# UTBot@Home 2015 Team Description Paper – Uso do robô Pioneer 3-AT com sensor Kinect para tarefas domésticas.

Rodrigo Longhi Guimarães , André Schneider de Oliveira, João Alberto Fabro, Thiago Becker, Hudo Cim Assenço, Felipe Ukan Pereira, André Lucas Zanellato, Vinícius Amilgar Brenner, Geovana Franco e Victor Jordão

UTFPR – Universidade Tecnológica Federal do Paraná, Brazil

LASER – Laboratório Avançado de Sistemas Embarcados e Robótica

rolongui@yahoo.com.br, andre@dainf.ct.utfpr.edu.br, fabro@utfpr.edu.br, beckerthiago@gmail.com, hudo@gmail.com, felipe.ukan@gmail.com, andre.zanellato@gmail.com, geovana.franco01@gmail.com, vj.jordao@gmail.com

***Abstract.*** This TDP describes the hardware and software implemented to allow a Pioneeer 3-AT robot to execute domestic tasks related to the Robocup@Home initiative. The robot uses the information provided by the sensor to execute navigation tasks such as environment mapping and auto-localization, and the "follow-me" task. The sensor it uses for so is the Microsoft Kinect. It also executes some other Robocup@Home tasks, like the voice related ones. The robot uses ROS(Robot Operating System) and a lot of its provided packages to accomplish its tasks.

## 1 Introdução

A Robocup@Home é uma competição que visa fomentar o desenvolvimento de robôs que venham a ajudar no ambiente doméstico, sendo a maior competição para robôs de serviço do planeta[1]. A fim de executar as tarefas domésticas, são desenvolvidos, dentre muitos outros campos da robótica, a interação homem-robô, a navegação e mapeamento de ambientes dinâmicos e a visão computacional.

Para controlar o robô pioneer 3-AT, utilizamos o ROS (Robot Operating System), um conjunto de bibliotecas de sofware e ferramentas que ajudam a desenvolver aplicações para robótica[3]. O sistema de controle do robô é responsável por adquirir e interpretar as leituras dos diversos sensores do robô e, a partir desses dados e do cenário que se está inserido, tomar decisões para cumprir suas tarefas.

Nesse documento, são apresentados os resultados parciais já obtidos e expectativas de projetos em desenvolvimento para o robô. A seção 2 apresenta uma descrição do hardware do robô, incluindo suas especificações e as características das melhorias em relação a base original. O software já desenvolvido, e o que ainda está em desenvolvimento, são descritos na seção 3. Conclusões e perspectivas futuras para o desenvolvimento do robô são apresentadas na 4.

## 2 Descrição do Hardware

### 2.1 Especificações do robô

O robô móvel utilizado para construir o UTBot@Home é do modelo Pioneer 3-AT[4]. Como pode ser visto na Figura 1, é um robô de 4 rodas de borracha, conectadas duas a duas por correias, que executa movimentos de rotação por derrapagem. Possui uma carcaça com 1.6mm de alumínio, pesa 12kg e mede 497mm de comprimento, 508mm de largura e 277mm de altura (Figura 2). Seus motores possibilitam uma velocidade

para frente ou para trás de até 0,7m/s e uma velocidade de rotação de até 140º/s. É alimentado por até três baterias ao mesmo tempo(12V, 7.2Ah cada), possibilitando de 2 a 4 horas de autonomia.

**Fig. 1.** Robô Pioneer P3AT, suporte para os sensores (nível intermediário) e notebook (nível superior)

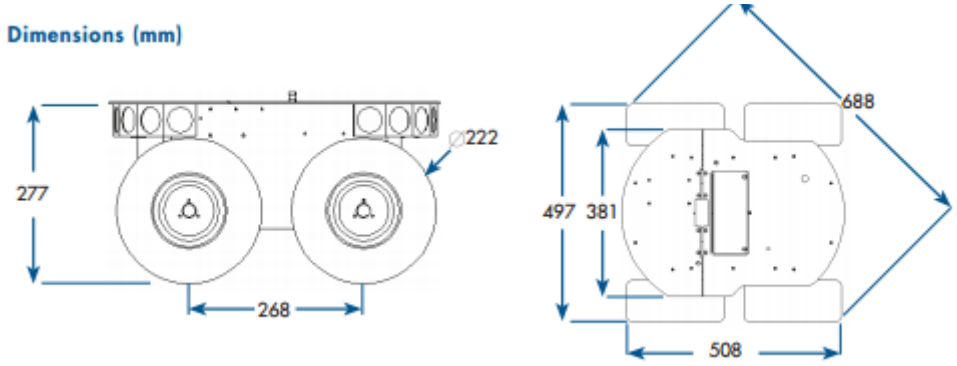


**Fig. 2.** Dimensões do robô

### 2.2 Suporte

Para suportar o notebook e os sensores do robô foi construído um suporte em alumínio (como pode ser visto na Figura 1) com dois níveis, com placas de alumínio de 30cm por 30cm, sendo que o primeiro nível está a 14,5cm do robô e o segundo a 30,5cm, fazendo com que o robô tenha uma altura total de aproximadamente 59cm. As placas são sustentadas por barras roscadas de 7mm de diâmetro, presas com roscas auto-travantes e com barras de alumínio em volta.

### 2.3 Sensores

A versão da base móvel que o time possui veio com poucos sensores de fábrica, sendo que a maioria dos sensores presentes no robô atualmente foram instalados no próprio laboratório. Os únicos sensores já presentes no robô são encoders nas rodas, que permitem inferir a velocidade e posição do robô em relação a referência inicial.

Após as modificações do time, o robô conta com um acelerômetro, GPS e giroscópio, além de um sensor de profundidade: o Kinect. Esse sensor é capaz de criar uma imagem em profundidade, posicionar suas câmeras verticalmente, rastrear a silhueta de uma pessoa, identificar comandos de voz, entre outras funções [5].

Para estabelecer a comunição entre o sensor Kinect e o ROS foi usado o projeto open source *Openni Kinect*[6] que já acompanha os drivers do Kinect e torna a comunicação entre o sensor e o ROS bastante simples. Os dados gerados pelo Kinect que são publicados em tópicos no ROS são peça chave para o funcionamento da maioria dos algoritmos usados nesse projeto, visto que é a partir da interpretação dessas imagens que o robô toma grande parte de suas decisões.

# 3 Software

## 3.1 ROS e comunicação

Foi instalado em um computador acoplado ao robô a versão Hydro do ROS, sobre o sistema operacional Ubuntu 12.04 LTS. O ROS é uma estrutura (*framework*), que traz uma coleção de ferramentas, bibliotecas e convenções que visam simplificar a tarefa de criar aplicações complexas e robustas para robôs[3]. Nesse sistema, é necessário executar um núcleo, o roscore, que funciona como um servidor, que gerencia tópicos e nós. Tópicos são barramentos de dados nomeados nos quais nós trocam mensagens, sendo que em um mesmo tópico vários nós podem estar inscritos (recebendo dados) ou publicando (enviando dados). Nós são processos que computam, aplicações propriamente ditas, e se baseiam nos valores de alguns tópicos específicos para gerar novos dados e publicar no mesmo ou em outro tópico.

O robô já possui a cinemática calculada e os encoders integrados, sendo que por um cabo serial-usb ele se comunica com o notebook e publica e recebe dados por meio do ROS. Esse notebook está configurado para ser acessado por SSH a partir de outros computadores, podendo assim ser controlado a distância por uma rede sem fio. Além disso, o acesso SSH juntamente com outro computador que contenha o ROS e a configuração do ROS para trabalhar com computadores múltiplos[7] possibilita que o processamento pesado seja feito em computadores externos ao robô.

## 3.2 Navegação e SLAM (*Simultaneous Localization and Mapping*, Localização e Mapeamento simultâneos)

Usando os dados dos encoders e corrigindo a posição atual usando de pacotes de localização autônoma baseados na leitura do Kinect, conseguimos uma boa posição relativa. O robô observa o ambiente a partir do Kinect, a fim de criar um modelo (mapa) do ambiente. São então realizadas medições através dessas observações para identificar onde o robô está e são definidos *landmarks* (pontos de referência) que ajudam o robô a se localizar dentro do mapa. Qualquer erro no mapa irá propagar erros na localização, mas, felizmente, os erros de estimativas de localização se correlacionam. Esses erros de correlação são corrigidos por meio do sistema EKF (Extended Kalman Filter)[8], que consegue armazenar a correlação entre todos os erros de estimativa e torna a localização dependente apenas da incerteza inicial de localização do robô.

Já foi implementado no robô o uso do SLAM (Localização e Mapeamento Simultâneos)[9], uma técnica utilizada por robôs e veículos autônomos para construir um mapa de um ambiente ao mesmo tempo que se localiza. Para realizar tal feito, o SLAM tem uma série de tarefas: extração de pontos de referência, associação de dados, estimação de estado, atualização de estado e atualização de ponto de referência. Para tal, dentre outras funções, ele usa do EKF. Essa implementação se deu usando funções já prontas do ROS, das quais os parâmetros foram ajustados para que a navegação tivesse boa performance em nossa base móvel.

## 3.3 Seguidor de Pessoas com Kinect

Adaptamos o algoritmo usado no robô turtlebot e disponível no pacote turtlebot_apps do ROS para funcionar como seguidor de pessoas e cumprir a tarefa da Robocup@Home. Estuda-se, até o prazo, integrar o nó openni_tracker com a navegação autônoma para se obter um resultado melhor do que o seguido de pessoas do turtlebot_apps[10].

## 3.4 Reconhecimento e síntese de voz

Usamos o pacote pocketsphinx[11] para reconhecimento de voz. Esse pacote já contém uma adaptação para o ROS, sendo que é necessário apenas configurar um dicionário e alguns parâmetros para fazê-lo funcionar. O funcionamento competente do pacote é um pouco mais difícil de ser obtido, sendo que depende tanto de uma boa escolha dos parâmetros e do dicionário do pacote quanto do sistema de captação de áudio presente no robô. Apesar de termos trabalhado com os parâmetros e conseguido alguns resultados interessantes com nossa configuração de áudio, ela é bastante limitada e ainda apresenta vários erros, principalmente em ambientes com muito ruído.

Quanto a síntese de voz, outro pacote do ROS, o sound_play[12], é usado. Esse pacote recebe uma sequência de caracteres e sintetiza a fala em inglês. O pacote não requer muita configuração e funciona dentro do esperado para a maioria das palavras que se passam para ele. Depois de estudar como se comportava para alguns números e símbolos, conseguimos um certo domínio desse pacote capaz de interagir com o usuário de forma adequada.

### 3.5 Garra

Uma garra está em fase final de desenvolvimento e espera-se que, até a competição, várias das funcionalidades requeridas estejam implementadas.

### 3.6 Mapeamento e visualização 3D

A categoria @Home da competição também contempla algumas provas extras, sendo que vai ser apresentado um projeto de mapeamento e visualização 3D, baseado no pacote octomap[13] do ROS. Usando de dados do Kinect e da odometria esse pacote consegue criar uma visualização 3D baseada na ocupação ou não de voxels (subespaços), conforme pode ser visto na figura 3.

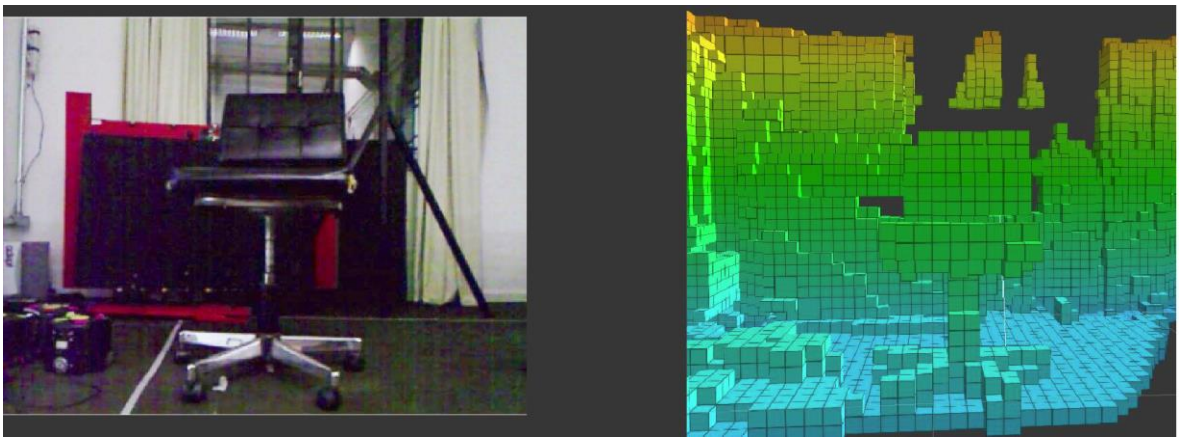

**Fig. 3.** Visualização 3D baseada no pacote octomap – visão do Kinect a esquerda e visualiação 3D a direita.

## 4 Conclusão

Com os trabalhos correntes e futuros é possível realizar várias das provas da categoria @Home com êxito, sendo que se espera, até a data, conseguir ainda mais resultados.

## 5 Agradecimentos

Os autores agradecem a Universidade Tecnológica Federal do Paraná e a Fundação Parque Tecnológico de Itaipu pelo apoio dado a este projeto.

## 6 Referências

[1] Site oficial Robocup@Home. Disponível em: http://www.robocupathome.org/. Acesso: 10 de julho, 2015.

[2] D. HOLZ, J.R. del SOLAR, K. SUGIURA, S. WACHSMUTH, "On RoboCup@Home – past, present and future of a scientific competition for service robots". Disponível em: http://goo.gl/fi7V51. Acesso: 11 de julho, 2015.

[3] About ROS – Página oficial. Disponível em: http://www.ros.org/about-ros/. Acesso em: 11 de julho, 2015.

[4] ADEPT, Pioneer 3-AT Specifications . Disponível em: http://www.mobilerobots.com/Libraries/Downloads/Pioneer3AT-P3AT-RevA.sflb.ashx. Acesso: 11 de julho, 2015.

[5] MICROSOFT, Develop Network: Kinect Sensor. Disponível em: http://msdn.microsoft.com/en-us/library/hh438998.aspx. Acesso: 12 de julho, 2015.

[6] V. RABAUD, T. FOOTE, openni_kinect ROS wiki page. Disponível em: http://wiki.ros.org/openni_kinect. Acesso: 12 de julho, 2015.

[7] ROS Multiple Machines wiki page. Disponível em: http://wiki.ros.org/ROS/Tutorials/MultipleMachines. Acesso em: 12 de julho, 2015.

[8] M. I. RIBEIRO, "Kalman and Extended Kalman Filters: Concept, Derivation and Properties". Disponível em: http://users.isr.ist.utl.pt/~mir/pub/kalman.pdf. Acesso: 12 de julho, 2015.

[9] S. RIISGAARD, M. R. BLAS, "SLAM for Dummies". Disponível em: http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam_blas_repo.pdf. Acesso em: 13 de julho, 2015.

[10] D. STONIER, T. FOOTE, M. WISE, turtlebot_apps ROS wiki page. Disponível em: http://wiki.ros.org/turtlebot_apps. Acesso em: 14 de Julho, 2014.

[11] M. FERGUSON, pocketsphinx ROS wiki page. Disponível em: http://wiki.ros.org/pocketsphinx. Acesso em: 14 de Julho, 2014.

[12] A. HENDRIX, B. GASSEND, sound_play ROS wiki page. Disponível em: http://wiki.ros.org/sound_play. Acesso em: 14 de Julho, 2014.

[13] A. HORNUNG, K. M. WURM, A. HORNUNG, octomap ROS wiki page. Disponível em: http://wiki.ros.org/octomap. Acesso em: 14 de Julho, 2014.