**Based on NVIDIA Jetson Vision Recognition Project**

**Contents:**

* Foreword
* Project Overview
* Features
* Environment and Dependencies
* Environment Configuration Files List
* Installation and Setup
  + 1. Setting up the NVIDIA Jetson Device
    2. Installing Docker and NVIDIA Container Toolkit
    3. Running Docker Containers
    4. Starting Services
* User Guide
  + Accessing JupyterLab
  + Accessing Gradio Interface
  + Camera Calibration
  + Hand-Eye Calibration
* Frequently Asked Questions and Troubleshooting
* License

**Foreword** This project is based on Section 1 of the Jetson AI Courses and Certifications—NVIDIA Deep Learning Institute's *Getting Started with AI on Jetson Nano* course, drawing inspiration from the *Realtime Language-Segment-Anything on Jetson Orin* project from the NVIDIA Jetson AI Lab. Key contributions include integrating camera calibration and industrial robot hand-eye calibration functionalities, enabling automated vision-guided picking with industrial robots.

Special thanks to NVIDIA for the detailed tutorials and to the *Realtime Language-Segment-Anything on Jetson Orin* project team for their outstanding work. This project is open for reference without requiring a license.

**Project Overview** This project is a vision recognition system based on the NVIDIA Jetson platform, utilizing deep learning to achieve real-time object detection and segmentation. Deployed using Docker containerization, the project provides both JupyterLab and Gradio interfaces for ease of development and testing.

**Features**

* Real-time video capture and object detection
* Support for camera calibration and hand-eye calibration
* Automated vision-guided picking for industrial robots
* JupyterLab and Gradio interfaces
* Docker containerized deployment for easy installation and management

**Environment and Dependencies**

* **Hardware:**
  + NVIDIA Jetson AGX Orin development board
  + Logitech C270 camera (or other compatible cameras)
* **Operating System:**
  + Ubuntu 22.04
  + JetPack 6.0 (corresponding to L4T 36.3.0)
* **Software:**
  + Docker CE
  + NVIDIA Container Toolkit
  + JupyterLab
  + Gradio
  + Python 3 and related libraries

**Environment Configuration Files List** During the installation and operation of this project, several configuration files are involved. Below is the list of configuration files and their descriptions:

1. **Docker Daemon Configuration File**
   * **Path:** /etc/docker/daemon.json
   * **Purpose:** Configures the Docker daemon to use the NVIDIA runtime.
   * **Sample Content:**

json

复制代码

{

"default-runtime": "nvidia",

"runtimes": {

"nvidia": {

"path": "nvidia-container-runtime",

"runtimeArgs": []

}

}

}

1. **Python Virtual Environment and Dependencies**
   * **Path:** requirements.txt in the project directory (if available)
   * **Purpose:** Lists the dependencies for the Python project, facilitating environment setup.
   * **Sample Content:**

复制代码

numpy

opencv-python

gradio

jupyterlab

1. **Camera Calibration Parameter Files**
   * **Path:** /nvdli-nano/Realtime\_Language\_Segment\_Anything/data/calibration/
   * **Purpose:** Stores parameters after camera calibration for subsequent image processing.
   * **Files:**
     + camera\_matrix.npy
     + dist\_coeffs.npy
2. **Hand-Eye Calibration Parameter File**
   * **Path:** Custom storage location based on project requirements
   * **Purpose:** Stores transformation matrix after hand-eye calibration, used for coordinate conversion.
3. **HTTP Service Directory**
   * **Path:** /nvdli-nano/Realtime\_Language\_Segment\_Anything/data/coords/
   * **Purpose:** Stores coordinate data files (coordinates.csv) for access by other clients.
4. **Gradio Application File**
   * **Path:** /nvdli-nano/Realtime\_Language\_Segment\_Anything/app\_gradio.py
   * **Purpose:** Main program for the Gradio application, including interface design and logic handling.
5. **JupyterLab Configuration File**
   * **Path:** ~/.jupyter/jupyter\_lab\_config.py (if available)
   * **Purpose:** Configures JupyterLab startup parameters, such as IP, port, and password.
6. **Docker Container Startup Script**
   * **Path:** /nvdli-nano/start\_services.sh
   * **Purpose:** Automatically starts necessary services when the container is launched.
   * **Sample Content:**

bash

复制代码

#!/bin/bash

# Start HTTP server

cd /nvdli-nano/Realtime\_Language\_Segment\_Anything/data/coords

python3 -m http.server 8080 &

# Start Gradio service

cd /nvdli-nano/Realtime\_Language\_Segment\_Anything

python3 app\_gradio.py &

# Prevent container from exiting

wait

1. **Camera Supported Resolution Configuration**
   * **How to Obtain:** Use the command v4l2-ctl --list-formats-ext
   * **Purpose:** Identifies supported resolutions and frame rates of the camera, helping to set appropriate parameters in code.

**Installation and Setup**

1. **Setting up the NVIDIA Jetson Device**
   1. **Flash the JetPack System:**
      * Use the sdkmanager tool to flash JetPack 6.0 onto the Jetson device on the host computer.
      * Follow the prompts to complete system installation and initial setup.
   2. **Configure the Network:**
      * Ensure the Jetson device is connected to the network.
      * Record the device's IP address, which will be used for subsequent access.
2. **Installing Docker and NVIDIA Container Toolkit**
   1. **Install Docker CE:**

bash

复制代码

# Download the required Docker packages and place them in ~/docker directory

sudo dpkg -i ~/docker/containerd.io\_1.6.33-1\_arm64.deb \

~/docker/docker-ce\_26.1.4-1~ubuntu.22.04~jammy\_arm64.deb \

~/docker/docker-ce-cli\_26.1.4-1~ubuntu.22.04~jammy\_arm64.deb \

~/docker/docker-buildx-plugin\_0.14.1-1~ubuntu.22.04~jammy\_arm64.deb \

~/docker/docker-compose-plugin\_2.27.1-1~ubuntu.22.04~jammy\_arm64.deb

# If dependency issues occur, run the following command

sudo apt-get install -f

1. **Start and Configure Docker Service:**

bash

复制代码

sudo systemctl start docker

sudo systemctl enable docker # Enable Docker at startup

sudo systemctl status docker # Check Docker service status

1. **Install NVIDIA Container Toolkit:**

bash

复制代码

# Add the NVIDIA repository and install

curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg \

&& curl -s -L https://nvidia.github.io/libnvidia-container/stable/ubuntu20.04/$(ARCH)/nvidia-container-toolkit.list | \

sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg] https://#g' | \

sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list

sudo apt-get update

sudo apt-get install -y nvidia-container-toolkit

sudo nvidia-ctk runtime configure --runtime=docker

sudo systemctl restart docker

1. **Running Docker Containers**
   1. **Create Data Directory:**

bash

复制代码

mkdir -p ~/nvdli-data

1. **Run the Container:**

bash

复制代码

sudo docker run --runtime nvidia -it --rm --network host \

--volume ~/nvdli-data:/nvdli-nano/data \

--device /dev/video0 \

nvcr.io/nvidia/dli/dli-nano-ai:v2.0.3-r36.3.0

If specific port mappings or other configurations are needed, adjust the command accordingly.

1. **Starting Services** After entering the container, follow these steps to start the services:
2. **Start JupyterLab:**

bash

复制代码

# Inside the container

jupyter-lab --allow-root --ip=0.0.0.0 --port=8888

1. **Start Gradio Service:**

bash

复制代码

# Inside the container

python3 /nvdli-nano/Realtime\_Language\_Segment\_Anything/app\_gradio.py

1. **Start HTTP Service (for coordinate file transfer):**

bash

复制代码

# Inside the container

cd /nvdli-nano/Realtime\_Language\_Segment\_Anything/data/coords

python3 -m http.server 8080

**User Guide**

* **Accessing JupyterLab:**
  + In your browser, enter: http://<Jetson\_IP>:8888
  + Enter password: dlinano
* **Accessing Gradio Interface:**
  + In your browser, enter: http://<Jetson\_IP>:7860

**Camera Calibration**

1. **Prepare the Chessboard Pattern:**
   * Use the provided chessboard.png (10x7 squares, each square 25 mm in size).
2. **Capture Calibration Images:**
   * In the Gradio or JupyterLab interface, click the "Capture" button to collect multiple images containing the chessboard.
   * The images will be saved in the directory /nvdli-nano/Realtime\_Language\_Segment\_Anything/data/Image/chessboard/.
3. **Execute Calibration:**
   * Click the "Calibrate" button, and the system will automatically read the captured images and compute the camera's intrinsic parameters.

**Hand-Eye Calibration**

1. **Prepare Calibration Data:**
   * Teach the positions of several feature points (e.g., chessboard corner points) in the robot's base coordinate system.
   * Record the corresponding feature point coordinates in the camera's coordinate system.
2. **Execute Hand-Eye Calibration:**
   * Run the hand-eye calibration algorithm to compute the transformation matrix from the camera coordinate system to the robot's base coordinate system.
   * This can be executed in a Python environment, and the result will be used for coordinate conversion.

**Frequently Asked Questions and Troubleshooting**

* **Cannot Access Services:**
  + Ensure the Docker container is running properly, and services have started.
  + Check if the Jetson device's IP address is correct and whether firewall settings block port access.
* **Camera Not Recognized:**
  + Verify that the camera is properly connected to the Jetson device's USB port.
  + Use v4l2-ctl --list-devices to check if the camera is recognized by the system.
* **Excessive or Garbled Log Information:**
  + Adjust the logging level by setting the logging module level to INFO and clean up unnecessary log handlers in the code.

**License** No license is required.