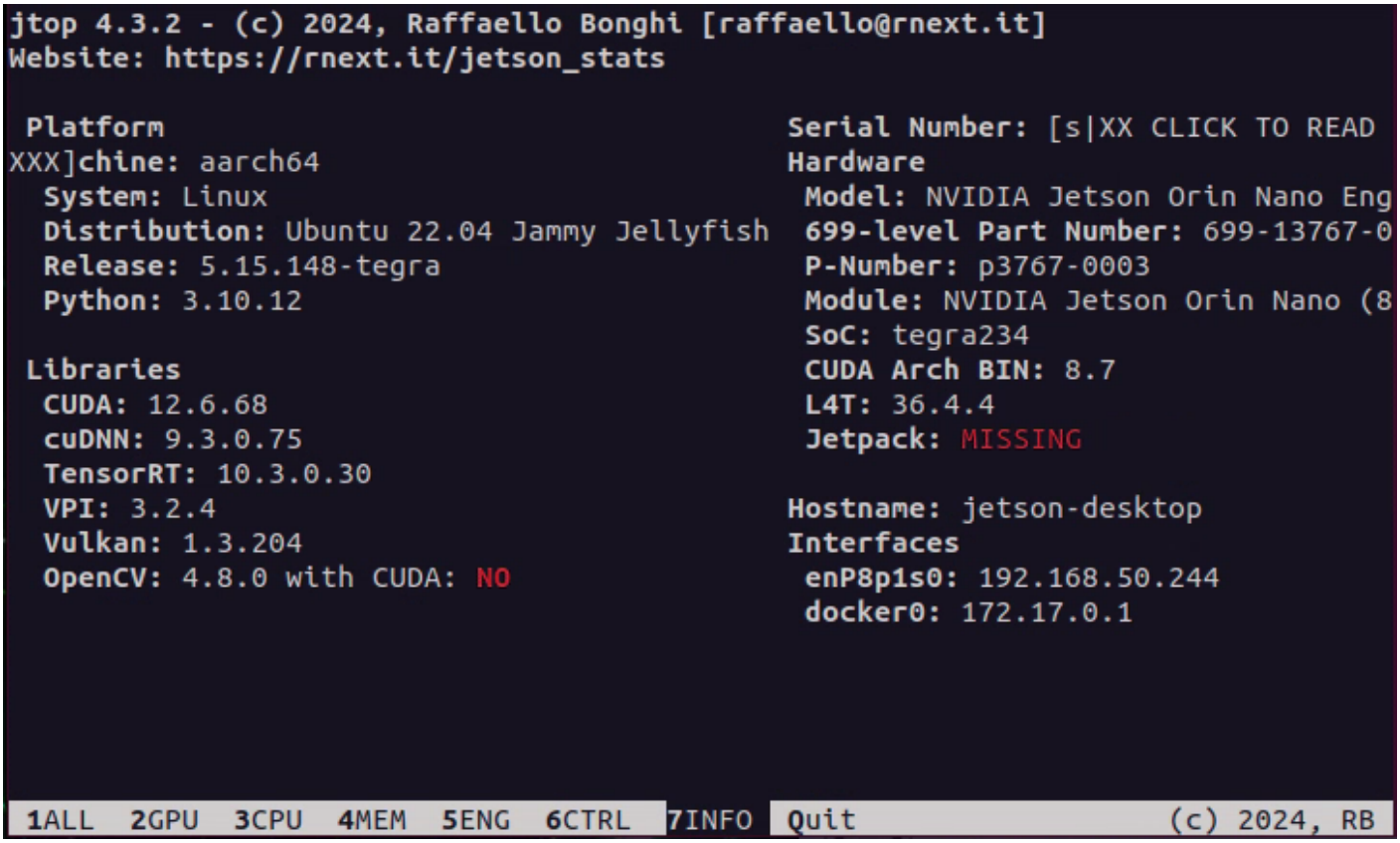
????opencv with cuda

> JetPack□□□ opencv□□□□ cuda□□□□□□□□□□



1. ??????

```
#!/bin/bash
# opencv_install.sh
# Modified from
https://github.com/AastaNV/JEP/blob/master/script/install_opencv4.10.0_Jetpack6.1.sh

version="4.10.0"
folder="workspace"
remove_old=""

set -e

# Parse command-line arguments
for arg in "$@"; do
```

```
    case $arg in
        --version=*)
            version="${arg#*=}"
            ;;
        --folder=*)
            folder="${arg#*=}"
            ;;
        --remove-old=*)
            remove_old="${arg#*=}"
            ;;
        --help|-h)
            echo "Usage: $0 [--version=4.x.x] [--folder=dir] [--remove-old=yes/no]"
            exit 0
            ;;
        *)
            echo "Unknown parameter: $arg"
            echo "Usage: $0 [--version=4.x.x] [--folder=dir] [--remove-old=yes/no]"
            exit 1
            ;;
    esac
done

# Create installation directory if it doesn't exist
if [ ! -d "$folder" ]; then
    echo "Creating directory: $folder"
    mkdir -p "$folder"
fi
cd "$folder" || exit

# Old OpenCV removal logic
if [ -z "$remove_old" ]; then
    read -rp "Do you want to remove system-installed OpenCV? (yes/no): " remove_old
fi

case "$remove_old" in
    [yY] | [yY][eE][sS])
        echo "** Removing system OpenCV packages"
        sudo apt -y purge *libopencv*
        sudo apt -y autoremove
        ;;
```

```
    *)
        echo "** Skipping system OpenCV removal"
        ;;
esac

echo "-----------------------------------"
echo "** Installing dependencies (1/4)"
echo "-----------------------------------"
sudo apt-get update
sudo apt-get install -y build-essential cmake git libgtk2.0-dev pkg-config \
    libavcodec-dev libavformat-dev libswscale-dev libgstreamer1.0-dev \
    libgstreamer-plugins-base1.0-dev python3-dev python3-numpy libtbb2 \
    libtbb-dev libjpeg-dev libpng-dev libtiff-dev libv4l-dev v4l-utils qv4l2 curl

# Verify essential dependencies installed
for dep in g++ cmake git pkg-config; do
    if ! command -v "$dep" > /dev/null; then
        echo "Error: $dep installation failed"
        exit 1
    fi
done

echo "-----------------------------------"
echo "** Downloading OpenCV ${version} (2/4)"
echo "-----------------------------------"

# Check if source files already exist
download_opencv=false
download_contrib=false

if [ ! -f "opencv-${version}.zip" ]; then
    echo "Downloading opencv-${version}.zip"
    wget -O opencv-${version}.zip https://github.com/opencv/opencv/archive/${version}.zip || {
        echo "Download failed! Check your internet connection or verify the version exists"
        exit 1
    }
    download_opencv=true
else
    echo "opencv-${version}.zip exists, skipping download"
fi
```

```
if [ ! -f "opencv_contrib-${version}.zip" ]; then
    echo "Downloading opencv_contrib-${version}.zip"
    wget -O opencv_contrib-${version}.zip
https://github.com/opencv/opencv_contrib/archive/${version}.zip || {
        echo "Download failed! Check your internet connection or verify the version exists"
        exit 1
    }
    download_contrib=true
else
    echo "opencv_contrib-${version}.zip exists, skipping download"
fi

# Unpack source files
if [ ! -d "opencv-${version}" ] || $download_opencv; then
    if [ -d "opencv-${version}" ]; then
        echo "Removing existing opencv-${version} directory"
        rm -rf "opencv-${version}"
    fi
    echo "Unpacking opencv-${version}.zip"
    unzip -q opencv-${version}.zip || {
        echo "Extraction failed! File may be corrupt"
        exit 1
    }
fi

if [ ! -d "opencv_contrib-${version}" ] || $download_contrib; then
    if [ -d "opencv_contrib-${version}" ]; then
        echo "Removing existing opencv_contrib-${version} directory"
        rm -rf "opencv_contrib-${version}"
    fi
    echo "Unpacking opencv_contrib-${version}.zip"
    unzip -q opencv_contrib-${version}.zip || {
        echo "Extraction failed! File may be corrupt"
        exit 1
    }
fi

# Clean up zip files after successful extraction
if [ $? -eq 0 ]; then
```

```
    rm -f opencv-${version}.zip opencv_contrib-${version}.zip
fi

cd opencv-${version} || exit

echo "-----------------------------------"
echo "** Building OpenCV ${version} (3/4)"
echo "-----------------------------------"
mkdir -p release
cd release

# Auto-detect CUDA architecture
cuda_arch=""
if command -v nvidia-smi &> /dev/null; then
    gpu_name=$(nvidia-smi --query-gpu=name --format=csv,noheader | head -n1)
    if [[ $gpu_name == *"Orin"* ]] || [[ $gpu_name == *"Jetson"* ]]; then
        cuda_arch="8.7"
    elif [[ $gpu_name == *"A100"* ]]; then
        cuda_arch="8.0"
    fi
fi

cmake_cmd="cmake -D WITH_CUDA=ON -D WITH_CUDNN=ON -D OPENCV_GENERATE_PKGCONFIG=ON "
cmake_cmd+="-D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib-${version}/modules "
cmake_cmd+="-D WITH_GSTREAMER=ON -D WITH_LIBV4L=ON -D BUILD_opencv_python3=ON "
cmake_cmd+="-D BUILD_TESTS=OFF -D BUILD_PERF_TESTS=OFF -D BUILD_EXAMPLES=OFF "
cmake_cmd+="-D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local "

# Add CUDA architecture if detected
if [ -n "$cuda_arch" ]; then
    echo "Detected NVIDIA GPU: ${gpu_name}, using CUDA_ARCH_BIN=${cuda_arch}"
    cmake_cmd+="-D CUDA_ARCH_BIN=${cuda_arch} -D CUDA_ARCH_PTX=\"\" "
else
    echo "No supported GPU detected, skipping CUDA architecture flags"
fi

# Execute CMake configuration
echo "CMake command: $cmake_cmd .."
$cmake_cmd .. || {
    echo "CMake configuration failed"
```

```
    exit 1
}

# Parallel build (leave one core for system stability)
cpu_cores=$(($(nproc) - 1))
[ $cpu_cores -lt 1 ] && cpu_cores=1
echo "Building with ${cpu_cores} CPU cores"
make -j${cpu_cores} || {
    echo "Compilation failed"
    exit 1
}

echo "-----------------------------------"
echo "** Installing OpenCV ${version} (4/4)"
echo "-----------------------------------"
sudo make install || {
    echo "Installation failed"
    exit 1
}

# Add environment variables to .bashrc (only if not already present)
bashrc=~/.bashrc
env_lines=(
    "export LD_LIBRARY_PATH=/usr/local/lib:\$LD_LIBRARY_PATH"
    "export PYTHONPATH=/usr/local/lib/python3.10/site-packages/:\$PYTHONPATH"
)

for line in "${env_lines[@]}"; do
    if ! grep -Fxq "$line" "$bashrc"; then
        echo "Adding to .bashrc: $line"
        echo "$line" >> "$bashrc"
    else
        echo "Environment variable already exists: $line"
    fi
done

source ~/.bashrc

echo "** OpenCV ${version} installation completed"
echo "Verification commands:"
```

```
echo "  pkg-config --modversion opencv4"
echo "  python3 -c 'import cv2; print(cv2.__version__)'"

echo "** Installation successful!"
```

2. ????

2.1 卸载自带 opencv

```
sudo apt-get purge libopencv*
sudo apt autoremove
sudo apt-get update
```

2.2 安装前置软件包

```
sudo apt-get update
sudo apt-get install -y build-essential cmake git libgtk2.0-dev pkg-config libavcodec-dev
libavformat-dev libswscale-dev
sudo apt-get install -y libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev python3.10-dev
python3-numpy
sudo apt-get install -y libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libv4l-dev v4l-
utils qv4l2
sudo apt-get install -y curl
```

2.3 获取 opencv源码 （以 4.10.0版本为例 ）

```
version=4.10.0
wget -O "opencv-${version}.zip" "https://github.com/opencv/opencv/archive/${version}.zip"
wget -O "opencv_contrib-${version}.zip"
"https://github.com/opencv/opencv_contrib/archive/${version}.zip"
unzip "opencv-${version}.zip"
unzip "opencv_contrib-${version}.zip"
rm "opencv-${version}.zip" "opencv_contrib-${version}.zip"
cd "opencv-${version}/"
```

2.4 编译源码

> 此步骤至少需要半小时以上。

>

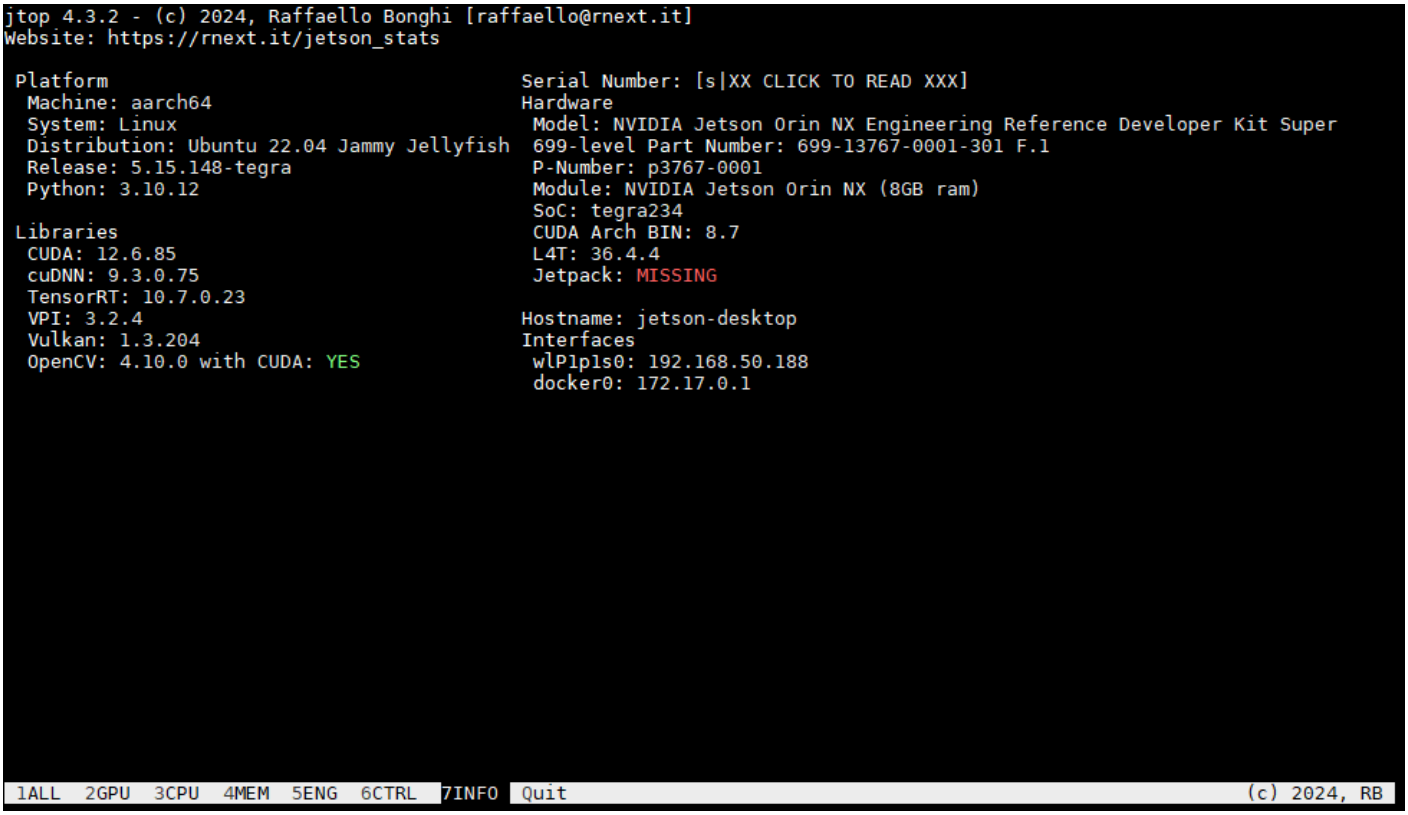
□□□□□□□□□□□□□□□□□□□□□□□□□□

```
mkdir build
cd build/
cmake -D WITH_CUDA=ON -D WITH_CUDNN=ON -D CUDA_ARCH_BIN="8.7" -D CUDA_ARCH_PTX="" -D
OPENCV_GENERATE_PKGCONFIG=ON -D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib-
${version}/modules -D WITH_GSTREAMER=ON -D WITH_LIBV4L=ON -D BUILD_opencv_python3=ON -D
BUILD_TESTS=OFF -D BUILD_PERF_TESTS=OFF -D BUILD_EXAMPLES=OFF -D CMAKE_BUILD_TYPE=RELEASE -D
CMAKE_INSTALL_PREFIX=/usr/local ..
make -j$(nproc)
```

2.5 □□

```
sudo make install
echo 'export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH' >> ~/.bashrc
echo 'export PYTHONPATH=/usr/local/lib/python3.10/site-packages/:$PYTHONPATH' >> ~/.bashrc
source ~/.bashrc
```

??????



```
#--test_cuda.cpp

#include <opencv2/opencv.hpp>
#include <opencv2/core/cuda.hpp>
#include <opencv2/cudaarithm.hpp>
#include <iostream>
#include <chrono>

// CPU □□□□
void cpu_matrix_mult(cv::Mat& a, cv::Mat& b, cv::Mat& result) {
    for (int i = 0; i < 50; i++) {
        result = a * b;
    }
}

// GPU □□□□
void gpu_matrix_mult(cv::cuda::GpuMat& d_a, cv::cuda::GpuMat& d_b, cv::cuda::GpuMat& d_result)
{
    cv::cuda::Stream stream;

    for (int i = 0; i < 50; i++) {
        cv::cuda::gemm(d_a, d_b, 1.0, cv::cuda::GpuMat(), 0, d_result, 0, stream);
        stream.waitForCompletion();
    }
}

int main() {
    try {
        std::cout << "--- OpenCV CUDA Matrix Multiplication Test ---\n";

        // □□□□ 1000x1000 □□□□□
        cv::Mat mat_a(1000, 1000, CV_32FC1);
        cv::Mat mat_b(1000, 1000, CV_32FC1);
        cv::randu(mat_a, 0.0f, 1.0f);
        cv::randu(mat_b, 0.0f, 1.0f);

        cv::Mat cpu_result;

        // CPU □□
        auto start_cpu = std::chrono::high_resolution_clock::now();
```

```
        cpu_matrix_mult(mat_a, mat_b, cpu_result);
        auto end_cpu = std::chrono::high_resolution_clock::now();
        double cpu_time = std::chrono::duration_cast<std::chrono::milliseconds>(end_cpu -
start_cpu).count();

        // GPU □□
        cv::cuda::GpuMat d_mat_a, d_mat_b, d_result;
        d_mat_a.upload(mat_a);
        d_mat_b.upload(mat_b);

        auto start_gpu = std::chrono::high_resolution_clock::now();
        gpu_matrix_mult(d_mat_a, d_mat_b, d_result);
        auto end_gpu = std::chrono::high_resolution_clock::now();
        double gpu_time = std::chrono::duration_cast<std::chrono::milliseconds>(end_gpu -
start_gpu).count();

        // □□□□□□□□□
        cv::Mat gpu_result;
        d_result.download(gpu_result);

        // □□□□(□□□□)
        double diff = cv::norm(cpu_result, gpu_result, cv::NORM_L2);
        std::cout << "Result difference: " << diff << "\n";

        std::cout << "Performance Results:\n"
                  << " - CPU time: " << cpu_time << " ms\n"
                  << " - GPU time: " << gpu_time << " ms\n"
                  << " - Speedup: " << cpu_time / gpu_time << "x\n";

        std::cout << "\n CUDA matrix multiplication test completed\n";
        return 0;

    } catch (const cv::Exception& e) {
        std::cerr << "OpenCV Error (" << e.err << "): " << e.what() << "\n";
        return -1;
    } catch (const std::exception& e) {
        std::cerr << "Standard Error: " << e.what() << "\n";
        return -2;
    }
}
```

编译运行

```
jetson@jetson-desktop:~/work$ g++ test_cuda.cpp -o test_cuda `pkg-config --cflags --libs opencv4`
jetson@jetson-desktop:~/work$ ./test_cuda
--- OpenCV CUDA Performance Test ---
Performance Results:
 - CPU time: 2451 ms
 - GPU time: 918 ms
 - Speedup: 2.66993x

 CUDA performance test completed
```

版本号 #11
由 edit 创建于 16 七月 2025 06:46:59
由 edit 更新于 28 七月 2025 02:05:02