Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Оренбургский государственный университет»

Кафедра вычислительной техники

Т.З. Аралбаев, А.Ю. Кручинин, Р.И. Хасанов

СИСТЕМЫ РАСПОЗНАВАНИЯ ОБРАЗОВ

Рекомендовано к изданию Редакционно-издательским советом
федерального государственного бюджетного образовательного учреждения
высшего профессионального образования
«Оренбургский государственный университет»
в качестве методических указаний для студентов, обучающихся по программам
высшего профессионального образования по направлению подготовки
090900.62 Информационная безопасность

Оренбург
2012

УДК 004.93(076.5)
ББК 32.973.2-018.02я7
А 79

Рецензент- кандидат технических наук, доцент А.В. Хлуденев

Аралбаев, Т.З.
А 79 Системы распознавания образов: методические указания к лабораторным работам / Т.З. Аралбаев, А.Ю. Кручинин, Р.И. Хасанов; Оренбургский гос. ун-т. – Оренбург: ОГУ, 2012. – 67 с.

Лабораторный курс содержит 8 лабораторных работ и методические указания к ним. Каждая работа включает теоретическое изложение материала, постановку задачи, порядок выполнения и контрольные вопросы для самоподготовки.

Лабораторный курс рекомендован преподавателям, как вспомогательный материал в организации и проведении занятий, а также студентам, обучающимся по специальности и профилю «Комплексная защита объектов информатизации» для аудиторного и самостоятельного освоения лабораторного курса дисциплины «Системы распознавания образов».

УДК 004.93(076.5)
ББК 32.973.2-018.02я7

Содержание

Обозначения и сокращения

БПФ – быстрое преобразование Фурье;

ГА – генетический алгоритм;

ГО – графический образ;

ДП – дорожное полотно автомобильной дороги;

ЗУ – запоминающее устройство;

ИП – информативный признак;

ППР – правило принятия решения;

РФ – разделяющая функция;

РП – разделяющее правило;

СА – солнечная активность;

СГ – сетевой график;

СПО – система параметрической оптимизации;

СРО – система распознавания образов;

УРО – устройство распознавания образов.

Введение

Настоящий лабораторный практикум предназначен для получения практических навыков студентами специальности и профиля подготовки – «Комплексная защита объектов информатизации» при изучении дисциплины «Системы распознавания образов» (СРО).

Лабораторный курс содержит восемь работ, рассчитанных на 36 часов аудиторных занятий. Предлагаемые задания охватывают основные разделы рабочей программы, связанные с изучением технологии и организации проектирования систем распознавания образов.

Общие методические рекомендации по использованию лабораторных работ и методических указаний:

– к выполнению лабораторной работы следует приступать после ознакомления с теоретической частью соответствующего раздела и рекомендациями, приведенными в конкретной работе;

– лабораторные работы рекомендуется выполнять в порядке их нумерации в аудиторное время, указанное в описании работы;

– рекомендуется для экономии времени отчеты о лабораторных работах оформлять в виде протоколов работы с обязательным указанием даты, номера, темы, цели работы и выводов с краткой характеристикой результата;

– дополнительные сведения по лабораторным работам содержатся в прилагаемом списке литературы.

Лабораторный курс может быть освоен на индивидуальном компьютере со средними техническими характеристиками. Обязательным для полной реализации курса является наличие комплекта офисных приложений и объектно-ориентированной среды программирования Delphi, желательно иметь выход в Интернет.

Практикум рекомендован преподавателям как вспомогательный материал в организации и проведении занятий, а также студентам – для аудиторного и самостоятельного освоения лабораторной части дисциплины СРО.

1 Лабораторная работа № 1. Классификация периодов солнечной активности

Цель работы: построить экспериментальную классификацию периодов солнечной активности (СА) на основе исследования и кодирования параметров спектров

1.1 Теоретическая часть

Исследование солнечной активности относится к перечню задач, которым длительное время уделяют внимание специалисты из разных областей науки. Предметом исследования являются данные временных рядов, так называемые числа Вольфа, регистрация которых известна с 1700 года по настоящее время. Цель проводимых исследований - выявление влияния солнечной активности на целый ряд земных процессов, например: эпидемии болезней, урожайность сельскохозяйственных культур, стихийные бедствия, изменения в обществе и другие. В настоящей и последующей работе рассматриваются вопросы начального изучения этой проблемы.

Систематизацией наблюдений за солнечными пятнами впервые занялся цюрихский астроном Р. Вольф. Им, в частности, предложена следующая формула для ежемесячной оценки солнечной активности:

$$W = k \times (10 \times g + f), \qquad (1.1)$$

где W – число Вольфа;

g – число групп и отдельных пятен на Солнце;

f – число пятен в группах и отдельно;

k – поправочный коэффициент, учитывающий характеристики измерительных приборов.

На сайте WWW.astronom2000.info/астрономия/солнечная активность представлены ежемесячные наблюдения значений чисел Вольфа, начиная с 1749 года. Известно, что периодичность изменения солнечной активности близка к 11,2 года и составляет 134 месяца.

Классификация периодов СА необходима для упорядочивания и систематизации множества периодов солнечной активности в процессе ее исследования. Классификация является результатом кластерного анализа периодов солнечной активности, в котором выявляются общие и отличительные признаки спектров солнечной активности для объединения периодов в отдельные группы.

В кластерном анализе обычно выполняются следующие операции:

а) отбор выборки для анализа и классификации;

б) определение признаков классификации;

в) определение меры сходства между объектами;

г) построение собственно классификации;

д) проверка классификации.

Согласно пункту «а» по данным источника [9] в распоряжении исследователя имеются данные по периодам солнечной активности с 1749 по 2009 год.

В общем случае исследование СА преследует различные цели, причем изучается как спектр сигналов целиком, так и его отдельные гармоники. Перечень гармоник определяется условиями задачи. В настоящей работе, учитывая лишь ее учебно-познавательную значимость, а так же специфику применения БПФ в системе MathCAD, принимаются следующие допущения:

– перечень гармоник выбирается произвольно, например, с периодами три, четыре, шесть и двенадцать месяцев;

– при расчете спектров использовать данные за первые 128 месяцев каждого периода, т.е. данные усечены на 6 замеров.

Определение признаков классификации является отдельной задачей. Учитывая учебный характер настоящей работы, выбор их производится произвольно, т.е. для классификации периодов СА используем гармоники с периодами 3, 4, 6 и 12 месяцев, т.е. число признаков $\boldsymbol{i} = 4$.

Меру сходства между периодами солнечной активности определим по величинам кодов $\boldsymbol{K_n}$, соответствующих спектрам N периодов СА ($\boldsymbol{n = 1..N}$).

Код $\boldsymbol{K_n}$ определяется по следующим выражениям:

$$K_n = k_1, k_2, ..., k_i, ..., k_n, \qquad (1.2)$$

$$K_i = \begin{cases} 1, & \text{если} \quad S_{i,n} \geq m_i; \\ 0, & \text{если} \quad S_{i,n} < m_i; \end{cases} \tag{1.3}$$

$$m_i = \frac{\sum_{n=1}^{N} S_{i,n}}{N}, \tag{1.4}$$

где $S_{i,n}$ – амплитудная оценка для ***i***-ой гармоники (из числа выбранных) в ***n***-ом периоде СА;

m_i – среднее значение амплитудных оценок спектра для ***i*** -ой гармоники по всем исследуемым спектрам СА.

Таким образом, число ***j*** определяет максимально возможное число классов, на которое можно разбить исходную выборку периодов СА. Построение классификаций периодов производится объединением в группы периодов с одинаковыми кодами.

Построенная классификация проверяется на соответствие числа разбиваемых на классы периодов сумме периодов, охваченных всеми классами, а так же на непревышение общего числа классов величины 2^j.

1.2 Порядок выполнения работы

1. Создайте документ MathCad. Прочитайте данные о СА в вектор U. Перечень исследуемых периодов СА и файл с данными предоставляется преподавателем или формируется студентом по данным источника [9].

Вариант документа представлен на рисунке 1.1.

2. Рассчитать спектральные оценки СА по заданным периодам (например, для N последовательно чередующихся периодов) для гармоник, соответствующих 3, 4, 6 и 12 месяцам.

3. Построить по N точкам графики изменения амплитудных оценок гармоник с периодами 3, 4, 6 и 12 месяцев.

4. Рассчитайте среднее значение амплитудных оценок спектра для каждой выбранной гармоники по N периодам по формуле (1.3).

5. Вычислите классификационные признаки ***K*** по N периодам согласно формуле (1.2).

Возможный вариант документа MathCad для N=24 представлен на рисунке 1.2.

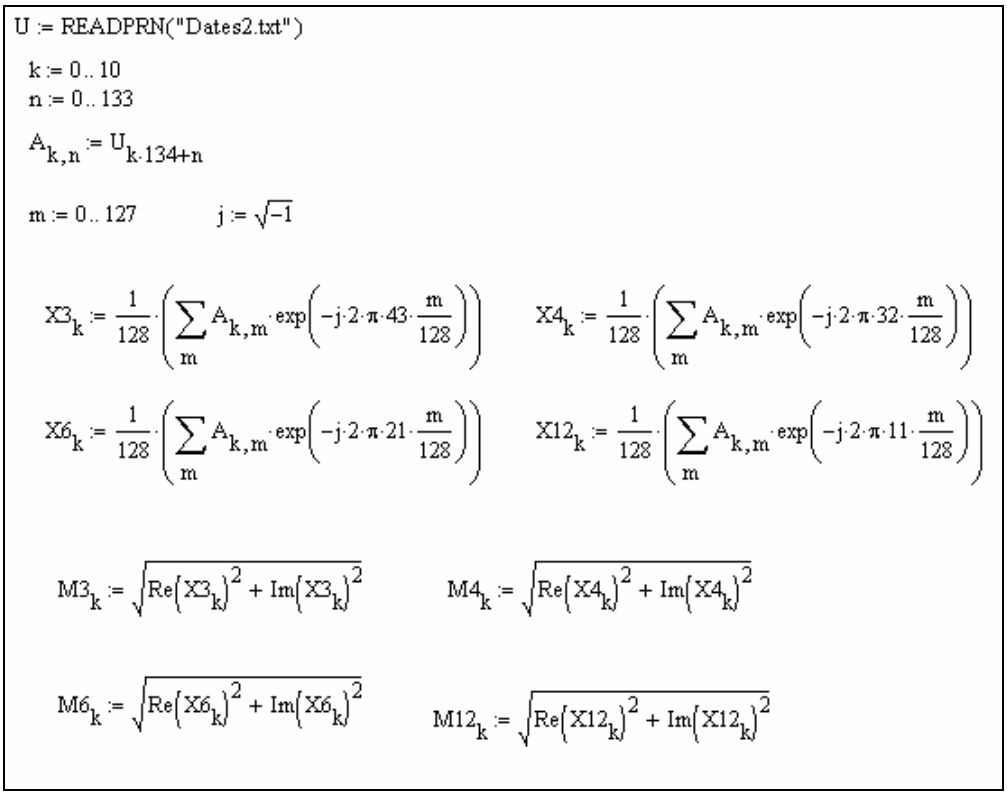


Рисунок 1.1 – Вычисление амплитудных оценок для исследуемых гармоник по периодам СА

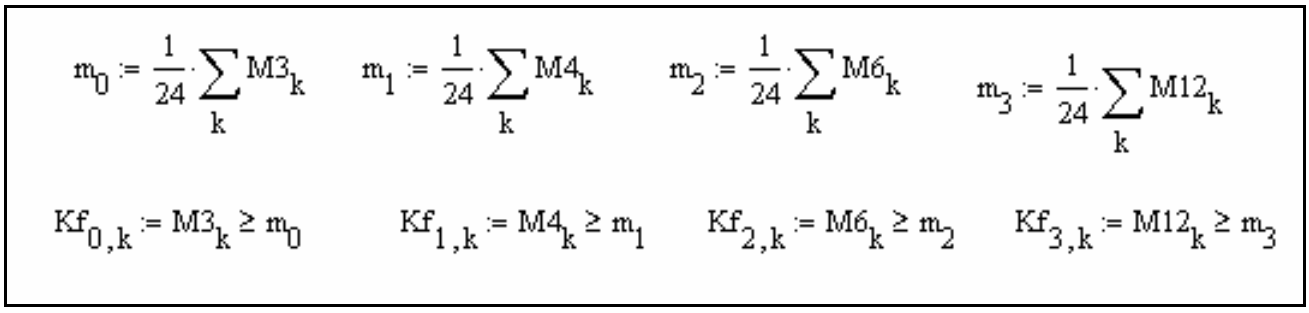


Рисунок 1.2 – Вычисление элементов классификационного кода периода СА

6. Произведите классификацию периодов СА, используя значения классификационных признаков ***K***. Периоды солнечной активности относятся к одному классу, если их классификационные коды полностью совпадают.

7. Сделайте выводы по работе и составьте отчет. В отчете должны быть представлены значения классификационных кодов каждого периода СА.

1.4 Контрольные вопросы

1. Порядок вычисления классификационного кода периода СА.
2. Как может быть использована на практике классификация периодов СА?
3. Как определить максимально возможное число классов по данной классификации?
4. Перечислите операции кластерного анализа.
5. Объясните понятие «корреляция данных».

2 Лабораторная работа № 2. Организация вычислительного процесса при решении задачи распознавания графического образа

Цель работы: освоить способ распараллеливания вычислительного процесса на примере задачи распознавания графического образа

2.1 Теоретическая часть

Эталонный графический образ (ГО) ***X*** представляется в формате BMP матрицей пикселов размером ***M*** столбцов на ***N*** строк. Исследуемый графический образ ***Y*** является частью образа ***X***, представляется в аналогичном формате и имеет размеры ***n*m***. Образ ***Y*** может быть зашумлен цветовыми помехами для исследования параметров распознавания. Процент зашумления (искажения) пикселов находится в диапазоне от 0 до 50%.

Суть задачи распознавания заключается в определении места положения фрагмента образа ***Y*** на исходном образе в условиях зашумления изображения фрагмента. Решение данной задачи в условиях больших размеров графических образов и значительной степени зашумления предоставляет для исследователя определенные сложности.

На рисунке 2.1 в качестве примера представлена постановки и решения задачи идентификации места положения фрагмента среза головного мозга при исследовании

его с помощью томографа. На рисунке 2.1 в пунктах “а” и “б” представлены соответственно образы **X** и **Y**, в пункте “в” – график коэффициента парной корреляции цветовых характеристик образов **X** и **Y** при скольжении образа **Y** по поверхности образа **X** слева на право.

Для формализации задачи распознавания введем следующие условные обозначения: $\boldsymbol{Q_Y}$ – образ фрагмента; $\boldsymbol{Q = \{q_1, q_2, \ldots, q_R\}}$ – множество вариантов образов основного образа; $\boldsymbol{P = \{p_1, p_2, p_3\}}$ – множество признаков распознавания; $\boldsymbol{RF = K_{XY}(P_X, P_Y)}$ – разделяющая функция; $\boldsymbol{RP}$ – разделяющее правило.

Число вариантов образов $\boldsymbol{R}$ при скольжении фрагмента образа по основному образу с шагом $\boldsymbol{\Delta M}$:

$$R = \frac{M - m}{\Delta M}. \tag{2.1}$$

Число признаков распознания соответствует числу основных цветов в цветовой модели графического образа ***RGB***, т.е. – трем.

В качестве разделяющей функции при распознавании места положения образа $\boldsymbol{Y}$ выбрана функция коэффициента парной корреляции $\boldsymbol{R_{XY}}$:

$$R_{xy} = \frac{K_{xy}}{\sigma_x \sigma_y} \tag{2.2}$$

$$K_{xy} = \frac{1}{L}\sum_{i=1}^{L}(x_i - \bar{x})(y_i - \bar{y}) \tag{2.3}$$

$$\sigma_z = \sqrt{\sigma_z^2} \tag{2.4}$$

$$\sigma_z^2 = \frac{1}{L}\sum_{i=1}^{L}(z_i - \bar{z})^2 \tag{2.5}$$

где $\boldsymbol{K_{xy}}$ – коэффициент ковариации;

$\boldsymbol{x_i}$ и $\boldsymbol{y_i}$ – соответственно текущие значения интенсивностей цвета по выборке;

$\bar{x}$ и $\bar{y}$ – соответственно, оценки средних значений интенсивностей цвета по выборке;

$\boldsymbol{L}$ – число элементов выборке;

z_i и $\overline{z}$ – соответственно текущие значения интенсивностей некоторого цвета и оценки среднего по выборке ($i = 1, L$);

σ – среднее квадратическое отклонение по выборке.

2.2 Порядок выполнения работы

На рисунке 2.1 представлена общая схема корреляционного метода для решения задачи.

Разделяющее правило для определения соответствия образа фрагмента Q_Y одному из вариантов образов основного образа Q выглядит следующим образом:

$$Q_Y \equiv Q^* \in Q : R_{XY}(P_X^*, P_Y) = R_{\max XY}(P_{X,}, P_Y), \qquad (2.6)$$

т.е. отнесение образа Q_y к одному из вариантов Q^*образов Q осуществляется по максимальной близости признаков образов.

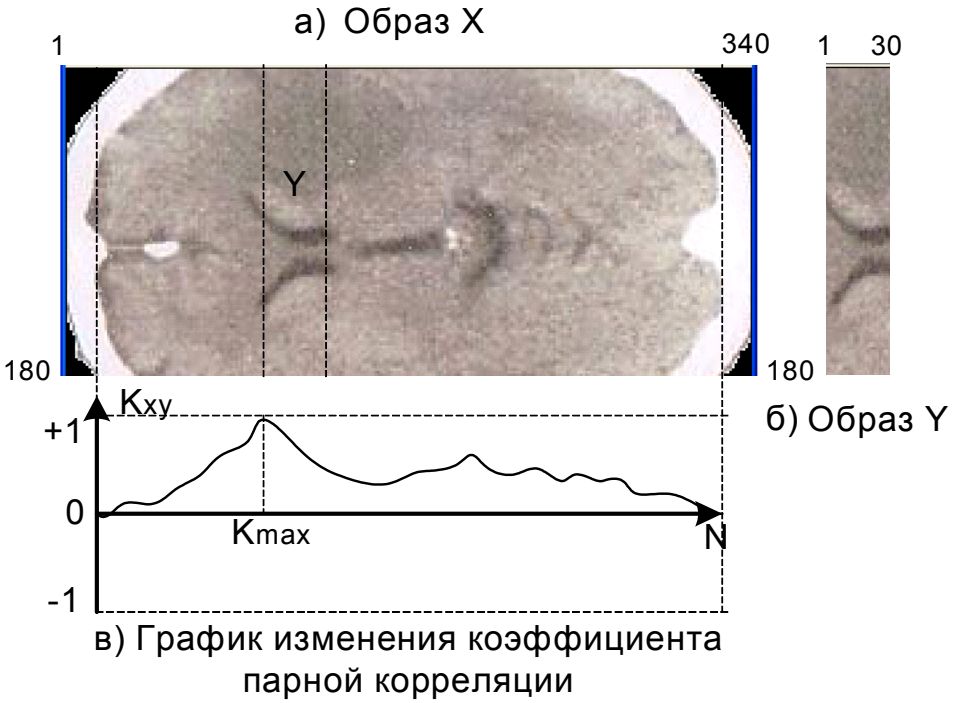


Рисунок 2.1 – Общая схема корреляционного метода решения задачи

На рисунке 2.2 представлена схема построения пространственных рядов признаков образов Q_Y и Q_X.

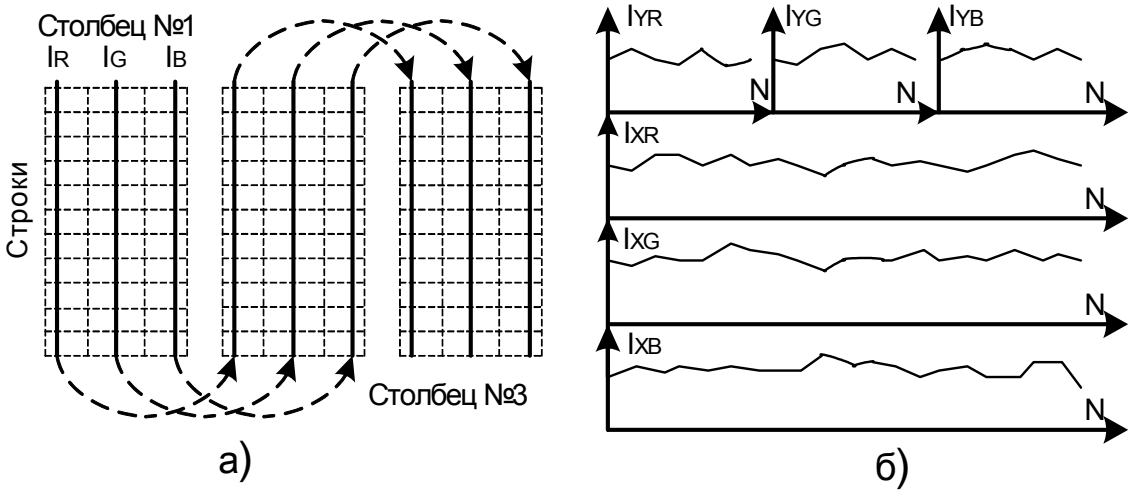


Рисунок 2.2 – Схема построения графиков интенсивностей цветов модели ***RGB*** графических образов

В частности, в пункте “а” представлен порядок формирования пространственных рядов интенсивностей цвета: красного ($\boldsymbol{I_R}$), зеленого ($\boldsymbol{I_G}$) и синего ($\boldsymbol{I_B}$) – по трем столбцам матрицы пикселов. Общее число столбцов соответствует числам $\boldsymbol{M}$ и $\boldsymbol{m}$ для исследуемых образов. В пункте “б” рисунка 2.2 представлены графики изменения интенсивности цветов для сравниваемых образов.

Расчет параметров по формулам (2.1) – (2.5) производится на данных пространственных рядов, представленных на рисунке 2.2 “б”, где $\boldsymbol{N}$ – число элементов пространственного ряда.

На рисунке 2.3 представлен общий вид демонстрационной программы идентификации места положения фрагмента $\boldsymbol{Q_Y}$ на поверхности образа $\boldsymbol{Q_X}$.

Как видно из рисунка 2.3, распознавание места положения фрагмента проведено по интенсивности зеленого цвета с уровнем зашумления в 10%. При этом $\boldsymbol{R_{maxXY}}$ = 0.896, номера начального и конечного столбца пикселов образа $\boldsymbol{Q_X}$, соответствующие фрагменту, имеют значения, соответственно, 105 и 135.

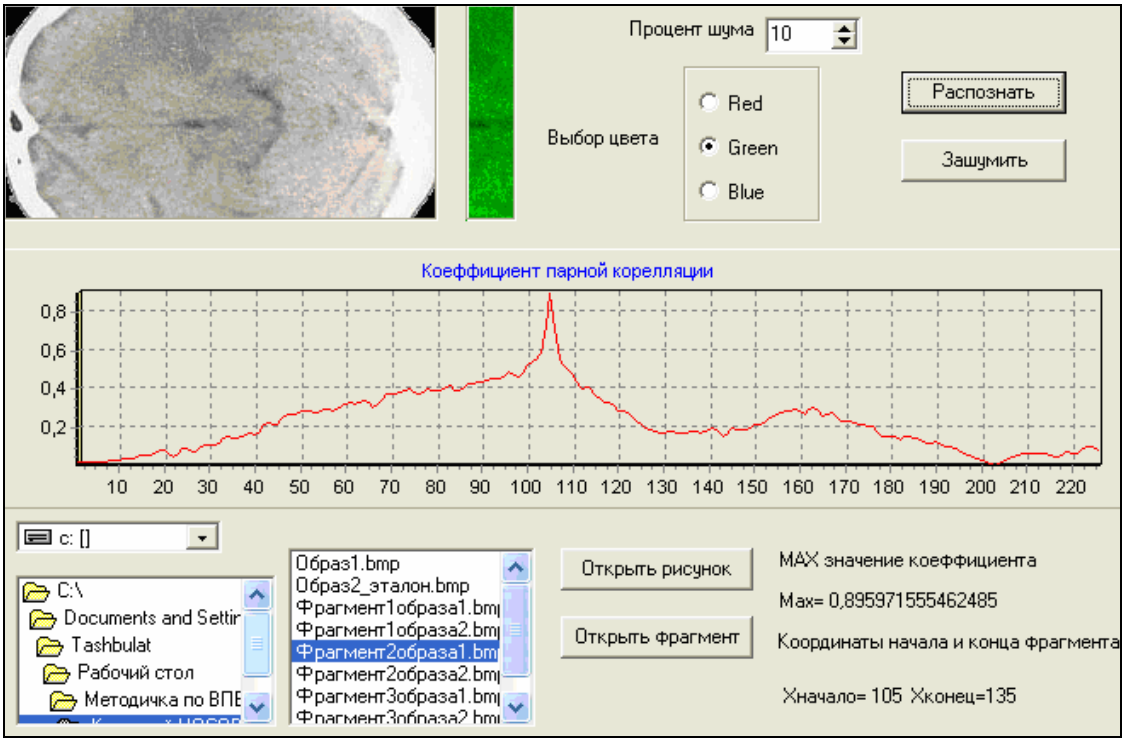


Рисунок 3.3 – Общий вид программы идентификации местоположения фрагмента

Анализ математического описания задачи и работы демонстрационной программы позволил сделать следующие выводы, которые необходимо учесть при решении задачи организации вычислительного процесса:

1. Задача распознавания графического образа представляет собой сложный вычислительный процесс;

2. Сложность вычислительного процесса обусловлена:

– размерами графических образов $\boldsymbol{Q_X}$ и $\boldsymbol{Q_Y}$;

– большой мощностью множества $\boldsymbol{Q}$;

– необходимостью идентификации по трем цветовым признакам $\boldsymbol{R}$, $\boldsymbol{G}$ и $\boldsymbol{B}$;

– большим объемом вычислений статистических оценок по выражениям (2.2) – (2.6).

3. Для эффективного решения задачи распознавания графического образа в условиях реального времени необходимо распараллеливание процесса обработки данных.

4. В качестве основных аспектов распараллеливания процесса рекомендуется использовать параллельную обработку данных: по цветовым признакам модели ***RGB***,

по зонам поиска, число которых вычисляется по формуле (2.1), по вычислению выражений (2.2) – (2.6).

Одним из методов организации параллельных процессов рекомендован метод сетевого планирования [6], предусматривающий выполнение следующих основных операций:

– определение всего перечня вычислительных операций, выполняемых при решении задачи;

– определение перечня операций, выполняемых параллельно и последовательно;

– построение сетевого графика (СГ);

– расчет элементов СГ и его оптимизация.

На рисунке 2.4 представлен вариант СГ некоторого вычислительного процесса.

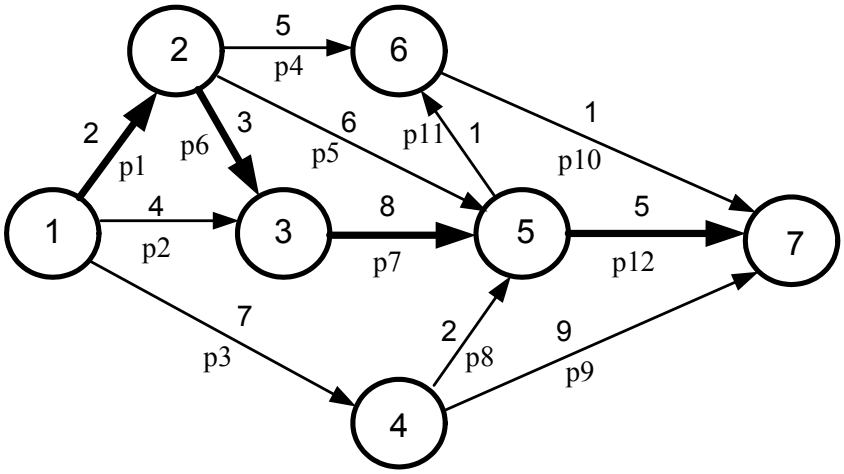


Рисунок 2.4 – Пример сетевого графика вычислительного процесса

Вершинами СГ являются события, соответствующие выполнению операций по обработке данных. Ребра СГ характеризуют направления вычислительного процесса, также время (на рисунке 2.4 показано над стрелками) и число процессоров, отведенных для каждой операции (***p*1** – ***p*12**).

События характеризуются началом и завершением вычислительной операции. Любая последовательность операций в СГ называется путем ***L***. Продолжительность

(длина) пути $T(L)$ равна сумме продолжительности составляющих его операций. Путь, имеющий наибольшую продолжительность $T_{кр}$, называется критическим.

В таблице 2.1 приведены характеристики путей СГ, представленного на рисунке 2.4.

Таблица 2.1 – Характеристики маршрутов сетевого графика

L_n	$T(L_n)$	№ п/п	L_n	$T(L_n)$
1 – 2 – 6 – 7	8	6	1 – 3 – 5 – 7	17
1 – 2 – 5 – 7	13	7	**1 – 2 – 3 – 5 – 7**	18
1 – 2 – 5 – 6 – 7	10	8	1 – 4 – 7	16
1 – 2 – 3 – 5 – 6 – 7	15	9	1 – 4 – 5 – 6 – 7	11
1 – 3 – 5 – 6 – 7	14	10	1 – 4 – 5 – 7	14

Ранний и поздний сроки свершения конкретного i-го события $t_p(i)$ и $t_n(i)$ определяются по максимальному из путей, проходящих через это событие, причем $t_p(i)$ равно продолжительности максимального из предшествующих событию путей; $t_n(i)$ равно разности между $T_{кр}$ и продолжительностью максимального из последующих за событием путей. Резерв времени для некоторого события характеризует запас времени, на который можно сдвинуть сроки свершения других событий. Он равен разности между $t_n(i)$ и $t_p(i)$. События критического пути резерва времени не имеют.

Ранний и поздний сроки свершения конкретного i-го события $t_p(i)$ и $t_n(i)$ определяются по максимальному из путей, проходящих через это событие, причем $t_p(i)$ равно продолжительности максимального из предшествующих событию путей; $t_n(i)$ равно разности между $T_{кр}$ и продолжительностью максимального из последующих за событием путей. Резерв времени для некоторого события характеризует запас времени, на который можно сдвинуть сроки свершения других событий. Он равен разности между $t_n(i)$ и $t_p(i)$. События критического пути резерва времени не имеют.

На рисунке 2.5 представлена временная диаграмма событий сетевого графика по рисунку 2.4 с распределением процессоров по операциям.

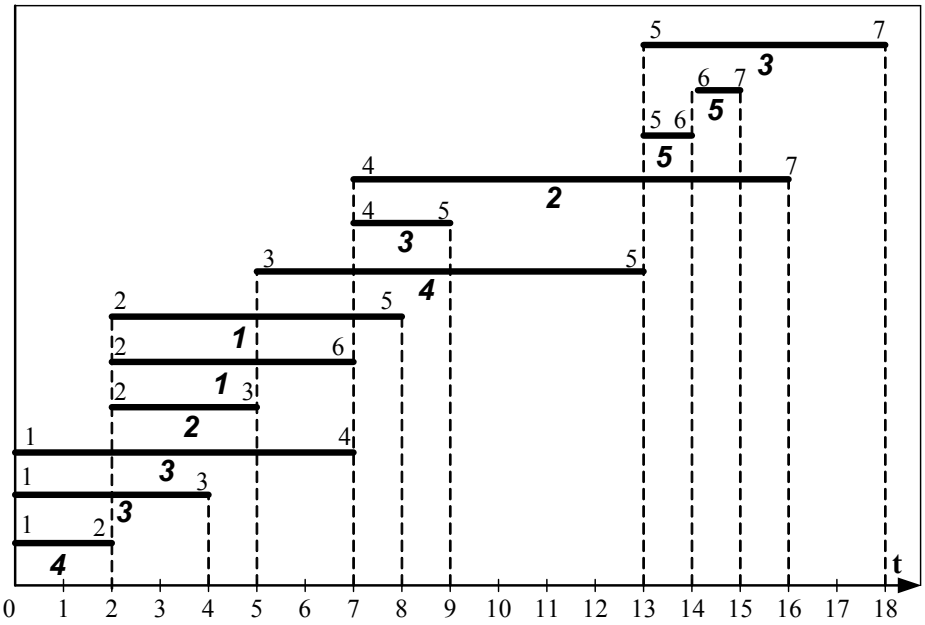


Рисунок 2.5 – Временная диаграмма событий сетевого графика с распределением процессов по операциям

Общее число процессоров, выделенных для реализации вычислительного процесса равно 10. Цифра над горизонтальными линиями соответствуют участкам СГ. Цифры под горизонтальными линиями соответствуют количеству процессоров, выделенных для выполнения конкретной операции. Общее число процессоров, находящихся в работе в каждый момент времени, не должно превышать 10.

2.3 Задание на лабораторную работу

В таблице 2.2 представлены варианты заданий на лабораторную работу.

Необходимо:

– ознакомится с работой демонстрационной программы распознавания ГО, предложенной преподавателем;

– построить СГ вычислительного процесса распознавания местоположения фрагмента ГО;

– построить временную диаграмму событий сетевого графика с распределением процессоров по операциям.

В качестве распараллеливания процесса в каждом задании рекомендуется использовать параллельную обработку данных по двум из трех способов.

Способ 1: по цветовым признакам модели ***RGB***.

Способ 2: по зонам поиска, число которых вычисляется по формуле (2.1).

Способ 3: по вычислению выражений (2.2) – (2.6).

2.4 Варианты заданий

Варианты заданий для выполнения лабораторной работы представлены в таблице 2.2.

Таблица 2.2 – Варианты заданий для выполнения лабораторной работы №3

Вариант	Способ распараллеливания	Общее число процессоров	Вариант	Способ распараллеливания	Общее число процессоров
1	1 и 2	7	11	1 и 2	10
2	1 и 3	6	12	1 и 3	6
3	2 и 3	8	13	2 и 3	8
4	1 и 2	10	14	1 и 2	5
5	1 и 3	6	15	1 и 3	6
6	2 и 3	8	16	2 и 3	8
7	1 и 2	8	17	1 и 2	10
8	1 и 3	6	18	1 и 3	6
9	2 и 3	5	19	2 и 3	5
10	1 и 2	10	20	1 и 2	7

3.5 Контрольные вопросы

1. Опишите модель распознавания образа и дайте определение коэффициенту парной корреляции и укажите диапазон его изменения.
2. Определите основные способы распараллеливания вычислительного процесса и дайте определение основным элементам сетевого графика.
3. Дайте определение критического пути.
4. Объясните алгоритм оптимизации сетевого графика.

3 Лабораторная работа № 3. Распознавание контуров геометрических объектов на графических изображениях

Цель работы: получить навыки использования методов контурного анализа при распознавания контуров геометрических объектов

3.1 Теоретическая часть

В общем случае контур – замкнутая линия, очерчивающая некоторый предмет на плоскости. Поскольку изображение дискретно, т.е. состоит из пикселей, то контур будет состоять из точек, ограничивающих какой-либо участок изображения. При этом сами точки можно получать двумя вариантами: а) точки описываемого объекта; б) точки фона. Принципиальной разницы между этими вариантами нет, и в любом случае особенностью этих точек является то, что их соседними точками являются как точки объекта, так и точки фона. Поэтому перед использованием контурного анализа необходимо выделить эти граничные точки.

Самым простым методом выделения граничных точек является бинаризация изображения (см. предыдущую работу). В этом случае происходит четкое разбиение на фон и объекты, например, черный цвет – фон, а белый – объекты. Хотя можно и наоборот. Однако при сложных изображениях перевод в монохромное может привести к потере значительной части информации, по которой можно восстановить контур.

Другим методом является применение известных операторов вычисления контуров, таких как, Робертса, Собела, Марра, Лапласа и Кенни. Данные операторы позволяют оценить перепады яркости на изображении, которые скорее всего и будут указывать на контур.

Для проверки работы некоторых операторов можно использовать следующую простую программу (Листинг 3.1), которая оперирует изображением.

Листинг 3.1. Пример работы с операторами Собела, Лапласа и Кенни в OpenCV

```
#include "opencv2/highgui/highgui_c.h"
#include "opencv2/core/core_c.h"
#include "opencv2/imgproc/imgproc_c.h"

int main()
{
	IplImage *img = cvLoadImage("lena.jpg", 1); // Загрузка изображения
	IplImage *img_gray = cvCreateImage(cvGetSize(img), 8, 1 ); // Создание 8-битного
изображения
	IplImage *img_gray2 = cvCreateImage(cvGetSize(img), 8, 1 );
	cvCvtColor(img, img_gray,CV_BGR2GRAY); // Перевод из полноцветного в градации
серого
	cvSobel(img_gray,img_gray2,1,0);
	cvSaveImage("SobelX.jpg", img_gray2);
	cvSobel(img_gray,img_gray2,0,1);
	cvSaveImage("SobelY.jpg", img_gray2);
	cvCanny(img_gray,img_gray2,50,100,3);
	cvSaveImage("Canny.jpg", img_gray2);
	IplImage* img16 = cvCreateImage(cvGetSize(img),IPL_DEPTH_16S,1 );
	cvLaplace(img_gray, img16, 3);
	cvConvertScale(img16, img_gray2);
	cvSaveImage("Laplace.jpg", img_gray2);
	return 0;
}
```

Результаты работы программы представлены на рисунке 3.1 (цвета для улучшения наглядности обращены).

После выделения границ каким либо способом – будь то описанные выше операторы или после преобразования в монохромное изображение – необходимо выделить контуры, в которые будут входить эти границы.

а – Собел (по X);

б – Собел (по Y);

в – Лаплас;

г – Кенни

Рисунок 3.1 – Операторы выделения границ

Для нахождения контуров обычно используют метод «жука», однако гораздо проще использовать уже готовые функции, которые есть в существующих бесплатных библиотеках, например в OpenCV. Ниже приведено описание функции ***cvFindContours*()** из данной библиотеки:

```
int cvFindContours(CvArr* image, CvMemStorage* storage,
CvSeq** first_contour,int header_size=sizeof(CvContour),
int mode=CV_RETR_LIST,
int method=CV_CHAIN_APPROX_SIMPLE,
CvPoint offset=cvPoint(0,0));
```

Параметр «***image***». Исходное 8-битное изображение. Отличные от нуля пиксели обрабатываются как 1, нулевые пиксели остаются 0 – т.е. изображение является монохромным. Чтобы получить такое изображение, можно использовать ***cvThreshold***, ***cvAdaptiveThreshold*** или ***cvCanny***. Функция изменяет исходное содержание изображения.

Параметр «***storage***». Контейнер найденных контуров.

Параметр «***first_contour***». Указатель на первый найденный контур.

Параметр «***header_size***». Размер заголовка последовательности, > = sizeof (CvChain) если method=CV_CHAIN_CODE, и > sizeof (CvContour) в противном случае.

Параметр «***mode***». CV_RETR_EXTERNAL – находятся только критические внешние контуры; CV_RETR_LIST – находятся все контуры, и помещает их в список; CV_RETR_CCOMP – находятся все контуры, и записывают их в иерархию с двумя уровнями: верхний уровень – внешние границы компонентов, второй уровень – границы отверстий; CV_RETR_TREE – находятся все контуры, и записывается полная иерархия вложенных контуров.

Параметр «***method***». Метод аппроксимации (для всех режимов, кроме CV_RETR_RUNS, который использует встроенную аппроксимацию). CV_CHAIN_CODE – на выходе очерчивает контур в цепном коде Фримена. Все другие методы выводят многоугольники; CV_CHAIN_APPROX_NONE – переводит все точки с цепного кода в точки; CV_CHAIN_APPROX_SIMPLE – сжимает горизонтальные, вертикальные, и диагональные доли; CV_CHAIN_APPROX_TC89_L1, CV_CHAIN_APPROX_TC89_KCOS – применяет одну из разновидностей алгоритма аппроксимации цепочки Teh-Chin. CV_LINK_RUNS – использует полностью различный алгоритм поиска контура через соединение горизонтальных долей. Только CV_RETR_LIST режим поиска может использоваться с этим методом.

Параметр «***offset***». Смещение, с которым каждая точка контура сдвинута. Это полезно, если контуры извлечены из изображения ROI, и затем они должны быть проанализированы в целом контексте изображения.

Функция ***cvFindContours*()** отыскивает контуры от монохромного изображения и возвращает число найденных контуров. Указатель ***first_contour*** заполняется функцией. Он будет содержать указатель на первый наиболее внешний контур или пустой указатель, если никакие контуры не обнаружены (если изображение полностью черно). Другие контуры могут быть достигнуты от ***first_contour***, используя ***h_next*** и связей ***v_next***. Последовательность контуров описана интересной структурой ***CvSeq***, которая предоставляет интерфейс для всех динамических структур в OpenCV.

После того, как контуры обнаружены – их можно вывести в изображение с помощью функции ***cvDrawContours*()**. Пример нахождения контуров и вывод их на изображение представлен в листинге 3.2.

Листинг 3.2. Нахождение контуров с использованием функций ***OpenCV***

```
IplImage* image = cvLoadImage( "image.jpg", 1 );
//Создание изображения в градациях серого
IplImage* img_gray= cvCreateImage( cvSize(image->width,image->height), 8, 1);
CvSeq* contours = 0;
CvMemStorage* storage = cvCreateMemStorage(0);
cvCvtColor( image, img_gray, CV_BGR2GRAY );
cvThreshold( img_gray, img_gray, 128, 255, CV_THRESH_BINARY );
/*cvAdaptiveThreshold(img_gray, img_gray, 255,
      CV_ADAPTIVE_THRESH_GAUSSIAN_C ,CV_THRESH_BINARY, 21, 7);*/
cvFindContours( img_gray, storage, &contours, sizeof(CvContour),
      CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, cvPoint(0,0) );
cvDrawContours( image, contours, CV_RGB(255,0,0), CV_RGB(0,255,0),2, 1, CV_AA,
cvPoint(0,0) );
cvSaveImage("out.jpg",image );
cvReleaseImage( &image );
```

Результат выделения контуров при использовании функции ***cvThreshold*()** представлен на рисунке 3.2 (а), а при использовании адаптивного построения монохромного изображения ***cvAdaptiveThreshold*()**, представлен на рисунке 3.2 (б).

Часто необходимо сгладить контуры. Для сглаживания контуров в OpenCV есть функция ***cvApproxPoly*()**.

Результат сглаживания показан на рисунке 3.3.

а – порогового преобразования; б – адаптивного преобразования

Рисунок 3.2 – Выделение контуров после приведения изображения к монохромному с помощью

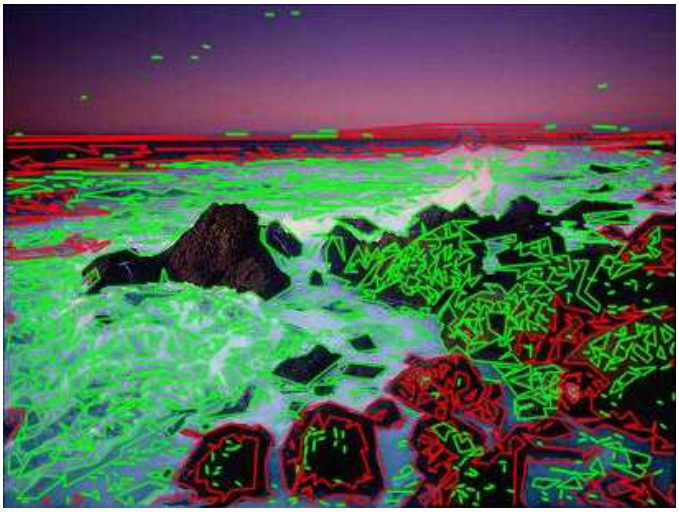


Рисунок 3.3 – Сглаженные контуры

Для анализа контуров используются так называемые моменты (полностью реализованные в OpenCV). Момент – это характеристика контура, объединённая (суммированная) со всеми пикселями контура. Момент (***p***, ***q***) определяется как:

$$m_{pq} = \sum_{i=1}^{n} I(x,y) x^p y^q \,, \tag{3.3}$$

где ***p*** – порядок ***x***;

q – порядок ***y***, где порядок означает мощность, на которой соответствующий компонент взят в сумме с другими отображенными;

***I*(*x*, *y*)** – пиксель контура (изображения);

n – количество точек контура.

Существуют центральные моменты, которые вычисляются по следующей формуле:

$$\mu_{pq} = \sum_{i=1}^{n} I(x,y)(x - x_{cp})^{\mathrm{p}} (y - y_{cp})^{\mathrm{q}} \,, \tag{3.4}$$

где x_{cp} и y_{cp} определяются из моментов, полученных в выражении (4.3), соответственно как m_{10}/m_{00} и m_{01}/m_{00}.

Центральные моменты можно нормализовать с помощью функции ***cvGetNormalizedCentralMoment*()**.

Нормализованные моменты вычисляются следующим образом:

$$\eta_{pq} = \frac{\mu_{pq}}{m_{00}^{(p+q)/2+1}} \,. \tag{3.5}$$

Комбинируя различные нормализованные центральные моменты, можно создавать инвариантные функции, представляющие различные аспекты изображения, которые не зависят от масштабов, вращения и отражения:

$$h_1=\eta_{20}+\eta_{02},\ h_2=(\eta_{20}-\eta_{02})^2+4\eta_{11}^2,\ h_3=(\eta_{30}-3\eta_{12})^2+(3\eta_{21}-\eta_{03})^2,$$

$$h_4=(\eta_{30}+\eta_{12})^2+(\eta_{21}+\eta_{03})^2,$$

$$h_5=(\eta_{30}-3\eta_{12})(\eta_{30}+\eta_{12})[(\eta_{30}+\eta_{12})^2-3(\eta_{21}+\eta_{03})^2]+(3\eta_{21}-\eta_{03})(\eta_{21}+\eta_{03})[3(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2], \quad (3.6)$$

$$h_6=(\eta_{20}-\eta_{02})[(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2]+4\eta_{11}(\eta_{30}+\eta_{12})(\eta_{21}+\eta_{03}),$$

$$h_7=(3\eta_{21}-\eta_{03})(\eta_{21}+\eta_{03})[3(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2]-(\eta_{30}-3\eta_{12})(\eta_{21}+\eta_{03})[3(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2].$$

Для получения моментов от контура или изображения в OpenCV есть ряд функций: ***cvMoments*()**, ***cvGetHuMoments*()**. Для сравнения – функция ***cvMatchShapes*()**.

Поддерживаются три метода сравнения инвариантных моментов.

CV_CONTOURS_MATCH_I1:

$$I_1(A,B)=\sum_{i=1}^{7}\left|\frac{1}{m_i^A}-\frac{1}{m_i^B}\right|, \quad (3.7)$$

где ***A*** и ***B*** – первое и второе изображение (контур).

CV_CONTOURS_MATCH_I2:

$$I_2(A,B)=\sum_{i=1}^{7}\left|m_i^A-m_i^B\right|. \quad (3.8)$$

CV_CONTOURS_MATCH_I3:

$$I_3(A,B)=\sum_{i=1}^{7}\left|\frac{m_i^A-m_i^B}{m_i^A}\right|. \quad (3.9)$$

Параметры ***m*** для выражений (3.7) – (3.9) высчитываются по следующим формулам:

$$m_i^A=sign(h_i^A)\log\left|h_i^A\right|, \quad (3.10)$$

$$m_i^B=sign(h_i^B)\log\left|h_i^B\right|. \quad (3.11)$$

Чем меньше возвращаемое значение выражений (3.7) – (3.9), тем более похожи изображения (контуры). Одним из направлений использования моментов и их сравнений является распознавания текста.

3.2 Порядок выполнения работы

1. Используя любую среду программирования (например, Visual Studio версии 2005 или старше и язык C/C++) и создайте новый проект.
2. Создайте файлы с эталонными образами в редакторе Paint в соответствии с вариантом. Размеры файлов 200 на 200 пикселей.
3. Напишите процедуру загрузки изображений в оперативную память.
4. Используя библиотеку OpenCV, найдите контуры всех эталонных образов и выведите информацию об этом или на экран или в графический файл.
5. Используя функции OpenCV ***cvMoments*()**, ***cvGetHuMoments*()**, определите моменты для каждого из образов в соответствии с постановкой задачи. Выведите эти значения на экран или в текстовый файл.
6. Модифицируете эталонные образы в соответствии с постановкой задачи и получите значения моментов.
7. Сравните значения эталонных и измененных образов. Определите, какие моменты можно использовать для отличия одного контура от другого.
8. Сделайте выводы о проделанной работе и составьте отчет.
9. Ответьте на контрольные вопросы.

3.3 Задание на лабораторную работу

1. Выбрать образы, которые необходимо распознать в соответствии с вариантом задания (таблица 3.1). Коды классов образов представлены на рисунке 3.4.

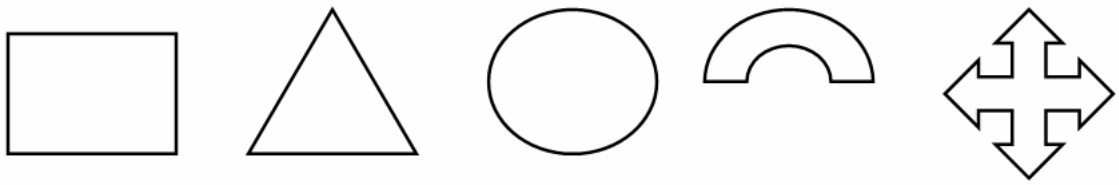


Рисунок 3.4 – Образы для распознавания

2. Написать программу, которая анализирует образы в соответствии с вариантом, выделяет внешние контуры, рассчитывает обычные, центральные и инвариантные моменты для них.

3. Протестируйте программу: а) на исходных образах; б) на образах, повернутых на угол из таблицы 3.1; в) на образах с искусственно произведенным разрывом контура; г) на образах с добавленной пересекающей произвольной линией. Сравните полученные значения.

4. Сделайте выводы о том, какие моменты можно использовать в случаях, когда нет поворота образа, в случаях поворота. Определите, как влияет повреждение и зашумление образа на его распознавание.

3.4 Варианты заданий

Варианты заданий для выполнения лабораторной работы представлены в таблице 3.1.

Таблица 3.1 – Варианты заданий для выполнения лабораторной работы №4

Вариант	Коды образов	Угол наклона, град.	Задание	Коды образов	Угол наклона, град.
1	1, 2, 3	10	9	1, 3, 4	70
2	1, 2, 4	20	10	1, 3, 5	80
3	1, 2, 5	30	11	1, 4, 5	10
4	1, 3, 4	45	12	2, 3, 4	20
5	1, 3, 5	60	13	2, 3, 5	30
6	1, 4, 5	70	14	3, 4, 5	45
7	2, 3, 4	80	15	1, 2, 3	60
8	2, 3, 5	10	16	1, 2, 4	10

3.5 Контрольные вопросы

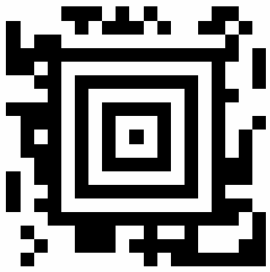
1. Что такое контур?
2. Какие методы выделения границ в изображении вы знаете?
3. Какие методы аппроксимации контуров используются на практике?
4. Что такое момент контура?
5. Какие виды моментов существуют?

4 Лабораторная работа № 4. Технологии распознавания двумерных штрих-кодов

Цель работы: получить знания о двумерном штриховом кодировании и технологиях распознавания

4.1 Теоретическая часть

На настоящий момент существует большое количество двумерных штриховых кодов (бар-кодов), наиболее популярными среди которых являются Aztec Code, DataMatrix, PDF417 и QR Code. Примеры кодов, содержащих одну и ту же фразу «Test code» представлены на рисунке 4.1.



а – Aztec Code;

б – DataMatrix;

в – PDF417;

г – QR Code

Рисунок 4.1 – Примеры кодирования фразы «Test code» в формате

У каждого из этих кодов есть свои достоинства и недостатки, что позволяет использовать их в различных условиях. Центральная мишень Aztec Code позволяет распознавать его в условиях пересечения других изображений с краями кода. DataMatrix – хорошо зарекомендовавший себя код, но требующий свободное место по краям. PDF417 сделан по подобию одномерных штрих кодов для облегчения сканирования, но может кодировать меньше информации по сравнению с другими

двухмерными бар-кодами. Популярный в Японии QR Code является одним из наиболее оптимальных кодов, хотя при малых размерах кода квадраты-мишени отнимают много места. Некоторые из кодов защищены стандартами и патентами, а некоторые переданы в общественное достояние.

В отличие от распознавания одномерных кодов, где необходимо прочитать и декодировать штриховую линию [6], для двухмерных кодов необходимо чётко определить не только границы, но и некоторые синхронизирующие элементы. Т.к. в большинстве случаев двухмерные бар-коды представляют собой квадраты или прямоугольники, то необходимо выделить четыре угловые точки. Большинство алгоритмов включают в себя следующие этапы:

- предварительная обработка изображения;
- детектирование углов кода;
- обратное перспективное преобразование;
- получение матрицы кода и декодирование.

На рисунке 4.2 слева показано правильное расположение двухмерного бар-кода при кодировании, а справа – пример того, как может находиться код на входном изображении. Обратное перспективное преобразование – это перевод изображения к тому, что было раньше.

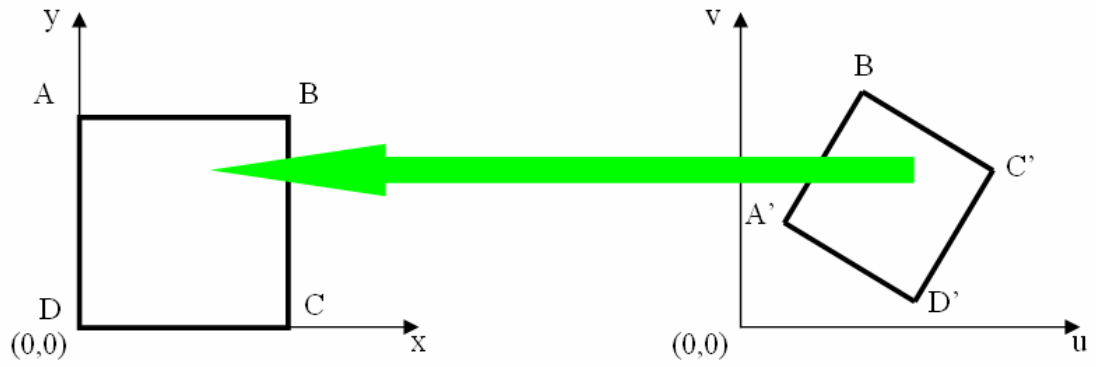


Рисунок 4.2 – Схема обратного перспективного преобразования

После того, как точки получены, можно приступать к наложению матрицы распознавания. Так при восстановлении матрицы кода «DataMatrix» нужно построить на полученных линиях сетку, и определить значения ячеек. Однако, поскольку размеры кода могут быть большими, то из-за наклона камеры возможно изменение

размера ячеек. Поэтому по размерам ячеек граничных пунктирных линий определяется зависимость их изменений с использованием метода наименьших квадратов. После этого строится сетка с различными значениями размеров ячеек, представленных на рисунке 4.3.

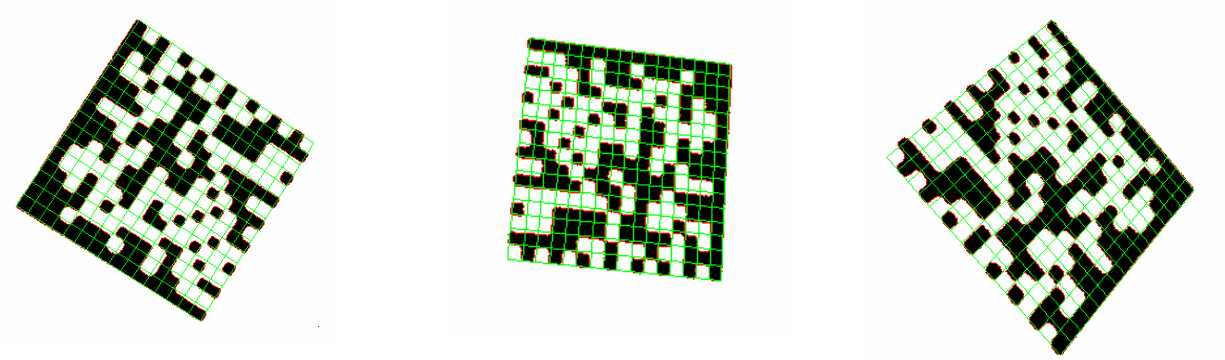


Рисунок 4.3 – Полученная сетка матрицы для различных положений кода DataMatrix

Полученная матрица кодируется определенным образом, причем в нее добавляются корректировочные коды Рида-Соломона. Т.е. часть информации несет в себе смысл, а другая часть – позволяет восстановить эту информацию при повреждении матрицы. Для восстановления данных необходимо знать правила кодирования конкретного кода. Ниже приведена информация кодирования кода «Aztec». Важнейшим элементом при детектировании кода является центральная мишень с битами ориентации, а также синхронизирующие пунктирные линии, представленные на рисунке 4.4.

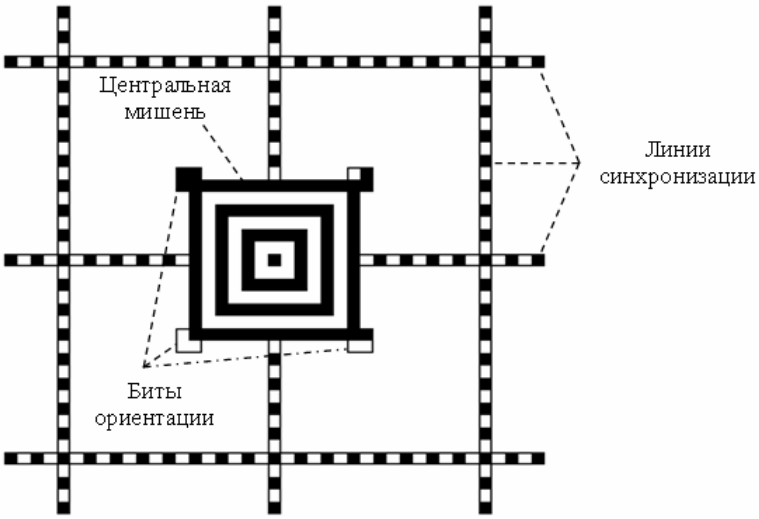


Рисунок 4.4 – Элементы для нахождения, ориентации и синхронизации Aztec кода

Наличие центральной мишени позволяет легко детектировать код, выяснить ориентацию с помощью элементов по краям и синхронизировать распознавание кода по соответствующим линиям.

Кодируемые данные располагаются по слоям (уровням), которые обвиваются вокруг центральной мишени по так называемому принципу «домино» по часовой стрелке (см. рисунок 4.5).

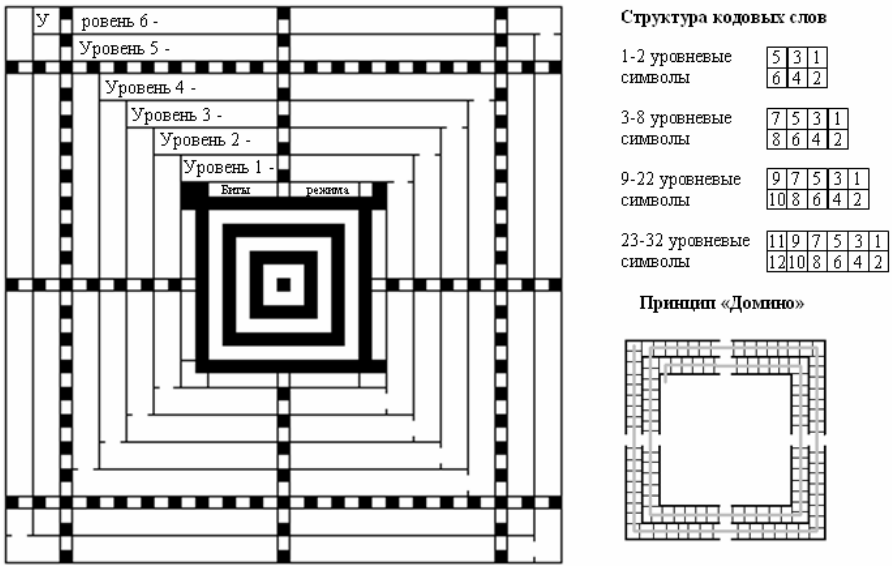


Рисунок 4.5 – Расположение данных, структура кодовых слов и принцип «домино»

Информация о режиме кодирования находится между ориентировочными элементами мишени по принципу «домино» и содержит в себе 40 бит. Уровни шириной в два бита и с различной длиной кодовых слов располагаются по принципу «домино», исключая синхронизирующие элементы – пунктирные линии сетки, которых при увеличении количества уровней становится больше. Центральные пунктирные линии присутствуют в любом стандартном коде Aztec. Остальные линии добавляются между 4 и 5 слоями, в слоях 12 и 27, разделяя кодовые слова.

В начальном уровне **#1** содержится 128 бит информации, а в последующих добавляется по 32 бита, поэтому общее количество бит информации в символе длины ***L*** может быть рассчитано по формуле:

$$Cb = (112 + 16 * L) * L \,. \qquad (4.1)$$

Коррекция Рида-Соломона осуществляется для кодовых слов различной длины:

– в 1-2 уровневых символах используются 6-битные кодовые слова;

– в 3-8 уровневых символах используются 8-битные кодовые слова;

– в 9-22 уровневых символах используются 10-битные кодовые слова;

– в 23-32 уровневых символах используются 12-битные кодовые слова.

Поэтому количество кодовых слов, о которых говорится в блоке данных, рассчитывается следующим образом:

$$Cw = Cb \ \ div \ \ K \ , \tag{4.2}$$

где ***K*** – количество бит в кодовом слове;

div – функция целого деления.

Возможные размеры символов Aztec Code представлены в таблице 4.1.

Таблица 4.1 – Размеры и вместимость символов «Aztec Code»

Уровни	Размер	Вме-стимост	Уровни	Размер	Вмести-мость	Уровни	Размер	Вмести-мость
1	19 × 19	21ь× 6	12	67 × 67	364 × 10	23	113 × 113	920 × 12
2	23 × 23	48 × 6	13	71 × 71	416 × 10	24	117 × 117	992 × 12
3	27 × 27	60 × 8	14	75 × 75	470 × 10	25	121 × 121	1066 × 12
4	31 × 31	88 × 8	15	79 × 79	528 × 10	26	125 × 125	1144 × 12
5	37 × 37	120 × 8	16	83 × 83	588 × 10	27	131 × 131	1224 × 12
6	41 × 41	156 × 8	17	87 × 87	652 × 10	28	135 × 135	1306 × 12
7	45 × 45	196 × 8	18	91 × 91	720 × 10	29	139 × 139	1392 × 12
8	49 × 49	240 × 8	19	95 × 95	790 × 10	30	143 × 143	1480 × 12
9	53 × 53	230 × 10	20	101 × 101	894 × 10	31	147 × 147	1570 × 12
10	57 × 57	272 × 10	21	105 × 105	940 × 10	32	151 × 151	1664 × 12
11	61 × 61	316 × 10	22	109 × 109	1020 × 10			

В таблице показаны физические размеры символов и их вместимость (количество кодовых слов умноженных на количество бит). Информация о режиме кодирования, как отмечалось выше, содержит 40 бит, из которых только 16 бит содержат полезную информацию, остальные 24 бита – выполняют функцию коррекции ошибок Рида-Соломона. Первые 5 бит содержат размер символа, который определяется через количество слоёв минус 1. Т.е. если первые 5 бит – нулевые, то код содержит только 1 уровень. Следующие 11 бит кодируют длину сообщения, которая определяется как количество кодовых слов минус 1, включая символы следующие за кодовыми словами для контроля ошибок.

Эти 16 служебных бит расположены в 4-битных словах, а остальные 6 проверочных слов добавляются кодированием Рида-Соломона с полем Галуа GF(16), основанным на главном полиноме модуля $\boldsymbol{x}^4+\boldsymbol{x}+\mathbf{1}$ (коэффициент генерации полинома = 19). Порождающий полином ($\boldsymbol{x}$-21)..($\boldsymbol{x}$-26) следующий:

$$x^6 + 7x^5 + 9x^4 + 3x^3 + 12x^2 + 10x + 12\,. \qquad (4.3)$$

Все десять слов следуют друг за другом, начиная с верхнего левого угла. Информационное сообщение (закодированное в уровнях символа) кодируется по-разному в зависимости от размеров.

В таблице 4.2 показаны характеристики распределения Рида-Соломона, необходимые для кодирования-декодирования информации.

Таблица 4.2 – Размеры кодовых слов и главные полиномы модуля

Уровни	Размер кодового слова	Поле Галуа	Полином	Коэффициент генерации
1 – 2	6 бит	GF(64)	$\boldsymbol{x}^6 + \boldsymbol{x} + 1$	67
3 – 8	8 бит	GF(256)	$\boldsymbol{x}^6 + \boldsymbol{x}^5 + \boldsymbol{x}^3 + \boldsymbol{x}^2 + 1$	301
9 – 22	10 бит	GF(1024)	$\boldsymbol{x}^{10} + \boldsymbol{x}^3 + 1$	1033
23 – 32	12 бит	GF(4096)	$\boldsymbol{x}^{12} + \boldsymbol{x}^6 + \boldsymbol{x}^5 + \boldsymbol{x}^3 + 1$	4201

Высокоуровневое кодирование состоит из двух этапов. На первом этапе, строка символов преобразуются в поток бит. На втором этапе поток бит подвергается преобразованию, которое предназначено для того, чтобы исключить длительного повторения одинаковых бит.

На этапе 1 берутся символы сообщения, которые кодируются с использованием таблицы 4.3.

Размеры символов могут быть 4, 5 и 8 бит – соответственно для цифр (Digit), остальных режимов таблицы 1.3 и побайтного кодирования.

Сообщение начинает кодироваться в режиме Upper и может быть переключено в другой режим с помощью UL (Upper), LL (Lower), ML (Mixed), PL (Punct), DL (Digit) или переведено в режим только для ввода единственного символа с помощью US (Upper) и PS (Punct). При переключении на режим Digit происходит смена количества бит на символ с 5 на 4.

Таблица 4.3 – Высокоуровневое кодирование «Aztec Code»

	Upper		Lower		Mixed		Punct		Digit	
Value	Char	ASCII	Char	ASCII	Char	ASCII	Char	ASCII	Char	ASCII
0	PS		PS		PS		FLG(n)	***	PS	
1	SP	32	SP	32	SP	32	CR	13	SP	32
2	A	65	a	97	SOH	1	CR LF	13,10	0	48
3	B	66	b	98	STX	2	.SP	46,32	1	49
4	C	67	c	99	ETX	3	,SP	44,32	2	50
5	D	68	d	100	EOT	4	:SP	58,32	3	51
6	E	69	e	101	ENQ	5	!	33	4	52
7	F	70	f	102	AVK	6	"	34	5	53
8	G	71	g	103	BEL	7	#	35	6	54
9	H	72	h	104	BS	8	$	36	7	55
10	I	73	i	105	HT	9	%	37	8	56
11	J	74	j	106	LF	10	&	38	9	57
12	K	75	k	107	VT	11	'	39	,	44
13	L	76	l	108	FF	12	(	40	.	46
14	M	77	m	109	CR	13	)	41	UL	
15	N	78	n	110	ESC	27	*	42	US	
16	O	79	o	111	FS	28	+	43		
17	P	80	p	112	GS	29	,	44		
18	Q	81	q	113	RS	30	-	45		
19	R	82	r	114	uS	31	.	46		
20	S	83	s	115	@	64	/	47		
21	T	84	t	116	\	92	:	58		
22	U	85	u	117	^	94	;	59		
23	V	86	v	118	_	95	<	60		
24	W	87	w	119	`	96	=	61		
25	X	88	x	120	|	124	>	62		
26	Y	89	y	121	~	126	?	63		
27	Z	90	z	122	DEL	127	[	91		
28	LL		US		LL		]	93		
29	ML		ML		UL		{	123		
30	DL		DL		PL		}	125		
31	BS		BS		BS		UL			

BS (Binary Shift) – переключение на ввод 8-битной строки символов с заданным размером. После BS следует 5-битное значение: если неравно нуля, то содержит в себе количество последующих байт строки; если 0, то следующие 11 бит содержат в себе количество байт минус 31. После окончания строки, кодирование возвращается в режим, который был до BS.

Символ, обозначенный FLG(n) в таблице 4.3, является специальным флагом, представляющим различные неинформационные символы, предоставляемые многими стандартными символиками. В битовом потоке значение FLG(n)

сопровождается 3 дополнительными битами, кодирующими аргумент «n» в бинарном режиме, поэтому значение ***n*** может быть от 0 до 7.

FLG(0) представляет «FNC1» - флажок данных происходящий от Code 128. Когда FNC1 используется в первой позиции данных, это сигнализирует об использовании правил EAN/UPC формата данных, использующих Application Identifiers, и установке бита 0 в модификаторе идентификаторе символики. Когда FNC1 используется во второй позиции данных или в третьей позиции следуют 2 цифры, то это сигнализирует об использовании некоторого другого индустриально-специфического формата, идентифицированного предыдущими данными и устанавливающим бит 1 в модификаторе. Когда FNC1 используется в дальнейших локациях, он служит в качестве разделителя области и вставляет ASCII код 29 в это место в выходных данных.

FLG(1) – FLG(6) предназначены для представления расширенного канального переходного символа ECE, который в выходной строке данных представлен как «\nnnnnn», наклонная черта влево, сопровождаемая 6 цифрами. Присутствие ECE где-нибудь в символе принуждает все наклонные черты влево внутри кодируемых данных быть удвоенным в выходной строке и также устанавливает бит 2 в модификаторе идентификаторе символики. Аргумент «n» показывает, сколько из этих 6 цифр явно кодируются, используя режим Digit, в символе, заполненном нулями. ECE #000123, например, кодируется FLG(3), после чего кодирование возвращается в режим предшествующий FLG(n). FLG(7) – не используется.

На втором шаге кодировании сообщения результирующий поток бит делится на последовательность из B-битных (B = 6, 8, 10 или 12) кодовых слов. Анализируя последовательность, первые B-1 бит проверяются на наличие одинаковых бит информации. Если все B-1 бит равны 0, тогда добавляется бит пустышки (dummy), который равен 1, сразу после первых B-1 бит кодового слова, после чего следует последний бит и дальнейшие элементы последовательности. Если все B-1 бит равны 1, тогда добавляется бит пустышки (dummy), который равен 0. Это делается для того, чтобы «разбавить» возможные одинаковые биты в коде, чтобы не принесло пользы для чтения кода.

В конце, символ и границы байта в первоначальном сообщении не имеют никаких необходимых отношений с границами ключевого слова. Поэтому добавляются дополнительные пустышки с 1 (или 0 в случае необходимости), чтобы устранить любую двусмысленность.

Возьмём пример кодирования фразы «Test code».

На рисунке 4.6. показано, как происходит перевод в биты информации и получения последовательности.

Перевод в биты

ASCII символ	Значение из табл.	Биты
T	21	10101
LL	28	11100
e	6	00110
s	20	10100
t	21	10101
SP	1	00001
c	4	00100
o	16	10000
d	5	00101
e	6	00110

Последовательность

10101+11100+00110+10100+10101+00001+00100+10000+00101+00110

Рисунок 4.6 – Перевод в биты информации и получение последовательности

Если используется одноуровневый «Aztec Code», то необходимо разбить последовательность по 6 бит. Затем проверить, необходимо ли добавлять dummy. Если нужно, то добавить (см. рисунок 4.7).

Перевод в блоки кодовых слов

Блоки бит	Кодовое слово
101011	43
110000	48
110101	53
001010	10
100001	33
001001	9
000000	0
101001	41
10	

Добавление dummy

Блоки бит	Кодовое слово
101011	43
110000	48
110101	53
001010	10
100001	33
001001	9
000001	1
010100	20
110111	55

Рисунок 4.7 – Перевод в кодовые слова и добавление dummy

После этого добавляются кодовые слова контроля ошибок, и генерируется код (см. рисунок 4.8).

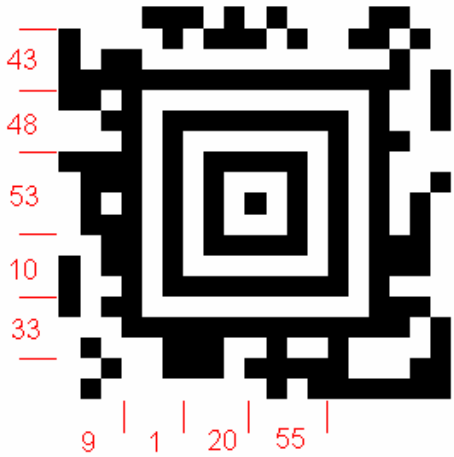


Рисунок 4.8 – 1-уровневый Aztec Code кодирующий фразу «Test Code»

Как видно из рисунка 4.8, значительную часть слоя занимают корректирующие кодовые слова Рида-Соломона.

4.2 Порядок выполнения работы

1. Выберете Aztec код в соответствии с вариантом из прилагаемых к работе данных.
2. Используя теорию кодирования кода Aztec вручную (без использования сторонних программ) определите закодированную фразу.
3. Запустите программу VidikonReader (прилагаемую к работе) и разберитесь с принципами работы.
4. Посмотрите результат распознавания вашего примера Aztec кода.
5. Создайте копию вашего примера и в редакторе Paint случайным образом с помощью инструмента «Карандаш» зашумите данные кода. Посмотрите результаты распознавания.
6. Создайте копию вашего примера и в редакторе Paint случайным образом с помощью инструмента «Карандаш» зашумите область мишени. Посмотрите результаты распознавания.

7. Создайте копию вашего примера и в редакторе Paint случайным образом с помощью инструмента «Линия» черного и белого цвета зашумите всю мишень. Посмотрите результаты распознавания.
8. Повторите шаги 5 – 7 с различными параметрами.
9. Сделайте выводы о проделанной работе и составьте отчет.
10. Ответьте на контрольные вопросы.

4.3 Задание на лабораторную работу

1. Выбрать Aztec код в соответствии с вариантом по имени Вариант[номер варинта].bmp.
2. Расшифровать кодовые слова так, как показано на рисунках 9.7 и 9.8, указав кодовые слова без корректирующих слов Рида-Соломона.
3. Используя программу «VidikonReader.exe», проверьте как распознается код с различными видами зашумлений: только данных кода пиксельным зашумлением, мишени кода пиксельным зашумлением, всего кода с помощью шумовых линий.

4.4 Контрольные вопросы

1. Какие виды двумерных кодов вы знаете?
2. Каков общий алгоритм распознавания двумерного бар-кода?
3. Зачем необходима коррекция Рида-Соломона?
4. Какова разрядность кодовых слов в Aztec коде?
5. Какие виды высокоуровневого кодирования Aztec вы знаете?

5 Лабораторная работа № 5. Распознавание текста в графическом изображении

Цель работы: получить навыки использования шаблонного метода распознавания текста в графическом изображении

5.1 Теоретическая часть

Алгоритмы распознавания текста (OCR), как и многие другие алгоритмы работают с бинаризированным изображением. Бинаризация изображений – это перевод RGB или в градациях серого изображения в монохромное, где присутствуют только два типа пикселей (темные и светлые). Существуют различные подходы к бинаризации, которые условно можно разделить на 2 группы: пороговые, адаптивные.

Если говорить кратко, то пороговые методы бинаризации работают со всем изображением, находя какую-то характеристику (порог), позволяющую разделить все изображение на чёрное и белое. Адаптивные методы работают с участками изображений и используются при неоднородном освещении объектов.

Для того, чтобы использовать пороговый метод бинаризации, необходимо или самостоятельно задать порог, например, значение 128 при анализе изображения в градациях серого, или воспользоваться какими-либо критериями автоматического определения оптимального порогового значения. Наиболее известные методы – Отсу, Бернсена, Эйквеля, Ниблэка и т.п. Наиболее эффективным по качеству и скорости обработки считается метод Отсу.

Суть данных методов такова: пусть все пиксели изображения распределяются по гистограмме яркости так, как показано на рисунке 5.1, тогда пороговым значением яркости будет то, которое делит оптимальным образом пиксели на черные и белые.

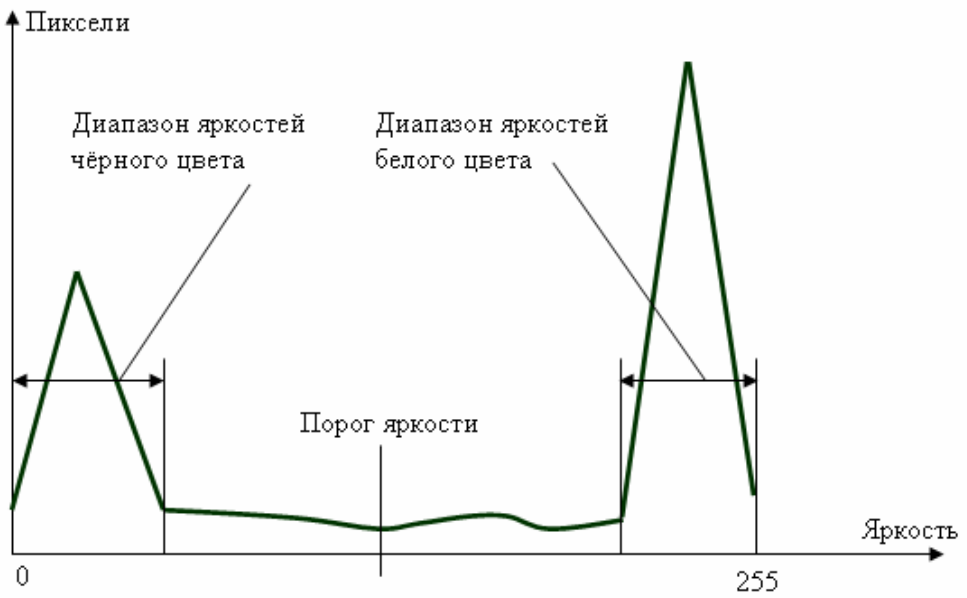


Рисунок 5.1 – Пример гистограммы яркости для изображения

Метод Отсу использует гистограмму распределений значения яркости. Диапазон яркости делится на ***L*** элементов, в котором ищется два класса значений (т.е. ищутся две области на гистограмме с распределением пикселей вокруг них). Каждому классу соответствуют относительные частоты:

$$\omega_0(k)=\sum_{i=1}^{k} p_i,\ \omega_1(k)=\sum_{i=k+1}^{L} p_i,\ \omega_1(k)=1-\omega_0(k), p_i=\frac{n_i}{N}, \tag{5.1}$$

где ***N*** – общее количество пикселей на изображении;

n_i – количество пикселей с уровнем яркости ***i***.

Средние уровни для каждых из двух классов изображения:

$$\mu_0(k)=\sum_{i=1}^{k}\frac{p_i i}{\omega_0(k)}, \mu_1(k)=\sum_{i=k+1}^{L}\frac{p_i i}{\omega_1(k)}. \tag{5.2}$$

Для каждого ***k*** из диапазона 1 ÷ ***L*** можно вычислить относительные частоты и средние уровни, после чего посчитать оценку качества разделения (качества порогового уровня) для каждого значения и найти максимум:

$$\eta(k)=\max_{k=1}^{L}\left(\frac{\sigma_{кл}^2(k)}{\sigma_{общ}^2}\right),\ \sigma_{кл}^2(k)=\omega_0(k)\omega_1(k)(\mu_1(k)-\mu_2(k))^2, \tag{5.3}$$

где $\boldsymbol{\sigma_{общ}^2}$ – общая дисперсия всего изображения.

Однако в случаях, когда изображение освещено неравномерно, описанный выше алгоритм применить достаточно сложно. Для этого необходимо использовать

адаптивное преобразование в монохромное изображение. Задаются блоки на изображении с одинаковыми шириной и высотой, значения которых являются нечётными, т.к. они являются окрестностями центрального пикселя.

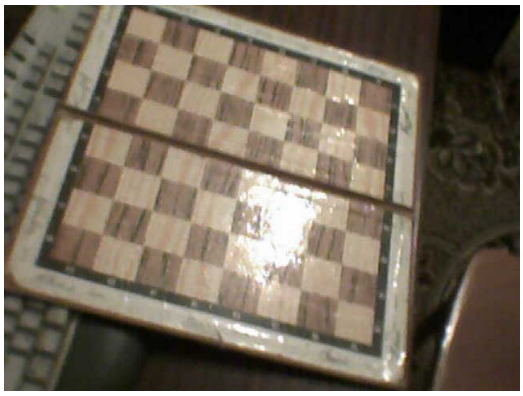
Суть метода такова: для каждого пикселя определяются окрестности, в которых вычисляется среднее значение яркости, которое и является пороговым значением ***T*(*x*,*y*)** для текущего пикселя. Для 24-битного изображения преобразование в монохромное изображение может осуществляться по следующей формуле:

$$dst(x,y) = \begin{cases} RGB(255,255,255), & \textit{если Яркость}(src(x,y)) > T(x,y) \\ RGB(0,0,0), & \textit{в другом случае} \end{cases} \quad (5.4)$$

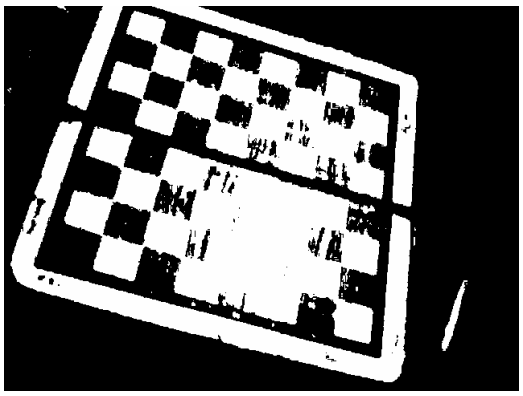
где ***src*(*x*, *y*)** – RGB-значения пикселя с координатами ***x*** и ***y*** на исходном изображении;

***dst*(*x*, *y*)** – соответственно конечного изображения.

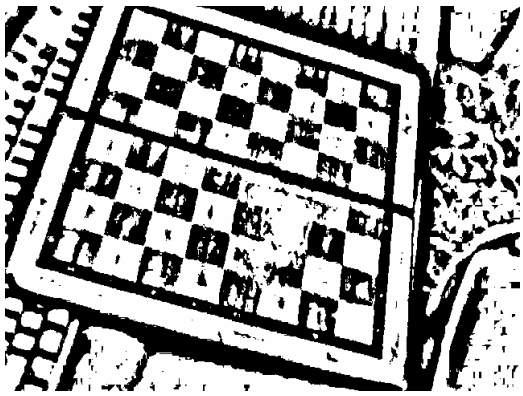
Пример отличия работы этих двух подходов на рисунке 5.2.



а – исходное изображение;



б – бинарное;



в – адаптивное преобразование

Рисунок 5.2 – Два подхода к преобразованию в монохромное изображение

И тот и другой подходы имеет смысл применять в различных ситуациях с учётом необходимых требований. В библиотеке OpenCV данные преобразования осуществляются с помощью функций ***cvThreshold*** и ***cvAdaptiveThreshold***. Помимо этих методов существуют еще множество комбинированных методов.

Предположим на некотором изображении есть текст, представленный на рисунке 5.3 (а).

а – исходное; б – монохромное

Рисунок 5.3 – Изображение с текстом

Для того чтобы распознать символы, необходимо определить их местоположение. Однако в данной лабораторной работе задача будет упрощена, и на вход будет поступать изображение с одним символом. Первоначально изображение надо привести к монохромному изображению, используя адаптивный или пороговый подход, представленный на рисунке 5.3 (б).

Определить границу символа для распознавания достаточно просто после проведенной бинаризации. Для этого можно воспользоваться выделением контуров (функция ***cvFindContours()*** в OpenCV – описано в следующей лабораторной работе) либо заранее заданным местом поиска символа.

Конечно, известны методы распознавание текста на базе нейронных сетей, но существует ещё более простой метод (в плане обучения), основанный на сравнении с эталоном (шаблоном). В ряде случаев сравнение с шаблоном будет достаточно для распознавания текста. Суть шаблонного метода заключается в том, чтобы взять изображение символа и сравнить их со всеми эталонами, каждый из которых разделен на небольшие блоки.

На рисунке 5.5 приведено несколько примеров различного построения шаблонов для буквы «С».

-1	1	1	1	-1
1	-1	-1	-1	1
1	-1	-2	-1	-1
1	-1	-2	-2	-2
1	-1	-2	-1	-1
1	-1	-1	-1	1
-1	1	1	1	-1

0.1	0.5	0.6	0.6	0.5
0.6	0.5	0.0	0.0	0.9
1.0	0.0	0.0	0.0	0.0
1.0	0.0	0.0	0.0	0.0
1.0	0.0	0.0	0.0	0.0
0.6	0.5	0.0	0.0	0.8
0.2	0.5	0.6	0.6	0.7

а – обыкновенный шаблон; б – шаблон по методу зон

Рисунок 5.5 – Два варианта шаблона для буквы «С»

В обыкновенном шаблоне проверяется, заполнена ячейка или нет (естественно округляя, если полупустая) и умножается на коэффициент. В идеальном случае совпадение с буквой «С» будет возвращать значение 13 (сумма всех единиц), а, например, с буквой «В» сумма будет складываться так: -1+1+1+1+1-1-2-2-1+1+1+1+1+1-1+1+1+1+1-1-1=3. По наибольшему совпадению будет определяться тот образ, который и соответствует неизвестному. Также необходимо предусмотреть пороговое значение, ниже которого образ считается нераспознанным.

На рисунке 5.5 (б) представлен другой вариант, где на каждую ячейку шаблона выделяется не полностью вариант – совпало / не совпало – а выделяется вес этой ячейки, который будет сопоставляться с совпадением пикселей неизвестного образа. Естественно в обоих случаях размеры шаблонов можно увеличивать для увеличения точности распознавания.

Для того, чтобы сравнить область изображения с символом необходимо привести эту область к размеру шаблона или наоборот – шаблон к размеру области. Можно просто использовать известные методы увеличения, или написать самому.

В листинге 5.1 показан С-код, в котором осуществляется приведение к размеру шаблона (в данном случае шаблон 5 на 7).

Листинг 5.1. Приведение размеров неизвестного образа к шаблону

```
double a[5*7];
double a1[4];
uchar* ptr = (uchar*) (Image->imageData);
int x,y;
```

```
int x1,y1;
double x1_,y1_;
int x2,y2;
double x2_,y2_;
..../* Точками обозначен участок, где необходимо получить зону (Rect), где находится символ */
memset(a,0,5*7*sizeof(double));
for(y=0;y<Rect.height;y++)
	for(x=0;x<Rect.width;x++){
		if (ptr[x+Rect.x+(y+Rect.y)* Image->widthStep]==0) continue;
		memset(a1,0,4*sizeof(double));
		x1_=(double)x*ratioW;
		y1_=(double)y*ratioH;
		x1=int(x1_);y1=int(y1_);
		x2_=(double)(x+1)*ratioW;
		y2_=(double)(y+1)*ratioH;
		x2=int(x2_);y2=int(y2_);
		if (x2==x1) { a1[0]=1;a1[1]=0;}
		else {
			a1[0]=(double)(x1+1-x1_)/(x2_-x1_);
			a1[1]=(double)(x2_-x2)/(x2_-x1_);
		}
		if (y2!=y1){
			a1[2]=(double)a1[0]*(y2_-y2)/(y2_-y1_);
			a1[3]=(double)a1[1]*(y2_-y2)/(y2_-y1_);
			a1[0]=(double)a1[0]*(y1+1-y1_)/(y2_-y1_);
			a1[1]=(double)a1[1]*(y1+1-y1_)/(y2_-y1_);
		}
		a[x1+y1*7]+=a1[0];
		if (x1<4)
			a[x1+1+y1*7]+=a1[1];
		if (y1<6)
			a[x1+(y1+1)*7]+=a1[2];
		if (x1<4 && y1<6)
			a[x1+1+(y1+1)*7]+=a1[3];
	}
```

В листинге 5.1 перебираются каждые точки, относящиеся к символу, после чего происходит их отнесение к той или иной ячейки изображения, причем не всегда полностью, но и частично – часть пикселя в одну ячейку, часть в другую. Далее можно сравнить полученный результат с каждым из эталонов, просто перемножив соответствующие элементы матрицы. И по наибольшему совпадению выбрать эталон.

На рисунке 5.6 приведено несколько примеров подобных искажений на базе автомобильных номеров.

а – наклон текста вглубь; б – шум на изображении; в – грязь; г – поворот символов

Рисунок 5.6 – Искажения текста и результаты бинаризации

В идеальном случае достоверность распознавания шаблонным методом будет высокой, однако, в реальности на изображения символов оказывают влияние следующие факторы: поворот в плоскости, наклон вглубь, загрязнение и зашумление символов.

5.2 Порядок выполнения работы

1. Используя любую среду программирования (например, Visual Studio версии 2005 или старше и язык C/C++) и создайте новый проект.
2. С помощью встроенного в операционную систему Windows редактора Paint создайте 6 эталонных образов типа и шрифта в соответствии с вариантом. Фон белый, текст черный. Эталонный образ (буква) должен быть полностью вписан в изображение, причем с максимально возможным размером. Размер изображения может быть произвольным, например 40 на 30 пикселей по ***X*** и ***Y*** соответственно.
3. Напишите процедуру загрузки изображений в оперативную память.
4. Напишите процедуру обучения алгоритма распознавания для типа и размера шаблона в соответствии с вариантом. При обучении вся входная картинка условно разбивается на клетки и информация о заполнении клетки (черный цвет) записывается в качестве обучающей в массив данных.
5. Напишите процедуру распознавания, которая будет: а) приводить размеры входного изображения к эталонному (функция ***cvResize*** в OpenCV); б) получать информацию о заполнении клеток распознаваемого изображения; в) сравнивать информацию о заполнении клеток с эталонами и выдавать результат о величине совпадения с каждым эталоном.

6. Протестировать работу программы в соответствии с постановкой задачи.
7. Сделайте выводы о проделанной работе и составьте отчет.
8. Ответьте на контрольные вопросы.

5.3 Задание на лабораторную работу

1. Написать программу, распознающую символы по методу шаблонов. В качестве эталонных символов использовать символы из соответствующего варианта (таблица 5.1), шаблонный метод по варианту. При обучении на вход программы должны поступать BMP-файлы, в каждом из которых находится по одному символу. При тестировании на вход программы поступает BMP-файл, а программа выдает результат – распознанный символ.

2. Тестирование должно проводиться по следующим выборкам (наборам BMP файлов):

а) эталонная;

б) эталонная, но с изменением масштабов;

в) те же символы, но набранные другим шрифтом;

г) наклоненные вглубь символы (например, с помощью Paint: рисунок → растянуть → наклон → по горизонтали);

д) случайным образом зашумленные (некоторое количество разноцветных точек на изображении).

3. Проанализировать полученные результаты, построив таблицы результатов распознавания для каждого варианта тестирования.

5.4 Варианты заданий

Варианты заданий для выполнения лабораторной работы представлены в таблице 5.1.

Таблица 5.1 – Варианты заданий для выполнения лабораторной работы №5

Вариант	Символы	Тип шаблона (рисунок 5.5)	Шрифт	Размер шаблона ($X \times Y$)
1	Q, W, E, R, T, Y	А	Arial	5 × 7
2	U, I, O, P, A, S	Б	Times	10 × 10
3	D, F, G, H, J, K	П	Verdana	10 × 14
4	L, Z, X, C, V, B	Б	Cambria	6 × 10
5	N, M, 1, 2, 3, 4	А	Courier	5 × 10
6	5, 6, 7, 8, 9, 0	Б	Tahoma	12 × 12
7	Ц, У, К, Е, Н, Г	А	Impact	5 × 7
8	Ш, Щ, З, Х, Ъ, Ф	Б	Arial	10 × 10
9	Ы, В, А, П, Р, О	А	Times	10 × 14
10	Л, Д, Ж, Э, Я, В	Б	Verdana	6 × 10
11	С, М, И, Т, Ь, Б	А	Cambria	5 × 10
12	1, 2, 3, 4, 5, 6	Б	Courier	12 × 12
13	A, B, C, D, E, F	А	Tahoma	5 × 7
14	I, J, K, L, M, N	Б	Impact	10 × 10
15	O, P, R, S, T, Q	А	Arial	10 × 14
16	X, Y, Z, 0, 1, 2	Б	Times	6 × 10
17	А, Е, И, О, У, Ю	А	Verdana	5 × 10
18	А, Б, В, Г, Д, Е, Ж	Б	Cambria	12 × 12
19	З, И, К, Л, М, Н	А	Courier	5 × 7
20	О, П, Р, С, Т, У	Б	Tahoma	10 × 10

5.5 Контрольные вопросы

1. Что такое бинаризация?
2. Чем отличается пороговая бинаризация от адаптивной бинаризации?
3. В чем суть шаблонного метода распознавания символов?
4. Какие методы оптимального определения порогового значения бинаризации вы знаете?
5. Какие помехи накладываются на реальные символы?

6 Лабораторная работа № 6. Ситуационный принцип в задаче распознавания образов

Цель работы: освоить принципы построения системы ситуационного распознавания образов

6.1 Теоретическая часть

Подход «ситуация управляет системой» закономерно возник в экономической отрасли, как инструмент оперативного контроля и управления экономическими субъектами в условиях быстроменяющихся финансовых ситуаций. Возможность применения данного подхода не ограничивается только социально-экономическими системами, например, ситуационное управление используется в поисковой системе Google, в менеджменте персонала, при управлении сложными техническими, эргатическими и организационными системами. Сложность задачи ситуационного управления распознаванием образов заключается в сравнительной оценке большого количества признаков (от нескольких десятков – до нескольких сотен) для выбора наиболее вероятных под лидирующие классы образов [7].

Модель системы распознавания образов имеет следующий вид: «значения признаков → ассоциативные оценки мер близости → коды сигнатур образов → коды ситуаций → оптимальные признаки».

Для формализации задачи использованы следующие условные обозначения [1, 7]: $\boldsymbol{q^x}$ – неизвестный образ, подлежащий распознаванию; $\boldsymbol{Q^*}$ – класс образов, к которому отнесен $\boldsymbol{q^x}$; $\boldsymbol{<s_i>}$ – зарегистрированное значение $\boldsymbol{i}$-го признака, $\boldsymbol{i = 1, m}$; $\boldsymbol{<S^x>}$ – вектор зарегистрированных значений признаков $\boldsymbol{q^x}$; $\boldsymbol{V\{q^x, Q_j\}}$ – мера близости между $\boldsymbol{q^x}$ и $\boldsymbol{j}$-ым образом из множества эталонов $\boldsymbol{Q}$; $\boldsymbol{v_{ij}\{<s_i>, Q_j\}}$ – частный параметр оценки ассоциативности значения $\boldsymbol{<s_i>}$ признака $\boldsymbol{s_i}$ из множества $\boldsymbol{S}$ для всех классов образов из $\boldsymbol{Q}$, $\boldsymbol{j = 1, n}$; $\boldsymbol{\Phi\{<S^x>, Q_j\}}$ – функционал для вычисления меры близости $\boldsymbol{V\{q^x, Q_j\}}$; $\boldsymbol{P(s)}$ – множество функций оценок распределений значений признаков из $\boldsymbol{S}$ для всех классов образов $\boldsymbol{Q}$; $\boldsymbol{w_{ij}\{v_{ij}, Q_j\}}$ – коэффициент сигнатуры значения $\boldsymbol{v_{ij}}$ для $\boldsymbol{j}$-го

образа из множества эталонов $\boldsymbol{Q}$; $\boldsymbol{W\{q^x, Q_j\}}$ – совокупность коэффициентов сигнатур, характеризующих ситуацию распознавания $\boldsymbol{j}$-го образа из множества эталонов $\boldsymbol{Q}$; $\boldsymbol{\Phi\{q^x, s_i\}}$ – функционал для нахождения оптимальных (наиболее вероятных) признаков $\boldsymbol{s_i}$ из множества $\boldsymbol{S}$ для образа $\boldsymbol{q^x}$, исходя из ситуации распознавания; $\boldsymbol{R_j}$ – сумма попаданий $\boldsymbol{<s_i>}$ для $\boldsymbol{i}$-го признака $\boldsymbol{j}$-го класса образов.

В общем случае в теории и практике ситуационного управления распознаванием образов в качестве правил отбора образов из множества $\boldsymbol{Q}$ могут быть использованы:

– доминирующий стиль (выбор оптимальных признаков только для одного лидера, набравшего максимальное количество баллов);

– тандемный стиль (выбор оптимальных признаков для двух лидеров);

– командный стиль (лояльный выбор оптимальных признаков сразу для нескольких классов образов, лидирующих в "эстафете" распознавания $\boldsymbol{Q_j}$).

По способу выбора признаков в алгоритме существуют следующие:

– рулеточный (кольцевой перебор признаков $\boldsymbol{s_i}$ из множества $\boldsymbol{S}$);

– вероятностный (выбор оптимальных признаков с учетом их рейтингов, исходя из вероятностей появления лидирующих образов).

Математическая модель распознавания $\boldsymbol{q^x}$ имеет следующий вид [1, 7]:

$$V\{q^{x}, Q_j\} = \Phi\{< S^x >, P, Q_j\}, j = 1, n; \tag{6.1}$$

$$\Phi\{< S^x >, P, Q_j\} = \sum_{i=1}^{m} v_{ij}\{< s_i{}^x >, P(s_i), Q_j\}, i = 1, m; \tag{6.2}$$

$$v_{ij}\{< s_i{}^x >, Q_j\} = \begin{cases} 1, & \text{если} \quad < s_i{}^x > \in D_{ij}; \\ 0, & \text{если} \quad < s_i{}^x > \notin Q_j; \end{cases} \tag{6.3}$$

$$v_{ij}\{< s_i{}^x >, P(s_i), Q_j\} = \begin{cases} 1, & \text{если} \quad P(s_i{}^x) = \max P(s_i); \\ 0, & \text{если} \quad P(s_i{}^x) \neq \max P(s_i); \end{cases} \tag{6.4}$$

$$M\{< S^x >, Q\} = \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1j} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2j} & \dots & v_{2n} \\ \dots & \dots & \dots & v_{ij} & \dots & \dots \\ v_{m1} & v_{m2} & \dots & v_{mj} & \dots & v_{mn} \end{pmatrix}; \tag{6.5}$$

$$q^x \in Q^* \in Q : V\{q^x, Q^*\} \equiv \max V\{q^x, Q_j\}, Q_j \in Q; \tag{6.6}$$

$$W\{q^x, Q_j\} = \Phi\{q^x, s_i\}, j = 1, n; \tag{6.7}$$

$$\Phi\{q^x, s_i\} = \begin{cases} 1, & \text{если} \quad P(s_i^x) = max\ P\{s_i, Q_j\}; \\ 0, & \text{если} \quad P(s_i^x) \neq max\ P\{s_i, Q_j\}; \end{cases} \tag{6.8}$$

$$P(s_i^x) = max\ P(s_i) : R_j = \max\left(\sum_{i=1}^{m} v_{ij} \{< s_i^x >, P(s_i), Q_j\} \right). \tag{6.9}$$

В матрице (6.5) каждый столбец соответствует частным мерам близости $\boldsymbol{v_{ij}}$ множества $\boldsymbol{<S^x>}$ для каждого класса образов по всем признакам. Сумма элементов каждой строки соответствует сумме в выражении (6.2). Выражение (6.6) описывает разделяющее правило, при котором отнесение $\boldsymbol{q^x}$ к одному из классов образов-эталонов $\boldsymbol{Q^*}$ производится на основе сигнатурного подхода и по максимальной величине меры близости, соответствующей сумме элементов в каждой строке матрицы (6.5). При обучении модели производится вычисление элементов матрицы (6.5) по выражениям (6.1) – (6.9).

Процесс распознавания $\boldsymbol{q^x}$ сводится к вычислению суммы элементов матрицы по каждой строке, соответствующих $\boldsymbol{<s^x>}$ и $\boldsymbol{<S^x>}$, и выбору искомого класса образа по максимальной сумме. Принятие решения по максимальной сумме элементов строки матрицы с учетом сигнатурного подхода позволяет реализовать мажоритарный принцип и повышает достоверность распознавания образов. Достоинством представленной модели распознавания образов является высокая производительность, обусловленная простотой алгоритма вычисления разделяющей функции и реализации разделяющего правила. Аппаратная и программная реализация алгоритма на порядок проще и быстрее аналогичного алгоритма распознавания, работающего, например, по методу Байеса.

На рисунке 6.1 представлена схема алгоритма выбора оптимальных (наиболее вероятных) признаков среди множества $\boldsymbol{S}$ для лидирующих образов $\boldsymbol{Q^*}$ на основе ситуационного принципа.

Входными данными алгоритма является: множество лидирующих классов образов $\boldsymbol{Q^*}$ и множество неопрошенных признаков $\boldsymbol{S^Y}$. Выходными – матрица оптимальных признаков $\boldsymbol{S^P}$ в порядке убывания приоритетов для лидирующих классов образов $\boldsymbol{Q^*}$.

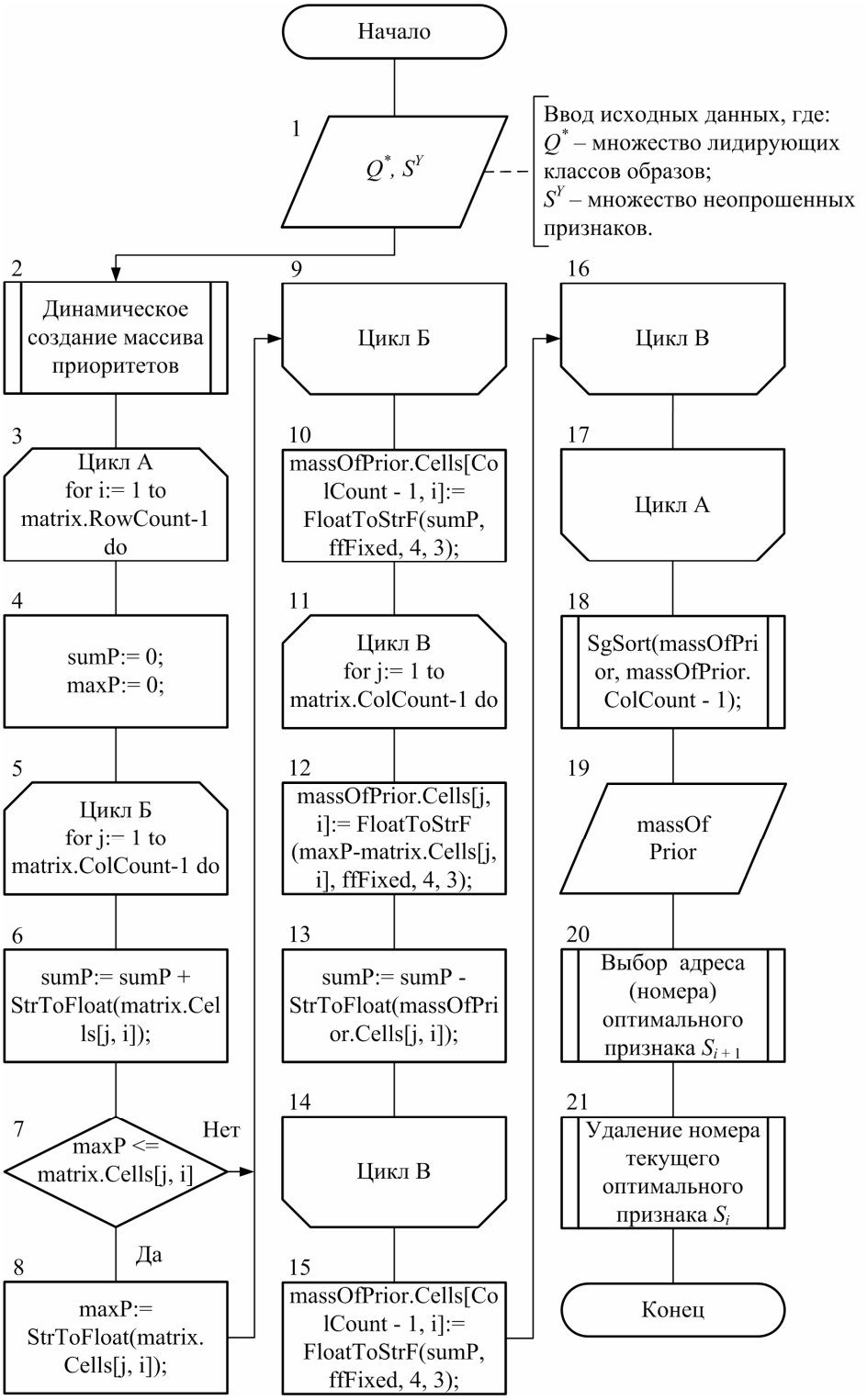


Рисунок 6.1 – Схема алгоритма выбора оптимальных признаков

В ходе работы алгоритма осуществляется формирование двумерного динамического массива, представленного в таблице 6.1, в ячейки (столбцы № 2 – 4) которого записываются вероятности $\boldsymbol{P}(\boldsymbol{Q_j}^*)$ появления ***j***-го лидирующего образа для ***i***-го неопрошенного признака $\boldsymbol{S_i^Y}$. В пятый столбец массива записывается сумма

вероятностей $\boldsymbol{P(Q^*)}$ для ***i***-го неопрошенного признака $\boldsymbol{S_i^Y}$. После заполнения матрицы осуществляется поиск максимальной вероятности $\mathbf{max}\boldsymbol{P(Q_j^*)}$ появления ***j***-го лидирующего образа для ***i***-го неопрошенного признака $\boldsymbol{S_i^Y}$.

Таблица 6.1 – Пример динамического массива, содержащего вероятности $\boldsymbol{P(Q_j^*)}$ появления ***j***-го лидирующего образа для ***i***-го признака $\boldsymbol{S_i^Y}$

Код признака	$P(Q_1^*)$	$P(Q_2^*)$	$P(Q_4^*)$	$\Sigma P(Q^*)$	$\Delta_1 = \max P(Q_j^*) - P(Q_1^*)$	$\Delta_2 = \max P(Q_j^*) - P(Q_2^*)$	$\Delta_4 = \max P(Q_j^*) - P(Q_4^*)$	Значение приоритета
1	2	3	4	5	6	7	8	9
A	5/20	6/20	4/20	15/20	1/20	0	2/20	0,6
B	7/20	4/20	1/20	12/20	0	3/20	6/20	0,15
C	3/20	1/20	4/20	8/20	1/20	3/20	0	0,20
D	8/20	0/20	3/20	11/20	0	8/20	5/20	-0,1

Темным цветом в таблице 6.1 выделены вероятности $\boldsymbol{P(Q^*)}$, которые являются максимальными $\mathbf{max}\boldsymbol{P(Q_j^*)}$.

Далее формируются вектора (см. таблицу 6.1, столбцы № 6 – 8), содержащие разницу $\boldsymbol{\Delta}$ между $\mathbf{max}\boldsymbol{P(Q_j^*)}$ и остальными $\boldsymbol{P(Q^*)}$. Полученные $\boldsymbol{\Delta}$ последовательно вычитаются из $\boldsymbol{\Sigma P(Q^*)}$ и формируют значение приоритета (см. таблицу 6.1, столбец №9). После вычисления значений приоритетов для всех неопрошенных признаков $\boldsymbol{S_i^Y}$, массив сортируется по убыванию. На выходе алгоритма (см. рисунок 6.1) содержится список наиболее вероятных кодов признаков $\boldsymbol{S_i^P}$ в порядке убывания приоритетов.

Далее алгоритм программы осуществляет опрос соответствующего признака, который получил наибольшее значение приоритета. В режиме идентификации на адресные входы ассоциативного ЗУ последовательно поступают значения признаков исследуемого образа. В подсистеме принятия решений передаются меры близости $\boldsymbol{v_{ij}}$ множества $\boldsymbol{<S^x>}$ для каждого класса образов по всем признакам. На основании этих данных в подсистеме принятия решений формируется совокупность коэффициентов сигнатур $\boldsymbol{W\{q^x, Q_j\}}$, характеризующих ситуацию распознавания ***j***-го образа из множества эталонов $\boldsymbol{Q}$. Через функционал $\boldsymbol{\Phi\{q^x, s_i\}}$ алгоритм программы определяет

оптимальные (наиболее вероятные) коды признаков s_i из множества S для лидирующих образов, исходя из ситуации распознавания.

Эффективность ситуационного алгоритма распознавания графических образов в среднем выполняется за три шага (итерации), тогда как для сигнатурного или классического ассоциативно-мажоритарного подхода требуется 5 – 6 итераций.

Таким образом, сигнатурный и ситуационный подходы позволяют оценить ситуацию по совокупности кодов сигнатур для выбора наиболее вероятных признаков под лидирующие классы образов и повысить оперативность принятия решения по распознаваемому классу образов.

6.2 Порядок выполнения работы

1. Запустить программу «Эмулятор системы ситуационного распознавания образов».

2. Используя теоретический материал, представленный в пункте 6.1, разобраться в ситуационном и сигнатурном принципах в задаче распознавания образов.

3. Изучить представленный программный эмулятор:

- режим настройки объема ассоциативного ЗУ;
- режим распознавания образов в соответствии с вариантом задания;
- режим пост обработки журнала вычислительного эксперимента.

4. В процессе работы эмулятора пользователь должен определить достоинства сигнатурного и ситуационного подходов, которые следует указать в отчете о проделанной работе.

6.3 Задание на лабораторную работу

1. Изучить прикладную программу «Эмулятор системы ситуационного распознавания образов».

2. Оценить длительность работы алгоритма распознавания образов в соответствии с вариантом задания.

3. Оформить отчет о проделанной лабораторной работе.

6.4 Варианты заданий

Варианты заданий для выполнения лабораторной работы представлены в таблице 6.2.

Таблица 6.2 – Варианты заданий для выполнения лабораторной работы №6

Вариант	Кол-во $\boldsymbol{Q}$	Кол-во $\boldsymbol{S}$	Длины $\boldsymbol{D_{ij}}$	Правило отбора $\boldsymbol{Q}$	Способ выбора $\boldsymbol{S}$
1	произвольное	1 ÷ 100	20 ÷ 58	доминирующий, тандемный, командный	вероятностный
2	100	1 ÷ 50	32 ÷ 64		рулеточный
3	1 ÷ 100	произвольное	16 ÷ 64		вероятностный
4	больше 1000	100 ÷ 1000	произвольные	доминирующий и тандемный	рулеточный
5	произвольное	100 ÷ 1000	1 ÷ 100		вероятностный
6	1 ÷ 20	произвольное	100 ÷ 1000	доминирующий и командный	рулеточный
7	100 ÷ 1000	произвольное	1 ÷ 50		вероятностный
8	произвольное	1 ÷ 50	64	тандемный и командный	вероятностный и рулеточный
9	произвольное	1 ÷ 100	128		
10	200 ÷ 500	100 ÷ 250	произвольное	доминирующий	
11	меньше 1000	произвольное	произвольное	тандемный	вероятностный
12	500 ÷ 1000	произвольное	100 ÷ 1000	командный	вероятностный
13	произвольное	1 ÷ 100	20 ÷ 58	доминирующий, тандемный, командный	вероятностный
14	100	1 ÷ 50	32 ÷ 64		рулеточный
15	1 ÷ 100	произвольное	16 ÷ 64		вероятностный
16	больше 1000	100 ÷ 1000	произвольные	доминирующий и тандемный	рулеточный
17	произвольное	100 ÷ 1000	1 ÷ 100		вероятностный
18	меньше 1000	произвольное	произвольное	тандемный	вероятностный
19	500 ÷ 1000	произвольное	100 ÷ 1000	командный	вероятностный
20	произвольное	100 ÷ 1000	1 ÷ 100	тандемный	рулеточный

6.5 Контрольные вопросы

1. Назовите преимущества ситуационного распознавания образов.
2. Назовите факторы, влияющие на достоверность распознавания образов.
3. Что такое порог распознавания и разделяющая функция?
4. Назовите критерии оценки эффективности систем распознавания образов.
5. Назовите достоинства сигнатурного подхода?

7 Лабораторная работа № 7. Ассоциативно-мажоритарная модель распознавания образов

Цель работы: освоить технологию оперативной оценки состояния протяженного объекта на основе ассоциативно-мажоритарной модели распознавания образов

7.1 Теоретическая часть

Распознавание состояния протяженных в пространстве объектов, например, автомобильных дорог является одним из основных процессов в системе управления качеством автомобильных дорог, требующих больших материальных и временных затрат [8]. Результаты распознавания состояния дорожного полотна (ДП) должны служить надежной информационной базой для решения управленческих задач.

На рисунке 7.1 представлены схема протяженного в пространстве объекта на примере автомобильной дороги.

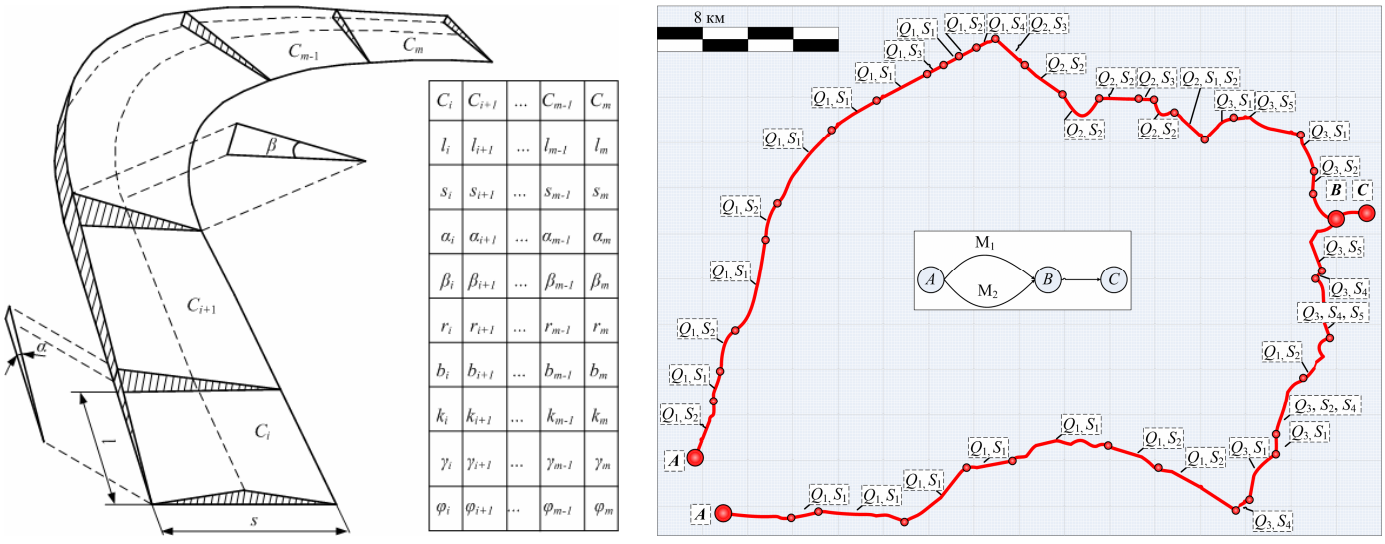


Рисунок 7.1 – Схема дорожного полотна и маршрут движения «Оренбург – Исянгулово» по трассам Р-314 и Р-336

Каждый участок C_i дороги длиной l и шириной s характеризуется следующими показателями, влияющими на безопасность дорожного движения: продольным α и

поперечным **β** уклоном, радиусом виража конкретной категории качества ДП ***r***, шероховатостью **γ** и коэффициентом продольного сцепления **φ**.

К исходным данным задачи идентификации относится [8]: множество классов образов $\boldsymbol{Q} = \{q_1, q_2, \ldots, q_j, \ldots, q_n\}$, характеризующих категорию ДП; множество информативных признаков $\boldsymbol{S} = \{s_1, s_2, \ldots, s_i, \ldots, s_m\}$ с известными распределениями значений $\boldsymbol{P(s)}$. Для формализации задачи распознавания использованы условные обозначения, представленные в лабораторной работе № 6.

Математическая модель распознавания q^x представлена в лабораторной работе №6 формулами (6.1) – (6.9).

Представленная модель универсальна и может быть использована для распознавания состояний сложных объектов при различных постановках задачи, в частности: при контроллинге и отборе персонала в социально-экономических системах; при определении категории дорожного полотна и соответствующей безопасной скорости транспорта по принятой в РФ классификации категорий автомобильных дорог согласно ГОСТ Р 52398-2005; в подсистемах мониторинга состояний региональных транспортных систем. При этом изменяется лишь выражение для вычисления частных параметров оценки ассоциативности значения $<s_i>$ признака s_i из множества $\boldsymbol{S}$.

На рисунке 7.2 представлена структурная схема высокопроизводительной системы идентификации на основе ассоциативно-мажоритарной модели распознавания образов, позволяющей оценивать состояние ДП во время движения автомобиля.

Входными данными для системы оценки являются:

– вектор параметров регистрации $\boldsymbol{R}=\{\boldsymbol{S,n,m}\}$;

– режим работы системы $\boldsymbol{U}$ (обучение или распознавание).

Потоки измерительной информации определяются перечнем датчиков и режимами работы средств регистрации и первичной обработки данных. Эффективность данной системы в основном определяется временем (производительностью) $\boldsymbol{H}$ проведения диагностирования и достоверностью $\boldsymbol{D}$ принимаемых решений.

Подсистема регистрации и первичной обработки данных о ДП формирует информационные потоки для подсистемы обучения ассоциативной модели и подсистемы идентификации ровности ДП.

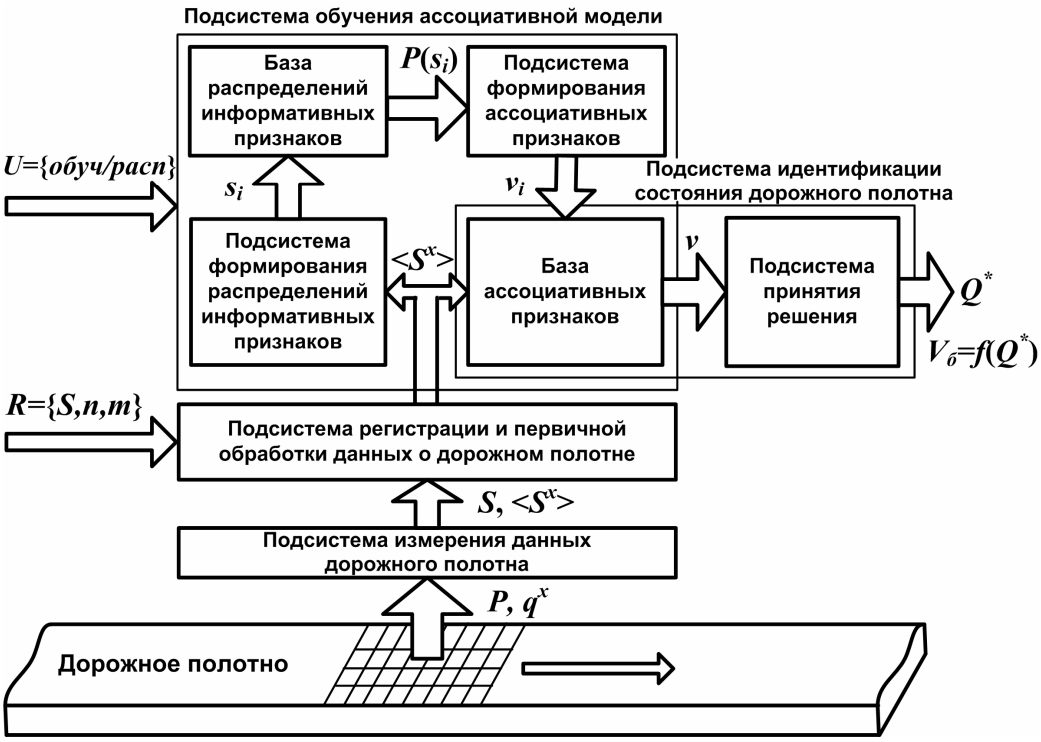


Рисунок 7.2 – Структурная схема системы идентификации состояния ДП

Формирование базы распределений информативных признаков ***P*** основано на принципе работы амплитудных анализаторов сигналов, что обеспечивает высокое быстродействие, обусловленное простотой алгоритма формирования произвольных распределений. На этапе обучения ассоциативной модели подсистема управления последовательно выбирает соответствующий признак дорожного полотна и организует регистрацию его распределения в системе. По окончании регистрации подсистема формирования ассоциативных признаков по полученным оценкам распределений формирует базу ассоциативных признаков. В результате идентификации система относит диагностируемое состояние ДП к одному из семи классов ровности ДП по международной классификации индексов ровности.

Применение предложенного подхода позволяет снизить затраты на проведение эксперимента. Ассоциативная модель распознавания образов реализована в виде

аппаратно-программного устройства и пакета программ [4]. Представленный подход может также найти применение в системах технической диагностики сложных промышленных объектов металлургического, нефтегазодобывающего профиля, а также в системах контроля и управления доступом в АСУ ТП.

7.2 Порядок выполнения работы

1. Войти в объектно-ориентированную среду программирования «Borland Delphi 7». Разработать общий вид экранной формы прикладной программы так, как показано на рисунке 7.3.

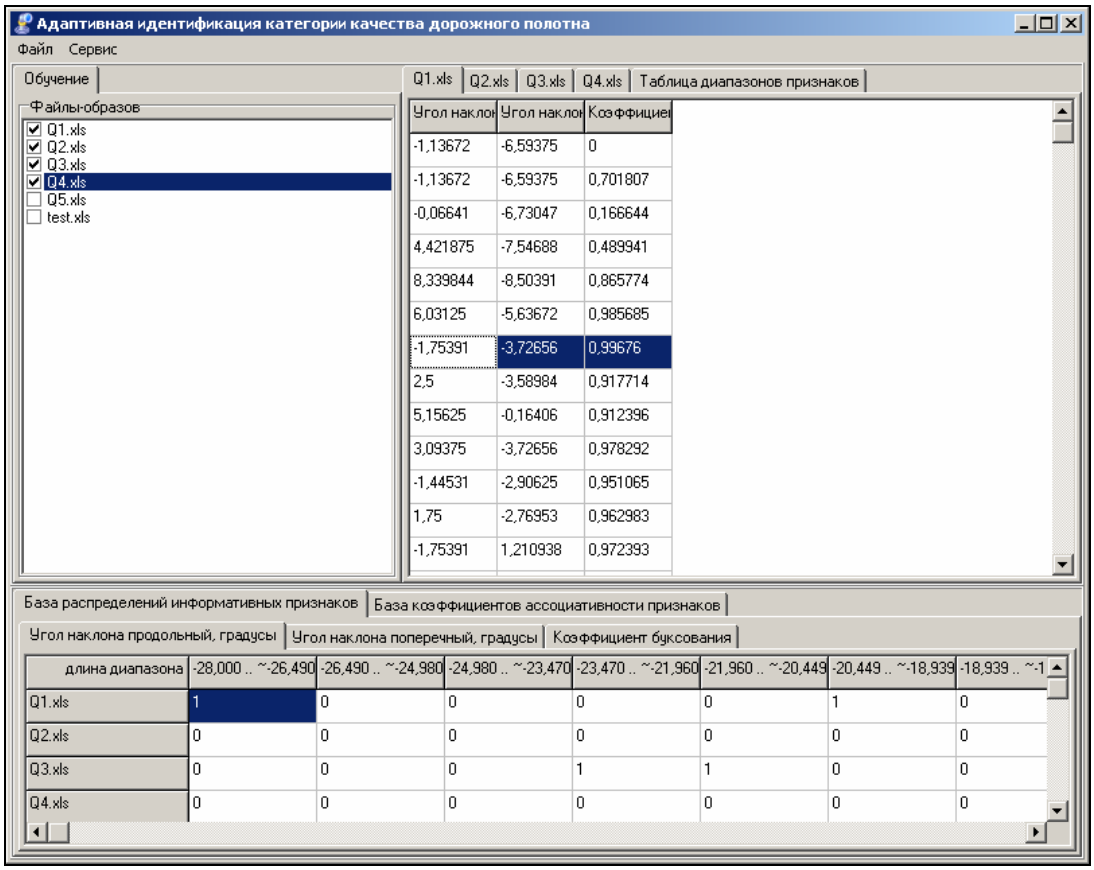


Рисунок 7.3 – Общий вид экранной формы в процессе работы программы

2. Используя теоретический материал, представленный в пункте 7.1, реализовать алгоритм оценки состояния ДП на основе ассоциативно-мажоритарной модели распознавания образов.

3. В программе должна быть предусмотрена возможность настройки

следующих параметров: количества информативных признаков и классов образов-эталонов; объема памяти (диапазона адресов), приходящегося на каждую страницу ассоциативного ЗУ; ведение журнала вычислительного эксперимента.

4. В программе должна быть предусмотрена возможность постобработки результатов вычислительных экспериментов и формирования диаграммы распределений значений информативных признаков.

Прикладная программа должна иметь гибкую структуру и возможность модификации.

7.3 Задание на лабораторную работу

1. Разработать прикладную программу «Адаптивная идентификация категории качества дорожного полотна» в соответствии с вариантом задания.
2. Оценить длительность работы алгоритма распознавания образов.
3. Разработать инструкции пользователя и программиста для работы с программой.

7.4 Варианты заданий

Варианты заданий для выполнения лабораторной работы представлены в таблице 6.1.

7.5 Контрольные вопросы

1. Объясните принцип работы амплитудного анализатора сигналов.
2. Как определяется общая ошибка при вычислении меры Хемминга?
3. Назовите способы выбора информативных признаков.
4. Назовите достоинства и недостатки ассоциативного и мажоритарного подходов.
5. Что такое вычислительная сложность алгоритма распознавания?

8 Лабораторная работа № 8. Генетический алгоритм распознавания образов

Цель работы: освоить принципы генетики в задаче распознавания образов

8.1 Теоретическая часть

Принципы генетики применительно к задаче распознавания образов позволяют определять варианты решений (совокупность параметров), которые обеспечивают заданные режимы работы технологических процессов. Достоинством генетических алгоритмов применительно к задаче выбора безопасного скоростного режима автомобиля является высокая производительность нахождения решений, обеспечивающих заданный безопасный скоростной режима автомобиля с учетом заданной величины погрешности. Сложность задачи оперативного выбора безопасного скоростного режима автомобиля заключается в сравнительной оценке большого объема значений совокупности эксплуатационных параметров системы «водитель – автомобиль – дорога – среда» (ВАДС), порядка 10^{13} [8].

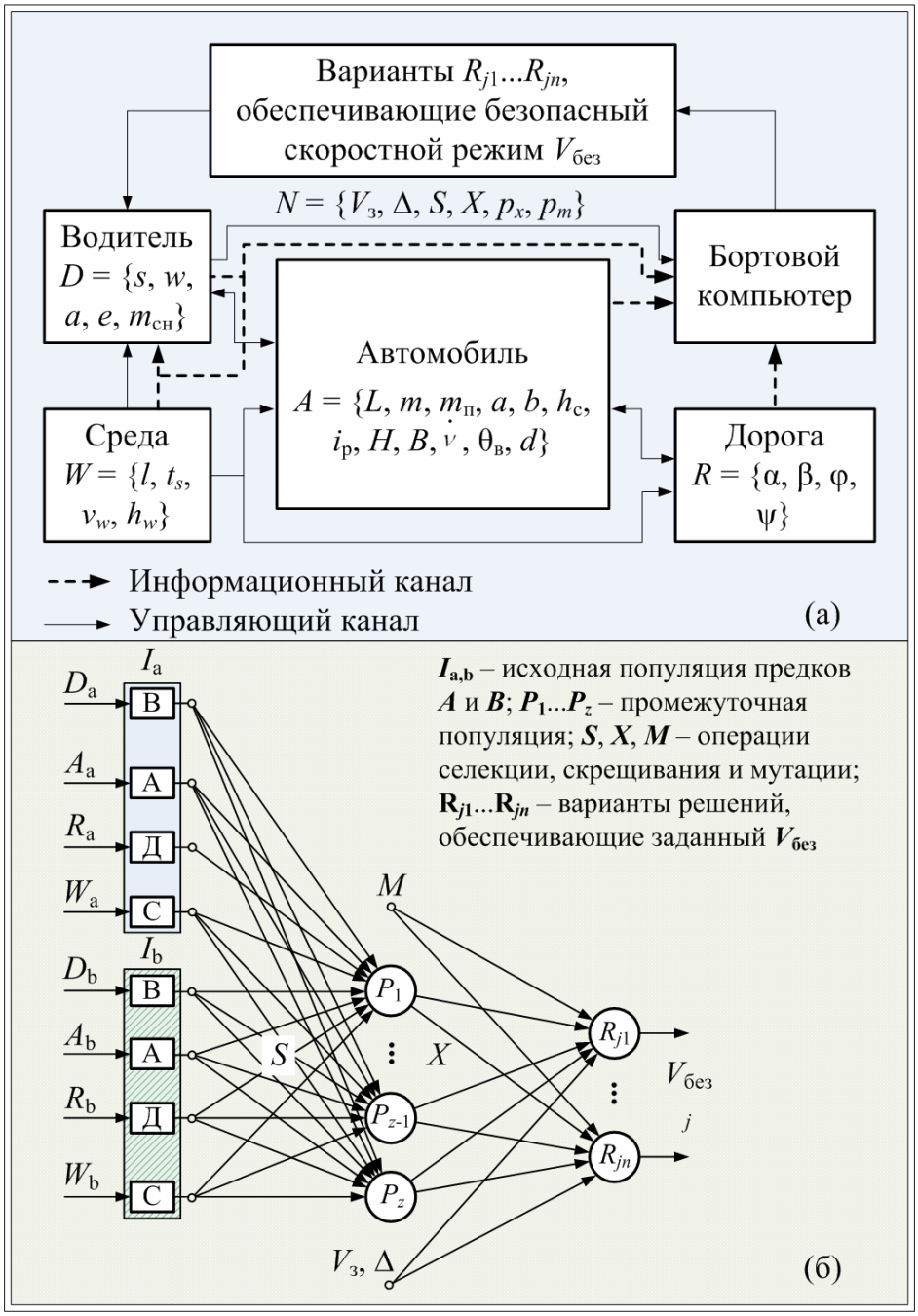
На рисунке 8.1 представлена структурная схема генетической модели системы оптимизации совокупности параметров системы ВАДС.

Входными данными модели являются следующие:

– совокупность векторов эксплуатационных параметров системы «водитель – автомобиль – дорога – среда», (см. рисунок 8.1);

– вектор настроечных параметров $\boldsymbol{N} = \{\boldsymbol{v_з}, \boldsymbol{\Delta}, \boldsymbol{S}, \boldsymbol{X}, \boldsymbol{p_x}, \boldsymbol{p_m}\}$ генетического алгоритма, содержащий значение заданного скоростного режима $\boldsymbol{v_з}$ автомобиля, величину погрешности $\boldsymbol{\Delta}$, виды селекции $\boldsymbol{S}$ и скрещивания $\boldsymbol{X}$, проценты скрещивания $\boldsymbol{p_x}$ и мутации $\boldsymbol{p_m}$ генов.

Выходными данными модели являются варианты решений $\boldsymbol{R_{j1}}\ldots\boldsymbol{R_{jn}}$, обеспечивающие диапазон заданного скоростного режима $\boldsymbol{v_з}$ автомобиля с учетом погрешности $\boldsymbol{\Delta}$.



а – модель системы ВАДС; б – структурная схема системы оптимизации $\boldsymbol{V_{без}}$

Рисунок 8.1 – Структурная схема генетической модели системы оптимизации сложной эргатической системы «водитель – автомобиль – дорога – среда»

К исходным данным задачи оптимизации скоростного режима относится: перечень векторов, содержащих параметры генетического алгоритма (ГА), системы ВАДС, и заданный скоростной режим $\boldsymbol{W = \{V, M, R, D, v_з\}}$.

Вектор входных параметров ГА имеет следующий вид:

$$V = \{h, g, p_c, p_m, g_{\min}, g_{\max}, e_t, z_t, m_c, m_s, i, z_i\}, \quad (8.1)$$

где $\boldsymbol{h}$, $\boldsymbol{g}$ – количество хромосом и генов;

p_c, p_m – процент скрещивания и мутации генов;

g_{min}, g_{max} – минимальное и максимальное значение гена;

e_t, z_t – тип экстремума целевой функции и выходных значений;

m_c, m_s – метод скрещивания и селекции;

i, z_i – тип и значение условия остановки процесса оптимизации.

На рисунке 8.2 представлена схема алгоритма оптимизации совокупности эксплуатационных параметров системы ВАДС.

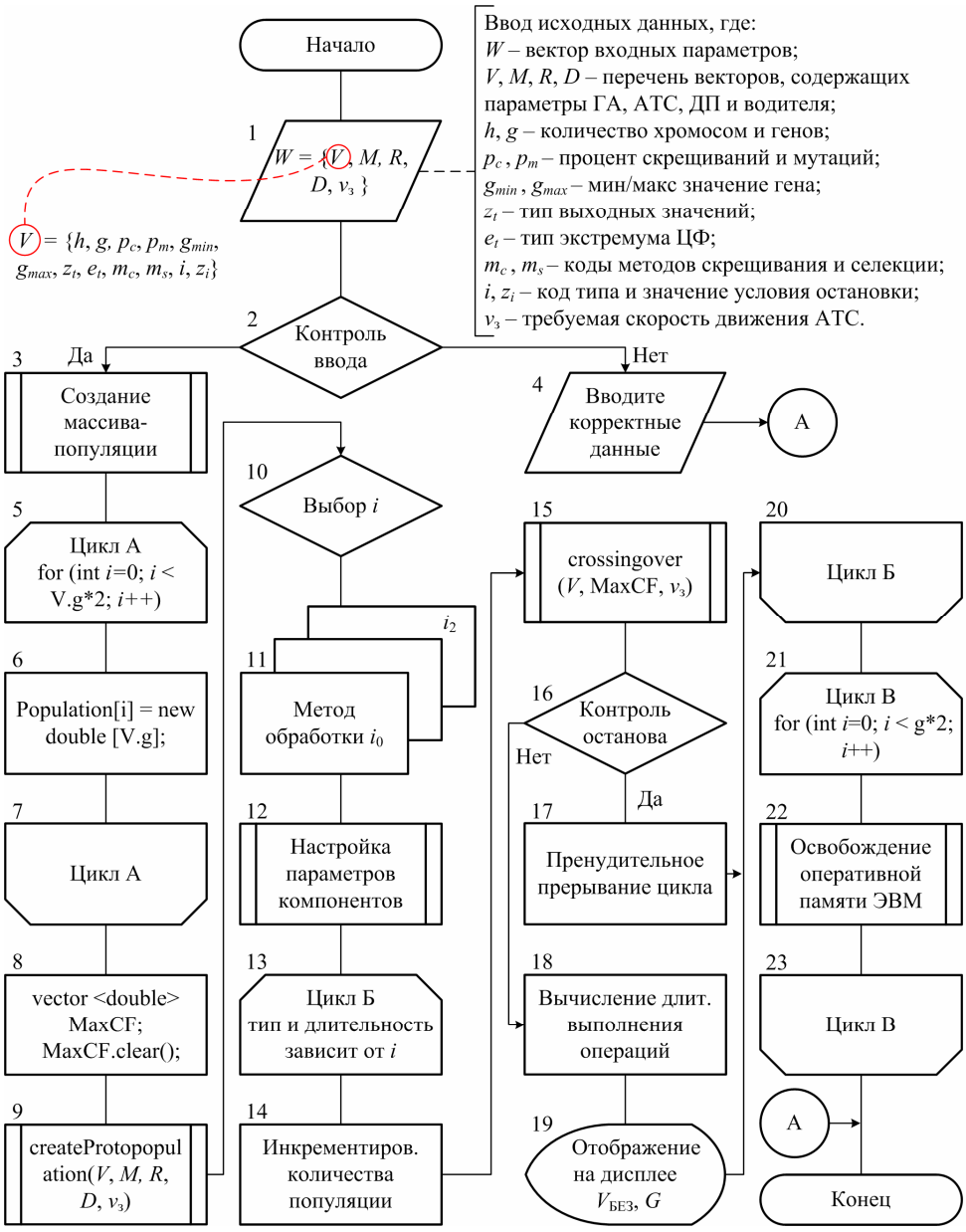


Рисунок 8.2 – Схема алгоритма оптимизации совокупности эксплуатационных параметров системы ВАДС

Алгоритм основан на принципах генетики, предусматривающих модификацию и обработку синтезированных векторов-поколений, содержащих значения эксплуатационных параметров системы ВАДС, и выбор оптимальных значений параметров исходя из условия обеспечения минимума рассогласования заданной $\boldsymbol{v}_3$ и безопасной скорости движения $\boldsymbol{v}_6$ автомобиля. При кодировании значений эксплуатационных параметров системы ВАДС, содержащихся в соответствующих векторах, использовалась фасетная система классификации с параллельным способом кодирования.

Отличительной особенностью алгоритма является наличие блоков 10, 11, которые осуществляют выбор метода обработки массива-популяции с учетом выбранного пользователем критерия остановки программы. В алгоритме программы заложены следующие условия остановки:

– по времени (длительности) нахождения оптимальных значений параметров исследуемых объектов;

– по количеству синтезированных и обработанных векторов-поколений, содержащих значения параметров исследуемых объектов;

– по экстремуму (максимуму или минимуму) целевой функции.

В блоках 14, 15 выполняется инкрементирование количества популяции и процедура скрещивания «crossingover». Данная процедура выполняет инициализацию колонии и обработку созданного массива-популяции на основе ГА с целью нахождения оптимальных параметров исследуемых объектов. В теле данной процедуры выполняются операции скрещивания и селекции, затем – мутация значений генов и вычисление целевой функции, при этом лучшие наборы векторов-поколений сохраняются в ОЗУ, дубликаты и оставшиеся наборы – удаляются.

8.2 Порядок выполнения работы

1. Запустить программу «Оптимизация системы ВАДС на основе генетического алгоритма».

2. Используя теоретический материал, представленный в пункте 7.1, разобраться в принципах генетики в задаче распознавания образов.

3. Изучить представленную прикладную программу:

– режим настройки исходной популяции, алгоритмов селекции и скрещивания генов;

– режим определения вариантов решений;

– режим сбора статистических данных.

4. В процессе работы программы пользователь должен определить достоинства и недостатки генетических принципов, которые следует указать в отчете о проделанной работе.

8.3 Задание на лабораторную работу

1. Изучить прикладную программу «Оптимизация системы ВАДС на основе генетического алгоритма».

2. Оценить длительность работы генетического алгоритма в соответствии с вариантом задания.

3. Оформить отчет о проделанной лабораторной работе.

8.4 Варианты заданий

Варианты заданий для выполнения лабораторной работы представлены в таблице 8.1.

Таблица 8.1 – Варианты заданий для выполнения лабораторной работы № 8

Вариант	Кол-во хромосом	Кол-во генов	Процент скрещиваний	Процент мутации	Алгоритм скрещивания	Алгоритм селекции	Значение ЦФ
1	15	8	95	10	одноточечный	рулетки	30 ÷ 60
2	20	6	90		поэлементный	турнирный	40 ÷ 70
3	25	8	85		двухточечный	рулетки	50 ÷ 80
4	30	6	80	15	одноточечный	турнирный	60 ÷ 90
5	35	8	75		двухточечный	рулетки	70 ÷ 100
6	произвольное	6	70	20	поэлементный	турнирный	80 ÷ 110
7	20	8	65		одноточечный и двухточечный	рулетки	90 ÷ 120
8	30	6	60	произ-вольный		турнирный	100 ÷ 130
9	произвольное	8	55			рулетки	110 ÷ 140
10	25	6	50	10		турнирный	120 ÷ 150

Продолжение таблицы 8.1

11	произвольное	8	произвольный	5	поэлементный	рулетки	130 ÷ 160
12	15	8	95	10	одноточечный	рулетки	140 ÷ 170
13	20	6	90		поэлементный	турнирный	150 ÷ 180
14	25	8	85		двухточечный	рулетки	160 ÷ 190
15	30	6	80	15	одноточечный	турнирный	170 ÷ 200
16	35	8	75		двухточечный	рулетки	180 ÷ 210
17	произвольное	6	70	20	поэлементный	турнирный	190 ÷ 220
18	15	8	95	10	одноточечный	рулетки	10 ÷ 120
19	20	6	90		поэлементный	турнирный	40 ÷ 170
20	25	8	85		двухточечный	рулетки	50 ÷ 180

8.5 Контрольные вопросы

1. Объясните алгоритм оптимизации совокупности эксплуатационных параметров системы ВАДС на основе генетического подхода.
2. Объясните принцип фасетной системы классификации с параллельным способом кодирования элементов.
3. Назовите достоинства и недостатки генетических алгоритмов.
4. Что такое сходимость генетического алгоритма?
5. Объясните понятие «мутация» применительно к генетическому алгоритму.

9 Литература, рекомендуемая для изучения тем

9.1 Основная литература

1. Аралбаев, Т.З. Контроль и управление доступом в АСУ ТП на основе биометрических характеристик пользователя / Т.З. Аралбаев, А.Г. Африн. – Уфа: Гилем, 2008. – 124 с. – ISBN 978-5-7501-0930-2.

2. Кудрявцев, В.Б. Теория тестового распознавания / В.Б. Кудрявцев, А.Е. Андреев, Э.Э. Гасанов. – М. : Физматлит, 2007. – 320 с. – ISBN 978-5-9221-0872-0.

3. Пат. 2430415 Российская Федерация, МПК G 06 K 9/00. Устройство для распознавания образов / Р.И. Хасанов, М.З. Масягутов, Т.З. Аралбаев; заявитель и патентообладатель Оренбургский государственный университет. – №2010116601/08; заявл. 26.04.2010 – опубл. 27.09.2011, Бюл. № 27. – 21 с.: ил.

4. Потапов, А.А. Новейшие методы обработки изображений: монография / под. общ. ред. А. А. Потапова. – М.: Физматлит, 2008. – 496 с. – ISBN 978-5-9221-0841-6.

5. Федотов, Н.Г. Теория признаков распознавания образов на основе стохастической геометрии и функционального анализа / Н.Г. Федотов. – М.: Физматлит, 2009. – 304 с. – ISBN 978-5-9221-0996-3.

9.2 Дополнительная литература

6. Аралбаев, Т.З. Распараллеливание вычислительного процесса в задаче распознавания распределенного графического образа: материалы IX всероссийской научно-практической конференции «Современные информационные технологии в науке, образовании и практике» / Т.З. Аралбаев, Р.И. Хасанов, Н.А. Носов. – Оренбург: ОГУ, 2010. – С. 74 – 78. – ISBN 978-5-9723-0070-9.

7. Аралбаев, Т.З. Ситуационная модель управления распознаванием образов: материалы X всероссийской научно-практической конференции «Современные информационные технологии в науке, образовании и практике» / Т.З. Аралбаев,

Р.И. Хасанов, С.В. Хасанова. – Оренбург: ОГУ, 2012. – С. 141 – 147. – ISBN 978-5-4417-0097-9.

8. Хасанов, Р.И. Определение категории дорожного полотна на основе ассоциативной модели распознавания образов / Р.И. Хасанов, Т.З. Аралбаев // Информационные системы и технологии. – 2011. – № 3 (65). – С.102 – 107. – ISSN 2072-8964.

9. Данные о солнечной активности [Электронный ресурс / Л.В. Константиновская]. – Режим доступа: WWW.astronom2000.info/астрономия/.