Разработка аппаратно-программного комплекса для адаптивной сортировки твердых бытовых отходов

Кириленко В. Д.

Руководитель: —.

ГБОУ «Школа № 1561», Москва, Россия, 2020 г.

Аннотация

В данном проекте представлена автономная система, позволяющая классифицировать твёрдые бытовые отходы, а также управлять манипулятором-сортировщиком мусора. В проекте так же представлены разработанный манипулятор-сортировщик и система моделирования для проведения тестов. Сортировка выполняется на основе материала, формы, либо конкретного класса объекта. Сортировка выполняется при помощи алгоритмов компьютерного зрения и машинного обучения. Разработка сосредоточилась на адаптивности и ускорении обучения, что позволяет системе приспосабливаться к изменениям в отходах.

ОГЛАВЛЕНИЕ

---

1. Введение

Загрязнение окружающей среды бытовыми отходами ведет к нарушению экологического баланса на всей планете. Однако, почти любой мусор пригоден для переработки и повторного использования. Одна из главных проблем цикла переработки мусора заключается в том, как его рассортировать на фракции, которые можно использовать для вторичной переработки. Роботизация этого процесса позволит как сократить затраты на весь цикл переработки отходов, так и уменьшить процент ошибки при сортировке[1].

Для реализации классификатора твёрдых отходов было выбрано использование компьютерного зрения для поиска и алгоритмов машинного обучения.

Цель проекта состоит в разработке аппаратно-программного комплекса, позволяющего в автономном режиме сортировать различные объекты в зависимости от их класса. Для корректной работы необходимо достичь точности не менее 90% и время переобучения не должно превышать 2-х минут. Процесс работы над проектом можно подразделить на следующие задачи:

1. Создание системы компьютерного зрения (далее — СКЗ), позволяющей с помощью закреплённой над рабочей зоной камеры расчитывать параметры ограничивающего прямоугольника для каждого сортируемого объекта
2. Изучение различных методов классификации объектов, основанных на визуальном представлении

3. Реализация алгоритма классификации в проекте
4. Разработка виртуальной среды, позволяющей тестировать алгоритм работы готового устройства
5. Разработка и создание собственного манипулятора, способного сортировать объекты массой до 1.5кг и шириной до 110мм для проверки работоспособности всех систем в реальных условия.

На данный момент разработкой подобных решений занимается всего несколько компаний, например Sadoko и Max AI. Эти компании используют в своих разработках глубинные нейронные сети. Такой подход позволяет добиться хорошей точности, однако для обработки необходимы большие вычислительные мощности[2], а процесс переобучения может занимать до нескольких часов(см табл. 1), что исключает возможность процесса адаптивного обучения.

---

2. Описание системы, принципов работы и используемых технологий

2.1 Система компьютерного зрения

Рассмотрим получившуюся СКЗ. Задача была разбита на две подзадачи:

- Устранение дефектов и искажений изображения, получившихся вследствие возникновения перспективы
- Создание бинарной маски для отделения рассматриваемых объектов от заднего фона.

Для решения первой проблемы было принято решение разместить четыре ArUco[3] маркера по углам рабочей области. Подобные метки минимально зависимы от таких внешних условий, как освещение, угол наклона относительно камеры и т.п. Основная идея состоит в поиске всех 4-х меток и, как следствие, 4-х углов рабочей зоны манипулятора (см. рис. 1.1), что позволяет вычислить матрицу преобразования и использовать функцию коррекции перспективы, в результате чего получаем изображение,

приближенное к ортогональному виду сверху (см. рис. 1.2). Такой результат позволит преобразовывать координаты объекта из глобальной системы отсчёта, связанной с камерой в локальную декартову систему рабочего поля. Матрица преобразования рассчитывается путём решения данной системы:

$$\begin{bmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 & -x_0 \cdot u_0 & -y_0 \cdot u_0 \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1 \cdot u_1 & -y_0 \cdot u_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2 \cdot u_2 & -y_0 \cdot u_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3 \cdot u_3 & -y_0 \cdot u_3 \\ 0 & 0 & 0 & x_0 & y_0 & 1 & -x_0 \cdot v_0 & -y_0 \cdot v_0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1 \cdot v_1 & -y_1 \cdot v_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2 \cdot v_2 & -y_2 \cdot v_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3 \cdot v_3 & -y_3 \cdot v_3 \end{bmatrix} \cdot \begin{bmatrix} c_{00} \\ c_{01} \\ c_{02} \\ c_{10} \\ c_{11} \\ c_{12} \\ c_{20} \\ c_{21} \end{bmatrix} = \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ v_0 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix},$$

где (x_i, y_i) - координаты i-й точки в глобальной системе отсчёта, (u_i, v_i) - координаты i-й точки в локальной системе отсчёта после преобразования, коэффициенты c_{ij} используются для расчёта координат $\begin{cases} u_i = \frac{c_{00} \cdot x_i + c_{01} \cdot y_i + c_{02}}{c_{20} \cdot x_i + c_{21} \cdot y_i + c_{22}} \\ v_i = \frac{c_{10} \cdot x_i + c_{11} \cdot y_i + c_{12}}{c_{20} \cdot x_i + c_{21} \cdot y_i + c_{22}} \end{cases},$

После того, как был получен наиболее удобный вид изображения, рассмотрим вторую подзадачу. Она была решена путём проведения предварительной калибровки среднего значения цвета, переводом изображения в цветовое пространство HSV[4] и применением инвертированного алгоритма поиска цветов в диапазоне, что на выходе даёт бинарную маску, соответствующую отклонению цвета пикселей от калибровочного стандарта (см. рис. 1.3). Подобный алгоритм подходит для поиска как непрозрачных, так и прозрачных объектов, которые ввиду преломления света всё равно будут найдены. Далее, выделяем отдельные объекты на получившейся маске, используя вспомогательные морфологические операции для повышения

точности. Выделенные объекты удобно вырезать из оригинального изображения для дальнейшей обработки и классификации.

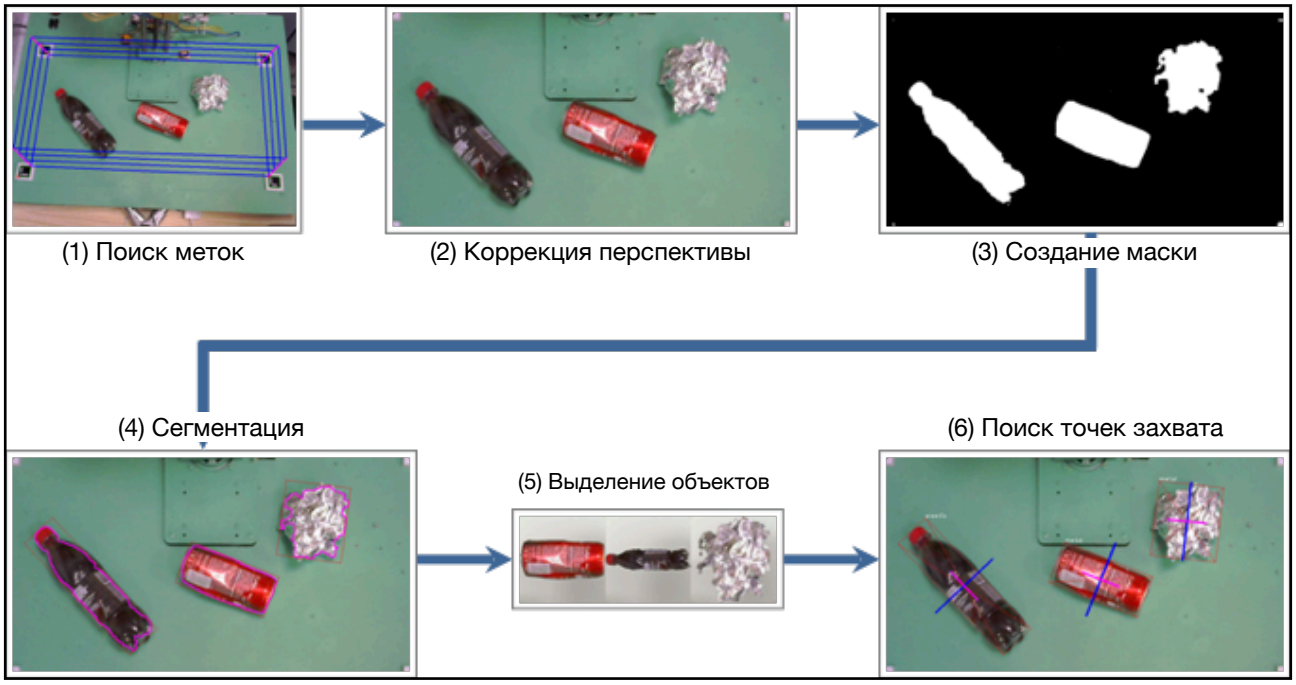

Рис. 1 — Этапы выделения объектов

2.2 Распознание образов

2.2.1 Выбор метода

Распознание образов является классической задачей в области машинного зрения и нейронных сетей и имеет множество путей решений в зависимости от задачи. В данном случае, так как проблема поиска объектов решена без применения алгоритмов машинного обучения, задача распознавания сводится к определению метки объекта на некотором входном изображении, классификации.

Глобально, для решения поставленной задачи используются несколько методов. Наиболее подходящими из них являются следующие:

- Обучение и применение свёрточной нейронной сети (CNN[5]) для классификации
- Обучение и применение глубинной нейронной сети (DNN) для классификации
- Использование одного из алгоритмов классификации или кластеризации (напр., метод k-ближайших соседей).

По результатам изучения имеющихся решений было выявлено следующее:

- Прямое применение CNN классификатора требует большого количества тренировочных данных и показывает точность в ~30%
- Прямое применение глубоких нейронных сетей показывает значительно большую точность по сравнению со свёрточными, однако остаётся недостаточно точными в рамках проекта, а так же имеют слишком большое время обучения (см. табл. 1)
- Прямое использование классификаторов/кластеризаторов имеет минимальное время обучения (в диапазоне от 0.02 до 1 мин.), однако имеет точность, недостаточную для решения задачи (около 10%).

Модель	Процессор	Видеокарта	Время тренировки	Точность
Inception V3	AMD Ryzen 3 3200G	MSI GeForce GTX 1050Ti	13ч 24мин	~50 %
Inception V4	AMD Ryzen 3 3200G	MSI GeForce GTX 1050Ti	21ч 3мин	~60 %
Inception V5	Intel Core i7 6700k	Gigabyte GeForce GTX 1070Ti	12ч 11мин	~61 %
MobileNet	AMD Ryzen 3 3200G	MSI GeForce GTX 1050Ti	20мин	~30 %

Табл. 1 — обучение различных моделей (8 классов по 100 примеров)

2.2.1 Разработанный метод

В процессе работы над проектом был разработан метод, обучающийся за 11.2 секунд с точностью 93.9%. Обучение и тестирование проводились на CPU Intel Core i5 2.4 GHz. Идея состоит в комбинации CNN ImageNet[6], позволяющей преобразовывать растровое RGB изображение в двумерный вектор признаков и одного из алгоритмов классификации/кластеризации. Подобная скорость обучения и точность классификации позволяют без длительной остановки работы добавлять новые классы и/или расширять уже имеющиеся.

2.2.1 Выбор классификатора

Для выбора наиболее подходящего классификатора был проведён сравнительный анализ 5-и наиболее распространённых алгоритмов. На рис. 2 приведены матрицы ошибок для сравниваемых алгоритмов, тестирование проводилось на одном и том же оборудовании и обучающей выборке. Матрица ошибок[7] представляет собой матрицу M, которая рассчитывается по следующей формуле: $M = \{m_{ij}\}_{i,j=0}^{C}$, $m_{ij} = \sum_{k=0}^{N} [a(x_k) = j][y_k = i]$ для некоторой выборки x_i ($i = 1,\ldots,N$, y_i - метка i-го класса $y_i \in \{1,2,\ldots,C\}$), каждый объект которой относится к одному из C классов и классификатор a, который эти классы предсказывает. Данная матрица показывает сколько объектов класса i были классифицированы как j. В таблице 2 приведены результаты тестирования изучаемых алгоритмов, на рис. 3 представлен график, позволяющий сравнить точность и время обучения алгоритмов.

На основании полученных данных было принято решение о выборе алгоритма SVC[8] для классификации. SVC, или C-Support Vector Classification является разновидностью метода опорных векторов. Основная идея состоит в переводе исходных векторов в пространство более высокой размерности и

поиск разделяющей гиперплоскости с максимальным зазором в этом пространстве. Формально, алгоритм может быть описан следующем способом: точки имеют вид $\{(x_1, c_1), (x_2, c_2), \ldots, (x_n, c_n)\}$, где c_iявляется индексом класса, которому принадлежит точка x_i. Строится разделяющая гиперплоскость, которая имеет вид: $w \cdot x - b = 0$. Вектор w - перпендикуляр к разделяющей плоскости, $\frac{b}{\|w\|}$ - расстояние от гиперплоскости до начала координат. Далее, задача сводится к минимизации $\|w\|$ для формулы $c_i \cdot (w \cdot x_i - b) \geq 1,\ 1 \leq i \leq n$.

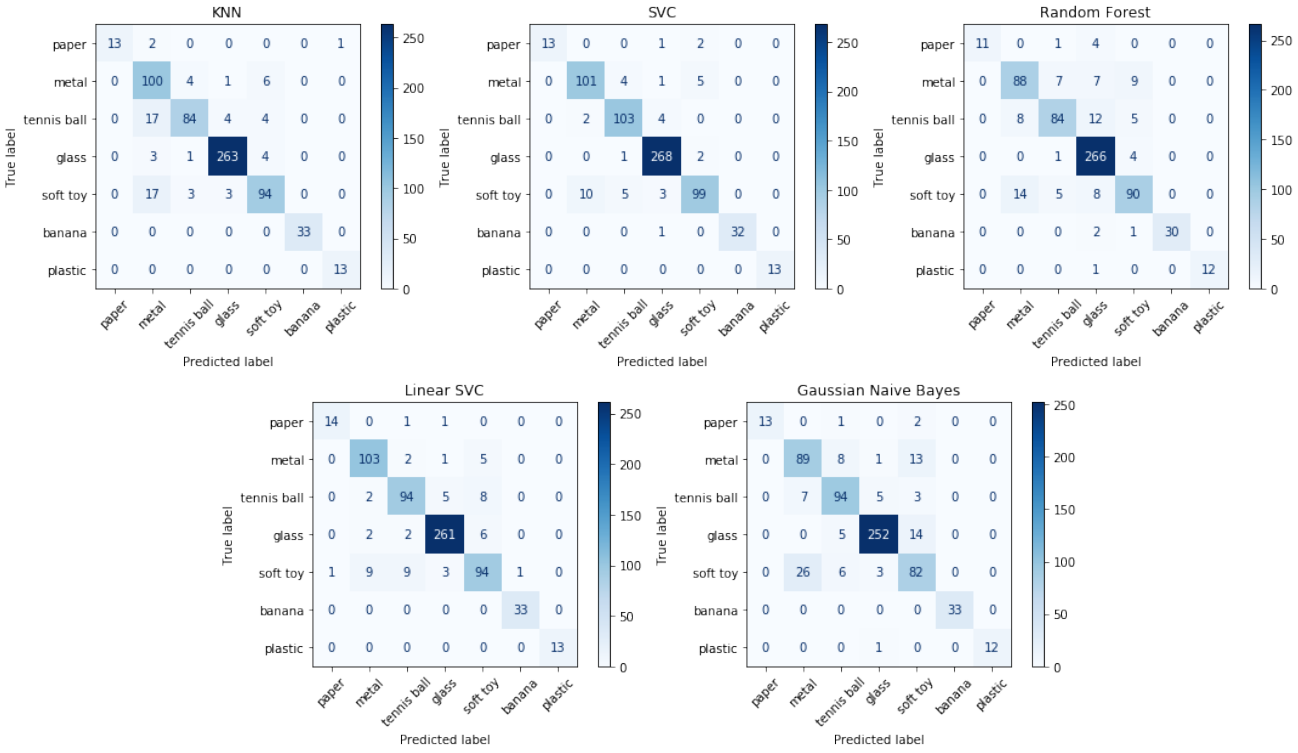

Рис. 2 — Матрицы ошибок для различных классификаторов

Метод	Время обучения, мс	Точность, %
SVC	11900	93.9
Linear SVC	55200	91.3
KNearestNeighbors	418	89.6
KNearestNeighbors	6800	86.7
Gaussian Naive Bayes	118	85.8

Табл. 2 — Сравнение времени обучения и точности алгоритмов классификации

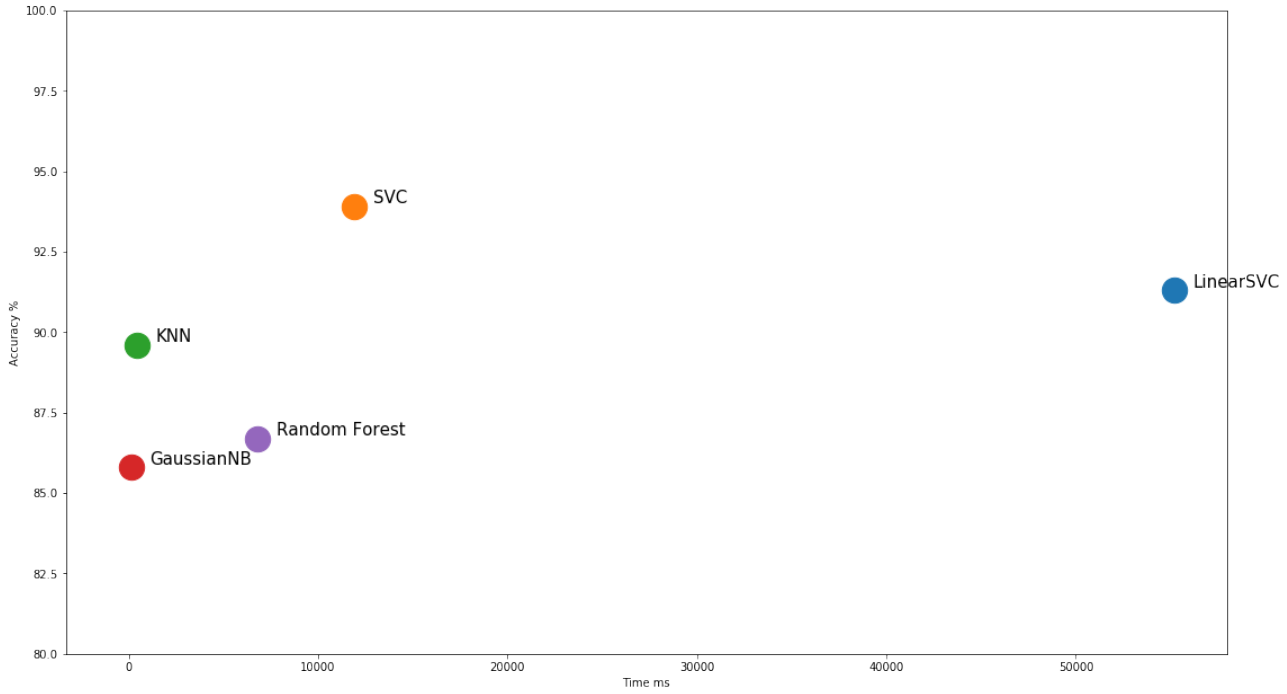

Рис. 3 — Распределение классификаторов по времени и точности

2.2.3 Реализация

Итоговая реализация СКЗ и классификатора выполнены на высокоуровневом языке Python с использованием следующих библиотек:

- OpenCV - для захвата видео с камеры, и обработки изображений в процессе работы
- TensorFlow 1.14 - для обработки ImageNet

- Scikit-learn - для обработки SVM
- OpenRV - библиотека собственной разработки для ускорения разработки.

Такой выбор обусловлен простотой использования и скоростью моделирования, что позволяет минимизировать сроки разработки. Далее, СКЗ и классификатор будут объединены под названием Система Зрительного Распознания Объектов (СЗРО).

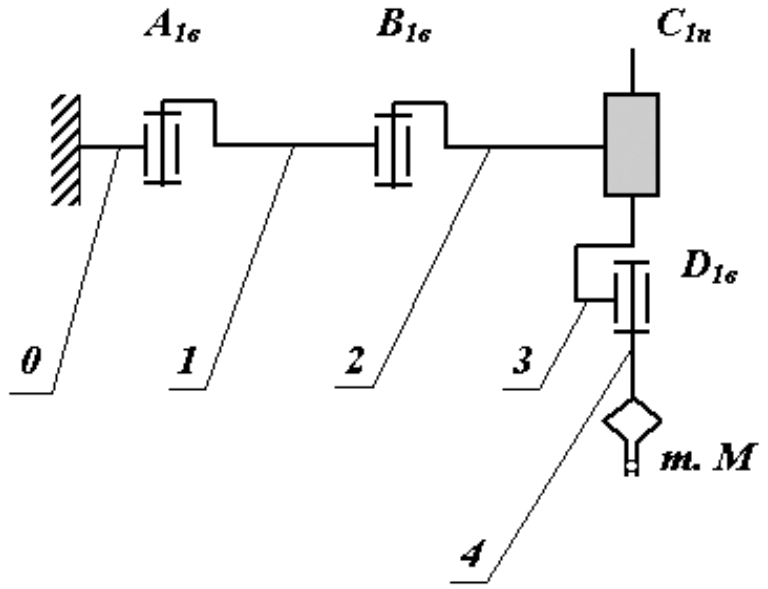

Рис. 4 — Кинематическая схема манипулятора SCARA

2.3 Виртуальная модель

Система моделирования должна отвечать следующим требованиям:

- Возможность тестирования алгоритмов работы реального устройства внутри СМ
- Возможность обмена данными между СМ и СЗРО через API
- Возможность масштабирования СМ до нескольких параллельно работающих устройств
- Поддержка различных кинематических систем манипулятора

- Возможность синхронизации СМ и реального манипулятора.

В процессе изучения готовых физических движков для создания *СМ*, были изучены следующие SDK:

1. Gazebo,
2. V-rep,
3. Webots,
4. MRS,
5. Unity 3D,
6. Unreal Engine

Наилучшим вариантом из перечисленных является Gazebo, однако от него пришлось отказаться ввиду высокой сложности воссоздания линейного перемещения системы. По аналогичным причинам, для проекта не подходят Webots, v-rep и MRS. Таким образом, для реализации *СМ* был выбран движок Unity 3D так, как он хорошо документирован, позволяет без особых затрат изменять кинематику и имеет встроенную поддержку последовательного соединения для синхронизации с реальным устройством, в отличие от Unreal Engine.

2.3 Манипулятор

Процесс создания манипулятора заключает в себе следующие задачи:

- Подобрать кинематическую схему
- Определить технические характеристики манипулятора исходя из веса предметов и размеров рабочей зоны
- Максимальный вес переносимого объекта - 1.5кг
- Размер рабочей зоны - 450мм × 900мм

Стоимость манипуляторов определяется в основном ценой электроприводов. Чем мощнее электропривод, тем выше его цена. Поэтому при выборе кинематической схемы основное внимание было уделено снижению нагрузки на двигатели манипулятора.

В большинстве кинематических схем приводы воспринимают статические и динамические нагрузки, как от сил веса перемещаемого груза, так и сил веса звеньев. Это требует значительного увеличения мощности электродвигателей для приводов и дополнительных тормозных устройств, а также противовесов.

В кинематической схеме SCARA (рис. 4) звенья манипулятора взаимно поворачиваются в горизонтальной плоскости, а захват совершает поступательные перемещения вверх и вниз. При этом силы веса звеньев и веса груза воспринимаются подшипниками кинематических пар и оказывают влияние только через силы трения в парах. Это позволяет использовать электроприводы малой мощности и, соответственно, меньшей стоимости, чем в любых других кинематических схемах. В результате все это удешевляет конструкцию. С учетом выше сказанного данном проекте была использована кинематическая схема SCARA.

Манипулятор приводится в действие тремя шаговыми моторами Nema 21, что позволяет с высокой точностью контролировать движение всех узлов. Для манипуляций с сортируемыми объектами используется комбинация из пневматического захвата с параллельными губками, установленного на серво-машинке и пневмоприсоски, сменяющих друг друга в зависимости от класса объекта (напр., для манипуляции с бутылкой используется грейферный захват, а для пакета - присоска). Серво-машинка используется для точного позиционирования, позволяющего эффективно брать такие объекты, как бананы, бутылки и т.п.

Для упрощения управления манипулятором, необходимо использовать формулы обратной кинематики, позволяющие преобразовывать координаты объекта из декартовой системы отсчёта поля в три угла для каждого из звеньев манипулятора. Исходя из схемы на рис. 5, угол $q_2 = \cos^{-1}\dfrac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1a_2}$,

$q_1 = \tan^{-1}\dfrac{y}{x} - \tan^{-1}\dfrac{a_2 \sin q_2}{a_1 + a_2 \cos q_2}$ и $q_3 = \cos^{-q}\langle \overrightarrow{a_2},\ \overrightarrow{L} \rangle$.

Низкоуровневое управление манипулятором реализовано на плате Arduino Mega 2560, для коммуникации с управляющим компьютером реализован API для общения с *СЗРО* и/или *СМ*.

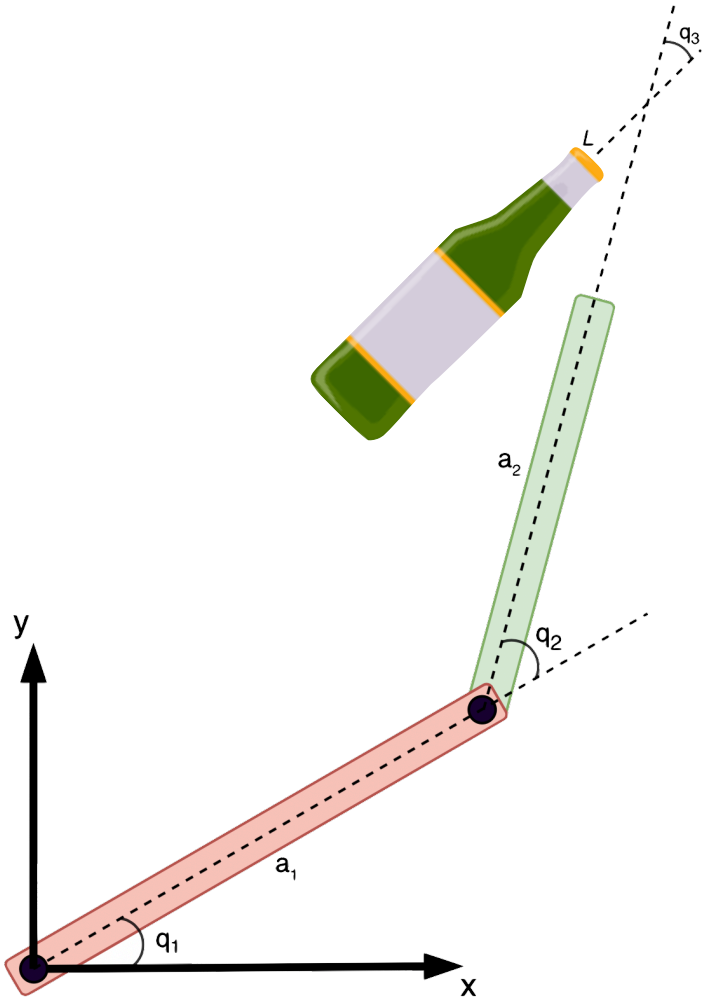

Рис. 5 — Обратная кинематика

3. Заключение

В результате проделанной работы была разработана автоматическая система для сортировки, способная распознавать множество классов объектов и управлять роботом-сортировщиком, а так же виртуальная среда, позволяющая проводить тестирование работоспособности программных модулей, в том числе, СЗРО. Проведён анализ различных способов классификации твердых отходов, и разработан метод, подходящий под условия проекта. Использование адаптивного обучения позволяет без длительной остановки работы увеличить число классов распознаваемых объектов, что позволяет перенастроить всю систему в соответствие с конкретными требованиями.

Было принято решение продолжать развивать проект в следующих направлениях:

- Уменьшение процента ошибки при классификации объектов, путём сотрудничества с мусоросортировочным комплексом с целью сбора достаточной обучающей выборки
- Повышение точности локализации объектов
- Доработка манипулятора, путём увеличения точности перемещений и снижения ограничений на объекты (напр., возможность брать объекты шире 110мм и тяжелее 1.5кг).

С пример работы получившегося манипулятора и *СКЗ* можно ознакомиться по ссылке: https://youtu.be/P_LQVBXe5EY

4. Литература

1. Исследование и разработка научно-технических решений в области проведения сортировочных операций в режиме реального времени, с объектами, имеющими сложные характеристики, с использованием высокоэффективных робототехнических средств автоматизации. Электронный ресурс. Режим доступа: http://fcpir.ru/upload/iblock/94b/corebofs000080000ldo38um1mqn7hvc_annotation.pdf
2. D. Jirak, S. Wermter Potentials and Limitations of Deep Neural Networks for Cognitive Robots — с.1-3
3. Detection of ArUco Markers. Электронный ресурс. Режим доступа: https://docs.opencv.org/trunk/d5/dae/tutorial_aruco_detection.html, – Проверено 08.01.2020
4. Цветовая модель HSV. Электронный ресурс. Режим доступа: https://en.wikipedia.org/wiki/HSL_and_HSV – Проверено 08.01.2020.
5. Ле Мань Ха Свёрточная нейронная сеть для решения задачи классификации // ТРУДЫ МФТИ. 2016. Том 8, № 3. 2016. — с.91-97
6. ImageNet. Электронный ресурс. Режим доступа: http://image-net.org/index – Проверено 08.01.2020.
7. Матрица ошибок. Электронный ресурс. https://learnmachinelearning.wikia.org/ru/wiki/ Матрица_ошибок_(Confusion_matrix) – Проверено 08.01.2020.
8. R. Berwick, An Idiot’s guide to Support vector machines (SVMs) – с.5-28.
9. SCRA Robot Kinematics. Электронный ресурс. Режим доступа: http://www.deltatau.com/Common/technotes/SCARA%20Robot%20Kinematics.pdf – Проверено 08.01.2020.
10. Вапник В.Н. The nature of statistical learning theory – с.138-176