ADOPTICUM

OAK-D-Lite Evaluation

This document gives our evaluation of the OAK-D-Lite. It is based on evaluation of the sensor performed by Stiftelsen Adopticum in a project financed by Kempestiftelserna. For more information about the sensor, please feel free to contact Adopticum.

Introduction

This is a document summarizing properties of the OAK-D-Lite stereo, color and smart 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:

- OAK-D-Lite camera (to be evaluated), with power and communication through Type-C to USB 3.0-cable. Early model of the camera, without an inertial measurement unit (IMU).
- Numerous measurement objects: Tables, chairs, phones, vehicles, people, etc.

Software

DepthAI is a spatial AI platform, for computer orientation in our 3D world, that uses a combination of 3D measurement techniques and (predefined or your own developed) AI models. The OAK cameras deliver 3D data. To communicate with an OAK camera DepthAI should be installed, which does include demo applications complete with a graphical user interface (GUI) where you can get the OAK camera up and running, try it out and even save image data.

DepthAI includes examples, written in python, that can be used for getting started with testing the camera. To use the camera for your own tailor-made applications you must develop your own python scripts, which means that you need knowledge of how to setup an environment for python and how to write python scripts, but the examples and the DepthAI documentation will help you get going.

Installation instructions:

[If you aim to get coding right away, see Creating your own applications below.]

Before you can use your OAK camera, follow the steps below.

- Visit Luxonis homepage and find the chapter First steps with DepthAI. https://docs.luxonis.com/en/latest/pages/tutorials/first_steps/

- Go to the section Installing DepthAI
- Choose your operating system and follow the instructions.
    - For macOS and Linux you get scripts to execute on the command line to install DepthAI.
    - Windows instructions (see Figure 1 below):
        - After clicking on the Windows tab in the Installing DepthAI-section, click on the link *Windows Installer* to download the installation file.
        - Run the installation file.
        - Follow the instructions of the installer. Choose the default settings unless you are more familiar with DepthAI. Choose to create a shortcut on the desktop.
- Otherwise, manual instructions are available on the same webpage.

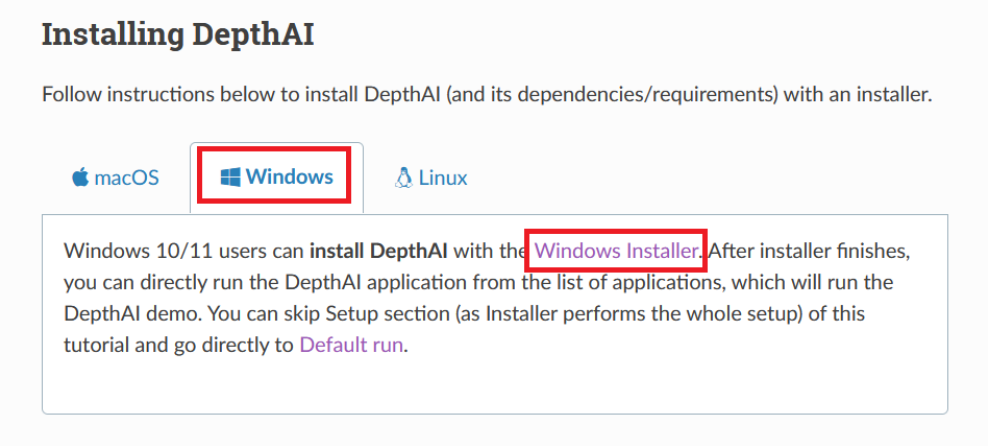

Figure 1: Instructions how to get the installer for DepthAI for WIndows through First step-instructions at Luxonis webpage. The red boxes show where to click to get the installer downloaded.

Getting started with the sensor:

Follow the instructions below to start using the camera (see Figure 2 for an overview of the GUI).

- Use a USBC to USB3 cable and connect it to the camera and the computer.
- Run the DepthAI-application (through the link on the desktop or by running the depthai_demo.py script).
- The application starts with the GUI showing the camera streaming rgb-images together with XYZ-coordinates for the objects that are identified with the AI-model "mobilinet-ssd".
- The camera settings can be adjusted, after which the "Apply and Restart" button is activated. Press it to activate the new settings.
- In the upper left droplist you can choose which camera image to show: Color, depth, left or right monochrome camera, etc.
- In the tabs AI, Depth, Camera and Misc you can choose different settings for the respective category:
    - AI: Choose Convolutional Neural Network(CNN)-model that is used and what camera stream (rgb, monochrome left/right) that the model uses as input.
    - Depth: Choose settings for the depth stream like choice of filter, confidence interval for the depth measurements and depth range.

- Camera: Frame rate and resolution for the color camera or left and right monochrome cameras. Other properties that can be adjusted are saturation, contrast, brightness and sharpness. RGB-Depth alignment is also available.
- Misc: Recording settings like what camera stream to record and recording frame rate and destination for recorded image data. Internal data for the camera, like temperature, CPU and memory usage can also be recorded.

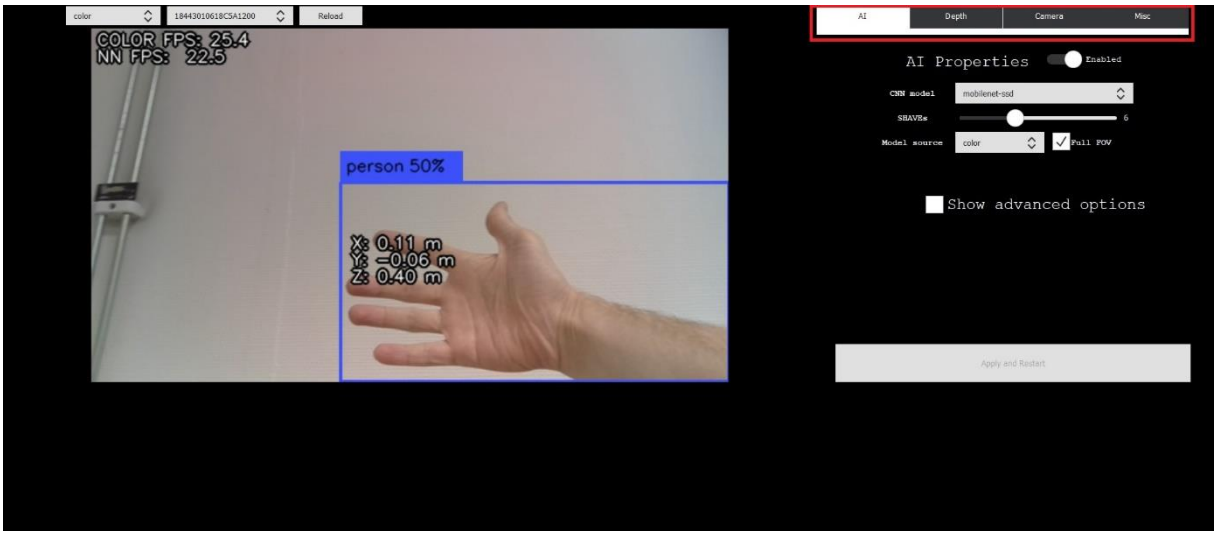

Figure 2: Graphical user interface for the depthai_demo-application with the OAK-D-Lite camera. What camera stream that is shown can be adjusted in the upper left corner, while the settings for the different categories AI, Depth, Camera and Misc are shown when you choose in the menu within the red box shown in the upper right corner of this image. In this image the category AI is highlighted, where you can see the output of the current (adjustable) CNN-model that is used in the current image stream shown.

Creating your own applications

The installation of DepthAI includes example code scripts for python that can be used to get started developing applications using DepthAI.

For developing applications, you can also go to github and clone the repositories, using git and the HTTPS link:

https://github.com/luxonis/depthai

https://github.com/luxonis/depthai-python

Several example python scripts will be available to get started using the camera and the available CNN models.

ADOPTICUM

Sensor settings

The OAK-D-Lite allows for many different settings for the different cameras: color, monochrome and stereo depth. This section goes through settings available by default from the depthAI SDK. Some of these, like filtering, can be implemented by the programmer, but if they are mentioned here there is an implementation ready from the SDK.

You can, in your own applications, stream/show any or all of them at the same time:

- Color (4K color camera)
- nnInput (image as input format to CNN model)
- left (left monochrome camera)
- right (right monochrome camera)
- rectifiedRight (calibration compensated image)
- rectifiedLeft (calibration compensated image)
- depth (depth data uint16)
- depthRaw (depth data from stereo in raw units from the stereo computation, used to calculate depth)
- disparity (Raw disparity, used for calculating raw depth. See Depth settings below)
- disparityColor (Disparity colorized for clearer visual representation)

Camera: Monochrome and color camera settings

- Frame rate (affected by resolution)
- Resolution (affects possible frame rate)
- Saturation level (intensity of the colors in an image)
- Contrast level (the ratio of different colors/tones in an image)
- Brightness level (how dark or light an image is)
- Sharpness level (how clear the details in an image are)

Depth: Depth processing parameters. The stereo depth processing can be turned off.

- Confidence threshold from 0 (max confidence that the value is valid) to 255 (minimum confidence, the value is more likely to be incorrect).
- Subpixel (the disparity will be estimated using 2 neighbouring pixels confidence values as well as for the current pixel for better depth precision).
- Extended disparity (runs the depth pipeline twice with full and downscaled resolution by 2 and the resulting disparity is a combination of the two, which will allow for closer range measurements)
- Left-Right Check (removes incorrect disparity data at object borders)
- Depth Range (the minimum and maximum distance calculated by the camera)
- Bilateral sigma (represents how many neighbourhood pixels are used together)
- LRC Threshold (Specifies the maximum disparity difference before a pixel value is not used)

System information:

- Temperature
- CPU usage
- Memory usage

AI: The camera to use the AI model on can be selected (monochromatic left, monochromatic right, color).

- CNN Models (Models to run with DepthAI, see section AI models below)
- Number of SHAVEs (number of vector processors to be used to compile the neural networks, higher number means faster network processing)
- OpenVINO (a free toolkit for converting AI models to the right format for the camera, different versions available)

Accuracy

The accuracy for the OAK-D-Lite camera has not been tested extensively. But the accuracy seems to be consistent with the 10 % error mentioned in the documentation for the camera (farther away from the OAK camera (or way too close), the 3D measurement gets more and more inaccurate). Which is similar to other a stereo cameras, with similar the resolution and distance between the stereo pair cameras. Instead, this document focuses on user cases the camera can be suited for and how it handles those situations. Worth pointing out is that stereo cameras work in the way that the same points in the images need to be seen AND identified/detected in both the left and right monochrome camera. Like uniformly colored surfaces can be problematic, as seen in Figure 3 below.

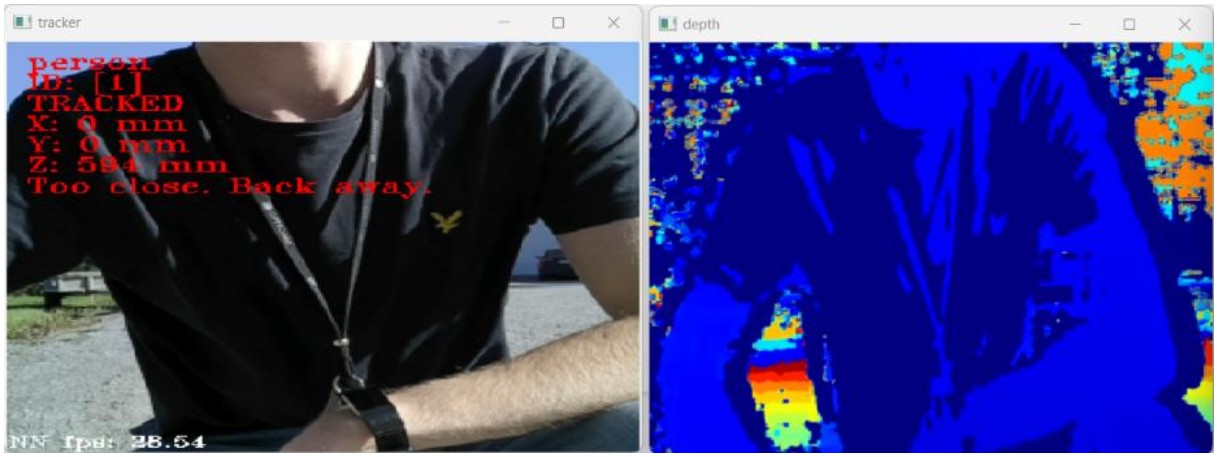

Figure 3: Color (left) and depth (right) images from the OAK-D-Lite camera, where shade makes parts of the person's clothes look uniformly black. Which will affect the depth measurement at those points, where those parts lack a depth value as seen in the image to the right where dark blue corresponds to empty points.

Sensor speed:

The sensor speed depends on different factors. Which camera stream are you using? Color or monochromatic, stereo depth or more than one at a time?

The color camera is limited to 30 fps, but only if the resolution isn't too high and the CNN model used is less computational heavy.

The monochrome cameras frame rate also depends on the resolution used for the images, as well as if a neural network model is applied to the images. Also, if stereo depth is calculated or not will affect the speed of the camera stream.

Which image stream to use depends on what the goal of the applications is. If the purpose is to use an AI model on a color camera only, then camera depth is not necessary to calculate, leaving more computational power at the applications disposal for other things.

Emitter usage

Unlike other variants of the OAK cameras, the D-Lite camera does not have an IR-emitter, meaning that it is completely dependent on the surrounding (sun, lamp, etc.) light to measure a scene.

Inertial measurement unit (IMU)

Early versions of the OAK-D-Lite camera does not have an Inertial measurement unit, but later versions of OAK-D-Lite do (the BMI270).

Price

You can get an OAK-D-Lite for about 149 USD. An OAK-D camera goes for about 249 USD, which is bigger, has higher power consumption, but also different specs (higher resolution for instance).

AI models

This section includes descriptions for the different default CNN models available with DepthAI. The confidence score for the identified objects with a model is useful for determining if the CNN model output is likely to be correct or not. This is important, since you will notice that the models can get things wrong at times. The XYZ-coordinates are also often included in the output when depth processing is activated. Sometimes the CNN model can track an object that fails to deliver accurate XYZ-coordinates from the 3D measurements, resulting in all coordinates getting the value 0.

Mobilinet-ssd

Tracks many different objects, including people, animals, furniture, vehicles and plants. The model is seen in action in Figure 4 below, for a few different objects. It also shows the area used within the bounding boxes of the tracked objects that are used for estimating the X-, Y- and Z-positions for the objects. This does affect the 3D position given in the example scripts when the bounding box centre is not at the objects, as seen in Figure 5 below, which needs to be handled differently in the python code to yield more accurate results in similar edge cases.

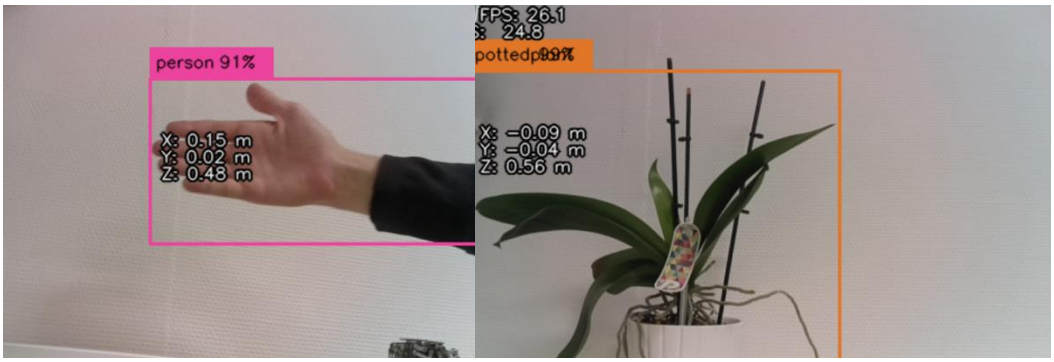

Figure 4: Showing (part of) frames from the OAK-D-Lite camera using the mobilinet-ssd CNN model, identifying certain objects. Left – Hand (of a person) is classified as a person. Right – A potted plant is classified as just that.

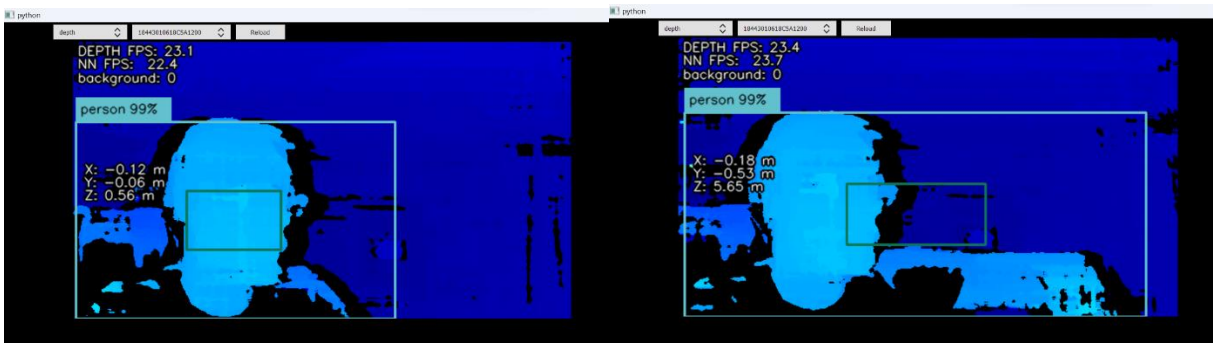

Figure 5: Depth frames from the OAK-D-Lite camera, using the mobilinet-ssd CNN model, identifying a person at about 0.6 m from the camera. The X-, Y- and Z-coordinates for the detected person are shown in text in the frames. Left – The person is at the centre of the bounding box, where the data points used for determining the 3D-coordinates of the tracked person are within the smaller bounding box, yielding a reasonable Z-coordinate value of 0.56 m. Right – The person, with the arm slightly stretched, results in a bounding box, where the data points used for determining the 3D-coordinates within the smaller bounding box include points mostly in the background, which causes the Z-coordinate to show an inaccurate value of 5.65 m.

The mobilinet-ssd model (or others) used for tracking objects can be run in an example script *spatial_object_tracker_with_depth_colour_alignment.py* from the depthai-python github. This script was adjusted to also use a distance limit, where the people in the color images are detected and highlighted with a bounding box that is green if the people are farther away from the camera baseline than 2 m, and red if the distance is closer, accompanied with a "Too close" text (see Figure 6 below). The depth images were also adjusted to visualize the people (but sometimes includes the area around them) in green or red depending on the distance from the camera, ignoring other points within the image (see Figure 7 below).

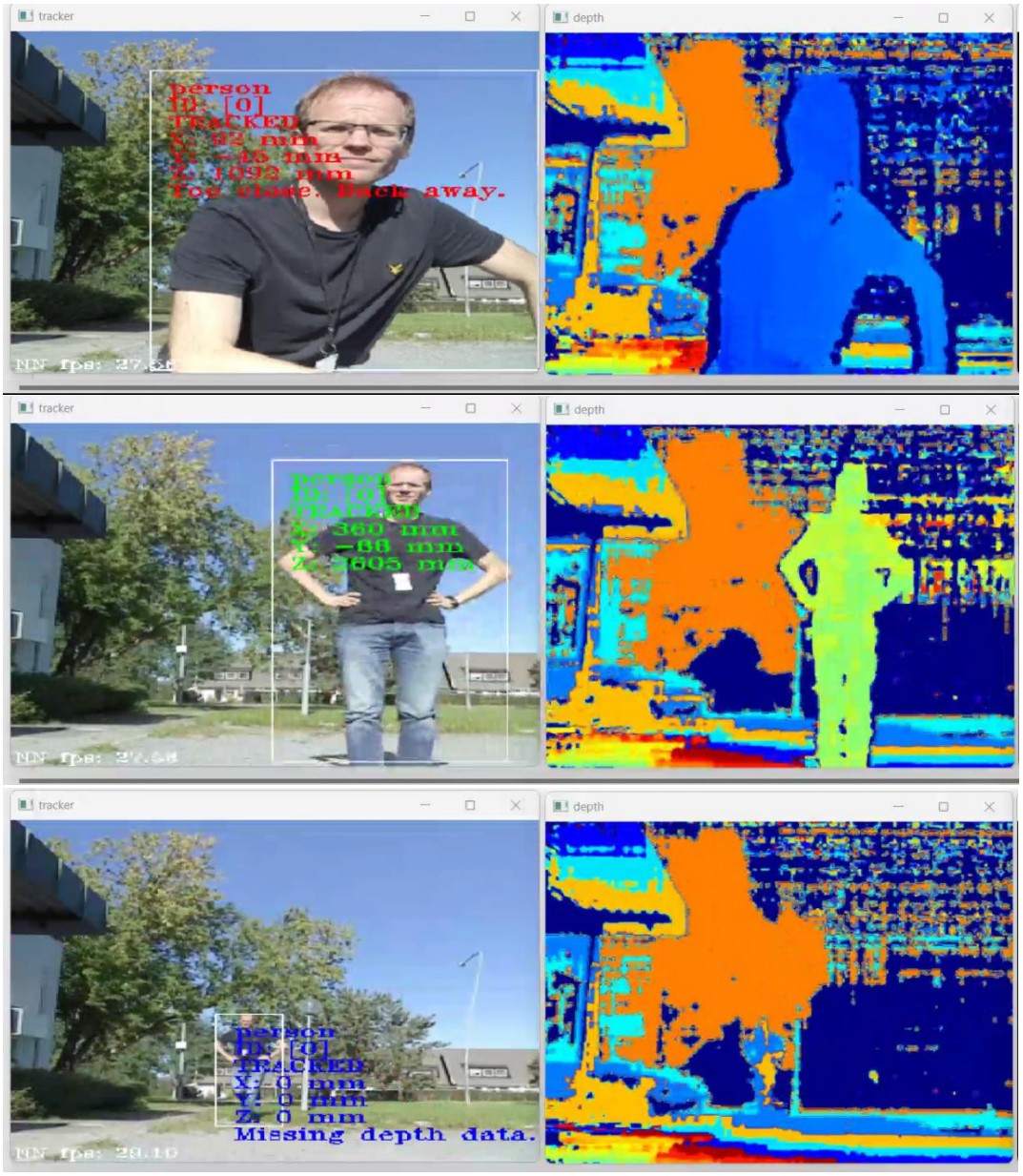

Figure 6: Images from the OAK-D-Lite camera, showing the results of an adjusted script: spatial_object_tracker_with_depth_colour_alignment.py from the depthai-python github. The tracking model used is the mobilinet-ssd, configured to only track people. Also added is the green color to the text when the person is farther away from the camera than 2 m (Z-direction, straight ahead from the camera). The text becomes red when the person is closer than that. The text "Too close. Back away." is also added. If the 3D-coordinates from the tracking of the AI model fails to use points in the depth images that correctly correspond to the tracked object (which can often happen if the person is too far away from the camera), the XYZ-coordinates will all be set to 0. In the bottom image, this can be seen, with the text displayed in blue.

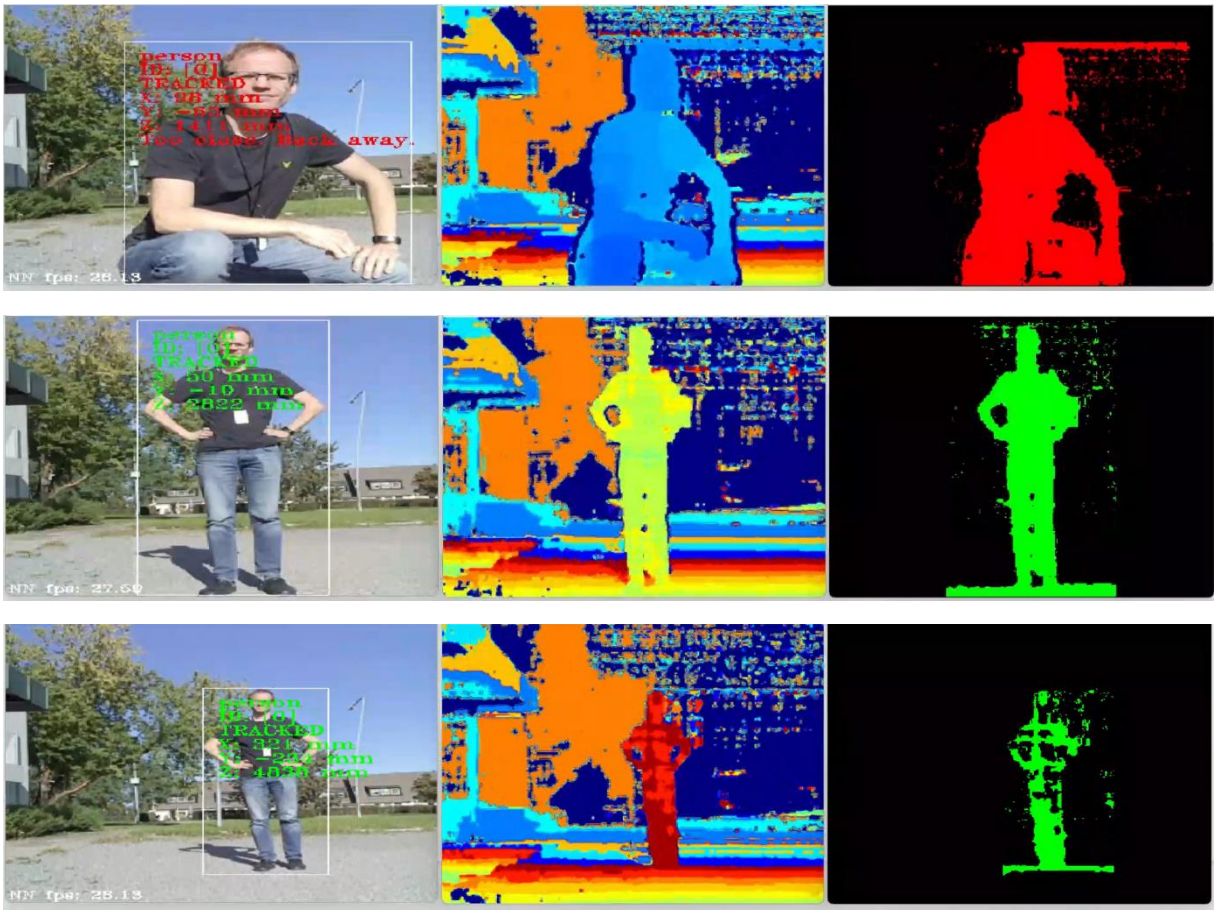

Figure 7: Images from the OAK-D-Lite camera, showing the results of an adjusted script: spatial_object_tracker_with_depth_colour_alignment.py from the depthai-python github. The tracking model used is the mobilinet-ssd, configured to only track people. The person is detected in the color image (the left images), but the corresponding area of the depth image (middle images) is used here for coloring the depth data at and around the detected person (the right images, the limit is 500 mm from the detection anchor Z-point of the person, returned by the AI model). Red means that the AI model returned a z-position closer than 2 m to the camera. When the person is moving farther away from the camera than 2 m, the data points at and around the detected person will be displayed in green. As seen in the bottom row we can see how the depth data gets more inaccurate farther away from the camera to the point that some parts of the detected person are excluded from the green-colored depth data to the right.

ADOPTICUM

Face-detections-adas

Classifies and tracks faces of people, even up to about 5 m from the camera, depending on the lighting conditions and resolution of the camera used (see Figure 8 below).

Figure 8: The CNN model face-detections-adas used for detecting humans faces, from 0.7 m up to about 5 m, before losing track of the detected face. The camera used is the OAK-D-Lite camera.

Human_pose_estimation_0001 and openpose2

These CNN models detect people and the “skeleton” of the people detected. The arms, legs, head, etc. are detected and tracked and illustrated in the frames from the camera. It takes a lot of processing time though, unlike for example Mobilinet-ssd, which affects how quick the position of the skeleton can keep up with the object location in the camera frames (see Figure 9 below), even with using more SHAVEs (12).

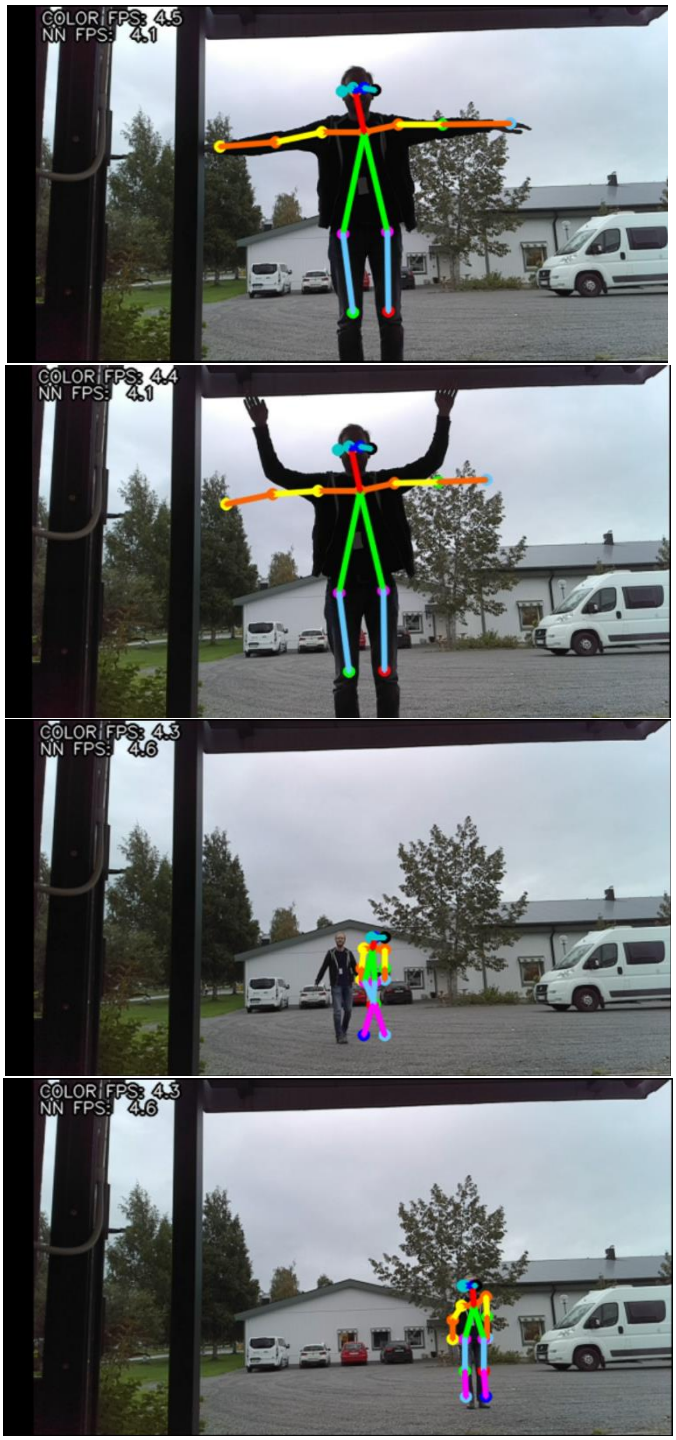

Figure 9: The CNN model human-pose-estimation used for detecting the positions of the human “skeleton”, from 2 m up to about 10 m. The process is intensive for the camera’s processing unit, and the skeleton positions have a hard time keeping up with the camera stream it uses to extract the skeleton position at higher frame rates (> 5 fps here), for the OAK-D-Lite camera.

deeplab_v3p_person

The depth measurements of people within the camera frames are extracted through the deeplab_v3p_person module (see Figure 10 below). The depth measurements synchronized with the color frame from the OAK-D-Lite can be used to extract measurement data of people in the scene only, removing the background. This does require coding of your own since the example code visually shows the output of the CNN model on the color image, but it is possible to extract the corresponding pixel values for the depth.

Figure 10: The deeplab_v3p_person module is used to mark the people in the images with the green colored blob, where the pixels of the images are marked where the person is detected, which can be used to extract depth pixel values at the objects, excluding points that do not include the person. The camera used is the OAK-D-Lite camera.

pedestrian_detections_adas and person_detection_retail

These models detect pedestrians in images. They are rather stable, but the tracking can be a bit behind when the objects move fast enough (see Figure 11 below), and it is needed to improve performance to perform better in that regard. Increase SHAVEs or reduce image resolution to try and improve the speed of the detection. Also, matching the frame rate of the camera with the speed of the Neural network performance can yield a better latency, where the CNN model has enough time to work on every frame. The models can track more than one person at a time.

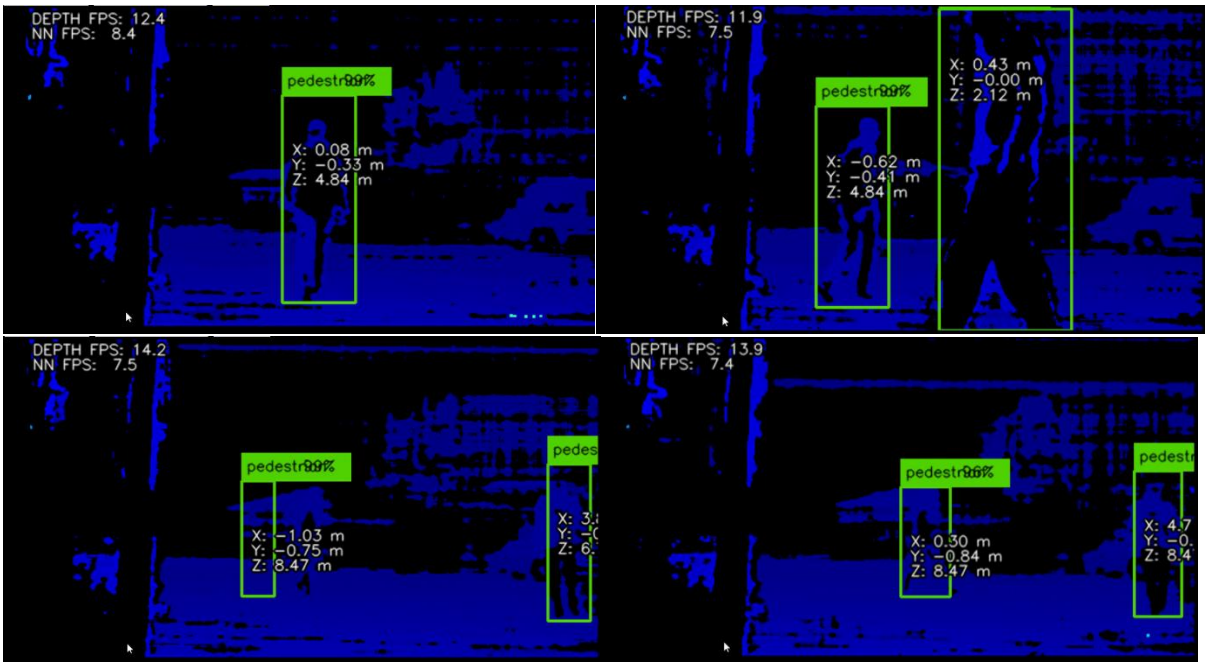

Figure 11: The Convolutional Neural Networks model pedestrian_detections_adas is tracking one or two people in this image during depth image streaming with the OAK-D-Lite camera. The model detected pedestrians in real time. In the lower left image, we can see the left person to the right of the pedestrian bounding box. This is because the depth image is updated faster than the CNN model can return an updated result for an image. In the lower right image, we see that the bounding box has caught up to the pedestrian position in the image.

person_vehicle_bike_detection_crossroads and vehicle_detection_adas

The person_vehicle_bike_Detection_crossroads and vehicle_detection_adas models find vehicles in images, with the former also detecting pedestrians. As can be seen in Figure 12 below, the objects are found within the images, with some lag for the tracking while streaming. The objects are detected even at about 20 m away, but the XYZ-coordinates yielded are either all 0 or otherwise incorrect at distances farther away than about 8 m. This means that work needs to be done in your own code if the objects that far away need to yield more accurate coordinates. The camera does also have its limitations when it comes to making accurate measurements at distances farther away.

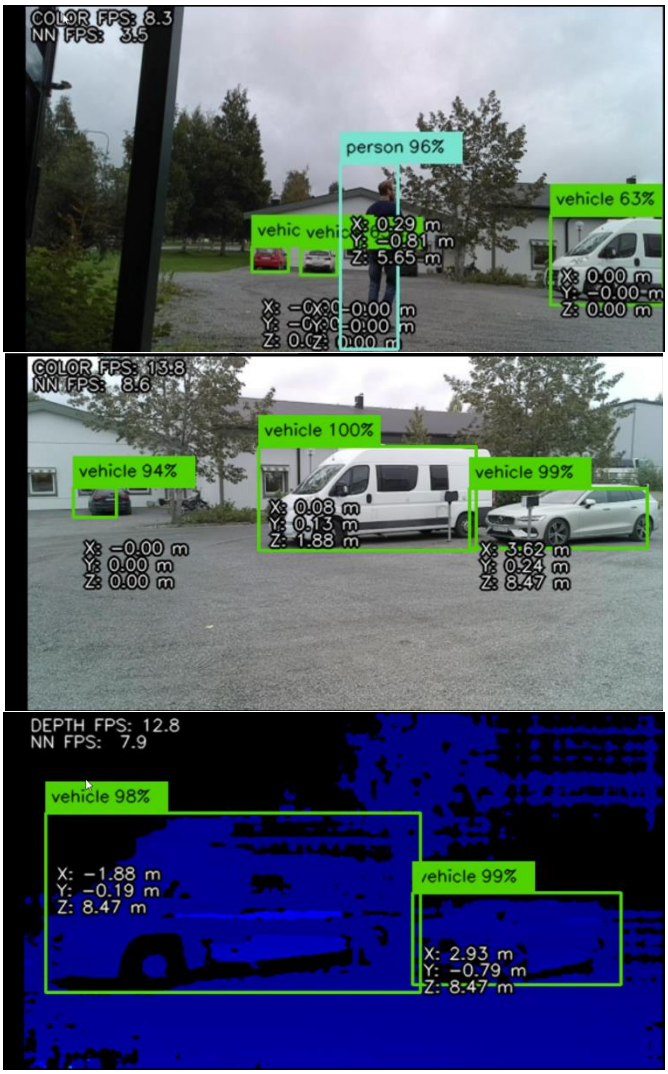

Figure 12: Top image: With the person_vehicle_bike_Detection_crossroads model, cars and one pedestrian are detected. Middle image: Vehicles detected in the image with the vehicle_detection_adas. Bottom image: Vehicles detected in the image with the vehicle_detection_adas, shown in the depth image. In all images there are objects where the XYZ-coordinates are either all 0.0 or otherwise incorrect for objects farther away from the camera. This happens about 8 m or farther from the OAK-D-Lite camera.

vehicle_license_plate_detection_barrier

The vehicle_license_plate_detection_barrier model detects not only license plates in the images, but also the vehicles themselves. During the testing of this model, the vehicle detection did not always work even when a vehicle was shown clearly in the image and close to the camera. Even more unreliable was the license plate detections, that were hard to get the camera to detect at all (see Figure 13 below).

Figure 13: The vehicle_license_plate_detection_barrier working on four different images from the OAK-D-Lite camera. The top two consecutive images from the same stream where the vehicle is not detected in one frame and then detected in the next. Unlike other vehicle detection algorithms, the detection was not very reliable, and the results often jumped back and forth even with images so close in view to each other as those two. The lower two show the same thing but for the license plate detection as well. The model often failed to detect the license plate and could often lose the object between frames.

road_segmentation_adas

The road_segmentation_adas AI model is detecting roads, shown by putting a green layer on the roads. As seen in Figure 14 below the road detection take some frames until the layer is positioned at the right position. The frame rate is rather low, 4.5 FPS for the neural network model to return a result, but it works well for asphalt roads.

Figure 14: The road_segmentation_adas AI model detecting an asphalt road and visualizing a green layer at the position and shape of the road. The two images show that it takes some frames for the road position to be updated. While holding the OAK-D-Lite camera in a static position the green layer will shape and cover the road, showing that the position of the road is detected.

Yolo_v3-tf and Yolo_v3_tiny_tf

Yolo is an object detection AI model that is used widely for training AI to detect objects within images, with focus on performance/speed. The yolo model available by default for the OAK camera can detect a wide variety of objects. For the OAK-D-Lite camera, the detections do affect the performance noticeably, but a way to increase the frame rate at which the model can operate on the camera stream, is to use the compressed version of the Yolo model, *yolo_v3_tiny_tf*. It uses a simpler network structure and reduces parameters, resulting in faster object detection. Speed seems to be prioritized over accuracy, since many objects are quickly labelled in the images, but often not correctly. See Figure 15 below.

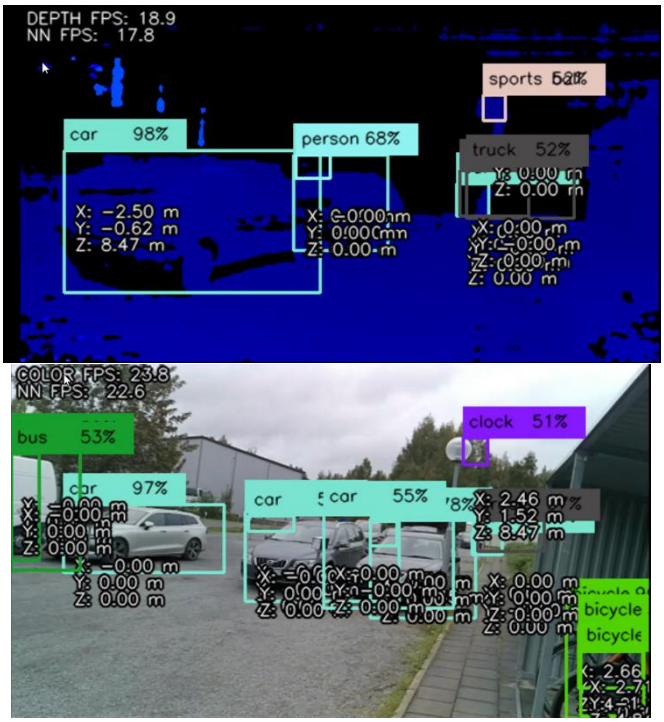

Figure 15: The yolo_v3_tiny_tf model detects the different objects and classifies them in the camera images. The top image shows the depth image and the bottom one shows the color image. The color image was used for detecting the objects in both cases. The vehicles like cars and bicycles can be detected rather well, but the model prioritizes speed above accuracy. In both images a lamppost can be seen detected as a clock or sports ball.

User cases

The camera can be used as a regular stereo camera, where the scenes of interest can be measured in 3D for both indoor and outdoor applications like:

- Automatic control of machines, robots and drones that need depth perception to navigate in the world.
- Measure distances to and shapes of objects for inspection and classification analysis or obstacle avoidance.
- Measure the complete dimensions of objects, even bigger ones, when cm to dm precision is enough, using several frames of depth data and combining them (needs additional development of your own).

With DepthAI the default models can be used for analysing 2D images and/or 3D data to identify, classify, measure and track all kinds of objects, like:

- People
- Vehicles
- Furniture
- Road segments
- Faces
- Animals
- Plants

You can even convert your own trained neural network models to a suitable format to run with the OAK-D-Lite. The user cases possible is up to your imagination.

ADOPTICUM

Summary

The OAK-D-Lite is a stereo camera and color camera with capacity to run neural network models on its own hardware, whether the default models or your own. It is thus a flexible sensor that is suitable for many different applications. The stereo camera pair can be combined with the rgb camera to get coordinated 3D and color data.

The camera has many different settings and a lot of functionalities through the DepthAI SDK. This gives the users many tools to work with, but also allows them to handle the image/3D data programmatically in their own code. There are tools and scripts for python to get the user started to familiarise themselves and learn how the SDK is used. This does mean that for someone who wants to use these cameras for their own applications, programming knowledge is needed to write your own code. The demo application is only going to get you so far.

Also, if you want to track objects not included in the default AI models available this means that you need to be able to either get other existing AI models or train the models yourself. Alternatively, you can also use traditional 2D and 3D processing tools for analysing the 2D and 3D images programmatically, either with the user's own code or with available software libraries (like OpenCv for expansive functionality in 2D image processing, which is also well documented and supported for python).

The sensor is relatively affordable at a price of 149 USD, which allows for many users to acquire one and test the sensor themselves.

The accuracy of the camera for 3D measurements is at best on several cm level (gets worse with increasing distance), which can inform your choice of objects to measure. Also, the object detection accuracy with the neural network models depends on the model used and won't be 100 % accurate all the time for any model, which is important to know when using the camera for object detection purposes. Testing your specific application thoroughly is always important.

The combination of color and depth perception for the OAK-D-Lite camera allows for 3D measurement and navigation, as well as identification of and tracking objects in the 2D images AND 3D data. And the same sensor can be used for different tasks, even at the same time as long as the processing power is enough for the specific application.