Intel® RealsensE ™
Distance & Object Detection

lidar-camera-l515

Autonomous robots - Professor Boaz Ben Moshe

Ariel University

Written by Saar Barel, Almog Reuveny, Ishay Levy.

Content

- Distance & Object Detection
  - Introduce RealSense l515 Camera
  - What is a Depth camera?
  - How can this data be used?
  - How we detect the distance?
  - How we detect the objects?
- Distance detection with Depth Camera - Pycharm
  - Code - Pycharm
  - Project result - Screenshots
  - Project result - YouTube Video
- Identify and Measure precisely Objects distance - Pycharm
  - Code - Pycharm
  - Project result - Screenshots
  - Project result - YouTube Video

RealSense

lidar-camera-l515

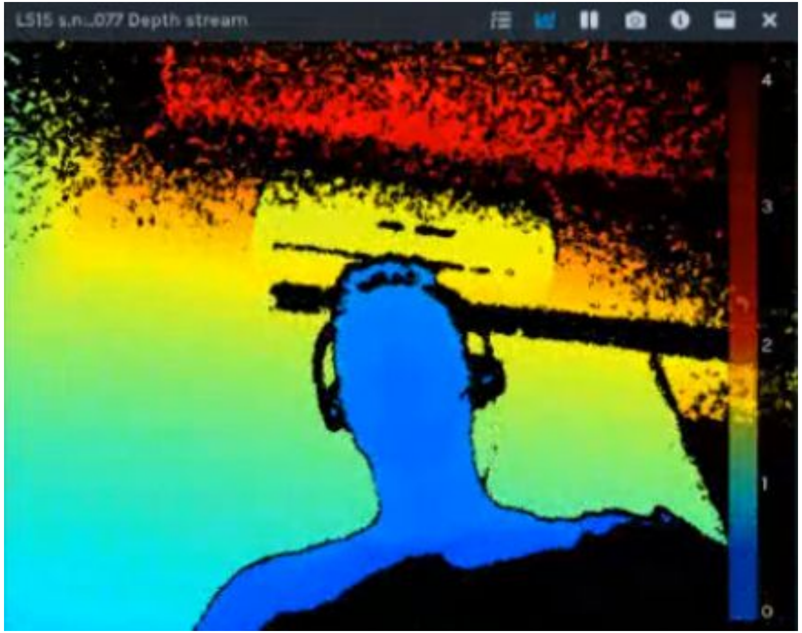
- Depth technology: LiDAR
- Depth frame rate: 30 fps
- Depth output resolution:Up to 1024 × 768
- RGB frame resolution: 1920 × 1080
- RGB frame rate: 30 fps

This is not an operation that can be done with a simple webcam because it does not provide depth information, you need a camera with an infrared sensor or other dedicated technology for distance measurement.

WHAT IS DEPTH?

Color Image from a Color Camera:
Each PIXEL has a Color (Red-Green-Blue) Value

Depth Image from a Depth Camera:
Each PIXEL has a Depth (distance from camera) Value



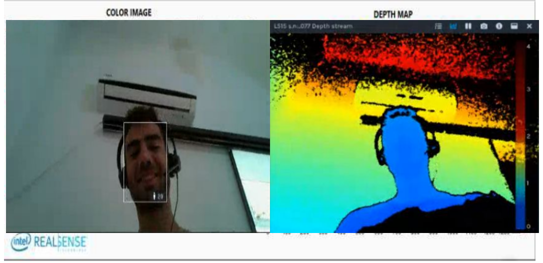
The Depth Map converts the distance information between points on an object's surface and the camera

HOW CAN THIS DATA BE USED?

Sense and Capture

Visualize & Process

Integrate into Applications & Products



Raw Depth Map

SDK Tools for prototyping

Samples Codes & APIs for applications

Design & Validation

Intelligence from depth information

Distance Detection
Object Detection
Deep Learning (R-CNN model)

Enhanced Depth perception makes robots/drones smarter devices and applications across multiple

How we detect the distance?

Depth frame - contains pixels: Each pixel has an array that consists of the distance from the pixel to the object that the camera is aimed at.

The distance that appears on the screen is the calculated distance from Depth frame to point / mouse / object

There are parts that the camera does not pick up and is wrong because it needs to be assigned filtering code to the code. In addition, the camera's prediction is damaged when there is a reflection of light on it.

How we detect the objects?

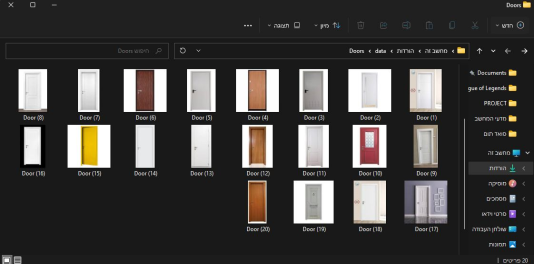
We are using R-CNN training model:

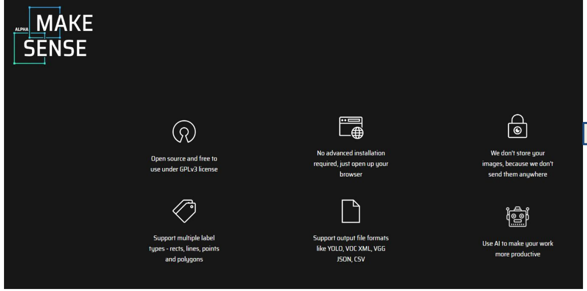
1. **Picture the object several times, in several positions, in different shapes and sizes**

2. **We used the MakeSense.AI website: the website allows you to take the photos we took and draw the shape of the object under a certain label (polygons,rects and more)**

3. **We used a trained model and added our object to it - the model's training was done through an open source in google colab notebook**

4. **We added the trained model to our code - each identified object is painted with a different color, with its own label, and with its own shape. In addition, the distance from the object is displayed on the screen**





Open source and free to use under GPLv3 license

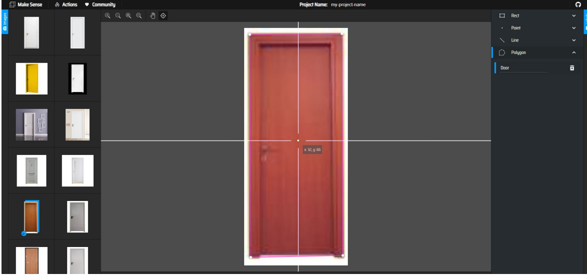
No advanced installation required, just open up your browser

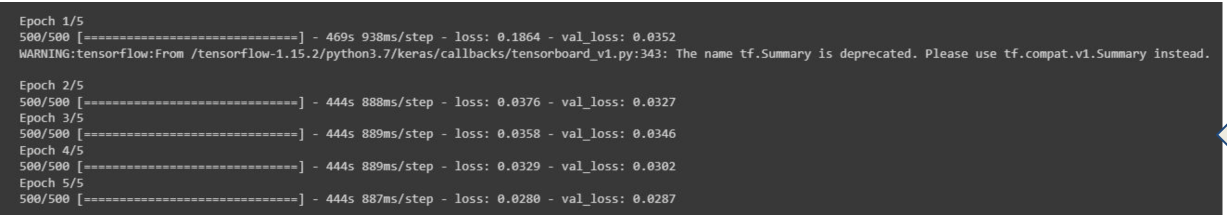
We don't store your images, because we don't send them anywhere

Support multiple label types - rects, lines, points and polygons

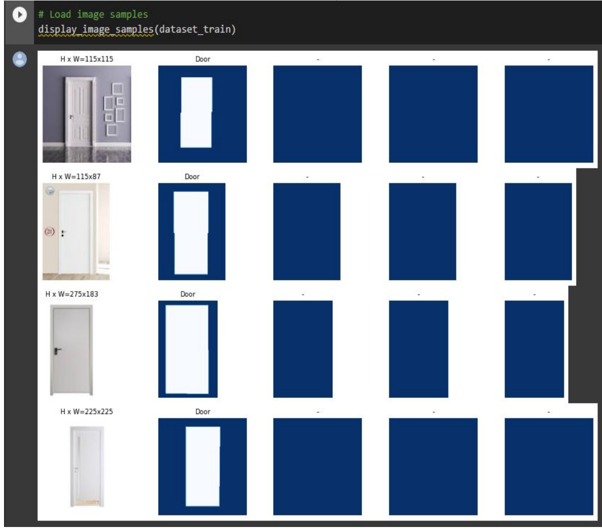
Support output file formats like YOLO, VOC XML, VGG JSON, CSV

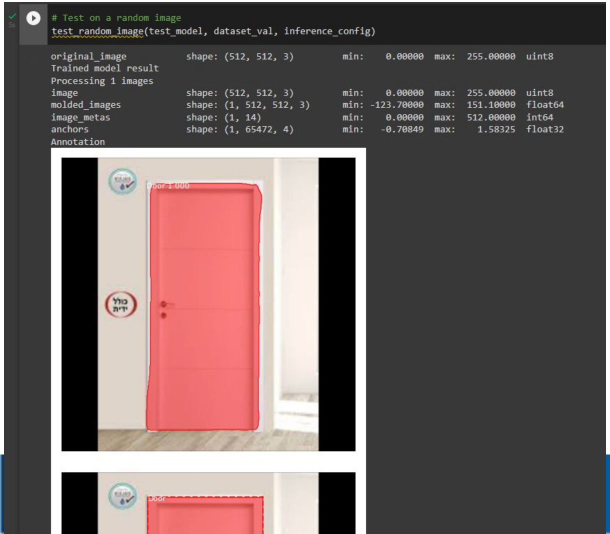
Use AI to make your work more productive

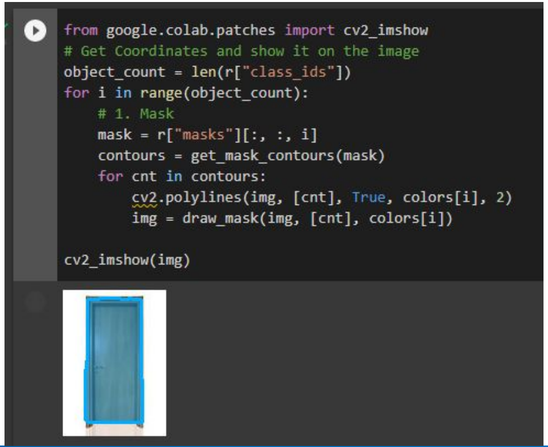



```
Epoch 1/5
500/500 [==============================] - 469s 938ms/step - loss: 0.1864 - val_loss: 0.0352
WARNING:tensorflow:From /tensorflow-1.15.2/python3.7/keras/callbacks/tensorboard_v1.py:343: The name tf.Summary is deprecated. Please use tf.compat.v1.Summary instead.

Epoch 2/5
500/500 [==============================] - 444s 888ms/step - loss: 0.0376 - val_loss: 0.0327
Epoch 3/5
500/500 [==============================] - 444s 889ms/step - loss: 0.0358 - val_loss: 0.0346
Epoch 4/5
500/500 [==============================] - 444s 889ms/step - loss: 0.0329 - val_loss: 0.0302
Epoch 5/5
500/500 [==============================] - 444s 887ms/step - loss: 0.0280 - val_loss: 0.0287
```


```
# Load image samples
display_image_samples(dataset_train)
```


```
# Test on a random image
test_random_image(test_model, dataset_val, inference_config)
```
```
original_image           shape: (512, 512, 3)        min:    0.00000  max:  255.00000  uint8
Trained model result
Processing 1 images
image                    shape: (512, 512, 3)        min:    0.00000  max:  255.00000  uint8
molded_images            shape: (1, 512, 512, 3)     min: -123.70000  max:  151.10000  float64
image_metas              shape: (1, 14)              min:    0.00000  max:  512.00000  int64
anchors                  shape: (1, 65472, 4)        min:   -0.70849  max:    1.58325  float32
Annotation
```


```
from google.colab.patches import cv2_imshow
# Get Coordinates and show it on the image
object_count = len(r["class_ids"])
for i in range(object_count):
    # 1. Mask
    mask = r["masks"][:, :, i]
    contours = get_mask_contours(mask)
    for cnt in contours:
        cv2.polylines(img, [cnt], True, colors[i], 2)
        img = draw_mask(img, [cnt], colors[i])

cv2_imshow(img)
```

Market Focus

Mobile Phones

Robotics

Broad Market

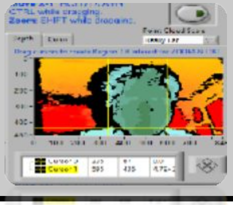
Virtual Reality

Drones

Autonomous Driving

Intel® RealSense™ SDK 2.0 – Tools

Intel® RealSense™ Viewer



Capture, View and Manipulate Depth Streams

Playback/Record

3D PointCloud

Presets, Filtering, Optical Power

Depth Quality Test tool for Intel® RealSense™ camera

Test Setup with camera position and target

Control settings– resolution, gain, exposure, distance to target, error etc.

Calculate z-accuracy, standard deviation

Debug Tools

Data Collection

FW logger

Terminal access

Camera Enumeration

https://github.com/IntelRealSense

* Other names and brands may be claimed as the property of others

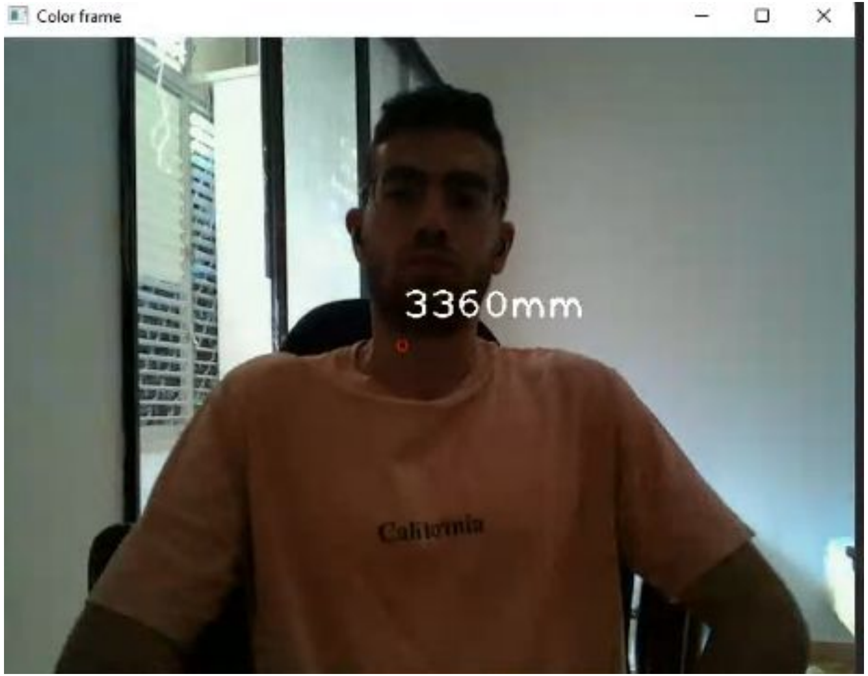
In this presentation we will present 2 parts of codes:

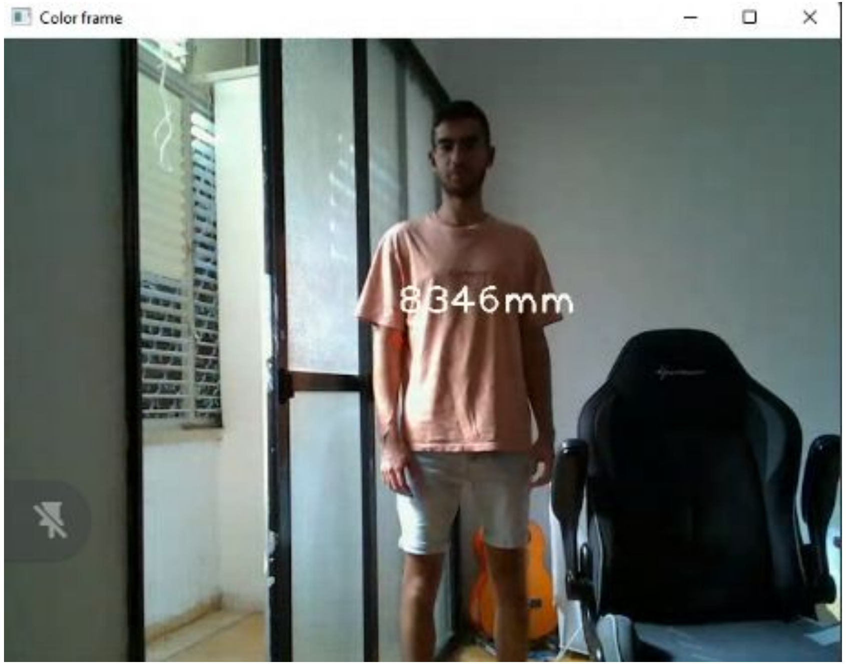
- **Distance detection with Depth Camera - Pycharm**
  - Get the frames from the depth camera.
  - Distance detection on mouseover.
  - Distance detection on Fixed point.

- **Identify and Measure precisely Objects distance - Pycharm**
  - Get the frames from the depth camera.
  - Detect object with AI.
  - using openCV to Identify Certain objects.
  - Distance detection from objects.

Distance detection with Depth Camera - Pycharm

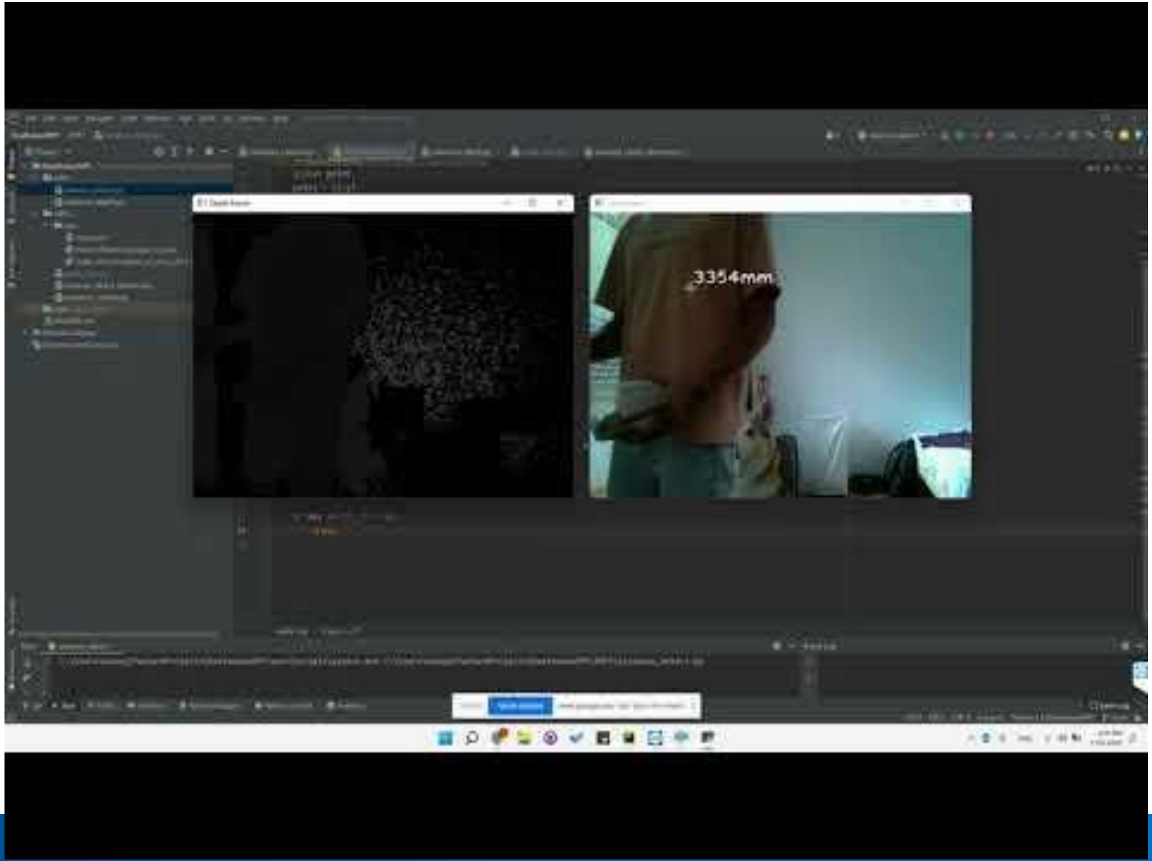
Distance detection on Fixed point.



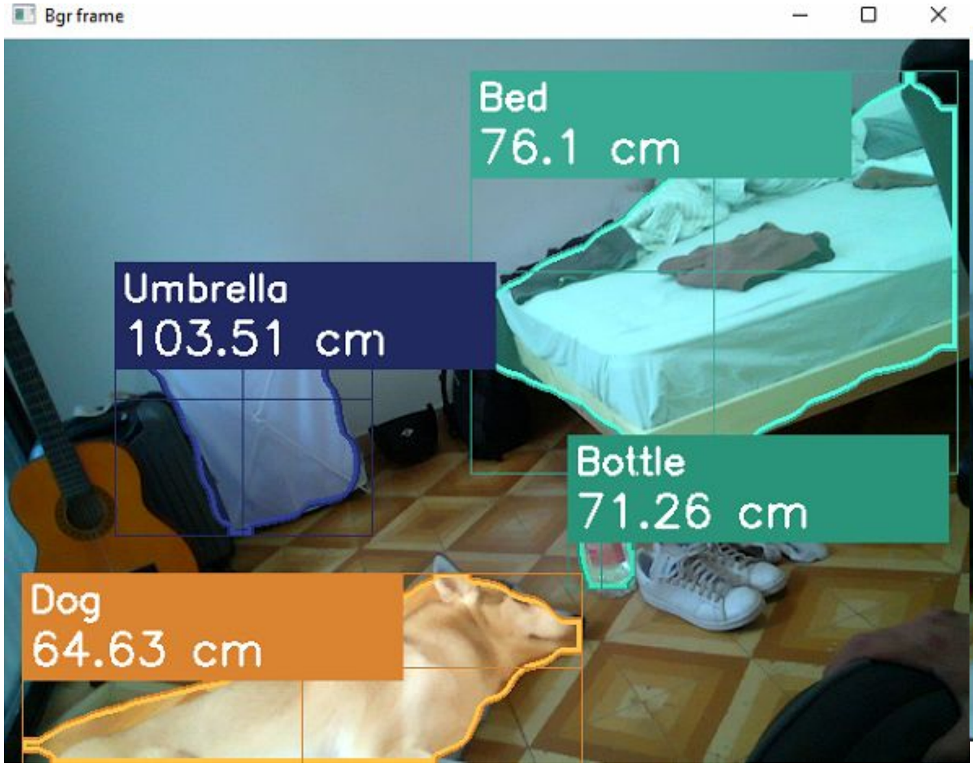


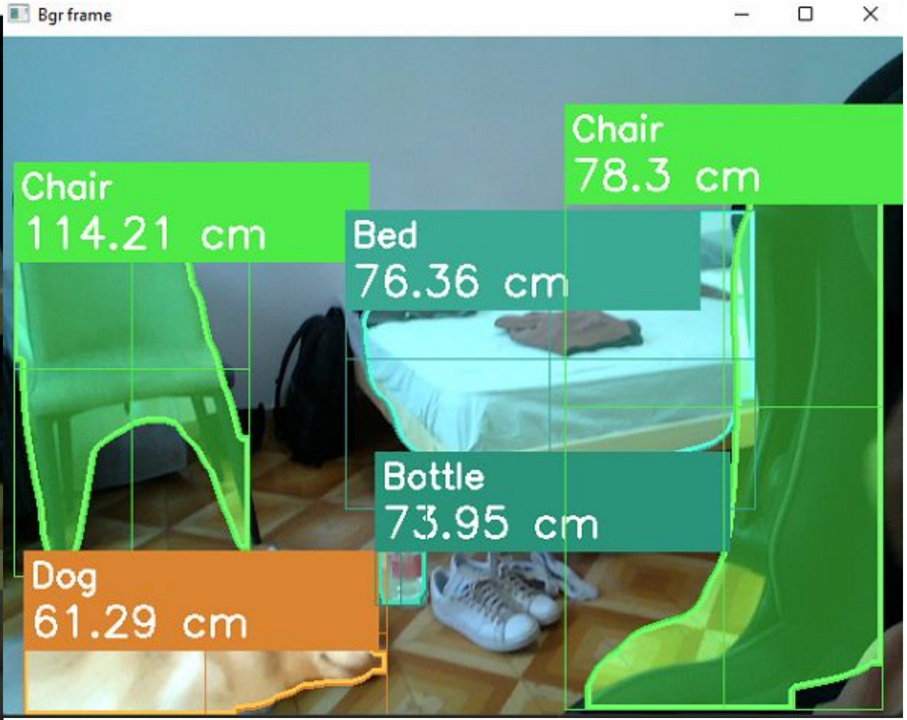
Distance detection with Depth Camera - Pycharm

Distance detection on mouseover.

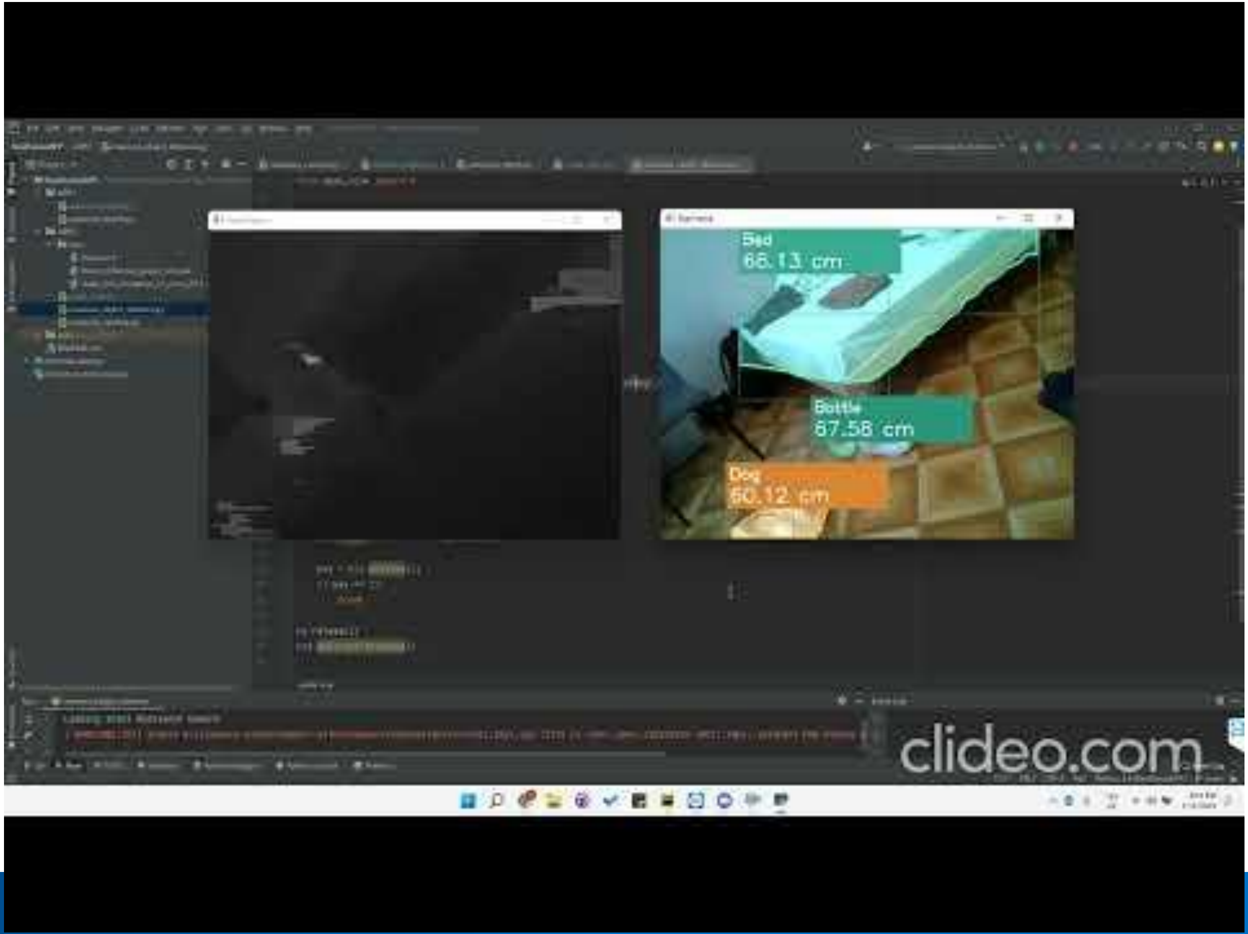


Identify and Measure precisely Objects distance - Pycharm





Identify and Measure precisely Objects distance - Pycharm



Points to follow

- We were unable to add filters that would improve the camera detection quality - there are points that the camera does not detect

- Connecting the code to an object such as: robot, drone, guide dog.
  The combination of the two will make the object smart (for example: the robot will know how to navigate without getting stuck between the objects by the detection.)

- Better GPU - In our project the FPS transfer slowly.