Санкт-Петербургский государственный университет
Кафедра системного программирования

Бабанов Пётр Андреевич

Реализация методов машинного обучения в задаче стереозрения.

Курсовая работа

Научный руководитель:
Александр Пименов

Санкт-Петербург
2017

Содержание

Введение

В последнее время проявляется большой интерес в области систем автоматического управления различными механизмами, такими как автомобили, автобусы, строительная и карьерная техника. Ведущие IT компании такие как Google и Nvidia, автоконцерны Audi и Tesla представляют свои реализации автопилотов. Одним из ключевых элементов таких систем является механизм стереозрения: алгоритмы, позволяющие по изображениям с камер определять окружающие объекты и расстояния до них.

Все алгоритмы, решающие эту задачу можно разделить на две категории:

1. Локальные алгоритмы.
2. Глобальные алгоритмы.

Локальные алгоритмы ищут соответствующий пиксель для заданного в небольших окрестностях, используя корреляционные окна (окрестности вокруг обрабатываемого пикселя, и такая же область вокруг предполагаемого соответствующего ему пикселя на втором изображении, внутри которой определяется схожесть фрагментов) вокруг них. Такие подходы дают высокую скорость обработки, так как не требуют затратных вычислений и допускают многопоточные расчеты. Отрицательной стороной таких подходов являются низкая точность и ограничение на содержание изображений (например в случае если на изображении есть шахматная доска или полосатый объект то корректность работы алгоритма не гарантируется)

Глобальные алгоритмы основываются на следующей идеи: паре пикселей на левом и правом изображении сопоставляется число которое отображает степень их схожесть (например яркость) в предположении что объекты локально непрерывны. Чем более похожи пиксели тем меньше это число. После этого считается сумма по всему изображению и если рассмотреть несколько вариантов таких сопоставлений то предпочтение отдается варианту с наименьшей суммой. Очевидно что такой алгоритм более сложный с вычислительной точки зрения и требует большего времени. На сегодняшний

день нет реализации такого алгоритма, которая бы могла работать в режиме реального времени.

В 2014 году Jure Žbontar и Yann LeCun в статье [1] предложили алгоритм который можно расположить между этими классами. Они предложили делить изображение на маленькие фрагменты и дальше обрабатывать их с помощью сверточной нейронной сети, которая определяет схожесть фрагментов и распределяет в два класса : «похожие» и «непохожие». После этого брать вероятность класса «непохожие» и использовать как число, аналогично глобальным методам. Такой подход помог добиться лучшего (по состоянию на 2014 год) показателя точности на KITTI dataset — наборе данных, разработанном технологическим институтом города Карлсруе.

Цели работы

Целью данной работы является повторение результатов работы [1] с использованием фреймворка для нейронной нейсети, написанного на C++, а так же исследовать методы параллельных вычислений для ускорения работы алгоритма.

Для достижения этой цели необходимо решить следующие задачи:

1. Провести сравнение существующих фреймворков для создания нейронных сетей, написанных на языке C++.
2. Подготовить набор данных для обучения нейронной сети.
3. Изучить различные виды архитектур нейронных сетей, предназначенных для сравнения изображений.
4. Реализовать нейронную сеть с использованием выбранного фреймворка и провести ее обучение.

Обзор существующих решений

В статье [1] авторы использовали Torch7 и язык Lua. Их реализация при тестировании на KITTI Dataset 2015 показала следующие результаты:

1. 2.89% ошибок;
2. 67 секунд.

При этом авторы утверждают что при более долгом обучении нейронной сети точность всего алгоритма продолжает увеличиваться.

Схожие идеи представлены в работе Сергея Загоруйко и Nikos Komodakis [2]. В ней авторы так же использовали Torch7 однако сведений об испытаниях их реализации на KITTI Dataset найти не удалось.

Реализация

Для реализации был выбран фреймоворк Caffe по нескольким причинам

1. Ориентация на работу с изображениями.
2. Реализация на языке C++.
3. Возможность конфигурации для работы на GPU вместо CPU.
4. Возможность интеграции в приложение с использованием библиотеки OpenCV

Подготовка данных

При подготовке данных для обучения нейронной сети использовался набор данных KITTI 2015. В соответствии с работой [1] были из больших изображений были нарезаны фрагменты размером 9x9 пикселей, для которых заранее известно смещение и таким образом для любого пикселя с известным смещением можно получить две пары изображений для обучения: одну с использованием правильного смещения и вторую, незначительно изменив значение смещения (Рис. 1).

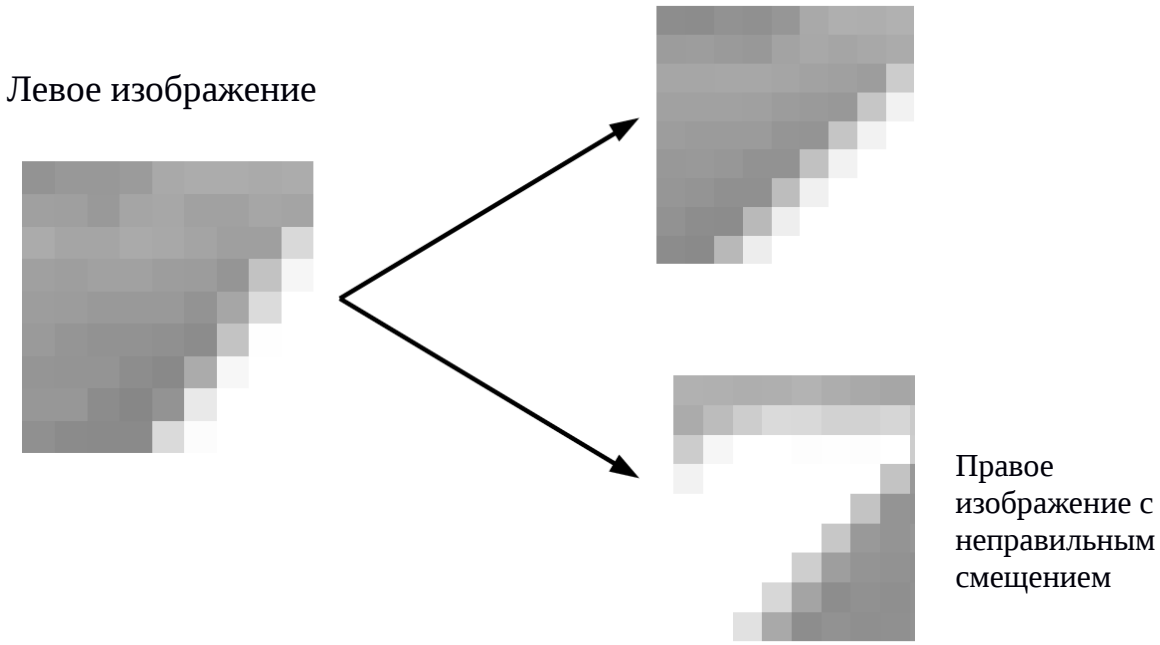

Рис. 1

Так как для соседних пикселей фрагменты будут похожими, то данные брались из непересекающихся областей. При нарезке использовалась технология CUDA с видеокартой GTX860m со следующими характеристиками:

- Тактовая частота 1029МГц
- Число потоков 640
- Объем памяти 2 Гб

В результате был сгенерирован набор данных из 699760 пар изображений.

Архитектура сети

В рамках работы были рассмотрены два типа архитектур, первая аналогична представленной в [3] (рис. 2), вторая взята из документации фреймворка caffe и используется там для сравнения изображений и идентификации их принадлежности к одному классу (рис. 3)

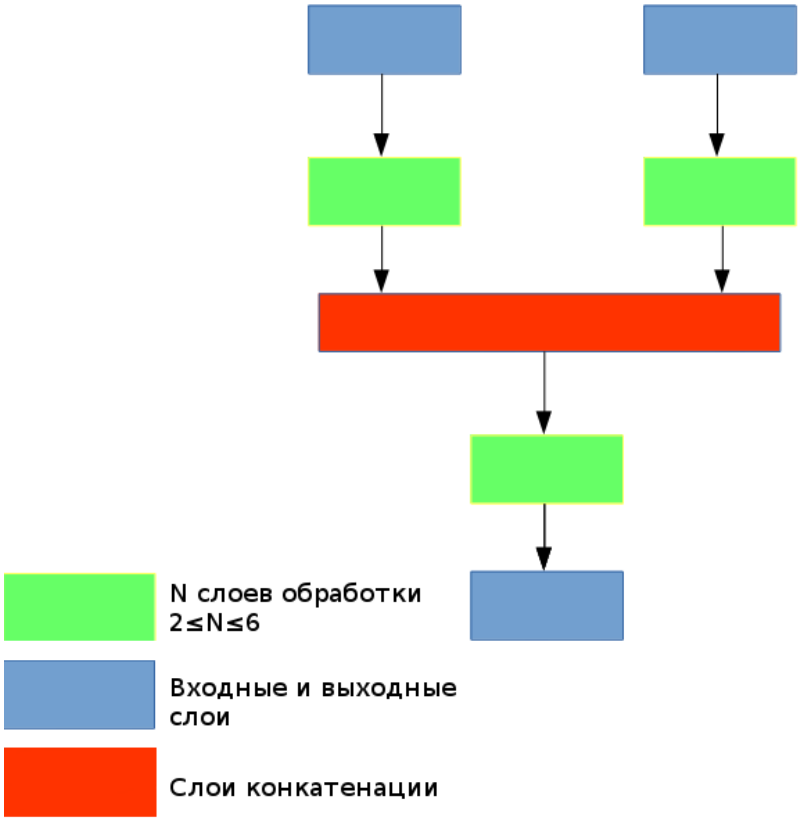
N слоев обработки
2≤N≤6
Входные и выходные
слои
Слои конкатенации

Рис. 2

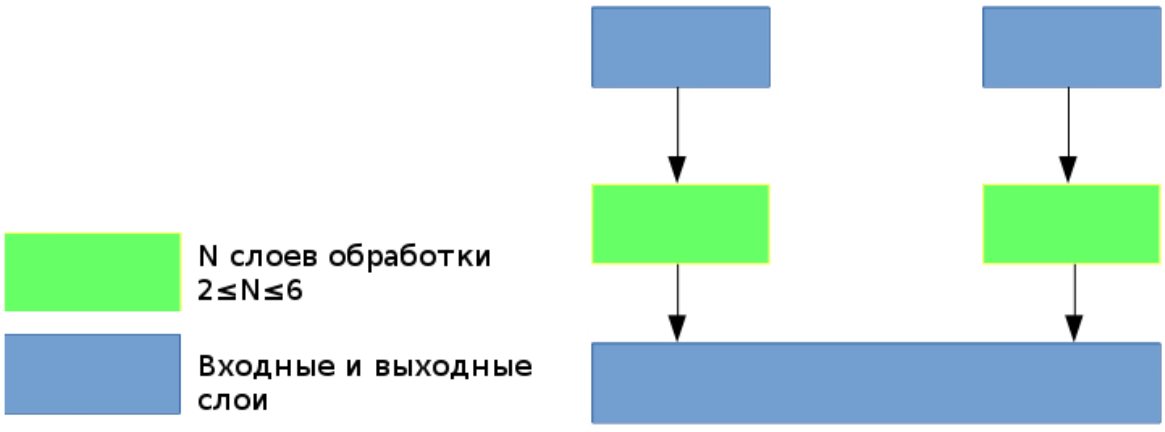
N слоев обработки
2≤N≤6
Входные и выходные
слои

Рис. 3

На вход нейронной сети подается пара фрагментов из набора данных, после чего каждое изображение из пары проходит обработку: применение матриц свертки. При этом применяется несколько ядер свертки — различных матриц, каждая из которых отвечает за нахождение своего признака на изображении (как правило на первом этапе это геометрические примитивы, а на следующих слоях — различные их комбинации). На выходе такого слоя образуется трехмерный массив размерности K×(N-M)×(N-M) в случае если матрица свертки применяется к исходному изображению с шагом 1, где K — количество ядер, N — размер изображения, M — размер матрицы свертки. В случае если шаг будет больше — то очевидно что вторые и третьи размерности размера надо разделить на размер шага.

После этого применяется слой подвыборки который уменьшает размеры массива и выделяет наиболее вероятные комбинации признаков. Затем к результату его работы снова применяется сверточный слой, и так далее.

В конце блока обработки образуется массив, который можно трактовать как вероятности наличия определенных фрагментов в определенных участках изображения. После этого происходит конкатенация этих массивов (в зависимости от параметров слоев выборки возможно вырождение в вектор) и дальнейшие слои аналогичным образом определяют схожесть этих двух векторов.

В случае второй архитектуры, результаты работы подсетей не объединяются а только лишь сравниваются (например с помощью косинусной меры подобия).

Для каждого вида архитектуры были опробованы варианты с разным количеством слоев в блоках обработки.

Обучение

Обучение проводилось с различными скоростями обучения и числом итераций от 100000 до 1000000. При обучении различных архитектур был получен приблизительно одинаковый результат. На рис. 3 изображен график ошибки в процессе обучения. По нему можно сказать что нейронная сеть не

обучилась так как в случае обучения ошибка стремится к 0 при увеличении числа итераций, и приемлемая точность получается уже в районе 100000-200000 тысяч итераций, а в нашем случае даже после 1000000 итераций ошибка достигает значений, близких к тем, которые были в начале обучения.

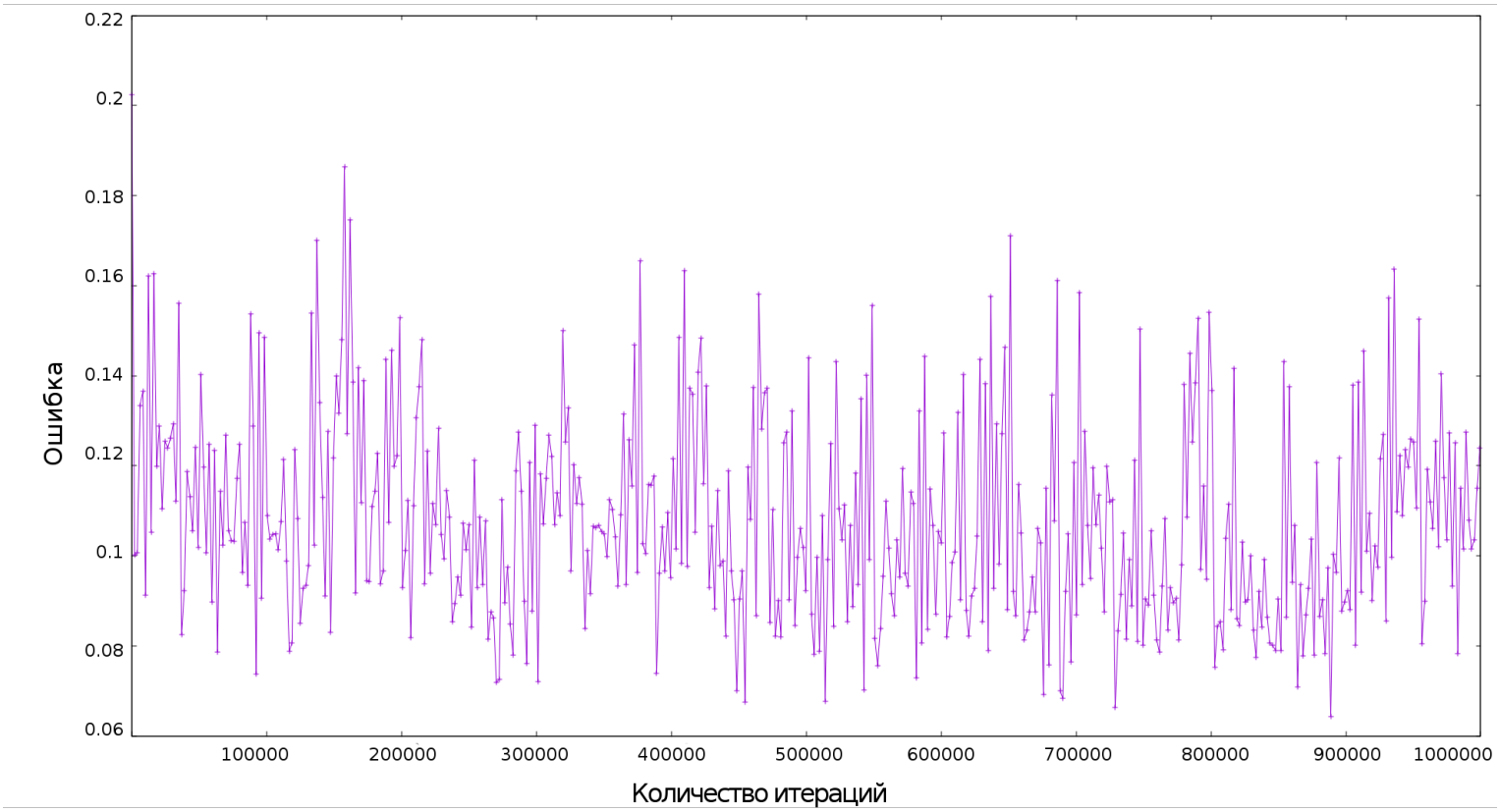

Рис. 4

Заключение

В результате данной работы был исследован фреймворк caffe, технология вычислений на GPU CUDA и библиотека OpenCV а так же создан набор данных для обучения нейронной сети, реализованы различные архитектуры нейронной сети для сравнения изображений.

Список литературы

1 Jure Žbontar, Yann LeCun Computing the Stereo Matching Cost with a Convolutional Neural Network (2014) [электронный ресурс] Режим доступа: https://pdfs.semanticscholar.org/2fdd/82708a99cec2bcd45c2c7f337a9beccced04.pdf (Дата обращения: 23.05.2017)

2 Sergey Zagoruyko, Nikos KomodakisLearning to Compare Image Patches via Convolutional Neural Networks [электронный ресурс] Режим доступа: https://arxiv.org/pdf/1504.03641.pdf (Дата обращения: 23.05.2017)

3 Jure Žbontar, Yann LeCun. Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. Journal of Machine Learning Research 17 (2016)