Distributed Networks for Robotics

Yongxin Zhang

November 2015

Contents

1 Introduction **2**

2 Tasks **2**

3 Set the environment **4**

4 Progress **4**
4.1 Start the project 4
4.2 Using individual process 5

5 Project Problems and solutions **12**
5.1 Cannot find USB webcam ID 12
5.2 Cannot interconnect with the process 12
5.3 Write data to file 13
5.4 Make all processes work together 14

1 Introduction

This project deals with the control networks for robotics to allow a large number of sensors and computers to transfer information between them in real time.

Key considerations:

- Network topology changes frequently during the mission
- The system will support REAL TIME constraints, and bandwidth limitations (video)
- The system will be based on a network of computers, which are connected together
- Each station is running multiple processes in fixed cycles (time-horizon)
  - At the beginning of the process: sampling the latest relevant inputs
  - According to the algorithm, deciding priorities to address those inputs (or ignore old information)
  - performs the specific algorithm calculations
  - Publishing the outputs (Multicast messages)
- Project requirements
  - Set up a network with multiple cameras, multiple screens, multiple mixers and control switches
  - The cameras send their video streams on the network
  - The processing system will perform manipulations on the image
  - The screens will display the selected image in accordance with switches definition

2 Tasks

Set up a network with multiple cameras, multiple screens, multiple mixers and panel switches. Able to add devices without disturbing the network

Cameras send the video streams over the network. The processing system performs manipulations on the images, for example: mixing, split-screen, three-dimensional display etc

The screens will display the selected image according to the switches (User Interface) input.

Need to define the messaging protocol and algorithm for each station in the system, deal with loss of messages (as UDP communication is not reliable) The

main controller should graphically display the network topology, and provide an interface that allows each node to update the algorithm by loading new code

Define the structure of messages that contain all relevant information

The system should support real-time messages that are published at regular intervals. Unlike ROS where communication is PTP and adding the listener creates a short pause when changing the topology (process the additional TCP protocol with the new station) In our system the communication never stops (assuming that the network's bandwidth is large enough). Adding a listener will not degrade the performance

It is advisable to run in parallel other processes, such as movement of the base towards the goal. Adding an image processor that identifies the target and reports them while instructing the SERVO motors to move towards the target

Start the project in the following orders [1]

- Start with one code on one computer. Don't care about networks
  - Takes picture from camera
  - Switch (User interface) defines what to do
  - Take command from a switch(which camera what command to the mux etc
  - Mux mixes 2 pictures and choose one based on algorithm (for example side by side)
  - Screen to display it

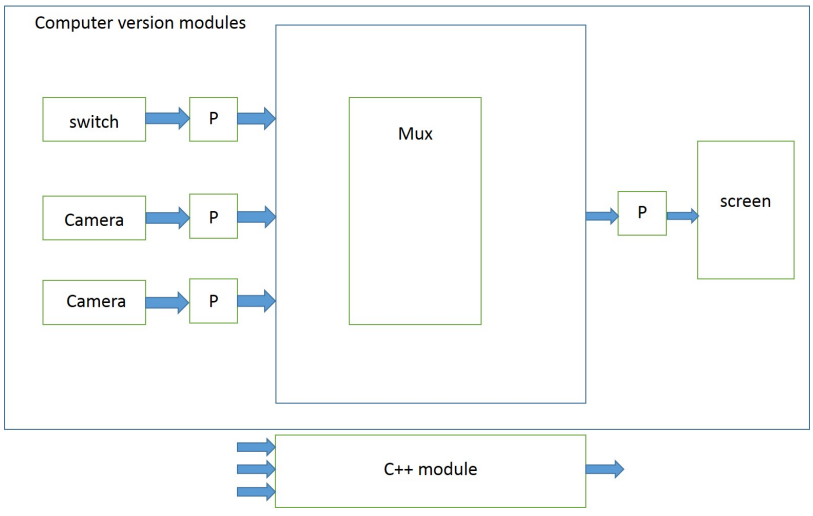

Figure 1: System diagram in memory

- Try to make each module (function) to be an independent process when running
  - 2 cameras take pictures and store pictures on disk as a jpg file
  - The Mux will choose one file based on the switch signal and store the file in disk as a file as well
  - The switch is another independent process
  - The display process just read the file from the Mux output and display it on screen
  - Each process should be an independent *.exe file in the working directory

3 Set the environment

I use Ubuntu 14.04 for the project. In order to set the working environment and get started, I follow this instruction [2].

4 Progress

The order of the work is based on the part I mentioned above. The first part is to implement the model in the Figure.

4.1 Start the project

- Take picture from camera

  By using the openCV library, we can implement it with the following code

```
VideoCapture cap1(0);
if (!cap1.isOpened())
{
    cout << "cannot open the camera \n";
    bFlagCM = true;
    return -2;
}

/*some details is not included,
  just the functions I used*/

bool bSucc1 = cap1.read(frame1);
imshow("YongxinWindow", frame);
```

- Switch the item

  The user interface is not available yet, but we can recieve command from the key board when the process is running

```
int key = waitKey(30);
cout << "Just get some outptut from waitKey function
    " << key << endl;

if (key == 27)
{
    cout << "esc key is pressed by the user\n";
    break;
}

else if (key == 99)
{
    frame = frame1;
    bIndi = true;
}

else if (key == 118)
{
    bIndi = false;
    frame = frame2;
}

else
{
    if (bIndi)
        frame = frame1;

    else
    {
        frame = frame2;
    }
}
```

Still many details are missing, but this is the main function for it to work properly

- All other functions actually have been implemented in the system, I will not shown them individually

4.2 Using individual process

For the final project, we need to make sure all the processes can work together and independent. So all the functions we mentioned above should be one individual process.

- Camera 1

Each camera/hardware for the network should be an independent process that can send message to the mux and let the switch know (currently I have no idea how this will be implemented)

The ID of the default camera is 0, so we have the following code for it.

```
Mat frame;
bool bSuccess = cap.read(frame);

if (!bSuccess)
{
    cout << "cannot read a frame from the camera\n";
    break;
}

fp=fopen("/home/yongxin/Documents/opencv/yossi/
    mulProcessRobot/camera1/status.txt","w");

if (fp == NULL)
{
    printf("File Not Found");
}

// need to consider read and write?
// one file for read, one for write and one for the
    next write
// what we have is one file in the memory (frame)
// new file and old file, which one is in memory....
// imWrite imageWriter("/home/yongxin/Documents/
    opencv/yossi/mulProcessRobot/camera1/camera1.jpg
    ", frame ,compression_params);

bool bSuccess2 = imwrite("/home/yongxin/Documents/
    opencv/yossi/mulProcessRobot/camera1/camera1write
    .jpg", frame, compression_params);

// if(!imageWriter.isOpened())
fputs("1",fp);
// this indicate a new file write
if(!bSuccess2)
{
    cout << "ERROR: failed to write the image file \n
        ";
    return -1;
}

imshow("YongxinWindow", frame);
```

```
if (waitKey(50) == 27)
{
    cout << "esc key is pressed\n";
    break;
}

fclose(fp);
```

Actually I have some problem when implementing the write process you mentioned in the email/last discussion.

Every time we need to store only one frame in the disk The readers will read the previous frame. Develop some mechanism to control it.

We write to file # and read from # 2.

When we finish writing to # 1 we need to start writing to file #2 instead, and the new readers will read from # 1.

Actually you need three files, one reading, one writing, and the third for the next write, as someone is reading from the second file.

A file will hold a single frame (picture), you can use JPG of BMP.

Files are marked Read/Write or /Old/New. You don't have to mark them you can see the file time stamp or change the file names

Step 1- reset
write to file Write
Read from file Read

Step 2 – new write
Write close the Write file and mark it New
Write start writing to the old file.

Step 3 - New Read
Read close the Read file and mark it Old
If there is a new File read start reading from it
If nor we wait.

The method I used here is by writing data to a file. For example, in the code, I write 1 to a status file when there is new frame/picture come. While for the mux, when it read the pciture, it will reset the data in this status file.

- Camera 2

  Camera 2 shares the same code as camera 1. Just the picture name and folder path are different. So I will not illustrate it here.

- Switch

  / ∗The purpose of switch is just to write command to a file.
  ∗And the mux will read the file and based on the result read

relevant pictures
*char 'c' for camera1
*char 'v' for camera2
*the file number is 1 and 2. We choose image based on this
*char 'esc' is used to stop the process
*/

The code for the switch is in the following

```
Mat img(650, 600, CV_16UC3, Scalar(0,65530, 65530));

FILE *fp;

bool bFlag = false;
int key = 0;
char chControl;

while (1)
{
    // if use parameter input, the process will strop
        ...
    fp=fopen("/home/yongxin/Documents/opencv/yossi/
        mulProcessRobot/switch/status.txt","w");
    // fp=fopen("/home/yongxin/Documents/opencv/yossi
        /mulProcessRobot/switch/status.txt","w");

    if (fp == NULL)
    {
        printf("File Not Found");
    }

    // imshow("MyWindow", img);
    //display the image which is stored in the 'img'
        in the "MyWindow" window
    key = waitKey(100);
    // cin >> chControl;
    cout << "The input key is " << key << " and the
        Falg is " << bFlag <<"\n";

    // if (key == 99)
    if (key == 99)
    {
        // cout << "Write 1 to the file \n";
        fputs("1",fp);
        bFlag = true;
        fclose(fp);
    }
```

```
        // else if (key == 118)
        else if (key == 118)
        {
            // cout << "Write 2 to the file \n";
            fputs("2", fp);
            bFlag = false;
            fclose(fp);
        }

        else
        {
            if (bFlag)
            {
                // cout << "Write 1 to the file by
                    default\n";
                fputs("1", fp);
                fclose(fp);
            }

            else
            {
                // cout << "Write 0 to the file by
                    default\n";
                fputs("2", fp);
                fclose(fp);
            }

            if (key == 27)
            {
                cout << "esc key is pressed\n";
                break;
            }
        }
    }
```

Also the switch write 1 and 2 to a file as the input to mux to choose

- Mux

  Mux reads the control command from switch and based on the selection signal, it will read relevant file from camera 1 or camera 2. After read the frame/picture, it will reset the status file of the switch (the OS will deal with the read/write at the same time for the frame)

```
fstream openFile;
openFile.open("/home/yongxin/Documents/opencv/yossi/
    mulProcessRobot/switch/status.txt");
```

```
int nTemp = -1;

if (!openFile.is_open())
{
    cout << "Cannot open the file \n";
}

else
{
    openFile >> nTemp;
    cout << "Enter the loop with nTemp = " << nTemp
       << "\n";

    if (nTemp == 1)
    {
        img = imread("/home/yongxin/Documents/opencv/
            yossi/mulProcessRobot/camera1/
            camera1write.jpg",
            CV_LOAD_IMAGE_UNCHANGED);
    }

    else if (nTemp == 2)
    {
        img = imread("/home/yongxin/Documents/opencv/
            yossi/mulProcessRobot/camera2/
            camera2write.jpg",
            CV_LOAD_IMAGE_UNCHANGED);
    }

    else if (nTemp == 0)
    {
        img = Mat(650, 600, CV_16UC3, Scalar(0,65530,
             65530));
    }
    if (img.empty()) //check whether the image is
       loaded or not
    {
        cout << "Error : Image cannot be loaded..!!"
           << endl;
        //system("pause"); //wait for a key press
        // return -1;
    }

    else
    {
        cout << "display the image \n";
```

```
            // namedWindow("MyWindow", CV_WINDOW_AUTOSIZE
                ); //create a window with the name "
                MyWindow"
            imshow("MyWindow", img); //display the image
                which is stored in the 'img' in the "
                MyWindow" window

            bool bSuccess2 = imwrite("/home/yongxin/
                Documents/opencv/yossi/mulProcessRobot/
                mux/muxWriter.jpg", img,
                compression_params);

            if(!bSuccess2)
            {
                cout << "ERROR: failed to write the image
                    file in the mux foler \n";
                return -1;
            }

            // waitKey(0); //wait infinite time for a
                keypress
            // destroyWindow("MyWindow"); //destroy the
                window with the name, "MyWindow"
        }
        // openFile.close();
        // openFile.open("/home/yongxin/Documents/opencv/
            yossi/mulProcessRobot/switch/status.txt");
        openFile.seekp(0,ios::beg);
        openFile << 0;
        openFile.close();
    }
```

- disPlay

  Display function is easy, just display the frame stored in the mux function

```
    img = imread("/home/yongxin/Documents/opencv/yossi/
        mulProcessRobot/mux/muxWriter.jpg",
        CV_LOAD_IMAGE_UNCHANGED);
    if (img.empty()) //check whether the image is loaded
        or not
    {
        cout << "Error : Image cannot be loaded..!!" <<
            endl;
        //system("pause"); //wait for a key press
        // return -1;
    }
```

```
else
{
    // cout << "display the image \n";
    namedWindow("MyWindow", CV_WINDOW_AUTOSIZE); //
        create a window with the name "MyWindow"
    imshow("MyWindow", img); //display the image
        which is stored in the 'img' in the "MyWindow
        " window
    // waitKey(0); //wait infinite time for a
        keypress
    // destroyWindow("MyWindow"); //destroy the
        window with the name, "MyWindow"
}

key = waitKey(100);

if (key == 27)
{
    cout << "esc is pressed \n";
    break;
}
```

5 Project Problems and solutions

This section describes the problems I encountered in this project. Just use to trace the project process and the difficulties I had during this process.

5.1 Cannot find USB webcam ID

It's an annoying situation. When I am trying to using 2 cameras in the laptop, the openCV cannot find the relevant camera with the VideoCapture(int ID). ID 0 and ID 1 all corresponding to the same physical camera... (Test folder – tcRobot)

5.2 Cannot interconnect with the process

I encountered a strange phenomena: when running a process and display a image from that process. The process can decode the key number from the waitKey() function. However, when I don't display the image (video), the process cannot decode the char I typed on the keyboard. So I feel I cannot use the switch function very well...
We can get the correct key value using the waitKey() function.
The corresponding test file is – /home/yongxin/Documents/opencv/yossi/mulProcessRobot/switch

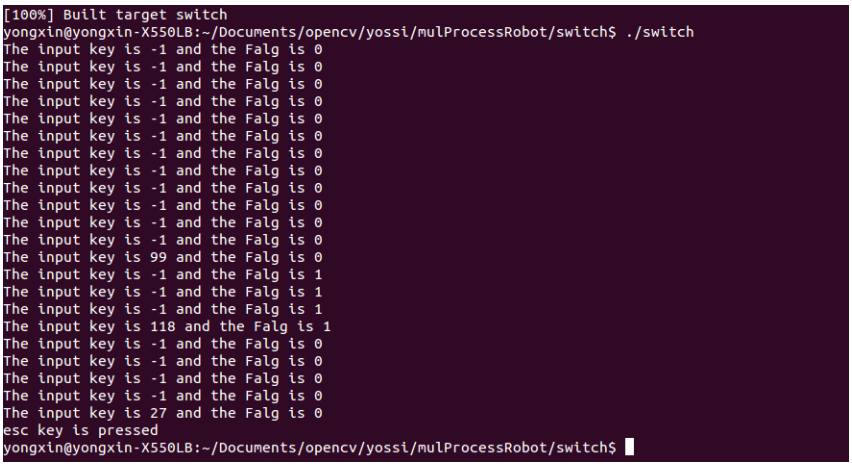

Figure 2: Capture the key when displaying an image

Now we cannot decode the key value, just the char symbol can appear on the screen.

5.3 Write data to file

The communication/message passing between processes are through the file folfer. However, when I write 0/1 to a file, I encountered very strange result. The code I am using is in the following

```
fstream openFile;
openFile.open("example.txt");
int nTemp = -1;
openFile >> nTemp;
cout << "Enter the loop with nTemp = " << nTemp << "\n";
// openFile.close();

// openFile.open("example.txt");
openFile.seekp(0,ios::beg);
openFile << 1 - nTemp << endl;
openFile.close();
```

Of course, the folder path is not the true path. For this code, if I didn't put the close() command/function before the write. It will not write the 0 to the folder. However, when I put the close() and open() in the code and run it, it will give the correct result. But if I comment the close() and open() again, it will write correctly. I have no idea to explain it ...
The relevant test files are in the folder /home/yongxin/Documents/opencv/yossi/mulProcessRobot/fileOO and folder /home/yongxin/Documents/opencv/yossi/mulProcessRobot/fileOP

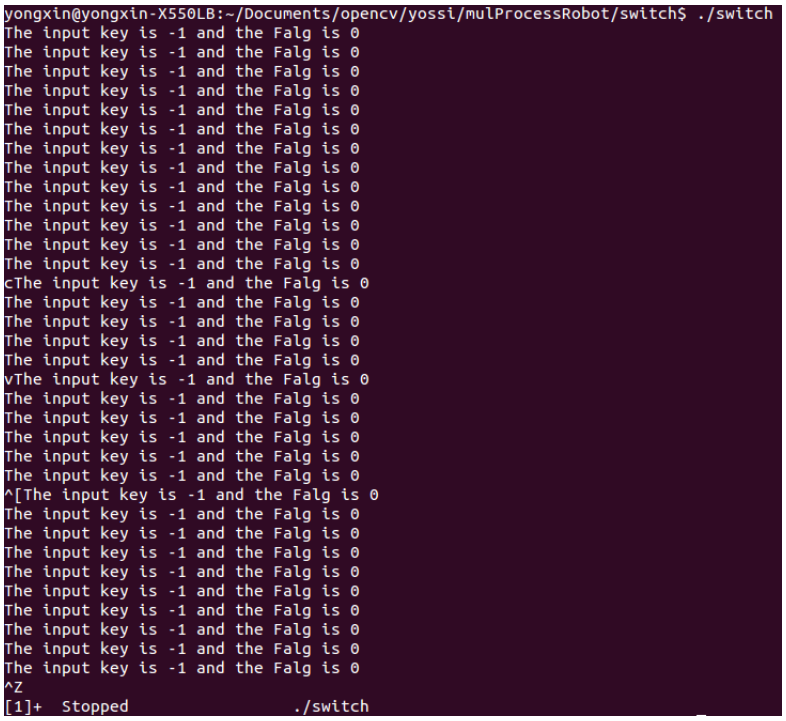

Figure 3: Capture the key when not displaying an image

5.4 Make all processes work together

Have no idea about how to make all processes work together

References

[1] Intel, "OpenCV open source community," http://opencv.org/.

[2] R. Berriel, "Installing OpenCV 3.0.0 on Ubuntu 14.04," http://rodrigoberriel.com/2014/10/installing-opencv-3-0-0-on-ubuntu-14-04/.