Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н. Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Р А С Ч Ё Т Н О - П О Я С Н И Т Е Л Ь Н А Я З А П И С К А

к курсовой работе на тему:

«Программа для бесконтактного оптического измерения геометрических параметров объекта»

Студент _____ Анисимов Н.С.
(Подпись, дата)

Руководитель курсового проекта _____ Куров А.В.
(Подпись, дата)

Москва 2017

Содержание

Введение

Промышленные измерительные системы позволяют проводить бесконтактные измерения геометрических размеров и формы сырья, заготовок, деталей и готовой продукции в процессе производства непосредственно на конвейере или в производственной линии.

В настоящее время ставится задача разработки программного обеспечения, позволяющего обрабатывать графические изображения (фотографии или видео) и определять геометрические параметры объектов (исходных заготовок произвольной формы), находящихся на движущемся конвейере или непосредственно в зоне обработки, для выдачи корректирующего сигнала в общую систему адаптивного контроля и управления кузнечно-прессовым оборудованием.

Целью данной работы является измерение геометрических параметров цилиндра, изображенном на фотографии, а так же визуализации процесса движения цилиндра по конвейеру.

1 Аналитический раздел

Данный раздел посвящен анализу предметной области и задачи распознавания цилиндра и нахождения его геометрических размеров. Здесь также проводится обзор существующих методов и алгоритмов, позволяющих решить эту задачу. Формально, алгоритм нахождения размеров состоит из нескольких шагов:

а) Выделение границ цилиндра

б) Нахождение боковых граней цилиндра

в) На основе расстояния между найденными гранями и параметров камеры, вычислить размеры

Далее так же приводится анализ алгоритм трехмерного моделирования.

1.1 Анализ алгоритмов нахождения границ изображения

Для решения данной задачи, было разработаны большое количество алгоритмов. Далее приведено описание наиболее популярных и используемых. Также для ускорения работы алгоритмов, изображение желательно перевести в градации серого путем вычисления средневзвешенного количества красного, зеленого и синего цветов. в. Используется формула

$$Y = 0.3 * R + 0.59 * G + 0.11 * B$$

эта величина также известна как свечение. Коэффициенты, используемые для расчета свечения, выбраны из соображений особенностей человеческого восприятия. Дело в том, что из базовых цветов, взятых в одинаковом количестве, человеческих глаз сперва выделяет зеленый, затем красный, а уже потом синий. Подразумевается, что когда зеленый и синий цвета излучаются монитором в одинаковом количестве, зеленый, тем не менее, выглядит ярче. Поэтому преобразование в градации серого цвета путем вычисления среднего арифметического цветовых компонент не отражает воспринимаемую человеком яркость оригинала. Для этого используется средневзвешенная величина.

1.1.1 Оператор Собеля

Оператор Собеля – дискретный дифференциальный оператор, вычисляющий приближенные значения производных разного порядка для функции яркости пикселей [1]. Наиболее распространенным примером практического использования является определение границ (ребер) объектов на изображении, т.е. точек резкого изменения яркости.

Данный оператор основан на свертке изображения с целочисленными фильтрами. В простейшем случае оператор построен на вычислении сверток исходного

изображения с ядрами G_x и G_y , обеспечивающими вычисление первых производных по направлениям:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{1.1}$$

Данный оператор используется для приближенного вычисления градиента функции интенсивности пикселей. Применение оператора G_x позволяет определить приближенное значение первой частной производной изменения интенсивности в горизонтальном направлении, G_y – в вертикальном. На основании данной информации можно вычислить магнитуду градиента для пикселя с координатами (i,j) согласно формуле $\mid G^{ij} \mid = \sqrt{(G_x^{ij})^2 + (G_y^{ij})^2}$. Также используя полученные данные, можно определить направление градиента как $\theta^{ij} = arctan\left(\frac{G_y}{G_x}\right)$.

1.1.2 Оператор Приютта

Оператор Прюитта похож на оператор Собеля, но используются другие ядра:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{1.2}$$

1.1.3 Оператор Робертса

Оператор Робертса — один из ранних алгоритмов выделения границ, который вычисляет сумму квадратов разниц между диагонально смежными пикселами. Это может быть выполнено сверткой изображения с двумя ядрами:

$$G_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, G_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \tag{1.3}$$

.

1.1.4 Оператор Кэнни

Данный алгоритм был разработан Джоном Кэнии в 1986 году. Его целью было разработать алгоритм алгоритм, который был бы оптимален по следующим критериям:

а) Определение: вероятность определения точек реальных ребер должна быть максимальна, в то время как вероятность нахождения точек ложных ребер должна быть минимальна.

б) Локализация: определенные ребра должны быть максимально близко к реальным ребрам.

в) Количество ответов: одно реальное ребро, должно давать не более одного распознанного ребра.

Алгоритм состоит из следующих пяти шагов.

Сглаживание. Неминуемо, что каждая фотография будет иметь некоторые шумы. Чтобы предотвратить ошибочное определение шумов как ребер, необходимо их удалить. Для этого к изображению применяется фильтр Гаусса. Ядро фильтра Гаусса с нормальным распределением $\sigma = 1.4$ выглядит следующим образом:

$$B = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \tag{1.4}$$

Поиск градиентов. Алгоритм Кэнни находит границы там, где интенсивность изображения меняется больше всего. Эти области находятся вычислением значений и направлений градиента. Градиент можно найти применением оператора Собеля, описанном в пункте 1.1.1.

Подавление немаксимумов. Целью данного шага является преобразование “размытых” границ на изображении градиентов в “четкие”. Это может сделано сохранением всех локальных максимумов и подавлением всего остального. Алгоритм для каждого пикселя выглядит следующим образом:

— Округление направления градиента θ до ближайшего значения, кратного 45°.

— Сравнение значения текущего пикселя, со значениями пекселей в положительном и отрицательном направлении градиента.

— Если значение текущего пикселя наибольшее, то сохраняем, иначе – подавляем.

Двойная пороговая фильтрация. Пиксели, оставшиеся после предыдущего шага, вероятнее всего действительно будут ребрами, но некоторые из них могут быть получены из-за шумов или изменения цвета. Детектор границ Кэнни использует двойную пороговую фильтрацию. Пиксели, значения которых больше, чем верхняя граница, помечаются как “сильные”. Пиксели, значения которых меньше, чем нижняя граница, подавляются, и пиксели, значения которых находятся между двумя границами помечаются как “слабые”.

Трассировка области неоднозначности. “Сильные” пиксели, оставшиеся после предыдущего шага сразу добавляются в результирующие изображение. “Сла-

бые" пиксели будут добавлены только в том случае, если они соприкасаются с "сильными".

1.2 Анализ алгоритма нахождения прямых линий

Наиболее распространенным алгоритмом поиска прямых является преобразование Хафа. В простейшем случае преобразование Хафа является линейным преобразованием для обнаружения прямых. Прямая может быть задана уравнением $y = mx+b$ и может быть вычислена по любой паре точек (x, y) на изображении. Главная идея преобразования Хафа — учесть характеристики прямой не как уравнение, построенное по паре точек изображения, а в терминах её параметров, то есть m — коэффициента наклона и b — точки пересечения с осью ординат. Исходя из этого прямая, заданная уравнением $y = mx + b$, может быть представлена в виде точки с координатами (b, m) в пространстве параметров.

Однако прямые, параллельные оси ординат, имеют бесконечные значения для параметра m. Поэтому удобней представить прямую с помощью других параметров, известных как r и θ. Параметр r — это длина радиус-вектора ближайшей к началу координат точки на прямой (т.е. нормали к прямой, проведенной из начала координат), а θ — это угол между этим вектором и осью абсцисс. При таком описании прямых не возникают бесконечные параметры.

Таким образом, уравнение прямой можно записать как:

$$y = \frac{-\cos\theta}{\sin\theta}x + \frac{r}{\sin\theta} \tag{1.5}$$

или

$$r = x\cos\theta + y\sin\theta \tag{1.6}$$

1.3 Анализ алгоритмов удаление невидимых линий

1.3.1 Алгоритм Робертса

Алгоритм Робертса представляет собой первое известное решение задачи об удалении невидимых линий. Это метод, работающий в объектном пространстве. В соответствии с алгоритмом, прежде всего удаляются из каждого тела те ребра или грани, которые перекрываются самим телом. Затем каждое из видимых ребер каждого тела сравнивается с каждым из оставшихся тел для определения того, какая его часть или части, если таковые есть, перекрываются этими телами.

Преимущества данного алгоритма в том, что математические методы, используемые в нем, просты, мощны и точны. Более поздние реализации алгоритма, например использующие предварительную сортировку вдоль оси z, демонстрируют почти линейную зависимость от числа объектов.

Минус этого алгоритма в том, что вычислительная трудоемкость алгоритма Робертса растет теоретически, как квадрат числа объектов. Реализация оптимизированных алгоритмов весьма сложна.

1.3.2 Трассировка лучей

В этом методе для каждого пикселя картинной плоскости определяется ближайшая к нему грань, для чего через этот пиксель выпускается луч, находятся все его пересечения с гранями и среди них выбирается ближайшая.

К достоинствам данного алгоритма можно отнести возможность получения изображения гладких объектов без аппроксимации их примитивами (например, треугольниками). Вычислительная сложность метода линейно зависит от сложности сцены. Нетрудно реализовать наложение света и тени на объекты. Качество полученного изображения получается очень реалистичным, этот метод отлично подходит для создания фотореалистичных картин.

Серьёзным недостатком алгоритма трассирования является производительность. Для получения изображения необходимо создавать огромное число лучей, проходящих через сцену и отражаемых от объекта. Это приводит к существенному снижению скорости работы программы.

1.3.3 Алгоритм, использующий z-буффер

Алгоритм, использующий z-буфер это один из простейших алгоритмов удаления невидимых поверхностей. Впервые он был предложен Кэтмулом. Работает этот алгоритм в пространстве изображения. Идея z-буфера является простым обобщением идеи о буфере кадра. Буфер кадра используется для запоминания атрибутов (интенсивности) каждого пиксела в пространстве изображения, z-буфер - это отдельный буфер глубины, используемый для запоминания координаты z или глубины каждого видимого пиксела в пространстве изображения. В процессе работы глубина или значение z каждого нового пиксела, который нужно занести в буфер кадра, сравнивается с глубиной того пиксела, который уже занесен в z-буфер. Если это сравнение показывает, что новый пиксел расположен впереди пиксела, находящегося в буфере кадра, то новый пиксел заносится в этот буфер и, кроме того, производится корректировка z-буфера новым значением z. Если же сравнение дает противоположный результат, то никаких действий не производится. По сути, алгоритм является поиском по x и y наибольшего значения функции $z(x,y)$.

Главное преимущество алгоритма – его простота. Кроме того, этот алгоритм решает задачу об удалении невидимых поверхностей и делает тривиальной визуализацию пересечений сложных поверхностей. Сцены могут быть любой сложности. Поскольку габариты пространства изображения фиксированы, оценка вычислитель-

ной трудоемкости алгоритма не более чем линейна. Поскольку элементы сцены или картинки можно заносить в буфер кадра или в z-буфер в произвольном порядке, их не нужно предварительно сортировать по приоритету глубины. Поэтому экономится вычислительное время, затрачиваемое на сортировку по глубине.

Основной недостаток алгоритма - большой объем требуемой памяти. Если сцена подвергается видовому преобразованию и отсекается до фиксированного диапазона значений координат z, то можно использовать z-буфер с фиксированной точностью. Информацию о глубине нужно обрабатывать с большей точностью, чем координатную информацию на плоскости (x, y); обычно бывает достаточно 20-ти бит. Буфер кадра размером 512x512x24 бит в комбинации с z-буфером размером 512x512x20 бит требует почти 1.5 мегабайт памяти. Однако снижение цен на память делает экономически оправданным создание специализированных запоминающих устройств для z-буфера и связанной с ним аппаратуры.

1.4 Анализ модели освещения по Фонгу

Расчет освещения по Фонгу требует вычисления цветовой интенсивности трех компонент освещения: фоновой, рассеянной и глянцевых бликов . Фоновая компонента — грубое приближение лучей света, рассеянных соседними объектами и затем достигших заданной точки; остальные две компоненты имитируют рассеивание и отражение прямого излучения.

Иллюстрация различных компонент, соединенных в модели Фонга

$$I = K_a I_a + K_d(\vec{n},\vec{l}) + K_s(\vec{n},\vec{h})^p \tag{1.7}$$

где

$\vec{n}$ — вектор нормали к поверхности в точке

$\vec{l}$— направление проецирования (направление на источник света)

$\vec{h}$ — направление на наблюдателя

K_a — коэффициент фонового освещения

K_s — коэффициент зеркального освещения

K_d — коэффициент диффузного освещения

1.5 Выводы и выбор алгоритмов для поставленной задачи

В качестве алгоритма выделения границ был использован на детектор границ Кэнни. Этот алгоритм дает наиболее точную границу, по сравнению со всеми выше описанными. Трансформация Хафа была использована для нахождения проекций цилиндрической поверхности.

Для удаления невидимых граней в трехмерной анимации был выбран алгоритм z-буффера. В силу простоты сцены, нет необходимости использовать эффек-

тивные алгоритмы, но в тоже время для сохранения плавности анимации нельзя использовать ресурсоемкие алгоритмы.

2 Конструкторский раздел

В данном разделе приводится описание общей структуры разрабатываемой системы, её основных модулей и структур данных. Подробно описываются методы и алгоритмы, выбранные для использования в системе.

2.1 Структуры данных, используемых при нахождении размеров цилиндра

Для нахождения геометрических размеров цилиндра использовались следующие структуры данных:

— `Vector2`. Двухмерный вектор. Содержит в себе два атрибута, координаты `x` и `y`.

— `CylinderSize`. Размер цилиндра. Содержит в себе два атрибута, высоту `H` и радиус `R`.

— `Line`. Отрезок. Содержит в себе два атрибута, вектор `v1` и вектор v2.

— `Color`. Цвет. Содержит в себе три атрибута, соответствующие значения цветовых каналов `r, g, b`.

— `ImageBase`. Базовый тип изображения, от него наследуются цветное изображение `ColorImage` и изображение в градациях серого `GrayscaleImage`. Содержит в себе массив значений пикселей изображения.

2.2 Алгоритм нахождения размеров цилиндра

Схема алгоритма нахождения размеров цилиндра представлена на рисунке 2.1.

2.3 Решение задачи определения геометрических размеров цилиндра

При съемке цилиндра, с указанной позиции, диаметр цилиндра не виден полностью, а лишь его часть, как показано на рисунке 2.2.

Зависимость размера видимой части цилиндра от высоты камеры $CD = H$ и радиуса R можно вычислить согласно уравнениям (2.1) - (2.4).

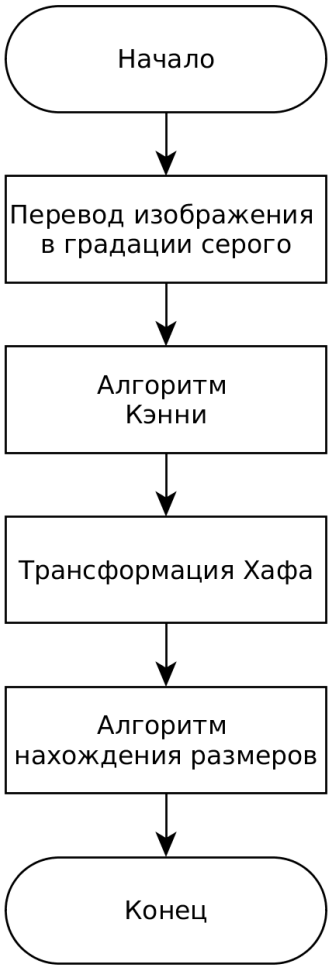

Рисунок 2.1 — Схема алгоритма нахождения размеров

$$\sin \angle ACO = \frac{AO}{CO} = \frac{R}{H - R} \tag{2.1}$$

$$\cos \angle ACO = \sqrt{1 - \sin^2 ACO} = \sqrt{1 - (\frac{R}{H - R})^2} \tag{2.2}$$

$$AE = R \cos \angle OAE = R \cos \angle ACO = R\sqrt{1 - (\frac{R}{H - R})^2} \tag{2.3}$$

$$AB = 2AE = 2R\sqrt{1 - (\frac{R}{H - R})^2} \tag{2.4}$$

Длина AB будет соответствовать одному определенному значению радиуса R на промежутке I, на котором длина AB возрастает. Верхнюю границу промежутка I можно найти из производной функции 2.1. Для случая, когда камера расположена на высоте $H = 1$ метр, действителен следующий график:

Из данной зависимости видно, что для корректной работы алгоритма, для данного случая, радиус цилиндра не должен превышать значение $R = 0.38197$ метра. В противном случае возникает неоднозначность. Таким образом для промежутка I значение AB однозначно соответствует радиусу цилиндра и возможно однозначно

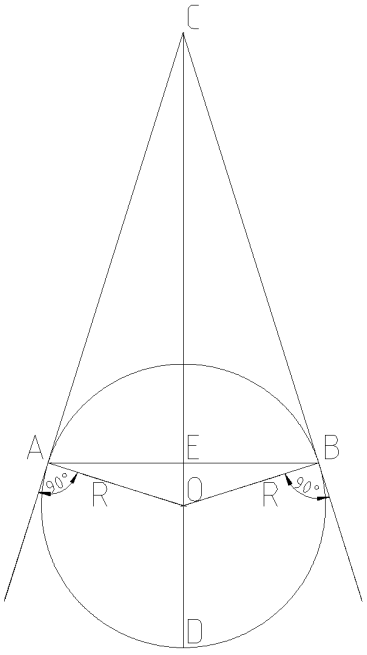

Рисунок 2.2 — Видимость цилиндра

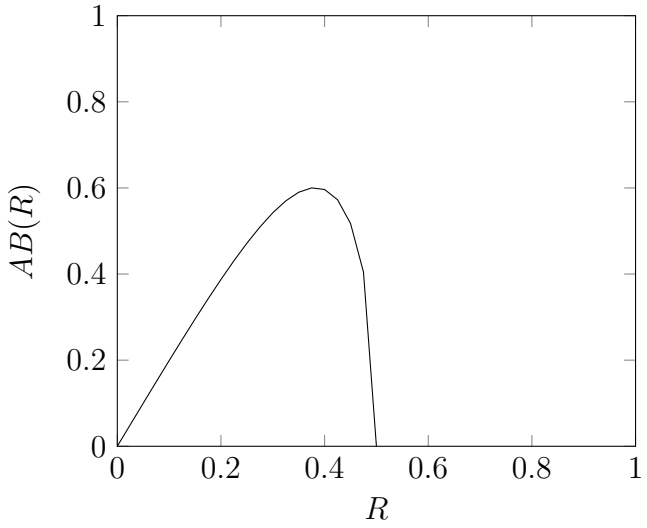

Рисунок 2.3 — Зависимость AB(R)

определить высоту на которой находится AB, и это значение можно использовать для вычисления масштабного коэффициента.

В вычислениях было сделано допущение, что лучи сходятся в одной точке. На самом же деле, лучи равномерно попадают на матрицу, которая имеет некоторые линейные размеры. Они были опушены, для упрощения вычислений и настройки программы.

2.4 Структуры данных, используемых для трехмерного моделирования

Для нахождения геометрических размеров цилиндра использовались следующие структуры данных:

— `Vector3`. Трехмерный вектор. Содержит в себе 3 атрибута, компоненты x, y, z.

— `Scene`. Сцена хранит все объекты в динамическом массиве, которые можно на ней разместить: модель, камера, свет.

— `SceneObject`. Абстрактный класс объекта сцены. От него наследуются модель `Model`, камера `Camera`, абстрактный класс света `Light`. Каждый из перечисленных классов содержит в себе атрибуты, необходимые для его корректной работы.

— `Light`. Абстрактный класс света. От него наследуются модель рассеянный свет `AmbientLight`, точечный источник света `PointLight`.

— `Vertex`. Вершина. Состоит из координат, и нормали в данной позиции.

— `Mesh`. Меш — набор вершин и многоугольников, определяющих форму трёхмерного объекта. Состоит из динамического массива вершин, и динамического массива треугольников `Triangle`. Треугольник содержит в себе три индекса вершин из массива меша.

— `Transformation`. Абстрактный класс преобразования объекта сцены. От него наследуются перемещение объекта сцены `MoveTransformation`, масштабирование – `scaleTransformation`, поворот вокруг соответствующих осей – `RotateXTransformation`, `RotateYTransformation`, `RotateZTransformation`, произвольное преобразование `CommonTransformation`, задаваемое матрицей `Mat4`. Классы содержат в себе матрицу преобразование и точку, относительно которой необходимо совершить преобразование.

— `Renderer`. Абстрактный класс преобразования объекта сцены. От него происходит рендер z-буффера со светом `LightZBufferRenderer`.

2.5 Алгоритм рендеринга сцены

Схема алгоритма рендеринга сцены представлена на рисунке 2.4.

2.6 Алгоритм закраски

Схема алгоритма закраски треугольника представлена на рисунке 2.5.

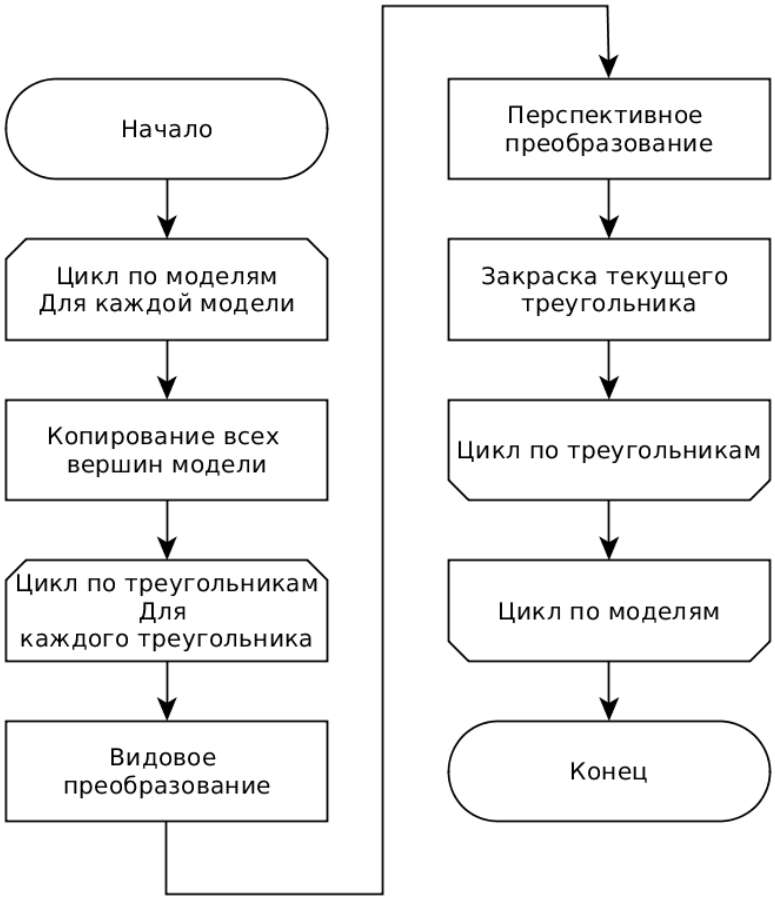

Рисунок 2.4 — Схема алгоритма рендеринга сцены.

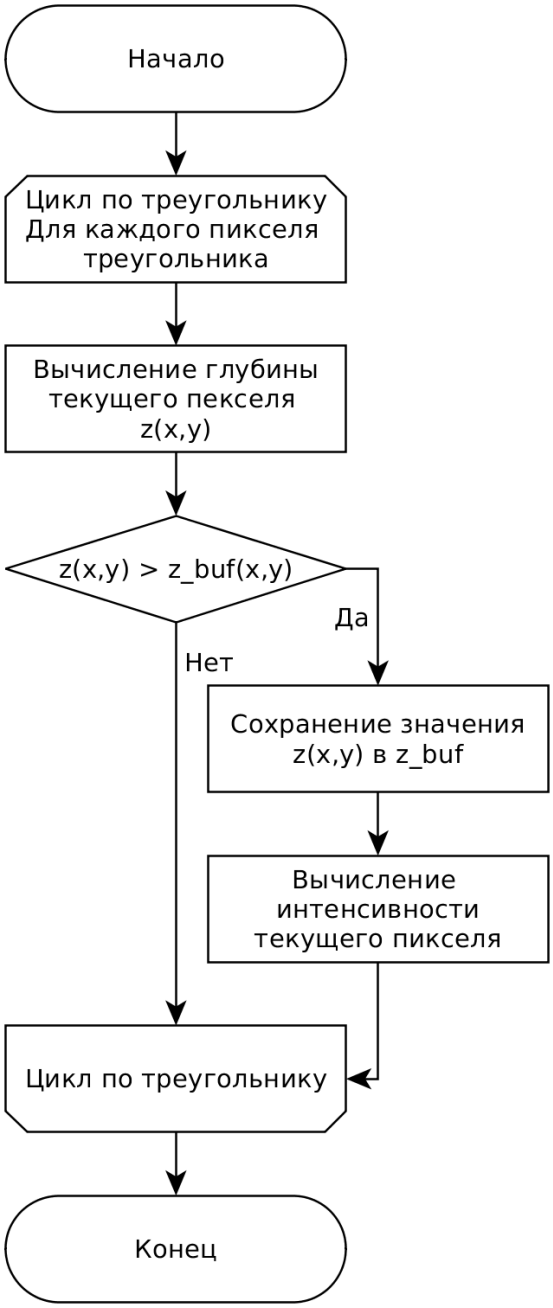

Рисунок 2.5 — Схема алгоритма закраски треугольника.

3 Технологический раздел

Цель технологического раздела – описание инструментов, средств разработки, языков программирования и технологий, использованных при реализации программного комплекса системы.

3.1 Выбор и обоснование языка программирования

В качестве языка разработка был выбран C++. Причины, по которым был выбран именно он следующие:

— Статическая типизация
— Высокая скорость выполнения
— Поддержка ООП
— Поддержка механизма исключений
— Последние стандарты расширили возможности языка, за счет чего разработка становится очень удобной

Выбор среды разработка пал на Qt Creator по следующим причинам:

— Удобный редактор
— Удобные средства отладки
— Поддержка псевдо-асинхронной модели программирования
— Большое количество встроенных библиотек
— Частичная платформонезависимость

3.2 Интерфейс пользователя

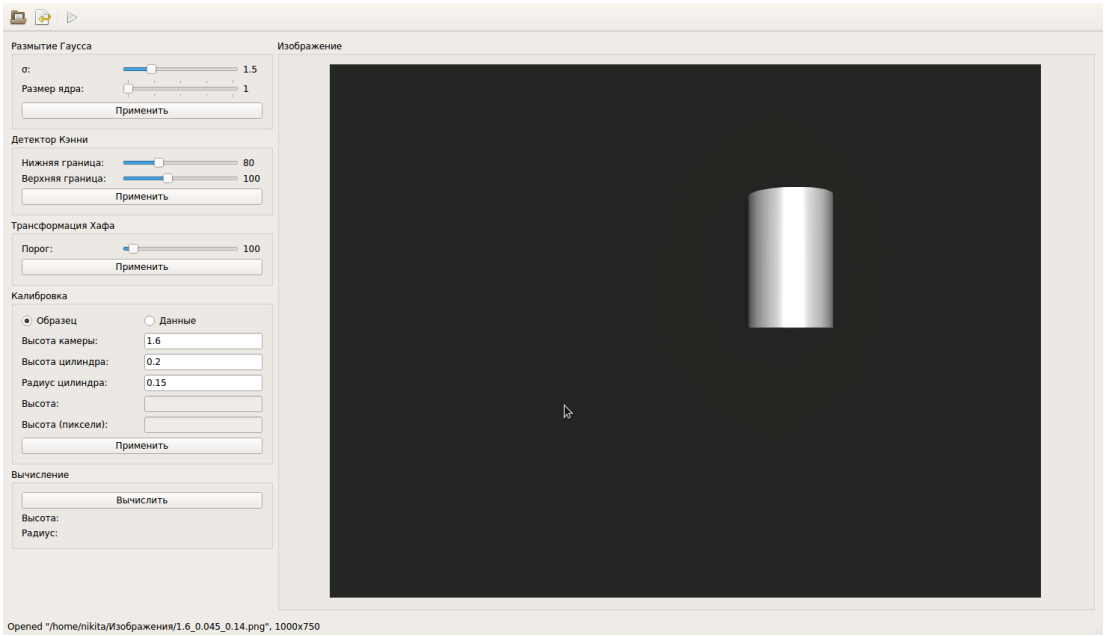

Рисунок 3.1 — Главное окно.

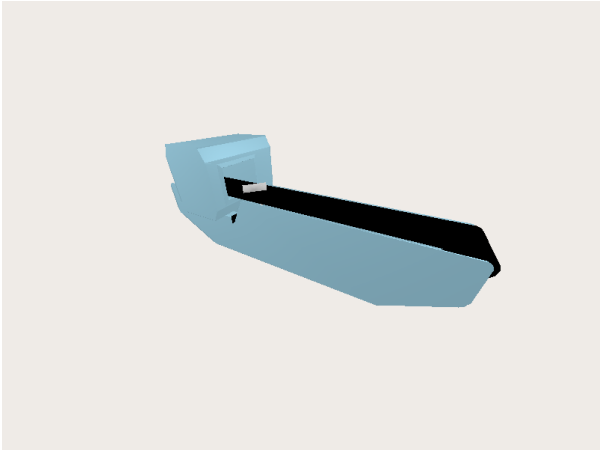

Рисунок 3.2 — Окно анимации.

3.3 Входные данные

Для работы алгоритма Кэнни необходимо задать: $\sigma \in [0..5]$ и размер ядра $\in \overline{1,3..9}$ для сглаживания Гаусса, нижнюю и верхнюю границу пороговой фильтрации $\in \overline{10,240}$.

Для настройки преобразования Хафа необходимо задать минимальную длину искомого отрезка $\in \overline{10,1000}$.

Также необходимо провести калибровку системы. Сделать это можно двумя путями:

а) На основе фотографии цилиндра, с известными размерами. В этом случае необходимо задать высоту камеры относительно ленты конвейера, геометрические размеры цилиндра, изображенного на фотографии.

б) На произвольного отрезка на самой ленте конвейера. Требуется указать действительную длину этого отрезка, и длину этого отрезка на изображении в пикселях.

3.4 Основные диаграммы классов модуля нахождения размеров

Диаграммы классов и пояснения к ним представлены на рисунках.

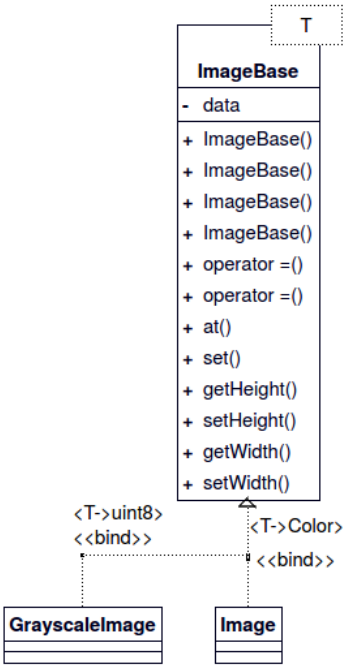

Рисунок 3.3 — Диаграмма классов изображения.

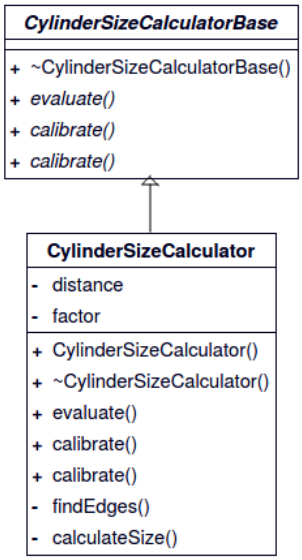

Рисунок 3.4 — Диаграмма классов вычислительного модуля. Использован паттерн стратегия, для возможности безболезненной замены.

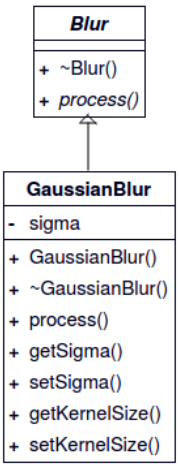

Рисунок 3.5 — Диаграмма классов размытия. Паттерн стратегия.

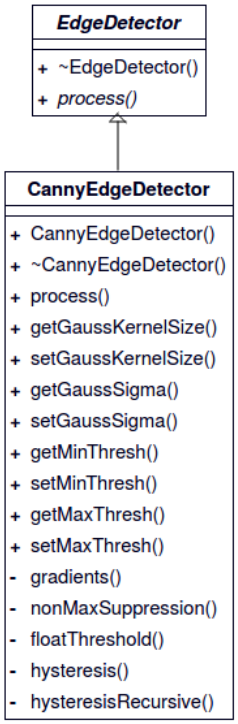

Рисунок 3.6 — Диаграмма классов детектора границ. Паттерн стратегия.

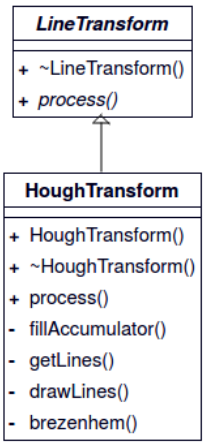

Рисунок 3.7 — Диаграмма классов детектора прямой. Паттерн стратегия.

3.5 Основные диаграммы классов модуля моделирования

Диаграммы классов и пояснения к ним представлены на рисунках.

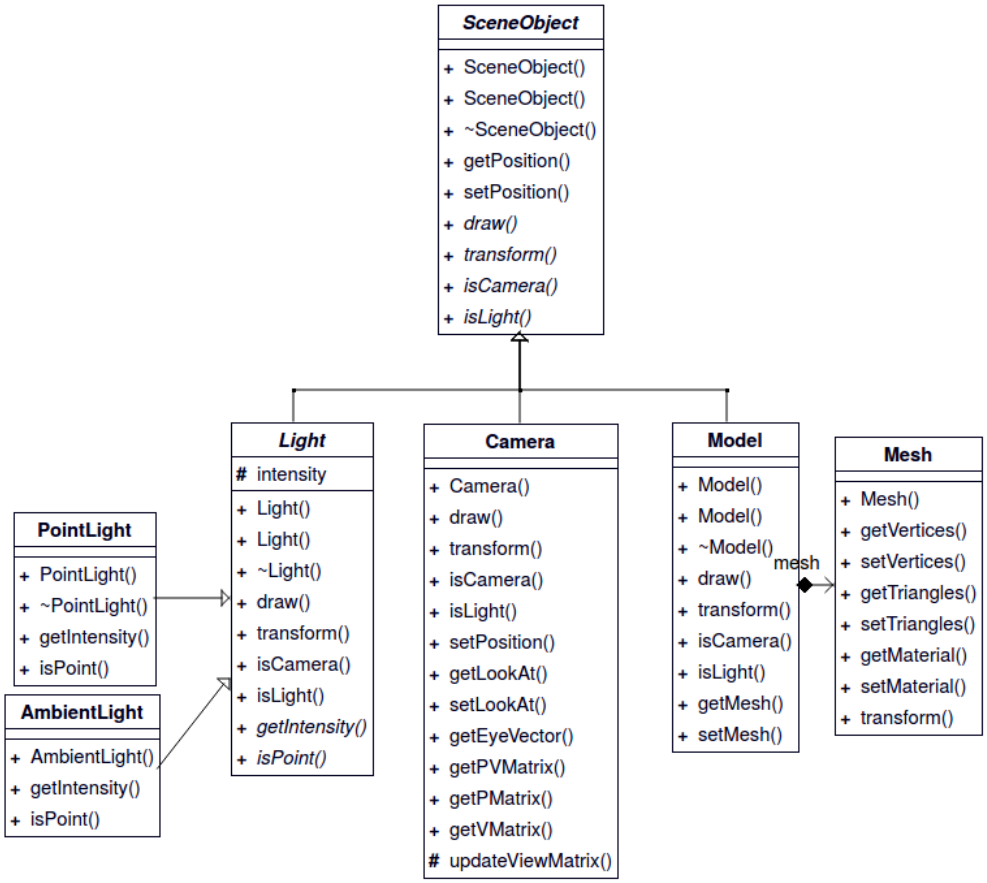

Рисунок 3.8 — Диаграмма класс объектов сцены.

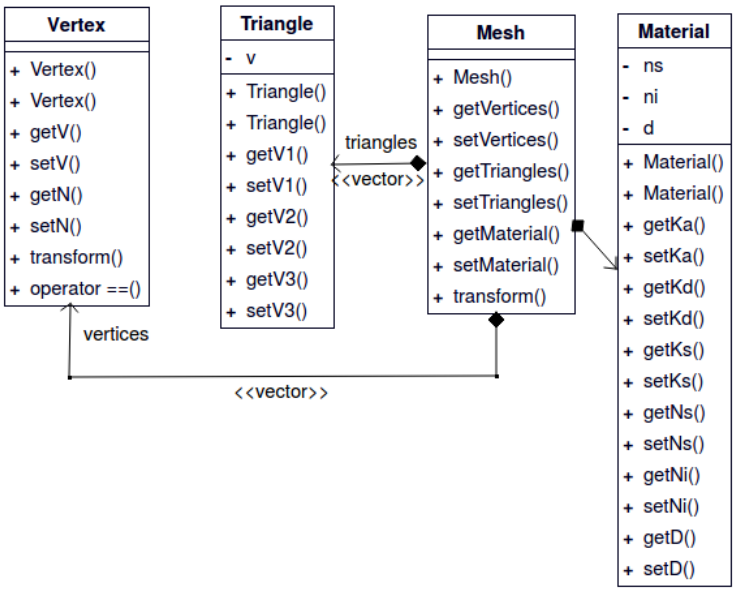

Рисунок 3.9 — Диаграмма класс объектов меша.

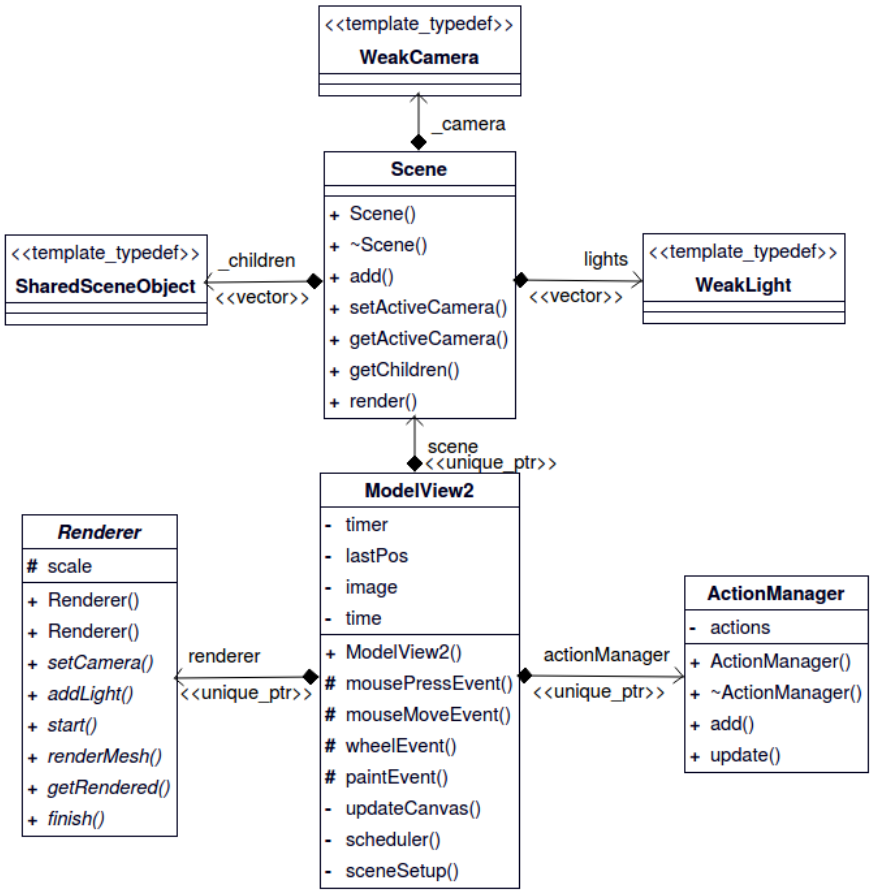

Рисунок 3.10 — Диаграмма классов сцены, рендера и менеджера анимаций. Для рендера использован паттерн посетитель, для абстрагирования от конкретных операций отрисовки для каждого объекта.

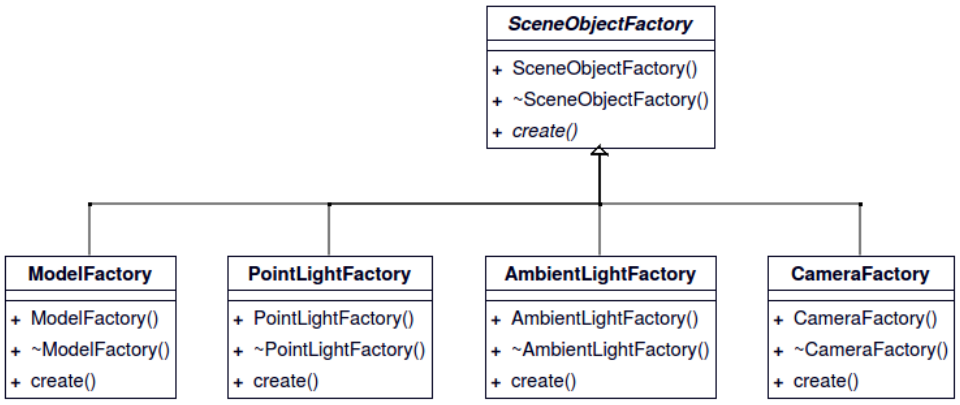

Рисунок 3.11 — Фабрика объектов. Паттерн фабричный метод.

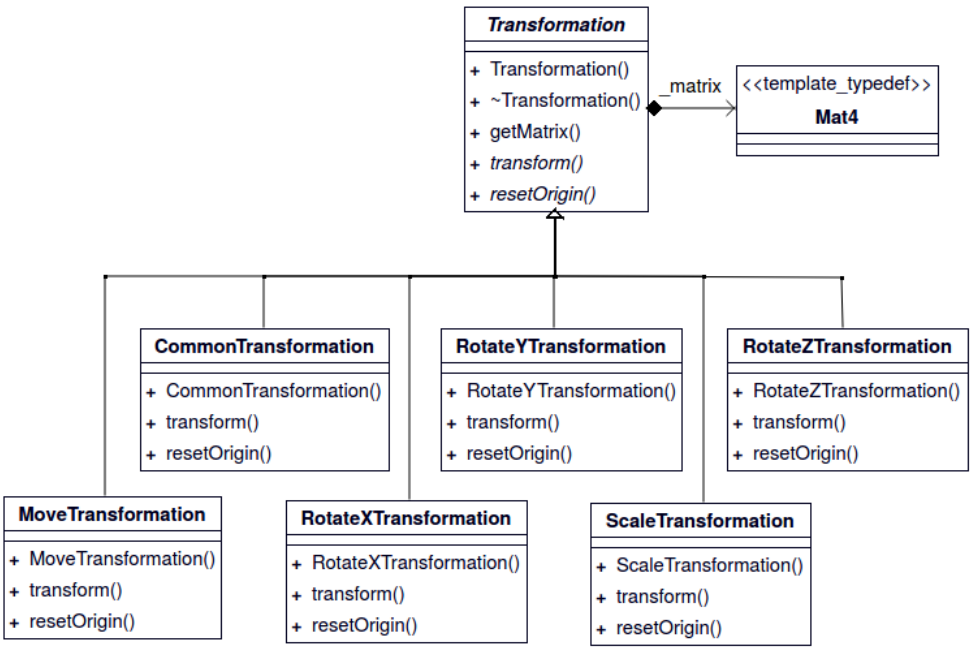

Рисунок 3.12 — Диаграмма классов трансформации объектов сцены. Паттерн стратегия.

4 Экспериментальный раздел

Цель экспериментального раздела – пример работы программы, расчет погрешности.

4.1 Погрешность измерения размером цилиндра

Для тестирования было взято три различных цилиндра, сфотографированных с высоты 1 метр 60 сантиметром:

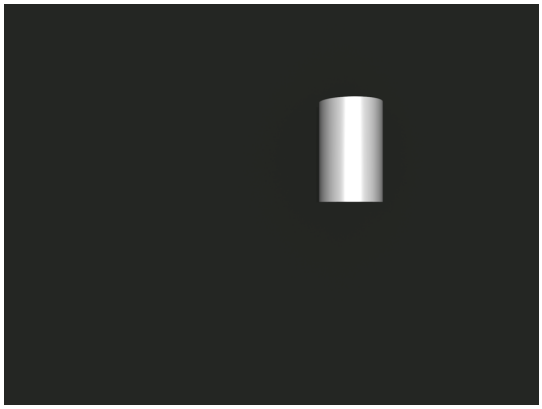

Рисунок 4.1 — Цилиндр 1. Высота 14 сантиметров, радиус 4.5 сантиметра.

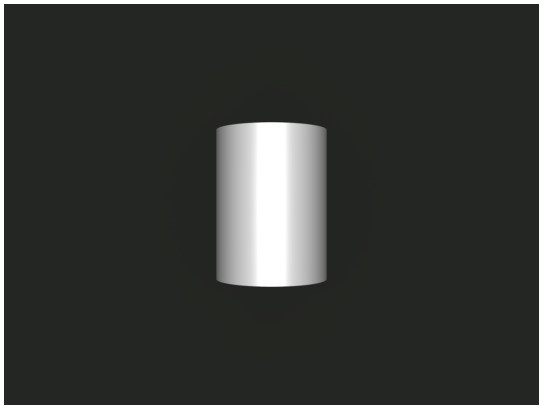

Рисунок 4.2 — Цилиндр 2. Высота 20 сантиметров, радиус 7.5 сантиметра.

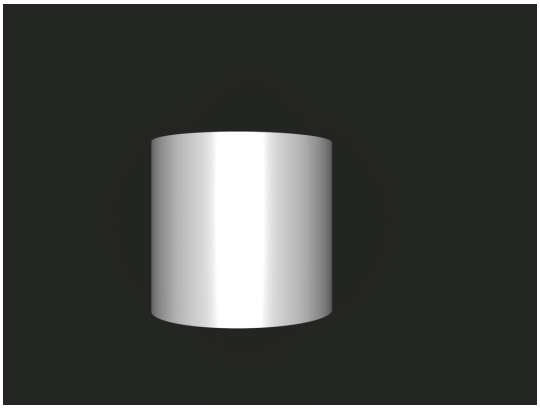

Рисунок 4.3 — Цилиндр 3. Высота 21 сантиметр, радиус 11.5 сантиметра.

В таблице 4.1 приведены результаты тестирования:

Измерение \ Калибровка		Цилиндр 1	Цилиндр 2	Цилиндр 3	Отрезок
Цилиндр 1	Радиус, м	0.044	0.043	0.041	0.046
	Погрешность, %	1.2	4.4	8.8	2.2
	Высота, м	0.14	0.136	0.129	0.144
	Погрешность, %	0	2.8	8.5	2.8
Цилиндр 2	Радиус, м	0.076	0.074	0.071	0.78
	Погрешность, %	1.3	1.3	5.3	4.4
	Высота, м	0.205	0.2	0.189	0.211
	Погрешность, %	2.5	0	5.5	4.4
Цилиндр 3	Радиус, м	0.121	0.117	0.112	0.124
	Погрешность, %	10	8.3	1.8	11.1
	Высота, м	0.227	0.222	0.21	0.234
	Погрешность, %	8.2	5.7	0	11

Таблица 4.1 — Таблица погрешностей

Погрешность можно уменьшить, путем увеличения разрешения изображения, использованием качественных линз, которые дает наименьшую дисторсию, увеличением расстояния между камерой и объектом, чтобы была видна наибольшая часть цилиндра.