Snackbot: Vision and Perception with Video and Audio Captures Using GStreamer

Hasani Burns and Chutima Boonthum-Denecke

Department of Computer Science
Hampton University
Hampton, VA 23668
hasani.burns@gmail.com, chutima.boonthum@hamptonu.edu

Abstract

The Snackbot, is a robot designed in collaboration between the Robotics Institute, and the Human Computer Interaction Institute of Carnegie Mellon University. The Snackbot was created to traverse the halls of Carnegie Mellon University, and deliver food items ordered by occupants of the offices. The goal of this development project for the Snackbot, was to refine the audio/video synchronization, and to also create a simple way to log, and stream that data over a network. Such a task requires that one not only carefully consider different pieces of software to use, but also that they can apply it across the necessary platform. For the Snackbot, the sight, and sound are important qualities, especially when testing out in the field using an operator. That ability is crucial when preparing an interactive robot to autonomously carry out its task efficiently.

Introduction

The Snackbot (Lee et al. 2009) needed to be refined from a software, backend side more so than on the physical, hardware side. The GUI needed to be revamped, as the Snackbot for the time being was a remote-controlled, remote-operated machine. Both the movement, as well as the dialogue were controlled through a user interface. Prior to my arrival for research, the team working on the Snackbot had some issues with the audio/video of the robot. The radio cameras, as well as audio receiver were out of synchronization which was an issue, as well as the fact that they were unable to securely and efficiently stream what was being captured from the Snackbot's eyes back to the operator. The only way to see how things turned out fully, was to record with the radio cameras the Snackbot used, and then play it back at the end. My job, as well as my partner's was to create a working, and easy to use set of code commands for capturing, and streaming audio and video feedback from the Snackbot during its trial runs. This would be not only to look through the robot's eyes when testing, but also to save that test as a form of video logging.

The Hypothesis

For a robot such as this to perform efficiently, as well as stream data over a network to an operator, the ability for the robot to "see" and "hear" is absolutely necessary.

There were many ways we could have gone about finding the software to use for this. At the time, we were working with Ubuntu Linux 10.04, while the other developers were using an older version, in 8.10. That being said, we not only had to find a piece of software that was easy to use, but one that was compatible with different versions of Ubuntu Linux, or could be easily converted or modified to work with them. The robot though, would need to be able to stream synchronized audio and video data in real 1 to 1 time, and as clearly as possible. This would be necessary in order to send the robot into the field, run tests, and see exactly what's going on through the eyes of the robot. With this, the operator would be able to monitor any and all interaction and responses recorded by the Snackbot.

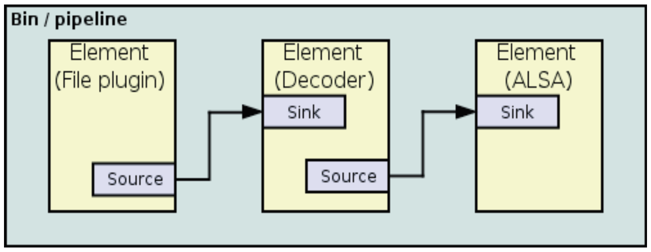
The Task

Early on, we come across GStreamer, a pipeline-based multimedia framework written in the C programming language. GStreamer basically opens the doors for a programmer to create a variety of media handling and manipulating components including what we needed audio/video playback, recording, streaming, and editing. When we first discovered this, we searched and found variations, augmentations of this software using Python programming. For a time, we did have working code that could separately record, send data, receive data, and log.

When telling our supervisor about our progress, he advised us that we should convert that code to C++ code.

Gstreamer processes media through a pipeline system, connecting a number of processing elements, each provided by an appropriate plug-in. Many of these plug-ins can be easily installed via the Ubuntu Linux Synaptic Package Manager.

This is an example of a filter model, which displays data as it is encoded, then decoded, then finally sent to ALSA drivers that in our case, were used to synchronize the data and feed it to whatever form of output necessary.



In order to start building our necessary pipeline, we used a simple workflow in which we defined the source, and then specifying a sink at the end. A sink can be anything from a file, to your own laptop or desktop monitor screen, to a network and use your computer as a streaming server.

```
$ gst-launch v4l2src device=/dev/video0 ! \
'video/x-raw-
yuv,width=640,height=480,framerate=30/1' ! \
xvimagesink
```

The above code, gst-launch represents the initiation of the GStreamer software, and it's using v4l2src, or video4linux2 as a stream source. The next command specifies the device, here it corresponds to one's laptop webcam, though they do vary. In the commands, it's also specified that video is wanted, and the height and width of the display is put in, as well as the frame rate. At the very end, everything is linked to xvimagesink. This sink displays the stream on your current screen. The above commands, instead of placing them in the terminal, can be placed in scripts, and executed from the terminal. This is the foundation for using GStreamer on linux to capture and manipulate audio and video.

```
$ gst-launch v4l2src ! ‘video/x-raw-
yuv,width=640,height=480,framerate=30/1' ! \
queue ! videorate ! 'video/x-raw-
yuv,framerate=30/1' ! theoraenc ! \ queue !
oggmux ! filesink
location=my_first_video.ogg
```

In order to not only access the video, but also record, and be able to play it back, there needs to be another element added to the pipeline. That next element is videorate, which will take each frame of the input and feed it to the next element at the frame rate that's requested. The data is then linked to theoraenc, which incodes the raw video into a theora stream.

Theoraenc can be translated to theora encoding. From there, the link is made to oggmux, which takes the stream, and places it into an ogg container. Gstreamer is also capable of containing the audio and video data in other formats, such as avi, mp4, and flv. This is necessary because GStreamer can very easily and smoothly create ogg files. Finally, the data is all linked to a filesink, which will write all of the data to a file that can be opened and played back.

This proved to be a difficult task, but along the way, we discovered something else. GStreamer itself, had launch commands built into it. These commands could activate the video from our webcams straight from the command terminal. We simply placed these commands into shell scripts, made them executable, and executed them from the command line. The major problem though, was putting together one script for displaying, one for recording, one for saving, and one for sending and receiving the synchronized audio and video feedback.

Currently, we have uncovered much of GStreamer, and have put together multiple shell scripts working with Linux, to display video, as well as stream with a simple running of our “send” and “receive” files. We're also able to, while receiving the stream, save the data to an .ogg file, for a virtual video log. There are still improvements that needs to be done.

Aside from this, we hope to refine the concept of the Snackbot even further by making its social, more interactive side, even that much more smooth and human like. This includes things such as facial recognition, and more in depth dialogue. The Snackbot though, has the potentiality of not only becoming much more than a robot that delivers snacks, but also being a beacon of innovation for robotics in general. A robot that can traverse, recognize, interact, and respond to humans, and real life situations, is where the future of robotics lies.

In this poster, we will present the work that was accomplished in developing the vision and perception using GStreamer.

Acknowledgments. This research was supported in part by the National Science Foundation (IIS-0624275, CNS 709077, CNS-0742252). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or corporate partners. A special thanks to Dr. Sara Kiesler, Dr. Paul Rybski, and SnackBot team at Carnegie Mellon University for their ARTSI REU support.

References

Lee, M.K., Forlizzi, J., Rybski, P.E., Crabbe, F., Chung, W., Finkle, J., Glaser, E., and Kiesler, S. (2009) The Snackbot: Documenting the design of a robot for long-term human-robot interaction. In Proceedings of HRI 2009, 7-14.