Intel Realsense D455 Evaluation

Introduction

This is a document summarizing properties of the Intel Realsense Depth stereo camera. The manual for the sensor includes technical data and information (see Software below).

This document, however, focuses on information acquired from our testing with a sensor available and analysing properties of interest. The goal is to make it clearer what kind of measurement cases this sensor would be useful for.

Equipment

The equipment used during testing was:

- Intel Realsense D455 (to be evaluated), with power and communication through Type-C to USB 3.0-cable. This product is classified as a Class 1 Laser Product under the EN/IEC 60825-1, Edition 3 (2014) internationally. This means that it is generally eye safe, but do not look right into the IR emitter through any optical instruments.
- Test objects for measurements:
  - Box 1 (see Figure 1 below).
    - Width: 65 mm
    - Depth: 65 mm
    - Height: 80 mm
  - Box 2 (see Figure 1 below).
    - Width: 73 mm
    - Depth: 73 mm
    - Height: 91 mm
    -
- Laptop with a USB 3.0-port.

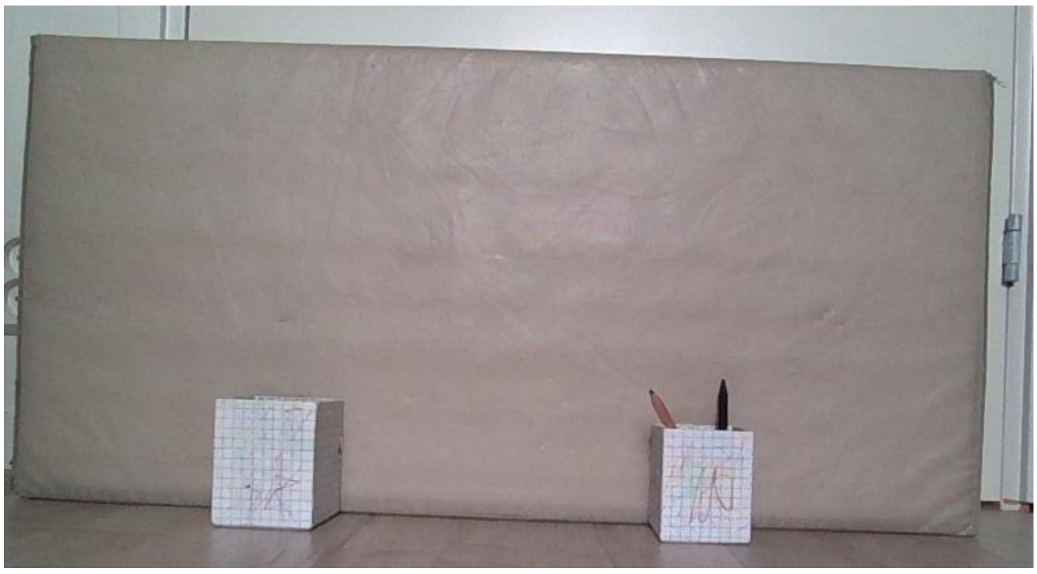


Figure 1: Boxes used for measuring different sizes and smaller differences in distances (cm level) at close range (1 – 2 m).

Software

Communication with the camera was done using the Intel Realsense SDK 2.0 in Python. Python scripts were written for this purpose, which allows the communication of the 3D-data from the camera to be handled programmatically as the user sees fit. Programming is therefore necessary to implement the camera with other software systems.

There is however an Intel Realsense viewer (Intel.RealSense.Viewer.exe) which can be used for a quick evaluation of the sensor on Windows, available at Intel Realsense GitHub repository.

Instructions can be found here: https://www.intelrealsense.com/get-started-depth-camera/).

Start the sensor:

- Download the viewer from Intel Realsense's GitHub webpage.
- Run the viewer.
- In the left column you can find the available cameras in a list.
- You can activate the stereo module (depth data), RGB module and the motion module by clicking the ON-buttons next to the respective module name. The data acquired from the activated modules can be seen in 2D or 3D perspective in the main subwindow to the right.
- There are several settings for the sensor that can be adjusted in the viewer.
- Press the Record-button to save measurement data to .bag-files. You will need other existing software or your own to read those files. The file will contain data only from the modules that are currently activated in the viewer.
- Using the viewer, you can also update the internal firmware for the camera. A box shows up in the upper right corner of the viewer asking to install available versions.

Other approach

Use the Intel Realsense SDK to write C++ or python code for communicating with the camera and processing the data. Connect the camera and run your scripts.

Suggestion: Open3D (separate evaluation in progress) is a 3D data processing and visualization SDK that can be used for writing code for communication, data processing and visualization of the 3D data from the D455 camera. This can work well together with the Intel Realsense SDK.

Sensor settings

Many different settings can be changed for the D455 camera, using the Intel Realsense SDK. Like in the Intel Realsense viewer, the camera can be controlled, as well as activating or adjusting post-processing tools for the data that the camera delivers.

Camera parameters include:

Stereo module

- Resolution
- Frame rate (depends on the resolution)
- Light emitter enabled.

- Available streams to show in 2D mode (Depth, Infrared camera 1 and Infrared camera 2, color)
- Enable auto exposure.
- Controls include:
    - Exposure time
    - Gain
    - Laser power
    - Thermal compensation

Also included are settings for post-processing, including adding different filters (threshold, spatial, temporal, hole filling, etc.). There are many other controls that can be tweaked to get the best results for measuring different scenes.

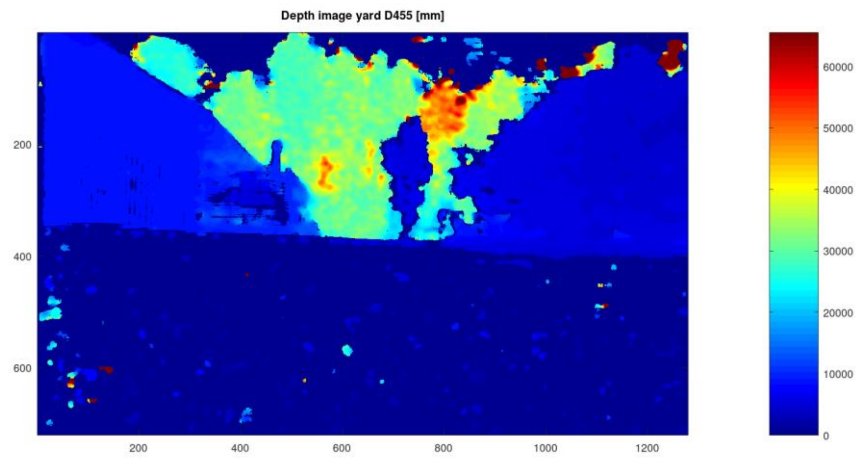
When running your own scripts, much of the functionality is available through the SDK where the settings for the camera can be adjusted, or you can do a lot of the filtering yourself. Depending on if accuracy or resolution is more important than the frame rate you will want to use different setting.

One property that can be adjusted for the camera is *DS_Second_Peak_threshold*. This parameter allows the user to decide what confidence interval the depth estimation of a data point must be in before that point is deemed inaccurate and is set to 0, meaning no data at that point. This is important in order to remove uncertain depth measurements that can behave like noise with unpredictable depth values, so that unreliable measurements from the camera aren't used.

Sensor properties of note

Range

The stereo camera can yield measurements from far away from the camera, even 20 m and farther away (see Figure 2 below). Also, measurement data from much closer, closer than 1 m, can be acquired. If the monochrome images from the stereo pair cameras get enough detail so that the matching between the two images can be performed, the camera can measure the distance to those points in the images. This means that the scene needs enough illumination, either from external light (sunlight, lamps, etc.) or from the structured light from the IR emitter of the camera (see Accuracy - Emitter usage below). The illumination as well as the distance from the camera to the objects in a scene, will affect the possibility of making accurate measurements (see Accuracy below). Flat uniform surfaces or repetitive patterns, like a fence, will make stereo matching difficult (again, Figure 2 below).



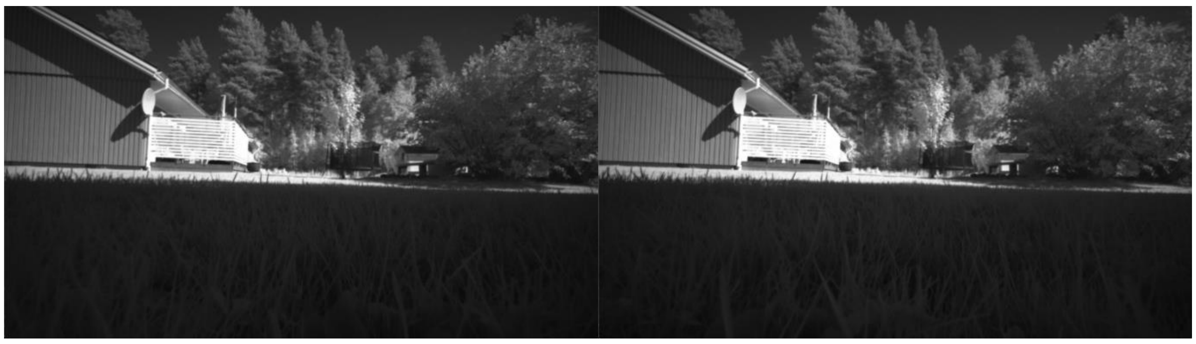


Figure 2:Top: Depth image from the Intel Realsense D455 camera of a backyard. The different colors represent different distances from the camera. Bottom images: These are the monochrome stereo pair camera images that are used for calculating the depth image. The white fence wind protection can be recognized, even in the depth image, but the accuracy of the measurement is reduced in that area, possibly due to the repeated pattern of bright white color resulting in problems when matching points between the two monochrome images. The trees, bushes and the house can be seen in the depth image as well. The camera can get measurement data from far away from camera (some trees are about >30 m from the camera), but the accuracy and precision of the camera will decrease with longer distances from the camera.

The minimum distance for the depth measurements varies depending on the settings of the camera. For instance, by reducing the (X-)resolution of the camera (see Figure 3 below), the camera can measure points closer to it, but this lower resolution also results in lower precision for other objects in the scene.

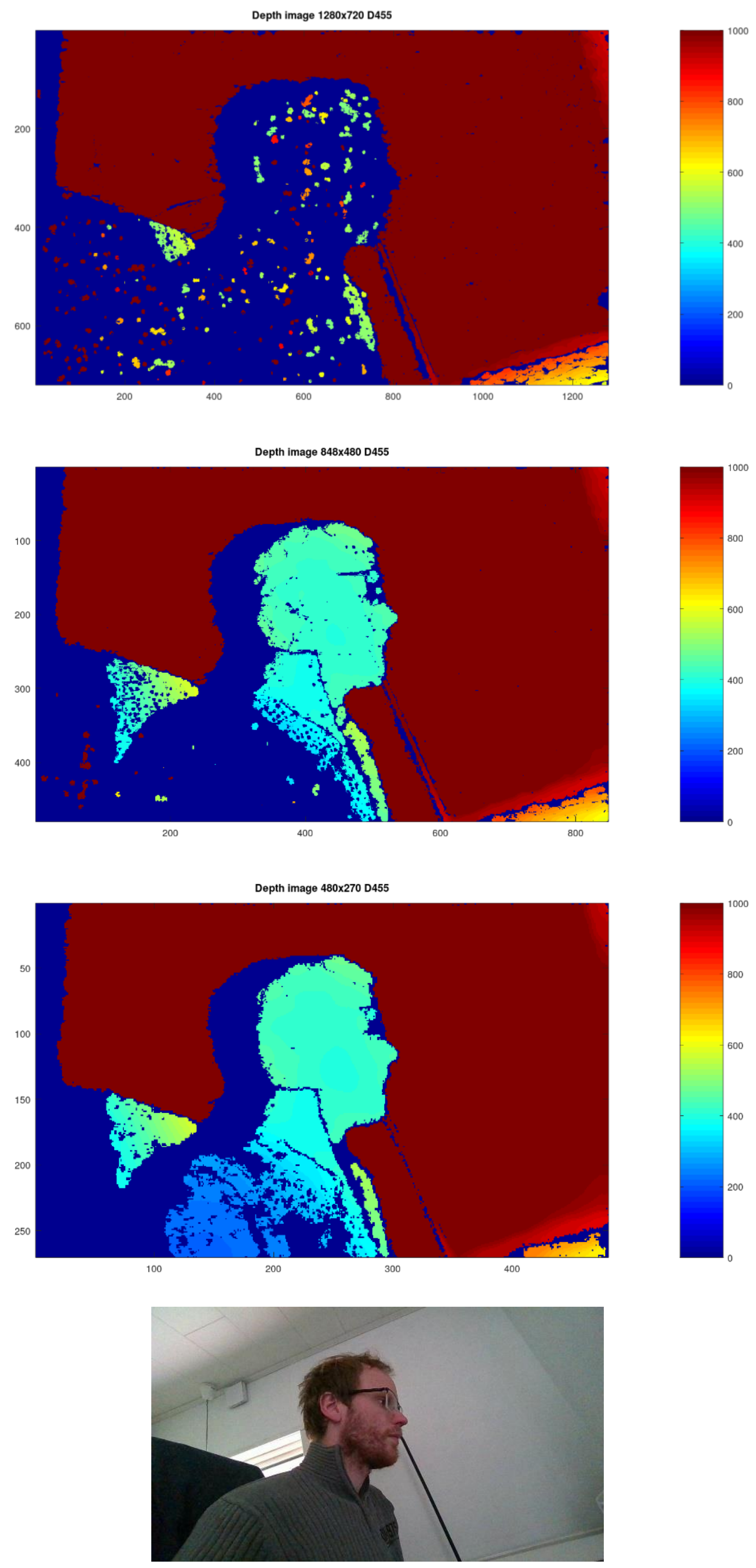


Figure 3: Depth images (measurements in mm) and a color image from the D455 3D camera. Illustrating how depth values are acquired for points closer to the camera when decreasing the X-resolution of the camera image. The highest resolution (top image) doesn't yield accurate values for a person at about 500 mm from the camera, but with decreasing resolution the images show more and more points at closer distances, down to about 200 mm from the camera for the lowest resolution.

Accuracy

The accuracy and resolution of the sensor can be found in data sheets and have not been analysed extensively. But the camera, for longer ranges, is less accurate for finer details of an object. The resolution is adjustable which is useful to get more details of the object but must be balanced against the frame rate. The camera is sometimes unable to accurately measure parts of a scene for reasons that can be hard to clearly identify. The camera is thus not the most accurate 3D camera on the market, but unlike many affordable 3D ToF or structured light cameras, the camera can perform measurements of outdoor scenes in direct sunlight and at distances of +30 m, while also measure darker scenes at closer range, thanks to the camera's IR emitter.

In Figure 4 and Figure 5 below, measurements of scenes including objects as far away as 20 m or more are displayed. The images show how the camera captures measurement data from a long-distance range for this kind of camera, but it must be noted that the accuracy is generally decreased with longer distances.

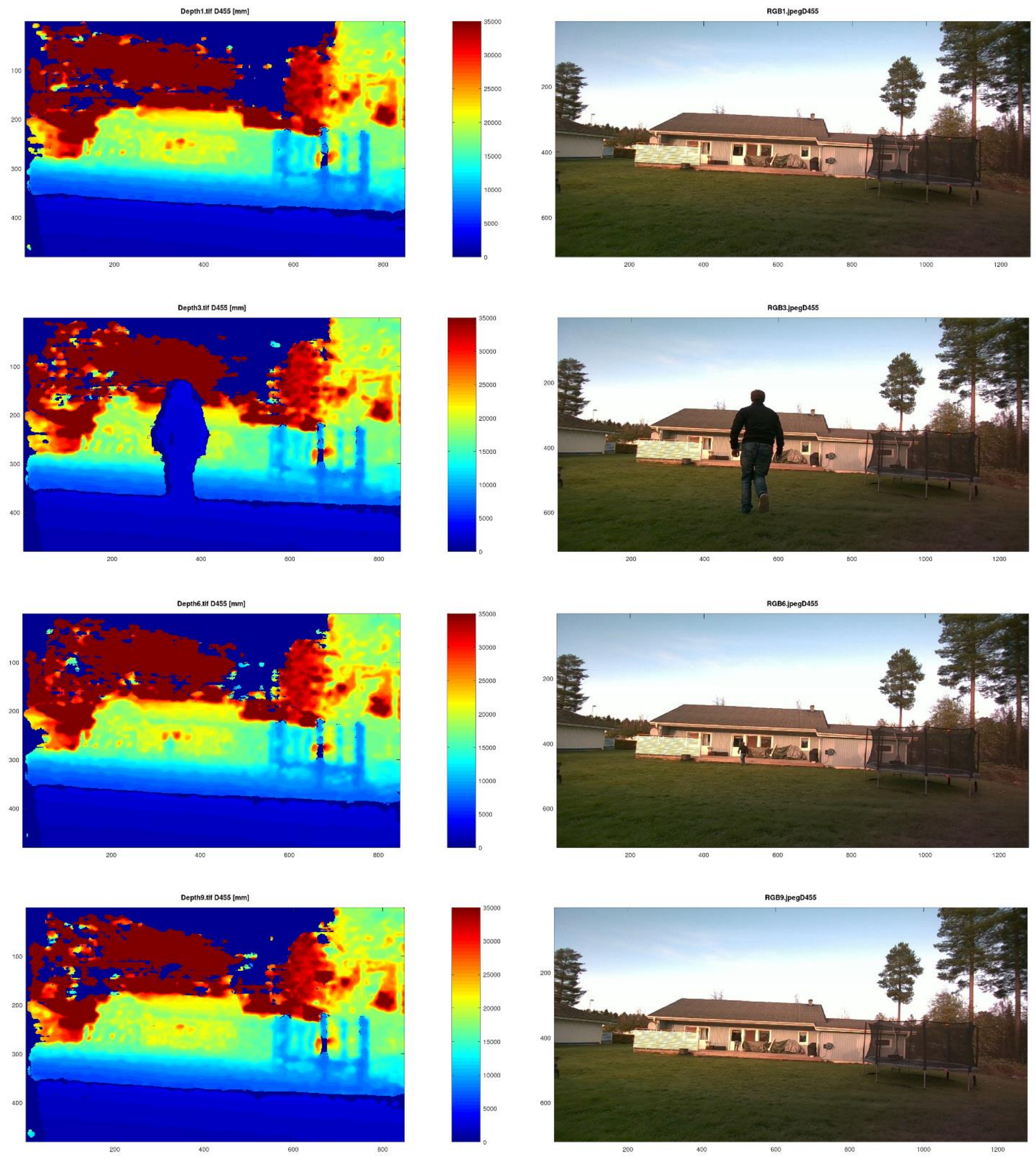


Figure 4: Depth images (to the left) and corresponding color images (to the right) from the D455 3D camera. A person is running farther from the camera, being closest in the second form the top image (Depth3) and being out of sight in the top image (Depth1). In the bottom image (Depth6) the person is still visible far away, about 20 m from the camera, in the color image but cannot be seen in the depth image. The accuracy of the depth measurement is decreased when moving farther away from the camera.

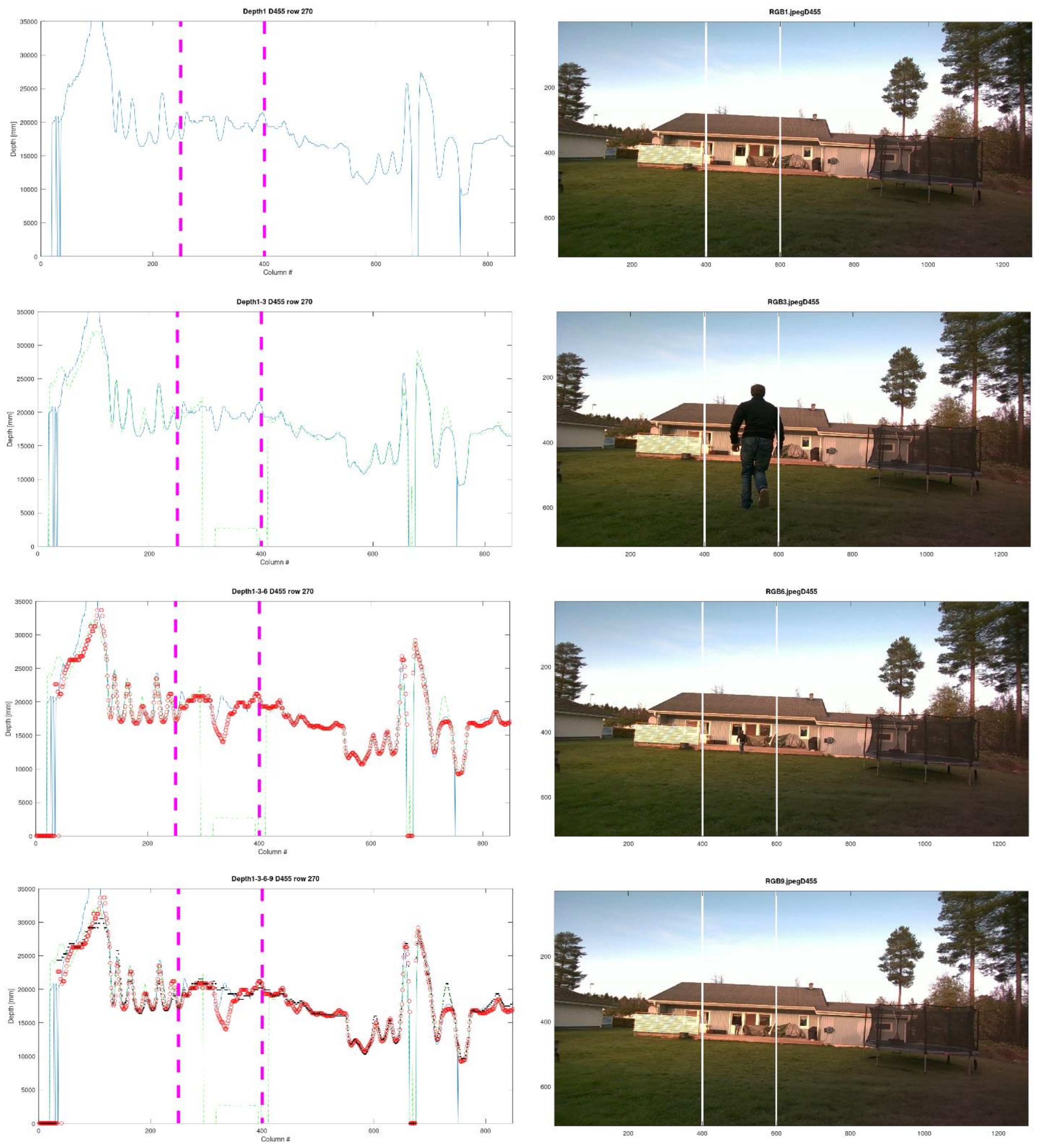


Figure 5: 2D plots across one row of different depth images (to the left) and full color images (to the right) from the D455 3D camera. A person is running farther from the camera, being closest in the second from the top image (RGB3) and being out of sight in the top image (RGB1). In the bottom image (RGB9) the person is still visible far away in the color image but cannot be seen in the depth data. The plots show the cross section of a row that includes depth data from the person, for all but the first top image where the person is out of sight for the camera. Every plot keeps the preceding cross sections above and adds the one related to the measurement shown in the color image to the right of the 2D plot. The striped vertical pink lines in the 2D plots correspond to about the same area as the vertical white lines in the color images, that includes the person running. The top and bottom images show how it is not easy to differentiate between the two 2D plots, even though in one measurement the person is still standing in front of the camera, but about 20 m away, and in the other the person cannot be seen in the image. While the camera can get measurement data from farther distances from the camera, the depth measurements in general become less and less accurate the farther away the scene is.

There are also data points that appear similar to noise of a signal, where measurement data is estimated but is not accurate, like there can be data points appearing in the sky in the depth images that should clearly not be there. By taking several consecutive images, for a static scene, those data points can be removed with filtering methods, where the points need to be measured as consistent values between frames. This causes a problem for measuring objects in motion, but that's where the *DS_Second_Peak_threshold setting* can be important. By only keeping points of high enough confidence level that they are accurate, the measurement results can be improved (see Figure 6 below).

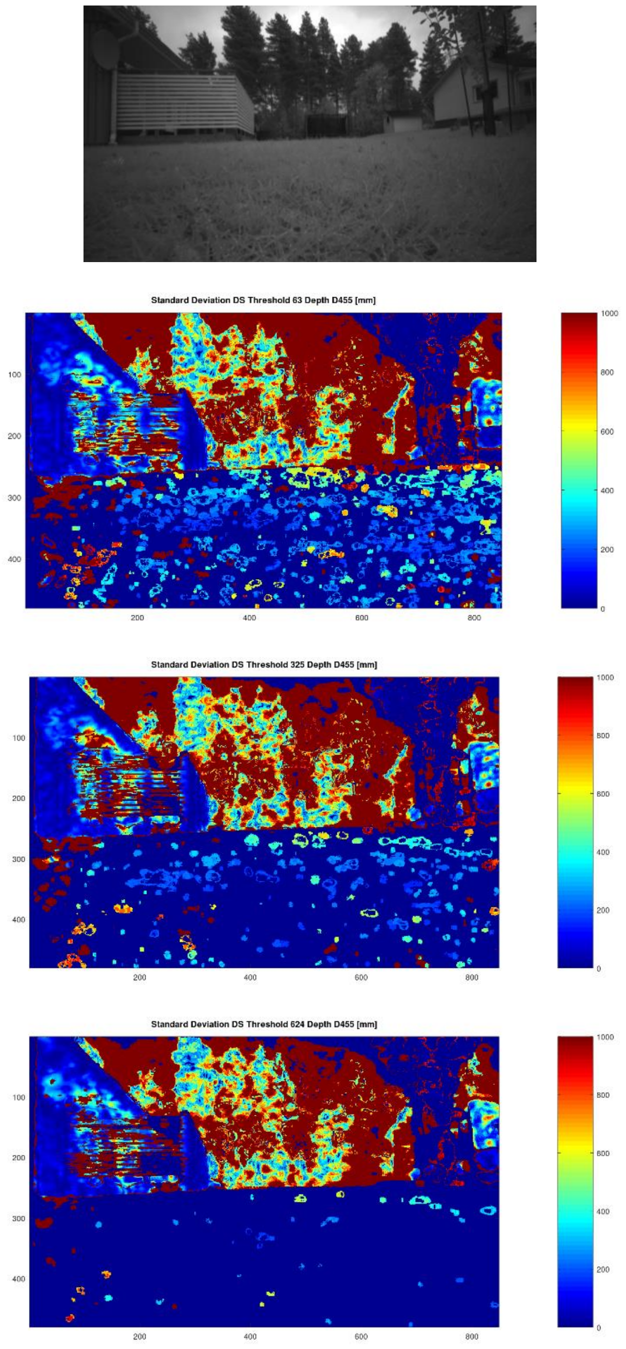


Figure 6: Top: Monochrome IR image. Below: Images from a D455 camera where the standard deviation of depth is at every pixel are shown, derived from 5 consecutive depth images. From top depth image to bottom the DS Second Peak threshold goes from lowest to highest. In the lower parts of every image, where the ground is, the standard deviation varies a lot for lower threshold values, while for higher thresholds more data points are neglected in order to keep valid data points to as high degree as possible and remove inaccurate depth measurement points.

Accuracy - Emitter usage

The IR emitter of the camera can be turned on and off, depending on your needs. With the emitter off you can get the IR image from a scene without the additional IR-dot pattern that the emitter adds. If you measure a scene that contains enough IR light, like measuring a scene in broad daylight, the camera can still measure the scene. However, if you are measuring a scene where there is little to no external light the camera might not be able to get any measurement data. By turning the emitter on it is possible to perform measurements even in these cases (see Figure 7 below). The emitter also allows for better measurement of flat and/or uniformly colored/shaped surfaces by adding a pattern to the scene. The edges of objects also tend to be more accurately measured when using the emitter.

For long distances the IR light intensity will be weaker when it reaches the objects, meaning that for long distances ambient light sources are important for performing depth measurements with the stereo camera. Long distance measurements are common in outdoor applications, where sunlight helps the stereo camera to perform measurements. Still, the accuracy and precision will decrease when measuring objects farther and farther away from the camera.

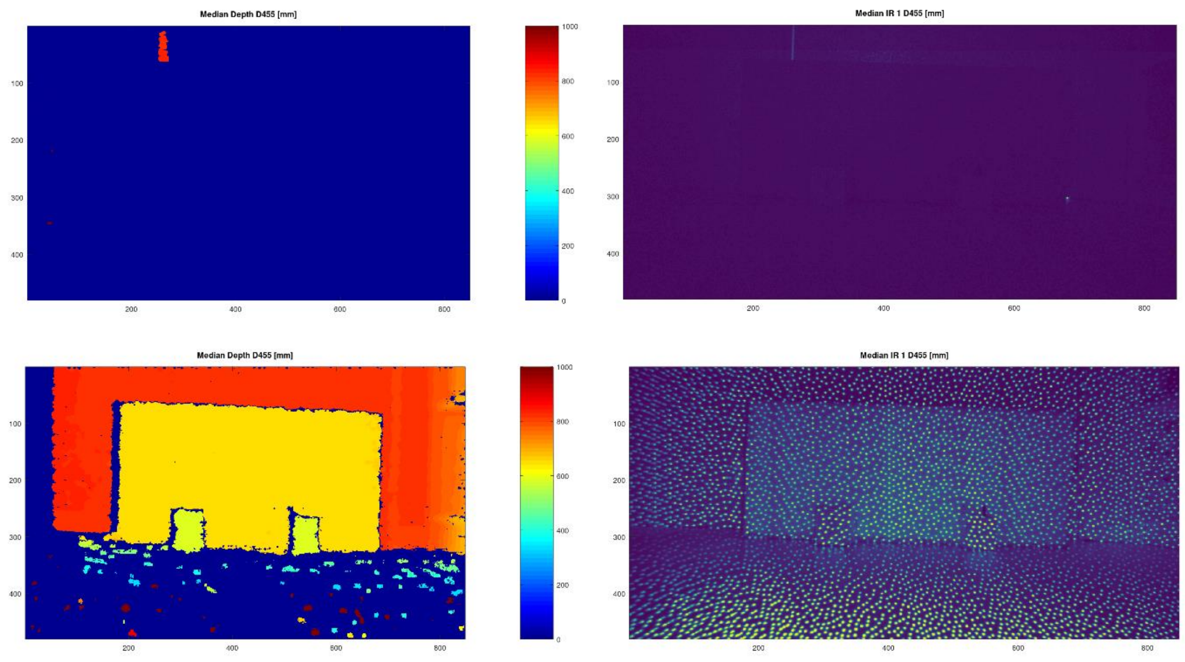


Figure 7: Depth images (to the left) and corresponding monochrome left IR images (to the right) from the D455 3D camera. These images were calculated as the median from 5 consecutive depth/IR images. The room was closed with windows covered and no lights were on. In the upper images no emitter was used, while the lower two images show the result when the camera's IR emitter was used. The bottom IR image shows the IR pattern from the emitter as several bright dots can be seen in the image. These help the stereo camera to perform well in the stereo pair matching, and thus the resulting depth measurements. In this case the scene can be measured when the emitter is used. When it is not, the room is simply too dark for the camera to see all the details in the scene to perform well, resulting in a mostly blank depth image.

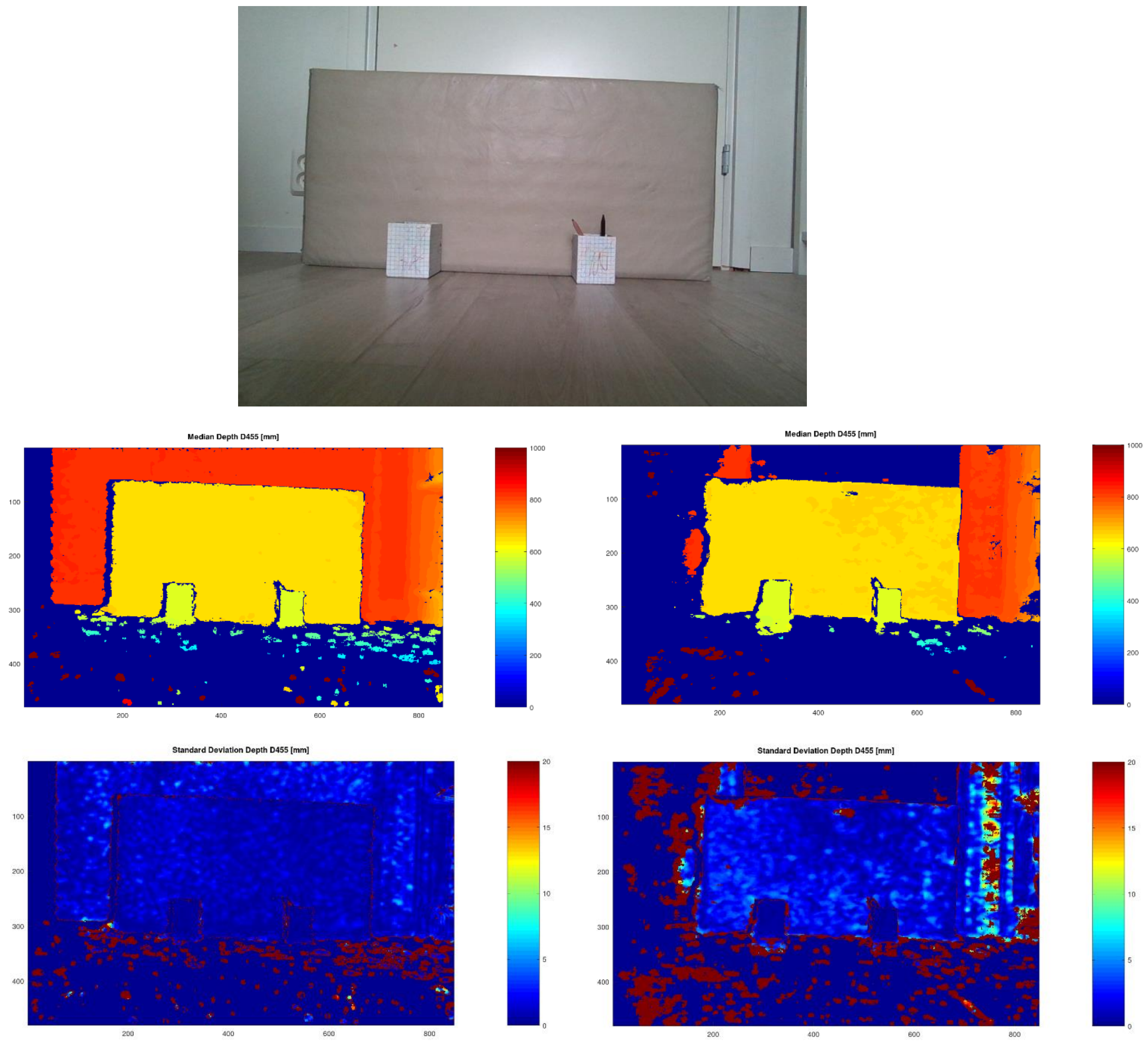
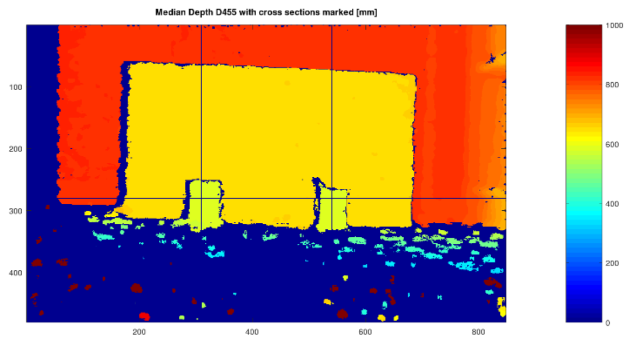
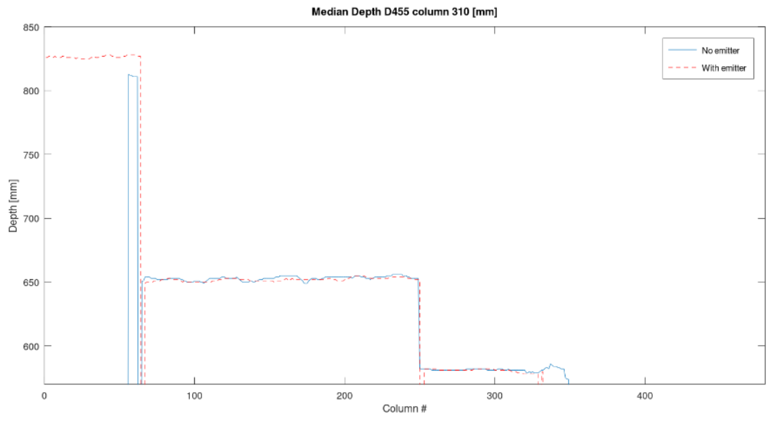
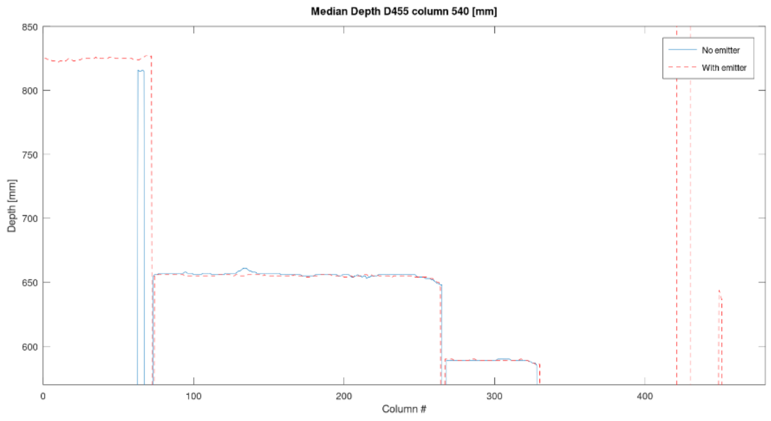


Figure 8: Color and depth images from the D455 3D camera. The depth images were calculated as the median from 5 consecutive depth/IR images. The room was illuminated by the light from a window during the bright daytime (cloudy weather). In the left image the emitter was active, and in the image to the right no emitter was used. The emitter allows for acquiring measurement data for a bigger part of the scene. The objects in the image have clearer edges and parts of the door and wall behind the objects could only be measured with the emitter, likely because the flat uniformly colored surface of a door/wall makes it hard to measure using stereo camera since those surfaces lack the details needed to correctly match points between the stereo images. The bottom images show the standard deviation for every pixel in the image. The color scale visualize that the standard deviation is higher (a few cm) for the measurements without the emitter (image to the right). At the edges of the objects the standard deviation gets very high, to a higher degree for the image where the emitter was not used during the measurements.







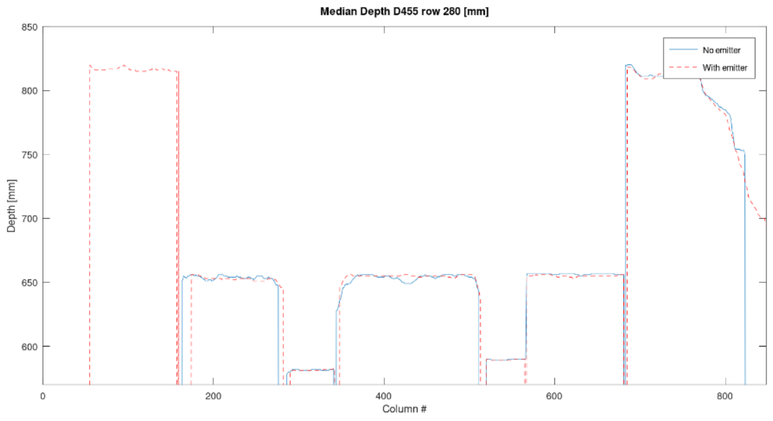


Figure 9: Top: Depth image showing the cross sections visualized in the 2D plots. The 2D plots show the cross sections at depth image columns 310 and 540 and row 280. The plots show that the measurement data is more stable when using the emitter, as well as yielding measurement data for the more uniform parts of the scene like the wall. But the measurements are still roughly the same for the areas where measurement data could consistently be acquired (rug, boxes). The jumps in

depth level should correspond to the real-life distances from left to right: 160 mm (missing for the no emitter data), 73 mm, 65 mm and 160 mm. Within a few cm, the measurement data is in line with this.

Sensor speed:

The camera frame rate depends on other settings of the camera, mainly the resolution.

The depth frame, calculated from both monochrome IR cameras, can handle a frame rate up to 100 frames per second for the resolution setting 848 x 100 only, which decreases the field of view for the camera significantly, but can be suitable for certain purposes. The IR frames can be streamed at the same time with these settings.

Otherwise, when streaming the monochrome IR frames and the depth frame together, the frame rate limit is 90 fps up to a resolution of 848 x 480. And at resolution 1280 x 720, the limit is 30 fps. Higher resolution than that does not seem to work for the depth frames.

For the IR frames only, the limit is 30 fps for resolution 1280 x 800.

It should be noted that some combinations of frame rates and resolutions don't work when using depth together with the IR frames. The Intel Realsense viewer is a good tool to test this.

For the RGB camera 90 frames per second is possible when using the resolution 640 x 360. The limit is 60 fps up to resolution 848 x 480. At higher resolutions the limit is 30 fps.

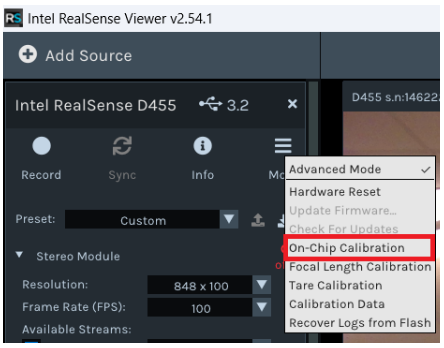
Inertial measurement unit

The sensor has an inertial measurement unit, Bosch BMI055, which allows the gyroscopic and acceleration data for the sensor to be acquired.

Recalibration

If a recalibration is ever needed, which can be seen if the camera starts to perform worse over time due to changes that have occurred internally for the sensor, due to for example exposure to shocks or very low or high temperatures, the instructions for this can be found here: https://dev.intelrealsense.com/docs/self-calibration-for-depth-cameras?_ga=2.251945481.1260781527.1697620687-211355368.1685949733

The Intel Realsense viewer has the tools for performing an On-Chip calibration (see Figure 10 below).



Intel RealSense Viewer v2.54.1

Intel RealSense D455 3.2

D455 s.n:14622

Preset: Custom

Stereo Module

Resolution: 848 x 100

Frame Rate (FPS): 100

Figure 10: A On-Chip calibration (improving depth image quality) can be performed using the Intel Realsense viewer on a target with over 50 % valid depth points/pixels.

Output data:

The camera can deliver:

- Depth image, where every pixel contains a distance measurement.
- RGB image, a regular colour image of the scene.
- Gyroscope and acceleration data for movement

The data can be saved from the viewer in .bag format, but if you write your own scripts you get the freedom to save the data however you want. Using python the frames from the camera, through the Intel Realsense SDK, can be converted to numpy arrays to be used for your specific purpose:

```
import pyrealsense2 as rs

# Configure depth and color streams
pipeline = rs.pipeline()
config = rs.config()
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 60)
config.enable_stream(rs.stream.color, 640, 480, rs.format.rgb8, 60)
config.enable_stream(rs.stream.infrared, 1, 640, 480, rs.format.y8, 60)
config.enable_stream(rs.stream.infrared, 2, 640, 480, rs.format.y8, 60)


# Start streaming
pipeline.start(config)

# Main loop
while True:

    # Wait for a new frame
    frames = pipeline.wait_for_frames()

    # Get depth and color frames
    depth_frame = frames.get_depth_frame()
    color_frame = frames.get_color_frame()
    ir_frame1 = frames.get_infrared_frame(1)
    ir_frame2 = frames.get_infrared_frame(2)


    # Convert depth and color frames to numpy arrays
    depth_img = np.array(depth_frame.get_data())
    color_img = np.array(color_frame.get_data())
    ir_img1 = np.array(ir_frame1.get_data())
    ir_img2 = np.array(ir_frame2.get_data())
```

Also, for calculating XYZ-coordinates instead of just the depth measurement, the SDK open3D can be used. That way a complete point cloud from every depth image can be acquired.

```python
import open3d as o3d


# Configure depth and color streams

pipeline = rs.pipeline()

config = rs.config()

# Enables all streams (reading from .bag file)
# config.enable_all_streams()

# Enable streams one by one
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 60)
config.enable_stream(rs.stream.color, 640, 480, rs.format.rgb8, 60)
config.enable_stream(rs.stream.infrared, 1, 640, 480, rs.format.y8, 60)
config.enable_stream(rs.stream.infrared, 2, 640, 480, rs.format.y8, 60)


# Start streaming
pipeline.start(config)

# Main loop
while True:


 # Wait for a new frame

    frames = pipeline.wait_for_frames()


    # Get depth and color frames

    depth_frame = frames.get_depth_frame()
    color_frame = frames.get_color_frame()
    ir_frame1 = frames.get_infrared_frame(1)
    ir_frame2 = frames.get_infrared_frame(2)


    if not depth_frame or not color_frame or not ir_frame1 or not ir_frame2:
        continue


    # Convert depth and color frames to numpy arrays

    depth_image = o3d.geometry.Image(np.array(depth_frame.get_data()))

    color_image = o3d.geometry.Image(np.array(color_frame.get_data()))
   # Create Open3D point cloud from depth and color images
   rgbd_image = o3d.geometry.RGBDImage.create_from_color_and_depth(color_image, depth_image)
   pcd = o3d.geometry.PointCloud.create_from_rgbd_image(rgbd_image, o3d.camera.PinholeCameraIntrinsic())

   # Get X, Y, and Z coordinates from the point cloud
   points = np.asarray(pcd.points)
   # Get first point X, Y and Z
   X = points[0,0]
   Y = points[0,1]
   Z = points[0,2]

   # Store points in array
   pcd_array.append(points)

   # Check for key press
   key = cv2.waitKey(1)
   if key == ord('q'):
       break

# Stop streaming
pipeline.stop()

# Write point cloud data to file
length_arrays = len(pcd_array)

for ind in range(0, length_arrays, 1):
    np.save("C:\\Python\\Python_all_purpose\\D455_images\\pcd" + str(ind+1) + ".npy", pcd_array[ind])
```

User cases

The properties of the D455 active stereo camera can make it useful for different measurement applications, including:

- Moving vehicle (like drones) measurement, for controlling the drone and avoid obstacles or even measure objects. Many measurements can be used together to create a 3D model of a larger object using a flying drone which films the area with the D455 camera (though you'll need to find/build and test your own software for doing this). The weight of the sensor is light enough (< 1 kg) that certain drones can handle it.
- Measurements in direct sunlight.
- Close range measurements with cm accuracy, even in dark areas, using the IR emitter.
- Measurements of moving objects due to the high framerate (limits the resolution).
- If the highest accuracy is not of most importance, even objects farther away (+10 m) can be measured if they are big enough to discern from other objects. Expect dm precision at best here. Farther away, +20 m, the camera can measure several meters wrong.

Summary

The D455 depth sensor can be easily up and running for a quick evaluation with the Intel Realsense viewer available. To use the measurement data from the sensor in real-time for other applications, some programming experience (C++, Python) will be needed. In which case the camera data is easily converted to data types that work with well-distributed image processing library OpenCv, as well as 3D data processing SDK:s like open3D, in python. There are tutorials available for getting things up and running quickly.

The camera can be used to get 3D measurements without any calibration. However, with time a recalibration can be needed if changes have occurred internally for the sensor, due to for example exposure to shocks or very low or high temperatures. The Intel Realsense SDK does offer code for performing a recalibration with the proper target (see Intel Realsense GitHub page), and the Intel Realsense viewer has tools for performing an On-Chip calibration (improving depth image quality).

The high framerate allows for measurement of fast-moving objects, with a decreased resolution affecting the precision of the measurements of the dimensions of the objects. Stationary objects can, with a lower frame rate, be measured in a bit more detail. The camera's weight is around 0.4 kg, which makes it easy to put on a moving vehicle for measurements while moving, where the gyroscopic and accelerometer data can be of use for positioning.

Knowing the limited accuracy of the sensor it can be useful for many applications where dm-accuracy is needed. For close range measurements the camera can perform better than that though, even within a few cm.

The cost for a D455 camera is around 5000 – 6000 sek.

The D455 depth camera can be a useful and affordable choice for measurements indoors as well as outdoors, stationary as well as moving applications, if the application doesn't require sub cm precision or resolution.