Санкт-Петербургский государственный университет

Кафедра компьютерных технологий и систем

Гальетов Владимир Валерьевич

Выпускная квалификационная работа бакалавра

«Система геодезического анализа карт местности»

Направление 010300

Фундаментальная информатика и информационные технологии

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Мисенов Б. А.

Санкт-Петербург

2016

Оглавление

Введение

В настоящее время происходит стремительное развитие компьютерной техники, информационных технологий. Каждый год появляется множество новых применений, внедрений этих технологий в жизнь человека. При этом тенденция развития такова, что наука и техника все больше применяются не только в технических, индустриальных сферах, но и в самых повседневных и бытовых областях деятельности человека. И здесь важную роль играет автоматизация, компьютеризация подобных механистических процессов, которые прежде приходилось выполнять человеку.

Одним из таких процессов является геодезический анализ, или дешифрирование изображений земной поверхности, полученных с помощью аэрофотосъемки, либо космофотосъемки. Если раньше распознаванием объектов земной поверхности занимались географы, то в настоящее время важной и актуальной задачей является разработка компьютерных систем, автоматически дешифрирующих эти фотоснимки.

Значимость выбранной темы заключается в том, что результаты работы подобной компьютерной системы будут полезны для решения целого ряда прикладных задач в самых разных областях. Основываясь на этих результатах, человек сможет заниматься принятием решений на более высоком уровне, а впоследствии, с еще большим развитием компьютерной индустрии, эти результаты могут быть использованы уже системами искусственного интеллекта, которые будут призваны освободить человека от необходимости принятия решений и на этом (более высоком) уровне.

Стоит заметить, что компьютерное зрение – одна из активно развивающихся областей на протяжении уже около 40 лет, имеющая множество практических приложений и играющая важную роль в процессе формирования интеллектуальных систем, разработка которых активно

ведется в настоящее время. Таким образом, решение различных задач и исследования в этой области сейчас являются актуальными.

В данной работе предлагается рассмотрение двух существующих методов дешифрирования (распознавания) изображений, их реализация с предложениями модификации для получения удовлетворительных результатов в конкретной предметной области, построение на их основе двух систем, способных распознавать несколько заранее определенных классов объектов. Кроме того, предлагается тестирование и анализ предлагаемых систем по эффективности.

Постановка задачи

Целью данной работы является исследование методов распознавания изображений и разработка двух систем, основанных на выбранных методах, анализ и тестирование полученных систем, выделение их достоинств и недостатков, заключение о возможности формирования на их основе самодостаточной системы распознавания и классификации объектов земной поверхности.

Для достижения указанной цели необходимо решить следующие задачи:

- провести исследование существующих методов распознавания изображений и выбор двух из них, потенциально подходящих для распознавания на фотоснимках объектов земной поверхности;
- осуществить необходимые модификации выбранных методов с целью учета особенностей предметной области;
- реализовать две системы, способные распознавать объекты земной поверхности;
- провести тестирование разработанных систем, их анализ и оценку эффективности.

Обзор литературы

Изучение литературы, научных статей и интернет-источников в сфере дистанционного зондирования Земли, мониторинга земной поверхности позволяет говорить о том, что такие задачи, как дешифрирование (интерпретация) аэрофотоснимков и космофотоснимков, а главное – автоматизация этого процесса, являются важными прикладными задачами, требующими эффективного решения.

При этом требования детализации решений задач дешифрирования, распознавания могут варьироваться от выделения на снимках объектов только одного вида до полного картографирования территории.

В настоящее время активно разрабатываются прикладные программные системы, содержащие подсистемы дешифрирования снимков. Назначение таких систем – анализ и получение новых данных (например, в военных целях), получение карт.

Подобные системы могут применяться в таких областях, как сельское хозяйство (задачи классификации, различных оценок, мониторинга), геодезия (в основном – геологические и военные задачи), исследования природной среды – мониторинг лесного покрова (задачи разграничения территории) и мониторинг поверхности в целом (создание более новых карт, планирование рационального использования природных ресурсов, расчет рисков, экологическая защита). [1]

Одной из наиболее сложных проблем является распознавание текстур природных объектов. Стоит упомянуть о том, что при решении задач распознавания текстур наиболее важным является нахождение такого алгоритма и его параметров, что они будут работать наиболее универсально для разных видов текстур, так как многие алгоритмы можно настроить таким образом, что для конкретного случая он будет работать практически идеально, а для другого – давать неудовлетворительные результаты [2, 3].

Для реализации таких алгоритмов были выбраны два метода – метод сравнения гистограмм и метод опорных векторов.

В [4, 5] подробно и формально описан метод опорных векторов, а также содержится общая информация об алгоритмах машинного обучения.

В [6, 7] описаны подробности реализации алгоритмов при помощи графической библиотеки OpenCV.

В [8] описаны основные подходы к распознаванию объектов на спутниковых снимках.

В [9] содержится информация о практическом тестировании метода опорных векторов.

Глава 1. Анализ и теоретические основы реализуемых методов

В этой главе подробно рассматриваются методы, выбранные для решения поставленной задачи, включая теоретические основы, необходимые модификации и улучшения.

1.1 Методы сравнения гистограмм

1.1.1 Гистограммы

Цветовые гистограммы часто используются в задачах распознавания объектов на изображениях, классификации, нахождения сходных изображений. В общем случае, гистограмма – это некоторый набор, коллекция данных, представленных в виде распределения значений, которые содержат информацию о количестве точек, отражающих какое-либо свойство объекта.

Гистограммы можно применять для описания изображений. С помощью гистограмм возможно несколько вариантов описания изображения: пиксели, градиенты, направления и т.д. В работе гистограмма рассматривается как распределение значений, каждое из которых отвечает количеству пикселей изображения с определенной интенсивностью. Здесь стоит отметить, что можно группировать несколько пикселей с разной интенсивностью (т.е. несколько точек), такие группы называются *бинами*. В этом случае, значение гистограммы, соответствующее такому бину, будет вычисляться как количество пикселей, которые содержит бин.

Стоит учитывать разницу между вычислением гистограмм для изображений в градациях серого и для цветных изображений. Чтобы вычислить гистограмму для изображения в градациях серого, необходимо просто подсчитать количество пикселей на изображении для каждого значения интенсивности, которое содержится в диапазоне от 0 до 255 (то есть

в данном случае изображение является одноканальным, для хранения одного пикселя достаточно одного байта информации). Однако, для того, чтобы вычислить гистограмму для цветного изображения, необходимо предварительно разделить исходное изображение на три «плоскости» (по числу каналов в цветном изображении в модели RGB), каждая из которых будет содержать значения интенсивности для каждого из трех каналов. На рисунке 1 можно видеть цветное изображение и построенные для нее гистограммы для каждого из RGB-каналов.

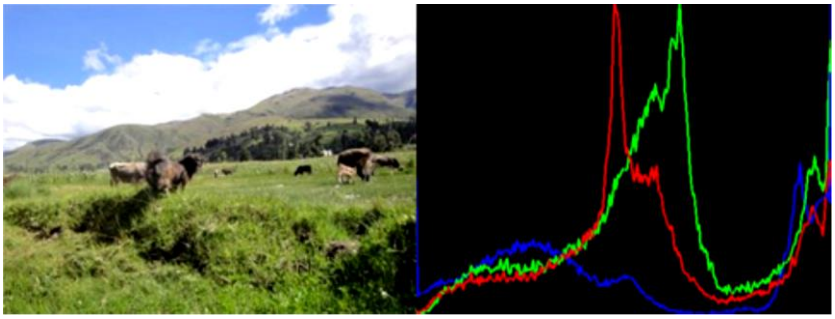

Рис. 1. Цветное изображение (слева), гистограммы для 3-х каналов (справа)

1.1.2 Особенности применения методов

Одним из наиболее простых и широко используемых методов поиска сходных изображений является метод сравнения гистограмм. В рассматриваемой работе этот метод используется для классификации текстур на изображении земной поверхности. Для идентификации текстур (лесного массива, водоемов и т.д.) на изображении используется подход сравнения с эталоном. Таким образом, процедура классификации, в этом случае, сводится к разбиению тестового изображения на множество квадратных участков небольшого размера, построению для них гистограмм и их последующему сравнению с гистограммами, построенными для эталонных участков текстур такого же размера. Такой метод последовательного сравнения гистограммы эталона с гистограммой «окна» – прямоугольной области, захватывающей часть тестового изображения, и сдвигающейся по всему этому изображению, называется методом «скользящего окна». Его недостаток заключается в том,

что при сравнении эталона с участком, на котором помимо искомого объекта есть что-то еще, скорее всего, метод сработает неправильно из-за искажения гистограммы. Недостаток можно исправить путем добавления в систему детектора границ, отслеживающего однородные блоки текстур, однако, в данной работе это не рассматривается ввиду работы с более крупным масштабом изображений, т.е. несущественности суммарной ошибки. Еще одной особенностью данного метода является достаточно высокая чувствительность к выбору количества бинов (при малом количестве различия будут критично несущественны, при большом количестве будет неестественно высокая уникальность каждого участка текстуры), поэтому важно эмпирическим путем отыскать оптимальные количества.

Поскольку цвет является одной из важных особенностей для каждой из текстур, присутствующих на большинстве изображений поверхностей Земли, то гистограммы имеет смысл вычислять именно для цветных изображений. В предыдущем подпараграфе был кратко рассмотрен подход к построению гистограмм для цветных изображений. Однако, в силу ряда причин, гистограммы желательно строить для изображений в HSV-модели (а не RGB), координатами которой являются цветовой тон, насыщенность и значение цвета (англ. Hue, Saturation, Value). Эта модель является нелинейным преобразованием модели RGB. Одними из причин, по которым используется именно HSV-модель, являются более высокая (чем в модели RGB) инвариантность к степени освещенности, устойчивость к затенению, более удобная с программной точки зрения эквализация и нормализация гистограмм, наконец, естественность координат модели для восприятия человеком.

1.1.3 Методы сравнения гистограмм

Замечание. Перед тем, как сравнивать, гистограммы требуется нормализовать, т.е. сумма значений бинов должна равняться единице.

Все методы вычисляют некое «расстояние», или уровень сходства между гистограммами – распределениями пикселей.

Обозначения:

$H_1(i)$ – значение функции распределения в i-м бине;

$\bar{H}_1$ – среднее по распределению;

Рассмотрим 4 основных метода сравнения гистограмм:

1. Корреляционный метод, где $d(H_1, H_2)$ находится в интервале [-1; 1], в котором нижняя и верхняя граница определяют максимальную степень соответствия. Ноль означает отсутствие корреляции;

$$d(H_1, H_2) = \frac{\sum_i (H_1(i) - \bar{H}_1)(H_2(i) - \bar{H}_2)}{\sqrt{\sum_i (H_1(i) - \bar{H}_1)^2 * (H_2(i) - \bar{H}_2)^2}}$$

2. Метод «Хи-квадрат», где $d(H_1, H_2)$ находится в интервале [0; $+\infty$), а ноль означает максимальную степень соответствия, в то время как степень минимального соответствия обусловлена количеством бинов;

$$d(H_1, H_2) = \sum_i \frac{(H_1(i) - H_2(i))^2}{H_1(i)}$$

3. Метод пересечения гистограмм, где $d(H_1, H_2)$ находится в интервале [0; 1], в котором нижняя граница определяет минимальную степень соответствия, а верхняя – максимальную;

$$d(H_1, H_2) = \sum_i \min(H_1(i), H_2(i))$$

4. Метод «Расстояние Бхаттачарья», где $d(H_1, H_2)$ находится в интервале [0; 1], в котором нижняя граница определяет максимальную степень соответствия, а верхняя – минимальную.

$$d(H_1, H_2) = \sqrt{1 - \sum_i \frac{\sqrt{H_1(i) * H_2(i)}}{\sqrt{\sum_i H_1(i) * \sum_i H_2(i)}}}$$

В силу того, что в методе «Хи-квадрат» степень минимального соответствия обусловлена, его использование в программной реализации является не вполне оправданным, поэтому было принято решение от него отказаться.

1.2 Метод опорных векторов

В этом параграфе рассматривается метод опорных векторов, в англоязычных источниках также известном, как Support Vector Machine (SVM). Также предлагаются некоторые дополнения, необходимые для применения метода в задачах классификации изображений.

1.2.1 Метод опорных векторов для случая линейной разделимости

Метод опорных векторов – это метод, который в теории машинного обучения рассматривается как метод *обучения с учителем*. Это означает, что перед тем как система, основанная на методе, допускается к распознаванию тестовых данных, ее «тренируют» при помощи *обучающей выборки*, то есть подают на вход системе множество примеров и соответствующих им ответов, после чего она выстраивает зависимость между ними и далее способна прогнозировать ответы уже для тестовых данных. Метод относится к категории дискриминативных линейных классификаторов. Первое означает, что для его работы не имеет значения, как были сгенерированы данные, а второе – что классификатор принимает решения на основе значения линейной комбинации характеристик объектов, называемых также

признаками, которые в машинном представлении хранятся в векторах, называемых векторами особенностей, или векторами признаков. Иными словами, метод строит линейную разделяющую поверхность (если имеются два класса, поверхность называется *разделяющей гиперплоскостью*).

Метод опорных векторов в случае, если обучающие данные линейно разделимы, осуществляется следующим образом: дано множество точек, принадлежащих к одному из двух классов, классификатор находит *оптимальную гиперплоскость,* оставляющую наибольшее количество точек одного класса с одной стороны, второго класса – с другой, причем расстояние от гиперплоскости до точек каждого из классов максимизируется.

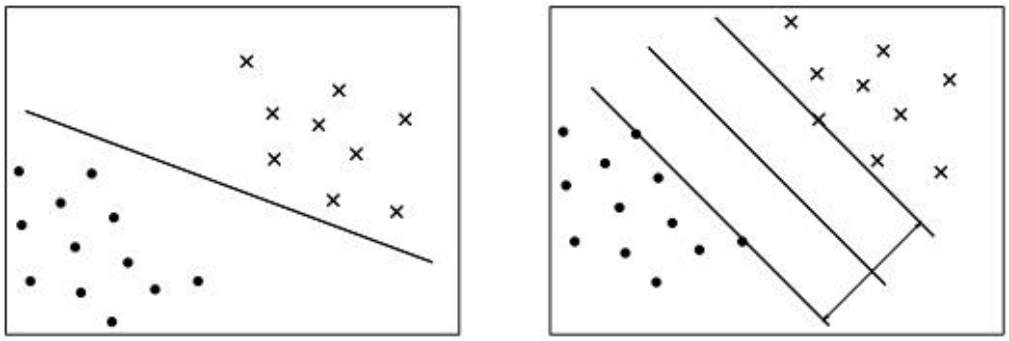

Рис. 2. Разделяющие гиперплоскости, справа – оптимальная

На рисунке 2 можно видеть две гиперплоскости, каждая из которых правильно разделяет множества точек на два класса, но только правая гиперплоскость является оптимальной, так как максимизирует зазор (margin) между параллельными гиперплоскостями, касающимися точек. Величина зазора и ошибка классификатора связаны между собой обратно пропорционально.

Формальная постановка задачи построения линейного классификатора по методу SVM.

Имеются обучающие данные из n точек

$$(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n),$$

где y_i может быть равным 1 или -1, указывая на класс, которому принадлежит одна из точек $\vec{x}_i$, из которых состоит вещественный вектор с

размерностью, равной p. Нужно отыскать оптимальную гиперплоскость, разделяющую множество точек, для которых $y_i = 1$ и множество точек, для которых $y_i = -1$, причем расстояние от нее до ближайших точек из обоих классов должно быть максимальным.

Гиперплоскость может быть выражена следующим уравнением:

$$\vec{w} \cdot \vec{x} - b = 0,$$

где вектор $\vec{w}$ – нормальный вектор к гиперплоскости, а величина $\frac{b}{\|\vec{w}\|}$ – расстояние от гиперплоскости до начала координат.

В случае линейной разделимости обучающих данных можно построить две параллельные гиперплоскости, с наибольшим зазором разделяющие данные. Оптимальная гиперплоскость параллельна им и лежит посередине между ними. Эти две гиперплоскости можно описать следующими уравнениями:

$$\vec{w} \cdot \vec{x} - b = 1 \text{ и } \vec{w} \cdot \vec{x} - b = -1.$$

Соответственно, расстояние между касающимися данных гиперплоскостями равно $\frac{2}{\|\vec{w}\|}$, то есть для его максимизации нужно минимизировать знаменатель. Для того чтобы не допустить попадания точек в зазор, нужно добавить следующее ограничение, тогда каждая из точек будет лежать на правильной стороне относительно гиперплоскости:

$$\vec{w} \cdot \overrightarrow{x_i} - b \geq 1, \text{если } y_i = 1 \text{ или } \vec{w} \cdot \overrightarrow{x_i} - b \leq -1, \text{если } y_i = -1.$$

Объединяя эти ограничения, можно выразить их следующим образом:

$$y_i(\vec{w} \cdot \overrightarrow{x_i} - b) \geq 1, \text{для всех } 1 \leq i \leq n. \quad (1)$$

Теперь требования максимизации зазора между гиперплоскостями и нахождения точек в своих классах можно выразить через постановку задачи оптимизации:

Минимизировать $\|\vec{w}\|$ при условии $y_i(\vec{w} \cdot \overrightarrow{x_i} - b) \geq 1,\ i = 1, \dots, n$

$\vec{w}$ и b, решающие задачу, определяют классификатор $\vec{x} \mapsto sign(\vec{w} \cdot \overrightarrow{x_i} - b)$.

Важно заметить, что оптимальная гиперплоскость полностью определяется теми точками $\overrightarrow{x_i}$, которые лежат к ней ближе всего. Точки $\overrightarrow{x_i}$ называются *опорными векторами*.

1.2.2 Метод опорных векторов для случая линейной неразделимости

В этом подпараграфе проведено обобщение метода на случай, если обучающие данные линейно неразделимы, так как большинство приложений, в которых используется SVM, требуют более мощный инструмент, нежели линейный классификатор, так как в подобных задачах обучающие данные редко можно разделить гиперплоскостью. Это связано в том числе с тем, что для обучения SVM необходимо загружать сложную совокупность позитивных и негативных примеров распознавания, а в этом случае может оказаться невозможно линейно отделить их друг от друга.

В случае линейной неразделимости в методе делается допущение, что некоторые из точек будут классифицированы неверно. Величина ошибочной классификации вводится как отдельная переменная в задачу оптимизации. Соответственно, задача может быть переформулирована как поиск разделяющей гиперплоскости с максимальным зазором при минимуме ошибок классификации.

Рассмотрим функцию потерь:

$$\max(0, 1 - y_i(\vec{w} \cdot \overrightarrow{x_i} - b)).$$

Она равна нулю в том случае, если выполнено ограничение (1) из предыдущего подпараграфа, то есть все $\overrightarrow{x_i}$ лежат на правильных сторонах относительно гиперплоскости, для остальных же $\overrightarrow{x_i}$ значения функции будут пропорциональны расстоянию от зазора. Тогда нужно минимизировать

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max(0, 1 - y_i(\vec{w} \cdot \overrightarrow{x_i} - b))\right] + \lambda\|\vec{w}\|^2,$$

где λ определяет компромисс между увеличением зазора и гарантией того, что точки $\overrightarrow{x_i}$ будут классифицированы верно. Эту минимизацию можно

переписать в виде оптимизационной задачи с ограничениями и дифференцируемой целевой функцией следующим образом:

Для каждого $i \in \{1, \ldots, n\}$ нужно ввести переменную, равную

$$\xi_i = \max(0, 1 - y_i(\vec{w} \cdot \vec{x_i} - b))$$

тогда и только тогда, когда ξ_i – наименьшее неотрицательное число, удовлетворяющее неравенству

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i.$$

Замечание. Геометрический смысл переменных ξ_i в том, что каждая из них обозначает расстояние от точки данных до правильной области решения. Если точка уже находится в этой области, то переменная равна нулю.

Таким образом, оптимизационную задачу можно представить в виде:

$$\text{minimize } \frac{1}{n}\sum_{i=1}^{n} \xi_i + \lambda\|\vec{w}\|^2,$$

при условии, что $y_i(x_i \cdot w + b) \geq 1 - \xi_i$ и $\xi_i \geq 0$ для всех i.

Метод опорных векторов способен эффективно решать задачу нелинейной классификации, используя способ, который называется в англоязычных источниках “kernel trick”, при этом происходит неявное отображение данных в пространство признаков большей размерности, в предположении, что там данные будут линейно разделимы. Идея способа заключается в том, что вместо скалярного произведения точек теперь используется нелинейная функция ядра, фактически выполняющая ту же роль, только в пространстве большей размерности. Функция, отвечающая оптимальной решающей функции в первоначальном пространстве, будет нелинейной в силу нелинейности ее отображения в пространство большей размерности.

Некоторые часто используемые ядра (в скобках – обозначения в библиотеке OpenCV):

- Полиномиальное (однородное) (POLY):

$$k(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j})^d$$

- Полиномиальное (неоднородное) (POLY):

$$k(\overrightarrow{x_i}, \overrightarrow{x_j}) = (\overrightarrow{x_i} \cdot \overrightarrow{x_j} + 1)^d$$

- Гауссовская радиальная базисная функция (RBF):

$$k(\overrightarrow{x_i}, \overrightarrow{x_j}) = e^{(-\gamma \|\overrightarrow{x_i} - \overrightarrow{x_j}\|^2)}, \text{для } \gamma > 0;$$

может использоваться параметр $\gamma = 1/2\sigma^2$

- Гиперболический тангенс (SIGMOID):

$$k(\overrightarrow{x_i}, \overrightarrow{x_j}) = \tanh(\gamma \overrightarrow{x_i} \cdot \overrightarrow{x_j} + c),$$

для некоторых (не всяких) $\gamma > 0$ и $c < 0$

Ядро связано с преобразованием $\varphi(\overrightarrow{x_i})$ уравнением

$$k(\overrightarrow{x_i}, \overrightarrow{x_j}) = \varphi(\overrightarrow{x_i}) \cdot \varphi(\overrightarrow{x_j}).$$

Величина w в преобразованном пространстве $\vec{w} = \sum_i \alpha_i y_i \varphi(\overrightarrow{x_i})$. Произведения данных с этой величиной могут быть вычислены с помощью “kernel trick”:

$$\vec{w} \cdot \varphi(\vec{x}) = \sum_i \alpha_i y_i \, k(\overrightarrow{x_i}, \vec{x}).$$

Глава 2. Реализация методов

В этой главе содержится обоснование выбора программных средств, используемых при реализации описанных выше методов, а также их краткий обзор. Помимо этого, глава содержит сведения о способе реализации систем распознавания, специфические детали, модификации.

2.1 Обзор инструментария

На данный момент существует довольно небольшое количество возможных путей программной реализации задач, относящихся к области распознавания и классификации изображений, в силу малого количества библиотек, связанных именно с компьютерным зрением. Для выполнения этой работы была выбрана библиотека OpenCV. Среди ее преимуществ можно отметить следующие:

- OpenCV имеет BSD лицензию, а это значит, ее можно свободное использовать как в академических, так и в коммерческих продуктах;
- OpenCV содержит большое количество модулей и алгоритмов, в том числе SVM и широкие возможности для работы с гистограммами;
- Поддержка многих языков программирования, платформ и операционных систем;
- Высокая популярность и большая поддержка – обширная документация, туториалы, интернет-поддержка, что особенно важно для начинающих разработчиков.

Разработка с использованием графической библиотеки OpenCV возможна на нескольких языках программирования, среди которых C, C++, Python, Java, MATLAB [10]. Для программной реализации был выбран C++ по следующим причинам:

- высокая производительность;

- содержательная OpenCV-документация;
- широкая поддержка конкретно связки OpenCV – C++ – большое сообщество разработчиков, множество примеров реализаций в литературе и интернет-источниках.

Для разработки систем классификации использовались следующие программные продукты:

- Графическая библиотека OpenCV версии 3.0.0;
- Интегрированная среда разработки Microsoft Visual Studio Community 2015.

2.2 Обзор программной реализации

2.2.1 Метод сравнения гистограмм

Известно, что при сравнении гистограмм необходимо загрузить тестовое изображение и эталонные участки изображений текстур распознаваемых классов в цвете. При сравнении используется так называемое «скользящее окно», при этом последовательно сравниваются гистограммы эталона и такого же по размеру элемента тестового изображения.

Так выглядит функция, в которой реализована концепция «скользящего окна»:

```
int rshift = 0, dshift = 0;
for (int i = 0; i < c_cols; i++) {
    for (int j = 0; j < c_rows; j++) {
        Rect r(0 + rshift, 0 + dshift, side, side);
        Mat copy = test;
        Mat crop = copy(r);
        if (hist_comparison(etalon, crop)) test(r) += CV_RGB(0, 0,
255);
```

```
            dshift += side;
            if (j == c_rows - 1) dshift = 0;
        }
        rshift += side;
    }
```

Приведем подробное описание реализации метода сравнения:

1. Сначала необходимо преобразовать два квадратных участка (эталон и такая же часть исходного изображения, например, размером 20x20 пикселей):

```
Mat hsv_etalon, hsv_test_crop;
cvtColor(etalon, hsv_etalon, COLOR_BGR2HSV);
cvtColor(test_crop, hsv_test_crop, COLOR_BGR2HSV);
```

2. Затем необходимо задать количество бинов для канала тона и насыщенности, диапазоны их изменения (H, S), и количество каналов:

```
int h_bins = 10; int s_bins = 60;
int histSize[] = { h_bins, s_bins };

float h_ranges[] = { 0, 180 };
float s_ranges[] = { 0, 256 };
const float* ranges[] = { h_ranges, s_ranges };
int channels[] = { 0, 1 };
```

3. Далее необходимо создать матрицы для хранения гистограмм и вызвать функции вычисления гистограмм (последние два аргумента – равномерность и отсутствие накопления данных), после чего нормализовать гистограммы (здесь 0 и 1 – нижняя и верхняя границы нормализации диапазона соответственно, 5-й аргумент указывает на способ нормализации внутренних значений, 6-й на то, что выходные данные будут иметь тот же тип, последний – тип маски):

```
Mat hist_etalon;
Mat hist_test_crop;

calcHist(&hsv_etalon, 1, channels, Mat(), hist_etalon, 2, histSize,
ranges, true, false);
```

```
normalize(hist_etalon, hist_etalon, 0, 1, NORM_MINMAX, -1, Mat());

calcHist(&hsv_test_crop, 1, channels, Mat(), hist_test_crop, 2,
histSize, ranges, true, false);
normalize(hist_test_crop, hist_test_crop, 0, 1, NORM_MINMAX, -1,
Mat());
```

4. Наконец, подготовленные данных выгружаются в библиотечную функцию сравнения гистограмм *compareHist*, последний ее аргумент указывает на метод сравнения гистограмм (в данном случае – пересечение). Далее устанавливается допустимый порог, отвечающий за требовательность к совпадению гистограмм. Функция сравнения паттернов – булева, в случае удовлетворительного результата сравнения возвращает *true*.

```
double test = compareHist(hist_etalon, hist_test_crop,
CV_COMP_INTERSECT);
double match_percent = 0.35;
if (test >= match_percent) return 1;
else return 0;
```

2.2.2 SVM

Стоит отметить, что при реализации метода опорных векторов существуют несколько потенциальных недостатков, среди которых:

- необходимость полной маркировки обучающих данных
- применимость только для двух классов
- затруднительность интерпретации параметров решаемой модели и их подбора для наилучшей классификации

В реализации метода SVM важной частью задачи является подготовка обучающих данных. Далее прилагается пример загрузки данных в SVM в случае, когда размер квадрата равен 15x15 пикселей, соответственно, размер

обучающей выборки составит $2 \cdot 88 \cdot 43 = 7568$ (при размере изображения, равном 1334x645, на котором помещается выборка).

Замечание. Данные в SVM загружаются в виде изображений в градациях серого, то есть распознающий алгоритм использует больше информацию о текстуре, нежели информацию о цветовой составляющей. Таким образом, процедура загрузки данных существенно ускоряется.

Ниже приведен пример заполнения половины обучающей выборки, которая сначала выгружается из изображения в двумерный массив матриц-изображений 15x15, откуда извлекается для загрузки в SVM-модель. Предварительно создается контейнер под всю выборку, последовательно заполняется половина выборки, содержащая «позитивные» примеры, таким же образом заполняется вторая половина, содержащая «негативные» примеры.

Замечание. SVM-модель OpenCV требует загрузки данных следующим образом – все обучающие примеры, в данном случае, изображения 15x15, необходимо «развернуть» построчно, таким образом, в общем контейнере каждая строка будет содержать один обучающий пример-изображение. То есть вектором особенностей в данном случае будет вектор-строка, содержащая цифровое представление изображения участка текстуры.

```
int file_num = 0, column = 0;
for (int k = 0; k < my_cols; k++) {
    for (int l = 0; l < my_rows; l++) {
        for (int i = 0; i < side; i++)
            for (int j = 0; j < side; j++)
                training_mat.at<uchar>(file_num, column++) =
arr_positive_sample[k][l].at<uchar>(i, j);
        file_num++;
        column = 0;
    }
}
```

Затем каждую строку, загруженную в большой контейнер, необходимо «пометить» либо 1, либо –1, поместив, таким образом,

обучающее изображение в один из двух классов. Для этого создается одностолбцовая матрица с количеством строк, равным количеству строк в большом контейнере.

```
Mat labelsMat(num_files, 1, CV_32SC1);
    for (int i = 0; i < num_files; i++)
        if (i < border) labelsMat.at<int>(i, 0) = 1;
        else labelsMat.at<int>(i, 0) = -1;
```

После этого устанавливаются необходимые параметры модели SVM, загружаются обучающие данные. Сначала выбирается тип модели с возможностью выбора параметра C (в предыдущей главе был обозначен как λ), отвечающего за штрафы при неправильной классификации. Далее выбираются параметр C, тип ядра, параметры γ, степень и коэффициент, которые имеют значение, если выбранный тип ядра использует их. Параметр *Termination Criteria* отвечает за ручную установку максимального числа итераций (это необходимо, чтобы регулировать точность нелинейной классификации), а так же ошибки допуска. Последняя строка запускает процесс обучения с указанной обучающей выборкой и метками.

```
Ptr<SVM> svm = SVM::create();
svm->setType(SVM::C_SVC);
svm->setC(0.3);
svm->setKernel(SVM::RBF);
svm->setGamma(0.0003);
svm->setDegree(2);
svm->setCoef0(0);
svm->setTermCriteria(TermCriteria(TermCriteria::MAX_ITER +
TermCriteria::EPS, 1000, 1e-4));
svm->train(trainingDataMat, ROW_SAMPLE, labelsMat);
```

После проведения обучения программа проходит все изображение «скользящим окном» и, получая ответы 1 или –1, закрашивает искомую текстуру. Запрашивая у обученной системы отклик, необходимо так же «разворачивать» изображение в строку.

```
int rshift = 0, dshift = 0;
    for (int i = 0; i < my_cols; i++) {
        for (int j = 0; j < my_rows; j++) {
            Rect r(0 + rshift, 0 + dshift, side, side);
            Mat crop = image(r);
            Mat copy(1, img_area, CV_32F);
            int column = 0;
            for (int i = 0; i < side; i++) {
                for (int j = 0; j < side; j++) {
                    copy.at<float>(0, column++) = crop.at<uchar>(i, j);
                }
            }
            float response = svm->predict(copy);
            if (response == 1) image_c(r) += CV_RGB(0, 200, 0);
            else if (response == -1) image_c(r) += CV_RGB(200, 0, 0);
            dshift += side;
            if (j == my_rows - 1) dshift = 0;
        }
        rshift += side;
    }
```

Глава 3. Тестирование систем

В этой главе рассмотрены результаты работы реализованных систем при изменении различных параметров.

Конфигурация компьютера:

Процессор: Intel Pentium CPU B980 2.40GHz x64

ОЗУ: 4 Gb DDR3

ОС: Windows 10 x64

Тесты проводились без применения аппаратного ускорения расчетов на GPU

3.1 Тестирование методов сравнения гистограмм

В этом параграфе проведено тестирование трех методов сравнения гистограмм – корреляции, пересечения и метода расстояния Бхаттачарья, и выбран лучший, исходя из оптимальности сочетания тройки качество-скорость-стабильность.

Тесты проводились на четырех изображениях размером 1320x645 пикселей, с различным содержанием, на примере распознавания лесных массивов. Количество бинов зафиксировано – *h_bins = 5; s_bins = 60*. При сравнении каждого участка размером 15x15 пикселей будет происходить сравнение гистограмм тестового участка и 9 примеров данного класса, при этом алгоритм оптимизирован таким образом, что при разовом закрашивании участка происходит выход из проверки оставшихся примеров. Качество измеряется количеством штрафных участков, которое составляет сумму из неправильно распознанных и не распознанных правильно участков. Для каждого метода порог распознавания устанавливается отдельно.

Таблица 1. Время обработки изображения (с/изображение 1320x645)

Номер изображения	Метод пересечения	Метод корреляции	Расстояние Бхаттачарья
1	3.185	3.854	3.076
2	4.294	3.629	4.054
3	4.549	4.563	4.952
4	5.641	6.018	6.376

Таблица 2. Качество обработки изображения (количество штрафных баллов)

Номер изображения	Метод пересечения	Метод корреляции	Расстояние Бхаттачарья
1	80	30	30
2	400	400	700
3	350	300	250
4	200	50	50

По результатам эксперимента можно сделать вывод, что лучшим является метод корреляции, так как он наиболее стабилен, а также является более быстрым методом, по сравнению с аналогичным по качеству «расстоянием Бхаттачарья».

3.2 Тестирование SVM

В этом параграфе предлагается, подобно предыдущему параграфу, зафиксировать остальные параметры в некоем стабильном состоянии и провести сравнение ядер. Эмпирическим путем было выяснено, что в качестве рабочих ядер для данной задачи можно использовать радиальную базисную функцию (RBF), ядро «Хи-квадрат» (CHI2) и ядро пересечения гистограмм (INTER). Параметры ядер на протяжении эксперимента

подобраны примерно оптимально. Тесты проводились на тех же данных, что и в предыдущем параграфе.

Таблица 3. Время обработки изображения (с/изображение 1320x645)

Номер изображения	RBF	CHI2	INTER
1	3.595	17.375	3.760
2	3.672	16.398	3.734
3	3.561	17.682	3.554
4	3.600	20.224	3.772

Таблица 4. Качество обработки изображения (количество штрафных баллов)

Номер изображения	RBF	CHI2	INTER
1	150	>>1000	400
2	200	>>1000	1000
3	700	>>1000	>>1000
4	150	>>1000	400

По результатам эксперимента можно сделать вывод, что наилучшим ядром для решения поставленной задачи оказалось ядро RBF, что подтверждает информацию в документации OpenCV, где говорится о том, что это ядро подходит во многих случаях. Остальные же ядра показали либо очень плохое время и результаты (CHI2), либо высокую нестабильность применительно к поставленной задаче.

Качественный результат распознавания изображения 2, где результат, полученный с помощью гистограмм, был неудовлетворительным, связан с качественным распознаванием форм объекта.

3.3 Сравнительное тестирование метода пересечения гистограмм и SVM с ядром RBF

В этом параграфе проведено сравнительное тестирование хорошо зарекомендовавших себя методов – метода пересечения гистограмм и метода опорных векторов, производящем переход в пространство большей размерности при помощи ядра RBF. Теперь методы распознают другой класс объектов – водоемы. Условия те же, что и в предыдущих параграфах, помимо корректировки некоторых параметров – порога в методе пересечения гистограмм и параметров γ и C в методе опорных векторов. Изображения другие, нежели в предыдущих методах.

Таблица 5. Время обработки изображения (с/изображение 1320x645)

Номер изображения	**Метод пересечения гистограмм**	**Метод опорных векторов**
1	6.417	2.448
2	5.217	2.827
3	5.428	2.376
4	6.937	2.323

Таблица 2. Качество обработки изображения (количество штрафных баллов)

Номер изображения	**Метод пересечения гистограмм**	**Метод опорных векторов**
1	50	15
2	300	300
3	400	30
4	20	25

На основе тестовых данных можно заключить, что метод опорных векторов более стабилен, быстр и работает более качественно, что же касается метода пересечения гистограмм, его имеет смысл использовать

только как дополнительный метод в комплексной системе, без перебора выборочных данных, так как это замедляет работу системы и не дает качественных результатов, так как существует большая зависимость между ними и выборкой.

3.4 Влияние различных параметров на качество распознавания

3.4.1 Параметры метода опорных векторов

Стоит заметить, что метод SVM можно использовать в режиме один-против-одного, когда все изображение распознается либо как один, либо как другой класс, однако данный режим практически не применим в реальных системах, так как работает только на изображениях, где присутствуют только два этих класса. Поэтому гораздо более предпочтителен режим один-против-всех и создание негативной выборки для каждого класса.

Негативная выборка является важнейшим аспектом системы SVM и ее правильный подбор может оказать колоссальное влияние на качество распознавания, при этом крайне важно учитывать соответствие масштаба выборки и тестовых изображений. Фактически, обучение системы, создание выборок определяет качество распознавания.

В методе SVM большая длительность обучения не сильно влияет на результат, т.е., после определенного порога с ростом времени обучения качество распознавания существенно не увеличивается.

Что касается других классов, то, например, водные объекты SVM распознает достаточно точно и быстро, это связано с тем, что текстура воды достаточно проста, практически без особенностей, поэтому этот вид объектов не так чувствителен к подбору негативной выборки.

Если брать более крупный размер участков, например 25x25, то, разумеется, распознавание более крупных, природных объектов будет более успешным и быстрым, но, например, в городах такой размер участка неприменим, в отсутствие в комплексной системе детектора границ. Это

замечание справедливо и для размера 15x15. На рисунках 3 и 4 приведены примеры распознавания лесных массивов в городе и природной среде.

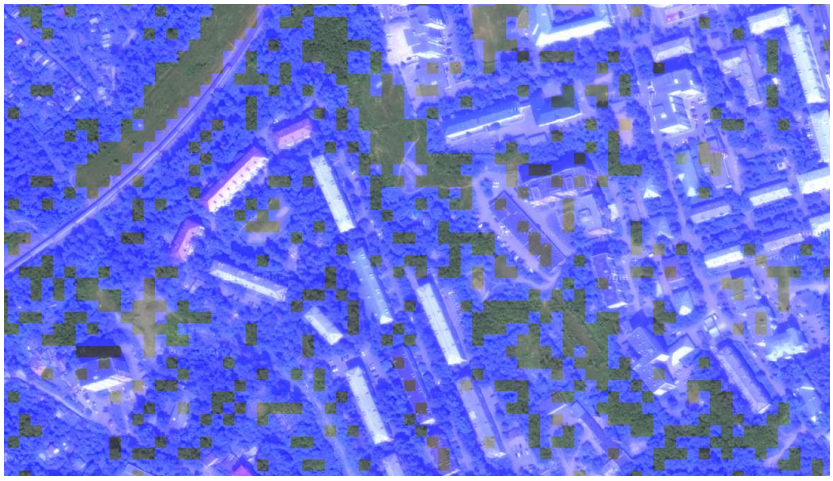

Рис. 3. Распознавание в городе

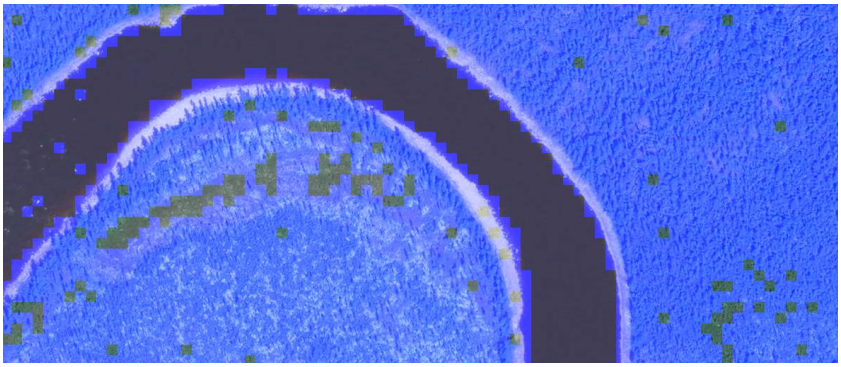

Рис. 4. Распознавание в природной среде

Соответственно, при более мелком (менее 10x10) размере участков, SVM не сможет уловить закономерности текстуры, распознавание будет более хаотическим, зашумленным.

3.4.2 Параметры метода сравнения гистограмм

В методе сравнения гистограмм в основном на качество распознавания влияет количество бинов канала Hue и порог распознавания. Правильный подбор этих параметров может немного улучшить качество и скорость распознавания и сделать метод пригодным в качестве использования в комплексной системе.

Выводы

Исходя из результатов тестирования, можно заключить, что система, основанная на методах сравнения гистограмм, показывает высокое качество распознавания, если выбирать эталон прямо на тестовом изображении, но как самодостаточная система с базой изображений, она не сможет работать достаточно быстро. В то же время, главным ее достоинством является скорость (в случае проверки только одного участка на всем изображении). Это означает, что такую систему можно использовать как дополнительную к одному из более качественных, но относительно медленных классификаторов, который бы распознавал класс с большой степенью уверенности, и передавал участок этого класса методу сравнения гистограмм.

Метод опорных векторов показал лучшее качество распознавания, по сравнению с методом сравнения гистограмм, и может использоваться, как самодостаточная система распознавания, однако, для ее качественной настройки необходимо дополнительно проанализировать влияние обучающих данных на конечный результат.

Работа может быть продолжена исследованиями в области детектирования границ, так как без этого распознавание и классификация мелких объектов (например, в черте города) будет достаточно сложной. Помимо этого, необходимо провести исследования, касающиеся инвариантности систем к масштабу распознаваемых изображений, так как сейчас они способны работать в достаточно ограниченном диапазоне масштабов.

Заключение

В представленной работе было проведено исследование двух методов распознавания изображений, на основе которых были разработаны системы классификации объектов на аэрофотоснимках земной поверхности. Системы были протестированы, а также были выявлены их достоинства и недостатки.
В работе были освещены также некоторые особенности использования методов применительно к предметной области. Помимо этого, был проведен анализ результатов тестирования и предложены варианты использования разработанных систем в качестве самодостаточной системы распознавания, а также направления дальнейших исследований.

Список литературы

1. Шовенгердт Р.А. Дистанционное зондирование. Модели и методы обработки изображений. М.: Техносфера, 2010. 560 с.
2. Шитова О. В., Пухляк А. Н., Дроб Е. М. Анализ методов сегментации текстурных областей изображений в системах обработки изображений // Научные ведомости Белгородского государственного университета. Серия: Экономика. Информатика. 2014. №8-1 (179). URL: http://cyberleninka.ru/article/n/analiz-metodov-segmentatsii-teksturnyh-oblastey-izobrazheniy-v-sistemah-obrabotki-izobrazheniy (дата обращения: 10.05.2016).
3. Мурзин Ф.А., Половинко О.Н., Лобив И.В. Распознавание текстур по пространственным закономерностям // Материалы междунар. конф. Новые информационные технологии в науке и образовании. Новосибирск, 2003. С. 256-268.
4. Воронцов К.В. Машинное обучение. Курс лекций // 2004
5. Хайкин С. Нейронные сети. Полный курс // 2006
6. Bradski G., Kaehler A. Learning OpenCV. Computer Vision with the OpenCV Library // O'Reilly // 2008
7. OpenCV 3.0.0 Documentation. http://docs.opencv.org/3.0.0/
8. Лабутина И.А. Дешифрирование аэрокосмических снимков. М.: АСПЕКТ ПРЕСС, 2004. 184 с.
9. A Practical Guide to Support Vector Classification. http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf (дата обращения: 14.04.2016).
10. OpenCV (C++ vs Python) vs MATLAB for Computer Vision. http://www.learnopencv.com/opencv-c-vs-python-vs-matlab-for-computer-vision/ (дата обращения: 17.02.2016).