Санкт-Петербургский Государственный Университет

Программная инженерия

Володин Вадим Евгеньевич

Библиотека для отслеживания экранов устройств в видео

Выпускная квалификационная работа бакалавра

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

Рецензент:
руководитель проектов ООО "Системы Компьютерного Зрения"
Пенкрат Н.А.

Санкт-Петербург
2020

SAINT-PETERSBURG STATE UNIVERSITY

Software Engineering

Vadim Volodin

A library for tracking screens of devices on video

Graduation Thesis

Scientific supervisor:
Candidate of Engineering Sciences Yurii Litvinov

Reviewer:
Project Manager «Computer Vision Systems» Nikolai Penkrat

Saint-Petersburg
2020

Оглавление

Введение

В настоящее время активно развивается область киноиндустрии и видеорекламы. В этой области используются специальные эффекты компьютерной графики для дополнения реальности при съёмке фильмов и видео. При этом некоторые графические эффекты могут быть реализованы с помощью алгоритмов компьютерного зрения. Одной из задач, возникающих в компьютерном зрении и необходимых в киноиндустрии, является задача отслеживания объектов. С помощью решения данной задачи возможно наложить эффект компьютерной графики на объект в видео.

Для решения задачи отслеживания в индустрии используются графические редакторы, предоставляющие пользовательский интерфейс для работы с компьютерной графикой. Одним из примеров таких редакторов является *GeoTracker* [2], разработанный компанией *KeenTools*[1]. Такие инструменты позволяют значительно сократить время на производство фильма или видео, позволяя накладывать эффекты на объекты на видео. Однако, недостатком таких инструментов является то, что результаты их работы необходимо поправлять для корректного наложения эффекта, что доставляет неудобства пользователю. Это говорит о недостаточной точности работы существующих алгоритмов отслеживания трёхмерных объектов. Необходимо улучшение качества подобных алгоритмов.

Экраны устройств являются частым объектом для отслеживания. Они обладают различными свойствами, которые могут быть использованы в алгоритмах и инструментах. Некоторые из этих свойств, такие как твердая прямоугольная рамка, выраженность сторон и монотонность рамки, могут упростить работу инструментов. Другие свойства, такие как изменчивость изображения, блики и отражения, могут ее усложнить. Существующие алгоритмы и инструменты рассчитаны на отслеживание произвольных объектов, не учитывая их свойства. Предполагается, что специализированный алгоритм, учитывающий свойства

[1]Домашняя страница компании KeenTools, URL: https://keentools.io/ (дата обращения: 30.05.2020)

экранов устройств, может повысить качество отслеживания для таких объектов.

Таким образом, необходимо разработать алгоритм, учитывающий свойства экранов устройств и реализовать его в виде библиотеки, которую можно будет использовать как в графическом редакторе, так и отдельно от него.

1. Постановка задачи

Целью данной работы является разработка и реализация алгоритма для отслеживания экранов устройств.

Для достижения цели были поставлены следующие задачи:

- провести обзор предметной области и рассмотреть алгоритмы и инструменты для отслеживания объектов;
- разработать алгоритм для отслеживания экранов устройств;
- реализовать алгоритм в виде библиотеки;
- произвести сравнение алгоритма с аналогами.

2. Обзор

В обзоре рассматривается задача отслеживания и её математические основы, метрики оценки качества алгоритмов отслеживания, существующие алгоритмы для решения задачи, а также используемые в работе инструменты.

2.1. Задача отслеживания

Рассмотрим процедуру наложения эффекта на объект, которым является экран. Он находится в модельной системе координат экрана, совпадающей с мировой системой координат. Для наложения эффекта используются точки эффекта в модельной системе координат и его текстура. Каждая точка эффекта проецируется на плоскость кадра с помощью модели перспективной проекции [4].

$$x = K[R|t]X,$$

где:

- x — точка на плоскости кадра;
- X — соответствующая ей точка в мировой системе координат;
- K — матрица внутренних параметров камеры, она зависит лишь от камеры и одинакова для всех кадров;
- R — поворот мировой системы координат относительно системы координат камеры;
- t — перенос мировой системы координат относительно системы координат камеры;
- $[R|t]$ — матрица внешних параметров камеры, она задаёт поворот и перенос модельной системы координат экрана относительно системы координат камеры, то есть трёхмерное положение экрана.

Для того чтобы наложить эффект на кадр, необходимо найти координаты эффекта на плоскости кадра. Это можно сделать с помощью матрицы внутренних параметров камеры, модельных координат объекта и трёхмерного положения по приведенной выше формуле.

Задачей отслеживания называется поиск трёхмерного положения по матрице внутренних параметров камеры и модельным координатам объекта, которые задаются пользователем. Приведенная формула кроме наложения эффекта может быть использована и для решения задачи отслеживания. Имея соответствия между точками в модельной системе координат и точками на плоскости кадра, можно найти трёхмерное положение экрана. Такая задача хорошо изучена и называется $PnP(Perspective\text{-}n\text{-}Point)$ [6]. Таким образом, задача отслеживания может быть решена с помощью нахождения координат объекта на плоскости кадра, соответствующих модельным координатам.

Для экрана искомыми точками могут быть четыре точки, которые являются углами экрана. Соответственно, можно найти координаты четырёх углов экрана на плоскости кадра, и затем по ним с помощью методов решения задачи PnP найти трёхмерное положение, имея которое можно спроецировать эффект на плоскость кадра с помощью модели перспективной проекции. Так задача отслеживания может быть сведена к нахождению координат углов экрана на плоскости кадра.

Имея трёхмерное положение экрана с предыдущего кадра и используя предположение о том, что экран находится близко к экрану на предыдущем кадре, с помощью алгоритма отслеживания можно найти точки экрана на данном кадре и затем трёхмерное положение на текущем кадре. Для этого необходимо, чтобы пользователь задал либо точки экрана на первом кадре, что можно сделать с помощью графического редактора, либо непосредственно трёхмерное положение на первом кадре.

Таким образом, на вход алгоритму подается видео, в котором необходимо отследить экран, матрица внутренних параметров камеры, соотношение сторон экрана (с помощью чего можно построить модель экрана в модельной системе координат). Также размечается первый

кадр видеопоследовательности: координаты четырёх углов экрана на плоскости кадра в пикселях.

Выходные данные алгоритма — трёхмерное положение для каждого кадра видео. После решения данной задачи эффект компьютерной графики может быть наложен на объект с помощью его проекции на кадр.

2.2. Метрики качества отслеживания

Метрики для оценки качества сравнивают трёхмерное положение, полученное алгоритмом отслеживания, и истинное трёхмерное положение. Метрики используют как изображение объекта, так и непосредственно трёхмерное положение, то есть вращение и перенос.

Метрика *projection error* [6] сравнивает координаты экрана на плоскости кадра, полученные с помощью проекции по трёхмерному положению. *Projection error* равна средней евклидовой дистанции между точками, полученными проекцией с помощью найденного трёхмерного положения и проекцией с помощью истинного трёхмерного положения по некоторому набору точек объекта. Например, по четырём углам экрана. Данная метрика отражает свойства алгоритма на изображении, но при этом не измеряет трёхмерное положение напрямую и не всегда выражает различия трёхмерных положений.

Следующая метрика — ошибка вращения. Разница между двумя вращениями также является вращением. Трёхмерное вращение выражается тремя двухмерными вращениями по различным осям. Утверждается [6], что можно найти единственные вектор и угол трехмерного вращения. Этот угол, который выражается в радианах, нормализуется на количество радиан, описываемых окружностью, и используется в качестве метрики.

Ошибка переноса является евклидовой дистанцией между переносом положения, полученным алгоритмом, и переносом истинного положения. При этом дистанция нормализована на диаметр объекта, то есть на длину диагонали экрана в модельной системе координат.

Метрика *reprojection error* [6] отличается от остальных тем, что для нее не требуется знание истинного положения. Хотя она и не отражает близость к истинному положению, эта метрика может быть использована для оценки соответствия модели взаимного расположения точек объекта. Использование может быть осуществлено в случае, если алгоритм производит поиск точек объекта до нахождения трёхмерного положения. Она отличается от *projection error* тем, что проекция по истинному положению заменяется на найденные алгоритмом точки объекта.

Метрика *tracking success rate* [11] объединяет ошибку вращения и ошибку переноса. Считается, что кадр был отслежен успешно, если ошибка вращения для него не больше конкретного порога и ошибка переноса не больше другого порога. Эта метрика достаточно распространена для задачи отслеживания и учитывает свойства работы пользователя с графическими редакторами. В связи с этим данная метрика и была выбрана в настоящей работе для оценки качества алгоритмов.

2.3. Методы отслеживания

Существует множество методов для отслеживания трёхмерного положения произвольного объекта. Обзор методов проводится в работах [6, 14]. Рассмотрим далее основные классы таких методов.

2.3.1. Маркерные методы

Маркеры — это визуальные образы, местоположение которых известно заранее и которые легко распознаются на изображении. Маркерные методы часто используются для съемки видео, в котором окружение известно и может быть изменено перед съемкой. Для этого на объекте нужно разместить маркер. Данный метод плохо подходит для задачи, так как не всегда возможно изменять сцену и размещать маркеры на объекте. Также это может вызвать необходимость переснимать сцену, если она была снята без маркеров.

2.3.2. Сопоставление шаблона

Отслеживание, основанное на сопоставлении шаблона. Для данного метода необходимо изображение объекта. Имея изображение объекта, можно найти его на другом изображении. Ограничение данного метода в том, что необходимо изображение экрана: в таком случае пользователю будет необходимо предоставлять данное изображение, это создает неудобство в работе с графическим редактором или другим инструментом. Также недостатком является то, что он плохо работает, когда объект меняется. На экране может смениться изображение. Смена ракурса, удаленность от камеры, блики и заслоны также значительно ухудшают качество метода.

2.3.3. Отслеживание особых точек

Другим методом является отслеживание особых точек на изображении. Особые точки — точки, которые выделяются по некоторым характеристикам среди остальных. Для устройства с экраном эти точки могут быть, например, углами экрана.

Одним из способов отслеживания особых точек является поиск и сопоставление дескрипторов этих точек. С помощью алгоритма получения дескриптора особых точки выделяются дескрипторы. Далее дескрипторы с разных изображений сравниваются друг с другом. Близкие дескрипторы говорят о схожести окрестностей точек. Так можно получить информацию о том, где находятся углы экрана на данном кадре, если известно их положение на предыдущем. Примером алгоритма вычисления дескрипторов особых точек является алгоритм *SIFT* [7]. Дескрипторы могут быть инвариантны относительно поворота, изменения яркости, масштаба и других величин.

Алгоритмы вычисления оптического потока также могут быть использованы для отслеживания особых точек. Примером таких алгоритмов является алгоритм Лукаса-Канаде [8], который использует предположение о том, что перемещение объекта за один кадр мало. В алгоритме происходит поиск закона, по которому переместились все пиксели,

находящиеся в окружении данной точки. Предполагается также, что яркость всех перемещенных пикселей мало отличается от кадра к кадру. В одной из модификаций алгоритма используется закон аффинного преобразования координат объекта.

Недостаток данных методов в том, что они не устойчивы к изменению изображения на экране, бликам и отражениям. Также для представленных алгоритмов не учитываются изменения, не подверженные закону аффинного преобразования.

2.3.4. Отслеживание контуров

Кроме методов отслеживания особых точек, существуют методы отслеживания контуров объекта. Примером такого алгоритма является *RAPiD* [3].

В алгоритме на предыдущем изображении определяется множество контрольных точек, распределенных по контуру объекта. На текущем изображении выделяются контуры изображения с помощью оператора *Кэнни* [1]. Для каждой контрольной точки производится поиск точки контура в направлении перпендикулярно касательной к контуру в обе стороны от него. Алгоритм хорошо показывает себя в случаях, когда важен контур объекта, а не его содержимое. Недостаток данного подхода в том, что при зашумленном изображении на экране и зашумленном фоне после применения оператора *Кэнни* на изображении находится множество линий, которые не являются стороной экрана, и точки на них могут быть выбраны методом вместо точек на контуре.

2.3.5. Преобразование Хафа

Одним из алгоритмов компьютерного зрения, который не является алгоритмом отслеживания, но использует свойство выраженности прямых линий, является алгоритм преобразования Хафа [13]. Данный алгоритм обнаруживает прямые линии на изображении, полученном после применения оператора *Кэнни* к исходному изображению. Прямая может быть задана уравнением $r = x \cdot cos(\theta) + y \cdot sin(\theta)$, где (x, y) —

декартовы координаты точки на изображении, (r, θ) — параметры прямой, которые можно интерпретировать как точки в полярной системе координат. Идея преобразования Хафа в том, чтобы учесть характеристики прямой не как уравнение, построенное по паре точек изображения, а в терминах её параметров. Исходя из этого прямая может быть представлена в виде точки в полярной системе координат. Таким образом, задача обнаружения прямых может быть сведена к задаче обнаружения точек.

В связи с тем, что вышеперечисленные алгоритмы не обладают достаточным качеством для удобства работы пользователя, для повышения качества было решено разработать отдельный алгоритм, который учитывает свойства экранов устройств. Важным свойством экранов является то, что рамка является прямоугольной, значит контуры экрана — прямые линии. При этом стороны экрана выражены на изображении. Эти свойства позволяют использовать преобразование Хафа для обнаружения линий. Таким образом, было решено использовать преобразование Хафа для разработки алгоритма отслеживания.

2.4. Инструменты

Язык программирования *Python* [9] часто используется для анализа данных, в том числе в компьютерном зрении. Он удобен для прототипирования, обширен по синтаксическому сахару. Программа может быть запущена в интерактивном режиме в интерпретаторе, таком как *Jupyter Notebook* [5], а также без интерактивного режима. На языке доступно множество различных библиотек для анализа данных. Учитывая эти преимущества, для работы был выбран язык *Python*.

Одной из доступных библиотек является *OpenCV* [10] — библиотека компьютерного зрения, в которой содержится множество алгоритмов. В библиотеке реализовано преобразование Хафа, с помощью которого можно найти прямые линии на изображении. *OpenCV* также предоставляет алгоритмы *Canny* для получения контуров на изображении, *solvePnP* для решения задачи *PnP*, *projectPoints* для проектирования

точек модели, реализацию оператора *Sobel* [12] для подсчета градиента изображения и другие алгоритмы. Таким образом, библиотека *OpenCV* была выбрана для работы с вышеперечисленными алгоритмами.

3. Разработка алгоритма

Алгоритм отслеживания для видеопоследовательности сводится к отслеживанию для одного кадра. То есть необходимо, имея изображения текущего и предыдущего кадров и положение экрана для предыдущего кадра, оценить положение на текущем кадре.

Экраны устройств имеют прямоугольную рамку и обладают свойствами выраженности сторон, изменчивостью изображения, бликами и отражениями. Учитывая эти свойства, было решено использовать алгоритм преобразования Хафа для выделения прямых линий на изображении и составлять четырехугольники из найденных линий для поиска экрана.

Сначала выполняется преобразование Хафа. Линии, полученные после преобразования Хафа, фильтруются и разделяются на внутренние и внешние. По линиям составляются внутренние и внешние четырехугольники и также фильтруются. После этого с помощью пар из внутренних и внешних четырехугольников составляются и фильтруются рамки. Затем выделяется лучший четырехугольник и по нему строится результирующее трёхмерное положение.

3.1. Выделение и фильтрация линий

Линии выделяются по изображению с помощью преобразования Хафа, после чего разделяются на внутренние и внешние и фильтруются.

В задаче отслеживания часто используется предположение о том, что перемещение экрана за один кадр мало́. Поэтому предполагается, что если на текущем кадре обрезать изображение так, что его центр будет совпадать с центром экрана на предыдущем кадре, и экран с предыдущего кадра, а также отступ будут находится в обрезанном изображении, то на текущем кадре будет присутствовать экран.

Предполагается, что сторона экрана на изображении имеет большой градиент, так как экраны устройств обладают свойством выраженности сторон. Поэтому после обрезания к изображению применяется оператор *Кэнни*, который выделяет контуры на изображении, которые соот-

ветствуют большому градиенту. Результатом данного оператора также является изображение. Этот шаг необходим для преобразования Хафа.

Далее используется непосредственно алгоритм преобразования Хафа для поиска всех прямых линий на изображении. Затем для каждой стороны экрана, внутренней и внешней, линии рассматриваются и проходят процедуру фильтрации отдельно.

В первую очередь линия отфильтровывается по относительным критериям, определяемым соответствующей линией одной из четырех внутренних сторон экрана с предыдущего кадра. Если расстояние между центрами линий велико, линия отфильтровывается. Также отфильтровывается, если велик угол между линиями. При этом учитывается то, что внешние линии располагаются дальше от внутренних сторон с предыдущего кадра, чем внутренние.

Далее для линии подсчитывается суммарный ориентированный градиент следующим образом. В каждом пикселе изображения, через который проходит линия, подсчитывается градиент с помощью оператора Собеля по двум направлениям, значения в которых составляют вектор. Этот вектор проецируется на перпендикуляр к линии. Таким образом, получается величина, которая говорит о том, насколько велик градиент в направлении перпендикуляра к линии. Если сумма этих величин по всем пикселям линии, нормированная на количество пикселей, мала, линия отфильтровывается. Данный подход учитывает свойство выраженности сторон экрана. Важно отметить, что длина линии, выделенной преобразованием Хафа, может быть больше длины стороны экрана и таким образом могут учитываться неподходящие пиксели.

Так выделяются и отфильтровываются линии экрана, которые затем могут быть использованы для составления четырехугольников.

3.2. Составление и фильтрация четырехугольников

Четырехугольники составляются из линий, разделяясь на внешние и внутренние, и фильтруются.

При имеющихся наборах внутренних линий для каждой из четырех

сторон экрана составляются всевозможные внутренние четырехугольники, каждый из которых является совокупностью четырех линий из различных наборов. При этом сторонами внутреннего четырехугольника являются отрезки, полученные после пересечения исходных линий. Аналогичным образом составляются внешние четырехугольники по внешним линиям.

Как внешние, так и внутренние четырехугольники оцениваются по сравнению с положением четырехугольника с предыдущего кадра. Для этого каждая из сторон сравнивается по отдельности со сторонами предыдущего кадра по дистанции и углу между сторонами аналогичным для линий образом и затем результат усредняется и четырехугольники отсеиваются по порогу. Также внутренние и внешние четырехугольники отсеиваются по ошибке репроекции для оценки соответствия модели экрана.

Внутренние четырехугольники проходят и другие этапы фильтрации. На первом этапе сравнивается текущий четырехугольник с найденным на предыдущем кадре четырехугольником. Если разница в площади для этих четырехугольников велика по сравнению с максимальной из двух площадью, то текущий четырехугольник подлежит фильтрации. Аналогичным образом рассматривается соотношение сторон четырехугольников, при этом шириной служит верхняя и нижняя стороны экрана, высотой служит левая и правая стороны.

Следующим этапом является фильтрация по градиенту перпендикулярно сторонам аналогично фильтрации для линий. Результаты для различных сторон усредняются и отсеиваются по порогу. Важно отметить, что в случае четырехугольников стороны не будут намного больше сторон экрана и градиент сторон не будет учитывать неподходящие пиксели.

Далее рассматриваются пиксели, которые находятся вблизи четырехугольника, но находятся вне его. Оценивается дисперсия их цвета. Если она мала, это означает, что вокруг данного четырехугольника цвет однотонный, что выражает свойство монотонности рамки экрана. На рис. 1 показан пример четырехугольника, вокруг которого цвет не

является однотонным.

Рис. 1: Пример четырехугольника с немонотонным внешним окружением

Таким образом составляются и отфильтровываются четырехугольники, которые затем могут быть использованы для составления рамок экранов.

3.3. Составление и фильтрация рамок

Зачастую получившиеся прямоугольники имеют внешнюю сторону рамки экрана как одну из сторон. Для фильтрации таких прямоугольников необходимо учесть наличие рамки экрана. Рамки составляются из четырехугольников и подлежат фильтрации.

При имеющихся наборах внутренних и внешних четырехугольников составляются всевозможные рамки, каждая из которых является совокупностью внутреннего и внешнего четырехугольника.

В первую очередь отсеивается пара из внутреннего и внешнего четырехугольника, где внутренний не находится внутри внешнего. На рис. 2 приведен пример рамки, не удовлетворяющей этому условию. Далее оценивается расстояние между соответствующими сторонами внутреннего и внешнего четырехугольника и усредняется по четырем сторонам. Четырехугольники фильтруются по порогу данного значения. Аналогично рассматриваются углы наклона между сторонами внутреннего и внешнего четырехугольника.

Рис. 2: Пример рамки, в которой внутренний четырехугольник не расположен во внешнем

Таким образом составляются и отсеиваются рамки, которые затем могут быть использованы для выделения лучшего четырехугольника.

3.4. Выделение лучшего четырехугольника

После фильтрации рамок рассматриваются все внутренние четырехугольники, из которых необходимо выбрать результат для данного кадра. Это может быть сделано с помощью метрики ошибки репроекции, так как она показывает соответствие точек модели экрана. Внутренний четырехугольник, у которого ошибка репроекции является наименьшей, выбирается для дальнейшей обработки.

Далее с помощью данного четырехугольника решается задача PnP, то есть находится трёхмерное положение, которое необходимо для наложения эффекта. Это положение и будет результатом для данного кадра. Из результатов работы для каждого кадра формируется итоговый результат алгоритма, то есть последовательность трёхмерных положений.

4. Реализация алгоритма

Для реализации алгоритма был выбран язык программирования *Python* и библиотека алгоритмов компьютерного зрения *OpenCV*. Алгоритм реализован в виде библиотеки, экспортирующей функцию. Исходный код данной библиотеки открыт. Библиотека может использоваться как самостоятельно, так и как часть графических редакторов и других программных продуктов. Диаграмма активности на рис. 3 показывает этапы работы алгоритма для одного кадра. На рис. 4 представлена диаграмма активности для фильтрации линий.

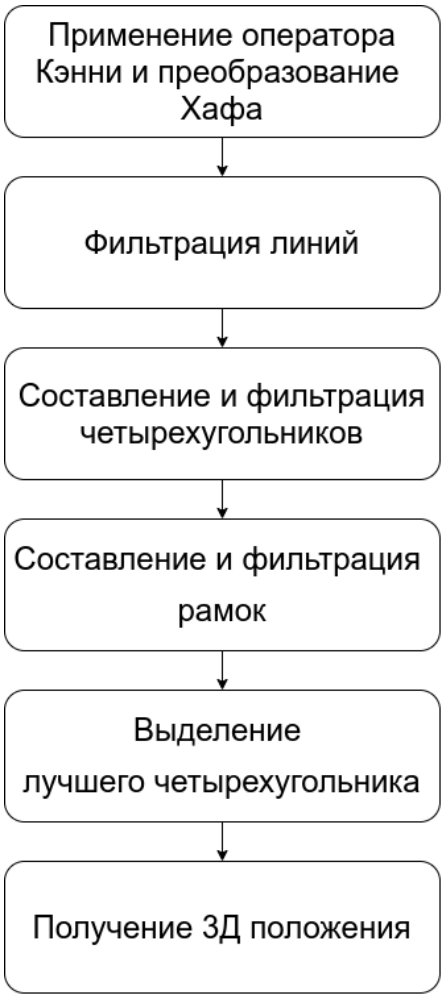

Рис. 3: Диаграмма активности алгоритма, определяющего трёхмерное положение для одного кадра

Аргументом экспортируемой функции является директория, в которой расположены файлы с данными для обработки. Для корректной работы алгоритма в этой директории должны располагаться: видеопо-

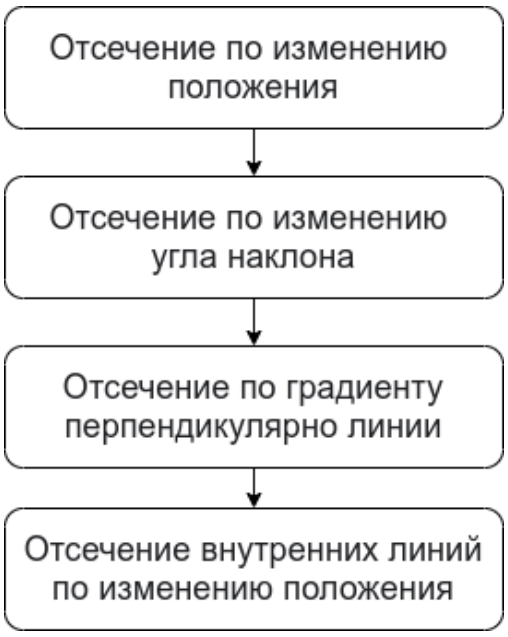

Рис. 4: Диаграмма активности, отображающая фильтрацию линий

следовательность, файл в формате *OBJ* для описания трёхмерной модели экрана, описание истинного положения в формате *YAML*, а также файл с названием *test_description.yml*. Файл *test_description.yml* включает в себя указание файлов с видеопоследовательностью, описанием модели и истинным положением, а также матрицу внутренних параметров камеры.

После работы алгоритма записывается файл результата в формате *YAML*. Результат включает в себя найденное трёхмерное положение для всех кадров видео, либо информацию о том, что алгоритм отслеживания не смог выявить трёхмерное положение объекта на кадрах.

Дополнительным аргументом функции является выбор алгоритма. Кроме реализованного алгоритма в библиотеке можно воспользоваться алгоритмом, основанным на *SIFT* дескрипторах, алгоритмом Лукаса-Канаде с аффинным преобразованием и алгоритмом отслеживания контуров *RAPiD*. Также могут быть выбраны этапы программы, среди которых сам алгоритм отслеживания, оценка метрики и запись видео с отображением найденного четырехугольника.

5. Сравнение с аналогами

Сравнение было произведено на тестовых данных, предоставленных компанией *KeenTools*. Тестовые данные включают набор из 20 видео с экраном телевизора. Данные содержат видео с включенным и выключенным экраном, бликами и отражениями, с различными изменениями ракурса. Для всех тестов матрица внутренних параметров камеры следующая:

$$K = \begin{pmatrix} 896.916 & 0.0 & 640.0 \\ 0.0 & 896.916 & 360.0 \\ 0.0 & 0.0 & 1.0 \end{pmatrix}$$

Тестирование алгоритма происходило на тестовом стенде с процессором Intel Core i7-8550U с номинальной тактовой частотой 1800MHz, объемом оперативной памяти 16GB и операционной системой Ubuntu 18.04.3.

Для метрики *tracking success rate* кадр считался отслеженным успешно, если ошибка вращения составляла не более 0.02 от полного круга, а ошибка переноса составляла не более 0.02 диагоналей экрана, так как при таких константах и матрице внутренних параметров камеры для тестовых данных дефекты не заметны визуально, что важно для удобства работы в графическом редакторе. Также для комфортной работы в графическом редакторе скорость работы алгоритма не должна быть больше пяти кадров в секунду.

Запуск тестирования проходил 30 раз. Для каждого запуска усредняется время обработки кадра по всем тестовым данным. Далее по 30 запускам вычисляются среднее и стандартное отклонение времени обработки кадра. Результат работы алгоритмов детерминирован, поэтому метрика качества вычислялась напрямую без учета среднего и стандартного отклонения. Результаты сравнения представлены в таблице 1.

По результатам сравнения видно, что реализованный алгоритм на основе преобразования Хафа превосходит алгоритм, основанный на сопоставлении особых точек *SIFT* в несколько раз как по скорости, так и

Алгоритм	**Время обработки кадра, сек.**		**Tracking success rate, %**
	Среднее	**Стандартное отклонение**	
Реализованный алгоритм	1.49	0.09	36.8
Сопоставление особых точек *SIFT*	42.8	0.14	4.4
RAPiD для отслеживания контуров	0.22	0.02	1.1
Метод Лукаса-Канаде с аффинным преобразованием	10.18	0.03	79.1
Инструмент *GeoTracker*	0.57	0.01	82.2

Таблица 1: Результаты сравнения алгоритмов

по качеству. Также качество алгоритма лучше, чем у алгоритма отслеживания контуров *RAPiD*, но при этом *RAPiD* выигрывает по скорости. Напротив, разработанный алгоритм является более быстрым, чем алгоритм Лукаса-Канаде с аффинным преобразованием, но уступает ему по качеству. Инструмент *GeoTracker* превосходит по качеству участвовавшие в сравнении алгоритмы, в том числе и разработанный алгоритм.

Заключение

В рамках данной дипломной были получены следующие результаты:

- проведен обзор предметной области и рассмотрены алгоритмы и инструменты для отслеживания объектов, а также метрики и технологии;
- разработан алгоритм для отслеживания экранов устройств, основанный на преобразовании Хафа и фильтрации линий и четырехугольников;
- алгоритм был реализован в виде библиотеки, которая экспортирует функцию, и может быть использован в графическом редакторе;
- произведено сравнение разработанного алгоритма с аналогичными алгоритмами и инструментами:
  - сопоставление особых точек *SIFT*;
  - RAPiD для отслеживания контуров объекта;
  - Метод Лукаса-Канаде с аффинным преобразованием;
  - Инструмент *GeoTracker*.

Таким образом, в результате работы был разработан алгоритм для отслеживания экранов устройств по видеопоследовательности, который основан на преобразовании Хафа и фильтрации линий и четырехугольников. Код проекта доступен в репозитории *GitHub* по ссылке[2].

[2]Исходный код разработанного алгоритма, URL: https://github.com/PolyProgrammist/screen_tracking (дата обращения: 30.05.2020)

Список литературы

[1] Canny J. A Computational Approach to Edge Detection // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 1986. — Nov. — Vol. PAMI-8, no. 6. — P. 679–698.

[2] GeoTracker homepage. — Accessed: 30.05.2020. URL: `https://keentools.io/geotracker`.

[3] Harris Chris. Tracking with Rigid Models // Active Vision. — Cambridge, MA, USA : MIT Press, 1993. — P. 59–73. — ISBN: 0262023512.

[4] Hartley Richard, Zisserman Andrew. Multiple View Geometry in Computer Vision. — 2 edition. — New York, NY, USA : Cambridge University Press, 2003. — ISBN: 0521540518.

[5] Jupyter homepage. — Accessed: 30.05.2020. URL: `https://jupyter.org/`.

[6] Lepetit Vincent, Fua Pascal. Monocular Model-Based 3D Tracking of Rigid Objects: A Survey // Foundations and Trends in Computer Graphics and Vision. — 2005. — 01. — Vol. 1.

[7] Lowe David G. Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image. — 2000.

[8] Lucas Bruce D., Kanade Takeo. An Iterative Image Registration Technique with an Application to Stereo Vision. — 1981.

[9] Oliphant T. E. Python for Scientific Computing // Computing in Science Engineering. — 2007. — Vol. 9, no. 3. — P. 10–20.

[10] OpenCV homepage. — Accessed: 30.05.2020. URL: `https://opencv.org/`.

[11] Real-Time Model-Based Rigid Object Pose Estimation and Tracking Combining Dense and Sparse Visual Cues / K. Pauwels, L. Rubio, J. Díaz, E. Ros // 2013 IEEE Conference on Computer Vision and Pattern Recognition. — 2013. — P. 2347–2354.

[12] Sobel Irwin. An Isotropic 3x3 Image Gradient Operator // Presentation at Stanford A.I. Project 1968. — 2014. — 02.

[13] A Survey on Hough Transform, Theory, Techniques and Applications / Allam Shehata, Sherien Mohammad, Mohamed Abdallah, Mohammad Ragab. — 2015. — 02.

[14] Wang Z., Shang Y., Zhang H. A Survey on Approaches of Monocular CAD Model-Based 3D Objects Pose Estimation and Tracking // 2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC). — 2018. — P. 1–6.