Skoltech

RobotX

Летний интенсив 2022

Компьютерное зрение

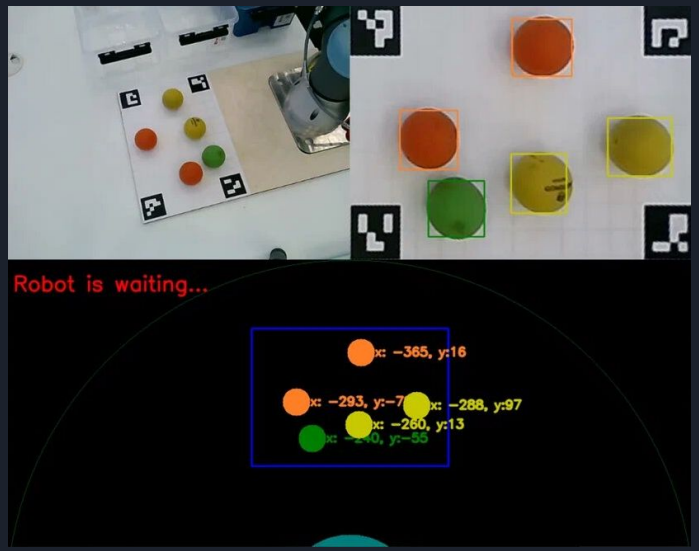
Преподаватели курса:

- Бурмистров Степан
- Федосеев Алексей

Руководитель лаборатории:

- Дмитрий Тетерюков

Задачи курса:



- Изучение алгоритмов компьютерного зрения
- Поиск и распознавание объектов
- Управление коллаборативным роботом-манипулятором UR3

Их разыскивает Россия!

Тимошин Иван
Смагин Никита
Жданов Степан
Палади Максим
Гартуев Максим
Громаков Максим
Громакова Варвара
Тегнирядноv Артём
Тихонов Константин

Калибровка камеры

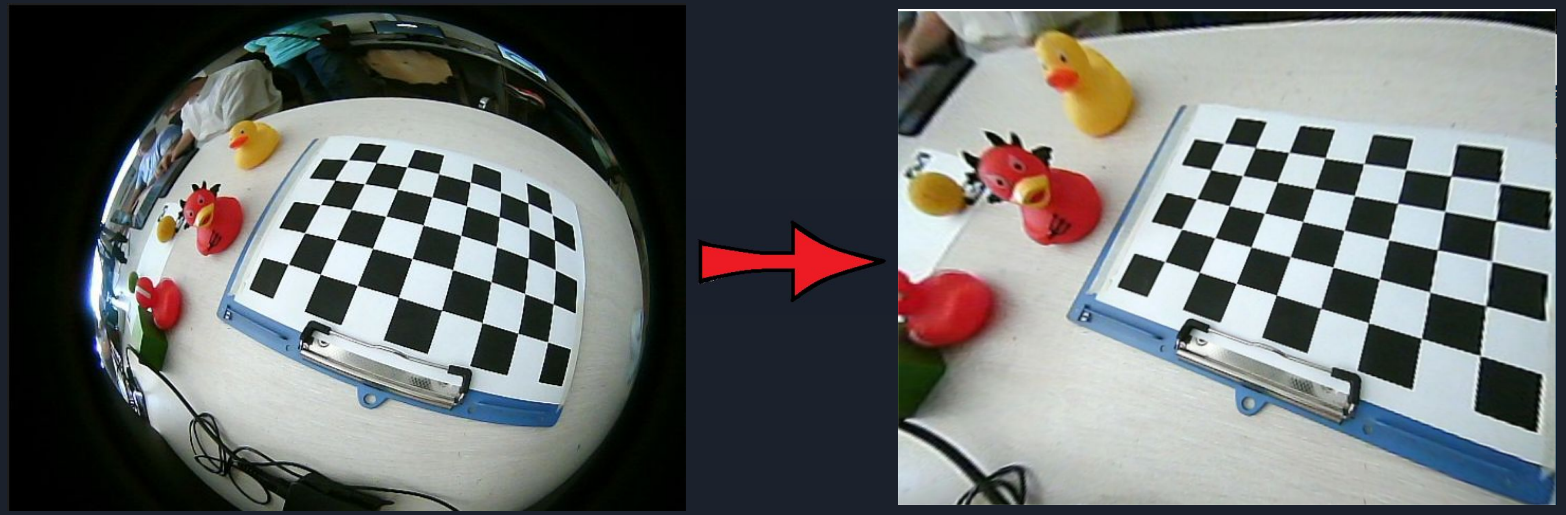
Skoltech

RobotX

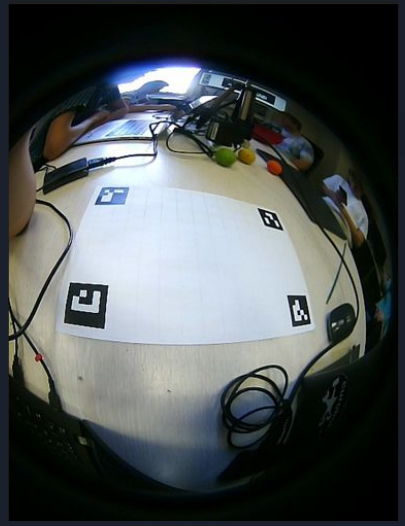
Шахматная доска

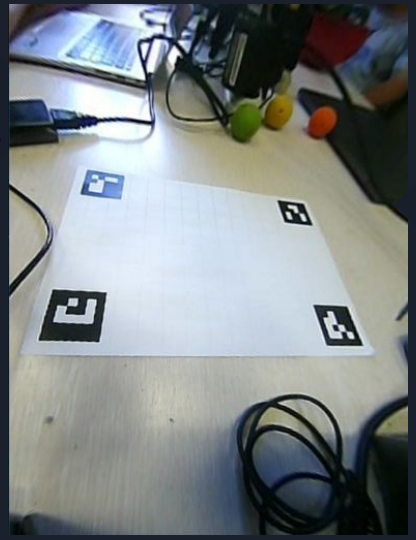
Skoltech

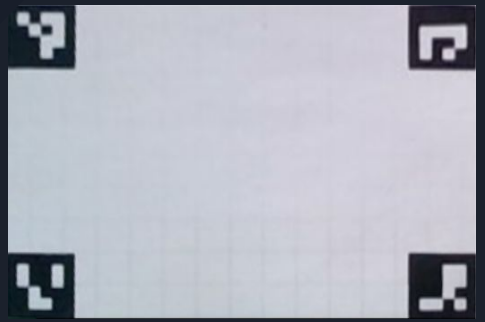
RobotX



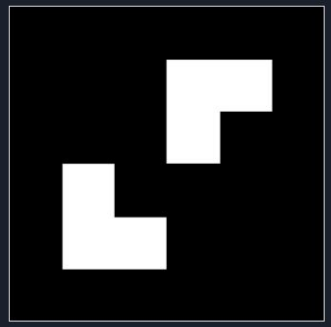
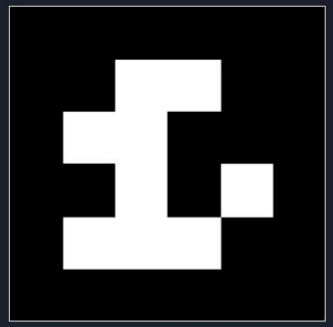
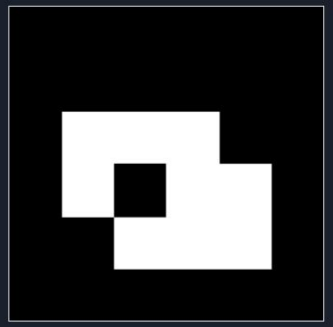
Результат





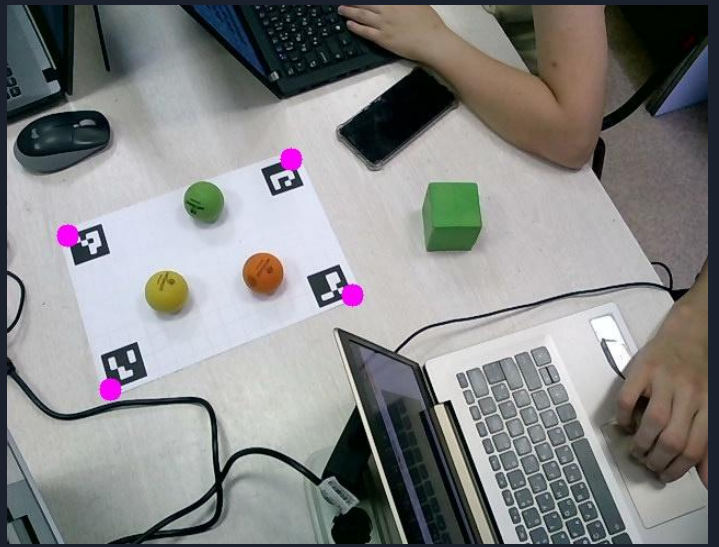


Aruco-маркеры

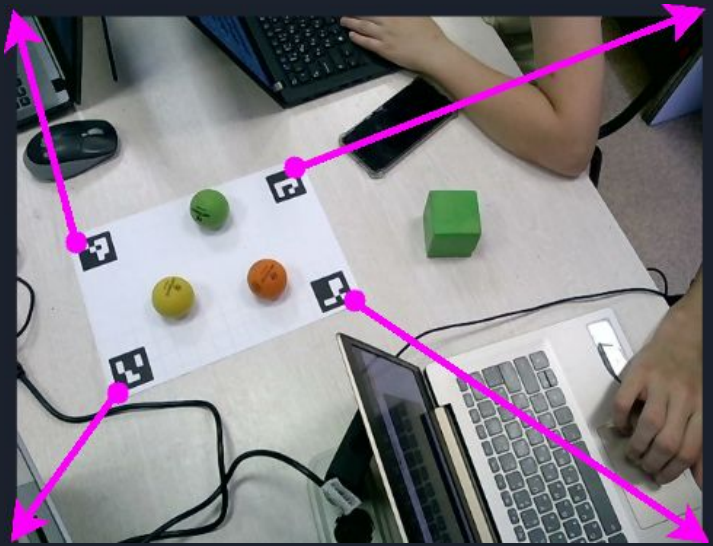

Поиск Aruco-маркеров

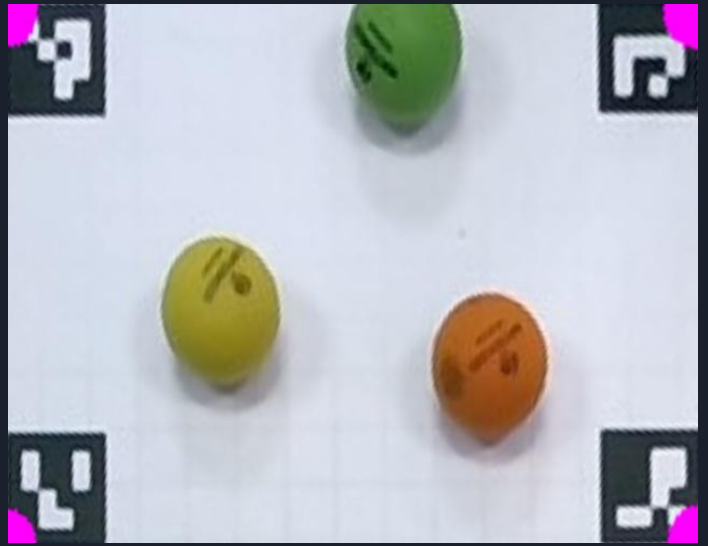
- Обнаружение маркеров с помощью cv2.aruco.detectMarkers.
- Нахождение их крайних точек.



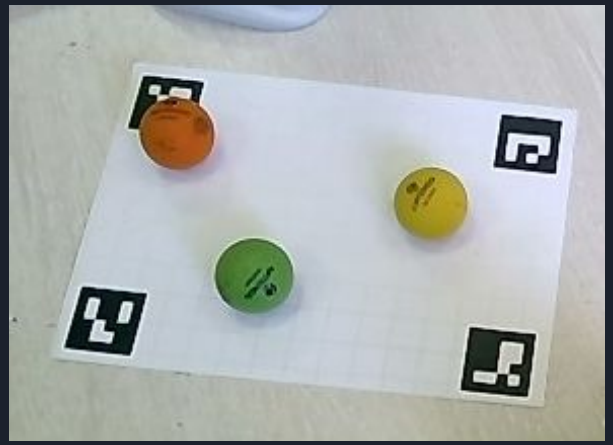
Исправление перспективы поля

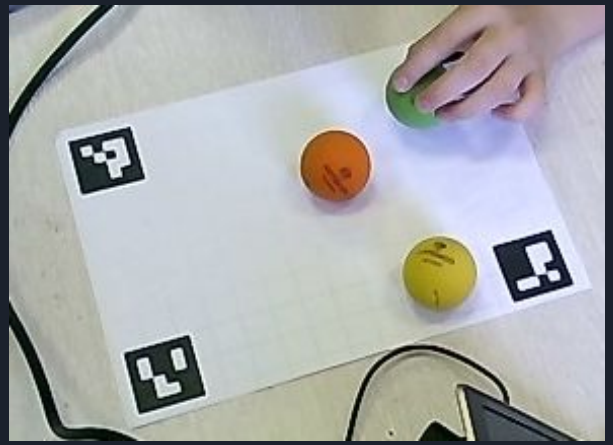
После получения крайних точек
растягиваем изображение



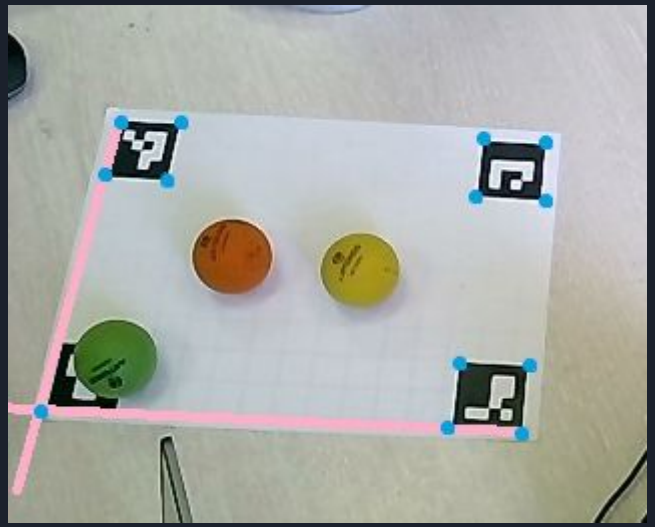


А что если маркер закрыт?





Математический способ



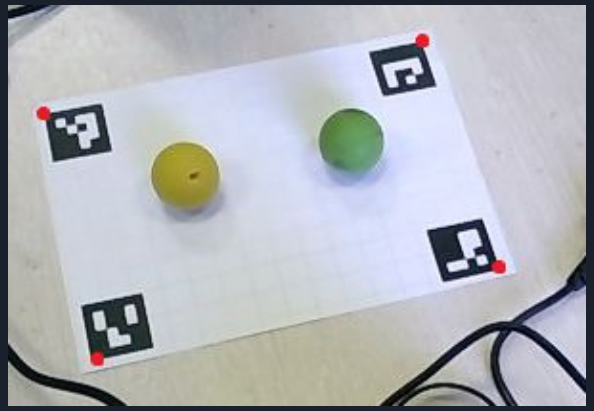
$$k = \frac{y2 - y1}{x2 - x1}$$

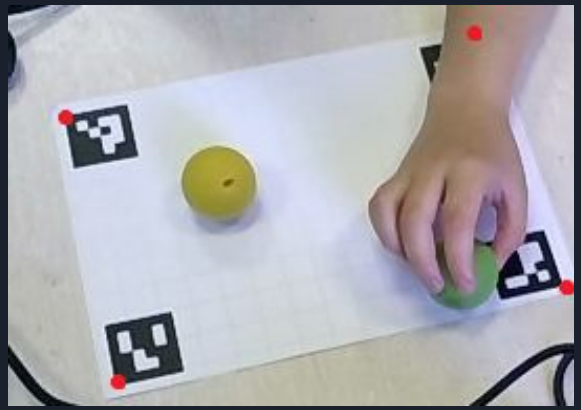
$$b = y2 - k * x1$$

$$x = \frac{b2 - b1}{k2 - k1}$$

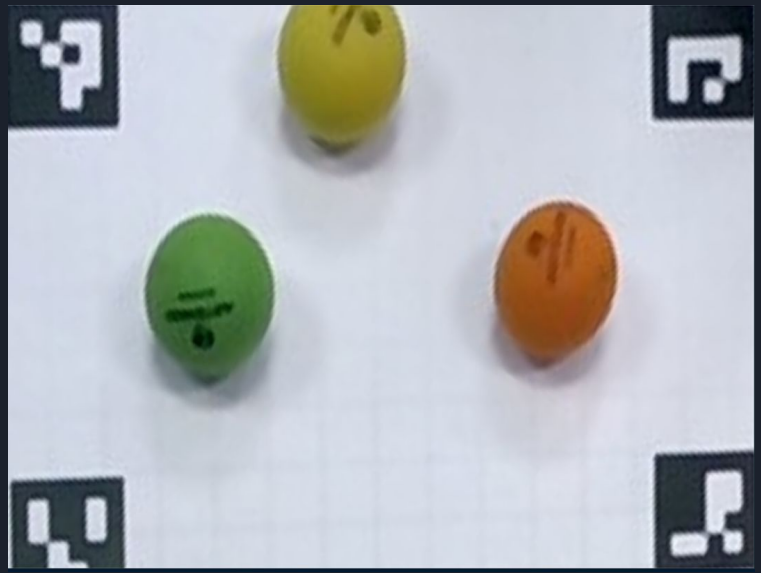
$$y = k2 - x * b1$$

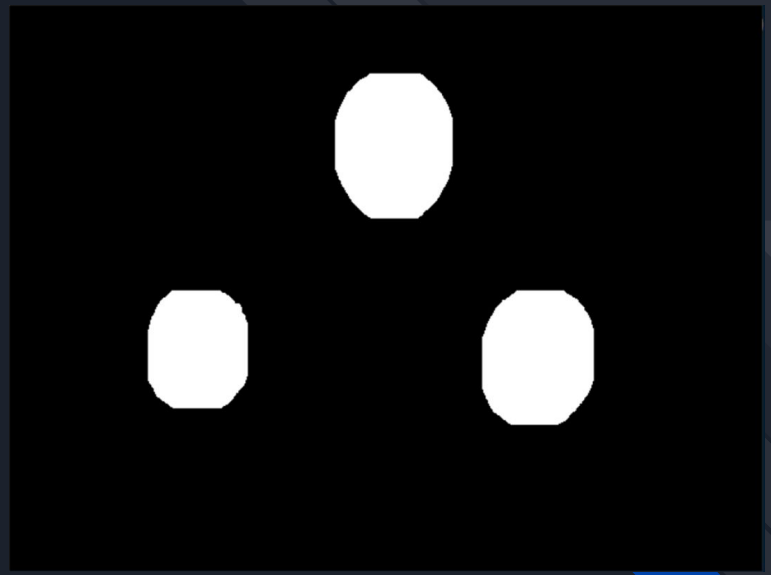
Метод сохранения данных



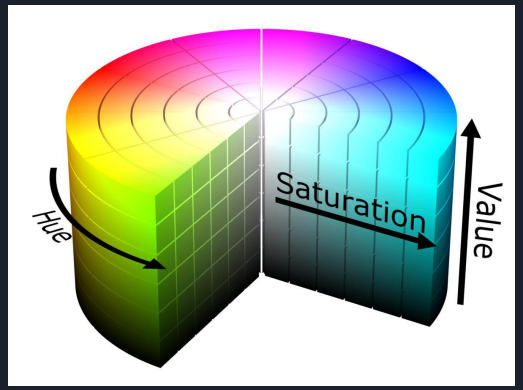


Создание бинаризованых масок для объектов разных цветов



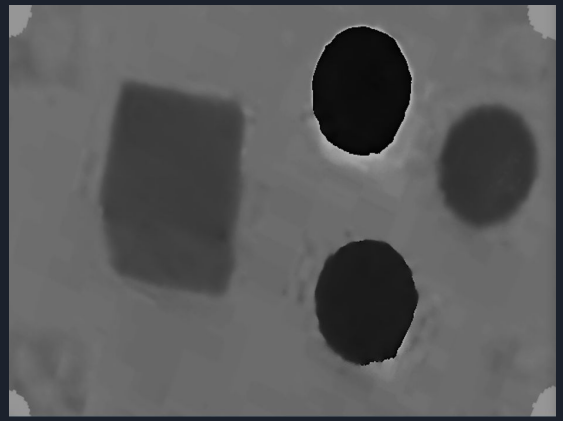


Skoltech
RobotX

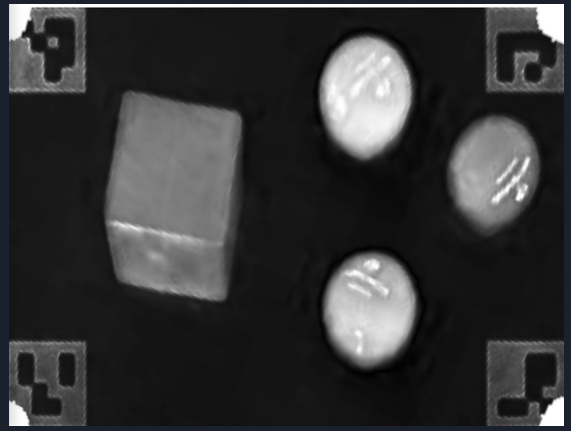


HSV

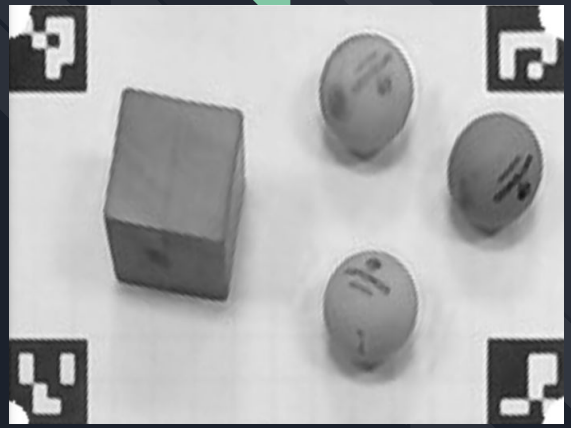
Hue

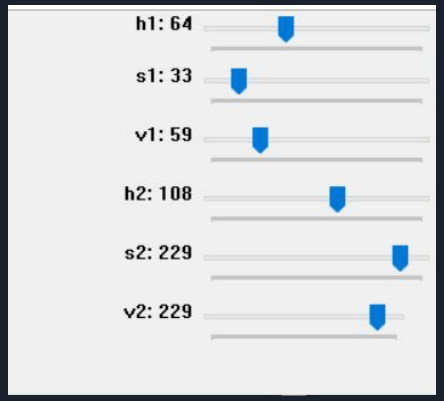


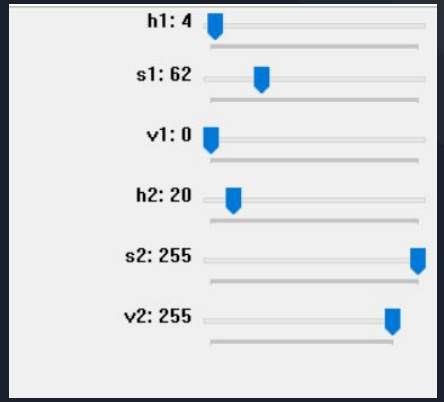
Saturation

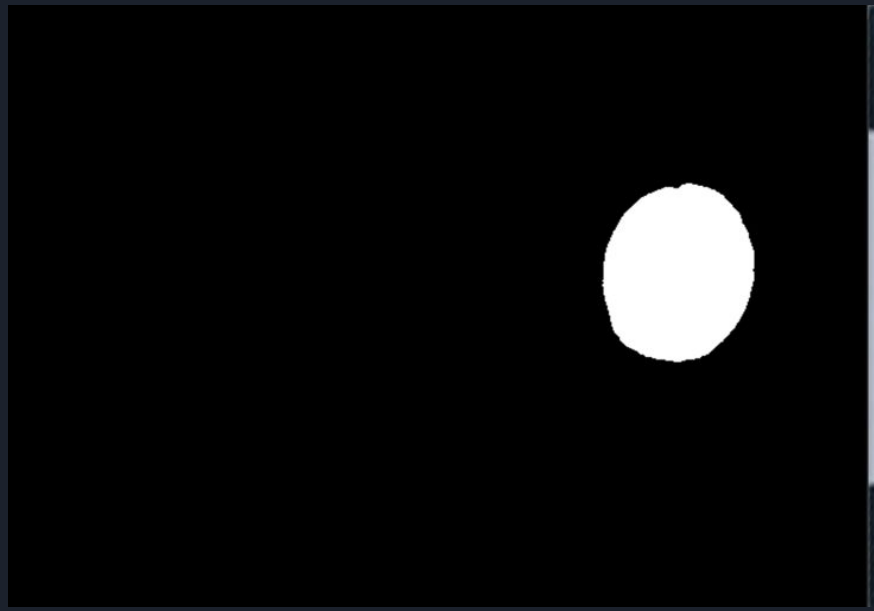


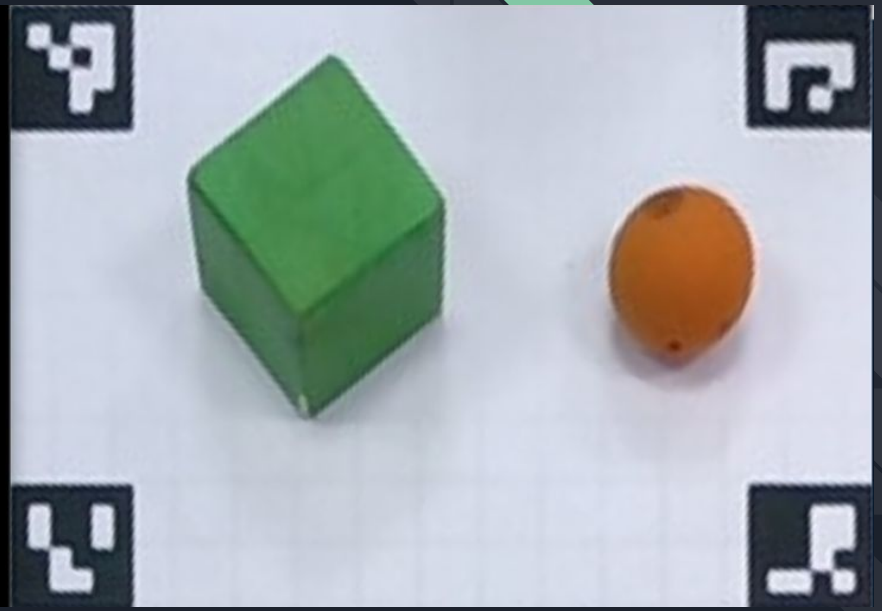
Value



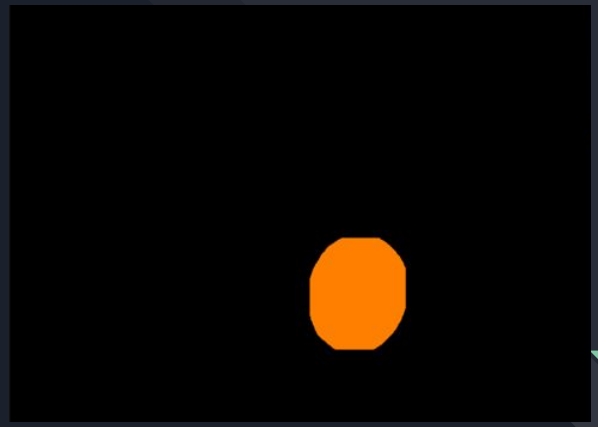


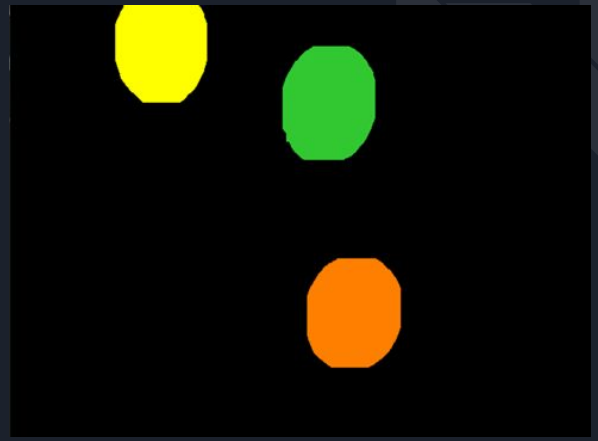






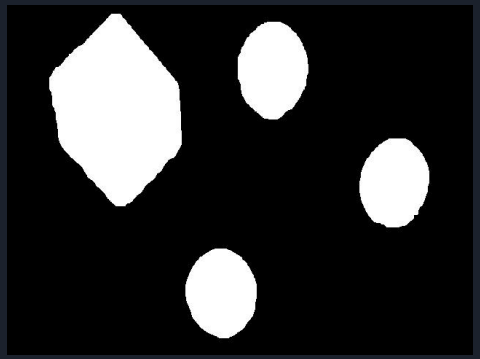
Skoltech RobotX

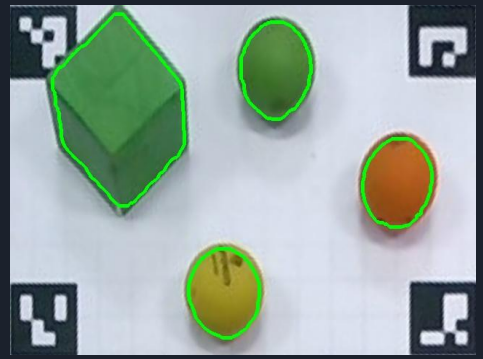




Нахождение контуров объекта

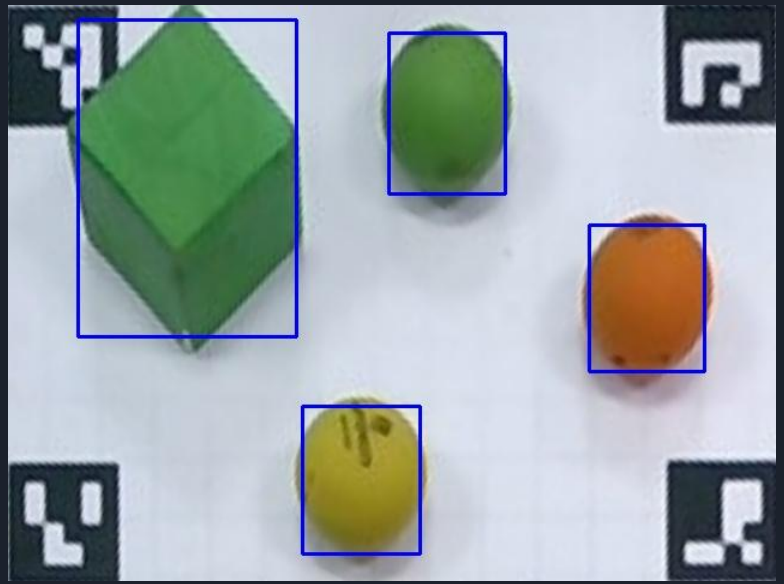
```
contours, hierarchy = cv2.findContours( img_bin,
                                        cv2.RETR_TREE,
                                        cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(ol_img, contours, -1, (0,255,0), 3)
```





Создание обводящей рамки

```
for i in range(len(contours)):
    x, y, w, h = cv2.boundingRect(contours[i])
    cv2.rectangle(field, (x, y), (x+w, y+h), (255,0,0), 2)
```

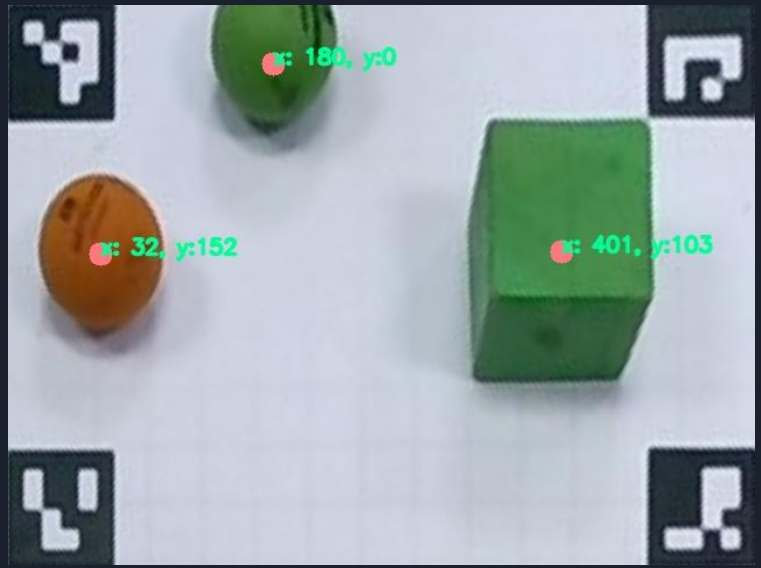


Определение центра объекта, зная его крайние точки

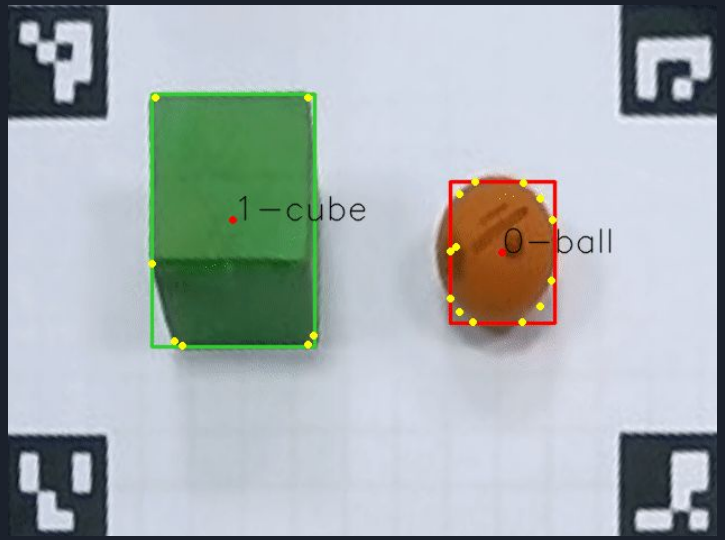
Skoltech

RobotX

```
x_center = x + w // 2

y_center = y + h // 2

x_mm = 287*x_center//width

y_mm = 200*y_center//height

cv2.circle(res_img, (x_center,y_center), 10, (128,120,255), thickness=-1)

cv2.putText(res_img, f"x: {x_mm}, y:{y_mm}",
            (x_center,y_center),
            cv2.FONT_HERSHEY_SIMPLEX,
            0.6, (0,255,0), 2, cv2.LINE_AA)
```

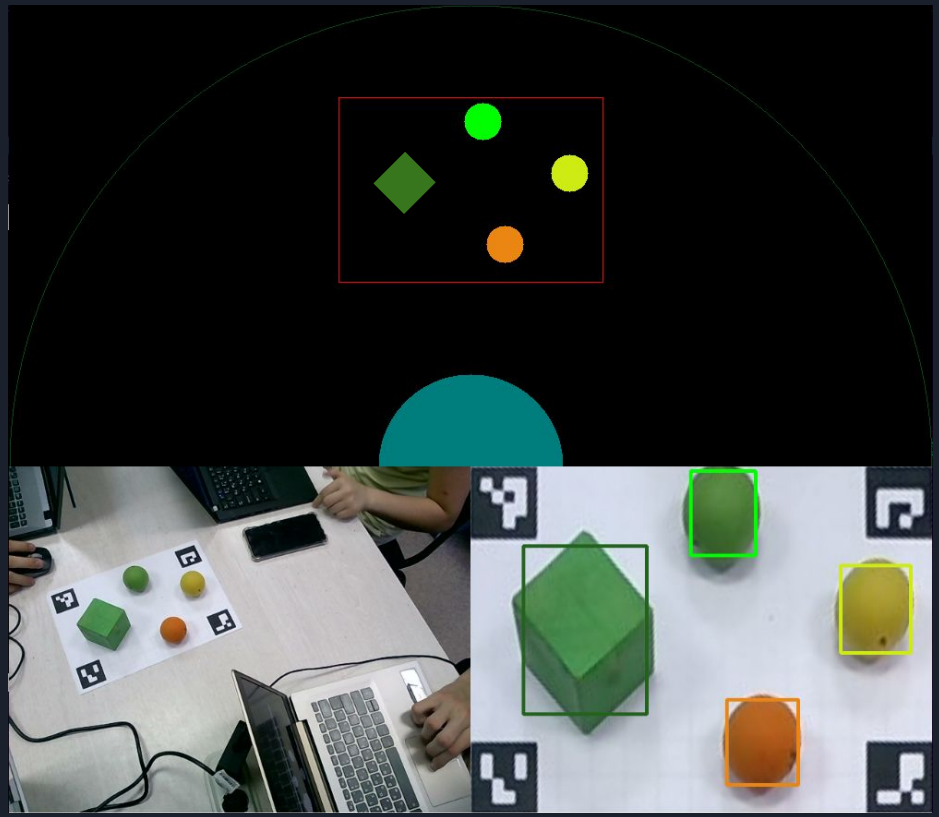


Нахождение формы объектов




```
area = cv2.contourArea(contour)

perimeter = cv2.arcLength(contour, True)
approx = cv2.approxPolyDP(contour, 0.005 * perimeter,
True)

x, y, w, h = cv2.boundingRect(contour)

if len(approx) <= 11:
   if w * h - area < 5000:
       shape = 1 # cube 0 deg
   else:
       shape = 2 # cube 45 deg
else:
   shape = 0 # ball
```

CNN для классификации объектов на поле

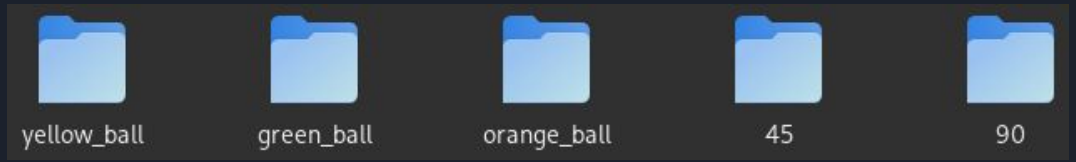


Датасет

Количество классов - 5
Классы:

- кубик 45°
- кубик 90°
- жёлтый шарик
- оранжевый шарик
- зелёный шарик

Размер изображений: **128x128x3**
Количество тренировочных картинок для каждого класса ~**200**



Архитектура CNN

```
import torch
from torch import nn


class CustomConvNet(nn.Module):
    def __init__(self, num_classes):
        super(CustomConvNet, self).__init__()
        self.layer1 = self.conv_module(3, 16)
        self.layer2 = self.conv_module(16, 32)
        self.layer3 = self.conv_module(32, 64)
        self.layer4 = self.conv_module(64, 128)
        self.layer5 = self.conv_module(128, 256)
        self.gap = self.global_avg_pool(256,
num_classes)

    def forward(self, x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = self.layer3(out)
        out = self.layer4(out)
        out = self.layer5(out)
        out = self.gap(out)
        out = out.view(-1, 5) # 5 - num classes
        return out
```

5 свёрточных слоёв

```
def conv_module(self, in_num, out_num):
        return nn.Sequential(
            nn.Conv2d(in_num, out_num, kernel_size=3,
stride=1, padding=1),
            nn.BatchNorm2d(out_num),
            nn.LeakyReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))
```

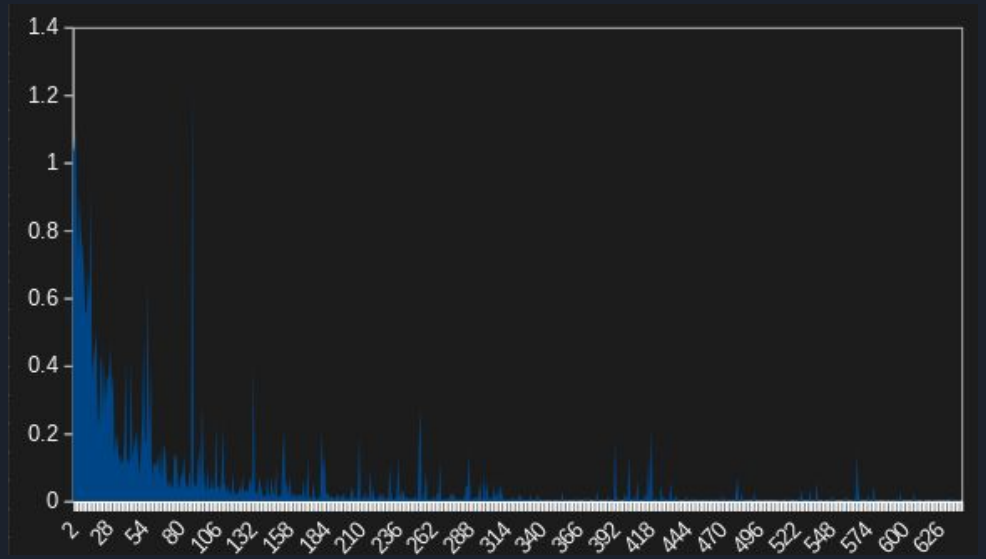
Выход нейросети:

```
tensor([[0.4581, 0.5374, 0.3437, 0.4208, 0.3797],
        [0.3316, 0.2519, 0.4646, 0.4176, 0.3997]],
grad_fn=<ViewBackward0>)
torch.Size([2, 5])
```

Обучение

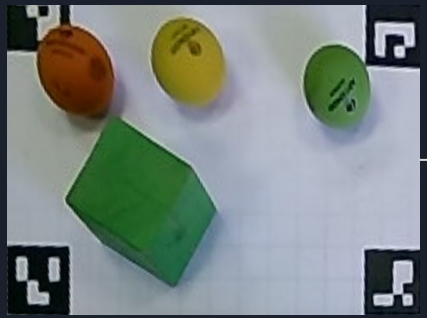
```
#Гиперпараметры
hyper_param_epoch = 80
hyper_param_batch = 8
hyper_param_learning_rate = 0.001
```

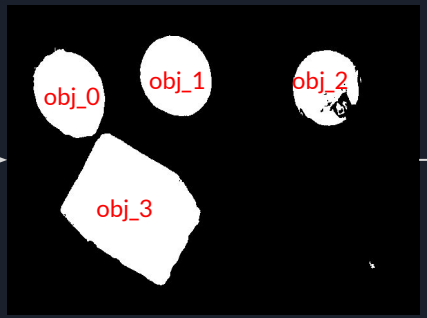
```
Epoch [78/80], Loss: 0.0009
Epoch [78/80], Loss: 0.0010
Epoch [78/80], Loss: 0.0004
Epoch [78/80], Loss: 0.0003
Epoch [79/80], Loss: 0.0008
Epoch [79/80], Loss: 0.0008
Epoch [79/80], Loss: 0.0002
Epoch [79/80], Loss: 0.0010
Epoch [79/80], Loss: 0.0008
Epoch [79/80], Loss: 0.0007
Epoch [79/80], Loss: 0.0003
Epoch [79/80], Loss: 0.0007
Epoch [80/80], Loss: 0.0132
Epoch [80/80], Loss: 0.0005
Epoch [80/80], Loss: 0.0065
Epoch [80/80], Loss: 0.0012
Epoch [80/80], Loss: 0.0012
Epoch [80/80], Loss: 0.0045
Epoch [80/80], Loss: 0.0007
Epoch [80/80], Loss: 0.0003
```

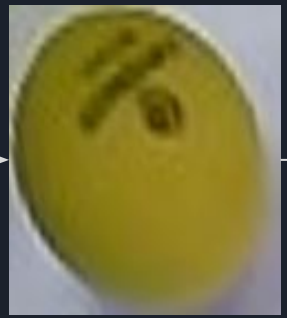


Инференс и обработка результатов

```
#Основные этапы обработки изображения
transforms_test = transforms.Compose([transforms.Resize((128, 128)),
                                      transforms.ToTensor()])

object_to_nn = field_img[y - 5: y + h + 10, x - 5: x + w + 10] # вырезка объекта из кадра

object_to_nn = transforms_test(object_to_nn).unsqueeze(0) # преобразование

out = model_classificator(object_to_nn) # инференс на нейронной сети

_, predicted = torch.max(out.data, 1) # объект с максимальной вероятностью

final_object_type_id = predicted.cpu().numpy()[0] # предполагаемый id класса объекта
classes = {0: "90 deg cube rotated", 1: "Ball orange", 2: "Ball yellow", 3: "Ball green", 4: "45 deg
cube rotated"}
```







tensor([[0.1567, -0.0217,
6.4791, 1.1930, -0.0247]])

Тестирование обученной модели

Инференс на Tesla K80 ~**390 FPS**

Инференс на CPU ~ **90 FPS**

GPU - Nvidia Tesla K80 spec

Куда ядра: **2496**

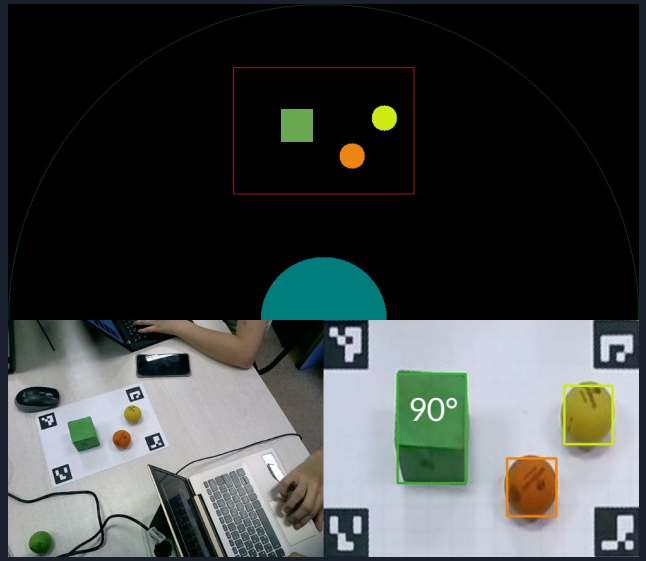
Видео-память: **12GB GDDR5**

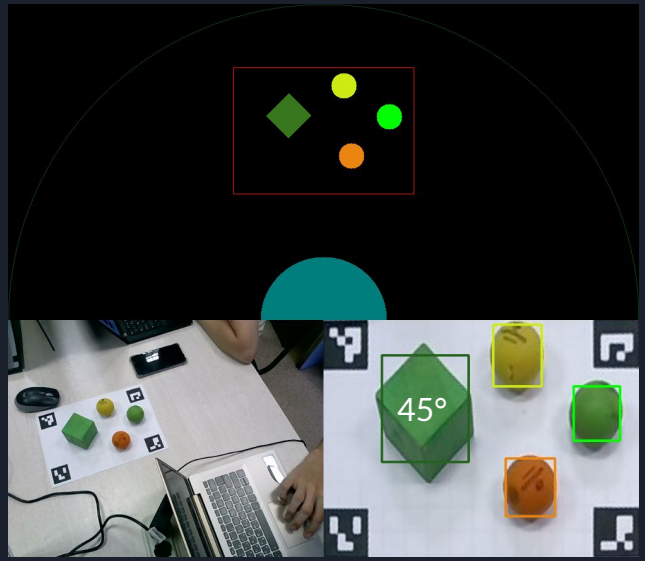
CPU - Intel Xeon spec

Частота: **2.3Ghz**

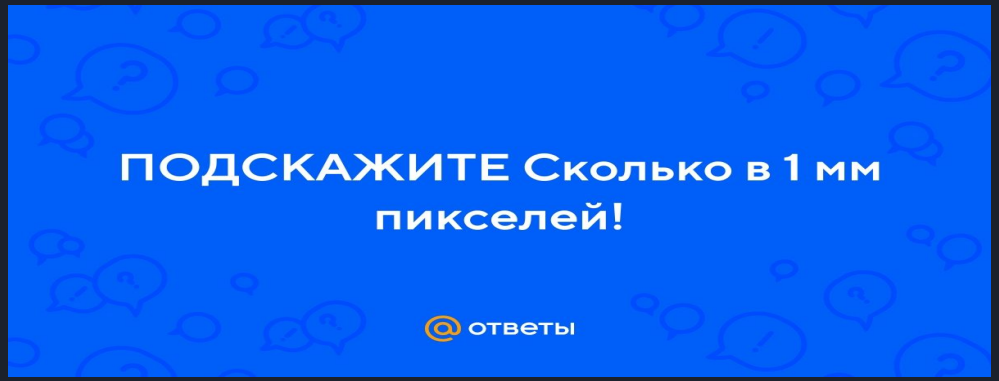
Ядра: **1 ядро, 2 потока**

Проверка на реальном поле



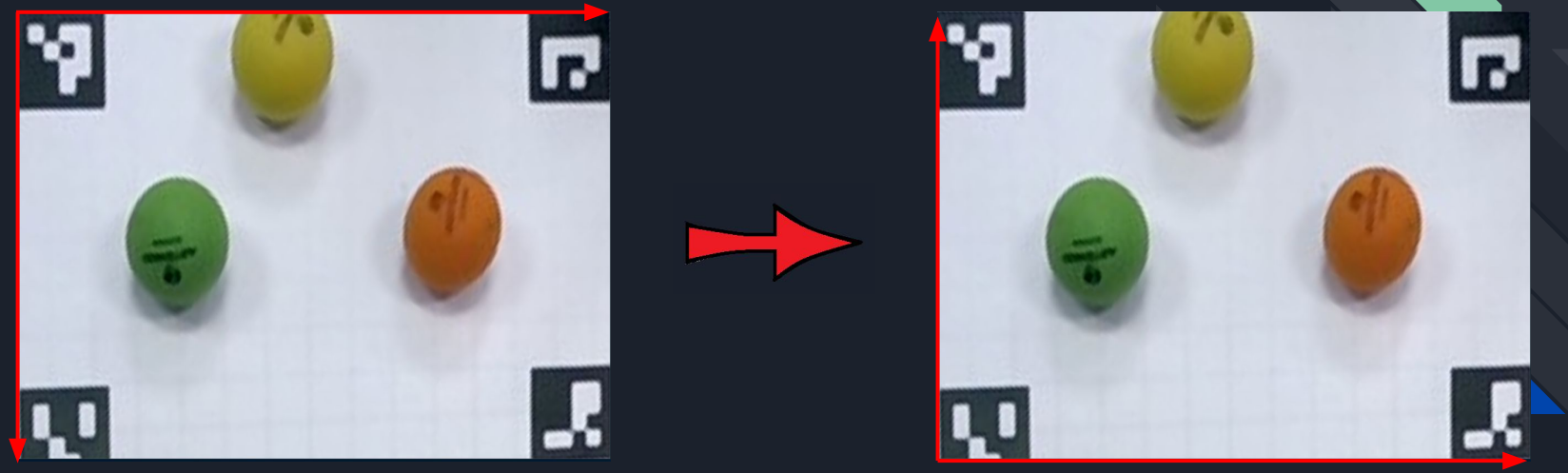


Перевод координат в миллиметры



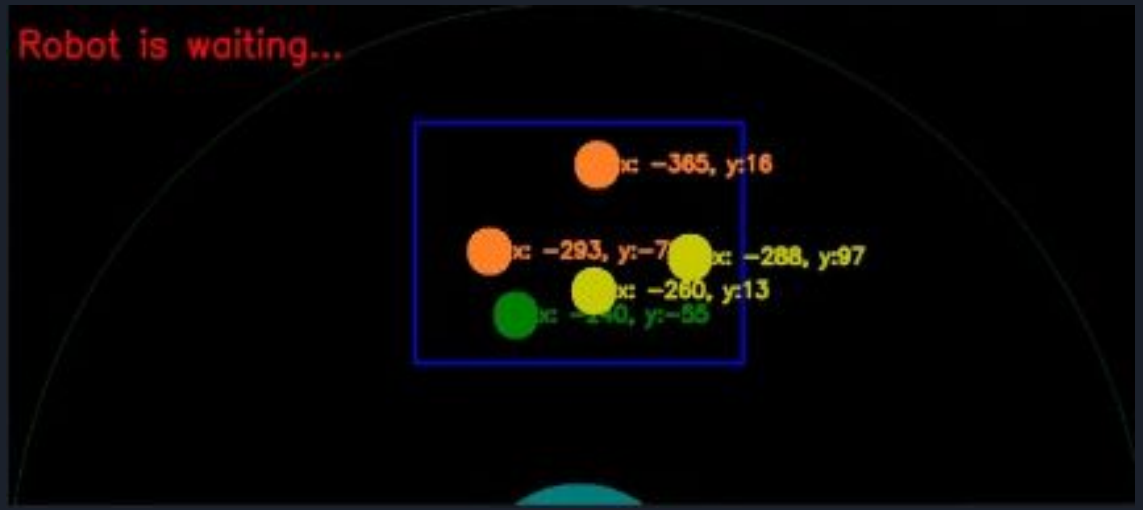

```
x_mm = int((287 / width) * x_center)
y_mm = int((200 / height) * y_center)
```

Изменяем систему координат поля

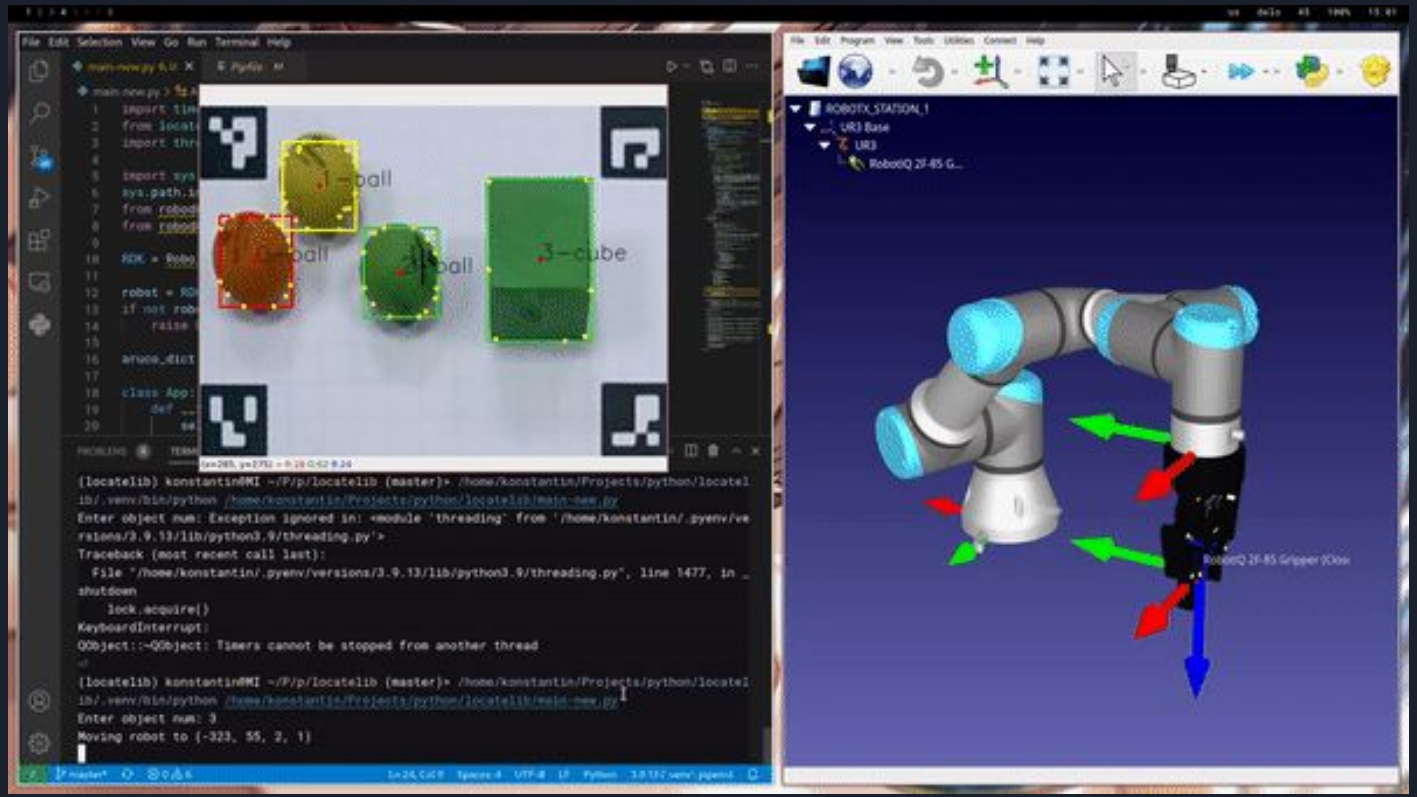


Skoltech RobotX

Перевод систему координат поля в систему координат робота



Симуляция в RoboDK



Симуляция в RoboDK

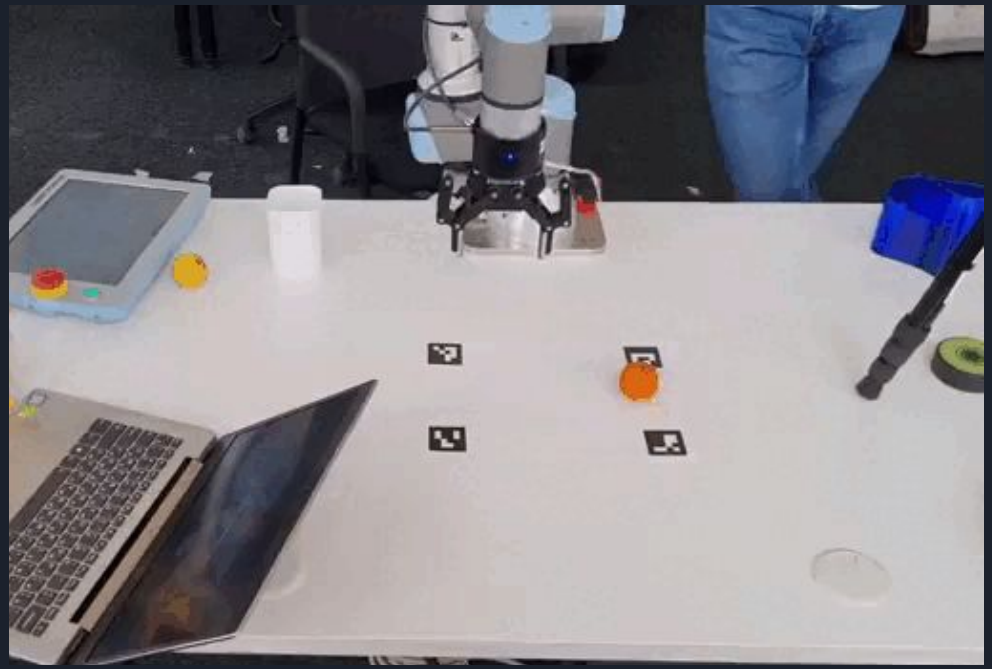
- Импорт

```
import sys
sys.path.insert(0, '/home/user/RoboDK/Python')
from robodk.robolink import *
from robodk.robomath import *

RDK = Robolink()

robot = RDK.ItemUserPick('Select a robot', ITEM_TYPE_ROBOT)
if not robot.Valid():
   raise Exception('No robot selected or available')
```

- Использование

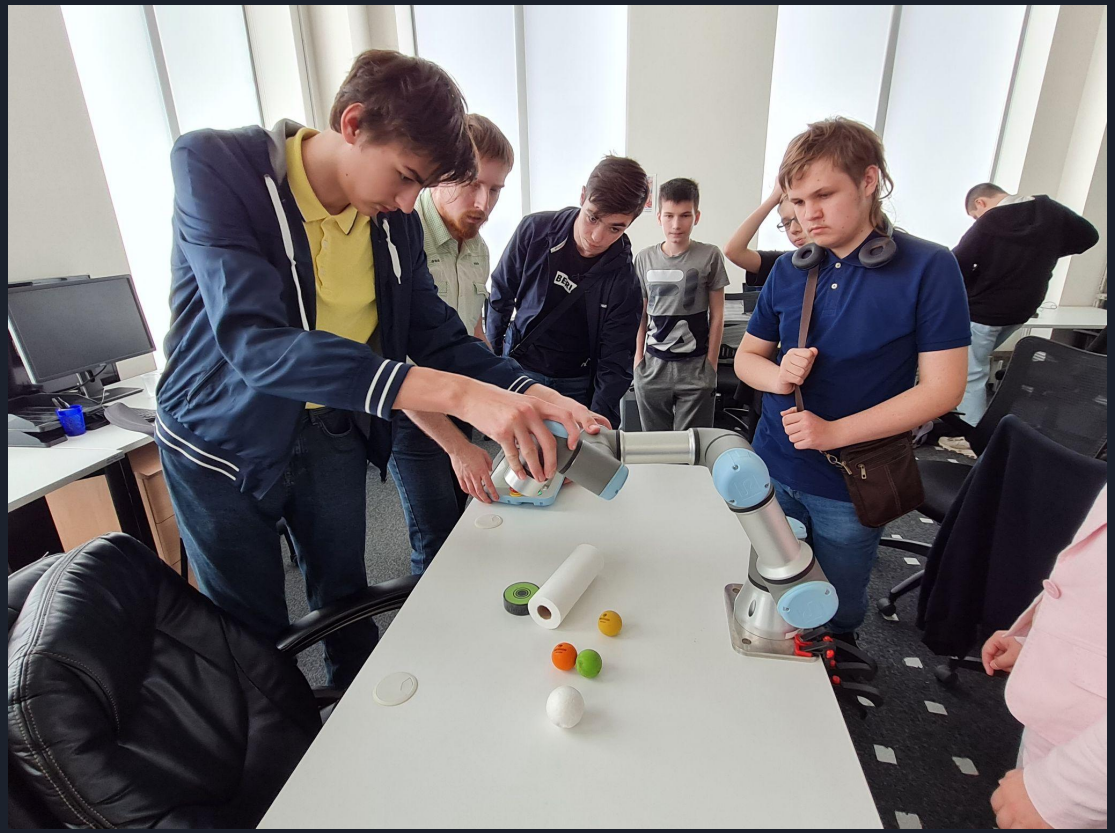
```
def movel(coord, *args, **kwargs):
       target = UR_2_Pose(coord)
       try:
           robot.MoveL(target)
       except TargetReachError:
           pass

movel([0, 0, 10, 0, 3.142, 0])
```

Робот-манипулятор UR3

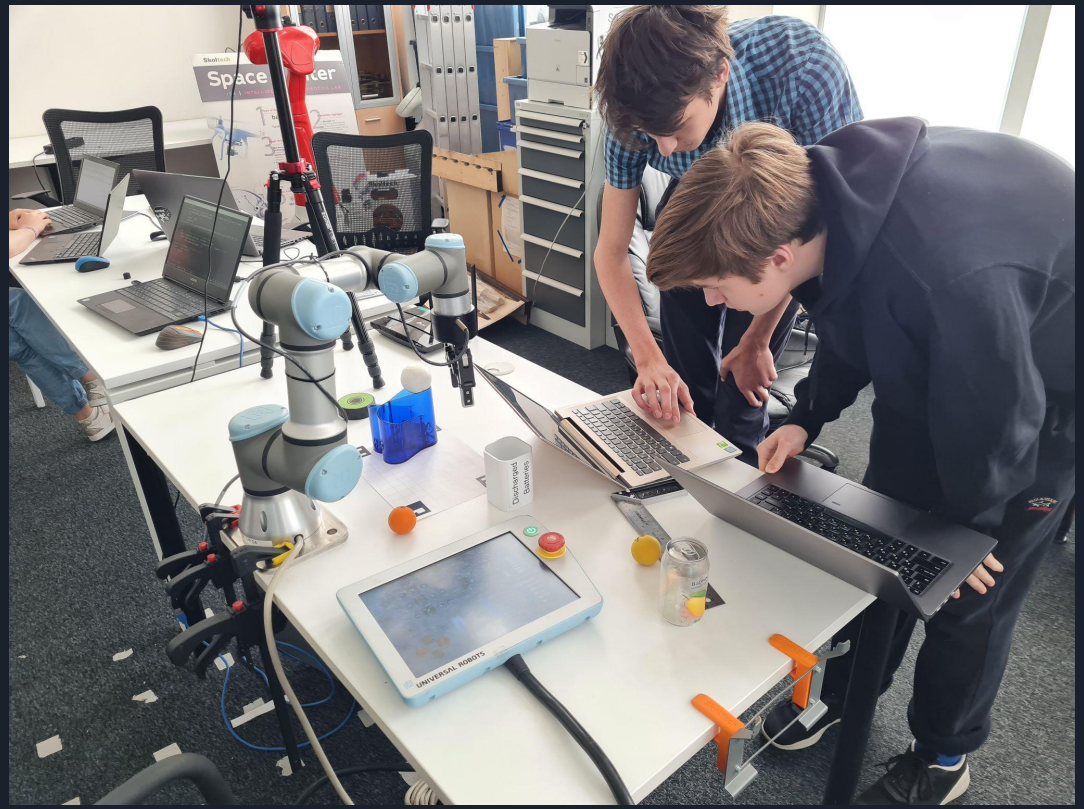


Работа манипулятора при помощи планшета

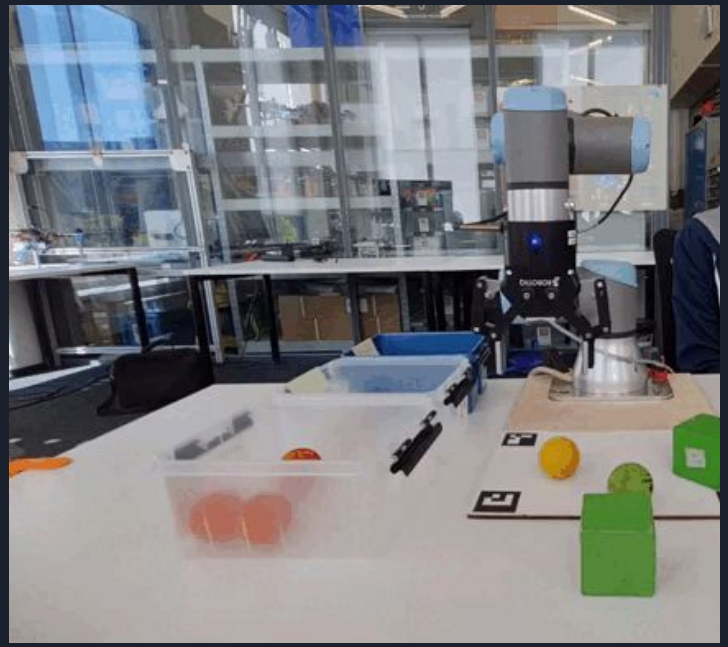
Free drive



Первое подключение к роботу



Построение маршрута робота



Работа с роботом

Movel

команда для линейного перемещение робота

координаты

ускорение

```
robot.movel([x, y, z, angle_1, angle_2, angle_3]), acc=acc , vel=velocity, wait=True)
```

максимальная скорость

Спасибо за внимание!

https://github.com/robotxschool/CV_June_2022

stepan_burmistrov

alexeyfas