OpenDR —
Open Deep Learning Toolkit for Robotics

Project Start Date: 01.01.2020
Duration: 48 months
Lead contractor: Aristotle University of Thessaloniki

Deliverable D7.2: First public version of the OpenDR toolkit

Date of delivery: 31 Dec 2021

Contributing Partners: AUTH, TAU, AU, TUD, ALU-FR, CYB, PAL

Version: v5.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449.

Title	**D7.2: First public version of the OpenDR toolkit**
Project	**OpenDR** (ICT-10-2019-2020 RIA)
Nature	Open Research Data Pilot
Dissemination Level:	**PU**blic
Authors	Anastasios Tefas (AUTH), Nikos Nikolaidis (AUTH), Nikolaos Passalis (AUTH), Pavlos Tosidis (AUTH), Paraskevi Nousi (AUTH), Charalampos Symeonidis (AUTH), Efstratios Kakaletsis (AUTH), Gizem Bozdemir (PAL), Thomas Peyrucain (PAL), Dias Daniel (CYB), Halil Ibrahim Ugurlu (AU), Illia Oleksiienko (AU), Lukas Hedegaard Morsing (AU), Daniel Honerkamp (ALU-FR), Niclas Vödisch (ALU-FR), Jenni Raitoharju (TAU), Anton Muravev (TAU), Dat Thanh Tran (TAU), Jelle Luijkx (TUD), Bas van der Heijden (TUD)
Lead Beneficiary	PAL (PAL Robotics)
WP	7
Doc ID:	OPENDR_D7.2.pdf

Document History

Version	**Date**	**Reason of change**
v1.0	1/10/2021	Deliverable structure template ready
v2.0	17/12/2021	Contributions from partners finalized
v3.0	20/12/2021	Final draft ready for internal review
v4.0	24/12/2021	Revised version including internal reviewer's comments
v5.0	28/12/2021	Final version

Contents

Executive Summary

This document aims at supplementing the first public version of OpenDR toolkit released at M24. It provides details about accessing, downloading and using the toolkit.

1 Introduction

OpenDR aims at developing an open, non-proprietary modular toolkit that can be easily used by robotics companies and research institutions to efficiently develop, evaluate and deploy AI and cognition technologies to robotics applications. At a high level, OpenDR contains a selection of cognition and perception algorithms, along with general-purpose functionalities that are necessary for common robotics tasks. This technical report (Deliverable D7.2) aims at supplementing the first public version of OpenDR toolkit released at M24. It provides details about accessing, downloading and using the toolkit.

2 Accessing the OpenDR toolkit

The toolkit is developed using the well-established GitHub platform, following robust development methodologies, including continuous integration and strict code review guidelines, as described in D2.1 and D7.1. The most recent version of the toolkit can be accessed at:

`https://github.com/opendr-eu/opendr`

The `master` branch will contain the latest stable version of the toolkit and the `develop` branch a version that includes the latest additions, refactors and module upgrades. Although CI tests will maintain stability and high quality in this experimental version, it is likely to change often so it is less adapted for daily usage. With every milestone of the project, content from the `develop` branch will be merged into the `master` branch, from which a new release candidate will be prepared, which after sufficient testing by the partners will be publicly announced. Although it is always possible to access the latest state of the toolkit from the Github page, in between these larger milestones shortcomings and bugs are an inevitability that needs to be addressed, therefore nightly (not necessarily literal, the frequency has not been decided yet) builds of the packages based on the `master` and `develop` branch will be automatically generated to ease their use.

OpenDR provides an intuitive and easy-to-use **Python interface**, a **C API** for selected tools, **a wealth of usage examples and supporting tools**, as well as **ready-to-use ROS nodes**. OpenDR is built to support Webots Open Source Robot Simulator, while it also extensively follows industry standards, such as ONNX model format and OpenAI Gym Interface. Detailed documentation can be found in OpenDR wiki, that accompanies the GitHub repository.

3 Installing and using the OpenDR toolkit

To maximize the visibility and ease-of-use of the toolkit, we provide three different ways for installing the toolkit:

1. By cloning the GitHub repository
2. Using *pip*
3. Using *docker*

The first way provides a fully functional version of the toolkit that can be installed in various platforms. *pip* is a straightforward way to install and experiment with the Python API of the toolkit, while docker images are provided to experiment with toolkit functionalities in a pre-configured environment with very little effort, as well as for other containerized applications.

The following subsection provides an overview of the installation process. OpenDR is designed to be easy-to-use and install in order to maximize its impact. To this end, installation scripts have been prepared to ensure that this process will be very easy, even for novice users. Up-to-date instructions and additional details are available on OpenDR's GitHub repository.

3.1 Installation by cloning the GitHub repository

3.1.1 Installation procedure

To install the toolkit on a Linux system (Ubuntu 20.04 is currently supported), please first make sure that *git* is available on the system:

```
sudo apt install git
```

Then, the toolkit should be downloaded locally:

```
git clone --depth 1 --recurse-submodules -j8  \
    https://github.com/opendr-eu/opendr
```

To install the toolkit an installation script is available:

```
cd opendr
./bin/install.sh
```

The installation script automatically installs all the required dependencies. Note that we can set the training/inference device using the `OPENDR_DEVICE` variable. The toolkit defaults to using CPU. If we want to use GPU, we can set this variable accordingly before running the installation script:

```
export OPENDR_DEVICE=gpu
```

The installation script creates a *virtualenv*, where the toolkit is installed. OpenDR environment can be activated similar to any other *virtualenv*:

```
source ./bin/activate.sh
```

All functionality (e.g., ROS, simulators, tools, demos, etc.) are then readily available.

3.1.2 Demo

For example, in order to run the *human gesture recognition* demo you can:

```
cd projects/perception/multimodal_human_centric
python3 gesture_recognition_demo.py \
    -input_rgb input_rgb.png -input_depth input_depth.png
```

where the two images mentioned are shown in Figure 1 (also available in the demo's folder). The result of the demo should be *Punch with confidence 0.606*.

Figure 1: (left) RGB input image and (right) depth input input image

3.2 Installation using *pip*

To increase the visibility of the toolkit, a PyPI package has been prepared.

3.2.1 Installation procedure

Installing the Python-API of the toolkit using this package is easy:

```
export DISABLE_BCOLZ_AVX2=true
sudo apt install python3.8-venv libfreetype6-dev git build-essential cmake \
    python3-dev wget
python3 -m venv venv
source venv/bin/activate
wget https://raw.githubusercontent.com/opendr-eu/opendr/master/dependencies/\
    pip_requirements.txt
cat pip_requirements.txt | xargs -n 1 -L 1 pip install
pip install opendr-toolkit
```

3.2.2 Demo

Assuming you have followed the procedure mentioned above the virtual environment should be active already. If it is not the case, you can activate it with:

```
source venv/bin/activate
```

Now, for example, let's retrieve the *semantic segmentation* demo from OpenDR repository:

```
wget https://raw.githubusercontent.com/opendr-eu/opendr/master/projects/\
    perception/semantic_segmentation/bisenet/inference_demo.py
```

When running `python3 inference_demo.py`, the model and test image should be downloaded and the result of the segmentation should be similar to what shown in Figure 2.

3.3 *docker* support

3.3.1 Procedure

Appropriate dockerfiles that can run on any Linux system have been prepared and docker images are publicly available on dockerhub. First, docker needs to be installed in your system. For

Figure 2: Result of running the semantic segmentation demo

Ubuntu you can follow this procedure. When installed, running the OpenDR docker image is very easy. For example, for the CPU image all you need is to execute:

```
sudo docker run -p 8888:8888 opendr/opendr-toolkit:cpu_latest
```

or for the cuda-enabled one:

```
sudo docker run --gpus all -p 8888:8888 opendr/opendr-toolkit:cuda_latest
```

Either command will pull the image and launch it, and in both cases a *Jupyter* notebook server that listens at port 8888 will run, which can be accessed by clicking on the link similar to `http://127.0.0.1:8888/?token=TOKEN` that appears in the console. Alternatively you can run an interactive session with:

```
sudo docker run -it opendr/opendr-toolkit:cpu_latest /bin/bash
```

or

```
sudo docker run --gpus all -it opendr/opendr-toolkit:cuda_latest /bin/bash
```

respectively for a cpu or cuda session. However, if you start an interactive session do not forget to enable the `venv` with the command:

```
source bin/activate.sh
```

3.3.2 Demo

For example, in order to run the *pose estimation demo* you must first pull the latest image from `dockerhub` and run it:

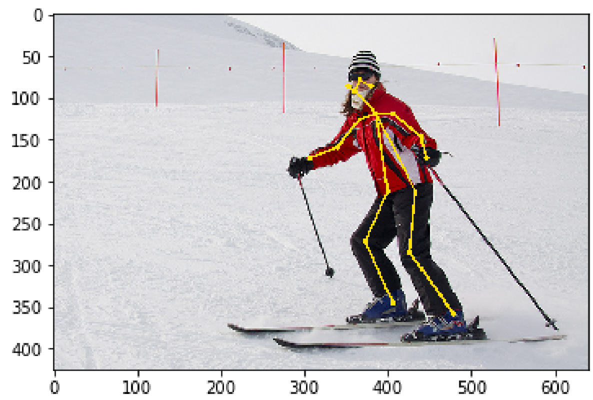

Figure 3: Result of running the pose estimation demo

```
sudo docker run -p 8888:8888 opendr/opendr-toolkit:cpu_latest
```

Now open the session in your browser, by clicking the link that appeared in the console, it should be similar to http://127.0.0.1:8888/?token=TOKEN.

Navigate to `projects/perception/lightweight_open_pose/demos/` and select the notebook `inference_tutorial.ipynb`. If you are using a CPU based docker, change the device to cpu in the second cell and execute the steps one by one. What you should expect at the end is shown in Figure 3.

To stop the Jupyter session you need to manually quit it.

3.4 Using OpenDR toolkit

OpenDR provides an extensive wiki available at:

https://github.com/opendr-eu/opendr/wiki

This wiki walks the user through all the available functionality providing all the necessary information to allow for directly using the toolkit. This includes: a) detailed documentation regarding all functionalities provided in the toolkit, among with usage examples, b) demos that showcase the use of the toolkit, c) ready to use ROS nodes for integrating OpenDR with robotics applications with minimal effort, and d) example use-case applications for some of the tools. Figures 4, 5 6, 7 and 8 demonstrates a few examples from OpenDR GitHub repository.

4 Conclusions

This document presented the work performed in WP7 about toolkit integration resulting in a first public version of the OpenDR toolkit.

activity_recognition module

The *activity_recognition* module contains the *X3DLearner* and *CoX3DLearner* classes, which inherit from the abstract class *Learner*.

Class X3DLearner

Bases: `engine.learners.Learner`

⋮

The X3DLearner class has the following public methods:

`X3DLearner` constructor

```
X3DLearner(self, lr, iters, batch_size, optimizer, lr_schedule, backbone, network_head,
```

Constructor parameters:

- **lr**: *float, default=1e-3* Learning rate during optimization.
- **iters**: *int, default=10* Number of epochs to train for.
- **batch_size**: *int, default=64* Dataloader batch size. Defaults to 64.
- **optimizer**: *str, default="adam"* Name of optimizer to use ("sgd" or "adam").
- **lr_schedule**: *str, default=""* Unused parameter.
- **network_head**: *str, default="classification"* Head of network (only "classification" is currently available).
- **checkpoint_after_iter**: *int, default=0* Unused parameter.
- **checkpoint_load_iter**: *int, default=0* Unused parameter.

Figure 4: Excerpts from OpenDR documentation

- **Inference and result drawing example on a test .jpg image using OpenCV.**

```
import cv2
from opendr.perception.pose_estimation import LightweightOpenPoseLearner
from opendr.perception.pose_estimation import draw, get_bbox
from opendr.engine.data import Image

pose_estimator = LightweightOpenPoseLearner(device="cuda", temp_path='./parent_dir
pose_estimator.download()  # Download the default pretrained mobilenet model in the
pose_estimator.load("./parent_dir/mobilenet_openpose")
pose_estimator.download(mode="test_data")  # Download a test data taken from COCO20

img = Image.open('./parent_dir/dataset/image/000000000785.jpg')
orig_img = img.opencv()  # Keep original image
current_poses = pose_estimator.infer(img)
img_opencv = img.opencv()
for pose in current_poses:
    draw(img_opencv, pose)
img_opencv = cv2.addWeighted(orig_img, 0.6, img_opencv, 0.4, 0)
cv2.imshow('Result', img_opencv)
cv2.waitKey(0)
```

Figure 5: Usage example from the documentation for an OpenDR tool

Pose Estimation Tutorial

This notebook provides a tutorial for running inference on a static image in order to predict keypoint locations of joints on a human, as well as draw the resulting pose.

First, we need to load our model. In this tutorial we are using the OpenPose estimator from OpenDR:

In [1]:
```
from opendr.perception.pose_estimation import LightweightOpenPoseLearner
```

We need to create our estimator:

In [2]:
```
 pose_estimator = LightweightOpenPoseLearner(device='cuda', num_refinement_stages=2
```

Note that we can alter the device (e.g., 'cpu', 'cuda', etc.), on which the model runs, the number of refinement stages that we are using, as well as a number of additional parameters that can make our model either faster or more accurate.

After creating our model, we need to download and load the pre-trained weights:

In [3]:
```
pose_estimator.download(path=".", verbose=True)
pose_estimator.load("openpose_default")
```

Figure 6: A Jupyter notebook prepared to showcase the usage of an OpenDR tool

4. Build the packages inside workspace

```
catkin_make
```

5. Source the workspace and you are ready to go!

```
source devel/setup.bash
```

Structure

Currently, apart from tools, opendr_ws contains the following ROS nodes:

Perception

1. Pose Estimation
2. 2D Object Detection
3. Face Detection
4. Panoptic Segmentation
5. Face Recognition
6. Semantic Segmentation
7. RGBD Hand Gesture Recognition
8. Heart Anomaly Detection
9. Video Human Activity Recognition
10. Landmark-based Facial Expression Recognition
11. Skeleton-based Human Action Recognition

Figure 7: Index of some of the ROS nodes available in the first version of OpenDR toolkit

Running the example

Human Activity Recognition using X3D

```
python demo.py --ip 0.0.0.0 --port 8000 --algorithm x3d --model xs
```

Human Activity Recognition using CoX3D

```
python demo.py --ip 0.0.0.0 --port 8000 --algorithm cox3d --model s
```

If you navigate your piano and http://0.0.0.0:8000 and pick up a ukulele, you might see something like this:

Figure 8: A showcase of OpenDR toolkit functionalities (from OpenDR demo projects)