Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем
Системное программирование

Пономарёва Наталья Александровна

Реконструкция нежёстких моделей людей в реальном времени по RGB-D видео

Выпускная квалификационная работа бакалавра

Научный руководитель:
доц. каф. СП, к. т. н. Литвинов Ю. В.

Консультант:
Корчёмкин Д. А.

Рецензент:
Ведущий разработчик ООО “Системы Компьютерного Зрения”, к. т. н. Федоренко С. И.

Санкт-Петербург
2020

SAINT-PETERSBURG STATE UNIVERSITY

Software and Administration of Information Systems
Software Engineering

Ponomareva Natalia

Real-time non-rigid reconstruction of human bodies from RGB-D video sequences

Bachelor's Thesis

Scientific supervisor:
C.Sc. Docent Yurii Litvinov

Consultant:
Dmitry Korchemkin

Reviewer:
Team lead, Computer Vision Systems LLC, C.Sc. Sergey Fedorenko

Saint-Petersburg
2020

Оглавление

Введение

На протяжении долгого времени детекция и реконструкция людей по видео с данными о глубине в реальном времени оставалась сложной и нерешённой задачей в компьютерном зрении. Ее целью является восстановление полноценной 3D-модели по последовательному ряду RGB-D изображений при помощи постепенного извлечения информации из каждого изображения и объединения этих данных. Особенностью RGB-D изображений является наличие четырёх каналов – красного, зеленого и синего, как в привычной нам цветовой модели, и так же карты глубины, в которой каждый пиксель задаёт расстояние до объекта в сцене. Первый существенный прорыв в области 3D-реконструкции был совершён в 2015 году Newcombe R. A. и др. [6]. Позже задача нежёсткой реконструкции по данным о глубине неоднократно рассматривалась в научных публикациях, например, [4], [5], [9], [3].

Нежёсткая реконструкция – это реконструкция поверхностей, не имеющих внешней формы, поддерживаемой неподвижным каркасом, например, людей, животных, в отличие от реконструкции жёсткой сцены – комнаты, земли. 3D-модель – это геометрический объект, при проецировании на дисплей визуально похожий на требуемый объект со всех сторон. Такие модели широко используются в компьютерной графике, компьютерных играх, кинематографе, телевидении и даже при создании макетов для печатной продукции.

Отдельным подклассом задач нежёсткой реконструкции является реконструкция человека, совершающего разнообразные активности. В подобном сценарии эксплуатация модели человеческого тела позволяет достичь лучшего качества реконструкции [1]. Однако, ни один из рассмотренных методов не позволяет полностью решить задачу реконструкции нежёсткой модели человека (с учётом предметов одежды и иных объектов) без наличия каких либо ограничений (например – требования к начальной позе).

Ввиду значительного удешевления датчиков глубины (в частности – основанных на Time-of-Flight измерениях), предполагается скорое внедрение подобных технологий в пользовательские устройства (в частности –

мобильные телефоны). В контексте технологии, ориентированной на простых пользователей, необходимо обеспечить максимально комфортное взаимодействие пользователя с устройством, в частности – максимально упростить требования, которые необходимо соблюсти для успешной реконструкции (в том числе – обеспечив возможность реконструирования из любой начальной позы, а также – реконструирования совместно с предметами одежды и иными аксессуарами). Кроме того, необходимы алгоритмы, обеспечивающие максимальное быстродействие, для обеспечения возможности производить реконструкцию на пользовательском устройстве в околореальном времени.

В данной работе предлагается объединение идей прямого предсказания параметров модели человеческого тела из RGB-изображений [2], оценки параметров модели человека и нежёсткой реконструкции "связанных" с ним объектов [1] и эффективного (с точки зрения скорости вычислений и затрат памяти) подхода к хранению промежуточных состояний нежёсткой реконструкции [3].

1 Постановка задачи

Целью данной дипломной работы является разработка и реализация алгоритма, который восстанавливает физическую модель человека по RGB-D видеопотоку из любой начальной позы человека и эффективно хранит промежуточные состояния сцены. Для достижения этой цели были поставлены следующие задачи:

- произвести обзор существующих алгоритмов реконструкции людей по видеоизображениям,
- разработать алгоритм восстановления модели человека по видеоизображениям с эффективным хранением промежуточных состояний сцены,
- разработать архитектуру и выполнить реализацию алгоритма,
- выполнить тестирование решения на эталонных данных.

2 Обзор

2.1 Алгоритм восстановления сцены на основе вокселеподобных структур данных

В статье [6] впервые была описана система, способная реконструировать нежёсткие изменяющиеся сцены в реальном времени при помощи слияния вместе RGB-D-изображений (причем в данной статье предполагалось использование исключительно данных о глубине). Авторы статьи описывают два ключевых отличия от всех предыдущих подходов, которые позволили им работать с движущимися объектами. Во-первых, важно хранить всю геометрию сцены в некотором фиксированном, каноническом пространстве, а для каждого нового кадра уметь построить отображение из него в это каноническое пространство (или наоборот). Во-вторых, невозможно оценить точное отображение для каждой точки пространства, так как это требует слишком больших вычислительных ресурсов. Поэтому предлагается использовать разреженные иерархические пространственные функции.

Деформирующее отображение сопоставляет определённому набору точек в пространстве (который называется граф деформирующего отображения) соответствующие им движения (чтобы уметь переводить из канонического пространства не только точки, но и нормали). Для того, чтобы вычислить значение отображения в конкретной точке, используется интерполяция с заранее заданными весами влияния по известным точкам.

Каноническое пространство предлагается хранить с помощью TSDF (truncated signed distance field, то есть усечённой функции расстояния со знаком). Производится выборка точек из пространства, эти точки сопоставляются центрам параллелепипедов – вокселей, и вводится функция, которая для каждого вокселя (элемента пространства) возвращает расстояние до ближайшей точки, лежащей на поверхности реконструированного объекта. Эта функция меняется при слиянии с новым полученным кадром по определенным формулам, в которых так же учитывается дальность от общей массы реконструированных точек (чем дальше – тем вероятнее, что это случайный выброс).

Для построения деформирующего отображения $\mathcal{W}_t$ из канонического пространства V в текущий кадр D_t вводится понятие энергии (1), которую пытаются минимизировать по отображению W_t, используя алгоритм ICP (Iterative Closest Point), который позволяет сопоставить между собой облака точек:

$$E(\mathcal{W}_t, V, D_t, \varepsilon) = Data(\mathcal{W}_t, V, D_t) + \lambda Reg(\mathcal{W}_t, \varepsilon) \tag{1}$$

По каноническому пространству, которое хранится как TSDF, строят поверхность при помощи алгоритма "шагающих кубиков" (marching cubes), который итеративно проходится по всем вокселям и по значению функции для текущего вокселя и восьми его соседей восстанавливает поверхность, проходящую через текущий воксель. Затем полученную геометрию отображают в текущий кадр деформирующим отображением. Data составляющая выражения (1) отвечает за соответствие между полученной картой глубин и текущей реконструкцией. Дополнительная составляющая выражения (1) обеспечивает как можно более жёсткие изменения между точками, соединёнными ребрами из множества ϵ, избегая таким образом негладких движений.

Пусть функция энергии представляется в виде суммы квадратов некоторых функций $f_i(\mathcal{W}_t)$: $E(\mathcal{W}_t) = \sum_i f_i^2(\mathcal{W}_t)$, тогда вектор, составленный из всех компонент обозначим $f(\mathcal{W}_t)$. В таком виде проблему минимизации $E(\mathcal{W}_t)$ можно рассматривать как стандартную разреженную нелинейную задачу наименьших квадратов, которая может быть решена итеративными подходами, основанными на алгоритме Гаусса-Ньютона. Для этого на каждом шаге алгоритма решается система $\mathbf{J}^{\mathrm{T}}\mathbf{J}(\mathcal{W}_t^{(k+1)} - \mathcal{W}_t^k) = \mathbf{J}^{\mathrm{T}} f(\mathcal{W}_t^k)$, где $\mathbf{J}$ – это якобиан функции f, а $\mathbf{J}^{\mathrm{T}}\mathbf{J}$ – это аппроксимация гессиана в области малых значений функции энергии E .

2.2 Алгоритм восстановления сцены на основе сёрфелеподобных структур данных

В отличие от предыдущего подхода, авторы статьи [3] отказались от использования объёмных структур данных (например, TSDF), так как они

требуют слишком больших объемов памяти и производительности. Предлагаемый подход позволяет избавиться от таких дорогих операций, как алгоритм "шагающих кубиков" (marching cubes) и слияние вокселе-подобных структур. В его основе лежит понятие сёрфел (surfel) – кортеж, который локально аппроксимирует поверхность объекта. Каждый сёрфел состоит из позиции $v \in \mathbb{R}^3$, нормали $n \in \mathbb{R}^3$, радиуса $r \in \mathbb{R}^+$, степени доверия $c \in \mathbb{R}$, времени появления $t_{init} \in \mathbb{N}$, времени последнего наблюдения $t_{observed} \in \mathbb{N}$.

Аналогично [6] предлагается использовать деформирующее отображение для перехода из канонического пространства (в котором объект хранится как массив сёрфелей) в текущий кадр. По полученному кадру строится новый массив сёрфелей, который пытаются сопоставить каноническому пространству, на которое подействовали деформирующим отображением. Эту задачу решают путём минимизации функции энергии:

$$E(\mathcal{W}_t, V, D_t, \varepsilon) = E_{data}(\mathcal{W}_t, V, D_t) + \lambda E_{reg}(\mathcal{W}_t, \varepsilon) \tag{2}$$

Слияние сёрфелей, полученных из нового кадра с имеющейся геометрией, производится в пространстве кадра, после чего при помощи деформирующего отображения итоговый набор переводится в каноническое пространство.

2.3 Представление модели человеческого тела

В статье [8] представляется SMPL[1] модель человеческого тела, которая зависит от конкретных формы и позы человека. Её основными преимуществами являются точность по сравнению с ранее существовавшими моделями и совместимость со многими существующими графическими 3D редакторами.

Модель представляет из себя полигональную сетку (набор N вершин – T, определяющих форму трёхмерного объекта) и набор суставов K – вершин, относительно которых могут двигаться и сгибаться части человеческого тела. Так же есть набор функций: $B_s(\beta) : \mathbb{R}^{|\beta|} \to \mathbb{R}^{3N}$ – сопоставляет коэффициентам, описывающим форму человеческого тела, измененный на-

[1]Skinned Multi-Person Linear Model

бор вершин, которые удовлетворяют этой форме; $J(\beta) : \mathbb{R}^{|\beta|} \to \mathbb{R}^{3K}$ – сопоставляет коэффициентам, описывающим форму человеческого тела, новый набор немного сдвинутых суставов; $B_p(\theta) : \mathbb{R}^{|\theta|} \to \mathbb{R}^{3N}$ – сопоставляет коэффициентам, описывающим позу человеческого тела, преобразованный в пространстве набор вершин, который более точно описывает форму человека в этой конкретной позе; $W(\cdot)$ – основная функция, которая вращает вершины относительно обновлённых расположений суставов и сглаживает результат с определёнными весами w.

В итоге, модель человеческого тела можно описать следующей функцией:

$$M(\beta, \theta) = W\Big(T + B_s\left(\beta\right) + B_p\left(\theta\right), J\left(\beta\right), \theta, w\Big) \tag{3}$$

2.4 Предсказание параметров SMPL-модели

В то время, как параметры, описанные в предыдущем параграфе, являются предпосчитанными для женщин и мужчин и представляют из себя обученную модель, то переменные β, θ характеризуют модель конкретного человека в конкретной позе. Поэтому должна быть возможность получить начальное приближение для данных параметров по первому кадру, что и делается методами машинного обучения в статье [2].

Большинство других существующих подходов восстанавливают 3D модель человека в две стадии – сначала в плоскости, а затем из плоскости переводят в пространство. Однако, таким образом теряется часть существенной информации, поэтому предлагается выводить параметры напрямую из пикселей. С таким подходом связано несколько проблем. Во-первых, датасеты с истинными значениями 3D моделей и подробными чёткими аннотациями обладают множеством ограничений, тогда как хотелось бы применять нейросеть для изображений из реального мира. Во-вторых, одной и той же проекции человека на плоскость может соответствовать множество поз в пространстве, однако большинство из них физически невозможны, поэтому хотелось бы их отсекать. Также есть некоторая неопределённость между размерами человека и расстоянием до камеры.

Для восстановления 3D представления используется итеративная 3D-регрессия с подкреплением. Далее запускается нейронная сеть, которая отделяет возможные позы от нереалистичных. Дескриптор – это визуальная характерная особенность, которая описывается формой, цветом, текстурой или движением, которые отличаются от других. В случае существования 2D-аннотаций (правильных описаний изображения в плоскости) можно построить отображение, которое 3D-дескрипторам сопоставляет 2D-дескрипторы и далее сравнивает их с верными ответами.

2.5 Алгоритм восстановления модели тела человека на основе вокселе-подобных структур данных

Авторы статьи предлагают новый подход, соединяющий в себе воксельное представление мира [6] и полную параметрическую форму тела [8], позволяя тем самым реконструировать детализированные поверхности и правдоподобное тело человека в реальном времени. Основная идея заключается в двойном уровне представления: внутри SMPL-модель тела человека, а снаружи с ней связана поверхность в каноническом пространстве. Таким образом, накладываются дополнительные ограничения для соответствия обоих уровней.

Теперь воксельная сетка, описанная в статье [6], разбивается на две части – нательную воксельную сетку и далекую от тела воксельную сетку. Причём выборка точек в нательную сетку производится по поверхности тела (SMPL-модели). При построении деформирующего отображения паралелльно так же оптимизируется поза человека (параметр θ) при помощи минимизации функции энергии:

$$E_{motion}(\mathcal{W}_t, \theta) = \lambda_{data} E_{data}(\mathcal{W}_t, V, D_t) + \lambda_{bind} E_{bind}(\mathcal{W}_{t_{on-body}}, \theta) + \\ \lambda_{reg} E_{reg}(\mathcal{W}_t, \varepsilon, w) + \lambda_{prior} E_{prior}(\theta), \quad (4)$$

где E_{data} отвечает за соответствие параметров новому кадру, E_{bind} – за соответствие SMPL-модели и нательной воксельной сетки, E_{reg} – за как можно

более жёсткие изменения между точками, чтобы избежать негладких движений, E_{prior} – за натуральность позы.

Дополнительно, после вывода деформирующего отображения для текущего кадра и слияния нового кадра с реконструкцией, предлагается оптимизировать форму и позу человеческого тела. Для этого определяется еще одна функция энергии:

$$E_{shape}(\beta, \theta) = E_{sdata}(\beta, \theta) + E_{sreg}(\beta, \theta, \beta_{prev}, \theta_{prev}) + E_{prior}(\theta), \qquad (5)$$

где E_{sdata} отвечает за соответствие параметров текущей реконструкции, E_{sreg} – за согласованность новых параметров со старыми, E_{prior} – за натуральность позы.

Однако, существенным минусом предлагаемого подхода является использование объёмных структур данных, которые влияют на производительность алгоритма и по памяти, и по времени.

3 Описание предлагаемого алгоритма

Общая идея заключается в последовательной обработке каждого кадра и слиянии результатов для построения 3D-модели человека. По первому поступившему кадру система инициализирует свои параметры. Вначале, при помощи машинного обучения инициализуются параметры позы и формы человеческого тела – β, θ. Затем инициализируются массивы сёрфелей S_{ref} и S_{live} (совпадающие для первого кадра), причём для каждого сёрфеля определяется – либо он лежит на теле человека (или достаточно близко), либо далеко от человека. Вершины для графа деформирующего отображения выбираются из S_{ref} отдельно для нательных сёрфелей – с использованием геодезических расстояний (на поверхности тела) и отдельно для далеко-лежащих сёрфелей – сеткой с радиусом 5см. Деформирующее отображение в начальный момент времени тождественно для всех вершин графа. Затем инициализируются ближайшие соседи и их веса для всех сёрфелей.

При получении нового кадра последовательно происходят следующие шаги: обработка входного изображения, вычисление нового деформирующего отображения, слияние полученной информации с имеющейся геометрией сцены и обновление всех параметров. Более подробно они будут описаны далее.

3.1 Вычисление деформирующего отображения

Предположим, что в данный момент уже существует оценка для вектора параметров β, описывающих форму человека, и нужно лишь понять, как он изменил свое положение в пространстве, и как переместились другие предметы в сцене. Деформирующее отображение $\mathcal{W}_t$ и позу человека θ_t предлагается оценивать с помощью ICP алгоритма, минимизируя следующую функцию энергии:

$$E_{motion}(\mathcal{W}_t, \theta) = \lambda_{data} E_{data}(\mathcal{W}_t, V, D_t) + \lambda_{bind} E_{bind}(\mathcal{W}_{t_{on-body}}, \theta) + \lambda_{reg} E_{reg}(\mathcal{W}_t, \varepsilon, w) \quad (6)$$

- Слагаемое E_{data} отвечает за соответствие между реконструированной двухуровневой поверхностью и поступившей картой глубины.

$$E_{data} = \sum_{(v_c, u) \in \mathcal{P}} \tau_1(v_c) * \psi\big(\tilde{n}_{v_c}(\tilde{v}_c - u)\big) + \big(\tau_2(v_c) + \tau_3(v_c)\big) * \psi\big(\hat{n}_{v_c}(\hat{v}_c - u)\big), \quad (7)$$

где $\mathcal{P}$ – это массив соответствий; $\psi(\cdot)$ – это устойчивая к выбросам функция потерь, определяемая формулой $\psi(x) = \frac{x^2}{x^2+\sigma^2}$; u – это точка карты глубины, а v_c – ближайшая соответствующая точка реконструкции. Индикаторные функции $\tau_1(v_c), \tau_2(v_c), \tau_3(v_c)$ определяют местоположение этой самой точки аналогично формуле 4.

$\tilde{v}_c, \tilde{n}_{v_c}$ – это позиция и нормаль сёрфеля, преобразованные с помощью деформирующего отображения.

$\hat{v}_c, \hat{n}_{v_c}$ – это позиция и нормаль, преобразованные с помощью основной функции SMPL-модели, то есть повёрнутые относительно расположений суставов и сглаженные с определенными весами по формуле:

$$\begin{gathered} G(v_c) = \sum_{i \in \mathcal{B}} \left(w_{i,v_c} \prod_{k \in \mathcal{K}_i} \exp\left(\theta_k \xi_k\right) \right), \\ \hat{v}_c = G(v_c) v_c \\ \hat{n}_{v_c} = rotation\left(G\left(v_c\right)\right) n_{v_c} \end{gathered} \quad (8)$$

где $\mathcal{B}$ – это набор суставов, $\mathcal{K}_i$ – это родительские индексы в подвешенном дереве суставов, коэффициенты ξ_k характеризуют положения суставов, w_{i,v_c} – взвешенное среднее весов, соответствующих ближайшим соседям вершины v_c в наборе вершин SMPL-модели, имеющей определенную форму человеческого тела.

- Слагаемое E_{bind} прикрепляет вершины, лежащие на теле человека к их ближайшим рёбрам в SMPL-модели:

$$E_{bind} = \sum_{i \in \mathcal{L}_s} \left\| \mathcal{W}_t \left(x_i \right) x_i - \hat{x}_i \right\|_2^2, \tag{9}$$

где $\mathcal{L}_s$ – множество вершин, лежащих на теле человека, $\mathcal{W}_t(x_i)x_i$ – вершина под воздействием деформирующего отображения, $\hat{x}_i$ – вершина, преобразованная основной функией SMPL-модели аналогично предыдущему пункту.

- Слагаемое E_{reg} отвечает за как можно более жёсткие изменения между точками, чтобы избежать негладких движений:

$$E_{reg\_far\_body} = \sum_{j} \sum_{i \in \mathcal{N}(j)} \left\| \mathcal{W}_t(x_i)x_j - \mathcal{W}_t(x_j)x_j \right\|_2^2 \tag{10}$$

Однако рядом с суставами потенциально могут как раз быть сильные изменения, поэтому для вершин, лежащих на теле человека эффект регуляризации контролируется отдельно:

$$E_{reg\_on\_body} = \sum_{i} \sum_{j \in \mathcal{N}(i)} \rho\big(\|W_i - W_j\|_2^2\big) \left\| \mathcal{W}_t(x_i)x_j - \mathcal{W}_t(x_j)x_j \right\|_2^2, \tag{11}$$

где W_i обозначает вектор коэффициентов – весов SMPL-модели, соответствующих всем суставам и определённой вершине x_i, а $\rho(\cdot)$ – взвешенная функция Хубера, задаваемая формулой с параметром $\delta = 0.2$:

$$\rho(x) = \begin{cases} \frac{x^2}{2}, & \text{если } |x| < \delta \\ \delta \left(|a| - \frac{\delta}{2} \right), & \text{иначе} \end{cases}$$

3.2 Слияние полученной информации с имеющейся геометрией сцены

Слияние данных происходит в пространстве текущего кадра – то есть с массивом сёрфелей S_{live}, затем отображается при помощи посчитанного деформирующего отображения обратно в S_{ref}, после чего обновляется информация об отображении для только что добавленных сёрфелей.

- Вначале строится специальная карта I: при помощи внутренних параметров камеры каждый сёрфел $S_{live}[k]$ проецируется в плоскость изображения камеры. После этого из карты выбирается подмножество сёрфелей, такое что одному пикселю карты глубины соответствует не более одного сёрфеля. Каждому пикселю u сопоставляется не более одного сёрфеля, удовлетворяющего следующим критериям:
  - расстояние между ними не больше, чем $\delta_{dist} = 1mm$,
  - их нормали выровнены, то есть $dot(n_{depth}, n_{surfel}) < \delta_{normal}$,
  - выбирается сёрфел с наибольшим коэффициентом доверия,
  - если существует несколько сёрфелей, то выбирается ближайший.

  Если в итоге найден подходящий сёрфел, то по пикселю u строится новый сёрфел, который сливается с найденным по определённым формулам.

- Если для пикселя не существует соответствующего сёрфеля, то его потенциально можно добавить в массив S_{live}. Однако, чтобы отобразить его в S_{ref} при помощи обратного деформирующего отображения, нужно знать его множество ближайших соседей, которое невозможно определить без знания позиции. Чтобы разрешить это противоречие, предлагается отобразить граф деформирующего отображения в пространство текущего кадра и определять множество ближайших соседей в нём же. Однако могут возникнуть топологические ошибки, например, в множество ближайших соседей точки, лежащей на руке, попадёт вершина, принадлежащая лицу. Чтобы этого избежать,

предлагается определять ближайщую вершину $node_0$ к текущему сёрфелю, а затем выкидывать из его множества ближайших соседей все вершины, чьи деформирующие отображения не соотносятся с отображением вершины $node_0$.

- Затем происходит обновление массива S_{live}:
  - добавляются новые сёрфели, такие что сумма весов их ближайших соседей больше δ_{nn}, иначе они являются потенциальными выбросами, так как лежат слишком далеко,
  - удаляются уже существующие сёрфели, если на протяжении долгого времени они являлись нестабильными или они слишком похожи на своих соседей.

  После массив S_{live} при помощи обратного деформирующего отображения переходит в S_{ref}. Затем определяется подмножество новых вершин, которые не покрываются текущим графом деформирующего отображения, из них "сэмплятся" вершины и обновляется граф, пересчитываются ближайшие соседи.

3.3 Обновление параметров SMPL-модели

После обновления геометрии сцены старые параметры позы и формы человеческого тела могут не удовлетворять новой поверхности, поэтому предлагается их оптимизировать, минимизируя функцию энергии:

$$E_{shape}(\beta, \theta) = E_{sdata}(\beta, \theta) + E_{sreg}(\beta, \theta, \beta_{prev}, \theta_{prev}) \tag{12}$$

- Слагаемое E_{sdata} отвечает за несогласованность между текущей геометрией и параметрами β, θ:

$$E_{sdata} = \sum_{\bar{v} \in \bar{T}} \psi\Big(D\Big(W\big(\bar{v} + B_s(\beta) + B_p(\theta), J(\beta), \theta\big)\Big)\Big), \tag{13}$$

где $D(\cdot)$ – это функция, которая возвращает расстояние до ближайшей точки из пространства текущего кадра. Для этого находятся k

ближайших соседей (из массива S_{live}) заданной вершины, в том случае, если они не лежат на теле человека – возвращается 0, иначе – расстояние от заданной вершины до интерполированных с определенными весами ближайших соседей.

- Слагаемое E_{sreg} не дает новым параметрам сильно отклоняться от старых значений:

$$E_{sreg} = \gamma_1 ||\beta - \beta'||_2^2 + \gamma_1 ||\theta - \theta'||_2^2 \tag{14}$$

4 Архитектура решения

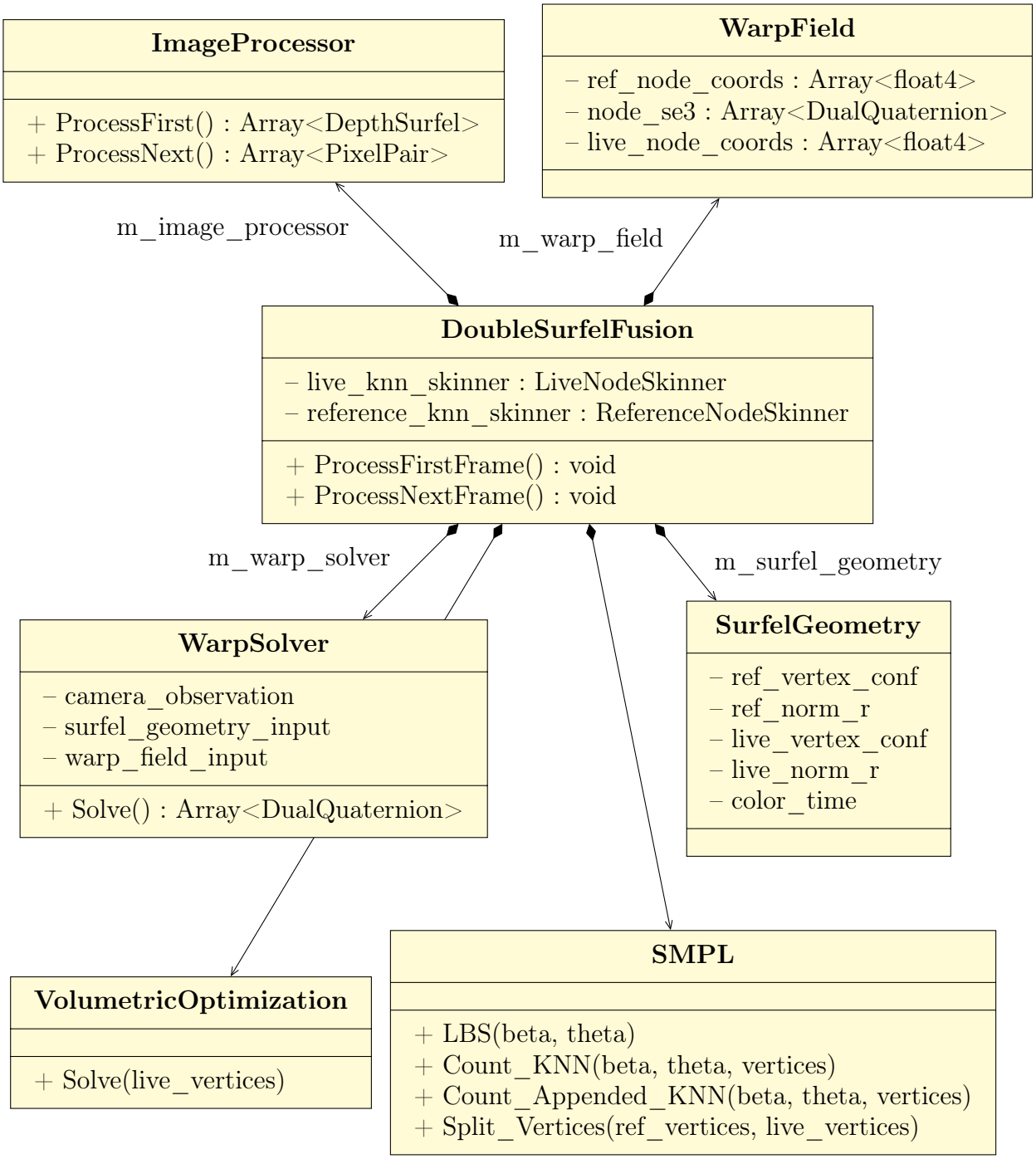


Рис. 1: Архитектура проекта

Диаграмму классов предлагаемого решения можно видеть на рисунке 1. Модуль `ImageProcessor` отвечает за обработку поступающих кадров – по первому кадру инициализирует массив сёрфелей, а для каждого следующего кадра строит массив соответствий $\mathcal{P}$ между текущей геометрией и

поступившим кадром глубины. Для получения массива соответствий $\mathcal{P}$ вычисляются видимые с текущей позиции камеры сёрфели и отбрасываются выбросы. Без потери общности мы всегда можем рассматривать камеру как фиксированную, рассматривая движение камеры как часть общего движения захваченного объекта. Таким образом, можем непосредственно спроецировать найденные сёрфели используя матрицу внутренних параметров камеры глубины K_d. Наконец, если для v – вектора позиции, характеризующего сёрфел, существует подходящий u_t с такими же координатами изображения в D_t, то пара (v, u_t) рассматривается как соответствие и сохраняется в $\mathcal{P}$. Эта схема основана на проективном алгоритме ICP [7].

`SurfelGeometry` хранит информацию о 2-х массивах сёрфелей – S_{ref} и S_{live} (S_{live} получено из S_{ref} при помощи деформирующего отображения по формуле 15). Позиция, нормаль, радиус и степень доверия сёрфелей хранятся оптимальным образом – по 4 числа с плавающей запятой (float), а цвет и время одинаковые для обоих массивов и хранятся отдельно.

`WarpField` хранит информацию о деформирующем отображении. Для этого из массива сёрфелей S_{ref} выбираются вершины, для которых хранится массив отображений `m_node_se3` в пространство текущего кадра. Модули `LiveNodeSkinner` и `ReferenceNodeSkinner` используются для вычисления деформирующего изображения в точке. Они содержат информацию о k ближайших соседях нужной вершины и их весах (влиянии), для которых деформирующее отображение (или обратное к нему) уже посчитано, что позволяет интерполировать значение в нужной вершине.

$$\begin{gathered} \mathcal{W}_t(x) = norm\left(\sum_{k \in N(x)} w_k\left(x\right) \hat{q}_k\right), \\ v_{live} = \mathcal{W}_t(v_{ref}) v_{ref} \\ n_{live} = rotation\Big(\mathcal{W}_t\left(v_{ref}\right)\Big) n_{ref} \end{gathered} \tag{15}$$

где $N(x)$ – k ближайших соседей вершины x, $w_k(x)$ – их веса, $\hat{q}_k$ – движение для вершины k, которое задается двойным кватернионом.

Модуль `WarpSolver` отвечает за вычисление деформирующего отображения, имея сведения о текущем кадре и соответствиях между геометрией

и кадром глубины (из `ImageProcessor`), о предыдущей геометрии и предыдущем деформирующем отображении.

`SMPL` хранит информацию о внутренних параметрах SMPL-модели человеческого тела. Функция LBS преобразовывает заранее заданные вершины и возвращает набор векторов, характеризующих человека, заданного параметрами формы β и позы θ. Для массива вершин сёрфелей можно найти ближайших соседей и их веса из набора предпосчитанных вершин SMPL-модели. Это используется для разделения сёрфелей на лежащие на человеческом теле и в окружающем мире, а так же для вычисления функции энергии в модуле `WarpSolver`.

`VolumetricOptimization` отвечает за оптимизацию параметров SMPL-модели β и θ так, чтобы модель человеческого тела лучше соответствовала обновленной геометрии.

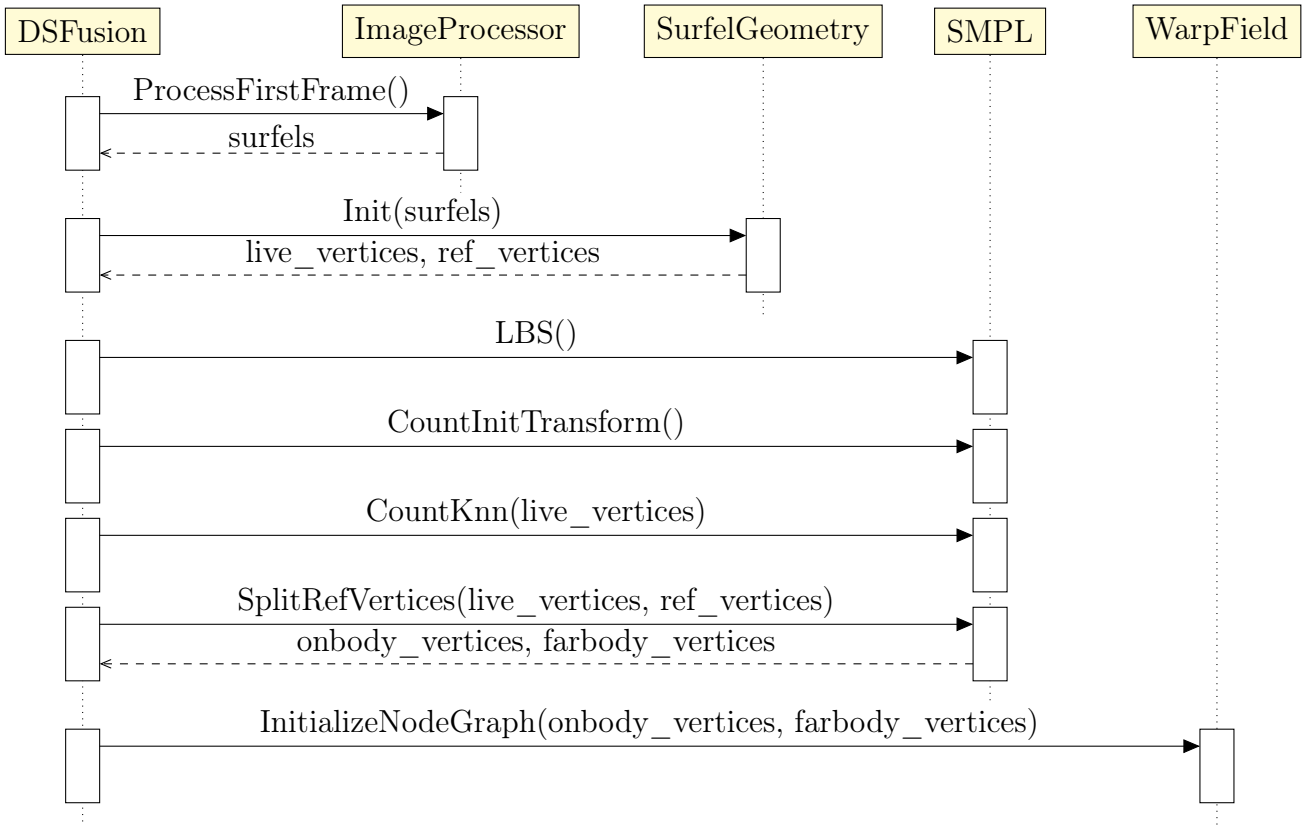


Рис. 2: Диаграмма последовательности для первого кадра

На рисунках 2 и 3 можно видеть диаграммы последовательностей для первого и последующих кадров.

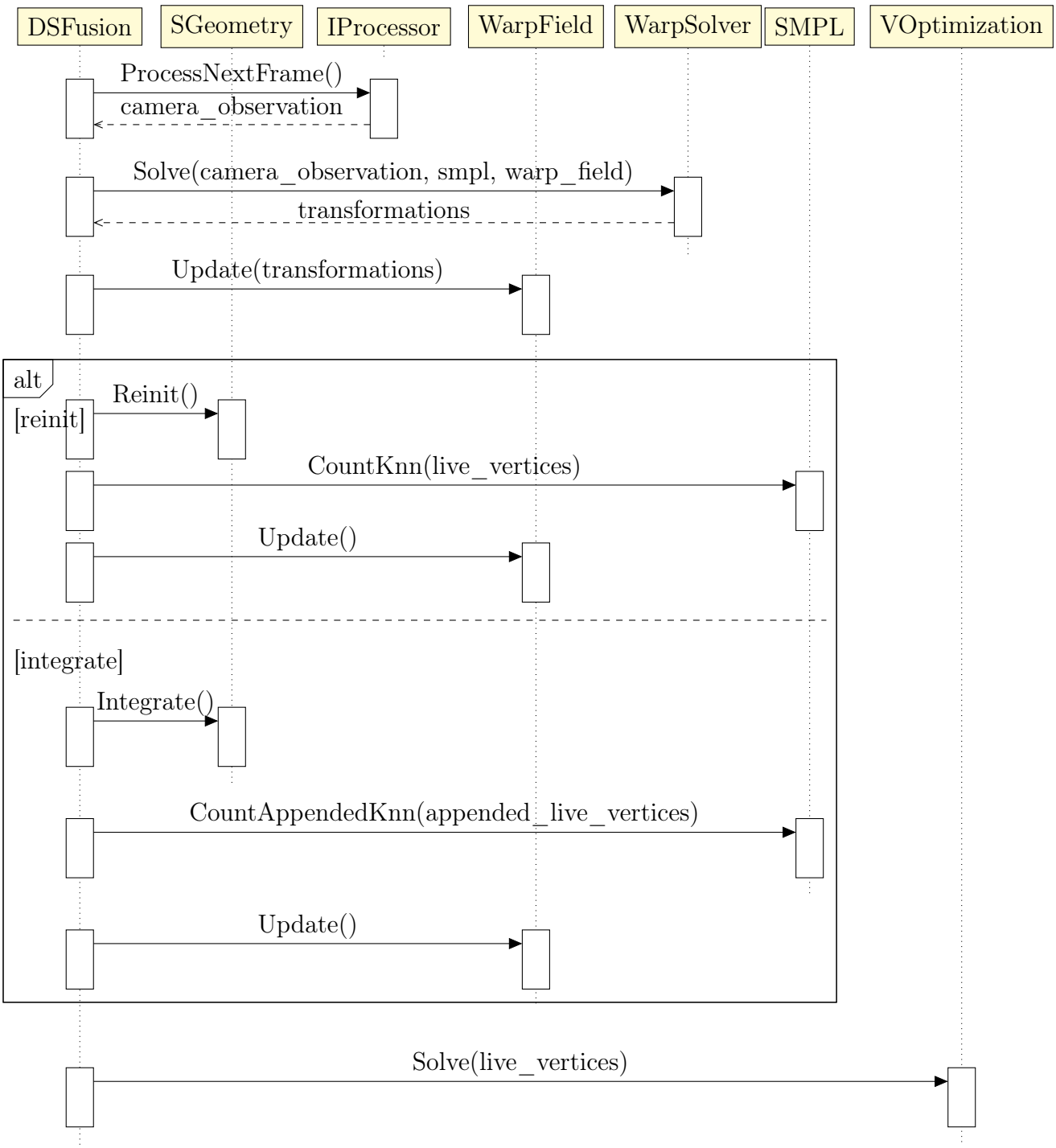


Рис. 3: Диаграмма последовательности для последующих кадров

5 Реализация

Для оптимальной скорости, производительности и простоты работы с видеокартой выбрано сочетание языка C++ и технологии CUDA. В качестве используемых библиотек выбраны: Eigen – библиотека линейной алгебры, cub – библиотека для работы с CUDA. Так же в работе используют-

ся открытые реализации статей SurfelWarp [2], и предсказания параметров SMPL-модели [3].

5.1 Реализация SMPL-модели

Реализация автора статьи [8] не поддерживает вычисления на GPU. Помимо неё было найдено несколько реализаций на Python с использованием тяжеловесных библиотек вроде tensorflow, от которых было решено отказаться. Была так же найдена реализация на C++, однако она в большей степени предназначена для подсчета моделей сразу для большого числа β и θ, что в рамках данной работы неактуально, так как на каждом шаге есть ровно один набор параметров, кроме того тот проект поддерживал не всю функциональность. Поэтому было принято решение реализовать SMPL-модель самостоятельно [4].

Параметры модели загружаются из файла в формате $json$, а затем копируются на GPU. Согласно оригинальной статье [8], построение 3D-модели состоит из четырёх стадий:

1. Генерация вершин $B_s(\beta)$, которые характеризуют форму человеческого тела, задаваемую параметрами β:

   ```
   Array<float> SMPL::shapeBlendShape(Array<float> beta);
   ```

   Генерация вершин $B_p(\theta)$, которые характеризуют форму человеческого тела в определённой позе, задаваемой параметрами θ:

   ```
   Array<float> SMPL::poseBlendShape(Array<float> theta);
   ```

   Вместе они задают форму человеческого тела $\bar{T} + B_s(\beta) + B_p(\theta)$.

2. Вращение суставов так, чтобы они соответствовали новой форме человеческого тела $J(\bar{T} + B_s(\beta))$

[2]Репозиторий проекта SurfelWarp: https://github.com/weigao95/surfelwarp (дата обращения: 30.05.2020)

[3]Репозиторий проекта Recovery of Human Shape and Pose: https://github.com/akanazawa/hmr (дата обращения: 30.05.2020)

[4]Репозиторий с реализацией SMPL-модели: https://github.com/NataliaPonomarevaMM/SMPL (дата обращения: 30.05.2020)

```
Array<float> SMPL::regressJoints(Array<float>
   shapeBlendShape)
```

3. Построение матриц поворота для всех вершин SMPL-модели (произведение из формулы (8)) по матрицам поворота для суставов и их новым положениям:

```
Array<float> SMPL::transform(Array<float> poseRotation,
   Array<float> joints)
```

4. Построение итоговых вершин, повёрнутых относительно суставов и сглаженных с определёнными весами по формуле (8):

```
Array<float> SMPL::skinning(Array<float> transformations,
   Array<float> weights, Array<float> vertices)
```

Кроме того, реализованы функции для вычисления ближайших соседей и соответствующих весов для набора сёрфелей: при инициализации и при слиянии с новыми сёрфелями. Для этого вычисляются расстояния до всех вершин SMPL-модели, и находятся кратчайшие четыре, которые и считаются ближайшими соседями. Так же, в зависимости от того, превышает ли кратчайшее расстояние от сёрфеля до SMPL-модели некоторое заранее заданное значение, определяется, лежит ли сёрфел на теле человека или в окружающем мире.

В таблице 1 представлено среднее значение и среднеквадратичное отклонение времени работы реализованных функций в миллисекундах.

Функция	Кол-во векторов	Среднее время (мс)	σ (мс)
lbs	6890	1.04866	0.098197
knn	53238	8.1845	0.876185
appended_knn	100	0.3283	0.361386

Таблица 1: Сравнение производительности реализованных функций

5.2 Интеграция с реализацией SurfelWarp

Инициализация SMPL-модели производится по начальным значениям параметров модели β_0 и θ_0, однако, после этого полученное облако точек необходимо соотнести с восстановленной с первого кадра геометрией сёрфелей. Среди возможных преобразований может быть поворот и параллельный перенос, поэтому оценивается 6 степеней свободы. Для этого используется жёсткий ICP алгоритм, реализованный в библиотеке cilantro.

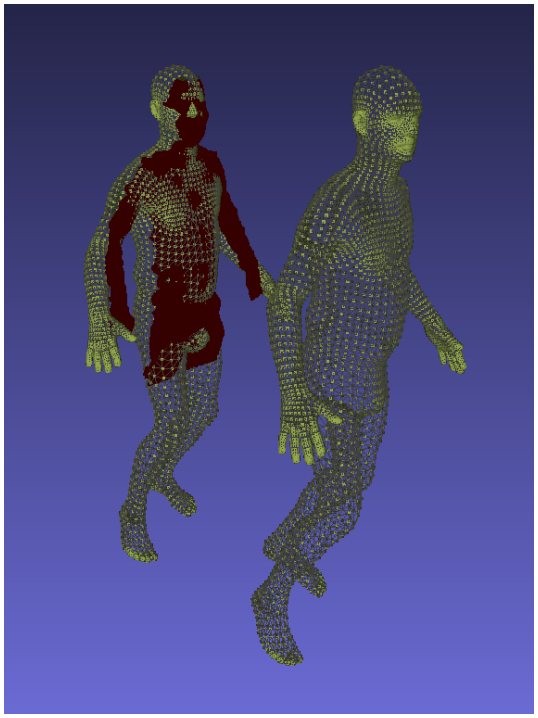


Рис. 4: Соотнесение облаков точек

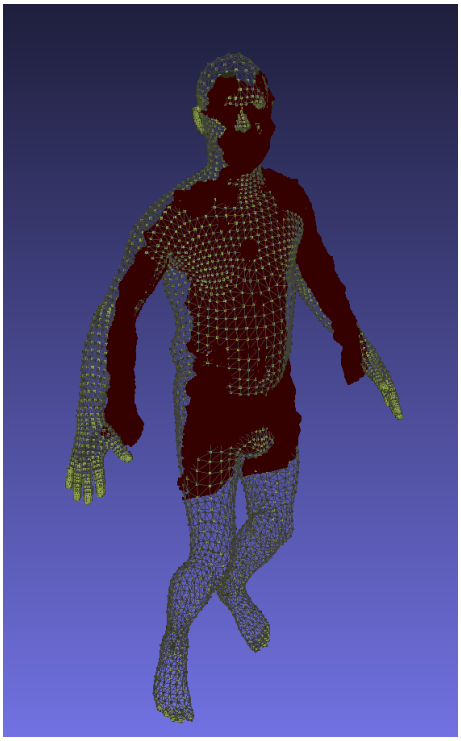


Рис. 5: Разделение точек

Затем все точки геометрии делятся на лежащие на теле человека или в окружающем мире. Это происходит при инициализации или же при повторной инициализации, которая происходит при наступлении некоторых условий во избежание ошибок при резких движениях или топологических изменениях. При добавлении же новых сёрфелей производится расчет ближайших соседей среди вершин SMPL-модели. Для них определяется, лежат ли они на человеческом теле, однако пересчета для старых сёрфелей не производится для ускорения времени работы. На рисунках 4, 5 можно увидеть пример соотнесения начального облака точек и SMPL-модели, а

так же разделение точек на лежащие на теле человека и в окружающем мире.

Так же в реализацию SurfelWarp была добавлена оптимизация параметров SMPL-модели. Для оптимизации функции энергии по формуле 13 необходимо найти пары вида (v_{smpl}, v_{live}), где v_{smpl} – вершина SMPL-модели, а v_{live} – ближайший сёрфел, причем лежащий на теле человека. Для этого так же используется поиск ближайших соседей, а затем происходит выборка только тех точек, для которых ближайший сосед ближе заранее заданного растояния. По этим парам и производится дальнейшее вычисление якобиана.

Для оптимизации параметров θ_t в процессе поиска деформирующего отображения и при дальнейшей оптимизации θ_t, β_t на t-ом кадре используется метод Гаусса-Ньютона. Пусть $X_t = [\beta_t, \theta_t] \in \mathbb{R}^D$ – это вектор параметров, которые оптимизируются, а $f(X_t) \in \mathbb{R}^C$ – это вектор остатков, причем минимизировать будем сумму квадратов компонент данного вектора: $\sum f_i^2(X_t)$. Тогда, начав с некоторого приближения $X_t^{(0)}$, будем последовательно выполнять итерации: $X_t^{(k+1)} = X_t^{(k)} - \alpha^{(k)}(\mathbf{J}^{\mathrm{T}}\mathbf{J})^{-1}\mathbf{J}^{\mathrm{T}} f(X_t^{(k)})$, где $\mathbf{J}$ – это якобиан функции f. Для этого на каждом шаге алгоритма считается якобиан, затем решается система $\mathbf{J}^{\mathrm{T}}\mathbf{J}(X_t^{(k+1)} - X_t^{(k)}) = \mathbf{J}^{\mathrm{T}} f(X_t^{(k)})$ при помощи метода сопряжённых градиентов, реализованного в библиотеке Eigen. Для обеспечения более надежной сходимости метода знак $\alpha^{(k)}$ выбирается таким образом, чтобы минимизировать вектор остатков $f(X_t^{(k+1)})$. Кроме того, для устойчивости и уменьшения сильных колебаний было взято $\alpha^{(k)} = 0.5$. В таблице 2 можно видеть эффективность работы метода.

Номер итерации	Значение минимизируемой функции
1	0.000381194
2	0.00023513
3	0.000153111
4	0.0000583831
5	0.0000432061
6	0.0000337034

Таблица 2: Пример работы метода для одного кадра

	Среднее время (мс)	σ (мс)
Вычисление якобиана	1.137	0.144848
1 итерация	4.3015	0.130088
6 итераций	25.9985	0.560912

Таблица 3: Время работы оптимизации для k-ого кадра

В таблице 3 приведено среднее время работы одной операции вычисления якобиана; одной итерации оптимизационного процесса, которая включает в себя вычисление якобиана, поиск нового $X_t^{(k+1)}$ и дважды вычисления вектора остатков $f(X_t^{(k+1)})$ для выбора знака α_k; всей оптимизации для одного кадра, состоящей из 6 итерации.

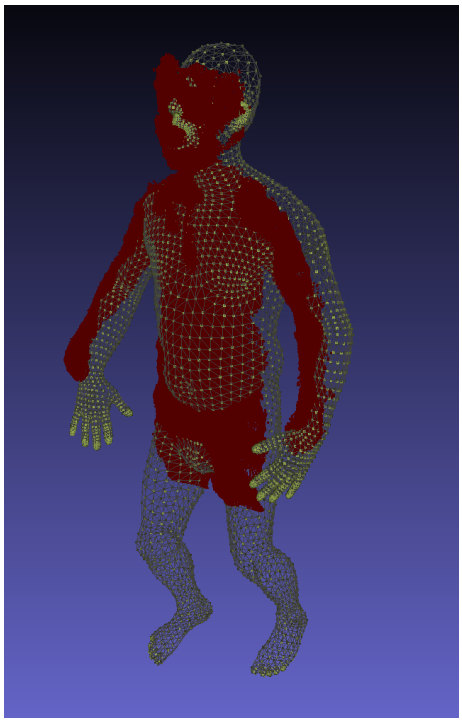


Рис. 6: Начальное положение модели

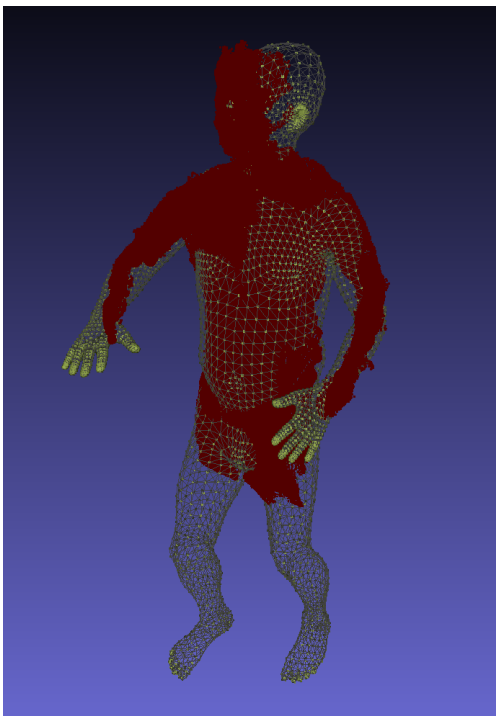
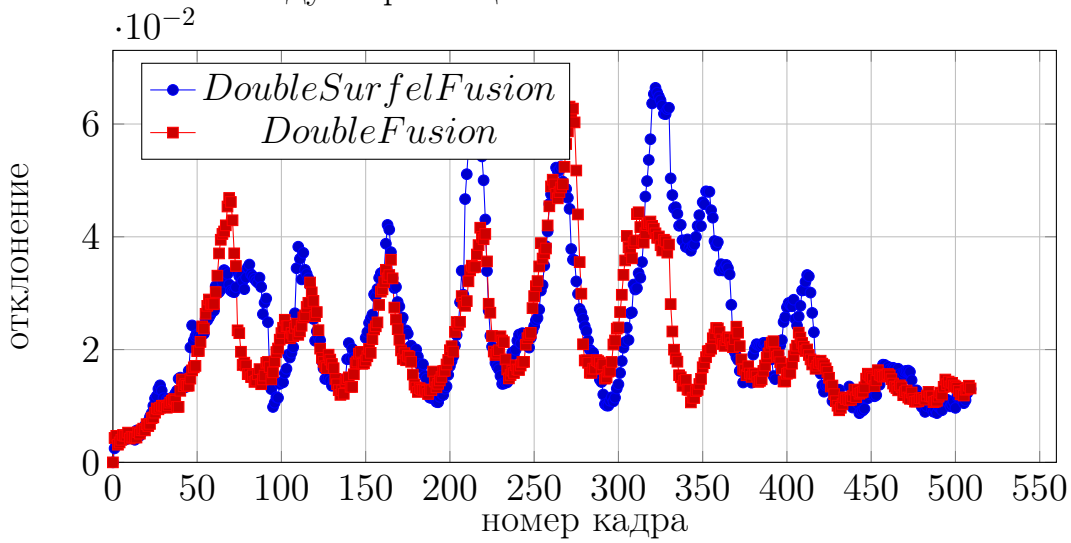


Рис. 7: Оптимизация при движении рук

На рисунках 6, 7 приведен пример оптимизации положения тела человека в движении.

6 Тестирование

Тестирование было произведено на BodyFusionVicon датасете, который содержит RGBD изображения и захваченные верные положения суставов человека на видеопоследовательности. Для всех кадров была посчитаны модули разниц между верными положениями суставов и полученными при помощи алгоритма, а затем за отклонение для каждого кадра была принята максимальная по модулю разница.



На графике 6 можно видеть посчитанные метрики для двух реализаций – предложенной в рамках данной работы и DoubleFusion (предложенной в статье [1]). Можно сделать вывод, что результаты отличаются незначительно, и предложенный алгоритм удовлетворяет необходимой точности.

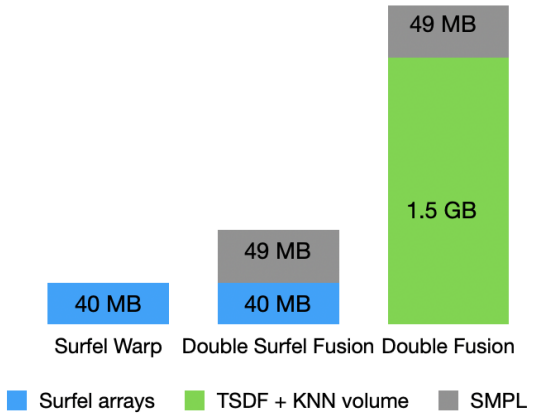


Рис. 8: Сравнение производительности по памяти

На рисунке 8 приведен график сравнения потребляемой памяти в те-

чение всего времени работы алгоритмов. Из него видно, что для хранения SMPL-модели и сопутствующих ресурсов необходимо больше памяти, чем этого требовала реализация SurfelWarp, однако это существенно меньше объема памяти, необходимого для хранения объемных структур данных в алгоритме DoubleFusion. Если сравнивать рассматриваемые алгоритмы по времени, то реализация SurfelWarp требует в среднем 16 мс на кадр, а оптимизация SMPL-модели добавляет еще 25 мс, что в сумме составляет в среднем 41 мс на кадр для предложенного алгоритма. Однако он уступает по времени работы реализации DoubleFusion, которой требуется в среднем 31 мс на кадр. Для устранения этого недостатка предлагаемый метод можно в дальнейшем улучшать путём ускорения вычисления якобиана или использования альтернативной реализации метода сопряжённых градиентов.

Заключение

В ходе работы были достигнуты следующие результаты:

- произведен обзор следующих алгоритмов реконструкции физических моделей людей по видеоизображениям: DoubleFusion, SurfelWarp, DynamicFusion,
- разработан алгоритм, объединяющий идеи представления тела человек в SMPL-модели и эффективного хранения информации об окружающем мире при помощи сёрфелей,
- разработана архитектура решения,
- реализована SMPL-модель человеческого тела на языке C++, с использованием технологий CUDA,
- SMPL-модель интегрирована с реализацией SurfelWarp,
- решение протестировано на датасете BodyFusionVicon.

Список литературы

[1] DoubleFusion: Real-Time Capture of Human Performances with Inner Body Shapes from a Single Depth Sensor / Tao Yu, Zerong Zheng, Kaiwen Guo et al. // 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2018. — P. 7287–7296.

[2] End-to-end Recovery of Human Shape and Pose / Angjoo Kanazawa, Michael J. Black, David W. Jacobs, Jitendra Malik. — 2018.

[3] Gao Wei, Tedrake Russ. SurfelWarp: Efficient Non-Volumetric Single View Dynamic Reconstruction. — 2019. — 04.

[4] KillingFusion: Non-rigid 3D Reconstruction without Correspondences / Miroslava Slavcheva, Maximilian Baust, Daniel Cremers, Slobodan Ilic. — 2017. — 07.

[5] Li Chao, Zhao Zheheng, Guo Xiaohu. ArticulatedFusion: Real-time Reconstruction of Motion, Geometry and Segmentation Using a Single Depth Camera // CoRR. — 2018. — Vol. abs/1807.07243.

[6] Newcombe R. A., Fox D., Seitz S. M. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time // 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2015. — June. — P. 343–352.

[7] Rusinkiewicz Szymon, Levoy Marc. Efficient Variants of the ICP Algorithm // Efficient Variants of the ICP Algorithm. — 2001. — 02. — P. 145–152.

[8] SMPL: A Skinned Multi-person Linear Model / Matthew Loper, Naureen Mahmood, Javier Romero et al. // ACM Trans. Graph. — 2015. — Oct. — Vol. 34, no. 6. — P. 248:1–248:16.

[9] VolumeDeform: Real-time Volumetric Non-rigid Reconstruction / Matthias Innmann, Michael Zollhöfer, Matthias Nießner et al. // CoRR. — 2016. — Vol. abs/1603.08161.