Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем
Кафедра системного программирования

Карнаухов Валерий Евгеньевич

Разработка модуля анализа текстур 3D-моделей при решении задач дерматологии

Выпускная квалификационная работа

Научный руководитель:
ст. преп. Смирнов М. Н.

Научный консультант:
техн. дир. ООО "Системы КМ" Петров А. Г.

Рецензент:
инженер-программист ООО "Системы КМ" Монькин С. А.

Санкт-Петербург
2018

SAINT-PETERSBURG STATE UNIVERSITY

Software and Administration Of Information Systems
Software Engineering Department

Valerii Karnaukhov

Development of Module for Human 3D Model Texture Analysis in Dermatology

Graduation Project

Scientific supervisor:
senior lecturer Mikhail Smirnov

Scientific consultant:
CTO at CM Systems LLC Alexander Petrov

Reviewer:
software engineer at CM Systems LLC Sergei Monkin

Saint-Petersburg
2018

Оглавление

Введение

Кожный покров играет важную роль в жизни человека, являясь защитным барьером между организмом и окружающей средой. Нарушение его целостности может повлечь за собой различные воспалительные процессы. Учитывая важнейшие функции кожи (такие, как иммунная, дыхательная, терморегуляторная и другие) — невозможно оставлять без внимания ее состояние.

Мониторинг структуры кожи необходим как в составе комплексных проверок состояния здоровья человека, так и при обнаружении любых тревожащих изменений: новообразований, шелушении, изменении цвета и т.д. К сожалению, не всегда бывает достаточно только визуального изучения пациента. Для того чтобы максимально точно оценить структуру кожного покрова, во многих медицинских центрах применяют аппаратно-электронную диагностику.

Современные компьютерные оборудования способствуют более качественному наблюдению за пациентом в процессе лечения. Они позволяют изучать те параметры кожи, которые трудно исследовать при внешнем осмотре: рельеф кожи, глубину морщин, степень нарушения пигментации, размер образований и другие. Результаты медицинского обследования представляются на экране монитора, что позволяет клиенту видеть детальную информацию о текущем состоянии здоровья в режиме реального времени. Также благодаря тому, что все данные сохраняются в памяти компьютера, можно контролировать динамику изменений.

Особый интерес при анализе кожи представляют родинки. Родинки — это доброкачественные образования на коже, которые, как правило, не причиняют проблем и дискомфорта. Однако в некоторых случаях они могут стать опасными для жизни человека. Если родинка повреждена и начинает расти, менять цвет или размер, то это первый признак того, что она может переродиться в злокачественную опухоль. Применяя методы обнаружения родинок, можно определять их границы, что позволяет с помощью систем 3D-моделирования вычислять их

размеры и отслеживать любые перемены в процессе наблюдения.

В данной работе описывается разработка модуля анализа пигментации кожи по 3D-моделям для системы планирования Phoenixcas 3D Viewer[1]. В системе встроена возможность реконструкции 3D-моделей, их визуализации, симуляции хирургических операций, выполнения замеров и другие.

[1]Phoenixcas 3D, `https://www.phoenixcas.com` (дата обращения: 14.04.2018)

1. Постановка задачи

Целью данной работы является добавление в систему Phoenixcas 3D Viewer возможности анализа родинок на теле и отслеживания изменения состояния кожи в динамике по трехмерным моделям пациента. Для достижения этой цели были поставлены следующие задачи.

1. Изучить способы обнаружения родинок по изображениям.
2. Реализовать библиотеку обнаружения родинок по изображениям на основе существующих алгоритмов.
3. Разработать модуль анализа пигментации по 3D-моделям:
   - реализовать инструменты детекции родинок:
     - автоматическое выделение родинок;
     - выделение и снятие выделения родинок "вручную";
     - вычисление размеров родинок;
   - реализовать инструменты визуализации для сравнения текстур:
     - проецирование областей анализа;
     - приближение выбранных участков модели.
4. Провести апробацию модуля.

2. Обзор

2.1. Выделение границ на изображениях

Одним из фундаментальных инструментов в области компьютерного зрения является выделение границ (контуров) объектов на изображениях. Оно применяется в решении большого числа задач, направленных на анализ графических данных (например, в робототехнике для навигации роботов, медицинских системах визуализации, системах видеонаблюдения).

Выделение контуров представляет собой выявление резких локальных изменений интенсивностей пикселей на изображении, называемых разрывами [10, с. 140]. Т.е. под контуром понимается внешнее очертание объекта на изображении, вдоль которого происходят разрывы значений яркости или цвета. На практике используют два метода выделения границ: градиентный метод и метод второй производной.

2.1.1. Градиентный метод

Градиентом некоторой функции $F(x_1, ..., x_n)$ в точке M называется вектор

$$(F_{x_1}(M), ..., F_{x_n}(M)) = \left(\frac{\partial F(M)}{\partial x_1}, ..., \frac{\partial F(M)}{\partial x_n} \right),$$

компонентами которого являются частные производные функции F по всем ее аргументам, взятых в точке M. Градиент изображения $f(x, y)$ в точке $M(x_0, y_0)$ представляет собой двумерный вектор G, направленный в сторону максимального изменения яркости в точке M, модуль которого равен

$$|G| = \sqrt{G_x^2 + G_y^2},$$

где $G_x = \frac{\partial f(M)}{\partial x}$, $G_y = \frac{\partial f(M)}{\partial y}$ — координаты этого вектора [10, с. 144].

Для изображений частные производные G_x и G_y аппроксимируются разностями. Приближение этих компонент можно определить с различной степенью точности. Одним из простейших приближений частных

производных является применение центральных разностей:

$$G_x(i,j) = \frac{1}{2}(f(i+1,j) - f(i-1,j)),$$

$$G_y(i,j) = \frac{1}{2}(f(i,j+1) - f(i,j-1)).$$

Исходя из определения градиента, следует, что он отличен от нуля на протяжении всех перепадов яркости изображения, при этом выделение контуров заключается в обнаружении резких изменений интенсивностей пикселей, поэтому для критерия детекции разрывов следует использовать некоторой дополнительный метод. Существенным разрывам соответствуют большие градиентные значения, поэтому процедуру выделения границ можно проводить, сравнивая величину градиента с некоторым заранее заданным порогом. Если модуль градиента превышает пороговое значение, то считается, что в точке присутствует разрыв, иначе в данной позиции нет разрыва.

Наиболее распространенными операторами приближения градиента изображения являются операторы Робертса, Превитта, Собеля [10, с. 147].

2.1.2. Метод второй производной

Градиентный метод основан на вычислении модуля градиента, и если он выше порога, то предполагается наличие разрыва. Это приводит к обнаружению слишком большого количества разрывов [10, с. 149]. Оптимальным подходом было бы найти только точки с локальными максимумами в градиентных значениях и рассмотреть их граничные точки. Если градиент изображения имеет максимум в некоторой точке, то в этой точке вторая производная равна нулю. Таким образом, разрывы могут быть обнаружены путем нахождения точек, в которых вторая производная равна нулю, при этом равномерные области нуля игнорируются (на данных участках яркость постоянна).

Вторые производные аппроксимируются с использованием различных разностных уравнений. Один из способов приближения выглядит

следующим образом:

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial G_x}{\partial x} = f(i, j+1) - 2f(i,j) + f(i, j-1),$$

$$\frac{\partial^2 f}{\partial y^2} = \frac{\partial G_y}{\partial y} = f(i+1, j) - 2f(i,j) + f(i-1, j).$$

Примерами операторов второго порядка являются Лапласиан и вторая производная по направлению градиента [10, с. 149].

2.1.3. Использование фильтра Гаусса

Операторы второго порядка редко отдельно используются для выделения границ, поскольку на любой оператор, содержащий вторые производные, шум оказывает большее влияние, чем на оператор с первой производной [10, с. 157]. Шум — это дефект изображения, заключающийся в появлении хаотически расположенных пикселей случайного цвета и яркости на изображении. Если применить оператор второго порядка к шумному изображению, результатом станет изображение с большим количеством выделенных мелких контуров, поскольку даже очень небольшие локальные максимумы в первой производной приведут к нулевым значениям во второй производной.

Для того чтобы оператор выделения границ был менее восприимчив к шуму, предварительно перед обнаружением контуров применяется размытие изображения, обычно фильтром Гаусса. Фильтр Гаусса относится к числу сглаживающих фильтров, основное применение которых — шумоподавление.

Оператором второго порядка, использующим фильтр Гаусса на первом этапе выделения границ, является Лапласиан Гауссиана (Laplacian of Gaussian, LoG [10, с. 157]). Стоит отметить, что фильтр Гаусса может применяться не только при работе с операторами, содержащими вторые производные. Так, оператор Канни детектирует контуры с помощью градиента, при этом предварительно применяет к изображению фильтр Гаусса [10, с. 169]. Также известен фильтр Разность Гауссиан (Difference of Gaussian, DoG), который применяет к изображению два

размытия по Гауссу с разными значениями степени сглаживания и затем вычисляет разницу их результатов [15, с. 152].

2.2. Способы обнаружения родинок по изображениям

В [11] для детекции родинок используется алгоритм среднего сдвига (Mean shift [5]), который устраняет шум и выделяет участки кожи, подозрительные на родинки. Для определения, является рассматриваемый участок родинкой или нет, проверяются такие признаки, как размер участка и средняя интенсивность пикселей в нем. В [4] для поиска объектов, возможно являющихся родинками, используется фильтр DoG. Далее, выявленные кандидаты классифицируются (как родинка или нет) путем применения алгоритма машинного обучения — метода опорных векторов (Support Vector Machine, SVM [16]).

В статье [8] выделение родинок осуществляется посредством использования фильтра LoG. В статьях [7], [13] для обнаружения родинок описывается метод сопоставления с шаблоном (Template matching [14]). Для формирования шаблонов применяется LoG, в качестве функции сопоставления выступает нормализованная взаимная корреляция (Normalized Cross Correlation, NCC [18]).

В работах [2], [9], [12] цель заключается не только в обнаружении родинок, но и в распознавании определенных биометрических характеристик человека (таких, как веснушки, морщины, шрамы). Для детекции этих признаков также используется фильтр LoG.

Таким образом, основным подходом к обнаружению родинок по изображениям является применение фильтра LoG.

2.3. Системы диагностики кожи

Компания Canfield[2] предоставляет продукты (такие, как Vectra M3, Reveal Imager, Visia), которые позволяют проводить диагностику со-

[2]Canfield, `https://www.canfieldsci.com` (дата обращения: 14.04.2018)

стояния кожи по 3D-моделям: выделение и отслеживание пигментных образований, шероховатостей и неровностей кожи, симуляция косметических и хирургических процедур.

В компании Miravex[3] представлен инструмент Antera 3D, который обеспечивает качественный и количественный анализ поверхности кожи пациента: выявляет нарушения рельефа кожи, любую неравномерную окраску, степень выраженности сосудистых дефектов.

Компании DermaQuip[4], EmageMedical[5], QuantifiCare[6] предлагают технологии Pear 3D, Image Pro 3D, 3D LifeViz Mini соответственно, которые позволяют оценить различные параметры кожи с помощью 3D-моделирования, продемонстрировать клиенту обнаруженные проблемы и возможные результаты после их устранения, следить за динамикой изменений в процессе лечения.

2.4. Используемые модели и инструменты

2.4.1. Барицентрические координаты

Определение

Пусть x_1, ..., x_n — вершины некоторого симплекса в аффинном пространстве A. Тогда каждая точка $P \in A$ единственным образом может быть представлена в виде барицентрической комбинации [17]:

$$P = m_1 x_1 + ... + m_n x_n, \tag{1}$$

где вещественные числа m_1, ... m_n удовлетворяют условию:

$$\sum_{i=1}^{n} m_i = 1.$$

Коэффициенты m_1, ..., m_n называют барицентрическими координатами точки P. Центр (m_1, ..., m_n) называют барицентром симплекса.

[3]Miravex, `http://miravex.com` (дата обращения: 14.04.2018)

[4]DermaQuip, `http://dermaquip.com` (дата обращения: 14.04.2018)

[5]EmageMedical, `http://www.emagemedical.com` (дата обращения: 14.04.2018)

[6]QuantifiCare, `https://www.quantificare.com` (дата обращения: 14.04.2018)

Барицентрические координаты на плоскости

Пусть имеется некоторый треугольник ABC и P — произвольная точка в его плоскости. Тогда ее можно представить в виде барицентрической комбинации точек A, B, C:

$$P = m_1 A + m_2 B + m_3 C,$$

где коэффициенты m_1, m_2, m_3 — барицентрические координаты точки P относительно треугольника ABC.

Пусть вершины A, B, C имеют координаты $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ соответственно, а точка P — (x_0, y_0). Тогда задача нахождения барицентрических координат сводится к решению системы линейных уравнений относительно неизвестных m_1, m_2, m_3:

$$\begin{cases} m_1 + m_2 + m_3 = 1, \\ m_1 x_1 + m_2 x_2 + m_3 x_3 = x_0, \\ m_1 y_1 + m_2 y_2 + m_3 y_3 = y_0. \end{cases} \tag{2}$$

2.4.2. Алгоритмы для обнаружения родинок

Алгоритм SimpleBlobDetector

SimpleBlobDetector (SBD) является классом библиотеки компьютерного зрения OpenCV[7], который, как следует из названия, базируется на простом алгоритме выделения капель[8] на изображении. Для упрощения изложения класс SBD будем называть алгоритмом. Он состоит из следующих шагов (курсивом выделены параметры алгоритма).

1. Исходное изображение преобразуется в двоичные изображения путем применения пороговых значений с несколькими уровнями, начинающихся от *minThreshold* и заканчивающихся *maxThreshold*, с шагом *thresholdStep* между соседними порогами.

[7] OpenCV, `http://opencv.org` (дата обращения: 14.04.2018)

[8] Капля — это область, в которой определенные свойства (цвет или яркость) являются одинаковыми или приблизительно одинаковыми; все точки в капле в некотором смысле можно считать похожими друг на друга

2. В каждом бинарном изображении связанные пиксели объединяются вместе с помощью нахождения контуров и вычисляются их центры.

3. Центры из бинарных изображений группируются по координатам, и те капли, которые расположены ближе, чем $minDistBetweenBlobs$, образуют одну группу.

4. Для каждой группы найденных капель есть свой счетчик. Если значение счетчика не меньше, чем $minRepeatability$, то все капли данной группы соединяются в одну, иначе удаляются из рассмотрения.

5. Вычисляются центры и радиусы новых объединенных блоков.

SBD предоставляет несколько фильтров для капель. Приведем два основных.

- По цвету. Фильтр сравнивает интенсивность двоичного изображения в центре капли с $blobColor$. Если они отличаются, капля пропускается. Установка $blobColor = 0$ означает, что будут извлекаться темные капли, $blobColor = 255$ — будут извлекаться светлые.

- По размеру. Фильтр извлекает только те капли, площадь (в пикселях) которых лежит между $minArea$ и $maxArea$. Например, $minArea = 50$ означает, что будут пропускаться все капли, которые содержат менее 50 пикселей.

Фильтр Лапласиан Гауссиана

Для описания фильтра Лапласиана Гауссиана рассмотрим вспомогательные понятия.

Фильтр Гаусса

Фильтр Гаусса (также известный как гауссовское сглаживание) — это фильтр размытия изображения, который заменяет значение каж-

дого пикселя средневзвешенным результатом окружающих пикселей на основе нормального распределения (называемое гауссовским распределением, отсюда название) [15, с. 123]. Уравнение распределения Гаусса в двумерном случае имеет вид:

$$G(x, y; \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{3}$$

где σ — стандартное отклонение распределения. Чем больше σ, тем сильнее сглаживание.

Оператор Лапласа

Оператор Лапласа (Лапласиан) — оператор второго порядка, который используется для выделения границ на изображении [15, с. 149]. Обозначается символом Δ. Результат действия оператора Лапласа на некоторую функцию F представляет собой сумму вторых частных производных по всем ее прямоугольным декартовым координатам. В двумерном случае Лапласиан выглядит следующим образом:

$$\Delta F = F_{xx} + F_{yy} \tag{4}$$

Если Лапласиан в некоторой точке равен нулю и при переходе через нее меняет знак, то данная точка считается точкой контура.

Описание фильтра Лапласиана Гауссиана

Оператор Лапласа содержит производные второго порядка, поэтому он очень чувствителен к шуму. Для того чтобы оператор Лапласа был менее восприимчив к шуму, к изображению сначала применяется размытие по Гауссу, которое его подавляет. Такой двухступенчатый процесс называют фильтром Лапласиана Гауссиана (LoG [15, с. 157]). В дальнейшем для упрощения изложения фильтр LoG будем называть алгоритмом. Из формул (3) и (4) следует, что композиция оператора

Лапласа и функции Гаусса имеет вид:

$$LoG(x, y) = \Delta G(x, y; \sigma) = \frac{x^2 + y^2 - 2\sigma^2}{\pi\sigma^4} exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \quad (5)$$

Данный алгоритм реализован в виде функции blob_log() в рамках библиотеки scikit-image[9] — библиотеки обработки изображений для языка Python. Функция принимает следующие параметры:

- *image* — входное бинарное изображение;
- *min_sigma* — минимальное стандартное отклонение распределения Гаусса;
- *max_sigma* — максимальное стандартное отклонение распределения Гаусса;
- *num_sigma* — количество промежуточных значений стандартных отклонений между *min_sigma* и *max_sigma*;
- *threshold* — пороговое значение локальных максимумов изображения. Если интенсивность пикселя меньше *threshold*, то он пропускается;
- *overlap* — значение между 0 и 1. Если пересечение двух выделенных капель[8] перекрывается на долю, превышающую overlap, меньшая капля устраняется;
- *log_scale* — булево значение. Если установлено, то промежуточные значения стандартных отклонений интерполируются с использованием логарифмической шкалы. Если нет, используется линейная интерполяция.

2.4.3. Метрики оценки точности работы алгоритма

Метрика — это численная характеристика качества алгоритма, позволяющая оценить точность его работы. Для описания метрик введем

[9]Scikit-image, `http://scikit-image.org` (дата обращения: 14.04.2018)

понятие матрицы ошибок. Матрицей ошибок называется таблица, в которой каждый столбец представляет результаты истинного ответа, а каждая строка — результаты ответа алгоритма (или наоборот) [6]. В случае, если таблица имеет размер 2x2, ее элементы разбиваются на четыре категории (класса) в зависимости от комбинации истинного значения и ответа алгоритма (см. рис. 1).

- True Positive (TP): результат верно определен алгоритмом как положительное значение.
- False Positive (FP): результат неверно определен алгоритмом как положительное значение.
- True Negative (TN): результат верно определен алгоритмом как отрицательное значение.
- False Negative (FN): результат неверно определен алгоритмом как отрицательное значение.

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Рис. 1: Матрица ошибок

Метрика *accuracy*

Доля верных ответов алгоритма (*accuracy*) определяется как отношение числа всех истинных значений к общему числу результатов:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$

У метрики *accuracy* есть один существенный недостаток: она зависит от соотношения классов [1]. Пусть, например, значение TN сильно больше значений трех остальных классов, тогда *accuracy* всегда будет показывать высокий процент точности, даже если алгоритм не определяет ни одного истинного ответа (TP = 0). Поэтому метрику *accuracy* целесообразно применять только при сбалансированных категориях.

Метрики *precision* и *recall*

Для задач с неравноценными классами используются такие метрики качества, как точность и полнота [1]. Точность (*precision*) определяется как отношение истинно-положительных ответов алгоритма ко всем его положительным ответам:

$$precision = \frac{TP}{TP + FP}.$$

Полнота (*recall*) определяется как отношение истинно-положительных ответов алгоритма ко всем истинно-положительным значениям:

$$recall = \frac{TP}{TP + FN}.$$

F-мера

Для нахождения баланса между точностью и полнотой используется F-мера (F_1-score [1]), которая позволяет совместить их в одну величину. Она определяется как их среднее гармоническое:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}.$$

2.4.4. Система Phoenixcas 3D Viewer

Данная работа является логическим продолжением проекта по разработке программного комплекса для исследования поражений кожи на теле по 3D-моделям пациента, проводившегося в рамках курсовой работы в прошлом году [20], для системы планирования Phoenixcas 3D Viewer. В этой системе есть модули для реконструкции 3D-объектов,

их визуализации, симуляции хирургических операций, моделирования имплантатов, выполнения различных замеров и другие.

Архитектура системы

Архитектура системы Phoenixcas 3D Viewer разделена на три уровня: уровень модулей, уровень общих компонент и уровень сторонних библиотек (рис. 2).

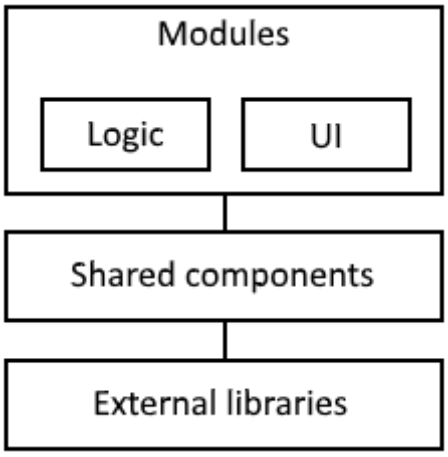


Рис. 2: Структура компонент системы Phoenixcas 3D Viewer

Модули первого уровня независимы друг от друга и выполняют определенные пользовательские задачи. Каждый из модулей состоит из двух частей: логика работы модуля и графический интерфейс. На уровне общих компонент реализованы алгоритмы, которые доступны всем модулям. На уровне сторонних библиотек подключены такие внешние библиотеки, как Ogre, Boost, OpenCV. Этот уровень закрыт от прямого доступа для модулей, работа с ним производится на уровне общих компонент.

Представление 3D-моделей

В системе Phoenixcas 3D Viewer трехмерные модели строятся с помощью пассивной фотограмметрии и представляют собой полигональные сетки. Полигональная сетка — это набор вершин, ребер и граней [3]. Множество граней, лежащих в одной плоскости, образуют полигон. В

системе в качестве полигонов выступают треугольники (рис. 3). Геометрия модели определяется списком вершин, треугольников и нормалей к ним.

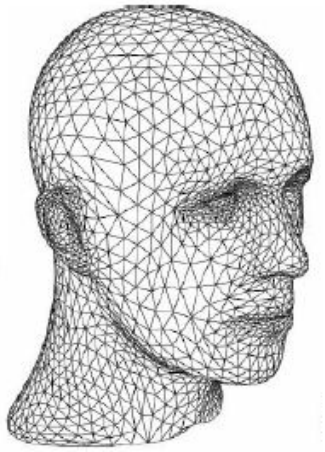


Рис. 3: Полигональная модель

Для каждой трехмерной модели задаются свои материалы. Под материалом понимается описание набора таких свойств отображения областей модели, как яркость, прозрачность, отражение, светимость. Также у каждой трехмерной модели есть текстура. Текстура — это изображение, накладываемое на поверхность модели. Существует соответствие между координатами на поверхности 3D-модели (X, Y, Z) и координатами на текстуре (U, V), которое называется UV-преобразованием, или разверткой [19].

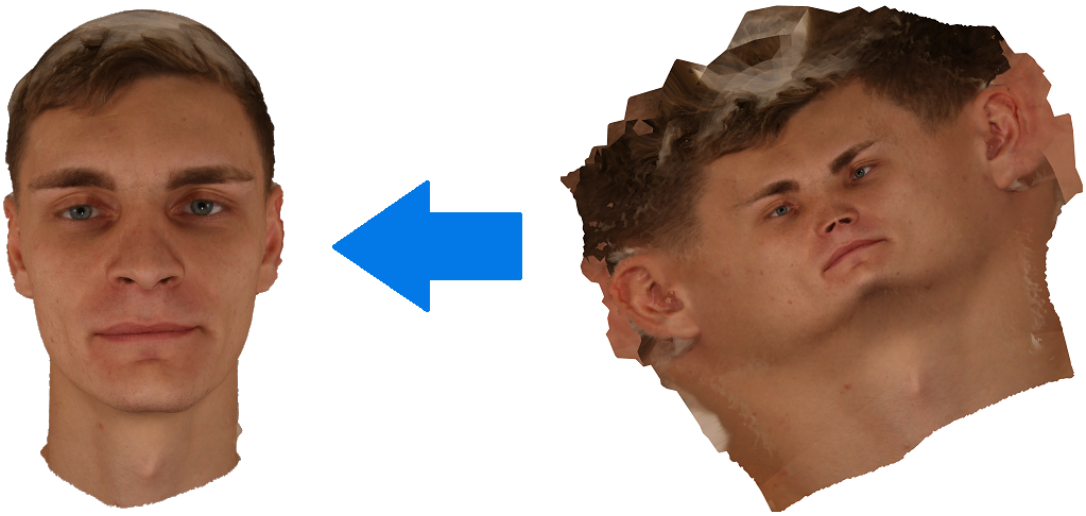


Рис. 4: Пример наложения текстуры на 3D-модель

Получение участков текстуры

В рамках курсовой работы [20] была реализована возможность получения участков текстуры для анализа. Этот процесс состоит из трех шагов.

1. Выделение областей анализа на 3D-модели (рис. 5).
2. Нахождение выбранных участков на текстуре.
3. Получение матриц пикселей (рис. 6).

Таким образом, по 3D-моделям выделяются интересующие области и по ним строятся матрицы пикселей. После этого к полученным матрицам можно применять алгоритмы обработки изображений.

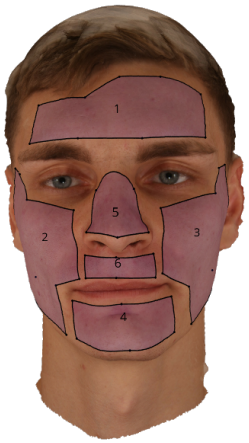


Рис. 5: Выделение областей на 3D-модели

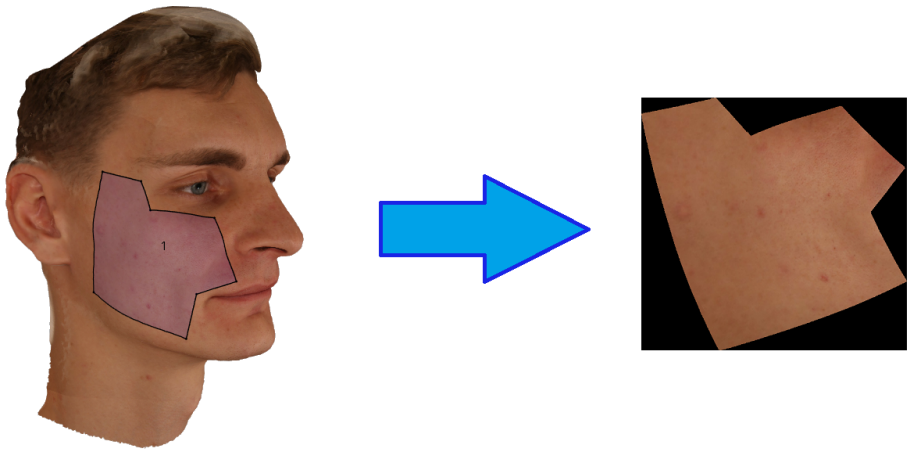


Рис. 6: Получение матрицы пикселей

Инструменты

Система Phoenixcas 3D Viewer разрабатывается в интегрированной среде Microsoft Visual Studio 2015[10]. Для визуализации трехмерной графики используется графический движок Ogre3D[11]. Логика разрабатываемого решения реализуется с помощью языков C++ и Python с использованием библиотек алгоритмов компьютерного зрения OpenCV[7] и scikit-image[9]. Графический интерфейс реализуется средствами языков JavaScript и HTML.

[10]Microsoft Visual Studio, `https://www.visualstudio.com` (дата обращения: 14.04.2018)

[11]Ogre3D, `http://www.ogre3d.org` (дата обращения: 14.04.2018)

3. Библиотека обнаружения родинок по изображениям

Библиотека содержит методы выделения границ родинок по изображениям и представляет собой объектный файл (.lib). Прототипы функций библиотеки определены в заголовочном файле (.h).

3.1. Архитектура библиотеки

Архитектура библиотеки обнаружения родинок по изображениям представлена на рис. 7.

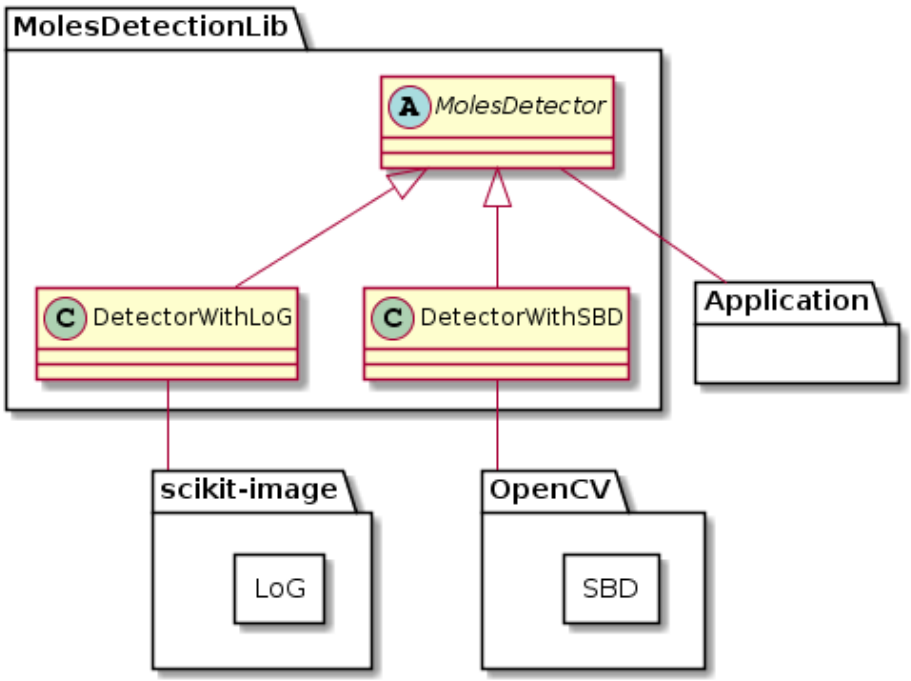


Рис. 7: Архитектура библиотеки

Абстрактный класс MolesDetector (MD) предоставляет интерфейс к методам обнаружения родинок. Производные классы DetectorWithLoG (DWL) и DetectorWithSBD (DWS) реализуют эти методы посредством применения алгоритмов LoG и SBD соответственно. Библиотека не зависит от разрабатываемого модуля и может быть использована отдельно. Есть отдельное консольное приложение, которое позволяет выделять родинки на изображении путем подключения библиотеки и ее заголовочного файла.

Поскольку логика библиотеки написана на языке C++, а LoG реализован в рамках библиотеки scikit-image на языке Python, было проведено встраивание интерпретатора Python в описываемую библиотеку. Программный интерфейс Python/C API предоставляет возможность вызывать модули Python в приложениях, написанных на C/C++. Подробно встраивание Python описано в документации по языку Python[12].

3.2. Оценки точности работы алгоритмов

Измерение оценок точности алгоритмов LoG и SBD проводилось с точек зрения обнаружения и сегментации родинок. В качестве метрики качества выступала F-мера, поскольку соотношение классов (TP, FP, TN, FN) не сбалансировано. Это связано с тем, что все участки кожи, на которых нет истинных и ложно-положительных родинок, относятся к категории TN.

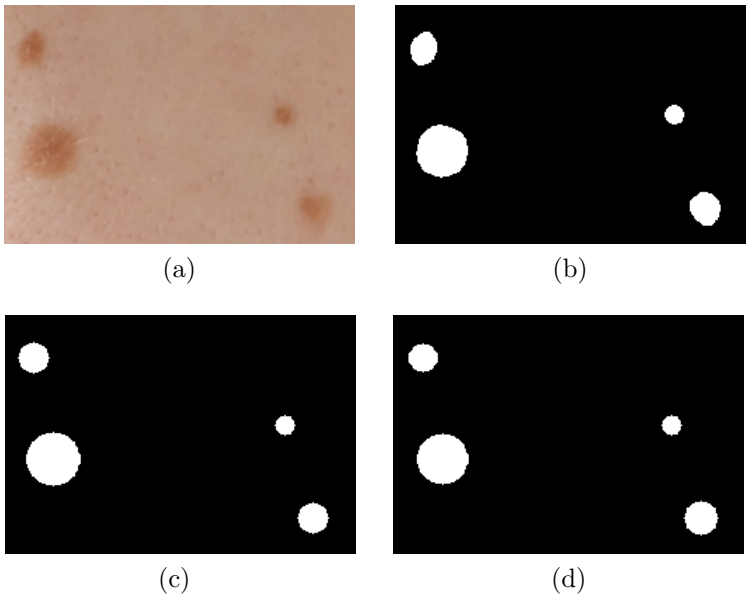
Оценивание результатов алгоритмов выполнялось на 20 изображениях, представляющих собой различные участки кожи разных людей с родинками. Для каждого изображения предварительно была создана размеченная двоичная маска, на которой вручную были закрашены все родинки. Внутри границ родинок пиксели имеют белый цвет, а вне границ — черный. Результаты алгоритмов аналогично представлялись в виде двоичных масок (рис. 8).

Основными параметрами алгоритмов LoG и SBD, влияющих на точность выделения родинок, являются пороговые значения *threshold* и *thresholdStep* соответственно, поэтому обнаружение и сегментация родинок оценивались путем варьирования данных порогов при фиксированных значениях остальных параметров алгоритмов. Эти значения были выбраны экспериментально и приведены в Таблицах 1, 2.

3.2.1. Оценка обнаружения родинок

При оценке обнаружения родинок внесение результатов алгоритма в классы TP, FP, FN определяется следующим образом: выделенная

[12]Документация по языку Python, `https://docs.python.org` (дата обращения: 14.04.2018)



(a) (b) (c) (d)

Рис. 8: Примеры представления изображений в виде двоичных масок: (a) исходное изображение, (b) маска ручного выделения родинок, (c) маска применения LoG, (d) маска применения SBD

Таблица 1: Параметры алгоритма LoG

Параметр	Значение
min_sigma	2
max_sigma	22
num_sigma	15
$overlap$	0

Таблица 2: Параметры алгоритма SBD

Параметр	Значение
$minThreshold$	45
$maxThreshold$	256
$minRepeatability$	3
$minDistBetweenBlobs$	1
$blobColor$	0
$minArea$	15

алгоритмом родинка, которая пересекает не менее 50% площади истинной родинки, считается истинно-положительной (TP), иначе — ложно-положительной (FP); количество ложно-отрицательных родинок (FN) вычисляется как разность между числом всех истинных родинок и числом истинно-положительных ответов.

На рис. 9, 10 представлена зависимость величины F-меры (F_1-score) от пороговых значений, принимаемых алгоритмами в качестве параметров. Самые высокие значения F-меры (0.964 и 0.957) алгоритмы LoG и SBD достигают при использовании порогов 0.037 и 6.0 соответственно.

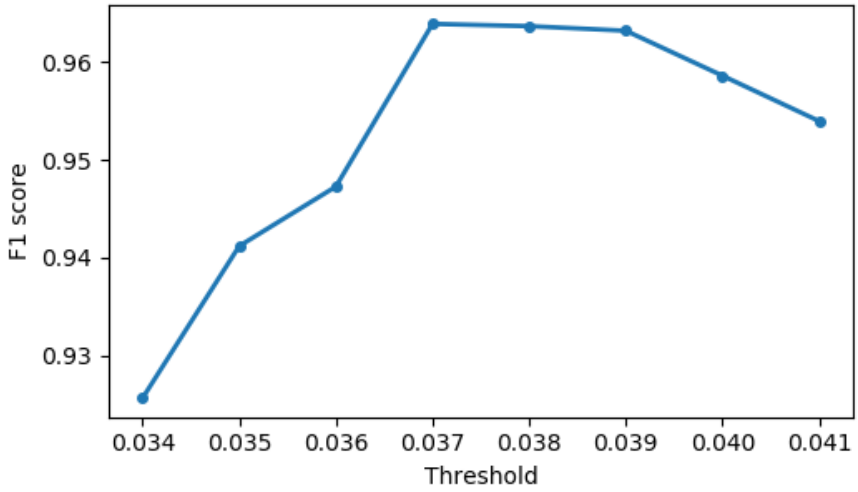


Рис. 9: Точность обнаружения алгоритма LoG при различных пороговых значениях

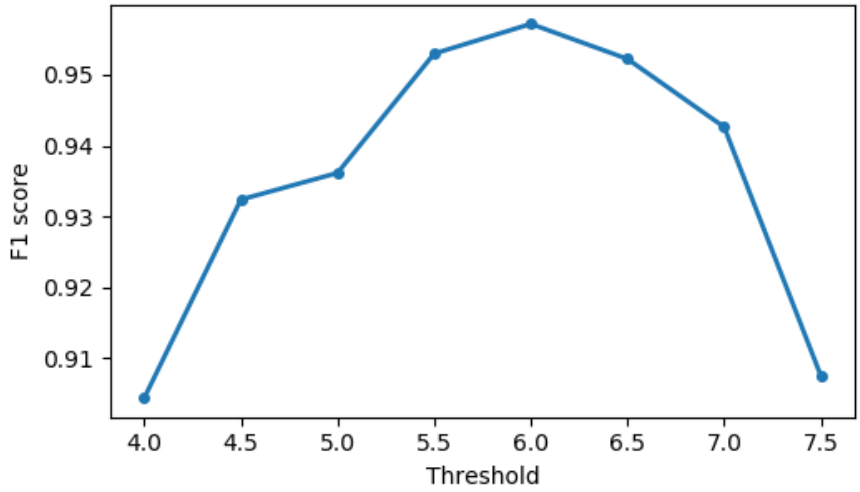


Рис. 10: Точность обнаружения алгоритма SBD при различных пороговых значениях

3.2.2. Оценка сегментации родинок

Сегментация показывает, насколько точно выделенные алгоритмом области покрыли истинные родинки. При оценке сегментации родинок отнесение ответов алгоритма к классам TP, FP, FN определяется путем прохода по всем пикселям истинной бинарной маски и сравнения их с соответствующими пикселями предсказанной бинарной маски.

На рис. 11, 12 представлена зависимость величины F-меры (F_1-score) от пороговых значений, принимаемых алгоритмами в качестве параметров. Самые высокие значения F-меры (0.856 и 0.812) алгоритмы LoG и SBD достигают при использовании порогов 0.037 и 6.0 соответственно.

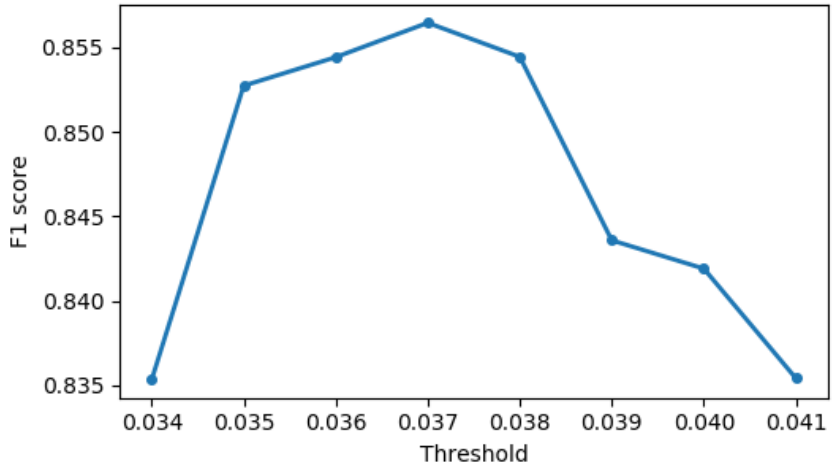


Рис. 11: Точность сегментации алгоритма LoG при различных пороговых значениях

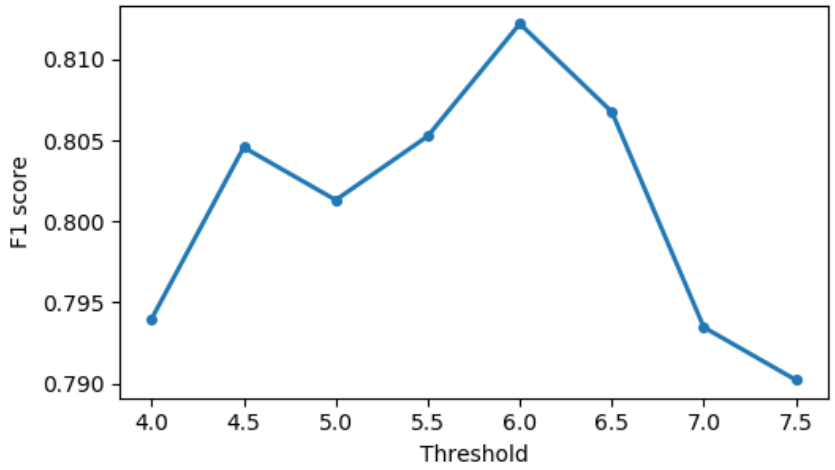


Рис. 12: Точность сегментации алгоритма SBD при различных пороговых значениях

Таким образом, алгоритмы LoG и SBD показывает наивысшую точность обнаружения и сегментации родинок при пороговых значениях 0.037 и 6.0 соответственно. В Таблице 3 представлены итоговые результаты алгоритмов, принимающих в качестве входных параметров значения, приведенные выше.

Таблица 3: Точность обнаружения и сегментации алгоритмов LoG и SBD

Алгоритм	Обнаружение, F1	Сегментация, F1
LoG	0.964	0.856
SBD	0.957	0.812

3.3. Примеры работы алгоритмов

На рис. 13, 14 представлены экспериментальные результаты алгоритмов LoG и SBD для двух изображений. Красным цветом выделены обнаруженные родинки. Слева располагается исходное изображение, по центру — результат алгоритма LoG, слева — результат алгоритма SBD.

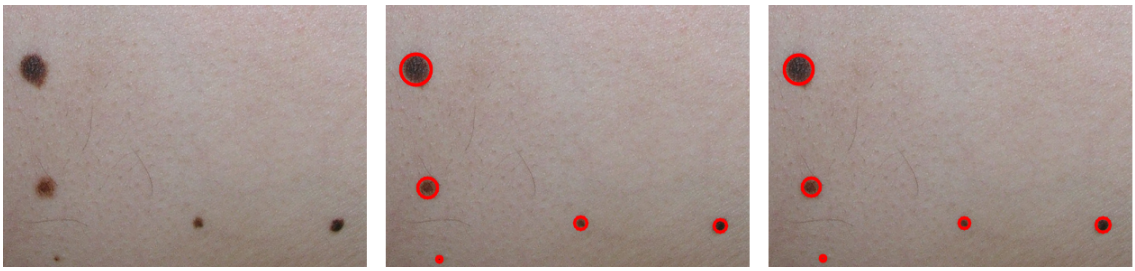


Рис. 13: Пример 1 обнаружения родинок

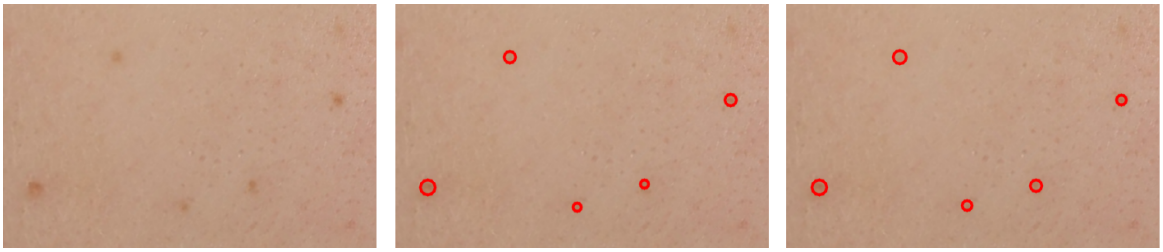


Рис. 14: Пример 2 обнаружения родинок

4. Модуль анализа пигментации по 3D-моделям

4.1. Описание модуля

В модуле реализуются графический интерфейс для ввода/вывода данных и логика обработки этих данных. Модуль предоставляет возможность автоматически выделять родинки на отмеченных участках 3D-моделей, "вручную" корректировать результаты детекции родинок, проводить визуальное сравнение текстур.

4.2. Архитектура модуля

Модуль анализа пигментации по 3D-моделям разрабатывается в системе Phoenixcas 3D Viewer на уровне модулей как динамически подключаемая библиотека. Его архитектура представлена на рис. 15. Поскольку данная работа является продолжением курсовой работы прошлого года [20], зеленым цветом выделены те классы, которые были реализованы или расширены в рамках текущей работы.

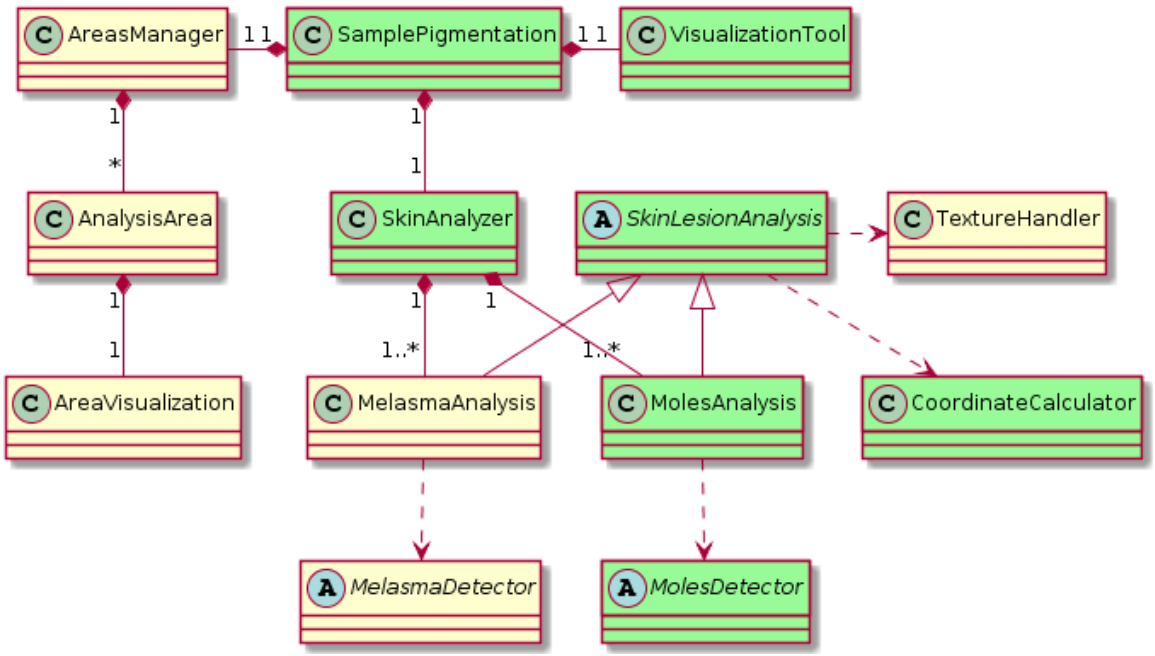


Рис. 15: Архитектура модуля

Класс SamplePigmentation осуществляет двустороннее взаимодействие между пользователем и модулем. Он обрабатывает поступающие

запросы от пользователя и вызывает методы графического интерфейса для отображения полученных результатов.

Абстрактный класс SkinLesionAnalysis определяет некоторое общее поведение инструментов анализа кожи. Конкретные классы MolesAnalysis и MelasmaAnalysis реализуют методы анализа родинок и мелазмы соответственно. Обращение к этим классам осуществляется через интерфейс класса SkinAnalyzer.

Абстрактные классы MolesDetector и MelasmaDetector предоставляют интерфейс к методам обнаружения родинок и мелазмы соответственно.

Класс CoordinateCalculator реализует методы вычисления барицентрических координат, нахождения трехмерных координат по текстурным и наоборот.

Класс VisualizationTool предоставляет инструменты визуализации для сравнения текстур: проецирование областей анализа, приближение выбранных участков модели.

Класс AreasManager содержит в себе области анализа и отвечает за их добавление, скрытие, удаление. Класс AnalysisArea предоставляет методы для построения области анализа, получения набора ее трехмерных и текстурных координат. Класс AreaVisualisation отвечает за отображение области анализа: ее цвет, границы, текст.

Класс TextureHandler предоставляет методы для работы с текстурой: получение и изменение текстуры, вычисление участков текстуры по областям анализа на 3D-модели.

4.3. Вычисление трехмерных координат точки по текстурным

Можно выделить задачи, когда требуется определить положение объекта на 3D-модели, зная только его текстурные координаты. Например, чтобы вычислить длину, площадь, объем объекта с текстуры, необходимо сначала найти его трехмерные координаты.

Для вычисления 3D-координат произвольной точки P на текстуре

был предложен следующий алгоритм.

1. Путем прохода по всем треугольникам модели извлекаются соответствующие им двумерные треугольники, и среди них ищется тот, внутри которого находится рассматриваемая точка.

2. Зная UV-координаты найденного треугольника, вычисляются барицентрические координаты m_1, m_2, m_3 точки P посредством решения системы (2).

3. Зная 3D-координаты треугольника и коэффициенты m_1, m_2, m_3, из уравнения (1) находятся координаты точки на 3D-модели.

4.4. Вычисление текстурных координат точки по трехмерным

Можно выделить задачи, когда требуется определить положение объекта на текстуре, зная только его трехмерные координаты. Например, чтобы отмеченный на 3D-модели объект нарисовать на текстуре, необходимо сначала найти его текстурные координаты.

Для вычисления UV-координат произвольной точки P на 3D-модели был предложен следующий алгоритм.

1. Путем прохода по всем треугольникам модели ищется тот, внутри которого находится рассматриваемая точка.

2. Зная 3D-координаты найденного треугольника, вычисляются барицентрические координаты m_1, m_2, m_3 точки P посредством решения системы (2).

3. Зная UV-координаты треугольника и коэффициенты m_1, m_2, m_3, из уравнения (1) находятся координаты точки на текстуре.

4.5. Инструменты детекции родинок

4.5.1. Автоматическое выделение родинок

Для автоматического выделения родинок в систему Phoenixcas 3D Viewer была добавлена библиотека обнаружения родинок по изображениям. Она подключается к модулю как статическая библиотека на уровне общих компонент. Библиотека предоставляет интерфейс для детекции родинок на участках текстуры, полученных с выделенных пользователем областей анализа на 3D-модели.

4.5.2. "Ручная" корректировка результатов детекции родинок

Поскольку обнаружение родинок осуществляется посредством применения алгоритмов компьютерного зрения, то нужно исходить из того, что 100% точного выделения не получится достичь в любом случае. Поэтому важно иметь инструменты для "ручной" корректировки результатов:

1. иметь возможность "вручную" выделять родинки на модели, которые алгоритм не обнаружил;

2. иметь возможность удалять выделение тех родинок, которые детектировались неверно или не совсем точно.

Опишем реализацию каждого из этих инструментов. В качестве иллюстрации работы инструментов выступает изображение участка манекена.

"Ручное" выделение родинок

Для того чтобы "вручную" выделить родинку на 3D-модели (рис. 16), была предложена следующая последовательность действий.

1. Пользователь задает две точки на модели.

2. Вычисляются UV-координаты этих точек по 3D.

3. Рисуется окружность на текстуре.

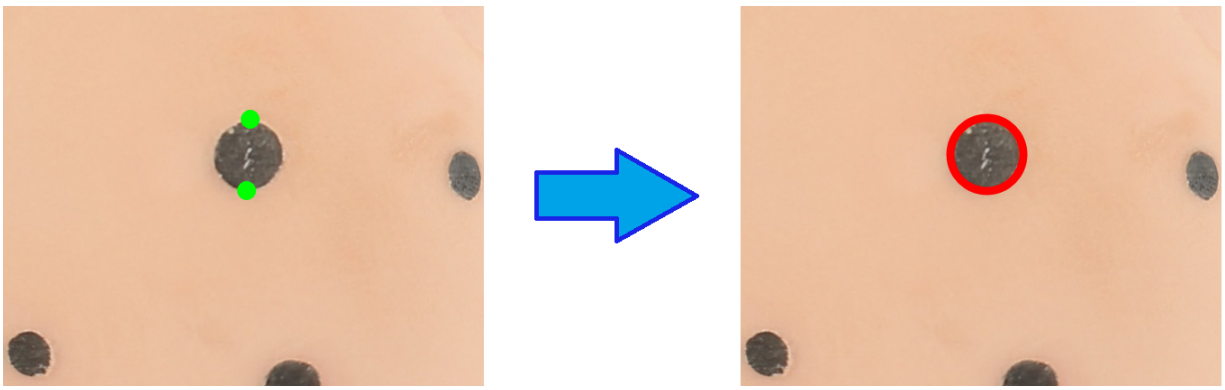


Рис. 16: ”Ручное” выделение

”Ручное” снятие выделения родинок

Для того чтобы ”вручную” снять выделение родинки на 3D-модели (рис. 17), была предложена следующая последовательность действий.

1. Пользователь задает точку на модели.
2. Путем прохода по всем 3D-координатам выделенных родинок ищется сфера, внутри которой лежит отмеченная точка.
3. Вычисляются UV-координаты сферы по 3D.
4. Перерисовывается участок изначальной текстуры на место сферы.

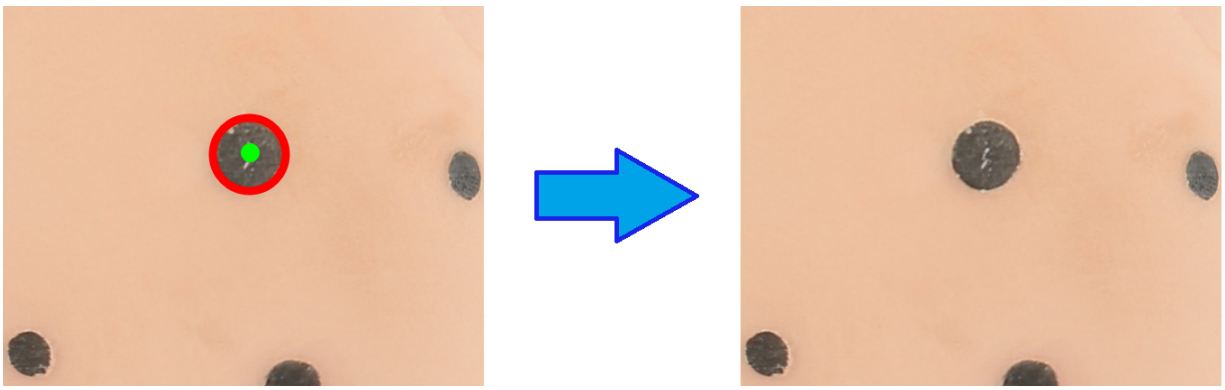


Рис. 17: ”Ручное” снятие выделения

4.5.3. Вычисление размеров родинок

Алгоритмы SBD и LoG выделяют родинки в виде окружности. Т.е. для всех родинок известны координаты их центров и радиусы в пиксе-

лях. Для того чтобы вычислить истинный радиус родинки, была предложена следующая последовательность действий.

1. Извлекаются центр и самая верхняя точка окружности.
2. Вычисляются 3D-координаты этих точек по UV.
3. Для пары точек $A(x_a, y_a, z_a), B(x_b, y_b, z_b)$ по формуле евклидова расстояния (6) находится расстояние между ними.

$$distance(A, B) = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2 + (z_b - z_a)^2} \qquad (6)$$

4.6. Инструменты визуализации для сравнения текстур

Использование 3D-моделей позволяет отслеживать изменения состояния кожи в динамике. Имея две (или больше) модели одного человека, сначала производится их регистрация (совмещение), после чего можно визуально сравнивать одни и те же участки на выбранных моделях. А благодаря "ручной" коррекции результатов можно достаточно точно судить о том, какие произошли перемены.

Для сравнения текстур 3D-моделей были реализованы проецирование областей анализа с одной модели на другую и приближение выбранных участков модели.

4.6.1. Проецирование областей анализа

Проецирование областей анализа с одной модели на другую позволяет автоматически строить точно такие же области на другой второй модели (рис. 18). Это дает возможность не тратить время на "ручное" определение областей и применять алгоритмы обработки изображений на одинаковых участках текстуры.

Для того чтобы спроецировать область анализа с одной модели на другую, была предложена следующая последовательность действий.

1. Для каждой отмеченной точки области ищется треугольник, который ее содержит, и извлекается нормаль найденного треугольника.

2. Вычисляется общая нормаль к области анализа как нормализованная сумма полученных нормалей.

3. В каждую отмеченную точку запускается луч, направленный противоположно вектору нормали области.

4. На второй модели строится область по точкам пересечения лучей с этой моделью.

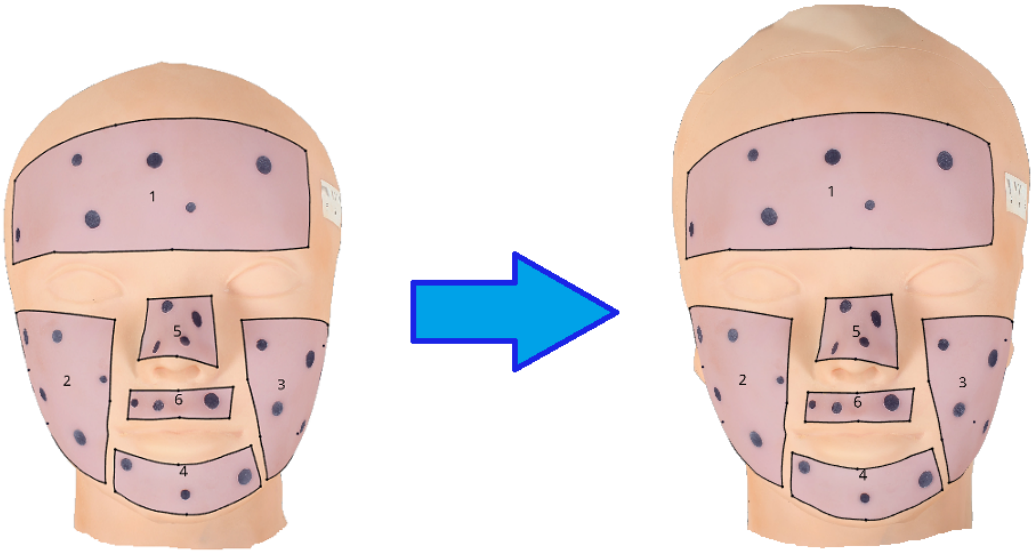


Рис. 18: Проецирование областей анализа с первой модели на вторую

4.6.2. Приближение выбранных участков модели

Приближение выбранных участков модели позволяет автоматически устанавливать близкое расположение камер к модели, направленных на интересующие области. Это дает возможность не тратить время на "ручную" настройку положения камер и детально демонстрировать указанный участок.

Чтобы автоматически задавать границы участков, было решено реализовать генерацию сетки, покрывающей модель (рис. 19). Количество строк и столбцов сетки задает пользователь.

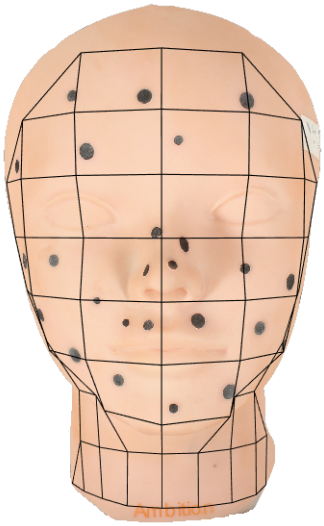


Рис. 19: Покрывающая сетка для 3D-модели

Для того чтобы приблизить выбранный участок модели (рис. 20), была предложена следующая последовательность действий.

1. Строится покрывающая сетка для модели.
2. Пользователь выбирает участок.
3. Определяются координаты данного участка и вычисляется его нормаль.
4. Вычисляется минимальный параллелепипед, ограничивающий указанный участок.
5. Находятся центр параллелепипеда и расстояние между его двумя самыми отдаленными друг от друга вершинами.
6. Извлекается угол обзора камеры.
7. Вычисляется и устанавливается положение камеры.

Приведем пояснение, как вычисляется положение камеры. На рис. 21 представлена схема поля зрения камеры. Пусть точка B является камерой, точка D — центром параллелепипеда, точки A и C — самые отдаленные друг от друга вершины параллелепипеда. Тогда задача определения координат точки B сводится к вычислению расстояния BD

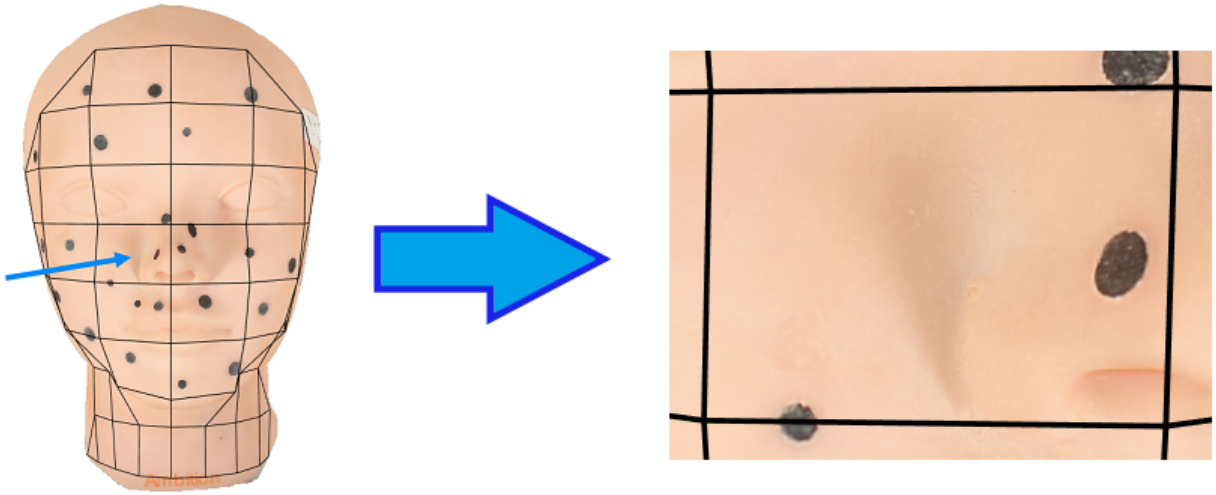


Рис. 20: Приближение выбранного участка

так, чтобы сторона AC составляла область видимости камеры. Из п.5 известно расстояние AC и координаты точки D, из п.6 известен $\angle ABC$ (обозначим его α). Исходя из этого, сторона BD вычисляется по формуле (7). Из п.3 известна нормаль выбранного участка. Тогда окончательно координаты точки B вычисляются путем откладывания от точки D величины BD в направлении вектора нормали.

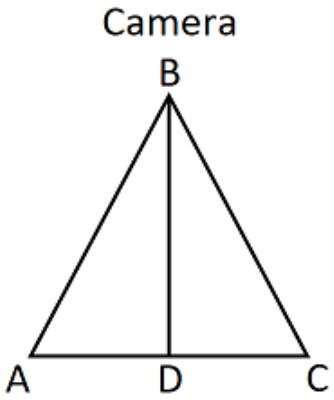


Рис. 21: Схема поля зрения камеры

$$BD = \frac{AD}{tg\frac{\alpha}{2}} \tag{7}$$

4.7. Пользовательский интерфейс

Взаимодействие с пользователем разбито на пять шагов.

1. Выбор 3D-моделей пациента и типа анализа (рис. 22).

2. Определение границ областей, участвующих в анализе (рис. 23).

3. Проецирование областей анализа с одной модели на другую и построение покрывающей сетки (рис. 24).

4. Отображение автоматически обнаруженных родинок и "ручная" корректировка результатов детекции (рис. 25).

5. Отображение выделенных родинок и их размеров (рис. 26).

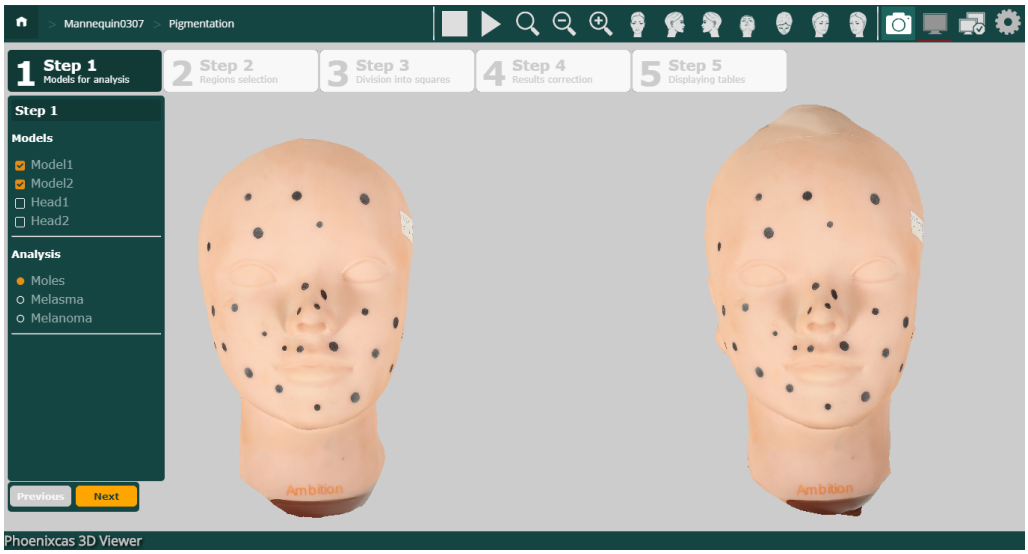


Рис. 22: Пользовательский интерфейс. Первый шаг

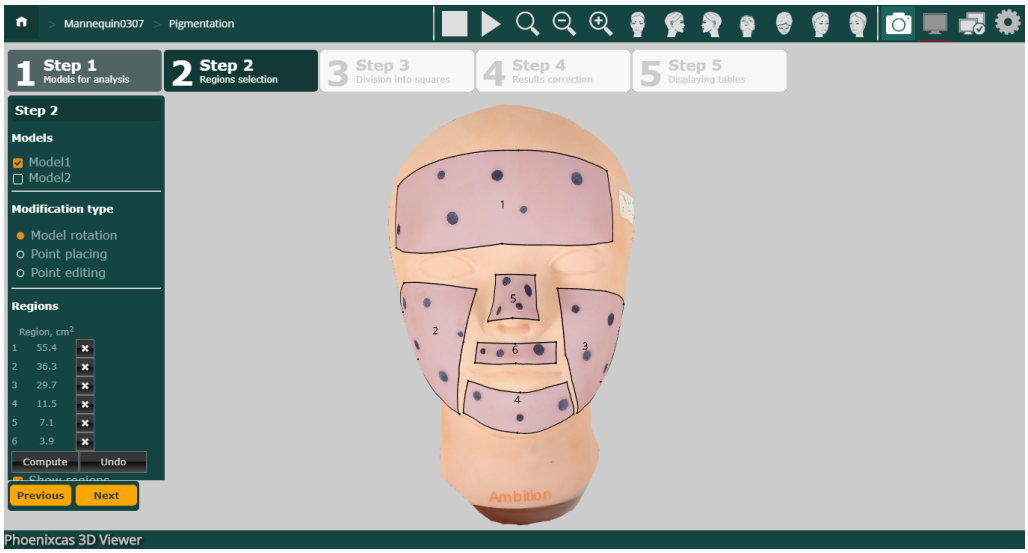


Рис. 23: Пользовательский интерфейс. Второй шаг

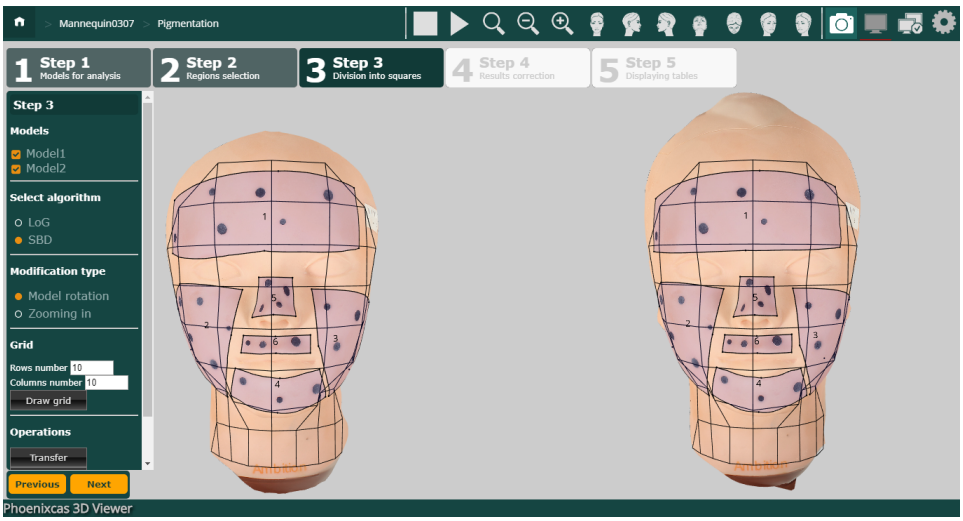


Рис. 24: Пользовательский интерфейс. Третий шаг

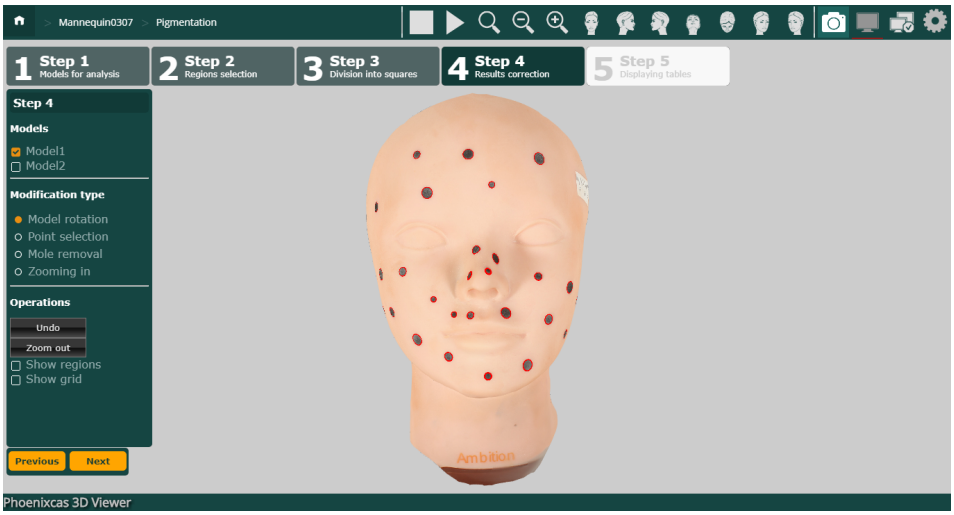


Рис. 25: Пользовательский интерфейс. Четвертый шаг

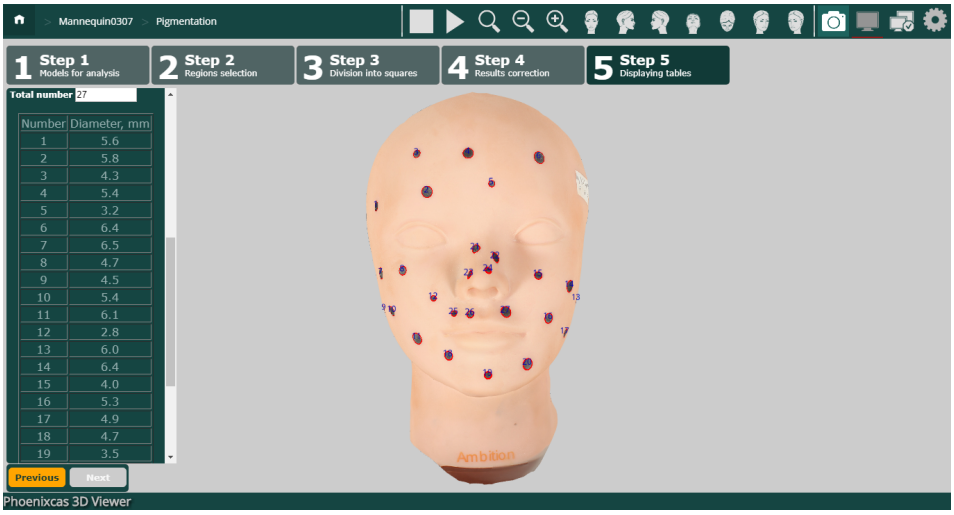


Рис. 26: Пользовательский интерфейс. Пятый шаг

5. Апробация модуля

Для проверки работоспособности модуля использовались трехмерные модели манекена и пациентов. Для симуляции родинок на манекене были вырезаны из бумаги черные круги различных размеров и наклеены на него. На рис. 26 представлено обнаружение "родинок" на манекене. Каждому выделенному кругу соответствует его порядковый номер, слева отображается таблица, в которой для каждого номера указывается диаметр круга в миллиметрах. Рис. 16, 17 иллюстрируют выделение и снятие выделения "вручную".

Для испытания инструмента проецирования области анализа была отдельно реконструирована вторая модель манекена. На рис. 18, 24 на левой модели изображены области анализа, отмеченные пользователем, на модели справа показано автоматическое проецирование данных областей. Результат приближения выбранного участка манекена представлен на рис. 20.

Также с целью проверки работоспособности модуля были приглашены пять пациентов из клиники, в которой планируется использование разработанного модуля. Для каждого пациента были произведены съемка и анализ родинок. Результаты работы были отправлены профессору из клиники для получения обратной связи по точности обнаружения и пользовательскому интерфейсу.

Примеры работы модуля с двумя моделями человека, построенных в разные промежутки времени, представлены на рис. 27 — рис. 29. Из рис. 29 видно, что при работе с двумя моделями окно отображения разбивается на две равные части, слева и справа показываются одинаковые участки двух рассматриваемых моделей. Благодаря проецированию областей анализа обнаружение родинок автоматически происходит на одних и тех же участках, а автоматическое приближение камеры позволяет проводить детальное сравнение указанных участков.

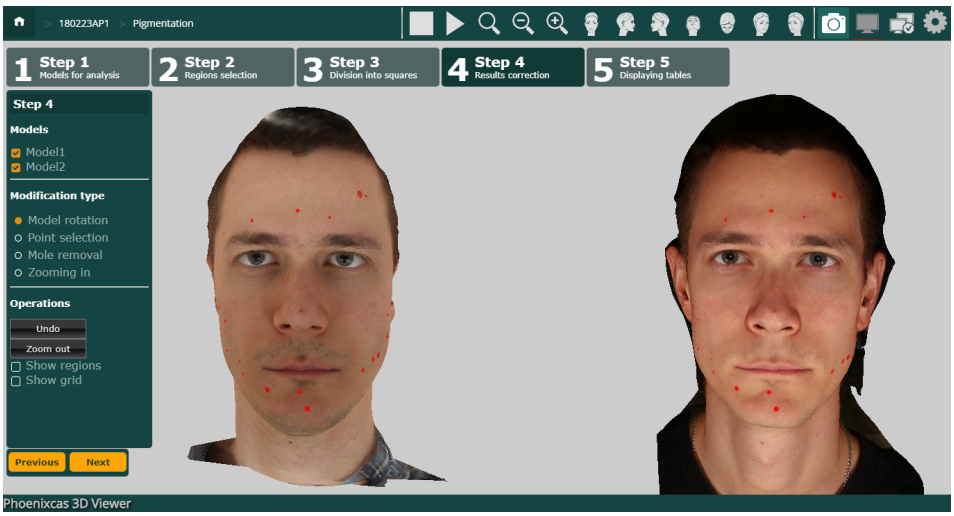


Рис. 27: Выделение родинок на двух моделях

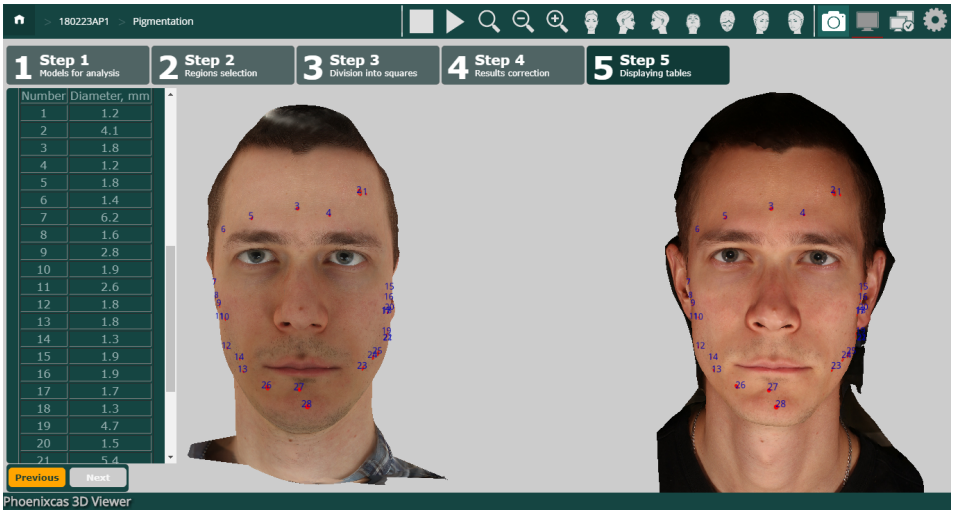


Рис. 28: Отображение размеров родинок

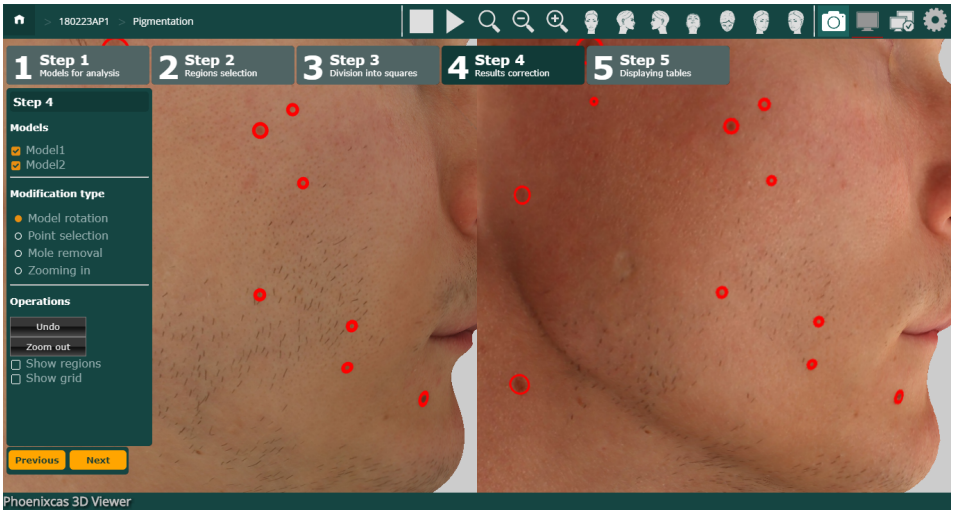


Рис. 29: Сравнение участков двух моделей

Заключение

В систему Phoenixcas 3D Viewer была добавлена возможность анализа родинок на теле и отслеживания изменений состояния кожи в динамике по трехмерным моделям пациента. В ходе работы были выполнены следующие задачи.

1. Изучены способы обнаружения родинок по изображениям.
2. Реализована библиотека обнаружения родинок по изображениям на основе алгоритмов LoG и SBD.
3. Разработан модуль анализа пигментации по 3D-моделям:
    - реализованы инструменты детекции родинок:
        - автоматическое выделение родинок;
        - выделение и снятие выделения родинок ”вручную”;
        - вычисление размеров родинок;
    - реализованы инструменты визуализации для сравнения текстур:
        - проецирование областей анализа;
        - приближение выбранных участков модели.
4. Проведена апробация модуля.

Список литературы

[1] Abma B.J.M. Evaluation of requirements management tools with support for traceability-based change impact analysis // Master's thesis, University of Twente, Enschede. — 2009.

[2] Becerra-Riera F., Morales-Gonzalez A. Detection and matching of facial marks in face images // Revista Cubana de Ciencias Informaticas. — 2016.

[3] Bischoff B.S., Botsch M., Steinberg S. et al. OpenMesh – a generic and efficient polygon mesh data structure // In openSG symposium. — 2002.

[4] Cho T.S., Freeman W.T., Tsao H. A reliable skin mole localization scheme // IEEE International Conference on Computer Vision. — 2007.

[5] Comaniciu D., Meer P. Mean shift: A robust approach towards feature space analysis // IEEE Transactions Pattern Analysis and Machine Intelligence. — 2002.

[6] Fawcett T. An introduction to ROC analysis // Pattern recognition letters. — 2006.

[7] Gogoi U.R., Bhowmik M.K., Saha P. et al. Facial mole detection: an approach towards face identification // Procedia Computer Science. — 2015.

[8] Hsieh C.C., Lai J.A. Face mole detection, classification and application // Journal of Computers. — 2015.

[9] Jain A.K., Park U. Facial marks: soft biometric for face recognition // IEEE International Conference on Image Processing. — 2009.

[10] Jain R., Kasturi R., Schunck B.G. Machine Vision / Ed. by E.M. Munson. — McGraw-Hill New York, 1995.

[11] Lee T.K., Atkins M.S., King M.A. et al. Counting moles automatically from back images // IEEE Transactions on Biomedical Engineering. — 2005.

[12] Park U., Jain A.K. Face matching and retrieval using soft biometrics // IEEE Transactions on Information Forensics and Security. — 2010.

[13] Pierrard J.S., Vetter T. Skin detail analysis for face recognition // IEEE Conference on Computer Vision and Pattern Recognition. — 2007.

[14] Swaroop P., Sharma N. An overview of various template matching methodologies in image processing // International Journal of Computer Applications. — 2016.

[15] Szeliski R. Computer vision: algorithms and applications / Ed. by D. Gries, F.B. Schneider. — Springer Science & Business Media, 2010.

[16] Tong S., Chang E. Support vector machine active learning for image retrieval // In Proceedings of the ACM International Conference on Multimedia. — 2001.

[17] Warren J., Schaefer S., Hirani A.N. et al. Barycentric coordinates for convex sets // Advances in computational mathematics. — 2007.

[18] Wei S.D., Lai S.H. Fast template matching based on normalized cross correlation with adaptive multilevel winner update // IEEE Transactions on Image Processing. — 2008.

[19] Weinhaus F.M., Devarajan V. Texture mapping 3D models of real-world scenes // ACM Computing Surveys (CSUR). — 1997.

[20] Карнаухов В.Е. Разработка модуля анализа мелазмы на лице по 3D моделям пациента. — URL: `http://se.math.spbu.ru/SE/YearlyProjects/spring-2017/344/344-Karnaukhov-report.pdf` (дата обращения: 14.04.2018).