

## 基于 AM57x 的 GigE 工业相机图像采集案例

## 目 录

|  |    |
|--|----|
| 1 开发环境.....                                | 3  |
| 2 GigE 相机 IP 配置.....                       | 4  |
| 2.1 安装工程源码.....                            | 4  |
| 2.2 GigE 相机 IP 地址的配置 .....                 | 5  |
| 2.3 IpConfigurator 查看不到相机设备解决方案 .....      | 7  |
| 3 PylonViewerApp 例程测试.....                 | 9  |
| 3.1 运行 PylonViewerApp 工具采集图像.....          | 9  |
| 3.2 GigE 相机参数的设置.....                      | 13 |
| 4 Grap 例程测试.....                           | 18 |
| 4.1 安装工程源码.....                            | 18 |
| 4.2 编译 Grap 例程.....                        | 19 |
| 4.3 设置环境变量.....                            | 21 |
| 4.4 Grap 例程测试.....                         | 23 |
| 5 GigEViewApp 测试.....                      | 25 |
| 5.1 安装工程源码.....                            | 25 |
| 5.2 编译 GigEViewApp 工程源码.....               | 26 |
| 5.3 运行 GigEViewApp 例程.....                 | 29 |
| 6 Gige_Egde_Detrction 例程测试 .....           | 31 |
| 6.1 安装、配置工程源码.....                         | 31 |
| 6.2 编译例程.....                              | 33 |
| 6.3 运行例程.....                              | 34 |
| 7 Gige_Egde_Detrction_GSTENC 测试 .....      | 37 |
| 7.1 安装、配置工程源码.....                         | 38 |
| 7.2 编译例程.....                              | 40 |
| 7.3 运行例程.....                              | 41 |
| 8 Gige_Egde_Detrction_GSTENC_RTSP 测试 ..... | 43 |
| 8.1 安装、配置工程源码.....                         | 43 |

8.2 编译例程 ..... 44

8.3 运行例程 ..... 45

更多帮助 ..... 49

## 1 开发环境

### ■ AM57x 开发板（广州创龙 TL5728-EasyEVM）

(本案例所有 Qt 程序在开发板运行时，默认使用 7 英寸的 LCD 显示屏。)

TL5728-EasyEVM 开发板简介：

- 基于 TI AM5728 浮点双 DSPC66x+双 ARM Cortex-A15 工业控制及高性能音视频处理器；
- 多核异构 CPU，集成双核 Cortex-A15、双核 C66x 浮点 DSP、双核 PRU-ICSS、双核 Cortex-M4 IPU、双核 GPU 等处理单元，支持 OpenCL、OpenMP、IPC 多核开发；
- 强劲的视频编解码能力，支持 1 路 1080P60 或 2 路 720P60 或 4 路 720P30 视频硬件编解码，支持 H.265 视频软解码；
- 支持高达 1 路 1080P60 全高清视频输入和 1 路 LCD + 1 路 HDMI 1.4a 输出；
- 双核 PRU-ICSS 工业实时控制子系统，支持 EtherCAT、EtherNet/IP、PROFIBUS 等工业协议；
- 高性能 GPU，双核 SGX5443D 加速器和 GC3202D 图形加速引擎，支持 OpenGL ES2.0；
- 外设接口丰富，集成双千兆网、PCIe、GPMC、USB 2.0、UART、SPI、QSPI、SATA 2.0、I2C、DCAN 等工业控制总线接口，支持极速接口 USB 3.0；
- 应用于工业 PC&HMI、工业机器人、机视觉、医疗影像、电力自动化等领域。



■ GigE 相机      品牌: BASLER      型号: acA640-120gm

■ VLC 多媒体播放器

安装包位于: 光盘资料\Tools\Windows\vlc-2.2.4-win32.exe

■ Ubuntu-14.04.4 64bit

参照《Linux Processor-SDK 安装》文档安装 Linux Processor-SDK 开发环境和交叉编译工具链; 参照《基于 AM57x 的 Linux QT 图形界面开发入门教程》文档搭建 Qt 开发环境。

更多关于 GigeE 工业相机的开发资料位于: 光盘资料“Demo\GigE\doc”目录下。

## 2 GigE 相机 IP 配置

### 2.1 安装工程源码

在 Ubuntu 上新建 “/home/tronlong/AM57xx/pylon/” 工作目录, 将光盘资料 “Demo\GigE\src\pylon-5.0.9.10389-x86\_64.tar.gz” 文件拷贝到该目录下。

Host# mkdir -p /home/tronlong/AM57xx/pylon/

```
tronlong@tronlong-virtual-machine:~$ mkdir -p /home/tronlong/AM57xx/pylon/
tronlong@tronlong-virtual-machine:~$ ls /home/tronlong/AM57xx/pylon/
pylon-5.0.9.10389-x86_64.tar.gz
tronlong@tronlong-virtual-machine:~$
```

图 1

在 pylon-5.0.9.10389-x86\_64.tar.gz 文件所在目录, 执行如下指令将其解压当前目录, 解压后生成 pylon-5.0.9.10389-x86\_64 文件夹, 进入新生成的 pylon-5.0.9.10389-x86\_64 文件夹, 解压 pylonSDK-5.0.9.10389-x86\_64.tar.gz 得到 pylon5 文件夹:

Host# tar -xzf pylon-5.0.9.10389-x86\_64.tar.gz -C ./

Host# cd pylon-5.0.9.10389-x86\_64

Host# tar -xzf pylonSDK-5.0.9.10389-x86\_64.tar.gz -C ./

```
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ ls
pylon-5.0.9.10389-x86_64.tar.gz
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ tar -xzf pylon-5.0.9.10389-x86_64.tar.gz -C ./
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ ls
pylon-5.0.9.10389-x86_64  pylon-5.0.9.10389-x86_64.tar.gz
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ cd pylon-5.0.9.10389-x86_64/
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-x86_64$ tar -xzf pylonSDK-5.0.9.10389-x86_64.tar.gz -C ./
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-x86_64$ ls
69-basler-cameras.rules  pylonSDK-5.0.9.10389-x86_64.tar.gz  Samples
INSTALL                  README                               setup-usb.sh
pylon5                   ReleaseNotes.txt
```

图 2

## 2.2 GigE 相机 IP 地址的配置

使用 GigE 相机配套的网线连接到 PC 的网口，另一端连接 GigE 相机网口，把 GigE 相机与 Ubuntu 连接到同网段网络。由于 IpConfigurator 需要对相机独占访问，需要确保没有其他程序访问相机。

执行如下指令，使用 Basler pylon 官方的 IpConfigurator 工具对 GigE 相机 IP 地址进行配置：

Host# cd /home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-x86\_64/pylon5/bin/

Host# ./IpConfigurator

```
tronlong@tronlong-virtual-machine:~$ cd /home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-x86_64/pylon5/bin/
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-x86_64/pylon5/bin$ ls
IpConfigurator          libQt5Core.so.5.6      libQt5XcbQpa.so.5
libPylonQtBase.so       libQt5Core.so.5.6.2    libQt5XcbQpa.so.5.6
libPylonQtBase.so.1     libQt5Gui.so           libQt5XcbQpa.so.5.6.2
libPylonQtBase.so.1.0   libQt5Gui.so.5         libQt5Xml.so
libPylonQtBase.so.1.0.0 libQt5Gui.so.5.6       libQt5Xml.so.5
libPylonQtWidgets.so    libQt5Gui.so.5.6.2     libQt5Xml.so.5.6
libPylonQtWidgets.so.1  libQt5Network.so       libQt5Xml.so.5.6.2
libPylonQtWidgets.so.1.0 libQt5Network.so.5     platforms
libPylonQtWidgets.so.1.0.0 libQt5Network.so.5.6   pylon-config
libPylonViewerSdk.so    libQt5Network.so.5.6.2 PylonFirmwareUpdater
libPylonViewerSdk.so.1  libQt5Widgets.so       pylon-setup-env.sh
libPylonViewerSdk.so.1.0 libQt5Widgets.so.5     pylon-start-with-logging
libPylonViewerSdk.so.1.0.0 libQt5Widgets.so.5.6   PylonViewerApp
libQt5Core.so           libQt5Widgets.so.5.6.2 StartPylonViewerWithLogging.sh
libQt5Core.so.5         libQt5XcbQpa.so
```

图 3



打开 IpConfigurator 工具后，将列出所有连接的摄像机以及它的一些信息，如下图所示。如果没有信息显示，尝试点击右边的 Refresh 刷新按钮，如果依然没有信息显示，请根据以下“IpConfigurator 查看不到相机设备解决方案”方法解决。点击选中相机设备，即可在下方看到相机的信息，GigE 相机提供支持 3 种方式对其 IP 地址进行配置：静态 IP 地址、动态 IP 地址、DHCP 自动分配 IP 地址。

## ■ 静态 IP 地址连接方式

最简单的连接 GigE 相机方法就是通过静态 IP 来实现，使用 GigE 相机配套的网线连接开发板网口与相机网口（直连）。如图所示可以对相机的 IP 地址进行配置，Gateway、DeviceUserID 可填也可不填，配置完成之后点击 Save 保存即可。保存成功之后这个 IP 配置将会永久的写入到相机设备中，新设置完成之后，相机将被复位，新设置将生效。

使用“静态 IP 地址连接方式”测试时，还需要手动设置开发板静态 IP 地址，使得开发板的 IP 地址与相机的 IP 地址在同一个网段。

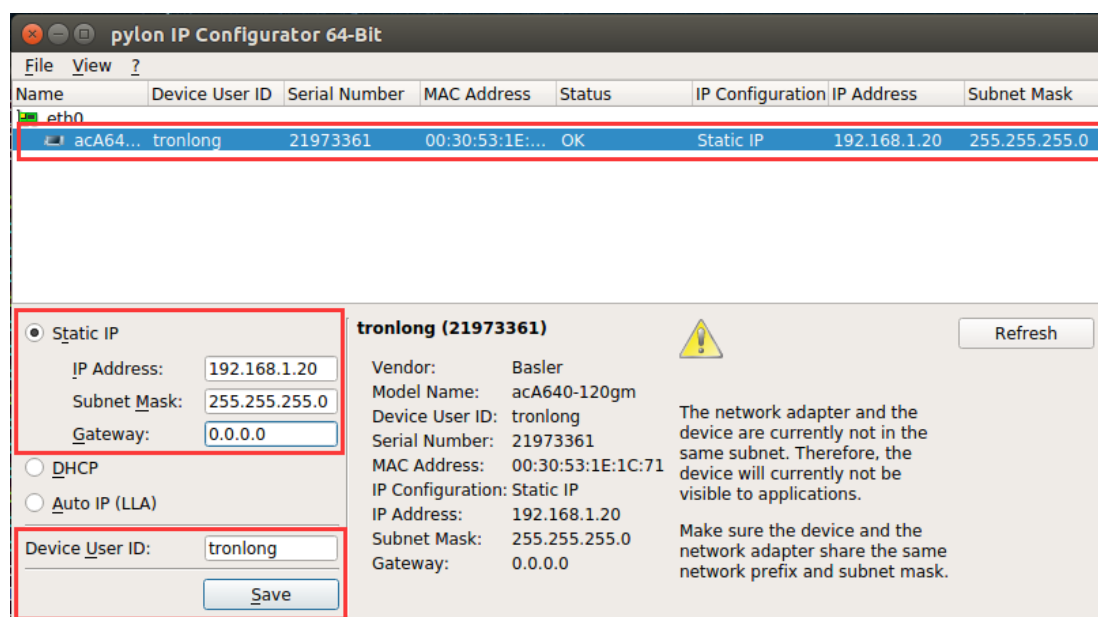


图 4

## ■ DHCP 自动分配 IP 地址连接方式

也可以使用 DHCP 服务器给 GigE 相机分配 IP 地址。分别将开发板和相机通过网线连接到路由器，在 Ubuntu 下按照下图方式设置，使用 DHCP 来分配 IP 地址到 GigE 相机。这样相机和开发板就接入相同网段路由，这种方法避免了手动设置 IP 地址步骤。

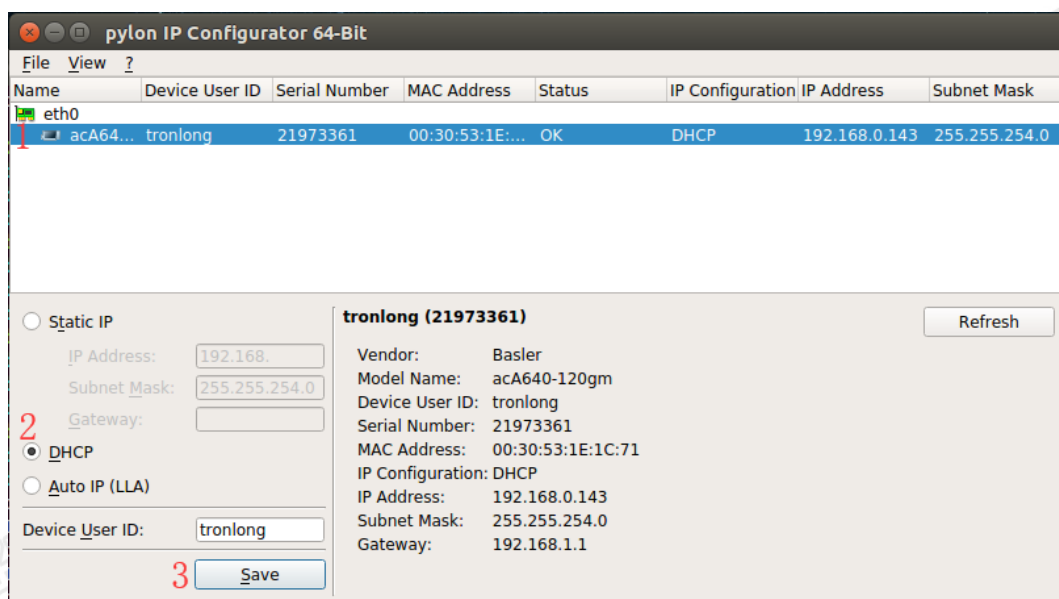


图 5

需要了解更多关于 GigE 相机 IP 配置方面的信息, 可以查看光盘资料“\Demo\GigE\doc”目录下的 INSTALL 文档和 README 文档。

### 2.3 IpConfigurator 查看不到相机设备解决方案

遇到 IpConfigurator 查看不到相机设备, 可关闭防火墙以及关闭内核中的反向路径过滤"(Reverse-path filtering)"来解决。

#### ■ 关闭防火墙

必须禁用相机连接的网络适配器的防火墙, 否则, 设备发送和接收数据可能无法正常工作。Ubuntu 下执行如下指令查看防火墙状态, inactive 表示禁用状态, active 表示开启状态。如果防火墙是开启的, 则执行“ufw disable”指令关闭防火墙。

```
Host# sudo ufw status
```

```
Host# sudo ufw disable
```

```
tronlong@tronlong-virtual-machine:~$ sudo ufw status
Status: inactive
tronlong@tronlong-virtual-machine:~$ sudo ufw disable
Firewall stopped and disabled on system startup
tronlong@tronlong-virtual-machine:~$
```

图 6

## ■ 关闭内核中的反向路径过滤"(Reverse-path filtering)"

关闭了防火墙后，刷新 IP 配置器如果仍然看不到相机设备，可能是内核中的反向路径过滤阻止了 IP 配置器检测相机。执行如下指令，检查是否打开过滤：

Host# sysctl -a 2>/dev/null | grep '\.rp\_filter'

```
tronlong@tronlong-virtual-machine:~$ sysctl -a 2>/dev/null | grep '\.rp_filter'
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.eth0.rp_filter = 1
net.ipv4.conf.lo.rp_filter = 1
tronlong@tronlong-virtual-machine:~$
```

图 7

- "net.ipv4.conf.all.rp\_filter"是一个全局开关，必须关闭；
- "net.ipv4.conf.eth0.rp\_filter"指示是否需要激活指定的网络适配器的过滤；
- eth0 是相机连接到的网络适配器。

执行如下指令关闭全部开关和过滤：

Host# sudo sysctl net.ipv4.conf.all.rp\_filter=0

Host# sudo sysctl net.ipv4.conf.eth0.rp\_filter=0

```
tronlong@tronlong-virtual-machine:~$ sudo sysctl net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.all.rp_filter = 0
tronlong@tronlong-virtual-machine:~$ sudo sysctl net.ipv4.conf.eth0.rp_filter=0
net.ipv4.conf.eth0.rp_filter = 0
tronlong@tronlong-virtual-machine:~$
```

图 8

上面操作只是临时关闭过滤器，如果需要永久关闭过滤器，可以在"/etc/sysctl.conf"文件末尾添加如下代码，将其关闭。

Host# sudo gedit /etc/sysctl.conf



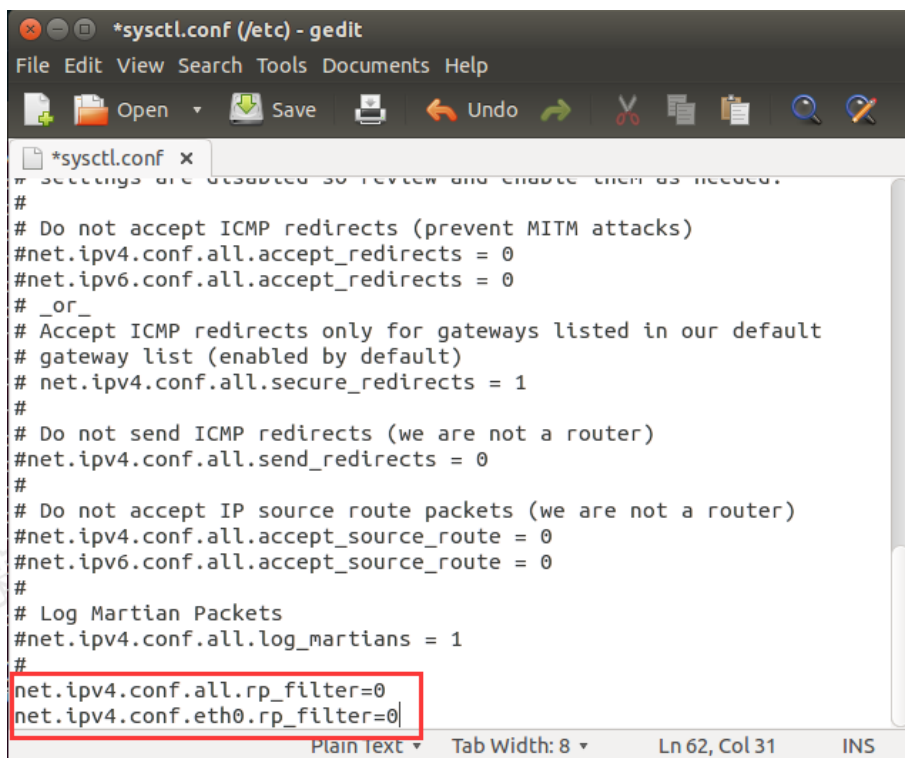


图 9

### 3 PylonViewerApp 例程测试

#### 3.1 运行 PylonViewerApp 工具采集图像

PylonViewerApp 是一个由 Basler pylon 官方提供，基于 Qt5.6 编写的可视化相机视频采集监视程序，执行如下指令运行 PylonViewerApp，如下图所示：

```
Host# cd /home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-x86_64/pylon5/bin
```

```
Host# ./PylonViewerApp
```

```
tronlong@tronlong-virtual-machine:~$ cd /home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-x86_64/pylon5/bin
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-x86_64/pylon5/bin$ ls
IpConfigurator          libQt5Core.so.5.6      libQt5XcbQpa.so.5
libPylonQtBase.so       libQt5Core.so.5.6.2   libQt5XcbQpa.so.5.6
libPylonQtBase.so.1     libQt5Gui.so          libQt5XcbQpa.so.5.6.2
libPylonQtBase.so.1.0   libQt5Gui.so.5        libQt5Xml.so
libPylonQtBase.so.1.0.0 libQt5Gui.so.5.6      libQt5Xml.so.5
libPylonQtWidgets.so    libQt5Gui.so.5.6.2    libQt5Xml.so.5.6
libPylonQtWidgets.so.1  libQt5Network.so      libQt5Xml.so.5.6.2
libPylonQtWidgets.so.1.0 libQt5Network.so.5    platforms
libPylonQtWidgets.so.1.0.0 libQt5Network.so.5.6  pylon-config
libPylonViewerSdk.so    libQt5Network.so.5.6.2 PylonFirmwareUpdater
libPylonViewerSdk.so.1  libQt5Widgets.so      pylon-setup-env.sh
libPylonViewerSdk.so.1.0 libQt5Widgets.so.5    pylon-start-with-logging
libPylonViewerSdk.so.1.0.0 libQt5Widgets.so.5.6  PylonViewerApp
libQt5Core.so           libQt5Widgets.so.5.6.2 StartPylonViewerWithLogging.sh
libQt5Core.so.5         libQt5XcbQpa.so
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-x86_64/pylon5/bin$ ./PylonViewerApp
```

图 10

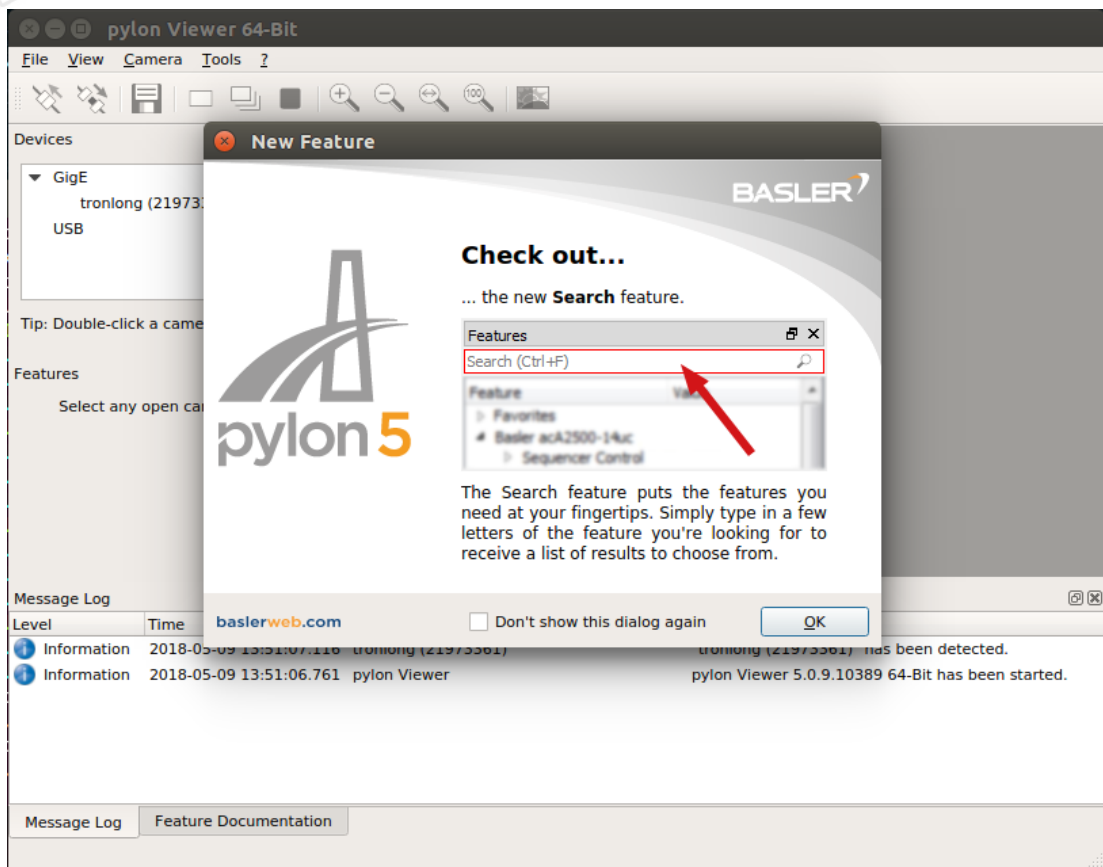


图 11

可以在 Devices 选项下看到当前连接的相机设备列表，由下图可见 GigE 相机已经连接成功：

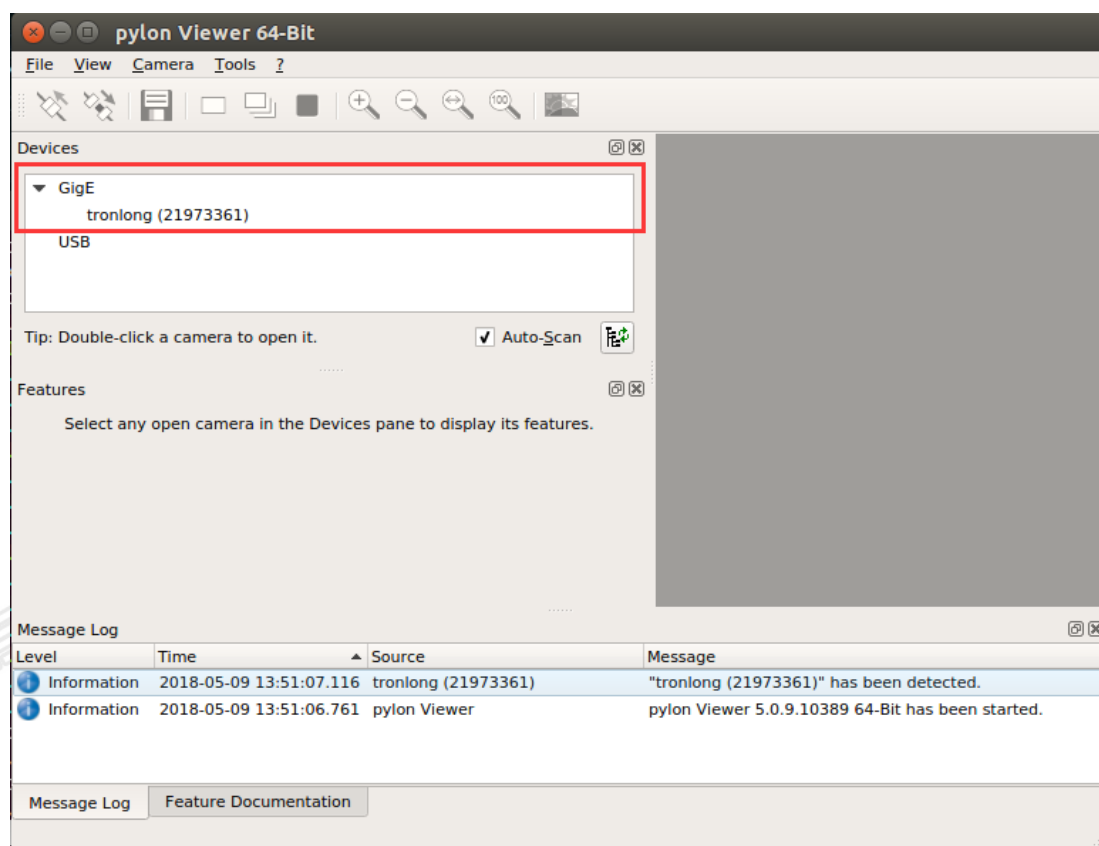


图 12

右键点击“tronlong(21973361)”相机设备，弹出菜单中选择 Open 选项打开设备，打印信息如下图所示：

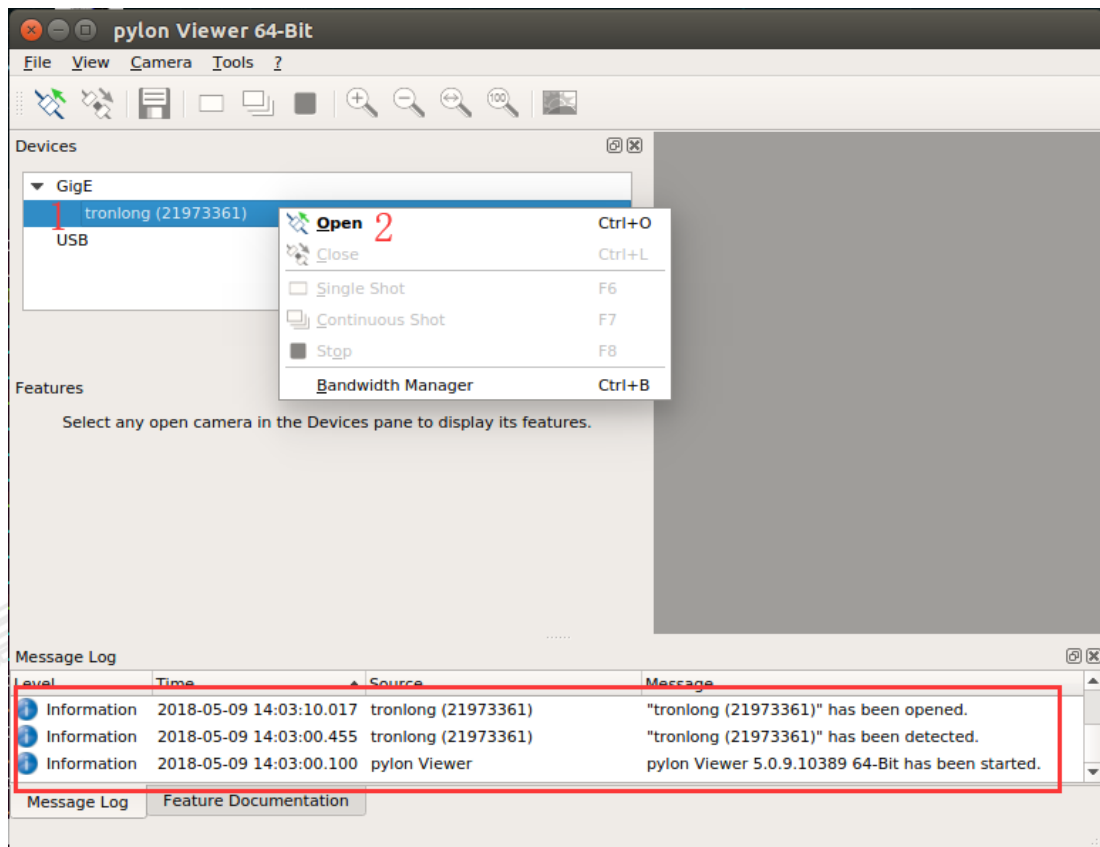




图 13

点击选项栏的单次采集  按钮或者连续采集  按钮，即可查看采集到的图像。

通过调整相机的调焦旋钮，可以采集到更加清晰的图像，如下图所示：

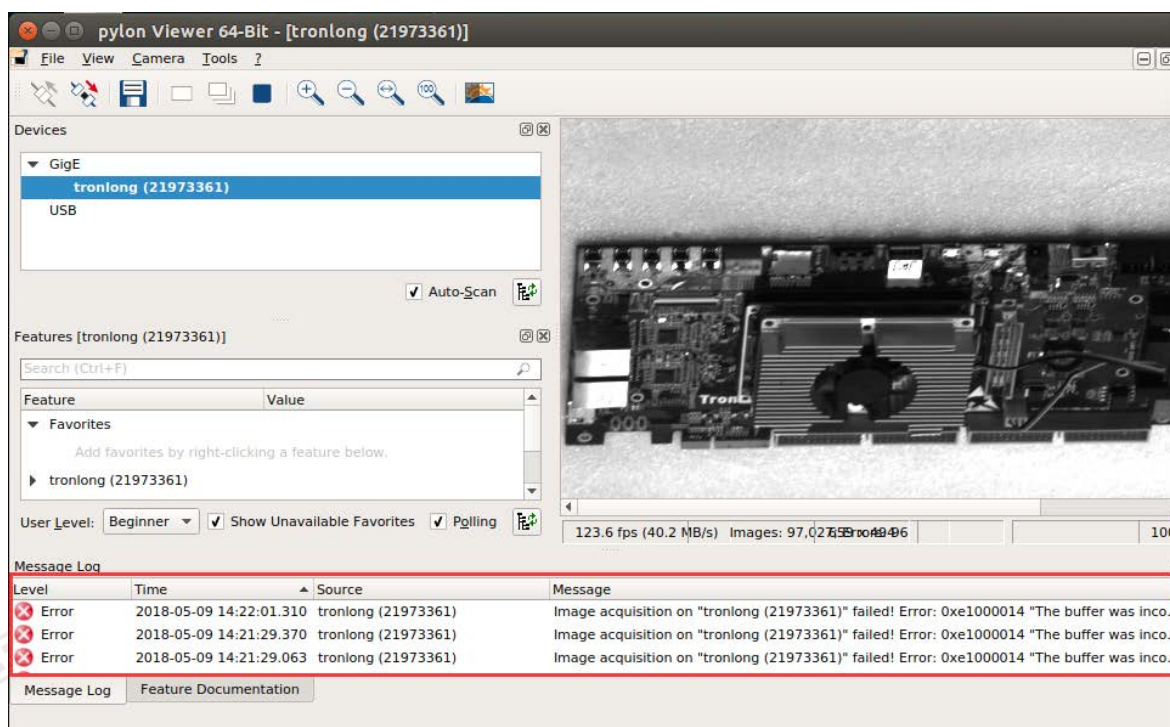

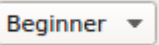



图 14

### 3.2 GigE 相机参数的设置

对于图像采集过程中，出现上图红框中的错误提示，需要按照如下方法设置“Inter-Packet Delay”参数和“Packet Size”参数。

点击选项栏中的  Stop 按钮，停止图像采集。在 View 菜单中勾选 Features[tronlong(21973361)]选项，然后点击  下拉按钮，选中  Guru 选项。可以看到 Features[tronlong(21973361)]功能特性配置界面下，tronlong(21973361)相机设备配置项如下图所示：



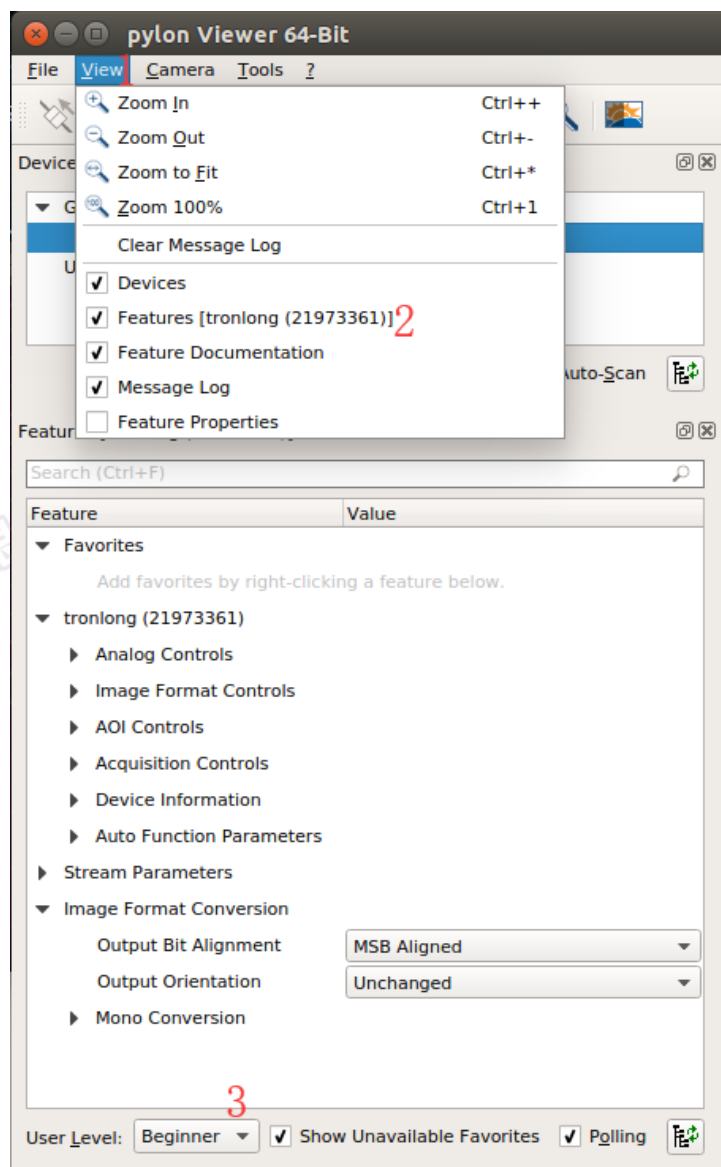


图 15

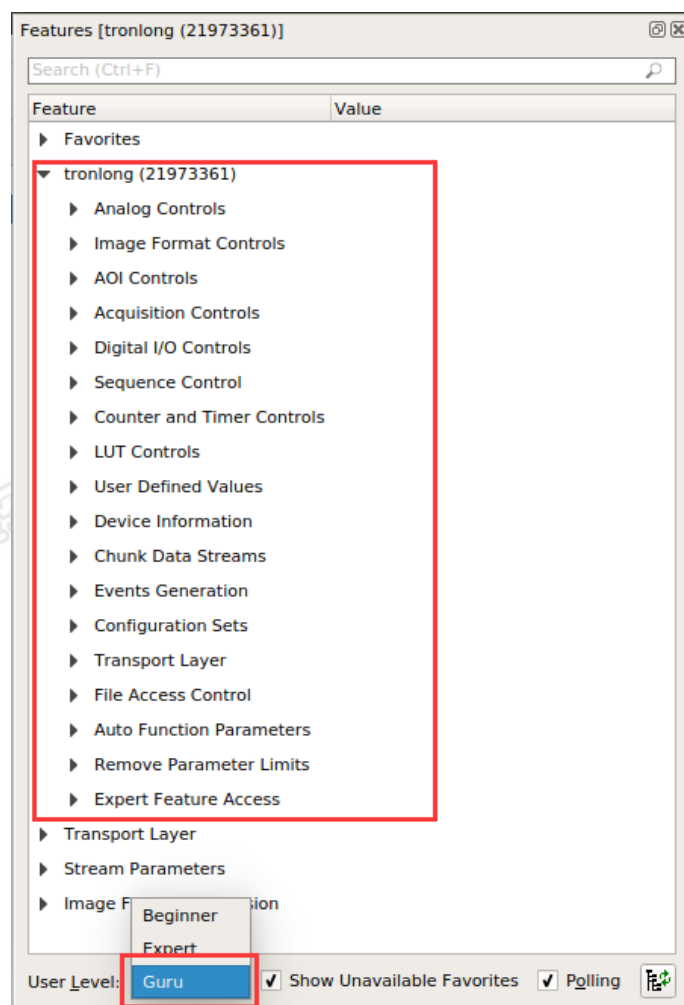


图 16

点击 Acquisition Controls 选项, 将其下的“Exposure Time(Abs)[us]”参数设置为 10000;  
点击 Transport Layer 选项, 将其下的“Packet Size”参数设置为 1500, “Inter-Packet Delay”  
参数设置为 9000:

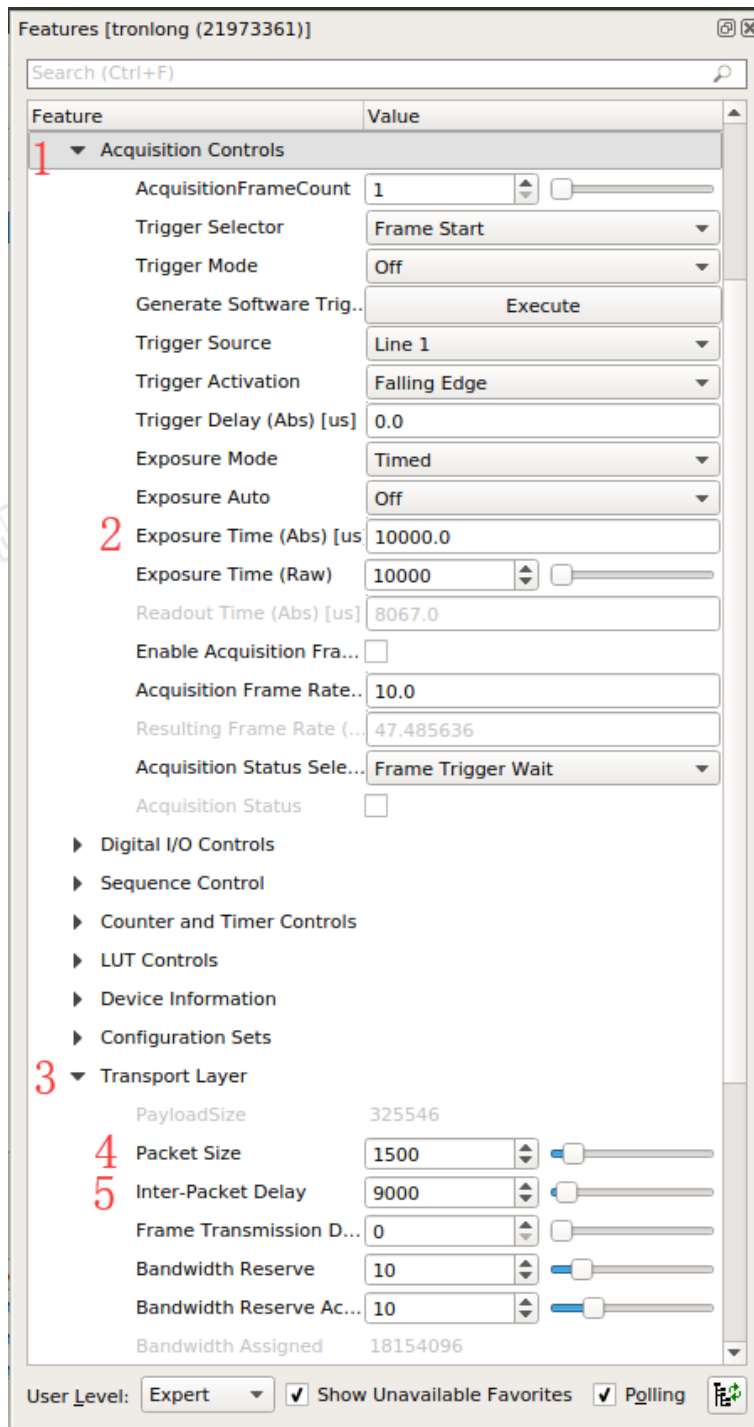


图 17

在 Ubuntu 下按“Ctrl+Alt+T”打开 Terminal 命令行终端，执行如下指令，将 Stream Parameters 选项下的“Socket Buffer Size”参数值修改为 2048:

```
Host# sudo sysctl net.core.rmem_max=2097152
```

```
tronlong@tronlong-virtual-machine:~$ sudo sysctl net.core.rmem_max=2097152
net.core.rmem_max = 2097152
tronlong@tronlong-virtual-machine:~$
```

图 18

修改完成后，可以查看到 Stream Parameters 选项下的“Socket Buffer Size”参数值已修改为 2048，如下图所示：

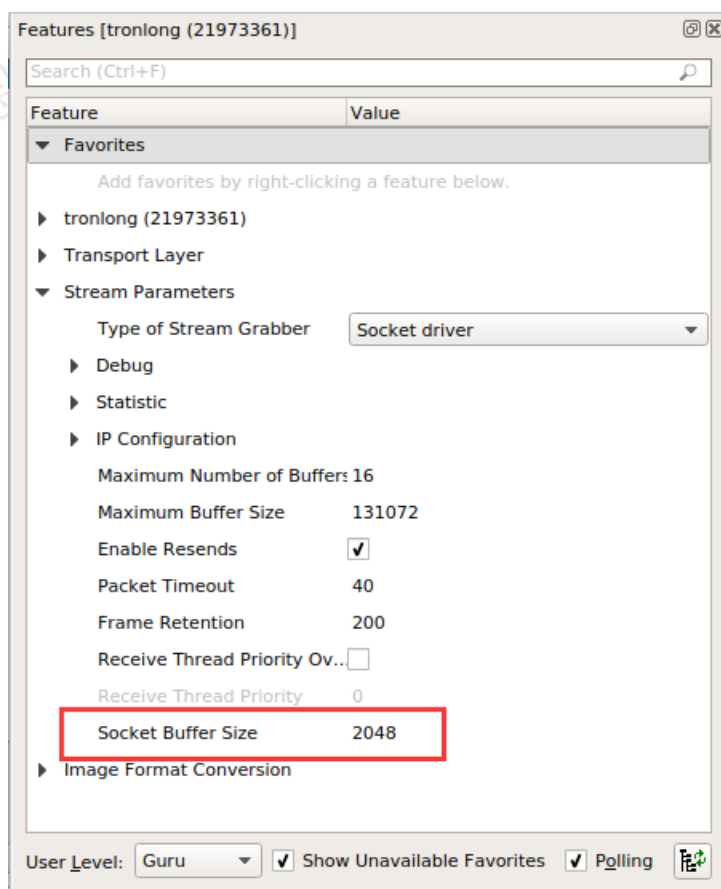



图 19

以上步骤设置的参数将会写入到相机设备中保存。

如果需要恢复初始参数值，先点击  按钮停止图像采集，打开 Configuration Sets 配置选项，将 Configuration Set Selector 选项参数设置为“[Default Configuration Set]”，将 User Set Load 选项参数设置为“Execute”，即可恢复默认设置。

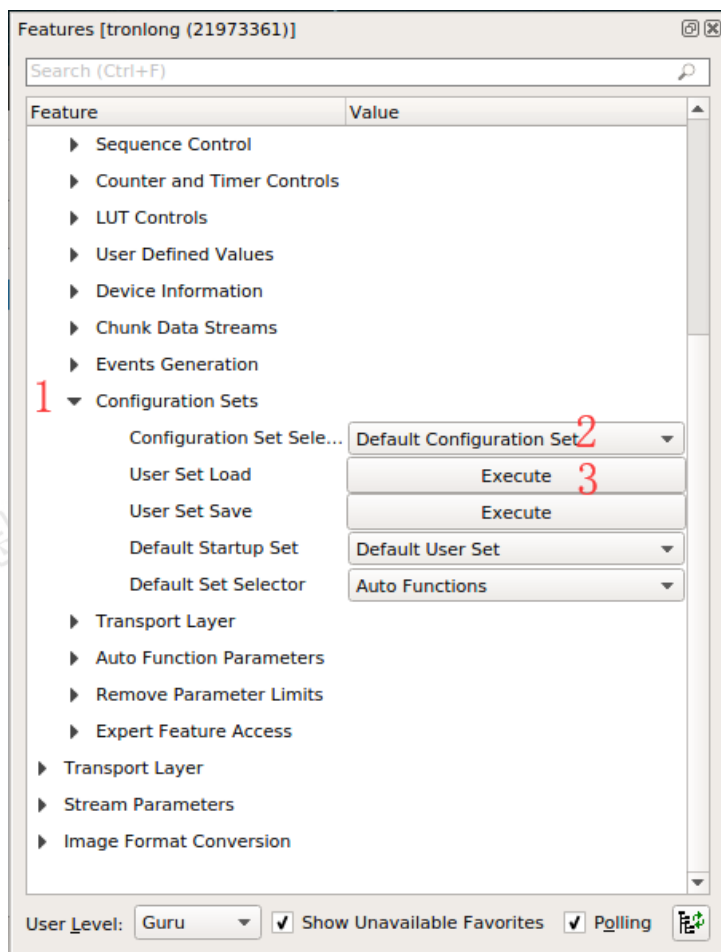


图 20

其他关于的 GigE 相机的调试方法可以参考“home/tronlong/pylon/AM57x/pylon/pylon-5.0.9.10389-x86\_64/”目录下的 README、INSTALL 文档。

```
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-x86_64$ pwd
/home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-x86_64
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-x86_64$ ls
69-basler-cameras.rules  pylonSDK-5.0.9.10389-x86_64.tar.gz  Samples
INSTALL                  README                               setup-usb.sh
pylon                    releaseNotes.txt
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-x86_64$
```

图 21

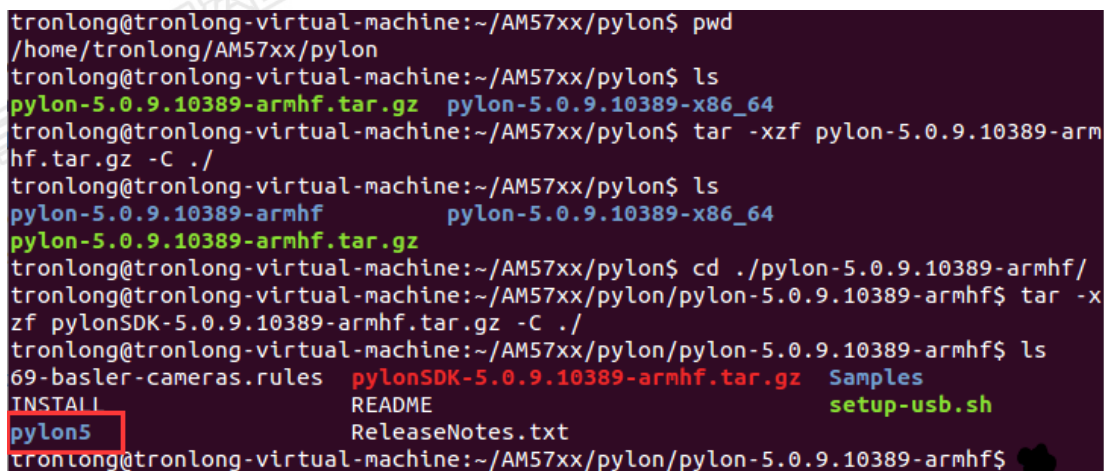
## 4 Grap 例程测试

### 4.1 安装工程源码



将光盘资料“Demo\GigE\src\pylon-5.0.9.10389-armhf.tar.gz”GigE 相机开发源码包拷贝到 Ubuntu 下的“/home/tronlong/AM57xx/pylon/”工作目录。在 pylon-5.0.9.10389-armhf.tar.gz 源码包所在目录执行如下指令将其解压得到 pylon-5.0.9.10389-armhf 目录，进入该目录解压 pylonSDK-5.0.9.10389-armhf.tar.gz 得到 pylon5 目录：

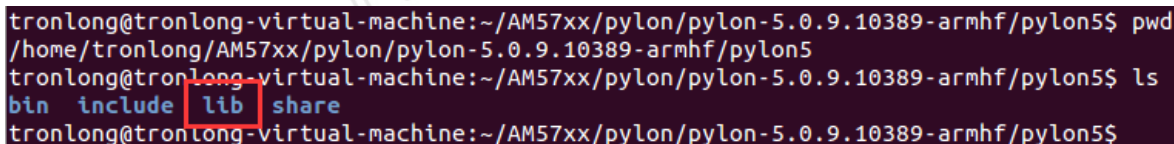
```
Host# tar -xzf pylon-5.0.9.10389-armhf.tar.gz -C ./
Host# cd ./pylon-5.0.9.10389-armhf/
Host# tar -xzf pylonSDK-5.0.9.10389-armhf.tar.gz -C ./
```



```
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ pwd
/home/tronlong/AM57xx/pylon
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ ls
pylon-5.0.9.10389-armhf.tar.gz  pylon-5.0.9.10389-x86_64
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ tar -xzf pylon-5.0.9.10389-armhf.tar.gz -C ./
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ ls
pylon-5.0.9.10389-armhf  pylon-5.0.9.10389-x86_64
pylon-5.0.9.10389-armhf.tar.gz
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ cd ./pylon-5.0.9.10389-armhf/
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-armhf$ tar -xzf pylonSDK-5.0.9.10389-armhf.tar.gz -C ./
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-armhf$ ls
69-basler-cameras.rules  pylonSDK-5.0.9.10389-armhf.tar.gz  Samples
INSTALL                  README                               setup-usb.sh
pylon5                   ReleaseNotes.txt
```

图 22

将 pylon5 目录下的 lib 库文件夹拷贝到 AM57x 开发板的文件系统“/home/root”目录下：



```
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5$ pwd
/home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5$ ls
bin  include  lib  share
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5$
```

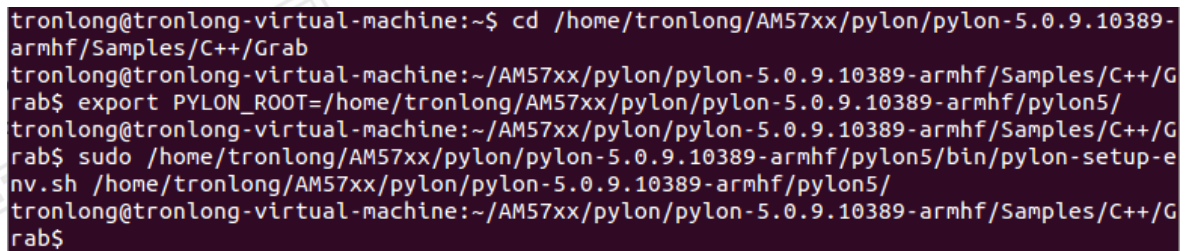
图 23

## 4.2 编译 Grap 例程

进入“/home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-armhf/Samples/C++/Grab”目录，Samples 目录主要包含了一些 Basler pylon 官方提供的基于“C/C++”的示例例程，包括 Grap 例程。执行如下指令设置环境变量，导出 pylon 所在目录到环境变量并执行环境变量初始

化脚本:

```
Host# cd /home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-armhf/Samples/C++/Grab
Host# export PYLON_ROOT=/home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-armhf/pyl
on5/
Host# sudo /home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5/bin/pylon-se
tup-env.sh /home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5/
```



```
tronlong@tronlong-virtual-machine:~$ cd /home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-
armhf/Samples/C++/Grab
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-armhf/Samples/C++/G
rab$ export PYLON_ROOT=/home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5/
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-armhf/Samples/C++/G
rab$ sudo /home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5/bin/pylon-setu
p-env.sh /home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5/
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-armhf/Samples/C++/G
rab$
```

图 24

执行如下指令打开 Makefile 文件, 在文件对应位置添加如下代码, 指定对应版本 Linux Processor-SDK 安装包交叉编译工具路径:

```
Host# gedit Makefile
```

➤ 添加代码如下:

```
CXX :=/home/tronlong/ti-processor-sdk-linux-am57xx-evm-03.01.00.06/linux-devkit/sysroots/
x86_64-arago-linux/usr/bin/arm-linux-gnueabi-g++
LD := $(CXX)
```

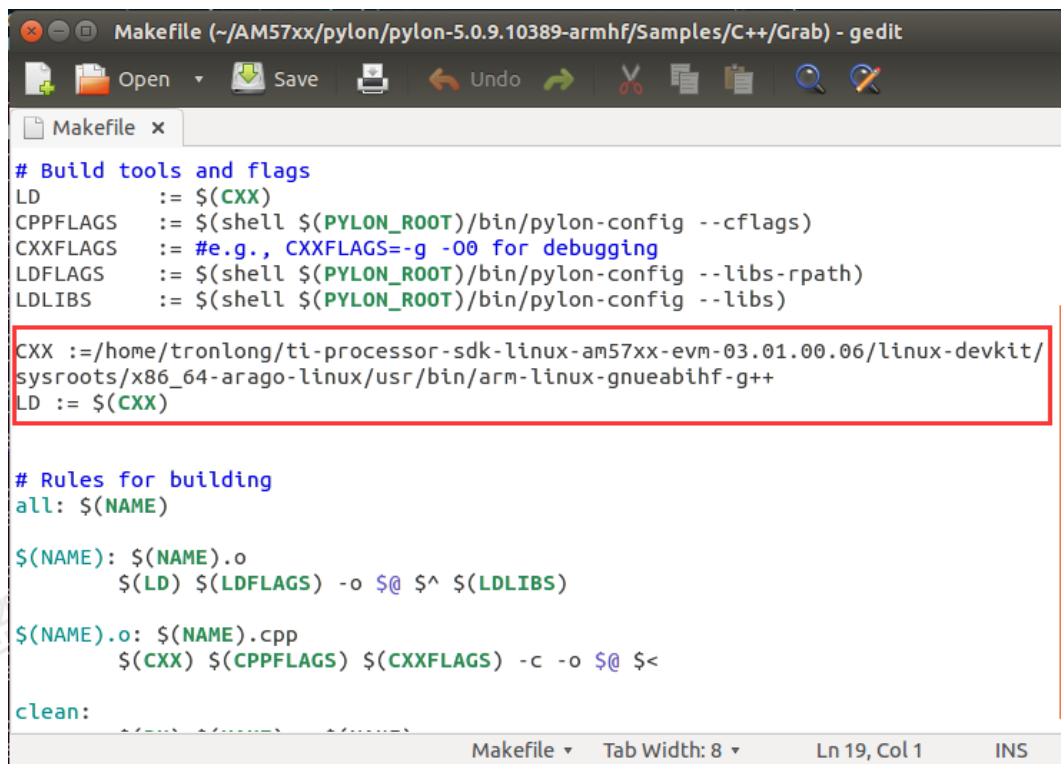


图 25

在 Grab 目录下执行 make 指令编译得到 Grab 可执行文件，如下图所示：

Host# make

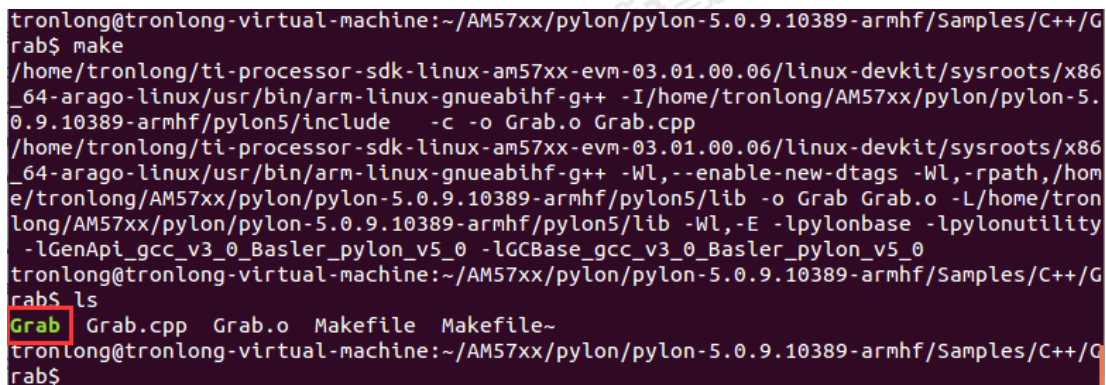


图 26

将编译生成的 Grab 可执行文件拷贝到开发板文件系统“/home/root/”目录下。

#### 4.3 设置环境变量

以下步骤主要设置开发板文件系统下的环境变量，根据如下步骤设置好的环境变量后，续例程的测试都会基于这样的环境变量运行。

将“/home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5”目录下的 lib 库文件夹拷贝到 AM57x 开发板的文件系统“/home/root”目录下：

```
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5$ pwd
/home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5$ ls
bin  include  lib  share
tronlong@tronlong-virtual-machine:~/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5$
```

图 27

开发板上电从 SD 系统卡启动，执行如下指令打开系统配置文件：

Target# vi /etc/profile

```
root@am57xx-evm:~# pwd
/home/root
root@am57xx-evm:~# ls
Grab  lib
root@am57xx-evm:~# vi /etc/profile
```

图 28

在打开的系统配置文件末尾添加如下代码，将 lib 库路径导出到系统环境变量当中，代码中 lib 库文件所在路径应以实际情况为准：

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/root/lib/
```

```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

PATH="/usr/local/bin:/usr/bin:/bin"
EDITOR="vi" # needed for packages like cron, git-commit
test -z "$TERM" && TERM="vt100" # Basic terminal capab. For screen etc.

if [ ! -e /etc/localtime -a ! -e /etc/TZ ]; then
    TZ="UTC" # Time Zone. Look at http://theory.uwinnipeg.ca/gnu/glibc/lib
    # for an explanation of how to set this to your local timezon
    export TZ
fi

if [ "$HOME" = "/home/root" ]; then
    PATH=$PATH:/usr/local/sbin:/usr/sbin:/sbin
fi

if [ "$PS1" ]; then
    # works for bash and ash (no other shells known to be in use here)
    PS1='\u@\h:\w\$'
fi

if [ -d /etc/profile.d ]; then
    for i in /etc/profile.d/*.sh ; do
        if [ -f $i -a -r $i ]; then
            . $i
        fi
    done
    unset i
fi

if [ -x /usr/bin/resize ]; then
    # Make sure we are on a serial console (i.e. the device used starts with /dev/tty),
    # otherwise we confuse e.g. the eclipse launcher which tries to use ssh
    test `tty | cut -c1-8` = "/dev/tty" && resize >/dev/null
fi

export PATH PS1 OPIEDIR QPEDIR QTDIR EDITOR TERM

omask 022
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/root/lib/

~
```

图 29

#### 4.4 Grap 例程测试

##### ■ 静态 IP 测试方法

当 GigE 相机使用“静态 IP 地址”配置模式时，需要按照如下方法连接硬件、设置开发板静态 IP 地址。

使用 GigE 相机配套的网线连接到开发板 RGMII ETH0 网络接口（eth0），另一端连接 GigE 相机网口，断开开发板其他网络连接，接上 GigE 相机电源。

由于前面步骤中已将 GigE 相机静态 IP 地址设置为 192.168.1.20，执行如下命令，设置开发板的 IP 地址与 GigE 相机的 IP 地址在同一个网段。

Target# ifconfig eth0 192.168.1.10

Target# ifconfig eth0 down

Target# ifconfig eth0 up

开发板 IP 地址修改完成之后，可以测试一下开发板与 GigE 相机是否能够 ping 通：

Target# ping 192.168.1.20



```
root@am57xx-evm:~# ifconfig eth0 192.168.1.10
root@am57xx-evm:~# ifconfig eth0 down
root@am57xx-evm:~# ifconfig eth0 up
[ 372.624718] net eth0: initializing cpsw version 1.15 (0)
[ 372.715140] net eth0: phy found : id is : 0x221622
[ 372.725006] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
root@am57xx-evm:~# [ 376.721351] cpsw 48484000.ethernet eth0: Link is up - 1Gbps/Full - flow
control rx/tx
[ 376.729252] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready

root@am57xx-evm:~# ping 192.168.1.20
PING 192.168.1.20 (192.168.1.20): 56 data bytes
64 bytes from 192.168.1.20: seq=0 ttl=255 time=0.390 ms
64 bytes from 192.168.1.20: seq=1 ttl=255 time=0.228 ms
64 bytes from 192.168.1.20: seq=2 ttl=255 time=0.223 ms
64 bytes from 192.168.1.20: seq=3 ttl=255 time=0.230 ms
^C
--- 192.168.1.20 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.223/0.267/0.390 ms
root@am57xx-evm:~#
```

图 30

### ■ 动态 IP 测试方法

使用“DHCP 自动分配 IP 地址”设置 GigE 相机 IP 地址，并将开发板和相机通过网线连接到路由器，使得开发板和相机由路由自动分配同一网段的 IP 地址。

在 Grab 可执行文件所在路径，执行如下命令赋予可执行权限后执行该程序。该程序执行后将采集 10 次相机图像数据，并将图像的像素宽度、像素高度以及第一个像素点的灰度值打印出来，如下图所示：

**Target#**      chmod 777 Grab

**Target#**      ./Grab

```
root@am57xx-evm:~# pwd
/home/root
root@am57xx-evm:~# ls
Grab          genicam_xml_cache  lib
root@am57xx-evm:~# chmod 777 Grab
root@am57xx-evm:~# ./Grab
Using device acA640-120gm
SizeX: 659
SizeY: 494
Gray value of first pixel: 61

SizeX: 659
SizeY: 494
Gray value of first pixel: 62

SizeX: 659
SizeY: 494
Gray value of first pixel: 99

SizeX: 659
SizeY: 494
Gray value of first pixel: 98

SizeX: 659
SizeY: 494
Gray value of first pixel: 97

SizeX: 659
SizeY: 494
Gray value of first pixel: 96

SizeX: 659
SizeY: 494
Gray value of first pixel: 98

SizeX: 659
SizeY: 494
Gray value of first pixel: 124
```

图 31

## 5 GigEViewApp 测试

GigEViewApp 例程是一个基于 Qt5.6.2 编写的简单可视化相机视频采集监视程序。

### 5.1 安装工程源码

将光盘资料“Demo\GigE\src\GigEViewApp.tar.gz”压缩文件复制到 Ubuntu 上的“/home/tronlong/AM57xx/pylon/”工作目录，进入 GigEViewApp.tar.gz 文件所在路径，执行如下指令将其解压到当前目录：

Host# tar -xzf GigEViewApp.tar.gz -C ./

```
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ pwd
/home/tronlong/AM57xx/pylon
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ ls
GigEViewApp.tar.gz  pylon-5.0.9.10389-armhf  pylon-5.0.9.10389-x86_64
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ tar -xzf GigEViewApp.tar.gz -C ./
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ ls
GigEViewApp  GigEViewApp.tar.gz  pylon-5.0.9.10389-armhf  pylon-5.0.9.10389-x86_64
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$
```

图 32

## 5.2 编译 GigEViewApp 工程源码

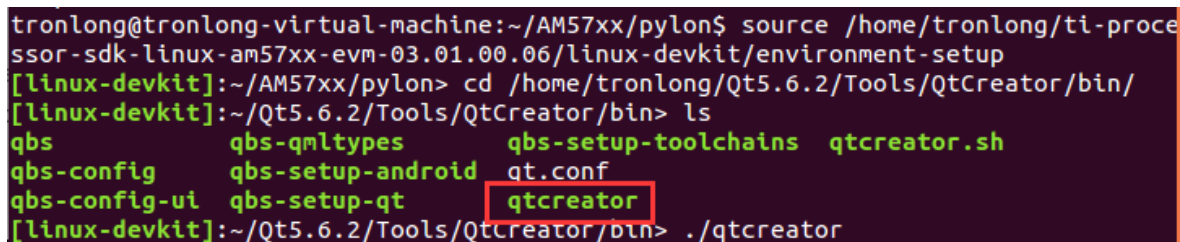
确认已参照《基于 AM57x 的 Linux Qt 图形界面开发入门教程》文档搭建好 Qt 开发环境。

执行如下命令使用对应平台的 Linux Processor-SDK 加载 Qt 等环境变量，加载后，在编译时会使用 Linux Processor-SDK 里面的 ARM 端 Qt 开发环境和运行库。进入 Qt Creator 软件所在路径将其打开：

```
Host# source /home/tronlong/ti-processor-sdk-linux-am57xx-evm-03.01.00.06/linux-devkit/  
/environment-setup
```

```
Host# cd /home/tronlong/Qt5.6.2/Tools/QtCreator/bin/
```

```
Host# ./qtccreator
```



```
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ source /home/tronlong/ti-proce  
ssor-sdk-linux-am57xx-evm-03.01.00.06/linux-devkit/environment-setup  
[linux-devkit]:~/AM57xx/pylon> cd /home/tronlong/Qt5.6.2/Tools/QtCreator/bin/  
[linux-devkit]:~/Qt5.6.2/Tools/QtCreator/bin> ls  
qbs                qbs-qmltypes      qbs-setup-toolchains  qtccreator.sh  
qbs-config          qbs-setup-android  qt.conf  
qbs-config-ui       qbs-setup-qt       qtccreator  
[linux-devkit]:~/Qt5.6.2/Tools/QtCreator/bin> ./qtccreator
```

图 33

点击 Qt Creator 菜单栏的“File->Open File or Project”，弹出对话框中选择“home/tronlong/AM57xx/pylon/GigEViewApp/GigEViewApp.pro”文件所在路径，将其选中并点击 Open 打开：

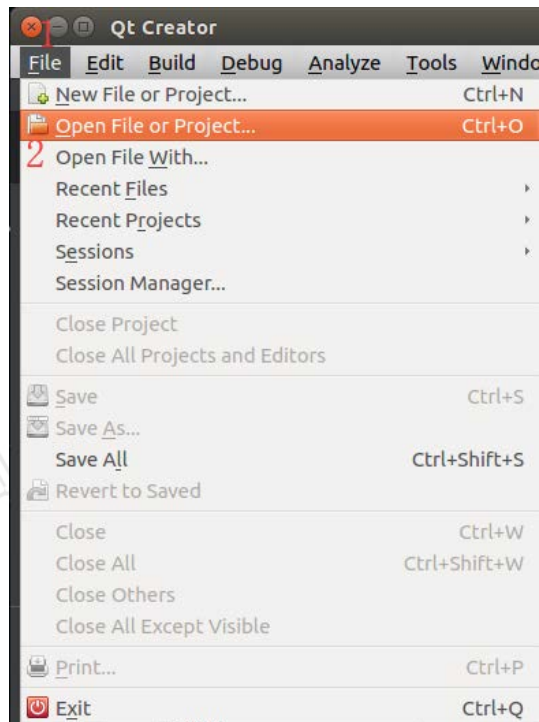


图 34

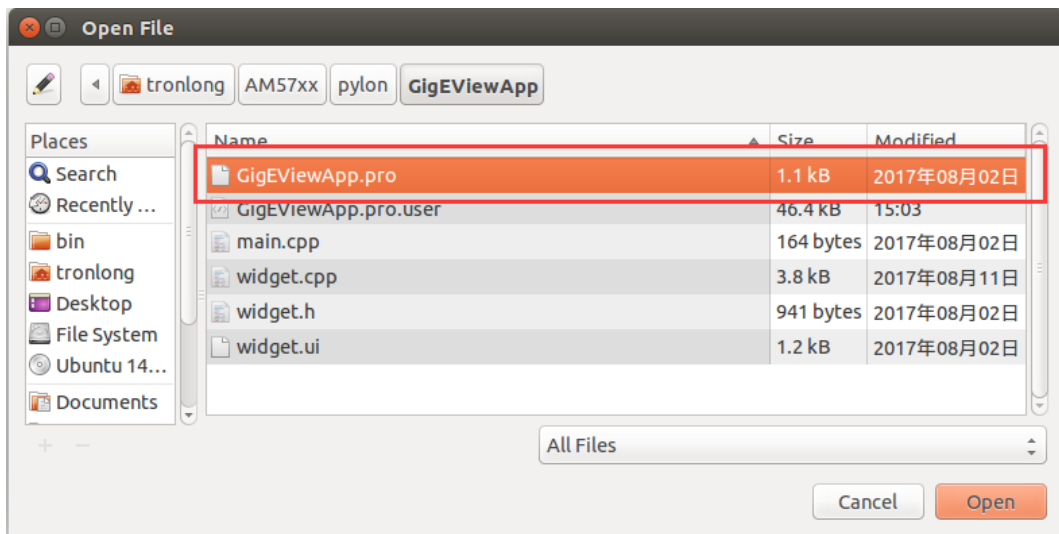


图 35

双击打开 GigEViewApp.pro 文件，按照如下方法，根据实际情况修改下图红框中第三方 GigE 库和头文件所在路径：

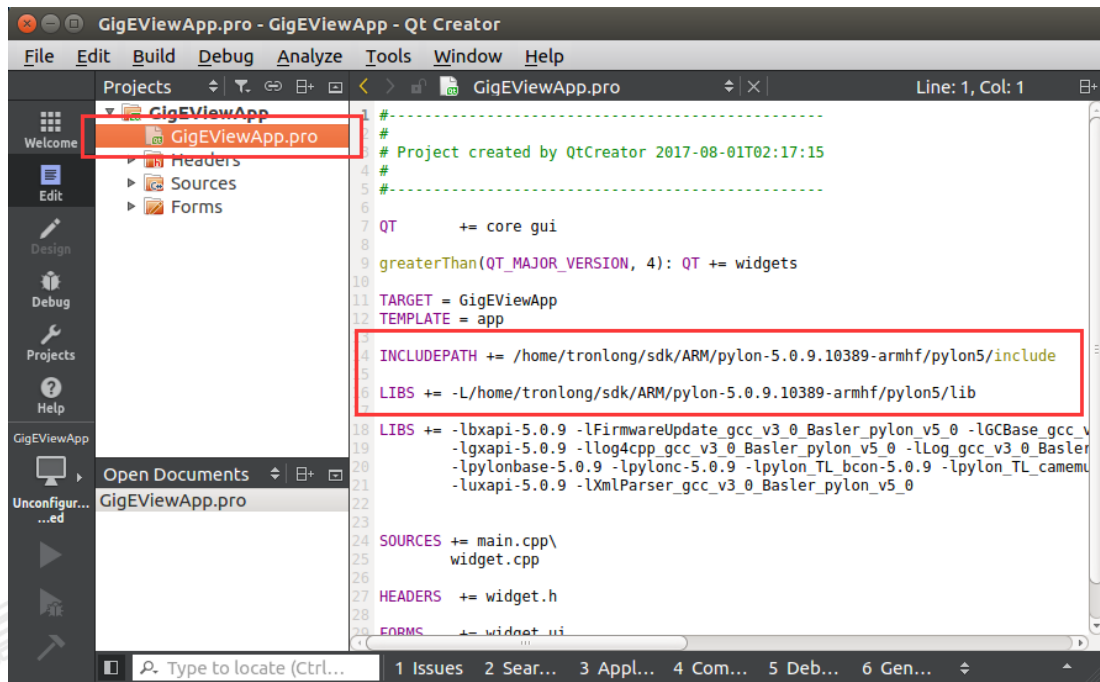


图 36

- 指定 GigE 相机开发源码头文件所在路径

INCLUDEPATH += /home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5/include/

- 指定 GigE 相机开发源码库文件所在路径

LIBS += -L/home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5/lib/

修改完成后内容如下图所示。点击菜单栏"File->Save All"保存修改，再点击菜单栏的"Builed->Run qmake"来使得修改的配置生效。



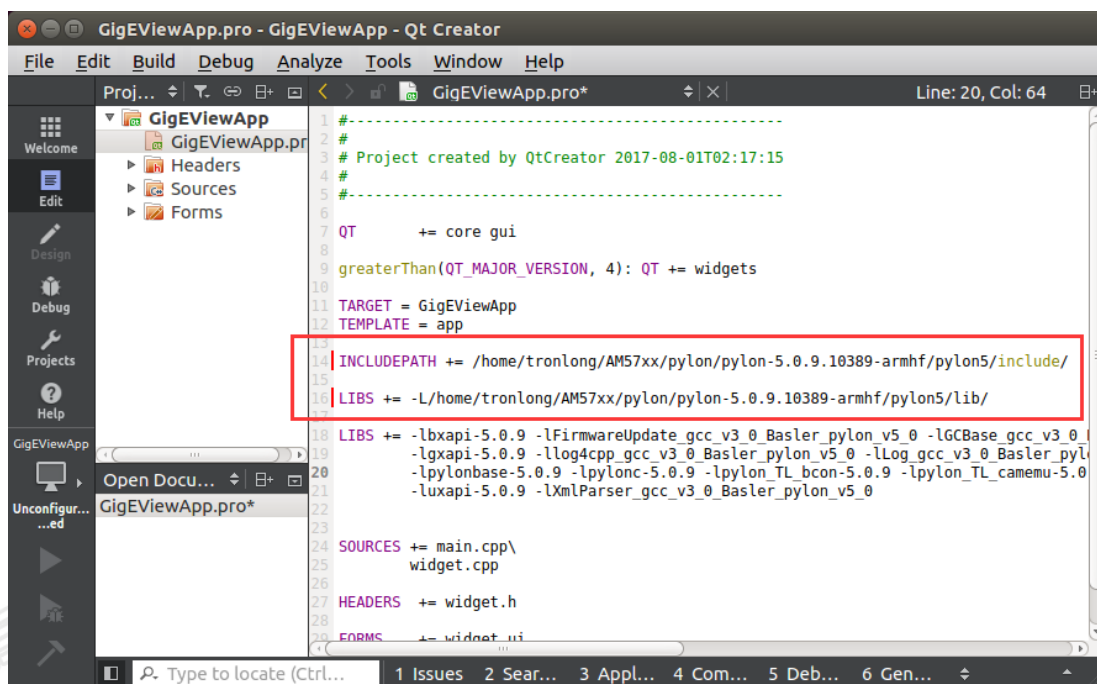



图 37

点击左下角的  按钮即可编译工程，编译完成后在“/home/tronlong/AM57xx/pylon/build-GigEViewApp-AM57xx-Debug/”目录下生成 GigEViewApp 可执行文件，将编译生成的 GigEViewApp 可执行文件拷贝到开发板文件系统“/home/root/”目录下。

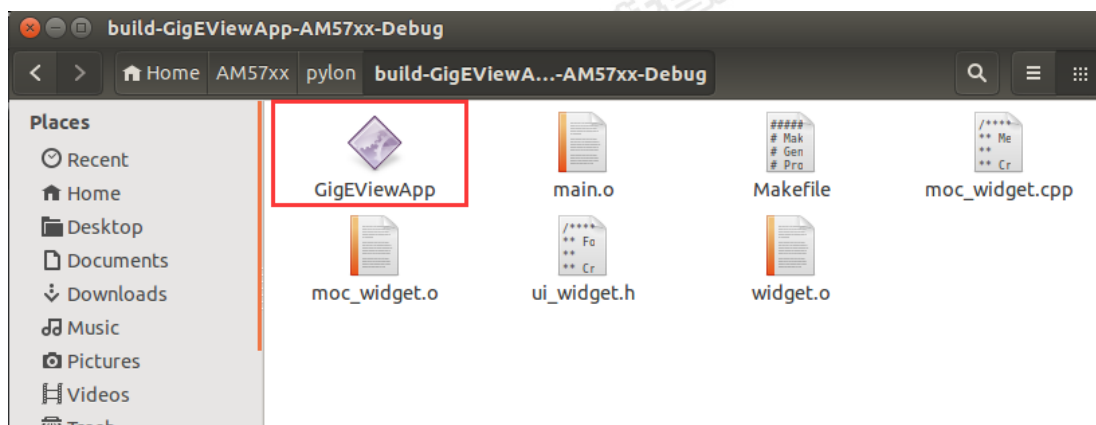


图 38

### 5.3 运行 GigEViewApp 例程

开发板连接触摸屏，上电正常启动进入文件系统，在 GigEViewApp 文件所在路径执行如下指令运行，程序运行后开发板 LCD 显示屏效果图如下：：

**Target#**      chmod 777 GigEViewApp

**Target#**      /etc/init.d/matrix-gui-2.0 stop      //关闭 Matrix 应用程序

**Target#**      ./GigEViewApp

```
root@AM57xx-Tronlong:~# pwd
/home/root
root@AM57xx-Tronlong:~# ls
GigEViewApp      Grab              lib
Gige_Edge_Detection  genicam_xml_cache
root@AM57xx-Tronlong:~# chmod 777 GigEViewApp
root@AM57xx-Tronlong:~# /etc/init.d/matrix-gui-2.0 stop
Stopping Matrix GUI application.
root@AM57xx-Tronlong:~# ./GigEViewApp
Using device aca640-120gm
GevCurrentIPConfiguration 5
GevLinkSpeed 1000
Using wayland-EGL
wlpvr: PVR Services Initialised
```

图 39

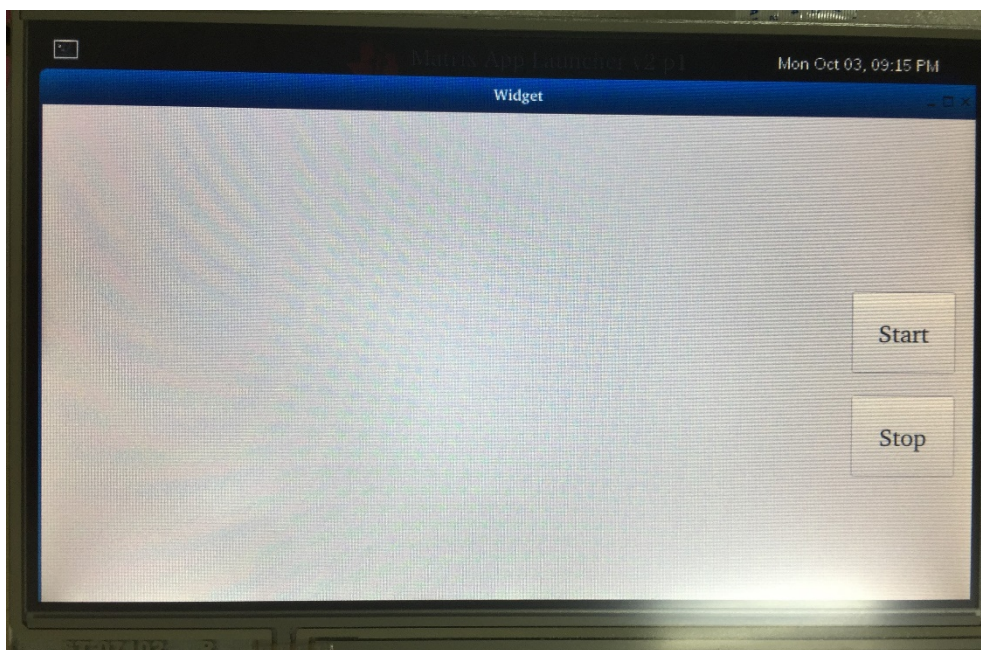


图 40

当串口打印如上信息时，点击 **Start** 按钮即可开始采集显示图像，点击 **Stop** 按钮即可停止采集显示，显示过程中，可以通过调节相机的调焦旋钮和光圈调节旋钮来调整图像的清晰度：

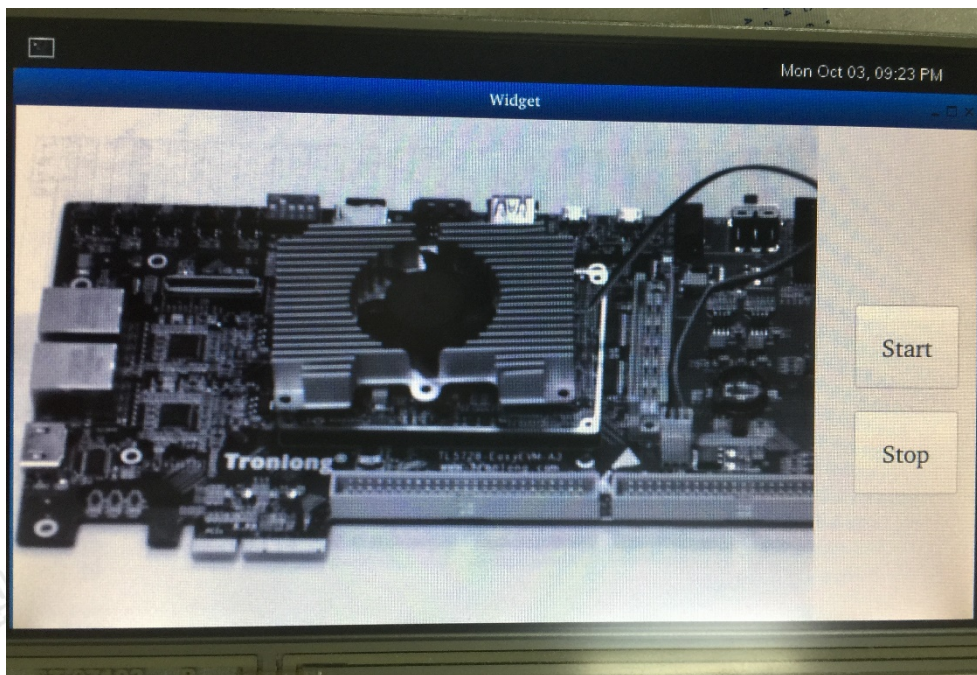


图 41

## 6 Gige\_Edge\_Detection 例程测试

本例程实现 GigE 工业摄像头采集图像后，由 DSP 核心基于 OpenCL 实现边缘检测，结果以.png 格式的图片保存。

### 6.1 安装、配置工程源码

将光盘资料“Demo\GigE\src\Gige\_Edge\_Detection.tar.gz”工程源码文件复制到 Ubuntu 上的“/home/tronlong/AM57xx/pylon/”工作目录，打开“Gige\_Edge\_Detection/src/Gige\_Edge\_Detection.pro”文件：

Host# gedit Gige\_Edge\_Detection/src/Gige\_Edge\_Detection.pro

```
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ pwd
/home/tronlong/AM57xx/pylon
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ ls
Gige_Edge_Detection  GigEViewApp  pylon-5.0.9.10389-armhf  pylon-5.0.9.10389-x86_64
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ gedit Gige_Edge_Detection/src/Gige_Edge_Detection.pro
```

图 42

按照如下方法，修改 Gige\_Edge\_Detection.pro 文件红框部分对应代码：



- 指定对应版本 Linux Processor-SDK 安装路径

TI\_SDK = /home/tronlong/ti-processor-sdk-linux-am57xx-evm-03.01.00.06/

- 指定 GigE 开发源码安装路径

Pylon5\_Dir = /home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5

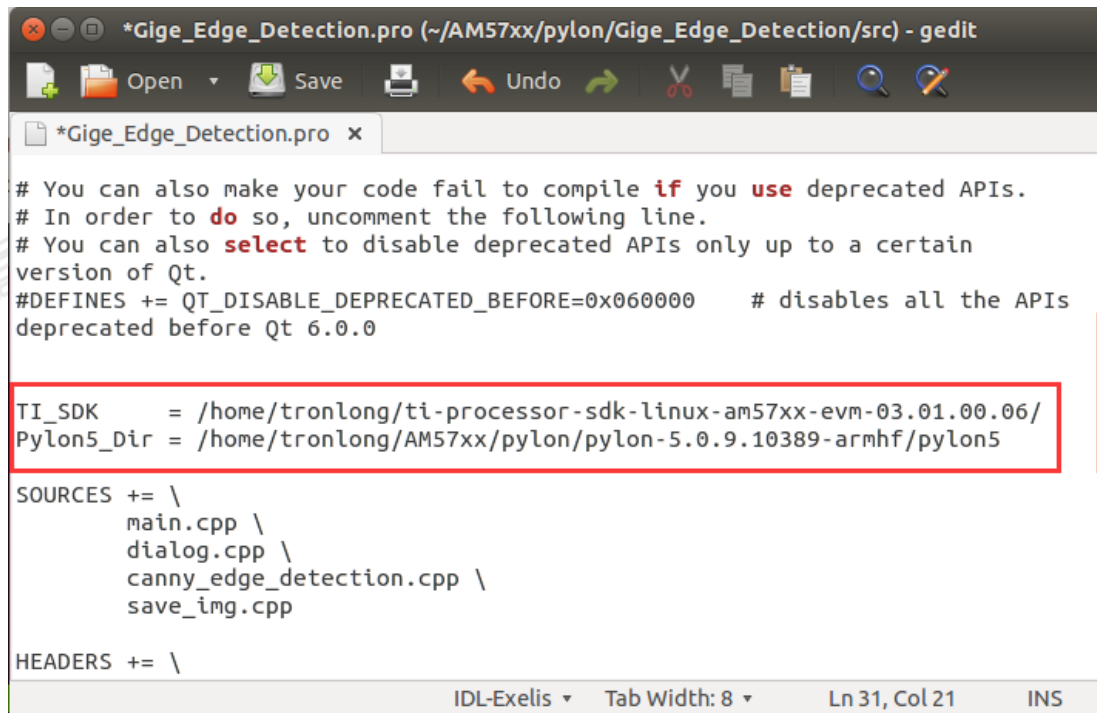


图 43

输入如下命令打开“Gige\_Edge\_Detection/src/kernel/Makefile”文件，按照下图方法指定 Linux Processor-SDK 安装包 Vlib.ae66 库文件所在路径：

Host# gedit Gige\_Edge\_Detection/src/kernel/Makefile

```
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ pwd
/home/tronlong/AM57xx/pylon
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ ls
Gige_Edge_Detection  GigeViewApp  pylon-5.0.9.10389-armhf  pylon-5.0.9.10389-x86_64
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ gedit Gige_Edge_Detection/src/kernel/Makefile
```

图 44

```
TI_VLIB_DIR = /home/tronlong/ti-processor-sdk-linux-am57xx-evm-03.01.00.06/linux-devkit  
/sysroots/armv7ahf-neon-linux-gnueabi/usr/share/ti/ti-vlib-c66x-tree/
```

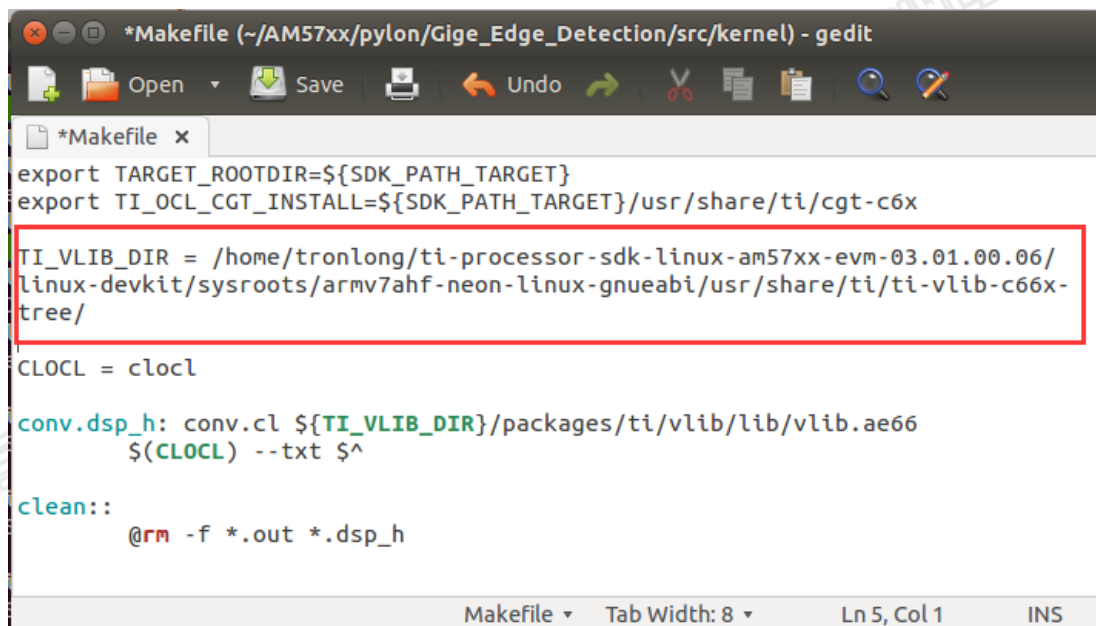


图 45

## 6.2 编译例程

执行如下命令使用对应平台的 Linux Processor-SDK 加载 Qt 等环境变量，加载后，在编译时会使用 Linux Processor-SDK 里面的 ARM 端 Qt 开发环境和运行库。进入到 Gige\_Edge\_Detection 例程源码所在路径，执行 make 指令编译例程：

```
Host source /home/tronlong/ti-processor-sdk-linux-am57xx-evm-03.01.00.06/linux-devkit  
/environment-setup
```

```
Host cd /home/tronlong/AM57xx/pylon/Gige_Edge_Detection/
```

```
Host make
```

```
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ source /home/tronlong/ti-proce
ssor-sdk-linux-am57xx-evm-03.01.00.06/linux-devkit/environment-setup
[linux-devkit]:~/AM57xx/pylon> cd /home/tronlong/AM57xx/pylon/Gige_Edge_Detection/
[linux-devkit]:~/AM57xx/pylon/Gige_Edge_Detection> ls
bin  doc  Makefile  src
[linux-devkit]:~/AM57xx/pylon/Gige_Edge_Detection> make
cd src/kernel && make
make[1]: Entering directory `/home/tronlong/AM57xx/pylon/Gige_Edge_Detection/src/kern
el'
make[1]: `conv.dsp_h' is up to date.
make[1]: Leaving directory `/home/tronlong/AM57xx/pylon/Gige_Edge_Detection/src/kern
el'
cd src && qmake && make
```

图 46

编译完成后，将在当前目录下生成的“src/Gige\_Edge\_Detection”文件复制到开发板文件  
件系统“/home/root”目录下。

```
[linux-devkit]:~/AM57xx/pylon/Gige_Edge_Detection> ls src/
canny_edge_detection.cpp  Gige_Edge_Detection  Makefile  save_img.h
canny_edge_detection.o    Gige_Edge_Detection.pro  moc_dialog.cpp  save_img.o
dialog.cpp                Gige_Edge_Detection.pro~  moc_dialog.o    ui_dialog.h
dialog.h                  kernel                  moc_save_img.cpp  moc_save_img.o
dialog.o                  main.cpp               moc_save_img.o
dialog.ui                 main.o                 save_img.cpp
```

图 47

### 6.3 运行例程

开发板连接触摸屏，上电正常启动进入文件系统，在 Gige\_Edge\_Detection 文件所在  
路径执行如下指令运行：

**Target#**     chmod 777 Gige\_Edge\_Detection

**Target#**     /etc/init.d/matrix-gui-2.0 stop     //关闭 Matrix 应用程序

**Target#**     ./Gige\_Edge\_Detection



```
root@AM57xx-Tronlong:~# pwd
/home/root
root@AM57xx-Tronlong:~# ls
GigEViewApp      Grab              lib
Gige_Edge_Detection  genicam_xml_cache
root@AM57xx-Tronlong:~# chmod 777 Gige_Edge_Detection
root@AM57xx-Tronlong:~# /etc/init.d/matrix-gui-2.0 stop
Stopping Matrix GUI application.
root@AM57xx-Tronlong:~# ./Gige_Edge_Detection
[ 1366.383462] omap-iommu 41501000.mmu: 41501000.mmu: version 3.0
[ 1366.390492] omap-iommu 41502000.mmu: 41502000.mmu: version 3.0
[ 1366.404689] omap-iommu 40d01000.mmu: 40d01000.mmu: version 3.0
[ 1366.411514] omap-iommu 40d02000.mmu: 40d02000.mmu: version 3.0
Using wayland-EGL
wlpvr: PVR Services Initialised
Using device : acA640-120gm
GevCurrentIPConfiguration : 5
GevLinkSpeed : 1000
Ready To Save Image
```

图 48

当串口打印如上信息时，点击 LCD 显示屏 Start 按钮程序开始运行，Pause 按钮暂停并保存当前图像到 Image.png 文件中，点击 Exit 按钮结束运行并退出程序。

```
root@AM57xx-Tronlong:~# ./Gige_Edge_Detection
[ 185.494266] omap-iommu 41501000.mmu: 41501000.mmu: version 3.0
[ 185.501429] omap-iommu 41502000.mmu: 41502000.mmu: version 3.0
[ 185.516098] omap-iommu 40d01000.mmu: 40d01000.mmu: version 3.0
[ 185.522301] omap-iommu 40d02000.mmu: 40d02000.mmu: version 3.0
Using Wayland-EGL
wlpvr: PVR Services Initialised
Using device : acA640-120gm
GevCurrentIPConfiguration : 6
GevLinkSpeed : 1000
Ready To Save Image
Canny Edge Detection Using Time : 0.003415 s
Canny Edge Detection Using Time : 0.041185 s
Canny Edge Detection Using Time : 0.000770 s
Canny Edge Detection Using Time : 0.000781 s
Canny Edge Detection Using Time : 0.000755 s
Canny Edge Detection Using Time : 0.000790 s
Canny Edge Detection Using Time : 0.000770 s
Canny Edge Detection Using Time : 0.000771 s
Canny Edge Detection Using Time : 0.000752 s
Canny Edge Detection Using Time : 0.000770 s
Canny Edge Detection Using Time : 0.000765 s
Canny Edge Detection Using Time : 0.000804 s
Canny Edge Detection Using Time : 0.000826 s
Canny Edge Detection Using Time : 0.000809 s
Canny Edge Detection Using Time : 0.000775 s
Canny Edge Detection Using Time : 0.000807 s
Canny Edge Detection Using Time : 0.000824 s
Canny Edge Detection Using Time : 0.000802 s
Gigeview exit
root@AM57xx-Tronlong:~# ls
GigeViewApp          Grab                  lib
Gige_Edge_Detection  Image.png            lib111
Gige_Edge_Detection_GSTENC genricam.xml_cache
root@AM57xx-Tronlong:~#
```

图 49

将 Image.png 图片文件上传至 PC 端，查看原始尺寸的效果图如下。

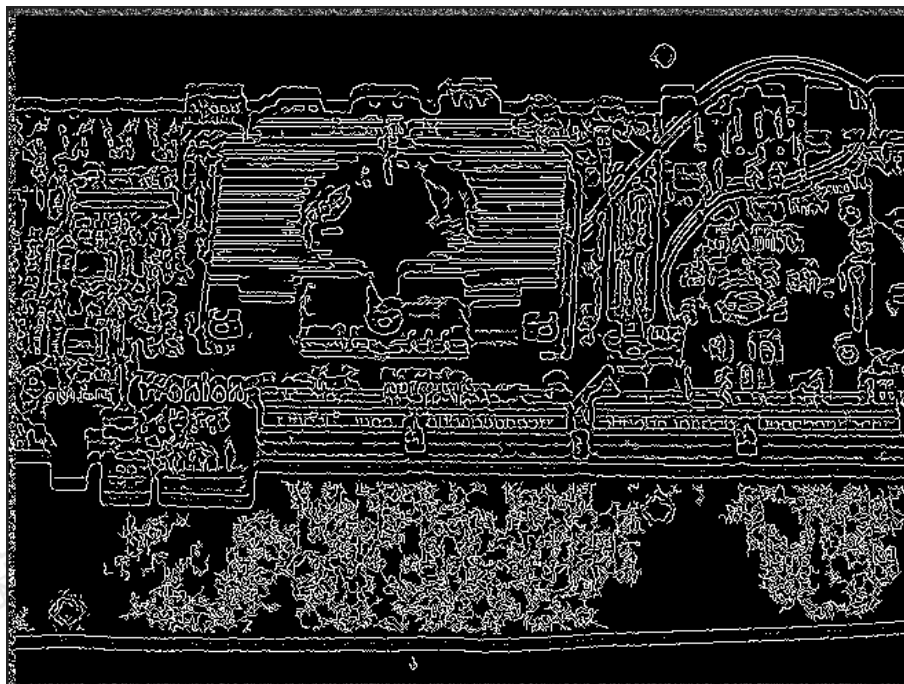


图 50



图 51

## 7 Gige\_Egde\_Detrction\_GSTENC 测试

创龙

公司官网: [www.tronlong.com](http://www.tronlong.com)  
技术论坛: [www.51ele.net](http://www.51ele.net)

销售邮箱: [sales@tronlong.com](mailto:sales@tronlong.com)  
技术邮箱: [support@tronlong.com](mailto:support@tronlong.com)

公司总机: 020-8998-6280  
技术热线: 020-3893-9734

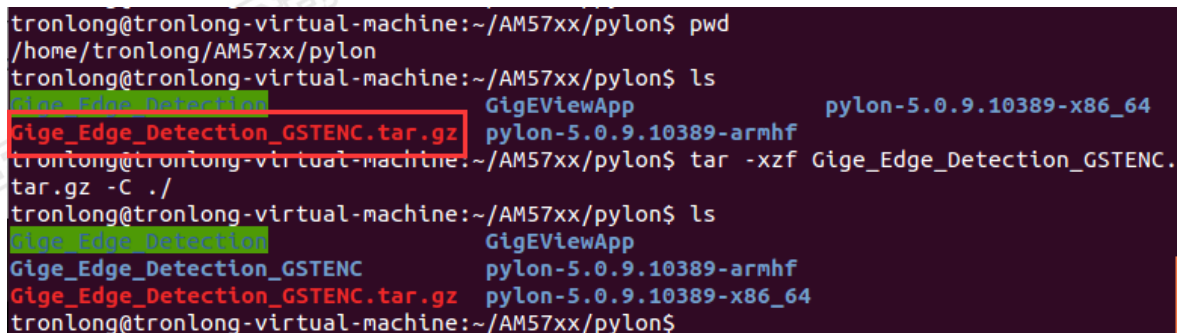
37/49

本例程实现 GigE 工业摄像头采集图像，DSP 核心基于 OpenCL 实现边缘检测，GST 硬件编码，结果保存在 test.264 文件。

## 7.1 安装、配置工程源码

将光盘资料“Demo\GigE\src\Gige\_Edge\_Detection\_GSTENC.tar.gz”工程源码压缩文件，拷贝到 Ubuntu 上的“/home/tronlong/AM57xx/pylon/”工作目录。进入 Gige\_Edge\_Detection\_GSTENC.tar.gz 文件所在路径，执行如下指令将其解压到当前目录下：

**Host** tar -xzf Gige\_Edge\_Detection\_GSTENC.tar.gz -C ./



```
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ pwd
/home/tronlong/AM57xx/pylon
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ ls
Gige_Edge_Detection  GigEViewApp  pylon-5.0.9.10389-x86_64
Gige_Edge_Detection_GSTENC.tar.gz  pylon-5.0.9.10389-armhf
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ tar -xzf Gige_Edge_Detection_GSTENC.tar.gz -C ./
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ ls
Gige_Edge_Detection  GigEViewApp
Gige_Edge_Detection_GSTENC  pylon-5.0.9.10389-armhf
Gige_Edge_Detection_GSTENC.tar.gz  pylon-5.0.9.10389-x86_64
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$
```

图 52

打开“Gige\_Edge\_Detection\_GSTENC/src/Gige\_Edge\_Detection\_GSTENC.pro”文件，按照如下方法，修改 Gige\_Edge\_Detection\_GSTENC.pro 文件红框部分对应代码：

- 指定对应版本 Linux Processor-SDK 安装路径

TI\_SDK = /home/tronlong/ti-processor-sdk-linux-am57xx-evm-03.01.00.06/

- 指定 GigE 开发源码安装路径

Pylon5\_Dir = /home/tronlong/AM57xx/pylon/pylon-5.0.9.10389-armhf/pylon5

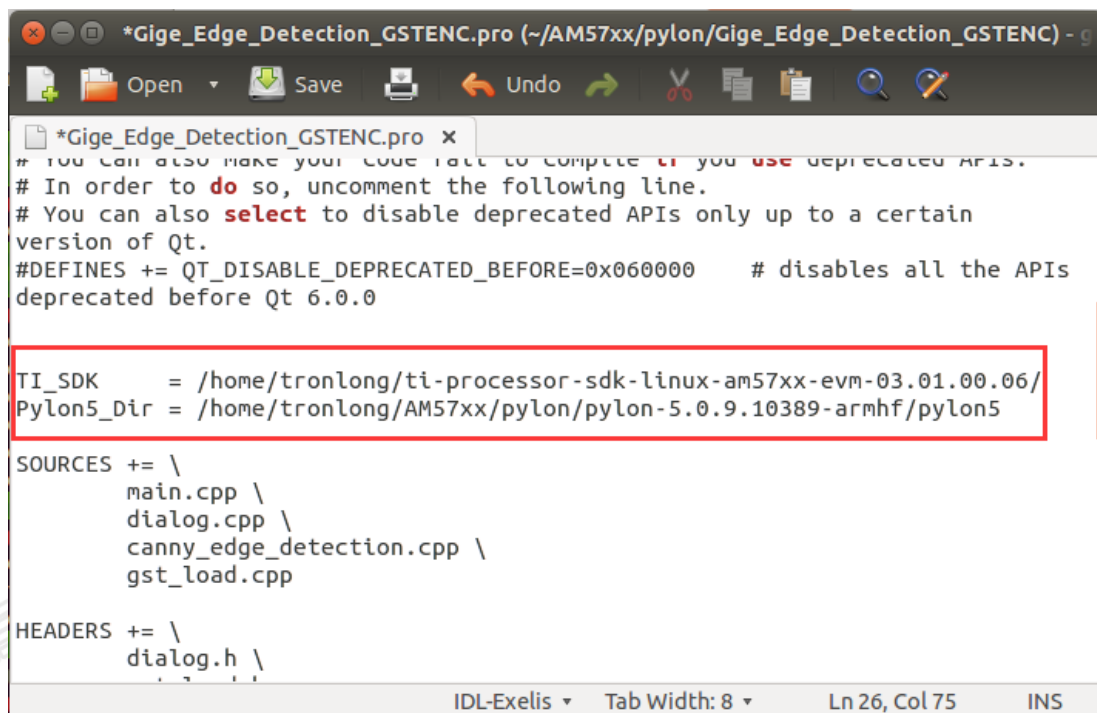


图 53

打开“Gige\_Edge\_Detection\_GSTENC/src/kernel/Makefile”文件，按照下图方法指定 Linux Processor-SDK 安装包 Vlib.ae66 库文件所在路径：

➤ 指定 Linux Processor-SDK 安装包 Vlib.ae66 库文件路径

```
TI_VLIB_DIR = /home/tronlong/ti-processor-sdk-linux-am57xx-evm-03.01.00.06/linux-devkit
/sysroots/armv7ahf-neon-linux-gnueabi/usr/share/ti/ti-vlib-c66x-tree/
```



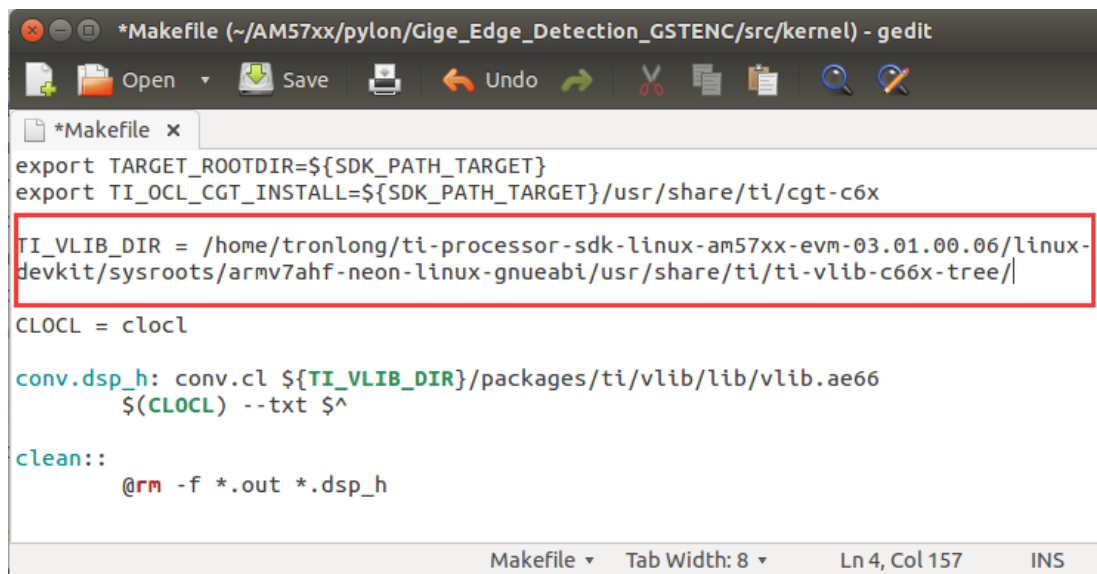


图 54

## 7.2 编译例程

执行如下命令使用对应平台的 Linux Processor-SDK 加载 Qt 等环境变量，加载后，在编译时会使用 Linux Processor-SDK 里面的 ARM 端 Qt 开发环境和运行库。进入到 Gige\_Edge\_Detection\_GSTENC 例程源码所在路径，执行 make 指令编译例程：

**Host** source /home/tronlong/ti-processor-sdk-linux-am57xx-evm-03.01.00.06/linux-devkit/environment-setup

**Host** cd /home/tronlong/AM57xx/pylon/Gige\_Edge\_Detection\_GSTENC/

**Host** make

```
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ source /home/tronlong/ti-processor-sdk-linux-am57xx-evm-03.01.00.06/linux-devkit/environment-setup
[linux-devkit]:~/AM57xx/pylon> cd /home/tronlong/AM57xx/pylon/Gige_Edge_Detection_GSTENC/
[linux-devkit]:~/AM57xx/pylon/Gige_Edge_Detection_GSTENC> ls
bin  Makefile  src
[linux-devkit]:~/AM57xx/pylon/Gige_Edge_Detection_GSTENC> make
cd src/kernel && make
make[1]: Entering directory `/home/tronlong/AM57xx/pylon/Gige_Edge_Detection_GSTENC/src/kernel'
```

图 55

编译完成后，会在当前目录下生成的“src/Gige\_Edge\_Detection\_GSTENC”可执行文件，将其复制到开发板文件系统“/home/root”目录下。

**Host#** ls src/Gige\_Edge\_Detection\_GSTENC



```
[linux-devkit]:~/AM57xx/pylon/Gige_Edge_Detection_GSTENC> ls src/Gige_Edge_Detection_GSTENC
src/Gige_Edge_Detection_GSTENC
[linux-devkit]:~/AM57xx/pylon/Gige_Edge_Detection_GSTENC>
```

图 56

### 7.3 运行例程

开发板连接触摸屏，上电正常启动进入文件系统，在 Gige\_Edge\_Detection 文件所在路径执行如下指令运行：

**Target#**      `chmod 777 Gige_Edge_Detection_GSTENC`

**Target#**      `/etc/init.d/matrix-gui-2.0 stop`    //关闭 Matrix 应用程序

**Target#**      `./Gige_Edge_Detection_GSTENC`

```
root@AM57xx-Tronlong:~# pwd
/home/root
root@AM57xx-Tronlong:~# ls
GigeViewApp      Gige_Edge_Detection_GSTENC  genicam_xml_cache
Gige_Edge_Detection  lib
root@AM57xx-Tronlong:~# chmod 777 Gige_Edge_Detection_GSTENC
root@AM57xx-Tronlong:~# /etc/init.d/matrix-gui-2.0 stop
Stopping Matrix GUI application.
root@AM57xx-Tronlong:~# ./Gige_Edge_Detection_GSTENC
[ 112.798571] omap-iommu 41501000.mmu: 41501000.mmu: version 3.0
[ 112.805693] omap-iommu 41502000.mmu: 41502000.mmu: version 3.0
[ 112.820482] omap-iommu 40d01000.mmu: 40d01000.mmu: version 3.0
[ 112.826539] omap-iommu 40d02000.mmu: 40d02000.mmu: version 3.0
Using wayland-EGL
wlpvr: PVR Services Initialised
```

图 57

当串口打印如上信息时，点击 LCD 显示屏 Start 按钮程序开始运行，点击 Exit 按钮退出程序，结果保存在当前目录下新生成 test.264 文件。

```
root@AM57xx-Tronlong:~# ./Gige_Edge_Detection_GSTENC
Using Wayland-EGL
wlpvr: PVR Services Initialised

Using device : aca640-120gm
GevCurrentIPConfiguration : 6
GevLinkspeed : 1000
Ready To Save Image ...
[ 555.495422] omap-iommu 55082000.mmu: 55082000.mmu: version 2.1

(Gige_Edge_Detection_GSTENC:1208): Glib-CRITICAL **: g_main_loop_run: assertion 'loop != NULL' failed
Got Frame , sleep 0 us
Got Frame , sleep 0 us
Got Frame , sleep 0 us
Got Frame , sleep 0 us
Got Frame , sleep 0 us
Got Frame , sleep 40959 us
Got Frame , sleep 14687 us
Got Frame , sleep 14687 us
Got Frame , sleep 6443 us
Got Frame , sleep 6443 us
Got Frame , sleep 6443 us
Got Frame , sleep 11136 us
Got Frame , sleep 11136 us
Got Frame , sleep 11136 us
Got Frame , sleep 11136 us
Got Frame , sleep 9101 us
Got Frame , sleep 9101 us
Got Frame , sleep 9101 us
Got Frame , sleep 9101 us
Got Frame , sleep 9101 us
Got Frame , sleep 7045 us
Got Frame , sleep 7045 us
Got Frame , sleep 7045 us
Got Frame , sleep 7045 us
Got Frame , sleep 7045 us
Got Frame , sleep 8283 us
Got Frame , sleep 8283 us
Got Frame , sleep 8283 us
Got Frame , sleep 8283 us
Got Frame , sleep 8164 us
Got Frame , sleep 8164 us
Got Frame , sleep 8164 us
Got Frame , sleep 8164 us
Got Frame , sleep 6517 us
Got Frame , sleep 6517 us
root@AM57xx-Tronlong:~#
root@AM57xx-Tronlong:~# pwd
/home/root
root@AM57xx-Tronlong:~# ls test.264
test.264
root@AM57xx-Tronlong:~#
```

图 58

将 test.264 文件重命名为 test.h264 上传至 PC，在支持 H264 解码的播放器上播放此文件，效果如下图所示：

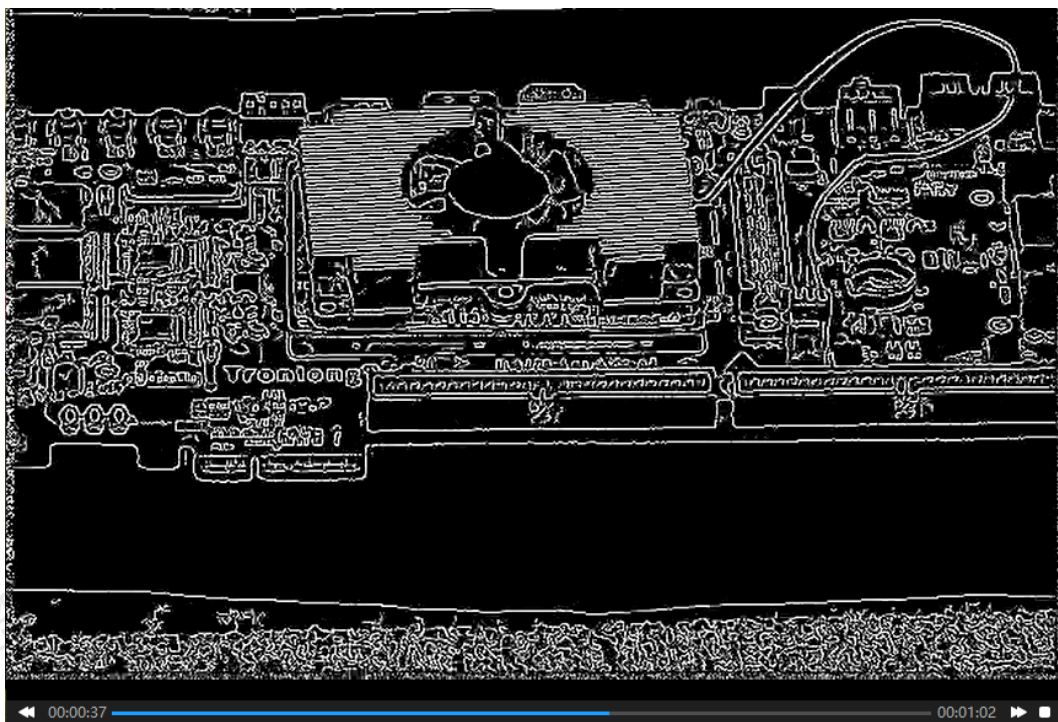


图 59

## 8 Gige\_Edge\_Detrction\_GSTENC\_RTSP 测试

本例程实现 GigE 工业摄像头采集图像，DSP 核心基于 OpenCL 实现边缘检测，GST 硬件编码，向 RTSP 服务器发送 H.264 媒体流。

双击光盘资料“Tools\Windows\vlc-2.2.4-win32.exe”VLC 多媒体播放器安装包，安装 VLC 多媒体播放器。

### 8.1 安装、配置工程源码

将光盘资料“Demo\GigE\src\Gige\_Edge\_Detection\_GSTENC\_RTSP.tar.gz”压缩源码文件拷贝到 Ubuntu 上的“/home/tronlong/AM57xx/pylon/”工作目录，执行如下指令将其解压到当前目录下：

```
Host tar -xzf Gige_Edge_Detection_GSTENC_RTSP.tar.gz -C ./
```

```
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ pwd
/home/tronlong/AM57xx/pylon
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ ls
Gige_Edge_Detection          GigeViewApp
Gige_Edge_Detection_GSTENC   pylon-5.0.9.10389-armhf
Gige_Edge_Detection_RTSP.tar.gz pylon-5.0.9.10389-x86_64
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ tar -xzf Gige_Edge_Detection_G
STENC_RTSP.tar.gz -C ./
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ ls
Gige_Edge_Detection          GigeViewApp
Gige_Edge_Detection_GSTENC   pylon-5.0.9.10389-armhf
Gige_Edge_Detection_RTSP     pylon-5.0.9.10389-x86_64
Gige_Edge_Detection_RTSP.tar.gz
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$
```

图 60

按照如上方法，依次修改“Gige\_Edge\_Detection\_GSTENC\_RTSP/src/Gige\_Edge\_Detection\_GSTENC\_RTSP.pro”和“Gige\_Edge\_Detection\_GSTENC\_RTSP/src/kernel/Makefile”文件。

## 8.2 编译例程

执行如下命令使用对应平台的 Linux Processor-SDK 加载 Qt 等环境变量，加载后，在编译时会使用 Linux Processor-SDK 里面的 ARM 端 Qt 开发环境和运行库。进入到 Gige\_Edge\_Detection\_GSTENC 例程源码所在路径，执行 make 指令编译例程：

```
Host source /home/tronlong/ti-processor-sdk-linux-am57xx-evm-03.01.00.06/linux-devkit
/environment-setup
```

```
Host cd /home/tronlong/AM57xx/pylon/Gige_Edge_Detection_GSTENC_RTSP
```

```
Host make
```

```
tronlong@tronlong-virtual-machine:~/AM57xx/pylon$ source /home/tronlong/ti-proce
ssor-sdk-linux-am57xx-evm-03.01.00.06/linux-devkit/environment-setup
[linux-devkit]:~/AM57xx/pylon> cd /home/tronlong/AM57xx/pylon/Gige_Edge_Detectio
n_GSTENC_RTSP/
[linux-devkit]:~/AM57xx/pylon/Gige_Edge_Detection_GSTENC_RTSP> make
cd ipc-rtsp-server && make
make[1]: Entering directory `/home/tronlong/AM57xx/pylon/Gige_Edge_Detection_GST
ENC_RTSP/ipc-rtsp-server'
arm-linux-gnueabi-g++ -O0 -ggdb -I. -Wall -I./live/liveMedia/include -I./live/
groupsock/include -I./live/BasicUsageEnvironment/include -I./live/UsageEnvironme
nt/include -c common.cpp -o .obj/common.o
```

图 61

编译完成后，将在当前目录下生成“src/Gige\_Edge\_Detection\_GSTENC\_RTSP”和“ipc-rtsp-server/mq-buf-live”可执行文件，将编译生成的两个可执行文件拷贝到开发板文件系统“/home/root”目录下。



Host# ls src/Gige\_Edge\_Detection\_GSTENC\_RTSP

Host# ls ipc-rtsp-server/mq-buf-live

```
ser_gcc_v3_0_Basler_pylon_v5_0 -L/home/tronlong/ti-processor-sdk-linux-am57xx-ev
m-03.01.00.06/linux-devkit/sysroots/armv7ahf-neon-linux-gnueabi/usr/lib -lQt5Wid
gets -lQt5Gui -lQt5Core -lGLESV2 -lpthread
make[1]: Leaving directory `/home/tronlong/AM57xx/pylon/Gige_Edge_Detection_GSTE
NC_RTSP/src'
[linux-devkit]:~/AM57xx/pylon/Gige_Edge_Detection_GSTENC_RTSP> pwd
/home/tronlong/AM57xx/pylon/Gige_Edge_Detection_GSTENC_RTSP
[linux-devkit]:~/AM57xx/pylon/Gige_Edge_Detection_GSTENC_RTSP> ls src/Gige_Edge_
Detection_GSTENC_RTSP
src/Gige_Edge_Detection_GSTENC_RTSP
[linux-devkit]:~/AM57xx/pylon/Gige_Edge_Detection_GSTENC_RTSP> ls ipc-rtsp-serve
r/mq-buf-live
ipc-rtsp-server/mq-buf-live
[linux-devkit]:~/AM57xx/pylon/Gige_Edge_Detection_GSTENC_RTSP>
```

图 62

### 8.3 运行例程

开发板连接触摸屏，上电正常启动进入文件系统。在 mq-buf-live 和 Gige\_Edge\_DetectionGige\_GSTENC\_RTSP 可执行文件所在路径执行如下指令运行：

Target# chmod 777 Gige\_Edge\_Detection\_GSTENC\_RTSP mq-buf-live

Target# /etc/init.d/matrix-gui-2.0 stop //关闭 Matrix 应用程序

Target# ./mq-buf-live &

Target# ./Gige\_Edge\_Detection\_GSTENC\_RTSP

```
root@AM57xx-Tronlong:~# pwd
/home/root
root@AM57xx-Tronlong:~# ls Gige_Edge_Detection_GSTENC_RTSP mq-buf-live
Gige_Edge_Detection_GSTENC_RTSP mq-buf-live
root@AM57xx-Tronlong:~# chmod 777 Gige_Edge_Detection_GSTENC_RTSP mq-buf-live
root@AM57xx-Tronlong:~# /etc/init.d/matrix-gui-2.0 stop
Stopping Matrix GUI application.
root@AM57xx-Tronlong:~# ./mq-buf-live &
[1] 1236
Play streams from this server using the URL
rtsp://192.168.1.99/b264_ch1_avstream
root@AM57xx-Tronlong:~# ./Gige_Edge_Detection_GSTENC_RTSP
[ 669.093821] omap-iommu 41501000.mmu: 41501000.mmu: version 3.0
[ 669.100488] omap-iommu 41502000.mmu: 41502000.mmu: version 3.0
[ 669.114357] omap-iommu 40d01000.mmu: 40d01000.mmu: version 3.0
[ 669.120704] omap-iommu 40d02000.mmu: 40d02000.mmu: version 3.0
Using Wayland-EGL
wlvpr: PVR Services Initialised
[ 669.554121] capability: warning: 'Gige_Edge_Detec' uses 32-bit capabilities (legacy support in use)
Using device : acA640-120gm
GevCurrentIPConfiguration : 6
GevLinkSpeed : 1000
Ready To Save Image ...
```

图 63

当串口打印如上信息时，点击 LCD 显示屏 Start 按钮程序开始运行，LCD 显示屏显示采集到的图像，在 PC 机使用流播放器输入 URL 地址进行视频图像播放，点击 EXIT 运行结束。

```
root@AM57xx-Tronlong:~# ./mq-buf-live &
[2] 1268
Failed to create RTSP server: bind() error (port number: 554): Address already in use
root@AM57xx-Tronlong:~# ./Gige_Edge_Detection_GSTENC_RTSP
[ 1152.060969] omap-iommu 41501000.mmu: 41501000.mmu: version 3.0
[ 1152.067857] omap-iommu 41502000.mmu: 41502000.mmu: version 3.0
[ 1152.081955] omap-iommu 40d01000.mmu: 40d01000.mmu: version 3.0
[ 1152.088365] omap-iommu 40d02000.mmu: 40d02000.mmu: version 3.0
Using Wayland-EGL
wlpvr: PVR Services Initialised
Using device : aca640-120gm
GevCurrentIPConfiguration : 6
GevLinkSpeed : 1000
Ready To Save Image ...
[ 1155.784997] omap-iommu 55082000.mmu: 55082000.mmu: version 2.1
(Gige_Edge_Detection_GSTENC_RTSP:1269): GLib-CRITICAL **: g_main_loop_run: assertion
'loop != NULL' failed
sleep 33195 us!
sleep 33248 us!
sleep 33306 us!
sleep 33247 us!
sleep 33217 us!
sleep 33207 us!
sleep 33293 us!
sleep 33298 us!
sleep 33156 us!
sleep 33271 us!
sleep 33302 us!
sleep 33279 us!
sleep 33233 us!
sleep 33124 us!
[2]+  Done(1)                  ./mq-buf-live
root@AM57xx-Tronlong:~#
```

图 64

双击打开 VLC 多媒体播放器，按照下图步骤打开网络串流窗口，在窗口中输入网络 URL 为“rtsp://192.168.1.99/h264\_ch1\_avstream”，网络 URL 中的 IP 地址为开发板的实际 IP 地址。



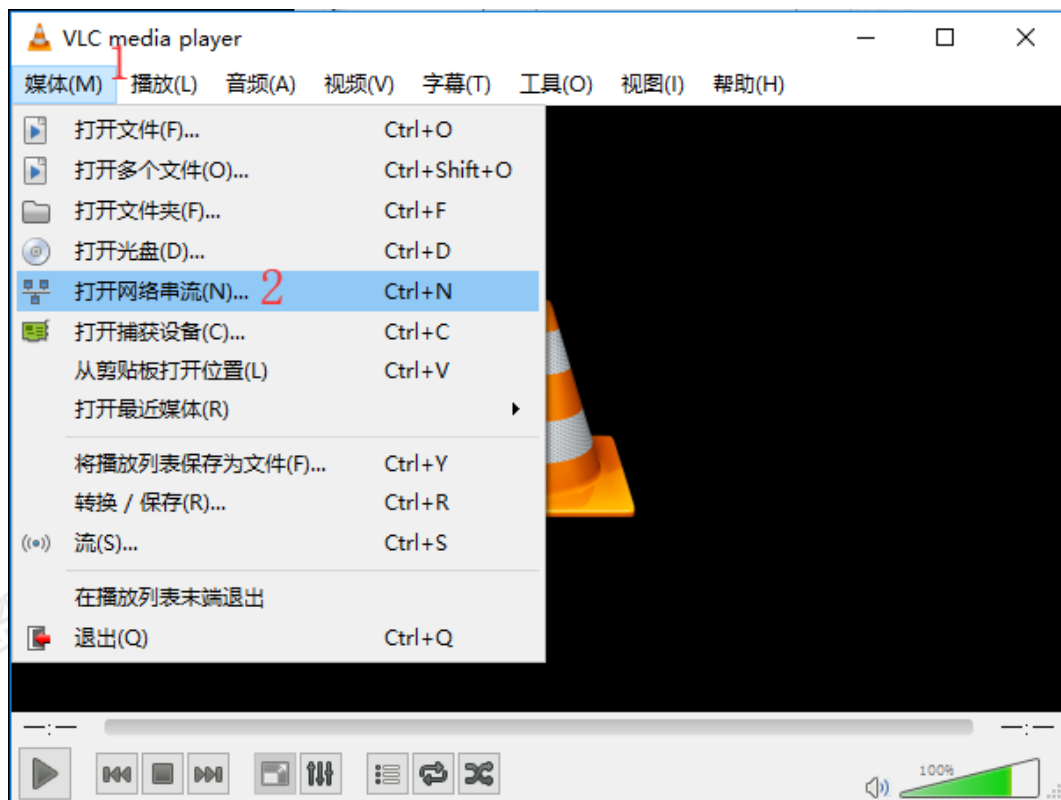


图 65

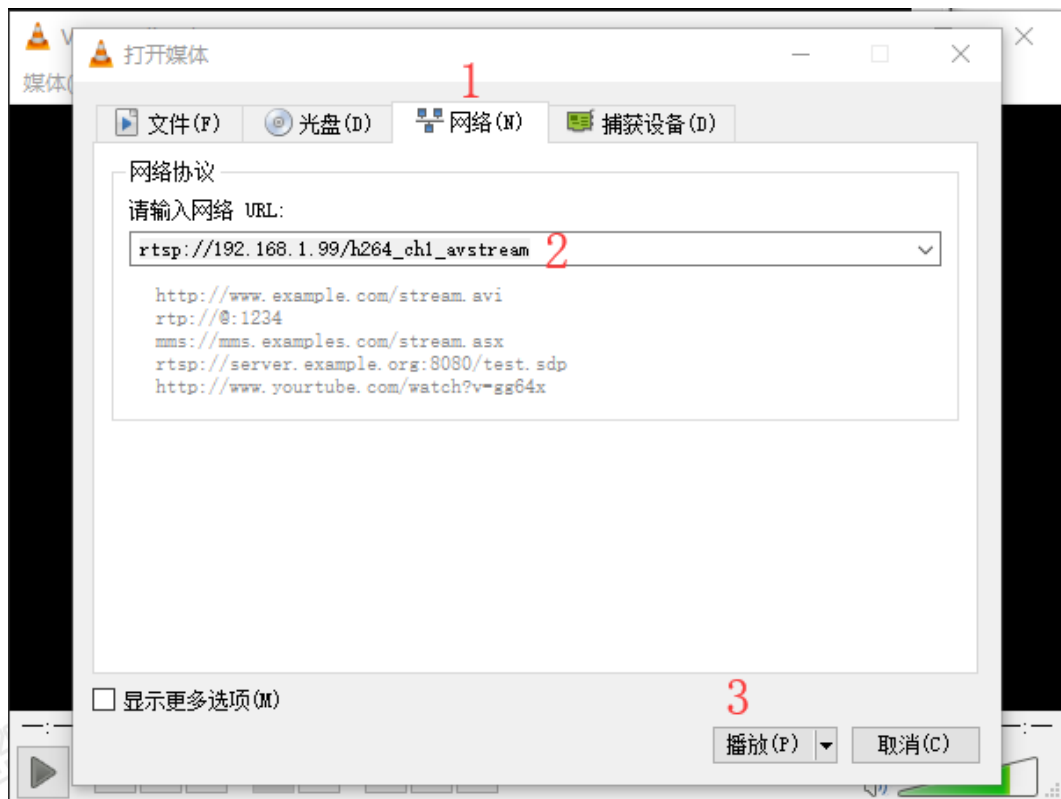


图 66

点击播放按钮后，可以看到 VLC 多媒体播放器播放 GigE 相机采集到的视频图像，显示的视频信息与开发板 LCD 显示屏所显示的同步，如下图所示：

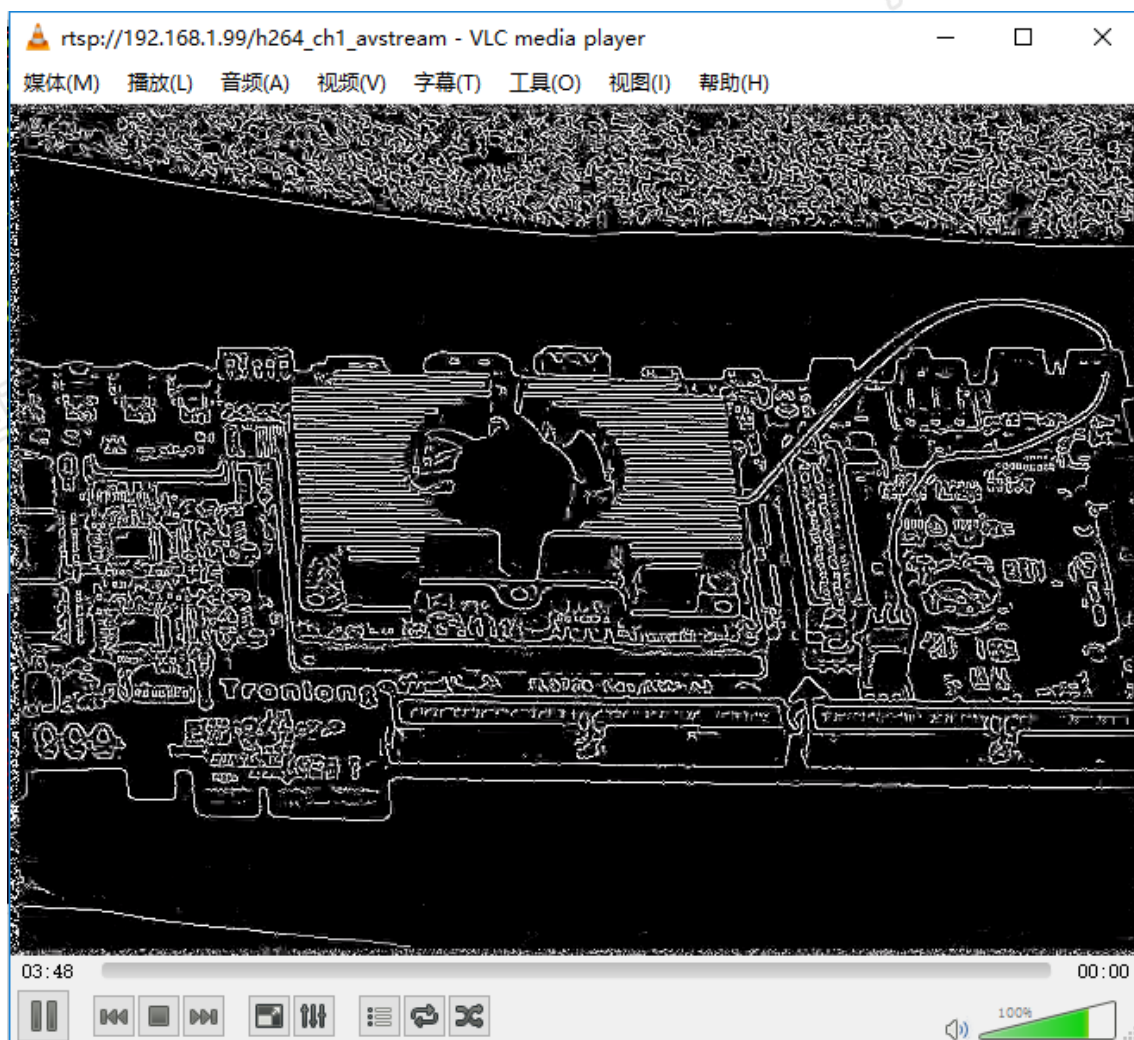


图 67

## 更多帮助

销售邮箱: [sales@tronlong.com](mailto:sales@tronlong.com)

技术邮箱: [support@tronlong.com](mailto:support@tronlong.com)

创龙总机: 020-8998-6280

技术热线: 020-3893-9734

创龙官网: [www.tronlong.com](http://www.tronlong.com)

技术论坛: [www.51ele.net](http://www.51ele.net)

线上商城: <https://tronlong.taobao.com>