# Halcon Software Evaluation Test

## I. Objectives

This test uses a laser target product as an example to explore the feasibility of the Halcon tool for developing related optoelectronic products.

* Ease of developing products based on Halcon;
* Performance of Halcon software on linux platform;

## II. Halcon-based development process

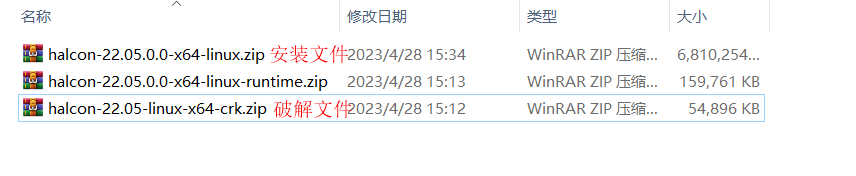
Development dependencies:

* Development Hosting Ubuntu 20.04 and Win10
* Halcon version 22.05
* Embedded development board for Mill development board, Linux kernel version 4.9.170

The main development steps consist of four parts: installing the Halcon software, configuring the cross-compilation environment, compiling and deploying.

### 2.1 Installing Halcon 22.05

Install Halcon22.05 via SOM in ubuntu 20.04, and after the installation is complete, just replace the two files under halcon-22.05-linux-x64-crk in the software installation directory.

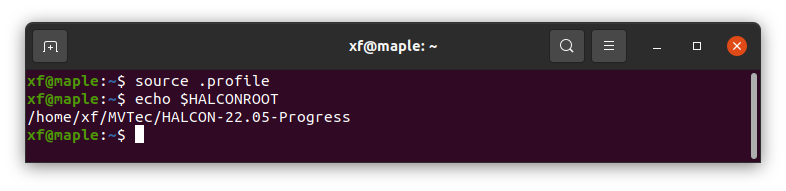


To configure environment variables, open the .profile file in the user directory and add the following:

HALCONROOT="/home/xf/MVTec/HALCON-22.05-Progress"; export HALCONROOT  
HALCONARCH="x64-linux"; export HALCONARCH  
PATH="${HALCONROOT}/bin/${HALCONARCH}:${PATH}"; export PATH  
LD\_LIBRARY\_PATH="${HALCONROOT}/lib/${HALCONARCH}:${LD\_LIBRARY\_PATH}"; export LD\_LIBRARY\_PATH  
HALCONEXAMPLES="/home/xf/MVTec/HALCON-22.05-Progress/examples"; export HALCONEXAMPLES

Note: HALCONROOT fills in the actual installation directory, for example: /home/xf/MVTec/HALCON-22.05-Progress.

Execute source .profile to make the configuration take effect, and verify it by typing echo $HALCONROOT (Note: If you are using a normal user, it is better to reboot the computer to make it take effect).



### 2.2 Installing the cross-compilation chain

sudo apt-get install gcc-aarch64-linux-gnu  
sudo apt-get install g++-aarch64-linux-gnu

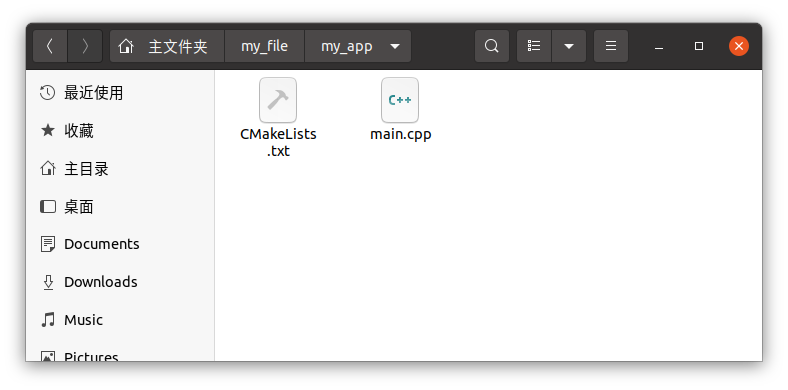
sudo apt-get install gcc-arm-linux-gnueabihf  
sudo apt-get install g++-arm-linux-gnueabihf



### 2.3 Compilation

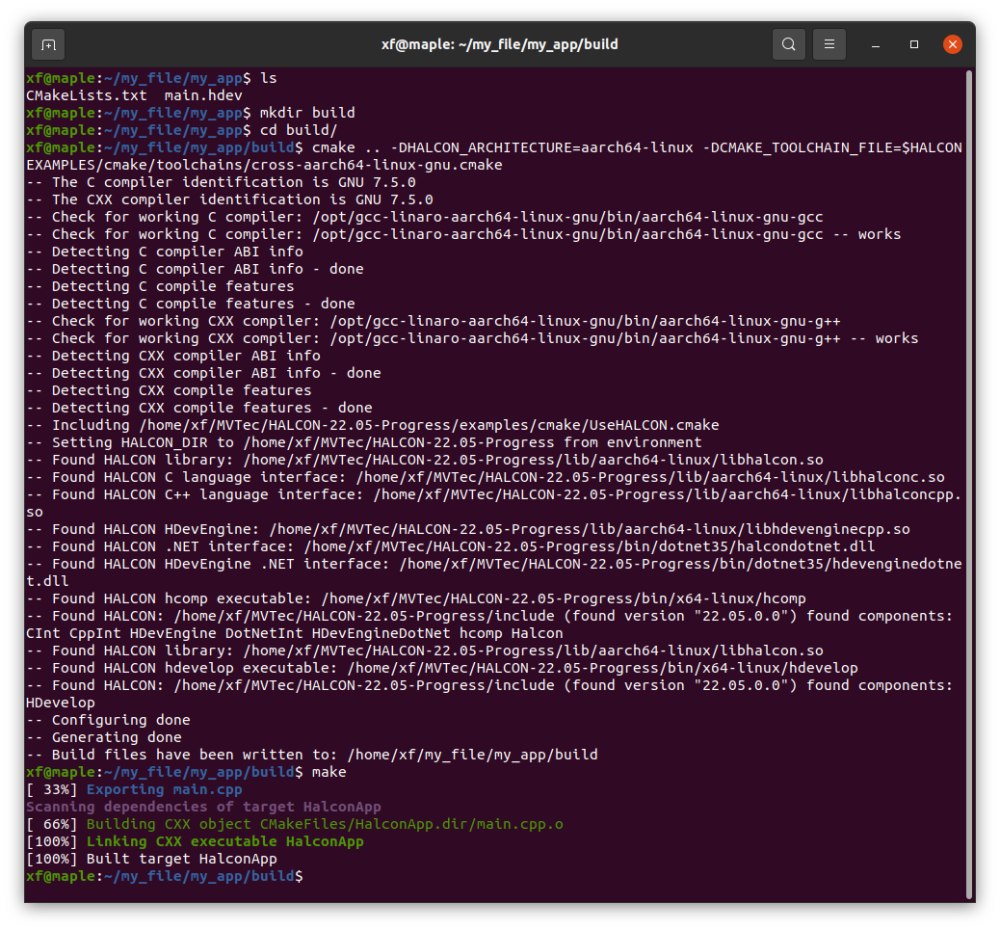
Copy the hdev file and create the CMakeLists.txt file.

cmake\_minimum\_required(VERSION 3.7.1)  
project(HalconApp VERSION 0.1.0 LANGUAGES C CXX)  
# Add the cmake directory for halcon  
list(APPEND CMAKE\_MODULE\_PATH $ENV{HALCONEXAMPLES}/cmake)  
# Embedded does not support xl  
option(HALCON\_XL "Use HALCON XL" OFF)  
# Includes generic configuration for halcon projects  
include(UseHALCON)  
# Generate c++ code  
add\_executable(HalconApp main.cpp)  
target\_link\_libraries(HalconApp HALCON::CppInt)



Compile source code

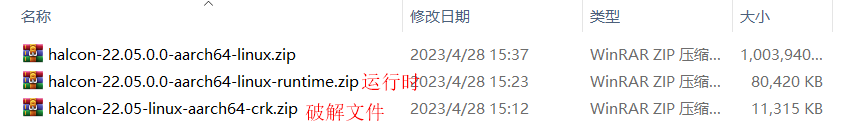
mkdir build  
  
cd build/  
  
cmake .. -DHALCON\_ARCHITECTURE=aarch64-linux -DCMAKE\_TOOLCHAIN\_FILE=$HALCONEXAMPLES/cmake/toolchains/cross-aarch64-linux-gnu.cmake  
  
make

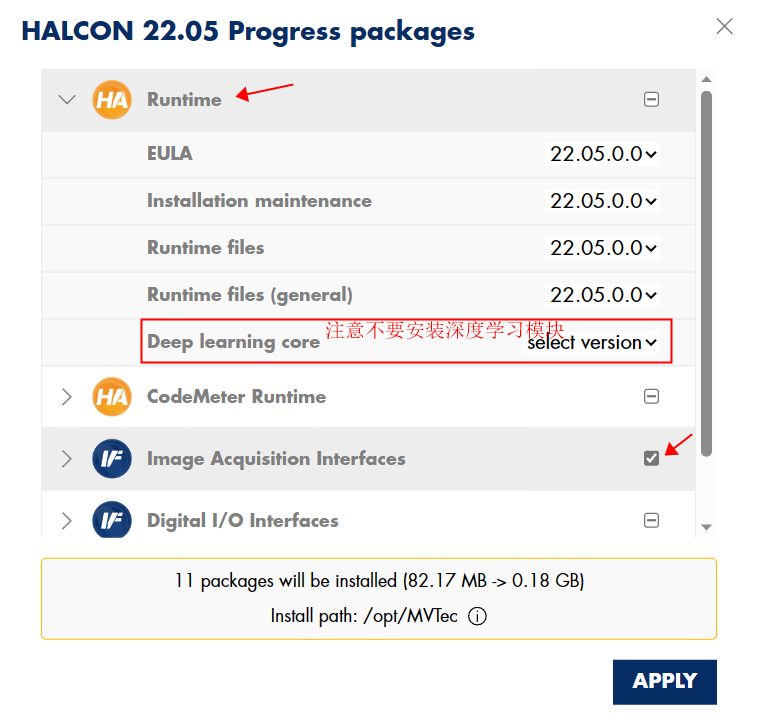


### 2.4 Deployment operations

Deploying the halcon runtime environment in the embedded board is similar to the host computer, and requires configuring the HALCONROOT and LD\_LIBRARY\_PATH environment variables according to the installation location.

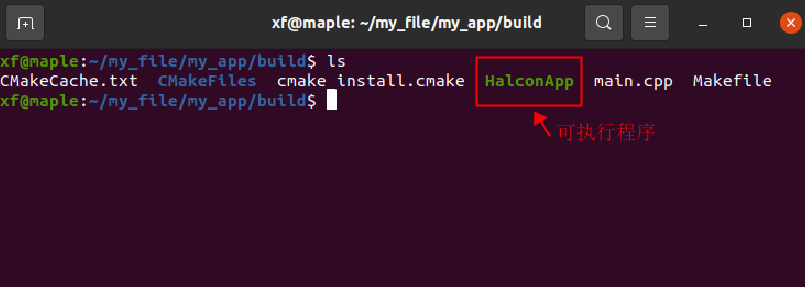
On the development board run . /som -H 192.168.0.232 -n, then you can open the installation interface on the host computer via a browser.





Note: If you are prompted for insufficient runtime space when installing halcon, you can delete the contents of the deep learning module (halcon-22.05.0.0-deep-learning-core-x64-win64\_x64-linux\_aarch64-linux\_armv7a-linux.zip) in the extracted contents, which takes up about 800MB of space. (which takes up about 800MB of space) Delete it.

Copy the compiled and generated executable to the embedded board to run it.



## III. Performance testing

In order to test the running performance of Halcon platform under linux platform, a multi-exposure mapping scenario was designed with the following steps:

1. Connect the Basler camera via GIGE communication and the inclinometer via serial port RS232;

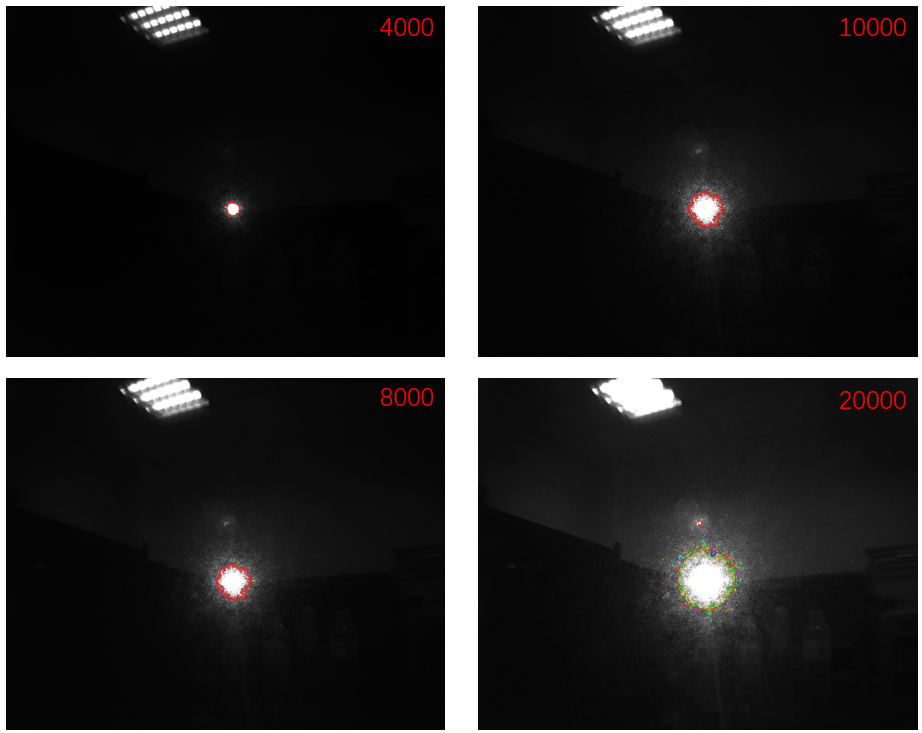
2. Capture camera images through different exposure times, one image for each exposure time, and spot processing;

3. Collect the inclinometer data (1S active out of the number), if it fails, then collect three times until the collection is successful;

4. Inclinometer and spot data fusion;

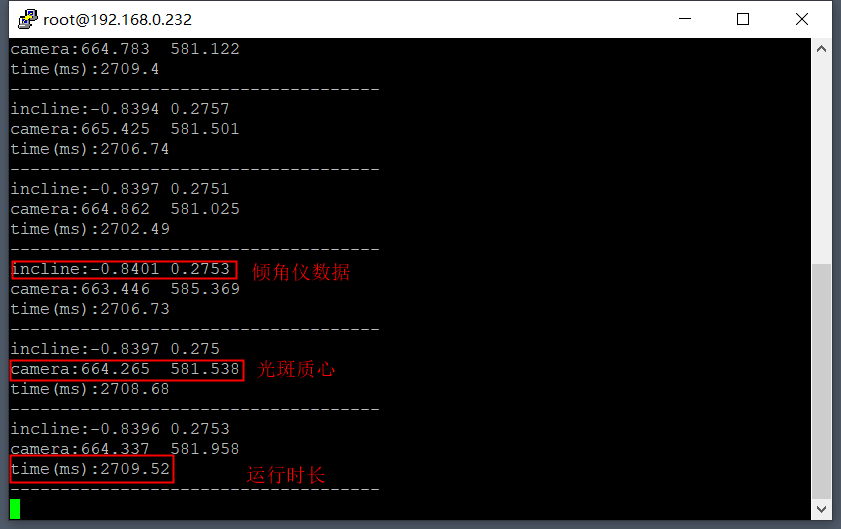
5. Cycle through steps 2-3 above and count the time spent on each cycle step;

By counting the time consumed in the whole process from map acquisition to data fusion, it provides a basis for evaluating the performance.



After testing, the average execution cycle under windows platform is 2.5S, and under Linux platform is 2.7S, which shows that the difference between the two is not much.

### 



## IV. Test conclusions

From the above test results, the following conclusions can be drawn:

1. Based on Halcon tools to develop laser target products (camera + inclinometer) technically feasible, compared with the traditional spot extraction and image processing process, Halcon has the advantages of convenience and speed; hDevelop provides a visual script development tools can facilitate the rapid verification of the image algorithms developers; programming, Halcon provides a code export function to support In terms of programming, Halcon provides code export function and supports multi-threaded concurrent operation, which is easy to combine with traditional development methods.

2. For the given scenarios of multiple exposures, shooting, processing, fusion, etc., Halcon's performance on the selected Mir development board is not much different from that on the Windows PC platform;

3. Test results show that the choice of 4 groups of exposure time to take pictures to extract the spot, it takes about 2.7S; if you choose 1 group of exposure time to take pictures, it takes about 600ms;

Since the whole guiding cycle is >10S, the multiple combinations of laser targets account for 10% to 20% of the exposure time, which is considered to meet the requirements for use.