

MYD-LD25X X-Linux-AI

Application Note



| | | |
|--|-----------------|----------------------------------|
| File status: [] Draft [√] Release | FILE ID: | MYIR-MYD-LD25X-SW-AN02-EN-6.1.82 |
| | VERSION: | V1.0[Doc] |
| | AUTHOR: | MSW0192 |
| | RELEASE: | 2024-09-25 |
| | UPDATED: | 2024-09-24 |

Revision History

| VERSION | AUTHOR | PARTICIPANT | DATE | DESCRIPTION |
|-----------|---------|-------------|------------|------------------|
| V1.0[Doc] | MSW0192 | MSW0041 | 2024-09-23 | Official Release |

CONTENT

| | |
|---|--------|
| Revision History | - 2 - |
| CONTENT | - 3 - |
| 1. X-Linux-AI Overview | - 4 - |
| 1.1. Hardware Resources | - 4 - |
| 1.2. Software Resources | - 4 - |
| 2. Installing X-Linux-AI on the Development Board | - 5 - |
| 2.1. Preparing the Configuration Environment | - 5 - |
| 2.2. Installing x-linux-ai-tool | - 9 - |
| 2.3. Installing the X-LINUX-AI Demo Package | - 9 - |
| 2.4. Running the Demo Launcher | - 9 - |
| 3. Running AI Application Examples | - 12 - |
| 3.1. Image Classification | - 12 - |
| 3.2. Object Detection | - 15 - |
| 3.3. Pose Estimation | - 18 - |
| 3.4. Semantic Segmentation | - 21 - |
| 4. References | - 24 - |
| Appendix A | - 25 - |

1. X-Linux-AI Overview

X-LINUX-AI is an STM32 MPU OpenSTLinux Expansion Package that targets artificial intelligence for STM32MP1 and STM32MP2 series microprocessors. It contains Linux® AI frameworks, as well as application examples to get started with some basic use cases.

The examples provided in X-LINUX-AI include a selection of optimized models for image classification, object detection, semantic segmentation, and human pose estimation. The face recognition application provided in X-LINUX-AI as a prebuilt binary is based on models retrained by STMicroelectronics. Contact the local STMicroelectronics support for more information about this application.

These examples rely on the STAI_MPU API based on either the TensorFlow™ Lite inference engine, ONNX Runtime, OpenVX™, or the Google Edge TPU™ accelerator. They all support Python™ scripting and C/C++ applications.

1.1. Hardware Resources

- MYD-LD25X development board with the pre-flashed MYiR release image
- MY-LVDS070C screen or any HDMI interface display
- MY-CAM003M MIPI-CSI camera module

1.2. Software Resources

All operations in this article are executed via the debug serial port of the MYD-LD25X development board. Please ensure that you have completed the basic content in the MYD-LD25X Quick Start Guide (the booklet included with the development board) and that the development board can connect to the internet. There are various ways to provide internet access, such as connecting to an internet-enabled router or using Wi-Fi. For specific Wi-Fi connection methods, please refer to the "MYD-LD25X Linux Software Evaluation Guide," particularly the STA connection section.

2. Installing X-Linux-AI on the Development Board

This chapter mainly introduces how to install X-Linux-AI and related demo components on the MYD-LD25X development board.

2.1. Preparing the Configuration Environment

1) Obtaining Calibration Parameters

If you are using an LVDS screen with the MYD-LD25X, the screen needs to be calibrated when entering Weston for the first time. The automated calibration script operation is added by default in the autorun script. If you have already performed the calibration or are using an HDMI screen, you can skip this subsection.

Run the autorun.sh script to perform the calibration operation:

```
# autorun.sh
```

After execution, touch points will appear on the LVDS screen. Click on them to complete the calibration, and you will not need to calibrate the screen again.

2) Closing HMI

To avoid display conflicts, after logging into the MYD-LD25X, execute the following command to close the mxapp2 program, which is MYiR's HMI interface:

```
# killall mxapp2
```

Also, comment out the line that starts mxapp2 in the autorun script so that it won't run automatically next time:

```
# vi /usr/bin/autorun.sh
```

```
#!/bin/sh
```

```
...(omitted)
sync
#!/usr/sbin/mxapp2 &
```

3) Updating Software Sources

Execute the following command to update the software sources:

```
# apt update
```

The software package is provided AS IS, and by downloading it, you agree to be bound to the terms of the software license agreement (SLA).

The detailed content licenses can be found at https://wiki.st.com/stm32mpu/wiki/OpenSTLinux_licenses.

```
Get:1 http://packages.openstlinux.st.com/5.1 mickledore InRelease [5,723 B]
```

```
Get:2 http://packages.openstlinux.st.com/5.1 mickledore/main arm64 Packages [725 kB]
```

```
Get:3 http://packages.openstlinux.st.com/5.1 mickledore/updates arm64 Packages [38.2 kB]
```

```
Get:4 http://packages.openstlinux.st.com/5.1 mickledore/untested arm64 Packages [1,338 kB]
```

```
Fetched 2,107 kB in 3s (690 kB/s)
```

```
Reading package lists... Done
```

```
Building dependency tree... Done
```

```
1 package can be upgraded. Run 'apt list --upgradable' to see it.
```

Updating the source requires the MYD-LD25X to be connected to the internet, so please ensure network connectivity. When executing the above command, the following issues may occur, leading to update errors:

- **Time Synchronization Issues**

E: Release file for <http://packages.openstlinux.st.com/5.1/dists/mickledore/InRelease> is not valid yet (invalid for another **1383d 8h 14min 14s**). Updates for this repository will not be applied.

E: Release file for <http://extra.packages.openstlinux.st.com/AI/5.1/dists/mickledore/InRelease> is not valid yet (invalid for another **1381d 8h 10min 47s**). Updates for this repository will not be applied.

The reason for this issue is that the current development board's time has not successfully synchronized with the network time. Execute the following operations to sync the network time. First, modify the timesyncd.conf configuration file by adding a FallbackNTP time server website as shown below:

```
# vi /etc/systemd/timesyncd.conf
...
[Time]
#NTP=
FallbackNTP=time1.google.com time2.google.com time3.google.com time4.google.com
...
```

Then enter the following command to restart the time synchronization service:

```
# systemctl restart systemd-timesyncd
```

Enter date again to check if the time has been successfully updated. Depending on network conditions, it may take some time to synchronize:

```
# date
2023 年 03 月 03 日 星期五 17:50:37 CST
# date
2024 年 09 月 20 日 星期五 15:45:15 CST
```

● DNS Issues

After running apt update, the following issues may occur:

```
# apt update
...
Err:1 http://packages.openstlinux.st.com/5.1 mickledore InRelease
Temporary failure resolving 'packages.openstlinux.st.com'
Reading package lists... Done
Building dependency tree... Done
All packages are up to date.
```

- / -

```
W: Failed to fetch http://packages.openstlinux.st.com/5.1/dists/mickledore/InRelease
Temporary failure resolving 'packages.openstlinux.st.com'
W: Some index files failed to download. They have been ignored, or old ones used instead.
```

This issue requires modifying /etc/resolv.conf by adding the following content to the file:

```
# vi /etc/resolv.conf
...
nameserver 8.8.8.8
nameserver 8.8.4.4
```


2.2. Installing x-linux-ai-tool

Once the environment configuration is complete, enter the following command to install x-linux-ai-tool:

```
# apt-get install -y x-linux-ai-tool
```

After installation, enter the following command to confirm the installation is complete:

```
# x-linux-ai -v  
X-LINUX-AI version: v5.1.0
```

For information on using the x-linux-ai tool, you can view it with x-linux-ai -h or browse the official wiki:

[X-LINUX-AI Tool - stm32mpu](#)

2.3. Installing the X-LINUX-AI Demo Package

To start using X-Linux-AI, you need to install the X-Linux-AI demo package, which provides all AI frameworks, application examples, tools, and utilities optimized for the specific target used:

```
# x-linux-ai -i packagegroup-x-linux-ai-demo
```

2.4. Running the Demo Launcher

The official ST demo launcher has been removed by default on the MYD-LD25X. To facilitate the use of X-Linux-AI demo applications, it needs to be re-added along with the official demo display program that starts automatically with Weston.

Enter the /usr/local/weston-start-at-startup directory and create a new script named start_up_demo_launcher.sh, adding the corresponding content to the script as follows:

```
# cd /usr/local/weston-start-at-startup
```

- 9 -

```
# vi start_up_demo_launcher.sh

#!/bin/sh
DEFAULT_DEMO_APPLICATION_GTK=/usr/local/demo/launch-demo-gtk.sh
if [ -e /etc/default/demo-launcher ]; then
    source /etc/default/demo-launcher
    if [ ! -z "$DEFAULT_DEMO_APPLICATION" ]; then
        $DEFAULT_DEMO_APPLICATION
    else
        $DEFAULT_DEMO_APPLICATION_GTK
    fi
else
    $DEFAULT_DEMO_APPLICATION_GTK
fi
```

Then, add run permissions to the script:

```
# chmod a+x start_up_demo_launcher.sh
```

Finally, after restarting the Weston service, you should see the launcher running successfully:

```
# systemctl restart weston-graphical-session.service
```

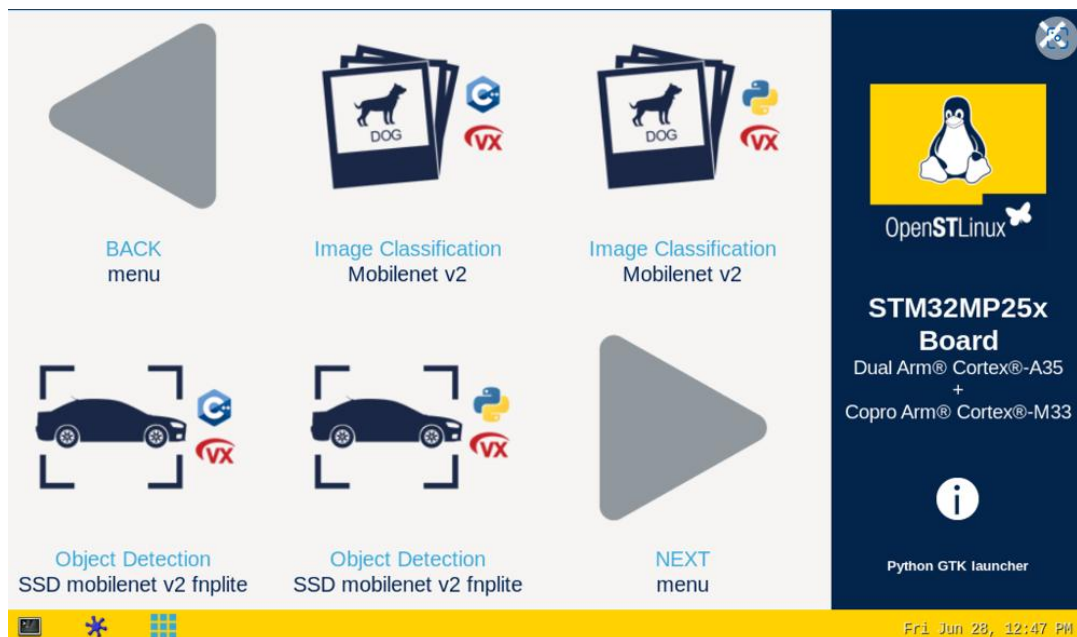


Figure 2-1. Demo Launcher - 1

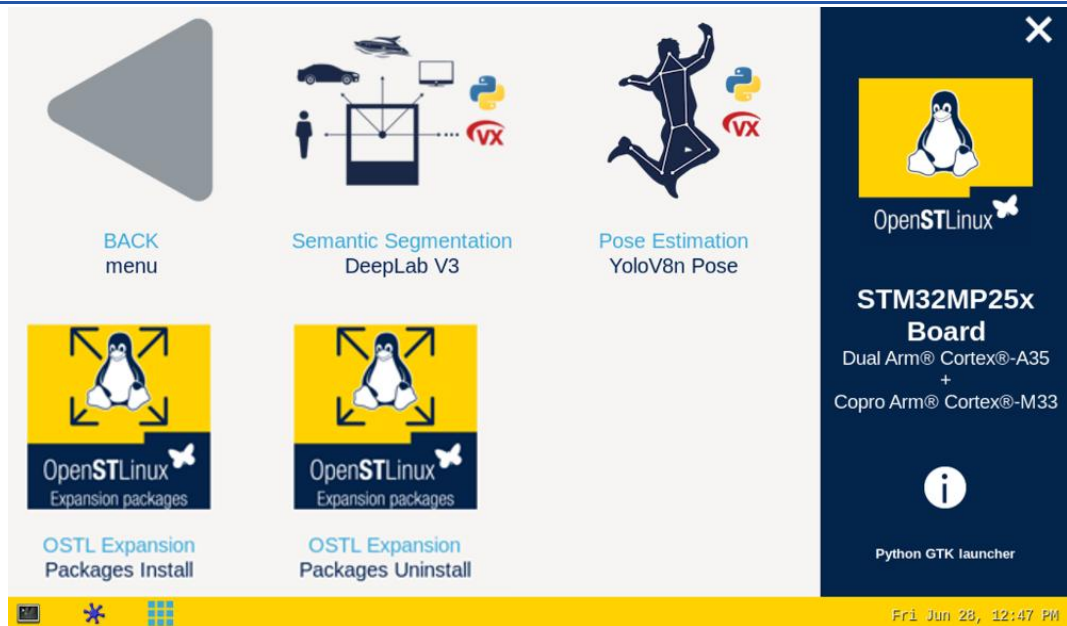


Figure 2-2. Demo Launcher - 2

3. Running AI Application Examples

This section will briefly demonstrate several demos after installation. Before executing the demos, ensure that the hardware resources from section 1.1 are prepared.

All demos require a MIPI-CSI camera. The camera model suitable for the MYD-LD25X development board is the MIL's MY-CAM003M. Before using this camera, initialize the configuration and preview it using the following command to ensure the camera operates correctly.

Enter the `/etc/myir_test` directory and run the `myir_camera_play` script:

```
# cd /etc/myir_test
# ./myir_camera_play
```

After execution, a camera preview will appear on the screen. Please ensure that the display is normal. If there are issues, refer to the section on MIPI-CSI cameras in the "MYD-LD25X Linux Software Evaluation Guide" to check the camera connection and troubleshoot. For further assistance, please consult MIL technical support.

3.1. Image Classification

The image classification neural network model can recognize the objects represented in images. It classifies images into different categories. This application demonstrates a computer vision use case for image classification, capturing frames from camera input (`/dev/videoX`) and analyzing them through neural network models parsed by OpenVX, TFLite, or ONNX frameworks.

3.1.1. Running via Demo Launcher

You can run this demo by clicking the Image Classification icon in the demo launcher. By default, the application installed in section 2.3 is OpenVX, available in both C/C++ and Python.



Figure 3-1. Image Classification

3.1.2. Running via Command

The C/C++ and Python applications for image classification are located in the `/usr/local/x-linux-ai/image-classification/` directory. You can get more help information by running the program with the `-h` parameter:

```
# cd /usr/local/x-linux-ai/image-classification/
# ls -la
stai_mpu_image_classification          # C++ executable
stai_mpu_image_classification.py      # Python executable
```

To simplify the demo startup, there are pre-configured launch scripts in the application directory:

- Using camera input to launch image classification demo

```
launch_bin_image_classification.sh    # C++ executable
launch_python_image_classification.sh # Python executable
```

- Using image input to launch image classification demo

```
launch_bin_image_classification_testdata.sh    # C++ executable
launch_python_image_classification_testdata.sh # Python executable
```

3.1.3. Demo Display

Starting the demo from the launcher defaults to using camera input, and the results will be consistent with the script used in command startup. Taking the C/C++ program as an example, the specific test is as follows:

```
# cd /usr/local/x-linux-ai/image-classification
# ./launch_bin_image_classification.sh    # or click the demo launcher icon
```



Figure 3-2. Camera Input

Before running the script for image input, prepare an image to be recognized. For example, place a teddy bear image in the directory /usr/local/x-linux-ai/image-classification/models/mobilenet/testdata, and then run the script, taking the C/C++ application as an example:

```
# cd /usr/local/x-linux-ai/image-classification/models/mobilenet/testdata
# ls -la
-rwxr--r-- 1 root root 102821  9 20 23:14 teddy.jpg
# cd /usr/local/x-linux-ai/image-classification
# ./launch_bin_image_classification_testdata.sh
```

The running results are as follows:



Figure 3-3. Image Input
- 14 -

3.2. Object Detection

This application demonstrates a computer vision use case for object detection, capturing frames from camera input (/dev/videoX) and analyzing them through neural network models parsed by OpenVX, TFLite, or ONNX frameworks. It uses a GStreamer pipeline to stream camera frames (using v4l2src), display previews (using gtkwaylandsink), and execute neural network inference (using appsink).

3.2.1. Running via Demo Launcher

You can run this demo by clicking the Object Detection icon in the demo launcher. By default, the application installed in section 2.3 is OpenVX, available in both C/C++ and Python.

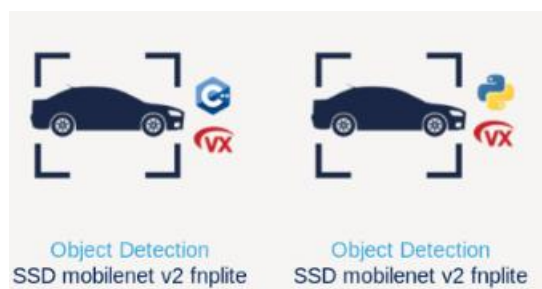


Figure 3-4. Object Detection

3.2.2. Running via Command

The C/C++ and Python applications for object detection are located in the `/usr/local/x-linux-ai/object-detection/` directory. You can get more help information by running the program with the `-h` parameter:

```
# cd /usr/local/x-linux-ai/object-detection/
# ls -la
stai_mpu_object_detection           # C++ executable
stai_mpu_object_detection.py        # Python executable
```

To simplify the demo startup, there are pre-configured launch scripts in the application directory:

- Using camera input to launch object detection demo


```
launch_bin_object_detection.sh          # C++ executable
launch_python_object_detection.sh       # Python executable
```

- Using image input to launch object detection demo

```
launch_bin_object_detection_testdata.sh  # C++ executable
launch_python_object_detection_testdata.sh # Python executable
```

3.2.3. Demo Display

Starting the demo from the launcher defaults to using camera input, and the results will be consistent with the script used in command startup. Taking the C/C++ program as an example, the specific test is as follows:

```
# cd /usr/local/x-linux-ai/object-detection
# ./launch_bin_object_detection.sh      # or click the demo launcher icon
```



Figure 3-5. Camera Input

Before running the script for image input, prepare an image to be recognized. For example, place an image of a teddy bear and a kitten in the directory:

```
/usr/local/x-linux-ai/object-detection/models/coco_ssd_mobilenet/testdata
```

Then run the script, taking the C/C++ application as an example:

```
# cd /usr/local/x-linux-ai/object-detection/models/coco_ssd_mobilenet/testdata
# ls -la
-rwxr--r-- 1 root root 102821  9 20 23:14 teddy-and-cat.jpg
# cd /usr/local/x-linux-ai/object-detection
# ./launch_bin_object_detection_testdata.sh
```


The running results are as follows:

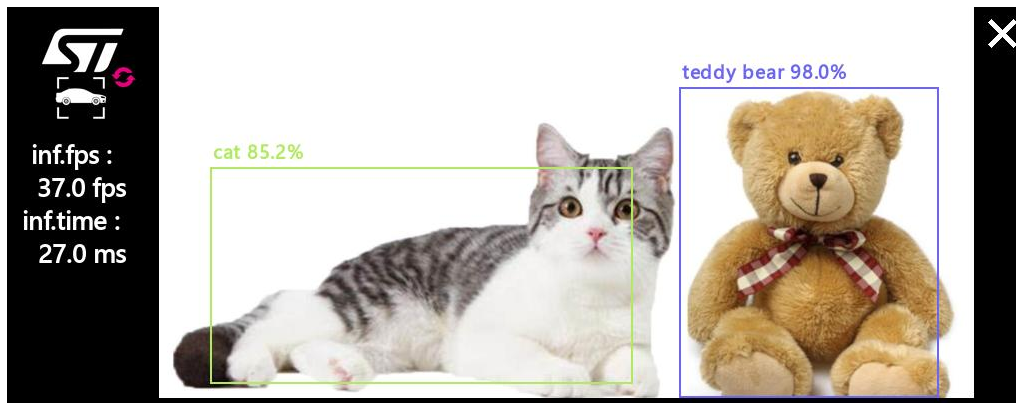


Figure 3-6. Image Input

3.3. Pose Estimation

This application demonstrates a computer vision use case for human pose estimation, capturing frames from camera input (/dev/videoX) and analyzing them through a neural network model parsed by the OpenVX framework. The model used in this application is ST YoloV8n-pose downloaded from the STM32-Hotspot Ultralytics GitHub branch.

3.3.1. Running via Demo Launcher

You can run this demo by clicking the Pose Estimation icon in the demo launcher. By default, the application installed in section 2.3 is OpenVX, and it defaults to a Python application.



Figure 3-7. Pose Estimation

3.3.2. Running via Command

The C/C++ and Python applications for pose estimation are located in the `/usr/local/x-linux-ai/pose-estimation/` directory. You can get more help information by running the program with the `-h` parameter:

```
# cd /usr/local/x-linux-ai/pose-estimation/  
# ls -la  
stai_mpu_pose_estimation.py
```

To simplify the demo startup, there are pre-configured launch scripts in the application directory:

- Using camera input to launch pose estimation demo

```
launch_python_pose_estimation.sh
```

- Using image input to launch pose estimation demo

launch_python_pose_estimation_testdata.sh

3.3.3. Demo Display

Starting the demo from the launcher defaults to using camera input, and the results will be consistent with the script used in command startup. Taking the C/C++ program as an example, the specific test is as follows:

```
# cd /usr/local/x-linux-ai/pose-estimation/
# ./launch_python_pose_estimation.sh      # or click the demo launcher icon
```



Figure 3-8. Camera Input

Before running the script for image input, prepare the image to be recognized. For example, place an image of a person running in the directory:

/usr/local/x-linux-ai/pose-estimation/models/yolov8n_pose/testdata . Then run the script, taking the C/C++ application as an example:

```
# cd /usr/local/x-linux-ai/pose-estimation/models/yolov8n_pose/testdata
# ls -la
-rwxr--r-- 1 root root 102821  9 20 23:14 person-run.jpg
# cd /usr/local/x-linux-ai/pose-estimation
# ./launch_python_pose_estimation_testdata.sh
```

The running results are as follows:

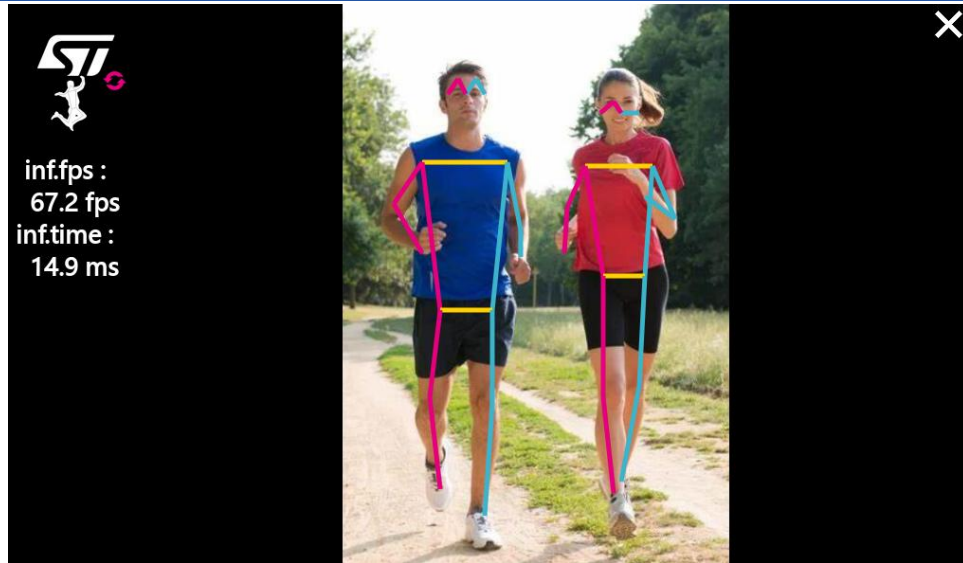


Figure 3-9. Image Input

3.4. Semantic Segmentation

This application demonstrates a computer vision use case for semantic segmentation, capturing frames from camera input (/dev/videoX) and analyzing them through a neural network model parsed by the OpenVX framework. It uses a GStreamer pipeline to stream camera frames (using v4l2src), display previews (using gtkwaylandsink), and execute neural network inference (using appsink). The inference results are displayed in the preview, with overlays implemented using GtkWidget and Cairo. The model used in this application is DeepLabV3, downloaded from TensorFlow™ Lite Hub.

3.4.1. Running via Demo Launcher

You can run this demo by clicking the Semantic Segmentation icon in the demo launcher. By default, the application installed in section 2.3 is OpenVX, and it defaults to a Python application.

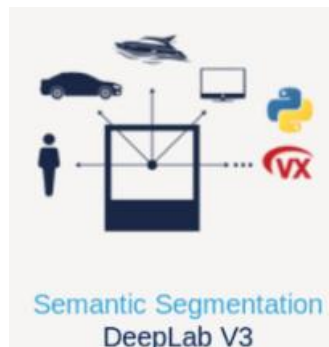


Figure 3-10. Semantic Segmentation

3.4.2. Running via Command

The C/C++ and Python applications for semantic segmentation are located in the /usr/local/x-linux-ai/semantic-segmentation/ directory. You can get more help information by running the program with the -h parameter:

```
# cd /usr/local/x-linux-ai/semantic-segmentation/  
# ls -la  
stai_mpu_semantic_segmentation.py
```

To simplify the demo startup, there are pre-configured launch scripts in the application directory:

- Using camera input to launch semantic segmentation demo

```
launch_python_semantic_segmentation.sh
```

- Using image input to launch semantic segmentation demo

```
launch_python_semantic_segmentation_testdata.sh
```

3.4.3. Demo Display

Starting the demo from the launcher defaults to using camera input, and the results will be consistent with the script used in command startup. Taking the C/C++ program as an example, the specific test is as follows:

```
# cd /usr/local/x-linux-ai/semantic-segmentation/
# ./launch_python_semantic_segmentation.sh #或者点击 demo 启动器图标
```

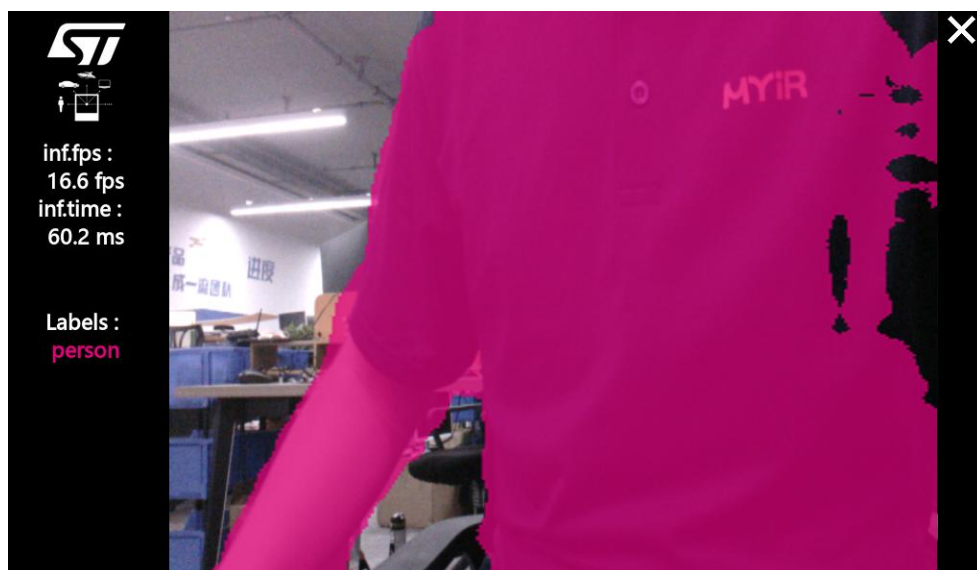


Figure 3-11. Camera Input

Before running the script for image input, prepare the image to be recognized. For example, place an image of a person working in the directory:

```
/usr/local/x-linux-ai/semantic-segmentation/models/deeplabv3/testdata
```

Then run the script, taking the C/C++ application as an example:

```
# cd /usr/local/x-linux-ai/semantic-segmentation/models/deeplabv3/testdata
# ls -la
```

```
-rwxr--r-- 1 root root 102821  9 20 23:14 person-work.jpg  
# cd /usr/local/x-linux-ai/semantic-segmentation  
# ./launch_python_semantic_segmentation_testdata.sh
```

The running results are as follows:

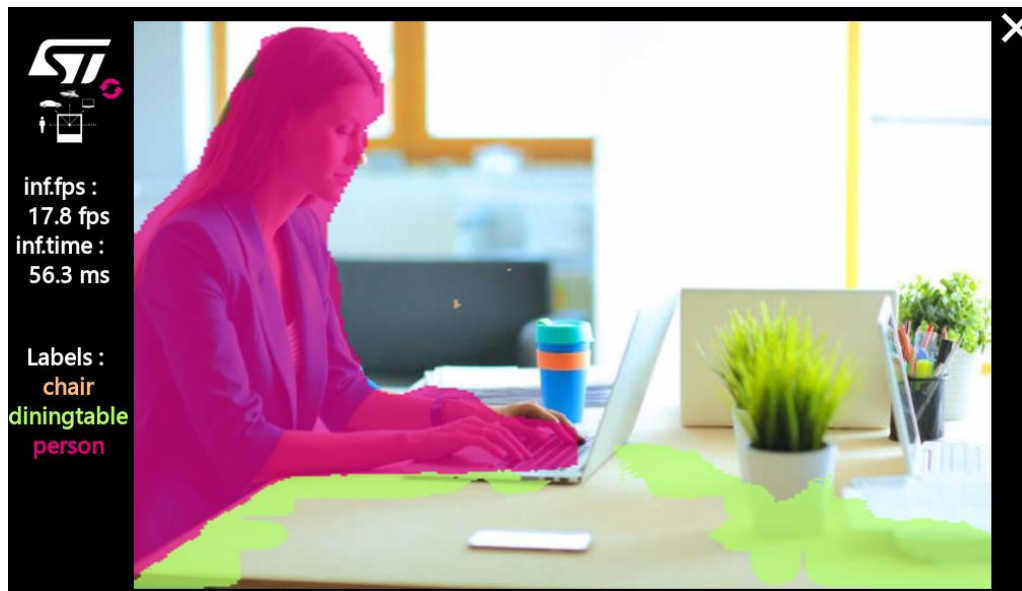


Figure 3-12. Image Input

4. References

- **X-Linux-AI-Tool**

https://wiki.st.com/stm32mpu/wiki/X-LINUX-AI_Tool

- **ST Wiki**

https://wiki.st.com/stm32mpu/wiki/Category:X-LINUX-AI_expansion_package

Appendix A

Warranty & Technical Support Services

MYIR Electronics Limited is a global provider of ARM hardware and software tools, design solutions for embedded applications. We support our customers in a wide range of services to accelerate your time to market.

MYIR is an ARM Connected Community Member and work closely with ARM and many semiconductor vendors. We sell products ranging from board level products such as development boards, single board computers and CPU modules to help with your evaluation, prototype, and system integration or creating your own applications. Our products are used widely in industrial control, medical devices, consumer electronic, telecommunication systems, Human Machine Interface (HMI) and more other embedded applications. MYIR has an experienced team and provides custom design services based on ARM processors to help customers make your idea a reality.

The contents below introduce to customers the warranty and technical support services provided by MYIR as well as the matters needing attention in using MYIR's products.

Service Guarantee

MYIR regards the product quality as the life of an enterprise. We strictly check and control the core board design, the procurement of components, production control, product testing, packaging, shipping and other aspects and strive to provide products with best quality to customers. We believe that only quality products and excellent services can ensure the long-term cooperation and mutual benefit.

Price

MYIR insists on providing customers with the most valuable products. We do not pursue excess profits which we think only for short-time cooperation. Instead, we hope to establish long-term cooperation and win-win business with customers. So we will offer reasonable prices in the hope of making the business greater with the customers together hand in hand.

Delivery Time

MYIR will always keep a certain stock for its regular products. If your order quantity is less than the amount of inventory, the delivery time would be within three days; if your order quantity is greater than the number of inventory, the delivery time would be always four to six weeks. If for any urgent delivery, we can negotiate with customer and try to supply the goods in advance.

Technical Support

MYIR has a professional technical support team. Customer can contact us by email

(support@myirtech.com), we will try to reply you within 48 hours. For mass production and customized products, we will specify person to follow the case and ensure the smooth production.

After-sale Service

MYIR offers one year free technical support and after-sales maintenance service from the purchase date.

The service covers:

Technical support service

MYIR offers technical support for the hardware and software materials which have provided to customers;

- To help customers compile and run the source code we offer;
- To help customers solve problems occurred during operations if users follow the user manual documents;
- To judge whether the failure exists;
- To provide free software upgrading service.

However, the following situations are not included in the scope of our free technical support service:

- Hardware or software problems occurred during customers' own development;
- Problems occurred when customers compile or run the OS which is tailored by themselves;
- Problems occurred during customers' own applications development;
- Problems occurred during the modification of MYIR's software source code.

After-sales maintenance service

The products except LCD, which are not used properly, will take the twelve months free maintenance service since the purchase date. But following situations are not included in the scope of our free maintenance service:

- The warranty period is expired;
- The customer cannot provide proof-of-purchase or the product has no serial number;
- The customer has not followed the instruction of the manual which has caused the damage the product;
- Due to the natural disasters (unexpected matters), or natural attrition of the components, or unexpected matters leads the defects of appearance/function;
- Due to the power supply, bump, leaking of the roof, pets, moist, impurities into the boards, all those reasons which have caused the damage of the products or defects of appearance;
- Due to unauthorized weld or dismantle parts or repair the products which has caused the damage of the products or defects of appearance;

- Due to unauthorized installation of the software, system or incorrect configuration or computer virus which has caused the damage of products.

Warm tips

1. MYIR does not supply maintenance service to LCD. We suggest the customer first check the LCD when receiving the goods. In case the LCD cannot run or no display, customer should contact MYIR within 7 business days from the moment get the goods.
2. Please do not use finger nails or hard sharp object to touch the surface of the LCD.
3. MYIR suggests user purchasing a piece of special wiper to wipe the LCD after long time use, please avoid clean the surface with fingers or hands to leave fingerprint.
4. Do not clean the surface of the screen with chemicals.
5. Please read through the product user manual before you using MYIR's products.
6. For any maintenance service, customers should communicate with MYIR to confirm the issue first. MYIR's support team will judge the failure to see if the goods need to be returned for repair service, we will issue you RMA number for return maintenance service after confirmation.

Maintenance period and charges

- MYIR will test the products within three days after receipt of the returned goods and inform customer the testing result. Then we will arrange shipment within one week for the repaired goods to the customer. For any special failure, we will negotiate with customers to confirm the maintenance period.
- For products within warranty period and caused by quality problem, MYIR offers free maintenance service; for products within warranty period but out of free maintenance service scope, MYIR provides maintenance service but shall charge some basic material cost; for products out of warranty period, MYIR provides maintenance service but shall charge some basic material cost and handling fee.

Shipping cost

During the warranty period, the shipping cost which delivered to MYIR should be responsible by user; MYIR will pay for the return shipping cost to users when the product is repaired. If the warranty period is expired, all the shipping cost will be responsible by users.

Products Life Cycle

MYIR will always select mainstream chips for our design, thus to ensure at least ten years continuous supply; if meeting some main chip stopping production, we will inform customers in time and assist customers with products updating and upgrading.

Value-added Services

1. MYIR provides services of driver development base on MYIR's products, like serial port, USB, Ethernet, LCD, etc.
2. MYIR provides the services of OS porting, BSP drivers' development, API software development, etc.
3. MYIR provides other products supporting services like power adapter, LCD panel, etc.
4. ODM/OEM services.

MYIR Electronics Limited

Room 04, 6th Floor, Building No.2, Fada Road,
Yunli Inteiligent Park, Bantian, Longgang District.

Support Email: support@myirtech.com

Sales Email: sales@myirtech.com

Phone: +86-755-22984836

Fax: +86-755-25532724

Website: www.myirtech.com