Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт Кибернетики
Направление подготовки Мехатроника и робототехника
Кафедра интегрированных компьютерных систем управления

БАКАЛАВРСКАЯ РАБОТА

Тема работы
Система оценки преодолимости мобильным роботом участков физически неоднородной среды

УДК 621.865.8-2:004

Студент

Группа	ФИО	Подпись	Дата
8Е21	Стучков Антон Витальевич		

Руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Старший научный сотрудник международной лаборатории систем технического зрения НИТГУ	Андраханов Анатолий Александрович			

КОНСУЛЬТАНТЫ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата

ДОПУСТИТЬ К ЗАЩИТЕ:

Зав. кафедрой	ФИО	Ученая степень, звание	Подпись	Дата

Томск – 2016 г.

РЕФЕРАТ

Выпускная квалификационная работа 78 с., 75 рис., 25 табл., 26 источников, 1 прил.

Ключевые слова: проходимость, преодолимость, мобильный робот, МГУА, классификатор поверхностей, оконтуривание, неоднородная среда.

Цель работы – разработка системы оценки преодолимости мобильным роботом участков физически неоднородной среды.

В процессе работы проводились: проведен обзор и систематизация известных технических решений, разработка структурной схемы системы, разработка алгоритмов идентификации параметров локальных участков неоднородной среды, проведение экспериментальных исследований.

В результате исследования были получены: зависимость качества классификатора от параметров сети и наполнения выборок.

Область применения: мобильная робототехника Outdoor-типа (в т.ч. роботы-разведчики, роботы-исследователи территорий, планетоходы, беспилотные транспортные средства).

Экономическая эффективность/значимость работы: экономическая значимость работы связана с повышением уровня автономности функционирования мобильных роботов в условиях пересеченной местности.

В будущем планируется улучшение точности работы классификатора за счет учета внутренних параметров робота, совершенствование методов обработки изображений.

Оглавление

Введение

Одной из проблем современной мобильной робототехники является движение и выполнение специализированных (разведывательных, исследовательских, поисковых) задач в условиях пересеченной местности. Такую среду можно назвать физически неоднородной, и при прохождении через нее робот встречает участки (снег, галька, болото, луг, песок и т.д.), попадание на которые может привести к невыполнению задачи, остановке, буксированию, поломке, сбою навигационных систем робота. Например, робот-разведчик может столкнуться с заболоченной территорией, планетоход с полупустынной. Следовательно, необходимой задачей робота является определение преодолимости участков, лежащих на его пути. Таким образом, разработка системы оценки преодолимости участков физически неоднородной среды является актуальной. В данном направлении работают исследователи из Государственного Университета Теннеси, Южного Федерального университета, Университета Токио, Корейского Морского Университета. На сегодняшний день популярной тенденцией в решении задачи определения проходимости является построение 3D-карт местности, анализ визуальных паттернов участков изображения и некоторые другие. Потенциальный экономический эффект такой системы состоит в увеличении степени автономности мобильных роботов в условиях физически неоднородной среды.

Глава 1. Обзор известных технических решений определения проходимости участков среды посредством системы технического зрения

Актуальной проблемой мобильной робототехники является определение проходимости естественной природной среды и выбор параметров движения с учетом данной проходимости. Данная проблема позволяет выделить задачи, перечисленные ниже.

1) Получение информации о среде и о состоянии мобильного робота (далее МР). В качестве датчиков получения информации о среде могут использоваться: стерео и глубинные видеокамеры; лидары; инфракрасные датчики; лазерные дальномеры.
2) Обработка полученной информации и составление модели поверхности/пространства вокруг МР. В составленной модели различные участки поверхности оцениваются и классифицируются по степени проходимости.
3) Выбор параметров движения робота.

Далее подробно рассмотрены методы обработки внешней информации и построения модели, а также проведен сравнительный анализ данных методов. В зависимости от поставленных задач и имеющихся ресурсов проходимость среды может вычисляться различными способами. Исследователи могут использовать понятие индекса проходимости (варианты записи: τ, Traversability Index, TI), а также заменять его другими (индекс неровности, карта проходимости, проходимость, состояние поверхности, коэффициент риска, коэффициент проходимости), либо классифицировать изображения без употребления подобного понятия. Среди рассмотренных методов основное внимание уделено методам с использованием систем технического зрения (СТЗ), поэтому методы оценки проходимости среды рассматриваются неразрывно с методами обработки изображений. В данном обзоре методы оценки преодолимости представлены авторами, а не

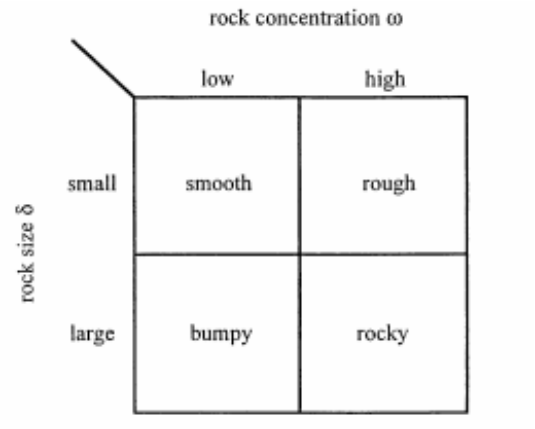
названиями методов, так как авторы используют комбинации и чередуют множество методов в рамках решения задач.

1. H. Seraji, Gennery, D. [1,2]

В методе [1,2] данные с СТЗ представляют собой набор неравномерно расположенных точек, каждая из которых описывается вектором $\mathsf{r} = [x\ y\ z]^T$, где x,y,z – координаты точки, и ковариационной матрицей этого вектора (Σrr). Для оценки параметров плоскости необходимо равномерно расположить точки (путём сглаживания и интерполирования), после этого оценить наклон и неровность поверхности. Данная задача может быть решена с помощью подгонки (fitting) (путём метода взвешенных наименьших квадратов (weighted least squares)) плоскостей до небольших площадей вокруг каждой желаемой выходной точки, используя коэффициент дисперсии каждой площади ($1/\sigma_i$) для определения высоты и наклона, и остатков от подгонки (residuals of the fit) для оценки неровности [2]. Далее чёткие значения (crisp values) неровности (roughness) и наклона (slope) подвергаются фаззификации для нахождения степени принадлежности в сопровождающих их нечетких наборах [1].

Существует другой способ вычисления неровности [1]: системой технического зрения проводятся измерения размера кочек (rock size, β) и их плотности (rock densitiy, ω). Оба параметра представляются в виде нечётких лингвистических наборов {большой, маленький} для размера кочек и {низкий, высокий} для концентрации кочек. Неровность представляется в виде нечёткого лингвистического набора, формируемого согласно правилам, представленным на рисунке 1, через нечеткие множества β и ω: {гладкий (smooth), неровный (rough), ухабистый (bumpy), скалистый (rocky)}.

Таблица 1 – Правила для формирования нечёткого множества оценки неровности



Таким образом, на первом этапе определяются нечёткие множества наклона и неровности.

Далее данные используются для подсчета индекса проходимости согласно правилам, представленным в таблице 2. Нечеткое множество индекса проходимости вычисляется умножением двух нечетких множеств наклона поверхности и неровности.

terrain roughness β

terrain slope α	smooth	rough	bumpy	rocky
low	high	medium	low	poor
medium	medium	medium	low	poor
high	low	low	poor	poor
very high	poor	poor	poor	poor

Таблица 2 – Правила формирования нечёткого множества индекса проходимости

Далее эта информация подается на дефаззификационную стадию, после чего осуществляется классификация поверхности согласно правилам соответствия (1).

POOR $\tau \rightarrow$ HIGHLY-IMPASSABLE TERRAIN.

LOW $\tau \rightarrow$ IMPASSABLE TERRAIN.

MEDIUM $\tau \rightarrow$ PASSABLE TERRAIN.

HIGH $\tau \rightarrow$ HIGHLY-PASSABLE TERRAIN. (1)

В дальнейшем значение проходимости поверхности определяет правила движения робота.

2. Sanjiv, S. [3]

Первый шаг - разбивание поверхности на патчи (участок размером 1,25м^2), а патчей на суб-патчи (25см^2). Проводится оценка угла крена (roll), угла тангажа (pitch) и неровности (roughness) каждого патча. Крен и неровность оцениваются с помощью метода наименьших квадратов (МНК). Метод заключается в подгонке (fit) плоскости под стерео-данные на каждом патче, что осуществляется путём нахождения такого параметра b, при котором сумма квадратов остатков от подгонки (residuals of the fit) будет минимальной (2) [14].

$$b = \arg\min \sum_{t=1}^{n} (y_t - f(x_t, b))^2 \quad (2)$$

Оценка неровности происходит следующим способом: для каждого суб-патча и патча считается χ^2-распределение остатков от подгонки (chi-squared residual of the fit). Максимальный остаток на обоих уровнях берётся как измеренная неровность. На выходе после первого этапа представлены значения покатости, высоты и неровности в диапазоне [0,1]. Общая добротность патча определяется как минимум этих трёх измерений. Также считается статистическая достоверность патча как функция числа и распределения точек (распределение зависит от количества достоверных значений суб-патчей). Статистическая достоверность предотвращает появление испорченных данных.

В результате формируется карта добротности, где для каждого патча определена проходимость.

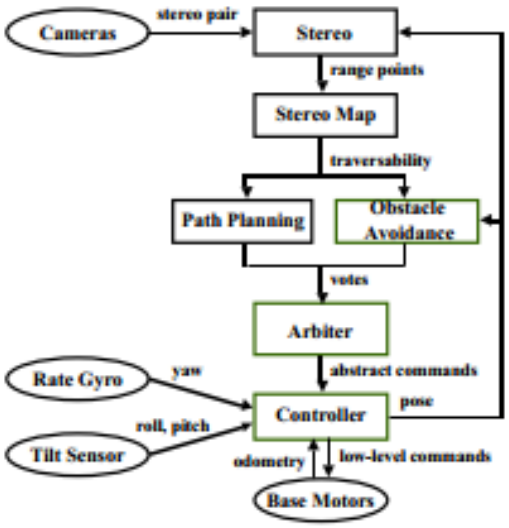


Рисунок 1 – Алгоритм метода разбивания на патчи

3. Shirkhodaie, A. [4]

Метод, предложенный A. Shirkhodaie в работе [4], предполагает анализ образа текстуры поверхности объекта (image-based object surface texture analysis). Изображение представлено как матрица интенсивностей пикселей ($I_{i,j}$) (256 оттенков серого от 0 до 255). Вариации в интенсивности пикселей изображения создают определенные повторяющиеся паттерны, называемые текстурой изображения. Для классификации условий поверхности применяется статистический анализ. Автор [4] вводит $p_s(i,j)$, $p_d(i,j)$ как сумму и разницу интенсивностей пары пикселей изображения i и j. Данные параметры используются для формулировки вектора суммы интенсивностей P_s и вектора разности интенсивностей P_d (1), где $h(x)$ – гистограмма x, α – максимальный диапазон изменения оттенка, i,j – координаты пикселя.

$$P_s = \frac{\hbar[p_s(i,j)]}{2\alpha} \qquad P_d = \frac{\hbar[p_d(i,j)]}{2\alpha} \tag{3}$$

Далее вводятся следующие параметры текстуры: вариативность текстуры изображения (Image Texture Variance) (2); контраст текстуры изображения (normalized image texture Contrast) (3); энергия текстуры изображения (Image Texture Energy), показывающая текстурную однородность (4); энтропия текстуры изображения (Image Texture Entropy), демонстрирующая наполненность информацией изображения (5).

$$V = \sum_{i=0}^{2\alpha} (i-b)^2 . P_s(i) \; + \sum_{i=0}^{2\alpha} (i-\alpha)^2 . P_d(i) \tag{4}$$

где $b = Max(P_s)$.

$$C = \frac{1}{2\alpha} \sum_{i=-\alpha}^{\alpha} i^2 . P_d(i+\alpha) \tag{5}$$

$$E = \frac{1}{2\alpha} \left(\sum_{i=0}^{2\alpha} P_s^2(i) + \sum_{i=0}^{2\alpha} P_d^2(i) \right) \tag{6}$$

$$En = \sum_{i=0}^{2\alpha} P_s(i).\log(P_s(i)) + \sum_{i=0}^{2\alpha} P_d(i).\log(P_d(i)) \tag{7}$$

После введения параметров изображения участки изображения классифицируются несколькими способами. Первый – эвристический способ на основе четко заданных правил классификации. Сперва изображения разделяются на суб-кадры (sub-terrain), далее параметры текстуры считаются для каждого суб-кадра, после чего из этих параметров формируется матрица характеристик местности (Terrain Feature Matrix, M_{fv}). В M_{fv} каждый элемент матрицы представляет собой вектор параметров суб-кадра. Применение 2D фильтра Калмана к матрице $M_{fv,}$ позволяет получить единые оценки суб-кадров. Для оценки проходимости используется эвристический классификатор, представляющий из себя детерминированную модель знаний (knowledge driven deterministic model). Она имеет набор предопределенных порогов для классификации различных параметров суб-кадров и придания им одной из характеристик среды (табл. 3). В результате на выходе формируется

конечная карта, состоящая из суб-кадров, каждый из которых классифицирован по возможности прохождения.

TERRAIN TEXTURE MEASURES	TERRAIN CONDITION
If (C > 0.01 Or En > 20.0)	Sand Piles
If (C >= 0.01 And C < 0.1) Or If (En>1.50 And En<=25.0)	Smooth Terrain
If (C >= 0.10 And C < 0.2) Or If (En<=1.5 And En > 0.60)	Rough Terrain
If (C >= 0.20 And C < 0.35) Or If (En<=0.6 And En > 0.1)	Rocky Terrain
If (C >=0.35 And C < 0.80) Or If (En<=0.10)	Very Rocky Terrain
If (C >=0.80)	Large Rocks

Таблица 3 – Правила классификации для суб-кадра

Второй способ использует классификатор на основе HNN (нейронная сеть Хэмминга), которая реализует классификацию на основе наименьшей погрешности для бинарных входных векторов, где расстояние Хэмминга (число битов, которые отличаются между двумя соответствующими входными векторами) определяет погрешность (рисунок 3).

Изображения сохраняются с помощью вычисления соответствующей весовой матрицы. Поэтому, начиная с произвольной конфигурации, память закрепляется на сохраненное изображение, которое ближе к первоначальной конфигурации с точки зрения расстояния Хэмминга. Таким образом, при наличии неполной или поврежденной версии текущего изображения, сеть способна вспомнить соответствующий оригинал изображения. Сеть имеет три слоя. Слой входов представлен множеством узлов в количестве, соответствующему количеству отдельных бинарных признаков (separate binary features) учебного образца. Далее располагается слой Хопфилда, где количество узлов определяется количеством категорий. В выходном слое число узлов соответствует слою категорий.

Обучение сети Хэмминга требует однопроходный обучающий набор (single-pass training set). Желаемый обучающий образ запечатлён на входном слое, в то время как желаемый класс, которому принадлежит вектор, запечатлён на выходном слое. Выход содержит только категорию выхода, к которой принадлежит входной вектор. Рекурсивная природа слоя Хопфилда обеспечивает средства регулировки весов всех связей. Авторами [4] была применена пересмотренная версия HNN, которая изначально конвертирует реальные значение свойств текстуры суб-кадра в бинарный входной вектор.

Суб-поверхностный вектор признаков (feature vector) содержит следующие величины: энергия, контраст, дисперсия и размер BLOB (Binary Large Object). Для обучения используется большое количество суб-местностей: в работе [4] обучение на основе 250 суб-изображений осуществляется в 200 итераций.

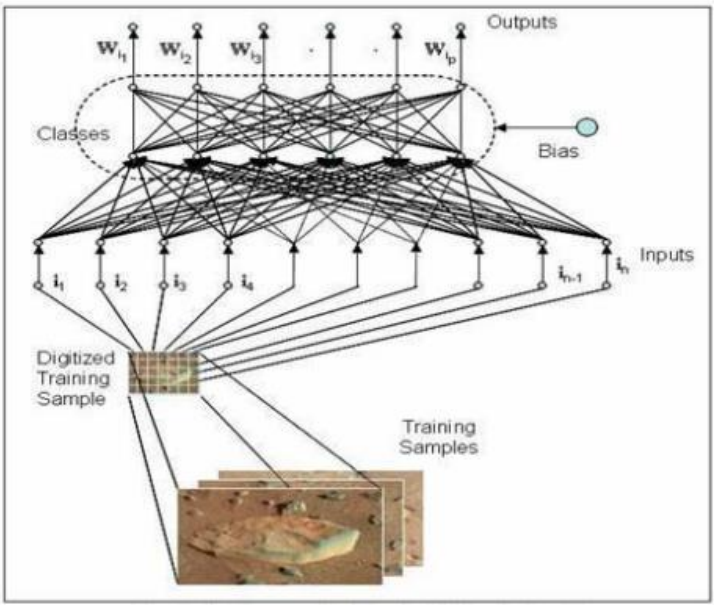


Рисунок 2 – Трёхслойная нейронная сеть Хэмминга, используемая для классификации изображений

4. Jin, G. [5]

Данный метод заключается в преобразовании цифровой модели поверхности (Digital Terrain Model, DTM) в карту проходимости (Traversability Map, TM) (рис. 3).

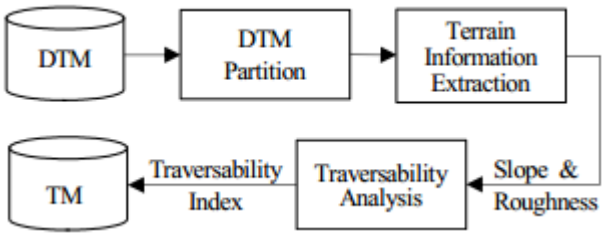


Рисунок 3 – Этапы метода преобразования DTM

DTM строится на основе данных стереокамеры или лазерных дальномеров. Модуль разделения DTM получает выборку данных с датчика, и делит XY плоскость DTM на равные промежутки W×W-сетки, состоящие из клеток. Модуль извлечения информации о поверхности использует интерполирование (curve fitting) и метод треугольной призмы на основе фракталов (fractal-based triangular prism method), имеет на выходе наклон и неровность отдельных участков. Для определения наклона поверхности используется стерео-камера или лазерные дальномеры, удаленные друг от друга. Применяя метод наименьших квадратов, составляется математическая модель, представляющая собой описание поверхности. Расчёты согласно методу наименьших квадратов приводят к выводу формулы индекса наклона (α). Неровность поверхности определяется схожим образом.

Модуль анализа проходимости оценивает проходимость путем соединения извлеченной информацией и нечеткого логического вывода для построения карты проходимости (traversability map).

5. EL-KABBANY, A. [6]

В некоторых работах индекс проходимости заменяется на индекс неровности (Roughness Index, (RI)) [6]. Данный индекс выводится из упрощенной модели поверхностной структуры (Рис. 4).

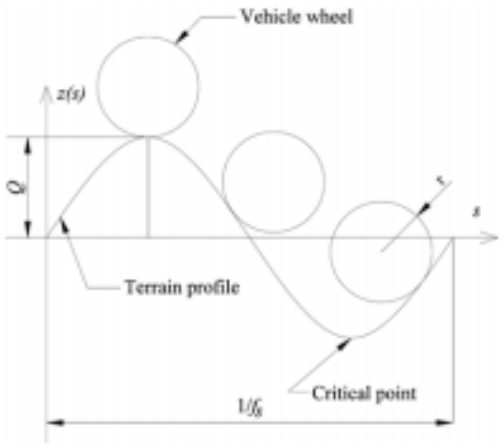


Рисунок 4 – Модель структуры поверхности, используемая для вычисления индекса неровности. Q – максимальная амплитуда профиля поверхности, r – радиус колеса, $1/f_s$ – период колебаний

Согласно данной модели, вводятся гармоническая частота поверхности, максимальная рассматриваемая частота и максимальная амплитуда поверхности. Исходя из вышеприведенных величин находится максимальная приемлемая частота неровности поверхности и грубость поверхности. Предложенный индекс неровности основан на стандартном отклонении высотного компонента точек поверхности. В результате, RI - это число в диапазоне от 0 до 1, где 0 соответствует наиболее неровной поверхности (невозможной для прохождения роботом), а 1 – наиболее гладкой поверхности. Данные расчёты позволяют учитывать аналитическую модель робота при планировании пути. Физическая реализация определения индекса неровности представлена следующей последовательностью. Массив светодиодов выпускает импульсы инфракрасного излучения, оптический фильтр на камере пропускает только длину волны светодиодов. Массив датчиков позади фильтра получает отражение инфракрасного света от объектов в целевом диапазоне камеры. Расстояние между камерой и

поверхностью считается как функция от времени возврата инфракрасного импульса. В результате поверхность воспринимается как облако точек в сферической координатной системе, имеющую центр координат в фокальной точке. Далее триангуляция Делоне используется для создания сетки точек поверхности. Поверхность представляется в виде набора точек в пространстве $[x, y, e]^T$, где e - высота объекта. Исходя из данных величин вычисляется стандартное отклонение высоты (σ) и коэффициент неровности: RI = 1- 2σ. Далее на основе индекса неровности строится карта поверхности и вычисляются скорости, необходимые для прохождения пути.

6. Tanaka, Y. [7]

В методе [7] (рис. 5) данные с лазерного дальномера используются для построения карты высот.

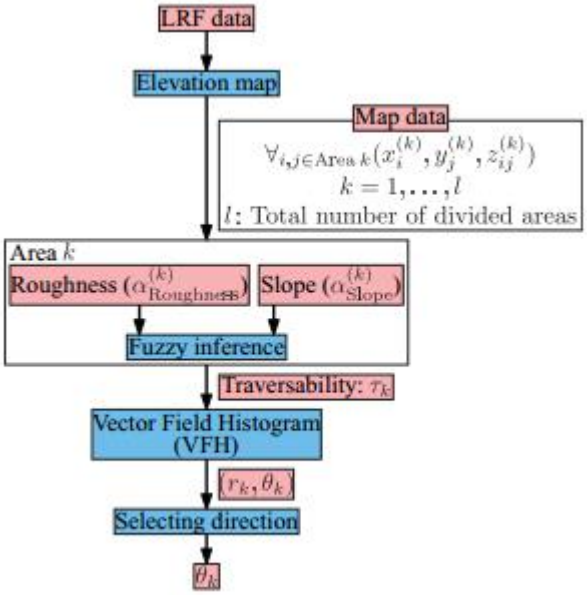


Рисунок 5 – Алгоритм метода с радиальным разделением поверхности

Все элементы карты высот состоят из трёх координат. x_i и y_j соответствуют клетке (i,j). z выражает высоту точки (x_i, y_j). Значение высоты (elevation value) определяется согласно следующему уравнению:

$$e_{ij}^{(t)} = \begin{cases} z_{ij}^{(t)} & (e_{ij}^{(t-1)} < z_{ij}^{(t)}) \\ e_{ij}^{(t-1)} & (\text{otherwise}) \end{cases} \quad (8)$$

где $e^{(t)}_{ij}$ значение высоты клетки (x_i, y_j) во время t; $z^{(t)}_{ij}$ – принимаемая датчиком (sensed) высота клетки (x_i, y_j) во время t. Таким образом, если новое считываемое значение высоты клетки выше чем предыдущее, оно заменяется на новое. Следующий этап – разделение карты высот на прямоугольные области. Прямоугольные области выстраиваются радиально по отношению к центру робота (RCP, robot center point) (рис. 6).

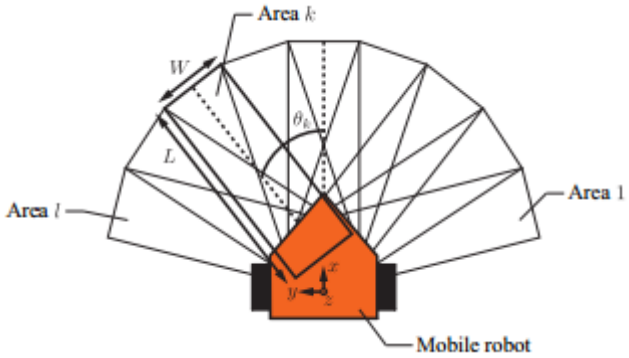


Рисунок 6 – Разделение плоскости вокруг робота на прямоугольные области.

L – длина области, W – ширина, θ k – угол вектора прямоугольной области относительно x координаты робота.

Далее рассчитывается наклон ($\alpha_{Slope}^{(k)}$) и неровность ($\alpha_{Roughness}^{(k)}$) для каждой области. Неровность берётся как стандартное отклонение высоты в каждой прямоугольной области.

$$\alpha_{\text{Roughness}}^{(k)} = \sqrt{\frac{1}{N_k} \sum_{i,j \in \text{Area } k} (z_{ij}^{(k)} - \overline{z_k})^2},$$

$$\overline{z_k} = \frac{1}{N_k} \sum_{i,j \in \text{Area } k} z_{ij}^{(k)}, \quad (7)$$

где N_k – число клеток в каждой прямоугольной площади, $z\,(k)_{ij}$ высота клетки (i, j) в прямоугольной области k, z_k средняя высота в каждой прямоугольной области.

Наклон считается из вектора нормали к подогнанной плоскости и из вектора направления прямоугольной области. Подгонка плоскости (plane fitting) считается исходя из информации о высоте для прямоугольной области.

Индекс проходимости определяется из нечетких множеств неровности и наклона аналогично [1]. Значение индекса проходимости используется для построения векторной гистограммы, которая показывает значение риска для каждой прямоугольной области (рис. 7).

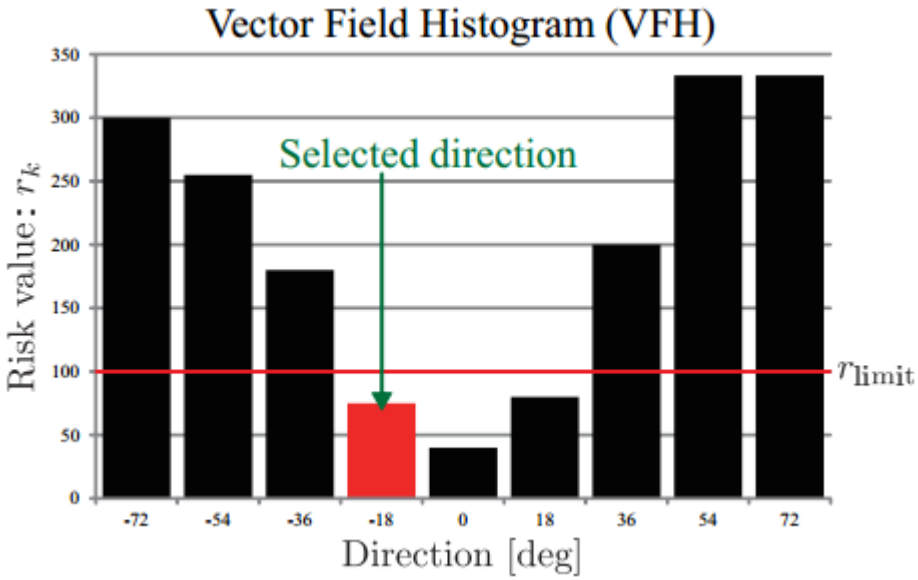


Рисунок 7 – Пример векторной гистограммы

7. Cappalunga, A. [8]

Данные с 2 AVT Stingray FireWire цветовых камер представляются в виде 3D-точек с тремя координатами ($p_{x,y,z}$). Пространство делится на продольные и боковые срезы *(slices)* фиксированной ширины, высоты и длины. Каждая 3D-точка проецируется на соответствующие 2D опорные плоскости (reference planes) P_i (продольный срез*)* и P_j (боковой срез), таким образом захватывается профиль, связанный с каждым срезом. Профиль включает в себя все точки среза, вне зависимости от того, что они представляют (местность, объект или шум). В результате формируется 2D проекция 3D-пространства.

Следующий шаг – оценка наклона каждого профиля. Здесь используется метод Оптимизации муравьиных колоний (Ant Colonies Optimizations),

название которого вдохновлено коммуникационной системой муравьев, которые всегда находят кратчайший путь от гнезда до еды. Здесь используется распределенная мета-эвристика (distributed meta-heuristic) для решения проблем комбинаторной оптимизации. Оценка наклона происходит следующим образом: для каждой 2D проекции строится «колония муравьев», перемещаясь от пикселя к пикселю, пытаясь найти оптимальный наклон кривой для каждой проекции.

Реализация оптимизации муравьиной колонии: проблема нахождения оптимальной проекции наклона формализуется в виде проблемы *кратчайшего пути через граф*. Авторы работы [8] задали начальные и конечные пиксели, правила движения и программно смоделировали движение «муравьев». В результате, для каждой проекции создана «колония муравьев», то есть точки, представляющие соответствующий наклон проекции.

Далее 3D-сетка строится путём аппроксимации полученных точек полилинией (рис. 8). Точки могут быть классифицированы на неискаженные (Inliers) и выбросы (Outliers), что даёт информацию о проходимости поверхности.

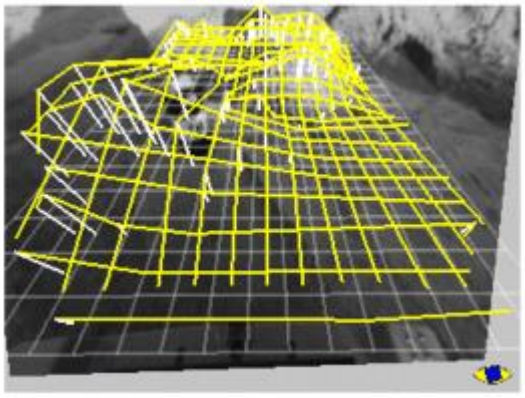


Рисунок 8 – Наилучшие пути «муравьев», аппроксимированные жёлтой полилинией

8. Moghadam, P. [9]

Первый этап – предварительная обработка изображения, где применяется фильтр низких частот. Далее обеспечивается согласование изображений стереопары (stereo matching) и создание карты диспаратности (disparity map) (рис. 9) путём корреляционного метода суммы абсолютных разностей (Sum of Absolute Differences (SAD) correlation method).

Рисунок 9 – Оригинальное изображение (слева) и диспаратное изображение (справа)

Далее применяется метод оценки плоскости земли (ground plane estimation method). Случайно выбираются 3 точки из ближней области диспаратного изображения. Затем перебираются близлежащие точки (supporting points) и та плоскость, у которой максимальное число близлежащих точек, определяется как основная.

Далее основная плоскость земли совмещается с диспаратным изображением для обнаружения препятствий, после чего происходит разбивание изображения на патчи, каждый из которых обозначается как препятствие либо как плоскость земли. Далее у патчей определяются характерные признаки (цвет, текстура). Для определения текстуры был использован банк фильтров Габора.

Далее для обучения робота используется метод опорных векторов (Support Vector Machines), что позволяет классифицировать изображения на наличие препятствий вне стерео-диапазона. Полный алгоритм представлен на рисунке 10.

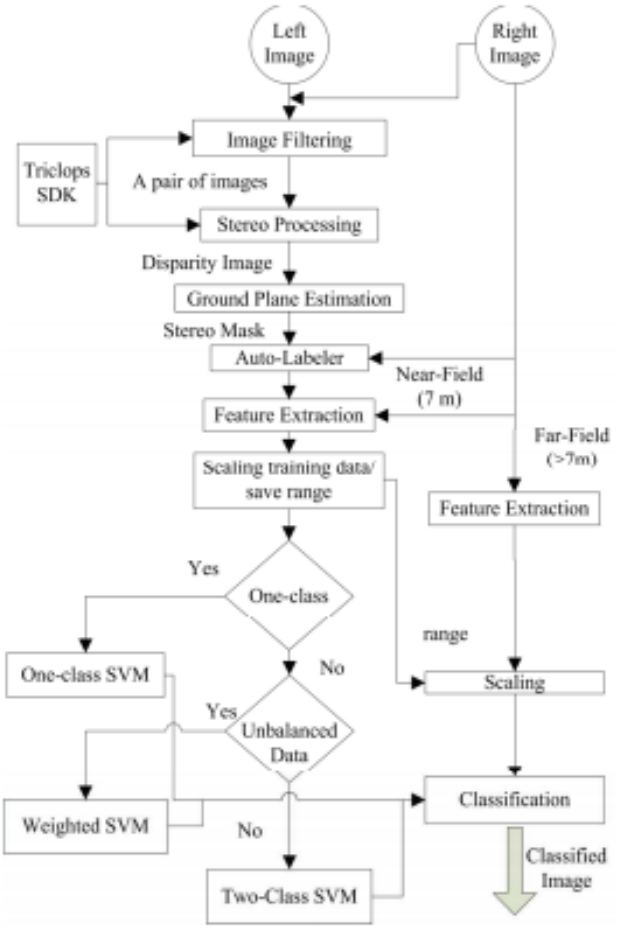


Рисунок 10 – Алгоритм метода диспарации изображения

9. Wang, W. [10]

Следующий метод [10] примечателен использованием JRDE (Joint Reliability based Disparity Expansion) алгоритма для конструирования 3D-модели. Метод реализуется в несколько этапов. Подробный алгоритм метода изображен на рисунке 11. Восприятие изображения начинается с получения 3D-распределения от направленных вниз стереокамер. Далее с помощью оценки несоответствия 3D-координат иерархически стоится цифровая карта высот (Digital Elevation Map, DEM) на основе 3D модели местности, полученной на предыдущем этапе. После этого конечная карта проходимости создается ограничением высот (оценка проходимости проходит в соответствии с четырьмя категориями: проходимый, непроходимый, неизвестный, неопределенный; оценка базируется на заданной пороговой

высоте) и оценкой дополнительного наклона (путём определения градиента поверхности), что, в целом, даёт индекс проходимости поверхности.

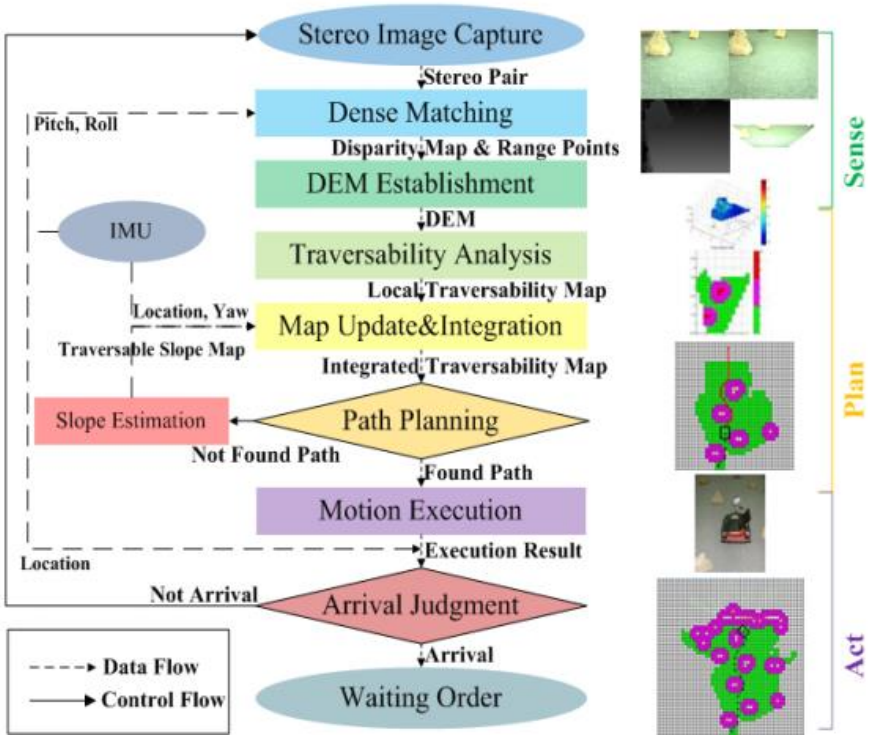


Рисунок 11 – Алгоритм метода JRDE

10.Чернухин, Ю. В. [11, 13]

В методе [11] локальная карта проходимости строится по изображению с камеры, преобразованному к оттенкам серого цвета, с помощью модели рецептивных полей ганглиозных on- и off- клеток сетчатки.

Клеткой с on-центром называется та, которая возбуждается, если пятно света находится внутри округлой центральной области, и вытормаживается, если пятно света попадает на кольцеобразную область с определенным внутренним и внешним диаметром. Клетка с off-центром характеризуется противоположным поведением – возбуждается, если пятно находится на периферии и вытормаживается, если пятно попадает на округлую центральную область [13].

Данные от датчика глубины представляются в графическом виде в форме гистограммы, а затем для каждого значения расстояния до объекта вычисляется цветовое значение пикселя:

$$Pi = 255 * (1 - Hi / N), \quad (8)$$

где P_i – цветовое значение пикселя для расстояния i мм, H_i – выборка элементов
гистограммы для расстояния i мм, N – общее число элементов гистограммы.

На полученном изображении производится выделение объемных объектов с помощью фильтра «Разница по Гауссу», соответствующего ответам ганглиозных on-клеток сетчатки. Фильтрация полученного изображения аппроксимацией математической модели рецептивного поля ганглиозных off-клеток сетчатки позволяет сформировать радиальные градиентные области вокруг каждого препятствия.

Для построения локальной карты проходимости производится комплексирование изображений:

$$P_{ij} = \begin{cases} Z, если\ I_{ij} \neq 0, \\ z_{ij}, если\ K_{ij} \neq 0, \\ 1, если\ K_{ij} = 0\ и\ I_{ij} = 0, \end{cases} \quad (9)$$

где P_{ij} – значение узла в локальной карте проходимости, Z – специальное значение,
обозначающее непроходимый участок, z_{ij} – коэффициент проходимости, I_{ij} – значение пикселя изображения с выделенными объемными объектами, K_{ij} – значение
пикселя изображения с коэффициентами проходимости.

Результат построения карты представлен на рисунке 12.

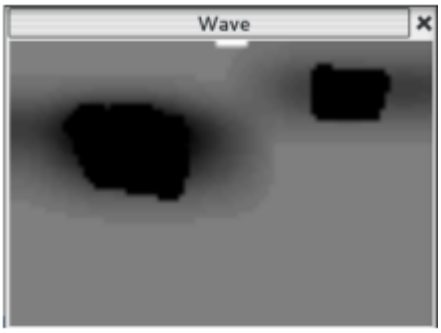


Рисунок 12 – Графическое представление карты проходимости

11.Dargazany, A.[12]

Согласно методу генерации облака точек [12], анализ проходимости проходит по следующей последовательности (рис. 13):

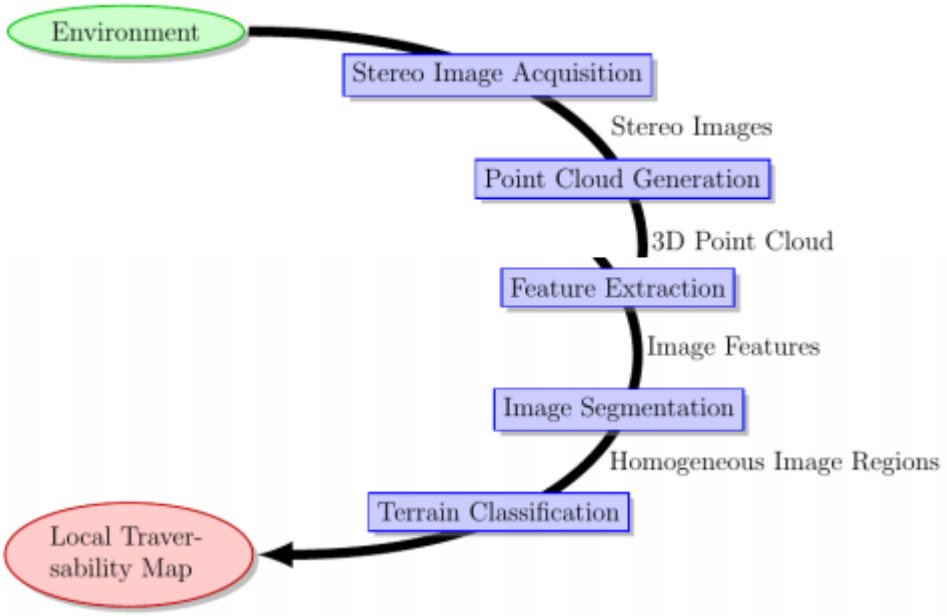


Рисунок 13 – Метод, использующий формирование облака точек

Данный метод состоит из двух основных шагов: генерация облака точек (point cloud generation) и анализ проходимости с использованием облака точек.

Облако точек может быть сгенерировано стереокамерой, RGB-D камерой (Xtion, Kinect) или ToF (Time of flight) камерой. Рассмотрим этапы генерации облака точек.

1) Получение стерео изображения.
2) Стерео калибровка и устранение шумов. Офлайн калибровка выполняется для очистки стереоизображений (stereo image rectification) с использованием инструментов стерео-калибровки библиотеки OpenCV.
3) Согласование изображений стереопары. Для генерации облака точек согласование применяется в стереоизображениях для обнаружения соответствующих пикселей на левом и правом изображениях с целью применения триангуляции. В результате генерируется карта диспаратности (disparity map). Пример метода: Block-based stereo matching (BB), Adaptive Cost - 2-Pass Scanline Optimization(ACSO).
4) 3D реконструкция. Составляется из карты диспаратности.

Следующий шаг – выделение существенных признаков (Feature extraction). Производится путем извлечения нормалей на основе пикселей (pixel-based normals) и текстур на основе пикселей (pixel-based textures) из сгенерированного облака точек окружающей среды. Существует три пути подсчета нормалей.

1) Метод ковариационной матрицы: создает 9 изображений для вычисления нормали к определённой точке из ковариационной матрицы локальной окрестности этой точки. С помощью этого метода можно оценить неровность поверхности.
2) Метод среднего 3D-градиента: Создает 6 встроенных изображений для вычисления сглаженных версий горизонтальных и вертикальных 3D-градиентов и вычисляет нормали при помощи перекрёстного произведения (cross-product) между этими двумя градиентами.
3) Метод изменения средней глубины. Создает только один целостный образ и вычисляет нормали от средних изменений глубины.

Авторы [12] использовали первый метод, вычислив неровность.

Далее следует сегментация изображения. Производится путём сбора и обнаружения всех поверхностей, используя извлеченный ландшафт. Это осуществляется путем технологии сегментации, которая преобразует сгенерированное облако точек в набор суперпикселей поверхности. Разница между нормалью к точке n_p и соседней нормалью n_{p2} должна быть меньше значения неровности (здесь α_r):

$$\vec{n_{p_1}} \bullet \vec{n_{p_2}} \leq \cos(\alpha_r) \tag{10}$$

Последний шаг – классификация поверхности, осуществляется в два этапа.

1) Оценка нормалей к поверхности суперпикселей. Сегменты (все обнаруженные суперпиксельные поверхности) анализируются методом главных компонент (Principal Component Analysis, PCA). Каждый сегмент аппроксимируется до плоскости.
2) Анализ проходимости с использованием плоскостей суперпикселей. Все оцененные плоскости суперпикселей анализируются на основе их наклона и шага.

   Сперва оценивается гравитационная нормаль:

$$\vec{g_c} = \vec{g} \bullet \cos(a_w + a_r) \tag{11}$$

   Наклон (α_{max}) вычисляется исходя из следующего неравенства:

$$\vec{n_p} \bullet \vec{g_c} \leq \cos(\alpha_{max}) \vec{n_p} \bullet \vec{g_c} \leq \cos(\alpha_{max}) \tag{12}$$

Исходя из наклона определяется индекс проходимости среды для каждого сегмента: проходимый, полу-проходимый и непроходимый.

В таблице 4 представлен сравнительный анализ рассмотренных методов, их достоинства и недостатки. Серым цветом выделены области, по которым не удалось найти информацию.

Таблица 4. Сравнительный анализ методов определения проходимости среды

Автор метода	**Используемые датчики**	**Преимущества**	**Недостатки**

H. Seraji, Gennery, D.	Стерео СТЗ	Не требуется априорных знаний о среде; Простое и интуитивное использование нечётких множеств [1];	При расчете индекса проходимости не используются геометрические параметры робота и его кинематическая схема.
Sanjiv, S.	Стерео СТЗ	Статистический анализ учитывает шум датчика; Низкая ошибка метода счисления пути (dead-reckoning error); Двухуровневый подход к навигации (локальный и глобальный) увеличивает скорость обработки информации [3];	При расчете индекса проходимости не используются геометрические параметры робота и его кинематическая схема.
Shirkhodaie, A. (общий)	Стерео СТЗ	За счёт заранее заданных параметров текстуры вычислительные требования сведены к минимуму; Параметры текстуры дают надежную статистическую оценку [4];	При расчете индекса проходимости не используются геометрические параметры робота и его кинематическая схема.
Shirkhodaie, A. (эвристический классификатор)	Стерео СТЗ	Высокая скорость классификации [4];	Низкая производительность (% успешных попыток определения проходимости) [4];
Shirkhodaie, A. (классификатор на основе HNN)	Стерео СТЗ	Высокий процент успешных попыток определения проходимости [4];	Критичность правильного набора обучающих шаблонов [4]
Jin, G.	Стерео СТЗ и лазерные дальномеры		
EL-KABBANY, A.	Инфракрасные датчики	При оценке проходимости учитывается аналитическая механическая модель робота [6];	При расчете индекса проходимости не используются геометрические параметры робота и его кинематическая схема.
Tanaka, Y.	Лазерный дальномер	Решение проблемы недоступности большой площади поверхности для анализа [7];	
Cappalunga, A.	Две цветовые	Сегментация не требует глобальной	Не учитывается цвет и

	камеры Stingray FireWire	модели поверхности [8];	текстура 3D точек; Возможность отрицательного влияния плохой видимости на восприятие [8];
Moghadam, P.	Short baseline (12cm) Point Grey Bumblebee colour stereo vision system.	Использует информацию с близкого расстояния, чтобы обучить классификатор оценивать дальнюю местность, недоступную стерео системе [9];	Классифицирует поверхность только на препятствия и проходимую зону, нет промежуточных значений [9];
Wang, W.	Стерео СТЗ	JRDE алгоритм устойчив к восприятию освещения и теневых областей [10];	
Чернухин, Ю. В.	Две аналоговые камеры и датчик глубины ASUS Xtion		Классифицирует поверхность только на препятствия и проходимую зону, нет промежуточных значений [11];
Dargazany, A.	RGB-D камера	Повышенная точность обнаружения поверхностей [12]	

Согласно таблице основными недостатками рассмотренных методов являются: отсутствие учета геометрических и кинематических параметров робота и отсутствие промежуточных значений проходимости при классификации поверхности.

Глава 2. Основная часть

Раздел 1. Разработка схемы системы оценки преодолимости

В качестве внешней среды используется составной полигон, где элементы представлены неоднородными по структуре, цвету и форме участками (рисунок 13) [15]. Темно-зеленые, темно-серые, коричневые и синие участки имеют щетинистые поверхности, слегка отличающиеся по жесткости друг от друга. Также есть светло-серые ковролиновые участки, светло-зеленые и розовые пенополистироловые, красные волокнистые, желтые поролоновые участки. Кроме этого, под верхним слоем щетинистых участков имеется слой либо поролона, либо органического стекла, что придает внешне одинаковым участкам различные свойства. Участки можно комбинировать множеством способов, что в купе с их отличающимися друг от друга свойствами позволяет создать упрощенную симуляцию физически неоднородной среды. В качестве мобильного робота используется учебная платформа Robotino 1.6 (рисунок 14). Robotino является голономным роботом, всенаправленные колеса которого расположены под углом 120° друг к другу [16]. Для получения информации о внешней среде используется камера Labtec WebCam Pro (рисунок 15). Параметры камеры представлены в таблице 5 [17].

Таблица 5. Параметры камеры

Матрица	0.3 млн пикс., CMOS
Разрешение	640x480
Максимальная частота кадров	30Гц
Фокусировка	ручная
Совместимые операционные системы	Windows 98/ME/2000/XP/7
Подключение	USB 1.1

Программирование осуществляется в среде Microsoft Visual Studio 2013. Robotino управляется с компьютера с помощью подключаемых библиотек по Wi-fi.

Рисунок 13 – Поле, используемое в качестве среды

Рисунок 14 – Платформа Robotino 1.6

Рисунок 15 – Labtec WebCam Pro

Понятие преодолимости не является однозначным. Если определять преодолимость как перемещение всех частей робота за границы целевого участка, возникает вопрос об этих границах. Если робот в процессе перемещения по участку изменил направление движения и выехал за границы не напротив точки вхождения, а сбоку, то является ли это преодолением? Следовательно, необходимо учитывать траекторию робота при вхождении в целевой участок. Помимо этого, можно уточнять понятие преодолимости различными целевыми установками: временем преодоления, затраченными ресурсами, отклонением от целевой траектории.

Для того чтобы определить понятие преодолимости, схематически изобразим робота, участок, через который робот проезжает и зона, где оказывается робот после преодоления участка (рисунок 23). Исходя из диаметра робота (36 см) и того факта, что размеры участков сопоставимы с размерами робота, определим преодолимость как попадание центра робота в окрестность точки А диаметром 40 см, лежащей за пределами преодолеваемого участка на пути движения робота.

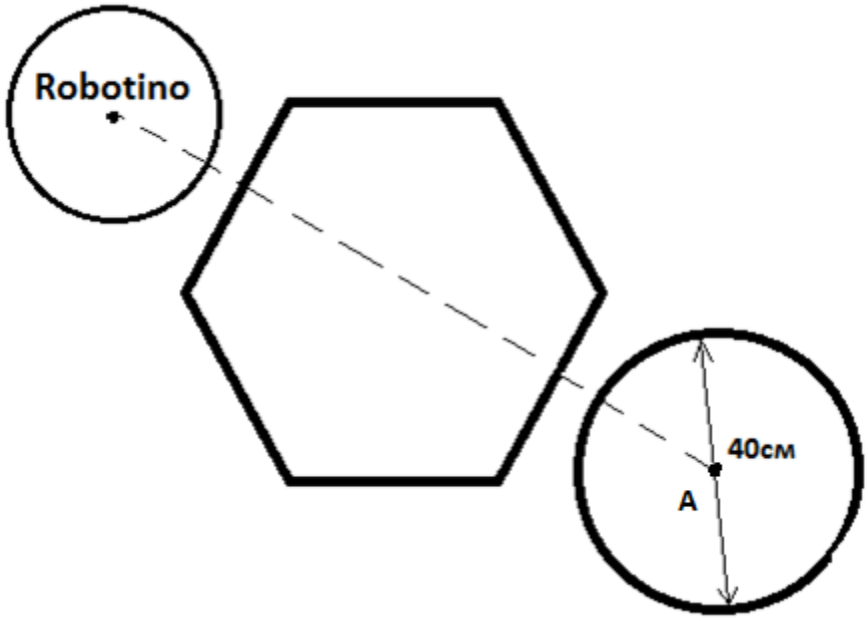


Рисунок 23 − Схематичное изображение робота, участка и зоны прибытия

Для оценки преодолимости участков вводятся следующие параметры:

1) Собственно преодолимость − бинарная переменная, принимающая значение 0 (препятствие) или 1 (не препятствие). Если робот преодолел участок, то оценка продолжается по следующим параметрам.

2) Отклонение цента Δx, (0..20) см., (радиус зоны попадания).
3) Отклонение Δy, (0..20) см.
4) Отклонение Δδ, (0..180)° (разворот робота вокруг своей оси).
5) Время t, мс.

Для классификации участка, необходимо, во-первых, этот участок увидеть, во-вторых, определить такие его параметры, которые связаны с преодолимостью. Для выделения контура участка на изображении, также как и для выделения параметров, используются средства библиотеки OpenCV. Таким образом, первым блоком проектируемой системы является блок обработки изображения. Далее, необходимо сопоставить параметры контура участка с его проходимостью, что можно реализовать путем тестирования участков на проходимость, фиксируя при этом их параметры. Действия по тестированию и фиксированию параметров контура можно обозначить как

блок обучения. Для того, чтобы создать классификатор, необходимо установить функциональные зависимости между параметрами контура и его преодолимостью и впоследствии использовать эти зависимости для оценки самой преодолимости. Далее подробнее описаны процессы разработки каждого блока.

1. Блок обработки изображения. Для оценки преодолимости впереди лежащих участков необходимо на изображении, получаемом с камеры, выделить интересующий контур и извлечь такие параметры, которые важны для будущей оценки. Следовательно, возникают две задачи:

1) Выбор важных параметров контура;
2) Реализация оконтуривания и выделения параметров инструментами VS2013.

Наиболее важным параметром является цвет (в данном случае используется пространство RGB), так как различие в цвете в большинстве случаев означает различие в структуре. Также следует учесть геометрические параметры участка (длина, площадь, ширина, центр масс), от которых может зависеть общее время движения робота, разница пути отдельных колес (при различном времени движения по участку).

Вторая задача решается средствами библиотеки OpenCV. Существует несколько методов выделения контуров (их тестирование описано в разделе 2). Каждый пригоден при различных параметрах изображения, к тому же любой из них требует настройки порогов бинаризации. Поэтому необходимо реализовать автоматический выбор одного из методов, настройку порогов бинаризации (у выбранного метода), а также провести фильтрацию шумов, связать контура из видеопотока с проходимостью на предыдущем такте (для исключения повторного анализа контуров). Кроме того, необходимо выделить параметры контура, выбранные выше. Подробнее алгоритм

обработки изображения описан в разделе 3, на схеме данный блок обозначен как «Блок обработки изображения».

2. Блок обучения. После выделения параметров контура необходимо привести в соответствие эти параметры и проходимость участков. Тестирование участков на проходимость описано в разделе 4, на схеме блок обозначен как: «Блок оценки преодолимости (человек)». Для того, чтобы система прогнозировала проходимость того или иного участка, необходимо создать классификатор. В данной работе классификатор создается с помощью дважды многорядной самоорганизующейся нейронной сети с активными нейронами на основе метода группового учета аргументов. В качестве аргументов используются параметры контура, полученные после блока обработки изображения, а также скорость робота по оси Y. Для полноты обучения использованы три значения скорости: 100; 250; 500. Так как при разных значениях скоростей робот может иметь различный характер соприкосновения с поверхностью, преодолимость отдельных участков будет зависеть от скорости. Выходной переменной является преодолимость участка.

Кроме прогнозирования преодолимости на основе параметров контура планируется также сделать на их основе оценку параметров самого робота. Для этого во время прохождения участка будут фиксироваться его токи, скорости моторов, показания энкодеров. Далее эти параметры используются в обучении сети как выходные переменные, а параметры контура и скорости как аргументы. Следующий этап – реализовать оценку преодолимости уже не только по параметрам контура, но и по внутренним параметрам робота. Такое обучение приведет к более точным прогнозам преодолимости.

3. Блок классификации участков. По результатам обучения создан классификатор, который при поступлении на вход параметров контуров классифицирует их на проходимые и непроходимые. Исходя из этого, формируется массив скоростей, который подается на двигатели робота. В

дальнейшем планируется включить в классификатор вариант «условное препятствие» [18], который принимает значение «проходимый» или «непроходимый» в зависимости от массива внутренних параметров. Здесь в качестве внутренних параметров рассматривается скорость движения робота, как один из ключевых для преодоления параметров. Скорости могут задаваться системой решения целевых задач (СРЦЗ), либо определяться на основе существующей модели.

На сегодняшний день элементы схемы, которые выделены черным цветом на рисунке 24, реализованы на практике. Элементы, выделенные оранжевым цветом, планируется реализовать в будущем.

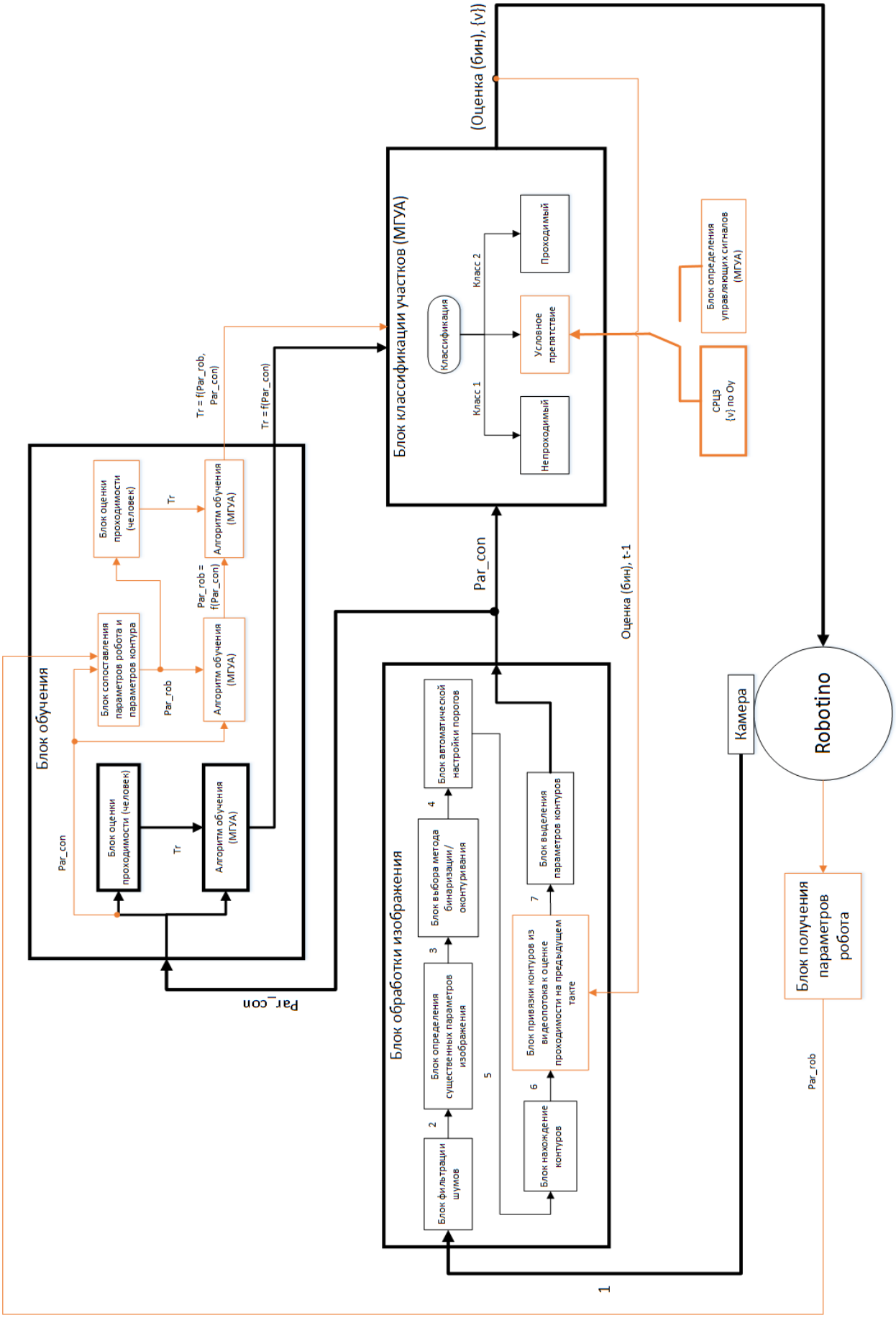


Рисунок 24 – Схема проектируемой системы

Ниже приведены обозначения, указанные на схеме.

1 – Исходное изображение с камеры;

2 – Отфильтрованное изображение;

3 – Яркость, градиент яркости;

4 – Изображение и выбранный метод оконтуривания;

5 – Матрица пикселей, метод оконтуривания, верхний и нижний пороги;

6 – Контур с текущего кадра;

7 – Проверенный контур;

Par_con – параметры контура: цвет (RGB), площадь (S), длина (L), ширина (D), расстояние до контура (A), центр масс (Cm);

Tr – проходимость;

Par_rob – параметры робота: токи (I1, I2, I3), скорости на моторах (M1, M2, M3), показания энкодеров (N1, N2, N3).

Раздел 2. Тестирование методов распознавания контуров при различных параметрах изображения

В рамках данного этапа были рассмотрены три метода нахождения контуров. Метод Канни, и два метода порогового преобразования бинарного изображения: метод на основе бинаризации по градациям серого и по RGB-каналам. Целью данного этапа являлось определение взаимосвязей между используемым методом, параметрами изображения и характеристиками выявляемых контуров. В эксперименте использованы два варианта освещения (дневной свет и сумеречный фон) и два типа конфигурации поверхности, содержащие наибольшее количество сочетаний участков.

Исходные изображения первой конфигурации участков при различной освещенности представлены на рисунке 25. В таблице 6 продемонстрированы результаты работы фильтров.

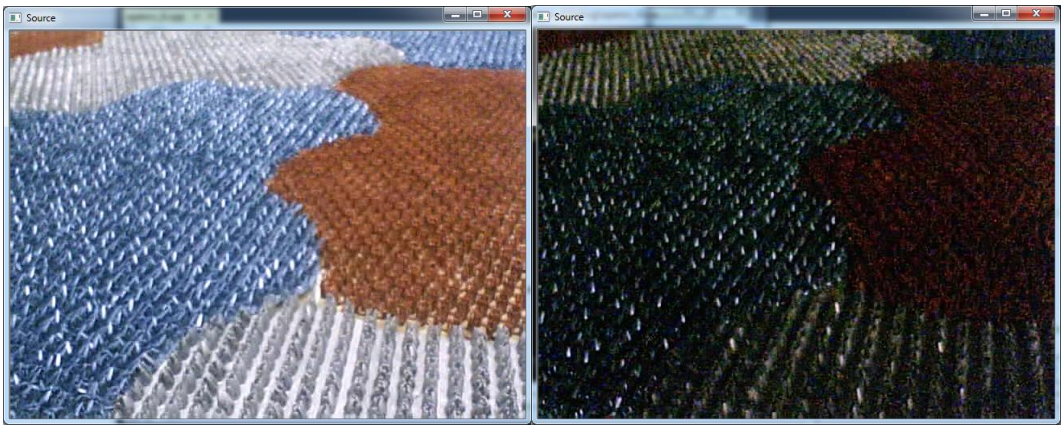


Рисунок 25 – Исходные изображения для тестирования методов оконтуривания

Таблица 6. Результаты работы фильтров

Метод нахождения контуров	Освещение	Среднее время выполнения одной итерации, мс	Минимальная и максимальная границы диапазона фильтров	Результат нахождения контуров

Канни	Светлое	40	60,550	Рисунок 26 (слева)
	Темное	25	60,150	Рисунок 26 (справа)
Порог по градациям серого	Светлое	45	33,123	Рисунок 27 (слева)
	Темное	160	27,236	Рисунок 27 (справа)
Порог по RGB-каналам	Светлое	95	Настройка фильтров на синий канал	Рисунок 28 (слева)
	Темное	118	R: 30,136 G: 11,97 B: 14,111	Рисунок 28 (справа)

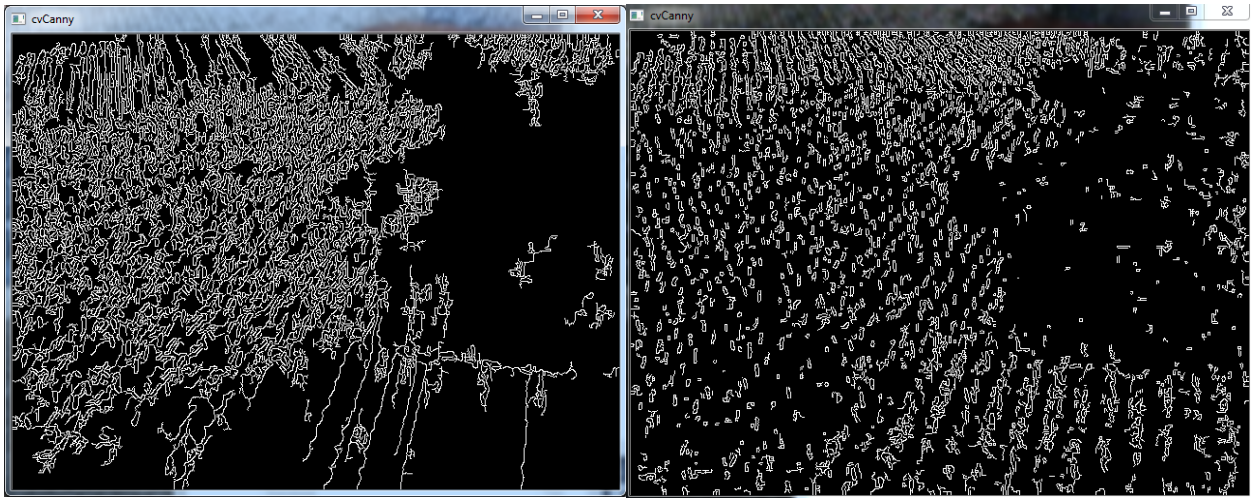


Рисунок 26 – Результаты работы метода Канни

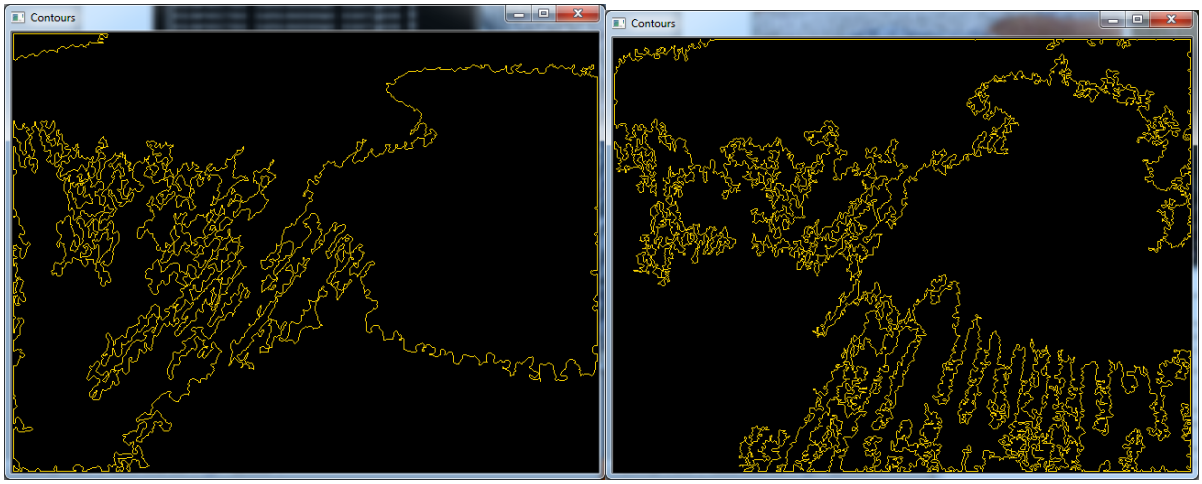


Рисунок 27 – Результаты работы порогового метода

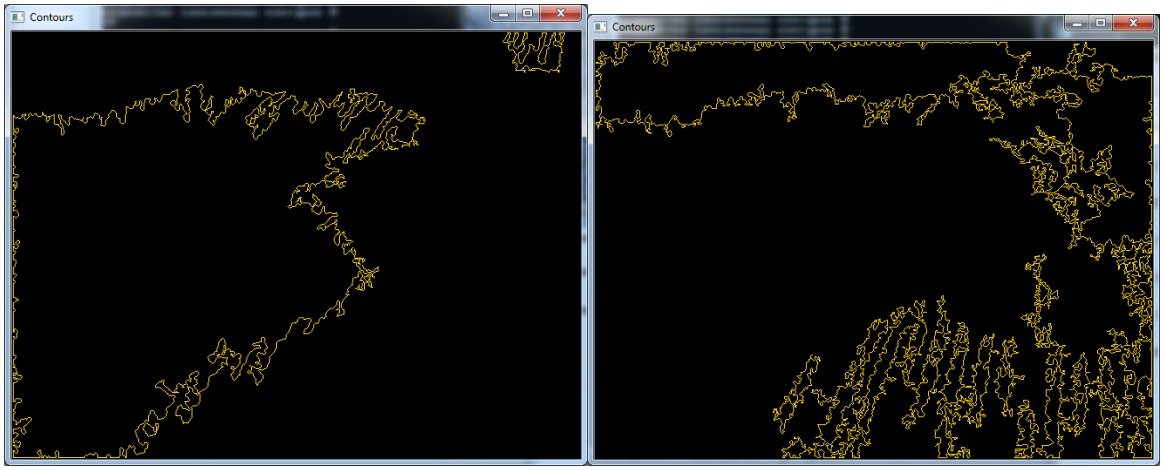


Рисунок 28 – Результаты работы RGB-выделения

На рисунке 29 показаны исходные изображения второй конфигурации участков поверхности при разной освещенности. В таблице 7 представлены результаты работы методов.

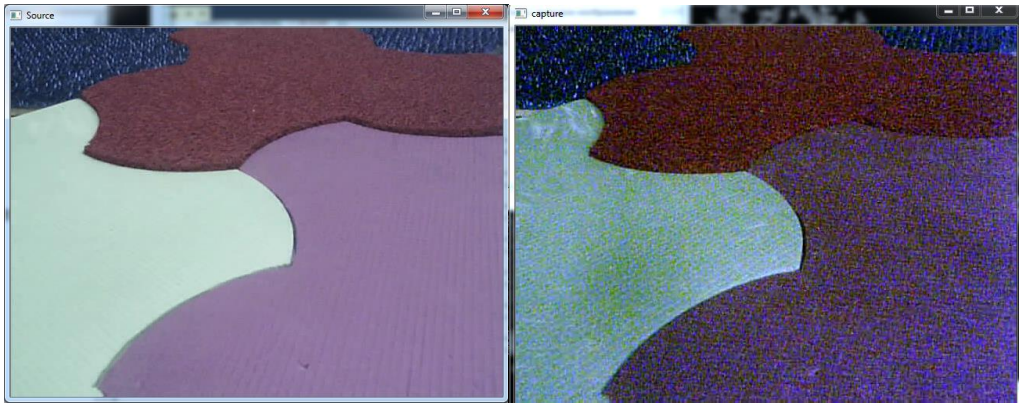


Рисунок 29 – Исходные изображения для тестирования методов оконтуривания

Таблица 7. Результаты тестирования методов оконтуривания

Метод нахождения контуров	Освещение	Среднее время выполнения одной итерации, мс	Минимальная и максимальная границы диапазона фильтров	Результат нахождения контуров
Канни	Светлое	30	40,150	Рисунок 30 (слева)
	Темное	25	60,470	Рисунок 30 (справа)
Порог по градациям серого	Светлое	43	35,154	Рисунок 31 (слева)
	Темное	120	42,123	Рисунок 31 (справа)
Порог по RGB-каналам	Светлое	25	R: 121,233 G: 72,249 B: 68,256	Рисунок 32 (слева)
	Темное	60	R: 31,255 G: 50,255 B: 50,255	Рисунок 32 (справа)

При настройке диапазона освещенность влияет на скорость работы метода Канни меньше.

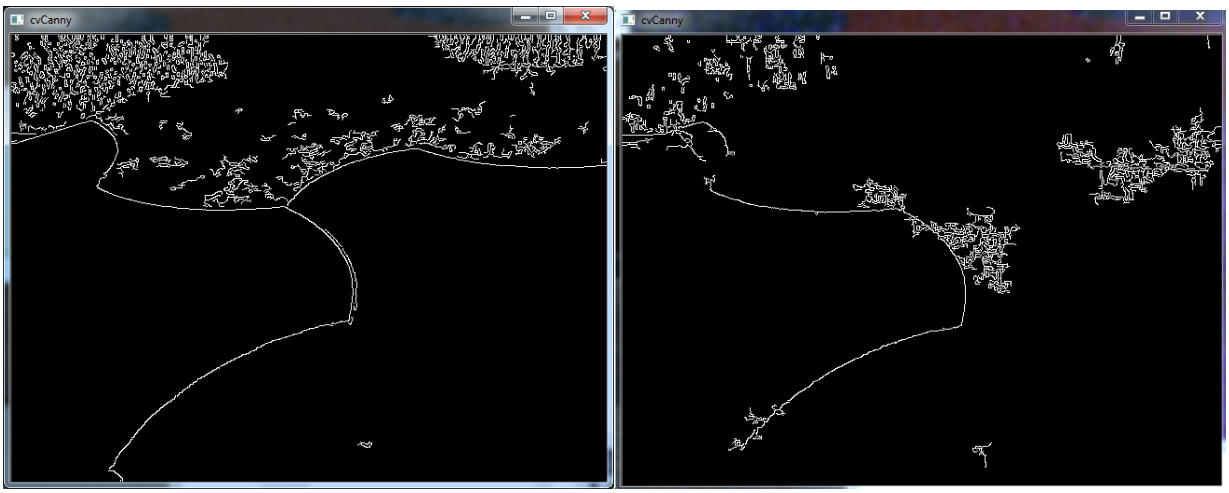


Рисунок 30 – Результаты работы метода Канни

Рисунок 31 – Результаты работы порогового метода

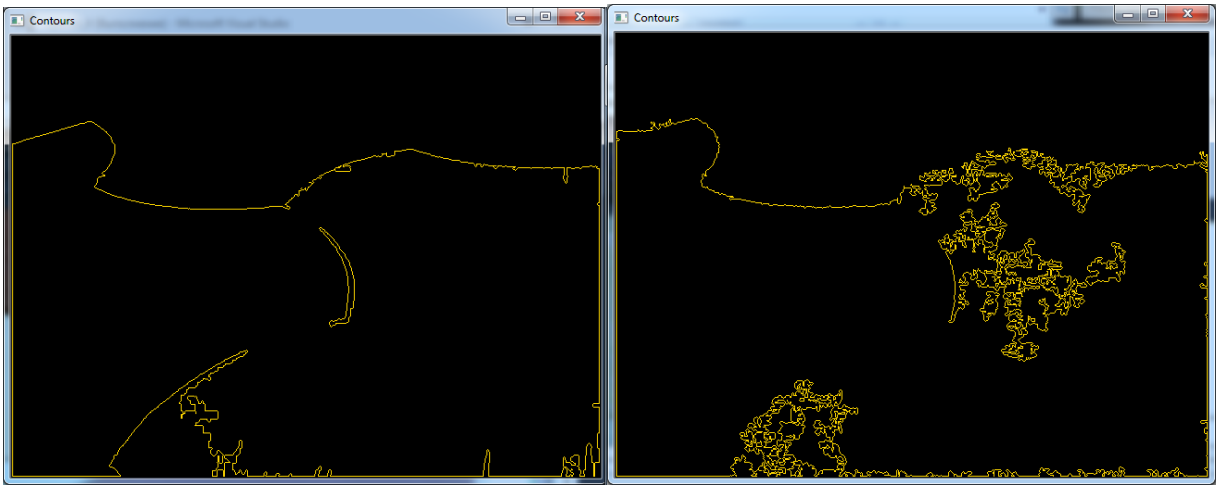


Рисунок 32 – Результаты работы RGB-выделения

Из проведенных опытов следуют несколько выводов. При более темном освещении контура распознаются хуже, чем при светлом всеми методами, но лучше чем другие ищет контура при плохой освещенности пороговый метод с градациями серого, причем как на щетинистых участках, так и на гладких. При нормальной освещенности контура щетинистых поверхностей также четче и быстрее (40мс) находит пороговый метод, использующий градации серого. При этом пороговый метод, использующий RGB каналы, показывает менее точные контура и за большее время, а метод Канни является наименее пригодным для щетинистых участков. Если говорить о гладких участках, метод Канни дает наиболее точные и явные контура, а RGB-пороговый наименее точные. Для реализации выбора метода оконтуривания необходимо извлечь из изображения параметры, оценка которых будет основой выбора. Такими параметрами являются: цвет, средняя яркость и градиент яркости по изображению.

Раздел 3. Разработка алгоритма обработки изображения

Как было указано в разделе 1, обработка изображения (одного кадра из видеопотока) состоит из нескольких этапов: фильтрация шумов, определение существенных параметров (для последующего выбора метода оконтуривания), выбор метода оконтуривания, автоматическая настройка порогов бинаризации, нахождение контуров, привязка контуров к преодолимости с предыдущего такта, определение параметров контура (рисунок 33). Фильтрация шумов производится стандартной процедурой. Существенными параметрами изображения названы такие параметры, которые играют критическую роль при выборе метода оконтуривания. Исходя из результатов раздела 2, этими параметрами являются: средний RGB, средняя яркость и градиент яркости. Но извлечение этих параметров из изображения целиком не позволит сделать достаточно точных выводов, поэтому было решено разделить изображение на так называемые зоны интереса (ROI – region of interest), в каждой из которых уже будут найдены существенные параметры (рисунок 34). Далее, чтобы выбрать метод бинаризации, существенные параметры каждой зоны интереса складываются и в зависимости от того, в какой диапазон попадет их сумма, выбирается метод. Окончательный метод выбирается исходя из наиболее частого метода в зонах интереса (рисунок 35). Автоматическая настройка порогов происходит по принципу вычисления нижних и верхних порогов как функций от существенных параметров зон интереса (рисунок 36). Далее подпрограммы нахождения контуров и их параметров выполняются с помощью стандартных функций OpenCV (рисунок 33).

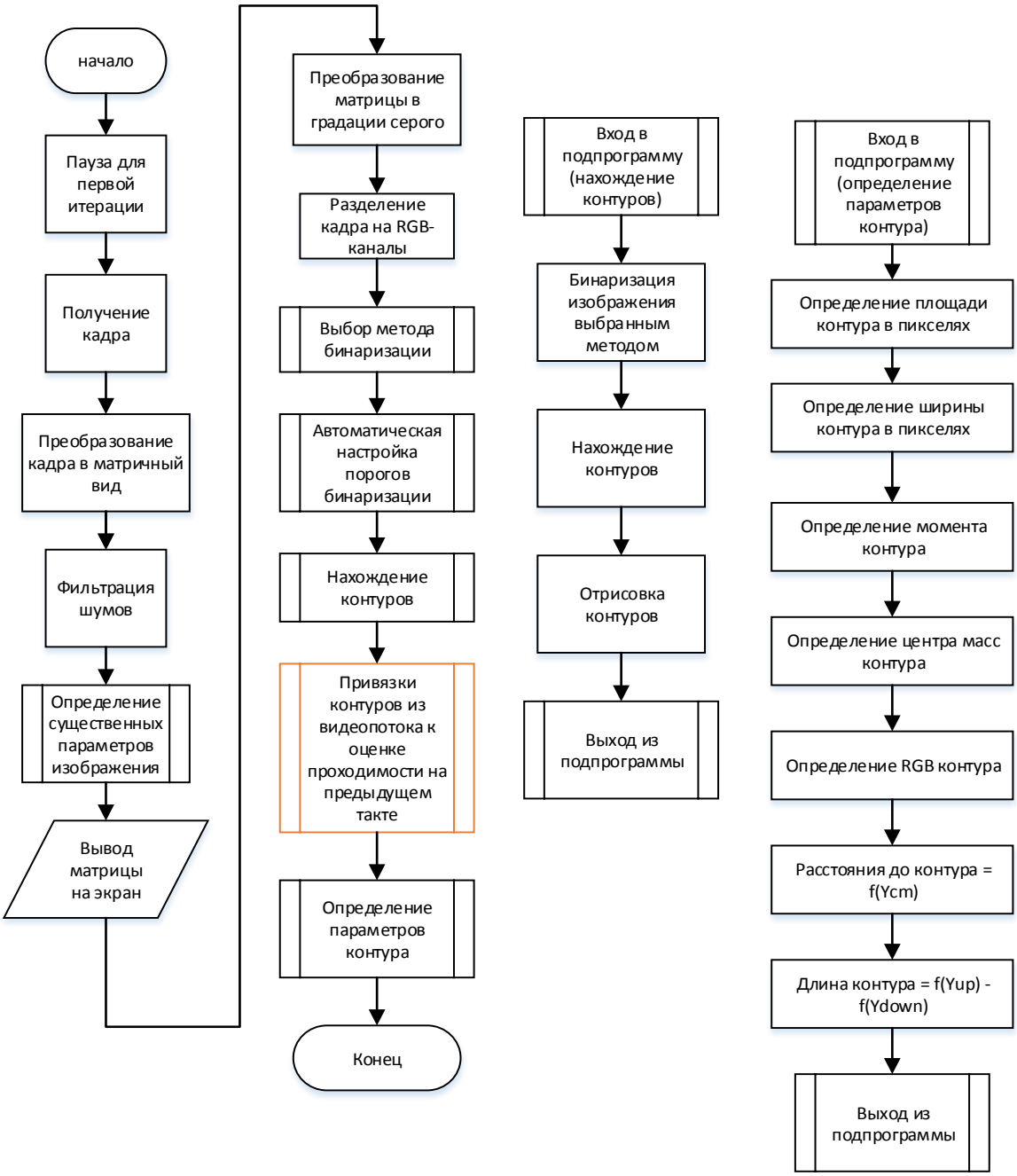


Рисунок 33 – Общий алгоритм обработки кадра (слева), подпрограмма нахождения контуров (центр), подпрограмма определения параметров (справа)

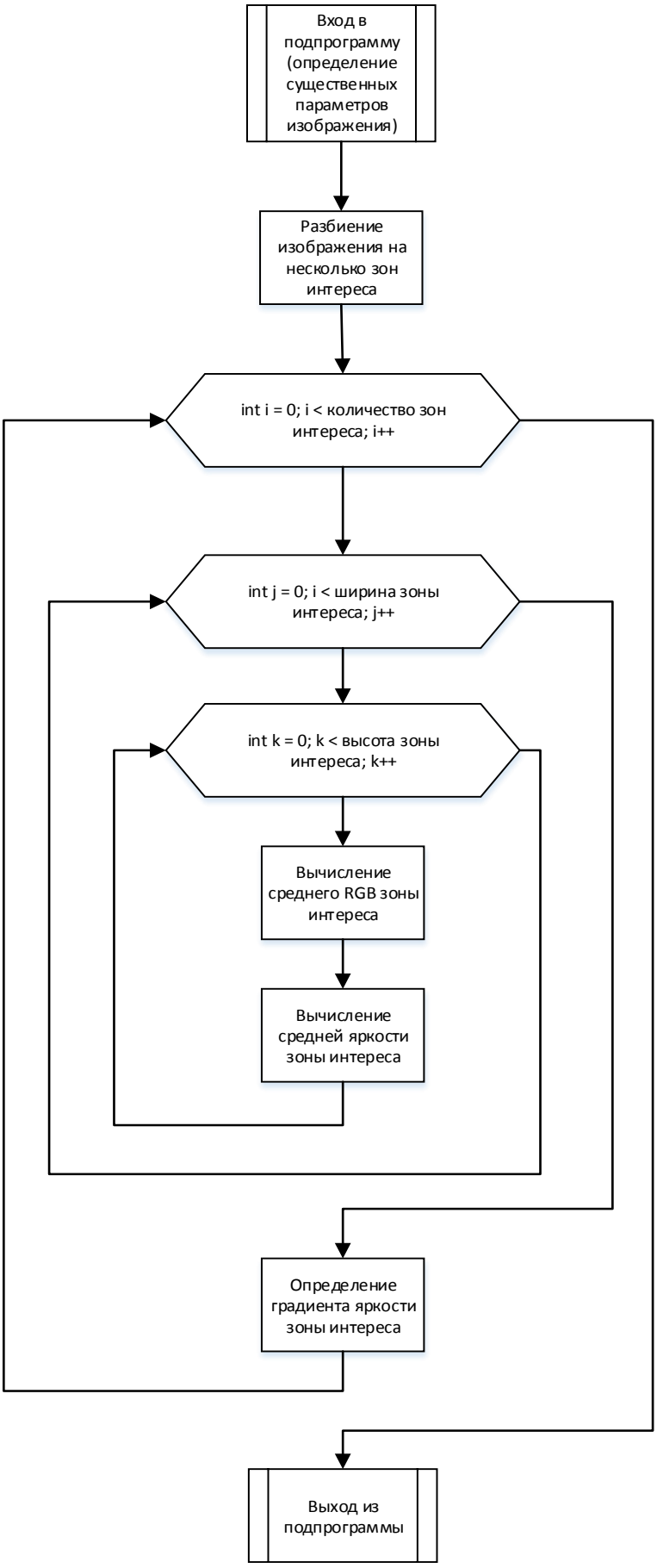


Рисунок 34 – Подпрограмма определения существенных параметров изображения

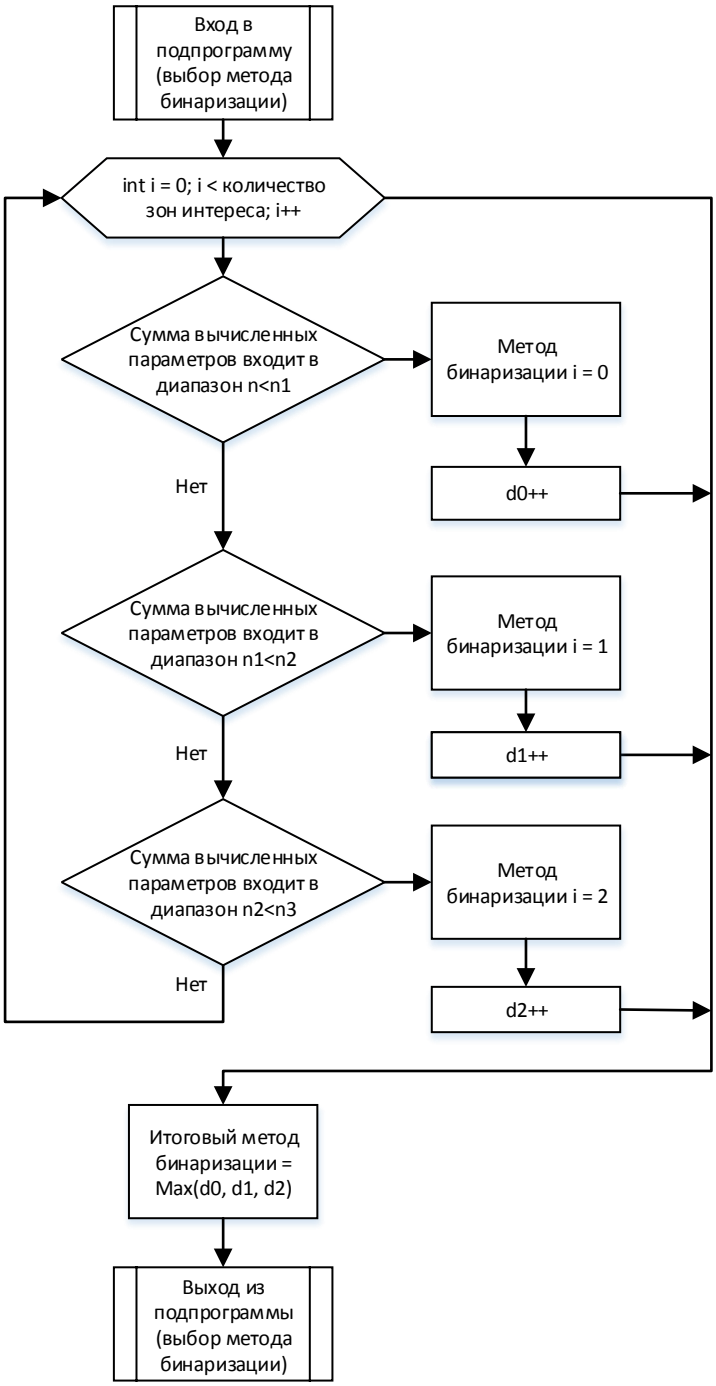


Рисунок 35 – Подпрограмма выбора метода бинаризации

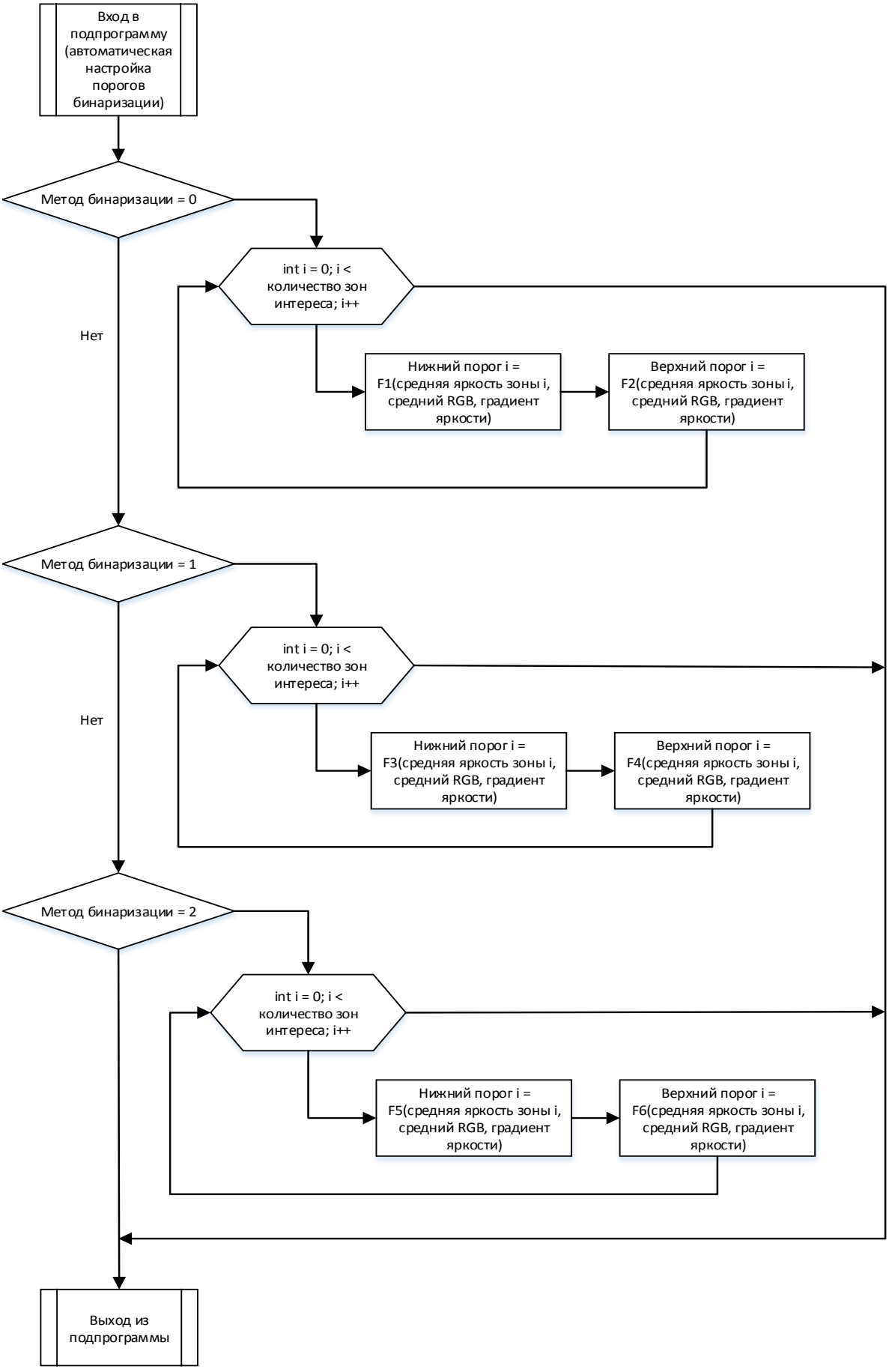


Рисунок 36 – Подпрограмма автоматической настройки порогов бинаризации

Раздел 4. Экспериментальное исследование влияние параметров локального участка на его преодолимость

Целью данного этапа является выявление зависимостей между параметрами участка и его преодолимостью. Используются следующие параметры участка (контура): площадь, длина, ширина, цвет (RGB), координаты центра масс и расстояние до центра масс.

Тесты проводится на участках различного цвета, размера и формы при различных скоростях робота (изображения участков представлены на рисунках 36-46). В таблице 7 приведены данные тестов. В таблице: № - номер теста; Vy – скорость робота по оси Y; L – длина контура; D – ширина контура; R, G, B – компоненты цветовой модели RGB; Xc – x-координата центра масс; Yc – y-координата центра масс; A – расстояние до контура; в – волокнистый, щ – щетинистый. В скобках после параметров указаны их номера при обучении.

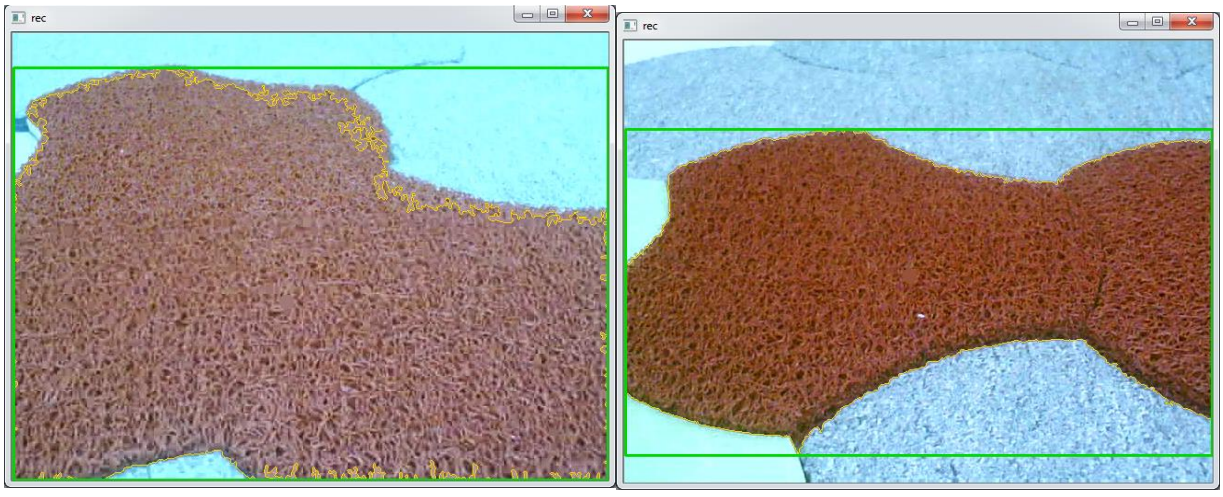


Рисунок 36 (а), 36(б)

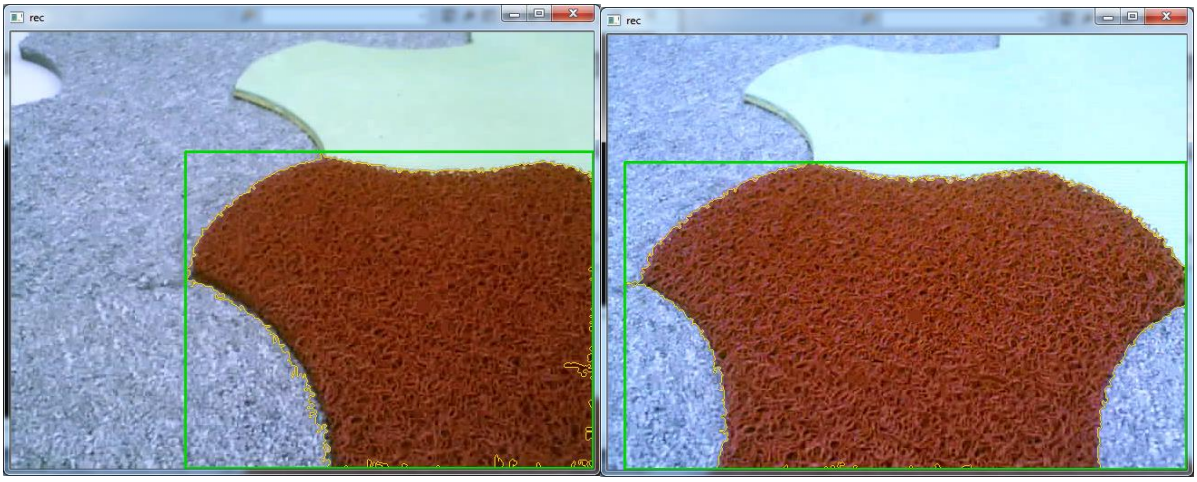


Рисунок 37 (а), 37 (б)

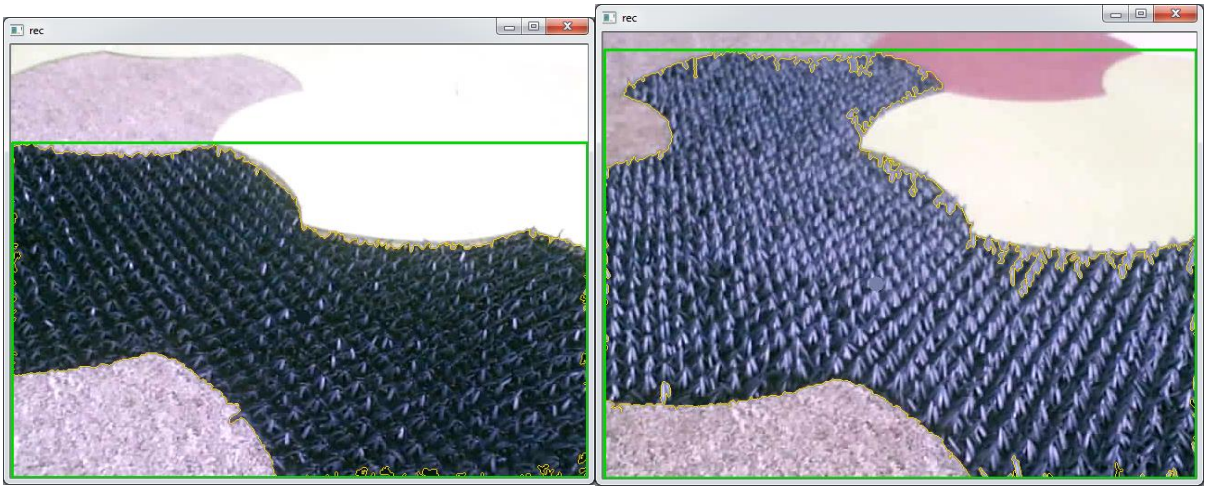


Рисунок 38 (а), 38 (б)

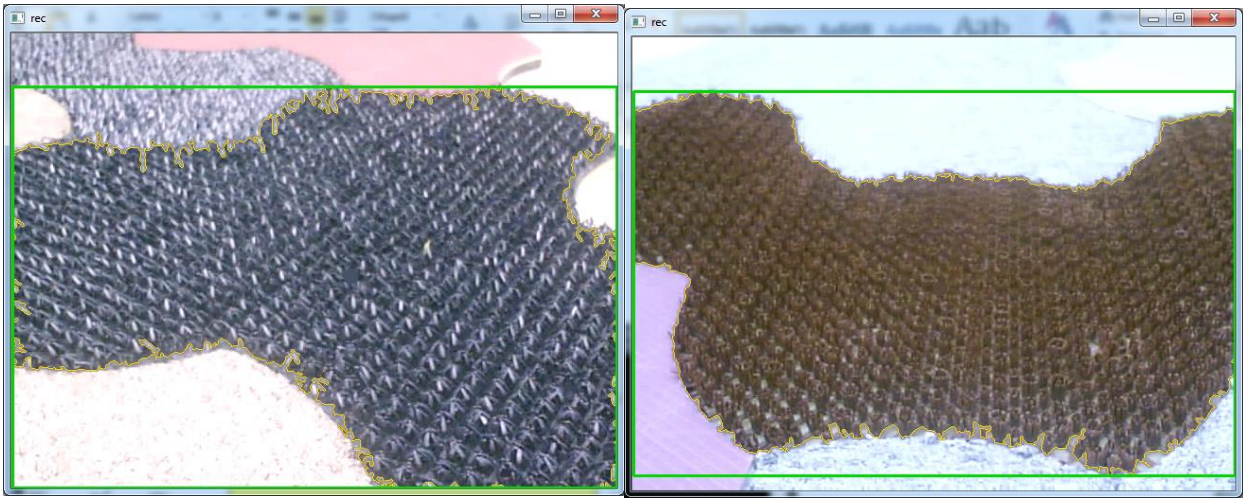


Рисунок 39 (а), 39 (б)

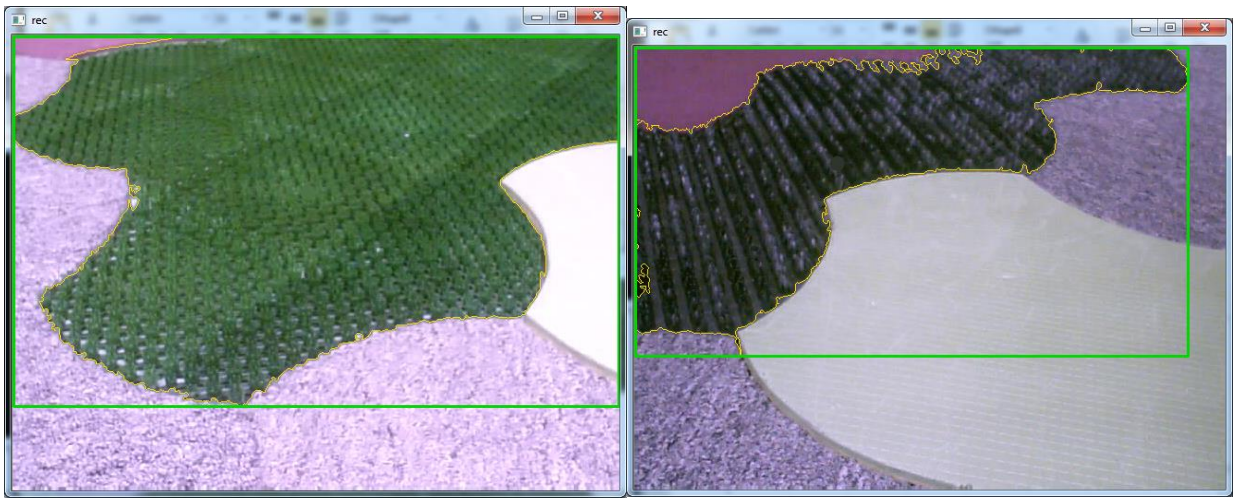


Рисунок 40 (а), 40 (б)

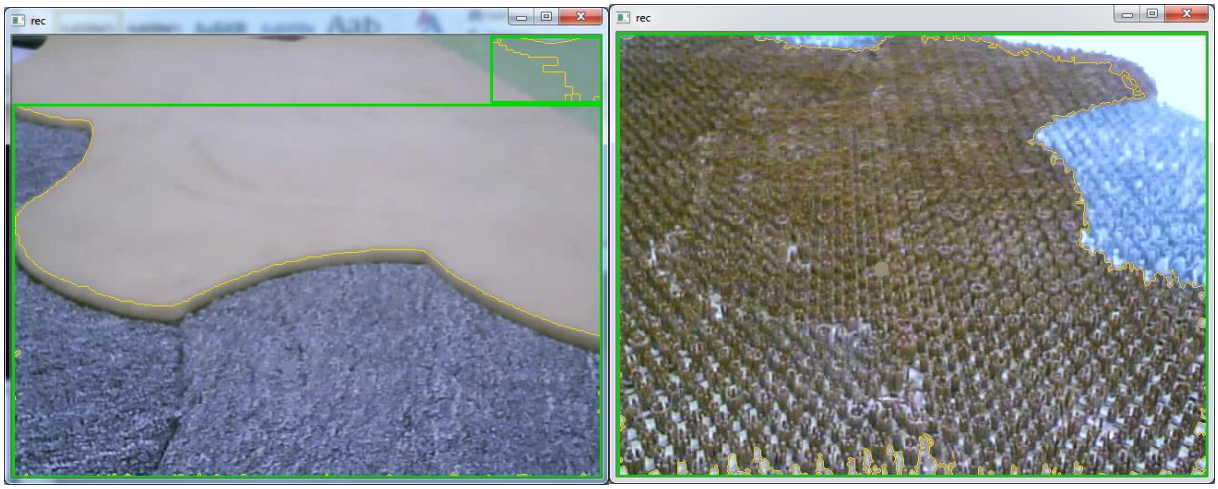


Рисунок 41 (а), 41 (б)

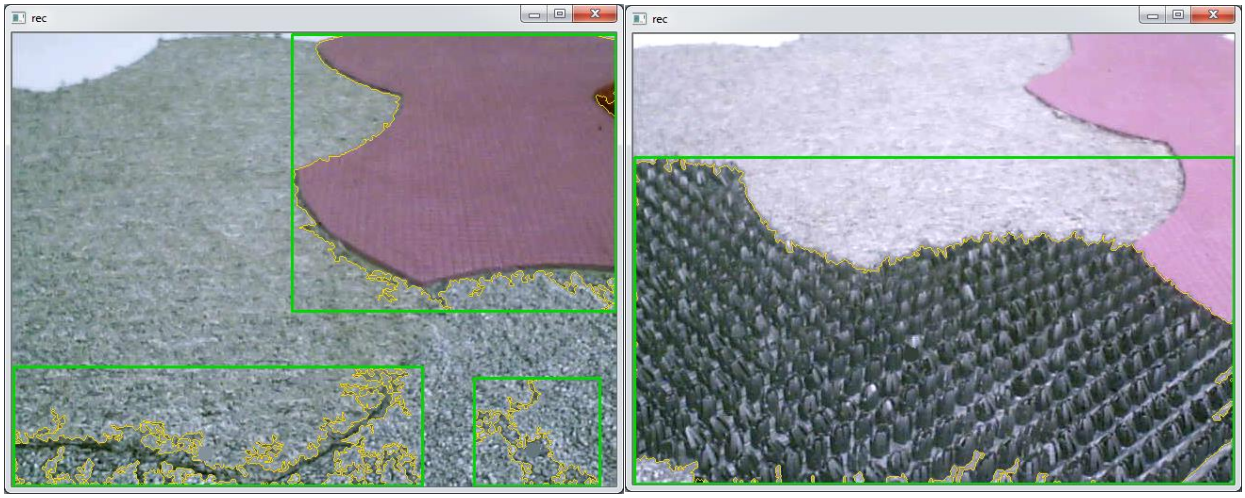


Рисунок 42 (а), 42 (б)

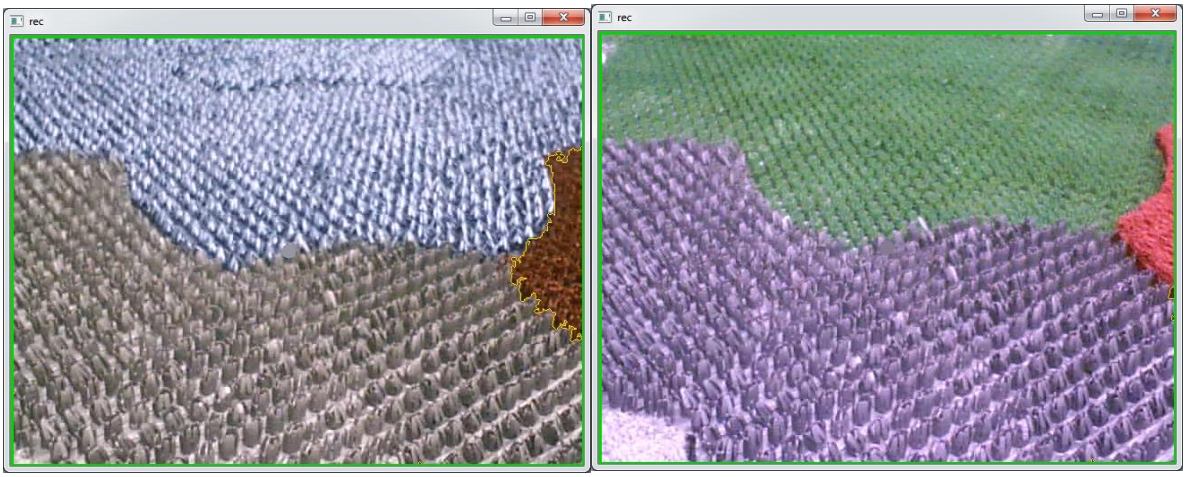


Рисунок 43 (а), 43 (б)

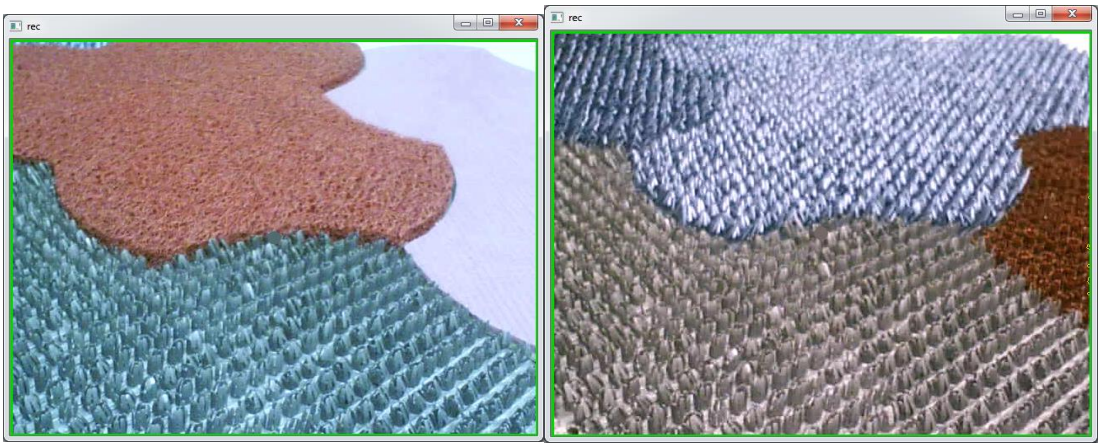


Рисунок 44 (а), 44 (б)

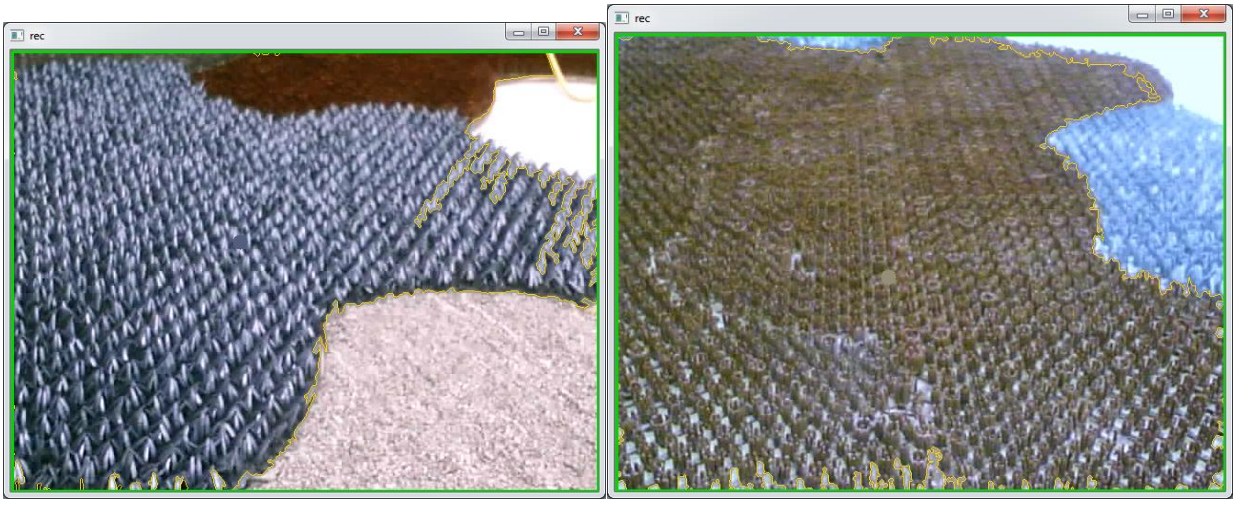


Рисунок 45 (а), 45 (б)

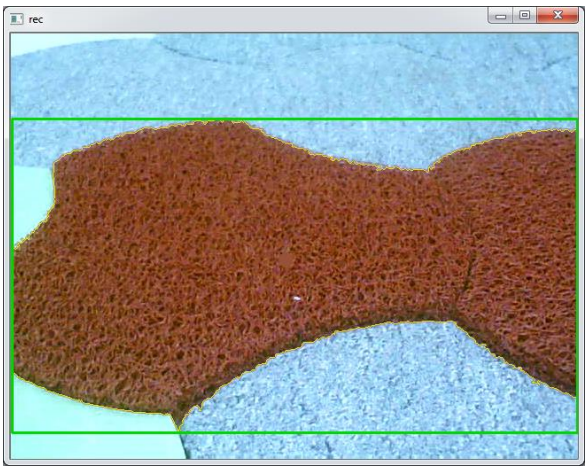


Рисунок 46

Таблица 8. Данные тестов участков на преодолимость

№	Словесная хар-ка	Vy (x0)	Параметры контура									Преод.
			S (x1)	L (x2)	D (x3)	R (x4)	G (x5)	B (x6)	Xc (x7)	Yc (x8)	A (x9)	Преодо-лел
1.1	Красный в, крупный	100	221880	51	638	50	71	111	310.5	275	40.5	0
1.2		250	221880	51	638	50	71	111	310.5	275	40.5	0
1.3		500	221880	51	638	50	71	111	310.5	275	40.5	0
2.1	Красный в, широкий	100	157410	35	638	60	85	160	310.5	255	42.3	1
2.2		250	157410	35	638	60	85	160	310.5	255	42.3	0
2.3		500	157410	35	638	60	85	160	310.5	255	42.3	0
3.1	Красный в, мелкий	100	119054	31	447	11	42	119	451.4	301	38.2	0
3.2		250	119054	31	447	11	42	119	451.4	301	38.2	0
3.3		500	119054	31	447	11	42	119	451.4	301	38.2	0
4.1	Красный в, по центру	100	154798	30	601	46	66	156	350	309	37.5	0
4.2		250	154798	30	601	46	66	156	350	309	37.5	0
4.3		500	154798	30	601	46	66	156	350	309	37.5	0
5.1	Синий щ, волокна вдоль	100	163163	34	638	55	45	28	324	300	38.2	1
5.2		250	163163	34	638	55	45	28	324	300	38.2	1
5.3		500	163163	34	638	55	45	28	324	300	38.2	1
6.1	Синий щ, волокна вдоль	100	195040	56	638	151	110	96	292	266	41.3	1
6.2		250	195040	56	638	151	110	96	292	266	41.3	1
6.3		500	195040	56	638	151	110	96	292	266	41.3	1
7.1	Синий щ,	100	193302	44	638	114	93	82	358	256	42.2	0

7.2	волокна	250	193302	44	638	114	93	82	358	256	42.2	0
7.3	против	500	193302	44	638	114	93	82	358	256	42.2	1
8.1	Коричневый	100	185519	43	638	53	69	89	326	266	41.3	1
8.2	щ	250	185519	43	638	53	69	89	326	266	41.3	1
8.3		500	185519	43	638	53	69	89	326	266	41.3	1
9.1	Зеленый щ	100	172605	53	638	73	120	83	311	165	52.6	1
9.2		250	172605	53	638	73	120	83	311	165	52.6	1
9.3		500	172605	53	638	73	120	83	311	165	52.6	1
10.1	Серый щ,	100	85005	49	598	26	41	30	218	127	58.1	0
10.2	волокна	250	85005	49	598	26	41	30	218	127	58.1	0
10.3	вдоль	500	85005	49	598	26	41	30	218	127	58.1	1
11.1	Поролоновый	100	135244	49	638	171	177	185	230	171	59.2	1
11.2		250	135244	49	638	171	177	185	230	171	59.2	1
11.3		500	135244	49	638	171	177	185	230	171	59.2	1
12.1	Синий щ,	100	162338	51	638	35	56	69	308	158	53.5	0
12.2	волокна	250	162338	51	638	35	56	69	308	158	53.5	0
12.3	против	500	162338	51	638	35	56	69	308	158	53.5	0
13.1	Розовый	100	43626.5	41	255	130	93	133	532.9	110	60.8	1
13.2		250	43626.5	41	255	130	93	133	532.9	110	60.8	1
13.3		500	43626.5	41	255	130	93	133	532.9	110	60.8	1
14.1	Серый щ, с	100	160200	31	638	62	63	54	294.3	341	35	0
14.2	последующим	250	151400	29	638	43	55	38	291	353	34.2	0
14.3	ровным	500	172520	32	368	58	85	93	306	323	36.3	0
15.1	Серый щ, с	100	202100	57	638	120	113	141	246	223	45.5	0
15.2	последующим	250	202100	57	638	120	113	141	246	223	45.5	0
15.3	щ (синий)	500	202100	57	638	120	113	141	246	223	45.5	0
16.1	Серый щ, с	100	136177	36	602	152	140	178	268	306	37.7	0
16.2	последующим	250	136177	36	602	152	140	178	268	306	37.7	0
16.3	щ (зеленый)	500	136177	36	602	152	140	178	268	306	37.7	0
21.1	Красный в,	100	180800	35	638	122	72	50	332	298	38.5	1
21.2	широкий	250	180800	35	638	122	72	50	332	298	38.5	1
21.3		500	180800	35	638	122	72	50	332	298	38.5	1

Как видно из тестов, наиболее непроходимыми участками являются красный и синий. Также было замечено, что существенное влияние

оказывает такой параметр как ориентация волокон участка, который не был включен в ход тестов по причине трудоемкости извлечения этого параметра из изображения. Влияние параметров контуров (а также скорости робота) на проходимость представлено в таблице 8 цифрами от 1 до 5, где 5 – наибольшее влияние, 1 – наименьшее.

Таблица 9. Влияние параметров контуров на их проходимость

Параметры преодолимости	Параметры контура						Скорость робота
	S	L	D	RGB	Координаты	A	
Общая преодолимость	2	3	1	5	2	2	4
Отклонение	3	3	4	5	4	1	2
Время	4	5	3	4	3	2	5

Раздел 5. Обучение блока оценки преодолимости участков среды

Классификатор представляет из себя результат обучения дважды многорядной самоорганизующейся нейронной сети с активными нейронами на основе метода группового учета аргументов. Нейронная сеть реализована на программном обеспечении, поставляемое на CD-диске к книге GMDH-methodology and implementation in C [19]. Верной классификацией считается такое значение выходной переменной МГУА (Y), при котором разница этой выходной переменной и исходного значения проходимости меньше 0.5. Исходные данные из таблицы 7 разделяются на обучающую (A) и проверочную (B) выборки. Кроме этого, классификатор проверяется на третьей, тестовой выборке (C), которая представлена в таблице 10.

Таблица 10. Тестовая выборка C

№	Словесная хар-ка	Vy (x0)	Параметры контура									Преод.
			S (x1)	L (x2)	D (x3)	R (x4)	G (x5)	B (x6)	Xc (x7)	Yc (x8)	A (x9)	Преодо-лел
16.1	Серый щ, с последующим щ (зеленый)	100	136177	36	602	152	140	178	268	306	37.7	0
16.2		250	136177	36	602	152	140	178	268	306	37.7	0
16.3		500	136177	36	602	152	140	178	268	306	37.7	0
17.1	Серый щ, с последующим красным	100	120613	28	600	123	167	192	268	339	35	0
17.2		250	120613	28	600	123	167	192	268	339	35	0
17.3		500	120613	28	600	123	167	192	268	339	35	0
18.1	Серый щ, с последующим щ (синий)	100	124211	33	599	109	99	98	263.2	318	36.8	0
18.2		250	124211	33	599	109	99	98	263.2	318	36.8	0
18.3		500	124211	33	599	109	99	98	263.2	318	36.8	0
19.1	Синий щ, волокна вдоль	100	174522	56	599	82	110	120	228	209	47	1
19.2		250	174522	56	599	82	110	120	228	209	47	1
19.3		500	174522	56	599	82	110	120	228	209	47	1
20.1	Коричневый щ, крупный	100	262013	57	638	124	115	120	299	250	42.7	0
20.2		250	258600	57	638	132	130	115	286	254	42.3	0
20.3		500	262250	57	638	110	120	115	288	257	42.6	0

21.1	Красный в, широкий	100	180800	35	638	122	72	50	332	298	38.5	1
21.2		250	180800	35	638	122	72	50	332	298	38.5	1
21.3		500	180800	35	638	122	72	50	332	298	38.5	1

Целью данного этапа является исследование влияния параметров МГУА (ёмкость нейронной сети, степень полинома, соотношение обучающей и проверочной выборок), а также расположения исходных данных в выборках A и B на процент верной классификации проходимости. Кроме этого, необходимо сравнить степень влияния аргументов на выходную переменную и выводы этапа 3. В таблице 10: № - номер эксперимента; p – максимальная степень полинома; vol – емкость сети, rat – соотношение выборок; % (A и B) – процент верной классификации на выборках A и B; % (C) – процент верной классификации на выборке C; In – номер слоя, на котором начинается индуцит (если число отсутствует, индуцита нет); pop - наиболее часто используемые переменные. В процессе проведения экспериментов в выборки были внесены изменения, причины указаны ниже таблицы.

Таблица 11. Параметры МГУА и выборок

№	p	vol	rat	Изменения в выборках	% A и B)	% (C)	In; pop
1	2	2x2	22:23		78.3	68.4	R,G,L
2	2	5x5	22:23		84.8	68.4	L,R,G
3	2	10x10	22:23		87	68.4	7; S,R,G,Vy
4	2	20x20	22:23		93.5	47.4	10; S,R,Vy
5	2	40x10	22:23		93.5	21.1	S,K,Vy
6	3	2x2	22:23		86.96	68.4	R,G,L
7	3	5x5	22:23		82.6	52.63	L,D,R,G
8	3	10x10	22:23		97.8	21.1	Vy,L,D,R,G

9	2	20x20	22:23	1	95.7	47.4	9; Vy,L,R,G
10	4	2x2	22:23	1	78.3	68.4	R,G,L
11	4	5x5	22:23	1	89.13	68.4	Vy,L,R,G
12	4	10x10	22:23	1	97.8	63.2	R,G,Vy,L
13	3	10x10	22:23	1; 2*	82.6	47.4	6; R,G,Vy
14	3	10x10	22:23	1; 2	80.44	84.2	6; Vy,R,B,Yc
15	3	10x10	15:30	1; 2; 3	95.6	57.9	Vy,L,R,G, Xc
16	3	10x10	30:15	1; 2; 3	91.3	52.6	5; D,G,B
17	3	10x10	22:23	3	76.9	52.6	6; Vy,L,Xc,B
18	3	10x10	22:23	4	100	30	Vy,L,R,G
19	3	10x10	30:15	4	82.6	52.6	3; R,G,B
20	3	10x10	15:30	4	54.3	52.6	7; S,L,B
21	3	10x10	25:26	4; 5	94.5	76.9	6; Vy,R,G
22	3	10x10	20:31	4; 5	90.4	76.9	7; Vy,D,R,G
23	3	10x10	31:20	4; 5	90.4	76.9	4; R,G,Yc

Изменения в выборках: 1 – параметр S уменьшен в 1000 раз, так как все остальные параметры на несколько порядков меньше; 2* – в A добавлены тесты (из B) с параметрами RGB, близкими к параметрам RGB из выборки C; 2 – в A добавлен еще один тест; 3 – 2-й тест сменен на 21-й, так как 2-й был снят неточно; 4 – исправлена ошибка учителя в тесте 2.1 (значение выходной

переменной изменено с 1 на 0); 5 – в исходные выборки добавлены 16 и 21 тесты из выборки C.

На рисунках 47-53 продемонстрированы графики преодолимости использованных участков. Черная кривая показывает исходные значения преодолимости, красная – выходную переменную классификатора. При малом количестве нейронов и слоев (2 либо 5) процент верной классификации не достигает 90%, и графики имеют вид, показанный на рисунке 47. Процент верной классификации на выборке C не достигает 70%. При большой емкости сети (>10:10) процент находится в диапазоне 90-95, графики имеют вид, продемонстрированный на рисунке 48.

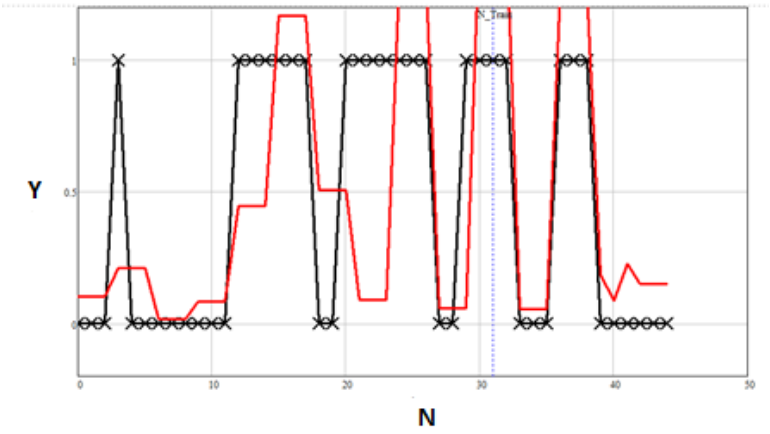


Рисунок 47 – График преодолимости на выборках A,B эксперимента 1

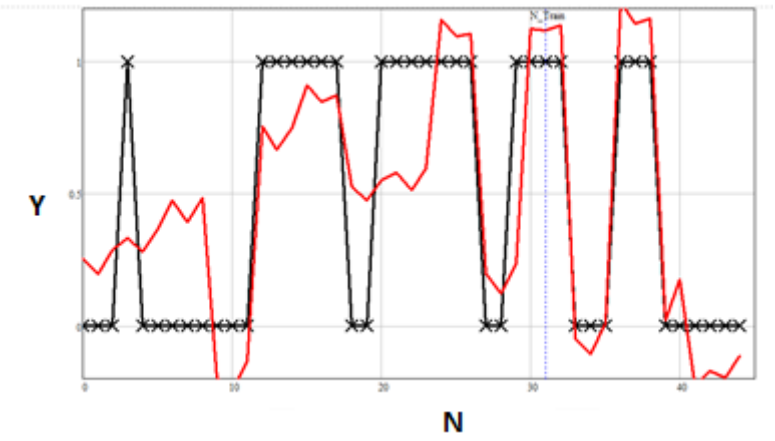


Рисунок 48 – График преодолимости на выборках A,B эксперимента 4

Если обучающая выборка меньше проверочной (15:30), то процент верной классификации существенно падает по сравнению с равными соотношениями (рисунок 49). Обратное соотношение (30:15) дает такой же результат, что и при равном соотношении (рисунок 50).

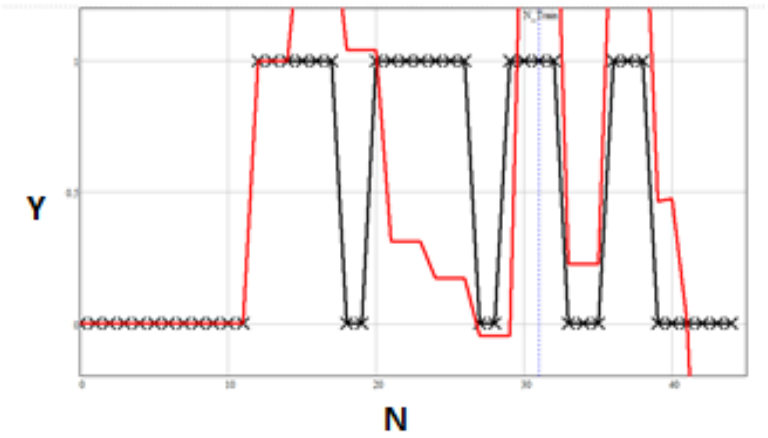


Рисунок 49 – График преодолимости на выборках A,B эксперимента 20

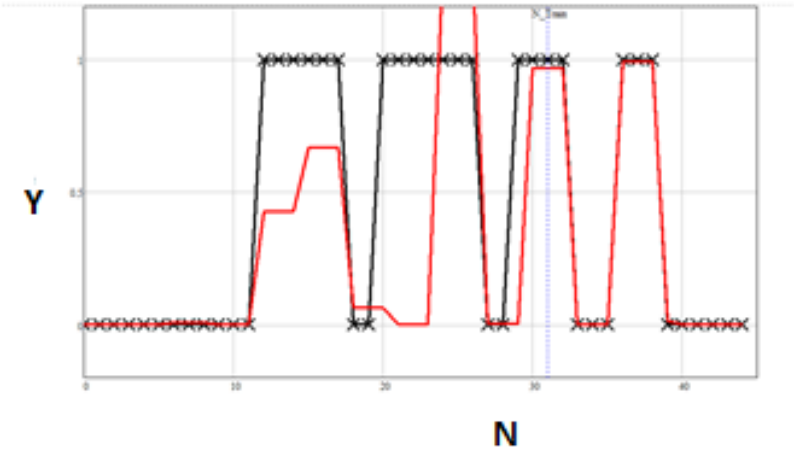


Рисунок 50 – График преодолимости на выборках A,B эксперимента 19

Объем сети 10:10 при равном соотношении выборок A и B обеспечивает верную классификацию более 95%. Как видно из рисунка 51, графики совпадают за исключением одной точки. Тест №2, обеспечивший это несоответствие, был признан ошибочным, так как время преодоления участка на этапе установления связей было на порядок больше обычного времени преодоления.

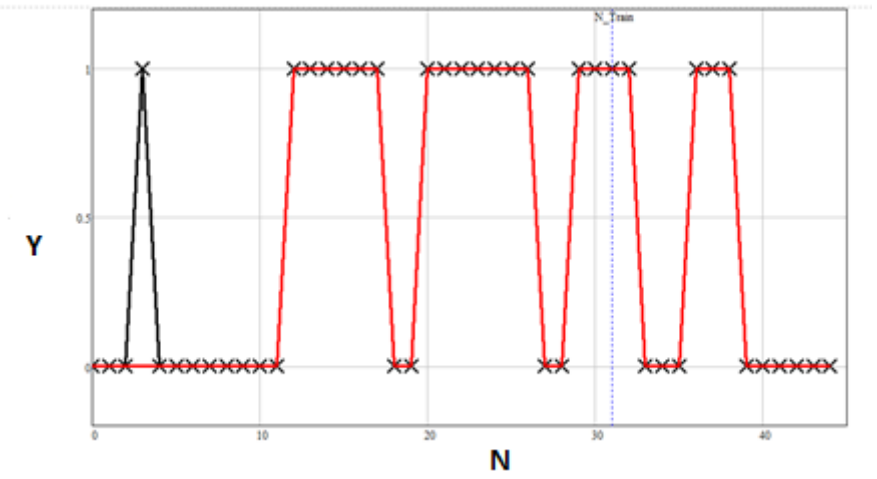


Рисунок 51 – График преодолимости на выборках A,B эксперимента 8

При подборе оптимальных параметров и исправлении ошибки эксперимента №2 исходная кривая и результат классификации полностью совпали (рисунок 52) на обучающей и проверочной выборках.

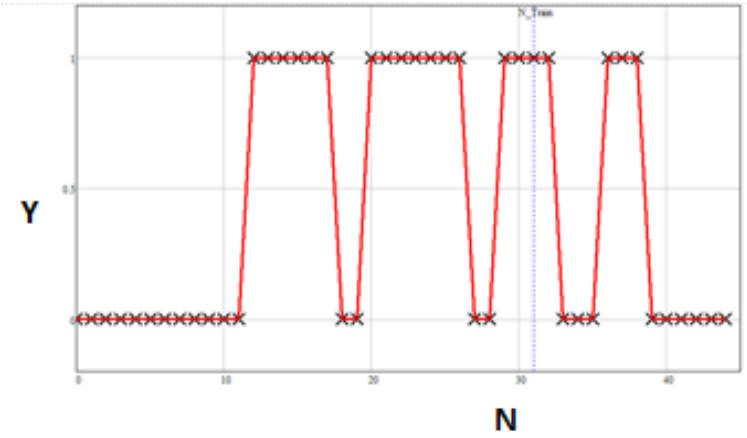


Рисунок 52 – График преодолимости на выборках A,B эксперимента 18

Этап обучения блока преодолимости можно подытожить следующими выводами. Наилучшие проценты верной классификации получены при максимальной степени полинома 3, размере сети 10x10, равном отношении обучающей и проверочной выборок (22:33 либо 25:26). Уменьшение параметра S на 3 порядка не повлияло на результаты, так как этот параметр в основном не используется сетью для формирования полиномов. Изменения в выборках (2, 3) также не оказали существенного влияния как на точность классификации на выборках A, B, так и на C. Однако исправление ошибки в

тесте 2 и увеличение выборок A и B (изменения 4 и 5) позволили увеличить процент верной классификации. Во всех экспериментах процент верной классификации на тестовой выборке C меньше, чем на обучающей и проверочной (за исключением 14-го эксперимента, но там преобладание тестовой выборки невелико). При построении сетей 10x10 и больше, часто наблюдается индуцит, что говорит о неэффективности построенных моделей. Аргументы, расположенные по убыванию частоты активного использования в построении классификатора: R(x4) –19 раз, G(x5) – 18, L(x2) –14, Vy(x0) – 14, S(x1) –4, D(x3) – 4, B(x6) – 4, Xc(x7) – 3, Yc(x8) – 1, A(x9) – 1. Таким образом, как и предполагалось в разделе 4, цвет является важнейшим аргументом. Однако нейронная сеть также активно использует длину контура и скорость, что не было замечено в разделе 4. Наименее востребованные параметры – центр масс и расстояние до контура, что близко к таблице 8.

Заключение

По итогам работы была создана система оценки преодолимости участков физически неоднородной среды. Разработаны алгоритмы обработки изображения, направленные на качественное оконтуривание и извлечение параметров. Создан классификатор на основе МГУА. Во время разработки проведены несколько исследований. По итогам исследования методов оконтуривания были получены данные для дальнейшего синтеза алгоритма автоматического выбора метода оконтуривания. По итогам тестирования классификатора выяснилось, что степени влияния параметров контура на преодолимость, оцененные автором, оказались близки к тем, на основании которых работал классификатор. Также исследовано влияние параметров нейронной сети и соотношения/наполнения выборок на качество работы классификатора. Создан задел для будущей работы, направленной на совершенствование методов обработки изображений и на разработку более точного классификатора.

Список использованных источников

1. Seraji, H. Traversability index: a new concept for planetary robots // Robotics and Automation in Space. – 1-3 June 1999. – Pages 159-166;
2. Gennery, D. B. Traversability Analysis and Path Planning for a Planetary Rover // Autonomous Robots 6. – 1999. – Pages 131-146;
3. Sanjiv, S., Simmons T., Smith T. Recent Progress in Local and Global Traversability for Planetary Rovers // Robotics and Automation. – 2000. – Pages 1194 - 1200 vol.2;
4. Shirkhodaie, A. Mobile Robots Traversability Awareness based on Terrain Visual Sensing Data Fusion // Unmanned Systems Technology IX. – 2007. – Vol. 6561, 65611U;
5. Jin, G. Development of a Traversability Map for Safe Navigation of Autonomous Mobile Robots // Journal of Institute of Control, Robotics and Systems. – 2014 20(4). – Pages 449-455;
6. EL-KABBANY, A., RAMIREZ-SERRANO, A. EFFECT OF NUMBER OF WHEELS ON HIGH SPEED UGV TRAVERSABILITY: ONLINE TERRAIN ASSESSMENT APPROACH // International Journal of Automotive Technology. – 2013. – Pages 249−257 Vol. 14, No. 2;
7. Tanaka, Y., Ji,T. Fuzzy Based Traversability Analysis for a Mobile Robot on Rough Terrain // International Conference on Robotics and Automation. – 2015. – Pages 3965-3970;
8. Cappalunga, A., Cattani, S., Broggi, A. Real Time 3D Terrain Elevation Mapping Using Ants Optimization Algorithm and Stereo Vision // IEEE Intelligent Vehicles Symposium. – 2010. – Pages 902-909;
9. Moghadam, P., Wijesoma W. S., Towards A Fully-Autonomous Vision-based Vehicle Navigation System in Outdoor Environments // Control Automation Robotics & Vision. – 2010. – Pages 597-602;
10. Wang, W., Shem, M. Visual Traversability Analysis for Micro Planetary Rover // International Conference on Robotics and Biomimetics. – 2009. – Pages 907-912;

11. Чернухин, Ю. В., Доленко, Ю. С., Бутов, П. А. НЕЙРОСЕТЕВОЙ ПОДХОД К РЕШЕНИЮ ЗАДАЧИ ЛОКАЛЬНОЙ НАВИГАЦИИ ИНТЕЛЛЕКТУАЛЬНЫМИ МОБИЛЬНЫМИ РОБОТАМИ В УСЛОВИЯХ, ПРИБЛИЖЕННЫХ К РЕАЛЬНОЙ СРЕДЕ // Известия Южного федерального университета. – 2013. – Страницы 80-84;
12. Dargazany, A., Berns, K. Terrain Traversability Analysis using Organized Point Cloud, Superpixel Surface Normals-based segmentation and PCA-based Classification // Workshop on Field and Assistive Robotics. – 2014/9;
13. Чернухин, Ю. В., Доленко, Ю. С., Бутов, П. А. БИОНИЧЕСКИЕ ПОДХОДЫ К ОБРАБОТКЕ СЕНСОРНОЙ ИНФОРМАЦИИ В НЕЙРОСЕТЕВЫХ СИСТЕМАХ УПРАВЛЕНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ МОБИЛЬНЫХ РОБОТОВ // // Известия Южного федерального университета. – 2012. – Страницы 194-199, том 130;
14. Метод наименьших квадратов. [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Метод_Наименьших_Квадратов. Дата обращения: 07.12.15.
15. Андраханов А.А. Технология управления автономным мобильным роботом на основе индуктивного метода самоорганизации моделей // Робототехника и техническая кибернетика. – Изд-во ЦНИИ РТК, Санкт-Петербург, 2014. – Т. 2. – № 1. – С. 38-44.
16. Robotino Manual. [Электронный ресурс] URL: http://www.festo-didactic.com/int-en/services/printed-media/technical-documentation/robotino-manual-544305.htm?fbid=aW50LmVuLjU1Ny4xNy4zMi45MDguNjcwOA (дата обращения: 20.05.2016)
17. Веб-камера Lab-Tech WebCam pro. [Электронный ресурс] URL: https://market.yandex.ru/product/4649858/spec (дата обращения: 20.05.2016).
18. Tyryshkin A.V., Andrakhanov A.A. Application of GMDH algorithms in the Obstacle Recognition Problem for Autonomous Mobile Robots // International Journal “Pattern Recognition and Image Analysis: Advanced in Mathematical Theory and Applications”, Vol. 19, №1, 2009. – P. 197-203.

19. Tyryshkin A.V., Andrakhanov A.A., Orlov A.A. GMDH-based Modified Polynomial Neural Network Algorithm // Chapter 6 in Book GMDH-methodology and implementation in C (With CD-ROM) / под общ. ред. G. Onwubolu. – London: Imperial College Press, World Scientific, 2015. – 304p. – ISBN: 978-1-84816-610-3.

Приложение А
(справочное)

Программный код обработки изображения и извлечения параметров контуров

```
#define _USE_MATH_DEFINES
#include <cmath>
#include <iostream>

#include "stdafx.h"
//#include "C:\opencv2.4\sources\include\opencv\cv.h"
#include <cv.h>
#include <highgui.h>
/*#include <imgproc.hpp>
#include <core.h>*/
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
//#include "opencv2/imgproc/imgproc.hpp"
//#include "opencv2/highgui/highgui.hpp"

#include "rec/robotino/com/all.h"
#include "rec/core_lt/utils.h"
#include "rec/core_lt/Timer.h"

using namespace cv;
using namespace std;
using namespace rec::robotino::com;

#define CV_RETR_EXTERNAL 0 // найти только крайние внешние контуры
#define CV_RETR_LIST     1 // найти все контуры и разместить их списком
#define CV_RETR_CCOMP    2 // найти все контуры и разместить их в виде 2-уровневой
иерархии
#define CV_RETR_TREE     3 // найти все контуры и разместить их в иерархии
вложенных контуров

#define CV_CHAIN_CODE               0 // цепной код Фридмана
#define CV_CHAIN_APPROX_NONE        1 // все точки цепного кода переводятся в
точки
#define CV_CHAIN_APPROX_SIMPLE      2 // сжимает горизонтальные, вертикальные и
диагональные сегменты и оставляет только их конечные точки
#define CV_CHAIN_APPROX_TC89_L1     3 // применяется алгоритм
#define CV_CHAIN_APPROX_TC89_KCOS   4 // аппроксимации Teh-Chin
#define CV_LINK_RUNS                5 // алгоритм только для CV_RETR_LIST

//параметры для изменения
      int act = 1;//для настройки изображения перед тестом
      int test = 0;

      int thresh = 100; // канни
      int max_thresh = 255;
      int Range_min = 30; //порог
      int Range_max = 137;

      int iterations_of_moving = 60; // движение робота
      int x_velocity = 500;
      int y_velocity = 0;
      int ang_velocity = 0;
```

```
//константы: движение робота

//для getch
int g = 0;

//счетчик для времени движения
int h = 0;

//измерение скорости
float velocity_sum_1 = 0;
int tic_tack = 0;
float average_velocity_1 = 0;

float velocity_sum_2 = 0;
float average_velocity_2 = 0;

float velocity_sum_3 = 0;
float average_velocity_3 = 0;

float current_sum_1 = 0;
float current_sum_2 = 0;
float current_sum_3 = 0;
float average_current_1 = 0;
float average_current_2 = 0;
float average_current_3 = 0;

int current_position1 = 0;
int current_position2 = 0;
int current_position3 = 0;

int last_position1 = 0;
int last_position2 = 0;
int last_position3 = 0;

/*int enc_pos = 0;
int enc_vel = 0;*/

class MyCom : public Com
{
public:
	MyCom()
	{
	}

	void errorEvent(Error error, const char* errorString)
	{
		std::cerr << "Error: " << errorString << std::endl;
	}

	void connectedEvent()
	{
		std::cout << "Connected." << std::endl;
	}

	void connectionClosedEvent()
	{
		std::cout << "Connection closed." << std::endl << std::endl;
	}
};

MyCom com;
Motor motor1;
Motor motor2;
Motor motor3;
OmniDrive omniDrive;
Odometry odometry;
```

```
Bumper bumper;
//MyInfo info;

void init(const std::string& hostname)
{

	// Initialize the actors
	motor1.setMotorNumber(0);

	motor2.setMotorNumber(1);

	motor3.setMotorNumber(2);

	// Connect
	std::cout << "Connecting..." << std::endl;
	com.setAddress(hostname.c_str());
	com.connect();

	std::cout << std::endl << "Connected" << std::endl;
}

void drive()
{
	rec::core_lt::Timer timer;
	timer.start();

	/*while (com.isConnected()
	&& false == bumper.value()
	&& timer.msecsElapsed() < time_of_moving)
	{*/

	//движение
	omniDrive.setVelocity(x_velocity, y_velocity, ang_velocity);
	com.waitForUpdate();
	//}
}

void destroy()
{
	com.disconnect();

}

//константы: обработка изображения
//IplImage* image = 0;
//IplImage* gray = 0;
IplImage* bin = 0;
IplImage* dst = 0;

IplImage* image = 0;

//для контуров по цвету
IplImage* rgb = 0;
IplImage* r_plane = 0;
IplImage* g_plane = 0;
IplImage* b_plane = 0;

IplImage* r_range = 0;
IplImage* g_range = 0;
IplImage* b_range = 0;

IplImage* rgb_and = 0;

//IplImage* frame;

//задаем минимальные значения каналов
int Rmin = 0;
```

```
int Rmax = 255;

int Gmin = 50;
int Gmax = 255;

int Bmin = 50;
int Bmax = 255;

int RGBmax = 256;

class Contour_finale {
	//friend ostream& operator<<(ostream& os, const Auto& at);
public:
	void set_number_of_contour(int number)
	{
		contour_number = number;
	}
	void set_area(double ar)
	{
		contour_area = ar;
	}
	void set_contour_output(vector<Point> contour_output_temp)
	{
		contour_output = contour_output_temp;
	};

	vector<Point> contour_output;
	int contour_number;
	double contour_area;
	/*ostream& operator<<(ostream& os, const Auto& at)
	{
	os << "номер контура " << at.contour_number << "\n";
	return os;
	}*/
private:
	//float contour_number;
	double perim;
	double FloorOrientation;
	char color[10];
};

int j = 0;
int fulled = 0;
double ar;
int mis_match = 0;
Mat src_gray;
Mat src_gray_1;
RNG rng(12345);

Mat myMat;
Mat myMat1;
Mat common;
//Mat drawing;

Mat Color_RangeS_output;

Contour_finale contour_finale[30];

//для войд
Mat canny_output;
Mat RangeS_output;

vector<vector<Point>> contours;
vector<Vec4i> hierarchy;

vector<Point> massiv_vector;
```

```
//для первого кадра
int for_first_frame = 0;

//для подсчета времени
int t;

void thresh_callback(int, void*);


int main(int argc, char* argv[])
{
	setlocale(LC_CTYPE, "rus");

	// получаем любую подключённую камеру
	CvCapture* capture = cvCreateCameraCapture(CV_CAP_DSHOW);
//cvCaptureFromCAM( 0 );

	if (!capture)
	{
		printf("Ooops! Camera Error");
	}
	assert(capture);

	// узнаем ширину и высоту кадра
	double width = cvGetCaptureProperty(capture, CV_CAP_PROP_FRAME_WIDTH);
	double height = cvGetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT);
	printf("[i] %.0f x %.0f\n", width, height);


	//Создание изображений
	//изображение записи кадра
	IplImage* frame1 = cvQueryFrame(capture);
	//IplImage* frame2;
	if (!frame1)
	{
		printf("Ooops! #1 cvQueryFrame Error");
		return -1;
	}

	//создание картинки для градаций серого
	/*IplImage* gray = cvCreateImage(cvGetSize(frame), IPL_DEPTH_8U, 1);
	if (!gray)
	{
	printf("Ooops! #2 gray Error");
	return -1;
	}
	//еще пару бинарных
	bin = cvCreateImage(cvGetSize(frame), IPL_DEPTH_8U, 1);
	dst = cvCreateImage(cvGetSize(frame), IPL_DEPTH_8U, 1);*/

	//изображения для разделения по каналам
	r_plane = cvCreateImage(cvSize(width, height), IPL_DEPTH_8U, 1);
	g_plane = cvCreateImage(cvSize(width, height), IPL_DEPTH_8U, 1);
	b_plane = cvCreateImage(cvSize(width, height), IPL_DEPTH_8U, 1);

	r_range = cvCreateImage(cvSize(width, height), IPL_DEPTH_8U, 1);
	g_range = cvCreateImage(cvSize(width, height), IPL_DEPTH_8U, 1);
	b_range = cvCreateImage(cvSize(width, height), IPL_DEPTH_8U, 1);
	rgb_and = cvCreateImage(cvSize(width, height), IPL_DEPTH_8U, 1);


	printf("[i] press Enter for capture image and Esc for quit!\n\n");

	int counter = 0;
	char filename[512];

	namedWindow("Source", CV_WINDOW_AUTOSIZE);
```

```
	namedWindow("rec", CV_WINDOW_AUTOSIZE);
	namedWindow("Trackbar", CV_WINDOW_AUTOSIZE);

	//робот. раскомментить init!!!

	std::cout << "example_forward" << std::endl;

	std::string hostname = "172.26.1.0";
	if (argc > 1)
	{
		hostname = argv[1];
	}

	//проверка изображения, настройка перед тестом
	if (test == 0){
		init(hostname);
	}

	//в одном цикле чередование снятя кадра и движения робота

	while (true){
		//1. работа с кадром
		// получаем кадр
		frame1 = cvQueryFrame(capture);
		//клонируем оригинальный кадр для рисования контуров
		dst = cvCloneImage(frame1);

		//сделать паузу для первого цикла чотбы камера настроила баланс
белого
		if (for_first_frame == 0){
			cvWaitKey(7500);
			for_first_frame = 1;
		}

		if (!frame1)
		{
			printf("Ooops! #2 cvQueryFrame Error");
			break;
		}

		//тестирование изображения

		Mat myMat(frame1);
		//разделяем кадр на каналы
		cvSplit(frame1, b_plane, g_plane, r_plane, 0);

		//инвертировать если светлый преобладает
			//Mat myMat1(480, 640, CV_32FC1);

			/*cv::Mat myMat1(480, 640, CV_32FC1);*/
			//invert(myMat, myMat1);
		//преобразуем в градации серого
		cvtColor(myMat, src_gray, CV_RGB2GRAY);
		//blur(src_gray, src_gray, Size(3, 3)); //
src_gray.inv(cv::DECOMP_CHOLESKY)
		imshow("Source", src_gray);

		//создаем трекбары//уточнить максимальные значения диапазона
		createTrackbar(" Canny thresh:", "Trackbar", &thresh, max_thresh,
thresh_callback);
		createTrackbar(" Range min:", "Trackbar", &Range_min, 100,
thresh_callback);
		createTrackbar(" Range max:", "Trackbar", &Range_max, 400,
thresh_callback);

		//трекбары для красного канала
		createTrackbar("Rmin", "Trackbar", &Rmin, RGBmax, thresh_callback);
```

```
            createTrackbar("Rmax", "Trackbar", &Rmax, RGBmax, thresh_callback);
            //трекбары для синего канала
            createTrackbar("Gmin", "Trackbar", &Gmin, RGBmax, thresh_callback);
            createTrackbar("Gmax", "Trackbar", &Gmax, RGBmax, thresh_callback);
            //трекбары для зеленого канала
            createTrackbar("Bmin", "Trackbar", &Bmin, RGBmax, thresh_callback);
            createTrackbar("Bmax", "Trackbar", &Bmax, RGBmax, thresh_callback);

            //вызов функции обработки порогов
            system("cls");
            thresh_callback(0, 0);

            //2. работа с роботом

            char c = cvWaitKey(33);
            if (c == 27) { // нажата ESC
                  break;
            }
            else if (c == 13) { // Enter
                  g++;
            }

            //счетчик этапов
            if (g == 1) {
                  std::cout << std::endl;
                  std::cout << "Движение до препятствия" << std::endl;
                  drive();
            }
            if (g == 2) {
                  drive();

                  std::cout << std::endl;
                  std::cout << "Движение по препятствию, запись параметров" <<
std::endl;

                  //вычисление средней скорости мотора на участке (активной либо
пассивной)
                  velocity_sum_1 = velocity_sum_1 + motor1.actualVelocity();
                  velocity_sum_2 = velocity_sum_2 + motor2.actualVelocity();
                  velocity_sum_3 = velocity_sum_3 + motor3.actualVelocity();

                  //вычисление позиции мотора
                  current_position1 = motor1.actualPosition() - last_position1;
                  current_position2 = motor2.actualPosition() - last_position2;
                  current_position3 = motor3.actualPosition() - last_position3;

                  //ток мотора
                  current_sum_1 = current_sum_1 + motor1.motorCurrent();
                  current_sum_2 = current_sum_2 + motor2.motorCurrent();
                  current_sum_3 = current_sum_3 + motor3.motorCurrent();
                  tic_tack++;
            }

            else if (g == 3){
                  std::cout << std::endl;
                  destroy();

                  //вывод параметров робота
                  std::cout << "Results: " << std::endl;
                  average_velocity_1 = velocity_sum_1 / tic_tack;
                  std::cout << "Motor_1 velocity (rpm): " << average_velocity_1
<< std::endl;

                  average_velocity_2 = velocity_sum_2 / tic_tack;
                  std::cout << "Motor_2 velocity: " << average_velocity_2 <<
std::endl;
```

```
                    average_velocity_3 = velocity_sum_3 / tic_tack;
                    std::cout << "Motor_3 velocity: " << average_velocity_3 <<
std::endl;

                    average_current_1 = current_sum_1 / tic_tack;
                    std::cout << "Motor_1 current: " << average_current_1 <<
std::endl;

                    average_current_2 = current_sum_2 / tic_tack;
                    std::cout << "Motor_2 current: " << average_current_2 <<
std::endl;

                    average_current_3 = current_sum_3 / tic_tack;
                    std::cout << "Motor_3 current: " << average_current_3 <<
std::endl;

                    std::cout << "Энкодер_1: " << current_position1 << std::endl;
                    std::cout << "Энкодер_2: " << current_position2 << std::endl;
                    std::cout << "Энкодер_3: " << current_position3 << std::endl;

                    std::cout << "Время: " << clock() - t << std::endl <<
std::endl;

                    rec::core_lt::waitForKey();
                    /*if (c == 27) { // нажата ESC
                    break;
                    }*/
              }
              if (tic_tack < 1){

                    //начало подчета времени
                    t = clock();

                    last_position1 = motor1.actualPosition();
                    last_position2 = motor2.actualPosition();
                    last_position3 = motor3.actualPosition();
              }

              /*if (h < iterations_of_moving){
              //подсчет его времени

              h++;
              }     */

       }

       // освобождаем ресурсы
       cvReleaseCapture(&capture);
       cvReleaseImage(&b_plane);
       cvReleaseImage(&g_plane);
       cvReleaseImage(&r_plane);
       cvReleaseImage(&frame1);
       //cvReleaseData(&myMat);

       cvDestroyWindow("Source");
       cvDestroyWindow("Trackbar");
       cvDestroyWindow("rec");

       return 0;
}
void thresh_callback(int, void*)
{
       //блок 1: выбор метода оконтуривания


       //блок 2: нахождение контуров с одного кадра/фильтрация
              //Mat drawing = Mat::zeros(RangeS_output.size(), CV_8UC3);
```

```
	//1 - нахождение контуров по канни
	/*Canny(src_gray, canny_output, thresh, thresh * 2, 3);
	findContours(canny_output, contours, hierarchy, CV_RETR_TREE,
CV_CHAIN_APPROX_SIMPLE, Point(0, 0));
	Mat common(dst);*/

	//2/ пороговое преобразование по градации серого

	inRange(src_gray, Scalar(Range_min), Scalar(Range_max), RangeS_output); //
atoi(argv[2])

	findContours(RangeS_output, contours, hierarchy, CV_RETR_TREE,
CV_CHAIN_APPROX_SIMPLE, Point(0, 0));

	//не получается аппроксимировать
		//vector<vector<Point>> contours_poly(contours.size());
	Mat common(dst);


	//3/ порог по цвету...
	//cvCopyImage(frame, rgb);
	/*cvInRangeS(r_plane, cvScalar(Rmin), cvScalar(Rmax), r_range);
	cvInRangeS(g_plane, cvScalar(Gmin), cvScalar(Gmax), g_range);
	cvInRangeS(b_plane, cvScalar(Bmin), cvScalar(Bmax), b_range);
	// складываем
	cvAnd(r_range, g_range, rgb_and);
	cvAnd(rgb_and, b_range, rgb_and);

	Mat Color_RangeS_output(rgb_and);

	findContours(Color_RangeS_output, contours, hierarchy, CV_RETR_TREE,
CV_CHAIN_APPROX_SIMPLE, Point(0, 0));
	Mat common(dst);*/

	j = 0;
	for (int i = 0; i< contours.size(); i++)
	{
		//фильтр по одному параметру (вычисляем площадь текущего контура)
		double area_init = fabs(contourArea(contours[i]));
		if (area_init > 3000){
			//обработка
			//approxPolyDP(Mat(contours[i]), contours_poly[i], 3, true);
			contour_finale[j].set_contour_output(contours[i]); //записываем
полезные контура в новый массив
			drawContours(common, contours, i, CV_RGB(255, 216, 0), 1, 8,
hierarchy, 0, Point()); //рисуем контура из старого массива (только те, что больше
барьера) (из нового массива не рисуется!)
			contour_finale[j].contour_area = area_init; //параметр
контуров: площадь
			j++;

			//сохраняем количество полезных контуров для нахождения
параметров
			fulled = j;

		}
	}

	//блок 5: нахождение остальных параметров полезных контуров
	//!!!записать эти свойства в методы классы
	//моменты контуров
	vector<Moments> mu(fulled);
	for (int i = 0; i < fulled; i++)
	{
		mu[i] = moments(contour_finale[i].contour_output, false);
		//std::cout << "Момент контура " << mu[i] << std::endl;
```

```
	}

	//центры масс (координаты)
	vector<Point2f> mc(fulled);
	float red;
	float green;
	float blue;

	for (int i = 0; i < fulled; i++)
	{
		mc[i] = Point2f(mu[i].m10 / mu[i].m00, mu[i].m01 / mu[i].m00);

		std::cout << "Центр масс контура " << i << ": " << mc[i] <<
std::endl;
		std::cout << "Площадь контура " << i << ": " <<
contour_finale[i].contour_area << std::endl;


		//определение цвета пикселя центра масс
		Vec3b intensity = common.at<Vec3b>(Point(mc[i].x, mc[i].y));
		blue = intensity.val[0];
		green = intensity.val[1];
		red = intensity.val[2];
		circle(common, mc[i], 8, CV_RGB(red, green, blue), -1, 8, 0);
		std::cout << "Цвет контура " << i << ": " << intensity << std::endl;

		//определение расстояния до центра масс
		float distance = pow(318, 3)*pow((mc[i].y + 617), -2);
		std::cout << "расстояние до центра масс контура " << i << ": " <<
distance << std::endl;
	}

	vector<Rect> rect_contour_output(fulled);
	for (int i = 0; i < fulled; i++){
		rect_contour_output[i] =
boundingRect(contour_finale[i].contour_output);
		//cvRectangle(dst, rect_contour_output[i].tl(),
rect_contour_output[i].br(), CV_RGB(0, 216, 0), 2, 5, 0);
		rectangle(common, rect_contour_output[i].tl(),
rect_contour_output[i].br(), CV_RGB(0, 216, 0), 2, 5, 0);

		//подсчет пикселей идет слева сверху!!!
		//определение нижней и верхней координаты прямоугольника
		float Y_up = rect_contour_output[i].y;
		float Y_down = rect_contour_output[i].y +
rect_contour_output[i].height;

		/*std::cout << "Верхняя координата " << i << " : " << Y_up <<
std::endl;
		std::cout << "Нижняя координата " << i << " : " << Y_down <<
std::endl;*/

		//ширина контура в пикселях
		int width = rect_contour_output[i].width;
		std::cout << "Ширина контура " << i << " : " << width << std::endl;

		//нахождение длины контура по
		float length = pow(318, 3)*pow((Y_up + 617), -2) - pow(318,
3)*pow((Y_down + 617), -2);
		std::cout << "Длина контура " << i << " : " << length;
	}
	//std::cout << std::endl;
	imshow("rec", common);
};
```