Docker Image Handover Document

Introduction

This handover document provides all the necessary details for the management and operation of two Docker images developed for the `siyiliu6/ai_redback` project. These images are designed to facilitate advanced AI and robotics operations using object detection, distance estimation, human posture detection functionalities and voice command functionality within ROS 2.

https://hub.docker.com/r/siyiliu6/ai_redback

Docker Images

1. vision Image

The Docker container is built using the provided, which sets up the necessary environment. The following components are included:

- **Ultralytics:** For object detection tasks.
- **PyRealSense:** For interfacing with Intel RealSense depth cameras. (Note: ARM architecture is not supported.)
- **CUDA & cuDNN:** For GPU acceleration (pre-installed in the NVIDIA Docker base image).
- **Caffe Prerequisites:** Required for OpenPose.
- **OpenPose:** For human posture detection.

Prerequisites

Before running the container, ensure the following prerequisites are met:

1. **Docker:** Docker must be installed on the host machine. Refer to the official Docker documentation for installation instructions.
2. **NVIDIA Drivers & Docker Integration:** If CUDA functionality is needed, ensure that the NVIDIA drivers and NVIDIA Container Toolkit are installed for Docker to utilize the GPU.

Running the Container

To run the Docker container, follow these steps:

1. **Build the Docker Image:** Navigate to the directory containing the Dockerfile and run:

   ```
   docker build -t object-detection-container .
   ```

   This command will build the Docker image with the tag `object-detection-container`.
2. **Start the Container:** Run the following command to start the container:

   ```
   docker run --gpus all -it --name object_detection_env object-detection-container
   ```

   The `--gpus all` flag enables GPU access within the container, which is necessary for CUDA-based components.

Usage Instructions

Object Detection

- **YOLO Detector Node:** After the container has been successfully built and started, the YOLO detector ROS2 node can be launched with the command:

  ```
  ros2 run object_detector yolo_detector
  ```

Distance Estimation

- **RealSense Depth Estimator Node:** The RealSense depth estimator node can be launched with the command:

  ```
  ros2 run depth_estimator realsense_depth_estimator
  ```

Human Posture Detection

- **OpenPose:** To run OpenPose after building, use the following command:

  ```
  ./build/examples/openpose/openpose.bin
  ```

Repository

The Docker image is located at:

docker pull siyiliu6/ai_redback:latest

Support and Maintenance

For additional support and documentation, refer to the following:

- YOLO Detector Node: [Docs/Link]
- RealSense Depth Estimator Node: [Docs/Link]
- OpenPose GitHub Repository: [https://github.com/CMU-Perceptual-Computing-Lab/openpose]

2. Voice2Command Image

Description

The Voice2Command image is a custom ROS 2 environment, prepared for voice-controlled robotics applications. This container is set up with ROS Foxy and Python 3.11.6, along with the necessary dependencies to run the voice command application.

Features

- Built on ROS Foxy base image.
- The application source is fully copied into the `/AI-RedBack/` directory in the container.
- Exposes port 8000 for external communication.
- The container starts with a Python voice command application as the default process.

How to Use

To deploy the Voice2Command Docker image, use:

```
docker run -it --rm -p 8000:8000 siyiliu6/ai_redback:voice2command-latest
```

Repository

The Docker image is located at:

docker pull siyiliu6/ai_redback:voice

Support and Maintenance

Documentation

- Full documentation for OpenPose can be found at OpenPose GitHub.
- Documentation for ROS 2 Foxy can be found at ROS 2 Wiki.