

ВЫСШАЯ ШКОЛА МАЙНОР
Институт инфотехнологии
Веб программирование

Александр Мочёнов
IT-3-Q-V-Tal

**Система (?) Метод (?) обнаружения и слежения за положением лица
человека на основе нейронных сетей и сегментации по цвету
кожного покрова (?) по цвету кожи?)**

Дипломная работа

Руководитель: Jelena Faronova, MSc

Таллинн 2010

Оглавление

Резюме	2
Введение	3
1 Введение в предметную область	5
1.1 Компьютерное зрение	5
1.2 Распознавание и обнаружение лиц	7
1.3 Существующие методы обнаружения лиц	7
1.3.1 Методы предварительной подготовки данных	8
1.3.2 Методы представления изображения	11
1.3.3 Задача классификации	13
1.4 Вспомогательные методы обнаружения лиц	16
2 Предлагаемый метод решения	18
2.1 Модуль нахождения лица	19
2.1.1 Предварительная обработка	19
2.1.2 Поиск зон с цветом кожного покрова	25
2.1.3 Выделение и объединение областей с цветом кожного покрова .	29
2.1.4 Классификация	33
2.2 Выбор цели для слежения	41
3 Результаты работы (Испытания?)	43
3.1 Автоконтраст и баланс белого	43
3.2 Поиск зон с кожным покровом	46
3.3 Объединение областей	46
3.4 (Результаты) работа с ИНС	48
3.4.1 (Результаты) обучения и тестирования	48
3.5 Испытание всей системы	51
Заключение и выводы	53
А Приложение. Отчёт по курсовой практике	54
Литература	54

РЕЗЮМЕ

TODO:

ВВЕДЕНИЕ

Роботы в различных вариациях являются частью жизни человека. Робототехника уже давно применяется, например, в индустриальном производстве, в детских игрушках, авиации и многих других местах. Так же роботы применяются в военными (беспилотные самолёты, роботы-сапёры), медицине и даже в космосе¹.

Тем не менее, применение роботов в сфере обслуживания сегодня не так распространено. Оно находится на рубеже науки робототехники и пока ещё широко не применяется. В данной работе автор разрабатывает небольшую часть робота, функционирующего в сфере обслуживания, главной целью которого является общение с человеком.

Цель автора данной работы - создать интерактивную систему слежения за человеческим лицом подобием головы робота, которая оборудована веб-камерами на месте глаз и серво-приводами, способными поворачивать её по двум осям. Вся система состоит из 3 модулей:

- Нахождение местоположения и размеры лиц людей на изображении с веб-камеры
- Выбор лица из найденных, за которым необходимо следовать
- Вычисления вектора движения и само общение с серво-приводами

Самой сложной из задач является поиск лица человека. В работе автор предлагает по-следовательный алгоритм поиска, который состоит из 3 подзадач, где результат предыдущей является источником данных для последующего:

- Предварительная обработка и подготовка изображения;
- Поиск, сегментация и кластеризация участков кадра, в которых высока вероятность обнаружения лица;
- Применение искусственных нейронных сетей для окончательной классификации (лицо или нет) по нескольким представлениям данного изображения;

TODO: Почему именно такой?

TODO: Про real-time сюда?

¹<http://robonaut.jsc.nasa.gov/default.asp>

Подобная система может применяться в любых роботах, обладающих подобием головы. Например: робот-консьерж в отеле, робот-официант или робот-домохозяйка. Это может упростить и улучшить впечатление от общения человека с машиной.

TODO: Содержание глав

1 ВВЕДЕНИЕ В ПРЕДМЕТНУЮ ОБЛАСТЬ

1.1 Компьютерное зрение

Основной частью данной работы является обработка изображений поступаемых с веб-камеры. Трансформация данных с видео камеры или из статичных изображений в новое представление или принимаемое решение называется - *Компьютерным Зрением* (*Computer Vision* или *CV*) (Bradski and Kaehler, 2008) Т.е. программы и алгоритмы, которые в своей работе используют визуальную информацию - всё это компьютерное зрение.

Человеку, в силу своей зрительной природы, может показаться, что обработка визуальной информации - это очень просто. Но эта представление крайне ошибочно. Наш мозг разделяет визуальную информацию на множество каналов, в которых зашифрованы различные виды информации и посылает их мозг человека. В мозгу есть системы распределения внимания, в ходе работы корой, часть информации обрабатывается, а часть остается незамеченной. (Bradski and Kaehler, 2008) Даже сетчатка глаза - внутренняя поверхность глаза, заполненная светочувствительными клетками (колбочками и палочками), отвечает за предварительную обработку сигнала. На поверхности глаза около 130 миллионов светочувствительных элементов, а нервных окончаний идущих к мозгу в 100 раз меньше. Это говорит о том, что сетчатка сжимает информацию. В частности одной из её функций является *обнаружение границ* (*Edge detection*) (RetinaOnWiki)

А что “видит” компьютер? Матрицу из чисел, представляющих собой интенсивность света в разных участках светочувствительной матрицы. При этом каждая ячейка этой матрицы кроме полезной информации содержит ещё и шум. И в этом наборе чисел надо найти машину или, например, идущего человека.

Компьютерное зрение широко применяется в медицине, где оно помогает человеку анализировать визуальные данные. Например по снимку с МРТ¹ указать на опухоль (рис. 1.1а) или анализировать клетки крови (рис. 1.1b). Это задача называется *распознаванием образов* (*Pattern recognition*). Распознавать можно так же и другие объекты и образы. Например, система автоматического замера скорости на дорогах распознаёт автомобили и их регистрационные номера, а система безопасности распознаёт передвижение людей (рис. 1.1d) или запрещённые объекты в багаже (рис. 1.1c).

¹Магнитно-резонансная томография

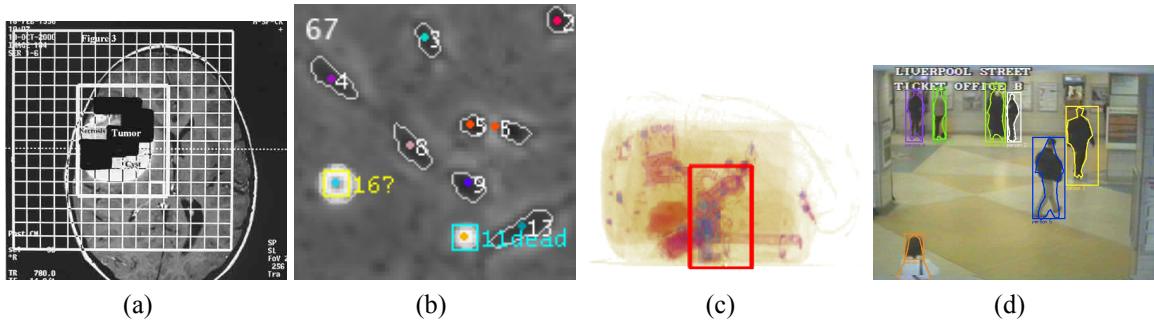


Рис. 1.1: Примеры применения Компьютерного зрения: (а) Обозначенная опухоль на снимке МРТ, (б) Анализ клеток с помощью CV, (с) Определение запрещённых предметов, (д) Слежение за людьми службой безопасности

Распознавание лица (Face recognition) человека является одной из наиболее популярных задач в области компьютерного зрения. Она заключается в обнаружении и определении по изображению лица кому именно оно принадлежит. Решения этой проблемы применяются в системах безопасности (авторизация) и системах управления базами данных лиц людей. С быстрым развитием более развитых методов в этой области, распознавание лица человека, как средство авторизации, представляет из себя более дешёвое решение по сравнению с системами распознавания сетчатки глаза или отпечатка пальца. (Kumar and Bindu, 2006)

В популярной программе для работы с цифровыми фотографиями Picasa (<http://picasa.google.com/>) есть эффективная система распознавания, маркировки и каталогизации найденных на фотографиях лиц людей (рис. 1.2).

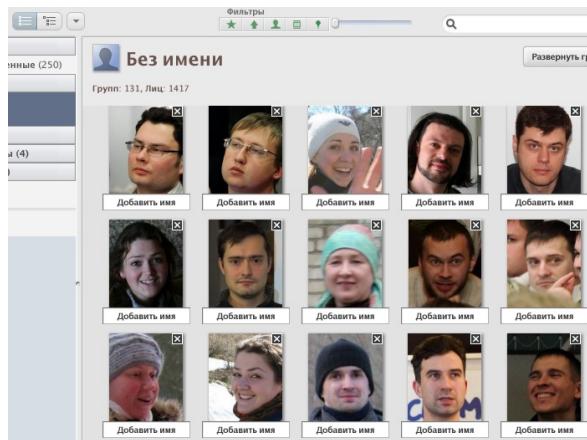


Рис. 1.2: Интерфейс программы Google Picasa с частью списка найденных на фотографиях лиц.

1.2 Распознавание и обнаружение лиц

Но для того, что бы распознать лицо, сначала необходимо найти его место положение и границы на изображении. Эта задача называется *обнаружение лица* (Face detection), что является частным случаем более общей проблемы *обнаружение объекта* (Object detection). Почти все алгоритмы распознавания лиц в качестве входных данных используют изображение, содержащее только лицо, которое надо распознать. По-этому обнаружение лица есть предварительная и очень важная задача, которую надо выполнить, перед распознаванием. Следовательно от точности и быстроты определения местоположения лица зависит эффективность всей задачи по распознаванию.

Но, обнаружение не обязательно должно вести к распознаванию. Обнаружение и слежение за лицом без определения принадлежности его к конкретному человеку является основной задачей как данной работы, так и схожих по своей сути работ: (Capi et al., 2010), (Luo et al., 2007), (Saxena et al., 2008).

Нахождение лица заключается в том, что бы по данному изображению определить, количество, место положения и размеры всех имеющихся лиц. (рис. 1.3) (Liu et al., 2010)

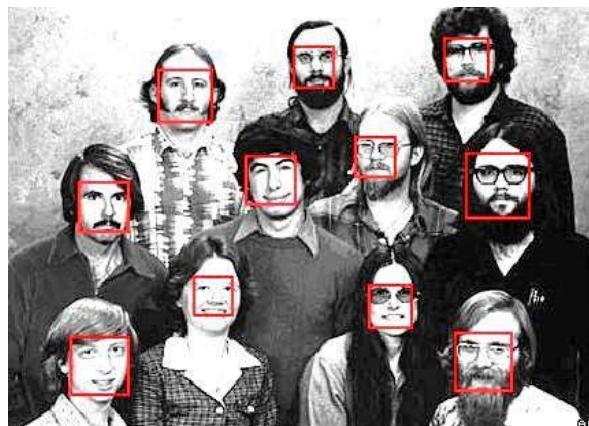


Рис. 1.3: Обнаружение всех лиц на изображении и обозначение их прямоугольником.

1.3 Существующие методы обнаружения лиц

Почти все современные подходы к решению задачи обнаружения лица, по мнению автора, содержат так или иначе 3 составляющих:

Подготовка данных

Для успешного обнаружения лица поступившие данные необходимо подготов-

вить для дальнейшей работы. Сюда входят и различные методы *предварительные обработки* изображения в целом (т.н. *preprocessing*) и методы позволяющие уменьшить область поиска или *область интереса* (*Region of interest* или *ROI*) для ускорения всего процесса обнаружения.

Представление данных

Изначальное изображение в виде матрицы интенсивностей светочувствительного элемента камеры часто трансформируют в иные, более компактные отображения или *представления* (*Representation*). Такие изменения чаще всего ведут к потере информации, но облегчают процесс классификации.

Классификация

Само определение наличия или отсутствия в данном участке картинки лица человека. Методов классификации в принципе (не только лиц) на сегодняшний день существует огромное количество и все они применимы для данной задачи.

1.3.1 Методы предварительной подготовки данных

Коррекция цвета и освещённости

Для эффективной работы с изображением его необходимо подготовить.(О проблеме освещения в принципе)

Некоторые механизмы применяются для всего изображения в целом. Например, коррекция контраста или интенсивности и баланса белого (рис 1.4а). Другие подготовительные процессы касаются изображений, которые поступают непосредственно в классификатор. Речь идёт о небольших частях изображения, которые получаются методов скользящего окна (*sliding window*) и маштабирования, которые называют *образцами* (об этом подробнее в (TODO)). Процессы коррекции контраста и баланса белого часто накладывают и на образцы (рис 1.4б), т.к. зачастую после исправления контрастности всего изображения, распределение интенсивности в образце всё ещё остаётся узким.

В работе (Rowley et al., 1998), в частности, применяется два метода предварительной обработки образцов. Во-первых производится нелинейное *уровновешивание гистограммы* (*Histogram equalization*), которое “растягивает” гистограмму, что компенсирует недостающие уровни интенсивности изображения, в результате чего изображение становится более контрастным. Во-вторых производиться линейная компенсация интенсивности. Для этого линейная функция аппроксимирует общую яркость в каждой из частей образца, после чего она может быть вычтена из образца в случае

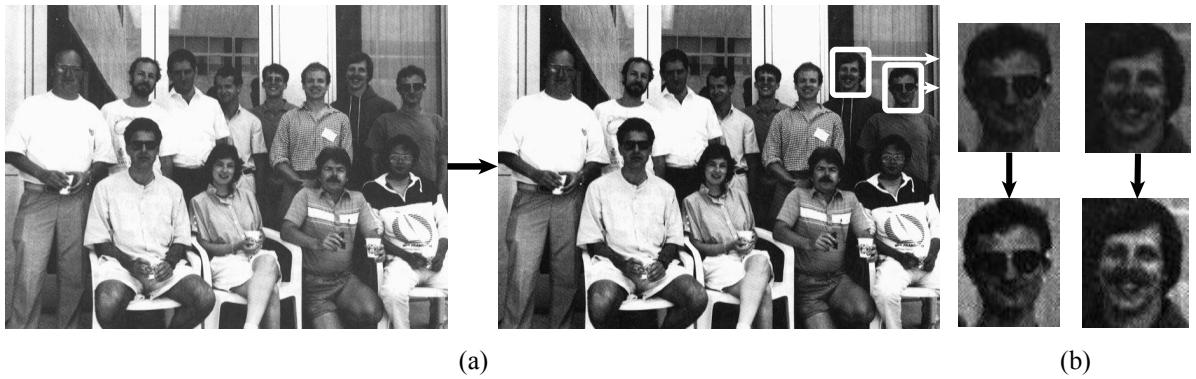


Рис. 1.4: Нормализация всего изображения (а) и дополнительная нормализация образца (б)

сильной разницы в освещённости разных частей образца. Это приводит к уравновешиванию контрастности в разных частях образца (рис. 1.5)

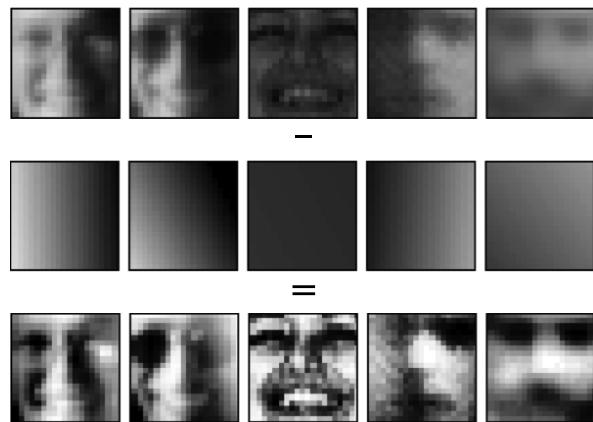


Рис. 1.5: Метод линейной коррекции контрастности. На рисунке сверху оригинальное изображение, ниже результат измерения суммарной интенсивности, ниже результат вычитания. (Rowley et al., 1998)

Похожий алгоритм нормализации интенсивности применяется и в работе (Lin et al., 2005). Перед классификацией каждый образец нормализуется по формулам 1.1 и 1.2.

$$\bar{I} = \frac{1}{N} \sum_{i=1}^N I_i \quad (1.1)$$

$$I'_i = (I_i - \bar{I}) + 128 \quad (1.2)$$

где

N – количество пикселей в образце

\bar{I} – средняя интенсивность по всем пикселям образца

I_i – интенсивность i ого пикселя образца

I'_i – нормализованная интенсивность

Таким образом если какие-то образцы были слишком тёмными или слишком светлыми

они все становятся единообразно освещены, что упрощает процесс машинного обучения. Пример показан на рисунке 1.6.



Рис. 1.6: Приминение формул 1.1 и 1.2.

Как видно, после преобразования образцы стали однообразны, но при этом в большинстве случаев теряется много полезной информации. В данной работе применяются схожие методы предварительной обработки образцов и изображения в целом. Подробнее в разделе(TODO)

Сегментация по цвету кожного покрова

Чем меньше изображение необходимо сканировать на наличие лица, тем меньше процессорного времени необходимо затрачивать, что ведёт к ускорению процесса обнаружение. Это можно достигнуть, например, уменьшением изображения, что приведёт к потере разрешающей способности данного подхода. Одним из способов позволяющих уменьшить площадь сканирования и одновременно не потерять разрешающей способности является эвристическое знание о том, что все лица людей покрыты кожным покровом. Это можно использовать для нахождения областей изображения содержащих цвета схожие с цветами кожного покрова (*skin-color detection*), и в дальнейшем осуществлять сканирование только этих областей.

Среди методов обнаружения лиц основанных на поиске признаков (*feature based*) подходы использующие информацию о цвете кожи, как признак обнаружения, получают всё большую популярность. Цвет легко и быстро обрабатывать и он инвариантен к геометрическим особенностям образов лиц людей. К тому же опыт подсказывает, что цвет кожи человека имеет отчётливый характерный цвет, который легко узнаваем людьми. (Vezhnevets et al., 2003) Цвет кожи - это один из тех признаков, что не зависит от положения лица, частичной закрытости и контраста, по-этому именно этот метод часто используют для локализации лиц. (Ruangyam and Covavisaruch, 2009)

Такой метод возможен благодаря тому факту, что различные цветовые вариации кожных покровов людей (даже среди представителей различных этнических групп) лежат

в достаточно узком диапазоне и отличаются только яркостью (Luminance), в тоже время цветность (chrominance) практически не меняется. Цвет кожи лежит в основном в красной части цветового спектра и определяется цветом крови. А яркость определяется прозрачностью эпидермиса (верхнего слоя кожи), за что в свою очередь отвечает концентрация меланина. (Xu and Zhu, 2006) Подробнее в разделе ... (TODO)

1.3.2 Методы представления изображения

Одно и тоже изображение можно представить разными способами. Это нужно для того, что бы обучаемые классификаторы (см. 1.3.3) могли обучаться на различных характеристиках или признаках, имеющихся в различных представлениях.

Пиксельная интенсивность

Самым простым и популярным представлением является информация об интенсивности в каждом пикселе изображения. Такой метод представления считается не имеющим потерь (*lossless*). Т.е. в нём присутствует вся информация об оригинальном изображении.(Bojkovic and Samcovic) Пример на рисунке 1.7a.

Информация о контурах

Любой метод обнаружения лица (да как и все другие алгоритмы в компьютерном зрении) должен быть стойким к таким вещам как поза головы, угол обзора, освещение и многим другим факторам.

Для решения проблемы с освещением используется *информация о контурах* (*edges*) или *градиент изображения*. Одно и то же лицо под разными источниками света с точки зрения пиксельного представления совершенно отличны друг от друга, что делает проблемой для классификатора найти среди них нечто общее. С другой стороны информация о пограничных областях в большей степени неизменна. (Ahmadyfard et al., 2008)

Это представление, в отличия от пиксельного представления, не содержит всей изначальной информации. Сохраняется информация лишь о пограничных контрастных зонах лица (глаза, брови, губы). Пример такого представления на рисунке 1.7b.

Собственные лица

Собственное лицо (*eigenface*) - это набор *собственных векторов* (*eigen vectors*), используемых для описания “стандартезированных компонентов лица”. Один образец лица принимается как точка в многомерном пространстве. Много образцов образуют некую область в этом пространстве. Задача заключается в нахождении *главных компонент* (*principal components*) этой области, параметрами которых её можно описать используя значительно меньшее количество переменных, нежели для описания всех точек изначального пространства. (Turk and Pentland, 1991)

Собственный вектор представляет из себя набор параметров, которыми можно описать лицо человека. Их же можно хранить например в базе данных, или использовать в качестве представления группы изображений. Этот набор параметров является своего рода выжимкой из многих тренировочных лиц и может использоваться для дальнейшей классификации. Эти параметры можно спроектировать в виде изображения. Пример собственного лица приведёт на рисунке 1.7с.

В работе (Tsai et al., 2006) автор для обнаружения лиц использует собственные лица для поиска кандидатов областей с лицами, а нейронная сеть проводит конечную валидацию.

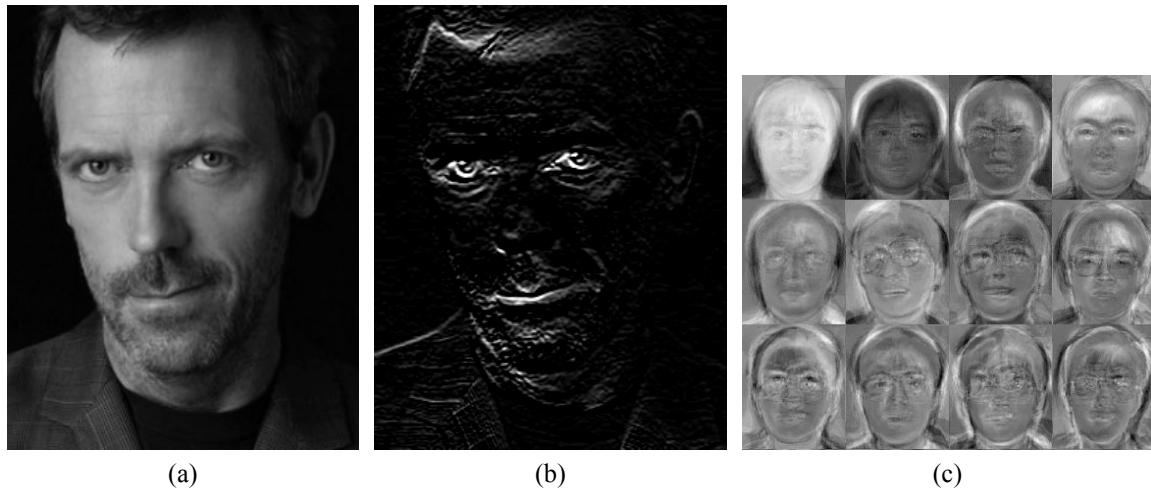


Рис. 1.7: Различные представления изображений: (a) обычное пиксельное представление, (b) нормализованный Собелев градиент по y , (c) набор из различных собственных лиц

Характеристики типа Хаара

Характеристики типа Хаара (*Haar-like features*) представляют изображения в виде набора характеристик полученных вычитанием сум интенсивностей одних областей

из других по определённому шаблону. Это представление применяется при использовании платформы по обнаружению объектов Виола-Джонса (*Viola-Jones object detection framework*). (Viola and Jones, 2001) Эта метод и применяемое в нём представление является одним из самых быстрых на сегодняшний день методов обнаружения объектов.

Использование нескольких представлений

Многие решения обнаружения лица используют сразу несколько представлений. Это позволяет использовать более широкий спектр характеристик, что делает классификатор более чувствительным к особенностям конкретного вида объектам (в данном случае к лицам).

Так в работе (Bojkovic and Samcovic) используются три представления: пиксельное, коэффициенты собственных лиц и профильные коэффициенты. В работе (Ahmadyfard et al., 2008) авторы используют две представления: пиксельное и информацию о контурах. По их словам “объединение информации об интенсивности и о контурах даёт более описательные характеристики для представления изображения с лицом”.

1.3.3 Задача классификации

Задача *классификации* - это проблема определение класса из всех возможных, к которому относятся классифицируемые объекты или наблюдения. Классификация тесно связана с *машинным обучением* (Machine Learning или ML), задачей которого является превращение данных в информацию (Bradski and Kaehler, 2008)

Задача обнаружения лица является классическим примером задачи классификации, где есть всего два класса “лицо” или “не лицо”. Классификатор должен по данному ему изображению уметь определить к какому классу из двух оно относиться.

Все методы можно грубо разделить на две части: классификаторы основанные на знаниях (*knowledge-based*) и статистически обучаемые. Основанные на знаниях используют эмпирические знания о местоположении и распределении органических характеристик лица. (Bojkovic and Samcovic) Например, факт, что глаза находятся в верхней половине картинки и они симметричны, а под ними находится рот, с которым они образуют равнобедренный треугольник. Такие алгоритмы легко реализовать, но они неустойчивы ко многим факторам (например наклон головы).

К статистически обучаемым алгоритмам классификации относятся те, что получают знания из данных при обучении с учителем или без него. Т.е. работа таких алгоритмов состоит из двух этапов: обучение и само использование. Самых методов очень много. Ниже приведены те, что чаще всего встречаются в обнаружении лиц.

Метод опорных векторов

Метод опорных векторов (Support vector machine или SVM) заключается в нахождении разделяющей гиперплоскости в более высокомерном пространстве, чем классифицируемые вектора. Конечная плоскость должна быть максимально удалено от представителей разных классов. Для этого находятся наиболее близкие к этой плоскости и тем самым более влиятельные векторы, которые называют опорными векторами. (Shavers et al., 2006)

Ограничением SVM подхода является проблематичность использования для классификации более чем по двум классам. Впрочем в данной задаче речь как раз и идёт о двух классах и SVM хорошо справляется с ней во многих работах. (Shavers et al., 2006), (Jee et al., 2004), (Saxena et al., 2008)

Искусственные нейронные сети

Одним из самых популярных (но не самым эффективным) методом классификации до сих пор является искусственные нейронные сети (далее ИНС, Artificial Neural Network или ANN). Этот способ применяется в большинстве научных работ изученных в ходе подготовки данной работы, что стало одним из факторов выбора её в качестве классификатора.

Появление ИНС было вдохновлено биологическими системами, в частности нервной системой. До определённой степени структура и функциональность ИНС напоминают структуры нейронных связей головного мозга (рис у8). Биологические нейроны это - нервная клетка, у которой есть сеть окончаний *дендритов*, через которые она получает сигнал от других нейронов. В ядре клетки происходит суммирование всех сигналов, после чего он передаётся по длинному выходному окончанию *аксону* на входы следующих нейронов. В ИНС всё похоже. У нейрона есть входной вектор (сеть дендритов), все значения которых умножаются на вес соответствующей связи и суммируются, после чего суммарный сигнал проходит через *активационную функцию* (ядро) и результат передаётся к следующему нейрону как один из компонентов его входного вектора.

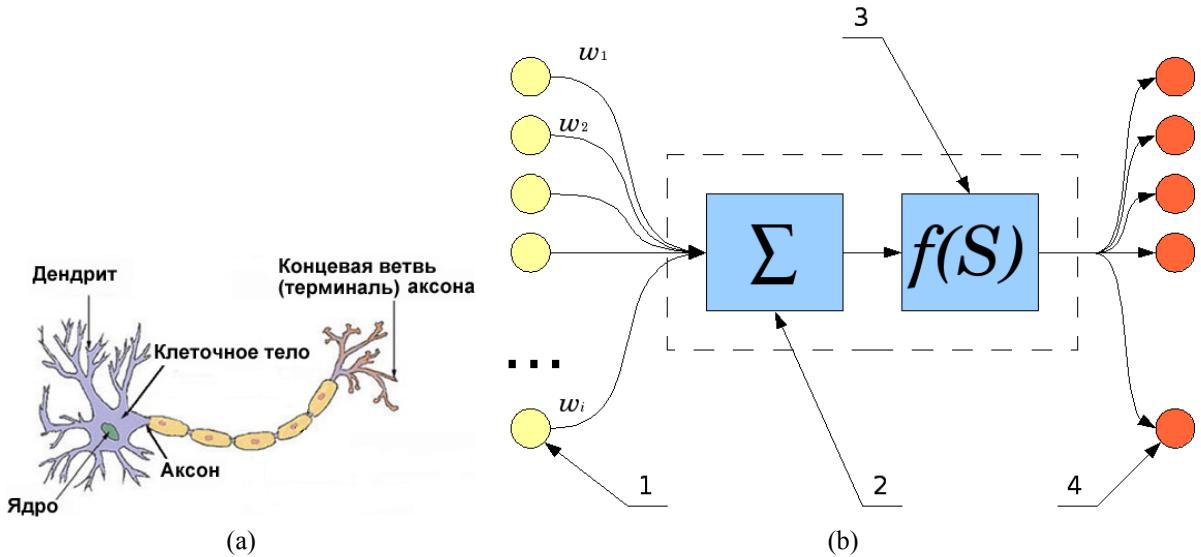


Рис. 1.8: Биологический (а) и искусственный (б) нейроны. На рисунке (б): $w_{1..i}$ - веса входящих соединений, 1 - входящий сигнал, 2 - суммирует входящие взвешенные сигналы, 3 - активационная функция преобразует сумму сигналов, 4 - результат передаётся на входы других нейронов.

Определяющими факторами работы ИНС являются: топология или структура сети, виды активационных функций и веса соединений. Веса соединений - это параметры сети, они меняются во время обучения сети, после чего сеть может описывать модель, которую её учат классифицировать. Главной задачей активационной функции является сжатие сигнала до определённых границ, а так же усиление слабых сигналов и гашение сильных. Самой популярной функцией является сигмойдная (TODO картинка). Она обладает всеми описанными выше требованиями к активационной функции и к тому же обладает простой производной, что очень удобно во время обучения методом обратного распространения. (Уоссерман, 1992)

Концептуально есть две структуры ИНС: с прямым распространением сигнала (*feedforward*) и рекурентные (т.е. с обратными связями). Второй тип применяется для обучения без учителя (*unsupervised learning*) и в обнаружении используется реже, чем первый тип, к которому применяют метод обучения с учителем (*supervised learning*).

Популярным методом обучения ИНС с прямым распространением является алгоритм *обратного распространения ошибки* (*backpropagation*) “Он имеет солидное математическое обоснование. Несмотря на некоторые ограничения, процедура обратного распространения сильно расширила область проблем, в которых могут быть использованы искусственные нейронные сети, и убедительно продемонстрировала свою мощь.” (Уоссерман, 1992) Принцип метода такой:

- сначала на вход сети подаётся входной вектор, который проходит через сеть и

- сеть выдаёт результат;
- разница полученного результата и заранее известного необходимого выхода (“правильного ответа”) сети посыпается в обратном направлении от выходов к входам;
 - по мере прохождения высчитывается ошибка по методу наименьших квадратов;
 - по этим ошибкам методом *градиентного спуска* высчитываются изменения весов соединений. Таким образом с каждым новым образцом веса меняются всё лучше описывая желаемую модель.

У ИНС метода есть и проблемы. Что бы получить результат выше среднего сеть должна быть очень точно отрегулирована (число слоёв, число узлов, скорость обучения и т.д.) (Jee et al., 2004) Авторы работы (Capi et al., 2010) так же считают, что эффективность работы зависит от качества подстроки всей системы, и что это трудоёмкая работа. Результат данной работы лишний раз подтверждает эти выводы.

1.4 Вспомогательные методы обнаружения лиц

На тему обнаружения лиц написано огромное количество научных работ. В базе данных IEEE Xplore за период с 2000 года можно найти свыше 6000 работ, так или иначе связанных с данной тематикой. В каждой из них есть какая-то своя, оригинальная идея.

Так в работе (Jee et al., 2004) одной из главных характеристик поиска являются глаза, которые обнаруживаются на изображении при помощи SVM классификатора.

В работе (Capi et al., 2010) решается проблема создания навигационной системы робота, способной следовать за человеком, используя визуальную информацию. Для этого сначала обнаруживается лицо человека, а после определяется основной цвет его одежды. Далее человек может развернуться, и робот будет следовать за ним ориентируясь по одежде, а не по лицу.

Похожие задачи стоят в работе (Luo et al., 2007), где решают проблему взаимодействия человека и робота (Human-Robot Interaction или HRI). Стоит задача обнаружения лица человека и дальнейшего слежения за ним. Особенностью этой работы является способ уменьшения области интереса, в которой высока вероятность обнаружения лица. Для этого информация о положении лица берётся из предыдущих кадров, высчитывается траектория движения лица и прогнозируемое следующее положение, после чего эта область расширяется на константу и в полученном месте изображения уже производится поиск лица.

В работе (Zhang and Liang, 2010) применяется простая идея о том, что лицо человека, как и сама голова, скорее всего в кадре будет двигаться. Так областью интереса можно

определить все движущиеся объекты. Но, такой метод предполагает статичную камеру и следовательно статичный фон.

(про real-time)

2 ПРЕДЛАГАЕМЫЙ МЕТОД РЕШЕНИЯ

В этой работе предлагается метод состоящий из 3 модулей, которые в совокупности образуют классический *интеллектуальный агент* (см. 2.1).

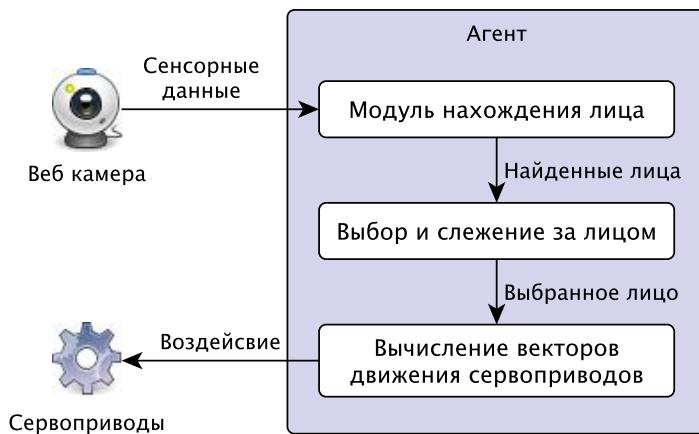


Рис. 2.1: Высокоуровневая структура всей системы.

Агент обладает способностью воспринимать окружающую среду через сенсоры, функцию которых выполняет веб-камера. После обработки полученных данных агент в каком-то роде может воздействовать на среду, вращая сервоприводы. (Рассел and Норвиг, 2006)

Информация с вебкамеры поступает в модуль нахождения лиц, от куда данные о найденных лицах поступают в модуль выбора и слежения. Тут происходит выбор того лица за которым следить. После выбора модуль работы с сервоприводами даёт команду на вращение. Такая модульная структура позволяет в случае необходимости заменить тот или иной модуль.

В предложенном решении основной программный код написан на языке Python. Что бы не тратить время на решение уже решённые кем-то проблемы, используются следующие программные библиотеки:

OpenCV

OpenCV¹ - самая популярная и используемая библиотека компьютерного зрения с открытым исходным. Библиотека написана на C\ C++ и работает под Linux, Windows и Mac OS X. Так же разрабатываются интерфейсы для языков Python ,

¹Open Source Computer Vision Library. <http://opencv.willowgarage.com/>

Matlab, Ruby, Lua и других. “Цель OpenCV - предоставить простую в использовании инфраструктуру компьютерного зрения, которая поможет людям строить достаточно сложные приложения.” (Bradski and Kaehler, 2008)

В данной работе используется очень интенсивно для всех манипуляций с изображением.

PyBrain

PyBrain² - модульная Python библиотека машинного обучения с открытыми исходным кодом. Имеет широкий набор алгоритмов для обучения с учителем, без учителя, обучения с подкреплением, оптимизации типа “чёрный-ящик” (black-box optimization).

В данной работе библиотека используется для конструирования, обучения и применения ИНС.

NetworkX

NetworkX³ - Python библиотека для работы с графами. В данной работе нужна для поиска минимального остовного дерева.

2.1 Модуль нахождения лица

Модуль нахождения лица является самой большую часть данной работы, т.е. в нём сосредоточена основная логика всей системы. Схематически весь алгоритм представлен на рис. 2.2. Каждый подмодуль соответствуетциальному алгоритму и он будет рассмотрен в данном разделе более подробно. (TODO Подробнее описать что да как?)

2.1.1 Предварительная обработка

В данной работе как для подготовки изображения для дальнейшей работы используется метод *нормализации гистограмм* каждого из каналов изображения. Необходимость данного этапа приведена в разделе 1.3.1. Цель данного этапа получить более контрастную картинку с минимальной потерей информации и исправление баланса белого.

²Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network Library. <http://pybrain.org/>

³<http://networkx.lanl.gov/>



Рис. 2.2: Алгоритм обнаружения лиц

Нормализации гистограмм

Гистограмма - это графическое представление, дающее наглядное представление о распределении данных. График состоит из прямоугольников, которые показывают количество наблюдений (ось ординат) на данном промежутке (ось абсцисс). На рисунке 2.3 показана гистограмма распределения пиксельных интенсивностей данного (2.3б) Ч\Б изображения. Стоит отметить, что в цветном изображении у каждого цветового канала своя гистограмма.

В данном примере по оси абсцисс промежутками являются промежутки возможных интенсивностей. На гистограмме справа (а) один промежуток соответствует одному конкретному значению интенсивностей. Для 8-и битного Ч\Б изображения значение может быть от 0 до 255, где 0 - это чёрный, а 255 - белый цвета. На гистограмме слева

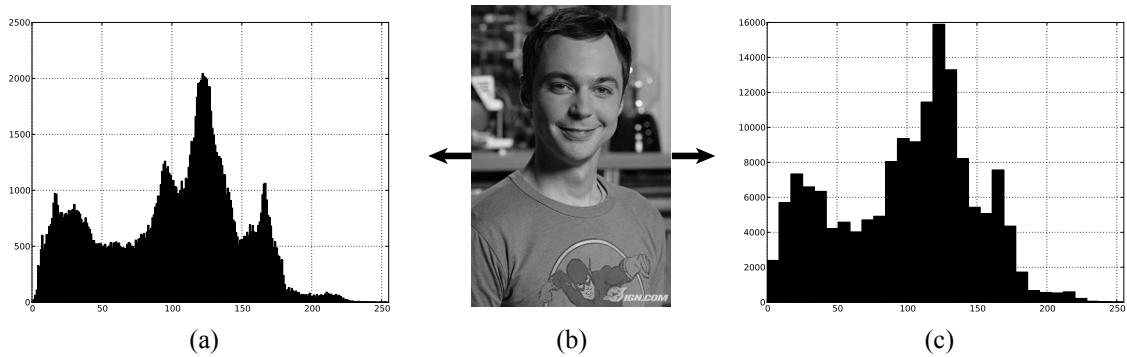


Рис. 2.3

(с) показано, что “столбики” (*bins*) могут объединять значения на равных интервалах , показывая тем самым менее детальное распределение между равными частями всех значений.

В случае изображений с низкой контрастностью рис. 2.4а гистограмма “сжата” и все значения сосредоточены на узком участке возможных интенсивностей рис. 2.4с. Как видно с гистограммы изображение содержит только пиксели средние интенсивности, т.е. серые цвета. По-этому на нём нету ни белого, ни чёрного цветов.

Что бы исправить положение и сделать изображение более интенсивным необходимо “рас2.4д. Как видно после такого растяжения изображение стало более контрастным, на нём появились более тёмные и светлый цвета 2.4б. На модифицированной гистограмме видны пробелы, они свидетельствуют о недостаточной информации и скачкообразных переходах интенсивностей изображения.

Баланс белого цвета

Баланс белого цвета (реже *цветовой баланс* от *color balance*) - это показатель нейтральности основных цветов (красный, зелёный и голубой) на изображении. Бывают ситуации, когда эта нейтральность нарушена и на изображении превалирует какой-то из цветов. Про такое изображение говорят, что баланс белого нарушен и необходимо совершил *цветокоррекцию*.

Глаз и мозг человека способны автоматически совершать цветокоррекцию, по-этому при любом освещении белый лист будет казаться белым. Но если тот же лист сфотографировать с неправильно выставленным балансом белого, тогда на изображении лист будет не белый, например желтоватый (если источник света - лампа накаливания).

На рисунке ?? видно, что на всём изображении превалирует красноватый цвет. Это

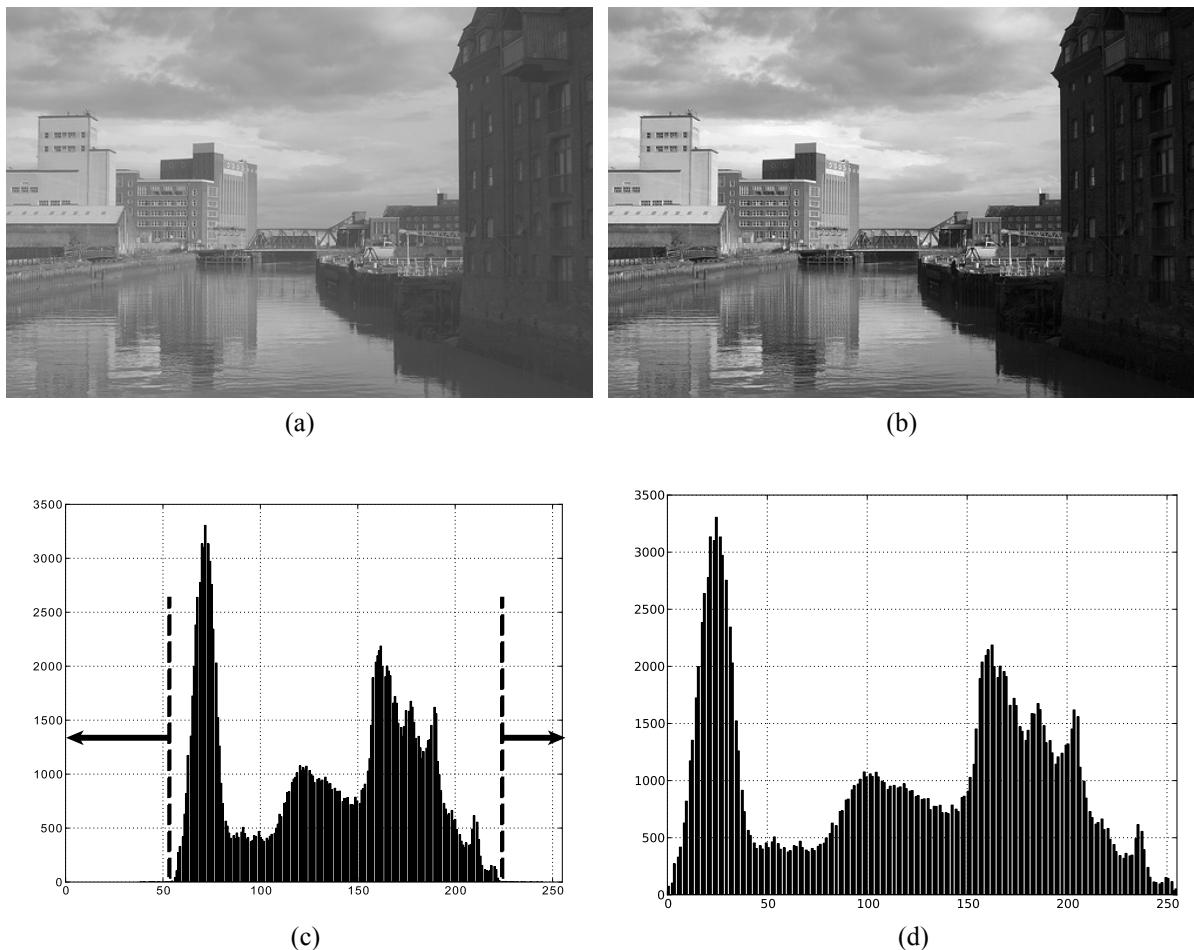


Рис. 2.4: Нормализация контраста “расстяжением” гистограммы.

говорит о том, что все цвета не совсем соответствуют своему истинному значению. Авто не нашёл в источниках информации о том, как производить цветокоррекцию. Но, наблюдая за тем, как меняются гистограммы (цветовых каналов) изображения после цветокоррекции в профессиональных фото-редакторах, была замечена закономерность. На рисунке 2.5с видно, что гистограммы не нормализованы (при чём по разному), а те же гистограммы 2.5д, но уже изображения с правильным балансом белого 2.5б - нормализованы. От сюда предположение, что для того чтобы поправить баланс белого нужно нормализовать гистограммы всех каналов по отдельности, что исправит и контрастность тоже.

В OpenCV уже есть функция, которая делает такое “расстяжение” гистограммы для всех интенсивностей изображения - `Normalize`. С помощью неё можно в любой матрице получить распределение в указанных пределах пропорционально тем значениям, что в ней уже есть. Для получения желаемого результата, нужно в качестве аргумента подать матрицу интенсивностей пикселей изображения, а в качестве границ 0 и 255.

Но иногда этого не достаточно. Бывают случаи, когда изображение мало-контрастно

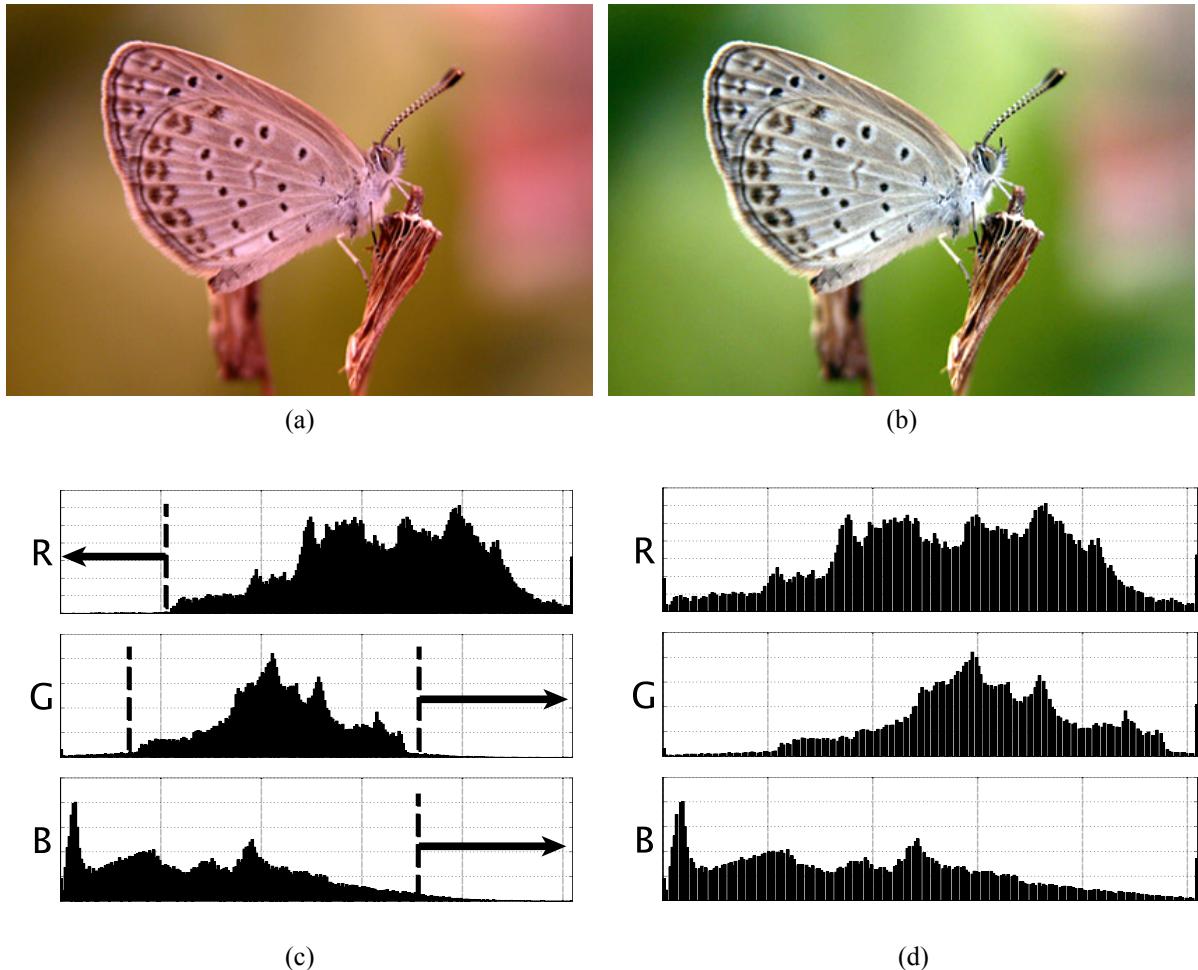


Рис. 2.5: Нормализация баланса белого “расстяжением” гистограммы разных каналов по отдельности.

на большей своей части, при этом имеются небольшие участки с сильно тёмными или светлыми областями. В таком случае Normalize не будет работать так как нужно. На рисунке 2.4 показан случай, когда метод работает, без дополнительных манипуляций. А на рисунке 2.6а видно, что большая часть интенсивностей так и осталась в среднем, “сером” промежутке.

Во время работы процедуры Normalize информация не теряется, а просто перераспределяется, т.е. это обратимый процесс. Что бы в случае 2.3а на картинке 2.3б “растянуть” зону до пунктирной линии 2.6б (верхняя гистограмма), т.е. сделать контрастным большую часть изображения 2.6б (нижняя гистограмма), необходимо лишиться некоторой информации. Следующий алгоритм находит верхнюю и нижнюю границы, область между которыми нужно “растянуть”:

- 1: $mostPopColor \leftarrow$ Самая распространённая интенсивность (цвет)
- 2: $mostPopColorCount \leftarrow histValueAt(mostPopColor)$
- 3: $threshold \leftarrow mostPopColorCount \times aggression$

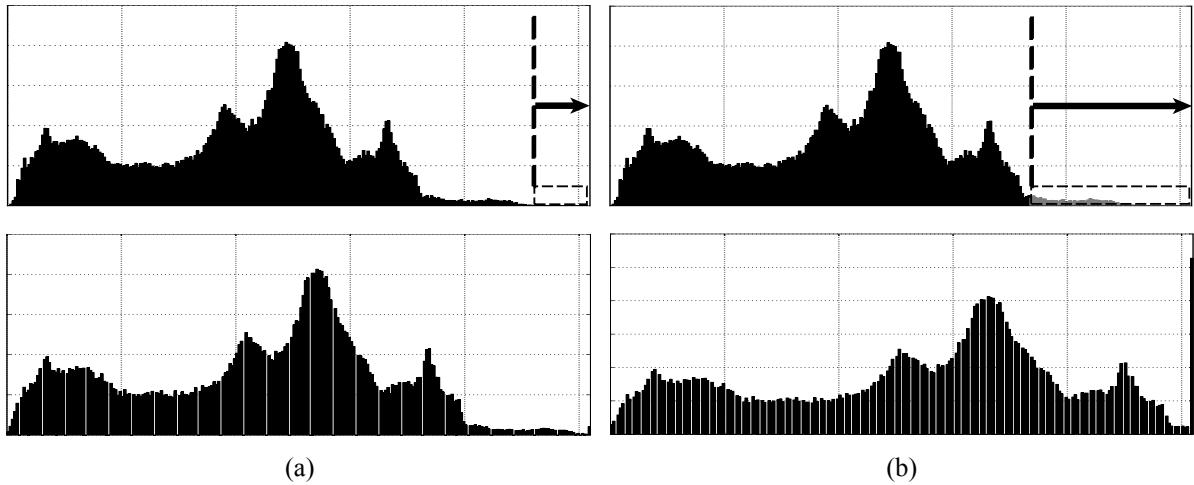


Рис. 2.6

```

4: for  $i = 0$  to 255 do
5:    $downV \leftarrow histValueAt(i)$ 
6:    $upV \leftarrow histValueAt(255 - i)$ 
7:   if  $downT \neq NIL$  and  $downV > threshold$  then
8:      $downT \leftarrow downValue$ 
9:   end if
10:  if  $upT \neq NIL$  and  $upV < threshold$  then
11:     $upT \leftarrow upV$ 
12:  end if
13: end for
14: for  $p \in pixels$  do
15:   if  $p < downT$  then
16:      $p \leftarrow downT$ 
17:   else if  $p > upT$  then
18:      $p \leftarrow upT$ 
19:   end if
20: end for

```

Здесь процедура $histValueAt(color)$ возвращает количество пикселей с данной интенсивностью (т.е. высоту “столбика” на гистограмме), а $aggression$ - константа, показывающая на сколько сильно будет “растянута” гистограмма. После применения данной процедуры все пиксели с интенсивностью до $downT$ и выше upT преобретают значения этих границ. Это проиллюстрировано на рисунке 2.7 ((a) и (b)).

После такой обработки можно уже применять процедуру `Normalize` (рис. 2.7c), которая пропорционально распределит все интенсивность от 0 до 255.

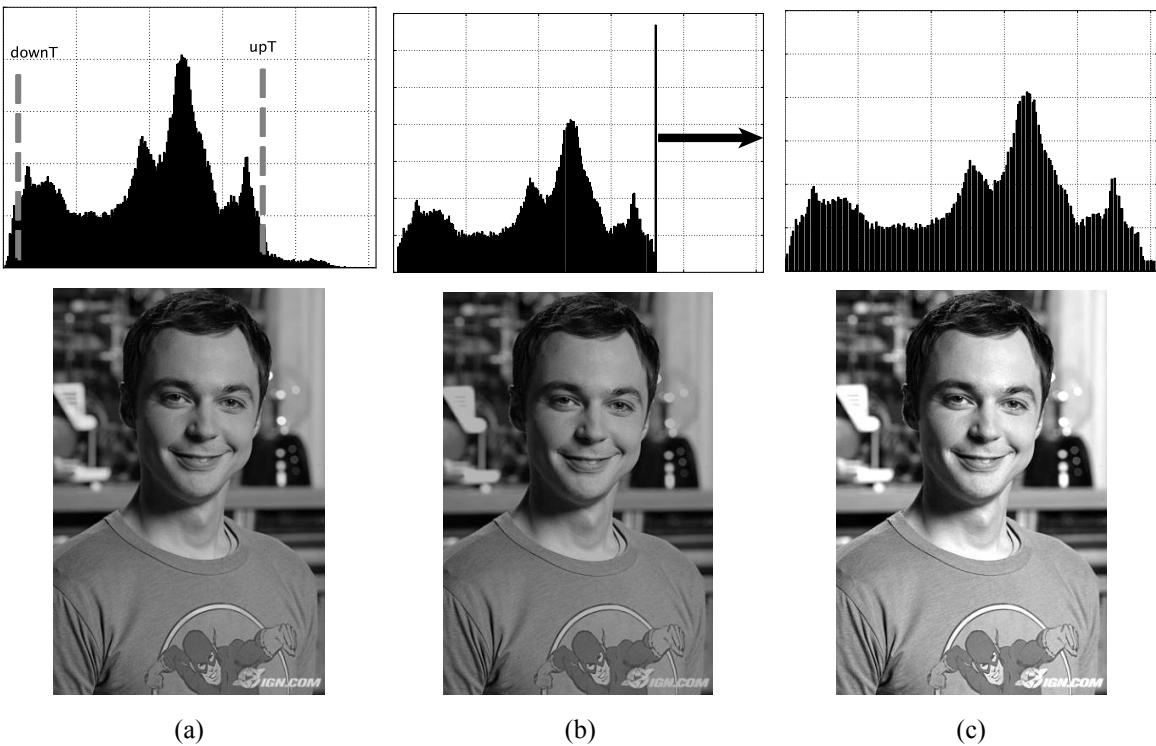


Рис. 2.7

2.1.2 Поиск зон с цветом кожного покрова

Как было отмечено в 1.3.1 для ускорения работы алгоритма обнаружения можно уменьшить область сканирования путём поиска частей изображения с цветом кожного покрова. Но это не единственное применение. Например, для того что бы оградить детей от изображений “только для взрослых”, применяют алгоритмы поиска областей с цветом кожи, т.к. подобные изображения часто имеют большую площадь этих областей. (Forsyth and Fleck, 1999) (Zheng et al., 2004)

Возможные пути решения

Задачу поиска области с цветом кожи как и задачу обнаружения лица можно представить как проблему классификации с двумя классами: “кожа” и “не кожа”. Следовательно как и классификацией лиц тут есть два основных направления: метод основанный на знании и статистические обучаемые, которые в свою очередь делятся на параметрические и непараметрические. (Vezhnevets et al., 2003)

Все обучаемые алгоритмы строят модель цвета кожи на основе тренировочных образцов. Параметрическая модель представляет собой набор параметров, описывающих некоторую область в цветовом пространстве, зачастую используя только плоскости цвет-

ности этих пространств. Плюсом этих методов является компактность полученной модели. Примерами таких алгоритмов являются нормальные распределения и различные эллиптические и модели.

Непараметрические методы строят модель на основе статистических распределений в тренеровых примерах. В результате получаются т.н. карта распределения цвета кожи (*Skin Probability Map*), которая определяет вероятность принадлежности каждого из возможных цветов к классу "кожа" или более формально $P(skin|c)$ - вероятность наблюдения кожи (*skin*), если дан цвет c . Примерами таких алгоритмов являются *Наивный байесовский классификатор*, *Самоорганизующаяся карта Кохонена* (*Self-organizing map* или *SOM*) и ИНС (в (Xu and Zhu, 2006) последний два называют полупараметрическими, т.к. параметры там всё же есть, но их число неопределено)

К основанным на знание относятся два метода: с статическим и динамическим определением границ. В обоих случаях для того, что бы классифицировать определённый цвет он должен входить в определённые рамки в цветовом пространстве (в трёхмерном цветовом пространстве - это ограничивающие плоскости). Конкретные границы определяются эмпирически. В методах со статическим определением они задаются изначально и не меняются, а в динамических границы могут адаптировать к текущему цвету лица или условиям освещённости.

Проблема выбора цветового пространства

Параллельно с выбором метода обнаружения кожи необходимо выбрать цветовое пространство в котором будет оперировать выбранный метод. Выбор цветового пространства можно считать важным шагом на пути классификации цвета кожи. Самым распространённым является RGB, компонентами которого являются интенсивности трёх основных цветов (красного, зелёного и синего). Координаты цвета в другом пространстве можно получить линейную или нелинейную трансформацию из RGB. Kakumanu et al. (2007)

RGB пространство не является удобным для поиска цветов кожи, т.к. само понятие "цвет кожи" - это не физическое свойство объекта, а скорее явление связанное с восприятие человеком, по-этому и хороших результатов в обнаружении можно достигнуть только использовав цветовые пространства обладающие схожими свойствами. К таким пространствам относится нормализованное RGB пространство, где сумма всех трёх компонент равны единице ($r + g + b = 1$) Это уменьшает зависимость от света. (Vezhnevets et al., 2003)

Другой популярной группой пространств являются HSV, HSI и HSL. Они описывают

ют цвет интуитивными понятиями, основаны на художественных представлениях о краске (*Tint*), насыщенности (*saturation*) и тон. Так *H* (*Hue*) показывает доминирующий цвет (красный, зелёный, жёлтый, фиалетовы и т.д.), *S* (*Saturation*) измеряет насыщенность краски по отношению пропорционально её освещённости. Последняя компонента *V*, *I* и *L* относяться к величине яркости. Таким образом *HSV* чётко отделяет цветность от яркости, что полезно в алгоритмах обнаружения кожи. (Vezhnevets et al., 2003)

Ещё одним очень популярным цветовым пространством является YC_rC_b . Цвет в этом пространстве представлен яркостной составляющей *Y* (*luma*), и двумя цветовыми составляющими C_r и C_b , показывающих разницу соответствующих компонент *RGB* от *Y*. Это цветовое пространство как и *HSV* разделяет цвет на цветность и яркость. (Vezhnevets et al., 2003)

Важным вопросом при выборе цветового пространства является его эффективность для конкретной задачи обнаружения кожи человека. В ряде работ приводятся доводы в пользу того или иного пространства, в других же эти выводы подвергаются сомнению и говориться, что от выбора цветового пространства эффективность сильно не зависит. Так же часто поднимается вопрос о необходимости вовлечении яркостной компоненты ($V \setminus I \setminus L$ для *HSV* и *Y* для YC_rC_b) в классификацию, ведь как говорилось выше (1.3.1) цвет кожи в основном меняется именно в нём, и было бы логичным отбросить этот компонент, что бы алгоритм стал более стабилен. Так поступают во многих изученных работах, например в (Mohamed et al., 2008). Но, в работе (Xu and Zhu, 2006) приводится сравнительный анализ использования различных цветовых пространств с и без яркостной составляющих, из которого видно, что её отбрасывание не ведёт к улучшению эффективности и даже ухудшает её.

Метод статического диапозона

В данной работе для обнаружения областей с цветом кожи, применяется самый простой, при этом достаточно действенный метод статических диапазонов. Для сравнения были опробованы 2 набора диапазонов из разных работ. В наборе условий 2.1 - 2.4 приведены условия, при которых пиксель принадлежит к коже по мнению авторов (Lin et al., 2005).

$$R > G, |R - G| \geq 11, \quad (2.1)$$

$$0.33 \leq r \leq 0.6, 0.25 \leq g \leq 0.37, \quad (2.2)$$

$$340 \leq H \leq 359 \vee 0 \leq H \leq 50, \quad (2.3)$$

$$0.12 \leq S \leq 0.7, 0.3 \leq V \leq 1.0 \quad (2.4)$$

(TODO где, ... надо ли. Прошли подробно все вроде)

В работе (Vezhnevets et al., 2003) приводиться похожий, но другой набор условий 2.5-2.7.

$$R > 95, G > 40, B > 20, \quad (2.5)$$

$$\max\{R, G, B\} - \min\{R, G, B\} > 15, \quad (2.6)$$

$$|R - G| > 15, R > G, R > B \quad (2.7)$$

Для реализации метода сначала выделяются и получаются отдельные каналы, после чего каждое отдельное условие приводит к получению бинарной маски. Т.е. например для реализации первой части условия 2.5 необходимо взять цветовой канал R и применив функцию $InRangeS(0, 95) - > mask$, получить изображение, на котором в тех местах, где условие удовлетворяется будет белый, а где нет - чёрный. Такие маски получаются для всех условий, после чего они объединяются логическим “И”, таким образом оставив в результирующей маске только те участки, который удовлетворяют всем условиям. Пример работы второго набора условий показан на рисунке 2.8.

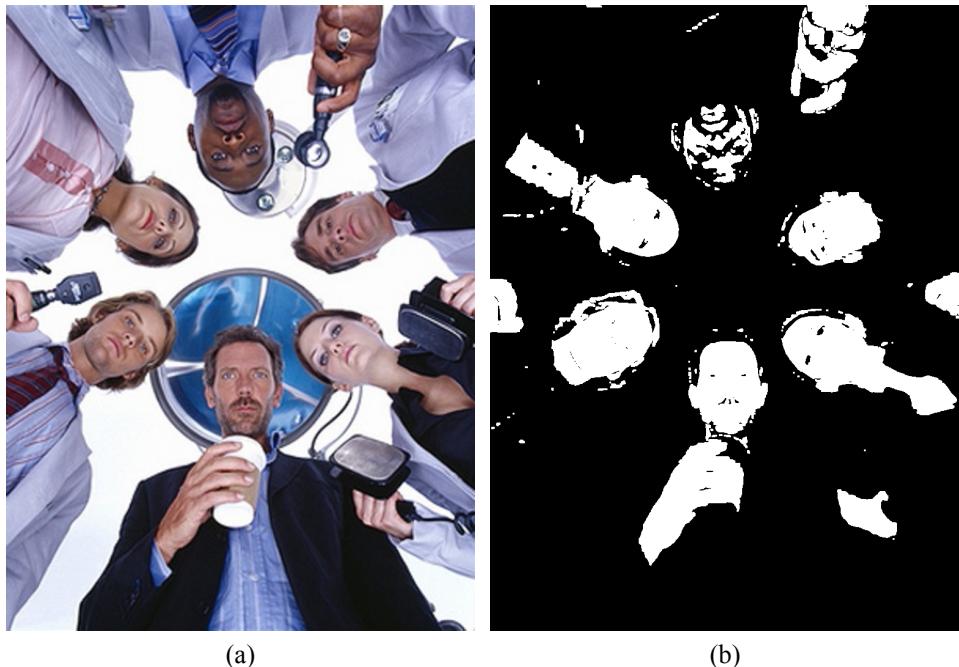


Рис. 2.8: Пример работы поиска областей с цветом кожи. Результатирующая маска (b), полученная применением условий 2.5-2.7 к изображению (a)

2.1.3 Выделение и объединение областей с цветом кожного покрова

Бинарное изображение, получаемое не выходе из 2.1.2 представляет из себя просто информацию о пиксельной интенсивности. Для дальнейшей работы, из этих данных нужно получить более интуитивные данные, например контуры.

Выделение найденных областей

В OpenCV есть ряд функций для работы с контурами, которые могут из любого изображения получить информацию о замкнутых линиях проходящих по границе контрастных районов изображения. Бинарное изображение как нельзя лучше подходит для получения чётких контуров.

На рисунке 2.9 показан процесс предобразования бинарной маски в отдельные контуры.

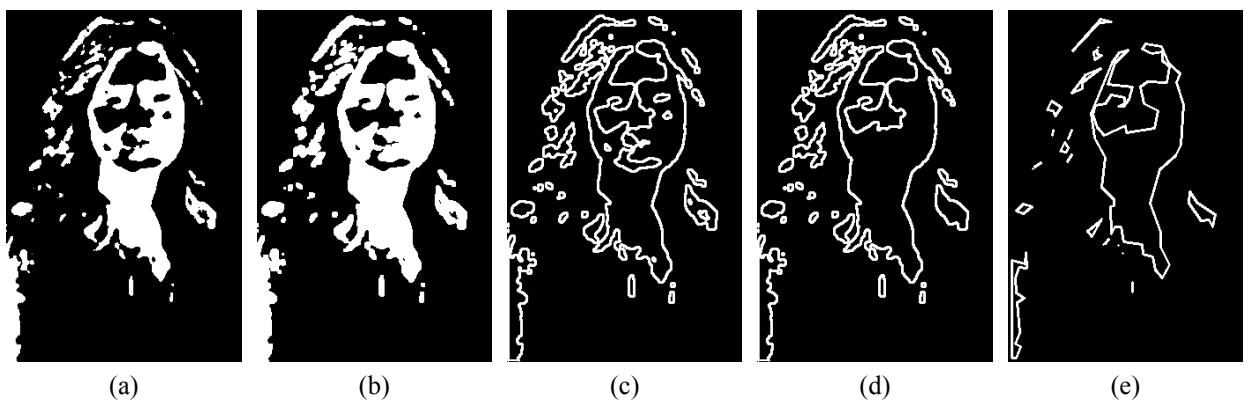


Рис. 2.9: Процесс получения контуров вокруг областей, содержащих цвет кожи.

Алгоритм выделения из маски контуров состоит из 4 этапов:

- (a) Сперва изображение с маской проходит процесс открытия или расширения (`dilate`) с одним шагом. Это необходимо для того, что бы замкнуть пограничные области. В OpenCV за это отвечает функция `dilate()`.
- (b) Далее высчитываются все имеющиеся контуры. За это в OpenCV отвечает функция `FindContours()`. Результат процедуры возвращает в виде списка отдельных замкнутых контуров, которые представлены в виде последовательности координат точек, образующих контур.
- (c) Из всех контуров остаются только внешние, тем самым отфильтровываются внутренние “дыры”, которые в будущем не потребуются.

- (d) Текущие контуры “заливаются” и получается бинарная маска, которая проходит процесс двухкратного закрытия или зжатия (*erode*). Это избавляет от очень мелких контуров и так же разъединяет те контуры, которые соединены маленькими “мостиками”, т.к. зачастую эти контуры описывают разные объекты.
- (e) Последним шагом убираются лишняя детальность контурной линии, она аппроксируется, в следствии чего уменьшается число описывающих контур вершин.

В результате имеется список контуров, описывающих небольшим количеством вершин отдельные области на изображении, где встречается цвет кожи.

Кластеризация

Как будет показано в выбранный способ классификации областей с цветом кожи не всегда работает. Бывают случаи, когда цвет кожи на лице человека обнаруживается не на всей площади, а небольшими участками. В результате эти участки будут просканированы и ничего не будет найдено. Для этого необходимо объединить эти участки в один 2.11. Ещё одна проблема, которую может решить объединение - это перекрывающиеся прямоугольники, которые обводят вокруг каждой области, для дальнейшей работы.

Подобные объединения называются кластерами, а сам процесс - кластеризацией. Методов кластеризации много. Самым простым и популярным является метод “*k-mean*”. Его реализация есть даже в библиотеке OpenCV. Минусом этого алгоритма является то, что необходимо указывать количество конечных кластеров. В данном случае это количество неизвестно. Но даже зная количество, *k-mean* кластеризует не достаточно интуитивно и полезно для данной задачи. Для кластеризации областей в качестве объекта кластеризации используются центры прямоугольников, обведённых вокруг контуров.

Интуитивным решением кажется объединять те точки, что находятся близко друг к другу, и разъединять те, что далеко. Для этого все точки надо соединить так, что бы расстояние этих соединений было минимальным. Это нужно, что бы найти самые удалённые друг от друга точки. В результате точки и их соединения образуют незамкнутый граф или дерево, а точнее *минимальное оствовое дерево* (МОД) (*Minimum spanning tree*). Для получения МОД используется библиотека NetworkX, а в качестве весов рёбер выступает евклидово расстояние между точками. Пример оствового дерева, полученного из центров прямоугольников описанных вокруг контуров, показан на рисунке 2.10.

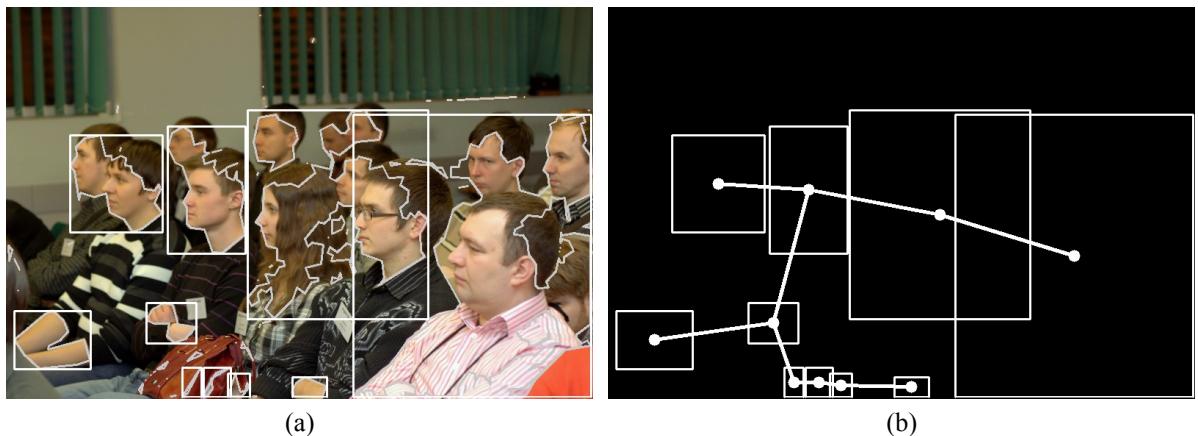


Рис. 2.10: Пример минимального оствовного дерева, где вершины - это центры прямоугольников обведённых вокруг контуров.

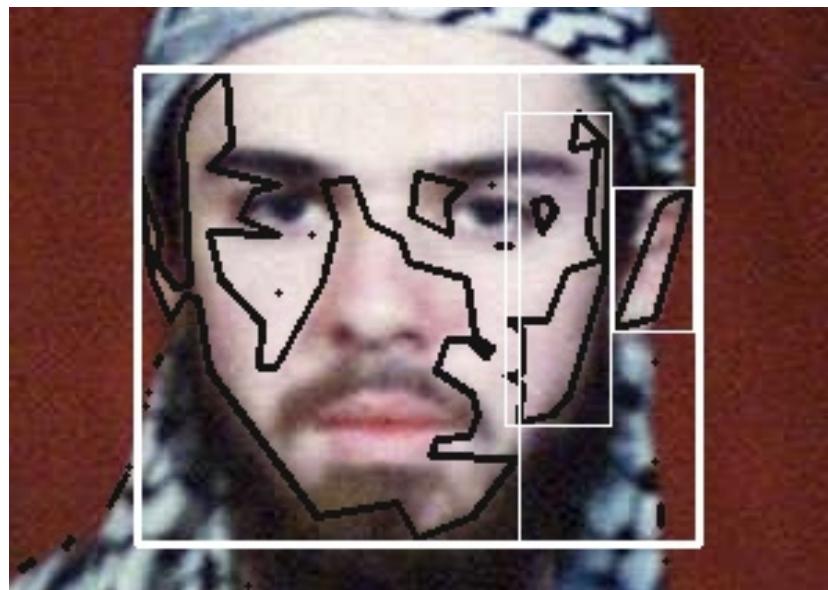


Рис. 2.11: Пример удачного использования объединения разрозненных участков в один.

Для кластеризации используется, описанный в работе (Grygorash et al., 2006) метод HEMST (*hierarchical euclidean minimum spanning tree*). Алгоритм как и k-means требует в качестве аргумента количество желаемых кластеров, но работает он значительно интуитивнее применительно к данной проблеме. Суть работы сводиться к удалению самых длинных рёбер в дереве, но до определённого предела (пока есть значительно длинные рёбра по отношению к остальным), после чего дерево аппроксимируется и опять удаляются самые длинные рёбра.

После объединения в кластеры, прямогольные области входящие в один кластер будут заменены прямогольным регионом, описанным вокруг тех, что входят в кластер. Для определения оптимального количества кластеров применяется алгоритм (см. ...), который сравнивает сумму площадей прямогольников, входящих в кластер с площа-

дью прямоугольника вокруг всего кластера (строка 11), если отношение удовлетворяет некоторому порогу (строка 12), то такое слияние принимается.

```

1: boxes  $\leftarrow$  initialBoxes
2: threshold  $\leftarrow$  initialThreshold
3: loan  $\leftarrow$  0
4: repeat
5:   initSize  $\leftarrow$  length[boxes]
6:   for k  $\leftarrow$  initSize to 1 do
7:     forMerge  $\leftarrow$  []
8:     for cluster  $\in$  HEMST(boxes, k) do
9:       bRect  $\leftarrow$  BOUNDING_RECT(cluster)
10:      clBoxes  $\leftarrow$  boxes  $\in$  cluster
11:      rel  $\leftarrow$  sqrt(BOXES_AREA(clBoxes)) / sqrt(площадь bRect)
12:      if rel  $>$  threshold then
13:        loan  $\leftarrow$  loan + (1 - rel)
14:        forMerge[length + 1]  $\leftarrow$  [bRect, clBoxes]
15:      end if
16:    end for
17:    boxes[i]  $\in$  forMerge[i][1]  $\leftarrow$  NIL
18:    boxes[length + 1]  $\leftarrow$  forMerge[] [0]
19:    threshold  $\leftarrow$  threshold + speedConst + loan/k
20:  end for
21: until initSize  $\neq$  length[boxes]
```

Тут *k* - число кластеров, *initialBoxes* - все начальные регионы вокруг контуров, *initialThreshold* - константа определяющая допустимое отношение квадратов площадей, *speedConst* - константа уменьшающая допустимый порок по мере расширяющегося последовательного объединения.

В алгоритме ... для подсчёта площади регионов, входящих в кластер, используется функция *BOXES_AREA(boxes)*. Это функция высчитывает только ту площадь, которую занимают прямоугольники все вместе. Т.е. в случае, если они накладываются друг на друга, то места наложения считаются один раз. Для этого используется алгоритм скользящей линии (*Sweep line algorithm*). Суть алгоритма в том, что вертикальная скользящая линия передвигается слева направа и регистрирует события встречаясь с правыми и левыми стороны прямоугольников. События встречи с левыми сторонами активируют эти прямоугольники, а с правой деактивируют. После каждого события горизонтальная скользящая линия проходит сверху вниз и точно так же регистрирует события верхних и нижних границ, но только активных прямоугольников. На момент

каждого события горизонтальной линии можно вычислить расстояние между текущей и предыдущей позициями как горизонтальной, так и вертикальной линий. Эти расстояния надо перемножать и получившиеся площади складывать. В результате получается площадь занятой прямоугольниками фигуры.

(картинки тут явно надо) (TODO - надо ли этот рассказ вообще)

2.1.4 Классификация

Выбор метода ИНС для классификации

В данной работе для конечной классификации используются искусственные нейронные сети. На выбор повлияли следующие факторы:

- Популярность метода среди изученных научных работ, что говорит об уместности данного выбора;
- Скорость работы ИНС позволяют использовать её даже в задачах реального времени;
- ИНС, в отличии от остальных методов, к моменту начала написания работы был единственным знакомым автору методом классификации;

TODO Как это делают другие?

Описание сети

В данной работе, для повышения чувствительности сети к различным характеристикам лица человека было использовано два представления: пиксельное и с информацией о границах. Для каждого из представлений создана своя сеть, которая обучается на обучающих образцах, представленных в соответствующем представлении. В результате имеется две сети, обученных на относительно одном и том же обучающем материале, но т.к. представления разные, то и разные параметры сети, и следовательно разные результаты на одни и те же входные данные.

Нейронная сеть состоит из входных нейронов, выходных нейронов и возможных слоёв со скрытыми нейронами. Входные нейроны образуют входной вектор, а выходные соответственно выходной вектор. Как писалось ранее структура сети, многие другие аспекты ИНС является важным фактором, от которого зависит эффективность обучения и работы сети в целом. В качестве входного вектора в данной работе берётся массив из

1024 элементов. Значениями этого массива принимают значения интенсивностей пикселей чёрно-белого изображения размером 32 на 32 пикселя. Выходной вектор состоит из двух элементов.

Оптимальное количество параметров должно коррелироваться с размером входного вектора, со сложностью модели и с количеством возможных классов. Слишком большое количество параметров приведут к “переобучению” сети, а слишком малое количество не дадут возможности описать более детальную модель. В случае полно связной сети без скрытого слоя получается 2048 связей (параметров). Следовательно количество нейронов в скрытом слое должно быть минимально или отсутствовать вовсе. Для того что бы найти оптимальную структуру необходимо протестировать оба варианта.

В стандартной нейронной сети кроме описанных выше нейронов также необходимо иметь особый нейрон смещения (*bias*). Нейрон смещения соединяется со всеми нейронами скрытых и выходных слоёв. Они участвуют формировании суммы, приходящей в качестве аргумента в активационную функцию. В частности они могут сдвигать функцию на право или на лево. Подсчёт сигнала с использованием нейрона смещения и сигмойдной активационной функцией приведёт в формуле 2.8.

$$out_i = \text{sig}\left(\left(\sum_{k=1}^N in_k \times w_k\right) + bias_i \times 1.0\right) \quad (2.8)$$

где

out_i	– выходной сигнал i -того нейрона
N	– число входных связей i -того нейрона
in_k	– входной значение от k -ого входного нейрона
w_k	– вес k -ого соединения
$bias_i$	– вес соединения между нейроном смещения и i -ым нейроном

Код создания нейронной сети при помощи библиотеки PyBrain приведёт в листинге 2.1.

```

1 def build_simple_ann(indim, outdim):
2     ann = FeedForwardNetwork()
3
4     ann.addInputModule(LinearLayer(indim, name='in'))
5     ann.addOutputModule(SoftmaxLayer(outdim, name='out'))
6     ann.addModule(BiasUnit(name='bias'))
7
8     ann.addConnection(FullConnection(ann['in'], ann['out']))
9     ann.addConnection(FullConnection(ann['bias'], ann['out']))
10
11    ann.sortModules()
12    return ann

```

```

13
14 def build_complex_ann(indim, outdim):
15     ann = FeedForwardNetwork()
16
17     ann.addInputModule(LinearLayer(indim, name='in'))
18     ann.addModule(SigmoidLayer(10, name='hidden'))
19     ann.addOutputModule(SoftmaxLayer(outdim, name='out'))
20     ann.addModule(BiasUnit(name='bias'))
21
22     ann.addConnection(FullConnection(ann['in'], ann['hidden']))
23     ann.addConnection(FullConnection(ann['hidden'], ann['out']))
24     ann.addConnection(FullConnection(ann['bias'], ann['out']))
25     ann.addConnection(FullConnection(ann['bias'], ann['hidden']))
26
27     ann.sortModules()
28
29     return ann

```

Listing 2.1: Создание сети со скрытым слоем и без него.

TODO -Если листинг катит, описание тут.

Обучение сети

Самым важным и первым этапом работы с сетью после её создания является процесс обучения. Он сопряжён с определённым проблемами и элементом вероятности - даже правильно спроектированная сеть с правильными образцами может не научиться правильной модели. Другие методы (типа SVM) более стабильны, по-этому в последнее время предпочтение отдается им, нежели ИНС.

Как и со многими другими методами непараметрической классификации (ИНС в данном случае называют полупараметрической (Xu and Zhu, 2006)) существуют две противоположные опасности : “недоучить” (*under fit*) и “переучить” (*over fit*). Суть в том, что когда классификатор обучают, его снабжают представителями различных классов, в данном случае образцами лиц и не лиц. Классификатор должен усвоить различные общие характеристики предоставляемых образцов, а детали, которыми образцы отличаются друг от друга не учитывать как шум. В результате чего параметры классификатора будут описывать модель данного класса. “Не доучить” - значит научить классификатор только некоторым из возможных характеристик истинной модели. В результате он будет описывать только часть модели, что приведёт к ошибкам конечной классификации. “Переучить” - значит научить классификатор не только основным характеристикам модели, но и мелким деталям, относящимся к конкретным образцам, а не всё модели в целом. В результате классификатор так же будет ошибаться. (Bradski

and Kaehler, 2008) Визуальное представление этих двух ошибок при обучении показаны на рисунке 2.12.

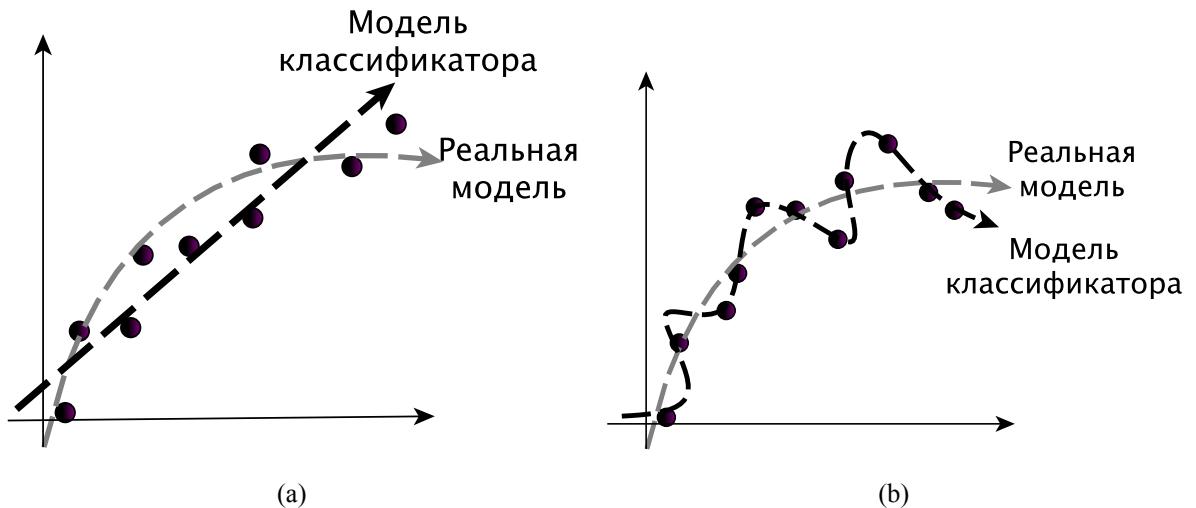


Рис. 2.12: Примеры ошибок во время тренировки сети. На (а) показана недоученная модель, а на (б) переученная. Чёрные точки символизируют образцы реальной модели. (Bradski and Kaehler, 2008)

Для того что бы во время обучения не произошло “переобучения”, т.е. что бы сеть не потеряла способность к обобщению, “запомнив” частные характеристики образцов из обучающего набора, применяют метод преждевременной остановки (early stopping). Суть метода показана на рисунке 2.13.

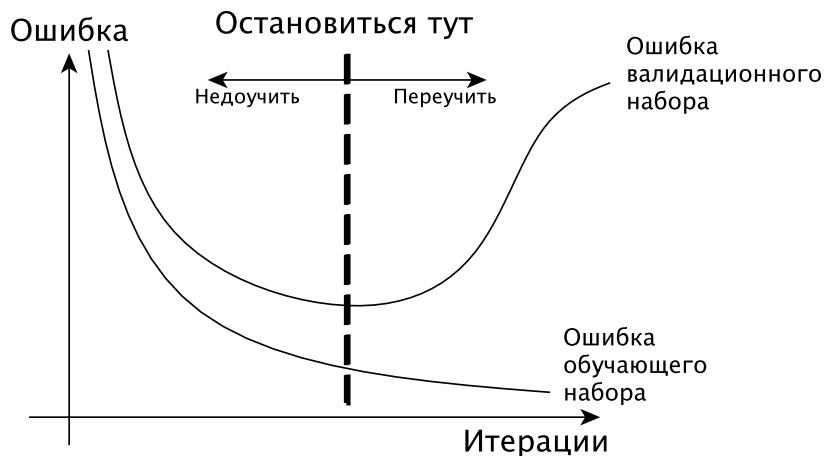


Рис. 2.13: Метод преждевременной остановки позволяет увидеть момент перенасыщения.

Суть метода в том, что перед началом обучения, обучающий набор делится на две части (обычно 8 к 2-ум) для обучения и параллельной валидации. Во время обучения с большим набором после каждой итерации высчитывается ошибка, применяя текущую сеть к валидационному набору. Сигналом к остановке обучения служит момент, когда

ошибка валидационного набора начинает расти. Библиотека PyBrain применяет этот метод во время обучения методом обратного распространения.

Ещё одной проблемой, которая появляется при работе с ИНС - это подбор правильных образцов. Этот этап, как и многие другие в основном выполняется методом проб и ошибок. "Легко найти представительные образцы изображений, содержащих лицо, но гораздо сложнее получить такие, где оно отсутствует"(Rowley et al., 1998) Проблема в том, что векторное пространство, в котором находятся входные вектора сети очень велико, большая часть которого содержит вектора класса "не лицо а вектора класса "лицо" локализованы плотной группой. Обучить сеть - значит научить её отличать эту компактную группу от всего остального пространства, а для этого нужно, что бы образцы описывающие класс "не лицо" представляли всё это пространство, что очень сложно из-за его многообразия. (TODO - переписать это)

В данной работе образцы класса "лицо" были взяты из трёх баз данных: (Samaria and Harter, 1994) (400 лиц), (GeorgiaFace) (750 лиц) и (Huang et al., 2007) (13233 лица). Размеры лиц в разных источниках разные, по-этому они были обрезаны и уменьшены до размера 32 на 32 пикселя. Примеры лиц из обучающего набора в двух представлениях приведёны на рисунке 2.14.

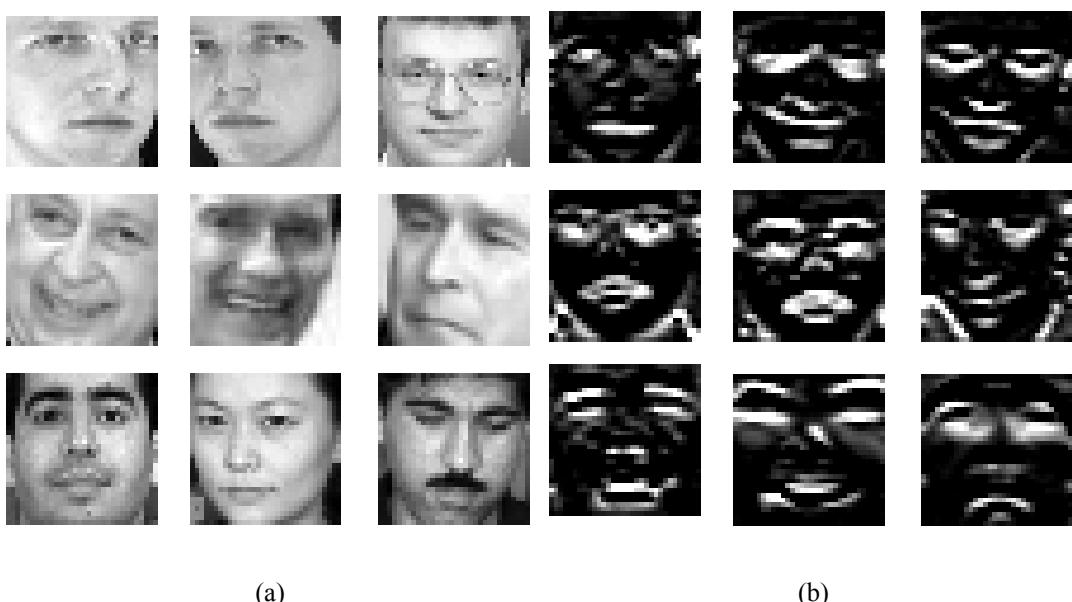


Рис. 2.14: Примеры образцов для обучения из разных баз данных. (а) пиксельное представление. (б) информация о границах

Образцы класса "не лиц" были получены из больших изображений не содержащих лиц методом скользящего окна (см. ниже). Для этого использовались изображения из интернета и собственные фотографии. Примеры фотографий и выборок показаны на рисунке 2.15. Всего было сделано более 40000 образцов "не лиц" из 20 фотографий.

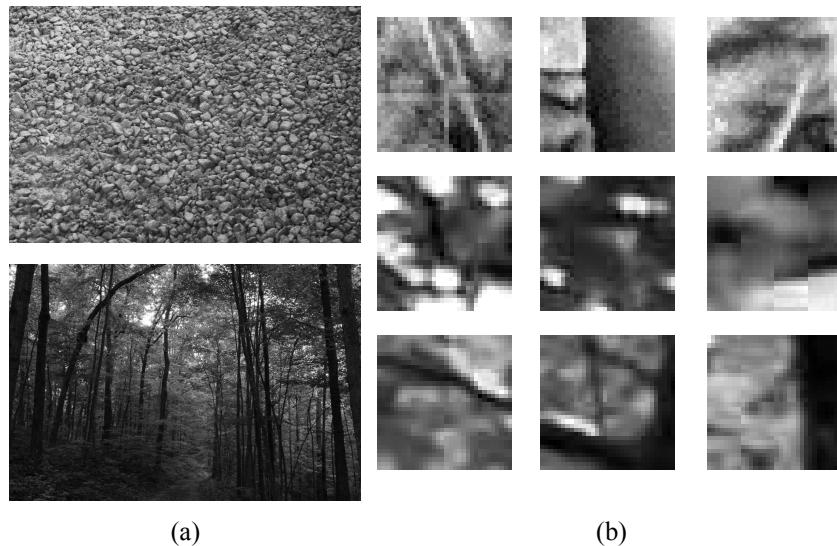


Рис. 2.15: Примеры образцов не содержащих лица (b), полученных из различных фотографий (a)

Что классификатор не обучился каким-то специфическим характеристикам, относящимся к одной сцене на однои из изображений из которых производились образцы “не лиц” или на фотографиях лиц, сделанных в студийных условиях, сделано 4 набора для обучения. В каждом набор находится по 2000 представителя из каждого класса. Каждый из источников (фотографии без лиц и базы данных) представлены в данных наборах в равных количествах.

В (Rowley et al., 1998) что бы получить представительные образцы не содержащие лиц используется следующий подход. Для обучения случайным образом выбираются представители обоих классов. Так же создаётся выборка для тестирования. После некоторого количества итерация обучения, производиться тестирование на тестовом наборе. Все образцы, не прошедшие тестирование (т.е. были неверно определены) переходят в набор для обучения. “Наличии таких примеров заставляют нейронную сеть выучивать чёткую границу между изображениями с и без лиц” (Rowley et al., 1998) Подобная методика используется и в данной работе.

Для того, что бы примерно представить чему в конечном счёте обучиться сеть можно взять всех представителей классов и получить усреднённое изображение 2.16.

Как видно на рисунке усреднённые лица содержат явные признаки лица, в то же время усреднив изображения из класса “не лицо” получиться серый квадрат без каких-либо деталей, что и должно быть.

Крайние части образцов, а особенно углы, часто содержат информацию ненужную информацию, т.к. представляют зачастую фон, а не часть лица. Для борьбы с этим в неко-

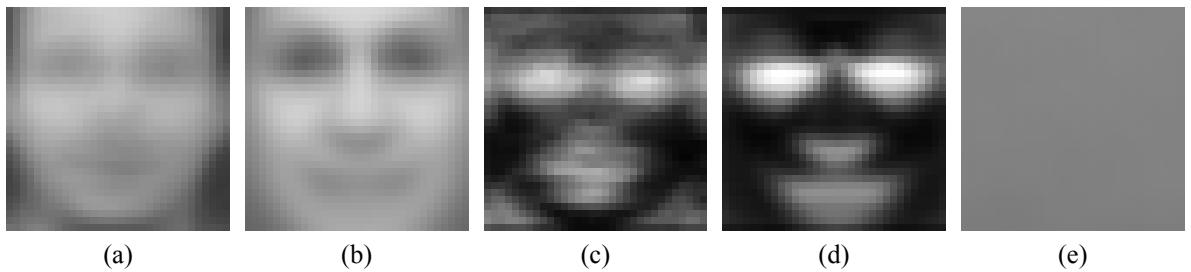


Рис. 2.16: Усреднённые лица представителей обоих классов. (а), (б) пиксельное представление лиц из разных баз данных; (с), (д) представление информацией о границах из разных баз данных; (е) усреднённое изображение класса “не лицо”.

торых работах используется специальная рамка, которая закрывает эту часть изображения и во время обучения и во время применения сети 2.17. Такая рамка используется и в данной работе.

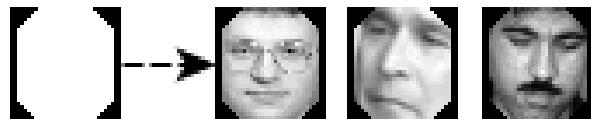


Рис. 2.17: Рамка и образцы с наложенное рамкой.

В листинге 2.2 приведёт метод, в котором происходит обучения сети по данному на вход обучающему набору.

```

1 def get_trained_ann(dataset, test_train_prop=0.25, max_epochs=50):
2     tstdata, trndata = dataset.splitWithProportion(test_train_prop)
3     trndata._convertToOneOfMany()
4     tstdata._convertToOneOfMany()
5
6     ann = build_simple_ann(trndata.indim, trndata.outdim)
7
8     trainer = BackpropTrainer(ann, dataset=trndata, learningrate=0.01, momentum=0.5,
9                               verbose=True)
10    trainer.trainUntilConvergence(maxEpochs=max_epochs)
11
12    trnresult = percentError( trainer.testOnClassData(), trndata['class'] )
13    tstresult = percentError( trainer.testOnClassData(dataset=tstdata ), tstdata['
14        class'] )
15
16    return ann, trnresult, tstresult

```

Listing 2.2: Обучение сети на основе данного обучающего набора.

На вход функции подаётся уже подготовленный набор с образцами обоих классов (`dataset`). В строке 2 набор делиться на две части: для обучения и для тестирования. На строке 6 создаётся сама ИНС (`ann`). На строке 8 создаётся объект для тренировки методом

обратного распространения (`trainer`), после чего в строке 9 запускается сам процесс обучения. Обучение длиться или до сходимости (2.13) или если пройдено максимальное количество итераций.

Применение сети

После того как сеть обучена, её можно сохранить. Это делается стандартными средствами для сериализации языка Python, т.к. сама сеть это обычный объект, где параметры сети - это свойство сети, представляющий собой обычный массив с данными. По-этому такой объект легко сохраняется на диск и вычитывается вновь как объект.

В данной сети два выхода, где каждый отвечает за свой класс. Во время обучения когда на вход подаётся лицо, то правильным ответом считается единица на одном из выходов, а ноль на другом и наоборот. Во время применения сети результаты обычно не равны идеально нулю и единице, а варьируются в ту или иную стороны. Использование активационной функции на выходных нейронах создаёт ситуацию, при которой на выходе сумма значений обоих нейронов всегда будет равна единице. Т.е. результатом после очередного применения могут быть 0.32 на одном и 0.68 на другом выходном нейроне. Для интерпретации ответа сети необходимо установить порог (`threshold`), который будет показывать в каком случае считать, что сеть показывает на лицо, а в каком на его отсутствие. Порог устанавливается экспериментально.

Одной из самых больших проблем описанного в данной работе метода является тот факт, что алгоритм не знает ни положения, ни предполагаемого размера лица, которое ему необходимо найти на изображении. Для этого он должен просканировать всё изображение на разных масштабах. Это делается методом скользящего окна рисунок 2.18.

Алгоритм скользящего окна следующий: квадратное окошко размером 32 на 32 пикселя становится в верхний левый угол изображения. Часть изображения под этим окошком и есть первый образец. После этого окошко сдвигается влева (на 1 или 2 пикселя, например) и получается второй образец. И т.д. до конца строки, потом на пиксель ниже и так далее до правого нижнего угла. После этого окошко становится в начальную позицию, а размер его увеличивается (например в 1.5 раза) и всё повторяется, только в этот раз полученные образцы уменьшаются до размера 32 на 32 пикселя. *sliding window* алгоритм. диаграммы, код.

Таким образом каждое изображение, а в случае потока изображений каждый кадр, должны быть отсканированы таким образом. Полученные образцы трансформируются в две представления и поступают каждый в свою сеть. Если обе сети дают положи-

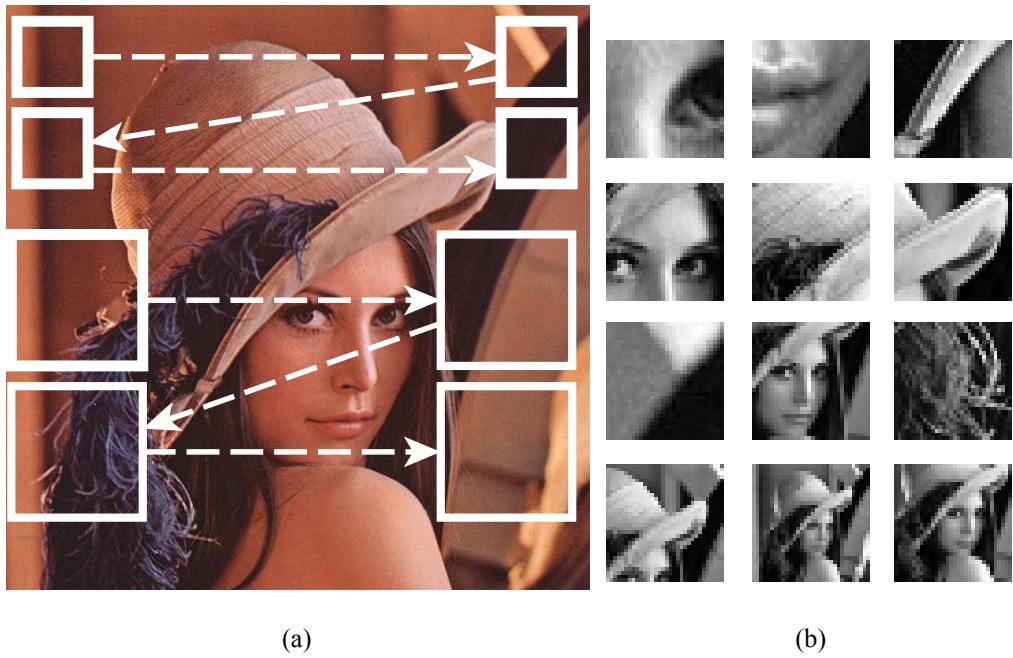


Рис. 2.18: Метод скользящего окна. (а) путь скользящего окна, (б) примеры полученных образцов.

тельный ответ, то считается, что изначальное окошко, из которого был сделан образец содержит лицо.

2.2 Выбор цели для слежения

После того, как все лица в кадре были найдены, необходимо определить за которым надо следить, что бы знать в какую сторону поворачивать камеру. Необходимо, что бы алгоритм не переключался с лица на лицо, а следил за выбранным лицом до определённого момента. Использовать алгоритмы распознавания нельзя, т.к. они вычислительно-ёмкие. Вместо этого можно использовать достаточно простой алгоритм, изображённый на рисунке 2.19.

Как видно на рисунке, сначала находится лицо, площадь описывающего прямоугольник вокруг которого наибольшая, что должно соответствовать ближайшему к камере лицу. После того как известны координаты выбранного лица из предыдущего кадра, можно предположить, что лицо не удалиться очень далеко за короткий промежуток времени и его можно найти в окрестности текущего. На случай, если появиться какое-то другое лицо со значительно большей площадью, чем предыдущее, то необходимо переключиться на него.

Для того, что бы повернуть камеру к выбранному лицу достаточно построить вектор из центра изображения камеры к центру прямоугольника лица. Компоненты полученного

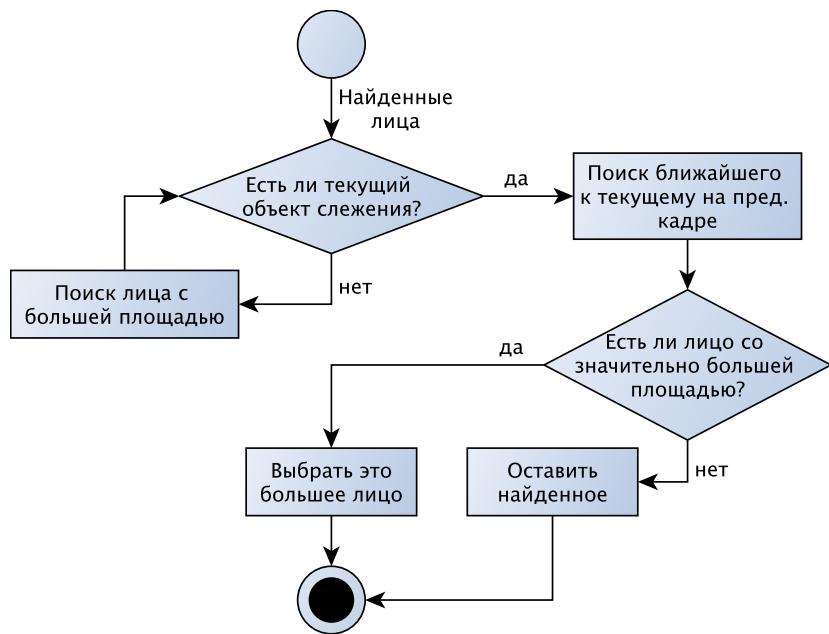


Рис. 2.19: Алгоритм выбора лица за которым следить.

вектора по x и y представляют собой то на сколько необходимо повернуть камеру. Исходя из полученного экспериментально соотношения одного пикселя к углу поворота сервопривода можно вычислить на какой угол и с каким знаком необходимо повернуть сервоприводы.

3 РЕЗУЛЬТАТЫ РАБОТЫ (ИСПЫТАНИЯ?)

Ниже представлены результаты испытаний всех частей приведённого в работе метода по отдельности и всех частей в целом.

3.1 Автоконтраст и баланс белого

Автоматическая коррекция контраста в большинстве случаев работает так как надо. Это касается прежде всего чёрно-белых изображений. Т.к. алгоритм коррекции контраста и баланса белого - это одно целое, то работу коррекции автоконтраста можно оценить только на чёрно-белых изображениях.

На рисунке 3.2 показаны случаи удачного применения автокоррекции контраста для на чёрно-белых изображениях, а на рисунке 3.1 для цветных.

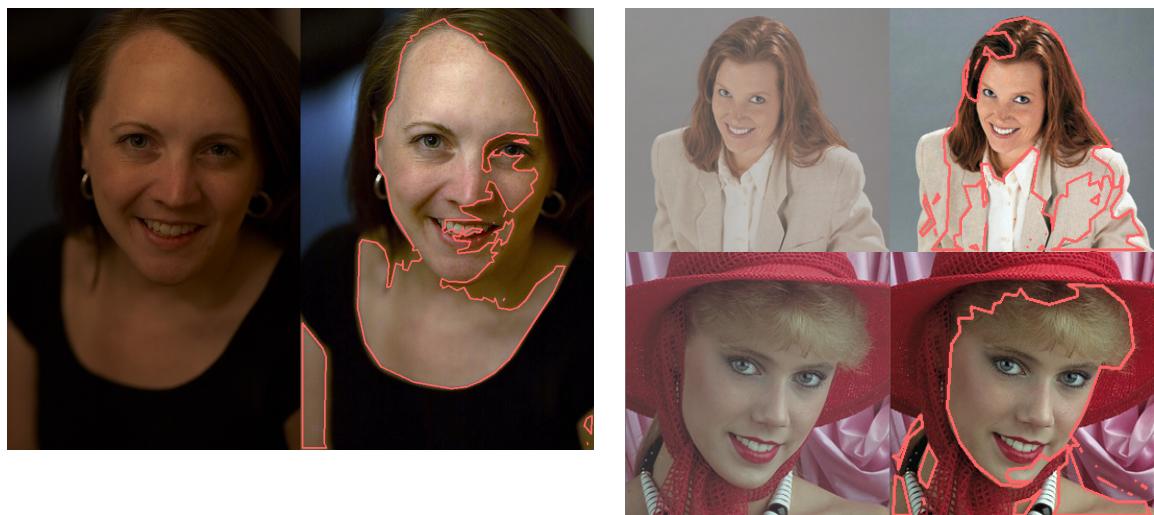


Рис. 3.1: Примеры удачного использования автокоррекции контраста для цветных изображений.

Алгоритм предварительной обработки (автокоррекция контраста и баланса белого) использует параметр *aggression* (см. ??). От присудствия этой переменной и её размера зависит на сколько агрессивно будет работать алгоритм. Если переменная слишком большая, то порой алгоритм ошибается, приводя к нежелательным результатам (рис. 3.3)



Рис. 3.2: Примеры удачного использования автокоррекции контраста для чёрно-белых изображений.

Как видно из приведённых примеров некоторым изображениям достаточно нулевого значения (первый ряд), другим же даже очень высокое значение идёт только на пользу (последний ряд), а у некоторых после определённого значения цветовая гамма резко нарушается. Причина в том, что это совершенно нормально, когда гистограмма одного канала существенно сдвинуто по отношению к другому, т.к. на изображение может быть просто много одного цвета и мало другого. В таком случае выравнивать их не нужно. Узнать есть ли на изображении доминантный цвет - задача сложная и данной работе не рассматривается.

В будущем что бы хоть как-то решить проблему с ненатуральными цветами в следствии неправильной автокоррекции для данной задачи (поиск кожного покрова) можно применить искусственный приём. Можно получить контуры кожного покрова до коррекции и после. Сравнив площади оставить тот вариант, в котором площадь больше. Дело в том, что если изображение имеет неправильный баланс белого и более “тёплое”, чем должно быть и при этом содержит лицо, то убрав “тёплые” тона со всего изображения,

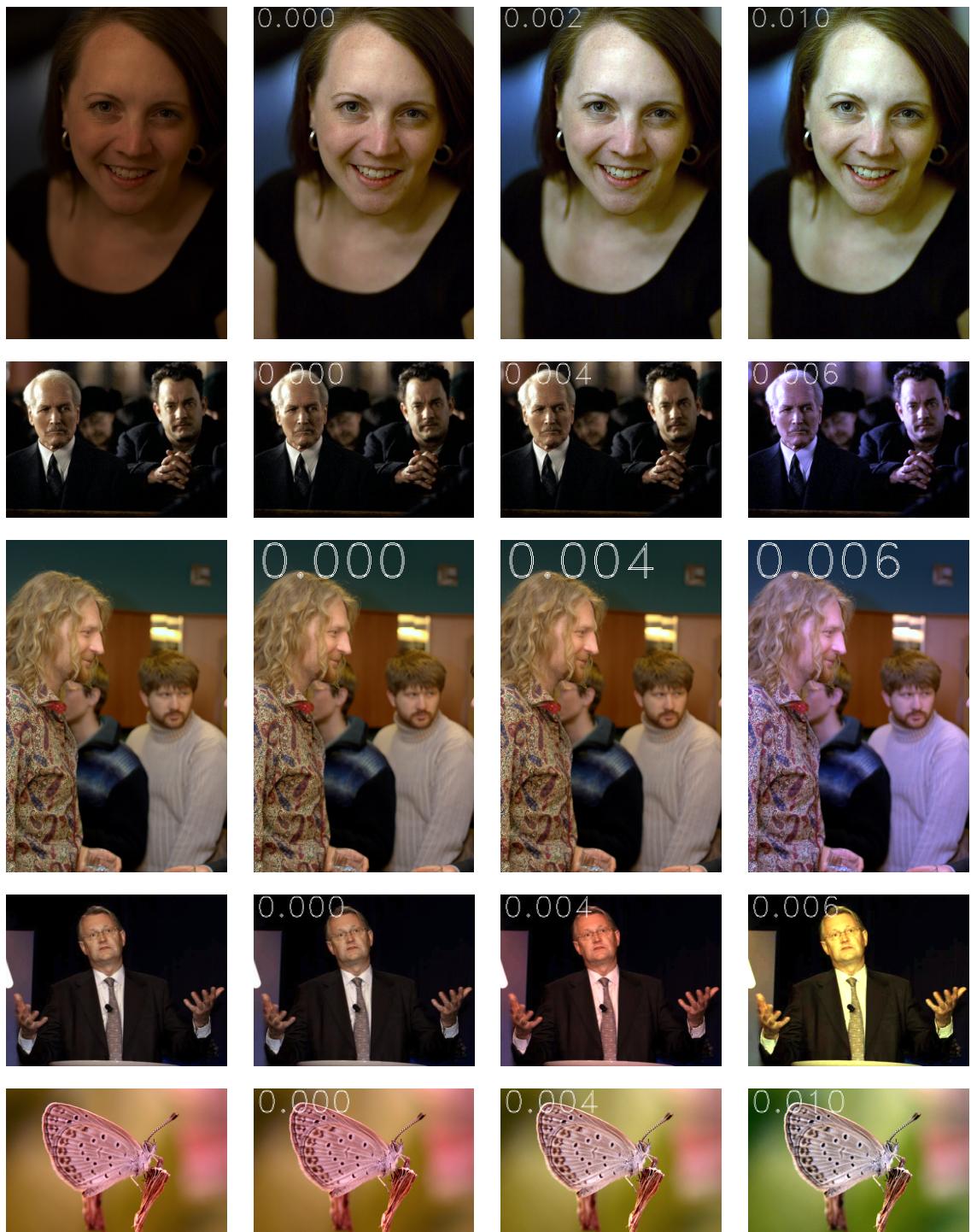


Рис. 3.3: Варианты работы автокоррекции баланса белого с разными константами *aggression*. В левой колонке оригинал.

цвет кожи так же потеряет этот тон, и вероятно перестанет определяться алгоритмом поиска кожи.

3.2 Поиск зон с кожным покровом

Выбранные в данной работе методы поиска цвета кожи статическим диапозоном даёт достаточно хороший результат. Как и требуется от специфики задача позитивно, что бы он мог ошибаться в ошибочно позитивную сторону (*true positive*), т.е. он может ошибочно показывать на кожу, когда её там нет. Если бы он больше ошибался, не показывая на места с кожей, когда они там есть, то это бы не давало возможности классификатору даже искать в тех областях, где лицо есть на самом деле.

На рисунке 3.4 показаны случаи, когда алгоритм хорошо находит участки с кожей. Как видно из примеров, модель кожного покрова применяемая в данной работе способен опознать цвет кожи различных национальностей. На рисунке изображено по два варианта каждого изображения: слева и сверху это первый набор формул (2.1-2.4), а справа и снизу второй (2.5 - 2.7). В ходе испытаний было замечено, что первый набор работает в большинстве случаев лучше (рис. 3.4 нижнее фото), но бывают случаи, когда второй лучше (рис. 3.4 второй ряд справа)

На рисунке 3.5 показаны примеры с большим количеством ложных положительных обнаружений. Как писалось выше, это не так страшно, как если бы цвет кожи не был найден совсем.

Что бы сделать алгоритм определения более эффективным можно выбрать любой другой метод классификации. Данная проблема, как писалось выше, представляет собой задачу классификации, где есть два класса "кожа" и "не кожа". Например что бы решить эту проблему при помощи ИНС. У такой ИНС на вход будет поступать вектор представляющий цвет, а на выходе будет один или два нейрона для определения одного из двух классов. Входов может быть, например, 6 или 9, где каждый из них будет принимать значения каналов из разных цветовых пространств (R, G, B, H, S, V т.д.) Такой метод может более гибко представить модель цвета кожи.

3.3 Объединение областей

Как показывают уже приведённые примеры обнаружения кожи (рис. 3.4 второй ряд справа, нижний ряд слева) объединение разрозненных контуров иногда необходимо. Примеры работы алгоритма показаны на рисунке 3.6.



Рис. 3.4: Примеры правильного определения кожного покрова для двух наборов условий.



Рис. 3.5: Примеры большого количества ложно-положительных обнаружений.

Как видно из примеров, иногда объединение областей необходимо что бы лицо попало в область последующего сканирования. В алгоритме ... есть константы, определяющие на сколько большими могут быть объединения. Для эффективной работы эта константа должна быть тщательно подобрана. Так же имеет смысл при подсчёте суммы всех площадей изначальных прямоугольников не вычитать общие части, тогда такая сумма будет показывать площадь, которую необходимо отсканировать и уже её сравнивать с площадью предлагаемого объединения.

3.4 (Результаты) работа с ИНС

3.4.1 (Результаты) обучения и тестирования

Было опробовано две структуры ИНС: сеть без скрытого слоя (рис. 3.7a) и сеть со скрытым слоем (рис. 3.7b).

Для сети без скрытых слоев проведя несколько вариантов с количеством обучающих образцов, было установлено, что повышать количество образцов выше 1500 (обоих классов) не имеет смысла, а только замедляет процесс обучения. Таким образом сети обучались на 500 образцах лиц и 1000 не лиц. На рисунке 3.8 приведены кривые ошибки обучения.

Верхняя красная линия показывает ошибку валидационных данных, не принимающих участие в обучении, нижняя синяя линия ошибку тестовых данных, на которых обучается сеть. После обучения через сеть было пропущено 2000 лиц и 2000 не лиц из другого набора. Средние показатели оказались таковы: для сети с информацией о границах

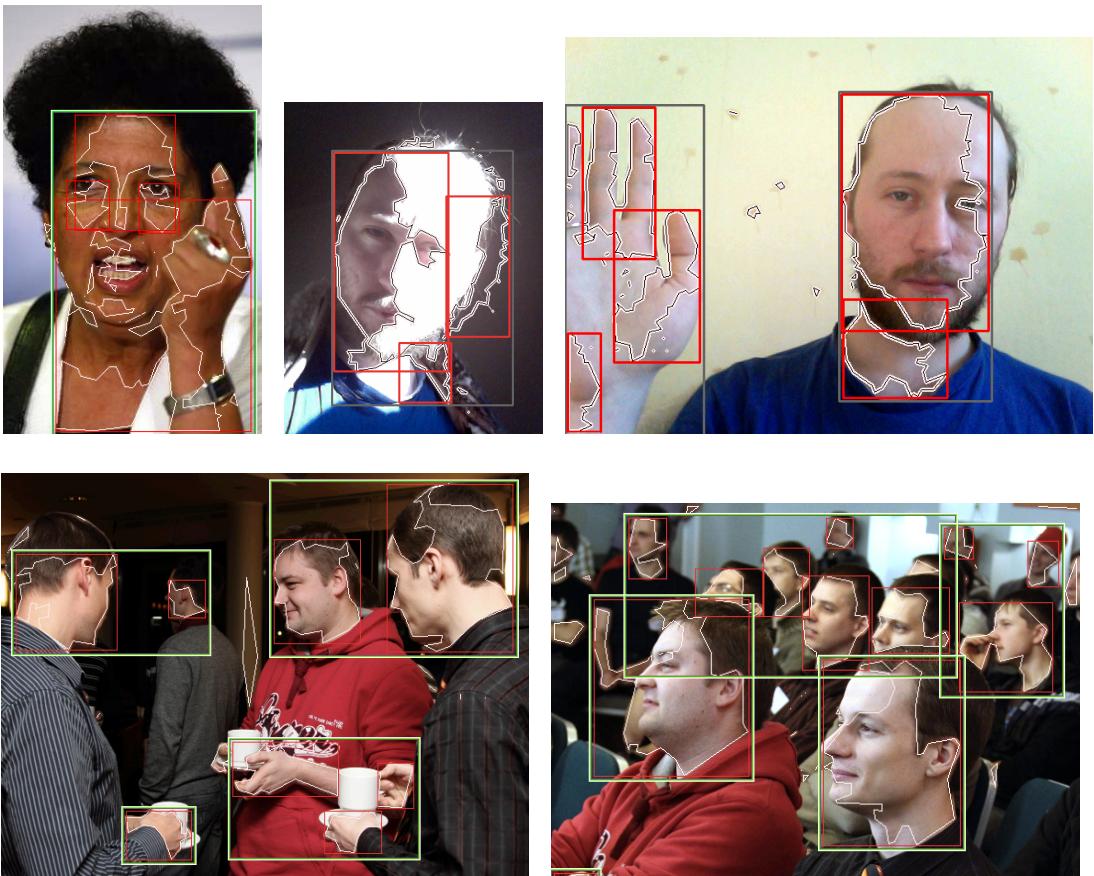


Рис. 3.6: Примеры работы кластеризации. Сверху справа серым, на остальных зелёным с белым отмечены объединённые области.

средний процент правильного ответа составил %81.372, для пиксельного представления %67.045. Основной часть ошибки составили ошибки типа неверное негативных ответов, т.е. когда сети давалось лицо, она отвечала не лицо.

Так же тестировалась сеть с 5-ю скрытыми нейронами 3.9.

Средний показатели после тестирования сети со скрытыми нейронами таковы: для сети с информацией о границах средний процент правильного ответа составил %49.945, для пиксельного представления %54.7325. Эти цифры говорят о том, что сеть ничего не научилась. Из этого следует, что сеть без скрытых слоёв в данной задаче более приемлема.

Интересно отметить как выглядит усреднённый образец всех тех, на которые при обучении сеть ошибочно дала ответи "лицо" (рис. 3.10). Особенно интересно усреднённое изображение всех ошибочно найденных лиц. Оно очень напоминает лицо, хотя сами представители этой группы не похожи.

Для того, что бы улучшить показатели необходимо сделать следующие вещи:

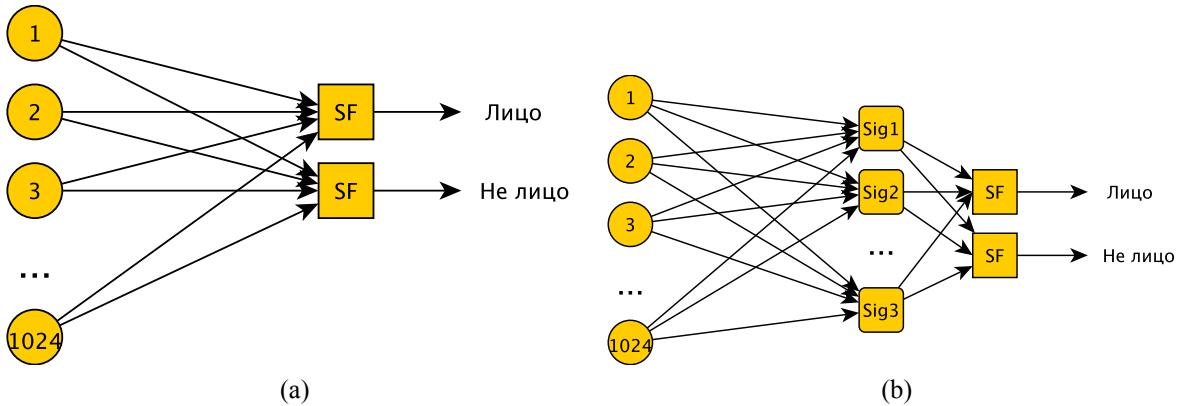


Рис. 3.7: Две структуры сети обработанные в данной работе. (а) без скрытого слоя, (б) со скрытым слоем.

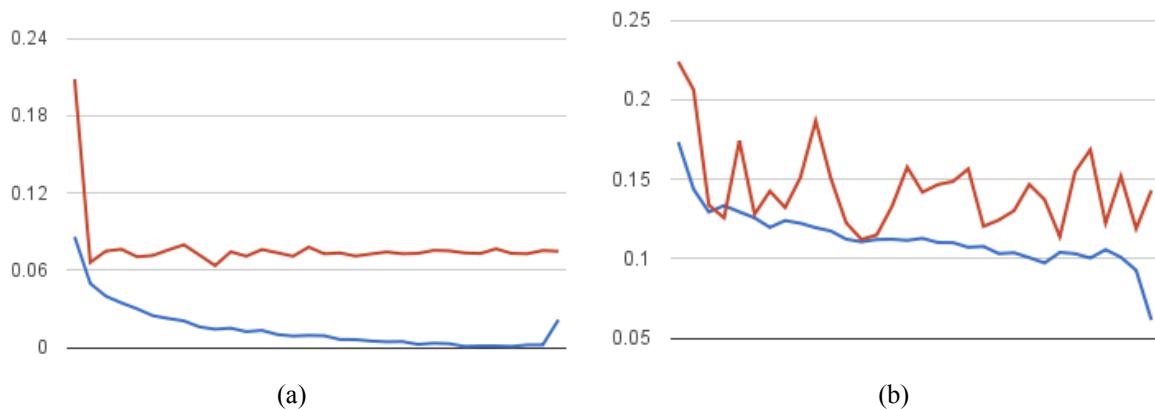


Рис. 3.8: Кривые ошибки обучения сети без скрытых слоёв. (а) обучение на пиксельном представлении, (б) обучение не образцах с информацией о границах

- Необходимо продолжить поиски правильной структуры сети. Уже понятно, что она не должна содержать много параметров, но возможны варианты. Например как это сделано в работе (Rowley et al., 1998), где есть скрытый слой, но он не полно связный, а состоит из участков, которые как бы выделяют характеристики присущие лицу.
 - Подбор образцов для обучения должен проводиться более тщательно. В используемых базах, особенно в (Huang et al., 2007) глаза, нос и рот находятся совершенно в разных положениях, а поза лица (направленность) сильно варьируется. Что бы сеть могла обучиться чёткой модели, образцы должны быть максимально однообразны по своей структуре, но разнообразны по содержанию. Т.е. лица должны быть разные, но все должны быть в одной позе и важные характеристики: глаза, нос и рот должны быть в одном месте.
 - Все образцы, которые не содержат чётко фронтального лица могут быть причислены к дополнительным классам, описывающим различные направления головы, куда эти образцы будут включены в качестве обучающих пар.

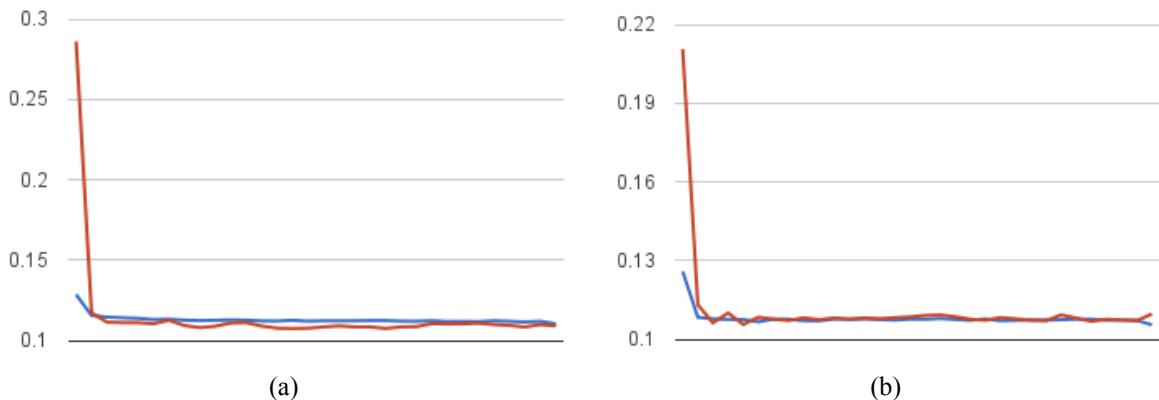


Рис. 3.9: Кривые ошибки обучения сети со скрытых слоёв. (а) обучение на пиксельном представлении, (б) обучение не образцах с информацией о границах

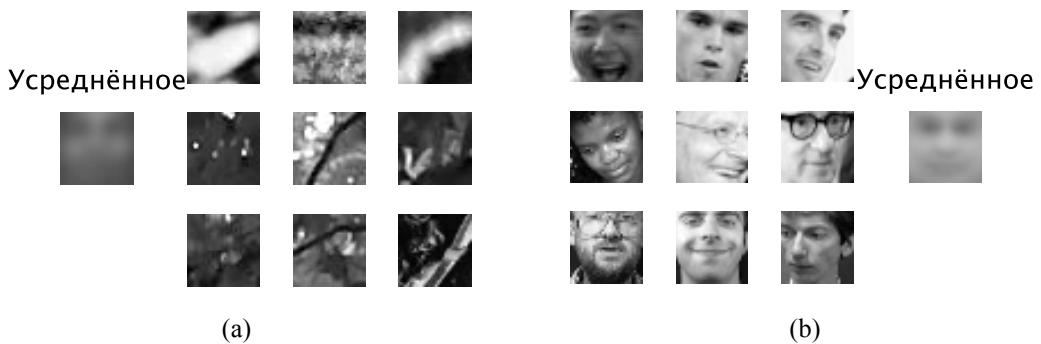


Рис. 3.10: Усреднённые лица и некоторые представители тех образцов лиц (а) и не лиц (б), на которые сеть дала неправильный ответ.

- Необходимо более тщательно и кропотливо подойти к моменту отбора неправильно опознанных образцов из валидационных наборов и включению их в обучающие наборы. Слишком много таких образцов могут “запутать” сеть слишком большим расхождением.
- Должен быть определён оптимальный порог сети. Это делается при помощи ROC таблицы.

3.5 Испытание всей системы

В результате несовершенной работы на некоторых этапах метода конечный результат не пригоден к использованию. Так обнаружение участков с кожей возвращает их слишком много, в связи с чем эффективность сканирования этих областей сравнима со сканированием всего изображения. Такая работа ставит под сомнение необходимость такого модуля.

Время затрачиваемое на сканирование небольшой картинки размером 512 на 512 занимает больше времени, чем допустимо для работы real-time приложения. Причиной является огромное количество образцов (около 97 000 для данной картинки), которые необходимо вырезать, уменьшить и применить к сети. Сама сеть работает очень быстро, а процесс вырезания и передачи в сеть занимает время. Частично в этом виновата реализация на Python, интерпретируемом языке, который не считается быстрым. С другой стороны OpenCV и PyBrain написаны на C, что даёт преимущество. Так же можно использовать специальную библиотеку psycos, которая ускоряет работу порой до 5-10 раз.

ЗАКЛЮЧЕНИЕ И ВЫВОДЫ

A ПРИЛОЖЕНИЕ. ОТЧЁТ ПО КУРСОВОЙ ПРАКТИКЕ

Литература

- A. Ahmadyfard, B. Yousefi, and S.M. Mirhassani. A Hierarchical Fuzzy Based Approach for Face Detection. In *Bioinformatics and Biomedical Engineering, 2008. ICBBE 2008. The 2nd International Conference on*, pages 2327–2330. IEEE, 2008.
- Z. Bojkovic and A. Samcovic. Face Detection Approach in Neural Network Based Method for Video Surveillance. In *Neural Network Applications in Electrical Engineering, 2006. NEUREL 2006. 8th Seminar on*, pages 44–47. IEEE. ISBN 1424404339.
- G.R. Bradski and A. Kaehler. *Learning opencv*. O'Reilly, 2008. ISBN 0596516134.
- G. Capi, H. Toda, and T. Nagasaki. A vision based robot navigation and human tracking for social robotics. In *Robotic and Sensors Environments (ROSE), 2010 IEEE International Workshop on*, pages 1–6. IEEE, 2010.
- D.A. Forsyth and M.M. Fleck. Automatic detection of human nudes. *International Journal of Computer Vision*, 32(1):63–77, 1999. ISSN 0920-5691.
- GeorgiaFace. Georgiatechfacedatabase, 2011. URL http://www.anefian.com/research/face_reco.htm.
- O. Grygorash, Y. Zhou, and Z. Jorgensen. Minimum spanning tree based clustering algorithms. In *Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on*, pages 73–81. Ieee, 2006. ISBN 0769527280.
- G.B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. *University of Massachusetts, Amherst, Technical Report 07*, 49:1, 2007.
- H. Jee, K. Lee, and S. Pan. Eye and face detection using SVM. In *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004. Proceedings of the 2004*, pages 577–580. IEEE, 2004. ISBN 0780388941.
- P. Kakumanu, S. Makrogiannis, and N. Bourbakis. A survey of skin-color modeling and detection methods. *Pattern Recognition*, 40(3):1106–1122, 2007. ISSN 0031-3203.
- CNR Kumar and A. Bindu. An efficient skin illumination compensation model for efficient face detection. In *IEEE Industrial Electronics, IECON 2006-32nd Annual Conference on*, pages 3444–3449. IEEE, 2006. ISBN 1424403901.

- H.J. Lin, S.Y. Wang, S.H. Yen, and Y.T. Kao. Face detection based on skin color segmentation and neural network. In *Neural Networks and Brain, 2005. ICNN&B'05. International Conference on*, volume 2, pages 1144–1149. IEEE, 2005. ISBN 0780394224.
- X. Liu, G. Geng, and X. Wang. Automatically face detection based on BP neural network and Bayesian decision. In *Natural Computation (ICNC), 2010 Sixth International Conference on*, volume 3, pages 1590–1594. IEEE, 2010.
- RC Luo, AC Tsai, and CT Liao. Face Detection and Tracking for Human Robot Interaction through Service Robot. In *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*, pages 2818–2823. IEEE, 2007. ISBN 1424407834.
- A. Mohamed, Y. Weng, J. Jiang, and S. Ipson. Face detection based neural networks using robust skin color segmentation. In *Systems, Signals and Devices, 2008. IEEE SSD 2008. 5th International Multi-Conference on*, pages 1–5. IEEE, 2008.
- RetinaOnWiki. Retina on wikipedia.org, 2011. URL <http://en.wikipedia.org/wiki/Retina>.
- H.A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(1):23–38, 1998. ISSN 0162-8828.
- P. Ruangyam and N. Covavisaruch. An efficient region-based skin color model for reliable face localization. In *Image and Vision Computing New Zealand, 2009. IVCNZ'09. 24th International Conference*, pages 260–265. IEEE, 2009.
- FS Samaria and AC Harter. Parameterisation of a stochastic model for human face identification. In *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, pages 138–142. IEEE, 1994. ISBN 081866410X.
- V. Saxena, S. Grover, and S. Joshi. A real time face tracking system using rank deficient face detection and motion estimation. In *Cybernetic Intelligent Systems, 2008. CIS 2008. 7th IEEE International Conference on*, pages 1–6. IEEE, 2008.
- C. Shavers, R. Li, and G. Lebby. An SVM-based approach to face detection. In *System Theory, 2006. SSST'06. Proceeding of the Thirty-Eighth Southeastern Symposium on*, pages 362–366. IEEE, 2006. ISBN 0780394577.
- CC Tsai, WC Cheng, JS Taur, and CW Tao. Face detection using eigenface and neural network. In *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, volume 5, pages 4343–4347. IEEE, 2006. ISBN 1424400996.
- M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991. ISSN 0898-929X.

- V. Vezhnevets, V. Sazonov, and A. Andreeva. A survey on pixel-based skin color detection techniques. In *Proc. Graphicon*, volume 3. Citeseer, 2003.
- P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. 2001. ISSN 1063-6919.
- Z. Xu and M. Zhu. Color-based skin detection: survey and evaluation. In *Multi-Media Modelling Conference Proceedings, 2006 12th International*, pages 10–pp. IEEE, 2006. ISBN 1424400287.
- L. Zhang and Y. Liang. A fast method of face detection in video images. In *Advanced Computer Control (ICACC), 2010 2nd International Conference on*, volume 4, pages 490–494. IEEE, 2010.
- H. Zheng, M. Daoudi, and B. Jedynak. Blocking adult images based on statistical skin detection. *Electronic Letters on Computer Vision and Image Analysis*, 4(2):1–14, 2004.
- C. Рассел and П. Норвиг. Искусственный интеллект: современный подход. М.: Вильямс, 2006.
- Ф. Уоссерман. *Нейрокомпьютерная техника. Теория и практика*. М.:–“Мир, 1992.