САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА МЕХАНИКИ УПРАВЛЯЕМОГО ДВИЖЕНИЯ

Горбунов Владислав Игоревич

Магистерская диссертация

Система технического зрения для робота манипулятора

Направление 01.04.02

Прикладная математика и информатика

Магистерская программа Математическое моделирование в задачах естествознания

Научный руководитель,
доктор технических наук,
профессор
Кулаков Ф. М.

Санкт-Петербург

2017

Оглавление

Введение

В последние годы роботы стали больше влиять на нашу жизнь. Но роботы несовершенны - им нужны органы чувств, для восприятия информации об окружающей среде для взаимодействия с объектами, а также система управления, которая позволит получить наибольший КПД от выполнения роботом поставленных задач.

Задачи дистанционного управления роботами имеют ряд особенностей, такие как: задержка в передаче сигнала, разрывы канала связи, сложность в управлении, недостаток информации, получаемой по обратной связи об окружающей среде [1]. Данные особенности могут существенно усложнить работу человека-оператора, осуществляющего управление роботом.

В качестве одного класса из прикладных задач в сфере дистанционного управления роботами, можно выделить проведение монтажно-сборочных работ манипуляторами, расположенными на значительном удалении от оператора. Данный класс задач подразумевает организацию телеуправления роботом с использованием силомоментного очуствления робота и передачи сил и моментов по обратной связи оператору через устройство вывода в виде очувствленной рукоятки [2]. Для обеспечения оператора информацией во время проведения работ, с робота необходимо передавать информацию об окружающей среде по обратной связи [3]. Большое расстояния и сложная модель коммуникационной сети между оператором и роботом означают наличие задержки в передаче управляющего сигнала и обратной связи, а также ограничения на пропускную способность канала связи [4]. Таким образом, данный класс задач содержит следующие проблемы:

- Организация передачи необходимой информации для принятия решения оператором с системы технического зрения с множества камер в виде нескольких видеопотоков высокого качества представляется дорогостоящей для реализации задачей [5].

- Работа оператора в копирующем режиме с использованием очувствленой задающей рукоятки с большой задержкой представляется невозможной в динамической среде без проведения коррекции траектории на стороне робота [6].

Для решения данных проблем при управлении роботом-манипулятором необходимо создать мехатронную систему - программно-аппаратный комплекс по управлению роботами в условиях больших задержек в передаче управляющего сигнала [7]. Данный комплекс должен состоять из следующих модулей:

- робот - манипулятор,

- система технического зрения (СТЗ) [8],

- система копирующего управления инструментом робота [9],

- система определения сил и моментов, прилагаемых к рабочему инструменту манипулятора, основанная на запястном силомоментном датчике [10],

- компьютерная модель рабочей сцены робота с интерфейсом пользователя [11],

- ИИ для коррекции траектории движения и уменьшения ошибки в управлении до допустимого уровня.

В рамках данной работы, предлагается к рассмотрению разработка одного из модулей программно-аппаратного комплекса по управлению роботами манипуляторами – системы технического зрения, удовлетворяющей требованиям к отказоустойчивости системы, а также к ограничениям, налагаемым на вычислительные мощности устройства обработки информации.

Сначала дается постановка задачи, описывающей требования к создаваемой системе технического зрения. Далее предлагается математическая модель описывающая процесс нахождения траектории перемещения звеньев робота–манипулятора в пространстве для взаимодействия с искомым объектом на рабочей сцене, рассматриваются модели детектирования искомого объекта и его положения в пространстве, и описывается реализация СТЗ. Целью работы становится разработка прототипа СТЗ для внедрения в качестве модуля в разрабатываемый комплекс по управлению роботами – манипуляторами. В выводах представлены полученные результаты, описывающие эффективность работы системы.

Постановка задачи

Разработать прототип системы технического зрения, фиксируемой на рабочем инструменте робота-манипулятора, позволяющий осуществлять распознавание искомого объекта с камер СТЗ и определять его месторасположение в пространстве, относительно робота манипулятора для взаимодействия робота с объектом.

Для этого необходимо решить следующие задачи:

- Построить математическую модель робота с СТЗ для нахождения позиций звеньев робота-манипулятора при взаимодействии с искомым объектом,

- Рассмотреть алгоритмы компьютерного зрения для детектирования объектов и их положения в пространстве,

- Провести модификацию существующих алгоритмов при заданных ограничениях системы на вычислительные ресурсы,

- Разработать архитектуру взаимодействия подсистем СТЗ,

- Разработать прототип встраиваемой СТЗ,

- Провести анализ работы созданного прототипа на основе предложенного алгоритма.

Обзор литературы

Изучение литературных источников позволяет сделать вывод о том, что задача создания мехатронных систем на базе роботов-манипуляторов впервые была рассмотрена в 1970х годах: в 1967 году в Стэнфордском университете был создан макет робота-манипулятора, снабженный техническим зрением для отработки системы управления «глаз-рука», в 1968 году в СССР был создан телеуправляемый подводный робот с очувствленным схватом «Манта» [12].

В настоящее время активно ведутся разработки в робототехнике в областях управления с задержками [1, 4, 6], силомоментного управления [2, 5, 10], технического зрения [3, 8] и применении методов искусственного интеллекта в телеуправлении для корректировки траектории движения [6, 8]. Данные исследования показывают актуальность темы построения мехатронной системы управления роботами-манипуляторами с использованием ИИ для корректировки траектории перемещения звеньев манипулятора.

В качестве наиболее значимых работ по техническому зрению в робототехнике, проводимых в настоящее время, можно выделить следующие публикации:

1. Цикл экспериментов Контур-2, проводимый немецким космическим агентством DLR при сотрудничестве с ЦНИИ РТК [5] – описывает предоставление избыточной информации об окружающей среде робота при помощи системы захвата отслеживания перемещений VICON для управления роботом-манипулятором с минимальными задержками. В данной системе используется сканирующий подход - строится компьютерная модель рабочей сцены на основе данных, полученных с системы статично закрепленных камер с использованием технологии Motion Capture. В качестве характерных точек в данной системе

используются шары с фиксированным диаметром, что налагает ряд ограничений на подготовку и калибровку рабочего пространства и на подготовку объектов для детектирования. Данный подход представляется неприменимым в реальных условиях для не статичной сцены.

2. В исследованиях, проводимых в МГТУ им. Баумана [13] применяется проецирующий подход – используется аппарат Microsoft Kinect для построения карты глубины изображения за счет проецирования множества инфракрасных точек и считывания информации о полученных искажениях сетки на изображении. Задачи идентификации объектов решаются за счет кластеризации множества точек методами машинного обучения. Данный аппарат может быть установлен в качестве рабочего инструмента робота манипулятора для осмотра рабочей поверхности. Излучения в инфракрасном диапазоне могут помешать корректному построению облака характерных точек.
3. В исследованиях, проводимых в университете Цюриха [3] предлагается метод построения 3D сцены с одной камеры движущегося объекта. Данный подход применим к использованию на роботе-манипуляторе, но предполагает дополнительные расходы на перемещение робота для построения 3D сцены, что может быть критично в условиях ограниченности ресурсов.

Особенности приведенные в перечисленных системах налагают ряд ограничений на систему, такие как: статичность рабочей поверхности, условия освещения рабочей поверхности, затраты на перемещение робота. В данной работе рассматривается создание комплексной системы, позволяющей работать в разных условиях окружающей среды.

В работе [14] описан подход в построении отказоустойчивых систем. В данной работе при построении системы применен принцип избыточности подсистем, обеспечивающий отказоустойчивость модуля.

В издании [15] описан подход в применении встраиваемых систем в управлении оборудованием. На основе данного подхода предложена модификация, позволяющая использовать разрабатываемую СТЗ в качестве модуля для корректировки траектории движения робота.

В главе 1 предложен подход к построению траектории перемещения звеньев манипулятора, основанный на решении прямой и обратной позиционной задачи и построении параметрического уравнения движения, которые описаны в работах [16,17].

В главе 2 предложен способ идентификации искомого объекта на проекции рабочей сцены, базированный на выделении и детектировании особых точек. Механизмы детектирования особых точек описаны в работах [22, 23]. Определение расположения в пространстве найденных особых точек основано на методах, описанных в работах [18-21].

В главе 3 представлена архитектура разрабатываемой системы с использованием подходов к построению систем, описанных в работах [7, 8, 14, 15].

1. Математическая модель

Для взаимодействия манипулятора с искомым объектом необходимо найти траекторию перемещения звеньев кинематической цепи. Для этого найдем начальную и конечную точки траектории заданные в обобщенных координатах, и зададим функцию скорости на промежутке перемещения.

1.1. Модель робота-манипулятора

Необходимо получить положение рабочего инструмента n – звенного робота - манипулятора.

Пусть дан вектор обобщенных координат $q = \{q_1, q_2, \ldots, q_n\}$ звеньев манипулятора. По вектору q необходимо найти положение и ориентацию инструмента $s = f(q)$.

Пусть A_i, где $i = 1, 2, \ldots, n$ — матрицы перехода от координат $i-1$ звена к системе координат n звена, тогда матрица (1) является решением поставленной задачи в форме матрицы однородного преобразования.

$$K_n = A_1 A_2 \ldots A_n \quad (1)$$

Пусть A_i - представление Денавита-Хартенберга [17] для совмещения системы координат $O_{i-1}X_{i-1}Y_{i-1}Z_{i-1}$ с системой координат $O_iX_iY_iZ_i$. Для этого необходимо выполнить следующую последовательность операций:

1. Совершаем поворот вокруг оси $O_{i-1}Z_{i-1}$ на угол q_i (оси $O_{i-1}X_{i-1}$ и O_iX_i параллельны).
2. Осуществляем сдвиг вдоль оси $O_{i-1}Z_{i-1}$ на di (оси $O_{i-1}X_{i-1}$ и O_iX_iсовпадают).
3. Осуществляем сдвиг вдоль оси $O_{i-1}X_{i-1}$ на a_i (O_{i-1}и O_i совпадают).
4. Совершаем поворот вокруг оси $O_{i-1}X_{i-1}$ на угол b_i (системы координат $O_{i-1}X_{i-1}Y_{i-1}Z_{i-1}$ и $O_iX_iY_iZ_i$ совпадают).

$$T_i^1 = \begin{pmatrix} \cos q_i & -\sin q_i & 0 & 0 \\ \sin q_i & \cos q_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad T_i^2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$T_i^3 = \begin{pmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad T_i^4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos b_i & -\sin b_i & 0 \\ 0 & \sin b_i & \cos b_i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

$$A_i = T_i^1 T_i^2 T_i^3 T_i^4,$$

$$A_i(qi, di, ai, bi) = \begin{pmatrix} \cos q_i & -\cos b_i \sin q_i & \sin b_i \sin q_i & a_i \cos q_i \\ \sin q_i & \cos b_i \cos q_i & -\sin b_i \cos q_i & a_i \sin q_i \\ 0 & \sin b_i & \cos b_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

1.2. Модель робота-манипулятора с СТЗ

«Виртуальное звено» кинематической цепи - звено, имеющее обобщенные координаты: углы относительного поворота - α, β, w, p, r и относительное смещение - l. При задании α, β – углов отклонения от центрального вектора на изображении камеры СТЗ, l – расстояния до выбранной точки на камере, а также w, p, r – углы, задающие положение рабочего инструмента, получим виртуальное звено, характеризующее положение $n+1$-звенного манипулятора в точке взаимодействия с искомым объектом.

Таким образом, зная положение на искомом объекте множества точек, получим множество виртуальных звеньев, соответствующее множеству конфигураций $n+1$ -звенного манипулятора. При наличии способности детектирования искомого объекта производим выбор нужного $n+1$ звена из множества виртуальных звеньев, соответствующего точке взаимодействия с объектом сложной формы, и получаем координату точки на объекте в

пространстве, решая прямую позиционную задачу для $n+1$-звенного манипулятора, полученную из (1):

$$K_{n+1} = K_n * A_{n+1}, \quad (2)$$

Найдем переход от n системы координат к n+1 системе координат для взаимодействия с искомым объектом:

$$A_{n+1} = L_1 L_2,$$

1. Матрицу переноса в искомую точку – L_1 необходимо найти из матрицы $C_1 C_2 C_3$, где C_1 – матрица поворота на угол α вокруг оси X_n, C_2 – матрица поворота на угол β вокруг оси Y_n, C_3 – матрица сдвига на l вдоль оси Z_n, выбрав последний столбец, обозначающий линейный сдвиг системы координат:

$$C_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$C_2 = \begin{pmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$C_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$C_1 C_2 C_3 = \begin{pmatrix} \cos\beta & 0 & \sin\beta & l\sin\beta \\ -\sin\alpha\sin\beta & \cos\alpha & \cos\beta\sin\alpha & l\cos\beta\sin\alpha \\ -\cos\alpha\sin\beta & -\sin\alpha & \cos\alpha\cos\beta & l\cos\alpha\cos\beta \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$L_1 = \begin{pmatrix} 1 & 0 & 0 & l\sin\beta \\ 0 & 1 & 0 & l\cos\beta\sin\alpha \\ 0 & 0 & 1 & l\cos\alpha\cos\beta \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

2. Матрицу поворота рабочего инструмента к объекту для взаимодействия на углы w, p, r вокруг осей X, Y, Z соответственно найдем следующим образом:

$$L_2 = C_4 C_5 C_6,$$

$$C_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos w & -\sin w & 0 \\ 0 & \sin w & \cos w & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$C_5 = \begin{pmatrix} \cos p & 0 & \sin p & 0 \\ 0 & 1 & 0 & 0 \\ -\sin p & 0 & \cos p & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$C_6 = \begin{pmatrix} \cos r & -\sin r & 0 & 0 \\ \sin r & \cos r & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$L_2 = \begin{pmatrix} L_2^1 & L_2^2 & L_2^3 & 0 \\ L_2^4 & L_2^5 & L_2^6 & 0 \\ L_2^7 & L_2^8 & L_2^9 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$L_2^1 = \cos p \cos r,$$

$$L_2^2 = -\cos p \sin r,$$

$$L_2^3 = \sin p,$$

$$L_2^4 = \cos w \sin r + \cos r \sin p \sin w,$$

$$L_2^5 = \cos r \cos w - \sin p \sin r \sin w,$$

$$L_2^6 = -\cos p \sin w,$$
$$L_2^7 = -\cos r \cos w \sin p + \sin r \sin w,$$
$$L_2^8 = \cos w \sin p \sin r + \cos r + \sin w,$$
$$L_2^9 = \cos p \cos w,$$

$$A_{n+1} = \begin{pmatrix} L_2^1 & L_2^2 & L_2^3 & l \sin \beta \\ L_2^4 & L_2^5 & L_2^6 & l \cos \beta \sin \alpha \\ L_2^7 & L_2^8 & L_2^9 & l \cos \alpha \cos \beta \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$A_{n+1} = \begin{pmatrix} \cos p \cos r & -\cos p \sin r & \sin p & l \sin \beta \\ \cos w \sin r + \cos r \sin p \sin w & \cos r \cos w - \sin p \sin r \sin w & -\cos p \sin w & l \cos \beta \sin \alpha \\ -\cos r \cos w \sin p + \sin r \sin w & \cos w \sin p \sin r + \cos r + \sin w & \cos p \cos w & l \cos \alpha \cos \beta \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (4)$$

Таким образом, для объектов сложной формы необходимо найти точку для взаимодействия, состоящую из 6 обобщенных координат, задающих матрицу A_{n+1}. Для этого надо найти точку, лежащую на искомом объекте для взаимодействия, используя предложенный в главе 2 алгоритм работы СТЗ.

1.3. Планирование траектории движения звеньев манипулятора

Для достижения позиции взаимодействия с искомой точкой на объекте рабочим инструментом робота-манипулятора необходимо решить обратную позиционную задачу для -звенного манипулятора в K_{n+1}. Решим данную задачу методом обратных преобразований [17]:

$$K_{n+1} = K'_n = A_1 A_2 \dots A_n,$$

$K'_n = K_n^{0*}$ - желаемое положение и ориентация манипулятора. (5)

Шаг 1.: Умножаем слева (5) на A_1^{-1}:

$$A_1^{-1} K_n^{0*}(q_1) = A_2 A_3 \dots A_n(q_2, q_3, \dots, q_n),$$

Ищем решение - q_1'в матричном равенстве и подставляем в

$$A_1^{-1}K_n^{0*}(q_1) = K_n^1(q_2, q_3, \dots, q_n) = K_n^{1*}.$$

Шаг i. ($i < n$):

$$K_n^{i-1*} = K_n^{i-1}(q_i, \dots, q_n), \quad (6)$$

Умножаем (2) на A_i^{-1}:

$$A_i^{-1}K_n^{i-1*}(q_i) = A_{i+1} \dots A_n(q_{i+1}, \dots, q_n),$$

Ищем решение - q_i' в матричном равенстве и подставляем в

$$A_i^{-1}K_n^{i-1*}(q_i) = K_n^i(q_{i+1}, \dots, q_n) = K_n^{i*}.$$

Шаг n-1.:

$$K_n^{n-2*} = K_n^{n-2}(q_{n-1}, q_n), \quad (7)$$

Умножаем (3) на A_{n-1}^{-1}:

$$A_{n-1}^{-1}K_n^{n-2*}(q_{n-1}) = A_n(\, q_n),$$

Ищем решение - q_{n-1}' в матричном равенстве и подставляем в

$$A_{n-1}^{-1}K_n^{n-2*}(q_{n-1}) = K_n^{n-1}(q_n),$$

Находим решение q_n'.

В результате шагов $\overline{1, n-1}$ метода обратных преобразований получим вектор обобщенных координат:

$$q' = \{q_1', q_2', \dots, q_n'\}.$$

В случае невозможности поиска решения методом обратного преобразования предлагается использовать метод Ньютона [17].

После получения обобщенных координат q' для n-звенного манипулятора осуществляем перемещение из точки q в точку q' за промежуток времени $[0, t]$:

$$q(t) = q + \lambda(t)(q' - q),$$

$$\lambda(t_0) = 0,\ \lambda(t_1) = 1, \tag{8}$$

$$q(t_0) = q,\ q(t_1) = q', \tag{9}$$

$$\lambda(t) = \frac{t - t_0}{t_1 - t_0}, \tag{10}$$

$$q(t) = q + \frac{q' - q}{t_1 - t_0}(t - t_0). \tag{11}$$

При выполнении условия (8) справедливо условие (9), при этом если условие (10) не выполняется, то движение осуществляется с переменной скоростью, иначе получаем параметрическое движение с фиксированной скоростью (11).

2. Алгоритм работы системы технического зрения

Для взаимодействия робота-манипулятора с объектом сложной формы, необходимо определить координаты особой точки искомого объекта x, y, z и углы поворота w, p, r, а также множество допустимых положений рабочего инструмента удобных для взаимодействия с объектом.

Данная задача представима в виде 2 подзадач:

1. Поиск обобщенных координат α, β, l и соответствующих им Декартовых координат x, y, z особой точки, расположенной на искомом объекте.
2. Поиск углов поворота объекта w, p, r относительно СТЗ, зафиксированной на рабочем инструменте манипулятора.

2.1. Поиск карты глубины на изображении

Современные ПЗС-камеры, используемые в СТЗ описываются с помощью модели проективной камеры (см. рис. 1) [18].

Вектор OC' перпендикулярен плоскости cx_py_p, задаваемой положением матрицы камеры $ABCD$ в пространстве $Oxyz$, проходит через c – центральную точку матрицы. Зная фокусное расстояние камеры – f равное длине Oc, координату точки m на плоскости cx_py_p- $m = \{x_p^m, y_p^m\}$ и $p = \{p_x, p_y\}$ – физический размер пикселя по осям x_p и y_p соответственно, определим углы поворота α и β для нахождения вектора OM:

$$\alpha = \mathrm{a}tan2\,\frac{y_p^m * p_y}{f} \tag{12}$$

$$\beta = atan2\,\frac{x_p^m * p_x}{f} \tag{13}$$

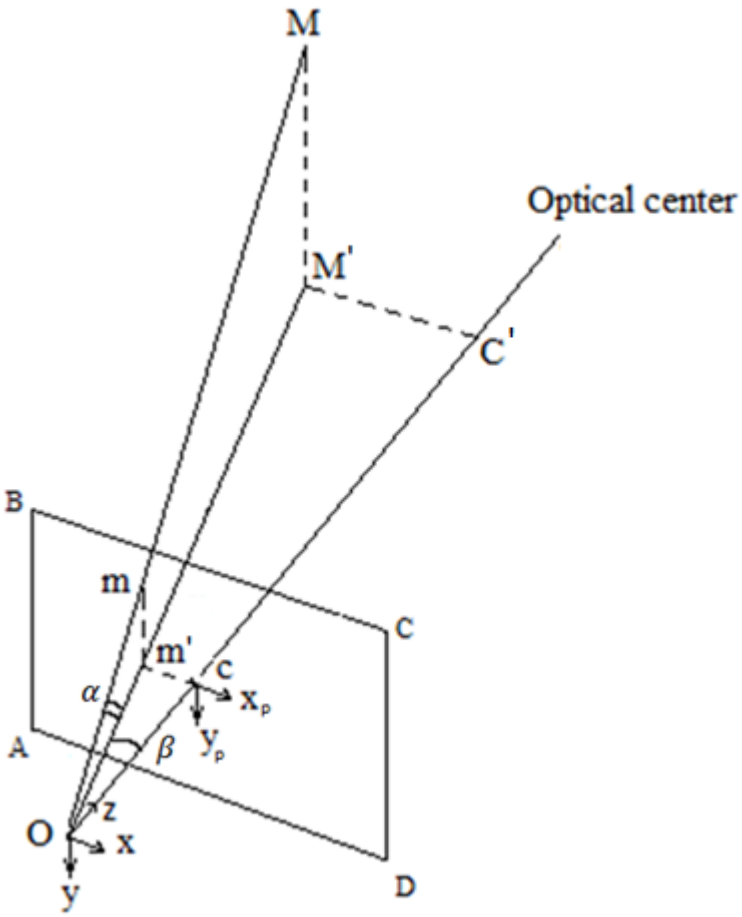

Рис.1 Модель проективной камеры

В поставленной задаче координаты x, y, z могут быть найдены по углам поворота α и β вокруг осей Ox и Oy соответственно, а также длине отрезка l по формуле (4).

2.1.1. Сканирующий подход

Сканирующий подход применяется при использовании двух кадров со сдвигом. Данный подход может быть применен при наличии двух откалиброванных камер или камеры с возможностью изменения ее месторасположения в пространстве между снимками на фиксированное расстояние.

В данном подходе, в общем случае, для каждого пикселя левого кадра с координатами x_0, y_0, выполняем поиск пикселя на правом кадре.

Предполагается, что пиксель на правом изображении должен иметь координаты $x_0 - d, y_0$, где d — величина смещения. Поиск соответствующего пикселя выполним путем вычисления максимума функции отклика, в качестве которой может выступать корреляция окрестностей пикселей (см. рис. 2) [19]. Для левого кадра СТЗ зависимость может быть вычислена по формуле:

$$\frac{T-d}{l-f} = \frac{T}{l} \rightarrow l = \frac{fT}{d}, \text{ где } l \text{ – расстояние до искомой точки} \qquad (14)$$

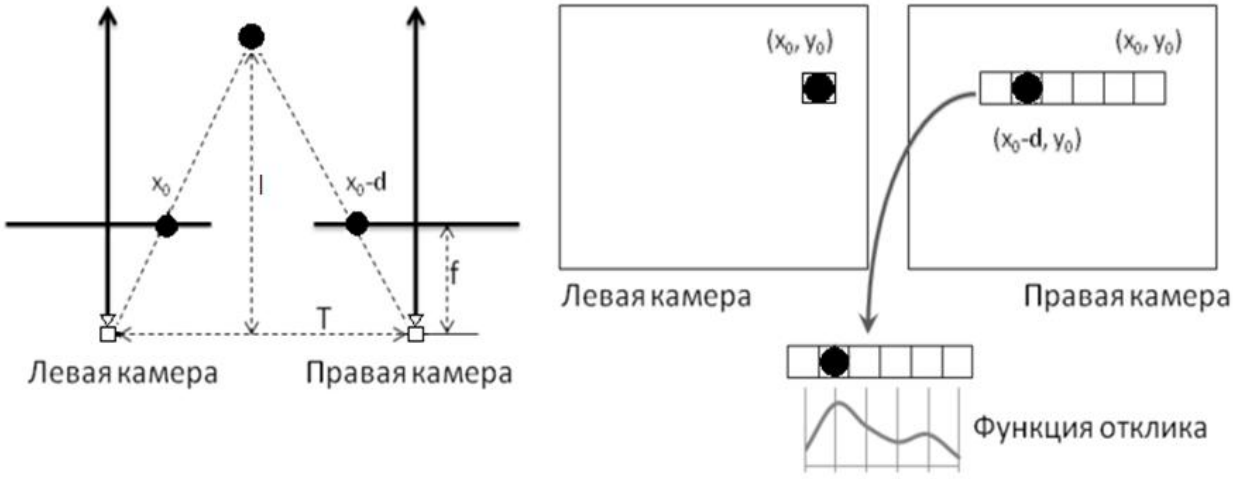

Рис. 2 Вычисление карты глубины изображения при помощи 2 изображений

При реализации необходимо провести расчет матриц стерео калибровки камер для выполнения аффинных преобразований проекций изображений рабочей сцены для минимизации ошибок. Для реализации построения карты глубины изображения был выбран алгоритм Block Matching [20] на одноканальном изображении, предварительно обработанном детектором границ для увеличения точности построения карты глубины в окрестностях особых точек, характеризующих граничный переход между объектами на рабочей сцене. В результате получим карту глубины изображения: $L = l_{i,j}$.

2.1.2. Проецирующий подход

При использовании одной камеры и лазерного целеуказателя, определим расстояния до точек, расположенных на рабочей сцене [21].

Рассмотрим случай на плоскости (см. рис. 3): Камера и лазерный модуль зафиксированы на расстоянии T друг от друга. Камера имеет угол обзора по оси Oy равный 2η, лазерный целеуказатель зафиксирован под углом θ к оси Oz. Определим l – расстояние до искомой точки P на рабочей сцене. Для этого определим lA, lB- расстояния до точек, фиксируемых лазерным целеуказателем в краевых положениях на матрице ПЗС-камеры:

$$lB = \frac{T}{\tan\theta - \tan\eta}$$

$$lA = \frac{T}{\tan\theta + \tan\eta}$$

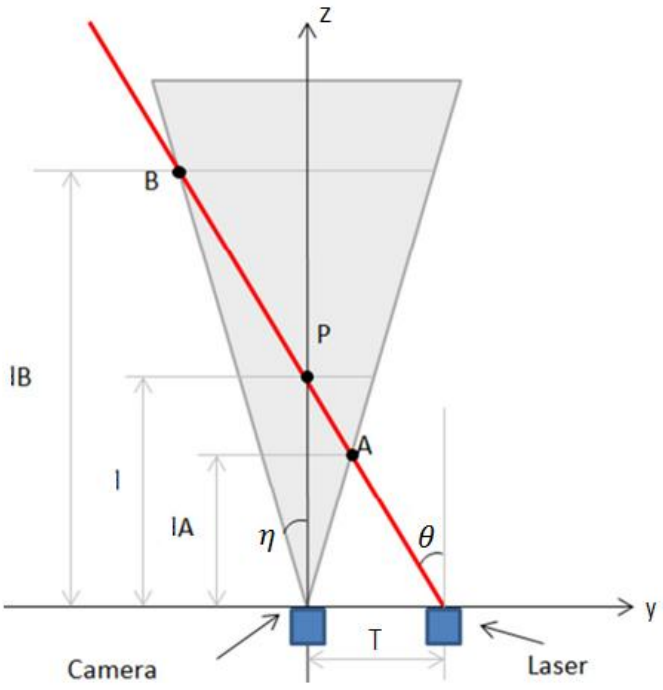

Рис. 3. Вычисление карты глубины изображения при помощи камеры и лазерного целеуказателя

$$p = \frac{n-k}{n} = \frac{AP}{AB} = \frac{l - lA}{lB - lA},$$

где k-номер строки в матрице, с найденной точкой, проецированной с лазерного модуля, полученной на изображении, n – количество строк в матрице.

$$l = \frac{T}{(\tan\theta + \tan\eta) - (2p\tan\eta)} \quad (15)$$

В трехмерном случае, с использованием целеуказателя – линии, установленного на сервомотор для получения каждого угла θ, получим точки в каждом столбце матрицы, полученной с изображения рабочей сцены. Таким образом, заполнив матрицу расстояний до каждого пикселя рабочей сцены с использованием сервопривода, получим карту глубины изображения: $L = l_{i,j}$.

2.1.3. Построение частичной карты глубины

При построении карты глубины изображения, для уменьшения затрат ресурсов на вычисление расстояний до каждого пикселя, спроецированного на матрицу ПЗС-камеры со сцены, возможно строить неполную карту глубины изображения.

При использовании метода Block Matching воспользуемся построением карты глубины для блоков, центральный пиксель которых соответствует особой точке, найденной на изображении. Выбор особых точек зависит от предлагаемого далее метода поиска углов вращения искомого объекта.

$$L' \subset L = l_{i,j}$$

2.2. Поиск углов вращения искомого объекта

Для получения углов вращения объекта важной задачей становится поиск и выбор множества реперных точек искомого объекта. Для взаимодействия с объектом сложной формы необходимо ввести унифицированный классификатор, который позволит однозначно определить на любом кадре СТЗ его двумерную проекцию из пространства R^3. Для этого рассмотрим создание массива шаблонов для сопоставления с рабочей сценой, а также способы детектирования особых точек.

2.2.1. Поиск углов на СТЗ с использованием массива шаблонов

Рассмотрим двумерную матрицу изображений $Mat_{N,N}$ составленную из изображений, полученных с ПЗС-камеры, расположенной на поверхности сферы с радиусом R (см. рис. 4). Оптический центр камеры пересекает центр полусферы с расположенным в нем объекте. Таким образом, матрица камеры $ABCD$ лежит в касательной поверхности к сфере в точке расположения камеры. Снимки с камеры сделаны с шагом $gr = 360/N°$. От выбора N зависит точность модели.

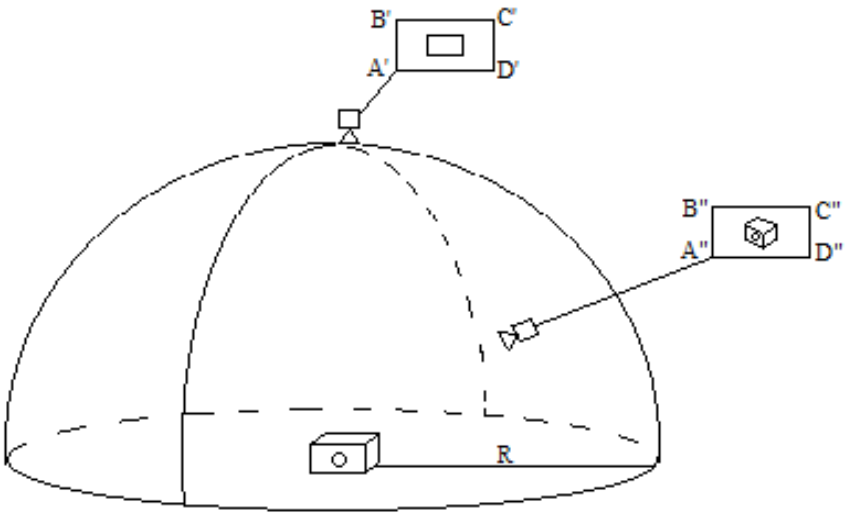

Рис. 4. Захват изображения объекта с СТЗ

Таким образом, если принять систему координат объекта, где центр объекта расположен в начале системы координат, то положения камеры для заполнения матрицы Mat можно рассчитать следующим образом: Углы вращения получим с помощью орта обратного вектора нормали $(\frac{-x_{n+1}}{\sqrt{(x_{n+1})^2+(y_{n+1})^2+(z_{n+1})^2}}, \frac{-y_{n+1}}{\sqrt{(x_{n+1})^2+(y_{n+1})^2+(z_{n+1})^2}}, \frac{-z_{n+1}}{\sqrt{(x_{n+1})^2+(y_{n+1})^2+(z_{n+1})^2}})$ при переходе от системы координат объекта к декартовым координатам робота.

Положение камеры в пространстве может быть рассчитано по формуле:

$$\begin{cases} x_{n+1} = R\ sin(w)\cos(p) \\ y_{n+1} = R\ sin(w)\sin(p) \\ z_{n+1} = R\ sin(w) \end{cases} \quad (16)$$

Для избегания уплотнения снимков в сферических координатах для заполнения Mat, для заданной точности gr по формуле расчёта длины хорд, расположенных под углом γ, параллельно к диаметру, $S = R\ cos\gamma$, где S – длина хорды, R – радиус окружности, получим $N * cos\gamma = K$, где N – размерность Mat, K – количество элементов в строке, соответствующей индексу $w = \alpha$.

Для упрощения модели предлагается в случае уплотнения шага снимков в 2 раза, соответствующие элементы Mat оставлять пустыми.

- $w = [0°, 60°]$ - Заполнение каждого элемента строки Mat.
- $w = (60°, 75°]$ - Заполнение каждого второго элемента строки Mat.
- $w = (75°, 83°]$ - Заполнение каждого четвертого элемента строки Mat.
- $w = (83°, 87°]$ - Заполнение каждого восьмого элемента строки Mat.

Таким образом, можно получить двумерный массив Mat, состоящий из бинарных изображений, с выделением особых точек, полученных на видимой сцене с камеры СТЗ, для которой углы w, p соответствуют строкам и

столбцам. Для заполнения Mat также возможно использование рендеринга CAD-модели объекта с обработкой детектором особых точек.

2.2.2. Детектирование границ

В качестве классификатора для комбинированной СТЗ предлагается использование множества характерных точек, принадлежащих границе проекции объекта. В общем случае, в методах, используемых для выделения границ, используется ступенчатый край, сглаженный функцией ошибки (функция Гаусса). Одномерное изображение f, имеющее один край в точке $x = 0$ в общем случае может быть смоделировано следующим образом:

$$f(x) = \frac{I_r - I_l}{2}\left(\mathrm{erf}\left(\frac{x}{\sqrt{2}\sigma}\right) + 1\right) + I_l \text{ , где } \mathrm{erf}(u) = \frac{2}{\sqrt{\pi}}\int_0^u e^{-t^2}\,dt,$$

$$I_l = \lim_{n\to-\infty} f(x), \quad I_r = \lim_{n\to\infty} f(x) \qquad (17)$$

Использование преобразования изображения, отображающего границу объекта, позволяет создать отпечаток объекта для заданного угла вращения, с точностью до симметрии граней.

В качестве детектора границ рассмотрим детектор Канни [22]. Для преобразования изображения выполняем следующие этапы алгоритма:

1. Преобразуем входное трехканальное изображение в одноканальное изображение в градациях серого цвета для уменьшения вычислительных затрат.
2. Производим сглаживание за счет размытия изображения для удаления шумов с использованием фильтра Гаусса (17).
3. Выделяем границы в окрестностях локальных максимумов градиентов при помощи оператора Собеля с использованием ядер фильтров:

$$MGx = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}, \qquad MGy = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

4. Подавляем немаксимумы. Только локальные максимумы отмечаем как границы вдоль направления вектора градиента.

5. Производим двойную пороговую фильтрацию для определения потенциальных границ в качестве порогов.
6. Производим трассировку области неоднозначности. Выделяем точки, не отнесенные к границам в отдельную группу в случае наличия соседних точек в восьми направлениях, иначе – удаляем точку.

2.2.3. Детектирование углов

Любой трехмерный объект может быть представлен в виде полигональной сетки, вершинами полигонов которой являются особые точки. Выберем в качестве особых точек углы - точки, формируемые из двух или более пересекающихся граней трехмерного объекта [23].

В общем случае объект может быть представлен в виде множества векторов $V = \{\overrightarrow{v_{i,j}}\},\ i \neq j,\ i,j = \overline{1,n}$ соединяющих 2 особые точки, задающих структуру объекта.

Рассмотрим представление объекта в виде неориентированого графа:

$G = (X, U)$, где $X -$ множество вершин, $U -$ множество ребер.

В качестве вершин графа X выберем особые точки, в качестве ребер U выберем длины векторов $\overrightarrow{v_{i,j}}$, соответствующие паре точек x_i, x_j.

$$u_{i,j} \in U,\ \ u_{i,j} = \left|\overrightarrow{v_{i,j}}\right|.$$

Таким образом, можно получить описание 3D объекта в виде связного графа G с точностью до углов поворотов и деформаций структуры объекта.

Для нахождения G необходимо выделить особые точки X и найти длины векторов U в пространстве.

В случае работы с изображением, получаемым с матрицы камеры СТЗ, при работе с выпуклым телом невозможно определить расположение всех

особых точек, лежащих на поверхности искомого объекта. Особые точки X будут спроецированы на ПЗС-матрицу камеры СТЗ: $X \to P$. Для определения координат особых точек в пространстве воспользуемся формулой (4) нахождения координат точек.

Найдем U по формуле: $u_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$.

Найдем проекции множества особых точек на изображении:

$$dst = D_x^2 D_{yy} + D_y^2 D_{xx} - 2D_x D_y D_{xy},$$

где D_x, D_y - производные первого порядка матрицы исходного изображения, D_{xx}, D_{yy} - производные второго порядка, D_{xy} - смешанные производные.

Для уменьшения затрат на вычисления во время сопоставления шаблона с изображением рабочей сцены, отсортируем наиболее явные углы на изображениях по мере качества каждой особой точки исходного изображения, полученной с помощью вычисления минимальных собственных значений матрицы градиента.

Для отсеивания шумов не рассматриваем углы с минимальным собственным значением меньше, чем заданный параметр. Оставшиеся углы сортируем по мере качества в порядке убывания и сохраняем их матричные координаты в динамический массив.

2.3. Алгоритм работы СТЗ

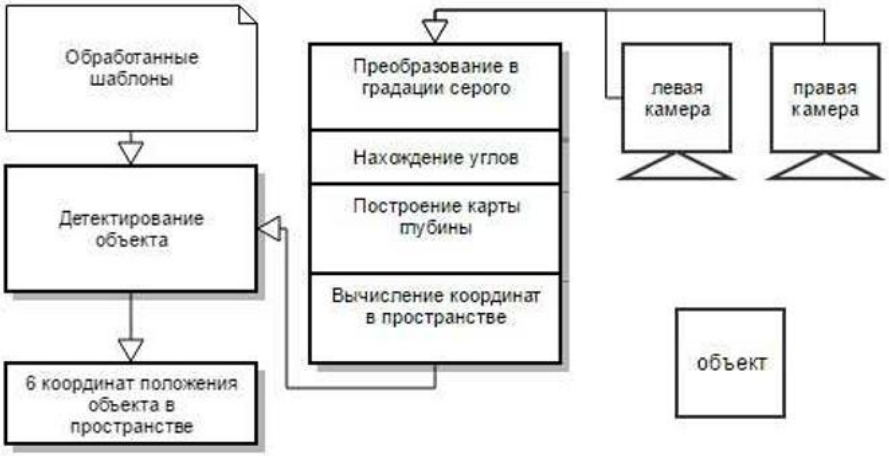

Рис. 5. Алгоритм работы системы технического зрения

Опишем общий алгоритм нахождения 6 координат с использованием системы технического зрения:

1. Получаем массив шаблонов объекта в виде матрицы изображений Mat с камер СТЗ в процессе обучения.

2. Обрабатываем каждый шаблон алгоритмом предобработки:

2.1. Изображение помещаем в структуру данных «матрица» и преобразуем в градации серого.

2.2. Вычисляем матрицу особых точек P_{templ}.

2.3. Строим карту глубины изображения $L = l_{i,j}$.

2.4. Получаем G_{templ}: вычисляем X_{templ} и длины векторов U_{templ}.

2.5. Сохраняем градусные меры углов между вектором, построенным от первой особой точки вертикально вверх, и другими векторами, построенными от выбранного угла к остальным особым точкам из массива для определения градусной меры поворота относительно шаблона (для определения обобщенной координаты r).

3. Получаем массив шаблонов - предобработанных изображений с построением G_{templ}.

4. Осуществляем поиск среди найденных на изображении рабочей сцены углов путем поэлементного сравнения U_{templ} и U_{scene}, где U_{scene} — длины векторов на рабочей сцене, полученные из X_{scene} – матрица особых точек в R^3, полученная из матрицы P_{scene} на изображении с камеры. При совпадении, вычисляем длины векторов от найденного угла до остальных особых точек на изображении сцены и сравниваем полученные данные с предварительно рассчитанными длинами на основе шаблонов. Таким образом осуществляем поиск подграфа графа G_{templ} на основе трехмерных координат точек X_{scene}.

5. При допустимом количестве совпадений длин векторов для соответствующих вершин графа, считаем, что объект находится на сцене. При наличии объекта производим вычисление обобщенной координаты r детектируемого объекта относительно шаблона.

3. Практическая реализация

На основе предложенного подхода к управлению взаимодействием робота-манипулятора с искомым объектом на рабочей сцене, в данной главе рассмотрим построение прототипа системы технического зрения, реализующего описанные методы.

3.1. Описание архитектуры системы

Для реализации СТЗ опишем архитектуру взаимодействия компонент разрабатываемой мехатронной системы по управлению роботами (см. рис. 6)

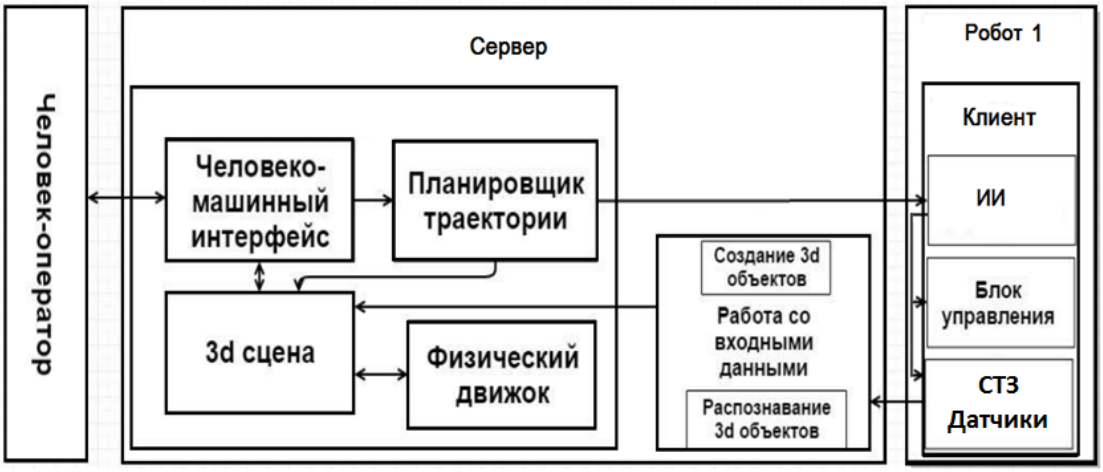

Рис. 6. Описание архитектуры мехатронной системы

Человек-оператор взаимодействует с серверным приложением за счет передачи высокоуровневых команд через человеко-машинный интерфейс. Команды из планировщика траектории передаются в блок ИИ встроенной системы установленной на роботе. С робота по обратной связи поступает информация с внутренних датчиков о его текущем состоянии и с внешних датчиков с данными об окружающей среде в блок обработки входных данных, где происходит добавление найденных объектов на 3D сцену, а также обновление данных о состоянии робота. Расчет взаимодействий между объектами на 3D сцене моделируется в физическом движке.

Таким образом, мехатронная система представляет из себя клиент-серверное приложение с множеством роботов – клиентов, которые обрабатывают поступившую информацию с сервера и с датчиков в блоке искусственного интеллекта, распознают объекты для взаимодействия и действия, которые необходимо совершить над объектами, планируют траекторию движения с учетом записанных директив и передают низкоуровневые команды для выполнения требуемых действий в блок управления (см. рис. 7). Поиск объектов на рабочей сцене для дальнейшего взаимодействия с роботом производится в системе технического зрения.

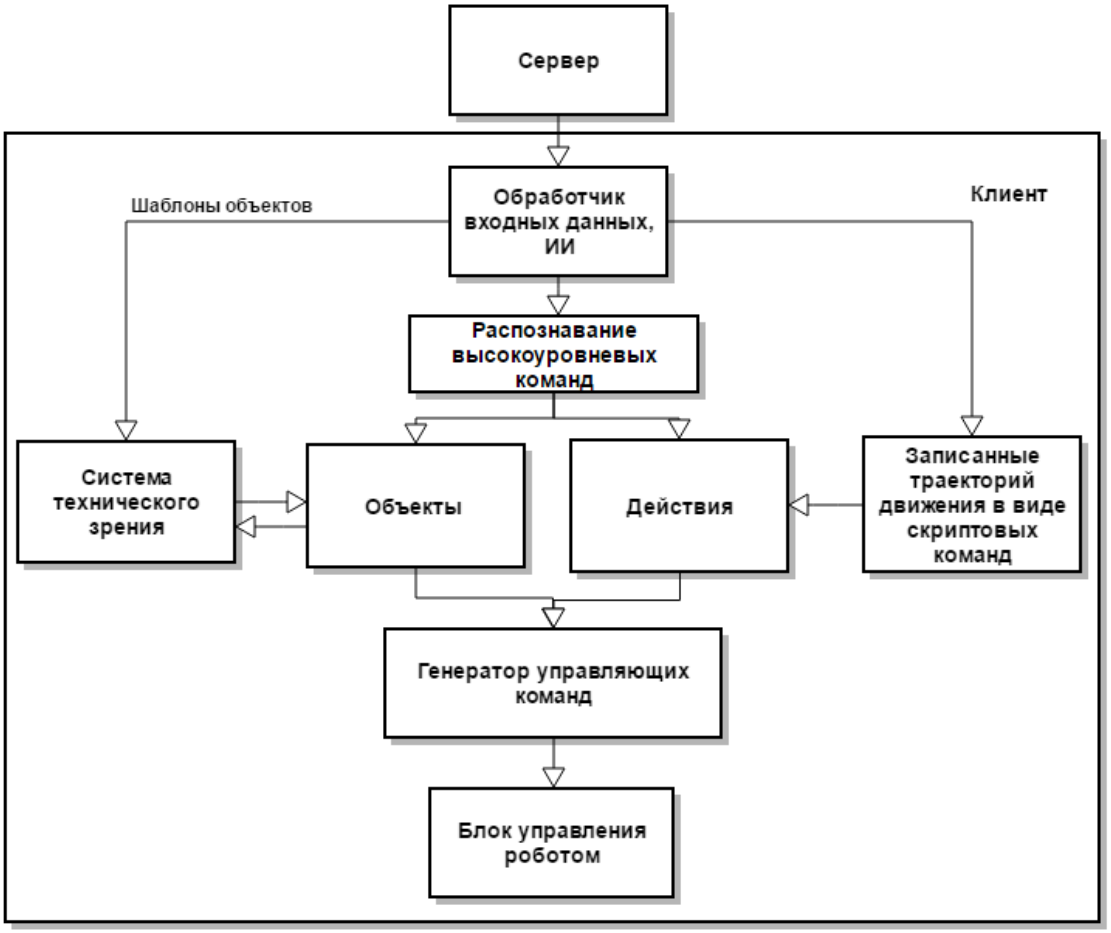

Рис. 7. Описание архитектуры клиентского приложения

Система технического зрения принимает на вход обработанные шаблоны объектов и команды на поиск объектов на рабочей сцене. Шаблоны хранятся в базе шаблонов и передаются в блок поиска объекта по шаблонам по уникальному идентификатору объекта. При получении команды на поиск

объекта на рабочей сцене происходит выбор подсистемы для съемки изображения и построения карты глубины изображения в соответствии с требованиями оператора, условиями на рабочей сцене и текущим состоянием модулей системы. На полученной проекции рабочей сцены осуществляется поиск особых точек и, с использованием карты глубины, находятся координаты выделенных особых точек в пространстве. В блоке поиска объектов по шаблонам происходит сопоставление особых точек изображения на основе представления структуры объектов в виде связных графов. После идентификации наиболее вероятного положения объекта на изображении рабочей сцены определяются 6 обобщенных координат объекта и передаются в блок «Объекты», содержащий описание всех объектов на рабочей сцене (см. рис. 8).

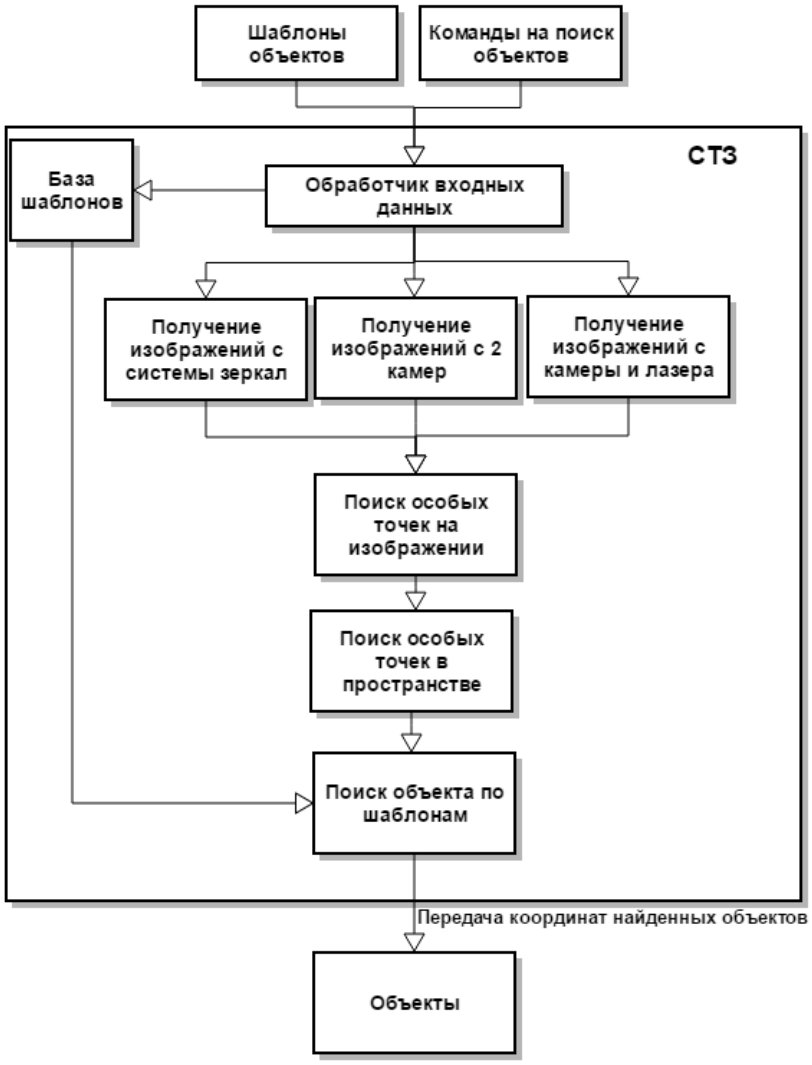

Рис. 8. Описание архитектуры СТЗ

3.2. Выбор оборудования для прототипа

В качестве оборудования для реализации стенда с прототипом отказоустойчивой СТЗ выбрано следующее:

1. Промышленный робот-манипулятор Fanuc M-20iA (см. рис. 9).

Рис.9. Промышленный робот-манипулятор Fanuc M-20iA

2. Микрокомпьютер Raspberry PI 3 с платой расширения для 4 камер (см. рис. 10).

Рис.10. Raspberry PI 3 с платой расширения

3. Система из 2 камер без ИК фильтра с ИК подсветкой (см. рис. 11).

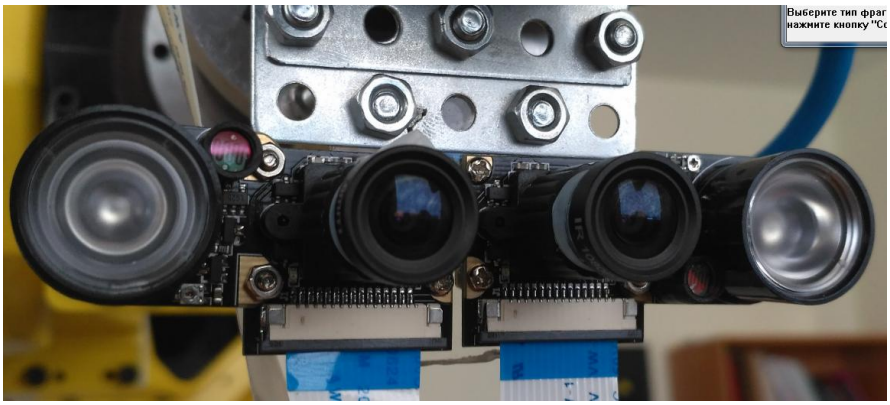

Рис.11. Система из 2 камер с подсветкой

4. Система зеркал с камерой (см. рис. 12).

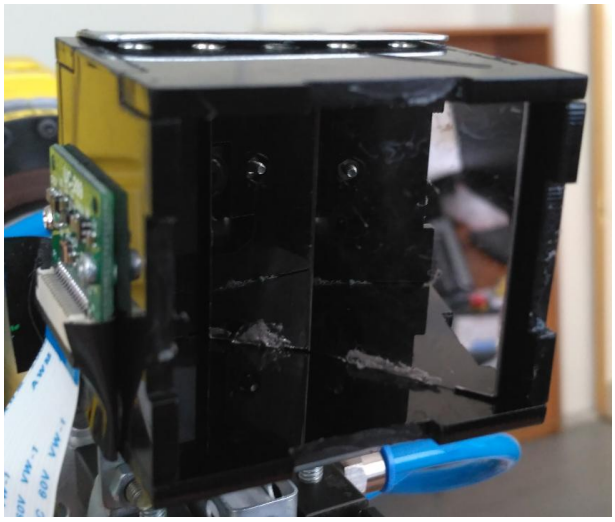

Рис.12. Система зеркал с камерой

Данная система зеркал была реализована с использованием оборудования ресурсного центра Научного парка СПбГУ «Инновационные технологии композитных наноматериалов» [24] по схеме, предложенной в проекте [25].

5. Система из камеры с аппаратным синим световым фильтром и лазерного модуля (5 mw) с проекцией линии зафиксированного на сервоприводе (см. рис. 13).

Рис.13. Система из камеры с синим фильтром и лазерного целеуказателя с сервоприводом

Схема подключения компонент прототипа системы технического зрения продемонстрирована на рисунке 14. На рисунке 15 продемонстрирован скомплектованный прототип.

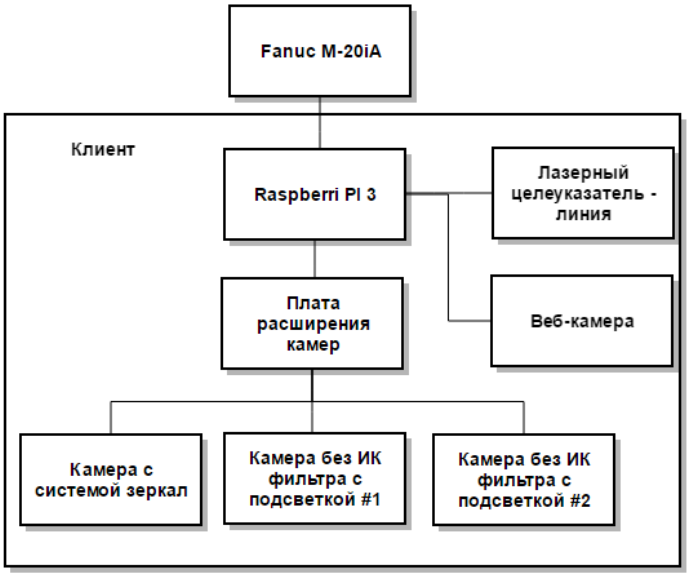

Рис.14. Схема подключения компонент прототипа

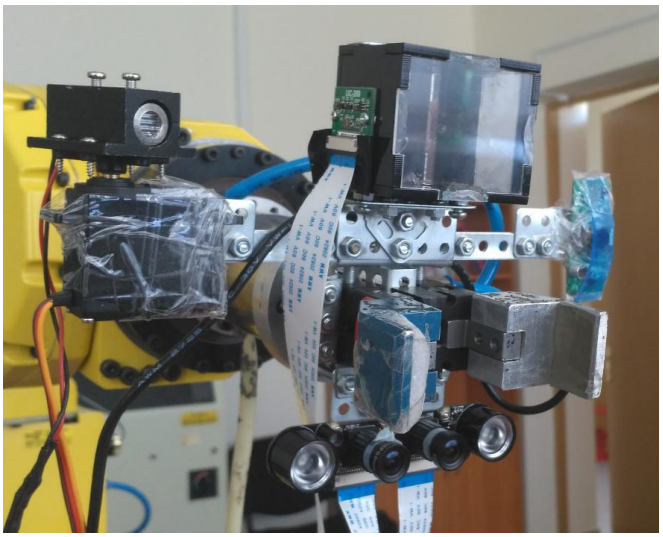

Рис.15. Прототип системы технического зрения

3.3. Анализ работы созданного прототипа

Алгоритмы, описанные в главе 2, реализованы на Raspberry PI 3 с использованием одного потока с максимальной частотой ядра 1 ГГц, а также на компьютере с CPU Intel Core i5 3210m с тактовой частотой ядра 2,5 ГГц.

3.3.1. Построение карты глубины изображения с использованием двух камер

В случае использования платы расширения для Raspberry PI 3 для создания снимков с двух камер наблюдается задержка на переключение между камерами для создания снимка сцены. Камеры без ИК фильтра с ИК подсветкой позволяют получать изображения без теней, а также изображения в условиях недостаточного освещения рабочей сцены. На рисунке 16 приведены результаты работы алгоритма построения карты глубины изображения. На рисунке 17 продемонстрирован результат построения частичной карты глубины изображения. На рисунке 18 представлена таблица с указанием времени работы этапов построения карты глубины изображения для случаев полной и частичной карты глубины изображения. На рисунке 19 продемонстрирована зависимость абсолютной максимальной ошибки в измерении карты глубины изображения от расстояния.

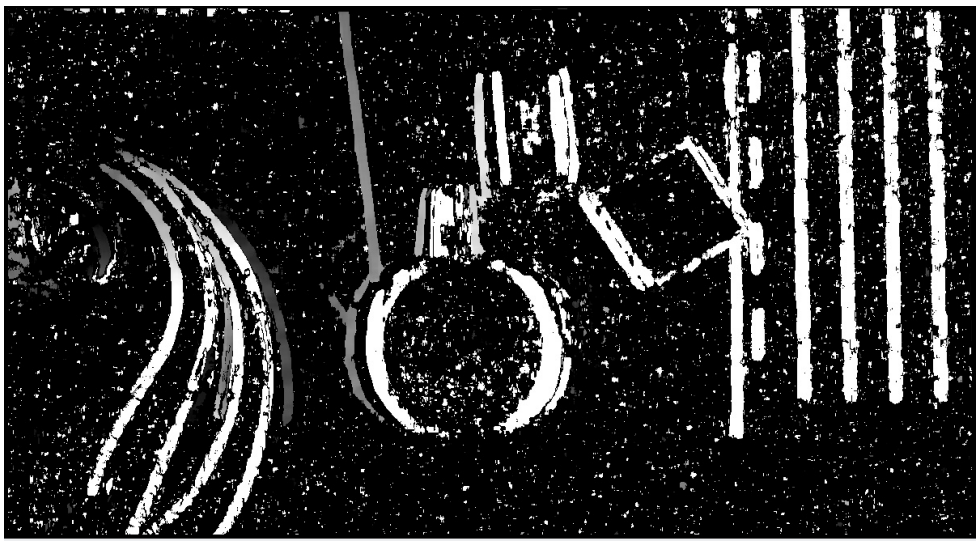

Рис.16. Построение полной карты глубины изображения

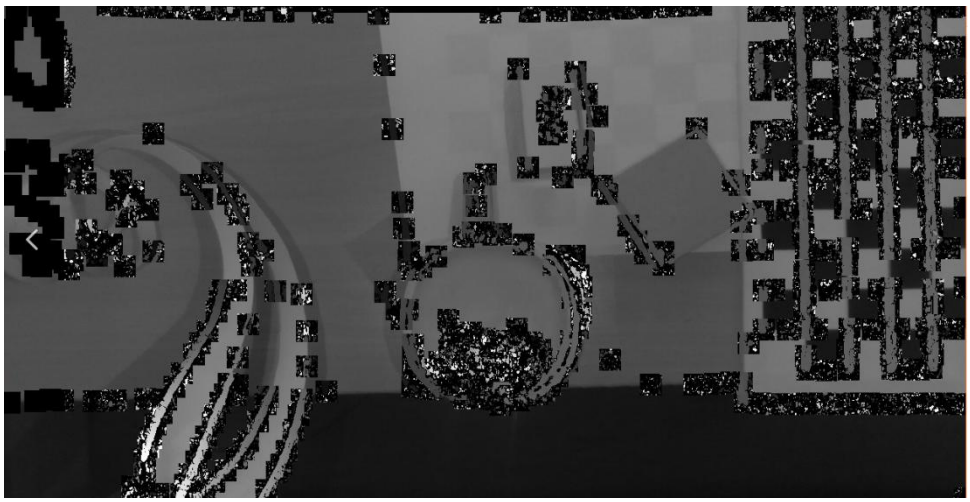

Рис.17. Построение частичной карты глубины изображения

Оборудование	Реализация	Среднее время работы алгоритма за 10 запусков (мс)
Raspberry PI 3	Съемка кадров	1003
Raspberry PI 3	Ректификация изображения	183
Intel Core i5 3210m	Ректификация изображения	103
Raspberry PI 3	Построение полной карты глубины (Python + NumPY)	1021
Intel Core i5 3210m	Построение полной карты глубины (C++)	816
Raspberry PI 3	Построение частичной карты глубины (Python + NumPY)	893
Intel Core i5 3210m	Построение частичной карты глубины (C++)	683

Рис.18. Таблица с описанием результатов работы

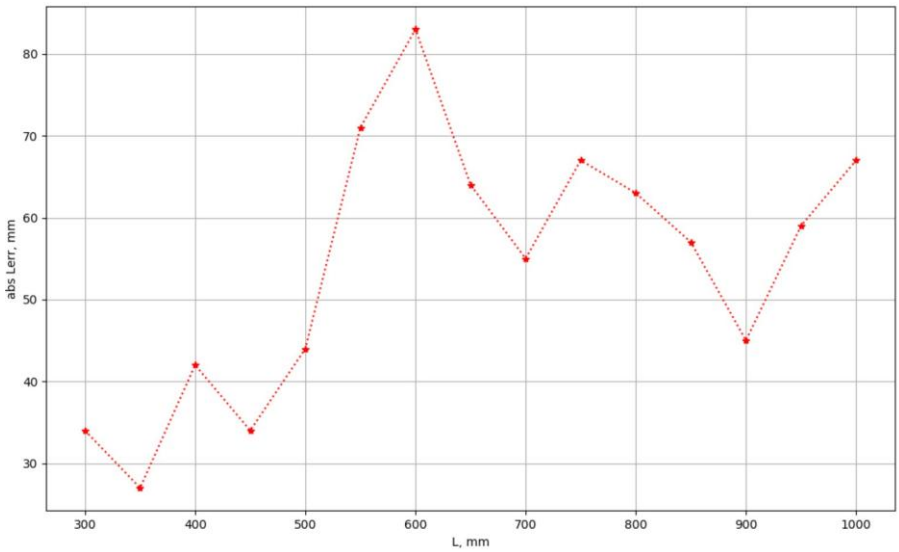

Рис.19. Зависимость ошибки в измерении карты глубины изображения от расстояния

Данная погрешность наблюдается вследствие проведения некачественной калибровки стереопары для получения матриц аффинного преобразования для использования алгоритма поиска диспаритета на блоках, полученных с изображений камер.

3.3.2. Построение карты глубины изображения с использованием камеры и системы зеркал

При использовании системы зеркал производится обрезка изображения для создания стереопары, и, таким образом, уменьшается угол обзора камеры. Использование одной камеры на плате расширения дает отсутствие задержки при получении двух снимков. Качество отражающей поверхности в оптической установке значительно влияет на построение карты глубины изображения. На рисунке 20 приведены результаты работы алгоритма построения полной карты глубины изображения. На рисунке 21 продемонстрирован результат построения частичной карты глубины изображения. На рисунке 22 представлена таблица с указанием времени работы этапов построения полной карты глубины изображения для случаев построения полной и частичной карты глубины изображения. На рисунке 23 продемонстрирована зависимость абсолютной максимальной ошибки в измерении карты глубины изображения от расстояния.

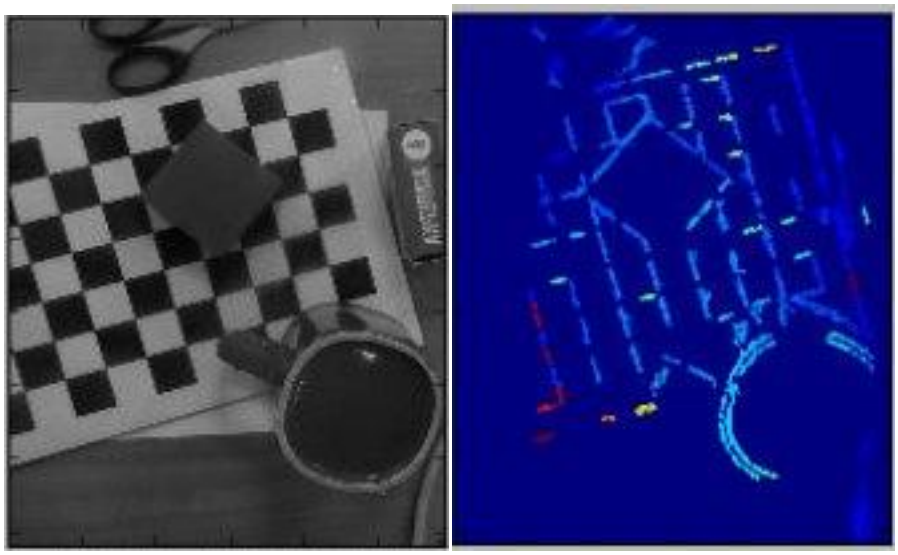

Рис.20. Построение полной карты глубины изображения

Рис.21. Построение частичной карты глубины изображения

Оборудование	Реализация	Среднее время работы алгоритма за 10 запусков (мс)
Raspberry PI 3	Съемка кадров	463
Raspberry PI 3	Ректификация изображения	114
Intel Core i5 3210m	Ректификация изображения	68
Raspberry PI 3	Построение полной карты глубины (Python + NumPY)	718
Intel Core i5 3210m	Построение полной карты глубины (C++)	513
Raspberry PI 3	Построение частичной карты глубины (Python + NumPY)	531
Intel Core i5 3210m	Построение частичной карты глубины (C++)	412

Рис.22. Таблица с описанием результатов работы

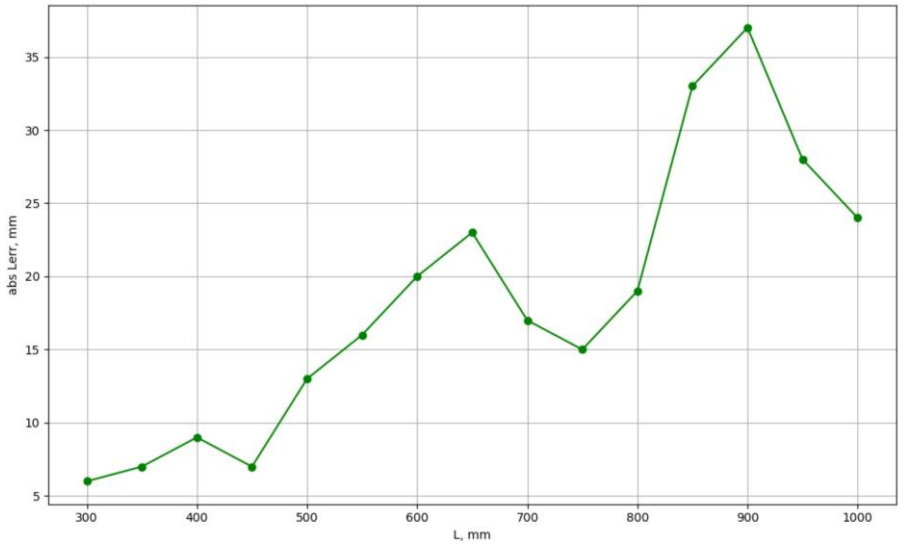

Рис.23. Зависимость ошибки в измерении карты глубины изображения от расстояния

3.3.3. Построение карты глубины изображения с использованием лазерного модуля с проекцией линии.

При использовании лазерного модуля возникает проблема детектирования точек на изображении. В работе [21] предложено использование аппаратного фильтра синего цвета для минимизации средней интенсивности красного цвета на изображении. Далее применяется пороговый фильтр канала красного цвета для нахождения пиксельных координат проецируемой линии с лазерного модуля на изображении. В случае работы со статичной сценой становится возможным проведение анализа изменений на полученных кадрах для выявления точек спроецированной линии.

На рисунке 24 приведены результаты работы алгоритма построения карты глубины изображения.

Рис.24. Результаты работы алгоритма построения карты глубины изображения

На рисунке 25 представлена таблица с указанием времени работы этапов построения полной карты глубины изображения. На рисунке 26 продемонстрирована зависимость абсолютной максимальной ошибки в измерении карты глубины изображения от расстояния. Для увеличения скорости построения карты глубины изображения рекомендуется установка высокоскоростной камеры с использованием двигателя постоянного тока с контролем оборотов. Для увеличения точности карты глубины изображения

рекомендуется использовать более мощный лазерный модуль и подобрать диапазон цвета и процент пропускания света для аппаратного конверсионного фильтра.

Оборудование	Реализация	Среднее время работы алгоритма за 10 запусков (мс)
Raspberry PI 3	Съемка кадров с изображениями линий	15569
Raspberry PI 3	Построение полной карты глубины (однопоточная, C++)	22326
Intel Core i5 3210m	Построение полной карты глубины (однопоточная, C++)	10679

Рис.25. Таблица с описанием результатов работы

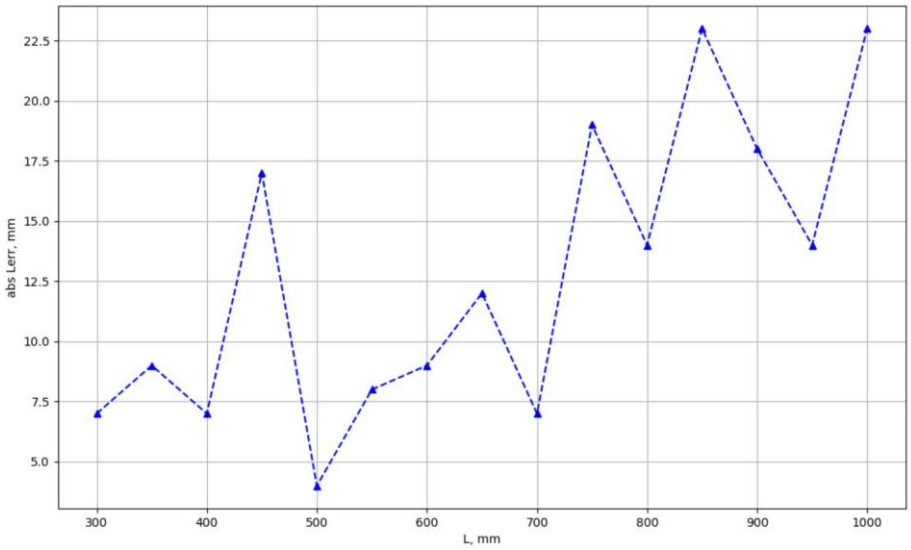

Рис.26. Зависимость ошибки в измерении карты глубины изображения от расстояния

3.3.4. Поиск угла поворота

На рисунке 27 приведены результаты работы модификации алгоритма детектирования особых точек на проекции рабочей сцены, описанного в разделе 2.2.3. В качестве графов для описания шаблонов объекта были выбраны G' - остовные подграфы полных графов, полученных на основе графов G для каждого угла поворота w, p.

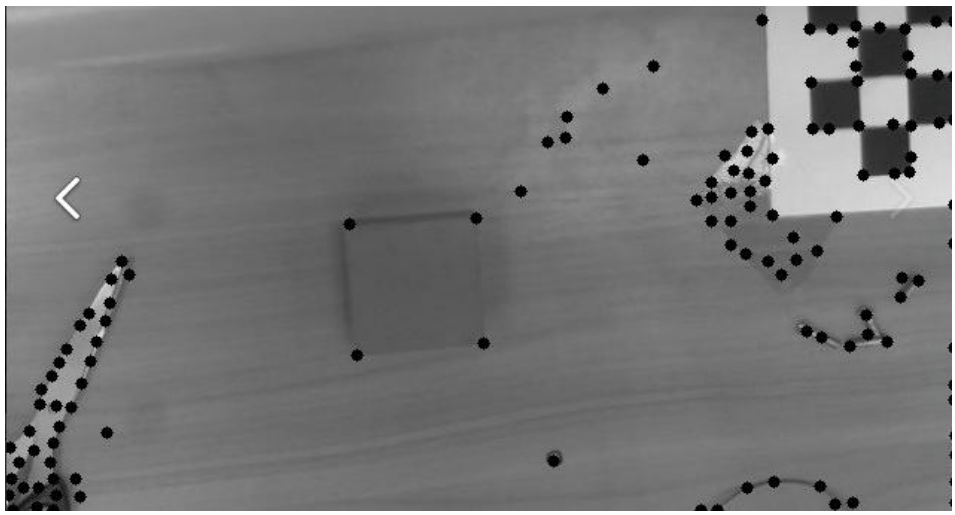

Рис.27. Результаты работы алгоритма детектирования особых точек на изображении

Для уменьшения затрат ресурсов на сопоставление изображения с 225 шаблонами был построен пороговый фильтр на основе подграфа G' с 6 вершинами. На рисунке 28 приведен пример работы алгоритма поиска наиболее подходящего шаблона на изображении сцены (с точностью до угла поворота шаблона – угол поворота однозначно определяется в разделе 2.3 на шаге 7).

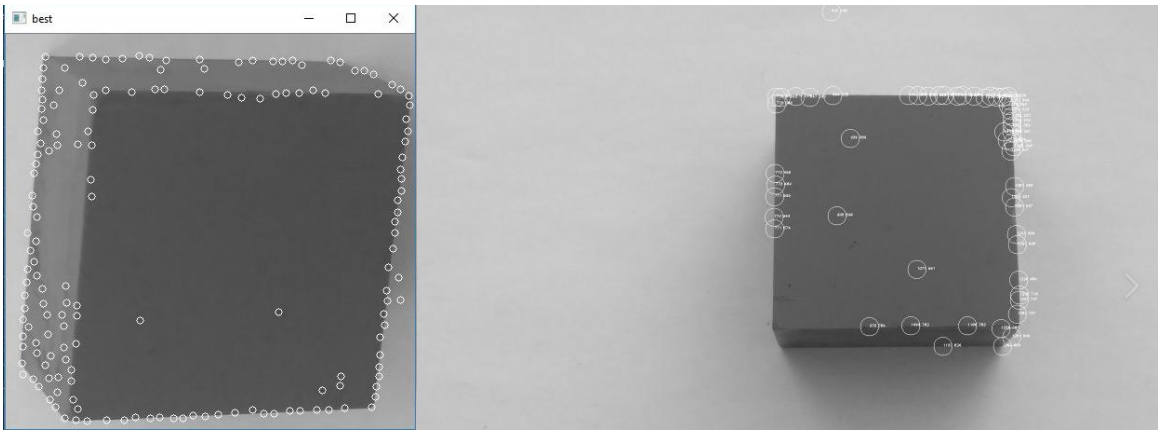

Рис.28. Пример работы алгоритма

На рисунке 29 представлена таблица с указанием времени работы алгоритма с различными параметрами для поиска наиболее релевантного шаблона на изображении, полученном с рабочей сцены.

Оборудование	Реализация	Среднее время работы алгоритма за 10 запусков (мс)
Raspberry PI 3	Поиск на сцене с 1000 особых точек	233
Intel Core i5 3210m	Поиск на сцене с 1000 особых точек	53
Raspberry PI 3	Поиск на сцене с 2000 особых точек	1109
Intel Core i5 3210m	Поиск на сцене с 2000 особых точек	235
Raspberry PI 3	Поиск на сцене с 4000 особых точек	3978
Intel Core i5 3210m	Поиск на сцене с 4000 особых точек	595
Raspberry PI 3	Поиск на сцене с 1000 особых точек 225 шаблонов с 200 особых точек (однопоточная, C++)	1575
Intel Core i5 3210m	Поиск на сцене с 1000 особых точек 225 шаблонов с 200 особых точек (однопоточная, C++)	315
Raspberry PI 3	Поиск на сцене с 2000 особых точек 225 шаблонов с 200 особых точек (однопоточная, C++)	5337
Intel Core i5 3210m	Поиск на сцене с 2000 особых точек 225 шаблонов с 200 особых точек (однопоточная, C++)	1059
Raspberry PI 3	Поиск на сцене с 4000 особых точек 225 шаблонов с 200 особых точек (однопоточная, C++)	11139
Intel Core i5 3210m	Поиск на сцене с 4000 особых точек 225 шаблонов с 200 особых точек (однопоточная, C++)	2020

Рис.29. Таблица с описанием результатов работы

Такие значения времени являются следствием неоптимизированности библиотеки OpenCV, используемой в проекте для реализации СТЗ, на микропроцессорах ARM. Для увеличения скорости работы алгоритма предлагается построить многопоточную реализацию алгоритма на CPU и GPU Raspberry PI 3 (80 ГФлопс), а также реализацию на платформе NVidia Jetson TX2 (1 ТФлопс) для тестирования предлагаемого алгоритма в качестве программной компоненты на встраиваемой системе технического зрения, работающей в режиме реального времени.

На рисунке 30 представлена точность сопоставления изображения сцены с массивом из 225 шаблонов, с поиском 4000 точек на изображении сцены и 200 точек на каждом шаблоне. Функция качества $Quality$ зависит от отношения количества совпавших длин ребер графа G_{scene} с длинами соответствующих ребер графа G_{templ} на шаблоне с учетом параметра погрешности длин l_{err}.

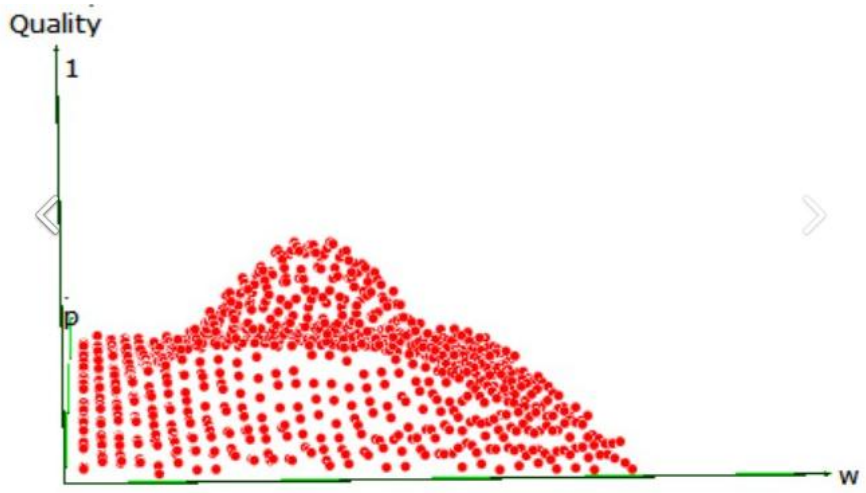

Рис.30. Точность сопоставления с массивом из шаблонов

Для увеличения точности работы алгоритма предлагается реализовать систему, автоматизирующую построение шаблонов на основе CAD модели, получаемой путем 3D сканирования и реконструкции объекта при помощи лазерного модуля с проекцией линии и сервопривода, задающего поворот модуля.

Выводы

В данной работе построена математическая модель для получения траектории перемещения звеньев кинематической цепи на основе заданных начальной точки и функции скорости на промежутке перемещения, а также поиска конечной точки траектории заданной в обобщенных координатах при помощи 6 обобщенных координат, полученных с системы технического зрения. Предложен подход к детектированию объектов на рабочей сцене с представлением объекта в виде графа, описан алгоритм поиска 6 обобщенных координат искомого объекта на рабочей сцене с использованием 2 подходов к определению карты глубины изображения. Предложена архитектура для построения мехатронной системы для управления роботом. Предложена архитектура встраиваемой отказоустойчивой системы технического зрения, предназначенной для поиска объектов на рабочей сцене и корректировки траектории движения, задаваемой оператором. Построен прототип СТЗ на основе Raspberry PI 3 с использованием платы расширения для камер, системы зеркал для получения стерео изображения, 2 камер без ИК фильтров с ИК подсветкой и лазерного модуля, проецирующего линию с сервоприводом. Построена программная реализация предложенного алгоритма для нахождения 6 обобщенных координат в виде программных компонент: построение карты глубины изображения на 3 подсистемах для реализации принципа избыточности и работы в различных условиях на сцене, детектирование объекта на изображении с определением углов поворота. Приведены примеры работы компонент СТЗ.

В подсистеме, состоящей из 2 камер, получена большая абсолютная погрешность. Это связано со сложностью проведения калибровки на изображении с выбранными камерами. В подсистеме, состоящей из камеры и системы зеркал, погрешность наблюдается в точках, лежащих в областях неровного покрытия зеркального напыления. В данной подсистеме

происходит более быстрый расчёт карты глубины изображения по сравнению с системой, состоящей из 2 камер за счет отсутствия задержки на переключение на плате расширения и создание второго снимка, а также из-за уменьшения угла обзора. Карта глубины, построенная с использованием лазерной линии и сервопривода, показывает наименьшую абсолютную ошибку по сравнению со сканирующими подсистемами, но зависит от засветки рабочей поверхности и построение карты глубины на используемом оборудовании занимает значительно больше времени.

Предложенный алгоритм поиска углов поворота искомого объекта выдает координаты углов поворота для наиболее подходящего шаблона. Предложенный алгоритм является масштабируемым на этапах построения карты глубины для найденных вершин графа G, а также сопоставления с шаблонами.

По полученным результатам можно сделать вывод о том, что построенная система позволяет получать 6 координат, обозначающих расположение искомого объекта в пространстве, но для использования в режиме реального времени в текущей реализации ПО и оборудования разработанная система не подходит. В работе предложены рекомендации по модификации существующих компонент системы технического зрения для увеличения точности и скорости определения координат искомого объекта на рабочей сцене.

Заключение

В ходе данной работы:

- Построена математическая модель робота с СТЗ для нахождения позиций звеньев робота-манипулятора при взаимодействии с искомым объектом,
- Рассмотрены алгоритмы компьютерного зрения для детектирования объектов и их положения в пространстве,
- Предложена модификация существующих алгоритмов при заданных ограничениях системы на вычислительные ресурсы,
- Разработана архитектура взаимодействия подсистем СТЗ,
- Разработан прототип встраиваемой СТЗ на основе предложенной архитектуры,
- Проведен анализ работы созданного прототипа на основе модификации предложенного алгоритма.

Дальнейшие направления развития работы:

- Реализовать алгоритм с использованием GPU Raspberry PI 3
- Реализовать СТЗ на основе платформы Nvidia Jetson TX2 с использованием откалиброванной стереокамеры ANTVR LucidCam,
- Автоматизировать процесс построения шаблонов на основе CAD модели объектов,
- Разработать систему слабого ИИ для коррекции траектории задаваемой оператором при перемещении звеньев манипулятора.

Список литературы

1. Кулаков Ф. М., Копирующее и супервизорное управление роботами при большом запаздывании сигналов управления и обратной связи. Часть I // Робототехника и техническая кибернетика. 2016. № 1 (10). С. 55–61.
2. Кулаков Ф. М., Алферов Г. В., Смирнов Е. Н., Управление роботами, очувствленными по усилиям // Проблемы механики и управления: Нелинейные динамические системы. 1996, №28. С. 6.
3. Rebecq H., Gallego G., Scaramuzza D. EMVS: Event-based Multi-View Stereo // British Machine Vision Conference (BMVC), York, 2016. P.18.
4. Artigas J., Ryu J., Preusche C., Hirzinger G. Network representation and passivity of delayed teleoperation systems // IEEE IROS, 2011. P. 177–183.
5. Artigas J., Riecke C., Weber B., Stelzer M., Balachandran R., Schaetzle S., Bayer R., Steinmetz M., Voegl J., Brunner B., Albu-Schaeffer A., Guk M. ,Zaborovskyi V., Kondratiev A., Muliukha V., Silinenko A., Shmakov O. Force-feedback teleoperation of on-ground robots from the international space station in the frame of the «KONTUR-2» experiment // Extreme robotics. Proceedings of the International Scientific and Technological Conference. Saint-Petersburg: «AP4Print». 2016. P. 50-56.
6. Кулаков Ф.М., Копирующее и супервизорное управление роботами при большом запаздывании сигналов управления и обратной связи. Часть II // Робототехника и техническая кибернетика. 2016. № 2 (11). С. 29-38.
7. Кулаков Ф.М., Алферов Г.В., Нечаев А.И., Чернакова С.Э. Информационные системы виртуальной реальности в мехатронике и робототехнике. Изд-во Санкт-Петербургского университета, 2009. 168с.

8. Gorbunov V. I., Kulakov F. M. Project of vision system for remote controlled space robots // Extreme robotics. Proceedings of the International Scientific and Technological Conference. Saint-Petersburg: «AP4Print». 2016. P. 276-281.
9. Горбунов В. И., Моисеев О. С. Система копирующего управления роботом-манипулятором // Сборник трудов XLVIII Международной научной конференции аспирантов и студентов «Процессы управления и устойчивость» Control Processes and Stability (CPS'17) ISSN 2313-7304, 2017. in print.
10. Кулаков Ф.М., Соловьев А.Е., Смирнов Е.Н., Беспальчиков Е.А., Наумов В.Б. Интегрированное программирование роботов на основе методов нечеткой логики с использованием "перчаточного" интерфейса. // Отчет о НИР № 98-01-01103 Российский фонд фундаментальных исследований.
11. Глазырин А.Г., Горбунов В.И., Гусев Д.А. Компьютерное моделирование окружающей среды в робототехнических комплексах// Сборник трудов XLVIII Международной научной конференции аспирантов и студентов «Процессы управления и устойчивость» Control Processes and Stability (CPS'17) ISSN 2313-7304, 2017. in print.
12. Фу К., Гонсалес Р., Ли К. Робототехника. М.: Мир, 1989. 624 с.
13. Leskov A.G., Moroshkin S.D. Computer vision system for objects pose estimation // Extreme robotics. Proceedings of the International Scientific and Technological Conference. Saint-Petersburg: «AP4Print». 2016. P. 287-290.
14. Gonzalez O., Shrikumar H., Stankovic J., Ramamritham K. Adaptive Fault Tolerance and Graceful Degradation // University of Massachusetts - Amherst 1997. P.33
15. Heath S. Embedded systems design. EDN series for design engineers (2 ed.). 2003 Newnes. P. 2.

16. Spong M.W., Hutchinson S., Vidyasagar M. Robot Modeling and Control. – Wiley, 2005. – 496 p.
17. Зенкевич С.Л., Ющенко А. С. Основы управления манипуляционными роботами // Издательство МГТУ им. Н.Э. Баумана 2004 480 с.
18. Hartley, R.~I. and Zisserman, A. Multiple View Geometry in Computer Vision, "2004","Cambridge University Press, ISBN: 0521540518"
19. Nalpantidis L., Sirakoulis G., Gasteratos A. Review of stereo vision algorithms: from software to hardware // International Journal of Optomechatronics, P. 435–462, 2008.
20. Konolige K. Improved occupancy grids for map building // AUTONOMOUS ROBOTS Springer Science+Business Media B.V., Formerly Kluwer Academic Publishers B.V., p. 351-367.
21. Baretto S., Remy E., Marcilio F. A Method for Image Processing and Distance Measuring Based on Laser Distance Triangulation // Electronics, Circuits, and Systems (ICECS), 2013 IEEE 20th International Conference
22. Canny J. A Computational Approach to Edge Detection // IEEE Transactions on pattern analysis and machine intelligence, VOL. PAMI-8, №6, November 1986. P. 2.
23. Shi J., Tomasi C. Good features to track // IEEE Conference on Computer Vision and Pattern Recognition (CVPR94) Seattle, June 1994. P. 2.
24. Ресурсный центр «Инновационные технологии композитных наноматериалов» СПбГУ. http://researchpark.spbu.ru/nanocomposites
25. Проект 3DBerry. http://3dberry.org/

Приложение А

Описание ООП реализации модуля поиска объекта на сцене:

```
#include <opencv2/core.hpp>
#include <opencv2/features2d.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#include <iostream>
#include <vector>
#include <list>
#include <algorithm>
#include <ctime>

class Scene {
	class Example {
		std::vector<cv::Point2i> example_keypoints;
		std::vector<std::vector<int>> example_len_vectors;
	public:
		cv::Mat example_image;
		Example(cv::Mat image = cv::Mat()) : example_image(image) {}
		~Example() {}
		void Compute(int maxCorners, double qualityLevel, double
minDistance) {...}
		std::vector<std::vector<int>>& LenVectors() { return
example_len_vectors; }
	};
	struct len_coords {
		int len;
		cv::Point2i coords;
		bool find_check_flag;
		bool check_flag;
		len_coords(int lenght = 0, cv::Point2i cords = cv::Point2i()) :
len(lenght), coords(cords), find_check_flag(0), check_flag(0) {}
		bool operator<(const len_coords& a) const
		{
			return (len < a.len);
		}
	};
	class Scope {
		std::vector<cv::Point2i> scope_keypoints;
		std::vector<std::vector<len_coords>> scope_len_vectors;
	public:
		cv::Mat scope_image;
		Scope(cv::Mat image = cv::Mat()) : scope_image(image){}
		~Scope() {}
		void Compute(int maxCorners, double qualityLevel, double
minDistance) {...}
		std::vector<std::vector<len_coords>>& LenVectors() { return
scope_len_vectors; }
	};
	std::vector<Example> examples;
	Scope ground;
	std::list<cv::Point2i> best_object_coords;
	double procent_of_coincidence;
public:
	Scene(const uint n, const char names[][128], int maxCorners = 10000,
double qualityLevel = 0.01, double minDistance = 10)
		: ground(Scope()), procent_of_coincidence(0.0) {
		examples.resize(n);
		for (uint i = 0; i < n; ++i) {
			examples[i] = Example(cv::imread(names[i], 0));
```

```
                  examples[i].Compute(maxCorners, qualityLevel, minDistance);
            }
      }
      ~Scene() {}
      void SetScope(cv::Mat scope_names, int maxCorners = 10000, double
qualityLevel = 0.01, double minDistance = 10) {...}
      void Find(Example &sample, double quality, uint count_for_binsearch, int
eps) {...}
      void ShowBestSearch() {...}
      void Detect(double quality = 0.5, uint count_for_binsearch = 4, int eps
= 0.0) {...}

};

int main() {
      const char templates_names[countOfTemplates][128] = {...}; //sending N
templates
      const char ground_names[countOfScenes][128] = {...}; //sending scene

      Scene my_scene(countOfTemplates, templatesNames, countOfPoints,
qualityLevel, minDistance);

      for (uint i = 0; i < countOfScenes; ++i) {
            my_scene.SetScope(cv::imread(ground_names[i], 0), 1000, 0.05, 10);
            my_scene.Detect(0.5, 4, 10);
            my_scene.ShowBestSearch();
      }

      system("pause");
      return 0;
}
```