Car Behavioral Clonning

Esse relatório tem o intuito de descrever as etapas utilizadas para criar uma conexão entre um computador com Ubuntu e o Jaguar (robô utilizado para o projeto) através do software ROS e adaptar o algoritmo de carro autônomo no Jaguar.

1. Rede Wireless + Câmeras

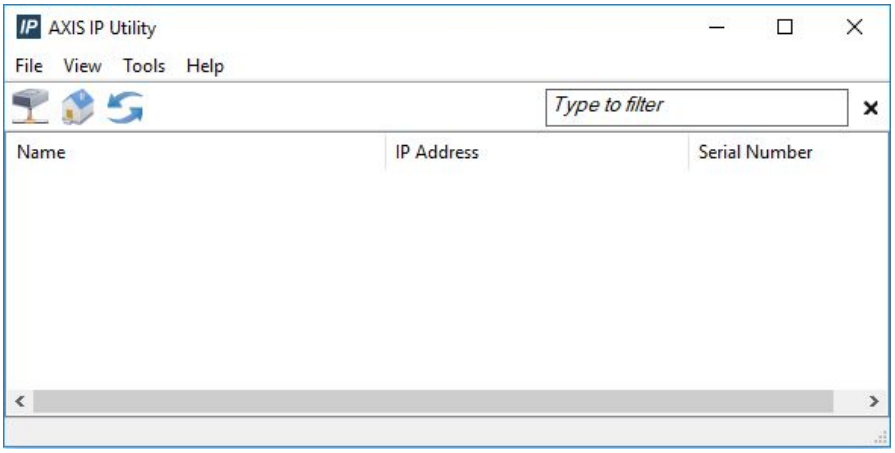
1.1 Consultar o manual de operação dos switches, antena, módulos IP e câmeras para melhor entendimento dos dispositivos.

Através dos manuais foi possível visualizar os parâmetros necessários para a configuração dos dispositivos, assim como para entender como que eles funcionam e interagem entre si. Todos os manuais necessários estão em "code" nesse relatório.

1.2 Configurar rede com IP estático dos dispositivos conforme manual do jaguar

Para configurar IP estático será necessário baixar o programa IPUtility (http://www.intelbras.com.br/empresarial/monitoramento/softwares-aplicativos/intelbras-ip-utility).

Ao abrir o programa, a seguinte tela irá aparecer:



Nela serão detectados automaticamente os dispositivos presentes na rede, assim sendo possível também definir um IP estático para esses dispositivos, simplesmente clicando com o botão direito do mouse no dispositivo e selecionando a opção de alterar o IP.

1.3 Testar cada câmera isoladamente

Para testar as câmeras isoladamente é preciso estar conectado à rede wireless do Jaguar e acessar o IP referente à camera, no caso: http://192.168.0.65/.

2. Behavioral Cloning

Há quatro programas essenciais para o funcionamento do algoritmo do Behavioral Clonning: savefile.py, Model.py, Utils.py e Drive.py. Cada um desses arquivos possui uma função essencial para fazer com que o Jaguar consiga andar de forma autônoma em um determinado circuito.

Inicialmente será necessário baixar a pasta "behavioral-cloning-one-camera", nessa pasta será possível ter acesso as pastas Drive.py, Model.py, Utils.py, as imagens que forem registradas pelas câmeras, as planilhas "driver_log.csv" e os modelos que serão criados. Cada uma das pastas são devidamente explicadas nos tópicos seguintes.

Para acessar a pasta é necessário digitar no Terminal de comando: "cd" e arrastar a pasta para o Terminal, assim será entendido que se está acessando a pasta "behavioral-cloning-one-camera".

Para executar os programas em python no Terminal, basta digitar, por exemplo: "python Model.py" ou "python Drive.py"

2.1 Savefile.py

Esse arquivo tem a função de criar um node na rede do ROS e “ouvir” (receber determinados dados enviados pelo node) os comandos que são enviados pelo drrobot_cmd_vel, que no caso é o node que envia os valores de movimentação para “X” linear e o “Z” angular, movimentando o Jaguar.
Após recebido os valores de X e Z, o programa savefile.py captura a imagem referente a esses dados, salvando-a no diretório Data/IMG e armazenado os dados lineares, angulares e localização da imagem referente a cada imagem baixada em um arquivo .CSV, organizando de tal forma que o arquivo Model.py tenha fácil acesso para criação do modelo.

2.2 Model.py

Model.py: Serve para criar o modelo, possuindo parâmetros Epoch, repetições. Nesse programa serão utilizadas as fotos que o programa "savefile.py" gerou e os dados que a planilha "driver_log.csv" gerou, assim se criará o modelo utilizando Deep Learning, aprendendo com essa aplicação e determinando qual modelo (arquivos "model-002.h5", por exemplo) criado será o mais efetivo.

2.3 Utils.py

Utils.py: Serve para auxiliar o Model.py (o qual chama o Utils.py), porque a imagem precisa ser alterada, devido seu tamanho, o que gera muito trabalho para a inteligência artificial processar todas as imagens. Nesse projeto foram geradas aproximadamente 1000 imagens. Esse programa corta a imagem, mantendo aspectos relevantes da pista na qual o Jaguar irá se locomover, onde foi alterada a cor da imagem, colocando-a em uma escala de cor preto e branco. O intuito principal desse programa é facilitar o processamento das imagens pela inteligência artificial, já que o programa irá examinar pixel por pixel, onde quando há uma mudança nesses pixels, o computador já considera como uma imagem diferente da outra que ao olho humano é imperceptível.

2.4 Drive.py

Drive.py: Serve para poder controlar o Jaguar. Esse programa trabalhará em conjunto com os arquivos modelo ("model-002.h5", por exemplo) criados pelo programa Model.py, os quais irão se comunicar com o Node do Jaguar. Ambos irão enviar juntamente os comandos de direção e velocidade e também filmando o que está sendo visualizado pelas câmeras do Jaguar.

2.5 Ler via rede dados a aceleração enviada a cada motor do Jaguar (Drive.py)

No arquivo Drive.py é possível realizar o controle do Jaguar para assim realizar o treinamento de movimentação do mesmo, onde ele vai captar as imagens que forem capturadas pelas câmeras e vai salvá-las na pasta "behavioral-cloning-one-camera/Data/IMG".

2.6 Criar interface de treinamento da rede neural, salvando dados dos encoders e câmeras

Inicialmente se executa o arquivo Model.py, o resultado dessa execução são arquivos ".h"(hierarchical data format) numerados referentes aos melhores modelos criados.

O diretório das imagens e os valores associados a elas, chamados de labels, são armazenados em um arquivo .csv, que é um formato de arquivo de texto que pode ser usado para trocar dados de uma planilha entre aplicativos, e será utilizado na hora da criação dos modelos.

3. ROS

O ROS (Robotic Operation System) é uma coleção de frameworks de softwares para desenvolvimento de robôs que trabalha com serviços padrões de um sistema operacional, tal como abstração de hardware, controle de dispositivo de baixo nível, passagem de mensagens e gerenciamento de pacotes.
Seu funcionamento se baseia em nós, onde cada dispositivo da rede pode ter um ou mais nós e pode enviar ou receber pacotes de dados através da rede do ROS.
Para o presente trabalho foi utilizado o ROS Kinetic.

4. Configurando o ROS Kinetic

Para Instalar o ROS Kinetic pode-se seguir o passo a passo descrito no próprio site http://wiki.ros.org/kinetic/Installation/Ubuntu

Linhas de código necessárias:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

```
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
```

```
sudo apt-get update
```

```
sudo apt-get install ros-kinetic-desktop-full
```

```
apt-cache search ros-kinetic
```

```
sudo rosdep init
```

```
rosdep update
```

5. Configurando a Rede Neural

Para a correta utilização da Rede Neural é preciso ter instalado na máquina do usuário o python de versão 2.7 e 3.5, já que os arquivos .py funcionam em versões diferentes. Para a instalação dos pacotes do python é preciso ter o pip instalado tanto para a versão 2.7 quanto para à versão 3.5.
Pacotes para à versão 2.7:
-csv
-re
-os
-urllib.request
-datetime
-time
-rospy
-geometry_msgs.msg

Pacotes para à versão 3.5:
-argparse
-base64
-datetime
-os
-shutil
-cv2
-numpy
-socketio
-eventlet
-eventlet.wsgi
-PIL
-flask
-io
-urllib
-pandas
-keras
-argparse

É sempre recomendado à instalação dos pacotes antes da execução do programa, porém, não ausência de qualquer um desses pacotes será informado para o usuário no terminal ou na sua IDE.

6. Executando o Programa

Após a instalação de todos os pacotes e arquivos necessário, podemos dar início a execução do programa.

6.1 Conectando ao Jaguar

Como primeiro passo é preciso conectar-se ao Jaguar. Para isso, ligue e robô e conecte o seu dispositivo à rede wireless do Jaguar. Garanta que à senha da rede está correta e o IP do seu dispositivo esteja diferente de todos os outros dispositivos conectados à rede.

6.2 Iniciando o ROS

Para iniciar o ROS é preciso abrir o primeiro Terminal e iniciar o node Master do Ros. Para isso, digite “roscore” no Terminal.
Após inicializado o master, vamos executar os nós de player e keyboard (controle pelo teclado) ou joy (controle pelo joystick) para controlar o Jaguar e capturar os primeiros dados. Para isso, é preciso abrir um novo terminal e executar os comandos:
source (seu ambiente do ROS)/devel/setup.bash
rosrun drrobot_jaguarV2_player drrobot_jaguarv2_player_node
Pronto. Agora o player_node já deve estar em execução. Para executar o node do controle remoto, digite em um novo terminal:
source (seu ambiente do ROS)/devel/setup.bash
rosrun drrobot_jaguarV2_player drrobot_jaguarv2_keyboard_teleop_node
para usar o teclado, ou
source (seu ambiente do ROS)/devel/setup.bash
rosrun joy joy_node
para usar o joystick.
Após esses passos provavelmente o ROS estará em execução e já era possível controlar o Jaguar pelo teclado ou joystick.

6.3 Salvando as imagens e criando o driver_log.csv

Com o ROS em execução no seu dispositivo juntamente com o controle, já é possível controlar o Jaguar para capturar as imagens e dados de movimentação. Para isso, é preciso abrir um Terminal na pasta do behavioral-cloning-one-camera e executar o savefile.py. Para isso, podemos executar o arquivo .py por uma IDE ou pelo terminal, executando o seguinte comando:
python2.7 savefile.py
Lembrando que o savefile.py é executável apenas no python de versão 2.7.
Após a inicialização do arquivo já é possível salvar as imagens e os dados do arquivo driver_log.csv, basta apenas controlar o Jaguar pela rota que se deseja obter os dados. Lembrando que é recomendado ter muitas imagens(mais de 1000) para aperfeiçoar o treinamento do para o Jaguar.

6.4 Criando o Model-xxx.h5

Depois de criado o arquivo driver_log.csv e obtidos as imagens, é possível começar a criar os primeiros modelos para o controle autônomo do Jaguar. Mas antes de qualquer coisa sugerimos à edição das imagens pelo o arquivo utils.py que será importado pelo model.py. Para isso há vários métodos dentro do utils.py que podem ser editados para reduzir o tamanho da imagem capturada e assim reduzir o tempo de execução do código de criação de modelo.
No model.py é recomendado fazer várias execuções com diferentes parâmetros de 'number of epochs', 'samples per epoch', 'batch size' para criar vários modelos e poder ter várias opções de teste para o drive.py. Para executá-lo, acesse a pasta onde se encontra o model.py, abra um Terminal e execute o seguinte código:
python3.5 model.py
Lembrando que o model.py só pode ser executado no python de versão 3.5.

6.5 Executando o drive.py

Por final, é possível executar o drive.py para Jaguar ser controlado de forma autonoma. Para isso, é preciso estar conectado à rede wireless do Jaguar com os nós Master e player do ROS em execução.
Para executar o drive.py é preciso abrir um Terminal na pasta que se encontra o arquivo e digitar o seguinte comando:
python3.5 drive.py model-xxx.h5 (onde o ‘xxx’ é o número do modelo escolhido pelo usuário)
É recomendado o teste de vários modelos até achar o correto. Se não obtiver nenhum erro, provavelmente o jaguar estará sendo pilotado de forma autonoma.