ВЕСТНИК ЮГОРСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
2020 г. Выпуск 4 (59). С. 33–41

DOI: 10.17816/byusu20200433-41
УДК 004.45

Е. И. Сафонов, П. В. Завьялов

РАЗРАБОТКА ПРОГРАММНО-АППАРАТНОГО ОБЕСПЕЧЕНИЯ РОБОТА, ИСПОЛЬЗУЮЩЕГО АЛГОРИТМЫ КОМПЬЮТЕРНОГО ЗРЕНИЯ

В статье приводится описание процесса проектирования и создания робота, способного использовать алгоритмы и методы компьютерного зрения для определения цветов граней, на примере кубика Рубика. Грани куба сканируются веб-камерой, установленной на роботе. Для избавления изображения от шумов используется размытие по Гауссу. Полученное изображение переводится в полностью черно-белый вид, из которого выделяется контур. Для наилучшего результата используется цветовое пространство HSV. Обмен данными между роботом и компьютером происходит по беспроводной технологии Bluetooth. Для поиска оптимального алгоритма сборки кубика используется сторонний сетевой Интернет-ресурс, в котором реализован алгоритм «Бога» сборки головоломки. Данные, полученные с Интернет-ресурса, считываются и преобразовываются в команды для робота. Робот построен на базе набора Lego Mindstorm NXT. Робот может выполнять набор команд, основанных на вращении кубика на плоскости, повороте линии кубика и перевороте кубика на другую грань.

Ключевые слова: робототехника, компьютерное зрение, кластеризация, Lego NXT

E. I. Safonov, P. V. Zavyalov

DEVELOPMENT OF SOFTWARE AND HARDWARE OF A ROBOT USING COMPUTER VISION ALGORITHMS

The article describes the process of designing and creating a robot capable of using algorithms and methods of computer vision to determine the colors of faces, using the example of a Rubik's cube. The edges of the cube are scanned by the webcam installed on the robot. To get rid of image noise, Gaussian blur is used. The resulting image is converted to a completely black and white view, from which the outline is highlighted. The HSV color space is used for best results. The exchange of data between the robot and the computer takes place via Bluetooth wireless technology. To find the optimal algorithm for solving the cube, a third-party network Internet resource is used, which implements the "God" algorithm for solving the puzzle. The data received from the Internet resource is read and converted into commands for the robot. The robot is based on the Lego Mindstorm NXT set. The robot can execute a set of commands based on the rotation of the cube on the plane, the rotation of the cube line and the flip of the cube to another face.

Key words: robotics, computer vision, clustering, Lego NXT

Введение

В работе рассматривается проектирование и разработка робота, использующего методы и алгоритмы компьютерного зрения. Примером использования подобного робота представлен процесс автоматической сборки кубика Рубика.

Кубик Рубика (далее – К.Р.) – одна из самых известных и продаваемых головоломок в истории, вызывает интерес у многих категорий специалистов. Математики доказывают теоремы о возможности нахождения решения в наименьшее число ходов. Программисты создают программные продукты, моделирующие К.Р. гигантских размеров (100×100×100 или

1000×1000×1000), а также невозможные в физическом мире – 4-, 5-, и даже 7-мерные аналоги. Инженеры ежегодно ставят мировые рекорды по созданию сверх быстрых роботов сборки К.Р., текущий рекорд 0,38 секунд был поставлен в 2018 году. А всё потому, что число всех достижимых различных состояний К.Р. 3x3x3 равно $(8!\times3^{8-1})\times(12!\times2^{12-1})/2$ = 43 252 003 274 489 860 000.

Выделим основные этапы создания программно-аппаратного комплекса:

1. Спроектировать и собрать робота, отвечающего следующим требованиям:

а) возможность вращения кубика;

б) поворот одной или двух линий кубика;

в) переворот кубика в одной из проекций.

2. Установить и настроить необходимое программное обеспечение для обеспечения связи с роботом.

3. Разработать программное обеспечение, которое позволит принимать данные сканирования цвета кубика.

4. Разработать программное обеспечение, которое позволит кластеризовать цвета по соответствующим классам.

5. Разработать программное обеспечение, которое позволит обращаться к сетевым ресурсам для составления алгоритма сборки кубика.

Схема программно-аппаратного комплекса представлена на рисунке 1.

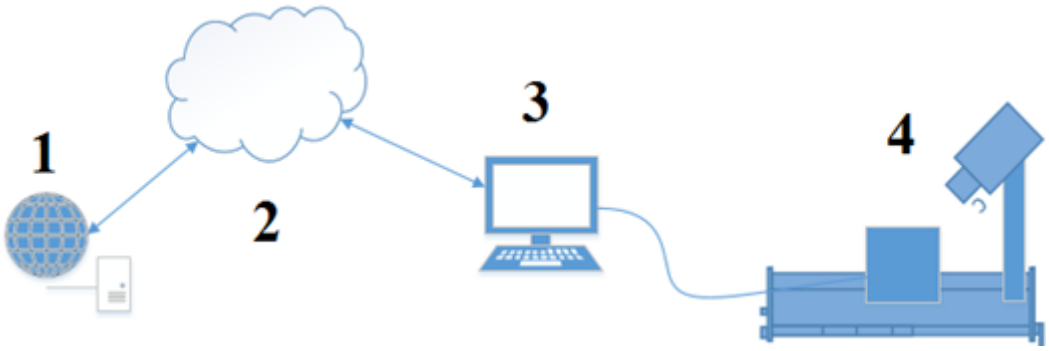


Рисунок 1 – Схематичное изображение программно-аппаратного комплекса

Программно-аппаратный комплекс можно разбить на четыре основные составляющие:

1. Удаленный сервер – необходим для расчёта оптимального алгоритма для сборки К.Р.
2. Сеть интернет – необходима для предоставления доступа к серверу.
3. Персональный компьютер – необходим для контроля и мониторинга системы.
4. Робот сборщик – необходим для сканирования куба и воздействия на него.

Требования к программно-аппаратному комплексу

Для того чтобы комплекс функционировал, необходимо наличие веб-камеры и доступ к сети Интернет и ПК. Веб-камера должна быть с разрешением не менее 480p. Скорость подключения к Интернету должна быть не менее 3 Мбит/с.

Кроме основных требований имеется ряд дополнительных:

1. Кроссплатформенность.
2. Высокая отказоустойчивость.
3. Быстродействие.
4. Отсутствие утечек памяти у приложения.

Для создания робота было решено использовать набор Lego Mindstorm NXT, ввиду скорости проектирования, сборки, надежности и взаимозаменяемости деталей.

Программно-аппаратный комплекс мог бы быть реализован только на аппаратной стороне NXT, но в данной реализации есть ряд недостатков:

1. В случае поломки одного из элементов полностью выходит из строя весь комплекс.

2. Нет возможности замены составных элементов, таких как моторы или веб-камера.

3. Ограничение по объему памяти. В контролере счёт памяти идёт на Кбайты и в таком случае, скорее всего, придется отказаться от ряда функций.

Для реализации проекта был выбран язык программирования Python, имеющий эффективные структуры данных высокого уровня и простой, но эффективных подход к объектно-ориентированному программированию, а так же библиотеки для работы с NXT и opencv [1] и библиотеки работы с матрицами – numpy [2].

В качестве IDE была выбрана профессиональная версия Pycharm, с наличием необходимых компонентов:

1. Профайлер – позволяет запускать код в специальной сессии и контролировать утечки памяти, загруженность участков кода, графически отрисовать цепочку выполнения функций.

2. Автокомплит – подсказывает пользователю, какие функции и модули уже импортированы и что готово к использованию.

3. Поиск документации – для каждого пакета пытается найти встроенное описание или же загрузить готовую документацию Sphinx.

Сборка робота

Для приведения примера работоспособности алгоритма обработки изображения построим простого робота, имеющего возможность собрать К.Р.

При сборке робота были выделены основные части:

1. «Лапа» – механизм переворота кубика для изменения его плоскости.
2. «Платформа» – держатель для кубика, позволяющий вращать его вокруг своей оси.
3. «Башня» – неподвижна и служит держателем для веб-камеры.
4. «База» – площадка, на которую установлены вышеописанные части.

Часть «Лапа» собрана из мотора для движения, двух балок по бокам для фиксации кубика в пространстве, задней перегородки движения кубика в крайнюю позицию, передней балки и двух двухмодульных фиксаторов для захвата грани кубика, во время переворота, рисунок 2.

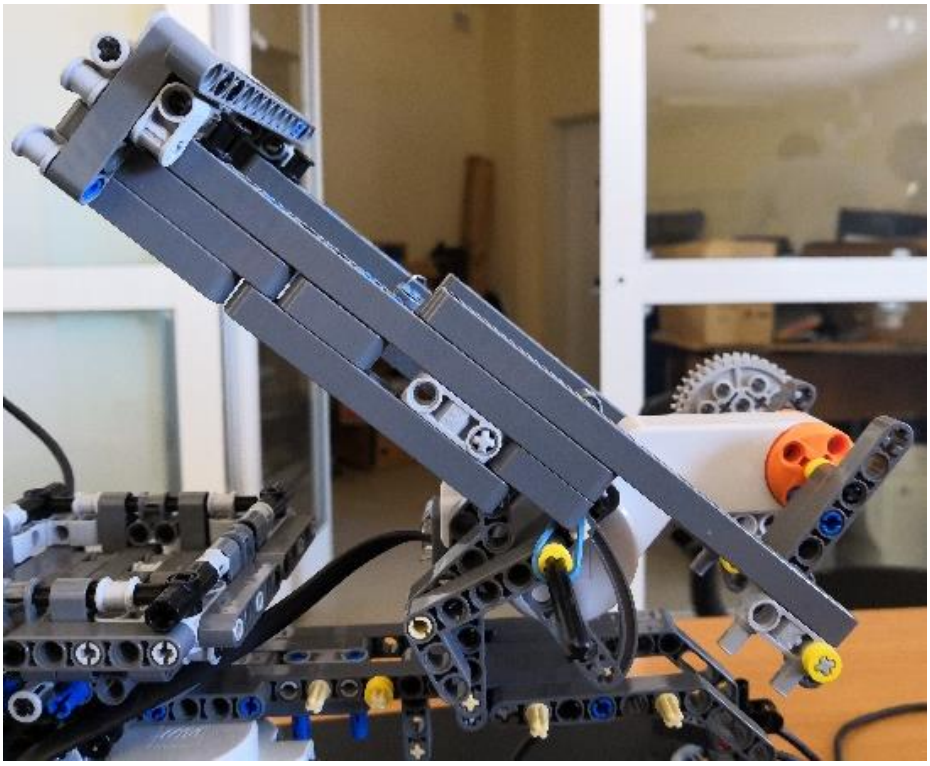


Рисунок 2 – Конструкция части «Лапа»

Во время сборки было найдено смещение «Лапы» относительно платформы на пол единицы. Для устранения дефекта были добавлены втулки желтые на пол единицы, балки изогнутые 3х5 также на пол единицы. Также был скрип при движении из-за несмазанных штифтов, плотно прилегающих к деталям. Данный момент иногда препятствовал правильной работе «Лапы», и были заменены на оси с фиксаций из резинки. Для отладочных работ к мотору была добавлена ось шестеренкой для удобства.

Часть «Платформа» была собрана по чертежам mindcuber [3], с заменой недостающих деталей, и с повышением цельности конструкции для устранения люфтов при движении конструкции, рисунок 3.

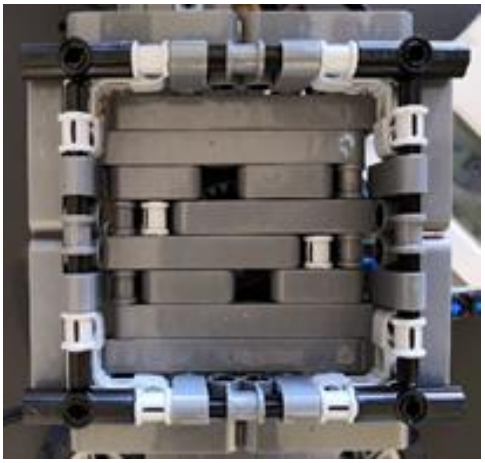


Рисунок 3 – Конструкция части «Платформа»

Часть «Башня» неподвижна для крепления веб-камеры под определенным углом и расстоянием, необходимым для обзора всей стороны кубика, и при этом не сильно отдаленным для корректной работы алгоритма классификатора и сенсора, рисунок 4.

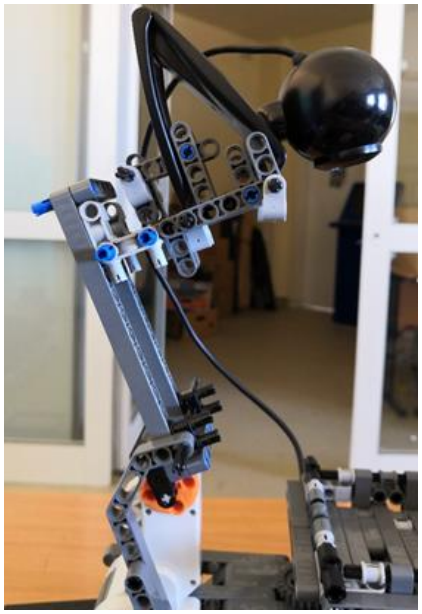


Рисунок 4 – Конструкция части «Башня»

Часть «База» имеет основу из самых длинных балок, имеющихся в распоряжении, и изогнутых балок 5х3х3, для установки ног, сделанных из колес, которые гасят лишние вибрации и крепко цепляются своей резиной к твердым поверхностям.

Управление роботом

Для управления роботом была подключена python-библиотека nxt-python, которая работает на всех операционных системах и поддерживает NXT и позволяет подключаться к роботу по Bluetooth, USB и Wi-Fi.

В нашем случае будет использоваться соединение через Bluetooth, так как связь по проводному USB-соединению занимало больше времени – 15 секунд, против 4 секунд. Последним вариантом было использование Wi-Fi, но в контроллере NXT такой модуль отсутствует. Соединение с блоком происходит через пакет bluesock, который является подпакетом NXT.

Для начала необходимо произвести сопряжение робота с компьютером по интерфейсу Bluetooth. Для этого необходимо:

1. Включить Bluetooth на обоих устройствах и установить режим, видимый для всех.
2. Включить поиск устройств на ПК и в появившемся списке выбрать своего робота. Или же узнать MAC адрес робота и попытаться соединить напрямую.
3. В ответ на запрос к подключению робот издаст звуковой сигнал и предложит пароль для установления соединения, нужно его запомнить и нажать на желтую кнопку, чтобы подтвердить свой выбор.
4. Ввести запомненный пароль с робота в появившуюся строку на компьютере.

MAC-адрес NXT используется по причине удобства взаимодействия с роботом во время разработки, т.к. он зашит глубоко в устройство и его сложно изменить простыми способами, а так же поскольку при запуске не нужно выбирать из большого числа устройств (наушники, приставки, джойстики, колонки, калькуляторы, клавиатуры, мышки и т. п.).

Схема управления моторами использует популярное решение MotorControl. Эта библиотека написана на Matlab. Таким образом, схема управления имеет вид:

1. Подключаемся к роботу и включаем программу на ПК.
2. Отправляем более сложные инструкции моторам по Bluetooth.
3. Опрашиваем датчик расстояния.
4. Закрываем соединение и строим поворот.

Используя данную программу, мы получаем прецизионное управление, позволяющее увеличить точность поворота, контролировать плавный старт и остановку, а также фиксировать окончание движения.

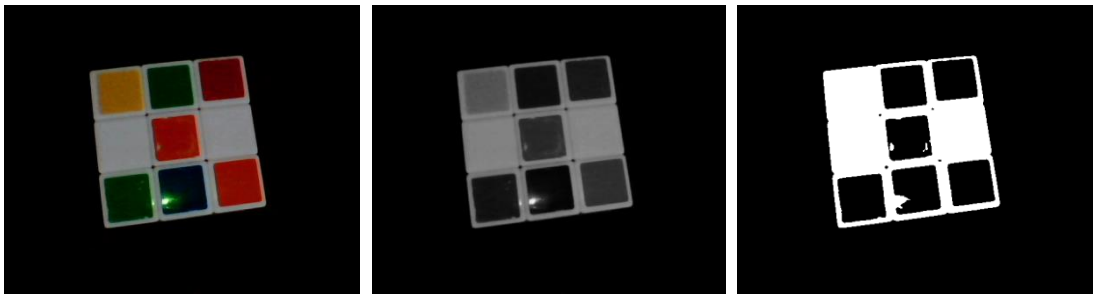
Сканирование и распознание цветов

Проблема распознавания цвета К.Р. является одним из важных этапов его сборки. Для её решения оператор затрачивает усилие – концентрируется на определенной точке и заполняет соответствующие метки.

Так как сканирование при помощи стандартного сенсора NXT занимает много времени (около 5 минут), было принято решение использовать в качестве считывающего устройства веб-камеру (сокращает время сканирования до 2 минут). Скорость сканирования повышается, поскольку сканируется сразу вся сторона кубика, а не каждый кусочек по отдельности.

В работе была использована камера LogitechWebcam c200. Данная камера имеет разрешение 640x480 пикселей и матрицу в 0,3 Мегапикселя. Несмотря на то, что камера имеет стандарт 480p и соотношение сторон 4:3, снимки при хорошем освещении не содержат шумов и прочих артефактов. Камера имеет встроенный чип, который отвечает за баланс белого, иными словами, для того чтобы снять видео или сделать фотографию, веб-камера должна получить несколько кадров для определения световой обстановки. Экспериментальным путем было выявлено, что камера при более низком разрешении (320x240) получает картинку быстрее, и тем самым баланс белого настраивается гораздо быстрее.

Процесс обработки изображения показан на рисунке 5.



а) б) в)

Рисунок 5 – а) исходное изображение, б) изображение после фильтра Гаусса, в) изображение, разбитое на 2 цвета

Из исходного изображения 5а получается монохромное изображение. Потом изображение претерпевает размытие по Гауссу в целях избавления от шумов 5б. Далее из монохромного изображения, содержащего полутона серого цвета, строится изображение, содержащее только белый и черный цвета 5в.

В полученном изображении происходит поиск контуров, среди которых приемлемыми считаются только те, которые превышают 10–20 % исходного изображения. Наилучшие контуры повергаются процессу аппроксимации, и таким образом контур содержит четыре точки вершины грани К.Р., рисунок 6.

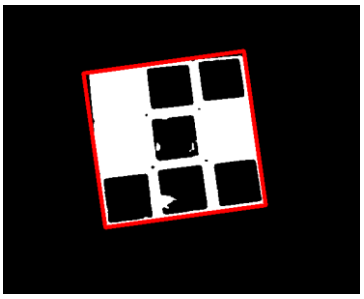


Рисунок 6 – Контур вокруг грани К.Р.

Поскольку контур имеет трапециевидную форму, и изображение может быть наклонено, происходит поиск матрицы перспективного преобразования, т. е. изображение в контуре перспективно приводится к квадратному изображению 300 на 300 пикселей, рисунок 7.

Рисунок 7 – Вырезанное изображение грани К.Р.

Полученное изображение делится на девять квадратных изображений размером 100 на 100 пикселей и помечается.

Кластеризация цветов

После того как К.Р. был отсканирован и соответствующим меткам присваиваются результаты сканирования, необходимо определить, какие метки представляют один цвет.

В стандартном пространстве цветов RGB из пакета sklearn [4] был задействован метод kmean (метод k-средних) [5]. В данном методе явно указывается число кластеров, подходящее для работы с небольшими объемами данных. В данном случае число кластеров = 6, так как у куба всего 6 граней и в каждом кластере должно быть 9 цветов. В качестве меры расстояния была выбрана стандартная мера похожести цветов, определенная как стандартное евклидово расстояние [6].

После нескольких тестирований выяснилось, что выбранный способ время от времени не может распределить белый и желтый цвета, а также красный и оранжевый. Таким образом, алгоритм не мог отнести от двух до пяти меток в соответствующие классы.

Также были протестированы алгоритмы – Spectral clustering, Wardhierarchical clustering, Agglomerative clustering. В них также можно явно задать число кластеров, но по сравнению с kmean они с задачей справлялись хуже – от семи до двенадцати меток были не распределены.

Для повышения надежности сканирования было принято решение преобразовывать исходные результаты сканирования в цветовое пространство HSV (Hue, Saturation, Brightness – тон, насыщенность, яркость).

При изменении освещения тон меняется незначительно и цвет, представленный в данном пространстве, устойчив к внешней обстановке. При переходе выходит, так что три составляющие цвета преобразуются только в одну, и фактически работа идет только с одной координатой вектора – шкалой оттенков, рисунок 8.

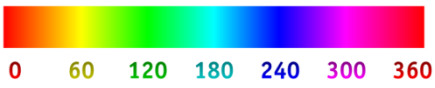


Рисунок 8 – Шкала оттенков

В данном случае экспериментальным путем были получены тоновые интервалы для каждого из 6 цветов куба.

Полученный алгоритм кластеризации работает следующим образом.

1. Определяется центральная опорная метка.
2. Из доступных цветов берутся все цвета, которые попадают в тоновый интервал, и происходит проверка на их количество.
3. Если число меток у одного цвета набирается в точности 9, то алгоритм переходит к следующему цвету, если же число меток не набирается в нужном количестве, то сканирование осуществляется повторно.

Сканирование белого и черного цветов происходит в связи с тем, что у белого цвета минимальна насыщенность, а у черного цвета – яркость. Если же в К.Р. присутствуют белый или черный цвет, то они сканируются в первую очередь.

Если кластеризация на основе k-средних в пространстве RGB или HSV проходит успешно, тогда можно не переходить к тональным интервалам в пространстве HSV. Подбор параметров в методе может увеличить вероятность успешного сканирования.

Алгоритмы «Бога» сборки К.Р.

Для демонстрации работы робота мы будем использовать алгоритм «Бога» сборки К.Р.

История поиска алгоритма «Бога» сборки К.Р. началась не позже 1980 года, когда открылся список рассылки для любителей К.Р. С тех пор математики, программисты и просто любители стремились найти алгоритм «Бога» – алгоритм, который бы позволил на практике решать К.Р. за минимальное число ходов. С этой проблемой была связана проблема определения числа «Бога» – числа ходов, всегда достаточного для сборки головоломки.

В июле 2010 года программист из Пало-Альто Томас Рокики, учитель математики из Дармштадта Герберт Коцемба, математик из Кентского университета Морли Дэвидсон и инженер компании Google Inc. Джон Детридж доказали, что каждая конфигурация К.Р. может

быть решена не более чем за 20 ходов. При этом любой поворот грани считался одним ходом. Таким образом, число «Бога» в метрике FTM оказалось равно 20 ходам.

Достаточно легко показать, что существуют разрешимые конфигурации, которые не могут быть решены менее чем в 17 ходов в метрике FTM или 19 ходов в метрике QTM.

Эту оценку можно улучшить, принимая во внимание дополнительные тождества, например, коммутативность поворотов двух противоположных граней (L R = R L, L2 R = R L2 и т. д.) Подобный подход позволяет получить нижнюю оценку для числа Бога, равную 18f или 21q.

Одной из конфигураций, для которой не удавалось найти короткое решение, был так называемый «суперфлип» (англ.), или «12-флип». Всё дело в том, что это была первая обнаруженная конфигурация, находящаяся на расстоянии 20f от начальной. «Суперфлип» представляет собой конфигурацию, в которой все угловые и рёберные кубики находятся на своих местах, но каждый рёберный кубик ориентирован противоположно. Оценка 20f в течение многих лет оставалась наилучшей известной. Кроме того, она вытекает из неконструктивного доказательства, так как оно не указывает конкретный пример конфигурации, требующей для сборки 18f или 21q.

Вершина, отвечающая «суперфлипу» в графе К.Р., является локальным максимумом: любой ход из этой конфигурации уменьшает расстояние до начальной конфигурации. Это дало основание предположить, что «суперфлип» находится на максимальном расстоянии от начальной конфигурации, то есть является глобальным максимумом.

В 1992 году Дик Т. Винтер нашёл решение «суперфлипа» в 20f. В 1995 году Майкл Рид доказал оптимальность этого решения, в результате чего нижняя оценка числа «Бога» стала равной 20 FTM.

Для получения комбинаций сборки алгоритмом «Бога» программное обеспечение обращается к сайту rubiks-cube-solver.com. Данный сайт частично написан на js из-за чего необходимо некоторое время ожидать подгрузки данных (~3 секунды). Передача данный на сайт происходит путем отправки запроса /solution.php?cube=0" + fc, где fc представляет из себя строку из цифр, определяющих цвет сторон кубика, полученных после кластеризации.

После соединения и загрузки готового алгоритма, мы начинаем поиск по странице. Если мы находим блок с «id = algoritmusHanyadik1», то понимаем, что алгоритм отработал правильно и выдал ответ, считываем из элемента данные с ответами на ПК. Преобразовываем полученный html код из кодировки Unicode в utf-8, а затем в команды для робота.

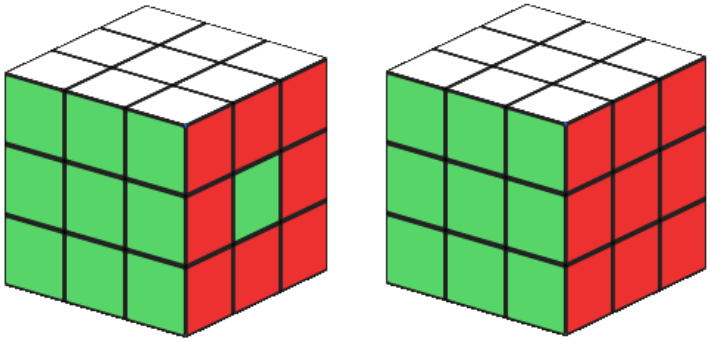
Тестирование

При разработке комплекса в первую очередь был написан класс Cube, который бы мог симулировать виртуальный кубик Рубика.

После того как были написаны модули оптимизации и сборки были созданы два модуля для тестирования класса. Первый модуль тестирования класса Cube работает следующим образом.

1. Создается виртуальный перемешанный куб и включается таймер отчета.
2. Программа на основе меток генерирует последовательность команд и исполняет их.
3. В конце выводится время выполнения, число ходов, процент успеха, число неудач.

Второй модуль использует фреймворк unitest и автоматически тестирует все методы класса. Фактически фреймворк проверяет каждый метод на его корректность выполнения. Так же тестируется возможность того что метки расставлены неправильно и данный куб не может существовать, рисунок 9.



а) б)

Рисунок 9 – Варианты меток К.Р. а) невозможный К.Р., б) возможный К.Р.

После того как был написан алгоритм сборки кубика и реализовано автоматическое управление при помощи selenium был написан соответствующий модуль тестирования, работающий следующим образом:

1. Программа пытается найти драйвера. В случае успеха импортирует их, а в случае неудачи выводит сообщение, что драйвера не найдены.
2. По желанию пользователя или в автоматическом режиме создается конфигурация К.Р.
3. Программа открывает веб-браузер, с которым ассоциирован драйвер, и отправляет запрос на сервер.
4. Selenium ожидает полной загрузки определенного элемента. В случае успеха сохраняет его и закрывает браузер. Если же время ожидания слишком велико (порядка пары минут) то выводится сообщение об ошибке.
5. Загруженный элемент из Unicode преобразуется в ASCII и выводится на экран.

Заключение

Полученный алгоритм компьютерного зрения обработки изображений и кластеризации цветов может получить применение в областях робототехники и беспилотных аппаратов. Использование цветового пространства HSV позволяет перейти к использованию тональных интервалов вектора цвета и не рассматривать насыщенность и яркость, как в случае с RGB.

Дополнительно спроектирован и создан робот, использующий компьютерное зрение для сборки кубика Рубика. Разработано программное обеспечение передачи данных на сетевой ресурс для обработки и передачи инструкций роботу. Данная работа обеспечивает 100 % сборку кубика Рубика в пределах 10 минут, что в рамках поставленной задачи не критично.

Литература

1. OpenCV-Python Tutorials. – Текст : электронный // OpenCV 3.0.0-dev documentation : сайт. – URL: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html (дата обращения: 25.11.2020).

2. NumPyReference. – Текст : электронный // NumPy v1.19 Manual. – URL: https://docs.scipy.org/doc/numpy/reference/ (дата обращения: 25.11.2020).

3. MindCuber for EV3 and NXT. – Текст : электронный // MindCuber. – URL: http://mindcuber.com/ (дата обращения: 25.11.2020).

4. Clustering. – Текст : электронный // Scikit-learn 0.23.2 documentation. – URL: http://scikit-learn.org/stable/modules/clustering.html (дата обращения: 25.11.2020).

5. Мюллер, А. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными / А. Мюллер, С. Гвидо. – Москва : Вильямс, 2017. – 480 с. – ISBN 978-5-9908910-8-1. – Текст : непосредственный.

6. Color difference. – Текст : электронный // Wikipedia. – URL: https://en.wikipedia.org/wiki/Color_difference (дата обращения: 25.11.2020).