоставления изображений после 5 итераций, а) стандартный алгоритм; б) модернизированный алгоритм.

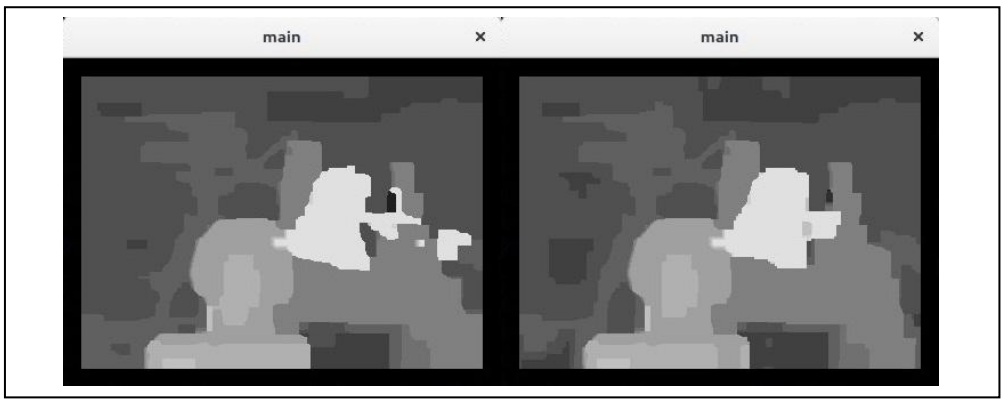

Рис. 9. Результаты сопоставления изображений после 10 итераций, а) стандартный алгоритм; б) модернизированный алгоритм.

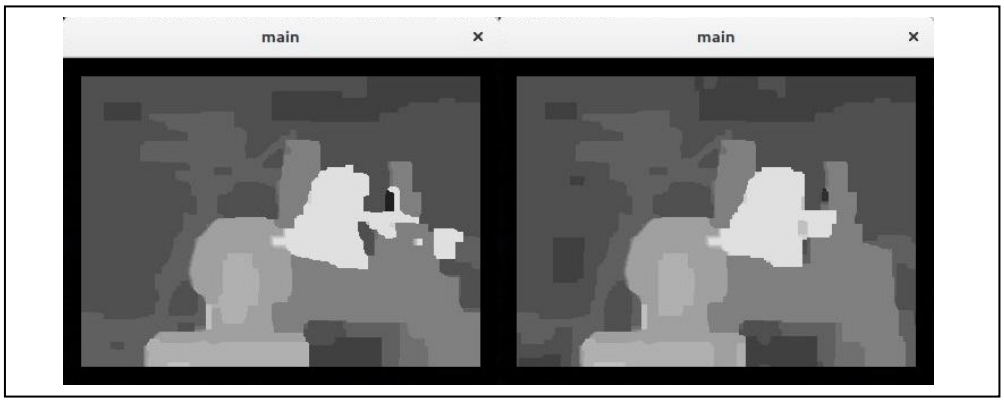

Рис. 10. Результаты сопоставления изображений после 15 итераций, а) стандартный алгоритм; б) модернизированный алгоритм.

Исходя из таблицы 1, модернизированный алгоритм даёт более точные результаты и выполняется быстрее примерно в 3 раза. Стоит отметить, что результаты сопоставления изображений являются приемлемыми уже после 5 итерации как видно на рис. 8, что подтверждает быструю сходимость алгоритма.

7. Заключение

В данной статье было рассмотрено 2 подхода увеличения производительности алгоритма сопоставления изображений, основанного на случайных Марковских полях. Был реализован модернизированный алгоритм сопоставления изображений, использующий пирамиду изображений при сопоставлении. Эксперименты показали, что полученный алгоритм не уступает исходному в скорости и качестве сопоставления. Так же был рассмотрен вариант распараллеливания алгоритма, а именно распараллеливание прогонки по направлениям.

Литература

[1] Форсайт, Д. Компьютерное зрение. Современный подход / Д. Форсайт, Ж. Понс // М: Издательский дом" Вильямс. – 2004.

[2] Depth Map from Stereo Images [Electronic resource]. Access mode: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_calib3d/py_depthmap/py_depthmap.html (12.10.2016).

[3] Loopy belief propagation, Markov random field, stereo vision [Electronic resource]. — Access mode: http://nghiaho.com/?page_id=1366 (12.10.2016).

[4] Stan, Z. Markov random field modeling in image analysis/ Z. Stan – London: Springer, 2009. – 351 p.

[5] Introduction to SIFT (Scale-Invariant Feature Transform) [Electronic resource]. – Access mode: http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html (7.11.2016).