

SegmentAnything

robohub example 13.

Features

- **User-Friendly Pythonic API:** A high-level Python interface that abstracts complex QNN C++ examples, allowing for rapid prototyping on Snapdragon platforms.
- **Real-time Detection:** A two-stage segmentation approach is employed: YOLOv8 is utilized to isolate the target object's surface, followed by the SAM (Segment Anything Model) to precisely segment surface defects.

Tech Stack

- **Core:** Python 3.12+
- **Backend:** QNN 2.36
- **Web Framework:** Flask (WSGI)
- **Computer Vision:** OpenCV (cv2)
- **OS Support:** Ubuntu 24.04 (Noble Numbat)

Core Workflow

1. **Object Detection:** YOLOv8 identifies and generates a mask for the target object (e.g., a cardboard box).
2. **Region Extraction (Crop):** The system utilizes the detection mask/bounding box to crop the target object from the original high-resolution frame.
3. **Defect Segmentation:** The cropped image is passed to the **SAM (Segment Anything Model)**, which focuses exclusively on the localized area to segment surface defects with high granularity.
4. **Visualization & Storage:** Final results are overlaid on the original feed, displayed on the Dashboard, and archived to the `--save-dir`.

Getting Started

1. Prerequisites

```
1 | cd .
2 | sudo apt update
3 | sudo apt install python3-venv python3-pip qcom-fastrpc1 qcom-fastrpc-dev -y
```

2. Installation

```
1 | cd .
2 | python3 -m venv venv
3 | source venv/bin/activate
4 | pip3 install -r requirements.txt
```

3. Get QNN Resource

To use this project, you need the Qualcomm QNN SDK v2.36.0.250627.

Steps:

1. Download Qualcomm QNN SDK v2.36.0.250627.
2. Unzip the SDK to your local machine.
3. Run this script with the path to the SDK

```
1 | ./extract_qnn_deps.sh /path/your/qnn/2.36.0.250627
```

4. Build Dependencies

⚠ Prerequisite: infertoypy Installation

This project depends on **infertoypy**, which we have included in the `opensource_qnn_sample/` folder as a compressed archive.

Due to its complex build requirements, we cannot document the installation steps here. You must:

```
1 | tar -zxvf opensource_qnn_sample/release_infertoypy_*.tgz
```

Next Steps:

1. Carefully read and follow the README file inside that folder. It contains detailed instructions on how to compile the library for your specific system.
2. Verify that infertoypy is successfully installed before continuing.

Note: Please do not skip this step, or the main application will fail to start.

5. Usage

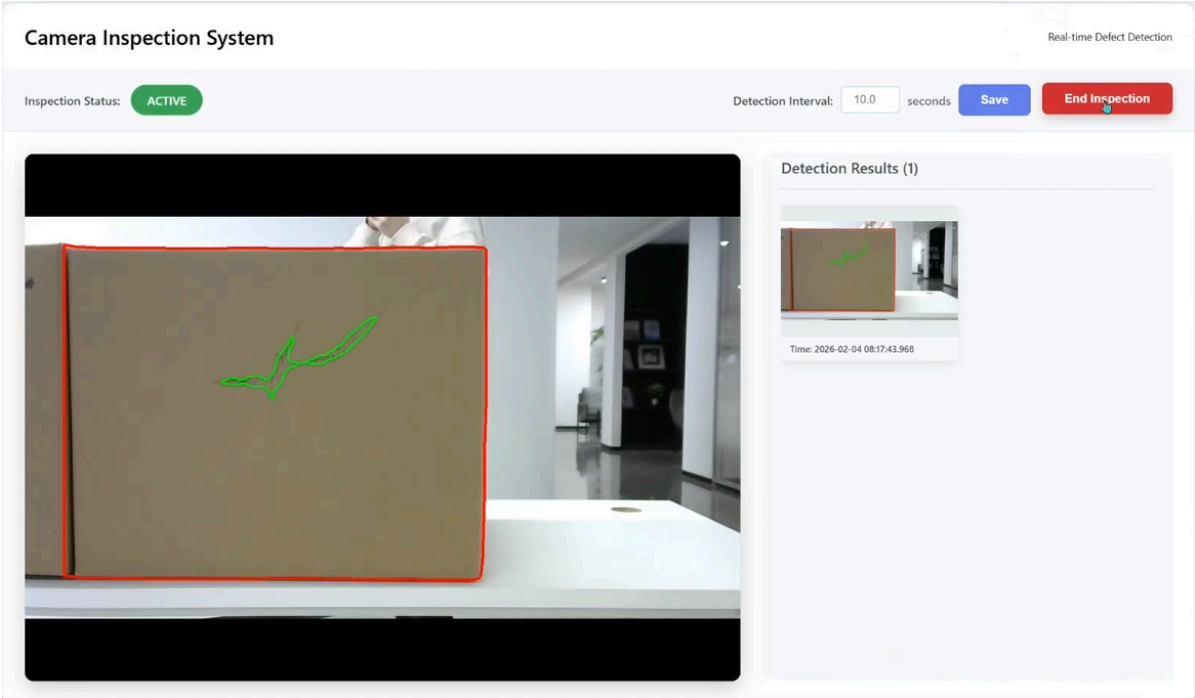
```
1 | ./start.sh --source 0 --resolution 1280 720 --class-id 0 --port 3333 --save-dir ./save
```

Access <http://localhost:3333> to open the Monitoring Dashboard.

The main interface displays a real-time camera feed. Once the detection algorithm is activated, red contours delineate the edges of the cardboard box, while green masks highlight any detected surface defects.

Captured defect frames are saved locally and populated in the right-hand panel for quick review (click to view the full-resolution image). The feed is sorted chronologically, with the most recent detections appearing at the top.

To ensure seamless integration with various production speeds, the detection interval can be customized to match your specific pipeline workflow.



Configuration

You can customize the execution of the YOLOv8 + SAM2 WebUI by passing the following command-line arguments:

Arguments Overview

Argument	Type	Default	Description
<code>--source</code>	<code>str</code>	Required	The source for the video stream (e.g., Camera ID <code>0</code> or a RTSP stream URL).
<code>--resolution</code>	<code>int</code> <code>int</code>	<code>1280</code> <code>720</code>	The desired width and height for the camera input stream.
<code>--class-id</code>	<code>int</code>	<code>0</code>	The specific class ID you want YOLOv8 to detect (e.g., <code>0</code> for box).
<code>--port</code>	<code>int</code>	<code>3333</code>	The port number on which the Web server will run.
<code>--save-dir</code>	<code>str</code>	<code>None</code>	The directory path where detection and segmentation results will be saved.

Troubleshooting

1. QNN Environment Errors (Initialization Failure)

Issue: The application fails to initialize models or throws errors related to missing `.so` libraries.

Cause: This is typically caused by missing or incorrect environment variables pointing to the QNN SDK dynamic libraries.

Solution: Ensure that `ADSP_LIBRARY_PATH` and `LD_LIBRARY_PATH` are exported to the directory containing the QNN libraries before starting the application:

```
1 # Get the absolute path of the project directory
2 export PROJ_ROOT=$(pwd)
3
4 export LD_LIBRARY_PATH=$PROJ_ROOT/resources/qnn236:$LD_LIBRARY_PATH
5 export ADSP_LIBRARY_PATH=$PROJ_ROOT/resources/qnn236
```

2. Camera Stream Warnings (JPEG Corruption)

Issue: The console is flooded with warnings like Corrupt JPEG data: 2 extraneous bytes before marker 0xd5.

Cause: This is a common issue with certain UVC cameras or low-level V4L2 drivers when the JPEG stream contains non-standard markers or padding bytes.

Solution: While these warnings are often non-fatal and can be ignored if the feed looks normal, you can try changing the pixel format to YUYV in the source code if your camera supports it.

Locate lines 54-55 in `python/run.py` and swap the commented lines as shown below:

Original Code (Default):

```
1 self.cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'MJPG'))
2 # self.cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'YUYV'))
```

Updated Code (YUYV):

```
1 # self.cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'MJPG'))
2 self.cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'YUYV'))
```

License

Copyright (c) 2026, Qualcomm Innovation Center, Inc. All rights reserved.

This project is licensed under the **BSD 3-Clause License (SPDX: BSD-3-Clause)**.

A copy of the license is included in the [LICENSE](#) file.