ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ т. 10, № 4(43), с. 111–139

ББК 3818:3813.53
ГРНТИ 27.35.43,28.23.15
УДК 004.94:591.484

А. П. Маштаков, К. А. Путинцева

Программный комплекс обработки изображений на основе модели зрения человека

Аннотация. Рассматривается задача компьютерного моделирования методов обработки изображений, основанных на принципах работы зрительной системы человека. Предложена структура программного комплекса (ПК), моделирующего первые этапы обработки зрительного сигнала мозгом человека: сглаживание изображений (удаление шума), выделение границ объектов (контуров изображения) и ориентационный анализ (определение угла наклона контуров). Эти этапы моделировались с помощью фильтров Гаусса для сглаживания и представления изображения, аппарата гауссовых производных для выделения границ и фильтра Габора для определения ориентаций. На языке Си, tcl/tk с использованием библиотек libpng, libgsl разработан ПК Visual Processing, выполняющий следующие функции: размытие изображений по Гауссу; дифференцирование изображений с помощью гауссовых производных, выделение границ объектов с помощью лапласиана гауссиана (LoG фильтра); определение направления контуров и подъем изображения на группу Ли SE_2 с помощью фильтров Габора. ПК основан на принципах открытой архитектуры и является платформой для реализации и тестирования алгоритмов обработки изображений в сфере математического моделирования зрения.

Ключевые слова и фразы: обработка изображений, модель зрения, зрительная система человека, гауссовы производные, фильтр Габора, группа движений плоскости.

Введение

Исследование биологических зрительных систем вызывает большой интерес среди исследователей во многих областях науки. Зрительная система устроена таким образом, что хранение и обработка огромных

Исследование выполнено за счет гранта Российского научного фонда (проект № 17-11-01387) в Институте программных систем им. А.К. Айламазяна Российской академии наук.

10.25209/2079-3316-2019-10-4-111-139

объемов визуальной информации происходит очень эффективным способом. Широко признанной теорией, предлагающей математическую модель зрения, является теория масштабируемых пространств (scale-space theory), впервые предложенная Т. Иидзимой [?Iijima] и Ж. Кондеринком [?Koenderink]. Эта теория в значительной степени вдохновлена свойствами ядра Гаусса и его производных как регуляризованных дифференциальных операторов, а также решений линейного уравнения диффузии. Позже эта теория была в значительной степени развита в работах Б. Ромени [?BartBook] применительно к проектированию систем технического зрения и задачам обработки цифровых изображений. В теории масштабируемых пространств биологическая зрительная система представляется как «геометрическая машина». На таком понимании основана модель зрения и в данной работе.

Основываясь на исследованиях в области нейрофизиологии зрительной системы млекопитающих, приводится математическая модель, описывающая процесс восприятия изображения на первых этапах обработки (биполярными и ганглиозными клетками сетчатки глаза, клетками латерального коленчатого тела LGN (lateral geniculate nucleus) и простыми клетками первичной зрительной коры головного мозга V1 (primary visual cortex), см. более подробно в разделе **??**. Рецептивным полем (receptive field) сенсорного нейрона называется участок с рецепторами, которые при воздействии на них определённого стимула приводят к изменению возбуждения этого нейрона. Согласно исследованиям [?DeAngelis, ?Jones], рецептивные поля ганглиозных клеток и нейронов LGN хорошо совпадают с профилями фильтров, основанных на ядре Гаусса и гауссовых производных, а рецептивные поля нейронов первичной зрительной коры — с профилями фильтров Габора. Подробнее см. в разделе **??**.

Моделирование работы первичной зрительной коры при обработке цифровых изображений приводит к задаче определения углов наклона контуров (ориентаций) на изображении. Ориентационный анализ изображений — широко распространенная задача компьютерного зрения. Она возникает при восстановлении поврежденных контуров и при поиске выделяющихся кривых на изображениях. Ориентационный анализ применяется в обработке медицинских изображений. Как пример, в работах Ремко Дайтса, см. например [?Duits2007], предложен подход к обработке изображений, основанный на поднятии изображения в расширенное пространство позиций и направлений

(invertible orientation scores). На основе этого подхода создано множество эффективных методов обработки изображений: улучшение, сегментация, восстановление, поиск объектов.

Математическая модель первичной зрительной коры, как субримановой структуры в пространстве позиций и направлений была предложена в работах Ж. Петито [?Petitot_B]. Затем эта модель была уточнена в работах Дж. Читти и А. Сарти [?Citti_Sarti_B]. Согласно этой модели процесс дополнения контуров объекта, скрытых от наблюдения, происходит путем минимизации энергии возбуждения нейронов, воспринимающих визуальную информацию в областях повреждения. Такой процесс моделируется действием оператора гипоэллиптической диффузии, изученного в работах У. Боскаина, Ж.П. Готье, Р. Чертовских и А. Ремизова [?Gauthier]. Результирующие кривые являются субримановыми кратчайшими. Применение субримановых кратчайших в обработке цифровых изображений дает эффективные методы восстановления изображений [?OptimalInpainting] и поиска выделяющихся кривых [?SIAM].

Первые этапы работы зрительной системы человека, от сетчатки до первичной зрительной коры, обширно изучены в литературе, и существуют многочисленные компьютерные модели, которые характеризуют анатомию зрительного анализатора и моделируют большинство биофизических функций. Существуют модели, способные очень детально воспроизвести конкретный физиологический эксперимент [?Amthor, ?Berry, ?Torre, ?Wilke], и модели, которые нацелены на имитацию первых этапов зрения в целом [?Herault, ?Morillas, ?Wohrer, ?Benoit]. Вычисления, выполняемые этими моделями, воспроизводят те аспекты поведения зрительной системы, для которых они были спроектированы, но не имеют возможности переконфигурирования для адаптации к новым экспериментам.

Программный комплекс (ПК) Visual Processing, описанный в этой статье, представляет собой универсальную среду моделирования, которая легко адаптируется к различным моделям зрения и предоставляет набор элементарных модулей с открытым кодом. ПК предназначен для использования в качестве эффективного инструмента моделирования и исследования механизмов зрительной системы человека.

Работа имеет следующую структуру. В разделе **??** приводятся основные сведения из нейрофизиологии зрения млекопитающих, описывающие первые этапы обработки визуальной информации.

В разделе **??** описывается математическая модель первых этапов обработки изображений зрительной системой млекопитающих и приводятся определения основных математических понятий, используемых в модели. Раздел **??** посвящен вопросам компьютерного моделирования рассматриваемой математической модели, в нем описана структура ПК Visual Processing и приведены фрагменты кода наиболее важных функций. В Заключении подводится итог проделанной работы и обсуждаются перспективы дальнейшего развития направления.

1. Архитектура зрительной системы млекопитающих

В процессе эволюции Природа создала эффективную систему восприятия визуальной информации. Зрительная система млекопитающих представляет собой совокупность структур, воспринимающих световую энергию в виде электромагнитного излучения с длиной волны 400 – 700 нм и формирующих зрительные ощущения. С помощью глаза воспринимается 80 – 90% всей информации об окружающем мире. В данном разделе приводятся основные сведения из нейрофизиологии зрения [?Physiology], описывающие первые этапы обработки зрительного сигнала.

В основе восприятия визуальной информации лежит реакция светочувствительных клеток сетчатки глаза. Лучи света, проходя через хрусталик глаза, фокусируются на сетчатке. Сетчатка глаза имеет сложную иерархическую структуру и представляет собой выделенный участок головного мозга, находящийся на периферии больших полушарий. В ней под действием световых лучей происходит активация светочувствительных рецепторов и выполняются начальные этапы обработки (кодирования) изображения для передачи в более высокие отделы головного мозга. Кодирование — процесс преобразования информации в условную форму (код), удобную для передачи по каналу связи. Универсальным кодом нервной системы являются нервные импульсы, которые распространяются по нервным волокнам. При этом содержание информации определяется не амплитудой импульсов, а их частотой. Передача сигнала от одной клетки к другой осуществляется с помощью химических веществ — нейромедиаторов.

Светочувствительные рецепторы сетчатки глаза человека делятся на два типа: палочки и колбочки, см. рисунок **??**. У человека насчитывается 6 – 7 млн колбочек и 110 – 125 млн палочек. Место выхода зрительного нерва из сетчатки не содержит фоторецепторов и

называется слепым пятном. Неподалеку от слепого пятна в области центральной ямки лежит участок наилучшего видения – желтое пятно (fovea), содержащее преимущественно колбочки. К периферии сетчатки число колбочек уменьшается, а число палочек возрастает.

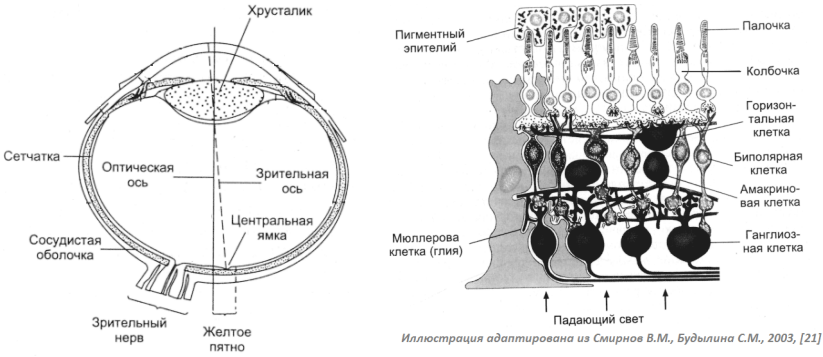


Рисунок 1. Слева: устройство глаза. Справа: схема строения сетчатки.

Палочки ответственны за ночное зрение и могут возбуждаться при очень низкой освещенности. Реакция палочек на свет осуществляется за счет содержащегося в них пигмента — родопсина. Под действием света молекула родопсина меняет свою структуру и высвобождает энергию, которая в дальнейшем используется на синтез глутаминовой кислоты — нейромедиатора, осуществляющего передачу информации в высшие участки мозга [?eye]. Колбочки же функционируют в условиях яркой освещенности и характеризуются разной чувствительностью к спектральным свойствам света (цветное зрение). В колбочках содержатся йодопсины — зрительные пигменты, ответственные за восприятия красного, зеленого и синего спектра дневного света. Механизм цветного зрения присутствует у человека и некоторых других млекопитающих.

Вследствие фотохимических процессов в фоторецепторах глаза при действии света возникает рецепторный потенциал, который представляет собой гиперполяризацию мембраны рецептора. Далее синаптические окончания фоторецепторов передают сигнал на биполярные нейроны сетчатки. Биполярные клетки разделяются на два типа: on-center и off-center. В одних биполярах на включение и выключение света возникает

медленная длительная деполяризация, а в других – на включение – гиперполяризация, на выключение – деполяризация.

Аксоны биполярных клеток в свою очередь передают сигнал на ганглиозные клетки. В результате на каждую ганглиозную клетку может передаваться сигнал с большого числа биполярных клеток, при этом, чем ближе к желтому пятну, тем меньше фоторецепторов передают сигнал на одну клетку. Именно это объясняет высокую остроту зрения в центральных отделах сетчатки. Рецептивные поля ганглиозных клеток на сетчатке имеют округлую форму, построены концентрически, каждое из них имеет возбудительный центр и тормозную периферическую зону в виде кольца. Различают рецептивные поля с on-центром (возбуждаются при освещении центра) и с off-центром (возбуждаются при затемнении центра).

В латеральных коленчатых телах (клетках LGN), куда приходят волокна из сетчатки, есть рецептивные поля, которые также имеют округлую форму, но меньше по размеру, чем в сетчатке. Благодаря работе этих клеток повышается контрастность изображения и выделяются границы объектов в зрительном поле.

Далее сигнал передается в первичную зрительную кору V1, отдел зрительного анализатора расположенный в затылочной доле. Исследование Хьюбела и Визеля [?Hubel] дало сильный прогресс в понимании принципов работы зрительной коры. В частности, они установили, что рецептивные поля нейронов зрительной коры имеют вытянутые а не округлые формы. Нейроны V1 способны выделять из цельного изображения части линий с различным расположением и ориентацией. В каждом участке коры сконцентрированы нейроны восприимчивые ко всему диапазону углов ориентаций, которые образуют гиперколонку, проходящую по глубине через все слои вертикально. Гиперколонки V1 являются детекторами угла наклона линий — ориентаций контуров объектов в поле зрения.

2. Математическое моделирование

В данном разделе описана математическая модель восприятия зрительной информации, собранная из результатов моделирования различных механизмов восприятия изображения зрительной системой человека [?BartBook, ?Wohrer, ?Benoit]. Модель основывается на знаниях из нейрофизиологии зрения, приведенных в разделе **??**. Согласно рассматриваемой модели обработка зрительного сигнала

происходит путем действия некоторого заданного фильтра. Действие фильтра на сигнал задается сверткой сигнала с некоторой функцией (ядром фильтра), обладающей схожими свойствами с рецептивными полями клеток на рассматриваемом этапе обработки. В частности, рецептивные поля биполярных клеток моделируются функцией Гаусса, см. далее пункт **??**, в то время как рецептивные поля ганглиозных клеток и нейронов LGN моделируются с помощью оператора лапласиана гауссиана, являющегося суммой вторых производных функции Гаусса по главным осям. Рецептивное поле зрительных нейронов V1 моделируется с помощью функции Габора, см. далее пункт **??**.

Моделирование биполярных клеток с помощью функции Гаусса предложено в теории масштабируемых пространств (scale-space theory) в компьютерном зрении, см. [?BartBook]. Благодаря уникальным свойствам функции Гаусса такое представление является физически обоснованным и удобным с математической точки зрения, см. [?BartFlorack].

В работе [?DeAngelis] исследованы рецептивные поля ганглиозных клеток и нейронов латерального коленчатого тела LGN. Установлено их сходство с профилем оператора лапласиана гауссиана. Этот оператор является ядром LoG фильтра, широко используемого для поиска границ объектов на изображениях в компьютерном зрении [?Shapiro].

Важным открытием нейрофизиологии зрения является геометрическая структура, соответствующая первичной зрительной коре головного мозга. Первичная зрительная кора осуществляет первичное (предшествующее всякой обработке) восприятие зрительной информации. Установлено, что для сохранения изображений первичная кора моделирует контактную структуру $\{(x, y, \theta)\} = D \times S^1$ на поверхности сетчатки глаза $D \subset \mathbb{R}^2$. Здесь угол θ соответствует наклону кривой $y(x)$ в данной точке. Таким образом, первичная зрительная кора осуществляет подъем изображения с плоскости в расширенное пространство позиций и направлений.

В работе [?Jones] показано, что рецептивные поля нейронов первичной зрительной коры V1 головного мозга точно приближаются профилями функций Габора. При этом рецептивные поля клеток, чувствительные к различным ориентациям, соответствуют различным значениям угла ориентации фильтра Габора. Таким образом, подъем изображения с поверхности сетчатки в расширенное пространство позиций и направлений моделируется действием на изображение однопараметрического семейства фильтров Габора.

Расширенное пространство позиций и направлений естественным образом имеет структуру группы Ли SE_2, группы движений евклидовой плоскости. Наличие групповой структуры позволяет эффективным образом организовать многие классические процессы обработки изображений, такие как сегментация, улучшение изображений, анизотропная диффузия и многое другое. Основные сведения о группе SE_2 приведены в пункте **??**.

2.1. Свертка функций

Свёртка функций — операция в функциональном анализе. По определению, свёртка — это математическая операция, применённая к двум функциям f и g, порождающая третью функцию, которая может рассматриваться как модифицированная версия одной из первоначальных. Это особый вид интегрального преобразования.

Операцию свертки можно интерпретировать как «схожесть» одной функции с отражённой и сдвинутой копией другой. Также, свёртка может быть описана как вес одной функции в случае, если другая функция, будучи отраженной и сдвинутой, является весовой.

Пусть $f, g : \mathbb{R}^d \to \mathbb{R}$ — две функции. Тогда их свёрткой называется функция $f * g : \mathbb{R}^d \to \mathbb{R}$, определенная формулой

$$(f * g)(x) \stackrel{\text{def}}{=} \int\limits_{\mathbb{R}^d} f(y)\, g(x-y)\, dy = \int\limits_{\mathbb{R}^d} f(x-y)\, g(y)\, dy.$$

В частности, при $d = 2$ формула принимает вид:

$$(f * g)(x, y) \stackrel{\text{def}}{=} \int\limits_{-\infty}^{\infty} \int\limits_{-\infty}^{\infty} f(\xi, \eta)\, g(x-\xi, y-\eta)\, d\xi\, d\eta.$$

2.2. Функция Гаусса и ее свойства

Немецкий ученый Карл Фридрих Гаусс — один из величайших математиков всех времен. С именем великого ученого связано большое число теорем и различных терминов в математике, физике и астрономии, к которым относится и нормальное распределение (распределение Гаусса). Это функция, записываемая формулой

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),$$

где присутствуют два вещественных параметра μ и σ: μ — коэффициент сдвига; $\sigma > 0$ — стандартное отклонение.

Нормальным распределением является распределение вероятностей (закон, описывающий область значений случайной величины и вероятности их появления), которое задается функцией плотности вероятности, для одномерного случая совпадающей с вышеназванной функцией Гаусса. Параметр μ — это среднее значение математического ожидания распределения, его медиана и мода, а σ – среднеквадратическое отклонение распределения (σ^2 – дисперсия). В природе нормальное распределение встречается часто, как и в различных приложениях. Широкое распространение нормального распределения основано на его бесконечной непрерывной делимости с конечной дисперсией. В теории вероятностей бесконечно делимым распределением называется такое распределение случайной величины, когда она может быть представлена в виде произвольного числа независимых, распределенных одинаково слагаемых.

Двумерная функция Гаусса имеет вид

$$g_\mu(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{1}{2}\left[\frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2}\right]\right),$$

где μ_x, μ_y определяют сдвиг по осям x и y; параметры σ_x, σ_y — стандартные отклонения Гауссова ядра по главным осям, определяют растянутость фильтра по осям x и y. Функция Гаусса широко используется в обработке изображений и лежит в основе многих фильтров. Отметим, что в обработке изображений, в частности в определении фильтров Габора, см. далее пункт **??**, используется функция Гаусса со значениями параметров $\mu_x = \mu_y = 0$:

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{1}{2}\left[\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right]\right). \tag{1}$$

Фильтр Гаусса, заданный ядром (**??**), используется для сглаживания (размытия) изображений (Gaussian blur). Размытие осуществляется путем свертки $I(x,y) * g(x,y)$. В теории масштабируемых пространств (scale-space theory) в компьютерном зрении, см. [?BartBook], принят подход, в соответствии с которым изображение $I(x,y)$ представляется как однопараметрическое семейство функций $J^\sigma(x,y) = I(x,y) * g^\sigma(x,y)$, где $g^\sigma(x,y)$ является семейством функций Гаусса (**??**), параметризованным значением $\sigma = \sigma_x = \sigma_y$. Параметр σ называется масштабом. Такой подход вдохновлен устройством биологических систем зрения и основан

на идее восприятия изображения путем его обозревания через проем изменяющейся величины. При этом глобальная информация о структуре изображения получается при больших значениях масштаба σ с низкой детализацией, в то время как локально в областях, содержащих важные части изображения, используется фильтр с меньшими значениями σ, обеспечивающий высокую детализацию. Такой подход обеспечивает эффективный способ представления и хранения изображений.

Помимо эффективного способа хранения, представление изображений в виде свертки с ядром Гаусса дает еще одно преимущество. Классическое представление цифровых изображений в виде матрицы дискретных значений функции яркости не позволяет напрямую определить дифференциальные операторы, действующие на изображение. В то время как благодаря свойству свертки

$$\left(\sum_{i,j} \frac{\partial}{\partial x^i \partial y^j} I(x,y)\right) * g(x,y) = I(x,y) * \left(\sum_{i,j} \frac{\partial}{\partial x^i \partial y^j} g(x,y)\right),$$

действие дифференциального оператора на изображение можно определить через производные функции Гаусса (гауссовы производные).

Гауссовы производные играют большую роль в обработке изображений. Они используются для обнаружения границ объектов и других особенностей на изображениях [?BartFlorack].

2.3. Фильтр Габора его свойства

Фильтр Габора — линейный фильтр, заданный в виде произведения тригонометрической функции и функции Гаусса. Двумерный фильтр Габора задается формулой

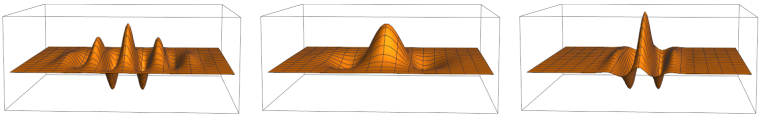
$$G(x,y,\theta) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{1}{2}\left[\frac{x_\theta^2}{\sigma_x^2} + \frac{y_\theta^2}{\sigma_y^2}\right]\right) \cos\left(\frac{2\pi x_\theta}{\lambda}\right), \tag{2}$$

где x_θ, y_θ – компоненты вектора $(x,y) \in \mathbb{R}^2$ при повороте на угол $\theta \in S^1$:

$$\begin{aligned} x_\theta &= x\cos(\theta) + y\sin(\theta), \\ y_\theta &= -x\sin(\theta) + y\cos(\theta). \end{aligned}$$

Параметры фильтра Габора имеют следующий смысл: $\lambda \in \mathbb{R}^+$ — длина волны, определяет частоту изменения знака (**??**) на заданной области (чем больше значение λ, тем длиннее интервалы знакопостоянства фильтра в направлении x_θ); а угол $\theta \in S^1$ — пространственная

направленность фильтра, определяет его ориентацию относительно осей x и y. Параметры σ_x, σ_y — стандартные отклонения гауссова ядра по осям, определяют растянутость фильтра по осям, см. рисунок **??**.



(*а*) $\sigma_x = \sigma_y = 1$, $\lambda = 1$ (*б*) $\sigma_x = \sigma_y = 1$, $\lambda = 3$ (*в*) $\sigma_x = 0.5$, $\sigma_y = 1.5$, $\lambda = 3$

Рисунок 2. Профиль фильтра Габора при фиксированном θ

2.4. Группа евклидовых движений плоскости SE_2

В математике группой G называется множество, на котором определена ассоциативная бинарная операция $\odot : G \times G \to G$, имеется нейтральный элемент $e \in G$, такой что $\forall g \in G : \; e \odot g = g \odot e = g$, и каждый элемент $g \in G$ имеет обратный $\exists g^{-1} \in G : \; g \odot g^{-1} = g^{-1} \odot g = e$.

Группой Ли называется группа, снабженная структурой гладкого многообразия, совместимой с групповыми операциями. Это означает, что операция $\odot$ и взятие обратного элемента являются гладкими.

Наиболее важные примеры групп Ли доставляют линейные группы Ли, т.е. группы линейных преобразований пространства $\mathbb{R}^n$. Пусть $X : \mathbb{R}^n \to \mathbb{R}^n$ есть линейное отображение. В фиксированном базисе $e_1, \ldots, e_n$ в $\mathbb{R}^n$ оператор X задается матрицей $X = (x_{ij})$, $i, j = 1, \ldots, n$, которую принято отождествлять с самим оператором. Таким образом, линейные группы Ли состоят из матриц.

Напомним, что движением называется преобразование, сохраняющее расстояние между точками. В данной работе рассматривается группа SE_2 евклидовых движений плоскости $\mathbb{R}^2$, состоящая из следующих матриц:

$$\mathrm{SE}_2 = \left\{ \begin{pmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{pmatrix} \;\middle|\; \theta \in S^1, \; x, y \in \mathbb{R} \right\},$$

где x, y — координаты параллельного переноса, θ - угол поворота.

В такой записи, групповая операция $\odot$ является матричным умножением, единичным элементом e является единичная матрица, и обратный элемент задается обратной матрицей.

3. Программный комплекс Visual Processing

На основе математической модели зрения человека, описанной в разделе **??**, был разработан ПК Visual Processing, реализующий функции обработки цифровых изображений, моделирующие первые этапы обработки зрительного сигнала мозгом человека: сглаживание изображений, выделение границ объектов (контуров изображения) и ориентационный анализ (определение углов наклона контуров). Эти этапы были реализованы следующим образом: обработка изображения осуществляется путем его свёртки с заданным ядром фильтра; фильтр Гаусса используется для сглаживания изображений, фильтры гауссовых производных — для выделения контуров изображений, фильтр Габора — для определения углов наклона контуров; подъем изображения на SE_2 осуществляется действием на изображение семейства фильтров Габора с различным значением угла поворота фильтра. Также были запрограммированы вспомогательные функции для представления изображений в разных форматах и нормализации изображений.

Для разработки ПК Visual Processing использовалась платформа Linux Debian 9. Вычислительные модули обработки изображений написаны на языке Си с использованием открытых библиотек libpng и libgsl. Графический интерфейс пользователя разработан на языке tcl/tk. Для работы с памятью используется стандартная библиотека malloc.

3.1. Графический интерфейс пользователя

Для удобства работы с функциями обработки изображений на языке tcl/tk был разработан графический интерфейс пользователя. Он дает пользователю возможность загрузить исходное изображение, выбрать метод обработки, означить параметры фильтра, обработать изображение в соответствии с заданной конфигурацией и сохранить результат на жесткий диск, см. рисунок **??**

3.2. Представление и нормализация изображения

ПК Visual Processing работает с изображениями в градациях серого глубиной 8 бит в формате PNG (portable network graphics). Для работы с изображениями (чтение / создание / запись) на жестком диске используется свободная библиотека libpng.

Библиотека libpng использует представление изображений в виде матрицы, элементы которой принимают целочисленные значения

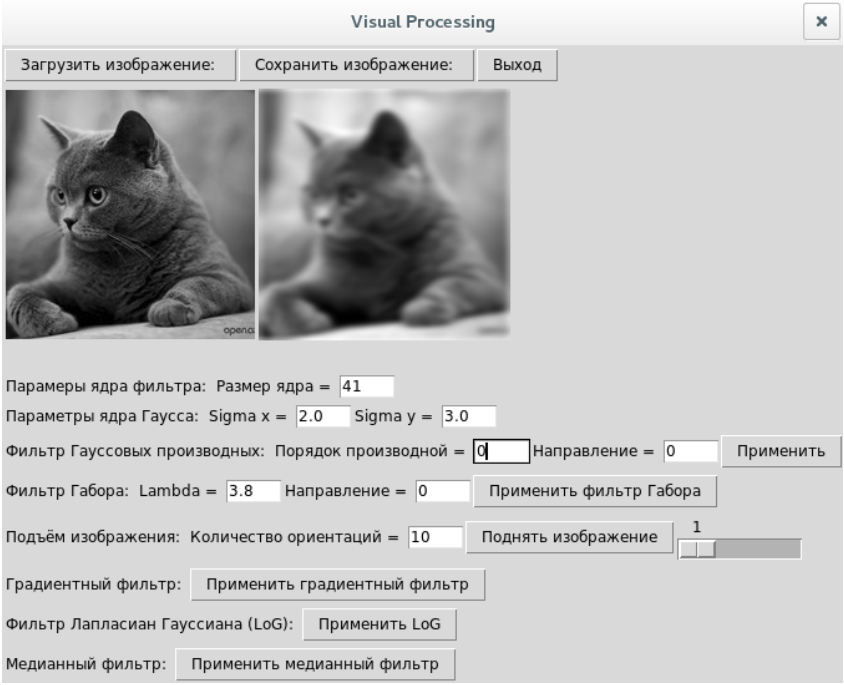


Рисунок 3. Графический интерфейс пользователя ПК Visual Processing

в отрезке [0, 255]. Для более прозрачной связи с математической моделью раздела **??** нам удобнее использовать диапазон вещественных значений из отрезка [0,1]. Для преобразования из одного представления в другое были написаны функции DtoCHAR и CHARtoD, принимающие на вход изображение Image и его размеры (ширину W и высоту H) и возвращающие изображение в другом представлении.

```
/* Преобразование изображения из~представления [0,255] в~[0,1] */
double* CHARtoD (unsigned char* Image, int W, int H){
  double* ImageN = (double*) malloc(W*H*sizeof(double));
  for (int j=0 ; j<H ; j++){ for (int i=0 ; i<W ; i++){
    ImageN[j*W + i] = ((double)(Image[j*W+i]))/255.0;}}
  return ImageN;} // Возвращает изображение в~представлении [0,1]

/* Преобразование изображения из~представления [0,1] в~[0,255] */
unsigned char* DtoCHAR (double* Image, int W, int H){
```

```
  unsigned char* ImageN = (unsigned char*) malloc(W*H*sizeof(unsigned char));
  for (int j=0 ; j<H ; j++){ for (int i=0 ; i<W ; i++){
    ImageN[j*W + i] = (unsigned char)(255.0*(Image[j*W+i]));}}
return ImageN;} // Возвращает изображение в˜представлении [0,255]
```

Функция norm нормализует значения яркости изображения в соответствии со средним значением. На вход подается изображение Image и его размеры (ширина W и высота H). В результате возвращается нормализованное изображение такое, что если наибольшее значение яркости исходного изображения отличалось от среднего значения сильнее чем наименьшее значение яркости, то максимальная яркость нормализованного изображения рана 1; иначе минимальная яркость нормализованного изображения равна 0.

```
double* norm (double* Image, int W, int H){
  double Mean; // Среднее значение яркости
  double Min = ImageD[0]; //минимальное значение
  double Max = ImageD[0]; // максимальное значение
  double L; // половина величины диапазона значений яркости
  double Sum = 0; // используется для вычисления среднего значения
  double t; // вспомогательная переменная для хранения значений яркости
  for (int j=0 ; j<H ; j++){ for (int i=0 ; i<W ; i++){
      t = ImageD[j*W+i];  if (t < Min) Min = t; if (t > Max) Max = t;
      Sum += ImageD[j*W+i];}}
  Mean = Sum/(W*H);  L = fmax(Max-Mean,Mean-Min);
  for (int j=0 ; j<H ; j++) { for (int i=0 ; i<W ; i++) {
      ImageD[j*W + i] = (ImageD[j*W + i]+L - Mean)/(2*L);}}
  return ImageD;} // Возвращает нормализованное изображение
```

3.3. Функция свёртки

Свертка изображения с ядром фильтра реализована функцией Convolve. На вход поступает изображение Image, ядро свертки Kernel, размеры изображения (ширина w и высота h) и размер ядра KS (ширина и высота ядра совпадают). Функция Convolve возвращает результат свёртки изображения Image с ядром Kernel.

```
double* Convolve (double* Image, double* Kernel, int w, int h, int KS){
  int s = KS/2; // Половина размера ядра
  double* ImageN = ImageExtend (Image, w, h, s); // Расширение изображения
  int H = h+(2*s); int W = w+(2*s); // Размеры расширенного изображения
  double* convD = (double*) malloc(H*W*sizeof(double)); //Выделение памяти
  int X = 0; int Y = 0; // Координаты пикселя расширенного изображения
```

```
  double Result = 0.0; // Значение свертки для заданного пикселя
  for (int y = s;  y < H-s; y++){ // цикл по~горизонтали изображения
    for (int x = s; x < W-s; x++){ // цикл по~вертикали изображения
      Result = 0.0;
      for (int i = KS - 1; i >= 0; i--){ // цикл по~горизонтали ядра
        for (int j = KS - 1; j >= 0; j--){ // цикл по~вертикали ядра
          X = x+(s-i); Y = y+(s-j); // Координаты текущего пикселя
          Result += ImageN[Y*W+X]*Kernel[i*KS+j];}}
          convD[y*W+x] = Result;}} // Значение свертки в~пикселе (x,y)
  double* ImageR = ImageCrop (convD, W, H, s); // Уменьшение изображения
  free(ImageN); free(convD); // Освобождение памяти
  return ImageR;} // Результат свертки изображения Image с~ядром Kernel
```

Для того, чтобы не проверять условие выхода ядра за границу изображения на каждой итерации цикла, изображение Image расширяется путём добавления пикселей по краям в количестве, равном половине размера ядра. Значение яркости в добавленных пикселях определяется зеркально относительно границы изображения. После применения к расширенному изображению функции свёртки, оно обрезается до первоначальных размеров. Такой подход позволяет существенно снизить время выполнения операции и традиционно используется в теории компьютерного зрения [?Shapiro].

Расширение изображения производится функцией ImageExtend. На вход поступает изображение Image, размеры изображения (ширина w и высота h) и величина расширения (количество пикселей s).

```
double* ImageExtend (double* Image, int w, int h, int s) {
  int H = h+(2*s); int W = w+(2*s); // Размеры расширенного изображения
  double* ImageN = (double*) malloc(H*W*sizeof(double)); //Выделение памяти
  int x, y; // Координаты пикселя
  /* Расширение изображения по~горизонтали */
  for (int r = 0;  r < h; r++){ for (int c = 0; c < w; c++){
    y = s+c; x = s+r;
    ImageN[x*W + y] = Image[r*w + c];}}
  /* Расширение изображения по~вертикали */
  for (int r = -s;  r < 0; r++){ for (int c = 0; c < w; c++){
    y = s+c; x = s+r;
    ImageN[x*W + y] = Image[(-r-1)*w + c];
    ImageN[(H - (x + 1))*W + y] = Image[(h + r)*w + c];} }
  /* Расширение изображения по~главной диагонали */
  for (int r = 0;  r < h; r++) { for (int c = -s; c < 0; c++){
    y = s+c; x = s+r;
```

```
    ImageN[x*W + y] = Image[r*w - (c+1)];
    ImageN[x*W + (W - (y + 1))] = Image[r*w + (w + (c + 1))];}}
  /* Расширение изображения по~побочной диагонали */
  for (int r = -s;  r < 0; r++){ for (int c = -s; c < 0; c++){
    y = s+c;  x = s+r;
    ImageN[x*W + y] = Image[(-r-1)*w-(c+1)];
    ImageN[x*W+(W-(y+1))] = Image[(-r-1)*w+(w+(c + 1))];
    ImageN[(H-(x+1))*W + y] = Image[(h+r)*w-(c+1)];
    ImageN[(H-(x + 1))*W + (W-(y + 1))] = Image[(h+(r+1))*w-(w+(c + 1))];}}
  return ImageN;} // Возвращает расширенное изображение
```

Уменьшение изображения производится функцией ImageCrop. На вход поступает расширенное изображение Image, размеры изображения (ширина W и высота H) и величина уменьшения (количество пикселей s). Функция ImageCrop обрезает изображение, удаляя по s пикселей от каждой из границ изображения.

```
double* ImageCrop (double* Image, int W, int H, int s) {
  int h = H - 2*s;  int w = W - 2*s;  // Размеры уменьшенного изображения
  double* ImageN = (double*) malloc(h*w*sizeof(double));//Выделение памяти
  int x, y; // Координаты пикселя
  for (int r = 0;  r < h; r++){ for (int c = 0; c < w; c++){
    y = s+c; x = s+r;  ImageN[r*w + c] = Image[x*W + y];}}
  return ImageN;} // Возвращает уменьшенное изображение
```

3.4. Фильтр Гаусса и гауссовы производные

Для создания ядра фильтра гауссовых производных используется функция GaussianDeriv. При этом производной нулевого порядка считается сама функция Гаусса. На вход поступает размер ядра KS (ширина и высота ядра совпадают), стандартные отклонения Sx и Sy гауссова ядра вдоль направлений x и y, порядок производной Order (может принимать значения 0, 1 или 2) и угол Th направления, вдоль которого считается производная. Функция GaussianDeriv возвращает ядро гауссовой производной порядка Order в направлении Th.

```
double* GaussianDeriv (int KS, double Sx, double Sy, int Order, double Th){
  double* GaussD = (double*)malloc(KS*KS*sizeof(double));//Выделение памяти
  double Sx2 = Sx*Sx; double Sy2 = Sy*Sy; // Дисперсии ядра Гаусса по~x и y
  double x, y; // Математические координаты в~[XMin,XMax]x[YMin,YMax]
  double xMin = -(KS-1)/2; double yMin = -(KS-1)/2;
  double xMax = (KS-1)/2; double yMax = (KS-1)/2;
  /* Размер шага дискретизации по~x и y */
```

```
double dx = (xMax - xMin)/KS; double dy = (yMax - yMin)/KS;
double xth; double yth; // Координаты, повернутые на~угол Th
switch (Order){ // Создает ядро Гауссовой производной порядка Order
  case 0: // Функция Гаусса (производная порядка 0)
    for (int j = 0; j < KS; j++) { y = yMin+dy/2 + j*dy;
      for (int i = 0; i < KS; i++) {x = xMin+dx/2 + i*dx;
       xth = x*cos(Th) + y*sin(Th); yth = -x*sin(Th) + y*cos(Th);
       GaussD[j*KS+i] = exp(-0.5*(xth*xth/Sx2 + yth*yth/Sy2));}}
  break; case 1: // Первая производная
    for (int j = 0; j < KS; j++) { y = yMin+dy/2 + j*dy;
      for (int i = 0; i < KS; i++) {x = xMin+dx/2 + i*dx;
        xth = x*cos(Th) + y*sin(Th); yth = -x*sin(Th) + y*cos(Th);
        GaussD[j*KS+i] =
          exp(-0.5*(xth*xth/Sx2 + yth*yth/Sy2))*(-xth/Sx2);}}
  break; case 2: // Вторая производная
    for (int j = 0; j < KS; j++) { y = yMin+dy/2 + j*dy;
      for (int i = 0; i < KS; i++) {x = xMin+dx/2 + i*dx;
        xth = x*cos(Th) + y*sin(Th); yth = -x*sin(Th) + y*cos(Th);
        GaussD[j*KS+i] =
          exp(-0.5*(xth*xth/Sx2+yth*yth/Sy2))*(xth*xth-Sx2)/(Sx2*Sx2);}}
  break; default: // Если порядок не равен 0, 1 или 2
    free(GaussD); GaussD = NULL;}
return GaussD;} //Возвращает производную порядка Order в~направлении Th
```

На рисунках **??**, **??**, **??** приведены примеры работы фильтра Гаусса, первой и второй гауссовой производной по заданным направлениям. При вычислении производных использовались значения $\sigma_x = \sigma_y = 2$.

Рисунок 4. Фильтр Гаусса. **Слева:** Исходное изображение. **Центр:** $\sigma_x = \sigma_y = 2$. **Справа:** $\sigma_x = \sigma_y = 5$

Рисунок 5. Первая гауссова производная в направлениях 0, $\frac{\pi}{2}$, π.

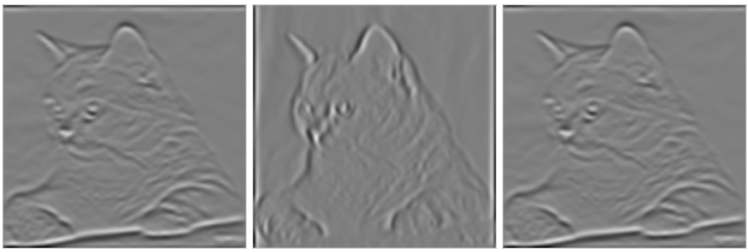


Рисунок 6. Вторая гауссова производная в направлениях 0, $\frac{\pi}{2}$, π.

3.5. Градиентный и LoG фильтры

Для реализации функций поиска границ объектов на изображении на основе аппарата гауссовых производных создан градиентный фильтр и фильтр лапласиан гауссиана (LoG фильтр). На вход подается изображение Image, его размеры (ширина W и высота H), размер ядра KS (ширина и высота ядра совпадают) и стандартные отклонения ядра Гаусса Sx и Sy по направлениям x и y.

Функция Gradient реализует градиентный фильтр.

```
double* Gradient (double* Image, int W, int H, int KS, double Sx, double Sy){
  double* ImageGradient = (double*) malloc(H * W * sizeof(double));
  double *Mx = GaussianDeriv (KS, Sx, Sy, 1, 0.0); // 1-ая производная по~x
  double *My = GaussianDeriv (KS, Sx, Sy, 1, M_PI/2); // производная по~y
  double* dx = (double*) malloc(H * W * sizeof(double));
  double* dy = (double*) malloc(H * W * sizeof(double));
  dx = Convolve(Image, Mx, W, H, KS);  dy = Convolve(Image, My, W, H, KS);
  double ddx;  double ddy; // Используется для хранения значений производных
```

```
  for (int j=0 ; j< H; j++) { for (int i=0 ; i< W; i++) {
      ddx = dx[j*W+i]; // Первая производная изображения по~x
      ddy = dy[j*W+i]; // Первая производная изображения по~y
      ImageGradient[j*W + i] = sqrt(ddx*ddx + ddy*ddy);}} // Норма градиента
return ImageGradient;} // Возвращает результат обработки градиентным фильтром
```

Функция Laplacian применяет LoG фильтр к изображению.

```
double* Laplacian (double* Image, int W, int H, int KS, double Sx, double Sy){
  double* ImageLaplacian = (double*) malloc(H * W * sizeof(double));
  double *Mx = GaussianDeriv (KS,  Sx, Sy, 2, 0.0);  // 2-ая производная по~x
  double *My = GaussianDeriv (KS,  Sx, Sy, 2, M_PI/2);  // производная по~y
  double* dx = (double*) malloc(H * W * sizeof(double));
  double* dy = (double*) malloc(H * W * sizeof(double));
  double ddx; double ddy; // Используется для хранения значений производных
  dx = Convolve(Image, Mx, W, H, KS);  dy = Convolve(Image, My, W, H, KS);
  for (int j=0 ; j< H; j++){ for (int i=0 ; i< W; i++){
    ddx = dx[j*W+i];  // Вторая производная изображения по~x
    ddy = dy[j*W+i];  // Вторая производная изображения по~y
    ImageLaplacian[j*W + i] = ddx+ddy;}} // значение лапласиана
  return ImageLaplacian;} // Возвращает результат обработки LoG фильтром
```

На рисунках **??**, **??** приведены примеры работы градиентного и LoG фильтров с указанными значениями параметров.

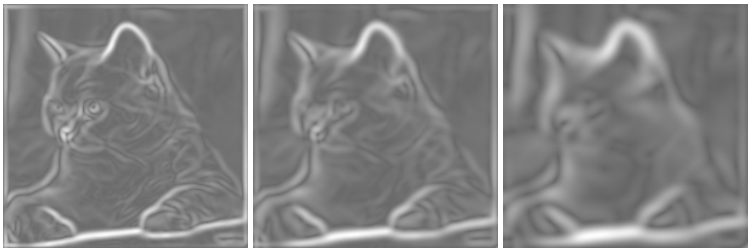


Рисунок 7. Градиентный фильтр при $\sigma_x = \sigma_y \in \{1, 2, 3\}$.

3.6. Фильтр Габора

Для создания ядра фильтра Габора используется функция Gabor. На вход поступает размер ядра KS (ширина и высота ядра совпадают), угол поворота фильтра th, стандартные отклонения Sx и Sy гауссова ядра вдоль направлений x и y и длина волны фильтра Lam. Математические функции exp, sin и cos определены в стандартной библиотеке

Рисунок 8. LoG фильтр при $\sigma_x = \sigma_y \in \{1, 2, 3\}$.

math, а константа (M_PI) определена в библиотеке GSL. Функция Gabor возвращает ядро фильтра Габора.

```
double* Gabor (int KS, double th, double Sx, double Sy, double Lam){
  double* GaborD = (double*)malloc(KS*KS*sizeof(double));//Выделение памяти
  const double Pi = M_PI; // Константа Pi
  double Sx2 = Sx*Sx; double Sy2 = Sy*Sy; // Дисперсии ядра Гаусса по~x и y
  double x, y; // Математические координаты в~[xMin,xMax]x[yMin,yMax]
  double xMin = -(KS-1)/2; double yMin = -(KS-1)/2;
  double xMax = (KS-1)/2; double yMax = (KS-1)/2;
  double Gab; // Значение функции Габора в~точке (x,y)
  /* Размер шага при дискретизации по~x и y */
  double dx = (xMax - xMin)/KS; double dy = (yMax - yMin)/KS;
  double xth, yth; //Преобразованные координаты (повернутые на~угол th)
  for (int j = 0; j < KS; j++) { y = yMin+dy/2 + j*dy;
    for (int i = 0; i < KS; i++) { x = xMin+dx/2 + i*dx;
      xth = x*cos(th) + y*sin(th); yth = -x*sin(th) + y*cos(th);
      Gab = exp(-0.5*(xth*xth/Sx2 + yth*yth/Sy2))*(cos(2*Pi*xth/Lam));
      GaborD[j*KernelSize+i]=Gab;}}
  return GaborD;} //Возвращает ядро Габора
```

На рисунке **??** приведены примеры работы фильтров Габора с указанными значениями параметров.

3.7. Функция подъёма изображения на SE_2

Подъем изображения с плоскости (x, y) на группу Ли SE_2 (в расширенное пространство позиций и направлений (x, y, θ)) производится функцией OrientationStack. На вход поступает изображение Image; его размеры (ширина W и высота H), размер ядра KS (ширина и высота ядра совпадают); количество norientation углов ориентации; стандартные отклонения Sx и Sy Гауссова ядра вдоль направлений x и

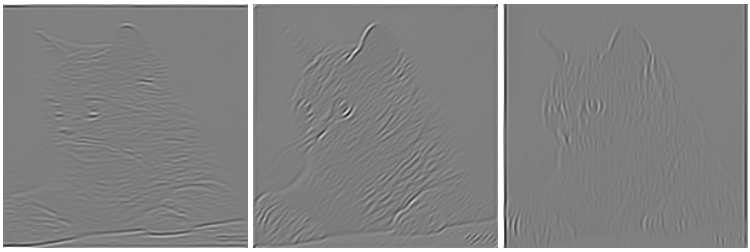


Рисунок 9. Фильтр Габора при $\sigma_x = \sigma_y = 1$, $\lambda = 4$, $\theta \in \{0, \frac{\pi}{4}, \frac{\pi}{2}\}$.

у и длина волны Lam фильтров Габора. Подъем на SE_2 осуществляется путем действия на изображения семейства фильтров Габора, чувствительных к различным углам ориентации. Функция OrientationStack возвращает массив из norientation изображений, полученных действием фильтров Габора с углом ориентации равномерно распределенным в диапазоне от 0 до 2π.

```
double** OrientationStack (double* Image, int H, int W, int KS,
              const int norientations, double Sx, double Sy, double Lam){
const double PI=M_PI; double Th;
double** OrStack = (double**) malloc(norientations*sizeof(double*));
for (int i = 0; i < norientations; i++){ Th = i*PI/norientations;
  double* OrStackN = (double*) malloc(H * W * sizeof(double));
  double* Kernel = Gabor (KS, Th, Sx, Sy, Lam); //Фильтр Габора с~углом Th
  OrStackN = (double*)Convolve(Image, Kernel, W, H, KS);
  OrStack[i] = OrStackN;} // Слой в~пространстве (x,y,th) с~заданным th=Th
return OrStack;}
```

3.8. Функции обработки изображения фильтрами

Для обработки изображений был разработан ряд функций, применяющих к изображению различные фильтры: фильтр гауссовых производных, фильтр Габора, подъем изображения на SE_2, градиентный фильтр и LoG фильтр. В качестве примера приведем код функции, применяющей градиентный фильтр к изображению.

```
int main(int argc, char *argv[]){ // Обработка градиентным фильтром
if (argc != 6) { // Проверка корректности формата входных данных
  fprintf(stderr, "Please specify input file, parametrs and output file");
  return 1;} // Выводит сообщение об ошибке при неверном кол-во параметров
  unsigned char* Image; // загружаемое с~диска исходное изображение
  int W; int H; // используются для хранения ширины и высоты изображения
```

```
int rdImg = readImage(argv[1], W, H, Image);  // считывает изображение
if (Image == NULL) {return 1;} // поверка, что изображение не пусто
/* Из~командной строки считываются значения параметров ядра */
int KS = atoi(argv[2]); // размер ядра
double Sx = (double)atof(argv[3]); // отклонение ядра Гаусса по~x
double Sy = (double)atof(argv[4]); // отклонение ядра Гаусса по~y
double* ImageD = CHARtoD(W, H, Image); // Преобразование в~формат [0,1]
double* ImageR = (double*)malloc(H *W*sizeof(double));//Выделение памяти
/* Применение градиентного фильтра с~последующей нормализацией */
ImageR = norm((Gradient (ImageD, W, H, KS, Sx,Sy)), W, H);
/* Сохранение обработанного изображения на~диск */
writeImage(argv[5], W, H,  DtoCHAR(W, H, ImageR), (char*) "Gradient");
free(ImageD); free(ImageR); // Освобождение памяти
return 0;} // Завершение программы в~штатном режиме
```

Заключение

В данной статье рассмотрена математическая модель обработки визуальной информации зрительной системой человека. На основе теории масштабируемых пространств (scale-space theory) охарактеризованы первые этапы обработки зрительного сигнала: обработка изображения биполярными клетками сетчатки моделируется путем свертки входного сигнала с фильтром Гаусса; обработка ганглиозными клетками и нейронами латерального коленчатого тела моделируется LoG фильтром, основанным на вычислении гауссовых производных порядка 2; работа клеток первичной зрительной коры по выделению ориентации контуров объектов на изображении моделируется фильтрами Габора. В рассмотренной модели ориентационные гиперколонки V1 осуществляют подъем изображения с плоскости сетчатки в расширенное пространство позиций и направлений (группу Ли SE_2) путем действия семейства фильтров Габора.

На основе математической модели была разработана компьютерная модель — ПК Visual Processing, реализующая функции обработки цифровых изображений методами, основанными на принципах работы зрительной системы человека на ранних этапах обработки визуальной информации. ПК Visual Processing выполняет следующие функции: размытие изображений по Гауссу; дифференцирование изображения с помощью гауссовых производных, выделение границ объектов с помощью лапласиана гауссиана (LoG фильтра); определение направления контуров и подъем изображения на группу Ли SE_2 с помощью фильтров

Габора. ПК основан на принципах открытой архитектуры и является платформой для реализации и тестирования алгоритмов обработки изображений в сфере математического моделирования зрения.

Логическим продолжением работы является расширение математической модели до более поздних этапов обработки зрительного сигнала — моделирование процессов в первичной зрительной коре: восстановление участков изображения, скрытых от наблюдения, и анизотропная диффузия (неравномерное размытие вдоль выделенных направлений), см. [?MarMas17]. В дальнейшем планируется расширение ПК Visual Processing путем включения соответствующих функций восстановления и анизотропной диффузии для цифровых изображений.

Список литературы

[1] T. Iijima. “Basic theory on normalization of a pattern (in case of typical one-dimensional pattern)”, *Bulletin of Electrical Laboratory*, **26** (1962), pp. 368–388 (in Japanese). ↑

[2] J.J. Koenderink. “The structure of images”, *Biological Cybernetics*, **50**:5 (1984), pp. 363–370. ↑

[3] B. M. ter Haar Romeny, *Front-end vision and multi-scale image analysis. Multi-scale computer vision theory and applications, written in mathematics*, Computational Imaging and Vision, vol. **27**, Springer, Dordrecht, 2003, ISBN 978-1-4020-1503-8. ↑

[4] G.C. DeAngelis, I. Ohzawa, R.D. Freeman. “Receptive-field dynamics in the central visual pathways”, *Trends Neurosciences*, **18**:10 (1995), pp. 451–458. ↑

[5] J.P. Jones, L.A. Palmer. “An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex”, *Journal of Neurophysiology*, **58**:6 (1987), pp. 1233–1258. ↑

[6] R. Duits, M. Felsberg, G. Granlund, B. Romeny,. “Image analysis and reconstruction using a wavelet transform constructed from a reducible representation of the Euclidean motion group”, *International Journal of Computer Vision*, **72**:1 (2007), pp. 79–102. ↑

[7] J. Petitot. “The neurogeometry of pinwheels as a sub-Riemannian contact structure”, *Journal of Physiology-Paris*, **97**:2–3 (2003), pp. 265–309. ↑

[8] G. Citti, A. Sarti. “A cortical based model of perceptual completion in the roto-translation space”, *Journal of Mathematical Imaging and Vision*, **24**:3 (2006), pp. 307–326. ↑

[9] U. Boscain, J.-P. Gauthier, R. Chertovskih, A. Remizov. “Hypoelliptic diffusion and human vision: a semidiscrete new twist”, *SIAM J. Imaging Sci.*, **7**:2 (2014), pp. 669–695. ↑

[10] A. P. Mashtakov, A. A. Ardentov, Yu. L. Sachkov. “Parallel algorithm and software for image inpainting via sub-Riemannian minimizers on the group of rototranslations”, *Numerical Mathematics: Theory, Methods and Applications*, **6**:1 (2013), pp. 95–115. ↑

[11] E. J. Bekkers, R. Duits, A. Mashtakov, G. R. Sanguinetti. “A PDE Approach to Data-driven Sub-Riemannian Geodesics in $SE(2)$”, *SIAM Journal on Imaging Sciences*, **8**:4 (2015), pp. 2740–2770. ↑

[12] F. R. Amthor, N. M. Grzywacz. “Nonlinearity of the inhibition underlying retinal directional selectivity”, *Visual neuroscience*, **6**:3 (1991), pp. 197–206. ↑

[13] M.J. Berry, I.H. Brivanlou, T.A. Jordan, M. Meister. “Anticipation of moving stimuli by the retina”, *Nature*, **398**:6725 (1999), pp. 334–338. ↑

[14] V. Torre, T. Poggio. “A synaptic mechanism possibly underlying directional selectivity to motion”, *Proceedings of the Royal Society of London. Series B. Biological Sciences*, **202**:1148 (1978), pp. 409–416. ↑

[15] S.D. Wilke, A. Thiel, C.W. Eurich, M. Greschner, M. Bongard, J. Ammermüller, H. Schwegler. “Population coding of motion patterns in the early visual system”, *Journal of Comparative Physiology*, **187**:7 (2001), pp. 549–558. ↑

[16] J. Hérault, B. Durette. “Modeling visual perception for image processing”, *IWANN 2007: Computational and Ambient Intelligence*, Lecture Notes in Computer Science, vol. **4507**, Springer, Berlin–Heidelberg, 2007, ISBN 978-3-540-73006-4, pp. 662–675. ↑

[17] C.A. Morillas, S.F. Romero, A. Martínez, F.J. Pelayo, E. Ros, E. Fernández. “A design framework to model retinas”, *Biosystems*, **87**:2–3 (2007), pp. 156–163. ↑

[18] A. Wohrer, P. Kornprobst. “Virtual retina: a biological retina model and simulator, with contrast gain control”, *J. Comput. Neurosci.*, **26**:2 (2009), pp. 219–249. ↑

[19] A. Benoit, A. Caplier, B. Durette, J. Herault. “Using Human Visual System modeling for bio-inspired low level image processing”, *Computer Vision and Image Understanding*, **114**:7 (2010), pp. 758–773. ↑

[20] P. Martínez-Cañada, C. Morillas, J.L. Nieves, B. Pino, F. Pelayo. “First stage of a Human Visual System simulator: the retina”, *CCIW 2015: Computational Color Imaging*, Lecture Notes in Computer Science, vol. **9016**, Springer, Cham, 2015, ISBN 978-3-319-15978-2, pp. 118–127. ↑

[21] В. М. Смирнов, С. М. Будылина. *Физиология сенсорных систем и высшая нервная деятельность*, Учеб. пособие для студ. высш. учеб. заведений, 4-е издание, стер., Издательский центр «Академия», М., 2009, ISBN 978-5-7695-5592-3, 336 с. ↑

[22] M. F. Land. *The Eye: A Very Short Introduction*, Oxford University Pres, 2014, ISBN 978-0199680306, 128 pp. ↑

[23] D.H. Hubel, T.N. Wiesel. “Receptive fiedls, binocular interaction and functional architecture in the cat’s visual cortex”, *J. Physiol.*, **160**:1 (1962), pp. 106–154. ↑

[24] B. M. ter Haar Romeny, L. M. J. Florack. “Front-end vision: a multiscale geometry engine”, *BMCV 2000: Biologically Motivated Computer Vision*, Lecture Notes in Computer Science, vol. **1811**, Springer, Berlin–Heidelberg, 2000, ISBN 978-3-540-67560-0, pp. 297–307. ↑

[25] Л. Шапиро, Дж. Стокман. *Компьютерное зрение*, Пер. с англ., БИНОМ. Лаборатория знаний, М., 2006, ISBN 5-94774-384-1, 752 с. ↑

[26] V. Markasheva, A. Mashtakov. “Existence of global fundamental solution to a class of Fokker–Planck equations”, *Program Systems: Theory and Applications*, **8**:4 (2017), pp. 149–162. ↑

Поступила в редакцию	26.11.2019
Переработана	10.12.2019
Опубликована	12.12.2019

Рекомендовал к публикации *д.ф.-м.н. Ю. Л. Сачков*

Пример ссылки на эту публикацию:

А. П. Маштаков, К. А. Путинцева. «Программный комплекс обработки изображений на основе модели зрения человека». *Программные системы: теория и приложения*, 2019, **10**:4(43), с. 111–139.
10.25209/2079-3316-2019-10-4-111-139
http://psta.psiras.ru/read/psta2019_4_111-139.pdf

Об авторах:

Алексей Павлович Маштаков

С.н.с. Исследовательского центра процессов управления ИПС им. А. К. Айламазяна РАН; к.т.н, тема диссертации "Математическое и программное обеспечение задач управления в робототехнике с приложением к машинной графике". Стажировался в Техническом университете Эйндховена, Нидерланды, и Болонском университете, Италия. Направления научных интересов: субриманова геометрия, геометрическая теория управления, оптимальное управление, инвариантные системы на группах Ли, приложения в робототехнике и обработке изображений, моделирование зрительной системы человека.

0000-0002-6378-3845
e-mail: alexey.mashtakov@gmail.com

Ксения Александровна Путинцева

студетка заочного факультета Ярославского Технического университета по специальности "Информационные системы"

0000-0001-6242-865X
e-mail: xenia.putintseva2015@yandex.ru

CSCSTI 27.35.43,28.23.15
UDC 004.94:591.484

Alexey P. Mashtakov, Kseniya A. Putintseva. *Image Processing Toolkit Inspired by Mechanisms of Human Visual Perception.*

Abstract. We consider a problem of computer simulation of human vision. We develop software for image processing by methods based on the principles of visual perception by humans. The software Visual Processing performs the first stages of the visual signal processing by the human brain: blurring images (removing noise), edge detection (image contours) and orientation analysis (determining the angle of inclination of the contours). These steps have been modeled by Gaussian filter to blur and represent the image, a Gaussian derivative apparatus for edge detection, and a Gabor filter for determining the orientations. Visual Processing has been developed in C, tcl / tk using the libpng and libgsl libraries. It performs the following functions: Gaussian image blur; differentiation of the image using Gaussian derivatives; edge detection using the Laplacian of Gaussian (LoG filter); determining the orientation of the contours and lift of the image to the Lie group SE_2 using Gabor filters. Visual Processing is an open-source software that aims to build a platform for implementing and testing different image processing algorithms in the field of mathematical modeling of vision.

Key words and phrases: image processing, vision model, human visual system, Gaussian derivatives, Gabor filter, roto-translation group.

2010 *Mathematics Subject Classification*: 65D19; 97N80, 97M60

10.25209/2079-3316-2019-10-4-111-139

References

[1] T. Iijima. "Basic theory on normalization of a pattern (in case of typical one-dimensional pattern)", *Bulletin of Electrical Laboratory*, **26** (1962), pp. 368–388 (in Japanese).↑

[2] J.J. Koenderink. "The structure of images", *Biological Cybernetics*, **50**:5 (1984), pp. 363–370. ↑

[3] B. M. ter Haar Romeny, *Front-end vision and multi-scale image analysis. Multi-scale computer vision theory and applications, written in mathematics*, Computational Imaging and Vision, vol. **27**, Springer, Dordrecht, 2003, ISBN 978-1-4020-1503-8. ↑

[4] G.C. DeAngelis, I. Ohzawa, R.D. Freeman. "Receptive-field dynamics in the central visual pathways", *Trends Neurosciences*, **18**:10 (1995), pp. 451–458. ↑

[5] J.P. Jones, L.A. Palmer. "An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex", *Journal of Neurophysiology*, **58**:6 (1987), pp. 1233–1258. ↑

[6] R. Duits, M. Felsberg, G. Granlund, B. Romeny,. "Image analysis and reconstruction using a wavelet transform constructed from a reducible representation of the Euclidean motion group", *International Journal of Computer Vision*, **72**:1 (2007), pp. 79–102. ↑

[7] J. Petitot. "The neurogeometry of pinwheels as a sub-Riemannian contact structure", *Journal of Physiology-Paris*, **97**:2–3 (2003), pp. 265–309. ↑

[8] G. Citti, A. Sarti. "A cortical based model of perceptual completion in the roto-translation space", *Journal of Mathematical Imaging and Vision*, **24**:3 (2006), pp. 307–326. ↑

[9] U. Boscain, J.-P. Gauthier, R. Chertovskih, A. Remizov. "Hypoelliptic diffusion and human vision: a semidiscrete new twist", *SIAM J. Imaging Sci.*, **7**:2 (2014), pp. 669–695. ↑

[10] A. P. Mashtakov, A. A. Ardentov, Yu. L. Sachkov. "Parallel algorithm and software for image inpainting via sub-Riemannian minimizers on the group of rototranslations", *Numerical Mathematics: Theory, Methods and Applications*, **6**:1 (2013), pp. 95–115. ↑

[11] E. J. Bekkers, R. Duits, A. Mashtakov, G. R. Sanguinetti. "A PDE Approach to Data-driven Sub-Riemannian Geodesics in $SE(2)$", *SIAM Journal on Imaging Sciences*, **8**:4 (2015), pp. 2740–2770. ↑

[12] F. R. Amthor, N. M. Grzywacz. "Nonlinearity of the inhibition underlying retinal directional selectivity", *Visual neuroscience*, **6**:3 (1991), pp. 197–206. ↑

[13] M.J. Berry, I.H. Brivanlou, T.A. Jordan, M. Meister. "Anticipation of moving stimuli by the retina", *Nature*, **398**:6725 (1999), pp. 334–338. ↑

[14] V. Torre, T. Poggio. "A synaptic mechanism possibly underlying directional selectivity to motion", *Proceedings of the Royal Society of London. Series B. Biological Sciences*, **202**:1148 (1978), pp. 409–416. ↑

[15] S.D. Wilke, A. Thiel, C.W. Eurich, M. Greschner, M. Bongard, J. Ammermüller, H. Schwegler. "Population coding of motion patterns in the early visual system", *Journal of Comparative Physiology*, **187**:7 (2001), pp. 549–558. ↑

[16] J. Hérault, B. Durette. "Modeling visual perception for image processing", *IWANN 2007: Computational and Ambient Intelligence*, Lecture Notes in Computer

Science, vol. **4507**, Springer, Berlin–Heidelberg, 2007, ISBN 978-3-540-73006-4, pp. 662–675. ↑

[17] C.A. Morillas, S.F. Romero, A. Martínez, F.J. Pelayo, E. Ros, E. Fernández. "A design framework to model retinas", *Biosystems*, **87**:2–3 (2007), pp. 156–163. ↑

[18] A. Wohrer, P. Kornprobst. "Virtual retina: a biological retina model and simulator, with contrast gain control", *J. Comput. Neurosci.*, **26**:2 (2009), pp. 219–249. ↑

[19] A. Benoit, A. Caplier, B. Durette, J. Herault. "Using Human Visual System modeling for bio-inspired low level image processing", *Computer Vision and Image Understanding*, **114**:7 (2010), pp. 758–773. ↑

[20] P. Martínez-Cañada, C. Morillas, J.L. Nieves, B. Pino, F. Pelayo. "First stage of a Human Visual System simulator: the retina", *CCIW 2015: Computational Color Imaging*, Lecture Notes in Computer Science, vol. **9016**, Springer, Cham, 2015, ISBN 978-3-319-15978-2, pp. 118–127. ↑

[21] V. M. Smirnov, S. M. Budylina. *Physiology sensory systems higher nervous activity*, Ucheb. posobiye dlya stud. vyssh. ucheb. zavedeniy, 4-ye izdaniye, ster., Izdatel'skiy tsentr "Akademiya", M., 2009, ISBN 978-5-7695-5592-3 (in Russian), 336 pp. ↑

[22] M. F. Land. *The Eye: A Very Short Introduction*, Oxford University Pres, 2014, ISBN 978-0199680306, 128 pp. ↑

[23] D.H. Hubel, T.N. Wiesel. "Receptive fiedls, binocular interaction and functional architecture in the cat's visual cortex", *J. Physiol.*, **160**:1 (1962), pp. 106–154. ↑

[24] B. M. ter Haar Romeny, L. M. J. Florack. "Front-end vision: a multiscale geometry engine", *BMCV 2000: Biologically Motivated Computer Vision*, Lecture Notes in Computer Science, vol. **1811**, Springer, Berlin–Heidelberg, 2000, ISBN 978-3-540-67560-0, pp. 297–307. ↑

[25] L. G. Shapiro, G. C. Stockman. *Computer Vision*, 1st ed., Pearson, 2001, ISBN 978-0130307965, 608 pp. ↑

[26] V. Markasheva, A. Mashtakov. "Existence of global fundamental solution to a class of Fokker–Planck equations", *Program Systems: Theory and Applications*, **8**:4 (2017), pp. 149–162. ↑

Sample citation of this publication:

Alexey P. Mashtakov, Kseniya A. Putintseva. "Image Processing Toolkit Inspired by Mechanisms of Human Visual Perception". *Program Systems: Theory and Applications*, 2019, **10**:4(43), pp. 111–139. (*In Russian*).

10.25209/2079-3316-2019-10-4-111-139

http://psta.psiras.ru/read/psta2019_4_111-139.pdf