**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**«ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО»**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Объектно-ориентированное программирование»

на тему: «3*D*-реконструкция поясничного отдела позвоночника человека методом *dual* *contouring*»

Исполнитель: студент гр. ИТИ-22

Ковшаров Г. Ю.

Руководитель: доцент

Курочка К. С.

Дата проверки: *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Дата допуска к защите: ­*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Дата защиты: *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Оценка работы: *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Подписи членов комиссии

по защите курсовой работы: *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Гомель 2024

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc161537659)

[1 Алгоритмы 3d-реконструкции и средства разработки приложения 4](#_Toc161537660)

[1.1 Введение в 3d-реконструкцию 4](#_Toc161537661)

[1.2 Алгоритмы 3d-реконструкции 5](#_Toc161537662)

[1.3 Обзор средств разработки приложения 8](#_Toc161537663)

[Список ипользуемых источников 13](#_Toc161537664)

# ВВЕДЕНИЕ

В современном мире наука и медицина всё больше интересуются развитием информационных технологий, особенно в области 3*D*-реконструкции. На данный момент 3*D*-реконструкция органов и тканей является одной из наиболее актуальных и перспективных технологий в современной медицине. Она позволяет создавать трёхмерные модели органов и тканей с высокой точностью, что может быть полезно для планирования хирургических вмешательств и обучения студентов медицинских вузов.

Позвоночник, как и любой другой орган человека, может подвергаться различным видам травм, так как ежедневно он испытывает большие нагрузки. Исследовать патологии позвоночника можно при помощи рентгеновской компьютерной томографии. Однако в некоторых случаях медикаментозное лечение является не эффективным, и пациент нуждается в хирургическом вмешательстве. В подобных ситуациях в целях повышения информативности данных и грамотного хирургического планирования, целесообразно использовать 3*D*-реконструкцию позвоночника.

Также в целях улучшения качества реконструкции в трёхмерную модель, можно воспользоваться нейронной сетью, которая позволяет очищать, полученные при помощи компьютерной томографии изображения от различных шумов. Данные действия приведут к улучшению точности и информативности изображения, что в свою очередь облегчит применение алгоритмов 3*D*-реконструкции и улучшит качество результата.

В данной работе рассматривается процесс 3*D*-реконструкции с использованием алгоритма *Dual* *contouring* и предварительной очистки изображения, получаемого при помощи компьютерной томографии поясничного отдела позвоночника человека, от шумов при помощи нейронной сети.

Целью данной работы является автоматизация и улучшение качества обслуживания пациентов в медицинских учреждениях.

Таким образом, использование 3*D*-реконструкции в медицине имеет огромный потенциал. В будущем, применение данной технологии в медицине может стать одним из ключевых направлений развития медицинской науки и технологий.

## 1 АЛГОРИТМЫ 3D-РЕКОНСТРУКЦИИ И СРЕДСТВА РАЗРАБОТКИ ПРИЛОЖЕНИЯ

## 1.1 Введение в 3d-реконструкцию

Технология 3*D*-реконструкции представляет собой передовой метод трансформации двухмерных изображений в трёхмерные модели. Этот процесс является чрезвычайно сложным и включает в себя детальное извлечение и анализ геометрической информации, такой как глубина, форма и текстура, из исходных данных. Основная задача этого метода заключается в создании цифровой модели, которая бы максимально точно воспроизводила структуру и внешний вид исходного, реального объекта.

История 3*D*-реконструкции уходит корнями в эпоху зарождения областей компьютерного зрения и компьютерной графики. Уже в 1960-е годы учёные начали исследования в области извлечения трёхмерной информации из плоских изображений. С тех пор исследование методов 3*D*-реконструкции представляет собой одну из наиболее сложных и важных задач в данной области. Применение этого метода позволяет не только определить трёхмерный профиль объекта, но и точно установить трёхмерные координаты любой точки этого профиля. 3*D*-реконструкция объектов стала ключевой задачей во многих научных и практических областях, включая медицинскую визуализацию, компьютерное геометрическое моделирование, компьютерную графику и анимацию, компьютерное зрение, вычислительную науку, виртуальную реальность, а также в сфере цифровых медиа.

В медицинской сфере, например, 3*D*-реконструкция позволяет визуализировать патологии пациентов в трёхмерных моделях, что открывает новые возможности для более точной и качественной диагностики и дальнейшего обслуживания пациента, имея при этом значительное клиническое значение. Точные 3*D*-модели анатомии человека значительно упрощают клиническую диагностику, наблюдение за состоянием пациентов, планирование и проведение хирургических операций. Основные преимущества 3*D*-реконструкции в медицине включают:

– увеличение точности диагностики за счёт использования множественных представлений;

– детальную оценку и анализ поверхности тканей;

– планирование и моделирование хирургических вмешательств, предоставление навигации и поддержки во время медицинских процедур;

– улучшенное и более полное представление о структуре анатомии пациента;

– поддержку в различных клинических направлениях, включая планирование радиотерапии, мониторинг лечения, операции на позвоночнике, замену суставов, нейрохирургические вмешательства и стентирование сосудов.

Современные достижения в области 3*D*-реконструкции, развитие датчиков изображений и прогресс в алгоритмах реконструкции значительно продвинули данную технологию вперёд, расширяя границы возможного в этой области и делая её более точной.

## 1.2 Алгоритмы 3d-реконструкции

Существует множество различных алгоритмов трансформации двумерных изображений в трёхмерные модели, например: *Marching cubes*, *Marching* *tetrahedra*, *Asymptotic decider*, *Dual* *contourin*.

Рассмотрим подробнее каждый алгоритм.

*Marching сubes* – это алгоритм для создания трёхмерных поверхностей из данных, представленных в виде трёхмерной скалярной функции. Данный алгоритм был разработан в 1987 году Эдвардом Лоренсом и Вейнди Бурком. Алгоритм *Marching cubes* был первым примером в области компьютерной графики, спровоцировавшим скандал в области патентования программного обеспечения. Он был запатентован несмотря на относительную очевидность решения задачи генерации поверхности.

*Marching сubes* работает с трёхмерной сеткой, разбивая её на кубические ячейки. Алгоритм проходит через скалярное поле, одновременно беря восемь соседних местоположений, формируя таким образом воображаемый куб, а затем определяя многоугольник(и), необходимые для представления части изоповерхности, проходящей через этот куб. Каждый куб содержит вершины, которые представляют значения функции *f* в соответствующих точках. Для каждой вершины куба вычисляется значение неявной функции вида *f* (*x*, *y*, *z*) = 0. Отдельные полигоны затем объединяются в желаемую поверхность.

Это делается путем создания индекса для предварительно рассчитанного массива из 28 возможных конфигураций многоугольников внутри куба, обрабатывая каждое из восьми скалярных значений как бит в восьми-битном целом числе. Если значение скаляра выше, чем значение изоповерхностного уровня (то есть оно находится внутри поверхности), тогда соответствующий бит устанавливается в единицу, а если оно ниже (снаружи), он устанавливается в ноль. Окончательное значение после проверки всех восьми скаляров является фактическим индексом массива индексов полигонов.

Наконец, каждая вершина сгенерированных многоугольников помещается в соответствующую позицию вдоль края куба путем линейной интерполяции двух скалярных значений, соединенных этим краем.

Данный алгоритм имеет ряд недостатков, связанных с тем, что при аппроксимации изоповерхности могут возникать ошибки, которые влияют на точность представления поверхности.

Давайте рассмотрим некоторые из этих проблем:

­– алгоритм *Marching сubes* работает с дискретными данными, разбивая пространство на кубы. Из-за этого полученные полигоны представляют приближённую аппроксимацию изоповерхности. В результате поверхность может быть недостаточно точной, особенно при низком разрешении данных;

– в некоторых случаях алгоритм может столкнуться с неоднозначностью. Например, если предположить, что углы помечены положительно, если их значение больше, чем у изолинии, или отрицательно, если оно меньше, то либо положительные углы разделены двумя изолиниями, либо положительные углы находятся в основной части квадрата, а отрицательные углы разделены двумя изолиниями;

– малые изменения в значениях скалярного поля могут привести к большим изменениям в генерируемых полигонах. Шум в данных может существенно повлиять на точность;

– алгоритм может иметь трудности с представлением сложных геометрических форм.

Позднее был разработан похожий алгоритм, названный Marching *tetrahedra*, который для того, чтобы обойти патент *Marching сubes*, использует вместо кубов тетраэдры.

*Marching tetrahedra* – это алгоритм в области компьютерной графики для рендеринга неявных поверхностей. Первоначально он был представлен в 1991 году Акио Дои и Акио Койде.

В *Marching tetrahedra* каждый куб разбивается на шесть неправильных тетраэдров путём трёхкратного разрезания куба пополам, разрезая по диагонали каждую из трёх пар противоположных граней. Таким образом, все тетраэдры имеют общую одну из главных диагоналей куба. Вместо двенадцати рёбер куба теперь девятнадцать рёбер: исходные двенадцать, шесть диагоналей граней и главная диагональ. Так же, как и в *Marching сubes*, пересечения этих рёбер с изоповерхностью аппроксимируются линейной интерполяцией значений в узлах сетки.

Соседние кубики имеют общие рёбра соединяющейся грани, включая одну и ту же диагональ. Это важное свойство для предотвращения трещин на визуализированной поверхности, поскольку интерполяция двух различных диагоналей грани обычно даёт несколько разные точки пересечения. Дополнительным преимуществом является то, что при обработке соседнего куба можно повторно использовать до пяти вычисленных точек пересечения. Сюда входят вычисленные нормали поверхности и другие графические атрибуты в точках пересечения.

Каждый тетраэдр имеет шестнадцать возможных конфигураций, которые делятся на три класса: без пересечения, пересечение в одном треугольнике и пересечение в двух (соседних) треугольниках.

*Asymptotic decider* – это алгоритм, разработанный Нильсоном и Хаманном в 1991 году, который создаёт изоповерхности из заданного скалярного поля. Он был предложен как улучшение алгоритма *Marching сubes* и решение одной из его проблем – неоднозначности, из-за которой может создаваться «плохая» топология, но также данный алгоритм может считаться самостоятельным. *Marching сubes* гарантирует, что полученные многоугольники всегда будут топологически корректными. Это важно для правильного представления поверхностей. Он описывает метод разрешения неоднозначных конфигураций последовательно. Это позволяет избежать ошибок в топологии. Таким образом, *Asymptotic* *decider* предоставляет более надежное и эффективное решение для создания изоповерхности из скалярных полей, чем *Marching* *cubes*.

Алгоритм сначала разбивает скалярное поле на равные кубы. Это делается для упрощения обработки и представления данных. Затем на боковых (граничных) поверхностях каждого куба рисуются топологически корректные контуры. Эти контуры представляют собой изолинии (линии постоянного значения скалярной функции). Контуры затем соединяются в многоугольники и триангулируются. Эти многоугольники могут быть треугольниками или другими простыми многоугольниками. Триангуляция многоугольников позволяет получить топологически корректные поверхности. В итоге все треугольники, полученные из каждого куба, формируют итоговую изоповерхность.

Однако, в процессе создания изоповерхностей возникают неоднозначные случаи, когда диагонально противоположные точки находятся с одной стороны изолинии, но с другой стороны относительно других точек в кубе. В этом случае алгоритм *Asymptotic decider* использует гиперболу для определения, как правильно разделить куб на две области. Благодаря этим действиям данный алгоритм как раз таки и решает проблему неоднозначности алгоритма *Marching cubes*, из-за которой может создаваться «плохая» топология.

*Dual contouring* ­– это тоже метод создания трёхмерных поверхностей из данных, представленных в виде скалярной функции. Данный алгоритм впервые был опубликован в 2002 году в сборнике работ конференции *SIGGRAPH*, авторами которого являются Тао Ю и Фрэнк Лосассо.

*Dual* *contouring* так же работает с трёхмерной сеткой, разбивая её на кубические ячейки. Для каждой ячейки воксельной сетки алгоритм вычисляет вершины сетки внутри ячейки. Эти вершины находятся на контурных линиях, которые соответствуют пересечениям поверхности с ячейкой. Алгоритм соединяет вершины, чтобы получить треугольники или квадраты. Это формирует поверхностную сетку. В отличие от *Marching cubes*, мы не можем оценивать ячейки независимо. Мы должны рассмотреть соседние, чтобы «соединить точки» и найти полную сетку. Но на самом деле это гораздо более простой алгоритм, чем *Marching cubes*, потому что здесь не так много отдельных случаев. Вы просто находите каждое ребро со сменой знака и соединяете вершины ячеек, соседних с этим ребром.

В *Marching cubes* нужно было просто вычислить функцию *f* на сетке. Теперь нам нужна информация о градиенте. Градиент функции *f* – это мера того, насколько быстро изменяется значение *f* в данной точке при движении в любом заданном направлении. Обычно он задается в виде пары чисел для каждой точки, указывая, насколько изменяется функция при перемещении по оси *X* или оси *Y*. Соединение точек через рёбра выполняется с изменением знака функции, используя информацию о градиенте. Так же для определения точки на сетке, в целях улучшения точности, используется функция квадратичной ошибки. Это позволяет *Dual* *contouring* более точно аппроксимировать поверхности, особенно в областях с резкими углами и краями, так как учитывается изменения функции в окрестности каждой вершины.

В рамках данного проекта будет использован алгоритм *Dual* *contouring*. Был выбран именно этот алгоритм, так как он решает некоторые проблемы алгоритма *Marching сubes* и позволяет получить более естественную модель.

Цель работы заключается в изучении принципов и методов для работы с 3*D*-реконструкцией. Также в работе предполагается создание нейронной сети для очистки изображения, полученного при помощи компьютерной томографии, от шумов и дальнейшей его реконструкции в 3*D*-модель, при помощи алгоритма *Dual* *contouring*.

## 1.3 Обзор средств разработки приложения

В настоящее время в мире существует несколько сотен реально используемых языков программирования. Для каждого есть своя область применения.

Любой алгоритм, как мы знаем, есть последовательность предписаний, выполнив которые можно за конечное число шагов перейти от исходных данных к результату. В зависимости от степени детализации предписаний обычно определяется уровень языка программирования – чем меньше детализация, тем выше уровень языка.

По этому критерию можно выделить следующие уровни языков программирования:

– машинные языки программирования, воспринимаемые аппаратной частью компьютера (машинные коды);

– машинно-ориентированные языки программирования (ассемблеры);

– языки программирования, которые отражают структуру конкретного типа компьютера;

– машинно-независимые (языки высокого уровня).

Машинные языки и машинно-ориентированные языки – это языки низкого уровня, требующие указания мелких деталей процесса обработки данных. Языки же высокого уровня имитируют естественные языки, используя некоторые слова разговорного языка и общепринятые математические символы.

До того, как компьютер сможет выполнить программу, написанную на языке высокого уровня, её приходится «переводить» на понятный компьютеру язык, то есть машинный код. Этот процесс перевода называют трансляцией, а программу-переводчик – транслятором. Трансляторы делятся на два класса: компиляторы и интерпретаторы.

Компиляция заключается в том, что программа в машинном коде (называемая компилятором) преобразует другую программу, написанную на языке программирования в машинный код. После этого полученный машинный код программы выполняется.

Интерпретация заключается в том, что программа в машинном коде (интерпретатор) записывает файл программы во внутреннюю память и начинает её построчно выполнять.

Языки высокого уровня делятся на несколько типов:

– процедурные;

– логические;

– объектно-ориентированные.

Процедурные языки программирования предоставляют возможность программисту определять каждый шаг в процессе решения задачи. Особенность таких языков программирования состоит в том, что задачи разбиваются на шаги и решаются шаг за шагом. Используя процедурный язык, программист определяет языковые конструкции для выполнения последовательности алгоритмических шагов.

Логические языки программирования – языки программирования, позволяющие выполнить описание проблемы в терминах фактов и логических формулах, а собственно решение проблемы выполняет система с помощью механизмов логического вывода.

Объектно-ориентированные языки программирования, в основе которых лежит понятие объекта, сочетающего в себе данные и действия над ними. Программа на объектно-ориентированном языке, решая некоторую задачу, по сути, описывает часть мира, относящуюся к этой задаче. Описание действительности в форме системы взаимодействующих объектов естественнее, чем в форме взаимодействующих процедур.

В настоящее время для разработчиков предлагается широкое разнообразие прикладных программных продуктов, позволяющее автоматизировать различные задачи проектировщиков и разработчиков.

В целях создания нейронной сети и реконструкции из двухмерного пространства в трёхмерное, также может быть использовано множество языков программирования разных уровней и типов. Ниже приведен обзор самых часто используемых языков программирования.

*Python* – это высокоуровневый, интерпретируемый язык программирования с динамической типизацией и автоматическим управлением памятью, к преимуществам которого относят высокую производительность программных решений и структурированный, хорошо читаемый код. Синтаксис *Python* максимально облегчен, что позволяет выучить его за сравнительно короткое время.

*Python* широко используется для разработки веб-приложений, работы с графикой, создания искусственного интеллекта, научных вычислений, системного администрирования. Благодаря своей простоте и читаемости, *Python* особенно популярен среди начинающих программистов и в образовательной сфере.

*Python* обладает богатой стандартной библиотекой и большим количеством сторонних модулей, что позволяет легко интегрировать различные системы и обрабатывать данные разных форматов. Он поддерживает множество парадигм программирования, включая объектно-ориентированное, процедурное и функциональное программирование.

Программы на *Python* выполняются в среде *Python Interpreter*, которая обеспечивает кроссплатформенную совместимость и возможность работы на большинстве операционных систем. Данный язык также является основой для многих популярных фреймворков, таких как *Django* для веб-разработки и *scikit-learn* для машинного обучения.

На сегодняшний день *Python* является одним из самых востребованных языков программирования в мире, благодаря своей гибкости, мощности и широкому применению в различных областях.

Язык поддерживает множество расширенных функций, таких как генераторы, декораторы и контекстные менеджеры, которые делают код более компактным и удобным для поддержки. Он продолжает активно развиваться, и с каждым новым релизом предлагает всё больше возможностей для разработчиков.

*Java* – язык программирования общего назначения. Относится к объектно-ориентированным языкам программирования, к языкам с сильной типизацией. Синтаксис языка *Java* похож на синтаксис других си-подобных языков. Программы на *Java* могут быть транслированы в байт-код, выполняемый на виртуальной *Java*-машине (*JVM*) – программе, обрабатывающей байт-код и передающей инструкции оборудованию, как интерпретатор, но с тем отличием, что байт-код, в отличие от текста, обрабатывается значительно быстрее. Реализация проекта была начата на языке *C++*, но вскоре возник ряд проблем, наилучшим средством борьбы с которыми было изменение самого инструмента – языка программирования. Стало очевидным, что необходим платформо-независимый язык программирования, позволяющий создавать программы, которые не приходилось бы компилировать отдельно для каждой архитектуры и можно было бы использовать на различных процессорах под различными операционными системами.

Три ключевых элемента объединились в технологии языка *Java:*

1) *Java* предоставляет для широкого использования свои апплеты (*applets*) – небольшие, надежные, динамичные, не зависящие от платформы активные сетевые приложения, встраиваемые в страницы *Web*. Апплеты *Java* могут настраиваться и распространяться потребителям с такой же легкостью, как любые документы *HTML*;

2) *Java* высвобождает мощь объектно-ориентированной разработки приложений, сочетая простой и знакомый синтаксис с надежной и удобной в работе средой разработки. Это позволяет широкому кругу программистов быстро создавать новые программы и новые апплеты;

3) *Java* предоставляет программисту богатый набор классов и объектов для ясного абстрагирования многих системных функций, используемых при работе с окнами, сетью и для ввода-вывода. Ключевая черта этих классов заключается в том, что они обеспечивают создание независимых от используемой платформы абстракций для широкого спектра системных интерфейсов.

*C#* – объектно-ориентированный, компилируемый, со статической типизацией, автоматическим управлением мусора, ориентированный на компоненты язык программирования. *C*# активно используется для реконструкции изображений из двухмерного пространства в трёхмерное и работы с нейронными сетями. Этот язык программирования обладает мощными объектно-ориентированными возможностями, которые идеально подходят для создания сложных систем. *C*# предоставляет разработчикам набор инструментов для эффективной работы с графикой и алгоритмами машинного обучения, что делает его отличным выбором для проектов, связанных с компьютерным зрением и искусственным интеллектом.

Программы *C#* выполняются в *.NET*, виртуальной системе выполнения, вызывающей общеязыковую среду выполнения (*CLR*) и набор библиотек классов. Среда *CLR* – это реализация общеязыковой инфраструктуры языка (*CLI*), являющейся международным стандартом, от корпорации Майкрософт. *CLI* является основой для создания сред выполнения и разработки, в которых языки и библиотеки прозрачно работают друг с другом.

На сегодняшний момент язык программирования *C#* один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

*C#* поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И *C#* продолжает активно развиваться, и с каждой новой версией появляется всё больше интересных функциональностей.

Когда говорят *C#*, нередко имеют в виду технологии платформы .*NET* (*Windows Forms, WPF, ASP.NET, Xamarin*). И, наоборот, когда говорят .*NET*, нередко имеют в виду *C*#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык *C*# был создан специально для работы с фреймворком .*NET*, однако само понятие .*NET* несколько шире.

Язык программирования *С*# является предпочтительным для осуществления реконструкции изображений из двухмерного пространства в трёхмерное, так как содержит большое количество системных функций и библиотек, которые делают работу с данной областью более комфортной.

# СПИСОК ИПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. *Dual* *Contouring* *Tutorial*. –Электрон. данные. – Режим доступа: [*https://www.boristhebrave.com/2018/04/15/dual-contouring-tutorial/*](https://www.boristhebrave.com/2018/04/15/dual-contouring-tutorial/). – Дата доступа: 16.03.2024.

2. Курочка,К. С.Адаптированные алгоритмы *Dual Contouring* и *Marching Cubes* для3*D*-реконструкции поясничного отдела позвоночника человека/К. С. Курочка, Т. С. Семенченя //Доклады БГУИР.– 2023. – Т. 21, № 6. – С. 99–105.

3. Семенченя, Т. С. Реконструкция 3*D*-модели поясничного отдела позвоночника человека методом *Dual* *Contouring* на основе анализа цифровых КТ-изображений / Т. С. Семенченя, К. С. Курочка // Информационные технологии и системы – 2022 (ИТС – 2022): матер. междунар. науч. конф., Минск, 23 ноября 2022 г. / Минск: Белорус. гос. ун-т информ. и радиоэлектр. – 2022. – С. 163–165.