Universidade Tecnológica Federal do Paraná (UTFPR)
Disciplina: CPGEI/PPGCA - Robótica Móvel

Experimentação Virtual

Prof. André Schneider de Oliveira
Prof. João Alberto Fabro

Experimentacao virtual

- São ambientes de simulação voltados a reproduzir o comportamento de sistemas reais em ambiente virtual

- Na robótica são aplicados para validar sistemas de locomoção, percepção, navegação e tomada de decisão

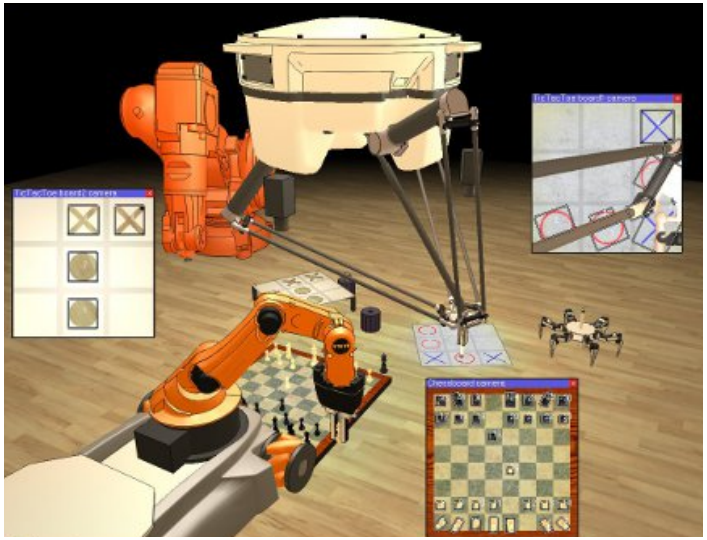
- Dentre os principais ambientes de experimentação para sistemas robóticos, com interface ROS, destacam-se

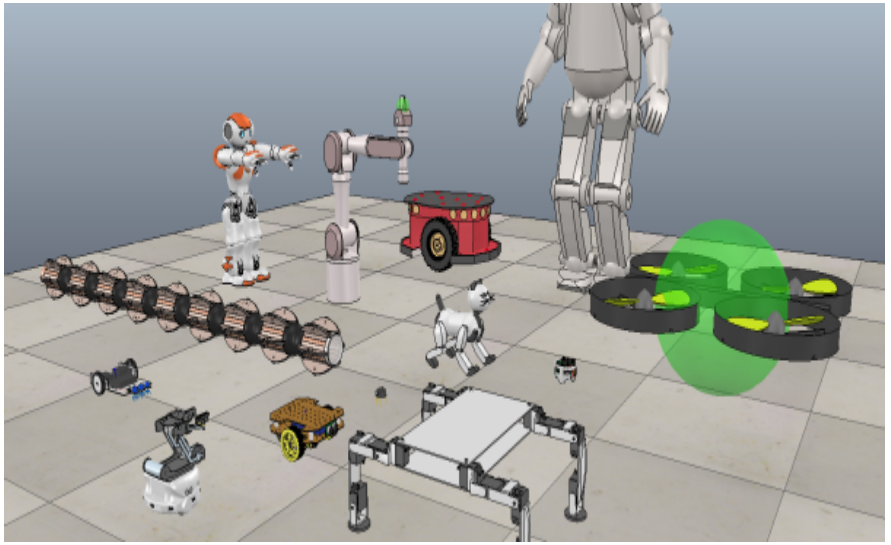
    1. **V-REP**
    2. **Gazebo**
    3. **Stage**

V-REP - Coppelia Robotics

Virtual Robot Experimentation Platform

http://www.coppeliarobotics.com/





V-REP - Coppelia Robotics

- Ambiente genérico para a simulação dos principais robôs, sensores e atuadores
- Modelos com comportamentos muito semelhantes aos componentes reais ***(utiliza engines de games)***
- Programação através de diferentes métodos, permitindo a integração com o ROS

V-REP - Coppelia Robotics

• A integração com o ROS depende da compilação de um plug-in externo

http://www.coppeliarobotics.com/helpFiles/en/rosTutorialIndigo.htm

• Para o ROS Kinetic pode-se baixar no site da disciplina o simulador com o plug-in já compilado **(no caso da maquina virtual já esta instalado)**

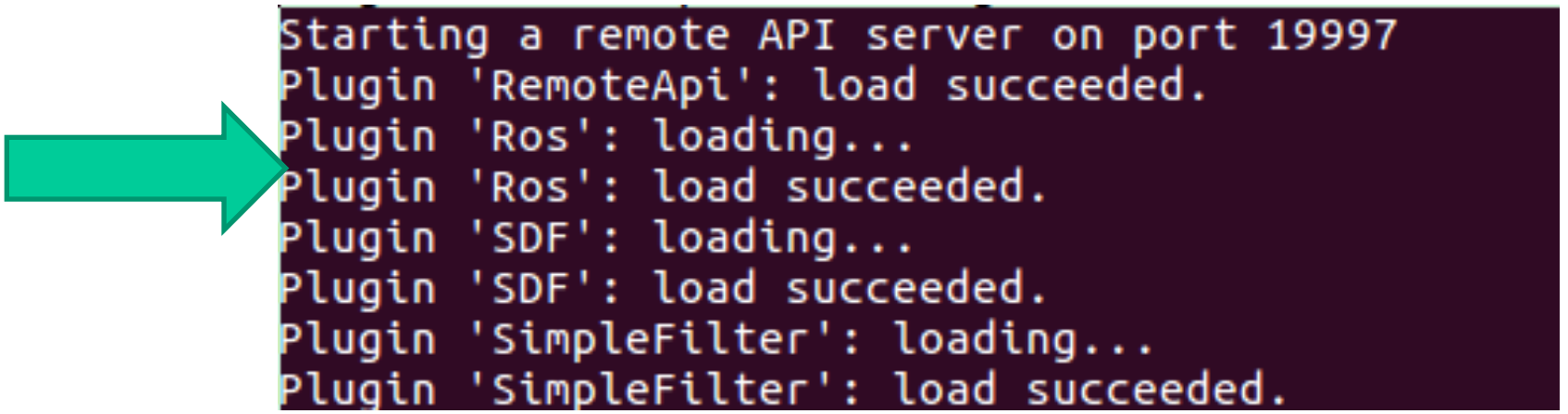
V-REP - Coppelia Robotics

• Para iniciar o ambiente entre na pagina do simulador e execute o comando

./vrep.sh (com o roscore ativo!)

**** sugestão descompactar dentro de ~/catkin_ws***

• O carregamento do plug-in ROS deve ser conferido no console



V-REP - Coppelia Robotics

- A integração com o ROS e realizada através de scripts internos programados em LUA

- Para todos os objetos deve-se armazenar o seu ***handle***

```
RightmotorHandle = simGetObjectHandle('Bob_leftMotor')
LeftmotorHandle =  simGetObjectHandle('Bob_rightMotor')
RobotHandle =  simGetObjectHandle('base_link')
OdomHandle =  simGetObjectHandle('odom')
```

- Depois é iniciado o **publisher** ou **subscriber**

```
simExtROS_enablePublisher('/odom',1,simros_strmcmd_get_odom_data,RobotHandle,OdomHandle,'OdomCovariance')
simExtROS_enableSubscriber('/cmd_vel',1,simros_strmcmd_set_twist_command,-1,-1,'cmd_vel_subdata')
```

Cinemática no V-REP

```
if (sim_call_type==sim_childscriptcall_actuation) then

    -- Put your main ACTUATION code here

     simExtROS_enableSubscriber('/cmd_vel',1,simros_strmcmd_set_twist_command,-1,-1,'cmd_vel_subdata')
     local packedData=simGetStringSignal('cmd_vel_subdata')
     if (packedData) then
          local twistData=simUnpackFloats(packedData)
              vx=twistData[1]
              w=twistData[6]
       end


    -- Base_controller --
    r = 1.0000e-01 -- (m) wheel radius
    L = 0.25

    Vright = - ((w*L)/(2*r)) + (vx/r)
    Vleft =  ((w*L)/(2*r)) + (vx/r)
    simSetJointTargetVelocity(LeftmotorHandle,Vleft)
    simSetJointTargetVelocity(RightmotorHandle,Vright)


end
```

Gazebo

- E um simulador interno ao ROS que normalmente já vem instalado no ***ros-full-desktop***

- A sua configuração se baseia em arquivos de descrição de modelos 3D, com objetivo de tornar a simulação virtual idêntica ao real

- Os mesmos pacotes de controle dos sistemas reais ROS são utilizados no simulador

- A configuração é realizada pela instalação de pacotes de simulação

Gazebo - Turtlebot 1&2

1. instalação

```
$ sudo apt-get update
$ sudo apt-get install ros-kinetic-turtlebot-simulator
```

2. execução

opcional

Turtlebot - versao 1

```
$ export TURTLEBOT_BASE=create
$ export TURTLEBOT_STACKS=circles
$ export TURTLEBOT_3D_SENSOR=kinect
```

Turtlebot - versao 2

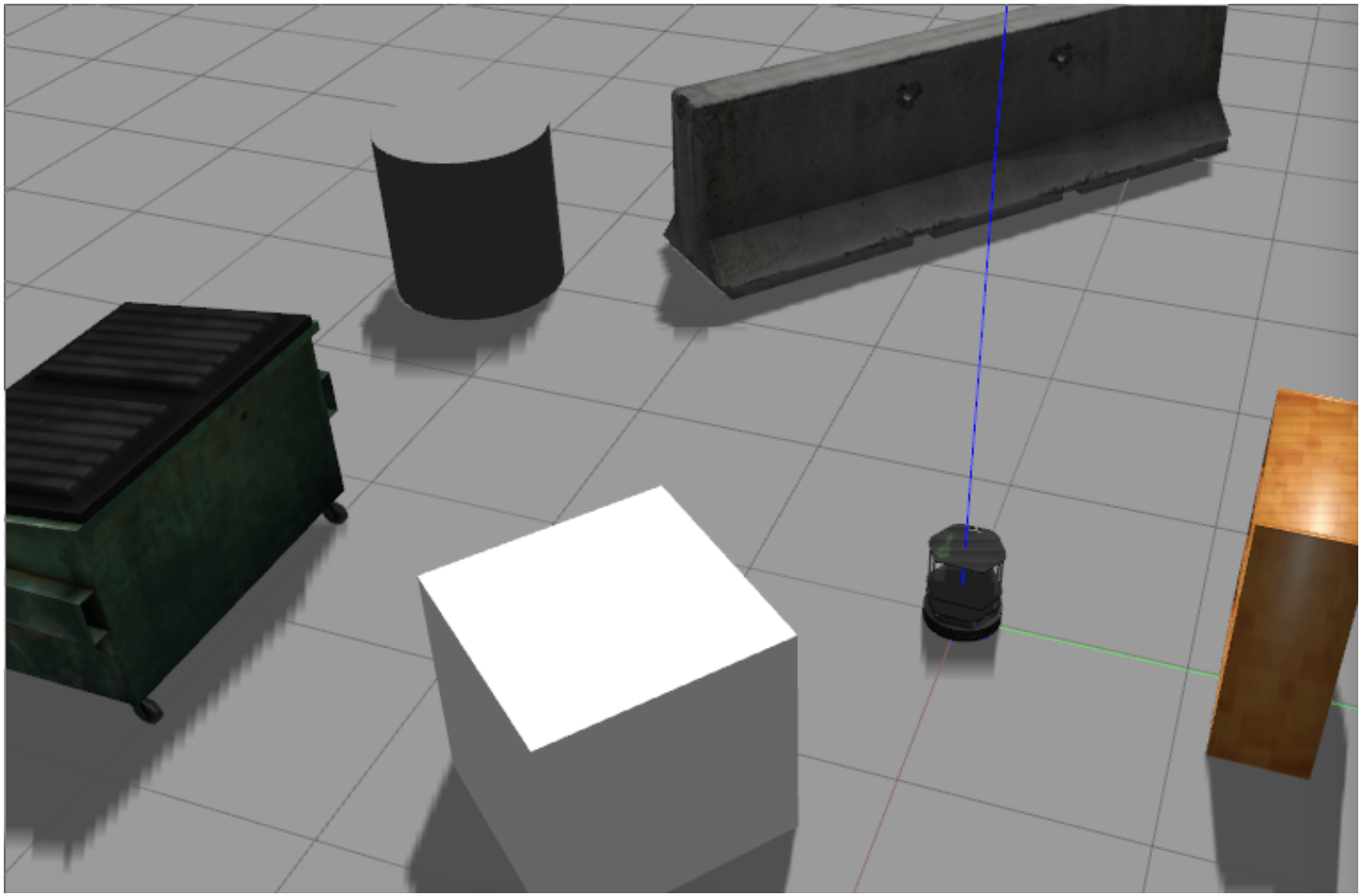
```
$ export TURTLEBOT_BASE=kobuki
$ export TURTLEBOT_STACKS=hexagons
$ export TURTLEBOT_3D_SENSOR=asus_xtion_pro
```

```
export TURTLEBOT_GAZEBO_WORLD_FILE=/opt/ros/kinetic/share/turtlebot_gazebo/worlds/playground.world
```

- obrigatório

```
$ roslaunch turtlebot_gazebo turtlebot_world.launch
```

Gazebo - Turtlebot 2



Gazebo - Turtlebot 3

1. instalação

```
$ sudo apt-get update
$ sudo apt-get install ros-kinetic-turtlebot3
$ cd ~/catkin_ws/src/
$ git clone https://github.com/ROBOTIS-
GIT/turtlebot3_simulations.git
$ cd ~/catkin_ws && catkin_make
```

2. execução

Selecionar uma das versões

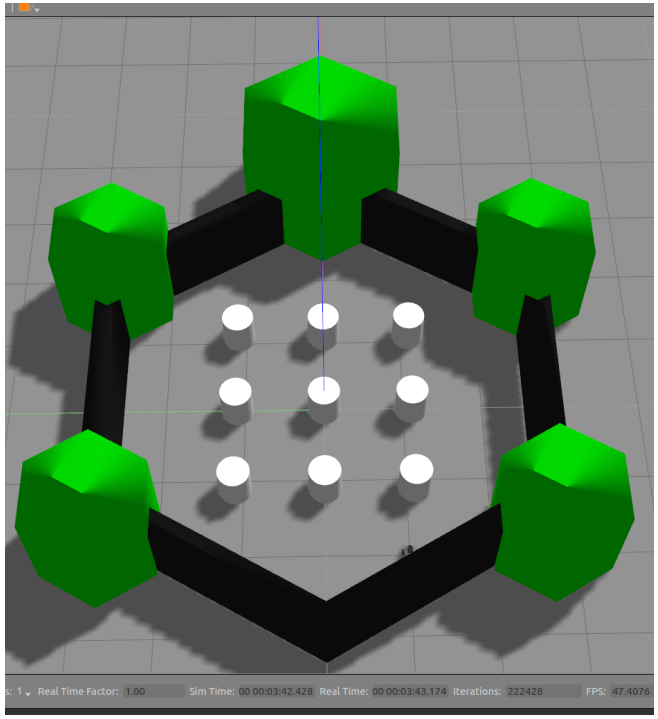
```
$ export TURTLEBOT3_MODEL=burger
$ export TURTLEBOT3_MODEL=waffle
```

```
$ roslaunch turtlebot3_gazebo turtlebot3_house.launch
```

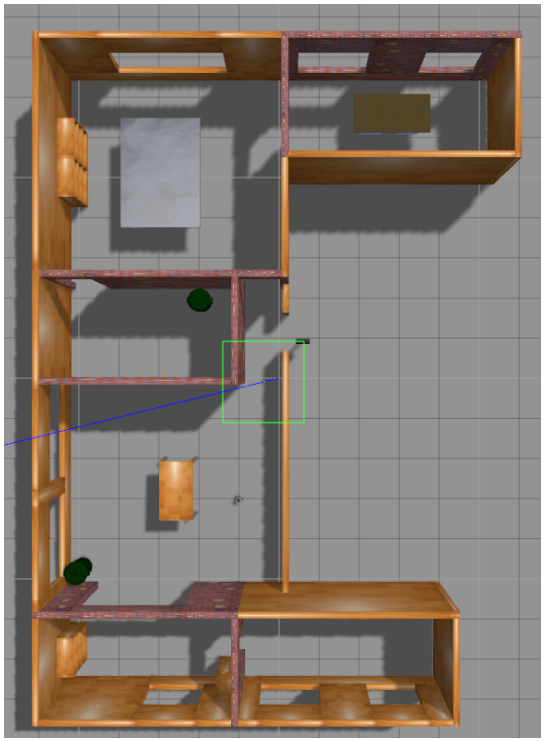
ou

```
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

Gazebo - Turtlebot 3



world



house

Gazebo - ARDrone

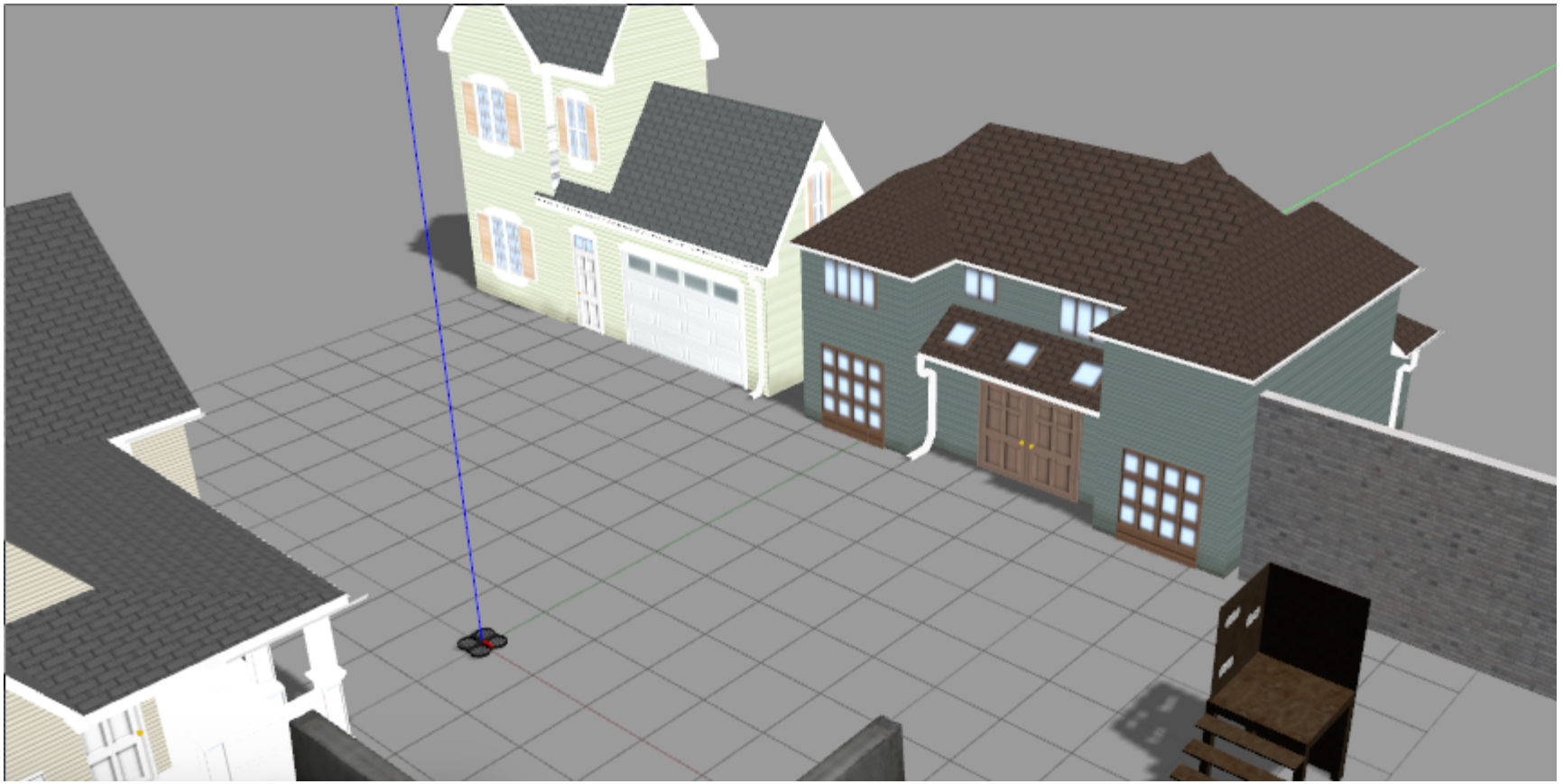
1. instalação

```
$ sudo apt-get update
$ sudo apt-get install ros-kinetic-ardrone-autonomy
$ cd ~/catkin_ws/src/
$ git clone https://github.com/RoboticaAI/tum_simulator-
1.git
$ cd ~/catkin_ws && catkin_make
```

2. execução

```
$ roslaunch cvg_sim_gazebo ardrone_testworld.launch
```

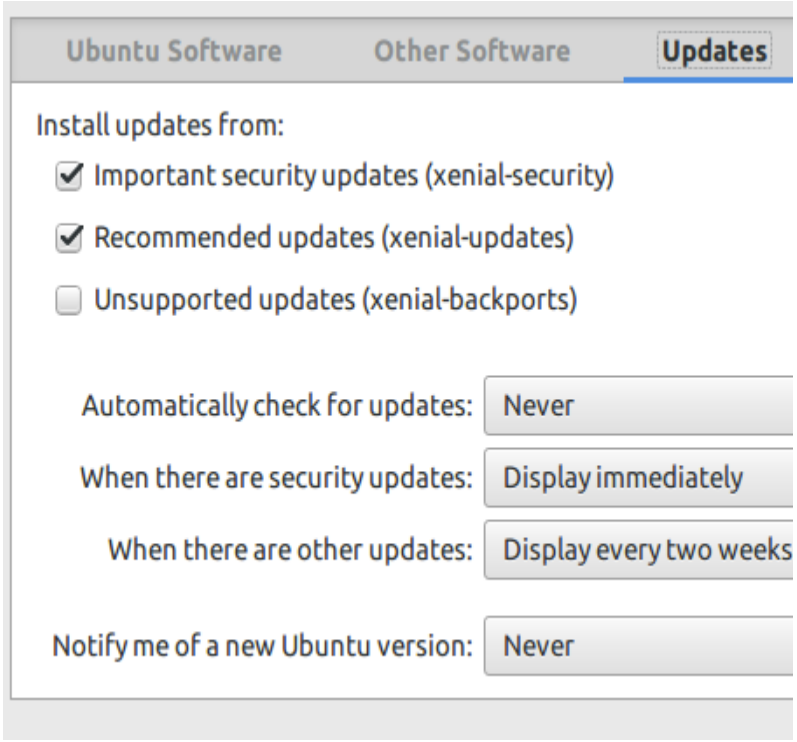
Gazebo - ARDrone



Gazebo na maquina virtual

E necessarop habilitar os repositorios de atualizacao

start/system tools/software updates/settings/updates



Stage

•E um simulador bastante simples, mas desenvolvido com o objetivo de ser rapido

•Contem os principais sensores e pode ser utilizado em computadores simples (como a VM)

•O simulador e instalado no ROS pelo comando

sudo apt-get install ros-kinetic-stage-ros

•Na pagina da disciplina existe um pacote simples de demonstracao do Stage **(ja no catkin_ws/src da VM)**

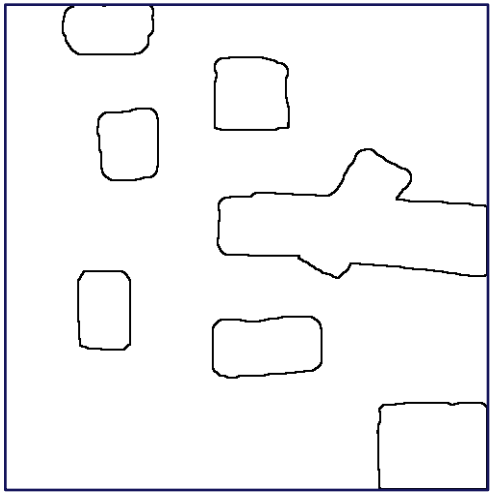
Stage

• Sua configuração baseia-se em arquivos de texto que especificam os parâmetros da simulação (robô, sensores, mapa)

http://wiki.ros.org/stage/Tutorials

http://rtv.github.io/Stage/modules.html

•O mapa e criado através de um arquivo de imagem PNG



Stage

- Para executar o simulador utiliza-se o comando

rosrun stage_ros stageros (arquivo de configuração).world

- O pacote exemplo já contem um arquivo denominado de utfpr.word

rosrun stage_ros stageros utfpr.world

Stage