Аннотация

Оглавление

Введение

Системы технического зрения применяются во многих областях робототехники: начиная от мобильных роботов, заканчивая стационарными промышленными манипуляторами. Одна из основных задач систем машинного зрения – анализ среды функционирования робота. Среду функционирования можно представить как в виде двумерного изображения, так и в формате трехмерного облака точек, по которому можно дать качественную характеристику окружающих робота объектов и местности.

Актуальными средствами регистрации облаков точек на данный момент являются камеры глубины с инфракрасным датчиком, стереокамеры и сканирующие лидары. На основе полученных данных можно построить трехмерную карту местности по методу SLAM, однако, это не позволяет дать оценку окружающей среде с точки зрения находящихся в ней объектов. Для полноценного взаимодействия робота с целевыми объектами необходима информация об их форме, габаритных характеристиках, положении и ориентации в пространстве. Также подобные данные требуется для определения положения самого робота, планирования траектории движения и обхода препятствий. Для достижения этой цели следует классифицировать группы точек в облаке таким образом, чтобы можно было описать их поверхностями третьего порядка или заданными формами-примитивами.

Совместив полученную информацию об объектах в поле зрения робота с данными о его положении, можно составить картину о среде функционирования путем нанесения на карту встреченных роботом объектов и препятствий, что позволит производить качественную разведку местности и поиск на ней целевых объектов.

Хорошие результаты при решении задач сегментации объектов в облаке точек показывают нейросети, однако, они требуют больших вычислительных мощностей, что накладывает ограничения на создание полностью автономной и независимой от соединения с удаленным сервером робототехнической системы.

В данной работе будет рассмотрен метод сегментации облаков точек на формы-примитивы, способный работать на маломощных встраиваемых вычислительных системах.

Определения, обозначения, сокращения

1. Обзор существующих методов визуальной оценки положения и формы объектов

Оценка положения и формы объектов в кадре может производиться в двумерном пространстве кадра изображения и в трехмерном облаке точек. Анализ двумерного изображения позволяет определить положение целевого объекта в кадре и форму его видимой области, однако, не позволяет доступно и качественно определить форму его составных частей. С помощью облака точек, составленного по карте глубины, можно дать характеристику формы, положения частей и ориентации объекта.

1.1. Методы анализа двумерных изображений

В системах технического зрения задача сегментации объектов в кадре решается с помощью нескольких подходов:

- Классические методы:
    1. Выделение контуров;
    2. Выделение по цветовой маске;
    3. Использование дескрипторов и детекторов особых точек;
- Нейросетевые методы с использованием сверточных нейронных сетей:
    1. Двухэтапный метод;
    2. Одноэтапный метод.

Выделение контуров

Среди множества алгоритмов поиска контуров на изображении широко используется оператор Кэнни. Алгоритм можно разбить на пять этапов:

1. Сглаживание изображения;

2. Поиск градиентов. Границы отмечаются там, где градиент изображения приобретает максимальное значение;
3. Подавление не-максимумов. Только локальные максимумы отмечаются как границы;
4. Двойная пороговая фильтрация. Потенциальные границы определяются порогами;
5. Трассировка области неоднозначности. Итоговые границы определяются путем подавления всех краев, несвязанных с сильными границами.

Прежде чем начать определять контуры объекта, нужно отфильтровать шумы на изображении. Для этого зачастую применяется размытие по Гауссу. Фильтр Гаусса представляет собой средневзвешенное значение яркостей соседних пикселей в области матрицы ядра с весом, уменьшающимся с расстоянием от центра ядра *p*.

Дадим определение свертке по функции Гаусса:

$$GC[I]_p = \sum_{q \epsilon S} G_\sigma\big(||p - q||\big) I_q \quad (1)$$

$$G_\sigma(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right) \quad (2)$$

Вес пикселя определяется *Gσ (|| p - q ||), L2* нормой матрицы (Евклидовой нормой), которая представляет собой геометрическое расстояние между точками в пространстве, где σ – параметр, определяющий размер окрестности. Сила этого влияния зависит только от пространственного расстояния между пикселями, но не от их значений. Так, яркий пиксель имеет сильное влияние на соседний темный пиксель, хотя значения яркостей этих пикселей сильно отличаются. В результате, края изображения размываются, так как пиксели на неоднородных участках усредняются все вместе.

1/16

1	2	1
2	4	2
1	2	1

1/273

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

1/1003

0	0	1	2	1	0	0
0	3	13	22	13	3	0
1	13	59	97	59	13	1
2	22	97	159	97	22	2
1	13	59	97	59	13	1
0	3	13	22	13	3	0
0	0	1	2	1	0	0

Рисунок 1. Примеры ядер фильтра Гаусса с размерами 3x3, 5x5 и 7x7

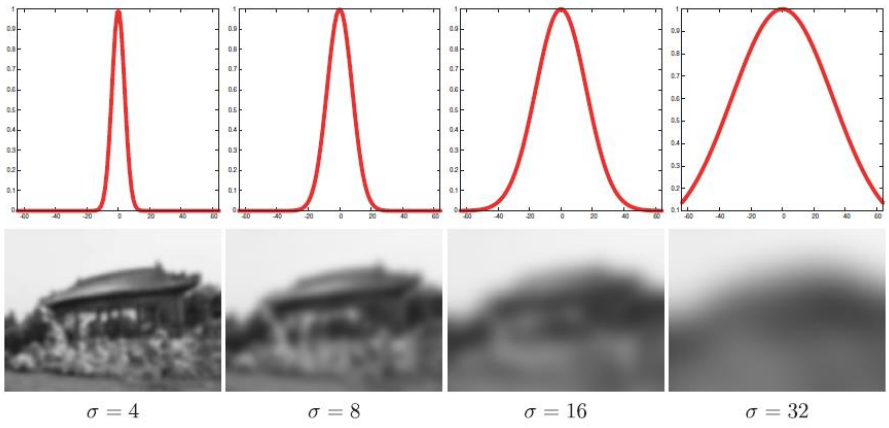

Рисунок 2. Пример Гауссова размытия при разных размерах ядра

Поиск градиента осуществляется следующим образом. Границы отмечаются там, где градиент изображения приобретает максимальное значение. Они могут иметь различное направление, поэтому алгоритм Кэнни использует четыре фильтра для обнаружения горизонтальных, вертикальных и диагональных ребер в размытом изображении.

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = arctg(\frac{G_y}{G_x})$$

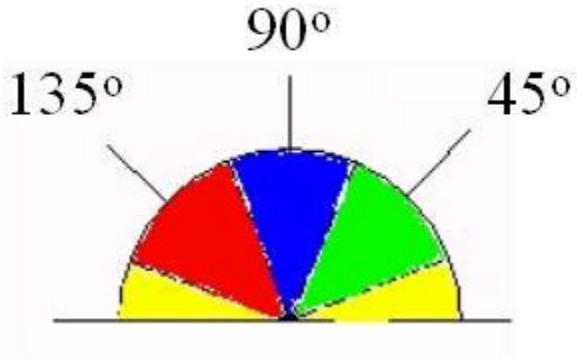

Рисунок 3. Определение направления градиента

Угол направления вектора градиента округляется и может принимать значения 0, 45, 90 и 135 градусов. Направление края, попадающего в каждую цветовую область будет установлено на определенное значение угла.

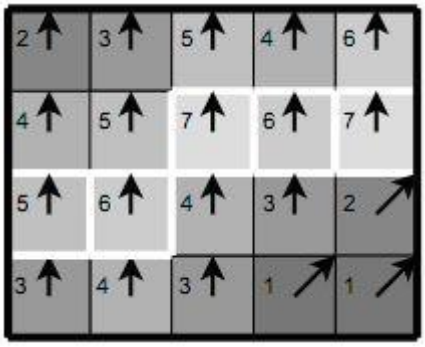

Рисунок 4. Визуализация градиента

Подавление не-максимумов – техника уменьшения толщины границ. Применяется для нахождения областей с резким изменением яркости. Алгоритм для каждого пикселя в градиентах:

1. Сравнение силы границ текущего пикселя с силой границ пикселей в положительных и отрицательных направлениях градиента
2. Если сила границ текущего пикселя больше значения в маске с пикселями с таким же направлением градиента, значение будет сохранено, в противном случае – подавлено.

Сила границ пикселя – производная первого порядка, например, величина градиента.

После подавления не-максимумов, оставшиеся пиксели дают более точное представление границ на изображении. Однако некоторые пиксели остаются из-за шума и цветового разброса. Чтобы отбросить эти выбросы, важно отфильтровать пиксели со слабым значением градиента и сохранить пиксели с сильным значением градиента. Для этого выбираются верхнее и нижнее пороговые значения. Если значение градиента краевого пикселя выше верхнего порогового значения, он помечается как сильный краевой пиксель. Если значение градиента краевого пикселя меньше верхнего порогового значения и больше нижнего значения, он помечается как слабый. Если значение меньше нижнего порога, оно будет подавлено. Пороговые значения определяются эмпирически и будут зависеть от содержимого изображения.

Пиксели с сильными границами обязательно будут задействованы в конечном результате, поскольку они извлекаются из истинных границ изображения. Однако возникает спорный момент с пикселями со слабыми границами, так как они могут быть извлечены и из истинных границ, и из шума. Для достижения более точного результата следует избавиться от пикселей вызванных шумом. Обычно пиксель со слабым значением, полученный от истинных границ будет связан с пикселем с сильным значением, в то время как шумовые характеристики ни с чем не связаны. Чтобы отследить связь, рассматривают слабый пиксель и соединенные с ним соседние 8 пикселей. Если есть хотя бы один сильный пиксель в этом квадрате, слабый пиксель можно сохранить.

В результате работы вышеперечисленных методов можно получить четкие контуры изображения.

Рисунок 5. Контуры изображения

Выделение по цветовой маске

Выделение объекта с помощью цветовой маски используется в случае, если целевой объект существенно выделяется на фоне по цвету. Осуществляется путем преобразования изображения из RGB в HSV пространство и выделения в нем нужных значений тона, насыщенности и значения.

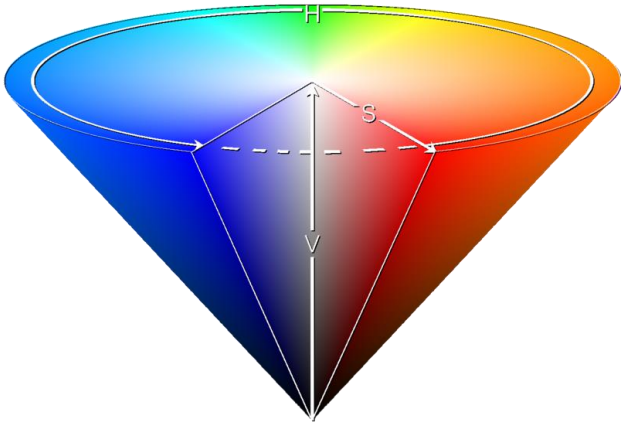

Рисунок 6. Цветовая модель HSV

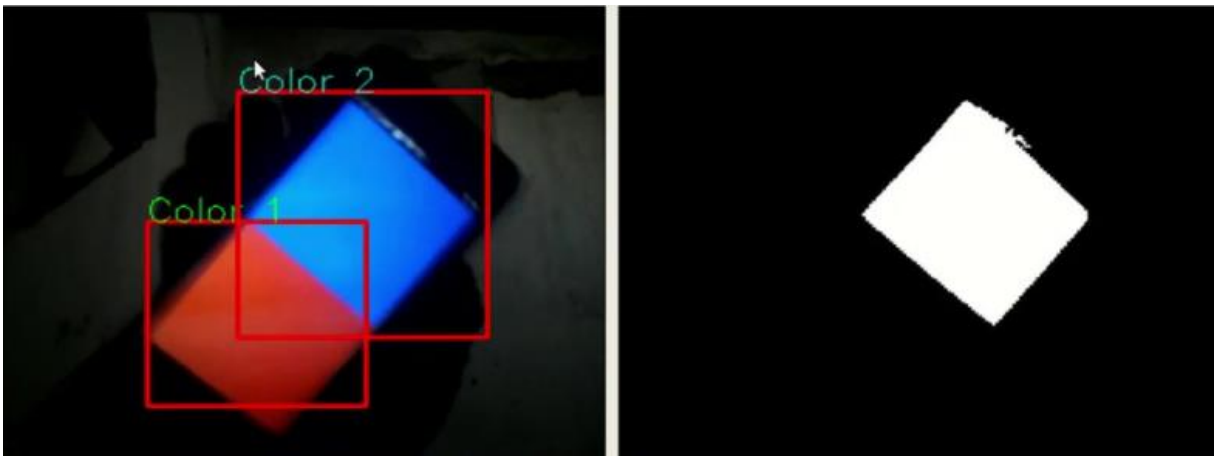

Рисунок 7. Выделение объектов по цветовой маске

Использование дескрипторов и детекторов особых точек

При одинаковой яркости с фоном или перекрытии объект может не иметь четких границ, поэтому до популяризации нейросетевых подходов для поиска объектов широко использовались детекторы признаков и дескрипторы. Детектор особых точек – алгоритм, который выбирает точки на изображении на основе некоторого критерия. Дескриптор – вектор значений, который описывает участок изображения вокруг интересующей точки. Вместе особая точка и ее дескриптор обычно называют локальной особенностью.

Основными критериями нахождения особых точек являются:

- Отличимость – особая точка должна явно выделяться на фоне и быть отличимой (уникальной) в своей окрестности;
- Инвариантность – определение особой точки должно быть независимо к аффинным преобразованиям;
- Стабильность – определение особой точки должно быть устойчиво к шумам и ошибкам;

- Уникальность – кроме локальной отличимости, особая точка должна обладать глобальной уникальностью для улучшения различимости повторяющихся шаблонов;
- Интерпретируемость – особые точки должны определяться так, чтобы их можно было использовать для анализа соответствий и выявления интерпретируемой информации из изображения.

Широкую популярность приобрели такие дескрипторы как SIFT (Scale Invariant Feature Transform), HOG (Histogram of Oriented Gradients), ORB (Oriented FAST and rotated BRIEF).

Дескрипторы могут решать задачу сопоставления изображений, что позволяет детектировать и классифицировать объекты в кадре. В общем случае получается схема на Рис.8:

Рисунок 8. Алгоритм сопоставления изображений

Основным этапом при детектировании особых точек является построение пирамиды гауссианов и разностей гауссианов. Разностью гауссианов называют изображение, полученное путем попиксельного вычитания

одного гауссина исходного изображения из гауссиана с другим радиусом размытия.

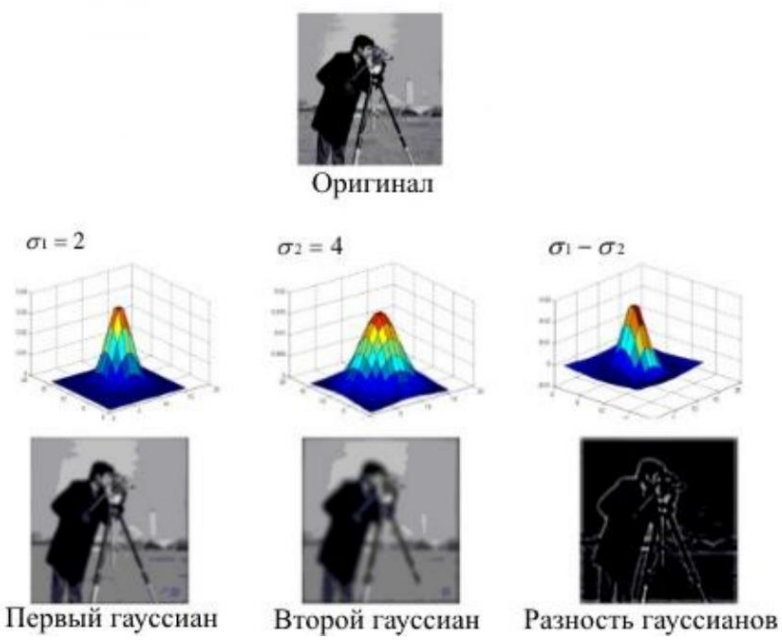

Рисунок 9. Построение разности гауссианов

Масштабируемое пространство изображения — множество различных, сглаженных некоторым фильтром, вариантов исходного изображения. Гауссово масштабируемое пространство является линейным и инвариантным относительно сдвигов, вращений, масштаба, не смещающим локальные экстремумы, и обладает свойством полугрупп. Различная степень размытия изображения гауссовым фильтром может быть принята за исходное изображение, взятое в некотором масштабе.[цитата]

Инвариантность относительно масштаба достигают за счет нахождения ключевых точек для исходного изображения, взятого в разных масштабах. Поэтому строят пирамиду гауссианов: все масштабируемое пространство разбивают на некоторые участки — октавы, причем часть

масштабируемого пространства, занимаемого следующей октавой, в 2 раза больше части, занимаемой предыдущей. При переходе от одной октавы к другой происходит уменьшение размеров изображения в 2 раза. Каждой октаве изображения соответствует бесконечное множество гауссианов изображения. В связи с этим необходимо строить только некоторое их число N с определенным шагом по радиусу размытия. С тем же шагом достраивают два дополнительных гауссиана, таким образом, получают N + 2 гауссиана. Масштаб первого изображения следующей октавы равен масштабу изображения предыдущей октавы с номером N. Сглаживая изображение гауссовым фильтром по приведенному алгоритму, получаем пирамиду гауссианов. [цитата]

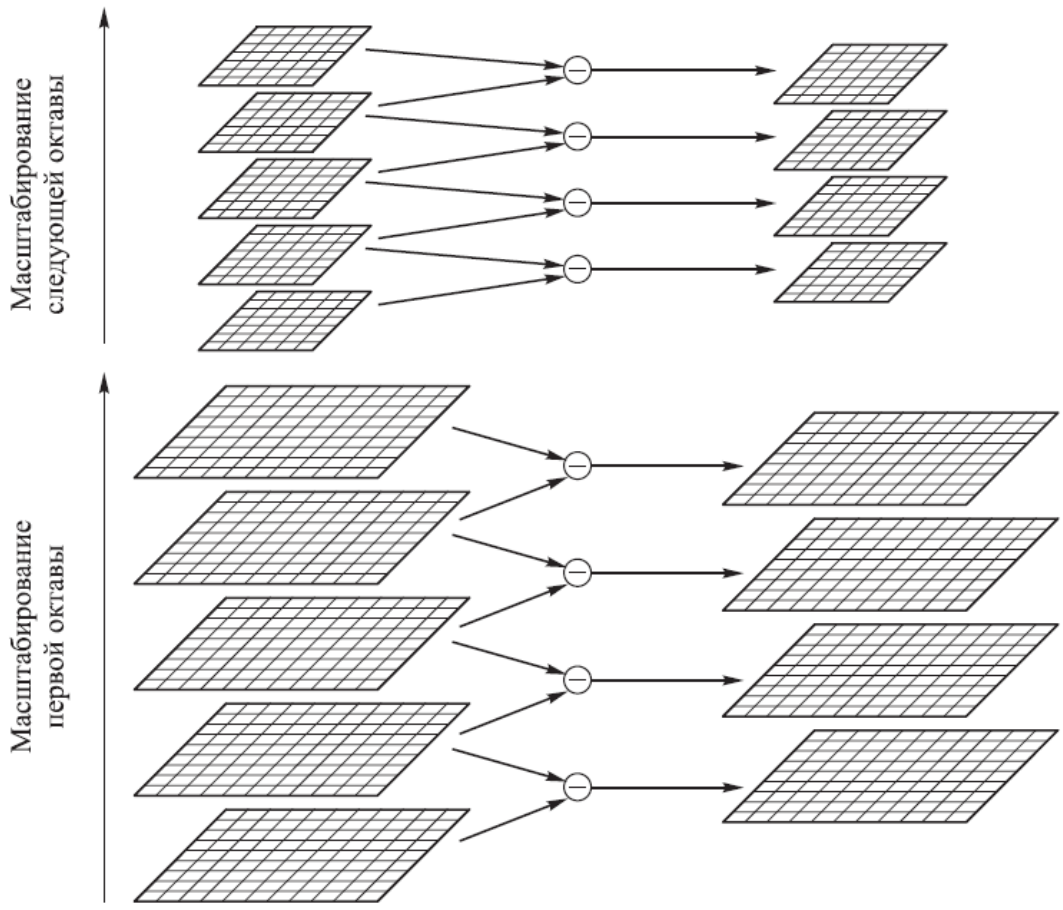

Рисунок 10. Построение разности гауссианов

После построения пирамид определяют является ли точка ключевой, а также определяют ее ориентацию. Точка является ключевой, если она представляет собой локальный экстремум разности гауссианов. Далее создается дескриптор ключевой точки.

Рисунок 11. Пример получения дескриптора

Рисунок 12. Пример работы дескриптора SIFT

Вышеперечисленные подходы успешно могут справляться с некоторыми задачами поиска объектов, однако, в значительной мере уступают в производительности нейросетевым подходам.

Нейросетевые методы с использованием сверточных нейронных сетей

С популяризацией в 2012 году сверточных нейронных сетей [Krizhevsky] в задачах классификации изображений и детекции объектов стали применять различные модели на их основе.

В общем случае детекция сводится к перебору ограничивающих рамок целевого объекта на изображении с порогом уверенности в правдивости классификации. За счет того что рамки могут быть различных размеров и иметь различные координаты, полный перебор становится длительным и неэффективным методом. Для уменьшения количества этих рамок выделяют два подхода:

1. Двухэтапный метод – на первом этапе с помощью селективного поиска или слоя нейронной сети выделяются регионы интереса, с некоторой вероятностью содержащие внутри целевые объекты. На втором этапе регионы рассматриваются классификатором с точки зрения принадлежности исходным классам и производится уточнение ограничивающих рамок.
2. Одноэтапный метод – метод, не использующий отдельный алгоритм для генерации регионов интереса, а вместо этого предсказывающий координаты определенного количества ограничивающих рамок с различными параметрами, такими как степень уверенности результатов классификации и корректируя в дальнейшем их положение.

Двухэтапные методы

Двухэтапные методы используются в таких известных моделях как R-CNN/Fast/Faster R-CNN/Mask R-CNN.

В качестве примера рассмотрим R-CNN (Region-based Convolutional Neural Network). Вместо использования для поиска сегментов скользящие окна фиксированного размера, на первом шаге алгоритм пытается найти селективным поиском "регионы" – прямоугольные области различных размеров, которые, могут содержать целевой объект. Размеры найденных регионов подаются на вход сверточной нейронной сети. На последнем этапе вектора признаков регионов анализируются с помощью метода опорных векторов, проводится классификация регионов.

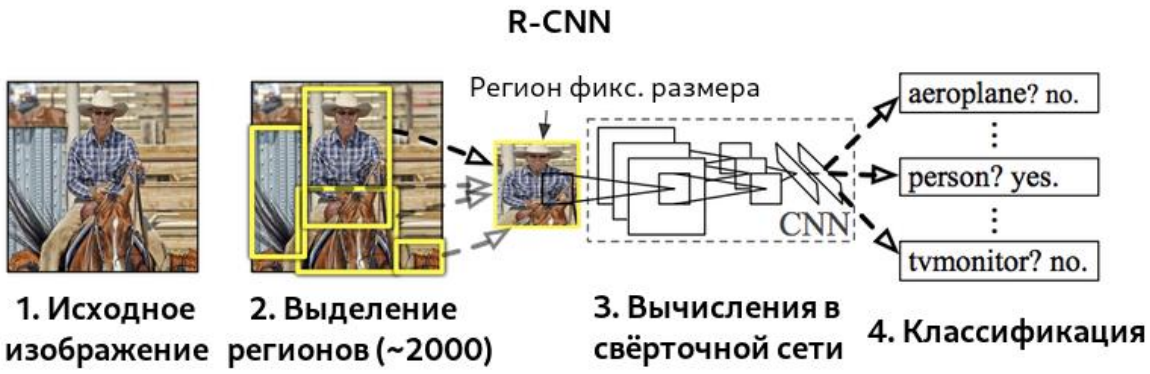

Рисунок 13. Схема работы R-CNN

Одноэтапные методы

По сравнению с двухэтапными методами, одноэтапные имеют более высокое быстродействие. Рассмотрим принцип работы на примере SSD.

Модель SSD использует принцип пирамидальной иерархии выходов сверточной нейросети для эффективного обнаружения объектов различных размеров. Изображение последовательно подается на уменьшающиеся в размерах слои свёрточной сети. Выход из последнего слоя каждой размерности участвует в принятии решения по детекции объектов, таким образом, складывается "пирамидальная характеристика" изображения. Это дает возможность обнаруживать объекты различных размеров, так как размерность выходов первых слоёв сильно коррелирует с ограничивающими рамками для крупных объектов, а последних – для небольших. Таким образом, крупные размеры могут быть обнаружены на более высоком уровне, а маленькие объекты – на низких уровнях. Как и в других моделях, функция потерь обеспечивает совместный вклад как в локализацию, так в классификацию.

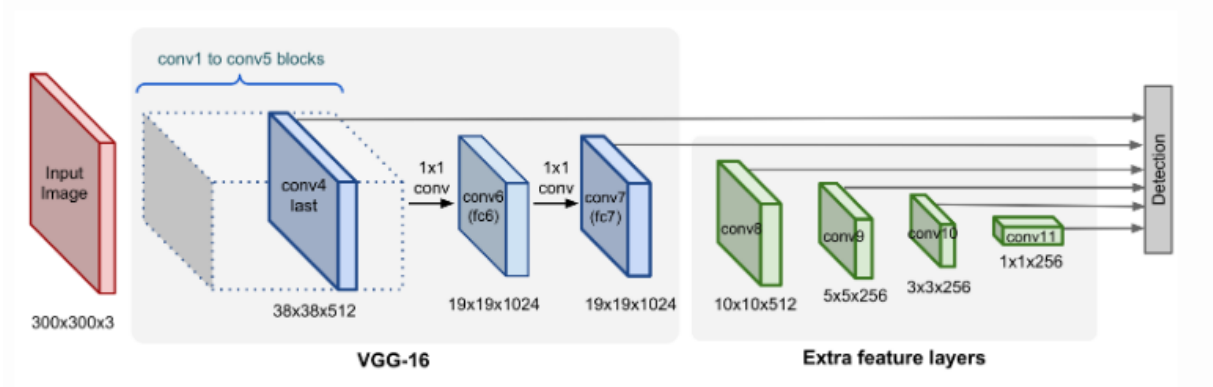

Рисунок 14. Архитектура нейронной сети для алгоритма SSD

Как показал обзор актуальных методов поиска объектов на изображении, для соблюдения баланса между эффективностью и быстродействием в поставленной задаче оптимальнее всего использовать алгоритм SSD.

1.2. Методы анализа трехмерных облаков точек

Развитие методов анализа и сегментации облаков точек началось с методов использования эвристических алгоритмов. Основной метод, который используется в некоторых задачах и до сих пор – алгоритм RANSAC и различные его модификации. В статье [schnabel] представлен способ сегментации и аппроксимации сложного облака точек с помощью данного алгоритма путем поиска в нем форм-примитивов плоскости, цилиндра, сферы и конуса.

Рисунок 15. Результат работы алгоритма RANSAC для аппроксимации облака точек примитивами

Алгоритм отличается хорошей устойчивостью к шумам и выбросам, а также способен сравнительно быстро работать в большом облаке точек и способен работать с формами различной степени замкнутости. Облако точек составленное по карте глубины не является полноценным результатом сканирования местности и объекта, а значит трехмерное облако ограничено ракурсом и углом обзора камеры, то есть задние части объектов не видны.

Еще один алгоритм основанный на RANSAC представлен в статье[baysac буряты]. Его отличием является возможность одновременного поиска нескольких примитивов и вероятностный подбор точек в окрестности

формы. Это позволяет выбирать точки для построения гипотезы примитива не вслепую, что ускоряет работу алгоритма и точность найденных гипотез.

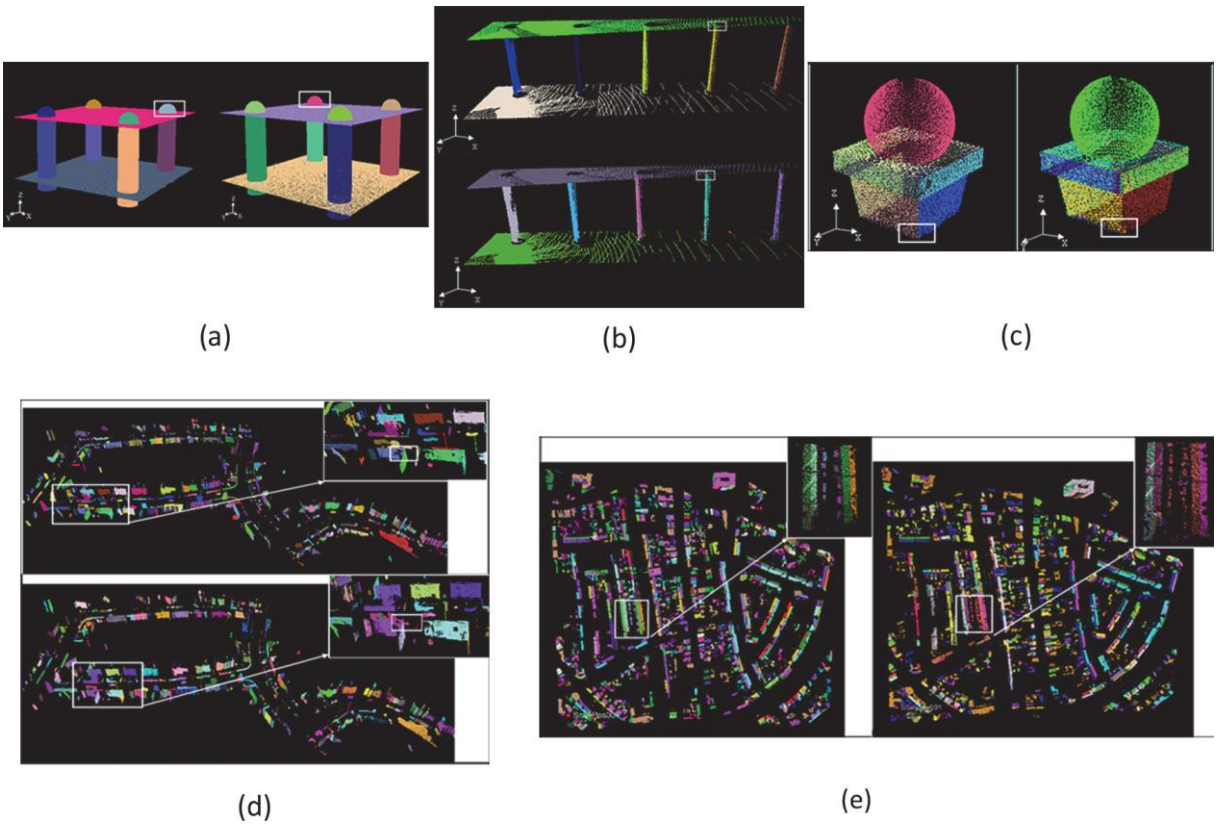

Рисунок 16. Результат работы MultyBaySAC

Современные способы сегментации облаков точек также бывают основаны на нейросетях. Самые популярные и эффективные – PointNet и SPFN (Supervised Primitive Fitting Network), основанный на PointNet.

Типичные сверточные архитектуры для перераспределений весов и оптимизации ядра требуют регулярных форматов входных данных, таких как сетки изображений или трехмерные воксели. Так как облака точек имеют нестандартный формат, обычно их преобразуют в трехмерные воксельные сетки, наборы изображений или проекции на координатные плоскости прежде чем подавать на вход нейронной сети. Однако такие

преобразования делают входные данные слишком объемными – появляются артефакты квантизации, которые могут скрыть естественную инвариантность данных. По это причине было решено в качестве входных данных использовать само облако точек. Выходом нейросети является класс для всего входа или его сегмента.

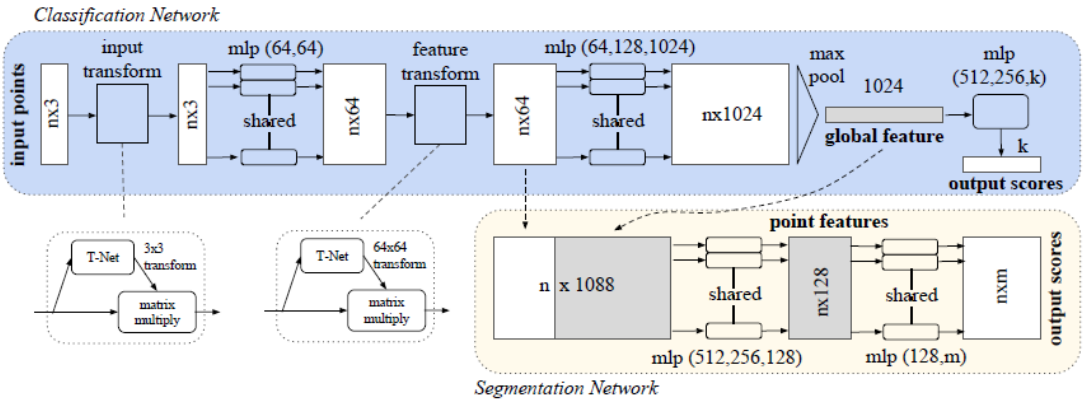

Рисунок 17. Архитектура PointNet

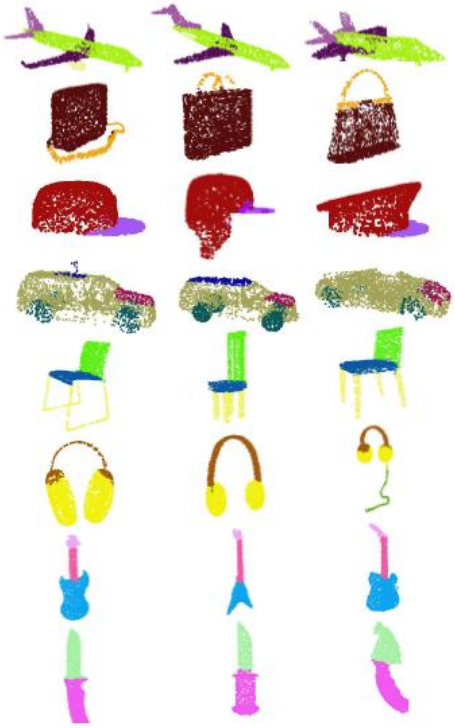

Рисунок 18. Результат работы PointNet при сегментации частей

Как показано на Рис.18, модель успешно справляется с задачей сегментации частей, а также может производить семантическую классификацию найденных частей.

Основанная на PointNet модель SPFN одновременно решает задачу сегментации и классификации методом разделения облака точек на примитивы. В набор входят плоскость, сфера, цилиндр и конус. В сравнении с RANSAC этот метод показывает гораздо более точные результаты, что делает его использование более выгодным в некоторых задачах.

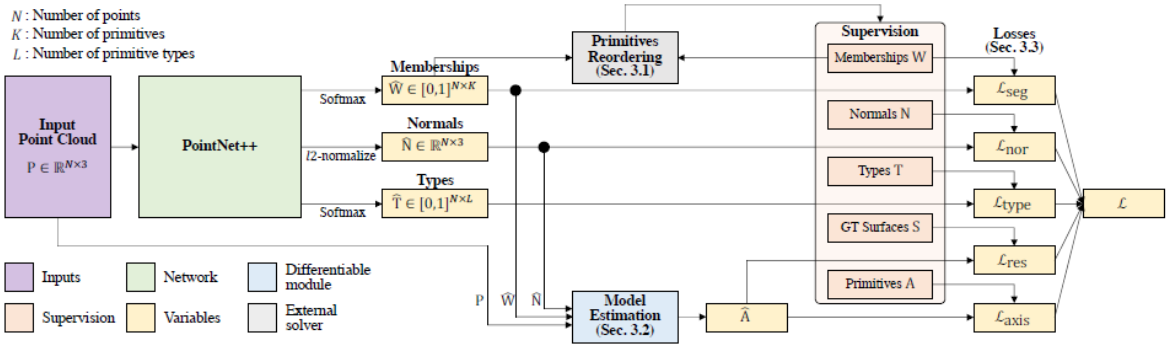

Рисунок 19. Архитектура SPFN

Ind	Method	Seg. (Mean IoU)	Primitive Type (%)
1	Eff. RANSAC [28]+J	43.68	52.92
2	Eff. RANSAC [28]*+J*	56.07	43.90
3	Eff. RANSAC [28]+J*	45.90	46.99
4	Eff. RANSAC [28]+J*+$\hat{\mathbf{W}}$	69.91	60.56
5	Eff. RANSAC [28]+J*+$\hat{\mathbf{W}}$+$\hat{\mathbf{t}}$	60.68	92.76
6	Eff. RANSAC [28]+$\hat{\mathbf{N}}$+$\hat{\mathbf{W}}$+$\hat{\mathbf{t}}$	60.56	93.13
7	DPPN (Sec. 4.4)	44.05	51.33
8	SPFN-$\mathcal{L}_{\text{seg}}$	41.61	92.40
9	SPFN-$\mathcal{L}_{\text{norm}}$+J*	71.18	95.44
10	SPFN-$\mathcal{L}_{\text{res}}$	72.70	96.66
11	SPFN-$\mathcal{L}_{\text{axis}}$	**77.31**	96.47
12	SPFN ($\hat{\mathbf{t}} \rightarrow$ Est.)	75.71	95.95
13	SPFN	77.14	**96.93**

Рисунок 20. Метрика IoU в сравнении с RANSAC

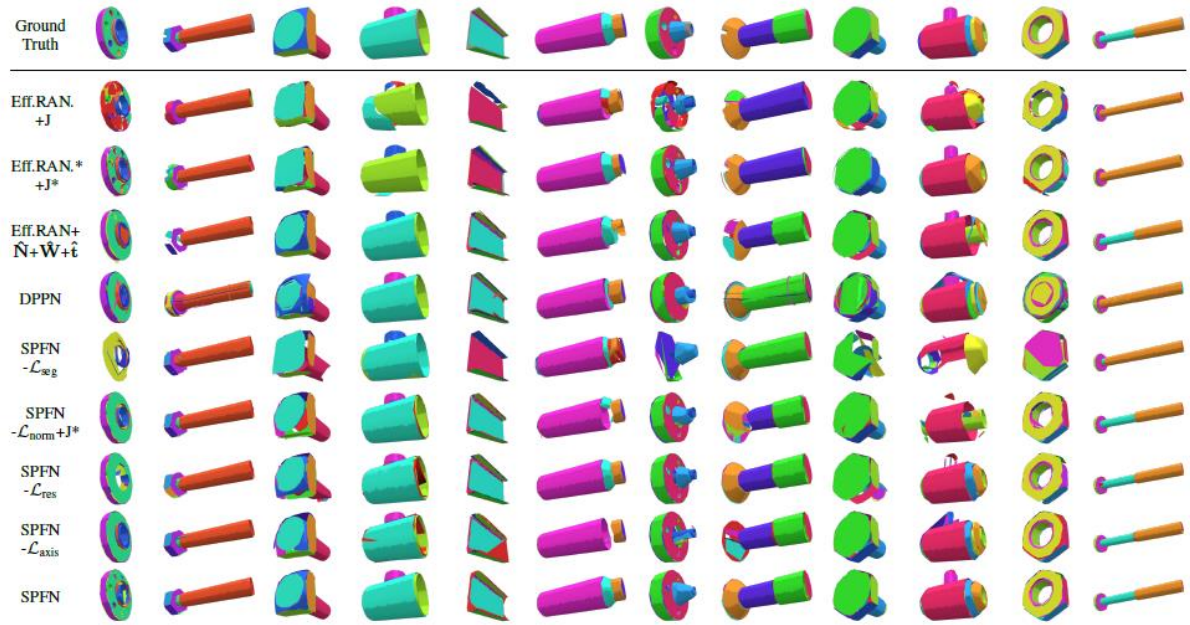

Рисунок 21. Результаты аппроксимации в сравнении с RANSAC

В сравнении с SPFN PointNet эффективнее работает в больших облаках точек и способен оперировать на целой сцене, в то время как PointNet хорош для сегментации и классификации трехмерных сканов объектов.

1.3. Интеллектуальные системы технического зрения в робототехнике

Система технического зрения в робототехнике – одна из важнейших частей всего программно-аппаратного комплекса робота, так как именно она позволяет наладить взаимодействие робота и окружающей среды.

В качестве аппаратных средств используются как стандарные цифровые RGB-камеры, стереокамеры и камеры с инфракрасным датчиком глубины. Также для достижения определенных целей, таких как устранение бликов и фильтрации цвета в промышленных решениях используют различные светофильтры и поляризаторы. Также с использованием камер глубины можно построить облако точек окружающей робота местности, что, например, позволяет промышленным манипуляторам определять способ

захвата целевого объекта, а мобильным просчитывать траекторию и способ перемещения, учитывая особенности рельефа.

Интеллектуальные системы технического зрения в мобильной робототехнике чаще всего применяются с использованием RGB-камер и нейросетевых моделей анализа изображений. Ярким примером является использование таких систем в беспилотных автомобилях, летательных аппаратах и космических аппаратах.

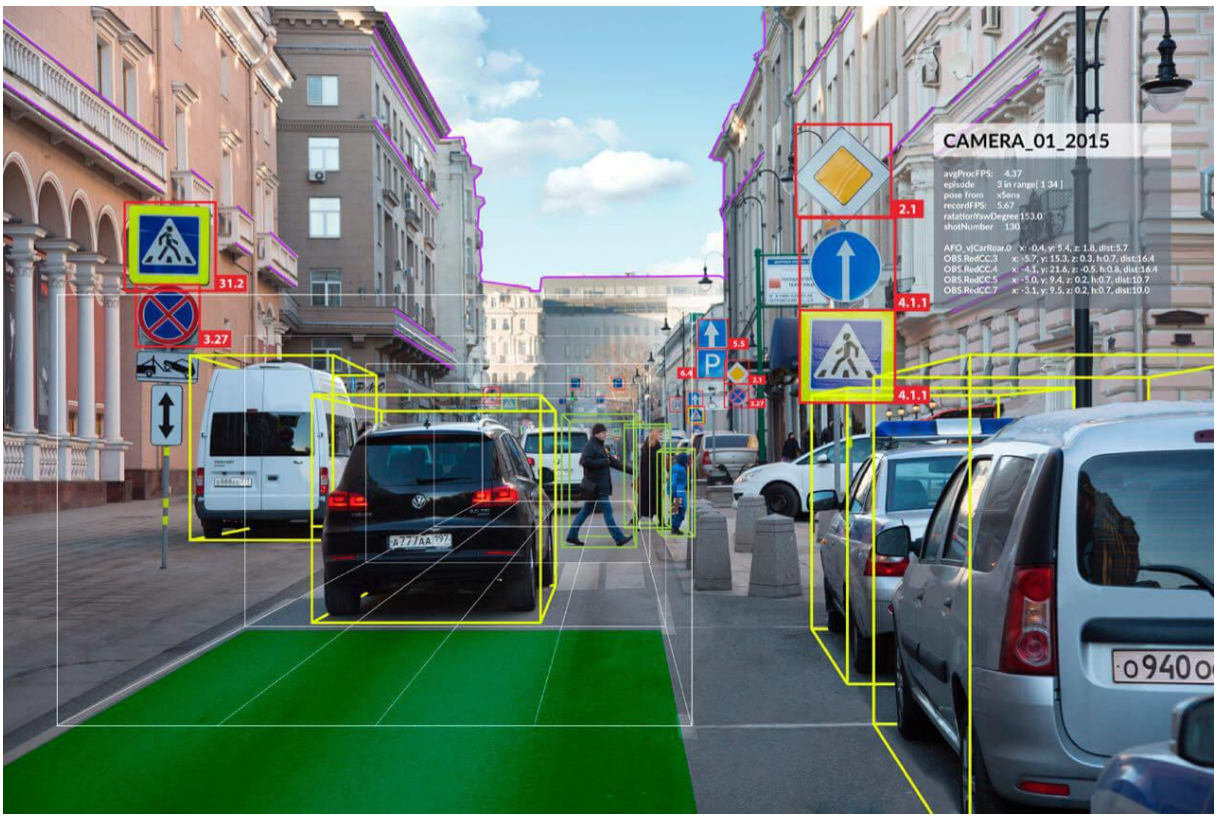

Рисунок 22. Пример использования интеллектуальной системы технического зрения в беспилотном автомобиле

В промышленной робототехнике также широко используют стереокамеры. Например, стерео сенсор Kuka_3D Perception распознает неструктурированное окружение в режиме реального времени и точно позиционирует объекты с разной кинематикой в пространстве. Робот

может фиксировать свое положение в пространстве с точностью до миллиметра и решать задачи еще быстрее и эффективнее. Благодаря интегрированной технологии распознавания KUKA_3D Perception обрабатывает данные изображения непосредственно в сенсоре и фиксирует текущее положение объекта с точностью до миллиметра. Камера оснащена встроенным графическим процессором, что позволяет запускать на ней нейросетевые модели, что также позволяет использовать ее в мобильной робототехнике.

Рисунок 23. Интеллектуальная камера KUKA_3D Perception

В ходе экспериментов при выполнении работы использовалась Intel RealSense, которая оснащена стереокамерой и инфракрасным датчиком, однако она не позволяет интегрировать интеллектуальные алгоритмы анализа и используется как сенсор.

Рисунок 24. Камера Intel RealSense

1.4. Заключение первой главы

Как показал обзор решений, интеллектуальные системы технического зрения широко применяются в современной робототехнике и активно развиваются. Использование нейросетевых моделей позволяет эффективно анализировать окружающую среду, однако такое решение не всегда является доступным для мобильных роботов и имеет высокую стоимость. Могут возникнуть сложности как со стороны аппаратного обеспечения, на котором должны работать алгоритмы, так и со стороны интеграции и масштабируемости решения. При решении некоторых задач также не всегда требуется настолько высокая производительность, поэтому использование аналитических и эвристических методов также не теряет своей актуальности.

2. Модели и алгоритмы нейро-эволюционной системы технического зрения

После изучения существующих решений был сделан вывод, что для маломощных вычислительных систем нужен алгоритм, способный производить анализ окружающей среды робота. В данной работе был разработан интеллектуальный алгоритм визуальной оценки положения и формы объектов с бортовой камеры робота, основанный на использовании сверточных нейросетей и эвристических алгоритмов.

2.1 Архитектура нейро-эволюционной системы технического зрения

По результатам экспериментов было предложено следующее решение: с помощью сверточной нейронной сети осуществляется поиск целевого объекта на RGB изображении, вырезается область интереса из RGB-D и строится облако точек, в котором с помощью RANSAC производится первичный поиск примитивов, которые впоследствии доводятся до лучшего состояния с помощью генетического алгоритма. Подобный подход должен обеспечить соблюдение баланса между объемом вычислительных мощностей, временем работы алгоритма и эффективностью, что может позволить использовать его в качестве встраиваемой системы для автономных мобильных роботов.

Рисунок 25. Архитектура алгоритма

2.2. Выбор нейросетевой модели для оценки положения объектов в кадре

В задачах детектирования объектов и классификации изображений хорошие результаты показывают сверточные нейронные сети. В отличие от нейронных сетей прямого распространения, помимо полносвязных слоев в них присутствуют сверточные слои и слои подвыборки, что позволяет определять больше признаков на входном изображении.

Одной из самых эффективных и быстродействующих моделей для детектирования объектов на изображении и сегментации является SSD MobileNetV2, разработанная специально для встраеваемых систем и активно использующаяся в мобильной робототехнике. Скорость работы – 19 мс, COCO mAP – 20.2.

2.2.1 Принципы построения сверточных нейронных сетей

Операция свертки в случае работы с изображениями представляет собой вычисление нового значения заданного пикселя, при котором учитываются значения окружающих его соседних пикселей. Главным элементом свертки является ядро, которое представляет собой квадратную матрицу нечетного размера, по умолчанию 3x3. Ядро является матрицей весов, которая «скользит» над двумерным изображением, выполняя поэлементное умножение с той частью, над которой оно находится в данный момент, суммируя затем полученные значения в новый пиксель. Независимо от того, попадает ли входной признак в то же место, он определяется в зависимости от того, находится он в зоне ядра, создающего выходные данные, или нет. Это значит, что размер ядра сверточной нейронной сети определяет количество признаков, которые будут объединены для получения нового признака на выходе. Такой способ перемещения называется паддинг (padding), что позволяет сохранить исходный размер матрицы входного изображения проходя по каждому пикселю.

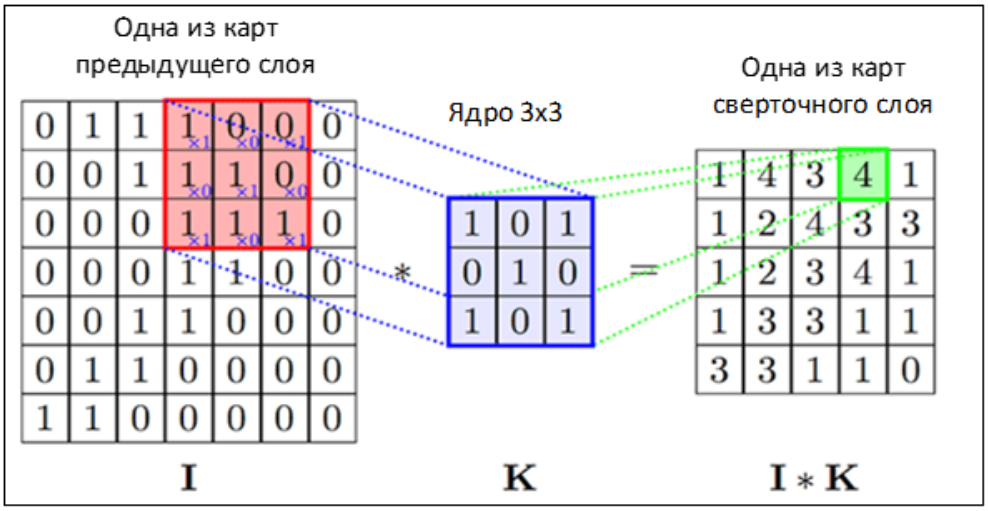

Рисунок 1. Пример свертки

Слой можно описать формулой

$$x^l = f(x^{l-1} * k^l + b^l),$$

где

$x^l -$ выход слоя l,

$f(x) -$ функция активации,

$b^l -$ коэффициент сдвига слоя l,

$*$ $-$операция свертки входа x с ядром k.

Цель слоя подвыборки (pooling layer) – уменьшение размерности карт сверточного слоя. Если на предыдущей операции свертки уже были выявлены некоторые признаки, то для дальнейшей обработки настолько подробное изображение уже не нужно, и оно уплотняется до менее подробного. К тому же фильтрация уже ненужных деталей помогает модели избежать переобучения. Как и в сверточном слое, в слое подвыборки есть ядро, как правило, размером 2x2, которая позволяет уменьшить предыдущие карты сверточного слоя в 2 раза. Карта признаков разделяется на ячейки размером 2x2 элемента, из которых выбирается один. Зачастую применяют Max-Pool – выбор наибольшего значения в ячейке. Обычно в подвыборочном слое применяется функция активации ReLU.

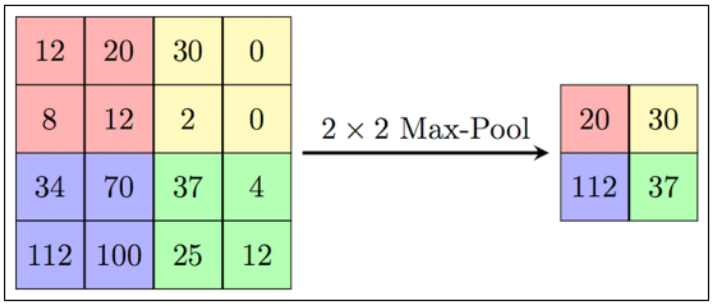

Рисунок 2. Пример работы слоя подвыборки

Формально подвыборочный слой можно описать формулой

$$x^l = f(a^l * subsample(x^{l-1}) + b^l),$$

где

$x^l -$ выход слоя l,

$f() -$ функция активации,

$a^l, b^l -$ коэффициент сдвига слоя l,

$subsample$ – операция выборки локальных максимальных значений.

Полносвязный слой представляет собой слой обычного многослойного перцептрона (Рис.), основной целью которого является классификация.

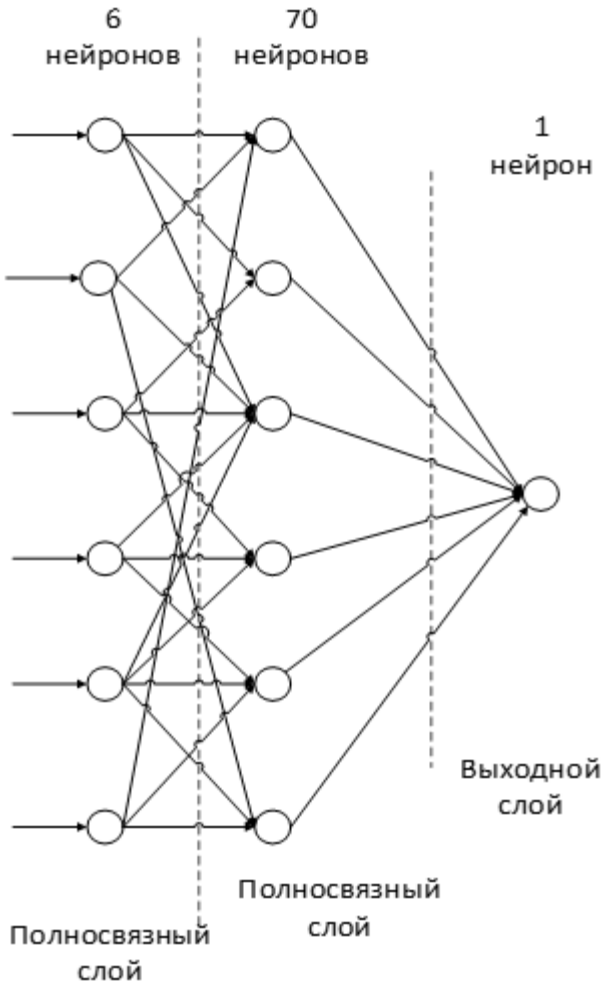

Рисунок 3. Пример полносвязных слоев

Математически полносвязный слой можно описать формулой:

$$x_j^l = f\left(\sum_i x_i^{l-1} * w_{i,j}^{l-1} + b_j^{l-1}\right),$$

где

$x_j^l -$ выход предыдущего слоя l,

$f -$ функция активации,

$b_j^{l-1} -$ коэффициент сдвига слоя l,

$w_{i,j}^l -$ матрица весовых коэффициентов слоя l.

Функции активации представляет собой непрерывную функцию, принимающую на вход вещественное число, а на выходе дает значение в интервале от -1 до 1.

Примерами функций активации являются сигмоида, гиперболический тангенс, ступенчатая функция, softsign, ReLU (Rectified Linear Unit) и др. В современных моделях широко применяется ReLU.

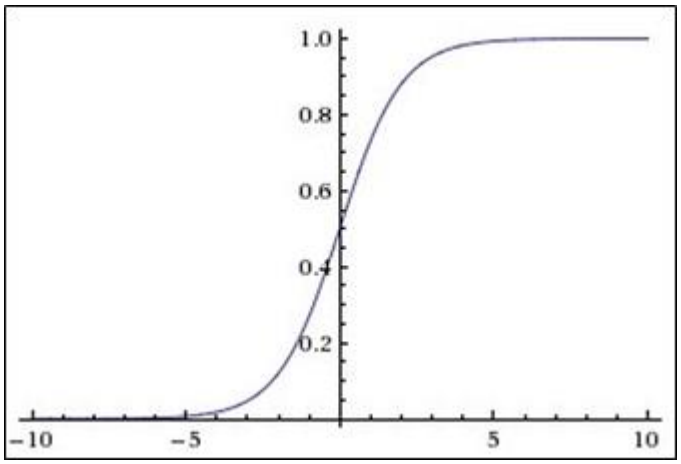

Рисунок 4. Пример сигмоидальной функции

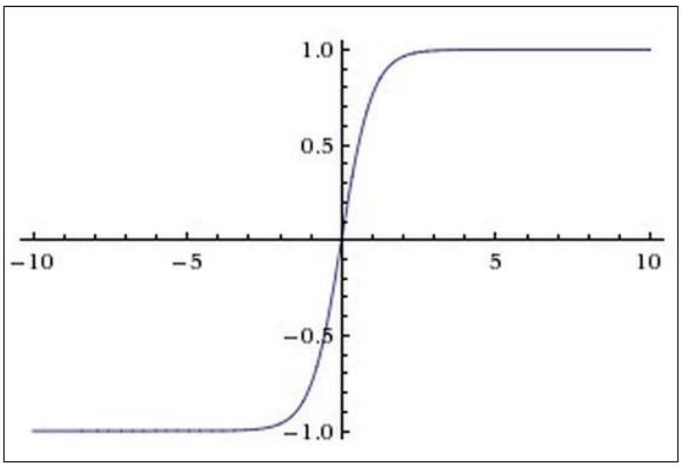

Рисунок 5. Пример функции гиперболического тангенса

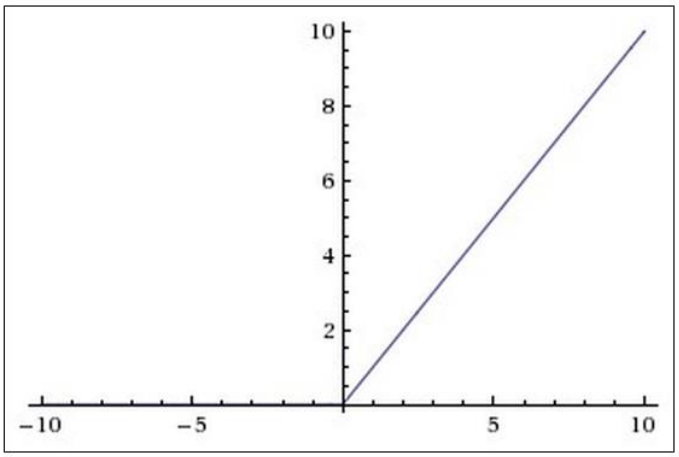

Рисунок 6. Пример функции ReLU

В общем виде топология сверточной нейронной сети имеет следующий вид, представленный на Рис.:

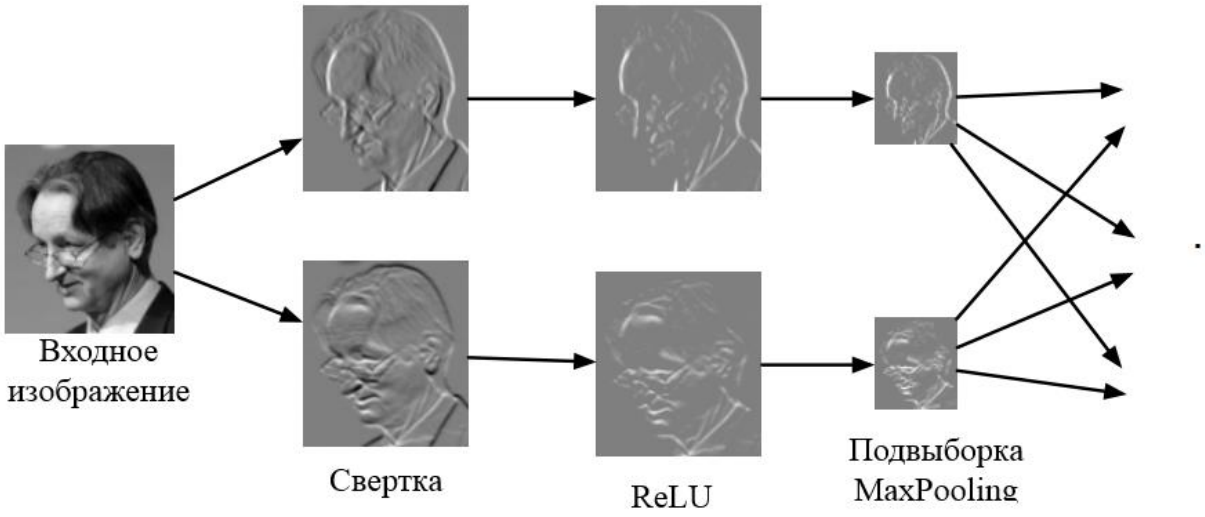

Рисунок 7. Топология сверточной нейронной сети

2.2.2 Архитектура модели

Архитектура выбранной модели SSD-MobileNetV2 имеет следующий вид (Рис., Рис.):

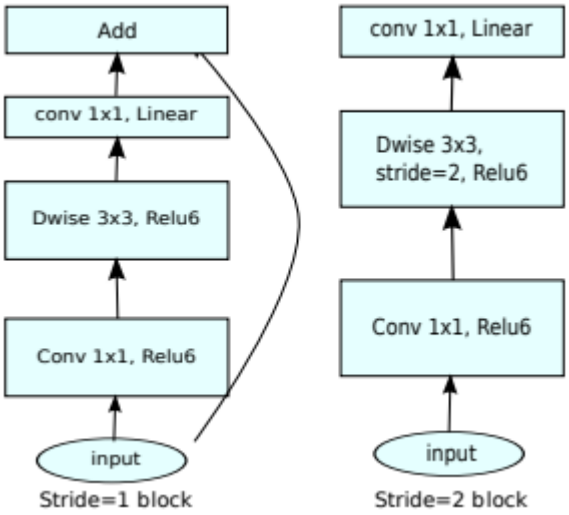

Рисунок 8. Блочная схема SSD MobileNetV2

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Рисунок 9. Архитектура SSD MobileNetV2

MobileNetV2 использует принцип Single Shot Detection (SSD). По изображению располагаются рамки (далее - bounding box) различных размеров, накрывающие участки пикселей, в которых потенциально может находиться целевой объект.

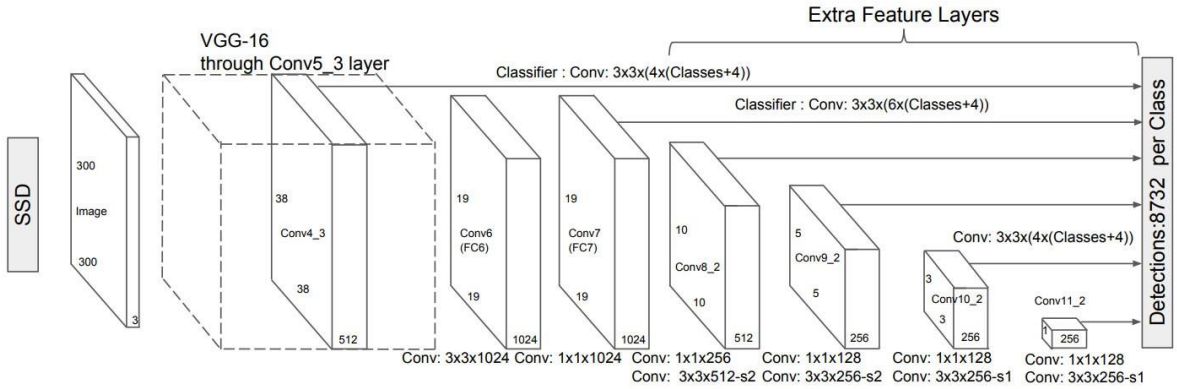

Рисунок 228. Архитектура модели SSD

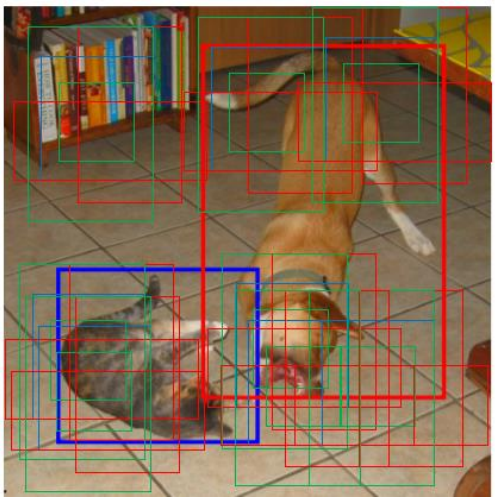

Рисунок 10. Single Shot Detection

В процессе обучения рамки по умолчанию сравниваются по соотношению сторон, местоположению и масштабу с истинными. Выбираются блоки с наибольшим перекрытием с блоком искомого объекта. Пересечение над объединением (IoU – Intersection over Union) между предсказанными рамками и истинными должно быть больше 0.5. В итоге выбирается bounding box с наибольшим попаданием в рамку искомого объекта.

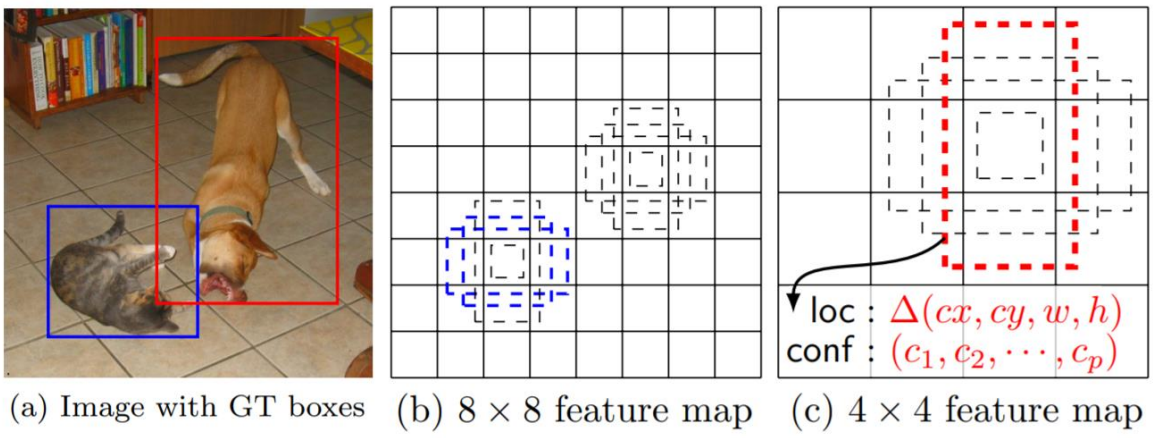

Рисунок 11. Пример работы SSD

Каждая гипотеза состоит из:

- Рамки со смещением положения. Δcx, Δcy, h и w представляют смещение от центра поля по умолчанию, его высоту и ширину.

- Уверенность в правильности найденного класса. Класс 0 зарезервирован под отсутствие объекта.

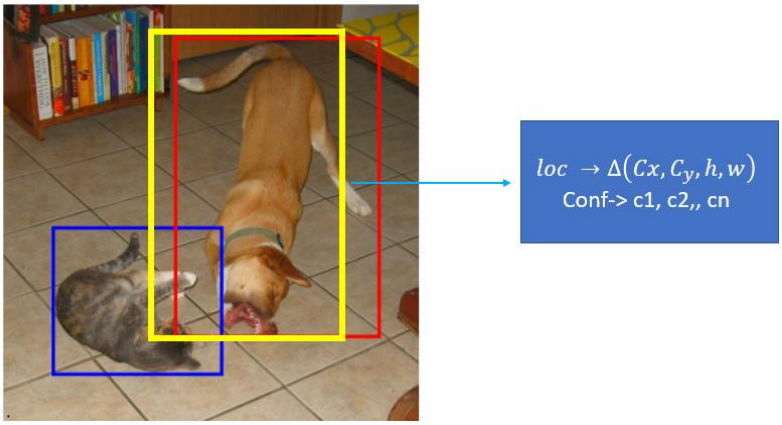

Рисунок 12. Гипотеза в SSD

Функция потерь в SSD называется MultiBox loss $L(x, c, l, g)$ и состоит из confidence loss и localization loss.

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} \log(\hat{c}_i^p) - \sum_{i \in N_{eg}} \log(\hat{c}_i^0),$$

где

$$\hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

N – количество совпавших рамок,

c – оценка найденного класса.

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^{k} smooth_{L1}\left(l_i^m - \hat{g}_j^m\right),$$

где

l – потери гипотезы,

g – smooth L1 loss,

cx, cy – отклонение от рамки d высоты h и ширины,

$x_{ij}^{p} = \begin{cases} 1 \\ 0 \end{cases}$ если $IoU > 0.5$ между найденной и истинной рамкой.

$$L(x, c, l, g) = \frac{1}{N}\left(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)\right),$$

где

N – число положительных совпадений,

$\alpha -$ вес $localization\ loss$

SSD-MobileNetV2 реализована во фреймворке Tensorflow Object Detection API и предобучена на датасете COCO. С его помощью можно находить объекты с видеопотока камеры мобильного робота.

Рисунок 13. Пример работы SSD MobileNevV2

2.2.3 Процесс обучения модели

В случае, когда используется готовая модель, эффективно производить ее дообучение (Fine tuning). Суть дообучения заключается в размораживании последних слоев нейронной сети и их последующем обучении. Таким образом, корректируются слои, которые имеют наиболее абстрактные представления. Производя дообучение только нескольких слоев, мы уменьшаем риск переобучения, а главное это позволяет сделать текущую модель ещё более подходящей для задачи.

Если верхние слои следует дообучить, то полносвязные слои нужно заменить на свои и также обучить. Процедура Fine tuning состоит в следующем:

1. Заморозить все слои предварительно обученной модели;
2. Добавить новые слои к обученной модели;
3. Обучить добавленные слои;
4. Разморозить несколько верхних слоев;
5. Обучить эти слои и добавленную часть вместе.

Прежде чем начинать обучение модели, необходимо корректно составить набор обучающих данных. Операция подразумевает под собой разметку набора изображений: выделение области, в которой должен находиться целевой объект и создание аннотации с положением и размерами области. Также для улучшения результата рекомендуется использовать аугментацию данных – искусственное зашумление, размытие, отражении изображение и прочие модификации.

После этого можно переходить к процессу обучения. Обучение нейронной сети, как правило, осуществляется с помощью вычисления прямого,

обратного распространения ошибки и алгоритма градиентного спуска. Основная цель алгоритма – расчет частных производных параметров модели и обновление весов на их основании.

Дадим несколько определений

Логистическая регрессия:

$$z = w^T x + b,$$

$b -$ ветор коэффициента смещения,

$w -$ вектор весовых коэффициентов,

$x -$ вектор входных данных.

Гипотеза:

$$a = \sigma(z),$$

$\sigma(z) -$ функция активации.

Функция потерь:

$$L(x, c, l, g) = \frac{1}{N}\left(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)\right)$$

На первом этапе алгоритма градиентного спуска рассчитывается прямое распространение ошибки путем вычисления z, a и $L(a,y$.

Возьмем производные:

$$da = \frac{d}{da}(L(x, c, l, g))$$

$$dz = a - y$$

$$dw = dz\, x$$

$$db = dz$$

После вычисления частных производных происходит обновление параметров за одну итерацию градиентного спуска:

$$w = w - \alpha\, dw,$$

$$b = b - \alpha\, db,$$

$\alpha -$ коэффициент скорости обучения.

Таким образом алгоритм находит локальный минимум функции с помощью движения вдоль градиента, минимизируя среднюю ошибку на выходе нейронной сети при подаче на вход обучающих данных. Для корректной работы алгоритма необходимо эмпирически выбрать коэффициент скорости обучения так, чтобы не перескочить локальный минимум и не застрять в одной точке. Также принято использовать различные оптимизаторы, такие как Adam, Momentum, Adagrad и т.д.

Также для улучшения результата следует осуществлять нормализацию признаков:

$$x = \frac{x - u}{s},$$

где

x – вектор значений входных данных,

u – среднеарифметическое всех данных,

s – разброс значений.

Так как входные значения признаков могут изменяться в очень большом диапазоне, работа алгоритма может оказаться некорректной. Нормализующие слои можно применять после каждого предыдущего слоя.

Обучение происходит в несколько эпох – количество раз, которое набор данных пройдет через нейронную сеть.

2.2.4 Оценка модели

Минимизация функции потерь говорит об успешном обучении модели.

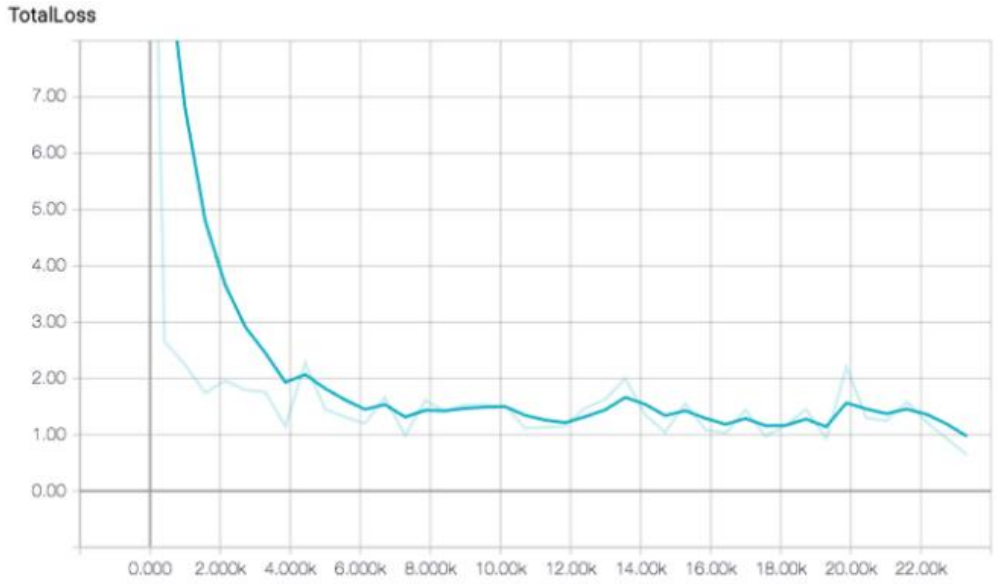

Рисунок 322. Изменение функции потерь

На Рис.12412 показана зависимость изменения функции потерь от выбранной скорости обучения.

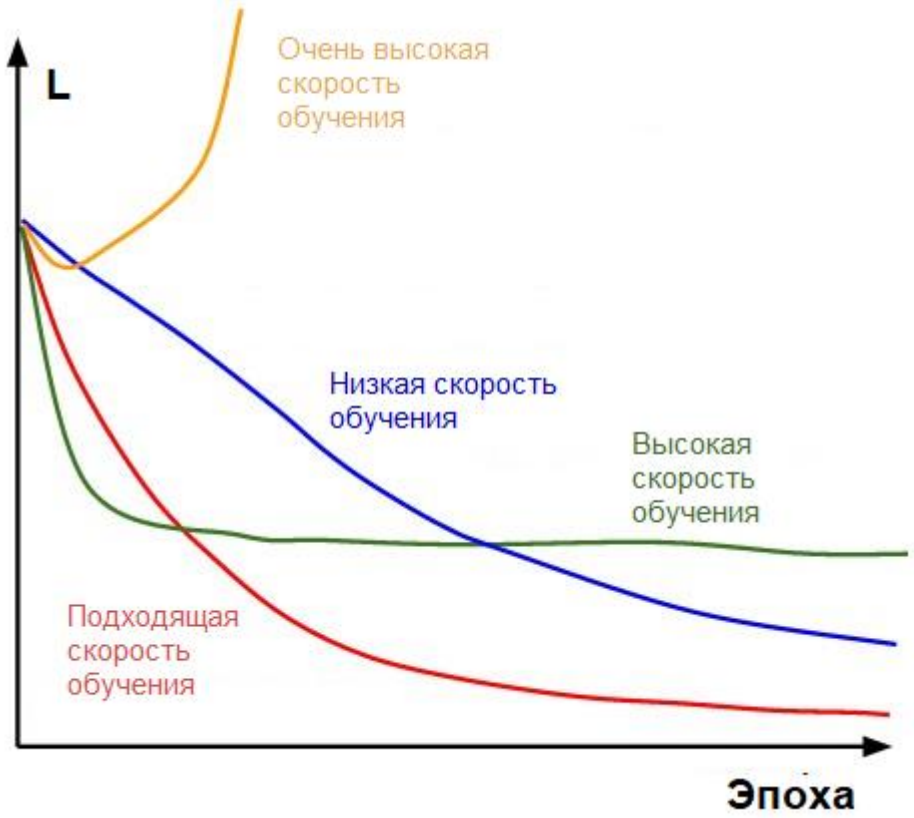

Рисунок 12412. Зависимость функции потерь от скорости обучения

Быстрая минимизация функции потерь на Рис.322 достигается за счет использования заранее предобученной модели.

При оценке работы нейросетевых моделей используется несколько метрик.

Перед переходом к самим метрикам необходимо ввести важную концепцию для описания этих метрик в терминах ошибок классификации — confusion matrix (матрица ошибок).

Допустим, что у нас есть два класса и алгоритм, предсказывающий принадлежность каждого объекта одному из классов, тогда матрица ошибок классификации будет выглядеть следующим образом:

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

Рисунок 1214. Матрица ошибок

В данном случае $\hat{y}$ – предсказание алгоритма, y – истинная метка класса. Таким образом, ошибки классификации бывают двух видов: False Negative (FN) и False Positive (FP).

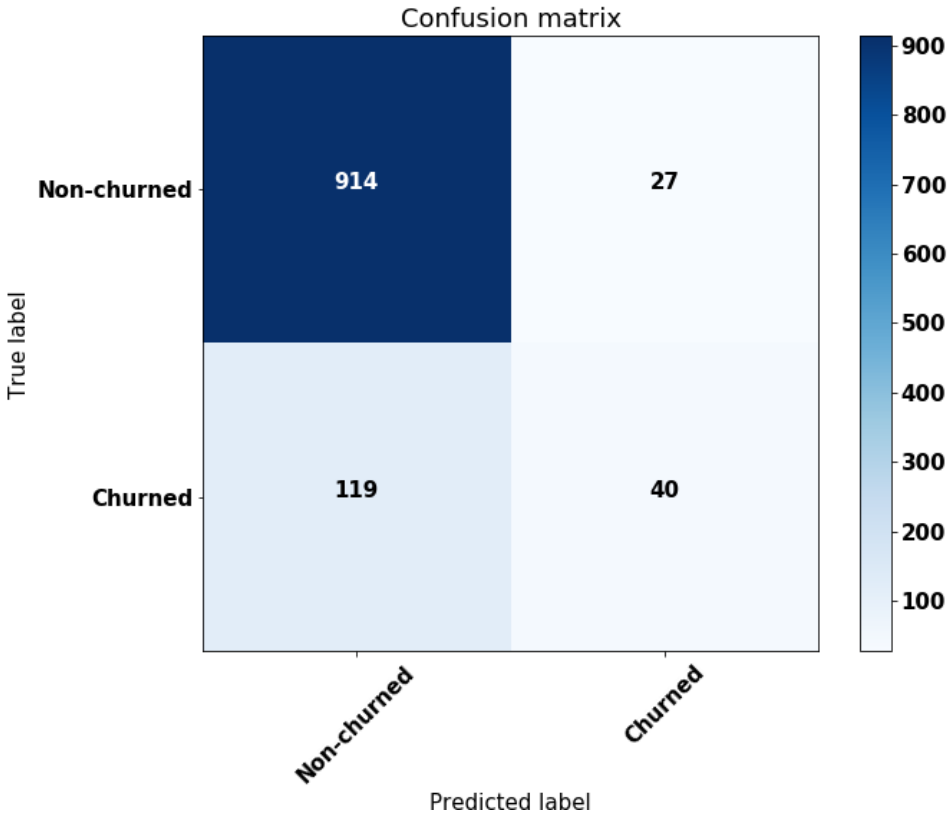

Рисунок 5134. Confusion matrix

Для оценки качества работы алгоритма используются метрики точности (precision) и полноты (recall). Точность можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а полнота показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

Метрика Average Precision (AP) производит выборку кривой по всем уникальным значениям recall, когда падает максимальное значение precision. С этим изменением можно измерить точную площадь под кривой precision-recall.

$$AP = \sum_{n} (recall_n - recall_{n-1}) precision_n$$

Метрика mAP (mean Average Precision) является усредненным значением AP. В некотором контексте вычисляется AP для каждого класса и усредняется.

Таким образом при успешном обучении должно возрастать значение метрики mAP, как показано на Рис.148.

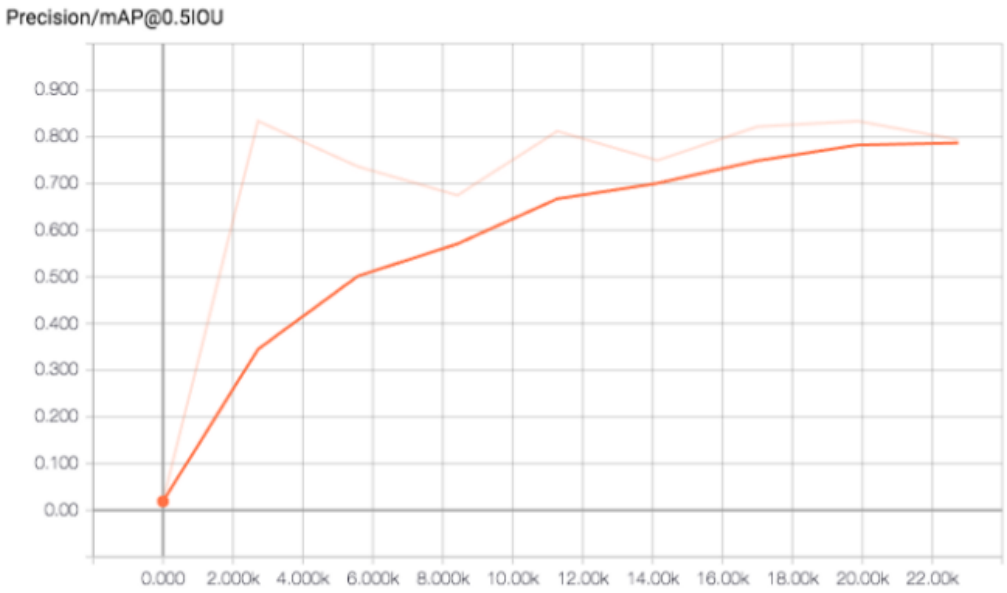

Рисунок 148. Изменение mAP

Также используется метрика Intersection over Union (IoU).

$$IoU(A,B) = \frac{A \cap B}{A \cup B}$$

Или другими словами – отношение площади пересечения к объему объединения. Сравнение следует проводить с размеченным набором данных.

2.3. Разработка эволюционного алгоритма для оценки формы объектов

После выделения зоны интереса с помощью SSD-MobileNet-V2 по полученной карте глубины составляется облако точек целевого объекта. Для определения его формы, положения и габаритных размеров был разработан двухэтапный эволюционный алгоритм на основе RANSAC и генетического алгоритма. Экспериментально было выяснено, что оперируя во всем трехмерном пространстве полученной зоны интереса, генетический алгоритм не может качественно аппроксимировать форму облака точек. Так как возникает дополнительная неопределенность из-за возможности изменения параметров во всем пространстве, различных вариантов выбора модели становится в разы больше. В лабораторных тестовых условиях данную проблему можно было решить путем ограничения максимального расстояния, поиска на которое примитив может удалиться от центра масс искомого облака, однако реальные объекты, состоящие из нескольких форм, как правило, имеют центр масс не строго посередине, и искомая форма может быть смещена. С целью уменьшения неопределенности предлагается использовать RANSAC, который оперирует в поле точек облака, а не в поле координат зоны интереса. После нахождения первоначальной модели примитива, можно использовать генетический алгоритм, который произведет более точную оценку формы объекта. В этом случае он будет ограничен первоначальными параметрами модели и расстояние от ее центра.

2.3.1 Описание алгоритма RANSAC

Алгоритм RANSAC позволяет путем случайно выбранных точек задать математическую модель объекта, качественно отфильтровав выбросы и лишние точки. Основной идеей является поиск такой модели, чтобы в окрестность ее математической функции попадало как можно большее количество точек из общего облака. В данной работе функция оценки найденной модели имеет вид

$$f = \frac{N_{inliers}}{N_{outliers} + 1},$$

где N *inliers* – количество точек, попавших в окрестность, N *outliers* – число точек, не попавших в нее, равно N всех точек на изображении – N *inliers*. Алгоритм в процессе работы стремится максимизировать функцию.

2.3.1.1 Метод работы RANSAC

Как было описано выше, RANSAC имеет недостаток, который состоит в слишком большом влиянии случайных величин на работу алгоритма. Т.к. точки для поиска моделей выбираются случайным образом, нахождение нужной модели может занять продолжительное время из-за длительного перебора точек. Как правило, при реализации алгоритма используется нормальное распределение вероятностей при выборе точек, из-за чего выбор производится вслепую.

В данной работе для решения этой проблемы была использована априорная и апостериорная вероятность при выборе точек. Рассмотрим принцип на примере плоскости.

Для построения уравнения плоскости необходимо выбрать три точки, по которым вычисляются коэффициенты уравнения плоскости в каноническом виде.

1) Выбирается первая точка в пространстве облака, используя нормальное распределение

2) После выбора первой точки производится перераспределение вероятностей выбора остальных точек:

    1 Рассчитываются расстояния от выбранной точки A до каждой точки в облаке;

    2 Округляются до некоторого порядка (подобрать в зависимости от разрешения входного облака и изображения), производится поиск уникальных элементов, т.е. точек, до которых округленное расстояние одинаково;

    3 Рассчитывается дискретное распределение вероятностей (Рис.2), т.е. считается отношение количества повторений n-го элемента списка уникальных элементов к общему числу элементов;

    4 Согласно новому распределению выбирается расстояние и набор точек, которые удалены на него от выбранной A;

    5 В этом наборе точек случайно выбирается одна, которая становится следующей выбранной точкой B

3) Рассчитывается средняя точка отрезка AB, относительно нее производится шаг 1 и далее.

Такой способ позволяет с большей вероятностью выбрать среди точек, находящихся в определенном радиусе от исходной, что важно для поиска в пространстве облака, где могут быть резкие скачки расстояний между точками, и может помочь с поиском небольших объектов.

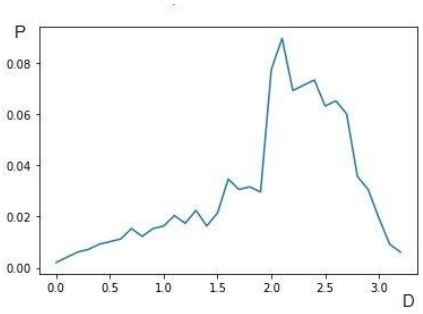

Рисунок 2. Распределение вероятностей при выборе точек в облаке куба

Модификация алгоритма, приведенная в работе, имеет следующую схему (Рис.3):

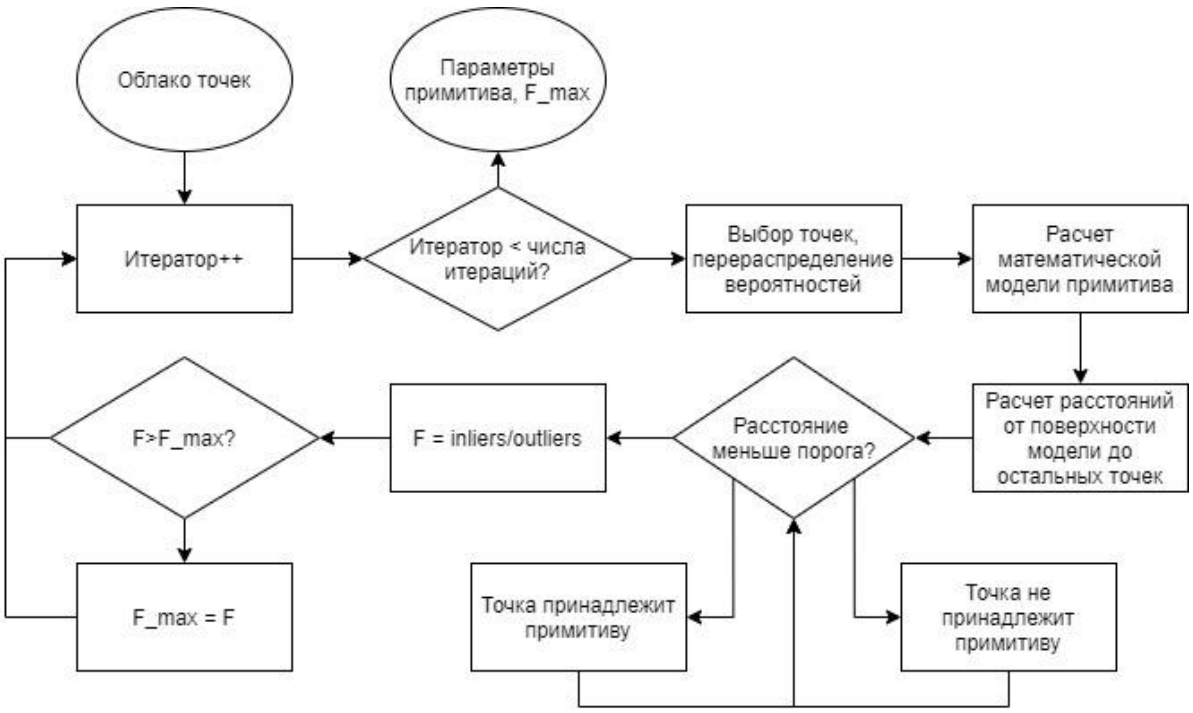

Рисунок 3. Блок-схема алгоритма RANSAC

2.3.1.2 Разработка математической модели примитива

Для демонстрации возможностей алгоритма были смоделированы примитивы плоскости и сферы. Первичный выбор примитивов основан на выборе таких точек, по которым можно составить их уравнение.

Для плоскости уравнение примет вид

$Ax + By + Cz + D = 0$,

Где $A, B, C \neq 0$.

Точки $M0(x0,y0,z0)$, $M1(x1,y1,z1)$, $M2(x2,y2,z2)$ не принадлежат одной плоскости.

Составим уравнение

$$\begin{matrix} x - x0 & x1 - x0 & x2 - x0 \\ y - y0 & y1 - y0 & y2 - y0 \\ z - z0 & z1 - z0 & z2 - z0 \end{matrix} = 0$$

Вычислим миноры по 1 столбцу

$$M1 = \begin{vmatrix} y1 - y0 & y2 - y0 \\ z1 - z0 & z2 - z0 \end{vmatrix}$$

$$M2 = \begin{vmatrix} x1 - x0 & x2 - x0 \\ z1 - z0 & z2 - z0 \end{vmatrix}$$

$$M3 = \begin{vmatrix} x1 - x0 & x2 - x0 \\ y1 - y0 & y2 - y0 \end{vmatrix}$$

Получаем коэффициенты

A = det(M1)

B = -det(M2)

C = det(M3)

D = -x0 det(M1) + y0 det(M2) – z0 det(M3)

Расстояние от точки до плоскости можно вычислить по формуле

$$R = \frac{|Ax0+By0+Cz0+D|}{\sqrt{A^2+B^2+C^2}}$$

В случае сферы для более качественного результата следует несколько усложнить дальнейшие операции.

Сферу можно описать двумя точками и их нормалями как к реальной поверхности. Нормали ищутся методом главных компонент по ковариационной матрице случайных векторов ближайших к точке соседей.

$$C = \frac{1}{k}\sum_{i=1}^{k}\left(p^i - \overline{p}\right) * \left(p^i - \overline{p}\right)^T, C * \overrightarrow{v_j} = \lambda_j * \overrightarrow{v_j}, j \in \{0,1,2\},$$

Где k – число точек-соседей точки pi, $\bar{p}$ – центроид всех ближайших соседей, λ_j – собственное значение матрицы ковариантности и $\overrightarrow{v_j}$ ее собственный вектор.

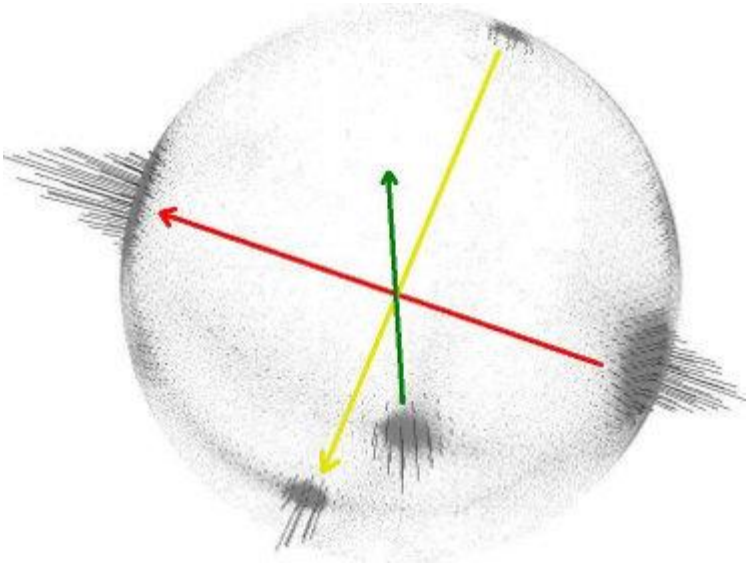

Рисунок 4. Пример нахождения нормалей точек

Точки (p1,p2) – выбранные случайным образом точки, (n1,n2) – их нормали.

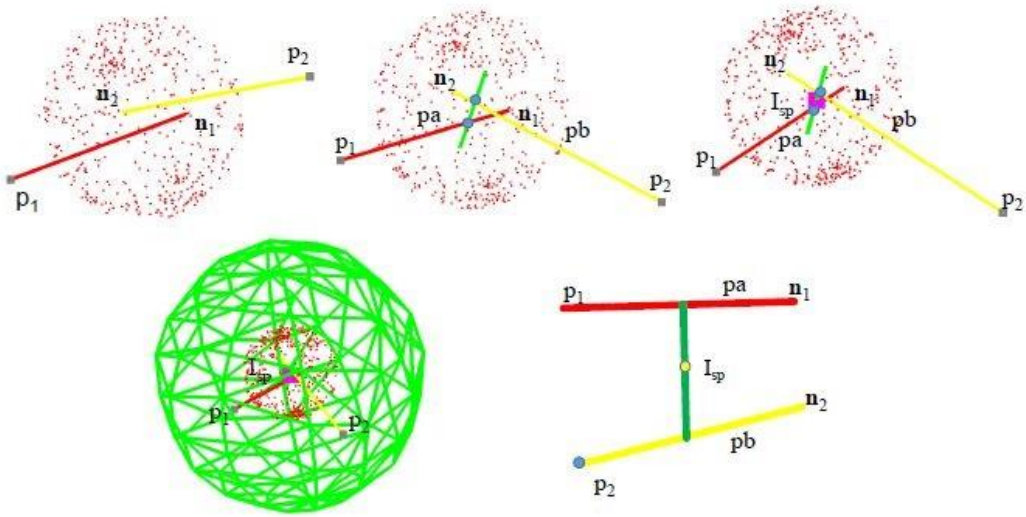

Рисунок 5. Нахождение центра и радиуса сферы по двум точкам и их нормалям

Построим скрещивающиеся прямые (p1,pa) и (p2,pb) в соответствии нормалям и найдем линию с кратчайшим расстоянием между ними. Центр этой линии и примем за центр сферы *c*. Найдем радиус

$$r = \frac{\left||p1 - c|\right| + \left||p2 - c|\right|}{2}$$

Так как точки мы выбираем случайным образом, стоит принять некоторую дистанцию *A*, на которую *p1* и *p2* могут отклоняться от найденного радиуса. Также перед началом вычислений стоит проверить, что угол между нормалями не превышает заданный угол *alpha*.

В это случае можно не строить сферу по какому-либо уравнению, достаточно проверять удаленность точек от центра сферы.

2.3.2 Описание генетического алгоритма

Генетический алгоритм – эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путем случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе. Отличительной особенностью генетического алгоритма является акцент на использование оператора «скрещивания», который производит операцию рекомбинации решений-кандидатов, роль которой аналогична роли скрещивания в живой природе. На Рис. 8 представлена блок-схема алгоритма.

В данном случае первоначальное поколение будет создаваться вокруг найденного примитива, а значения мутации будут несколько снижены, чтобы объект не отходил слишком далеко от оригинала. Это позволит более точно определить ориентацию той или иной фигуры, что должно перекрыть недостатки RANSAC.

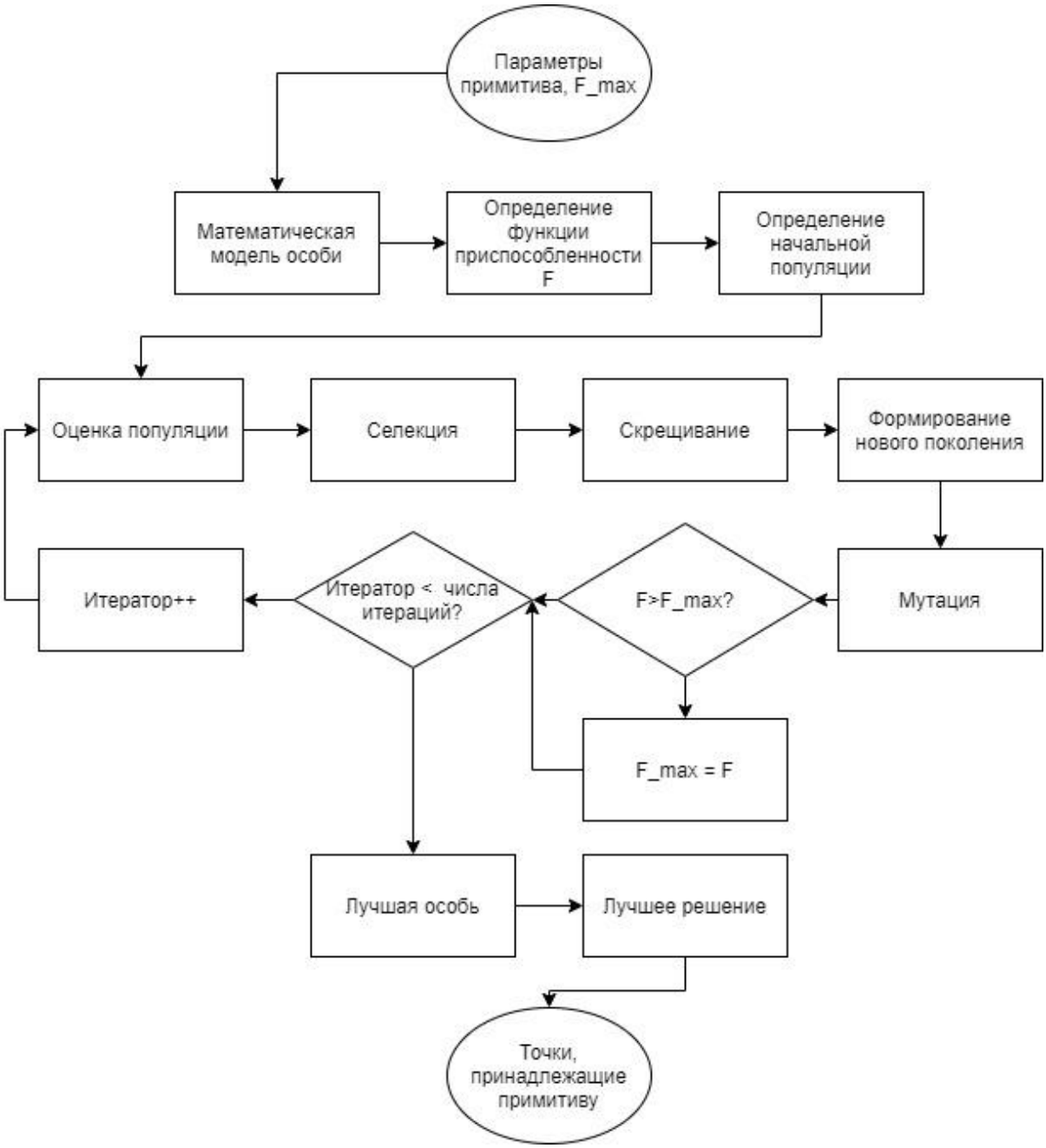

Рисунок 8. Блок-схема генетического алгоритма

В качестве генов для плоскости используются коэффициента A,B,C,D, в качестве генов для сферы ее центр и радиус. Функция приспособленности используется такая же как в RANSAC и имеет вид:

$$f = \frac{N_{inliers}}{N_{outliers} + 1},$$

После определения модели и функции можно перейти к инициализации начальной популяции размера S. Параметры моделей выбираются путем прибавления случайного числа в неком диапазоне к параметрам найденной RANSAC моделью

Оценка популяции и селекция

После инициализации каждую особь нужно оценить по определенной выше функции приспособленности. В результате нужно составить таблицу соответствий особи и ее значением. После этого нужно провести селекцию. Существует множество методов селекции, перечислим самые распространенные:

1) Рулетка – вероятностный способ отбора. По результатам оценки популяции, для каждой особи вычисляется значение вероятности попадания в следующее поколение.

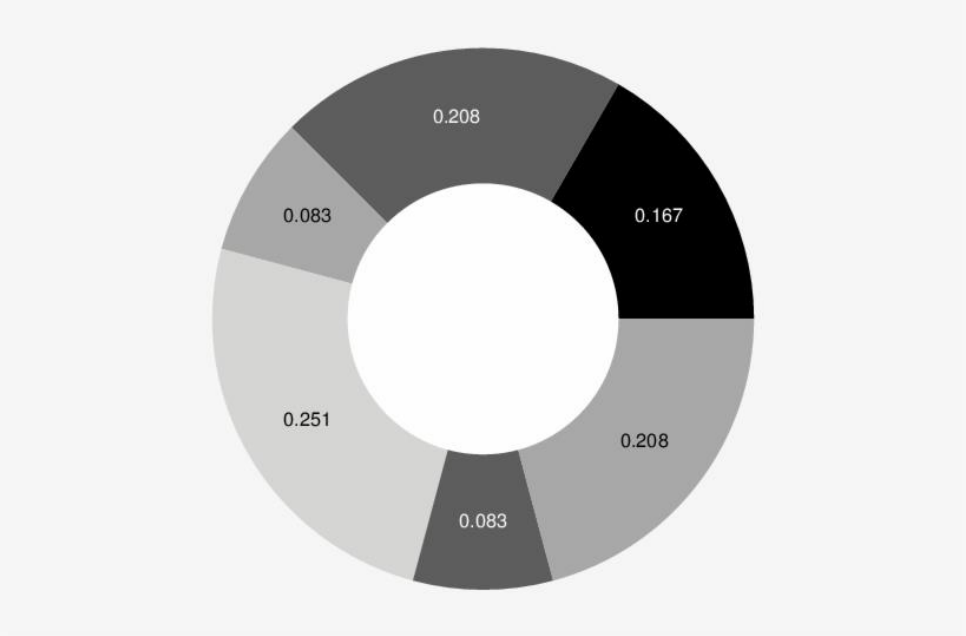

Рисунок 9. Вероятности пропорциональны значениям функции приспособленности

После этого для каждого члена популяции случайно выбирается число от 0 до 1. Сравнивая полученные значения с результатами кумулятивной суммы всех вероятностей, переставляется порядок особей в поколении, так, чтобы самыми первыми были те, кто попал в диапазон между первыми двумя элементами массива из кумулятивных сумм.

Такой метод позволяет разнообразить популяцию, что впоследствии приводит к большему числу вероятных решений и возможностям для мутации.

1) Отбор по рангу – метод, когда происходит сортировка особей по убыванию их значения приспособленности. Такой способ позволяет быстро оценить всю популяцию и не потерять ценные особи, однако может привести к малой вариативности возможных гипотез.

2) Турнирная селекция – особи разбиваются на «команды». Победитель состязаний из каждой «команды» попадает в следующее поколение.

3) Элитизм - Зачастую для получения наилучших параметров используются стратегии с частичным воспроизведением. Небольшая часть лучших особей из последнего поколения без каких-либо изменений передается следующему.

В данной работе был использован метод отбора по рангу. После сортировки особей, вторая половина удаляется. Их место занимает потомство.

Скрещивание и формирование нового поколения

Размножение в генетическом алгоритме требует для производства потомка нескольких родителей, обычно двух.

Можно выделить несколько операторов выбора родителей:

1. Панмиксия – оба родителя выбираются случайно, каждая особь популяции имеет равные шансы быть выбранной

2. Инбридинг – первый родитель выбирается случайно, а вторым выбирается такой, который наиболее похож на первого родителя

3. Аутбридинг – первый родитель выбирается случайно, а вторым выбирается такой, который наименее похож на первого родителя

Инбридинг и аутбридинг бывают в двух формах: фенотипной и генотипной. В случае фенотипной формы похожесть измеряется в зависимости от значения функции приспособленности (чем ближе значения целевой функции, тем особи более похожи), а в случае генотипной формы похожесть измеряется в зависимости от представления генотипа (чем меньше отличий между генотипами особей, тем особи сильнее похожи). Также существует несколько вариантов вероятностного скрещивания.

В работе был использован метод панмиксии. Скрещивание осуществляется путем сложения параметров родителей, но каждое слагаемое умножается на случайно сгенерированный весовой коэффициент в диапазоне от 0 до 1.

После того как скрещивание завершилось, потомки добавляются в конец списка, восполняя собой удаленные ранее особи.

Мутация

В генетическом алгоритме мутация – вероятностный процесс. Как правило, частота и сила мутаций определяется специальным параметром, который выбирает n-е количество параметров из параметров всех особей и случайным образом меняет их. Такой способ позволяет регулировать степень изменения генофонда популяции и изменить ее для получения лучшего результата.

Завершение работы алгоритма

После окончания мутаций, особи в популяции перемешиваются случайным образом, и цикл начинается заново. Чтобы не потерять хорошие гипотезы, нужно сохранять особи с самым высоким значением функции приспособленности и обновлять каждый раз, как только будет найдена новая гипотеза с лучшим значением.

2.3.3 Оценка работы эволюционного алгоритма

Результат работы эволюционного алгоритма можно оценить метрикой IoU, как и в задаче детекции объектов на изображении. Однако здесь вместо площади используется количество точек.

2.4 Заключение второй главы

3. Разработка программного обеспечения для анализа изображения с бортовой камеры автономного робота

3.1 Структура программного обеспечения

3.2 Используемые инструменты

3.3 Порядок работы с программным обеспечением

4. Проведение экспериментальных исследований

4.1 Результаты работы эволюционного алгоритма

4.2 Результаты работы нейро-эволюционного алгоритма на данных с бортовой камеры

4.3 Заключение четвертой главы

5. Организационно-экономическая часть

Заключение

Используемая литература

Приложения