Qualcomm Developer Project objectTracker-demo2.0

Project Submission

|  |  |  |
| --- | --- | --- |
| **Attribution:** |  | |
| **Email address** | [zhangzz6687@thundersoft.com](mailto:zhangzz6687@thundersoft.com)  [zhanglei0706@thundersoft.com](mailto:sunzhen@thundersoft.com)  Yuandk0305@thundersoft.com | |
| **Project Title**\* | objectTracker-demo2.0 | |
| **Images**  *Upload up to 5 images of your project*  *Please submit/send the original JPEG/PNG files for all images included in the document* | **Qualcomm® QCS610 SoC.png**    [Alt tag: “objectTracker-demo using The Qualcomm® QCS610 SOC Open Kit ”]  **usb.png**   |  | | --- | | **typc** |   [Alt tag: “using the USB line to develop on Qualcomm® QCS610 SOC development board” ]  **dpline.jpg**  dpline  [Alt tag: “using the universal DP line to connect LED displayer to Qualcomm® QCS610 SOC development board”]  **LED-Displayer.png**  LED-Displayer  [Alt tag: “using a LED Displayer to display the objectTracker-demo interface from Qualcomm® QCS610 SOC development board”]  **charger.jpg**  charger  [Alt tag: “using round-hole charger to power Qualcomm® QCS610 SOC development board”] | |
| **Description**\*  *High level description of the project* ***(75 words or less)*** | This project is based on Qualcomm Neural processing SDK for AI. Using the Qualcomm® QCS610 SOC Open Kit as core developent board. Then, its powerful computing ability will been show by a object tracking demo is completed that combine with Qualcomm Neural processing SDK for AI, Gstreamer, OpenCV and open source of YUV. Among them, the model adopted is converted from Caffe's Goturn model to Qualcomm Neural processing SDK for AI's DLC model. Gstreamer is used to get video streams that need to be tracked for objects.OpenCV is mainly used for simple processing of pictures.YUV is used to convert video frames to image formats. | |
| **Objective**   * *What inspired you to create this project?* * *What is your desired outcome?* | AI integration based on Qualcomm® QCS610 SOC open kit and Qualcomm Neural processing SDK for AI for implementing object tracking. | |
| **Materials Required / Parts List / Tools** | Part Name | Link to purchase |
| Qualcomm® QCS610 SOC Open Kit | https://www.thundercomm.com/app\_zh/product/1593776185472315 |
| USB line | https://item.jd.com/40759941966.html |
| DP line |  |
| LED Displayer |  |
| Charger |  |
|  |  |
|  |  |
| **Source Code / Source Examples / Application Executable**  *Link to open source / shareable code repository* | Description | Link |
| [Source Code](https://github.com/ThunderSoft-XA/C610-objectTracker-demo) | https://github.com/ThunderSoft-XA/C610-objectTracker-demo2.0 |
|  |  |
|  |  |
|  |  |
| **Additional Resources**  *List related links or resources such as websites, videos, presentations, or other materials* | Resource Title | Link or File Name (and provide file) |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| **Build / Assembly Instructions** | Sample outline:   1. Overall design framework and Test environment construction method.      1. Software Build Instructions    1. Prepare a PC (Ubuntu 16.04) with Yocto Project;    2. Install adb ;    3. Configure the compilation environment according to the release note document    4. Configure the DP display environment    5. Write BB file(<objecttrack\_0.1.bb>),Executing the “bitbake objecttrack” command generates an “objectTracker” executable file 2. Start objectTracker-demo    1. Copy “objectTracker” executable file to project root directory, then “adb push ” whole project file to Qualcomm® QCS610 SOC,ex./data/ dir.    2. Execute adb root && adb shell. Next, enter the objectTracker-demo project root directory, execute ./weston\_dp\_client objectTracker. | |
|  | Sample outline:   1. How does it work?   The running process of demo is relatively clear. After the program runs, the environment initialization of GStreamer and Qualcomm Neural processing SDK for AI will be carried out immediately. Then, the local video stream pipeline will be built and the Qualcomm Neural processing SDK for AI model will be loaded; After receiving the signal from the pipeline, the captured video frame is transformed into picture frame format, and the object is tracked by neural network, and the object coordinate value is returned.  int main(int argc, char \*\*argv)  {  cout << \_\_FILE\_\_ << \_\_LINE\_\_ << endl;  // if(setCameraEnv()) {  // printf("camera env init failed\n");  // return -1;  // }  // system("source /etc/gstreamer1.0/set\_gst\_env.sh");  setenv("MALLOC\_TRACE", "mtrace.log", 1);  mtrace();  int count = 2;  char config\_file[1024] = "/home/user/Desktop/test/objectTracker/res/config.ini";  MulitGstCamera::GstEnvInit();  GMainLoop \*main\_loop = g\_main\_loop\_new(NULL,false);  StreamConf gstCameraConf;  DEBUG\_FUNC();  for(int i=0; i< count; i++)  {  memset(&gstCameraConf, 0, sizeof(StreamConf));  sprintf(gstCameraConf.gst\_dic, "gst\_%d",i);  gst\_param\_load(config\_file,&gstCameraConf);  DEBUG\_FUNC();  GstCamera \*my\_gst\_camera = new GstCamera();  DEBUG\_FUNC();  my\_gst\_camera->set\_pipe\_name(gstCameraConf.gst\_name);  my\_gst\_camera->set\_sink\_name(gstCameraConf.gst\_sink);  my\_gst\_camera->set\_gst\_type(gstCameraConf.gst\_type);  my\_gst\_camera->set\_ai\_type(gstCameraConf.ai\_type);  my\_gst\_camera->set\_width(gstCameraConf.width);  my\_gst\_camera->set\_height(gstCameraConf.height);  my\_gst\_camera->set\_decode\_type(gstCameraConf.decode);  my\_gst\_camera->set\_format(gstCameraConf.format);  my\_gst\_camera->set\_path(gstCameraConf.path);  my\_gst\_camera->set\_hw\_dec(false);  my\_gst\_camera->rgb\_object\_frame = std::make\_shared<BufManager<cv::Mat> > ();  if(AIType::TARGETTRACK == my\_gst\_camera->get\_ai\_type()) {  DEBUG\_FUNC();  std::shared\_ptr<cv::Mat> object\_frame;  my\_gst\_camera->Init(MulitGstCamera::GetPipeline());  std::cout << my\_gst\_camera->get\_pipeline\_str() << std::endl;  \_listGstCam.push\_back(my\_gst\_camera);  } else if (AIType::OBJECTDETECT == my\_gst\_camera->get\_ai\_type()) {  //waitting compact  DEBUG\_FUNC();  } else if (AIType::OBJECTDETECT == my\_gst\_camera->get\_ai\_type()){  //waitting compact  DEBUG\_FUNC();  } else {  cout << "Disable AI function" << endl;  my\_gst\_camera->Init(MulitGstCamera::GetPipeline());  std::cout << my\_gst\_camera->get\_pipeline\_str() << std::endl;  \_listGstCam.push\_back(my\_gst\_camera);  }  }  DEBUG\_FUNC();  std::thread handleThread(thread\_get\_frame);  handleThread.detach();  sleep(2);  // According to the AI type run corresponding thread after enable SUPPORT\_AI  #if SUPPORT\_AI  std::thread trackThread(thread\_object\_track);  trackThread.detach();  #endif  #if SHOW\_VIDEO  DEBUG\_FUNC();  {  std::thread showThread(thread\_show);  showThread.join();  }  #endif // SHOW\_VIDEO  DEBUG\_FUNC();  g\_main\_loop\_run(main\_loop);  MulitGstCamera::GstEnvDeinit();  g\_main\_loop\_unref(main\_loop);  return 0;  } | |
| **Usage Instructions** | The object tracking results are as follows：    Note:  The gray part in the middle of the image is the track map of the athlete. Because the athlete is always near the center of the video, her track is almost concentrated in that area.You may be surprised that the frame rate is only 1 FPS, but it runs in CPU mode. It should be much better to switch to GPU mode. | |
| **Contributor(s) Info**  *Feel free to include headshots!* | Name | Title  Company |
|  |  |
|  |  |
|  |  |

––– Continued on next page –––

Filters and Tags for QDN projects page

|  |  |  |
| --- | --- | --- |
| **Platform/Hardware** | CSR 101x/102x Bluetooth  DragonBoard 410c  mangOH Red/Yellow  √ Qualcomm QCS610 SoC | MDM920x LTE for IoT  QCA-402x WiFi/BLE/Zigbee  Qualcomm Robotics RBx Dev Kit |
| **Software Tools** | 3D Audio Plugin for Unity  Adreno GPU SDK  Hexagon DSP SDK | √ Neural Processing SDK for AI  　Snapdragon Profiler |
| **Operating System** | Android  √ Linux  ThreadX RTOS | Ubuntu Core  Windows 10 IoT Core |
| **Cloud Services/Platform** | Sierra Wireless AirVantage  Gizwits Cloud Platform  AT&T M2X  IBM Bluemix | IBM Watson IoT  Microsoft Azure IoT  Amazon AWS IoT |
| **Skill Level Required** | Advanced  Beginner  √ Intermediate |  |
| **Areas of Focus** | 3D Printing & Modeling  Alexa Voice Service  √ Artificial Intelligence  Bluetooth  √ Computer Vision  Digital Signage  Education  √ Embedded  Gaming | Healthcare  √ IoT  √ Robotics  √ Security  Sensors  √ Smart Cities  Smart Home  Toys |

*By submitting your content (“Submission”), you are granting Qualcomm a royalty-free, perpetual, non-exclusive, unrestricted, worldwide license to: (a) post, use, copy, sublicense, adapt, transmit, publicly perform or display any such Submission, (b) use, reproduce, modify, adapt, publish, translate, create derivative works from, distribute, perform, play, host, communicate, make available and publish your Submission without restriction and (c) sublicense to third parties the unrestricted right to exercise any of the foregoing rights granted with respect to the Submission. The foregoing grants shall include the right to exploit any ideas, concepts, intellectual property, or proprietary rights in such Submission, including but not limited to rights under copyright, trademark, servicemark or patent laws under any relevant jurisdiction without Qualcomm owing any monies to you whatsoever. You represent and warrant that you own all right, title and interest in and to the Submission, or you have been granted sufficient rights in and to the Submission allowing the foregoing use of such Submission.*