МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра компьютерных технологий и систем

СИСТЕМА ДОПОЛНЕННОЙ РЕАЛЬНОСТИ ДЛЯ АНАЛИЗА ПОСЛЕДОВАТЕЛЬНОСТИ СНИМКОВ

Курсовой проект

Захарчени Вадима Валерьевича

студента 4 курса 4 группы,
специальность
«информатика»

Научные руководители:
доктор технических наук
Недзьведь Александр Михайлович

ассистент
Лагуто Анна Андреевна

Минск, 2017

АННОТАЦИЯ

Захарченя В.В. Исследование возможностей дополненной реальности для анализа последовательности снимков: Курсовой проект / Минск: БГУ, 2017. – 36 с.

Исследуются варианты реализации дополненной реальности, способы анализа изображений, методы калибровки камер. Рассматриваются особенности реализации программного средства, а также технологии для реализации.

АНАТАЦЫЯ

Захарчэня В.В. Даследаванне магчымасцяў дапоўненай рэальнасці для аналізу паслядоўнасці здымкаў: Курсавы праэкт / Мінск: БДУ, 2017. - 36 с.

Даследуюцца варыянты рэалізацыі дапоўненай рэальнасці, спосабы аналізу малюнкаў, метады каліброўкі камер. Разглядаюцца асаблівасці рэалізацыі праграмнай сродкі, а таксама тэхналогіі для рэалізацыі.

ANNOTATION

Zakharchenya V.V. Exploring the possibilities of augmented reality for analyzing the sequence of images: Course project / Minsk: BSU, 2017. - 36 p.

The variants of realization of augmented reality, methods of image analysis, methods of camera calibration are explored. Features of the implementation of the software, as well as technologies for implementation are considered.

РЕФЕРАТ

Курсовой проект, 36 с., 13 рис., 7 источников.

Ключевые слова: ДОПОЛНЕННАЯ РЕАЛЬНОСТЬ, ВИРТУАЛЬНАЯ РЕАЛЬНОСТЬ, АНАЛИЗ ИЗОБРАЖЕНИЙ, АФФИННЫЕ ПРЕОБРАЗОВАНИЯ, HOLOLENS

Объект исследования **–** системы анализа последовательности снимков(видеоряда). Предмет исследования – программное средство для анализа видеопотока.

Цель работы – изучение систем дополненной реальности и алгоритмов в рамках анализа изображений. Изучение технологий для реализации соответствующего программного средства.

Методы исследования – анализ существующих систем, разработка алгоритма, реализация программного средства.

Результатом является программная реализация для анализа изображений из видеопотока.

Областью применения является различные сферы деятельности человека.

РЭФЕРАТ

Курсавой праект, 36 с., 13 мал., 7 крыніц.

Ключавыя словы: дапоўненая рэальнасць, віртуальная рэальнасць, аналізу малюнкаў, АФФИННЫЕ пераўтварэнняў, HOLOLENS

Аб'ект даследавання - сістэмы аналізу паслядоўнасці здымкаў (відэашэрагу). Прадмет даследавання - праграмнае сродак для аналізу відэаструменю.

Мэта работы - вывучэнне сістэм дапоўненай рэальнасці і алгарытмаў ў рамках аналізу малюнкаў. Вывучэнне тэхналогій для рэалізацыі адпаведнага праграмнага сродкі.

Метады даследавання - аналіз існуючых сістэм, распрацоўка алгарытму, рэалізацыя праграмнага сродкі.

Вынікам з'яўляецца праграмная рэалізацыя для аналізу малюнкаў з відэаструменю.

Вобласцю ўжывання з'яўляецца розныя сферы дзейнасці чалавека.

ABSTRACT

Course project, 36 pp., 13 pic., 7 sources.

Keywords: ADDITIONAL REALITY, VIRTUAL REALITY, ANALYSIS OF IMAGES, AFFINE TRANSFORMATIONS, HOLOLENS

The object of the study is the sequence analysis system of images (video sequence). The subject of the study is a software tool for analyzing the video stream.

The aim of the work is to study the systems of augmented reality and algorithms in the framework of image analysis. The study of technologies for the implementation of the relevant software.

Methods of research - analysis of existing systems, development of an algorithm, implementation of a software tool.

The result is a software implementation for analyzing images from a video stream.

The sphere of application is various spheres of human activity.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время мы можем наблюдать как стремительное развитие технологий в различных областях науки, так и объём информации, который необходимо обрабатывать.

С давних времён для воздействия на окружающие предметы, управления ими, человек использовал по большей части свои руки. После появления компьютера – одного из величайших изобретений человечества, принцип управления кардинально не изменился.

Но с ростом производительности компьютеров появляется необходимость увеличения потоков информации между человеком и компьютером. Как следствие, возникает потребность в новых интерфейсах взаимодействия с компьютером, и, соответственно, новых методах анализа входного потока информации и вывода результатов этого анализа в приемлемой и удобочитаемой форме.

О технологиях виртуальной и дополненной реальности в последние годы было много противоречивых прогнозов – с момента самой ранней эры «четвёртой волны» развития подобных технологий, начавшейся в ещё несколько лет назад.

Именно в наше время VR-технологии может ждать стремительный рост: производители первых VR-устройств, VR-контента впервые будут активно собирать обратную связь от «раннего большинства», обычных пользователей.

К тому же, к гонке за лидерство на рынке VR подключаются все новые именитые компании. Это вселяет уверенность, что нас ждёт бум пользовательских устройств.

Организации уже могут использовать VR-технологии для симуляции, тренировок и обучения, демотуров, коммуникации, прототипирования и других применений. Возможно, именно активность компаний по развитию подобных проектов будет способствовать росту популярности VR и AR.

Дополненная реальность описывалась в научно-фантастических произведениях на протяжении последних десятилетий. Даже сейчас, когда в кармане почти каждого человека лежит смартфон — миниатюрный компьютер, вычислительные мощности которого превосходят даже самые смелые предсказания ученых начала девяностых, перспектива таких технологий кажется очень далекой. Но, если смотреть правде в глаза, дополненная реальность все сильнее и сильнее интегрируется в нашу жизнь, например, через социальные сети. Так что же такое Augmented Reality и чего ждать от нее в ближайшие годы?

Дополненная реальность показывает информацию, относящуюся к физическим объектам вокруг человека. Простой пример: в машине сломались тормоза. Дополненная реальность позволила бы провести диагностику

автомобиля с помощью одного лишь смартфона. Включив его камеру, можно было бы посмотреть, в каком именно месте произошла поломка, что конкретно сломалось, а также узнать, что следует сделать, чтобы вернуть машину в рабочее состояние.

Целью данной работы является исследование возможностей систем дополненной реальности для анализа и обработки входного потока информации в виде видеоряда, последовательности изображений, а также способов вывода результатов работы.

1 Существующие интерфейсы и системы, а также методы анализа изображений.

1.1 Виртуальная реальность

Виртуальная реальность — созданный техническими средствами мир (объекты и субъекты), передаваемый человеку через его ощущения: зрение, слух, обоняние, осязание и другие. Виртуальная реальность имитирует как воздействие, так и реакции на воздействие. Для создания убедительного комплекса ощущений реальности компьютерный синтез свойств и реакций виртуальной реальности производится в реальном времени.

Объекты виртуальной реальности обычно ведут себя близко к поведению аналогичных объектов материальной реальности. Пользователь может воздействовать на эти объекты в согласии с реальными законами физики (гравитация, свойства воды, столкновение с предметами, отражение и т. п.). Однако часто в развлекательных целях пользователям виртуальных миров позволяется больше, чем возможно в реальной жизни (например: летать, создавать любые предметы и т. п.).

Не следует путать виртуальную реальность с дополненной. Их коренное различие в том, что виртуальная конструирует новый искусственный мир, а дополненная реальность лишь вносит отдельные искусственные элементы в восприятие мира реального.

Системами «виртуальной реальности» называются устройства, которые более полно по сравнению с обычными компьютерными системами имитируют взаимодействие с виртуальной средой, путём воздействия на все пять имеющихся у человека органов чувств.

В настоящее время существует несколько основных типов систем, обеспечивающих формирование и вывод изображения в системах виртуальной реальности:

Шлем / очки виртуальной реальности (HMD - display).

Современные шлемы виртуальной реальности представляют собой скорее очки, нежели шлем, и содержат один или несколько дисплеев, на которые выводятся изображения для левого и правого глаза, систему линз для корректировки геометрии изображения, а также систему трекинга, отслеживающую ориентацию устройства в пространстве. Как правило, системы трекинга для шлемов виртуальной реальности разрабатываются на основе гироскопов, акселерометров и магнитометров. Для систем этого типа важен широкий угол обзора, точность работы системы трекинга при отслеживании

наклонов и поворотов головы пользователя, а также минимальная задержка между детектированием изменения положения головы в пространстве и выводом на дисплеи соответствующего изображения (Рисунок 1.1).

Рисунок 1.1 – Пример игры, основанной на технологии виртуальной реальности

MotionParallax3D дисплеи

К устройствам этого типа относится множество различных устройств: от некоторых смартфонов до комнат виртуальной реальности (CAVE). Системы данного типа формируют у пользователя иллюзию объёмного объекта за счёт вывода на один или несколько дисплеев специально сформированных проекций виртуальных объектов, сгенерированных исходя из информации о положении глаз пользователя. При изменении положения глаз пользователя относительно дисплеев, изображение на них соответствующим образом меняется. Все системы

данного типа задействуют зрительный механизм восприятия объёмного изображения параллакс движения (Motion Parallax). Также, в большинстве своём, они обеспечивают вывод стереоизображения с помощью стереодисплеев, задействуя стереоскопическое зрение. Системы трекинга для MotionParallax3D дисплеев отслеживают координаты глаз пользователей в пространстве. Для этого используются различные технологии: оптическая (определение координат глаз пользователя на изображении с камеры, отслеживание активных или пассивных маркеров), существенно реже - ультразвуковая. Зачастую системы трекинга могут включать в себя дополнительные устройства: гироскопы, акселерометры и магнитометры. Для систем данного типа важна точность отслеживания положения пользователя в пространстве, а также минимальная задержка между детектированием изменения положения головы в пространстве и выводом на дисплеи соответствующего изображения. Системы данного класса могут выполняться в различных форм-факторах: от виртуальных комнат с полным погружением до экранов виртуальной реальности размером от трёх дюймов.

Виртуальный ретинальный монитор

Устройства данного типа формируют изображение непосредственно на сетчатке глаза. В результате пользователь видит изображение, «висящее» в воздухе перед ним. Устройства данного типа ближе к системам дополненной реальности, поскольку изображения виртуальных объектов, которые видит пользователь, накладываются на изображения объектов реального мира. Тем не менее, при определённых условиях (тёмная комната, достаточно широкое покрытие сетчатки изображением, а также в сочетании с системой трекинга), устройства данного типа могут использоваться для погружения пользователя в виртуальную реальность.

Также существуют различные гибридные варианты: например, система CastAR, в которой получение корректной проекции изображения на плоскости достигается за счёт расположения проекторов непосредственно на очках, а стереоскопическое разделение - за счёт использования световозвращающего покрытия поверхности, на которую ведётся проецирование. Но пока такие устройства широко не распространены и существуют лишь в виде прототипов.

На данный момент самыми совершенными системами виртуальной реальности являются проекционные системы, выполненные в компоновке комнаты виртуальной реальности (CAVE). Такая система представляет собой комнату, на все стены которой проецируется 3D-стереоизображение. Положение пользователя, повороты его головы отслеживаются трекинговыми системами,

что позволяет добиться максимального эффекта погружения. Данные системы активно используются в маркетинговых, военных, научных и других целях.

1.2 Дополненная реальность

Дополненная реальность – результат введения в поле восприятия любых сенсорных данных с целью дополнения сведений об окружении и улучшения восприятия информации (Рисунок 1.2).

Дополненная реальность – воспринимаемая смешанная реальность (англ. mixed reality), создаваемая с использованием «дополненных» с помощью компьютера элементов воспринимаемой реальности (когда реальные объекты монтируются в поле восприятия).

Среди наиболее распространенных примеров дополнения воспринимаемой реальности – параллельная лицевой цветная линия, показывающая нахождение ближайшего полевого игрока к воротам при телевизионном показе футбольных матчей, стрелки с указанием расстояния от места штрафного удара до ворот, «нарисованная» траектория полета шайбы во время хоккейного матча, смешение реальных и вымышленных объектов в кинофильмах и компьютерных и т. п.

Существует несколько определений дополненной реальности: исследователь Рональд Азума (англ. Ronald Azuma) в 1997 году определил её как систему, которая:

1. совмещает виртуальное и реальное;
2. взаимодействует в реальном времени;
3. работает в 3D.

В 1994 году Пол Милгром и Фумио Кисино описали континуум «виртуальность-реальность» – пространство между реальностью и виртуальностью, между которыми расположены дополненная реальность (ближе к реальности) и дополненная виртуальность (ближе к виртуальности). Дополненная реальность — результат добавления к воспринимаемым как элементы реального мира мнимых объектов (обычно в качестве вспомогательной информации).

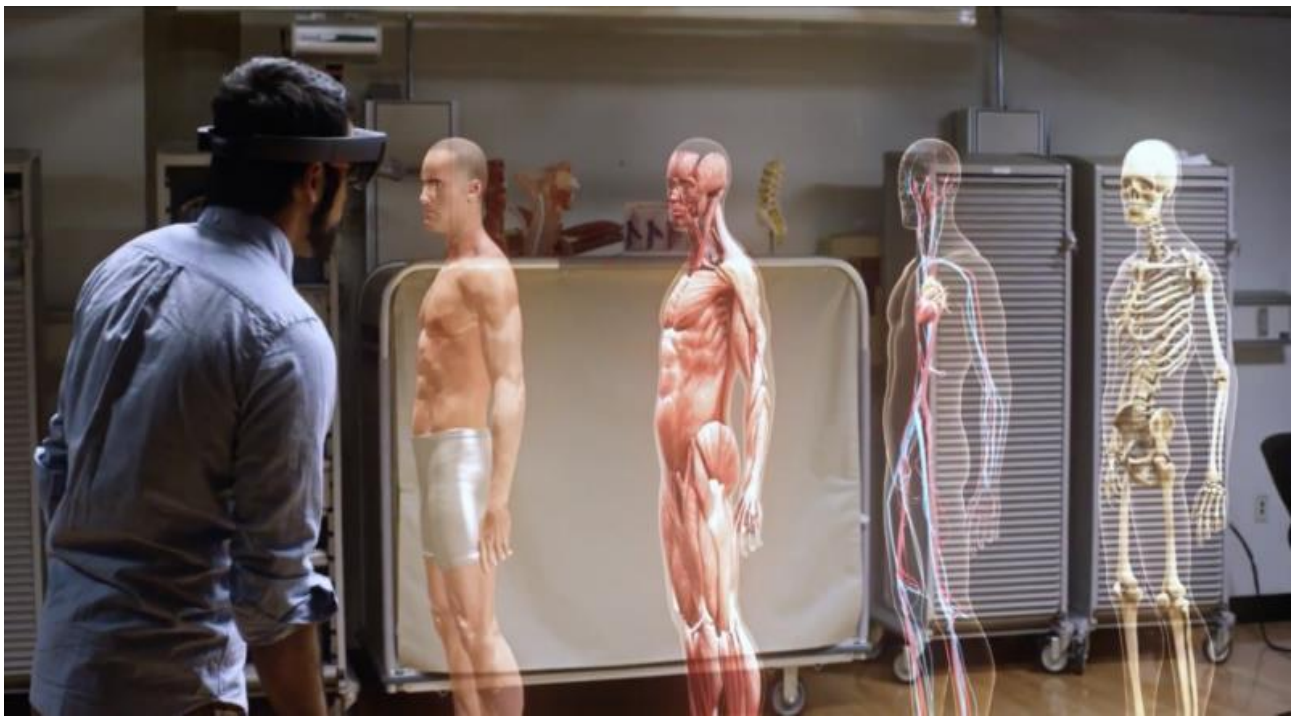


Рисунок 1.2 – Пример реализации дополненной реальности

Иногда в качестве синонимов используют термины «расширенная реальность», «улучшенная реальность», «обогащённая реальность», «увеличенная реальность». Правда, такое использование названных терминов в общем случае неправильно, — термины «расширенная реальность», «увеличенная реальность», «обогащённая реальность» применимы лишь для обозначения определённых форм и и аспектов практического применения дополненной реальности, тогда как применимость термина «улучшенная реальность» вовсе сомнительна.

1.3 Аффинное преобразование

Аффинное преобразование – отображение плоскости или пространства в себя, при котором параллельные прямые переходят в параллельные прямые, пересекающиеся в пересекающиеся, скрещивающиеся в скрещивающиеся.

Аффинное преобразование евклидова пространства – взаимно однозначное точечное отображение плоскости или пространства на себя, при котором трем точкам, лежащим на одной прямой, соответствуют три точки, также лежащие на одной прямой. Таким образом, при аффинном преобразовании прямые переходят в прямые. Аффинное преобразование плоскости переводит пересекающиеся прямые в пересекающиеся, параллельные – в параллельные. При аффинном преобразовании пространства каждая плоскость аффинно

отображается на некоторую плоскость; при этом пересекающиеся плоскости переходят в пересекающиеся, параллельные – в параллельные. Кроме того, сохраняется взаимное расположение двух прямых: пересекающиеся прямые переходят в пересекающиеся, параллельные - в параллельные, скрещивающиеся – в скрещивающиеся.

Примерами аффинных преобразований являются:

- преобразование движения
- преобразование растяжения
- преобразования подобия

Движение — преобразование метрического пространства, сохраняющее расстояние между соответствующими точками. Иначе говоря, движение — это изометрия пространства в себя.

Несмотря на то, что движение определяется на всех метрических пространствах, этот термин более распространён в евклидовой геометрии и смежных областях. В метрической геометрии (в частности, в римановой геометрии) чаще говорят: изометрия пространства в себя. В общем случае метрического пространства (например, для неплоского риманова многообразия) движения могут существовать далеко не всегда.

В евклидовом (или псевдоевклидовом) пространстве движение автоматически сохраняет также углы, так что сохраняются все скалярные произведения (Рисунок 1.3).

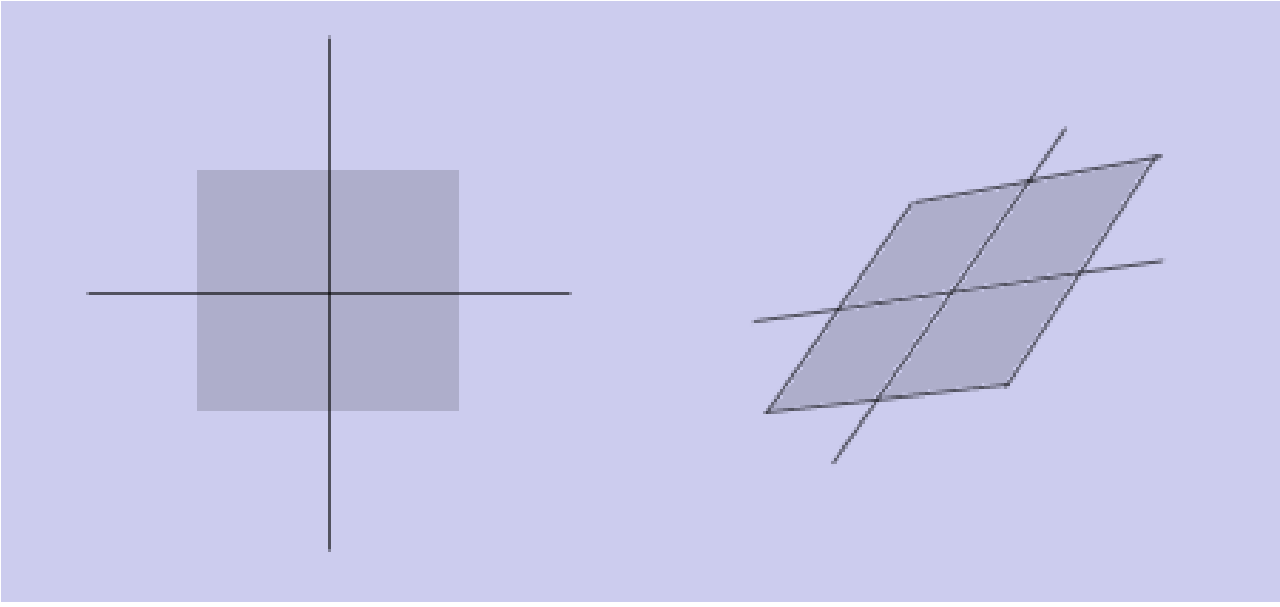


Рисунок 1.3 – Скос – объединение нескольких аффинных преобразований

Растяжение плоскости относительно оси l с коэффициентом k — преобразование плоскости, при котором каждая точка M переходит в такую точку M', что расстояние от прямой l, до M' в k раз больше, чем до точки M, и проекции точек M и M' на прямую l совпадают.

Подобие — преобразование евклидова пространства, при котором для любых двух точек A, B и их образов A', B' имеет место соотношение $|A'B'| = k|AB|$ где $k \neq 0$ — коэффициент подобия.

1.4 Калибровка камер

Современные видеокамеры, имеющие многолинзовые объективы (Рисунок 1.4), вносят геометрические искажения в получаемое изображение. Для задач компьютерного зрения наличие искажений нежелательно или недопустимо, поэтому требуется их устранение программным способом. Это достигается путем использования математической модели оптической системы видеокамеры и определенных экспериментально в процессе калибровки ее внутренних и внешних параметров.

Рисунок 1.4 – Современная камера с двумя линзами

Калибровка камеры — это задача получения внутренних и внешних параметров камеры по имеющимся фотографиям или видео, отснятыми ею. Для

калибровки видеокамер предлагается использовать реализованные в программной библиотеке OpenCV алгоритмы, учитывающие радиальное и тангенциальное искажение.

Радиальное искажение описывается следующей формулой:

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Наличие радиального искажения проявляется в виде «бочки» или эффекта «рыбий глаз».

Тангенциальное искажение происходит потому, что плоскость линзы не идеально параллельно плоскости изображения. Это может быть исправлено с помощью формул:

$$x_{corrected} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$

$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2 xy]$$

Позиция каждой точки пикселя с координатами (x, y) в исходном изображении будет скорректирована на новую точку с координатами $(x_{corrected}, y_{corrected})$.

Таким образом, у нас есть пять параметров искажения, которые в OpenCV представлены в виде одной строки матрицы с 5 столбцами:

$$Distortion_{coefficents} = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$$

Теперь для блока преобразования мы используем следующую формулу:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

При этом присутствие w объясняется использованием гомографии системы (и координат $w = Z$). Неизвестные параметры f_x и f_y (фокусные расстояния) и (c_x, c_y), которые являются оптическими центрами. Если для обеих осей общее фокусное расстояние используется с заданным a соотношением сторон (обычно $a = 1$), а затем $f_y = f_x * a$ и в верхней формуле мы будем иметь одно фокусное расстояние f. Матрица, содержащая эти четыре параметра, называется матрицей камеры.

Процесс определения этих двух матриц называется калибровкой. Расчет этих параметров осуществляется с помощью основных геометрических уравнений. Уравнения, используемые в зависимости от выбранных объектов калибрующие.

Проведенные учёными натурные эксперименты с широкоугольной камерой для задач стабилизации видео и целеуказания показали эффективность указанных подходов к устранению оптических искажений.

1.5 Устройство дополненной реальности как средство достижения поставленной задачи

Microsoft HoloLens – очки дополненной реальности, разработанные компанией Microsoft. Используют 32-разрядную операционную систему Windows Holographic (Рисунок 1.5).

HoloLens представляет собой надеваемый на голову обруч с расположенными перед глазами тонированными линзами с волнообразной призматической структурой, которые преломляют и отправляют в глаза пользователя изображения с расположенных по бокам микродисплеев. Для использования HoloLens должно быть откалибровано межзрачковое расстояние. Размер устройства может быть приспособлен под размер головы пользователя с помощью специального колёсика.

Рисунок 1.5 – Очки дополненной реальности Microsoft HoloLens

В верхней части расположены 2 пары кнопок – для управления яркостью экрана (над левым ухом) и громкостью звука (над правым). Соседние кнопки имеют разную форму (одна выпуклая, другая вогнутая) с той целью, чтобы их

можно было различать на ощупь. Динамики расположены у нижнего края устройства; они позволяют слышать как звуки виртуальной реальности, так и звуки, исходящие извне.

В отличие от большинства других устройств виртуальной, дополненной или смешанной реальности, HoloLens автономны и не требуют подключения к ПК, смартфону или игровой консоли.

HoloLens использует 64-разрядный 4-ядерный процессор Intel Atom x5-Z8100 с частотой 1,04 ГГц. В дополнение к центральному и графическому процессорам HoloLens имеет голографический процессор (англ. holographic processing unit), разработанный Microsoft специально для HoloLens. Голографический процессор, размещённый в корпусе 12 × 12 мм, использует 28 цифровых сигнальных процессоров производства Tensilica для обработки и интеграции данных, поступающих со всех сенсоров, а также пространственного сканирования (англ. spatial mapping) помещения, распознавания жестов, голоса и речи. По утверждению разработчиков, голографический процессор обрабатывает «терабайты информации». SoC и голографический процессор имеют 8 Мб встраиваемой памяти SRAM и по 1 Гб LPDDR3.

Встроенное хранилище данных имеет объём 64 Гб, из них около 10 Гб занимает операционная система, так что пользователю остаются доступны 54,09 Гб.

Объём оперативной памяти — 2 Гб.

Приложения для HoloLens не могут использовать больше 900 Мб памяти, в случае превышения этого лимита работа приложения прерывается.

HoloLens обладают 4 камерами (по 2 с каждой стороны) для сканирования окружения и ориентации в пространстве, 4 микрофонами, гиростабилизатором, датчиком глубины, 2MP видеокамерой, сенсором окружающего освещения.

Оптика HoloLens устроена очень сложно (если сравнивать, например, с устройствами виртуальной реальности), что обусловлено необходимостью не просто выводить изображение на экран, но ещё и правильно совмещать его с объектами реального мира. Жидкокристаллические проекторы с разрешением сторон 16:9, которые Microsoft назвала «световыми движками» (англ. light engines), создают изображение, которое затем проходит через визуализационную оптику (англ. imaging optics), волновод, combiner (устройство, совмещающее проекцию и изображение реального мира) и дифракционные решётки. Линзы имеют 3 слоя — для синего, зелёного и красного цветов — каждый со своими дифракционными свойствами.

Теми, кто пользовался HoloLens, отмечается, что они обладают небольшим полем зрения (при этом субъективные оценки размера поля зрения существенно разнятся), особенно по сравнению со шлемами виртуальной реальности, что часто характеризуется как их основной недостаток; при этом на Electronic

Entertainment Expo 2015 Кудо Цунода, вице-президент Microsoft, сообщил, что поле зрения вряд ли значительно изменится к выходу окончательной версии устройства.

HoloLens может генерировать бинауральный звук, что позволяет имитировать направление звука, создавая иллюзию того, что он исходит от виртуального источника.

HoloLens содержат электрический аккумулятор объёмом 16 500 мА·ч, которого должно хватать на 2-3 часа активного использования или 2 недели в спящем режиме. HoloLens можно использовать во время зарядки аккумулятора.

2 Модель системы анализа изображений

2.1 Проектирование модели системы анализа изображений

Итак, задача состоит в том, чтобы проанализировать входные данные и предоставить результаты анализа в форме, удобной для восприятия любым человеком.

Необходимо учитывать целевую аудиторию, т. е. стремиться к тому, чтобы отсутствовала необходимость в определённых знаниях и навыках для корректного считывания информации (Рисунок 2.1).

Рисунок 2.1 – Пример сложного интерфейса дополненной реальности

Проанализировав особенности и возможности виртуальной и дополненной реальностей, логичным шагом будет выбор в пользу дополненной реальности как наиболее подходящей для данной задачи.

2.2 Особенности анализа

Инструментом, с которым необходимо будет работать, является вышеописанные очки дополненной реальности HoloLens от компании Microsoft.

Данные очки дополненной реальности имеют различные датчики и сенсоры, из которых наиболее весомыми являются 4 камеры, которые предоставляют видеоряд, который как раз и является последовательностью снимков.

При анализе последовательности снимков подразумевается, что данные снимки приблизительно одних и тех же объектов, но снятых под разными углами, на разных расстояниях и с различных ракурсов.

При всей динамике содержимого изображений для корректного анализа необходимо использовать определённые маркеры, которые должны соответствовать требованиям для успешного позиционирования объекта (Рисунок 2.2).

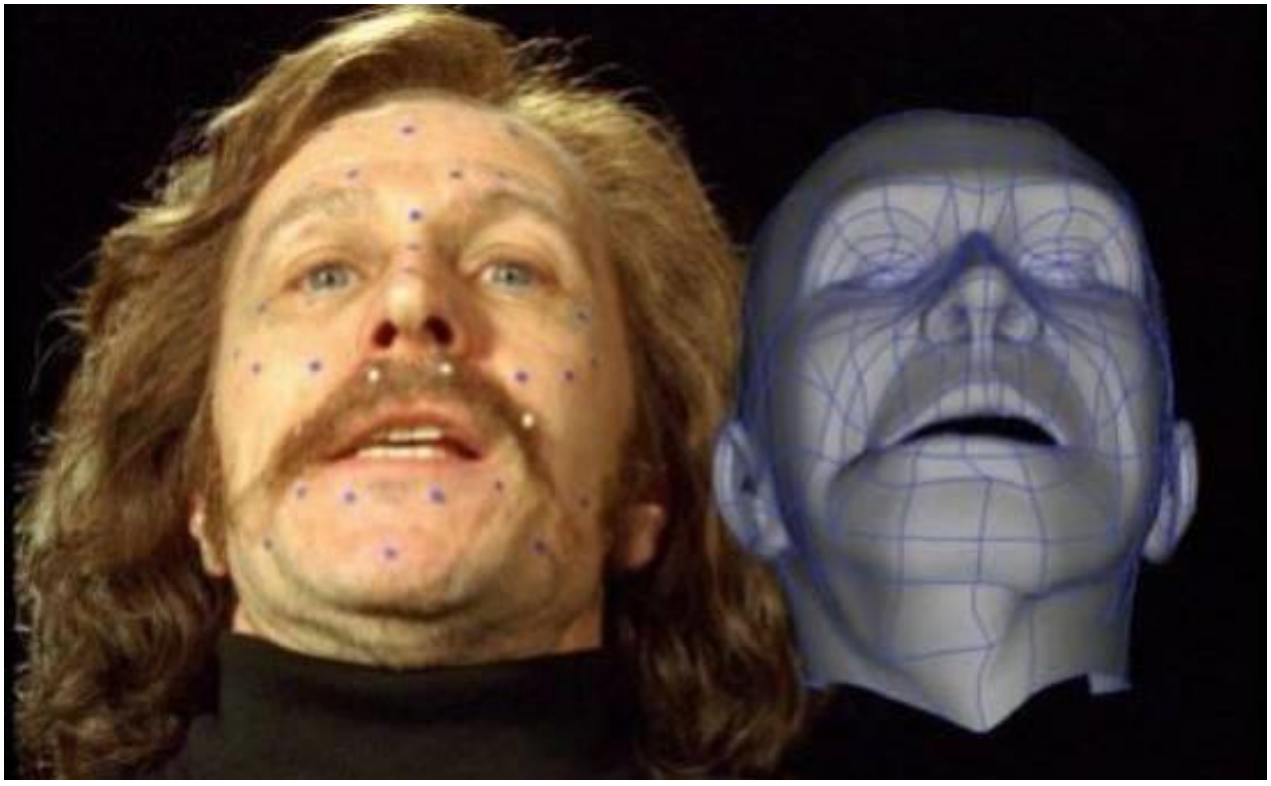


Рисунок 2.2 – Пример использования маркеров на лице для позиционирования его в пространстве

В целях упрощения реализации в данной системе будет использоваться один маркер в виде некоего прямоугольно предмета определённого цвета.

Так как ожидается, что одной из областей использования данной системы может быть медицина, то примером маркера может служить бинт.

3 Подготовка алгоритма анализа

Для начала необходимо определить, что алгоритм будет состоять из нескольких частей.

Во-первых, нужно найти вышеуказанный маркер на изображении.

Во-вторых, необходимо определить позиционирование этого маркера в пространстве, и, соответственно, при наличии заранее подготовленного описания объекта - определить позиционирование всего объекта в пространстве.

Т. е. получаем третий шаг: экстраполирование позиционирования маркера на весь объект.

3.1 Поиск определённого маркера на изображении

Для поиска маркера на изображении воспользуемся готовыми функциями из библиотеки OpenCV.

Детектирование объектов — поиск объекта по шаблону (Template matching)

Детектирование объекта по шаблону может пригодиться во множестве случаев. Самый простой пример — поиск заранее заданного объекта.

В OpenCV для этого есть замечательная функция cvMatchTemplate()

$CVAPI(void)\ cvMatchTemplate(const\ CvArr*\ image,\ const\ CvArr*\ templ, CvArr*\ result, int\ method\ )$; — сравнение шаблона и перекрывающего окна на исходном изображении

- **image** — изображение для поиска (8-битное или 32F)
- **templ** — шаблон для поиска (не должен превышать исходное изображение и иметь тот же тип)
- **result** — карта результата сравнения (32FC1) если image WxH и templ wxh, то result = (W-w+1)x(H-h+1)
- **method** — метод сравнения областей изображения

Функцию поиска можно приблизительно представить следующим образом: изображение шаблона templ последовательно накладывается на исходное изображение image и между ними вычисляется корреляция, результат которой заносится в результирующее изображение result.

Очевидно, что корреляцию между двумя изображениями можно считать разными способами.

Эти методы собраны в перечислении и просто меняют формулу расчёта корреляции:

- method= CV_TM_SQDIFF

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

- method=CV_TM_SQDIFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

- method=CV_TM_CCORR

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

- method=CV_TM_CCORR_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

- method=CV_TM_CCOEFF

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))$$

where

$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$$
$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'')$$

- method=CV_TM_CCOEFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$

3.2 Позиционирование объекта на изображении

Учитывая изображение рисунка, мы можем использовать приведенную данную информацию для расчета того, как объект находится в пространстве, например, как он вращается, как он смещён и т. д.

Для планарного объекта мы можем предположить, что Z = 0, так что теперь проблема заключается в том, как камера помещается в пространство, чтобы увидеть наш образ рисунка. Итак, если мы знаем, как объект лежит в пространстве, мы можем нарисовать в нем 2D-диаграммы для имитации 3D-эффекта. Давайте посмотрим, как это сделать.

Проблема заключается в том, что мы хотим нарисовать нашу трехмерную координатную ось (оси X, Y, Z) на первом углу шахматной доски. Ось X - синим цветом, ось Y - зеленым цветом, а ось Z - красным. Таким образом, во-вторых, ось Z должна чувствовать, что она перпендикулярна нашей шахматной плоскости.

Во-первых, давайте загрузим матрицу камеры и коэффициенты искажения из предыдущего результата калибровки.

```
import cv2
import numpy as np
import glob

# Load previously saved data
with np.load('B.npz') as X:
    mtx, dist, _, _ = [X[i] for i in ('mtx','dist','rvecs','tvecs')]
```

Теперь давайте создадим функцию, которую рисуем, которая принимает углы в шахматной доске (полученные с помощью cv2.findChessboardCorners ()) и осевые точки для рисования трехмерной оси.

```
def draw(img, corners, imgpts):
    corner = tuple(corners[0].ravel())
    img = cv2.line(img, corner, tuple(imgpts[0].ravel()), (255,0,0), 5)
    img = cv2.line(img, corner, tuple(imgpts[1].ravel()), (0,255,0), 5)
    img = cv2.line(img, corner, tuple(imgpts[2].ravel()), (0,0,255), 5)
    return img
```

Затем, как и в предыдущем случае, мы создаем критерии завершения, точки объекта (3D-точки углов в шахматной доске) и осевые точки. Точки оси - это точки в 3D-пространстве для рисования оси. Мы рисуем ось длины 3 (единицы будут в виде квадратного размера шахмат, так как мы откалиброваны на основе этого размера). Таким образом, наша ось X оттягивается от (0,0,0) до

(3,0,0), поэтому для оси Y. Для оси Z она выводится из (0,0,0) в (0,0, -3). Отрицательный означает, что он направлен к камере.

```
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
objp = np.zeros((6*7,3), np.float32)
objp[:,:2] = np.mgrid[0:7,0:6].T.reshape(-1,2)

axis = np.float32([[3,0,0], [0,3,0], [0,0,-3]]).reshape(-1,3)
```

Теперь, как обычно, мы загружаем каждое изображение.

Затем для вычисления вращения и трансляции используется функция cv2.solvePnPRansac ().

Как только мы получим эти матрицы преобразования, мы используем их для проектирования наших точек оси на плоскость изображения.

Простыми словами мы находим точки на плоскости изображения, соответствующие каждому из (3,0,0), (0,3,0), (0,0,3) в трехмерном пространстве. Как только мы получим их, мы рисуем линии от первого угла до каждого из этих пунктов, используя нашу функцию draw.

```
for fname in glob.glob('left*.jpg'):
    img = cv2.imread(fname)
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    ret, corners = cv2.findChessboardCorners(gray, (7,6),None)

    if ret == True:
        corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)

        # Find the rotation and translation vectors.
        rvecs, tvecs, inliers = cv2.solvePnPRansac(objp, corners2, mtx, dist)

        # project 3D points to image plane
        imgpts, jac = cv2.projectPoints(axis, rvecs, tvecs, mtx, dist)

        img = draw(img,corners2,imgpts)
        cv2.imshow('img',img)
        k = cv2.waitKey(0) & 0xff
        if k == 's':
            cv2.imwrite(fname[:6]+'.png', img)

cv2.destroyAllWindows()
```

Ниже приведены некоторые результаты (Рисунок 3.1). Обратите внимание, что каждая ось имеет длину 3 квадрата.

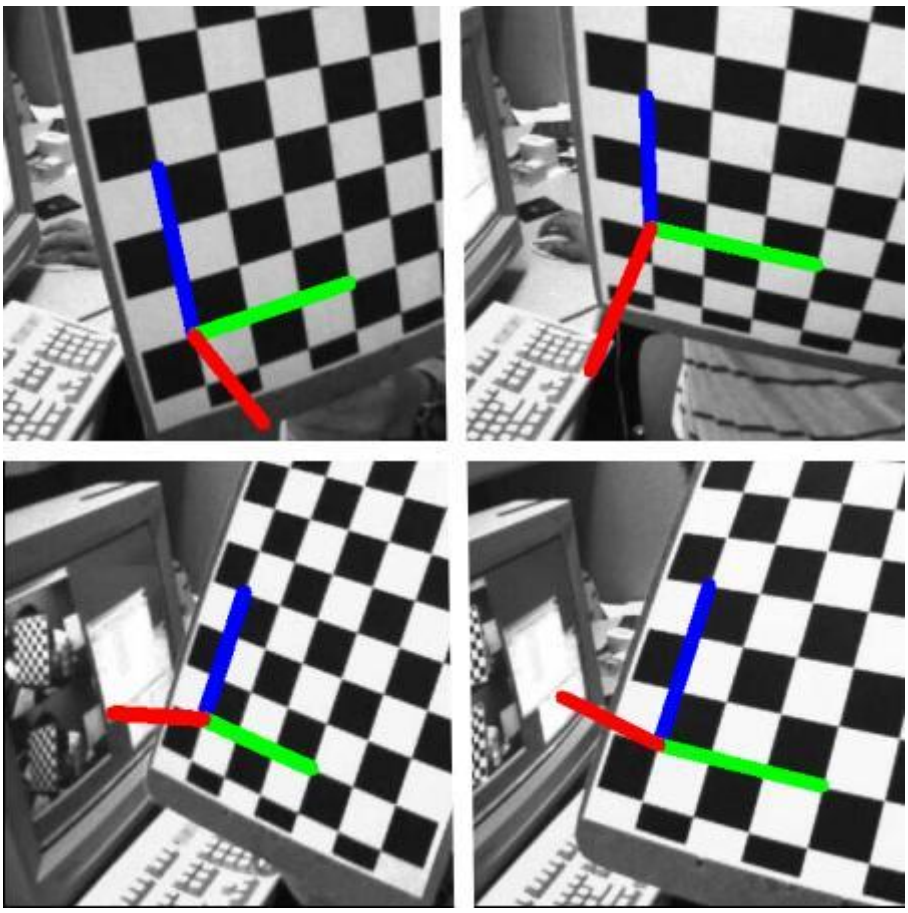


Рисунок 3.1 – Результаты позиционирования в пространстве

Измененная функция draw ():

```
def draw(img, corners, imgpts):
    imgpts = np.int32(imgpts).reshape(-1,2)

    # draw ground floor in green
    img = cv2.drawContours(img, [imgpts[:4]],-1,(0,255,0),-3)

    # draw pillars in blue color
    for i,j in zip(range(4),range(4,8)):
        img = cv2.line(img, tuple(imgpts[i]), tuple(imgpts[j]),(255),3)

    # draw top layer in red color
    img = cv2.drawContours(img, [imgpts[4:]],-1,(0,0,255),3)

    return img
```

Модифицированные осевые точки. Это 8 углов куба в трехмерном пространстве:

```
axis = np.float32([[0,0,0], [0,3,0], [3,3,0], [3,0,0],
                   [0,0,-3],[0,3,-3],[3,3,-3],[3,0,-3] ])
```

И посмотрите на результат ниже (Рисунок 3.2).

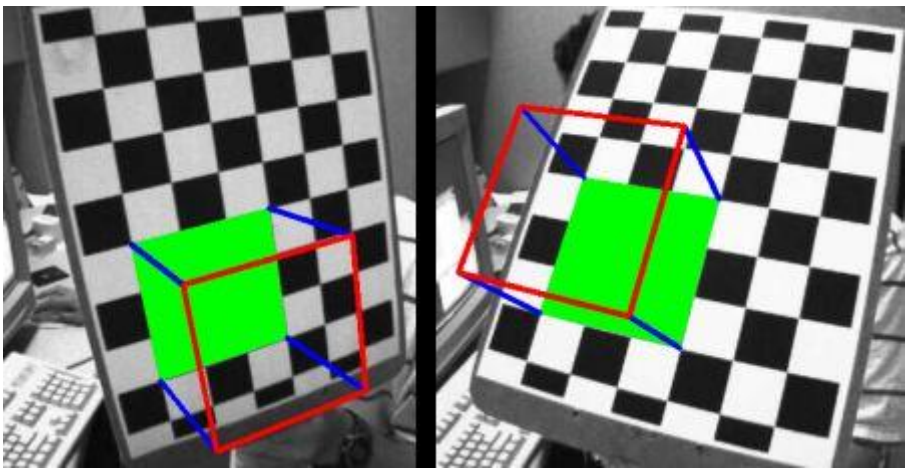


Рисунок 3.2 – Результаты позиционирования в пространстве

Можно использовать OpenGL для визуализации более сложных фигур.

4 Исследование существующих технологий для реализации программного средства

4.1 Используемый язык программирования

Реализация программы, системы сдачи выдачи домашнего задания проходит на языке программирования Java. Java — объектно-ориентированный язык программирования, разрабатываемый компанией Sun Microsystems с 1991 года и официально выпущенный 23 мая 1995 года. Изначально новый язык программирования назывался Oak (James Gosling) и разрабатывался для бытовой электроники, но впоследствии был переименован в Java и стал использоваться для написания апплетов, приложений и серверного программного обеспечения.

Программы на Java могут быть транслированы в байт-код, выполняемый на виртуальной java-машине (JVM) — программе, обрабатывающей байт-код и передающей инструкции оборудованию, как интерпретатор, но с тем отличием, что байт-код, в отличие от текста, обрабатывается значительно быстрее.

Язык Java зародился как часть проекта создания передового программного обеспечения для различных бытовых приборов. Реализация проекта была начата на языке C++, но вскоре возник ряд проблем, наилучшим средством борьбы с которыми было изменение самого инструмента — языка программирования. Стало очевидным, что необходим платформо-независимый язык программирования, позволяющий создавать программы, которые не приходилось бы компилировать отдельно для каждой архитектуры и можно было бы использовать на различных процессорах под различными операционными системами.

Язык Java потребовался для создания интерактивных продуктов для сети Internet. Фактически, большинство архитектурных решений, принятых при создании Java, было продиктовано желанием предоставить синтаксис, сходный с C и C++. В Java используются практически идентичные соглашения для объявления переменных, передачи параметров, операторов и для управления потоком выполнением кода. В Java добавлены все хорошие черты C++.

Три ключевых элемента объединились в технологии языка Java:

- Java предоставляет для широкого использования свои апплеты (applets) — небольшие, надежные, динамичные, не зависящие от платформы активные сетевые приложения, встраиваемые в страницы Web. Апплеты Java могут настраиваться и распространяться потребителям с такой же легкостью, как любые документы HTML
- Java высвобождает мощь объектно-ориентированной разработки приложений, сочетая простой и знакомый синтаксис с надежной и удобной в

работе средой разработки. Это позволяет широкому кругу программистов быстро создавать новые программы и новые апплеты

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

4.2 Программный инструмент для разработки приложения

Unity — это инструмент для разработки двух- и трёхмерных приложений и игр, работающий под операционными системами Windows, Linux и OS X. Созданные с помощью Unity приложения работают под операционными системами Windows, OS X, Windows Phone, Android, Apple iOS, Linux, а также на игровых приставках Wii, PlayStation 3, PlayStation 4, Xbox 360, Xbox One и MotionParallax3D дисплеях (устройства для воспроизведения виртуальных голограмм), например, Nettlebox. Есть возможность создавать приложения для запуска в браузерах с помощью специального подключаемого модуля Unity (Unity Web Player), а также с помощью реализации технологии WebGL. Ранее была экспериментальная поддержка реализации проектов в рамках модуля Adobe Flash Player, но позже команда разработчиков Unity приняла сложное решение по отказу от этого (Рисунок 3.2).

Рисунок 3.2 – Unity

Приложения, созданные с помощью Unity, поддерживают DirectX и OpenGL. Активно движок используется как крупными разработчиками (Blizzard,

EA, QuartSoft, Ubisoft), так и разработчиками Indie-игр (например, ремейк Мор. Утопия (Pathologic), Kerbal Space Program, Slender: The Eight Pages, Slender: The Arrival, Surgeon Simulator 2013, Baeklyse Apps: Guess the actor и т. п.) в силу наличия бесплатной версии, удобного интерфейса и простоты работы с движком (Рисунок 3.3).

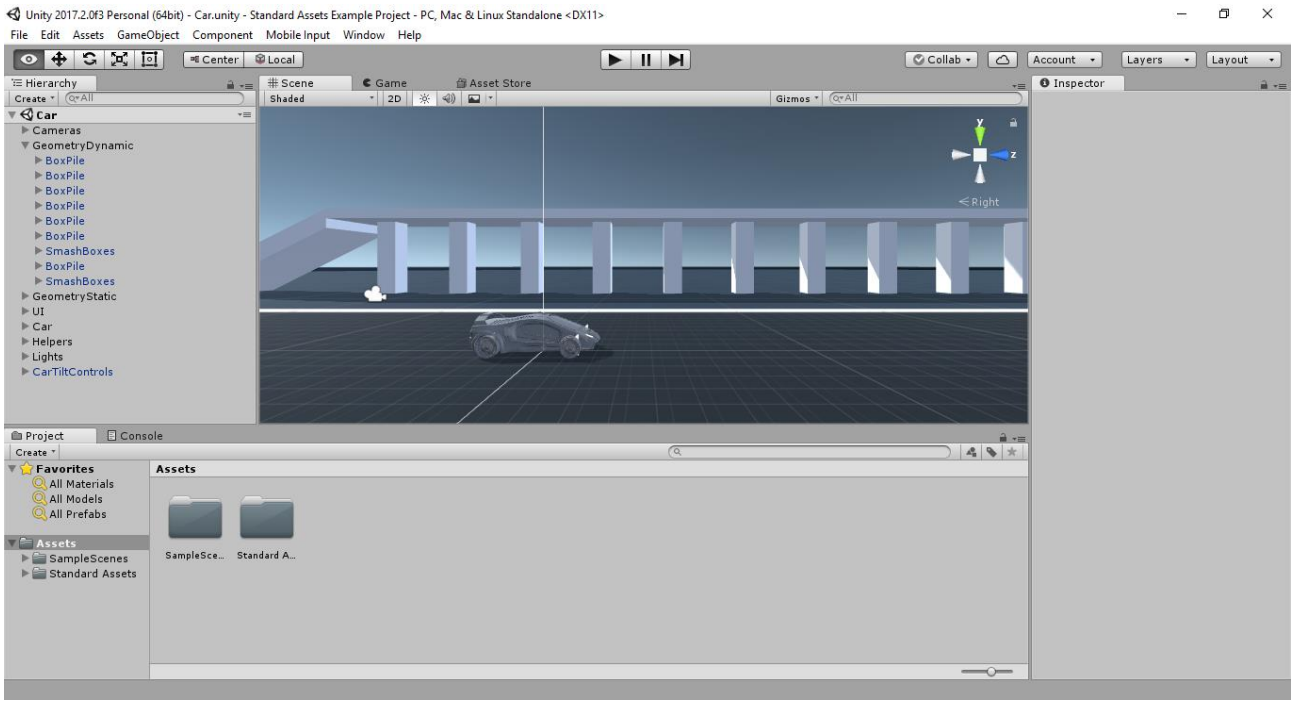


Рисунок 3.3 – Интерфейс управления

Редактор Unity имеет простой Drag&Drop интерфейс, который легко настраивать, состоящий из различных окон, благодаря чему можно производить отладку игры прямо в редакторе. Движок поддерживает два скриптовых языка: C#, JavaScript (модификация). Ранее была поддержка Boo (диалект Python), но его убрали в 5-ой версии. Расчёты физики производит физический движок PhysX от NVIDIA.

Проект в Unity делится на сцены (уровни) — отдельные файлы, содержащие свои игровые миры со своим набором объектов, сценариев, и настроек. Сцены могут содержать в себе как, собственно, объекты (модели), так и пустые игровые объекты — объекты, которые не имеют модели («пустышки»). Объекты, в свою очередь содержат наборы компонентов, с которыми и взаимодействуют скрипты. Также у объектов есть название (в Unity допускается наличие двух и более объектов с одинаковыми названиями), может быть тег (метка) и слой, на котором он должен отображаться. Так, у любого объекта на сцене обязательно присутствует компонент Transform — он хранит в себе координаты местоположения, поворота и размеров объекта по всем трём осям. У

объектов с видимой геометрией также по умолчанию присутствует компонент Mesh Renderer, делающий модель объекта видимой.

5 Практическая реализация

5.1 Схема прототипа приложения

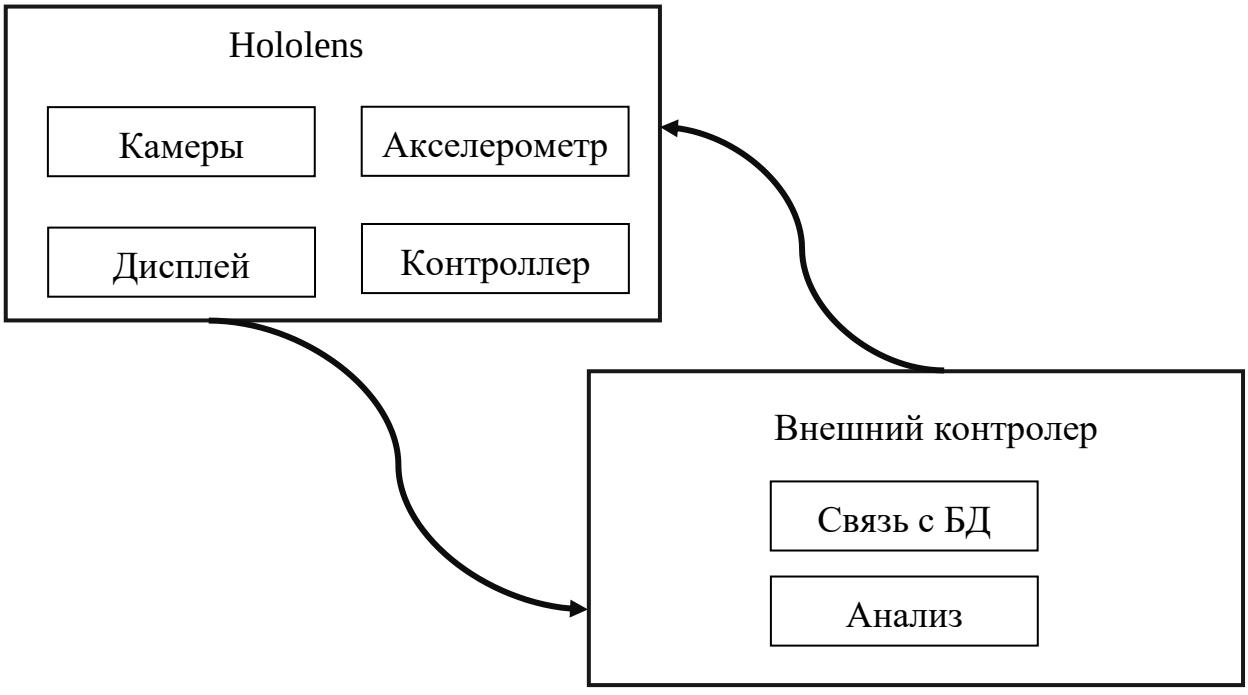
При создании программной реализации системы анализа на основе технологии дополненной реальности следует учитывать особенности интерфейса, с которым необходимо будет работать пользователю.

Следует понимать, что из-за особенностей очков дополненной реальности, единственными входными данными является видеопоток, т. е. для управления можно использовать только анализ изображений.

Итак, можно построить схему программного средства.

Сенсоры акселерометра, а также камеры AR-очков передают видеопоток, которые «раздваивается»: одна часть идёт сразу на дисплей очков, а другая отправляется в контроллер для дальнейшего анализа.

Далее, после того, как видеопоток проанализирован и получены определённые результаты, а также проведена привязка дополнительных элементов как результата работы анализирующего алгоритма после применения функционала Unity – второй видеопоток отправляется на интерфейс пользователя, где он соединяется с оригинальным видеорядом (Рисунок 4.1).



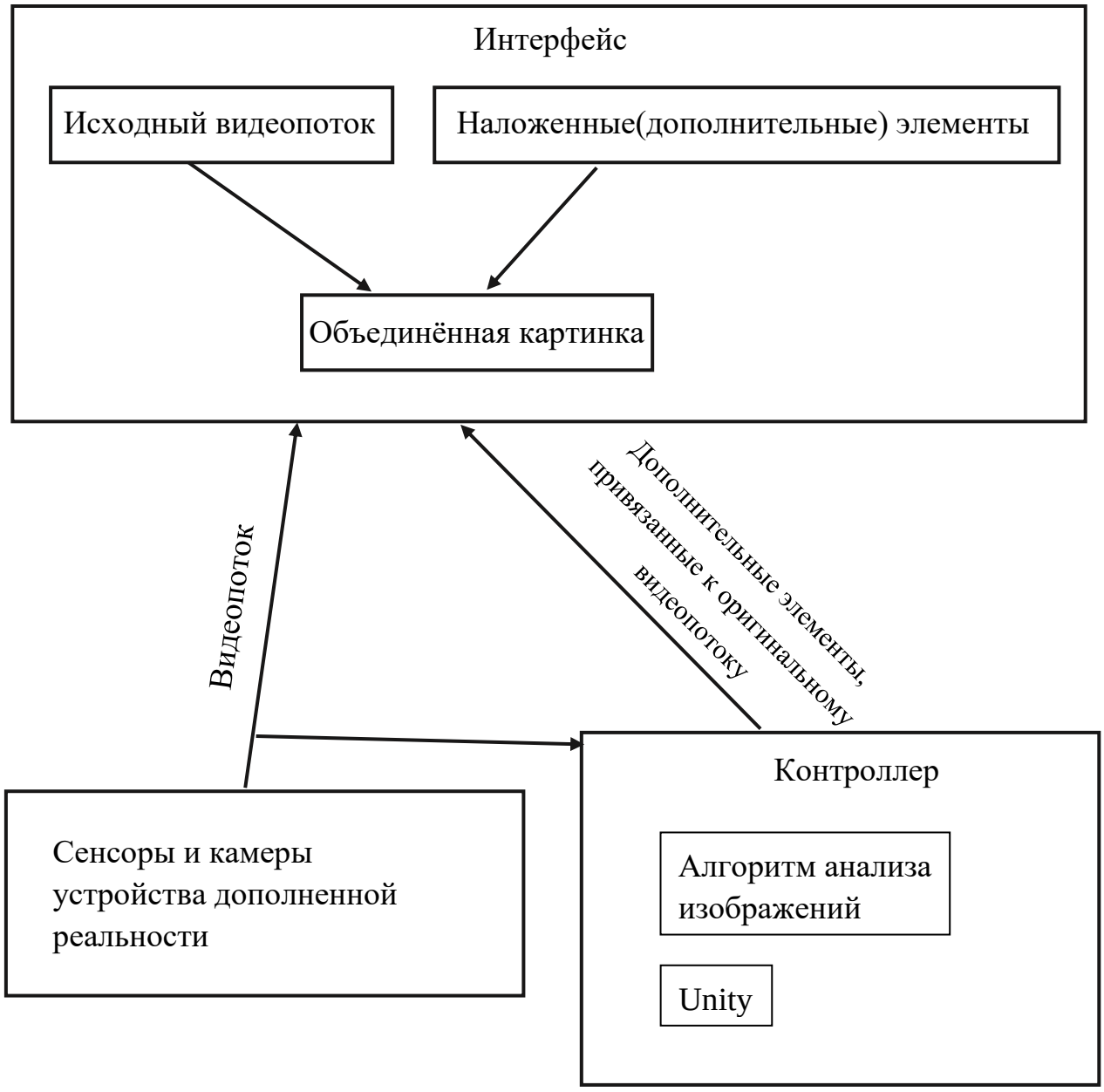


Рисунок 4.1 – Схемы работы программного средства для анализа

Исходя из вышеприведённой схемы, встаёт вопрос о задержке, которая возникает при анализе изображения. И в самом деле, если оригинальный видеопоток будет уходить сразу на экран пользователя, а элементы дополненной реальности появляться только после анализа и обработки, могут происходить некоторые задержки.

Но при этом стоит учитывать достаточно высокую производительность современных процессоров, а также и тот факт, что экраны для глаз пользователя хоть и выдают качественную картинку, но тем не менее не имеют высокого разрешения – соответственно, не требуют больших вычислительных затрат.

ЗАКЛЮЧЕНИЕ

В ходе данной работы исследованы возможности систем дополненной реальности для анализа и обработки входного потока информации в виде видеоряда, последовательности изображений, а также способов вывода результатов работы.

Проанализированы дополненная и виртуальная реальность, а также существующие системы реализации дополненной реальности.

Определены необходимые технологии для реализации системы. Реализована схема прототипа программного средства анализа последовательности снимков и построен алгоритм анализа видеопотока.

Практическая значимость данного исследования состоит в разработке алгоритма и подготовке для программной реализации системы анализа видеопотока.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ричард, A. Grome Terrain Modeling with UDK, and Unity3D / A. Ричард. – СПб: Packt Publishing Ltd, 2013. – 162 с.
2. Гриффитс, И. Программирование на C# 5.0 / И. Гриффитс. – Москва: Эксмо, 2014. – 1136 с.
3. Макфарланд, Д. Большая книга CSS3 / Д. Макфарланд – СПб: Издательский дом «Питер», 2011. – 560 с.
4. Гонсалвес, Э. Изучаем Java EE 7 / Э. Гонсалвес. – СПб: Питер, 2017. – 640 с.
5. Эккель, Б. Философия Java / Б. Эккель. – СПб: Питер, 2017. – 1168 с.
6. Фримен, A. Pro ASP.NET MVC 4 / A. Фримен, С. Сандерсон. – 4-е изд. – Москва: Вильямс, 2011. –672 с.
7. Роджерс, Д. Математические основы машинной графики /Д. Роджерс. – Москва: Мир, 2001. - 604 с.