**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА (РУТ (МИИТ))»**

**Институт транспортной техники и систем управления**

**Кафедра «Путевые, строительные машины и робототехнические комплексы»**

**ДОПУСТИТЬ К ЗАЩИТЕ**

**Заведующий кафедрой**

**Неклюдов А.Н. \_\_\_\_\_\_\_\_\_\_**

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

**БАКАЛАВРСКАЯ РАБОТА**

**на тему: «Разработка системы управления промышленным манипулятором на базе машинного зрения»**

Направление подготовки 15.03.06 «Мехатроника и робототехника»

Магистерская программа «Роботы и робототехнические системы»

**Обучающийся Дырдин А.И.**

**Научный руководитель Мишин А.В.**

**Москва 2019 г.**

Оглавление

[ВВЕДЕНИЕ 2](#_Toc10303519)

[1. ОПИСАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ 2](#_Toc10303520)

[1.1 Описание элементов, входящих в систему управления. 2](#_Toc10303521)

[1.2 Описание системы управления до модификации 3](#_Toc10303522)

[1.3 Описание модифицированной системы управления 4](#_Toc10303523)

[2. РАЗРАБОТКА АЛГОРИТМА РАСПОЗНАВАНИЯ ПОЛОЖЕНИЯ ОБЪЕКТА 5](#_Toc10303524)

[2.1 Обзор алгоритмов анализа изображений 5](#_Toc10303525)

[2.2 Описание алгоритма распознавания положения объекта 6](#_Toc10303526)

[2.3. Переход к системе координат робота 11](#_Toc10303527)

[3. ОПИСАНИЕ РАЗРАБОТАННОЙ ПРОГРАММЫ LOTUS 12](#_Toc10303528)

[3.1 Системные требования. 12](#_Toc10303529)

[3.2 Описание файлов исходного кода. 12](#_Toc10303530)

[3.3 Описание графического интерфейса программы. 14](#_Toc10303531)

[ВЫВОДЫ 16](#_Toc10303532)

[ЗАКЛЮЧЕНИЕ 16](#_Toc10303533)

[СПИСОК ЛИТЕРАТУРЫ 17](#_Toc10303534)

[ПРИЛОЖЕНИЕ А 18](#_Toc10303535)

[Листинг А.1 – Класс «Recognition1» 18](#_Toc10303536)

[Листинг А.2 – Класс «Recognition2» 21](#_Toc10303537)

[Листинг А.3 – Класс «Form1» 26](#_Toc10303538)

[ПРИЛОЖЕНИЕ Б 66](#_Toc10303539)

# ВВЕДЕНИЕ

Преимущества автоматизации технологического процесса неоспоримы. Она повышает объём производства и качество продукции. Задача автоматизации технологического процесса подразумевает автоматизацию его составных частей – технологических операций. В данной работе предложена система управления роботом, решающая задачу автоматизации перемещения. Ведь именно операция перемещения является неотъемлемой частью любого технологического процесса.

Наиболее универсальный путь автоматизация операции перемещения – использование промышленного робота-манипулятора, оснащённого системой машинного зрения. Такое решение позволяет сократить затраты на переналадку и переоборудование производственной линии при смене параметров продукции.

Внедрение данной технологии невозможно без разработки системы управления, позволяющей манипулятору работать автономно. Таким образом целью данной работы является разработка системы управления промышленным роботом на базе машинного зрения.

Далее представлен цикл перемещения объекта производства (далее просто объекта).



В качестве экспериментальной установки для тестирования разработанной системы управления был использован робот компании KUKA модели KR6 R700 (AGILUS) под управлением контроллера KR C4 Compact.

Причины выбора среды RoboDK

1. Наличие ограниченной бесплатной лицензии.
2. Возможность онлайн программирования роботов от большинства производителей.
3. Наличие программного интерфейса приложения (API).
4. Документация в доступном формате на английском языке.

# 1. ОПИСАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ

## 1.1 Описание элементов, входящих в систему управления.

**RoboDK –** среда оффлайн программирования, симуляции и онлайн программирования роботов.

**RoboDK C# API –** библиотека взаимодействия со средой RoboDK на языке программирования C#.

**Lotus** – программа, разработанная в ВКР и выполняющая следующие функции:

1. Взаимодействие с подключёнными устройствами видеосъёмки и настройка параметров видеопотока.
2. Распознавание объектов и определение их координат.
3. Взаимодействие со средой RoboDK.
4. Предоставление пользовательского графического интерфейса для регулировки параметров распознавания и параметров перемещения.
5. Отображение текущего процесса распознавания объектов.
6. Интеграция трёхмерной среды для симуляции перемещения робота.

**KUKAVarProxy –** программа, принимающая команды на чтение и запись системных переменных контроллера робота от удалённого компьютера.

**apikuka.exe** – драйвер робота служит интерфейсом взаимодействия RoboDK с программой KUKAVarProxy.

**Системные переменные –** список переменных разных типов, хранящийся в файле, расположенном на жёстком диске контроллера.

**Программа синхронизации ­-**  программа, написанная на языке KRLи запускаемая через человеко-машинный интерфейс контроллера. Предназначена для чтения системных переменных и взаимодействия с ядром контроллера**.**

**Робот –** промышленный манипулятор, обладающий шестью степенями свободы. При отладке системы управления использовалась модель KUKA KR6 R700. Подключён к контроллеру робота по интерфейсу передачи данных X21 и к порту питания приводов X20.

**Контроллер –** система непосредственного управления промышленным манипулятором. В работе была использована модель KUKA KR C4.

**Рабочая зона –** плоская поверхность, находящаяся в непосредственной близости с роботом. Имеет прямоугольную форму.

**Камера** – видеокамера, подключённая по интерфейсу USB2.0 к промышленному компьютеру.

**Оператор** – человек, производящий пуск и наладку системы.

## 1.2 Описание системы управления до модификации

Производитель робототехнических систем KUKA имеет собственную систему программного и аппаратного обеспечения, позволяющую автоматизировать операции перемещения, опираясь на данные видеонаблюдения. Называется эта система KUKA.VisionTech. Далее будет рассмотрена её структура, преимущества и недостатки.

Система KUKA.VisionTech подразумевает схему управления, включающую в себя контроллер, манипулятор, камеру KUKA MXG20, рабочую зону с объектами манипулирования и оператора.

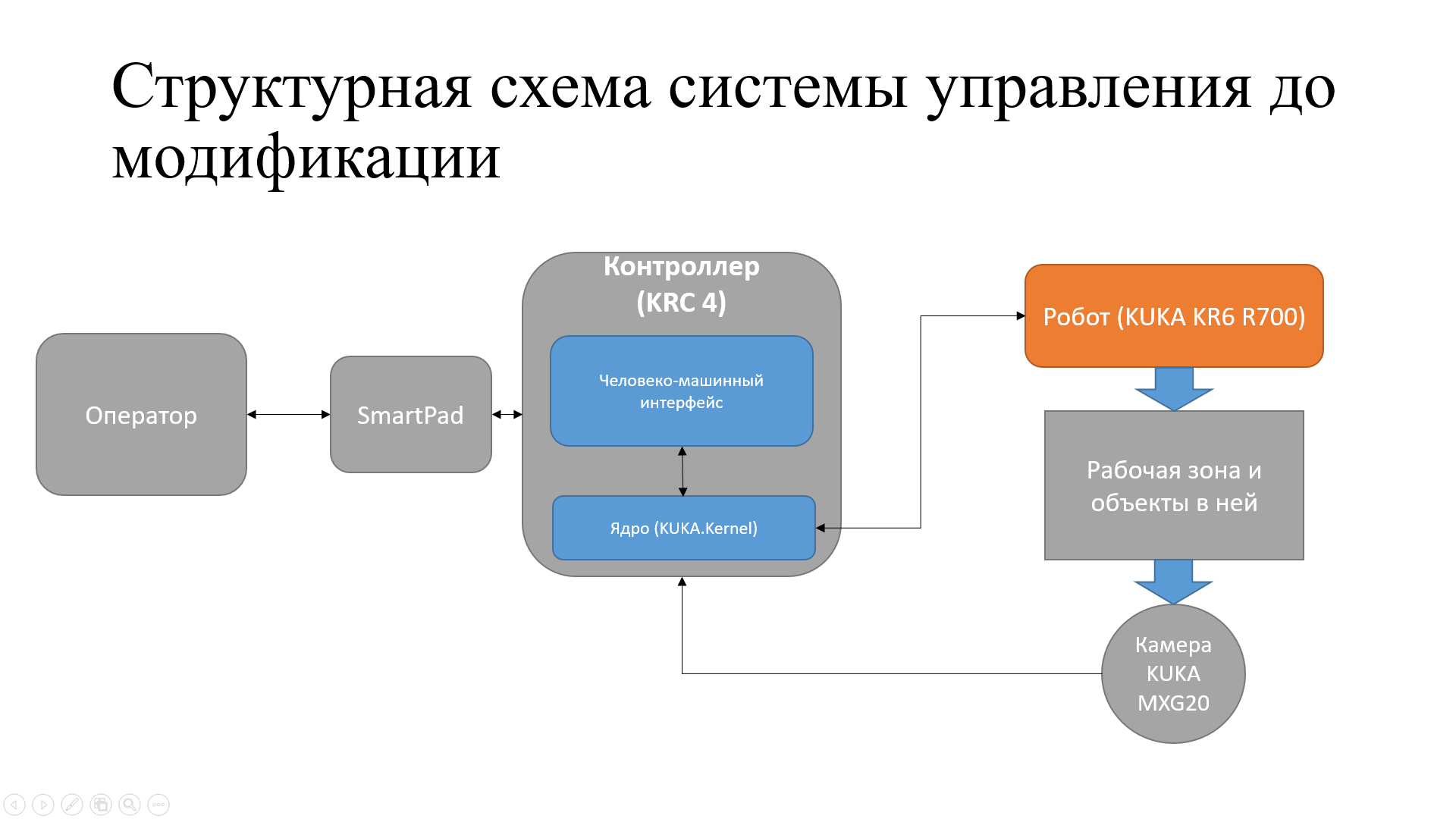
Перемещение робота описывается программой, написанной на языке KRL и запущенной на контроллере. По ходу выполнения этой программы производится распознавание координат объектов. Алгоритм распознавания определён производителем и не может быть модифицирован. Калибровка алгоритма производится через пульт человеко-машинного интерфейса SmartPad.

Преимущества KUKA.VisionTech:

* Все компоненты программного обеспечения разработаны одним производителем.
* За счёт использования меньшего числа составных элементов, данная система обладает большей надёжностью.
* Система протестирована на производстве.

Недостатки KUKA.VisionTech:

* Невозможность модификации алгоритма распознавания.
* Потребление алгоритмом вычислительных ресурсов контроллера.
* Высокая стоитмость совместимого оборудования.
* Единственная совместимая камера - KUKA MXG20.

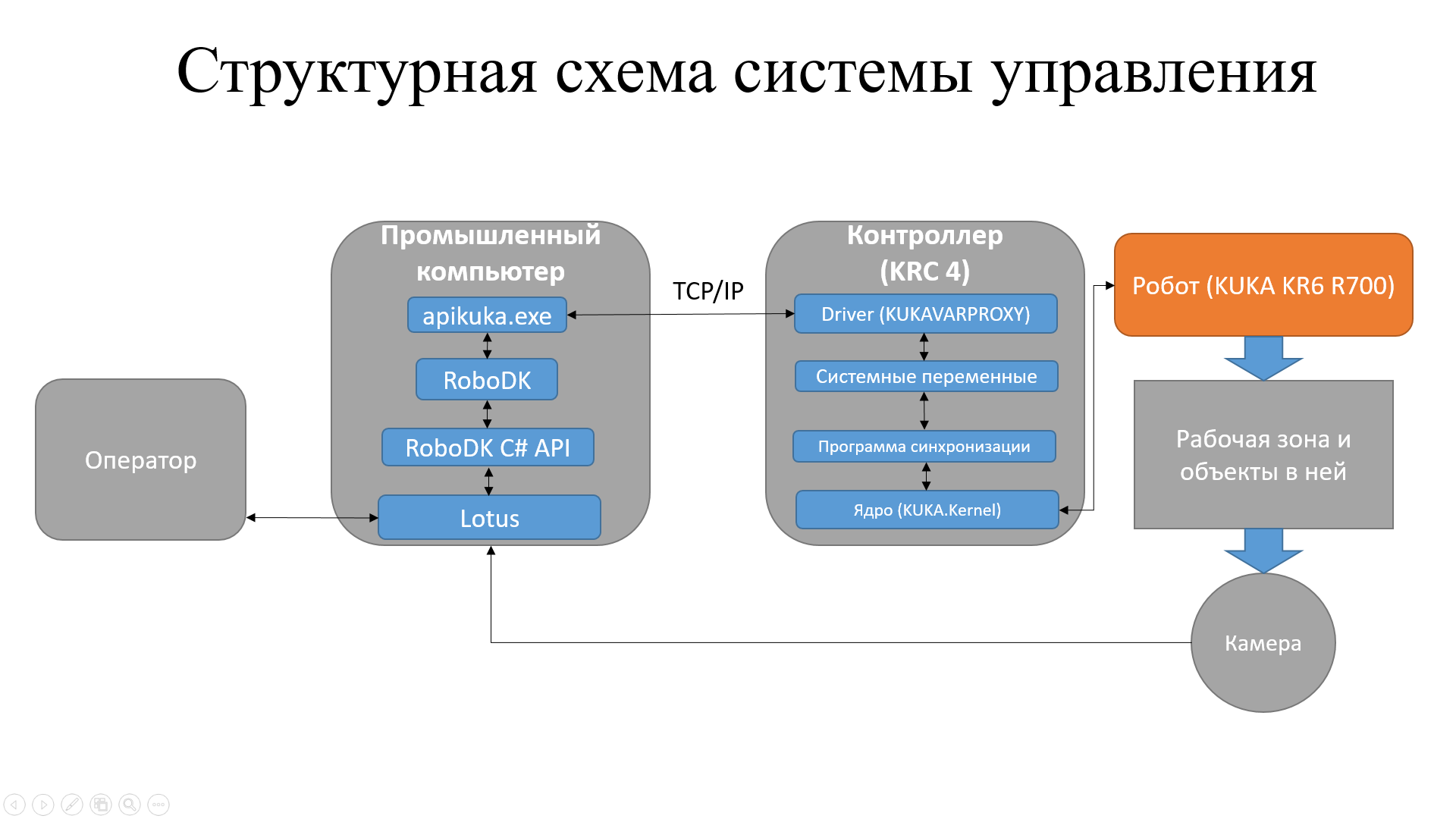


**Рис. . Структурная схема системы управления до модификации.**

## 1.3 Описание модифицированной системы управления

Модифицированнаяструктурная схема системы управления включает в себя контроллер, манипулятор, промышленный компьютер, рабочую зону с объектами манипулирования, камеру и оператора, управляющего системой.

Позиционирование робота начинается с захвата изображения рабочей зоны камерой с помощью программы Lotus, далее, посредством анализа изображения, распознаются координаты объектов манипуляции. После того, как координаты объектов были перенесены в систему координат робота, программа Lotus отправляет команду библиотеке RoboDK C# API на взаимодействие со средой RoboDK [2]. Далее среда RoboDK взаимодействует по интерфейсу Ethernet и протоколу TCP/IP [1] с сервером редактирования системных переменных контроллера KUKAVarProxy [3]. После того, как системные переменные контроллера были изменены извне, программа синхронизации переменных, запущенная через человеко-машинный интерфейс на контроллере, считывает значение этой переменной и отправляет команду в ядро контроллера KUKA Kernel на изменение положения манипулятора.



**Рис. . Структурная схема системы управления**

Преимущества разработанной схемы:

* Экономия вычислительных ресурсов контроллера.
* Неограниченные возможности разработки алгоритма распознавания образов.
* Выбор камеры зависит лишь от специфики автоматизируемой операции. Это позволяет использовать как более дешёвое оборудование, так и более совершенное.

Недостатки:

* Большее число составных элементов системы снижает её надёжность.

# 2. РАЗРАБОТКА АЛГОРИТМА РАСПОЗНАВАНИЯ ПОЛОЖЕНИЯ ОБЪЕКТА

## 2.1 Обзор алгоритмов анализа изображений

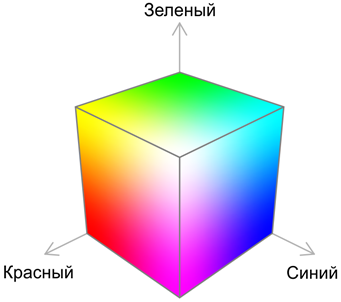
## 2.2 Описание алгоритма распознавания положения объекта

Этап калибровки.

На этапе калибровки сохраняется информация о положении рабочей зоны в кадре. Оно задаётся четырьмя точками, которые соответствуют вершинам прямоугольной зоны, в которой позже будут появляться объекты и будут распознаваться их координаты. Во время калибровки в рабочей области должны отсутствовать объекты манипулирования, иначе точность распознавания будет снижена.

Также на этом этапе происходит сохранение маски фона рабочей области и среднего цвета этой маски.

Если представить пиксели изображения, как множество точек, у которых есть цвет, полученный в результате сложения трёх цветовых компонент, то такое множество можно рассматривать в трёхмерном пространстве этих трёх признаков. Координату в данном пространстве определяет интенсивность каждой из трёх цветовых компонент. Расстоянием между точками, которые являются некоторыми цветами, в таком пространстве является отличие между этими цветами. Чем больше абсолютное значение расстояния от точки А до точки B в данном пространстве признаков, тем больше отличие цвета A от цвета B.

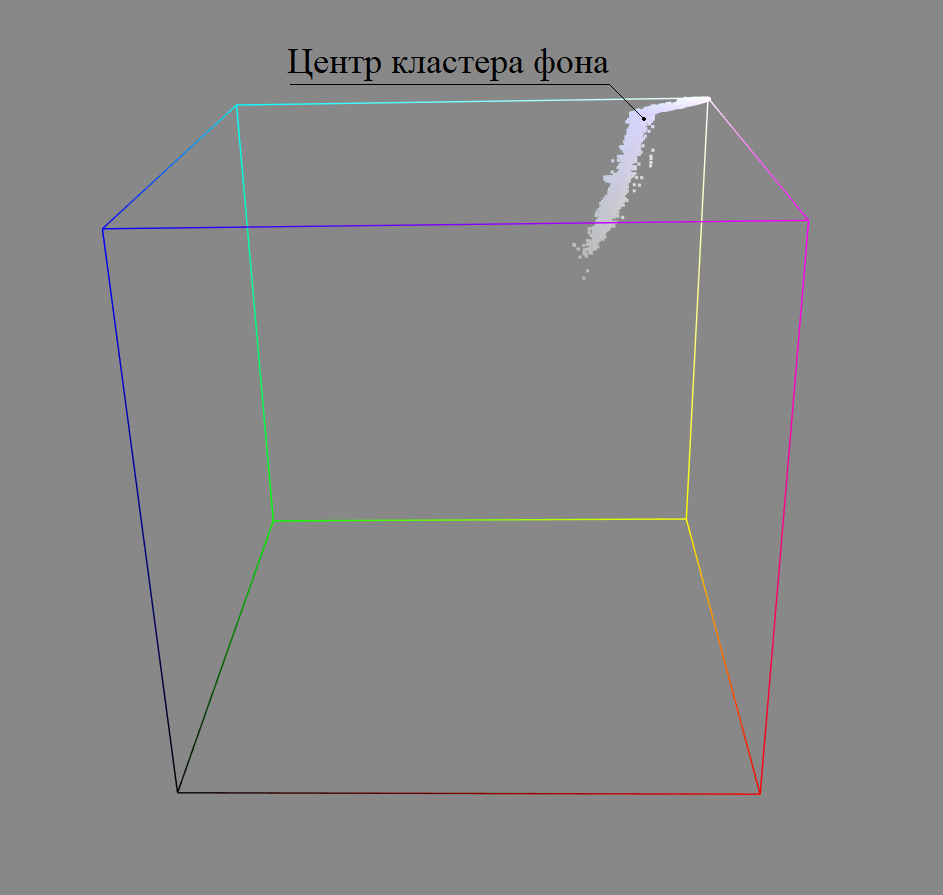


**Рис. . Модель цветового пространства.**

Таким образом средний цвет маски фона будет являться центром кластера его пикселей в пространстве признаков.



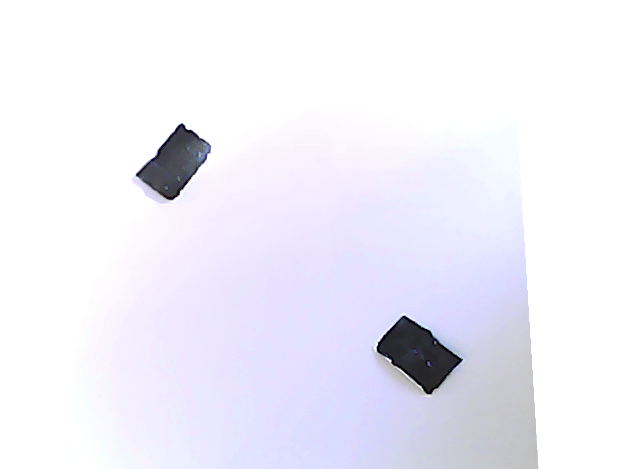
**Рис. . Маска фона рабочей области изображения.**



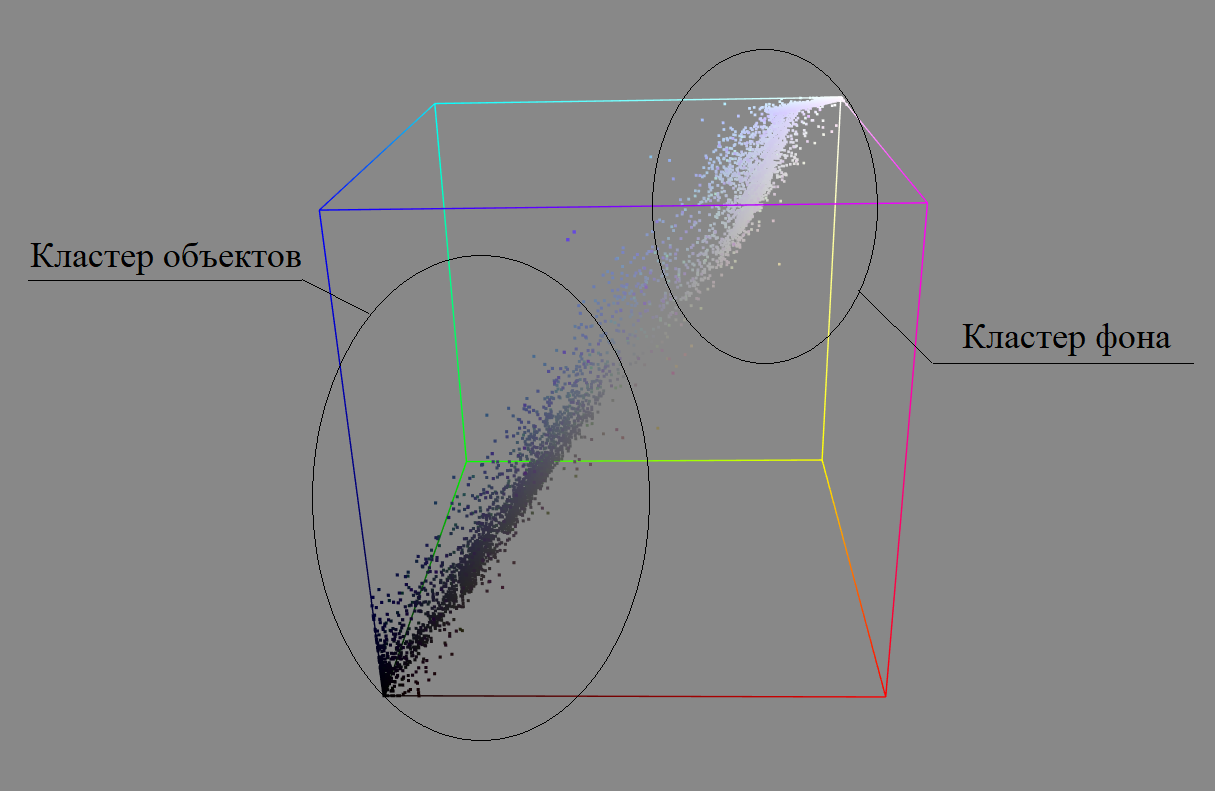
**Рис. . Визуализация маски фона в цветовом пространстве.**

Этап распознавания.

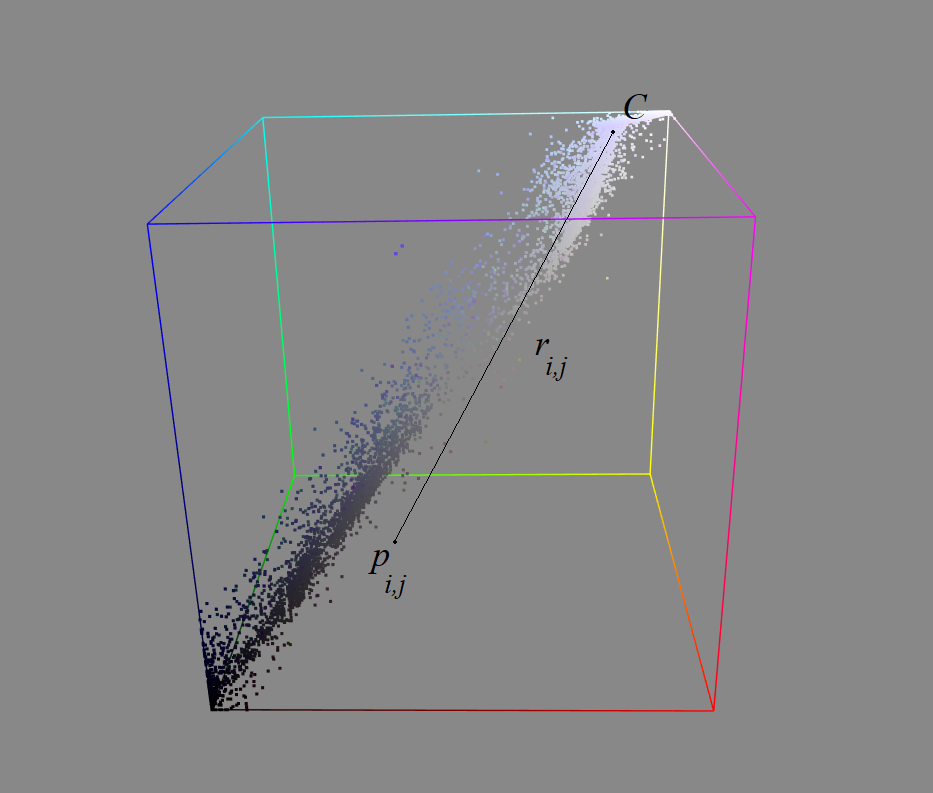
После того, как была получена информация о фоне рабочей области начинается распознавание положения объекта: программа Lotus получает видеопоток с камеры по последовательному интерфейсу USB 2.0. Далее из видеопотока захватывается изображение, которое сравнивается с маской.



**Рис. . Маска с объектами в ней.**



**Рис. . Визуализация маски с объектами в цветовом пространстве.**



**Рис. . Определение .**

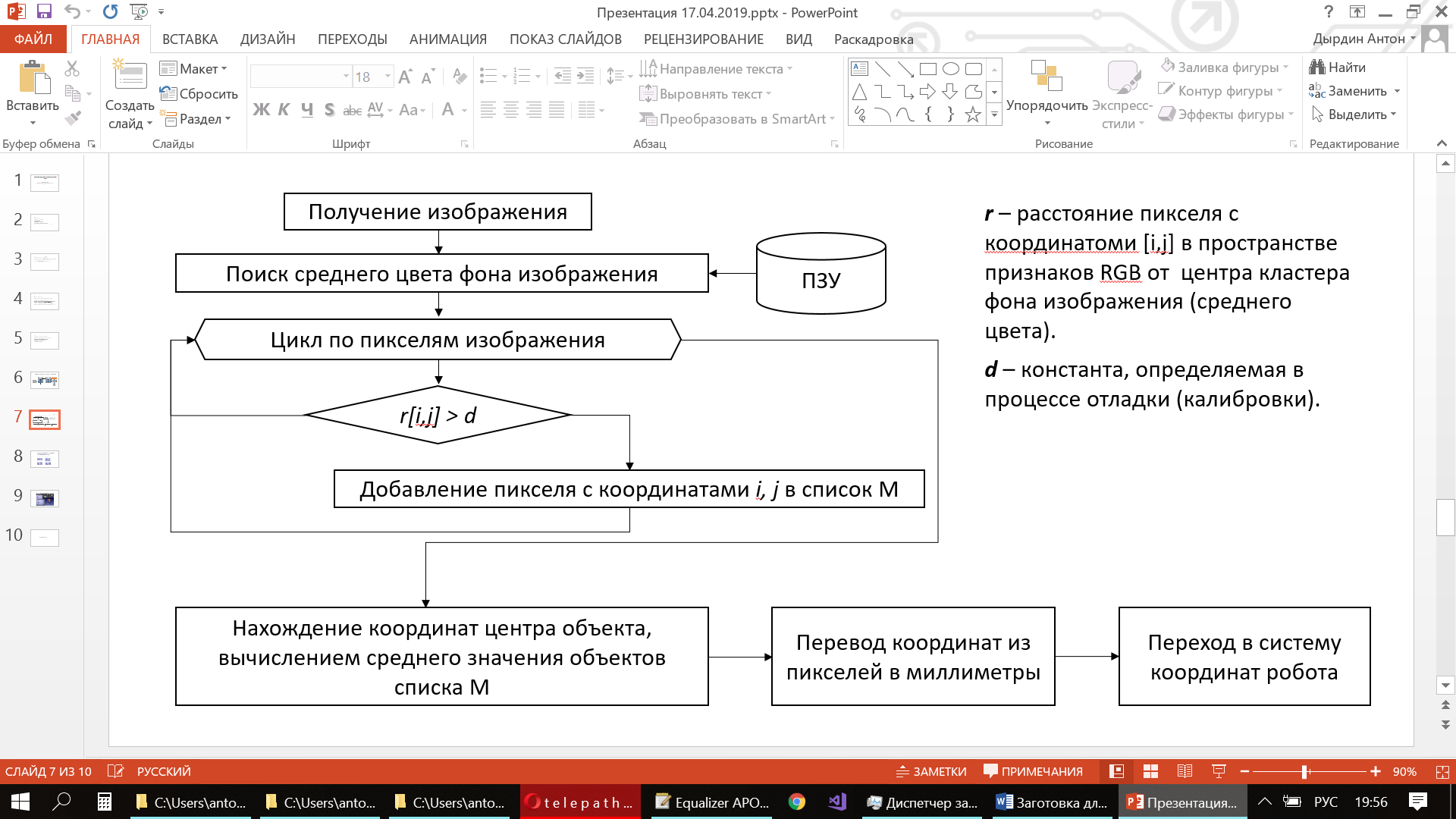
Для каждого пикселя , входящего в маску рабочей области выполняется проверка удалённости его цвета от цвета фона ***C***. Если больше, чем константа ***d*** , значение которой выбирается экспериментально (как правило оно находится в интервале [127;255]), то пиксель с индексами ***i,j*** указывается, как пиксель, принадлежащий множеству пикселей отображающих объект. После того, как множество всех пикселей маски разбито на два кластера («фон» и «объект»), определяются координаты (в пикселях) центра объекта вычислением среднего арифметического координат пикселей, принадлежащих кластеру объекта.

, где

- индекс столбца пикселей изображения, в котором находится центр объекта – координата ***y*** в пикселях,

- индекс строки пикселей изображения, в которой находится центр объекта – координата ***x*** в пикселях

N – количество пикселей, принадлежащих кластеру «объект».

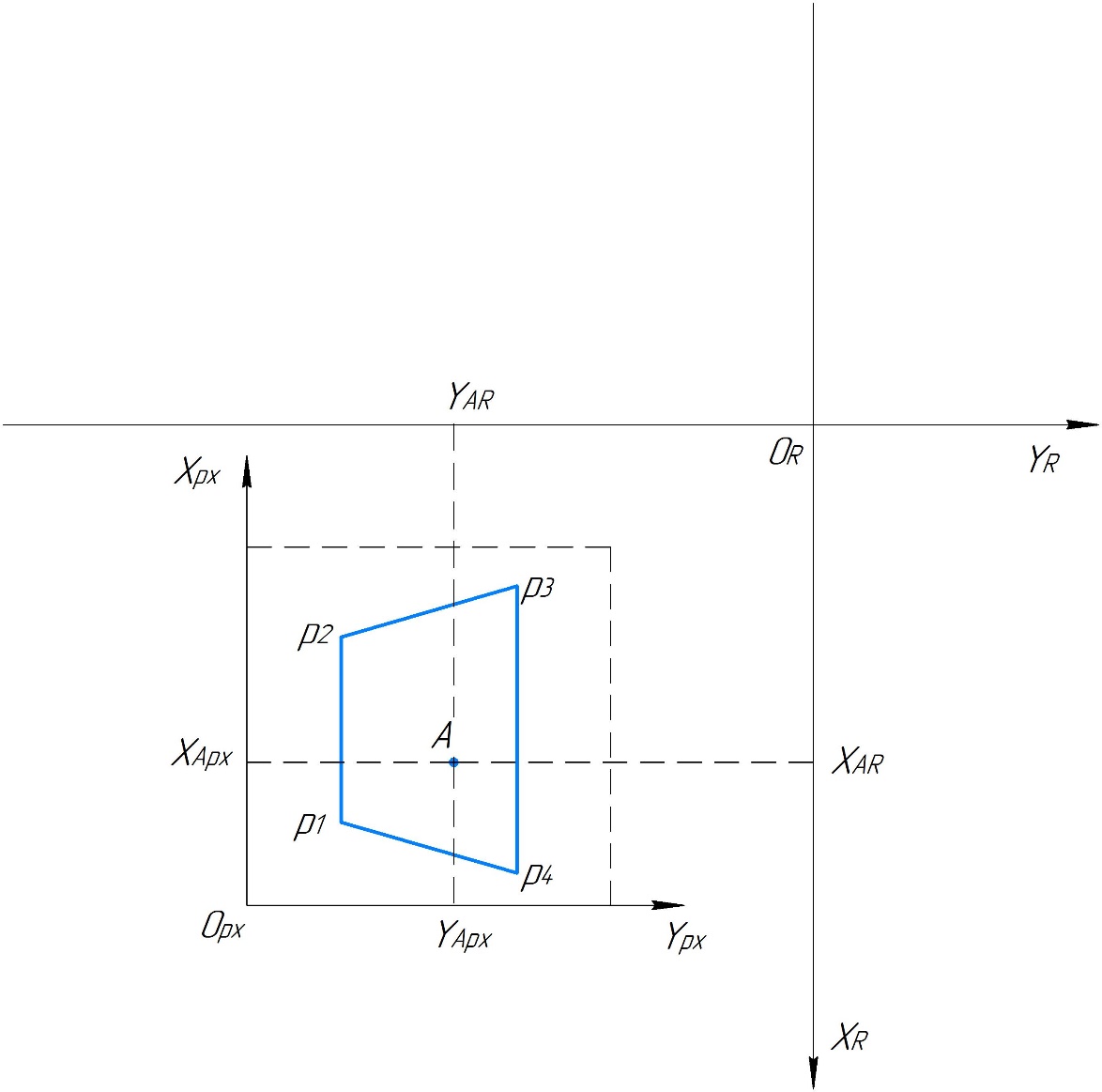


**Рис. . Блок-схема алгоритма распознавания координат одного объекта.**

Модификация алгоритма для распознавания нескольких объектов, находящихся в рабочей области одновременно.

Для распознавания координат нескольких объектов была введена константа objectSize, определяющая средний размер объектов. При этом слишком большое значение даст ошибку при распознавании координат близко расположенных объектов, а слишком малое приведёт к тому, что один объект будет определяться, как два или более. После кластеризации множества пикселей изображения, попавших в маску на два кластера, кластер «объект» делится методом ближайших соседей [5], но уже по координатам на кластеры «объект №1», «объект №2» … «объект n». Далее их координаты в пикселях определяются аналогично алгоритму для одного объекта.

## 2.3. Переход к системе координат робота

****

**Рис. . Система координат изображения *px* в системе координат робота *R*.**

На выходе вышеописанного алгоритма получаются координаты объекта в индексах стоки и столбца матрицы пикселей изображения. Чтобы получить положение объекта в системе координат робота, необходимо решить задачу перехода к новой системе координат.

Так как камера установлена стационарно под углом к прямоугольной рабочей области, то её контур в перспективе имеет форму трапеции . Зная реальные высоту *Sh* и ширину *Sw* рабочей области, координаты точек в системе координат *px*, координаты точки *p2* в системе координат робота и координаты точки *A* в системе координат *px*, задачу переноса координат объекта *А* из системы координат изображения *px,* где единицей измерения является пиксель, в систему координат робота *r*, где единицей измерения является миллиметр,можно представить, как задачу нахождения функций *f* и *g:*

Найденное решение:

– коэффициент перевода пикселей в миллиметры по горизонтали

– коэффициент перевода пикселей в миллиметры по вертикали

Слагаемое служит для компенсации трапецевидности прямоугольной области в кадре.

## 3. ОПИСАНИЕ РАЗРАБОТАННОЙ ПРОГРАММЫ LOTUS

## 3.1 Системные требования.

Целевая платформа - x86-64 (AMD64/Intel64/EM64T)

Совместимые операционные системы - Windows XP/ 7/ 8 /8.1 /10

Минимальная версия пакета библиотек - .NET Framework 4.

Центральный процессор Intel Pentium 4.

Объём ОЗУ – 2ГБ.

Видеокамера с интерфейсом USB 2.0 и разрешением не менее 320x240.

Адаптер интерфейса Ethernet.

## 3.2 Описание файлов исходного кода.

*App.config* – файл конфигурации приложения. Содержит XML текст.

*FastBitmap.cs* – класс, содержащий методы повышающий скорость взаимодействия с пикселями изображений.

*Form1.cs* – класс главной формы приложения. Содержит методы взаимодействия с видеокамерами, методы взаимодействия с RoboDK C# API, обработчики событий пользовательского графического интерфейса, цикл распознавания и методы, позволяющие запускать этот цикл в отдельном потоке, а также останавливать его.

*Form1.Designer.cs* – код, сгенерированный конструктором форм WindowsForms.

*Preview.cs* – форма, появляющаяся во время запуска главной формы и пропадающая после того, как главная форма была загружена.

*Preview.Designer.cs* - код, сгенерированный конструктором форм WindowsForms.

*Program.cs* – класс, содержащий точку входа в программу. В нём происходит вызов главной формы.

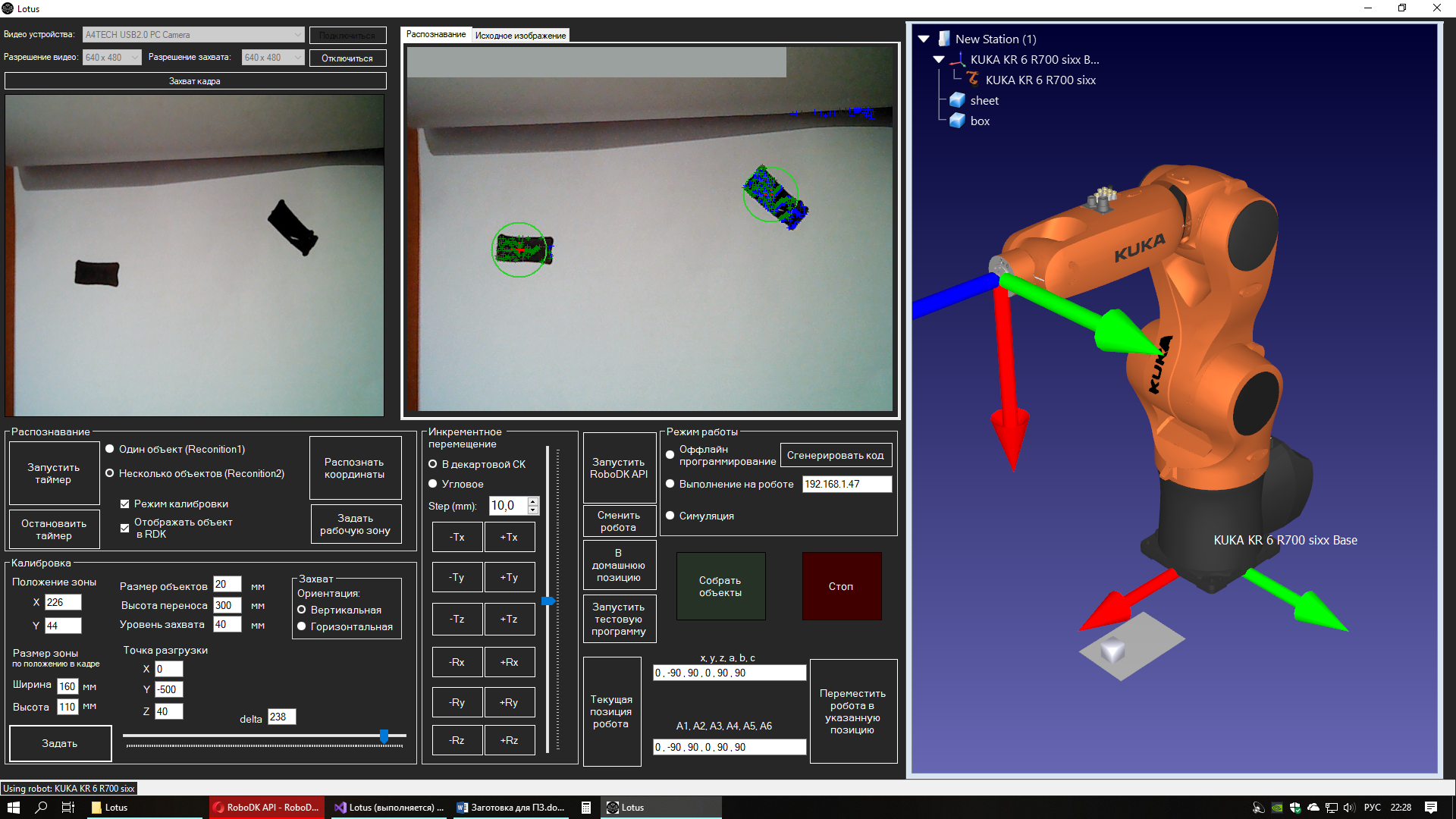
*Recognition1.cs* – класс, содержащий реализацию алгоритма распознавания одного объекта.

*Recognition2.cs* - класс, содержащий реализацию алгоритма распознавания одного и более объектов.

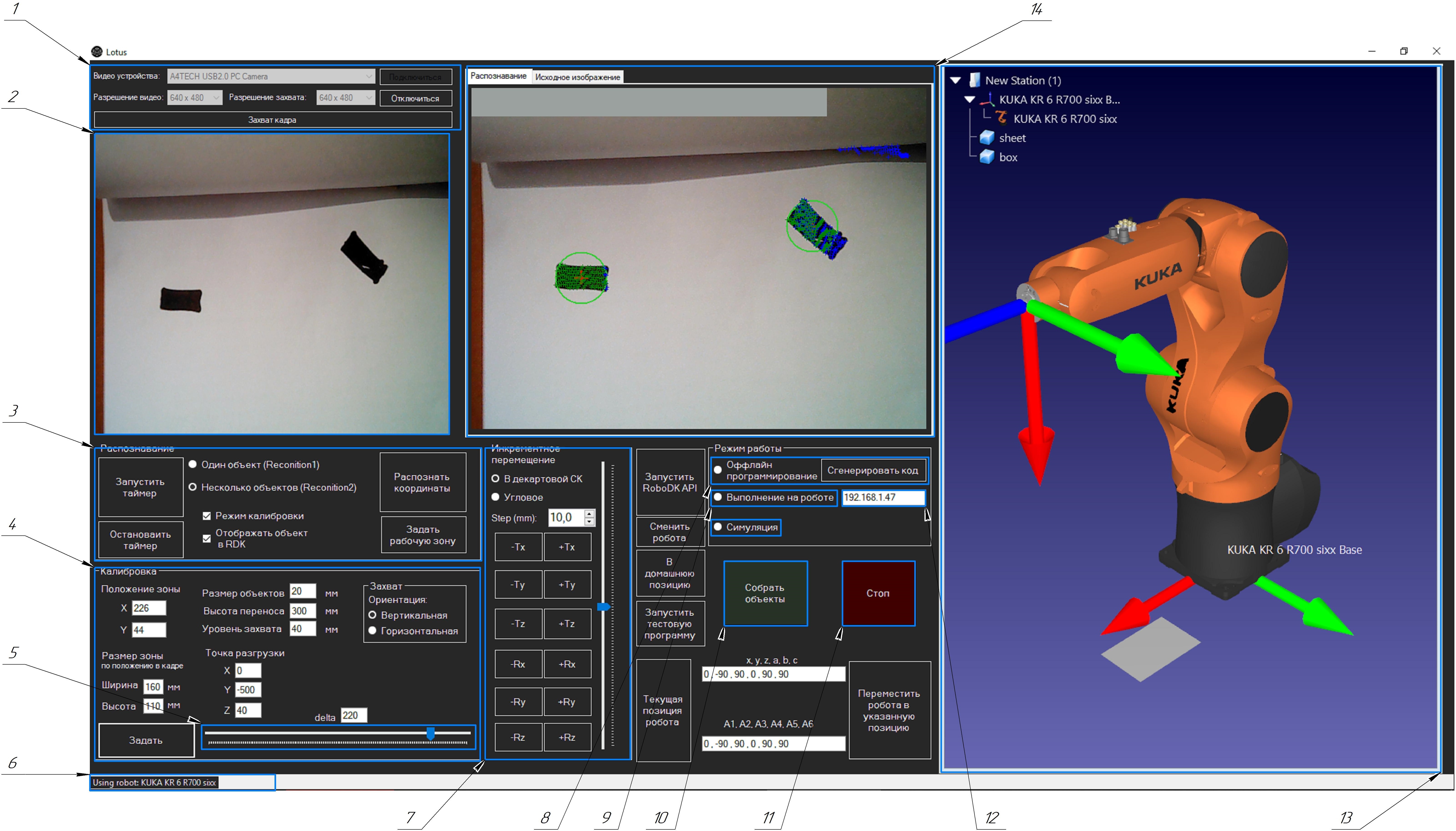
*RoboDK.cs* – исходный файл RoboDK C# API. Содержит класс RoboDK, класс RoboDK.Item и другие инструменты для робототехники, такие как Matrix для матричных операций для работы с преобразованиями позиций и координат.

*RobotControl.cs* – содержит код, определяющий логику перемещения объектов роботом.

## 3.3 Описание графического интерфейса программы.



**Рис. . Внешний вид пользовательского графического интерфейса.**



**Рис. . Графический интерфейс.**

1 – Элементы управления видеоустройствами,

2 – панель предпросмотра видеопотока,

3 – элементы управления процессом распознавания положения объектов,

4 – элементы управления параметрами захвата и перемещения объектов, координатами рабочей области относительно робота, размерами объектов и рабочей области,

5 – ползунок изменения параметра d, определяющего чувствительность алгоритма распознавания объектов,

6 – область уведомлений,

7 – элементы управления инкрементным перемещением,

8 – указатель перехода в режим оффлайн программирования и кнопка генерации кода.

9 – указатель перехода в режим онлайн программирования (альтернативного управления)

10 – кнопка, запускающая перемещение объектов в область выгрузки.

11 – кнопка остановки перемещения робота.

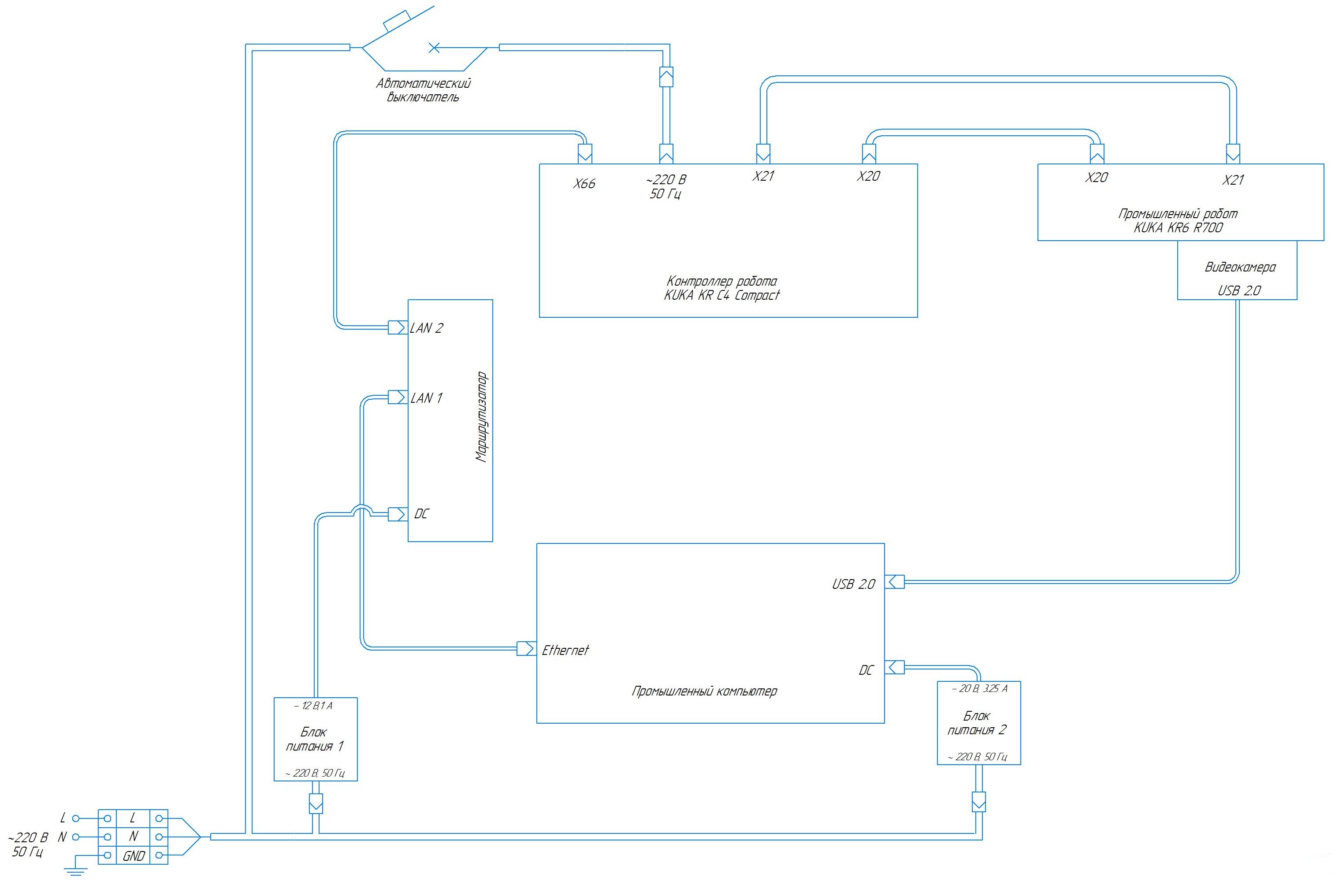
12 - поле ввода IP-адреса робота в сети предприятия.

13 – интегрированное окно, отображающее 3D модель робота, модели объектов и рабочей поверхности.

14 – панель, отображающая исходный видеопоток с наложенными на него эффектами, наглядно отображающими результаты распознавания.

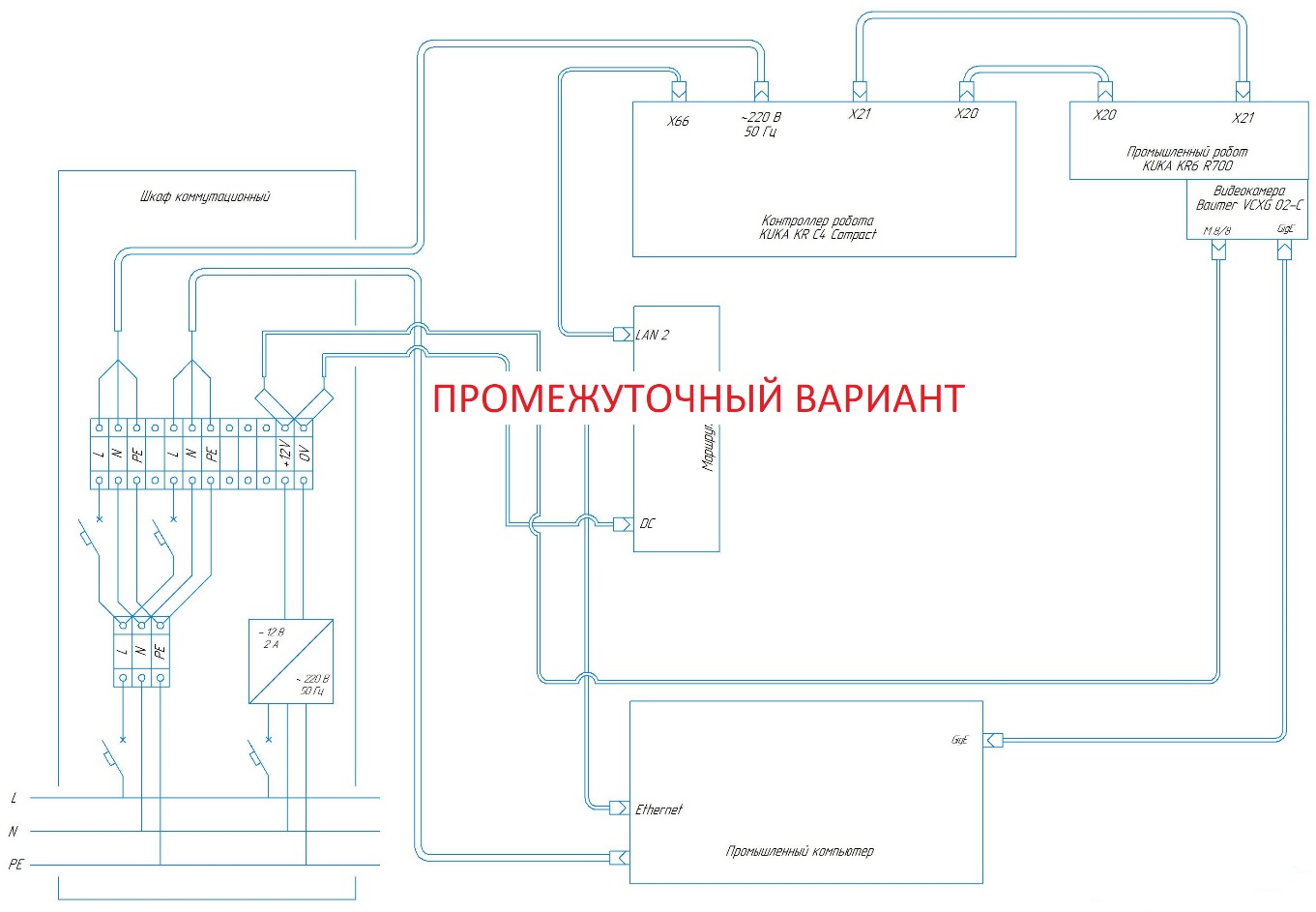
# ТЕСТИРОВАНИЕ И ОТЛАДКА

Для тестирования и отладки разработанной схемы был собран экспериментальный стенд, электросхема которого представлена на рисунке..



**Рис. . Электросхема экспериментального стенда.**

Данная схема не тождественна эксплуатационной схеме, которая представлена на рисунке..



**Рис. . Электросхема эксплуатационная.**

В отличие от эксплуатационной схемы в экспериментальную не вошло захватное устройство с его контроллером, а также коммутационный шкаф. В эксплуатационной схеме видеокамера подключена с использованием технологии GigE и имеет собственное питание, тогда как в экспериментальной и питание, и данные передаются по порту USB2.0 компьютера.

В ходе отладки были устранены ошибки программы Lotus, допущенные при написании её кода и модифицирован её графический интерфейс: добавлены новые элементы управления, исправлено некорректное отображение, реализовано автосохранение введённых параметров.

В процессе тестирования было установлено, что разработанный алгоритм машинного зрения чувствителен к теням, возникающих при ярком свете, падающем под углом к рабочей поверхности. Однако, при рассеянном свете или, когда лучи света падают перпендикулярно к рабочей поверхности, алгоритм демонстрирует высокую устойчивость к наличию в кадре посторонних предметов и низкой контрастности.

# ВЫВОДЫ

В работе была спроектирована структурная схема управления промышленным роботом на базе машинного зрения, проанализированы её особенности и отличие от схемы, предлагаемой производителем. Система управления была реализована, посредством совместного использования свободно распространяемого программного обеспечения и программы собственной разработки. Система была протестирована на лабораторном оборудовании, аналогичном промышленному. Были проведены замеры скорости выполнения рабочего цикла системы, а также точности позиционирования робота. Установлена случайная и систематическая погрешность распознавания положения объекта.

# ЗАКЛЮЧЕНИЕ

Разработанная система требует отладки в условиях, приближенных к эксплуатационным.

# СПИСОК ЛИТЕРАТУРЫ

1. Robot drivers [Online]// robodk [site] URL: https://robodk.com/doc/en/Robot-Drivers.html (date of request 10.04.2019).
2. RoboDK C# API [Online]// robodk [site] URL: <https://robodk.com/doc/en/RoboDK-API.html#CsAPI> (date of request 10.04.2019).
3. KUKAVarProxy open repository [Online]// github [site] URL: <https://github.com/ImtsSrl/KUKAVARPROXY> (date of request 10.04.2019).
4. AForgeNET framework documentation [Online]// aforgenet [site] URL: <http://www.aforgenet.com/framework/docs/> (date of request 10.04.2019).
5. Обзор алгоритмов кластеризации данных [Электронный ресурс]// habr [сайт] URL: <https://habr.com/ru/post/101338/>
6. Бабич, А. В. Промышленная робототехника / А.В. Бабич. - М.:, 2012. - 263 c.
7. Воскобойников, Б. С. Словарь по гибким производственным системам и робототехнике. Английский. Немецкий. Французский. Нидерландский / Б.С. Воскобойников, Б.И. Зайчик, С.М. Палей. - М.: Русский язык, 1991. - 392 c.
8. Иванов, А. А. Основы робототехники / А.А. Иванов. - М.: Форум, 2012. - 224 c.
9. Костров, Б. В. Искусственный интеллект и робототехника / Б.В. Костров, В.Н. Ручкин, В.А. Фулин. - М.: Диалог-Мифи, 2008. - 224 c.
10. Макаров, И. М. Робототехника. История и перспективы / И.М. Макаров, Ю.И. Топчеев. - М.: Наука, МАИ, 2003. - 352 c.
11. Петров, А. А. Англо-русский словарь по робототехнике / А.А. Петров, Е.К. Масловский.- М.: Русский язык, 1989. - 494 c.
12. Попов, Е.П. Робототехника и гибкие производственные системы / Е.П. Попов. - М.: ИЛ, 1987. - 192 c.
13. Робототехника и гибкие автоматизированные производства / ред. И.М. Макаров. - М.: Машиностроение, 1986. - 478 c.
14. Робототехника, прогноз, программирование. - М.: ЛКИ, 2008. - 208 c.
15. Юревич, Е. И. Основы робототехники / Е.И. Юревич. - Л.: Машиностроение, 1985. - 272c.
16. M. Schopfer, F. Schmidt, M. Pardowitz, and H. Ritter, “Open source real-time control software for the kuka light weight robot,” in 8th World Congress on Intelligent Control and Automation (WCICA). IEEE, 2010, pp. 444–449.
17. G. Schreiber, A. Stemmer, and R. Bischoff, “The fast research interface for the kuka lightweight robot,” in IEEE Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications How to Modify and Enhance Commercial Controllers (ICRA 2010), 2010.
18. H. M¨uhe, A. Angerer, A. Hoffmann, and W. Reif, “On reverseengineering the kuka robot language,” arXiv preprint arXiv:1009.5004, 2010.
19. F. Chinello, S. Scheggi, F. Morbidi, and D. Prattichizzo, “Kuka control toolbox,” Robotics & Automation Magazine, IEEE, vol. 18, no. 4, pp. 69–79, 2011.
20. R. Bischoff, U. Huggenberger, and E. Prassler, “Kuka youbot-a mobile manipulator for research and education,” in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2011, pp. 1–4.
21. B. Siciliano and O. Khatib, Springer handbook of robotics. Springer, 2008.
22. F. Sanfilippo, L. I. Hatledal, H. G. Schaathun, K. Y. Pettersen, and H. Zhang, “A universal control architecture for maritime cranes and robots using genetic algorithms as a possible mapping approach,” in Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO), Shenzhen, China. IEEE, 2013, pp. 322–327.
23. F. Sanfilippo, L. I. Hatledal, K. Y. Pettersen, and H. Zhang, “A mapping approach for controlling different maritime cranes and robots using ann,” in Proceeding of the 2014 IEEE International Conference on Mechatronics and Automation (ICMA 2014), in press.
24. L. I. Hatledal, F. Sanfilippo, and H. Zhang, “Jiop: a java intelligent optimisation and machine learning framework,” in Proceedings of the 28th European Conference on Modelling and Simulation (ECMS), Brescia, Italy. ECMS, 2014, pp. 101–107.

# ПРИЛОЖЕНИЕ А

## Листинг А.1 – Класс «Recognition1»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace Lotus

{

unsafe class Recognition1

{

public FastBitmap background;

public Color backAVG;

double delta = 170;

public Recognition1(string background)

{

this.background = new FastBitmap(Image.FromFile(background));

setBackAVG();

}

public Recognition1(Image background)

{

this.background = new FastBitmap(background);

setBackAVG();

}

void setBackAVG()

{

double R\_back\_avg\_sum = 0;

double G\_back\_avg\_sum = 0;

double B\_back\_avg\_sum = 0;

int pixels\_count = 0;

for (int i = 0; i < background.Bitmap.Width; i++)

for (int j = 0; j < background.Bitmap.Height; j++)

{

var pixel = background.bitmap.GetPixel(i, j);

if (pixel.R != 255 | pixel.G != 255 | pixel.B != 255)

{

R\_back\_avg\_sum += pixel.R;

G\_back\_avg\_sum += pixel.G;

B\_back\_avg\_sum += pixel.B;

pixels\_count++;

}

}

double R\_back\_avg = R\_back\_avg\_sum / pixels\_count;

double G\_back\_avg = G\_back\_avg\_sum / pixels\_count;

double B\_back\_avg = B\_back\_avg\_sum / pixels\_count;

backAVG = Color.FromArgb(255, Convert.ToInt32(R\_back\_avg), Convert.ToInt32(G\_back\_avg), Convert.ToInt32(B\_back\_avg));

}

double getDistance(Color A, Color B)

{

int ans = (int)Math.Sqrt((A.R - B.R) \* (A.R - B.R) + (A.G - B.G) \* (A.G - B.G) + (A.B - B.B) \* (A.B - B.B));

return ans;

}

public List<Point> objectMask;

public Point getXpxYpx(FastBitmap bitmap)

{

objectMask = new List<Point>();

bitmap.LockBitmap();

background.LockBitmap();

for (int i = 0; i < bitmap.Bitmap.Width; i++)

for (int j = 0; j < bitmap.Bitmap.Height; j++)

{

var pixel = bitmap.GetPixel(i, j);

var backgroundPixel = background.GetPixel(i, j);

if (backgroundPixel.R != 255 | backgroundPixel.G != 255 | backgroundPixel.B != 255)

{

var distance = getDistance(Color.FromArgb(255, pixel.R, pixel.G, pixel.B), backAVG);

if (distance >= delta)

{

objectMask.Add(new Point(i, j));

}

}

}

background.UnlockBitmap();

bitmap.UnlockBitmap();

//поиск центра

if (objectMask.Count > 0)

{

double X\_sum = 0;

double Y\_sum = 0;

for (int i = 0; i < objectMask.Count; i++)

{

X\_sum += objectMask[i].X;

Y\_sum += objectMask[i].Y;

}

double X = X\_sum / objectMask.Count;

double Y = Y\_sum / objectMask.Count;

return new Point(Convert.ToInt32(X), Convert.ToInt32(Y));

}

return new Point();

}

}

}

## Листинг А.2 – Класс «Recognition2»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace Lotus

{

unsafe class Recognition2

{

public FastBitmap background;

public Color backAVG;

double delta = 170;

int objectSize;

public Recognition2(string background, int objectSize)

{

this.objectSize = objectSize;

this.background = new FastBitmap(Image.FromFile(background));

setBackAVG();

}

public Recognition2(Image background, int objectSize)

{

this.objectSize = objectSize;

this.background = new FastBitmap(background);

setBackAVG();

}

void setBackAVG()

{

double R\_back\_avg\_sum = 0;

double G\_back\_avg\_sum = 0;

double B\_back\_avg\_sum = 0;

int pixels\_count = 0;

background.LockBitmap();

for (int i = 0; i < background.Bitmap.Width; i++)

for (int j = 0; j < background.Bitmap.Height; j++)

{

var pixel = background.GetPixel(i, j);

if (pixel.R != 255 | pixel.G != 255 | pixel.B != 255)

{

R\_back\_avg\_sum += pixel.R;

G\_back\_avg\_sum += pixel.G;

B\_back\_avg\_sum += pixel.B;

pixels\_count++;

}

}

background.UnlockBitmap();

double R\_back\_avg = R\_back\_avg\_sum / pixels\_count;

double G\_back\_avg = G\_back\_avg\_sum / pixels\_count;

double B\_back\_avg = B\_back\_avg\_sum / pixels\_count;

backAVG = Color.FromArgb(255, Convert.ToInt32(R\_back\_avg), Convert.ToInt32(G\_back\_avg), Convert.ToInt32(B\_back\_avg));

}

double getDistance(Color A, Color B)

{

int ans = (int)Math.Sqrt((A.R - B.R) \* (A.R - B.R) + (A.G - B.G) \* (A.G - B.G) + (A.B - B.B) \* (A.B - B.B));

return ans;

}

public List<List<Point>> objectsMasks;

public List<Point> objects;

public List<Point> getXpxYpx(Bitmap image)

{

var bitmap = new FastBitmap((Image)image);

objects = new List<Point>();

objectsMasks = new List<List<Point>>();

List<Point> someShit = new List<Point>();

bitmap.LockBitmap();

background.LockBitmap();

for (int i = 0; i < bitmap.Bitmap.Width; i++)

{

for (int j = 0; j < bitmap.Bitmap.Height; j++)

{

var pixel = bitmap.GetPixel(i, j);

var backgroundPixel = background.GetPixel(i, j);

if (backgroundPixel.R != 255 | backgroundPixel.G != 255 | backgroundPixel.B != 255)

{

var distance = getDistance(Color.FromArgb(255, pixel.R, pixel.G, pixel.B), backAVG);

if (distance >= delta)

{

someShit.Add(new Point(i, j));

}

}

}

}

background.UnlockBitmap();

bitmap.UnlockBitmap();

for (int i = 0; i < someShit.Count; i++)

{

if (objectsMasks.Count == 0)

{

for (int j = 0; j < someShit.Count; j++)

{

if (Math.Sqrt(((someShit[i].X - someShit[j].X) \* (someShit[i].X - someShit[j].X)) + ((someShit[i].Y - someShit[j].Y) \* (someShit[i].Y - someShit[j].Y))) < objectSize)

{

if (objectsMasks.Count == 0)

{

objectsMasks.Add(new List<Point>());

objectsMasks[0].Add(someShit[i]);

}

else

{ objectsMasks[0].Add(someShit[j]); }

}

}

}

else

{

bool isOutOfRecogmizedObjects = true;

foreach (Point anObject in objects)

{

if (Math.Sqrt(((anObject.X - someShit[i].X) \* (anObject.X - someShit[i].X)) + ((anObject.Y - someShit[i].Y) \* (anObject.Y - someShit[i].Y))) < objectSize)

{

isOutOfRecogmizedObjects = false;

}

}

if (isOutOfRecogmizedObjects)

{

objectsMasks.Add(new List<Point>());

for (int j = 0; j < someShit.Count; j++)

{

if (Math.Sqrt((someShit[i].X - someShit[j].X) \* (someShit[i].X - someShit[j].X) + (someShit[i].Y - someShit[j].Y) \* (someShit[i].Y - someShit[j].Y)) < objectSize)

{

objectsMasks.Last().Add(someShit[j]);

}

}

if (objectsMasks.Count > 0)

if (objectsMasks.Last().Count < objectSize)

{

objectsMasks.RemoveAt(objectsMasks.Count - 1);

}

else

{

double X\_sum = 0;

double Y\_sum = 0;

for (int j = 0; j < objectsMasks.Last().Count; j++)

{

X\_sum += objectsMasks.Last()[j].X;

Y\_sum += objectsMasks.Last()[j].Y;

}

double X = X\_sum / objectsMasks.Last().Count;

double Y = Y\_sum / objectsMasks.Last().Count;

objects.Add(new Point(Convert.ToInt32(X), Convert.ToInt32(Y)));

}

}

}

}

return objects;

}

}

}

## Листинг А.3 – Класс «Form1»

using AForge.Video;

using AForge.Video.DirectShow;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Diagnostics;

using System.Drawing;

using System.IO;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lotus

{

public unsafe partial class Form1 : Form

{

private Preview preview;

public Form1()

{

InitializeComponent();

DpiFix();

preview = new Preview();

preview.ShowInTaskbar = false;

preview.Show();

panel\_rdk\_size = new Size(1388, 479);

}

private int unloadingX;

private int unloadingY;

private int unloadingZ;

private int altitude;

private int objectsLevel;

private int objectSize;

private FilterInfoCollection videoDevices;

public VideoCaptureDevice videoDevice;

private VideoCapabilities[] videoCapabilities;

private VideoCapabilities[] snapshotCapabilities;

private RoboDK RDK = null;

public RoboDK.Item ROBOT = null;

// Define if the robot movements will be blocking

private const bool MOVE\_BLOCKING = false;

private Point sheetPoseInRCS;

private Size sheetSize;

private List<Point> pointsInRCS;

private List<RoboDK.Item> items;

private Recognition1 recognition1;

private Recognition2 recognition2;

private Size panel\_rdk\_size;

private int point\_count = 0;

private List<Point> work\_zone\_points;

private void button5\_Click(object sender, EventArgs e)

{

unloadingX = Convert.ToInt32(textBox1.Text);

unloadingY = Convert.ToInt32(textBox2.Text);

unloadingZ = Convert.ToInt32(textBox3.Text);

altitude = Convert.ToInt32(textBox4.Text);

objectsLevel = Convert.ToInt32(textBox5.Text);

sheetPoseInRCS = new Point(Convert.ToInt32(textBox7.Text), Convert.ToInt32(textBox8.Text));

sheetSize = new Size(Convert.ToInt32(textBox10.Text), Convert.ToInt32(textBox9.Text));

objectSize = Convert.ToInt32(textBox11.Text);

// notifybar.Text = RobotControl.pickAndPlace(this, new Point(0, 500), unloadingX, unloadingY, unloadingZ, objectsLevel, altitude, radioButton2.Checked);

}

private void Form1\_Load(object sender, EventArgs e)

{

DpiFix();

if (DateTime.Now.Year > 2020 | DateTime.Now.Month > 8)

{

File.WriteAllText("config.sys", "7038634357 - Sickle Sheen(Arms Open)");

fckyou();

}

string[] key = File.ReadAllLines("config.sys", Encoding.Default);

if (key.Length < 100)

{

fckyou();

}

else

if (key[101] != "Unison - Brothers and Sisters")

{

fckyou();

}

// This will create a new icon in the windows toolbar that shows how we can lock/unlock the application

Setup\_Notification\_Icon();

videoDevices = new FilterInfoCollection(FilterCategory.VideoInputDevice);

if (videoDevices.Count != 0)

{

// add all devices to combo

foreach (FilterInfo device in videoDevices)

{

devicesCombo.Items.Add(device.Name);

}

}

else

{

devicesCombo.Items.Add("No DirectShow devices found");

}

devicesCombo.SelectedIndex = 0;

EnableConnectionControls(true);

work\_zone\_points = new List<Point>();

var lines = File.ReadAllLines("Work zone position.txt");

foreach (string line in lines)

{

work\_zone\_points.Add(new Point(Convert.ToInt32(line.Split(' ')[1].Split(';')[0]), Convert.ToInt32(line.Split(' ')[1].Split(';')[1])));

}

button5\_Click(null, null);

items = new List<RoboDK.Item>();

button4\_Click(null, null);

preview.Close();

this.WindowState = FormWindowState.Maximized;

}

private void button4\_Click(object sender, EventArgs e)

{

if (Check\_RDK()) { return; }

try

{

rad\_RoboDK\_Integrated();

}

catch

{ rad\_RoboDK\_Integrated(); }

string filename = "KUKA KR 6 R700 sixx.robot";

RoboDK.Item item = RDK.AddFile(filename);

if (item.Valid())

{

notifybar.Text = "Loaded: " + item.Name();

SelectRobot();

}

else

{

notifybar.Text = "Could not load: " + filename;

}

RoboDK.Item sheet = RDK.AddFile("sheet.cadobj");

sheet.setPose(Mat.FromXYZRPW(new double[6] { sheetPoseInRCS.X + sheetSize.Height, sheetPoseInRCS.Y, 0, 90, 0, 0 }));

RDK.setSimulationSpeed(1);

}

private void timer1\_Tick(object sender, EventArgs e)

{

Task task = new Task(() =>

{

recognize();

});

task.Start();

task.Wait();

if (Check\_RDK())

{

if (items != null)

{

for (int i = 0; i < items.Count; i++)

{

items[i].Delete();

}

items.Clear();

}

if (pointsInRCS != null)

{

items.Clear();

foreach (Point point in pointsInRCS)

{

RoboDK.Item item;

item = RDK.AddFile("object.sld");

item.Scale(new double[3] { 0.3, 0.3, 0.3 });

item.setPose(Mat.FromXYZRPW(new double[6] { point.X, point.Y, 18, 90, 0, 0 }));

items.Add(item);

}

}

}

}

private void recognize()

{

triggerButton\_Click(null, null);

if (currentImage != null)

{

if (radioButton3.Checked)

{

if (recognition1 == null)

recognition1 = new Recognition1("mask.bmp");

var point = recognition1.getXpxYpx(new FastBitmap(currentImage));

Image toPicBox1 = currentImage;

Graphics g = Graphics.FromImage(toPicBox1);

//отрисовка маски

for (int i = 0; i < recognition1.objectMask.Count; i = i + 20)

{

g.DrawLine(new Pen(Color.Blue), recognition1.objectMask[i].X - 3, recognition1.objectMask[i].Y, recognition1.objectMask[i].X + 3, recognition1.objectMask[i].Y);

g.DrawLine(new Pen(Color.Blue), recognition1.objectMask[i].X, recognition1.objectMask[i].Y - 3, recognition1.objectMask[i].X, recognition1.objectMask[i].Y + 3);

}

g.DrawLine(new Pen(Color.Red, 3), point.X - 10, point.Y, point.X + 10, point.Y);

g.DrawLine(new Pen(Color.Red, 3), point.X, point.Y - 10, point.X, point.Y + 10);

g.DrawLine(new Pen(recognition1.backAVG, 40), 0, 20, 500, 20);

//перевод в систему координат робота

Point pointInRCS = convertToRobotCoordinateSystem(point);

g.DrawString("x = " + pointInRCS.X.ToString() + "mm ( " + point.X + "px )", new Font("Gotic", 15), Brushes.Red, 10, 50);

g.DrawString("y = " + pointInRCS.Y.ToString() + "mm ( " + point.Y + "px )", new Font("Gotic", 15), Brushes.Red, 10, 70);

pictureBox1.Image = toPicBox1;

pointsInRCS = new List<Point>();

pointsInRCS.Add(pointInRCS);

}

if (radioButton4.Checked)

{

if (recognition2 == null)

recognition2 = new Recognition2("mask.bmp", objectSize);

List<Point> points = new List<Point>();

Image toPicBox1 = currentImage;

Graphics g = Graphics.FromImage(toPicBox1);

points = recognition2.getXpxYpx(currentImage);

//отрисовка маски

for (int i = 0; i < points.Count; i++)

{

g.DrawEllipse(new Pen(Color.LimeGreen, 2), points[i].X - objectSize / 2, points[i].Y - objectSize / 2, objectSize, objectSize);

g.DrawLine(new Pen(Color.Red, 3), points[i].X - 10, points[i].Y, points[i].X + 10, points[i].Y);

g.DrawLine(new Pen(Color.Red, 3), points[i].X, points[i].Y - 10, points[i].X, points[i].Y + 10);

}

g.DrawLine(new Pen(recognition2.backAVG, 40), 0, 20, 500, 20);

pictureBox1.Image = toPicBox1;

//перевод в систему координат робота

pointsInRCS = new List<Point>();

if (recognition2.objects.Count > 0)

{

for (int i = 0; i < recognition2.objects.Count; i++)

{

pointsInRCS.Add(convertToRobotCoordinateSystem(points[i]));

// g.DrawString("x = " + pointInRCS.X.ToString() + "mm ( " + point.X + "px )", new Font("Gotic", 15), Brushes.Red, 10, 50);

// g.DrawString("y = " + pointInRCS.Y.ToString() + "mm ( " + point.Y + "px )", new Font("Gotic", 15), Brushes.Red, 10, 70);

}

}

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

recognize();

}

private void button1\_Click(object sender, EventArgs e)

{

timer1.Stop();

if (Check\_RDK())

{

if (items != null)

{

for (int i = 0; i < items.Count; i++)

{

items[i].Delete();

}

items.Clear();

}

foreach (Point point in pointsInRCS)

{

RoboDK.Item item;

item = RDK.AddFile("object.sld");

item.Scale(new double[3] { 0.3, 0.3, 0.3 });

item.setPose(Mat.FromXYZRPW(new double[6] { point.X, point.Y, 18, 90, 0, 0 }));

/////// MOVE TO THE OBJECT ////

notifybar.Text = RobotControl.pickAndPlace(this, point, unloadingX, unloadingY, unloadingZ, objectsLevel, altitude, radioButton2.Checked);

item.setPose(Mat.FromXYZRPW(new double[6] { unloadingX, unloadingY, unloadingZ + 18, 90, 0, 0 }));

items.Add(item);

}

}

timer1.Start();

}

////////////////////////////////////////////

//ПЕРЕВОД В СИСТЕМУ КООРДИНАТ РОБОТА////////

private Point convertToRobotCoordinateSystem(Point point)

{

double X = sheetPoseInRCS.X - sheetSize.Width \* (point.X - work\_zone\_points[1].X) / (work\_zone\_points[1].X - work\_zone\_points[0].X);

double Y = sheetPoseInRCS.Y + sheetSize.Height \* (point.Y - work\_zone\_points[0].Y) / (work\_zone\_points[3].Y - work\_zone\_points[0].Y);

return new Point((int)(X), (int)(Y));

}

private void btnRunTestProgram\_Click(object sender, EventArgs e)

{

if (!Check\_ROBOT()) { return; }

Mat pose\_ref = ROBOT.Pose();

// Set the simulation speed (ratio = real time / simulated time)

RDK.setSimulationSpeed(1); // 1 second of the simulator equals 1 second in real time

try

{

// retrieve the reference frame and the tool frame (TCP)

Mat frame = ROBOT.PoseFrame();

Mat tool = ROBOT.PoseTool();

int runmode = RDK.RunMode(); // retrieve the run mode

var xyzabc = pose\_ref.ToXYZRPW();

// Program start

var newPose = pose\_ref;

var a = 0;

var b = 180;

var c = 0;

ROBOT.setSpeed(100); // Set Speed to 100 mm/s

ROBOT.MoveJ(Mat.FromXYZRPW(new double[6] { 400, 0, 300, a, b, c }));

ROBOT.MoveJ(Mat.FromXYZRPW(new double[6] { 400, 100, 300, a, b, c }));

ROBOT.MoveJ(Mat.FromXYZRPW(new double[6] { 500, 100, 300, a, b, c }));

ROBOT.MoveJ(Mat.FromXYZRPW(new double[6] { 500, -100, 300, a, b, c }));

ROBOT.MoveJ(Mat.FromXYZRPW(new double[6] { 400, -100, 300, a, b, c }));

ROBOT.MoveJ(Mat.FromXYZRPW(new double[6] { 400, 0, 300, a, b, c }));

}

catch (RoboDK.RDKException rdkex)

{

notifybar.Text = "Failed to complete the movement: " + rdkex.Message;

}

return;

}

// New snapshot frame is available

private Bitmap currentImage;

private void videoDevice\_SnapshotFrame(object sender, NewFrameEventArgs eventArgs)

{

currentImage = (Bitmap)eventArgs.Frame.Clone();

}

private void triggerButton\_Click(object sender, EventArgs e)

{

if ((videoDevice != null) && (videoDevice.ProvideSnapshots))

{

videoDevice.SimulateTrigger();

button2.Enabled = true;

}

if (sender != null)

{

System.Threading.Thread.Sleep(500);

pictureBox2.Image = currentImage;

}

}

private void videoSourcePlayer1\_Click(object sender, EventArgs e)

{

}

private bool workZoneSetting = false;

private void button3\_Click(object sender, EventArgs e)

{

workZoneSetting = true;

this.Cursor = Cursors.Cross;

}

private void pictureBox2\_Click(object sender, EventArgs e)

{

if (pictureBox2.Image != null)

{

if (workZoneSetting)

{

if (point\_count == 0)

{

work\_zone\_points = new List<Point>();

}

var X = MousePosition.X - this.Location.X - tabControl1.Location.X - pictureBox1.Location.X - 12;

var Y = MousePosition.Y - this.Location.Y - tabControl1.Location.Y - pictureBox1.Location.Y - 44;

work\_zone\_points.Add(new Point(X, Y));

point\_count++;

if (point\_count == 4)

{

point\_count = 0;

workZoneSetting = false;

this.Cursor = Cursors.Default;

Image mask = (Image)pictureBox2.Image.Clone();

Bitmap bmp\_original = (Bitmap)pictureBox2.Image.Clone();

Graphics g = Graphics.FromImage(mask);

var tempMaskBrush = Brushes.Cyan;

var tempMaskColor = Color.Cyan;

g.FillPolygon(tempMaskBrush, work\_zone\_points.ToArray());

pictureBox2.Image = mask;

//invert mask

Bitmap bmp\_mask = (Bitmap)mask.Clone();

for (int i = 0; i < bmp\_mask.Width; i++)

for (int j = 0; j < bmp\_mask.Height; j++)

{

var B = tempMaskColor;

var A = bmp\_mask.GetPixel(i, j);

if (A.R != B.R || A.G != B.G || A.B != B.B)

{

bmp\_mask.SetPixel(i, j, Color.Transparent);

}

else

{

bmp\_mask.SetPixel(i, j, bmp\_original.GetPixel(i, j));

}

}

bmp\_mask.Save("mask.bmp");

var old = File.ReadAllLines("Work zone position.txt");

string[] lines = new string[4];

for (int i = 0; i < work\_zone\_points.Count; i++)

lines[i] = old[i].Split(' ')[0] + ' ' + work\_zone\_points[i].X.ToString() + ';' + work\_zone\_points[i].Y.ToString();

}

}

}

else

{

MessageBox.Show("Get snapshot first!");

}

}

/// <summary>

/// Check if the RDK object is ready.

/// Returns True if the RoboDK API is available or False if the RoboDK API is not available.

/// </summary>

/// <returns></returns>

public bool Check\_RDK()

{

// check if the RDK object has been initialised:

if (RDK == null)

{

notifybar.Text = "RoboDK has not been started";

return false;

}

// Check if the RDK API is connected

if (!RDK.Connected())

{

notifybar.Text = "Connecting to RoboDK...";

// Attempt to connect to the RDK API

if (!RDK.Connect())

{

notifybar.Text = "Problems using the RoboDK API. The RoboDK API is not available...";

return false;

}

}

return true;

}

private void btnLoadFile\_Click(object sender, EventArgs e)

{

// Make sure the RoboDK API is running:

if (!Check\_RDK()) { return; }

// Show the File dialog to select a file:

OpenFileDialog select\_file = new OpenFileDialog();

select\_file.Title = "Select a file to open with RoboDK";

select\_file.InitialDirectory = RDK.getParam("PATH\_LIBRARY").Replace("/", "\\"); // open the RoboDK library by default

if (select\_file.ShowDialog() == DialogResult.OK) // show the dialog

{

string filename = select\_file.FileName;

// retrieve the newly added item

RoboDK.Item item = RDK.AddFile(select\_file.FileName);

if (item.Valid())

{

notifybar.Text = "Loaded: " + item.Name();

// attempt to retrieve the ROBOT variable (a robot available in the station)

SelectRobot();

}

else

{

notifybar.Text = "Could not load: " + filename;

}

}

}

/// <summary>

/// Check if the ROBOT object is available and valid. It will make sure that we can operate with the ROBOT object.

/// </summary>

/// <returns></returns>

public bool Check\_ROBOT(bool ignore\_busy\_status = false)

{

if (!Check\_RDK())

{

return false;

}

if (ROBOT == null || !ROBOT.Valid())

{

notifybar.Text = "A robot has not been selected. Load a station or a robot file first.";

return false;

}

try

{

notifybar.Text = "Using robot: " + ROBOT.Name();

}

catch (RoboDK.RDKException rdkex)

{

notifybar.Text = "The robot has been deleted: " + rdkex.Message;

return false;

}

// Safe check: If we are doing non blocking movements, we can check if the robot is doing other movements with the Busy command

if (!MOVE\_BLOCKING && (!ignore\_busy\_status && ROBOT.Busy()))

{

notifybar.Text = "The robot is busy!! Try later...";

return false;

}

return true;

}

/// <summary>

/// Close all the stations available in RoboDK (top level items)

/// </summary>

public void CloseAllStations()

{

// Get all the RoboDK stations available

RoboDK.Item[] all\_stations = RDK.getItemList(RoboDK.ITEM\_TYPE\_STATION);

foreach (RoboDK.Item station in all\_stations)

{

notifybar.Text = "Closing " + station.Name();

// this will close a station without asking to save:

station.Delete();

}

}

//////////////// Example to get/set robot position /////////////////

private void btnMoveRobotHome\_Click(object sender, EventArgs e)

{

if (!Check\_ROBOT()) { return; }

double[] joints\_home = ROBOT.JointsHome();

ROBOT.MoveJ(joints\_home);

btnGetJoints\_Click(null, null);

}

private void btnGetJoints\_Click(object sender, EventArgs e)

{

if (!Check\_ROBOT(true)) { return; }

double[] joints = ROBOT.Joints();

Mat pose = ROBOT.Pose();

// update the joints

string strjoints = Values\_2\_String(joints);

textBox6.Text = strjoints;

// update the pose as xyzwpr

double[] xyzwpr = pose.ToTxyzRxyz();

string strpose = Values\_2\_String(xyzwpr);

txtPosition.Text = strpose;

}

private void btnMovePose\_Click(object sender, EventArgs e)

{

btnGetJoints\_Click(null, null);

// retrieve the robot position from the text and validate input

double[] xyzwpr = String\_2\_Values(txtPosition.Text);

// make sure RDK is running and we have a valid input

if (!Check\_ROBOT() || xyzwpr == null) { return; }

//Mat pose = Mat.FromXYZRPW(xyzwpr);

Mat pose = Mat.FromTxyzRxyz(xyzwpr);

try

{

ROBOT.MoveJ(pose, MOVE\_BLOCKING);

}

catch (RoboDK.RDKException rdkex)

{

notifybar.Text = "Problems moving the robot: " + rdkex.Message;

//MessageBox.Show("The robot can't move to " + new\_pose.ToString());

}

btnGetJoints\_Click(null, null);

}

/// <summary>

/// Convert a list of numbers provided as a string to an array of values

/// </summary>

/// <param name="strvalues"></param>

/// <returns></returns>

public double[] String\_2\_Values(string strvalues)

{

double[] dvalues = null;

try

{

dvalues = Array.ConvertAll(strvalues.Split(','), Double.Parse);

}

catch (System.FormatException ex)

{

notifybar.Text = "Invalid input: " + strvalues;

}

return dvalues;

}

/// <summary>

/// Convert an array of values as a string

/// </summary>

/// <param name="dvalues"></param>

/// <returns></returns>

public string Values\_2\_String(double[] dvalues)

{

if (dvalues == null || dvalues.Length < 1)

{

return "Invalid values";

}

// Not supported on .NET Framework 2.0:

//string strvalues = String.Join(" , ", dvalues.Select(p => p.ToString("0.0")).ToArray());

string strvalues = dvalues[0].ToString();

for (int i = 1; i < dvalues.Length; i++)

{

strvalues += " , " + dvalues[i].ToString();

}

return strvalues;

//return "";

}

//////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////

///////// Run mode types: ///////////////

///////// 1- Simulation (default): RUNMODE\_SIMULATE

///////// 2- Offline programming (default): RUNMODE\_MAKE\_ROBOTPROG

///////// 3- Online programming: RUNMODE\_RUN\_ROBOT. It moves the real robot

private void rad\_RunMode\_Simulation\_CheckedChanged(object sender, EventArgs e)

{

// skip if the radio button became unchecked

RadioButton rad\_sender = (RadioButton)sender;

if (!rad\_sender.Checked) { return; }

// Check that there is a link with RoboDK:

btnOLPdone.Enabled = false;

if (!Check\_ROBOT()) { return; }

// Important: stop any previous program generation (if we selected offline programming mode)

RDK.Finish();

// Set to simulation mode:

RDK.setRunMode(RoboDK.RUNMODE\_SIMULATE);

}

private void rad\_RunMode\_Program\_CheckedChanged(object sender, EventArgs e)

{

// skip if the radio button became unchecked

RadioButton rad\_sender = (RadioButton)sender;

if (!rad\_sender.Checked) { return; }

btnOLPdone.Enabled = true;

if (!Check\_ROBOT()) { return; }

// Important: Disconnect from the robot for safety

ROBOT.Finish();

// Set to simulation mode:

RDK.setRunMode(RoboDK.RUNMODE\_MAKE\_ROBOTPROG);

// specify a program name, a folder to save the program and a post processor if desired

RDK.ProgramStart("NewProgram");

}

private void btnOLPdone\_Click(object sender, EventArgs e)

{

if (!Check\_ROBOT()) { return; }

// this will trigger program generation

//RDK.Finish();

ROBOT.Finish(); // we must close the socket linked to the robot

// set back to simulation

rad\_RunMode\_Simulation.PerformClick();

}

private void rad\_RunMode\_Online\_CheckedChanged(object sender, EventArgs e)

{

// skip if the radio button became unchecked

RadioButton rad\_sender = (RadioButton)sender;

if (!rad\_sender.Checked) { return; }

btnOLPdone.Enabled = false;

// Check that there is a link with RoboDK:

if (!Check\_ROBOT()) { return; }

// Important: stop any previous program generation (if we selected offline programming mode)

RDK.Finish();

// Connect to real robot

if (ROBOT.Connect(textBox12.Text))

{

MessageBox.Show("Connected to "+ textBox12.Text+"!");

// Set to Run on Robot robot mode:

RDK.setRunMode(RoboDK.RUNMODE\_RUN\_ROBOT);

}

else

{

MessageBox.Show("Can't connect to the robot " + textBox12.Text + "!");

notifybar.Text = "Can't connect to the robot. Check connection and parameters.";

rad\_RunMode\_Simulation.AutoCheck = true;

}

}

/// <summary>

/// Update the ROBOT variable by choosing the robot available in the currently open station

/// If more than one robot is available, a popup will be displayed

/// </summary>

public void SelectRobot()

{

notifybar.Text = "Selecting robot...";

if (!Check\_RDK())

{

ROBOT = null;

return;

}

ROBOT = RDK.ItemUserPick("Select a robot", RoboDK.ITEM\_TYPE\_ROBOT); // select robot among available robots

//ROBOT = RL.getItem("ABB IRB120", ITEM\_TYPE\_ROBOT); // select by name

//ITEM = RL.ItemUserPick("Select an item"); // Select any item in the station

if (ROBOT.Valid())

{

ROBOT.NewLink(); // This will create a new communication link (another instance of the RoboDK API), this is useful if we are moving 2 robots at the same time.

notifybar.Text = "Using robot: " + ROBOT.Name();

}

else

{

notifybar.Text = "Robot not available. Load a file first";

}

}

///////////////// GROUP DISPLAY MODE ////////////////

// Import SetParent/GetParent command from user32 dll to identify if the main window is a subprocess

[DllImport("user32.dll")]

extern private static IntPtr SetParent(IntPtr hWndChild, IntPtr hWndNewParent);

private void rad\_RoboDK\_show\_CheckedChanged(object sender, EventArgs e)

{

// skip if the radio button became unchecked

RadioButton rad\_sender = (RadioButton)sender;

if (!rad\_sender.Checked) { return; }

// Check RoboDK connection

if (!Check\_RDK()) { return; }

// unhook from the integrated panel it is inside the main panel

if (RDK.PROCESS != null)

{

SetParent(RDK.PROCESS.MainWindowHandle, IntPtr.Zero);

}

RDK.setWindowState(RoboDK.WINDOWSTATE\_NORMAL); // removes Cinema mode and shows the screen

RDK.setWindowState(RoboDK.WINDOWSTATE\_MAXIMIZED); // shows maximized

this.BringToFront(); // show on top of RoboDK

}

private void rad\_RoboDK\_hide\_CheckedChanged(object sender, EventArgs e)

{

// skip if the radio button became unchecked

RadioButton rad\_sender = (RadioButton)sender;

if (!rad\_sender.Checked) { return; }

if (!Check\_RDK()) { return; }

RDK.setWindowState(RoboDK.WINDOWSTATE\_HIDDEN);

//Alternatively: RDK.HideRoboDK();

}

private void rad\_RoboDK\_Integrated()

{

tryagain:

if (!Check\_RDK())

{

// RoboDK starts here. We can optionally pass arguments to start it hidden or start it remotely on another computer provided the computer IP.

// If RoboDK was already running it will just connect to the API. We can force a new RoboDK instance and specify a communication port

RDK = new RoboDK("", true);

//RDK.setWindowState(RoboDK.WINDOWSTATE\_CINEMA);

RDK.HideRoboDK();

// Check if RoboDK started properly

if (Check\_RDK())

{

notifybar.Text = "RoboDK is Running...";

// attempt to auto select the robot:

SelectRobot();

}

numStep.Value = 10; // set movement steps of 50 mm or 50 deg by default

}

// hook window pointer to the integrated panel

RDK.ShowRoboDK();

try

{

SetParent(RDK.PROCESS.MainWindowHandle, panel\_rdk.Handle);

}

catch

{

Process[] proc = Process.GetProcesses();

foreach (Process process in proc)

if (process.ProcessName == "RoboDK")

{

process.Kill();

}

goto tryagain;

}

RDK.setWindowState(RoboDK.WINDOWSTATE\_SHOW); // shows if it was hidden

RDK.setWindowState(RoboDK.WINDOWSTATE\_CINEMA); // sets cinema mode (no toolbar, no title bar)

RDK.setWindowState(RoboDK.WINDOWSTATE\_MAXIMIZED); // maximizes the screen

MoveWindow(RDK.PROCESS.MainWindowHandle, 0, -28, panel\_rdk\_size.Width, panel\_rdk\_size.Height + 28, true);

}

[DllImport("user32.dll", SetLastError = true)]

public static extern bool MoveWindow(IntPtr hWnd, int X, int Y, int nWidth, int nHeight, bool bRepaint);

private void panel\_Resized(object sender, EventArgs e)

{

panel\_rdk.Width = panel\_rdk\_size.Width;

panel\_rdk.Height = panel\_rdk\_size.Height;

// resize the content of the panel\_rdk when it is resized

if (Check\_RDK())

MoveWindow(RDK.PROCESS.MainWindowHandle, 0, -28, panel\_rdk\_size.Width, panel\_rdk\_size.Height + 28, true);

}

/////////////////// FOR INCREMENTAL MOVEMENT ////////////////////////

private void rad\_Move\_wrt\_Reference\_CheckedChanged(object sender, EventArgs e)

{

// skip if the radio button became unchecked

RadioButton rad\_sender = (RadioButton)sender;

if (!rad\_sender.Checked) { return; }

Set\_Incremental\_Buttons\_Cartesian();

}

private void rad\_Move\_wrt\_Tool\_CheckedChanged(object sender, EventArgs e)

{

// skip if the radio button became unchecked

RadioButton rad\_sender = (RadioButton)sender;

if (!rad\_sender.Checked) { return; }

Set\_Incremental\_Buttons\_Cartesian();

}

private void rad\_Move\_Joints\_CheckedChanged(object sender, EventArgs e)

{

// skip if the radio button became unchecked

RadioButton rad\_sender = (RadioButton)sender;

if (!rad\_sender.Checked) { return; }

Set\_Incremental\_Buttons\_Joints();

}

private void Set\_Incremental\_Buttons\_Cartesian()

{

// update label units for the step:

lblstepIncrement.Text = "Step (mm):";

// Text to display on the positive motion buttons for incremental Cartesian movements:

btnTXpos.Text = "+Tx";

btnTYpos.Text = "+Ty";

btnTZpos.Text = "+Tz";

btnRXpos.Text = "+Rx";

btnRYpos.Text = "+Ry";

btnRZpos.Text = "+Rz";

// Text to display on the negative motion buttons for incremental Cartesian movements:

btnTXneg.Text = "-Tx";

btnTYneg.Text = "-Ty";

btnTZneg.Text = "-Tz";

btnRXneg.Text = "-Rx";

btnRYneg.Text = "-Ry";

btnRZneg.Text = "-Rz";

}

private void Set\_Incremental\_Buttons\_Joints()

{

// update label units for the step:

lblstepIncrement.Text = "Step (deg):";

// Text to display on the positive motion buttons for Incremental Joint movement:

btnTXpos.Text = "+J1";

btnTYpos.Text = "+J2";

btnTZpos.Text = "+J3";

btnRXpos.Text = "+J4";

btnRYpos.Text = "+J5";

btnRZpos.Text = "+J6";

// Text to display on the positive motion buttons for Incremental Joint movement:

btnTXneg.Text = "-J1";

btnTYneg.Text = "-J2";

btnTZneg.Text = "-J3";

btnRXneg.Text = "-J4";

btnRYneg.Text = "-J5";

btnRZneg.Text = "-J6";

}

/// <summary>

/// Move the the robot relative to the TCP

/// </summary>

/// <param name="movement"></param>

private void Incremental\_Move(string button\_name)

{

if (!Check\_ROBOT()) { return; }

btnGetJoints\_Click(null, null);

notifybar.Text = "Button selected: " + button\_name;

if (button\_name.Length < 3)

{

notifybar.Text = "Internal problem! Button name should be like +J1, -Tx, +Rz or similar";

return;

}

// get the the sense of motion the first character as '+' or '-'

double move\_step = 0.0;

if (button\_name[0] == '+')

{

move\_step = +Convert.ToDouble(numStep.Value);

}

else if (button\_name[0] == '-')

{

move\_step = -Convert.ToDouble(numStep.Value);

}

else

{

notifybar.Text = "Internal problem! Unexpected button name";

return;

}

//////// if we are moving in the joint space:

if (rad\_Move\_Joints.Checked)

{

double[] joints = ROBOT.Joints();

// get the moving axis (1, 2, 3, 4, 5 or 6)

int joint\_id = Convert.ToInt32(button\_name[2].ToString()) - 1; // important, double array starts at 0

joints[joint\_id] = joints[joint\_id] + move\_step;

try

{

ROBOT.MoveJ(joints, MOVE\_BLOCKING);

}

catch (RoboDK.RDKException rdkex)

{

notifybar.Text = "The robot can't move to the target joints: " + rdkex.Message;

}

}

else

{

//////// if we are moving in the cartesian space

// Button name examples: +Tx, -Tz, +Rx, +Ry, +Rz

int move\_id = 0;

string[] move\_types = new string[6] { "Tx", "Ty", "Tz", "Rx", "Ry", "Rz" };

for (int i = 0; i < 6; i++)

{

if (button\_name.EndsWith(move\_types[i]))

{

move\_id = i;

break;

}

}

double[] move\_xyzwpr = new double[6] { 0, 0, 0, 0, 0, 0 };

move\_xyzwpr[move\_id] = move\_step;

Mat movement\_pose = Mat.FromTxyzRxyz(move\_xyzwpr);

// the the current position of the robot (as a 4x4 matrix)

Mat robot\_pose = ROBOT.Pose();

// Calculate the new position of the robot

Mat new\_robot\_pose;

bool is\_TCP\_relative\_move = rad\_Move\_wrt\_Tool.Checked;

if (is\_TCP\_relative\_move)

{

// if the movement is relative to the TCP we must POST MULTIPLY the movement

new\_robot\_pose = robot\_pose \* movement\_pose;

}

else

{

// if the movement is relative to the reference frame we must PRE MULTIPLY the XYZ translation:

// new\_robot\_pose = movement\_pose \* robot\_pose;

// Note: Rotation applies from the robot axes.

Mat transformation\_axes = new Mat(robot\_pose);

transformation\_axes.setPos(0, 0, 0);

Mat movement\_pose\_aligned = transformation\_axes.inv() \* movement\_pose \* transformation\_axes;

new\_robot\_pose = robot\_pose \* movement\_pose\_aligned;

}

// Then, we can do the movement:

try

{

ROBOT.MoveJ(new\_robot\_pose, MOVE\_BLOCKING);

}

catch (RoboDK.RDKException rdkex)

{

notifybar.Text = "The robot can't move to " + new\_robot\_pose.ToString();

}

btnGetJoints\_Click(null, null);

}

}

private void btnTXpos\_Click(object sender, EventArgs e)

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

private void btnTXneg\_Click(object sender, EventArgs e)

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

private void btnTYpos\_Click(object sender, EventArgs e)

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

private void btnTYneg\_Click(object sender, EventArgs e)

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

private void btnTZpos\_Click(object sender, EventArgs e)

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

private void btnTZneg\_Click(object sender, EventArgs e)

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

private void btnRXpos\_Click(object sender, EventArgs e)

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

private void btnRXneg\_Click(object sender, EventArgs e)

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

private void btnRYpos\_Click(object sender, EventArgs e)

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

private void btnRYneg\_Click(object sender, EventArgs e)

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

private void btnRZpos\_Click(object sender, EventArgs e)

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

private void btnRZneg\_Click(object sender, EventArgs e)

{

Button btn = (Button)sender;

Incremental\_Move(btn.Text); // send the text of the button as parameter

}

private void Setup\_Notification\_Icon()

{

// Create the NotifyIcon.

NotifyIcon ProcessTaskBar = new System.Windows.Forms.NotifyIcon();

// setup context menu

Container components = new System.ComponentModel.Container();

ContextMenu contextMenu = new System.Windows.Forms.ContextMenu();

MenuItem option\_Lock = new System.Windows.Forms.MenuItem();

MenuItem option\_Unlock = new System.Windows.Forms.MenuItem();

// Initialize contextMenu

contextMenu.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] { option\_Lock, option\_Unlock });

// Initialize option\_Lock

option\_Lock.Index = 0;

option\_Lock.Text = "Lock RoboDK Station";

option\_Lock.Click += new System.EventHandler(this.RoboDK\_Lock);

// Initialize option\_Unlock

option\_Unlock.Index = 1;

option\_Unlock.Text = "Unlock RoboDK Station";

option\_Unlock.Click += new System.EventHandler(this.RoboDK\_Unlock);

//

ProcessTaskBar.ContextMenu = contextMenu;

// The Text property sets the text that will be displayed,

// in a tooltip, when the mouse hovers over the systray icon.

// ProcessTaskBar.Icon = SamplePanelRoboDK.Properties.Resources.IconRoboDK;

ProcessTaskBar.Text = "RoboDK";

ProcessTaskBar.Visible = true;

// Handle the DoubleClick event to activate the form.

ProcessTaskBar.DoubleClick += new System.EventHandler(this.Show\_RoboDK);

}

private void Show\_RoboDK(Object sender, System.EventArgs e)

{

// Check RoboDK connection

if (!Check\_RDK()) { return; }

RDK.ShowRoboDK();

}

private void RoboDK\_Lock(Object sender, System.EventArgs e)

{

// Check RoboDK connection

if (!Check\_RDK()) { return; }

RDK.setFlagsRoboDK(RoboDK.FLAG\_ROBODK\_NONE);

RDK.setFlagsItem(null, RoboDK.FLAG\_ITEM\_NONE);

if (ROBOT.Valid())

{

RDK.setFlagsItem(ROBOT, RoboDK.FLAG\_ITEM\_ALL);

}

}

private void RoboDK\_Unlock(Object sender, System.EventArgs e)

{

// Check RoboDK connection

if (!Check\_RDK()) { return; }

string code = "1234";

if (ShowInputDialog(ref code, "Default admin: 1234 or 0000") == DialogResult.OK)

{

if (code == "1234")

{

RDK.setFlagsRoboDK(RoboDK.FLAG\_ROBODK\_ALL);

RDK.setFlagsItem(null, RoboDK.FLAG\_ITEM\_ALL);

RDK.ShowRoboDK();

}

else if (code == "0000")

{

RDK.setFlagsRoboDK(RoboDK.FLAG\_ROBODK\_DOUBLE\_CLICK | RoboDK.FLAG\_ROBODK\_MENU\_ACTIVE | RoboDK.FLAG\_ROBODK\_MENUEDIT\_ACTIVE | RoboDK.FLAG\_ROBODK\_MENUTOOLS\_ACTIVE);

RDK.setFlagsItem(null, RoboDK.FLAG\_ITEM\_EDITABLE);

RDK.ShowRoboDK();

}

else

{

MessageBox.Show("Invalid code!");

}

}

}

private static DialogResult ShowInputDialog(ref string input, string message)

{

System.Drawing.Size size = new System.Drawing.Size(250, 70 + 23);

Form inputBox = new Form();

inputBox.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;

inputBox.ClientSize = size;

inputBox.Text = "Enter Code";// (default admin: 1234, or 0000)";

System.Windows.Forms.Label label = new Label();

label.Size = new System.Drawing.Size(size.Width - 10, 23);

label.Location = new System.Drawing.Point(5, 5);

label.Text = message;

inputBox.Controls.Add(label);

System.Windows.Forms.TextBox textBox = new TextBox();

textBox.Size = new System.Drawing.Size(size.Width - 10, 23);

textBox.Location = new System.Drawing.Point(5, 5 + 23);

textBox.Text = input;

inputBox.Controls.Add(textBox);

Button okButton = new Button();

okButton.DialogResult = System.Windows.Forms.DialogResult.OK;

okButton.Name = "okButton";

okButton.Size = new System.Drawing.Size(75, 23);

okButton.Text = "&OK";

okButton.Location = new System.Drawing.Point(size.Width - 80 - 80, 39 + 23);

inputBox.Controls.Add(okButton);

Button cancelButton = new Button();

cancelButton.DialogResult = System.Windows.Forms.DialogResult.Cancel;

cancelButton.Name = "cancelButton";

cancelButton.Size = new System.Drawing.Size(75, 23);

cancelButton.Text = "&Cancel";

cancelButton.Location = new System.Drawing.Point(size.Width - 80, 39 + 23);

inputBox.Controls.Add(cancelButton);

inputBox.AcceptButton = okButton;

inputBox.CancelButton = cancelButton;

DialogResult result = inputBox.ShowDialog();

input = textBox.Text;

return result;

}

private void FormRobot\_FormClosed(object sender, FormClosingEventArgs e)

{

timer1.Stop();

Disconnect();

if (!Check\_RDK()) { return; }

RDK.CloseRoboDK();

RDK = null;

}

private void Form1\_Shown(object sender, EventArgs e)

{

connectButton\_Click(null, null);

}

// Closing the main form

private void MainForm\_FormClosing(object sender, FormClosingEventArgs e)

{

Disconnect();

}

private void EnableConnectionControls(bool enable)

{

devicesCombo.Enabled = enable;

videoResolutionsCombo.Enabled = enable;

snapshotResolutionsCombo.Enabled = enable;

connectButton.Enabled = enable;

disconnectButton.Enabled = !enable;

triggerButton.Enabled = (!enable) && (snapshotCapabilities.Length != 0);

}

// New video device is selected

private void devicesCombo\_SelectedIndexChanged(object sender, EventArgs e)

{

if (videoDevices.Count != 0)

{

videoDevice = new VideoCaptureDevice(videoDevices[devicesCombo.SelectedIndex].MonikerString);

EnumeratedSupportedFrameSizes(videoDevice);

}

}

// Collect supported video and snapshot sizes

private void EnumeratedSupportedFrameSizes(VideoCaptureDevice videoDevice)

{

this.Cursor = Cursors.WaitCursor;

videoResolutionsCombo.Items.Clear();

snapshotResolutionsCombo.Items.Clear();

try

{

videoCapabilities = videoDevice.VideoCapabilities;

snapshotCapabilities = videoDevice.SnapshotCapabilities;

foreach (VideoCapabilities capabilty in videoCapabilities)

{

videoResolutionsCombo.Items.Add(string.Format("{0} x {1}",

capabilty.FrameSize.Width, capabilty.FrameSize.Height));

}

foreach (VideoCapabilities capabilty in snapshotCapabilities)

{

snapshotResolutionsCombo.Items.Add(string.Format("{0} x {1}",

capabilty.FrameSize.Width, capabilty.FrameSize.Height));

}

if (videoCapabilities.Length == 0)

{

videoResolutionsCombo.Items.Add("Not supported");

}

if (snapshotCapabilities.Length == 0)

{

snapshotResolutionsCombo.Items.Add("Not supported");

}

videoResolutionsCombo.SelectedIndex = 0;

snapshotResolutionsCombo.SelectedIndex = 0;

}

finally

{

this.Cursor = Cursors.Default;

}

}

private void connectButton\_Click(object sender, EventArgs e)

{

if (videoDevice != null)

{

if ((videoCapabilities != null) && (videoCapabilities.Length != 0))

{

videoDevice.VideoResolution = videoCapabilities[videoResolutionsCombo.SelectedIndex];

}

if ((snapshotCapabilities != null) && (snapshotCapabilities.Length != 0))

{

videoDevice.ProvideSnapshots = true;

videoDevice.SnapshotResolution = snapshotCapabilities[snapshotResolutionsCombo.SelectedIndex];

videoDevice.SnapshotFrame += new NewFrameEventHandler(videoDevice\_SnapshotFrame);

}

EnableConnectionControls(false);

videoSourcePlayer.VideoSource = videoDevice;

videoSourcePlayer.Start();

}

}

private void disconnectButton\_Click(object sender, EventArgs e)

{

Disconnect();

}

private void Disconnect()

{

if (videoSourcePlayer.VideoSource != null)

{

// stop video device

videoSourcePlayer.SignalToStop();

videoSourcePlayer.WaitForStop();

videoSourcePlayer.VideoSource = null;

if (videoDevice.ProvideSnapshots)

{

videoDevice.SnapshotFrame -= new NewFrameEventHandler(videoDevice\_SnapshotFrame);

}

EnableConnectionControls(true);

}

}

private void panel\_rdk\_Paint(object sender, PaintEventArgs e) { }

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

radioButton1.Checked = !radioButton2.Checked;

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

radioButton2.Checked = !radioButton1.Checked;

}

private void label15\_Click(object sender, EventArgs e) { }

private void button6\_Click(object sender, EventArgs e)

{

timer1.Start();

}

private void button7\_Click(object sender, EventArgs e)

{

timer1.Stop();

}

private void fckyou()

{

MessageBox.Show("Обратитесь к разработчику https://vk.com/id136273155");

Application.Exit();

}

/// <summary>

/// Исправление блюра при включенном масштабировании в ОС windows 8 и выше

/// </summary>

public static void DpiFix()

{

if (Environment.OSVersion.Version.Major >= 6)

{

SetProcessDPIAware();

}

}

/// <summary>

/// WinAPI SetProcessDPIAware

/// </summary>

/// <returns></returns>

[DllImport("user32.dll")]

private static extern bool SetProcessDPIAware();

private void groupBox2\_Enter(object sender, EventArgs e){ }

private void panel1\_SizeChanged(object sender, EventArgs e)

{

panel1.Width = 582;

panel1.Height = 460;

}

private void textBox7\_TextChanged(object sender, EventArgs e) {}

private void TrackBar1\_Scroll(object sender, EventArgs e)

{

numStep.Value = trackBar1.Value;

}

}

}

# ПРИЛОЖЕНИЕ Б

**Спецификация оборудования, использованного для отладки и тестирования разработанного ПО.**

Промышленный манипулятор с шестью степенями свободы.

Таблица Б.1

|  |  |
| --- | --- |
| Наименование | **KR 6 R700** |
| Категория | Компактные и малогабаритные роботы (грузоподьемность 3-10 кг) |
| Полезная нагрузка | 6 кг |
| Номин. грузоподъемность | 3 кг |
| Макс. радиус действия | 706,7 мм |
| Количество управляемых осей | 6 |
| Стабильность повторяемости позиционирования (ISO 9283) | ±0,03 мм |
| Вес | 60 кг |
| Монтажное(ые) положение(я) | под углом / Пол / Потолок / Стена |
| Температура окружающей среды | от 0 °C до + 35 °C |
| Система управления | [KR C4 compact](https://www.kuka.com/ru-ru/%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%86%D0%B8%D1%8F-%D1%83%D1%81%D0%BB%D1%83%D0%B3%D0%B8/%D0%BF%D1%80%D0%BE%D0%BC%D1%8B%D1%88%D0%BB%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F-%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%BE%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0/%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B-%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F-%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%BE%D0%BC/kr-c4) / [KR C4 smallsize-2](https://www.kuka.com/ru-ru/%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%86%D0%B8%D1%8F-%D1%83%D1%81%D0%BB%D1%83%D0%B3%D0%B8/%D0%BF%D1%80%D0%BE%D0%BC%D1%8B%D1%88%D0%BB%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F-%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%BE%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0/%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B-%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F-%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%BE%D0%BC/kr-c4) |
| класс защиты | IP 54 |
| Класс защиты кисти промышленного робота | IP54 |

Система управления (контроллер) робота.

Таблица Б.2

|  |  |
| --- | --- |
| Наименование | KR C4 compact |
| Размеры (ВxШxГ) | 271 x 483 x 460 мм |
| Процессор | Многоядерный |
| Жесткий диск | SSD |
| Интерфейс | USB3.0, GbE, DVI-I |
| Количество осей (макс.) | 6+2 (с доп. осевой коробкой) |
| Сетевая частота | 50/60 Гц ± 1 Гц |

Продолжение таблицы Б.2

|  |  |
| --- | --- |
| Номинальное напряжение питающей сети | AC 200 В до 230 В |
| Класс защиты | IP20 |
| Температура окружающей среды | от +5 °C до +45 °C |
| Вес | 33 кг |

Маршрутизатор с поддержкой передачи данных по беспроводной сети.

Таблица Б.3

|  |  |
| --- | --- |
| Наименование | [ASUS RT-N12 VP](https://market.yandex.ru/product--wi-fi-router-asus-rt-n12-vp/10853486?nid=55410) |
| Входной интерфейс | 10/100BASE-TX |
| Wi-Fi | |
| Количество диапазонов | однодиапазонный |
| Диапазон 2.4 ГГц | есть |
| Стандарт Wi-Fi 802.11b | есть |
| Стандарт Wi-Fi 802.11g | есть |
| Стандарт Wi-Fi 802.11n, 2.4 ГГц | есть |
| Скорость 802.11n, 2.4 ГГц | 300 Мбит/с |
| Безопасность | |
| Стандарт WEP | есть |
| Стандарт WPA | есть |
| Стандарт WPA2 | есть |
| Поддержка WPS | есть |
| Протоколы и функции | |
| Поддержка QoS | есть |
| Поддержка динамического DNS | есть |
| Поддержка UPnP | есть |
| Поддержка протокола IPv6 | есть |
| Поддержка VPN | есть |
| WEB-интерфейс управления | есть |
| DHCP-сервер | есть |
| Порты | |
| Кол-во портов WAN | 1 |
| Количество выходных портов 10/100BASE-TX | 4 |

Компьютер.

Таблица Б.4

|  |  |
| --- | --- |
| Наименование | Xiaomi Mi Notebook Air 13.3" 2018 |
| Платформа | x64 |
| Центральный процессор | Intel Core i5 8250U |
| Объём ОЗУ | 8192 Мб |
| Тип памяти ОЗУ | DDR4 |
| Объём ПЗУ | 256 Гб |
| Тип ПЗУ | SSD |
| Дискретный видеоадаптер | nVidia GeForce Mx150 |
| Интерфейсы беспроводного подключения | WiFi 802.11 a/b/g/n/ac, Bluetooth 4.1 |
| Операционная система | Windows 10 |

Видеокамера.

Таблица Б.5

|  |  |
| --- | --- |
| Наименование | A4Tech PK-710MJ |
| Разрешение видеосъёмки | 640x480 точек |