Санкт–Петербургский государственный университет

Михельсон Маргарита Николаевна

Выпускная квалификационная работа

Применение Bundle Adjustment и лендмарок в 3D-трекинге головы на основе деформируемой 3D-модели

Уровень образования: бакалавриат
Направление: 02.03.01 «Математика и компьютерные науки»
Основная образовательная программа СВ.5152.2019 «Математика, алгоритмы и анализ данных»

Научный руководитель:
доцент, Факультет математики и компьютерных наук СПбГУ, к. ф.-м. н.
Авдюшенко Александр Юрьевич

Рецензент:
инженер-программист ООО «ЖивойСофт»
Назаренко Владимир Владимирович

Санкт-Петербург
2023 г.

Содержание

Введение

Трехмерный трекинг головы — это задача отслеживания на видео положения головы в трехмерном пространстве и деформаций, отвечающих за выражение лица. Трекинг лица применяется для различных целей, как то создание визуальных эффектов в киноиндустрии, накладывание масок на видео в реальном времени, создание цифровых аватаров и так далее. В зависимости от целей варьируются требования, выдвигаемые к решению: в одних случаях ключевым фактором может быть скорость работы и полный автоматизм, в других упор делается на точность решения.

В рамках данный работы трекинг лица рассматривается применительно к киноиндустрии, что определяет специфику задачи. Трекинг должен быть как можно более точным; настолько, чтобы обмануть зрителя, даже если тот смотрит видео в хорошем разрешении на большом экране. На данный момент полностью автоматические алгоритмы трекинга не дают требуемую точность, поэтому для достижения нужного результата приходится вручную итерациями корректировать работу алгоритма, при необходимости запуская его несколько раз.

Ошибки, возникающие по ходу трекинга, можно условно разделить на две категории: сползание и дрожание. При сползании решение остается визуально гладким, но его качество падает со временем. Такую ошибку, пока она не успела существенно накопиться, зрителю не так просто заметить. Осуществляющий же трекинг пользователь может ее исправить, уточнив решение на том кадре, где ошибка по его мнению перестала быть приемлемой. Дрожание, в свою очередь, проявляется в том, что на каждом кадре возникают хаотичные ошибки, которые возможно и меньше ошибки сползания, но выливаются в неприятное глазу дергание. Зритель хорошо замечает эти рывки, с точки же зрения пользователя, исправить дрожание намного сложнее, поскольку для этого приходится поправлять трекинг почти на каждом кадре.

Даже незначительное улучшение качества трекинга может оказаться большим шагом вперед, если оно удлиняет последовательность кадров, на которых алгоритм выдает удовлетворительное с точки зрения пользователя качество. Действительно, большая часть времени, затраченная на трекинг ли-

ца, приходится именно на ручные действия, а не на работу алгоритма, так что возможность реже уточнять решение существенно сэкономит пользователю силы, время и нервы. С другой стороны, если алгоритм сможет на протяжении длинного отрезка кадров сохранять пусть и недостаточную, но не катастрофически маленькую точность, пользователю будет проще поправить результат, что также сохранит его время.

К моменту начала выполнения данной работы уже был реализован алгоритм покадрового трекинга, основанный на отслеживании на изображении двумерных точечных особенностей — точек на изображении, для которых можно определить перемещение от кадра к кадру. Этот алгоритм обеспечивает гладкое решение, но подвержен ошибке сползания.

Целью данной работы является улучшение алгоритма, для чего рассматриваются два подхода: добавление в него 2D ландмарок и использование Bundle Adjustment. Ландмарки — определенные точки на лице, определяемые нейросетью — призваны уменьшить возникающую в процессе работы алгоритма ошибку сползания, а Bundle Adjustment — метод уточнения решения благодаря одновременной оптимизации всех кадров — эксплуатирует ту особенность рассматриваемой задачи, что все кадры известны до начала трекинга.

1. 3D трекинг лица

1.1. Постановка задачи 3D трекинга лица

Трехмерный трекинг лица — это задача отслеживания на видео положения головы в трехмерном пространстве относительно камеры и деформаций, отвечающих за выражение лица.

Входными данными являются:

- Видео, состоящее из RGB кадров с номерами $t = 1, \ldots, T$, на котором есть голова человека.

- Трёхмерная модель $\mathcal{M}$ головы этого человека, задающаяся набором вершин $V = (v_1, \ldots, v_n)$, $v_i \in \mathbb{R}^3$ и набором треугольных граней $F = (f_1, \ldots, f_m)$, $f_i = (f_{i1}, f_{i2}, f_{i3})$, $f_{ij} \in \{1, \ldots, n\}$. Точка на поверхности модели $P \in \mathcal{M}$ задается гранью f_i, на которой она лежит, и барицентрическими координатами относительно этой грани.

- Набор деформаций $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_{|\mathcal{D}|})$, каждая из которых ставит в соответствие каждой вершине модели трехмерный вектор: $\mathcal{D}_k = (w_{k1}, \ldots, w_{kn}), w_{ki} \in \mathbb{R}^3$. Для параметров деформации $D = (d_1, \ldots, d_{|\mathcal{D}|})$ деформированная модель $\mathcal{M}(D)$ будет содержать тот же набор граней F, что и исходная модель $\mathcal{M}$, а координаты ее вершин будут вычисляться как
$$\tilde{v}_i = v_i + \sum_{k=1}^{|\mathcal{D}|} d_k w_{ki}$$

- Положение модели в трехмерном пространстве относительно камеры $Pose_1$, состоящее из вращения и параллельного переноса, и параметры деформации D_1 на первом кадре.

- Внутренние параметры камеры C, задающиеся матрицей 3×3:
$$C = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

Замечание 1. Вначале опишем как точка с поверхности модели проецируется на плоскость кадра. Для положения модели $Pose$ и параметров деформации D проекция $\pi(P, Pose, D)$ точки $P \in \mathcal{M}$ на плоскость кадра определяется следующим образом:

1. *Вычисление 3D точки.*

   Пусть P лежит на грани f_i модели и имеет барицентрические координаты (α, β, γ). Рассматриваются вершины деформированной модели $\mathcal{M}(D)$, образующие интересующую нас грань f_i, и трёхмерная точка, соответствующая P, вычисляется как

   $$\tilde{P} = \alpha\tilde{v}_{f_{i1}} + \beta\tilde{v}_{f_{i2}} + \gamma\tilde{v}_{f_{i3}}$$

   То есть это точка с теми же барицентрическими координатами относительно грани деформированной модели.

2. *Применение движения.*

   К точке $\tilde{P} \in \mathbb{R}^3$ применяется вращение $R \in SO(3)$ и параллельный перенос $tr \in \mathbb{R}^3$, соответствующие $Pose$:

   $$\bar{P} = R\tilde{P} + tr$$

3. *Проекция на изображение.*

   Полученная на предыдущем шаге точка $\bar{P} \in \mathbb{R}^3$ проецируется на плоскость кадра с учетом внутренних параметров камеры C:

   $$\begin{pmatrix} sx \\ sy \\ s \end{pmatrix} = C\bar{P}$$

   Таким образом, получается искомая точка на изображении $p = (x,\ y) \in \mathbb{R}^2$. Более подробную информацию о математической модели перспективной камеры можно найти, например, в [7].

   Целью трекинга является нахождение для каждого кадра $t = 1, \ldots, T$:

- положения модели относительно камер $Pose_t$
- параметров деформации D_t

1.2. Обзор подходов для решения задачи трекинга лица

Задача трекинга перекликается с задачей детектирования — позиционированием объекта в пространстве для одного кадра. Разумеется, задачу трекинга можно решить с помощью детектирования: достаточно спозиционировать объект независимо на каждом кадре. Однако обычно под трекингом понимается именно отслеживание движения объекта, а начальное положение считается известным. Благодаря тому, что входное изображение от кадра к кадру меняется не очень сильно, появляется дополнительная информация (по сравнению с задачей детектирования), которую можно и нужно использовать.

Итак, рассмотрим существующие подходы для решения задачи трекинга.

1.2.1 Классические методы компьютерного зрения

Под классическими методами компьютерного зрения применительно к задаче трекинга мы понимаем все подходы, не использующие машинное обучение. Такие методы извлекают из входных изображений некую информацию, строят на ее основе целевую функцию, оптимизируют ее и таким образом получают искомые положение головы и параметры деформации. Эти методы можно условно классифицировать по типу извлекаемой из изображений информации:

- точечные особенности [1]
- оптический поток [2]
- контуры [3]
- фотометрическая ошибка [4]
- комбинация вышеперечисленного [5]

Наиболее распространены методы из первой группы, поэтому опишем принцип их работы более подробно.

Ключевая точка (она же точечная особенность) — это точка изображения вместе со свой окрестностью, которая заметно отличается от других точек. Различными способами на кадрах входного видео вычисляются такие ключевые точки [8], [12] и определяется, какие из них, будучи на разных кадрах, соответствуют одной и той же трехмерной точке [9], [13]. Эти трехмерные точки на поверхности модели вычисляются в процессе трекинга на основе уже полученных положений модели в пространстве на предыдущих кадрах, и, таким образом, получаются новые соответствия между точками на модели и двумерными точками на следующих кадрах входного видео.

Чтобы спозиционировать модель на кадре, вводится ошибка репроекции: по положению модели в пространстве определяется несоответствие между спроецированными трехмерными точками и соответствующими им двумерными особенностями. Как правило, эта ошибка представляет собой сумму квадратов, которая итеративно минимизируется, в результате чего и определяется положение модели. Дополнительно могут использоваться такие техники как фильтр Калмана [6] или фильтры частиц.

1.2.2 2D ландмарки

Ландмарки — это точки в характерных частях лица (например уголки глаз, губ и т.д.), которые детектируются на изображении методами машинного обучения. Набор ландмарок может варьироваться в зависимости от выбранной нейросети, но он не зависит от входного видео, поэтому точки на модели, соответствующие ландмаркам, можно заранее разметить вручную. Таким образом, для каждого кадра входного видео получаются соответствия между двумерными точками на изображении и точками на поверхности модели, по которым, как и в случае точечных особенностей, можно спозиционировать модель. Более подробно этот процесс описан в разделе 3.1.

Были протестированы несколько наиболее известных нейросетей, находящихся в открытом доступе: MediaPipe [14], Fan2D68 [15], InsightFace [16]. Большинство из них хорошо справляется, когда лицо на изображении по-

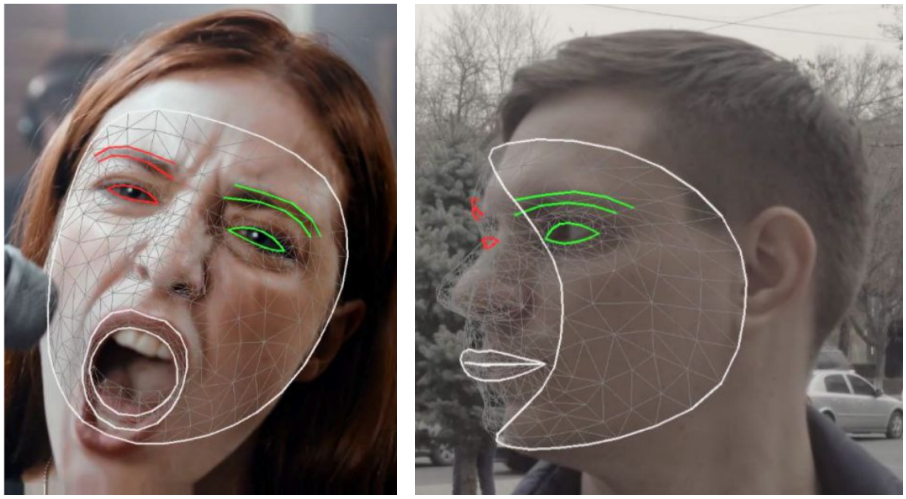

Рис. 1: Ландмарки MediaPipe. На левом изображении неточно определились подбородок и нижняя губа; на правом из-за ракурса в профиль плохо сидят нос и рот.

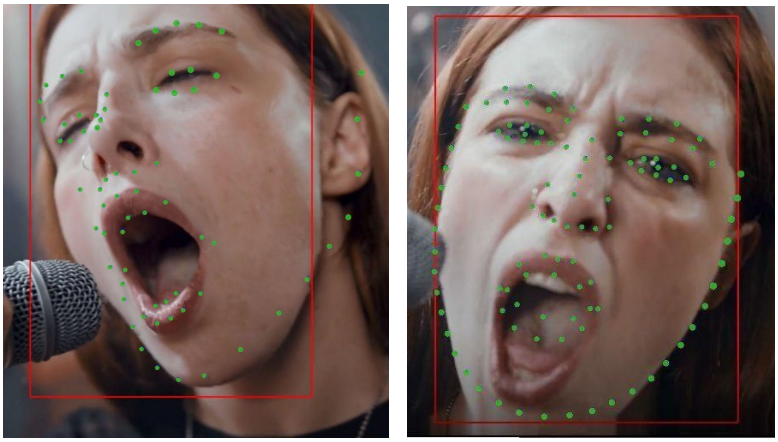

Рис. 2: Ландмарки InsightFace. Слева результат нейросети, выдающей 68 ландмарок, и снова ошибается контур лица; справа набор из 106 ландмарок, и неверно определился рот.

вернуто в фас и нет сильного проявления эмоций. В противном случае, как видно на иллюстрациях 1, 2, 3, они могут достаточно сильно ошибаться. Таким образом, становится понятно, что точности нейросетей недостаточно для желаемого качества трекинга.

1.2.3 3D ландмарки и End2End нейросети

В отличие от двумерных ландмарок (см. раздел 1.2.2), 3D ландмарки — это точки в трехмерном пространстве, а не в плоскости изображения. End2End нейросети, в свою очередь, по входному изображению выдают сразу поворот и параллельный перенос, соответствующие положению модели в пространстве, а также параметры деформации лица, на которых обучалась данная нейросеть.

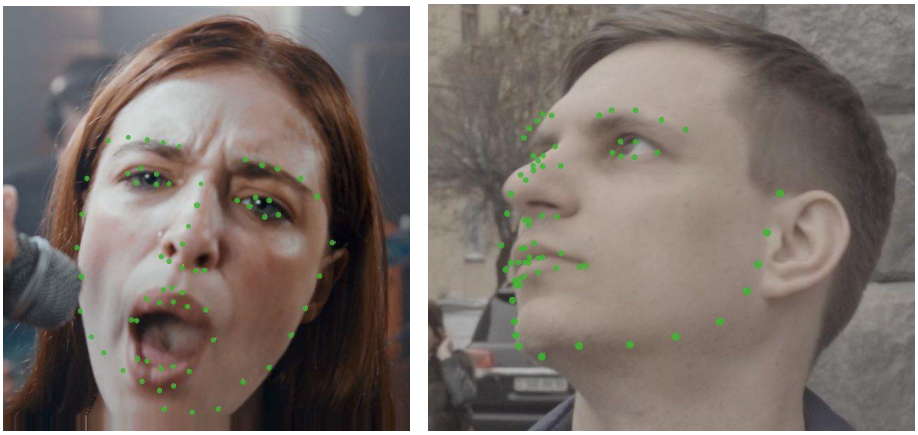

Рис. 3: Ландмарки Fan2D68. Слева ошибается подбородок и верхняя губа; справа контур лица и бровь слетают на фон.

Такие методы подходят для полностью автоматического трекинга, но в нашем случае их применение может быть затруднительно. Задача трекинга была сформулирована для конкретной модели головы $\mathcal{M}$ и набора деформаций $\mathcal{D}$, поэтому и 3D ландмарки не попадут на поверхность $\mathcal{M}$, и параметры деформации, найденные нейросетью, не совпадут с искомыми D. В связи с этим, методы этой группы не облегчают задачу трекинга по сравнению с 2D ландмарками, а, возможно, наоборот переусложняют.

1.3. Цель работы

До начала данной работы данной работы уже был реализован алгоритм трекинга, основанный на отслеживании точечных особенностей классическими методами компьютерного зрения (см. раздел 2). Цель работы заключается в том, чтобы улучшить его качество с помощью 2D ландмарок и благодаря более полному использованию того факта, что информация обо всех кадрах известна до начала трекинга.

Отметим сразу, что задача обучения ландмарок перед нами не стоит, это отдельное направление исследований, мы же используем готовые ландмарки и фокусируемся именно на встраивании их в трекинг.

Как видно из раздела 1.2.2 использование ландмарок из коробки не позволяет достигнуть желаемой цели, поэтому были поставлены следующие задачи:

1. Проведение экспериментов с прямой оптимизацией по ландмаркам.

2. Реализация двухэтапного трекинга: на первом этапе в решении используются ландмарки, а на втором — точечные особенности.

3. Использование Bundle Adjustment для дальнейшего улучшения результата.

2. Описание исходного алгоритма трекинга

2.1. Основная идея алгоритма

Исходный алгоритм трекинга основан на отслеживании на изображении двумерных точек: исходя из их движения вычисляется движение модели.

Сначала входное видео обрабатывается, как описано в разделе 2.2, в результате чего получается набор треков — последовательностей 2D точек на разных кадрах, соответствующих одной трехмерной точке.

После этого последовательно перебираются кадры $t = 2, \ldots, T$ и для каждого из них сначала вычисляются новые 3D точки для некоторых треков на основе уже известных $Pose_{t-1}$ и D_{t-1}, затем строится набор соответствий между точками с поверхности модели и двумерными точками на кадре t и, наконец, оптимизируется функция ошибки, благодаря чему вычисляются $Pose_t$ и D_t. Подробное описание каждого шага алгоритма представлено в разделе 2.3.

2.2. Определение на видео двумерных особенностей

На первом кадре видео с помощью алгоритма Ши-Томаси [8] определяются локальные особенности и из них выделяются лучшие так, чтобы весь кадр был покрыт равномерно. После этого эти особенности отслеживаются на следующий кадр алгоритмом Лукаса-Канаде [9]. Некоторые особенности могли не отследиться, в таком случае в образовавшихся “пустых” местах на втором кадре доопределяются новые особенности. Далее особенности со второго кадра отслеживаются на третий, и так продолжается до конца видео.

Таким образом, для i-й трехмерной точки получается трек — набор двумерных точек $p_{i,t_\ell},\ p_{i,t_\ell+1}, \ldots, p_{i,t_r}$, где p_{it} находится на кадре t. Некоторые из этих треков попадают на фон, другие же соответствуют точкам на лице, но на данном этапе невозможно определить какой 3D точке соответствует данный трек, именно поэтому особенности отслеживаются сразу на всем кадре.

Следует отметить, что поскольку в процессе отслеживания локальной особенности неизбежно появляется ошибка, двумерные точки соответствуют

трехмерной не идеально, а с некоторой погрешностью, обычно увеличивающейся к концу трека.

2.3. Реализация непосредственно трекинга

Алгоритм трекинга последовательно обрабатывает кадры $t = 2, \dots T$ и для каждого из них проделывает следующие шаги.

1. *Вычисление новых 3D точек.*

   - На кадре $t - 1$ определятся область, в которую проецируется модель с учетом уже известных $Pose_{t-1}$ и D_{t-1}, и рассматриваются точки $p_{i,t-1}$, попавшие в эту область и не участвовавшие до этого в трекинге.
   - Для каждой из них определяется точка на поверхности модели $P_i \in \mathcal{M}$, соответствующая $p_{i,t-1}$ на кадре $t - 1$. Другими словами P_i находится из уравнения $\pi(P_i, Pose_{t-1}, D_{t-1}) = p_{i,t-1}$.

2. *Определение множества 3D-2D соответствий.*

   Введем обозначение $\mathcal{P}_t = \{i \mid \exists p_{it},\ \exists P_i\}$ — индексы точек, для которых известно как 2D положение на кадре t, так и положение на поверхности модели.

3. *Вычисление $Pose_t$ и D_t.*

   Вводится ошибка репроекции, которая по данному положению модели $Pose$ и параметрам деформации D определяет несоответствие между каждой двумерной точкой p_{it} и спроецированной точкой с поверхности модели P_i:

$$Err_t(Pose, D) = \sum_{i \in \mathcal{P}_t} ||\pi(P_i, Pose, D) - p_{it}||^2 \xrightarrow[Pose,\ D]{} \min$$

   Err_t итеративно минимизируется с помощью алгоритма Левенберга-Марквардта [10][11], в результате чего определяются искомые $Pose_t$ и D_t.

2.4. Причины возникновения ошибки в описанном алгоритме

В описанном алгоритме трекинга благодаря свойствам оптического потока и усреднению по большому количеству точек не возникает дрожания, но, к сожалению, быстро появляется ошибка сползания. Рассмотрим более подробно почему так происходит.

Одна из причин возникновения этой ошибки состоит в том, что каждая конкретная точечная особенность отслеживается от кадра к кадру все менее точно, и со временем может значительно отклониться от своего изначального положения.

Другая же проблема кроется в вычислении трехмерной точки. Полученная на шаге 1 точка P_i зависит не только от двумерной точки $p_{i,t-1}$, но и от положения модели $Pose_{t-1}$ и параметров деформации D_{t-1}, которые, если $t > 2$, были вычислены алгоритмом трекинга на предыдущих шагах, а следовательно могут содержать ошибку, которая выливается в ошибку определения P_i.

Точки P_i, вычисленные на первом кадре не подвержены этой ошибке, но соответствующие им треки рано или поздно заканчиваются, в результате чего по ходу работы трекинга все большую долю в ошибке будут занимать слагаемые, соответствующие трехмерным точкам, вычисленным на следующих кадрах, то есть трехмерным точкам, вычисленным с ошибкой. Более того, для всех точек P_i, вычисленных на кадре t, эта ошибка будет систематической, поскольку для них всех использовалось одно и то же неточное значение $Pose_t$ и D_t. На следующих шагах трекинга эти точки будут тянуть оптимизацию к заведомо неверному положению, из-за чего ошибка при вычислении $Pose$ и D на следующих кадрах увеличится, новые трехмерные точки станут еще менее точными, и так далее.

Таким образом, ошибка трекинга быстро накапливается, положение головы становится настолько неточным, что на шаге 1 в оптимизацию включаются особенности с фона, и это окончательно сбивает трекинг.

3. Оптимизация по ландмаркам

В этом разделе описывается первый реализованный вариант использования ландмарок в трекинге — использование их в оптимизируемой функции ошибки. И хотя качество получившегося трекинга осталось ниже желаемого, некоторые из описанных идей пригодились для более эффективной модификации, описанной в разделе 4.

3.1. Совмещение ландмарок и деформируемой модели головы

Используемая нейросеть вычисляет на изображении 68 двумерных ландмарок, которые по аналогии с локальными особенностями будут обозначаться p_{it}. Отметим, что сами по себе ландмарки ничего не знают ни про нашу геометрию головы $\mathcal{M}$, ни про набор доступных деформаций $\mathcal{D}$, так что задача определения положения головы и параметров деформаций по ландмаркам не такая простая, как может показаться на первый взгляд.

Трехмерные точки $P_i \in \mathcal{M}$, соответствующие ландмаркам, были размечены вручную, и таким образом каждая ландмарка на кадре стала, равно как и локальная особенность, порождать соответствие между двумерной точкой и точкой на модели. Однако ландмарки, в отличие от особенностей, не подвержены ни одной из ошибок сползания, описанных в разделе 2.4: двумерные точки, соответствующие ландмаркам, определяются независимо на каждом кадре, и, следовательно, их качество не деградирует, а трехмерные точки-ландмарки фиксированы и не зависят от результатов трекинга на прошлых кадрах. Именно по этим причинам и предполагается, что ландмарки позволят улучшить трекинг.

Вычисление $Pose_t$ и D_t по ландмаркам сводится к оптимизации следующей функции ошибки:

$$Err_t(Pose, D) = \sum_{i \in \mathcal{Q}_t} ||\pi(P_i, Pose, D) - p_{it}||^2 \xrightarrow[Pose,\ D]{} \min$$

где $\mathcal{Q}_t$ — множество ландмарок на кадре t.

Однако, найденные таким образом $Pose_t$ и D_t не улучшают, а даже

ухудшают, качество трекинга. Так происходит потому, что ландмарки могут во-первых ошибаться, а во-вторых дрожать от кадра к кадру, что приводит к дрожанию решения. В связи с этим предлагается именно совместить ландмарки и точечные особенности, а не использовать что-то одно.

3.2. Алгоритм трекинга с ландмарками в оптимизации

Модифицируем алгоритм, предложенный в разделе 2.3, следующим образом. На шаге 3 будем использовать в ошибке репроекции как слагаемые, соответствующие точечным особенностям, так и слагаемые, соответствующие ландмаркам:

$$Err_t(Pose, D) = \sum_{i \in \mathcal{P}_t \cup \mathcal{Q}_t} \phi_i(||\pi(P_i, Pose, D) - p_{it}||) \xrightarrow[Pose,\ D]{} \min$$

где $\mathcal{Q}_t$ — множество ландмарок на кадре t, а ϕ_i — лосс-функция, $\phi(x) = x^2$.

Интуитивно это означает, что теперь положение и деформации головы подстраиваются так, чтобы как можно лучше соответствовать не только потоку, но и ланмаркам.

Тем не менее такой подход не дает желаемого улучшения качества трекинга, что связано в первую очередь с тем, что ландмарок мало а в оптическом потоке сотни или даже тысячи точек, поэтому вклад ландмарок в сумму оказывается незначительным. Эта проблема легко решается: слагаемым, соответствующим ландмаркам назначается вес, пропорциональный размеру $\mathcal{P}_t$, то есть $\phi_i(x) = C \cdot |\mathcal{P}_t| \cdot x^2$.

3.3. Взвешивание ландмарок в оптимизируемой функции

Возникает желание уточнить трекинг благодаря взвешиванию ландмарок и между собой тоже, поскольку среди них есть и стабильно более точные (глаза, нос), на которые хочется больше полагаться во время оптимизации, и стабильно менее точные (контур лица). Более того, веса ландмарок можно брать динамическими, зависящими от уверенности нейросети в данной ландмарке на данном кадре. Этим экспериментами и посвящен данный раз-

Рис. 4: Сравнение исходного трекинга (сверху), и трекинга с ландмарками (снизу) на кадрах 1, 25, 40 и 50

дел, хотя забегая вперед скажем, что ни один из предложенных вариантов взвешивания не оказался удачным.

3.3.1 Взвешивание ландмарок по группам

Разобьем ландмарки на группы (глаза, нос, рот и т.д.) и для каждой группы возьмем $\phi_i(x) = C_i \cdot x^2$, где C_i — вес для ландмарок из i-й группы. Для оценки C_i брались небольшие видео (длиной 20-40 кадров), лицо на них отслеживалось вручную, благодаря чему определялись истинные двумерные положения ландмарок. После этого строились графики отклонения ландмарки, найденной нейроетью, от ее истинного положения для каждой группы (см. рис.5).

Однако такое взвешивание не привело к улучшению трекинга по нескольким причинам. Во-первых, дисперсия ландмарки зачастую сильно зависит от ракурса. Во-вторых, для разных людей на разных видео дисперсия также отличается. В-третьих, ландмарки определяются заметно менее точно, когда соответствующая часть лица чем-то закрыта (occlusion). Выбранные же C_i не учитывают ни один из этих факторов.

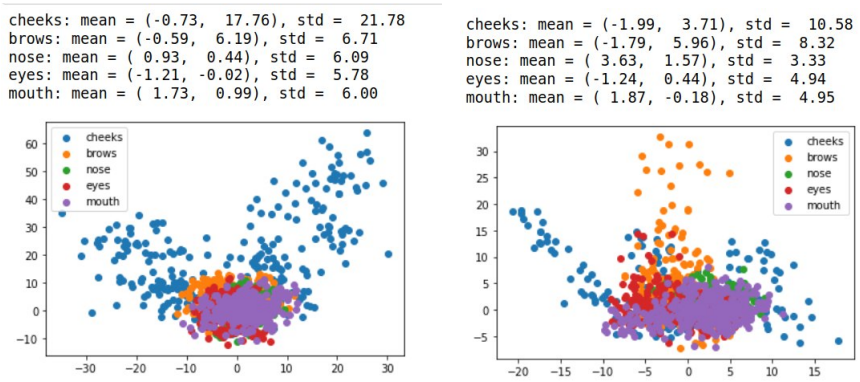

Рис. 5: Отклонение в 2D ландмарок от истинного положения для каждой группы: контур лица, брови, нос, глаза, рот

3.3.2 Выбор веса на основе уверенности нейросети

Используемая нейросеть по входному изображению выдает для каждой ландмарки не только ее координаты, но и двумерный массив вещественных чисел (карту), показывающий уверенность нейросети в том, что ландмарка находится в том или ином месте изображения (см. рис. 6).

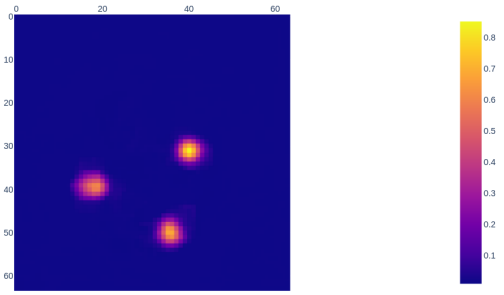

Рис. 6: Объединение карт, выданных нейросетью, для трех ландмарок

Один вариант использования таких карт заключается в том, чтобы подобрать на их основе тип функции ϕ_i, как описано в [17]. К сожалению, такой подход не дал результата, поскольку распределения, задаваемые картами, хорошо аппроскимируются нормальным распределением, что соответствует квадратичной функции ϕ, которая и использовалась изначально.

Другая, связанная с картами идея, заключается в том, чтобы динамически выбирать вес ландмарки в зависимости от уверенности в ней нейросети, где под уверенностью можно понимать, например, максимум карты или дисперсию соответствующего распределения. Чтобы такое взвешивание имело смысл, необходимо чтобы выбранная “уверенность” коррелировала с реаль-

ной точностью ландмарки, то есть расстоянием между найденным нейросетью и истинным положениями. На рисунке 7 представлены соответствующие графики, полученные на основе сгенерированных тестовых видео. Несмотря на наличие корреляции в обоих случаях, ни одна из рассмотренных функций не является хорошим кандидатом на вес ландмарки, поскольку на ее основании не отделить наиболее точные ландмарки от остальных. Нахождение же наименее точных ландмарок не представляет интереса, так как такие ландмарки будут автоматически определены на шаге 3 алгоритма.

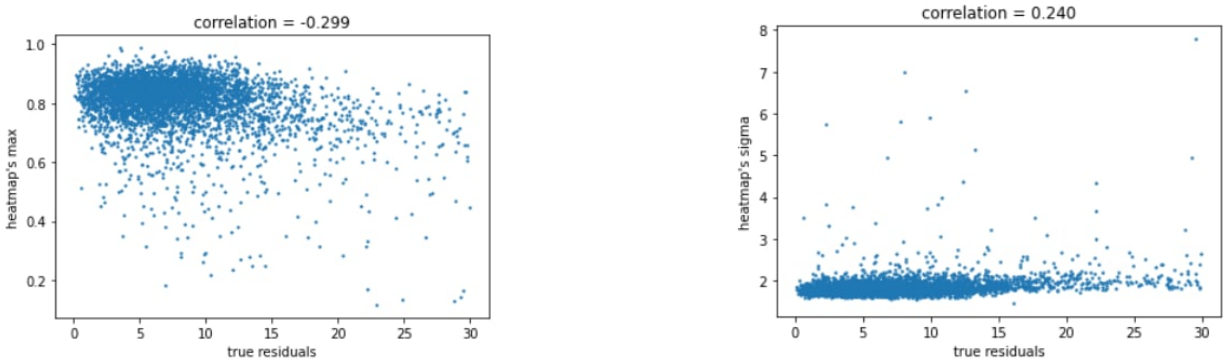

Рис. 7: Зависимость максимума карты (слева) и дисперсии (справа) от расстояния между истинным положением ландмарки и найденным нейросетью

Таким образом, взвешивание ландмарок в функции ошибки не привело к заметному улучшению качества трекинга, и решено было исследовать принципиально другие подходы.

4. Ландмарки как начальное приближение

Как уже обсуждалось в разделе 2.4 качество трекинга падает из-за неверно вычисляющихся точек на модели P_i, соответствующих локальным особенностям. Добавление ландмарок в оптимизацию, описанное в разделе 3 несколько замедляет накопление ошибки в $Pose_t$ и D_t, но основная проблема остается. Точки P_i вычисляются все менее точно, из-за этого, несмотря на ландмарки, локальные особенности тянут оптимизацию к неверным $Pose_t$ и D_t на следующем кадре, и ошибка быстро накапливается. Взвешиванием ландмарок эта проблема не решается главным образом потому, что ландмаркам приходится назначать слишком большой вес, а это делает решение чересчур неустойчивым к ошибкам ландмарок, что выливается в дрожание.

Рассмотрим другой подход к решению описанной проблемы. Сделаем алгоритм трекинга трёхэтапным. На первом этапе $Pose_t$ и D_t на всех кадрах будут грубо оцениваться исключительно по ландмаркам. На втором этапе по ним будут вычисляться трехмерные точки P_i для локальных особенностей. И, наконец, на третьем этапе трекинг будет запускаться повторно и использовать только локальные особенности с вычисленными на предыдущем этапе P_i.

Опишем теперь каждый этап более подробно.

1. *Вычисление всех $Pose_t$ и D_t по ландмаркам.*

   Для каждого кадра $t = 2, \ldots, T$ запускается оптимизация

   $$Err_t(Pose, D) = \sum_{i \in \mathcal{Q}_t} ||\pi(P_i, Pose, D) - p_{it}||^2 \xrightarrow[Pose,\ D]{} \min$$

   где $\mathcal{Q}_t$ — множество ландмарок на кадре t.

   Таким образом вычисляются $Pose_t^0$ и D_t^0 для всех t. Они оказываются не очень точными, зато не содержат ошибку сползания.

2. *Вычисление P_i.*

   Рассмотрим произвольный трек с двумерными точками $p_{i,\ell}, p_{i,\ell+1}, \ldots, p_{i,r}$. Для каждого кадра $t = \ell, \ldots, r$, вычислим точку $P_{it} \in \mathcal{M}$, соответствующую p_{it} на кадре t с учетом полученных на предыдущем этапе $Pose_t^0$

и D_t^0: P_{it} находится из уравнения $p_{it} = \pi(P_{it}, Pose_t^0, D_t^0)$. Не для всех кадров 2D точка $p_{i,t}$ может попасть на проекцию модели, так что получится облако точек $\{P_{it}\}$ для некоторых $t \in [\ell, \dots, r]$. Если бы особенности идеально соответствовали трехмерным точкам, и положения головы также были бы вычислены идеально, то вместо облака конечно получилась бы одна точка, но надеяться на такой расклад не приходится. Поэтому точки из облака усредняются, притягиваются к поверхности модели и полученная трехмерная точка P_i запоминается для данного трека. Если же $p_{i,t}$ не попала в проекцию модели ни на одном кадре, и облако получилось пустым, трек просто пропускается. Такая операция проделывается для всех треков.

3. *Вычисление финальных $Pose_t$ и D_t.*

   Наконец запускается исходный алгоритм трекинга, описанный в разделе 2.3, с тем лишь изменением, что шаг 1 пропускается, а на шаге 2 трехмерные точки P_i берутся с предыдущего этапа 2.

Главное достоинство получившегося алгоритма трекинга в том, что он сохраняет пусть и не всегда достаточное, но относительно приемлемое качество даже на очень длинных последовательностях кадров, где исходная версия трекинга не справляется с тем, чтобы выдать хоть сколько-то вразумительный результат (Рис. 8). С другой стороны, в случаях некоторых коротких видео данная модификация проигрывает исходному алгоритму (Рис. 9) из-за того, что ландмарки на изображении определяются хуже опического потока. Особенно хорошо это заметно на поворотах головы, где ландмарки часто ошибаются, причем ошибаются сходным образом на соседних кадрах, поскольку кадры похожи. В результате на этапе 2 точки P_i вычисляются с ошибкой, а это влечет за собой и ошибку трекинга на этапе 3.

Рис. 8: Сравнение исходного алгоритма трекинга (слева) и описанной модификации (справа), кадр 25 (сверху) и 60 (снизу) из 120

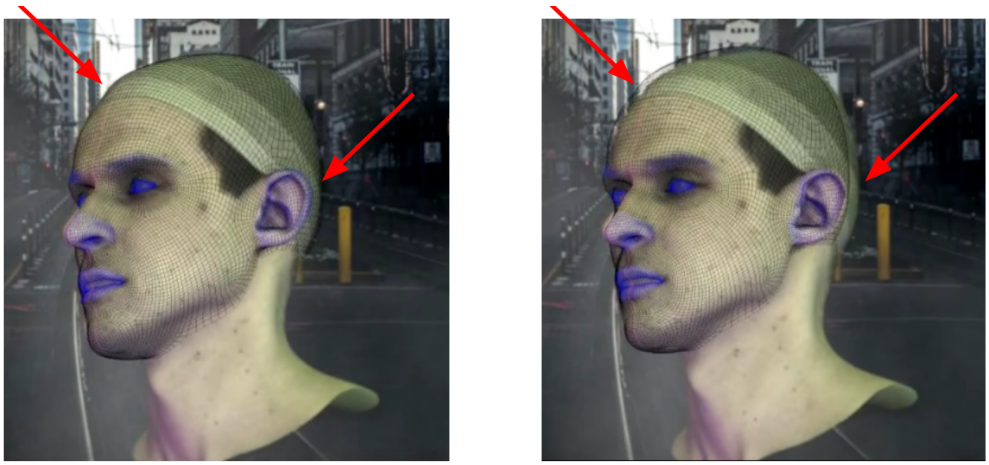

Рис. 9: Сравнение исходного алгоритма трекинга (слева) и описанной модификации (справа) на синтетическом тесте, кадр 10 из 80

5. Bundle Adjustment

До сих пор мы рассматривали исходный алгоритм трекинга и его модификации, которые вычисляют положение модели $Pose_t$ и параметры деформации D_t, по очереди запуская процесс оптимизации для кадров $t = 2, \ldots, T$. Основная же идея метода Bundle Adjustment [19] заключается в уточнении полученного решения благодаря одновременной оптимизации всех имеющихся параметров. Таким образом будет активно использоваться тот факт, что информация обо всех кадрах известна до начала трекинга.

В классической формулировке Bundle Adjustment используется для уточнения облака трехмерных точек и положений, с которых эти точки наблюдаются. Это базовый алгоритм в области фотограмметрии, использующийся в большинстве современных программ для трехмерной реконструкции.

Задача оптимизации для Bundle Adjustment может быть сформулирована как:

$$Err(Pose, P) = \sum_t \sum_{i \in \mathcal{P}_t} ||\pi(P_i, Pose_t) - p_{it}||^2 \xrightarrow[Pose,P]{} \min$$

Оптимизация ведется по положениям на всех кадрах $Pose = \{Pose_t\}$ и всем трёхмерным точкам $P = \{P_i\}$, $P_i \in \mathbb{R}^3$.

5.1. Model-Based Bundle Adjustment

Главное отличие нашей ситуации от описанной выше в наличии деформируемой модели $\mathcal{M}$. Это приводит, во-первых, к добавлению в оптимизацию параметров деформации на каждом кадре $D = \{D_t\}$, но намного более существенные изменения касаются точек P_i, которые теперь должны лежать не где угодно в пространстве, а строго на поверхности модели. Чтобы соблюсти это условие можно продолжить оптимизировать P_i как трехмерный вектор, но добавить регуляризацию, запрещающую точке сильно отклоняться от поверхности модели. Однако как утверждается в [20] лучшие результаты дает другой подход — параметризация 3D точек. Именно он и был реализован.

Итак, точка $P_i \in \mathcal{M}$ вычисляется теперь по двумерному вектору-параметру u_i:

$$P_i = \Psi_i(u_i),\ u_i \in \mathbb{R}^2$$

Задача минимизации таком случае записывается как

$$Err(Pose, D, U) = \sum_t \sum_{i \in \mathcal{P}_t} \phi(||\pi(\Psi_i(u_i), Pose_t, D_t) - p_{it}||) + Reg(D) \xrightarrow[Pose,D,U]{} \min \tag{1}$$

Здесь $U = \{u_i\}$ — множество оптимизируемых параметров точек (подробное описание параметризации представлено в разделе 5.2), а $Reg(D)$ — регуляризация на параметры деформации, ей посвящен раздел 5.3.

5.2. Параметризация точек на поверхности модели

Сформулируем основные критерии удобной для использования параметризации:

1. Каждой точке может соответствовать своя функция $\Psi_i(u)$, например Ψ_i могут выбираться из некоторого семейства.

2. Функция $\Psi_i(u)$ должна быть определена либо на всем $\mathbb{R}^2$, либо на простом его подмножестве, например прямоугольнике $[l_1, r_1] \times [l_2, r_2]$, для удобства оптимизации.

3. Функция $\Psi_i(u)$ должна быть (кусочно) дифференцируема, поскольку и в классическом Bundle Adjustment, и в исходном алгоритме трекинга используются методы оптимизации, для которых требуется дифференцируемость функции ошибки. От этих методов не хочется отказываться.

4. Образ $\Psi_i(\cdot)$ не обязан совпадать со всей поверхностью модели $\mathcal{M}$, однако если истинное положение i-й 3D точки не попадет в область значений Ψ_i, оптимизация гарантированно не сможет сойтись к истинному решению.

Подобрать такую параметризацию, которая удовлетворяет описанным выше требованиям, и которую при этом возможно реализовать на практике, оказывается не так просто.

Сначала была реализована параметризация по плоскости кадра (см. раздел 5.2.1), идея которой была почерпнута из [20]. После чего были проанализированы ее недостатки, и была реализована другая параметризация: по UV-развертке (см. раздел 5.2.2), лишенная большей части этих недостатков.

5.2.1 Параметризация по плоскости кадра

Опишем семейство функций $\{\Psi_{Pose,\ D}\}$, каждая из которых будет задаваться положением модели в пространстве $Pose$ и параметрами деформации D.

Рассмотрим произвольные такие параметры $Pose$ и D и мысленно спроецируем модель на плоскость кадра с учетом этих $Pose$ и D. Пусть теперь u — координаты в плоскости этого кадра. Определим $\Psi_{Pose,\ D}(u)$ как точку на модели, проекция которой совпадает с u. Если же таких точек несколько, из них выбирается ближайшая к камере. Более формально,

$$\Psi_{Pose,\ D}(u) = \underset{P \in \mathcal{M}}{\operatorname{argmin}}\{dist(P, C_0) \mid \pi(P, Pose, D) = u\}$$

где C_0 обозначает центр камеры.

Замечание 2 (Вычисление $\Psi_{Pose,\ D}(u)$). Для вычисления $\Psi_{Pose,\ D}(u)$ не обязательно проецировать модель на изображение, достаточно уметь находить пересечение геометрии нашей модели и луча из центра камеры в пиксель изображения, соответствующий u.

Замечание 3 (Область определения $\Psi_{Pose,\ D}$). Функция $\Psi_{Pose,\ D}$ определена только для тех точек $u \in \mathbb{R}^2$, которые лежат в образе $\pi(\cdot, Pose, D)$. Это множество в общем случае не описывается простым образом (например, оно не является прямоугольником), поэтому возникает необходимость доопределить $\Psi_{Pose,\ D}$.

Если точка $u \in \mathbb{R}^2$ не попала на проекцию модели, среди граней модели выбирается f_i — грань, проекция которой оказалась ближайшей к u — после чего луч из центра камеры в u пересекается с f_i. Получившаяся трехмерная точка не будет лежать на поверхности модели (она будет находиться лишь в плоскости, образованной f_i), зато $\Psi_{Pose,\ D}$ останется дифференцируе-

мой, и при небольших отклонениях u от проекции модели производная будет "тянуть" оптимизацию обратно, к корректному значению u.

Замечание 4 (Область значений $\Psi_{Pose,\ D}$). Функция $\Psi_{Pose,\ D}$ параметризует только ту часть модели, которую видит камера с учетом выбранного ракурса $Pose$ и деформаций D. Другими словами, не для любой точки $P \in \mathcal{M}$ существует параметр u, такой что $P = \Psi_{Pose,\ D}(u)$.

Теперь естественным образом встает вопрос выбора параметров $Pose$ и D для параметризации i-й трехмерной точки. Хорошим выбором были бы такие $Pose$ и D, что и исходная трехмерная точка, и истинная хорошо видны на соответствующем кадре, равно как виден и путь межу ними. Однако истинная точка нам неизвестна, мы можем лишь предполагать, что она достаточно близка к исходной. Поэтому мы хотим выбрать $Pose$ и D так, чтобы окрестность i-й исходной точки была бы хорошо видна, то есть повернута к камере.

Для параметризации каждой точки P_i выбирается свой кадр: из кадров начального приближения $\{Pose_t,\ D_t\}$ брется тот, на котором грань модели, содержащая исходную точку P_i, повернута к камере под углом, максимально близким к прямому. Таким образом, каждой P_i назначается своя функция $\Psi_{Pose,\ D}$.

Перечислим недостатки описанной параметризации:

- У функции $\Psi_{Pose,\ D}$ могут быть разрывы (например, если $Pose$ соответствует ракурсу в три четверти, малое изменение u может привести к тому, что точка "перескочит" с носа на щеку, см. рис. **??**). Это негативно влияет на сходимость.

- Параметр u может не соответствовать точке на поверхности модели, это также ухудшает сходимость.

- Истинное положение трехмерной точки может не соответствовать никакому параметру u.

5.2.2 Параметризация по UV-развертке

Рассмотрим другой вариант параметризации 3D точки. Для этого нам понадобится UV-развертка [18] — понятие, широко использующееся в компьютерной графике. Более формально, UV-развертка $\mathcal{U}$ для каждой грани модели $f_i = (f_{i1},\ f_{i2},\ f_{i3})$ задает координаты этой грани на плоскости: $q_i = (q_{i1},\ q_{i2},\ q_{i3})$, $q_{ij} \in \mathbb{R}^2$.

Функция $\Psi : \mathcal{U} \to \mathcal{M}$ по двумерным координатам u определяет грань q_i, в которую попадает u, затем вычисляет барицентрические координаты относительно q_i и возвращает точку на грани модели f_i с этими барицентрическими координатами. То есть если раньше параметр u задавал координаты в плоскости кадра, то теперь это uv-координаты, а функция $\Psi_i = \Psi$ преобразует их в точку на поверхности модели.

Рис. 10: UV-развертка.

Заметим, что у произвольного объекта, с одной стороны, вполне может быть сколь угодно много UV-разверток, а с другой стороны, если развертка не дана во входных данных, то сгененрировать ее — отдельная задача. Мы остановили свой выбор на развертке, изображенной на рис. 10 по нескольким причинам:

1. На лице она непрерывна, что важно для оптимизации ошибки. Единственный же разрез на затылке, через который u не сможет “перескo-

чить" во время оптимизации, не является существенной проблемой, поскольку исходная задача заключается в трекинге именно лица, то есть затылок в кадр почти никогда не попадает.

2. У этой развертки большая площадь приходится непосредственно на лицо.

Кроме непрерывности, у параметризации по UV-развертке есть еще одно преимущество по сравнению с параметризацией по плоскости кадра — отображение $\Psi : \mathcal{U} \to \mathcal{M}$ сюръективно, то есть истинному положению 3D точки всегда будет соответствовать какой-то параметр u. Таким образом нет необходимости выбирать Ψ_i для каждой точки свою, как это было в параметризации по плоскости кадра. Одна описанная выше функция Ψ годится для параметризации любой точки.

Тем не менее, одна старая проблема остается: Ψ определена не на всем прямоугольнике, изображенном на рис. 10, а только на его подмножестве (черным областям на изображении не соответствует никакая точка на модели). Как и в случае параметризации по плоскости кадра Ψ приходится доопределить: если u не попадает ни в какой треугольник на UV-развертке, берется ближайший к u треугольник q_i, и барицентрические координаты вычисляются относительно него. Да, точка окажется не на поверхности модели, но такое будет происходить только для точек на затылке, которые можно предварительно отфильтровать.

5.3. Регуляризация

Еще одна особенность предлагаемого Bundle Adjustment заключается в том, что необходима регуляризация на параметры деформации. Деформации заметно увеличивают как количество аргументов в оптимизации, так и гибкость модели, что приводит к тому, что модель стремится неестественно изогнуться, чтобы как можно лучше подстроится под данные. В связи с этим вводится регуляризация:

$$Reg(D) = C \sum_t \sum_k d_{tk}^2$$

где $D_t = (d_{t,1}, \ldots, d_{t,|\mathcal{D}|})$ и $d_{t,k}$ — коэффициент применения k-й деформации на кадре t, а C — константа, подобранная эмпирически.

Параметры $D = (0, \ldots, 0)$ соответствуют нейтральному выражению лица, а чем большее значение принимает $|d_{tk}|$, тем сильнее деформируется лицо, и тем менее вероятно встретить его во входных данных. Если предположить, что деформации описываются нормальным распределением, предложенная регуляризация соответствует как раз логарифму вероятности встретить данные параметры деформации. Минимизация ошибки в формуле 1 в таком случае становится эквивалента максимизации апостериорной вероятности в Байесовском подходе [21].

Рис. 11: Результат работы исходного алгоритма трекинга (слева), его модификации с ландмарками в качестве начального приближение (в центре) и Bundle Adjustment (справа) на кадрах 1, 30 и 60.

6. Тестирование

Главная сложность, сопряженная с тестированием, заключается в том, что для произвольного видео с лицом человека не существует идеально точного решения задачи трекинга. Это связано с тем, что, во-первых, модель головы $\mathcal{M}$ не идеально приближает голову реального человека, а, во-вторых, набор деформаций $\mathcal{D}$ не идеально приближает возможные деформации лица. Поэтому для оценки качества алгоритмов брались как искусственно сгенерированные тестовые видео, для которых известны истинные положения модели и деформации на каждом кадре, так и реальные, где "истинным" считалось решение, построенное экспертом вручную.

Для сравнения качества решений используется описанная ниже метрика. Для каждого кадра t рассматриваются полученные алгоритмом значения $Pose_t$ и D_t а также истинные $\widehat{Pose}_t$ и $\hat{D}_t$. После этого для каждой вершины модели v вычисляется положение в пространстве, соответствующее этим $Pose_t$ и D_t, и рассматривается расстояние до ее истинного положения. Эти расстояния усредняются по всем вершинам:

$$\delta_t = \frac{100\%}{d_{\mathcal{M}}} \frac{1}{N} \sum_{v \in V_{\mathcal{M}}} ||P(v, Pose_t, D_t) - P(v, \widehat{Pose}_t, \hat{D}_t)||$$

Здесь $P(v, Pose, D)$ — функция, возвращающая по точке v на поверхности модели соответствующую ей точку $\bar{v} \in \mathbb{R}^3$, полученную с учетом деформаций модели D и ее положения в пространстве $Pose$ (см. 1), N — количество вершин у $\mathcal{M}$, а $d_{\mathcal{M}}$ — ее диаметр.

После этого строится график, показывающий для какой доли кадров ошибка δ_t не превосходит данное значение, и рассматривается значение AUC — площади подграфика — чем оно больше, тем точнее решение.

На графике 12 представлены результаты работы исходного алгоритма трекинга, алгоритма с ландмарками, описанного в разделе 4, и Bundle Adjustment для двух сгенерированных тестов. В первом случае использование ландмарок улучшает трекинг, тогда как во втором (правый график) качество ухудшается из-за того, что, как уже было упомянуто, ландмарки ошибаются во время поворота головы (правый график построен для того же теста, кадры

из которого приведены на рисунке 9). Так или иначе, использование Bundle Adjustment увеличивает AUC по сравнению с исходным алгоритмом в первом случае на 10%, а во втором на 18%.

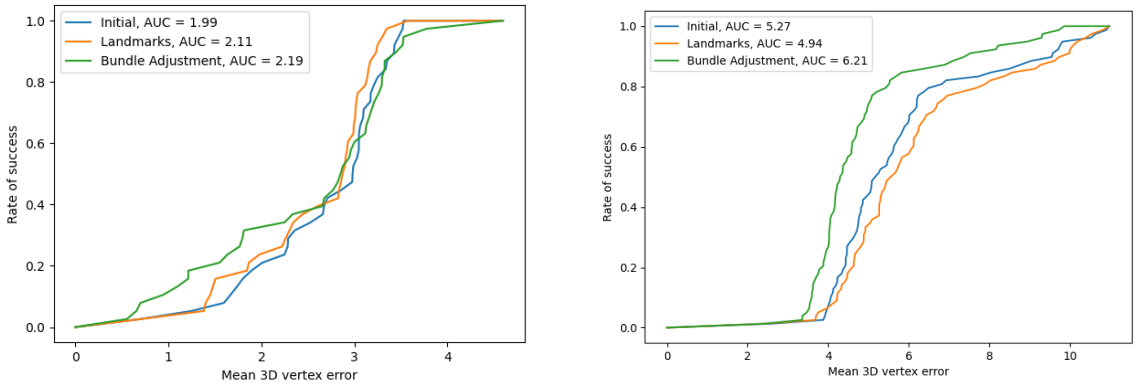

Рис. 12: Сравнение исходного алгоритма трекинга, алгоритма с ландмарками, описанного в разделе 4, и последующим его улучшением с помощью Bundle Adjustment. Результаты для двух различных искусственно сгнерированных видео: 40 кадров (слева), 80 кадров (справа).

На графике 13 проиллюстрировано преимущество параметризации трехмерных точек по UV-развертке по сравнению с параметризацией по плоскости кадра. И, наконец, график 14 показывает, что Bundle Adjustment хорошо исправляет даже не очень точное начальное приближение (AUC увеличивается в 2.7 раза).

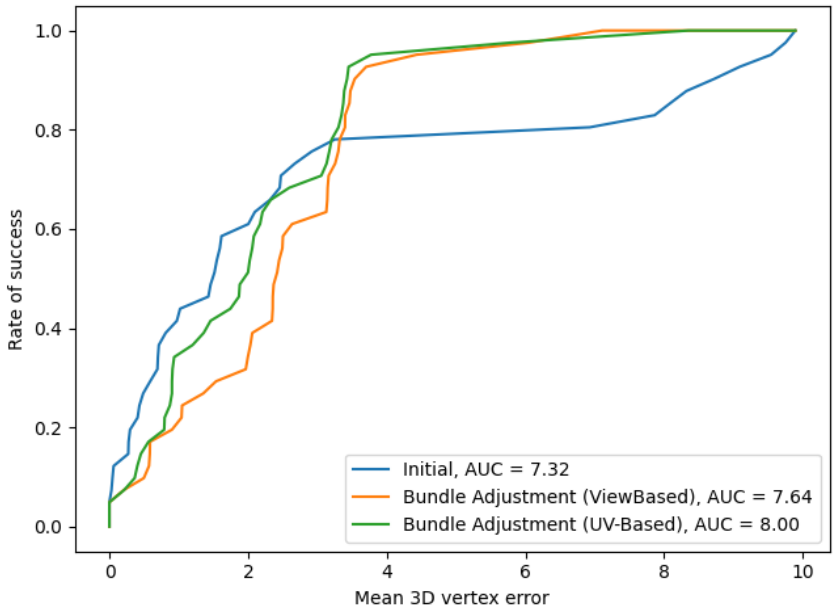

Рис. 13: Сравнение исходного алгоритма трекинга и Bundle Adjustment с параметризацией по плоскости кадра и по UV-развертке на реальном видео.

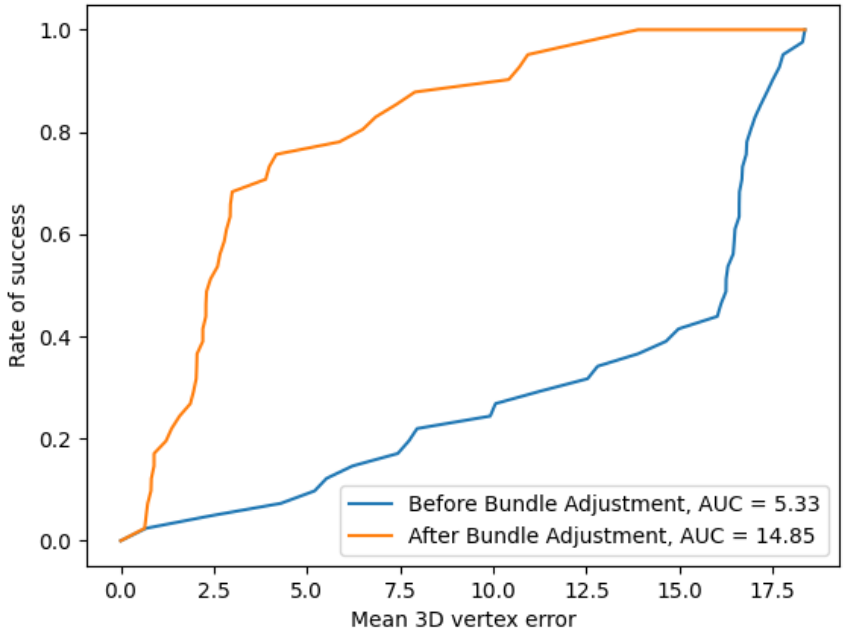

Рис. 14: Bundle Adjustment, запущенный на реальном видео с достаточно неточным начальным приближением.

Заключение

В данной работе рассматривалась задача трехмерного трекинга лица на основе деформируемой модели. Сначала был описан реализованный до начала выполнения работы алгоритм трекинга, основанный на отслеживании на видео точечных особенностей. Его недостатки были проанализированы, исходя из чего были предложены и реализованы два независимых его улучшения: первое использует 2D ландмарки, второе — Bundle Adjustment.

Использование в трекинге 2D ландмарок позволяет уменьшить накапливающуюся в процессе работы алгоритма ошибку. Чем длиннее входное видео, тем заметнее это улучшение, однако на некоторых коротких видео данная модификация может проигрывать исходному алгоритму из-за неточного определения на изображении ландмарок.

Bundle Adjustment же, используемый для уточнения уже имеющегося решения, показал хорошие результаты как на искусственно сгенерированных тестовых видео, так и на реальных. Отметим, что использование этого метода оказалось возможно благодаря специфике рассматриваемой задачи — тому факту, что решение не требуется строить в реальном времени, то есть информация обо всех кадрах известна до начала работы трекинга.

Логичным продолжением исследований может стать совмещение двух описанных подходов. Отметим, что эта задача не такая простая, как может показаться на первый взгляд. Модификация трекинга, использующая ландмарки, лучше всего себя показывает на длинных видео, в то время как Bundle Adjustment, наоборот, имеет определенные ограничения на длину входного видео, поскольку с длиной видео растет количество параметров в задаче оптимизации. В связи с этим остается простор для дальнейших экспериментов, связанных, например, с выбором подмножества кадров для запуска на них Bundle Adjustment, чтобы с одной стороны выиграть во времени, а с другой не потерять в качестве решения.

Список литературы

[1] L. Vacchetti, V. Lepetit and P. Fua, "Stable real-time 3D tracking using online and offline information,"in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 10, pp. 1385-1391, Oct. 2004

[2] S. Basu, I. Essa, and A. Pentland, “Motion regularization for model-basedhead tracking,” International Conference on PatternRecognition(Vienna,Austria), 1996.

[3] Pengfei Han, Gang Zhao, A review of edge-based 3D tracking of rigid objects, Virtual Reality Intelligent Hardware, Volume 1, Issue 6, 2019, pp 580-596

[4] M. Cascia, S. Sclaroff, and V. Athitsos, “Fast, reliable head tracking undervarying illumination: An approach based on registration of texture-mapped 3Dmodels,”IEEE Transactions on Pattern Analysis and Machine Intelligence,vol. 22, April 2000.

[5] D. DeCarlo and D. Metaxas, “Optical flow constraints on deformable modelswith applications to face tracking,”International Journal of Computer Vision,vol. 38, pp. 99–127, 2000.

[6] Z. Shaik and V. Asari, "A Robust Method for Multiple Face Tracking Using Kalman Filter,"36th Applied Imagery Pattern Recognition Workshop (aipr 2007), Washington, DC, USA, 2007, pp. 125-130, doi: 10.1109/AIPR.2007.21.

[7] Lepetit V., Fua P. Monocular Model-Based 3D Tracking of Rigid Objects: A Survey. Foundations and Trends in Computer Graphics and Vision. — 2005.

[8] Shi Jianbo, Tomasi Carlo. Good Features to Track 1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR’94). — 1994. — P. 593–600.

[9] B. D. Lucas and T. Kanade (1981), An iterative image registration technique with an application to stereo vision. Proceedings of Imaging Understanding Workshop, pages 121–130

[10] Levenberg K. (1944). A Method for the Solution of Certain Non-Linear Problems in Least Squares. Quarterly of Applied Mathematics. 2 (2): 164–168.

[11] Marquardt D. (1963). An Algorithm for Least-Squares Estimation of Nonlinear Parameters. SIAM Journal on Applied Mathematics. 11 (2): 431–441

[12] Lowe D. (1999). "Object recognition from local scale-invariant features"(PDF). Proceedings of the International Conference on Computer Vision. Vol. 2. pp. 1150–1157

[13] Baumberg A. (2000) “Reliable feature matching across widely separated views,” in Conference on Computer Vision and Pattern Recognition, pp. 774–781

[14] MediaPipe https://developers.google.com/mediapipe/solutions/vision/face_landmarker/

[15] Face Alignment https://github.com/1adrianb/face-alignment

[16] InsightFace https://github.com/deepinsight/insightface

[17] Saragih, J.M., Lucey, S. Cohn, J.F. Deformable Model Fitting by Regularized Landmark Mean-Shift. Int J Comput Vis 91, 200–215 (2011).

[18] Wikipedia. UV mapping. Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/UV_mapping (online; accessed: 25.05.2023).

[19] Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W. (2000). Bundle Adjustment — A Modern Synthesis. In: Triggs, B., Zisserman, A., Szeliski, R. (eds) Vision Algorithms: Theory and Practice. IWVA 1999. Lecture Notes in Computer Science, vol 1883. Springer, Berlin, Heidelberg.

[20] Ying Shan, Zicheng Liu and Zhengyou Zhang, "Model-based bundle adjustment with application to face modeling,"Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001, Vancouver, BC, Canada, 2001, pp. 644-651 vol.2.

[21] Wikipedia. Bayesian inference. Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Bayesian_inference (online; accessed: 25.05.2023).