Progress review #11

Gauri Gandhi

Team G Robographers

Teammates:

Rohit Dashrathi

Jimit Gandhi

Tiffany May

Sida Wang

ILR #10

March 31, 2016

a. <u>Individual Progress</u>

For the eleventh progress review, I worked on integrating the complete detection sub-system using the new hardware with Sida. We also worked together to write a ROS node for writing data on Arduino for pan-tilt unit. I worked with Jimit to implement the localization of the turtlebots and implement the flocking algorithm.

I. Integrating Detection Sub-system with the hardware

For this progress review, I majorly worked on setting up the new Chromebooks and integrating the complete detection sub-system. I worked in the following order:

(i) The first thing I did was to install the Trusty version of Ubuntu 14.04 on each of the Chromebooks with Sida. To accomplish this, we completely cleared the ChromeOS partitions so that the complete 32GB space of the Chromebooks is available for Linux. After that I installed all the required software components like ROS and all of its supporting libraries required for our system.

(ii) After that Sida merged the code for extracting the single person of interest from the frame into the main code. I merged the code for voice commands into the main code for best image selection in the server computer. I also optimized the code for the best image selection by making it modular.

(iii) Once the code was ready for both the Chromebooks and the server computer, I built the codes in the respective machines. Then I made the final.launch files for each of the machines to launch all the server client launch files for ROCON. For the Chromebook clients, I included the nodes with the merged code for person tracking and emotion recognition in the final launch file. For the server computer, I included the node for the best image selection in the final launch file.

(iv) I created a start_rapps.bash file on the server to automatically do rosservice call for the three Chromebooks. Now, we can do the service calls on all the clients by simply sourcing the bash file once.

(v) After everything on the software front was done, I connected the respective Chromebooks with the turtlebots, the new cameras on the pan-tilt units and the Arduino units.

Hence, I finished the complete integration of the software and the hardware for the detection sub-system. The system can be launched by launching all the final.launch files on the respective machines and sourcing the start_rapps.bash file on the server. The Figure 1(a) shows the person of interest with the April Tag standing in front of the three turtlebots. The photo as shown in Figure 1(c) is clicked in just the single selected Chromebook.

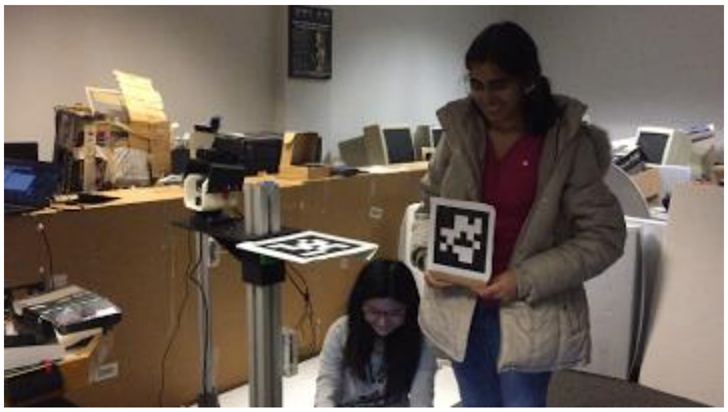


Figure 1(a) Person of Interest with the April Tag standing in front of the three robots

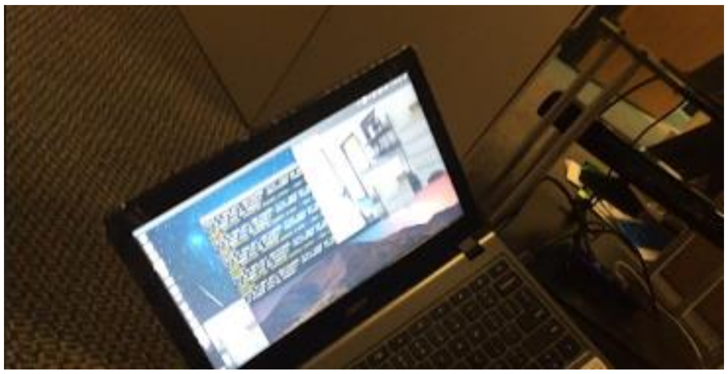


Figure 1(b) The robots getting lesser smile percentage do not click the photo

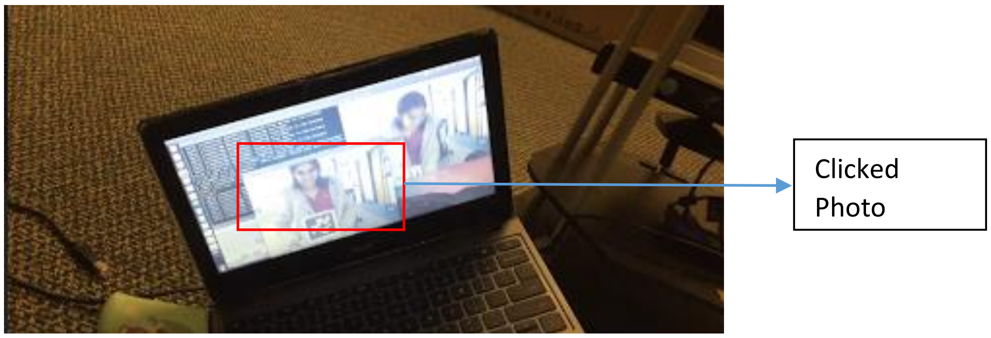


Figure 1(c)The robot getting maximum smile percentage clicks the photo

II. Understanding and implementing flocking

To implement flocking, Jimit and I first decided to understand our mentor Sasanka Nagavalli's localization and flocking code on Bitbucket. It really took us a lot of time to understand all the frame transformations involved between the world coordinates and the camera coordinates.

After that, Jimit implemented the flocking algorithm for the three turtlebots and I created the rapps for the ROCON setup between them. These rapps allowed the setup to subscribe to April Tag locations and odometry readings for each of the turtlebots and publish on their base velocities. I also changed the topic names involved in the communication to allow proper remappings through the rapps. To try out this complete framework for navigation, we used the other three turtlebots in the Advanced Agents Lab with netbooks.

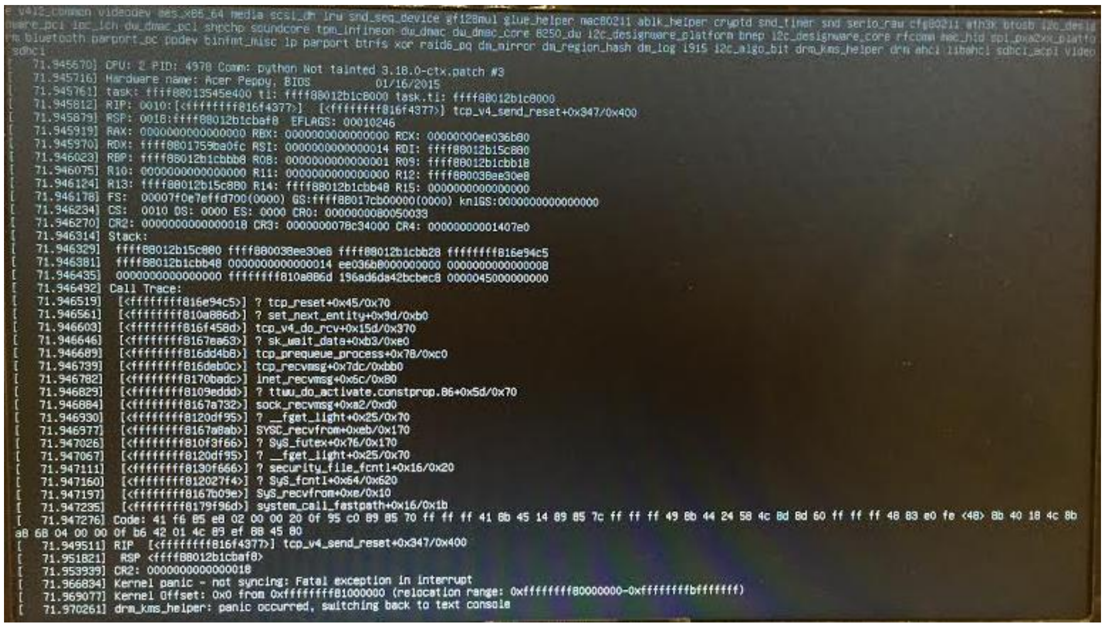
III. Node for writing data on Arduino

Sida proposed to operate the pan-tilt unit by using the x coordinate of the April Tag pose for controlling the pan motor and the y coordinate of the landmarks associated with the nose obtained from the Intraface for the tilt motor. She and I worked together to write a node that subscribes to both the April Tag pose data and the Intraface landmarks data. The April Tag x coordinate data is of the type Float64. The landmarks data is of the type Integer. The node then writes the different types of information on the same Arduino port.

b. <u>Challenges</u>

I. The major challenge faced by us this time was the unlikely crashing (Figure 2) of the three new Chromebooks each time we launched the ROCON master-client files. We could perform hardly two or three successful runs before the final demonstration when all the three laptops worked without crashing. At first, we thought that the problem was arising due to some video buffering problems. Hence, we decided to work with the USB cameras each time we launched the complete system. But that didn't help. After the demonstration, we discussed the problem with Sasanka and found out that the present version of Ubuntu on the Chromebooks is not able to support the ROCON network configurations completely. We now plan to change the versions of the Ubuntu on each of the three laptops. We also plan to keep our own laptops as a backup for the final demonstration.

II. Another major issue I faced this time while integrating everything was when I was trying to ssh into each of the three Chromebooks from the server (my laptop). I was doing to this to launch the final launch files in each of the clients. But as soon as I did this, my laptop would hang each time. The reason behing this was that each client laptop was streaming atleast three video streams as part of processing. All the video streams were getting transferred on my laptop at the same time. This meant nine video streams getting transferred over a single network to a single master machine. For the demonstration purposes, I preferred to individually launch the required files in each of the clients without doing ssh. Now, I will change all the codes that are opening up multiple image windows such that only the final clicked video is opened up on each of the clients and transferred to the server.



```
e v4l2_common videodev aes_x86_64 media scsi_dh lru snd_seq_device gf128mul glue_helper mac80211 ablk_helper cryptd snd_timer snd serio_raw cfg80211 ath3k btusb i2c_desig
nware_pci ipc_ich dw_dmac_pci shpchp soundcore tpm_infineon dw_dmac dw_dmac_core 8250_dw i2c_designware_platform bnep i2c_designware_core rfcomm mac_hid spi_pxa2xx_platfo
rm bluetooth parport_pc ppdev binfmt_misc lp parport btrfs xor raid6_pq dm_mirror dm_region_hash dm_log i915 i2c_algo_bit drm_kms_helper drm ahci libahci sdhci_acpi video
 sdhci
[   71.945670] CPU: 2 PID: 4978 Comm: python Not tainted 3.18.0-ctx.patch #3
[   71.945716] Hardware name: Acer Peppy, BIOS          01/16/2015
[   71.945761] task: ffff88013545e400 ti: ffff88012b1c8000 task.ti: ffff88012b1c8000
[   71.945812] RIP: 0010:[<ffffffff816f4377>]  [<ffffffff816f4377>] tcp_v4_send_reset+0x347/0x400
[   71.945879] RSP: 0018:ffff88012b1cbaf8  EFLAGS: 00010246
[   71.945919] RAX: 0000000000000000 RBX: 0000000000000000 RCX: 00000000ee036b80
[   71.945970] RDX: ffff8801759ba0fc RSI: 0000000000000014 RDI: ffff88012b15c880
[   71.946023] RBP: ffff88012b1cbbb8 R08: 0000000000000001 R09: ffff88012b1cbb18
[   71.946075] R10: 0000000000000000 R11: 0000000000000000 R12: ffff880038ee30e8
[   71.946124] R13: ffff88012b15c880 R14: ffff88012b1cbb48 R15: 0000000000000000
[   71.946178] FS:  00007f0e7effd700(0000) GS:ffff88017cb00000(0000) knlGS:0000000000000000
[   71.946234] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[   71.946270] CR2: 0000000000000018 CR3: 0000000078c34000 CR4: 00000000001407e0
[   71.946314] Stack:
[   71.946329]  ffff88012b15c880 ffff880038ee30e8 ffff88012b1cbb28 ffffffff816e94c5
[   71.946381]  ffff88012b1cbb48 0000000000000014 ee036b8000000000 0000000000000008
[   71.946435]  0000000000000000 ffffffff810a886d 196ad6da42bcbec8 0000045000000000
[   71.946492] Call Trace:
[   71.946519]  [<ffffffff816e94c5>] ? tcp_reset+0x45/0x70
[   71.946561]  [<ffffffff810a886d>] ? set_next_entity+0x9d/0xb0
[   71.946603]  [<ffffffff816f458d>] tcp_v4_do_rcv+0x15d/0x370
[   71.946646]  [<ffffffff8167ea63>] ? sk_wait_data+0xb3/0xe0
[   71.946689]  [<ffffffff816dd4b8>] tcp_prequeue_process+0x78/0xc0
[   71.946739]  [<ffffffff816deb0c>] tcp_recvmsg+0x7dc/0xbb0
[   71.946782]  [<ffffffff8170badc>] inet_recvmsg+0x6c/0x80
[   71.946829]  [<ffffffff8109eddd>] ? ttwu_do_activate.constprop.86+0x5d/0x70
[   71.946884]  [<ffffffff8167a732>] sock_recvmsg+0xa2/0xd0
[   71.946930]  [<ffffffff8120df95>] ? __fget_light+0x25/0x70
[   71.946977]  [<ffffffff8167a8ab>] SYSC_recvfrom+0xeb/0x170
[   71.947026]  [<ffffffff810f3f66>] ? SyS_futex+0x76/0x170
[   71.947067]  [<ffffffff8120df95>] ? __fget_light+0x25/0x70
[   71.947111]  [<ffffffff8130f666>] ? security_file_fcntl+0x16/0x20
[   71.947160]  [<ffffffff812027f4>] ? SyS_fcntl+0x64/0x620
[   71.947197]  [<ffffffff8167b09e>] SyS_recvfrom+0xe/0x10
[   71.947235]  [<ffffffff8179f96d>] system_call_fastpath+0x16/0x1b
[   71.947276] Code: 41 f6 85 e8 02 00 00 20 0f 95 c0 89 85 70 ff ff ff 41 8b 45 14 89 85 7c ff ff ff 49 8b 44 24 58 4c 8d 8d 60 ff ff ff 48 83 e0 fe <48> 8b 40 18 4c 8b
a8 68 04 00 00 0f b6 42 01 4c 89 ef 88 45 80
[   71.949511] RIP  [<ffffffff816f4377>] tcp_v4_send_reset+0x347/0x400
[   71.951821]  RSP <ffff88012b1cbaf8>
[   71.953939] CR2: 0000000000000018
[   71.966834] Kernel panic - not syncing: Fatal exception in interrupt
[   71.969077] Kernel Offset: 0x0 from 0xffffffff81000000 (relocation range: 0xffffffff80000000-0xffffffffbfffffff)
[   71.970261] drm_kms_helper: panic occurred, switching back to text console
```

Figure 2 The Crash Screen

c. <u>Team Work</u>

For this progress review, I collaborated with Sida to complete the integration of the detection sub-system. Sida worked on the voice command feature for a single robot system. Jimit and I worked together on the localization and flocking part. Jimit and Sida also helped me in setting up the network and launching all the required files for the integrated detection sub-system for every trial. Rohit finished the manufacturing of all the hardware components for the system. He also worked on improving the aesthetics of the system. Tiffany worked on integrating the Arduino code with the ROS code. She also worked with Sida to make all the Arduino code ROS compatible.

d. <u>Future Plans</u>

For the next and final Progress Review, I will be working on the following tasks:

I. Implement the flocking of the three turtlebots and localizing them using the April Tags and the global cameras set up in the Advanced Agents Lab. I will be collaborating with Jimit and Sida for this task.

II. Integrating the navigation sub-system with the detection sub-system.

III. Updating the version of Ubuntu on each of the new Chromebooks with Jimit.

IV. Debug the sound play code such that it does not run on a loop.

As a team, we will finish the following goals:

I. Calibrate all the new cameras and the three turtlebots.

II. Integrating the pan-tilt functionality with the main system.

III. Implement arrangement of the robots around the person of interest.

IV. Improve the aesthetics.

e. References

1. Turtlebot ROS tutorials
wiki.ros.org

2. ROCON tutorials for ROS Indigo
wiki.ros.org/rocon/indigo/Guide

3. Gateway model tutorials
http://redmine.robotconcert.org/projects/multimaster/wiki/Gateway_Model

4. Sasanka’s bitbucket repository
https://bitbucket.org/snagavallis/turtlebot-swarm-scripts/