3rd Person Dash-Camera (3PDC): Georgia Edition

CS 4365 Semester Project

Usage

After completing the setup instructions, 3PDC should be ready for use. In order to manually invoke the code, from the `/Scheduler` directory of 3PDC, run `python3 scheduler.py`. This will start up the scheduler, run the initial setup such as building the geojson and setting up logging, and then enter into the check for broadcast state.

Details about the status of the scheduler are tracked in the `scheduler.log` file in the root of the 3PDC Archive Directory on the NAS, or in the root of the project directory if the NAS directory is not available. This file can be read with a text editor in real-time to ensure that the scheduler initialized properly and is ready to capture and record a session. Each session will be saved in a directory structure with date/time, tracked person name, camera name, and event number metadata. At the top level of the session folder will be a session log containing all of the coordinates captured, distances and speed calculated, and cameras invoked.

On the side of the phone, simply open Google Maps, click on your profile icon in the top right corner, select Location Sharing from the list of options, type in the email address of the account connected to the Pi or select it from a list of previously shared with accounts, set the sharing duration (should be longer than expected driving session time), then click share. Within the next check for broadcast polling interval (set in config file), 3PDC should detect that a session has been started and write to the log file as well as create a folder on the NAS for the session. It is strongly encouraged that you use a Maps navigation session at the same time for best location accuracy, even if you know where you are going. Failure to do so means that location data is only updated every 3 minutes. At the end of the drive, simply go back to locating sharing and stop sharing with your Pi. The Pi will then finalize the session and reset the state machine in preparation for the next session.

For a more permanent setup, 3PDC can be configured to run at boot so it is always ready as long as the Pi is on. Additionally, a reboot with rebuild the geojson if something gets out of sync. To do this run `sudo nano /etc/rc.local` to edit the startup scripts. Then insert the following inside, taking care to use your specific path:

```
sudo python3 <path_to_3PDC_root>/Scheduler/scheduler.py &
```

3PDC will now auto-run at boot and can be checked on remotely on LAN via the NAS. This includes checking the status in the scheduler log as well as viewing captured footage. If any changes are made to the cookie file, config file, or if the geojson data is stale, a reboot of the Pi will restart 3PDC and catch the changes.

The state machine for the scheduler is as follows:

1. Check for Broadcast (waiting for location data)
2. Initialize Session (got location data, start session)
3. Handle Session (location data is coming in and the handler is recording cams according to location and the algorithm)

4. Teardown/Reset (session has finished, state machine reset)

   Back to #1

Setup

Materials Needed:

1. Raspberry Pi 3B+ (others may work but are untested)
2. MicroSD Card for Boot Drive (Min 16GB, Recommend Class 10)
3. USB Type-A Storage Device (Necessary size depends on number of cams and time to store, we are using 128GB flash storage drives)
4. 5V Power Supply for Pi (refer to model specific amperage recommendations)
5. A Google Account dedicated to your Raspberry Pi (For location sharing component)

Software Installation:

1. Install Raspberry Pi OS to MicroSD Card to boot a clean instance. You can reuse a previous install but some pre-existing applications may interfere with the 3PDC code.

2. (Optional) - Enable VNC and/or SSH on the Pi for remote configuration. You may opt to use the Pi locally if you choose.

3. Run the following command to get the latest repository updates for the following software installs: `sudo apt-get update` .

4. Check that Python 3 is installed using the following command: `$python3 --version` . Our OS came preinstalled with Python 3.7.3 however anything newer than this should be supported. If issues arise, consider matching our version.

5. Install & Configure the Samba NAS Software using the following steps:

   1. Run the following command: `sudo apt install samba samba-common-bin`
   - Answer yes/y for the installation as well as the extra networking component (if asked)

   2. Next backup the Samba Config file before changing for future reference: `sudo cp /etc/samba/smb.conf /etc/samba/smb.conf_backup`

   3. Open `/etc/samba/smb.conf` using `sudo nano /etc/samba/smb.conf` and add the following lines at the end:

   ```
   [RaspberryPi NAS]
   comment = Pi Server
   public = yes
   writeable = yes
   browsable = yes
   path = /mnt/usbdrive/NAS
   ```

   - While still in `smb.conf` find the following individual lines under the "Share Definitions" section after the respective comments and replace the existing value with 0777:

"# File creation mask..."

`create mask = 0700`

"# Directory creation mask..."

`directory mask = 0700`

- See the `Example Files` folder in repository for an example conf file. You can opt to replace your `smb.conf` with the sample and skip the modifications made in step 3 if desired.

4. With your USB Drive Connected to the Pi, run the following command: `sudo blkid`

- You should get something similar to the following:

```
/dev/mmcblk0p1: LABEL_FATBOOT="boot" LABEL="boot" UUID="XXXX-XXXX"
TYPE="vfat" PARTUUID="XXXXXXXX-XX"
/dev/mmcblk0p2: LABEL="rootfs" UUID="XXXXXXXX-XXXX-XXXX-XXXX-
XXXXXXXXXXXX" TYPE="ext4" PARTUUID="XXXXXXXX-XX"
/dev/sda1: LABEL="SanDiskUSB" UUID="1234-ABCD" TYPE="exfat"
PARTUUID="XXXXXXXX-XX"
/dev/mmcblk0: PTUUID="XXXXXXXX" PTTYPE="dos"
```

- Find the corresponding UUID for your USB Drive. All ID's in the example have been redacted or replaced however the target for our USB in this exaple has a UUID of "1234-ABCD". Copy this UUID for the next step.

5. Using the UUID from Part 4, open the fstab file using the following command: `sudo nano -Bw /etc/fstab`.

Warning! Any invalid changes in this file may render the Pi unbootable. Procced with caution. Using the -Bw flag with nano provides a backup to revert to in case of damage.

- Insert the following line at the end of the fstab, taking care to set the UUID to your specific USB Drive's UUID found in step 4 :

```
UUID=1234-ABCD /mnt/usbdrive auto defaults,nofail,umask=000,noatime 0 2
```

- See the `Example Files` folder in repository for an example fstab file. You cannot reuse this as your UUID values will be different for both internal and external storage, however it may give you an idea of what the fstab should look like after modification.

6. Navigate to your USB Drive via the existing mount-point under `media\pi\<your_drive_name>` and create the following directories:

```
<your_drive_name>/NAS
<your_drive_name>/NAS/3PDCArchive
```

- Running the `tree` command from the root of the USB Drive should yeild the following output if the directories were created properly. The System Volume Information folder is negligible and simply an artifact of formatting the USB Drive as FAT32 on Windows.

```
$ tree
.
├── NAS
│   └── 3PDCArchive
└── System Volume Information
    ├── IndexerVolumeGuid
    └── WPSettings.dat
```

7. Now that the Samba NAS is configured and the proper directories are created and mapped, restart the Pi. Alternately, you can opt to restart Samba if the USB Drive is already mounted at `/mnt/usbdrive` using `sudo service smbd restart`

- The NAS should now be exposed on the local network. You can connect to it using your file explorer. On Windows, open Explorer and go to "This PC". From the top ribbon under the "Computer" tab, select "Add a network location". From here, a popup will go through a couple of explanation pages before asking for an input address. Browsing will sometimes work but if everything is configured properly you should be able to use the following address: "`\\raspberrypi\RaspberryPi NAS`". If a connection cannot be established, try swapping `raspberrypi` with the IP of your Pi. Once done, you should be able to connect to the NAS and see the `3PDCArchive` folder within. Test reading and writing files to this location just to be sure it is working as expected. If so, you have completed this part of the setup.

6. Install Python Dependencies

- Use the following commands to install the necessary Python dependencies to run the code:

```
$ pip3 install locationsharinglib
$ pip3 install pandas
$ pip3 install sklearn
$ pip3 install opencv-python
$ pip3 install numpy
$ pip3 install requests
$ sudo apt-get install libatlas-base-dev
```

7. Generate Google Maps Cookie File (Location Sharing Authentication)

In order for the location collecting portion of 3PDC to work properly it must be able to authenticate using the Google Account created for the Pi. In order to accomplish this, it uses a combination of the email address in the `config.py` file (filled out later), along with a cookie file. You can generate your specific cookie file using these steps:

1. In your Pi's web browser or your local machines browser, install the Get cookies.txt extension. This requires Chrome or Chromium however you are free to try other methods (untested) of generating the cookie file if you'd like.

2. Once installed. Sign out of all active Google Accounts.

3. Navigate to maps.google.com and sign in using the account created for locating sharing with the Pi.

4. Once authenticated, while still on the Google Maps landing page, click the extension and have it export a cookie file. The format will be something like `google.com_cookies.txt`. You may opt for a different filename however it will need to be changed in the config file if it is different than the default above. Save or copy the file to the `Scheduler` directory in the project. This should replace the temporary/template file.

8. Update the `config.py` file with the following details:

> set `person_full_name` to equal the name on the Google Account of the end user (the phone)
> set `rpi_google_email` to equal the email of the Google account used by the Pi (must be the same account as what generated the cookie file)

The config file also includes other parameters to adjust such as platform, max cam number, the GA 511 endpoint, and the check for broadcast polling interval however these should not be changed unless necessary for testing, diagnostics, or development.