Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Computational Engineering and Technical Physics
Computer Vision and Pattern Recognition

Aleksandr Gubanov

SMART GRASPING OF KNOWN OBJECTS

Master's Thesis

Examiners:	Associate Professor Arto Kaarna
	Professor Arkadii Soloviev
Supervisors:	Associate Professor Arto Kaarna
	Professor Arkadii Soloviev

ABSTRACT

Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Computational Engineering and Technical Physics
Computer Vision and Pattern Recognition

Aleksandr Gubanov

Smart grasping of known objects

Master's Thesis

2020

46 pages, 26 figures.

Examiners: Associate Professor Arto Kaarna
Professor Arkadii Soloviev

Keywords: computer vision, image processing, pattern recognition, robotics, machine vision, deep learning

Smart grasping means that a robot can automatically decide which object and how an object can be grasped. Firstly, a neural network should find the objects and get a 2D bounding box. The convolution neural network is used for processing the frames to detect the objects. Secondly, a proposed approach should support the robot to understand where it is possible to grasp the object. The decision should be done independently and autonomously such that reliable grasping becomes possible. In the experiment there are only a limited set of known objects. The quick addition of new objects is possible since an automatic process for extracting new objects from video was developed. The result of the thesis provides an automated system which connects the recognition system and the robot.

CONTENTS

LIST OF ABBREVIATIONS

2-D	Two-dimensional
3-D	Three-dimensional
API	Application Program Interface
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
CVPRL	Computer Vision and Pattern Recognition Laboratory
DIB-R	Differentiable interpolation-based renderer
FPS	Frames Per Second
GPU	Graphics Processing Unit
HOG	Histogram of Oriented Gradients
LSD	Large-Scale Direct
KF	Key Frame
MRF	Markov Random Field
ORB	Oriented fast and Rotated Brief
PWM	Pulse-Width Modulation
R-CNN	Region-based Convolutional Neural Networks
ROS	Robot Operating System
SLAM	Simultaneous localization and mapping
SSD	Single Shot Detector
SVM	Support vector machine
YOLO	You Only Look Once

1 INTRODUCTION

1.1 Background

One of the most important modern trends in software is software with computer vision [1]. This technology has made it possible to analyze information in images and video files. A distinctive feature of computer vision is the extraction of useful and important information for us from images or image sequences. Computer vision concentrates on dealing with 3D scenes created for several images. One to several images can reconstruct the structure or other 3D scene information. Computer vision applications are extremely broad: manipulation systems (industrial robots, autonomous vehicles), video monitoring systems, data management systems (e.g., for categorization image databases), objects (Fig.1 [2]) or environmental simulation systems (medical image analysis), and interaction systems (e.g., human-machine interaction systems).

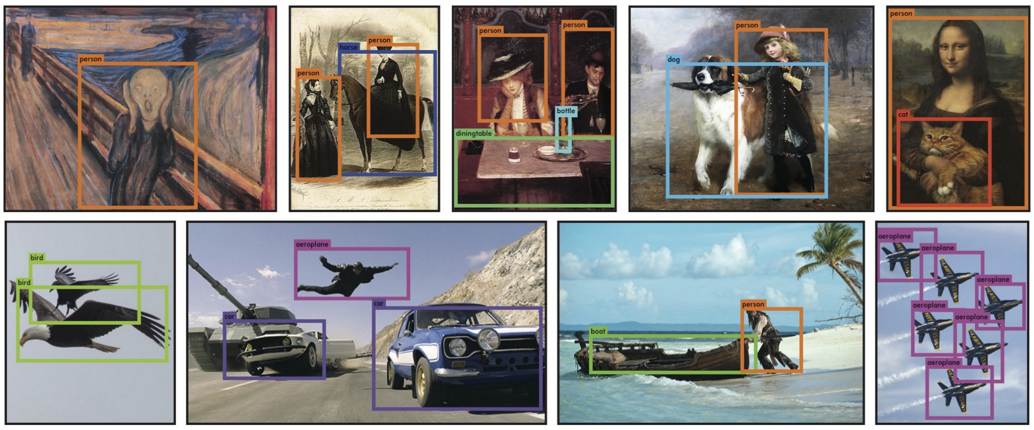


Figure 1. Examples of object recognition. [2]

Programming a modern robot to perform almost any complexity of tasks is very easy and simple. The robots have an application program interface (API), which is usually very uncomplicated to understand and implement. One such example of this technology is Robot Operating System (ROS) [3]. This is a set of various tools, libraries and specific rules aimed at simplifying the tasks of robot software development. However, solving problems where objects are unknown is still a complicated task. By all means, using already known objects facilitates the task of grabbing, as in this thesis. In the Computer Vision and Pattern Recognition Laboratory (CVPRL) [4], there is a Mitsubishi Industrial Robot RV-3SB-S312 [5] with Robotiq 3-finger gripper [6] for grasping objects.

For many robotic systems, it is necessary to have sufficient information about the surrounding area. Based on this information, the robot chooses a behavior model. Thus, the robot uses information about the structure of surrounding objects and tracks their movement, which makes the robot possible to work stably in a dynamically changing environment.

The main task of the motion control system is to plan the robot's movements to a certain target point taking into account various factors such as distance to the object (Fig. 2, green part). Information about these factors is contained in the video stream received from the camera located on a mobile platform. To do so, it is necessary to recognize the images of surrounding objects. Then, the robot control system can, on the basis of these data, correctly form the appropriate behavior of the robot to perform the task. See the algorithm of proposed system in Fig. 2. However, before starting using the system, it is necessary to prepare data for training the neural network and to train on this data (the blue and red areas in Fig. 2)

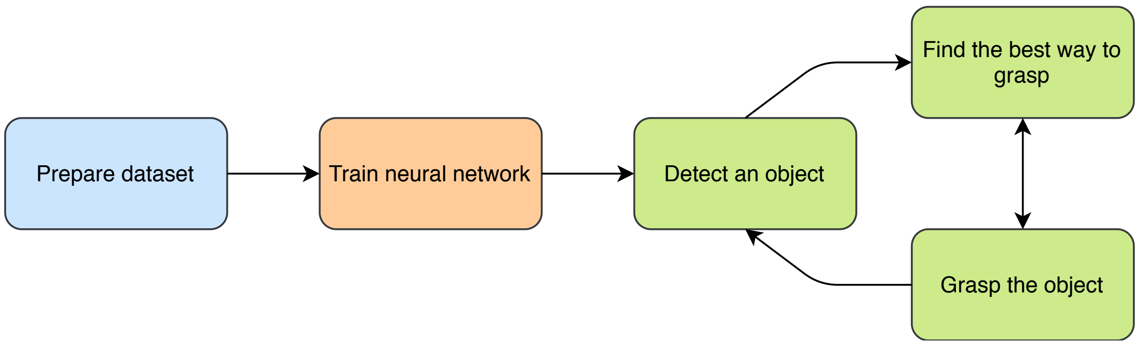


Figure 2. Proposed system.

1.2 Objectives and delimitations

The main goal of this work is to develop an automatic system which can provide some base interface for grasping a limited set of known objects of complex shapes (owl, shampoo, two different packages of salt, cup; see in Fig.3a) in the CVPRL [4]. Also, the possibility to reconstruct 3D model of a table where there are objects under research is studied in the thesis. In addition, this work studies the technology of working with ROS in combination with a web-camera and the script required for the recognition of objects.

For that purpose, the following questions are considered:

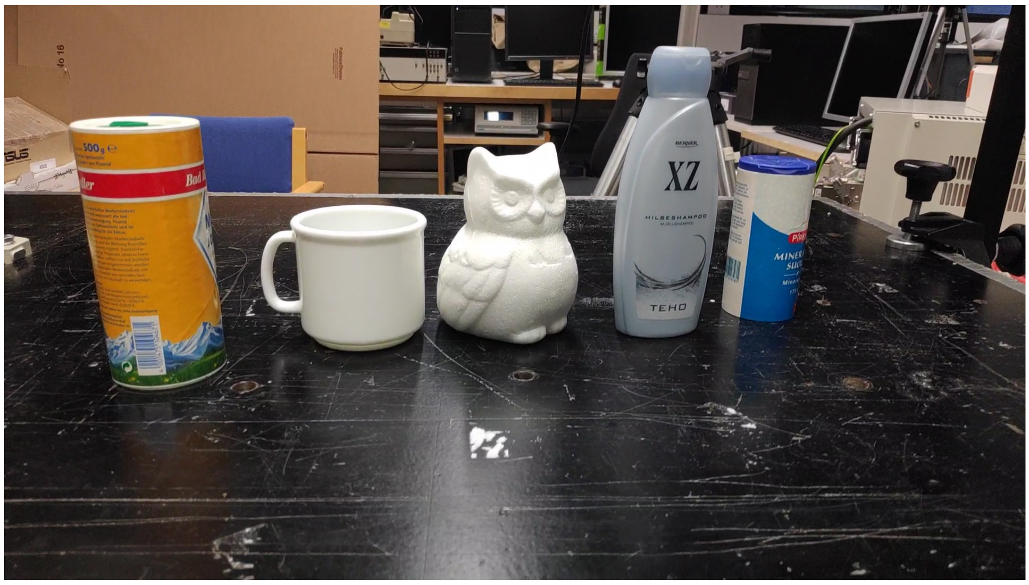
- What are the best ways for object recognition in real-time.
- Transforming from 2D to 3D world using multiple images.
- Linking ROS and the object detection system (the web-camera and the program).

The thesis considers only grasping of a limited set of object from the CVPR laboratory. In the current research existing approaches are used for CNN and ROS without creating new methods.

Since it was a coronavirus epidemic, it was decided to simulate the operation of the robot. The operation was replaced by a system consisting of an Arduino micro-controller and its sensors. The original set of objects for the new one had to be changed. The new set consists of a red bird toy, an orange cup, and a blue jug. See new set in Fig.3b.

1.3 Structure of the thesis

The rest of the thesis has the following structure. Chapter 2 gives an overview of state-of-the-art object detection algorithms. Chapter 3 and Chapter 4 contain existing approaches and algorithms for solving the grasping task. Moreover, Chapter 3 includes the answer on the question: how the robot should understand a scene. Chapter 5 contains information about proposed method. Chapter 6 defines the process of setting up and linking the robot and the recognition system into one structure and reports conducted experiments and how they were performed. Chapter 7 discusses the current results and the future study needed. Chapter 8 is the summary of the thesis.


(a) Original objects

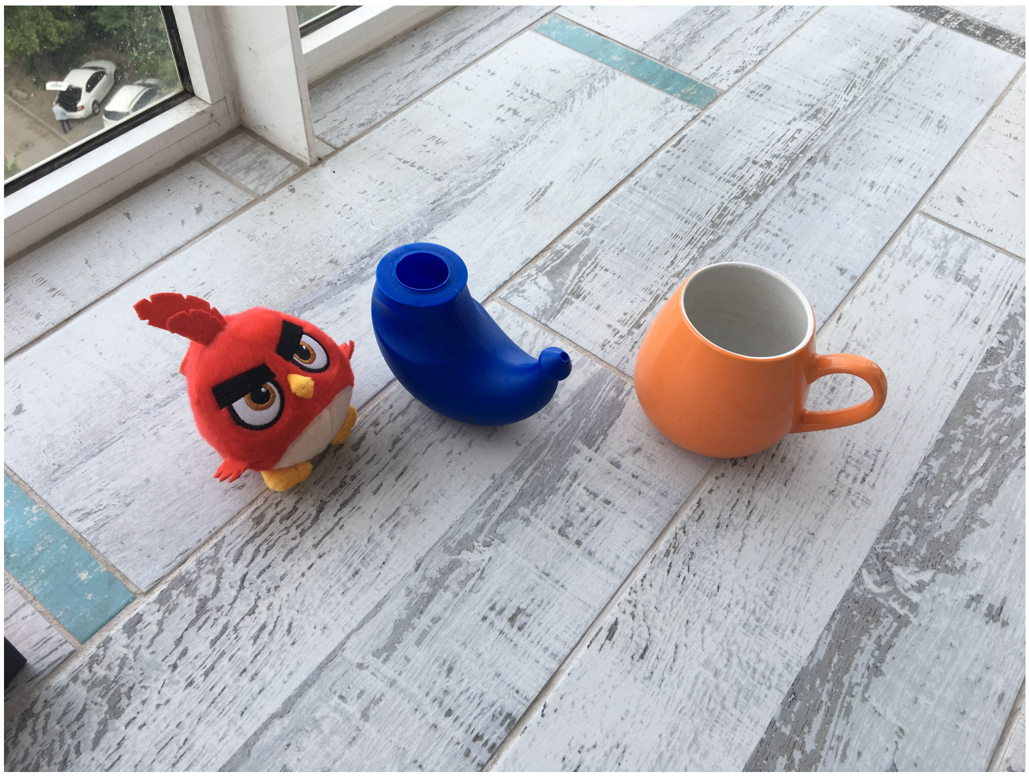

(b) New objects due to the exchange of the experimental setup

Figure 3. Objects of interest.

2 DETECTION OF AN OBJECT

2.1 Convolutional neural networks

Nowadays, recognition tasks have a vast number of solutions that can provide an answer about what is on a given image or pose of the object under a study with good accuracy. The use of new technologies has made it possible to use them in a variety of environments ranging from video control systems mounted on significant structures to self-driven vehicles. Object detection is not a classifier [7]. The main difference between these two tasks is that object identification helps to get the exact location and boundaries of the object in the image.

The foundations of the modern architecture of twin neural networks were laid in one of the first widely known twin neural networks, LeNet-5 by Yann LeCun and Yoshua Bengio [8]. A Convolutional Neural Network (CNN) is a neural network where a convolutional layer is presented [9]. Usually, convolutional neural networks also contain a sub-sampling layer (pooling layer) and a fully connected layer. In convolutional networks, the convolution and sub-sampling layers are made up of several "levels" of neurons called feature maps or channels. Each neuron in this layer relates to a slight area of the preceding layer, called a receptive field [10]. In the case of an image, the feature map is a 2D array of neurons, or simply a matrix. The architecture of CNN is shown in Fig. 4 [11].

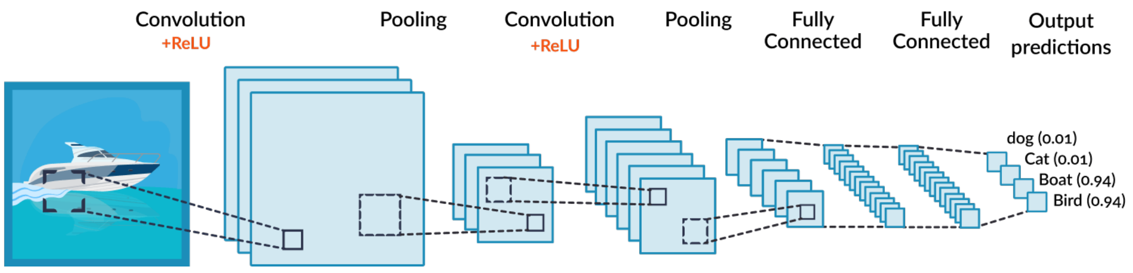


Figure 4. The architecture of CNN. [11]

In the convolution layer of each feature map there is one kernel of the convolution, also called a filter [12]. Each neuron performs the operation of convolution or mutual correlation with its receptive layer as its output value. Since there is one convolution kernel for each feature map, this allows the neural network to learn how to distinguish features regardless of their location in the input image and also leads to a significant decreasing in the number of parameters.

According to the established notation, a reconciliation layer uses a filter W $\times$ H, if each filter in this layer has dimension

$$D = W \times H \times C$$

where D is dimension, W is width, H is height, C is a dimension of channels in the previous layer. The sub-sampling layer compresses the feature maps of the preceding layer. Furthermore, the layer does not change the number of maps. Each map of the features of a layer is associated with a corresponding map of the features of the preceding layer, each neuron "abbreviate" its susceptibility field.

The most popular types of this layer are Max Pooling (the maximum value is selected from the receptive layer), Average Pooling (the average value is selected) and L2 Pooling (L2 norm is selected). With the help of the sub-sampling layer, resistance to small shifts in the input image is achieved, and the size of subsequent layers is reduced. A fully connected layer — a usual hidden layer of multi-layer perceptron associated to each neurons of the preceding layer.

Standard method of Convolutional Neural Network for object detection is not valid, because it uses various regions of interest of an image, additionally the method uses a CNN to predict the classes of targets inside the regions [13]. Moreover, diverse spatial locations and aspect of ratios can be one of the main problems of such way. Thus, it has to be used an enormous variety of regions that would require complex computer calculations.

2.2 Approaches of object identification

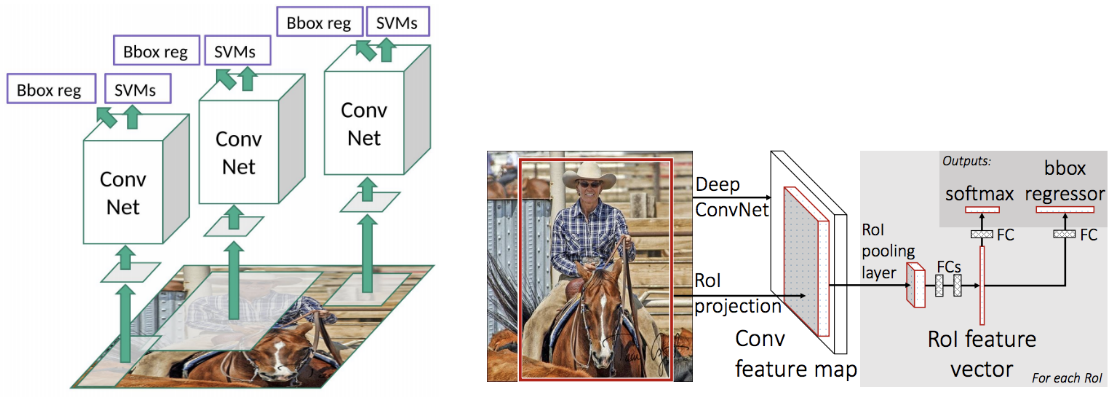
In the current work a CNN-based network is used. There are a lot of state-of-the-art algorithms which solve that kind of problem of standard method for object detection: Region-based Convolutional Neural Networks (R-CNN), Single Shot Detector (SSD), You Only Look Once (YOLO).

R-CNN

In 2013 Ross Girshick *et al.* [14] proposed an R-CNN method where selective search [15] was used to cut 2000 regions from the image. However, the R-CNN has got a small amount of drawbacks, as follows:

1. Still, computational cost if very big (2000 regions per image).
2. No real-time implementation because of a time-consuming testing for each image (around 50 seconds) [16].
3. No training takes place at this stage for the reason the selective algorithm is fixed [17].

Ross Girshick *et al.* upgraded R-CNN in 2015 [18]. The new algorithm got a name Fast R-CNN. The approach can be comparable with usual R-CNN. Though, instead of providing the regions of the image to CNN, Fast R-CNN takes the input image, and then the algorithm generates a convolutional feature map. The map's purpose is to extract the regions. The next step uses these regions to get Regions-of-Interest feature vector and using a softmax layer to predict the type of the region. The architecture of the R-CNN and Fast R-CNN is shown in Fig.5 [14, 18].



(a) (b)

Figure 5. The architecture of the algorithms: (a) R-CNN [14]; (b) Fast R-CNN [18].

Shaoqing Ren *et al.* proposed another algorithm called Faster R-CNN [19]. It is similar to Fast R-CNN, but a separate network was used to predict regions instead of using an algorithm of selective search to identify them.

Single Shot Detector

In 2015 C. Szegedy *et al.* introduced Single Shot Detector (SSD) [20]. The technology is inexpensive by computation and have a high accuracy for real-world tasks. SSD uses a single network for object detection. The model's main components are a base network block and several multi scale feature blocks connected in a series. Therefore, for recognizing purpose SSD needs to take one single shot (this means that the tasks of localization and classification of objects are performed in one direct pass of the network) to find multiple target within the image. Meanwhile, approaches with regional proposal network (for example, R-CNN) need two shots, one for creating regions, one for classify the object of each location. Therefore, SSD is vastly faster in contrast with the two-shot region ways. The architecture of SSD is shown in Fig. 6 [21].

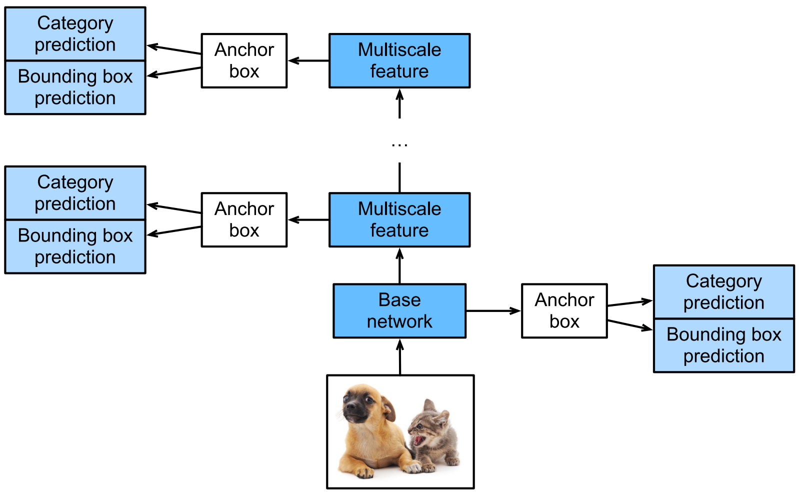


Figure 6. The architecture of SSD. [21]

You Only Look Once

The first version YOLO was introduced in 2015 [2]. After that the authors have upgraded YOLO three times. There are three main steps in the YOLO detection system (see Fig. 7 [22]). In the first step the system resizes the input image. Then YOLO applies a segregate CNN on the image. In the end the system reduces the results of recognition of the model with the help of a threshold.

The biggest advantage of the YOLO model is reflected in the name — You Only Look Once. This model applies a grid to the image, dividing it into cells. Each cell tries to predict the coordinates of the detection area with an estimate of confidence for these fields and the probability classes. The confidence estimate for each detection area is then multiplied by the probability of the class to get the final estimate. The more advanced explanation of YOLOv3 is described below [2,22].

First, during training, the YOLOv3 network is supplied with input images for predicting 3D tensors (which are the last map of objects) correlating with three scales, as shown in the graph in the center of Fig.7. These three scales are constructed to detect objects of various sizes. For the 13×13 scale, as an example [22], the input image is separated into 13×13 grid cells, each grid cell matches to a cell inside the 3D tensor with dimension:

$$N_{scale} \times N_{scale} \times N_{boxes} \times (T_{coord} + S + N_{classes}) \tag{1}$$

where N_{scale} is a scale size, N_{boxes} is a number of boxes (each grid predicts 3 boxes), T_{coord} is a box coordinates (tx, ty, tw, th), S is an objectness score(how likely it is an object), $N_{classes}$ is a number of classes, see the right part of Fig.7. Substituting the corresponding values into (1), the following formula is obtained:

$$1 \times 1 \times 3 \times (4 + 1 + 80) = 1 \times 1 \times 255$$

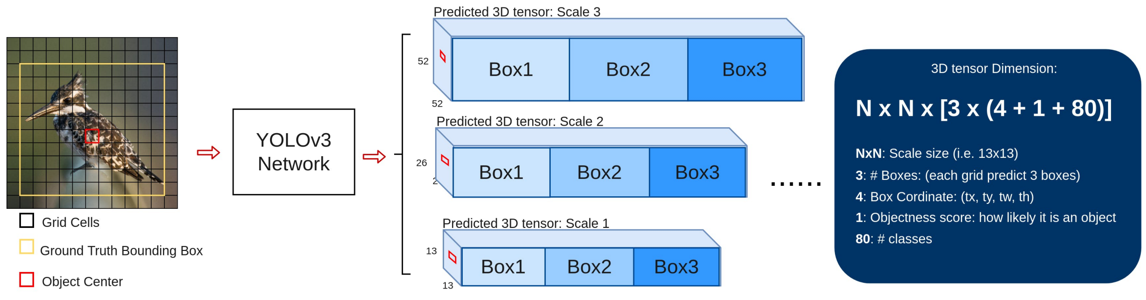
Values within the 3D tensor, for instance, the coordinates of the bounding cells, objectivity score, and class confidence, are shown to the right of the diagram (see Fig.7 [2,22]).

Secondly, if the center of the bounding boxes of the truth of the object falls into a certain grid cell, which is in charge of foretelling the bounding boxes of the object. The corresponding objectivity rate is "one" for a given grid cell and "zero" for others. For each grid cell, three previous cells of different sizes are assigned. During training, the cell learns to choose the correct square and calculate the exact offset/coordinates.

Finally, it needs to choose the initial length of these 3 early boxes. The authors use K-tool clustering to classify common blocks from a COCO collection to nine clusters in front of training [2]. This results in nine sizes selected from nine clusters, three for 3 scales. These previous parameter values are useful to study how to accurately calculate the coordinates of the boxes, because the intuitive, poor choice of container size makes learning step complicated and time-consuming for the network.

2.3 Comparison

Provided modern algorithms of object detection are the best choice for the thesis. However, there is an important requirement. The selected recognition algorithm must have a high FPS for real time operation and be of low-cost in terms of computing



Grid Cells
Ground Truth Bounding Box
Object Center
YOLOv3 Network
Predicted 3D tensor: Scale 3
52
52
Box1
Box2
Box3
Predicted 3D tensor: Scale 2
26
26
Box1
Box2
Box3
Predicted 3D tensor: Scale 1
13
13
Box1
Box2
Box3
......
3D tensor Dimension:
N x N x [3 x (4 + 1 + 80)]
NxN: Scale size (i.e. 13x13)
3: # Boxes: (each grid predict 3 boxes)
4: Box Cordinate: (tx, ty, tw, th)
1: Objectness score: how likely it is an object
80: # classes

Figure 7. YOLOv3 algorithm, consist of the three components: Left – image scale, Center – 3D tensor, Right: 3D tensor dimension. [2, 22]

resources. Derakhshani *et al.* [23] compared these algorithms and concluded that YOLOv3 is the best choice for real-time, because it has vast FPS values relative to the accuracy of the method.

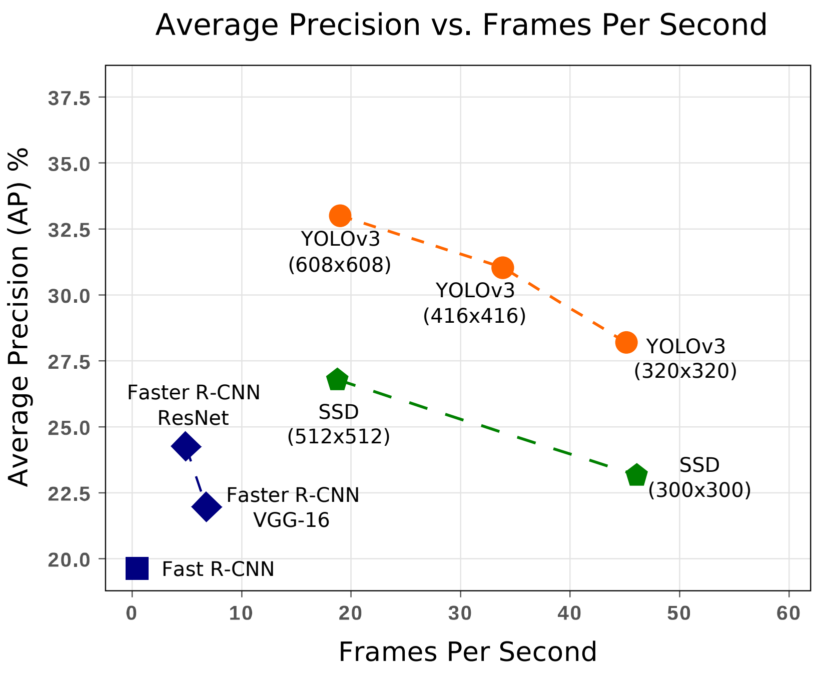


Figure 8. Comparison of modern algorithms. [23]

3 3D SCENE STRUCTURE

3.1 3D scene from 2D images

3D image processing and analysis nowadays has an essential role in many areas of research, especially in industry. In the present subsection articles of expansion of the model of the 2D image and its application to 3D images are considered.

In 2009 Ashutosh Saxena *et al.* [24] introduced the method of studying 3D area plan from an image. The researchers use Markov Random Field (MRF) for conclusion about a batch of "flat criterions" that find not only a 3D location, but also a 3D orientation of an object . MRF is a graphical model, which is used to represent joint distributions of a set of several random variables. By and large, the Markov random field is made up of the parts [25]:

- an undirected graph, where all vertexes are arbitrary variables and all edges serve as a relationship between random variables.
- a set of potential functions, one for each click (full subgraph of an undirected graph) on a graph. The function assigns to each possible click state a certain non-negative material number.

The peaks, which are not adjoining, should conform with contingently independent random values. A range of adjoining vertices forms a click, the set of vertex states is the argument of the matching possible function.

The authors segmented the images in many identical thumbnails and use MRF to make a conclusion about the 3D position and orientation of each of them [24]. However, the proposed algorithm does not make assumptions about what is exactly on the image such as the scene consist of vertical surfaces, which stands on a horizontal floor. The approach is used to show that there are several small surfaces.

Another method for 3D modeling from 2D images based on machine learning techniques. The researchers from NVIDIA have created a framework for rendering volume models from 2D images [26]. The researchers created a special framework which is called Differentiable interpolation-based renderer (DIB-R). DIB-R processes the picture and then transforms it into a highly accurate 3D model. The shapes, texture, color and lighting of the object are taken into account. It involves the architecture of the encoder-decoder, a type of neural network that converts input data into a vector used to predict specific information. All the processing takes less than 100 milliseconds, but training of the neural network takes two days on the NVIDIA V100 GPU. On other GPUs the process, according to the company, lasts for weeks.

The system gives the depth perception of autonomous robots, thus increasing their accuracy and safety, as well as improving their orientation on the ground and the ability to manipulate objects. The DIB-R framework is integrated into the Kaolin library developed by NVIDIA PyTorch [27]. It helps to accelerate research in 3D technology and in-depth learning.

3.2 Simultaneous localization and mapping (SLAM)

When controlling the movement of a stand-alone mobile robot or group of robots, they often rely on an accurate assessment of the robot's position in the space or accurate information about the environment. However, in numerous situations, this information may not be available. In this case, to solve the problem of planning trajectories when performing a mission by an autonomous robot, it is necessary to first determine its location and assess the environment, which deals with the issue of simultaneous localization and mapping — SLAM [28, 29]. The SLAM task implies that the robot is placed in an environment about which it has not enough information. Using the only on-board sensors, the robot has to move in this environment from its initial position to the specified one, and simultaneously construct a plan of the area and get an accurate assessment of its own trajectory.

The most popular methods are those that use a camera to map and track robot movement in space. This is due to the compactness and cheapness of the camera. These methods can be divided into two classes: those that use the entire camera image and build a dense terrain map (direct based SLAM), and feature based methods which build a sparse map: key points are extracted from the frame — the most prominent points of the image, and the rest of the information is discarded.

There are two main types of realizations of SLAM [30]: feature-based Oriented FAST and Rotated BRIEF SLAM (ORB) and direct-based Large-Scale Direct SLAM (LSD).

ORB SLAM

One of the most popular implementations of SLAM [31]. This method is based on ORB key point detector. High speed of ORB detector allows the method to work in real time in conditions of limited computing resources. Its work is divided into three main threads.

- Frame Tracking approximately defines the current camera position by searching for a similar frame in the local map and comparing key points with the found frame.
- Local Mapping builds a map near the current camera position and optimizes the map.
- Loop Closing searches for and combines similar frames.

LSD SLAM

The algorithm LSD SLAM includes three main modules: tracking (continuously monitors new images from the camera and detects camera movement), depth map estimation (compares the new frame with the current one and then clarifies or completely replaces the current one) and map optimization (performs map optimization). Fig. 9 [32] shows the main component of the algorithm.

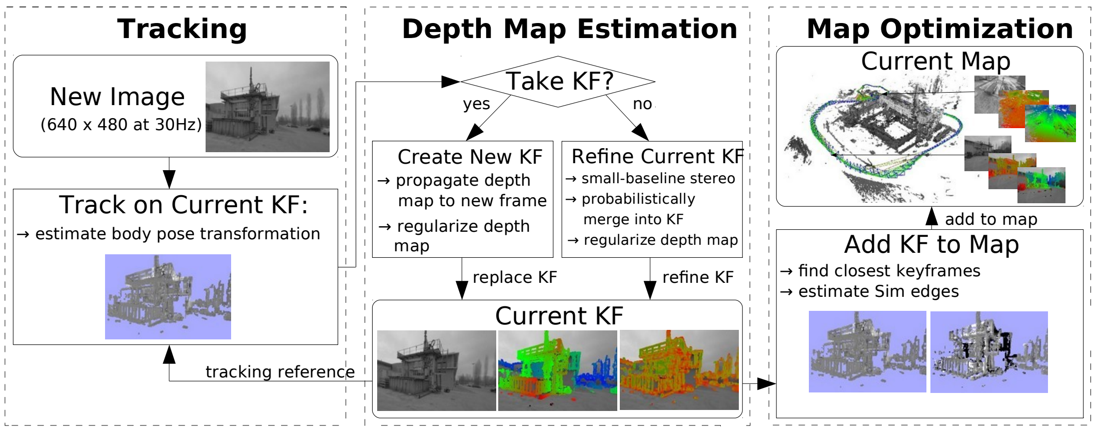


Figure 9. Overview LSD-SLAM algorithm [32]. KF is Key Frame.

SLAM would allow a complete understanding of the scene, but in conditions where it was necessary to quickly create a system for localizing the object, it was decided to use the system with relatively good accuracy and quick action. That's why it was decided to use a neural network connection with an Arduino. However, to further improve the system, it is recommended to use SLAM, since the method will make it possible to increase the accuracy of grasping. In addition to the neural network, SLAM will provide complete information about the 3D world, which help to understand the object boundaries to simplify grasping from anywhere.

4 GRASPING

Grasping objects is a challenging task for industrial robots. It is not always easy for modern robots to control compression and regulate forces for various objects, like humans, even though there are available force-feedback sensors for the grippers. It has to set a separate mode for each new object. This is especially crucial for delicate objects, for instance, vegetables or fragile glassware: squeezing too hard can damage them. Thus, the necessary for smart grasping is very important part for industries.

The article about ways of learning of grasping using visual information was introduced by Ishay Kamon *et al.* in 1996 [33]. Current work solves two subquestions: 1) choosing grasp points; 2) predicting the quality of a grasp. The system computes the coordinates of given objects and its geometry. Then it chooses grasp points, and performs a grasp. For each grasp the method stores location parameters and quality parameters.

One more approach for grasping of objects was proposed by Ashutosh *et al.* in 2008 [34]. In the article, researchers suggested a way to grasp, which evaluates the stability of various grasps, taking into account only estimates of the visible forms of the object, for example, using a depth sensor. Mixing this with kinematic information of hand and the robot's arm, the algorithm allows to calculate the particular locations of the robot's fingers to grasp. The algorithm was tested on various types of controllers (see in Fig.10 [34]).

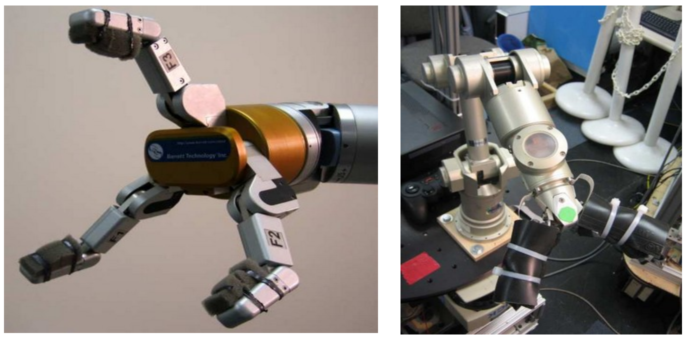


Figure 10. (Left) Barrett 3-fingered hand. (Right) Katana parallel plate gripper [34].

In 2020, Morrison *et al.* introduced a new approach of grasping objects using image depth through deep neural networks [35]. Their neural network predicts pixel capture character which can be utilized in grasp closed loop scripts. This method solves the significant drawbacks of existing techniques, such as, a discrete sample of candidates for capture and a long calculation time. The neural network copes with the grasping task significantly more efficiently than other modern solutions. Moreover, the researchers were able to obtain higher performance, even in cases of a random scatter of objects. The result of capturing objects using their algorithm is about 85% for previously unknown objects and about 90% with already known objects.

In 1998, Bologni presented a solution to one of the main problems of robot grasping [36]: determining the best grasping position and required force values. In this paper, the author presented a method for solving the problem under the condition of the presence of three locations of connection in 2D. Finding the optimum reply of the energy structure to the request of a disturbing outer power helps to choose between several possible solutions. The proposed method on the grounds of simple geometric transformation and requires a fairly small estimated time, which is an important advantage for real-time operation.

However, force feedback is not important for this thesis, as the main purpose is to find the object and then grasp it from any point. Therefore, the force of grasping can be neglected.

5 PROPOSED METHOD

The left part of the scheme (Fig. 11) shows the part with image processing. The first part consists of creating data for training YOLO using a specialized program. Next is the process of training the neural network. In addition, the process of finding an object in the image. The right part of the diagram shows the part with the connection of all parts of the system using Python and Arduino with sensors.

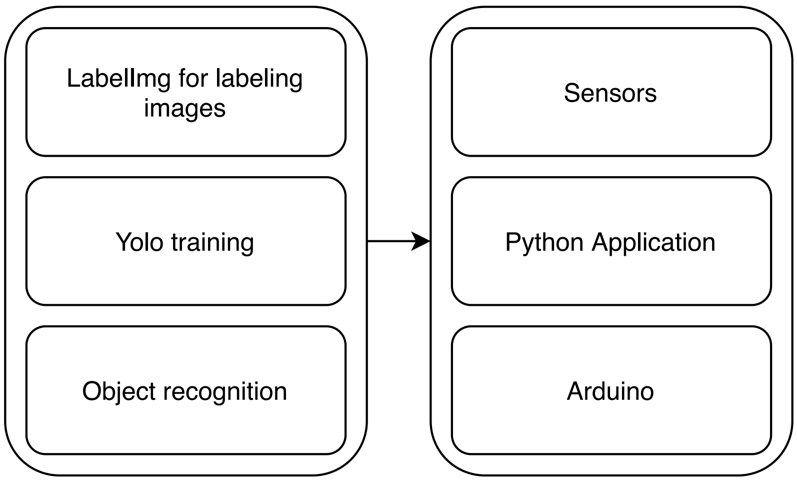


Figure 11. Proposed method schematic design.

5.1 Object recognition

For fast and accurate recognition it was necessary to use an effective system of object recognition in images. As shown in the section 2, the YOLO network has the best ratios of FPS and accuracy for real-time tasks.

The advantages of YOLO are that the network looks at the whole image at once and takes into account the context when detecting and recognizing an object. Moreover, YOLO is about 1000 and 1500 times faster than R-CNN and Fast R-CNN respectively. YOLOv3 (Fig. 12 [2, 22]) is one of the best and highest quality among YOLO architectures at the moment. This neural network uses 106 convolution layers, which allow for easy detection of small objects compared to previous versions. One of the features of YOLOv3 are 3 layers, which are designed to find objects of various sizes.

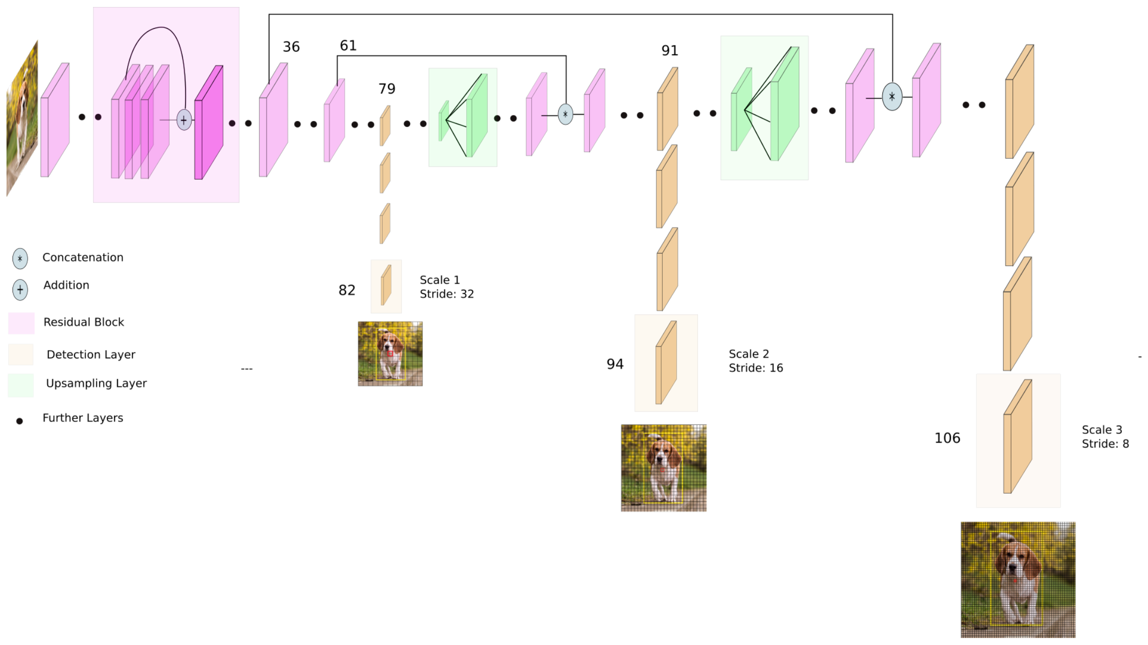


Figure 12. YOLOv3 schematic architecture [2, 22].

5.2 Smart grasping

When developing the system, it was decided to change the initial use of the robot from the laboratory due to the coronavirus pandemic infection. Previously, the system without any sensors could easily determine the location of the end point of the robot arm. This allowed the system to determine the object and calculate the trajectory to the detected object.

Since earlier my bachelor's work was related to Arduino, it was decided to use this particular system as fundamental for design. Moreover, a vast number of sensors were produced for Arduino that could help in creating the right system. The sensors helps to obtain data about the direction of movement using a compass, as well as the distance to the object using a distance sensor, and of course about the angle of inclination relative to given axes.

It was necessary to develop a system that could simulate the action of end-effector of the robot. For this, a complex of Arduino and sensors was developed.

Arduino

The Arduino Mega 2560 (Fig.13 [37]) debug board is built on the ATmega2560 micro-controller. It has 54 digital Input/Output pins (15 of which can be used as Pulse-Width Modulation (PWM) outputs), 16 analog inputs, 4 UART (hardware serial ports), a 16 MHz crystal, a USB connection, a power connector, an ICSP connector, and a reset button. It contains everything that are needed to work with the micro-controller. The Arduino Mega 2560 is compatible with most expansion cards designed for the Arduino Uno.

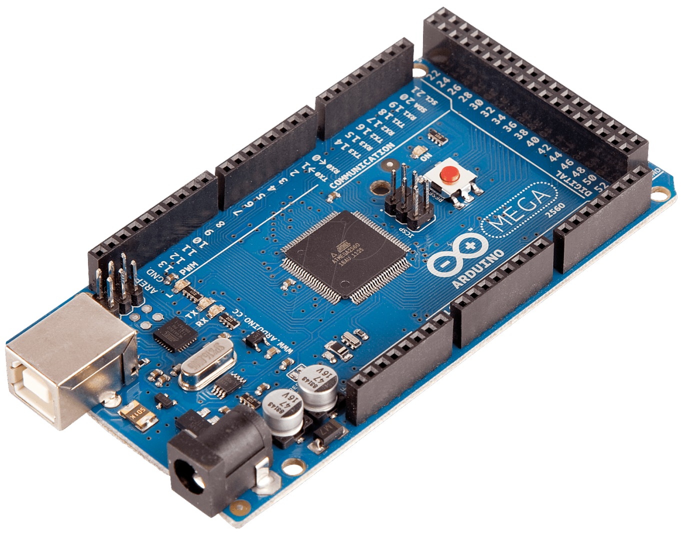


Figure 13. Arduino Mega 2560 [37].

Arduino sensors

HC-SR04. The Arduino distance sensor (Fig. 14 [38]) is a non-contact type instrument and provides high-precision measurement and stability. The principle of operation of the ultrasonic rangefinder is based on the emission of ultrasound and its reflection from objects in front. Based on the time the sound returns, using a simple formula, the distance to the object is calculated. The HC-SR04 rangefinder has good performance at a low price. The range of its measurement range is from 2 to 400 cm. Electromagnetic radiation and solar energy do not significantly affect its operation. To reduce errors and measurement errors, values are averaged (measure several times, remove bursts, then find the average);

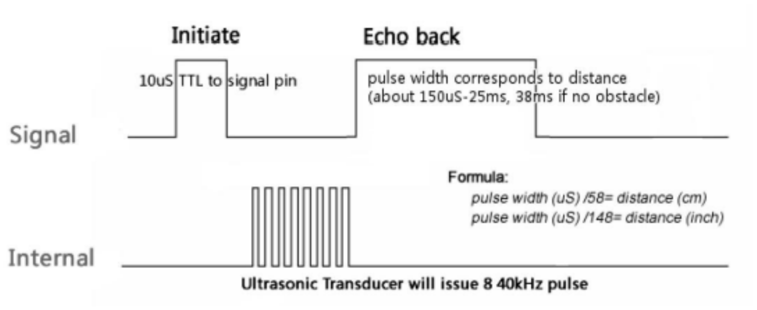
The sequence of actions for obtaining data is as follows (see scheme in Fig.15 [39]):

Figure 14. HC-SR04 [38].

- give a pulse of 10 μs duration to the Trig pin;
- inside the rangefinder, the input pulse is converted into 8 pulses with a frequency of 40 kHz and sent forward through the emitter T;
- reaching the obstacle, the sent impulses are reflected and received by the receiver R, as a result we get the output signal at the Echo pin;
- directly on the controller side we translate the received signal into the distance according to the formula:

$$\frac{pw}{58} = d_{cm} \text{ and } \frac{pw}{148} = d_{inch}$$

where the d_{cm} is distance in centimeters, the d_inch is distance in inches, pw is pulse width in μs.



Initiate
Echo back
10uS TTL to signal pin
pulse width corresponds to distance
(about 150uS-25ms, 38ms if no obstacle)
Signal
Formula:
pulse width (uS) /58= distance (cm)
pulse width (uS) /148= distance (inch)
Internal
Ultrasonic Transducer will issue 8 40kHz pulse

Figure 15. Principle of operation of the distance sensor [39].

Troyka-Accelerometer. The accelerometer from the Troyka-module (Fig.16 [41]) line allows to measure acceleration relative to own X, Y, and Z axes. This property is very useful if assemble a shock force meter, a balancing robot, or a control system using gestures are needed.

The module is based on the LIS331DLH chip from STMicroelectronics [40]. Exactly the same accelerometer is installed on the iPhone 4S. The chip has a built-in temperature sensor. This will allow for accurate sensor operation even in extreme conditions. The module communicates with control electronics, such as Arduino, via the I^2C protocol. This means that two contacts to connect are needed.

A voltage regulator and a special I^2C buffer are installed on the module. Therefore, it can be safely used with a control electronics voltage of 3.3 ... 5V.

The connection of this Troyka-module is different from the standard: it has two three-pin connectors. One connector is used to supply voltage, the other to connect to the I^2C bus. Two 3-wire cables for connection are included.

The accelerometer is needed to obtain data on the angle of inclination relative to a given plane, as well as on obtaining data on acceleration in one direction or another in the XYZ-plane

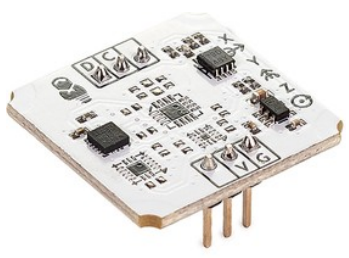


Figure 16. Troyka-Accelerometer [41].

Troyka-Magnetometer/Compass. A triaxial magnetometer (Fig.17 [42]) will help measure magnetic induction. This magnetometer/compass from the line of Troyka-modules allows to determine the angles between the eigen axes of the sensor X, Y, Z and the lines of force of the Earth's magnetic field. This means that it is very easy to find the cardinal points in whatever position it is.

The module is based on the LIS3MDL chip from STMicroelectronics. The chip has a built-in temperature sensor. This will allow for accurate sensor operation even in extreme conditions. The module communicates with control electronics, such as Arduino, via the I^{2}C protocol. This means that two contacts to connect are needed.

Using a compass, it becomes possible to obtain data in which direction the sensor is looking relative to the pole, respectively, this will help in detecting the direction of movement of the system

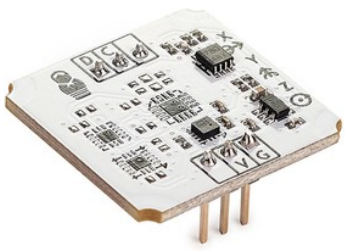


Figure 17. Troyka-Magnetometer/Compass [42].

Algorithm

As the coronavirus pandemic was the reason for the current setup, a developed system should be able to simulate the endpoint of the robot. Its algorithm is as follows:

The 3 step in Alg.1 is shown in Fig.19. After finding the object, the system will center the object on the image, as shown in Figure 4b. After the object is in the middle of the image, the distance necessary to reach the object along line 2 will be shown in the lower left corner (see Fig. 19b, orange rectangle). In the bottom middle, the distance to the object will be shown (see Fig. 19b, turquoise rectangle). Colored lines in the Fig.19a correspond to the following values:

1. Distance to the object;
2. The necessary gap that the system needs to go through to be above the objects;
3. Distances to the object to make a grasp.

In the next step (see Fig. 20), the system moves along the line 2 to the point at which it will be located above the object. The system will turn down in order to check the possibility of

Algorithm 1 Smart grasping

1. The system begins to collect information, passing a path similar to a zig-zag.
2. At certain points, the system moves around its axis in order to understand the surrounding environment.
3. If the specified object is found and it's possible to grasp. Since in this paper Robotic 3-finger gripper is considered as the option, it can be assumed that there is enough space for the gripper to grasp the object from anywhere. In addition, a sufficient place for grasping is assumed from the fact that the boundaries of the object have a sufficient amount of space for grasping. Thereafter, the system centers the object in the middle of the image. The distance to the object is calculated. The system passes the distance to the object. Turns down and informs about the possibility of grasping. If grasping is possible, the system will go down to grasp.

   If the system does not find the object during the intermediate point, then the system goes further along the path until the object is found. If after some time the object is not found, the system will inform about the absence of the object in the area.

grasp, as well as the presence of the desired object (see Fig. 21b). After the previous step, the system will go along the line 3 (see Fig. 19) in order to take an object. The system will notify the user that the object is grasped.

To determine the distance necessary to reach the object along the line ② (see Fig.18), the formula for calculating the sides in a right triangle is used.

$$d_2 = d_1 \cos(\alpha)$$

where d_2 is the horizontal distance (Fig.18, line ②) and d_1 is the measurement from the camera to the object (Fig.18, line ①) and α is an angle between the horizontal distance d_2 and the measurement from the camera to the object d_1. The angle α is obtained using the accelerometer sensor, as it can show information about how much the system has tilted relative to the current axis. In this case, relative to the Y axis, which is line numbered as ② in Fig. 18.

The starting point at the start of the system is taken as the reference point in the world frame. While moving along the plane, the robot uses the world frame as the basis for searching and establishing the presence of an object in the image. After the object is found, it is necessary to take robot frames as a basis. In this frame, as it is shown in Fig. 18, the line ③ is the projections along the Z axis, the line ② along the X axis, and the line ① along the X and Z axes. In Fig.18 it is the robot frame and the origin of the frame is the starting position for the robot motion.

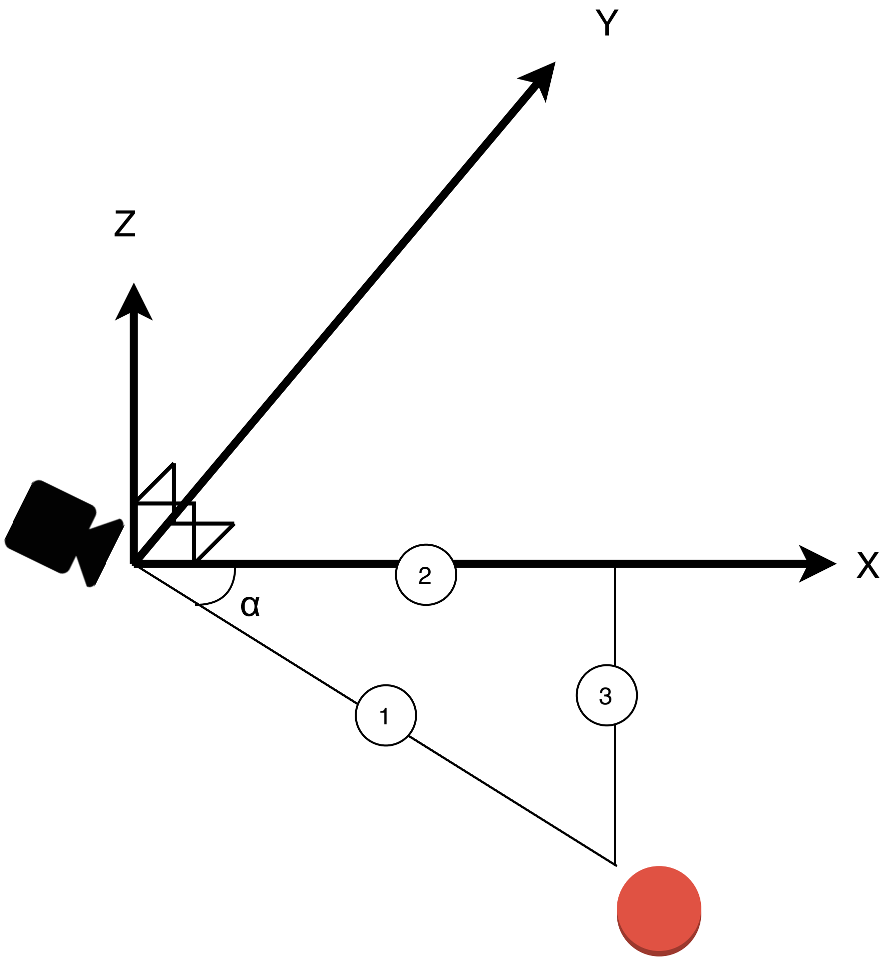
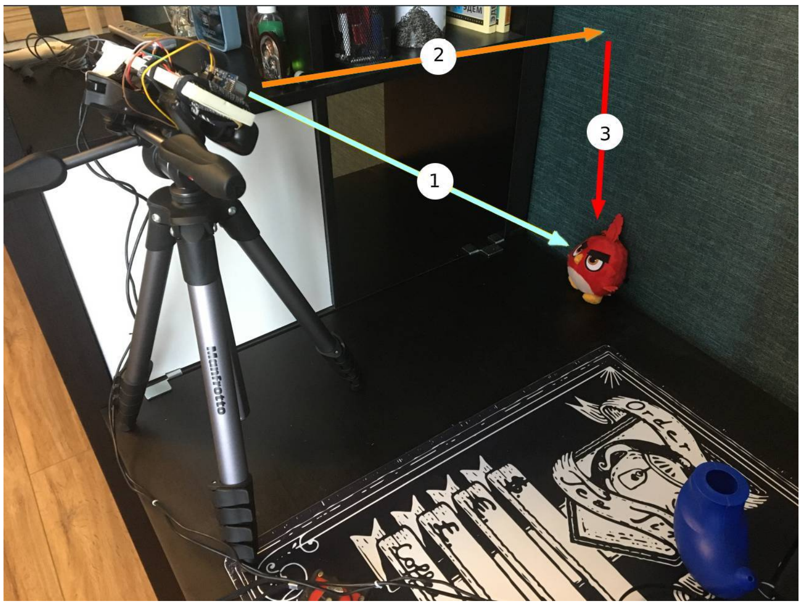
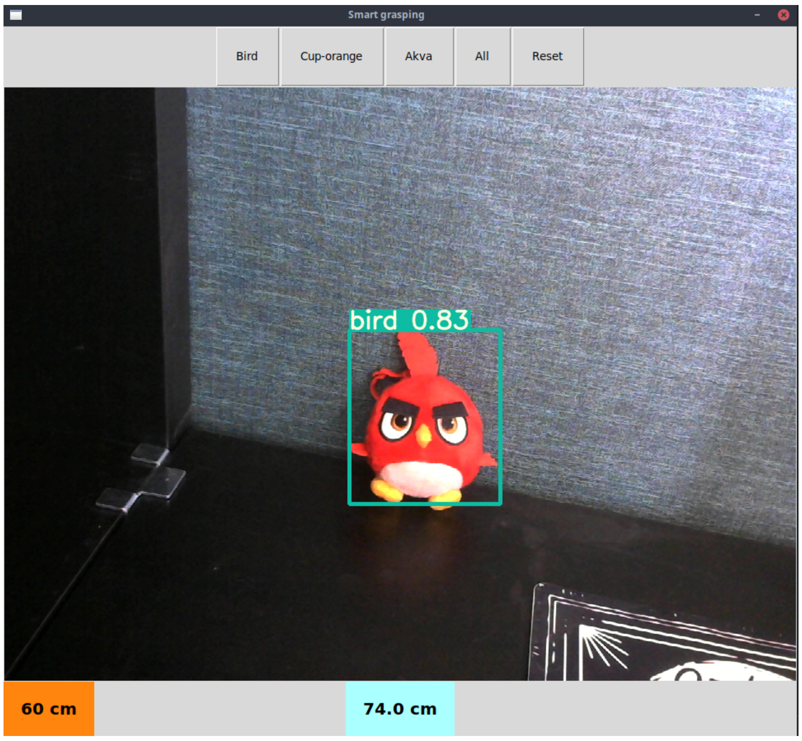


Figure 18. The scheme for calculating the distance to the object. Camera on the left, the target object (red) on the right. .



(a) System



(b) Applicaton for grasping

Figure 19. Step 3. Start.

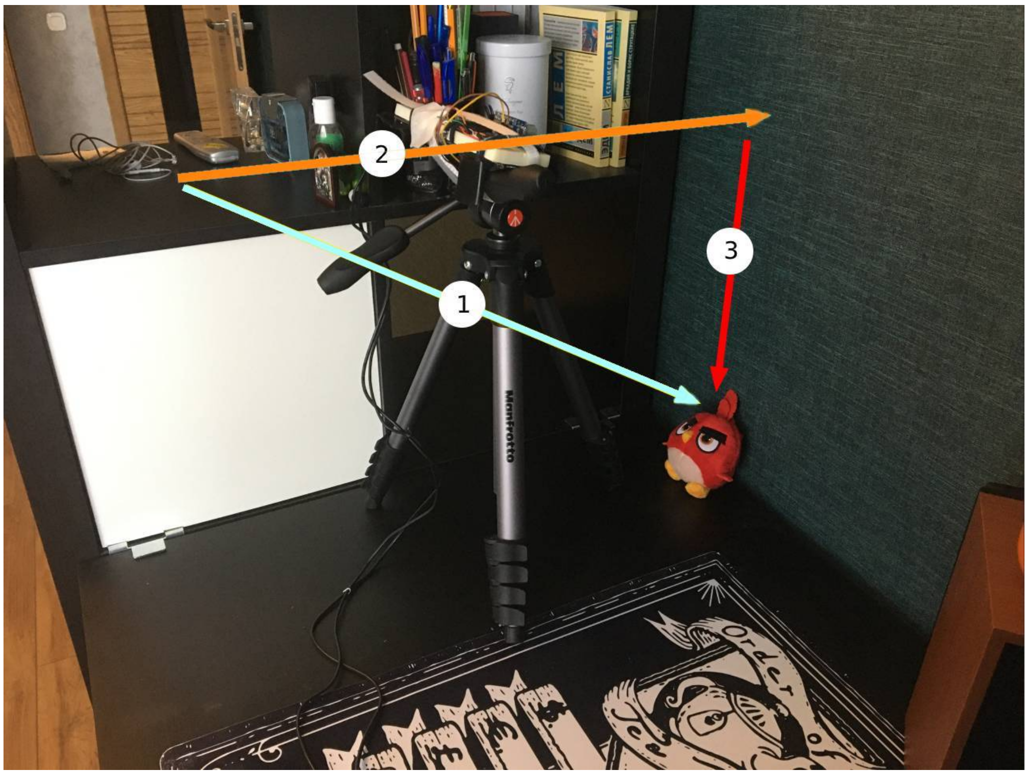
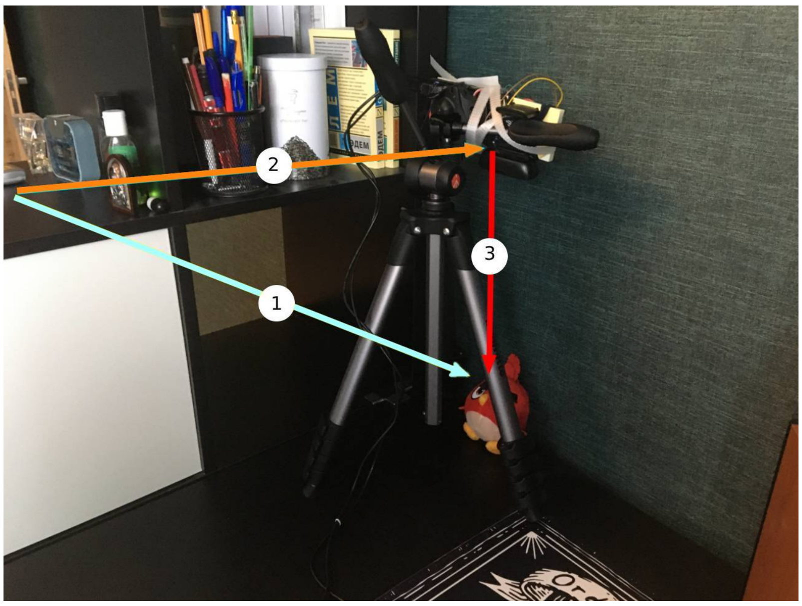
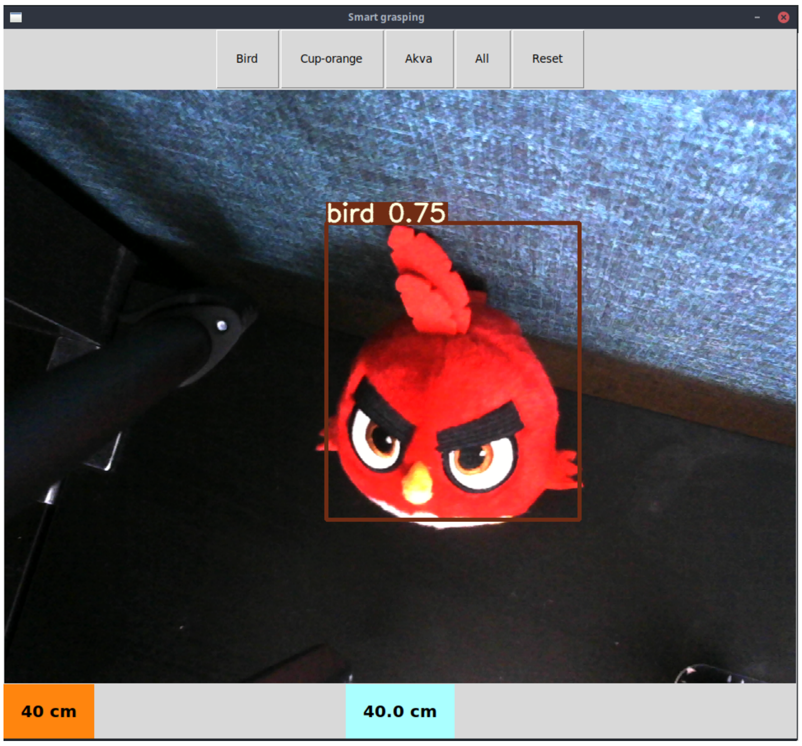


Figure 20. Step 3. Motion on the second line.



(a) System



(b) Applicaton for grasping

Figure 21. Step 3. Finish.

6 EXPERIMENTS

The main goal of the experiments is to show the accuracy of the system, as well as the success of the identification of objects. Two types of experiments were performed: a system for recognizing objects and the accuracy of measuring the distance to an object that the system must pass to be above the object.

6.1 Data collection

In order to prepare data for a neural network, an algorithm was initially proposed to automatically create various options from a video with a static background. However, this method has shown low efficiency for the object recognition problem. Therefore, it was decided to create a dataset of about 400 photographs that were manually labeled. One object accounted for about 100 images. It took about 2 hours to create the entire dataset. To create a specialized dataset, the LabelImg program was used (see Fig.22, [43])

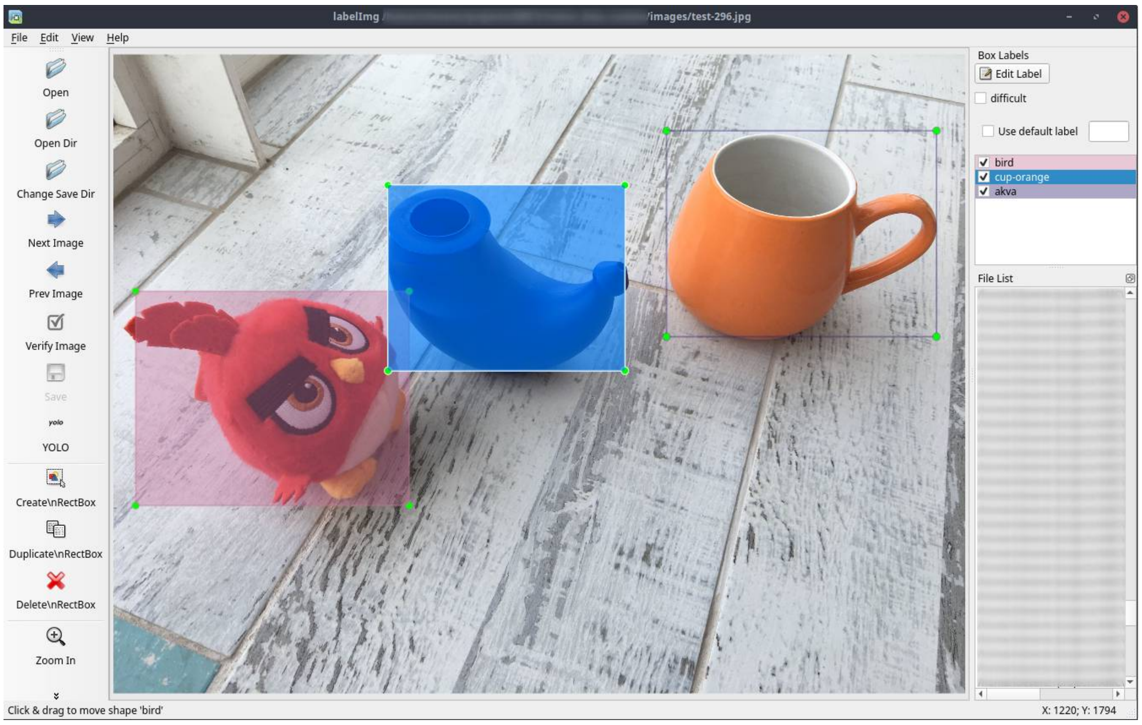


Figure 22. LabelImg [43].

Adding new objects to the system is a very simple step. This requires the use of about

100 pictures with different conditions, such as various viewpoints and objects, as shown in Fig.23. Using the LabelImg program, new items are added to the data folder for YOLO. Then there is training on existing weights.

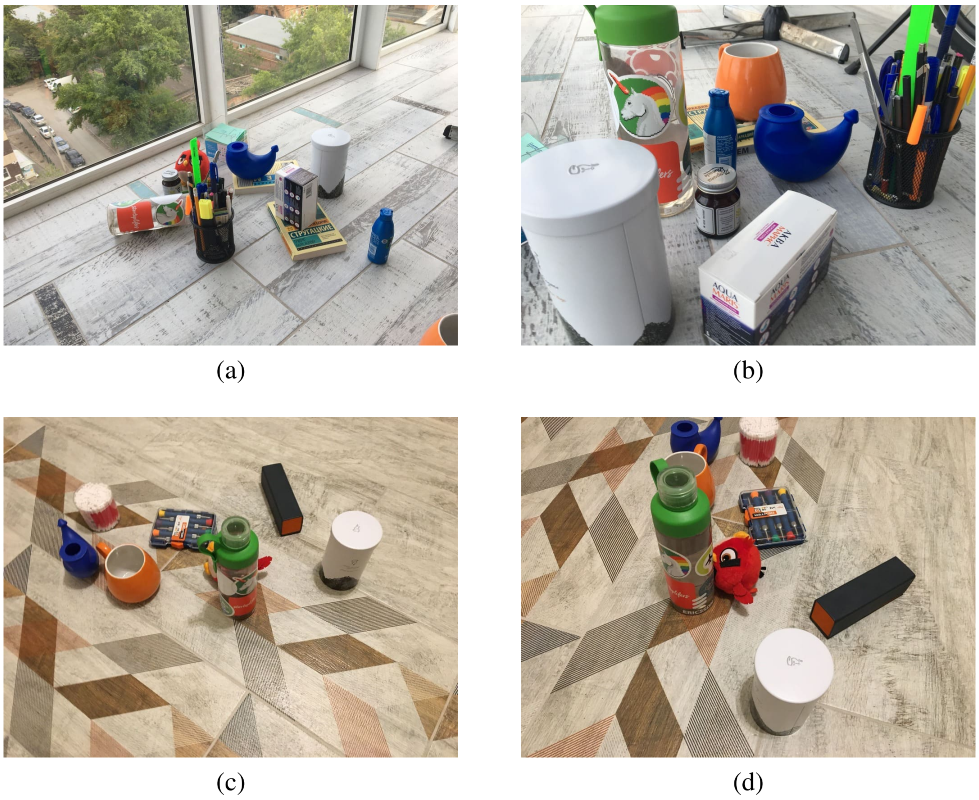


Figure 23. Examples of different conditions.

6.2 Experimental setup

Components of the system

Ultrasonic rangefinder HC SR04 has such technical parameters:

- Supply voltage: 5V;
- The operating parameter of the eye force is 15 mA;
- Current strength in passive state < 2 mA;
- The viewing angle is 15°;
- Sensor resolution: 0.3 cm, main parameter;
- Measuring angle: 30°;
- Pulse width: 10-6 μs.

Troyka-Accelerometer has following technical parameters:

- Sensitivity: 9.8×10-3 m/s^2, main parameter;
- Range of measurement: ±2/±4/±8 g;
- Supply voltage: 3.3-5 V;
- Current Consumption: less than 10 mA.

Troyka-Magnetometer/Compass has following technical parameters:

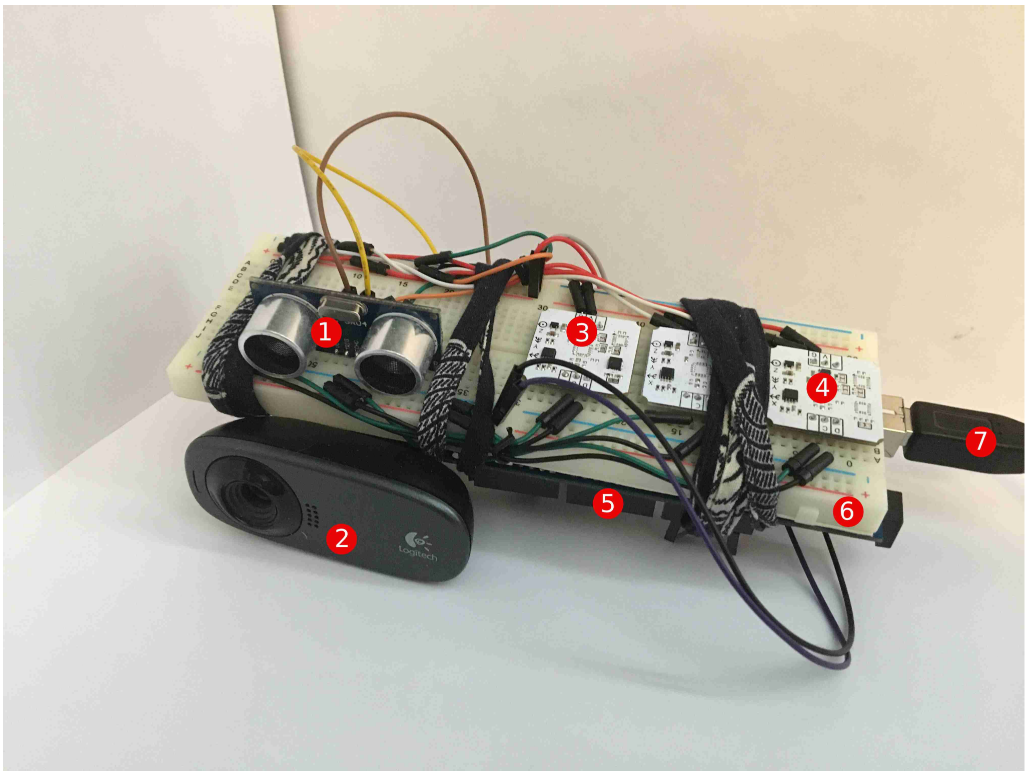
- Sensitivity: 1.46×10-4 Gs, main parameter;
- Range of measurement: ±4/±8/±12/±16 Gs;
- Supply voltage: 3.3-5 V;
- Current Consumption: less than 10 mA.

Web-camera Logitech HD Webcam C310 has following technical parameters:

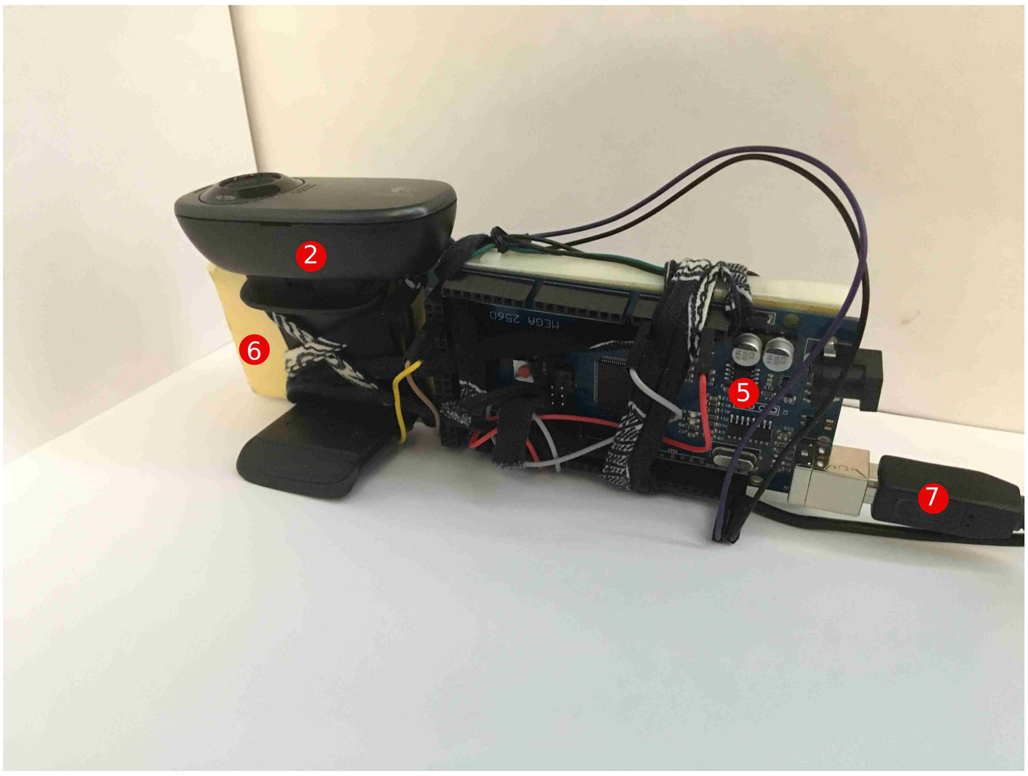
- Video resolution: 1280x720px;
- Frame rate: 30 frames per second.

Fig.24 shows the developed complex to simulate the endpoint of the robot using Arduino and necessary sensors. The table below shows the correspondence of numbers and components of the developed system.

1. Ultrasonic rangefinder HC SR04
2. Logitech HD Webcam C310
3. Troyka-Magnetometer/Compass
4. Troyka-Accelerometer
5. Arduino Mega 2560
6. Breadboard
7. USB cable



(a) View from above



(b) View from below

Figure 24. The proposed system

The YOLOv3 network was trained on new objects using CVPR GPU powered computers (see results in Fig.25):

- GeForce GTX 1080 12 GB
- 1 step - 10 epochs; image size - 256x256; batch size - 12;
- 1 step - 20 epochs; image size - 512x512; batch size - 12;
- All other YOLO parameters were standard from YOLO repository [44].

There is an explanation of YOLO result table in Fig. 25

- **Precision** measures how accurate is your predictions;

  $$P = \frac{TP}{TP + FP}$$

  where P is a precision, TP is true positive, FP is false positive.

- **Recall** measures how good you find all the positives.

  $$R = \frac{TP}{TP + FN}$$

  where R is a recall, TP is true positive, FN is false negative

- **mAP** is mean Average Precision;
- **F1** score is the harmonic mean of precision and recall

  $$F1 = 2 \cdot \frac{P \cdot R}{P + R}$$

  where F1 is F1 score, P is a precision, R is a recall.

- **GIoU** is Generalized Intersection over Union.

The results from Fig. 25 show rather high accuracy, which in turn gives us the right to conclude that the neural network has trained correctly. Moreover, the graph helps to adjust the training for new data, if necessary.

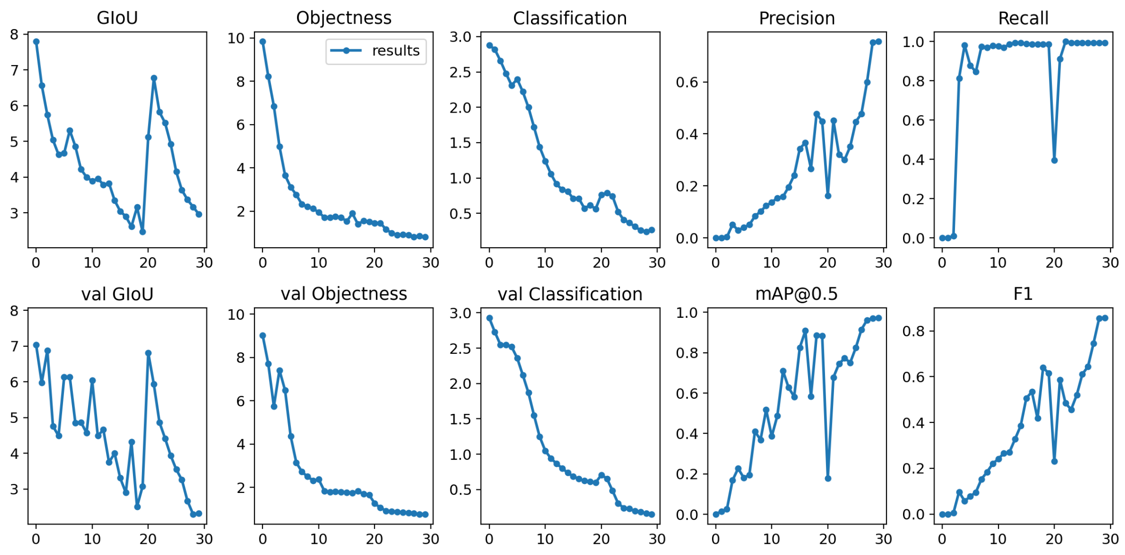


Figure 25. Training results of YOLO. X-axis is the number of epochs.

6.3 Tests

In order to check the quality of determining objects during the execution of the system, 10 attempts for each object were made with various conditions (top view, bottom view, only a part of the object is visible, etc.). See these attempts for the red toy in Fig 26. As can be seen from Table 1, on average, an object is recognized in 9 cases out of 10, which is a good indicator of the accuracy of training.

Table 1. Checking the quality of the object recognition.

№	Object	Not found	Found
1	red toy	1	9
2	orange cup	2	8
3	blue jug	1	9

An important parameter that could show the quality of the system is the measurement of distance. Five attempts were made in each case, where the distance was 10 and 35 cm for all three objects. The table 2 shows that the averaged measurement of the distance that was necessary to go to the object (see Fig.21a, the second line) is not always accurately determined. This is due to the distance sensor, since it is not a very accurate tool.

The distance is measured by HC-SR04 sensor between the camera and the object, which has own accuracy. Therefore, the accuracy of these measuring instruments is not considered in these tests.When moving the camera, the issue of measurement accuracy is not considered, since at this time the system considers only the issue of the presence of the desired object in the image.

Table 2. Checking the quality of the measured distance.

№	True distance	Object	Averaged Measured distance	Number of attempts
1	10 cm	red toy	11.5 cm	5
2	35 cm		33 cm	
3	10 cm	orange cup	10 cm	
4	35 cm		35 cm	
5	10 cm	blue jug	9 cm	
6	35 cm		34.5 cm	

At the moment, this system is only an imitation of a real robot and the necessary tools for a deeper and fully intelligent grasping. In the future, using a real robot, it will be possible to use kinematics, as well as using the Robotiq 3-finger gripper it will be easy to find out how much space is nearby to perform smart grasping.

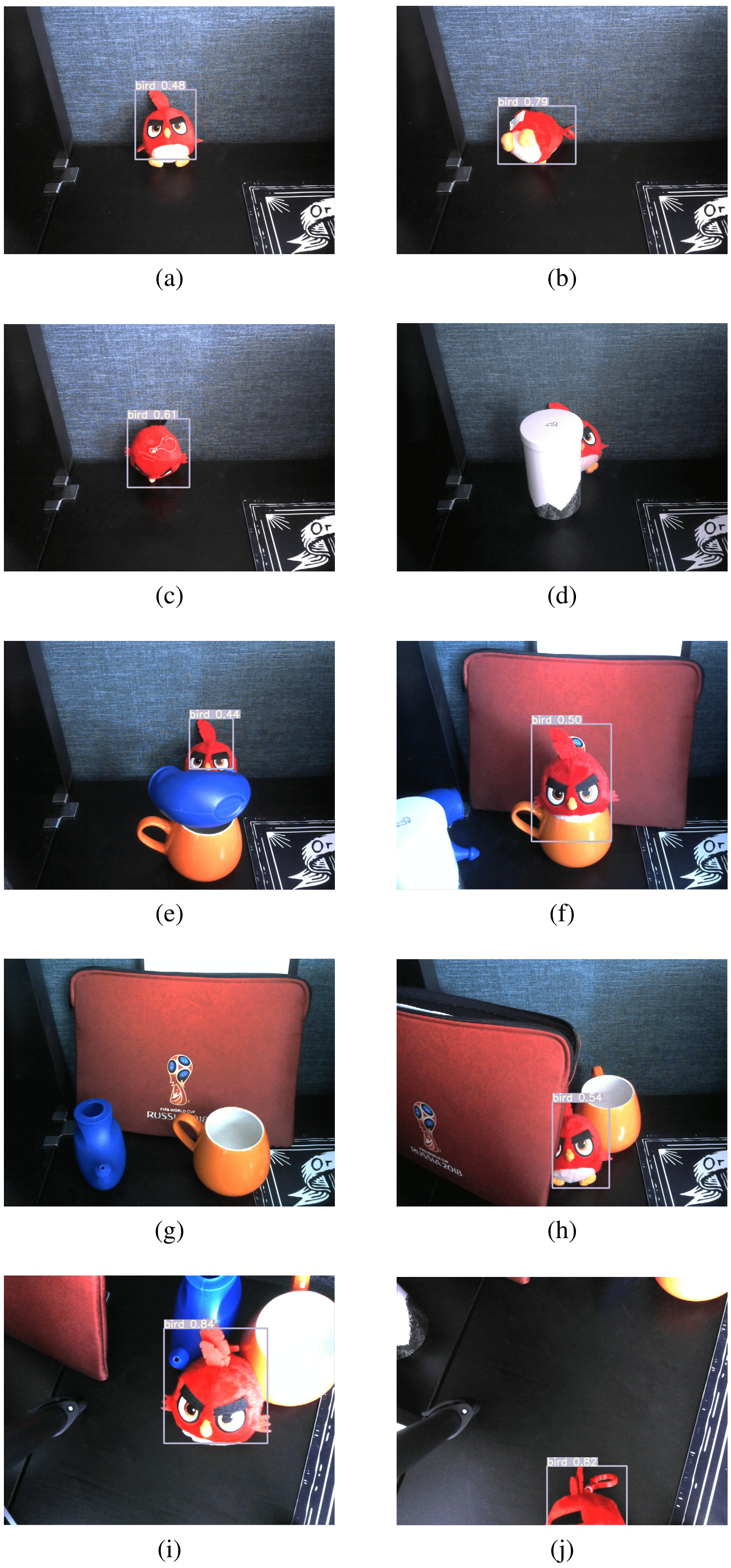


Figure 26. Examples of different conditions for testing.

7 DISCUSSION

During the review many useful articles were seen, which explain the minimum necessary knowledge about current research questions. Based on the reviewed articles it was easy to build the recognition system. Using the high performance existing implementations such as YOLOv3 or SLAM, which speeded up the process of building. Moreover, a library was built as part of the study to facilitate the proposed method.

The coronavirus pandemic was the reason for changing the initial design of the system, which was tied to a robot in the CVPR laboratory. As a result, a system consisting of Arduino and its sensors appeared, which made it possible to identify objects in real time and with good accuracy, as well as to localize them for further grasping. Although the system requires further development, as well as the possibility of replacing some of the sensors to obtain more accurate results, the overall system performance targets for the thesis. Moreover, the created system simplifies further integration into the system with robots to complete the connection of the robotic system and automatically find objects with the gripper.

At the moment, 3 new objects have been added, and a basis has also been created for easily adding new objects to the existing system without compromising the accuracy of determining previous ones. For each of the 3 objects, 10 tests were carried out, which showed a good result of the trained system with various conditions by type, the object is not or partially visible, the object is identified among similar ones in color, etc.

8 CONCLUSION

After successful literature review the chance of getting a high quality results have increased. All necessary information about objects recognition, understanding 3D scene, and smart grasping was achieved. Despite the fact that it was not possible to connect the system with the original robot, the system for determining and localizing the object was created with good performance.

Despite the fact that the system performs all the necessary functions and does them with good accuracy, not all the goals that were set were originally fulfilled. The transformation from 2D to 3D was not performed, since this was not necessary due to the simplicity of the Arduino system. At the same time, the accuracy of executing and adding new objects, as well as performing real-time searches, works both efficiently and with high FPS. As a result of the thesis, an Arduino system with sensors was created from scratch, using Compute Unified Device Architecture (CUDA) and parallelizing the processes of obtaining data from Arduino and outputting frames from the camera, as well as performing calculations related to finding the distance.

One of the important factors for more successful completion of the task is to improve the accuracy of the sensors, and it is possible to use more complex systems to fully understand the world around the system. In the future, it will be necessary to integrate the current system with ROS, as this will replace Arduino and create a more efficient system for grasping the objects.

REFERENCES

[1] Xin Feng, Youni Jiang, Xuejiao Yang, Ming Du, and Xin Li. Computer vision algorithms and hardware implementations: A survey. *Integration*, 69:309 – 320, 2019.

[2] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[3] The robot operating system. `https://www.ros.org/about-ros/`, February 2020.

[4] LUT CVPR Laboratory. `https://www.it.lut.fi/cvprl/`, January 2020.

[5] Industrial Robots-MELFA | MITSUBISHI ELECTRIC FA. `https://www.mitsubishielectric.com/fa/products/rbt/robot/index.html`, February 2020.

[6] 3-Finger Adaptive Robot Gripper. `https://robotiq.com/products/3-finger-adaptive-robot-gripper`, May 2020.

[7] Olga Russakovsky, Jia Deng, and Hao Su. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[8] Yann LeCun and Yoshua Bengio. Object Recognition with Gradient-Based Learning. In *Shape, Contour and Grouping in Computer Vision*, Lecture Notes in Computer Science, pages 319–345. Springer, 1999.

[9] Genevieve Sapijaszko and Wasfy B. Mikhael. An Overview of Recent Convolutional Neural Network Algorithms for Image Recognition. In *IEEE International Midwest Symposium on Circuits and Systems*, pages 743–746, 2018.

[10] Suresh Arunachalam T, Shahana R, Kavitha T. Advanced Convolutional Neural Network Architecture : A Detailed Review. *International Journal of Engineering Trends and Technology*, pages 183–187, 2019.

[11] Convolutional Neural Network Tutorial: From Basic to Advanced. `https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-tutorial-basic-advanced/`, January 2020.

[12] Yu Vankov, Aleksey Rumyantsev, Shamil Ziganshin, Tatyana Politova, Rinat Minyazev, and Ayrat Zagretdinov. Assessment of the condition of pipelines using convolutional neural networks. *Energies*, 13:618–630, 2020.

[13] S Anitha Elavarasi, J Jayanthi, and N Basker. Trajectory object detection using deep learning algorithms. *International Journal of Recent Technology and Engineering*, 8:7895–7898, 2019.

[14] Ross Girshick and Jeff Donahue. Rich feature hierarchies for accurate object detection and semantic segmentation. *Computing Research Repository*, October 2013. arXiv: 1311.2524.

[15] Jasper R. R. Uijlings and Koen E. A. van de Sande. Selective Search for Object Recognition. *International Journal of Computer Vision*, 104(2):154–171, September 2013.

[16] Yeunghak Lee, Israfil Ansari, and Jaechang Shim. Rear-approaching vehicle detection using frame similarity base on faster r-cnn. *International Journal of Engineering and Technology*, 7(4.44), 2018.

[17] B Vinoth Kumar, S Abirami, R J Bharathi Lakshmi, R Lohitha, and R B Udhaya. Detection and content retrieval of object in an image using YOLO. *IOP Conference Series: Materials Science and Engineering*, 590:12–32, 2019.

[18] Ross Girshick. Fast R-CNN. *IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[19] Shaoqing Ren and Ross Girshick. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2016.

[20] Wei Liu and Dragomir Anguelov. SSD: Single Shot MultiBox Detector. In *Computer Vision*, volume 9905, pages 21–37. Springer International Publishing, 2016. arXiv: 1512.02325.

[21] Single Shot Multibox Detection (SSD) — Dive into Deep Learning. `https://d2l.ai/chapter_computer-vision/ssd.html`, January 2020.

[22] Z. Ding, R. Huang, and B. Hu. Robust indoor slam based on pedestrian recognition by using rgb-d camera. In *Chinese Automation Congress*, pages 292–297, 2019.

[23] Mohammad Mahdi Derakhshani and Saeed Masoudnia. Assisted Excitation of Activations: A Learning Technique to Improve Object Detectors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9193–9202, 2019.

[24] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Make3D: Learning 3D Scene Structure from a Single Still Image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):824–840, May 2009.

[25] A. Soloshenko, Y. Orlova, Vladimir Rozaliev, and A.V. Zaboleeva-Zotova. *Automated Mind Map Generation from News Texts Based on Link Grammar*, volume 535, pages 637–654. 01 2015.

[26] Wenzheng Chen, Jun Gao, Ling Huan, Edward J. Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer. *Conference on Neural Information Processing Systems*, pages 9605–9616, 2019.

[27] Krishna Murthy Jatavallabhula and Edward Smith. Kaolin: A PyTorch Library for Accelerating 3D Deep Learning Research. *Computing Research Repository*, November 2019. arXiv: 1911.05063.

[28] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part I. *IEEE Robotics Automation Magazine*, 13(2):99–110, June 2006.

[29] Juan-Antonio Fernández-Madrigal and José Luis Blanco Claraco. *Simultaneous Localization and Mapping for Mobile Robots: Introduction and Methods*. IGI Global, January 2012.

[30] Søren Riisgaard and Morten Rufus Blas. Slam for dummies: A tutorial approach to simultaneous localization and mapping. Technical report, 2005.

[31] Raul Mur-Artal, J. Montiel, and Juan Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31:1147 – 1163, 10 2015.

[32] Jakob Engel and Thomas Schöps. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *European Conference on Computer Vision*, Lecture Notes in Computer Science, pages 834–849. Springer, 2014.

[33] Ishay Kamon, Tamar Flash, and Shimon Edelman. Learning to Grasp Using Visual Information. In *IEEE International Conference on Robotics and Automation*, pages 2470–2476, 1994.

[34] Ashutosh Saxena, Lawson L S Wong, and Andrew Y Ng. Learning Grasp Strategies with Partial Shape Information. In *AAAI Conference on Artificial Intelligence*, pages 1491–1494, 2008.

[35] Douglas Morrison, Peter Corke, and Jürgen Leitner. Learning robust, real-time, reactive robotic grasping. *The International Journal of Robotics Research*, 39(2-3):183–201, 2020.

[36] L. Bologni. Robotic grasping: How to determine contact positions. *IFAC Proceedings Volumes*, 21(16):395 – 400, 1988.

[37] Arduino Mega 2560 Rev3. `https://store.arduino.cc/arduino-mega-2560-rev3`, June 2020.

[38] HC-SR04 Ultrasonic Distance Rangefinder/Obstacle Detection Module. `https://www.amazon.co.uk/HC-SR04-Ultrasonic-Distance-Rangefinder-Detection/dp/B0066X9V5K`, June 2020.

[39] HC-SR04 Ultrasonic Distance Rangefinder/Obstacle Detection Module.

[40] LIS331DLH chip from STMicroelectronics. `https://www.st.com/en/mems-and-sensors/lis331dlh.html`, May 2020.

[41] Accelerometer (Troyka-module). `https://amperka.ru/product/troyka-accelerometer`, June 2020.

[42] Magnetometer/compass (Troyka module). `https://amperka.ru/product/troyka-magnetometer-compass`, June 2020.

[43] Tzutalin. LabelImg. `https://github.com/tzutalin/labelImg`, June 2020.

[44] YOLOv3 Github Repository. `https://github.com/ultralytics/yolov3`, May 2020.