Robô Recepcionista Pioneer P5-DX

Pioneer 5
Pedro Isidro - 67220, Diogo Silva - 75136, João Pedrosa - 79833

***Abstract*—Abstract (summary of the work)**

***Index Terms*—Pioneer P5-DX, ROS, autónomo.**

I. Introdução

Os requisitos para do nosso projecto exigiam ter um robô Pioneer P3-DX capaz de executar tarefas modeladas por uma máquina de estados finitos. Essas tarefas incluem o robô ser capaz de mapear, auto-localiza-se e navegar o seu ambiente e deslocar-se para diferentes localizações indicadas através de interação com utilizadores.

A nossa motivação foi ter um robô recepcionista capaz de receber visitantes num edifício, e.g. um bloco de escritórios. O robô seria capaz de mapear o edifício *a priori* para futura utilização, de receber um conjunto de coordenadas já conhecidas (e.g. escritórios de certas pessoas) associadas a palavras-chave, de interagir com utilizadores através de síntese e reconhecimento de voz e, por fim, de guiar os mesmos aos seus destinos.

Durante a implementação do projecto simplificámos algumas destas funcionalidades, nomeadamente o mapeamento e o reconhecimento de voz. O mapeamento foi executado apenas no quinto piso da Torre Norte do IST. O reconhecimento de voz foi restringido a um pequenos dicionário para reduzir a complexidade e a necessidade de adaptar modelos acústicos e treinamento.

December 6, 2013

II. Algoritmos e Implementação

A projecto foi desenvolvido inteiramente sob a plataforma ROS (*Robot Operating System*) com uso de soluções *off-the-shelf*, preferencialmente *open-source*.

A. Mapeamento

O mapa utilizado é uma grelha de ocupação, onde cada célula contém um valor binário indicando se está ocupada ou se é um espaço livre:

$$m = \sum_i m_i \tag{1}$$

Os dados recebidos de odometria vão ser utilizados para estimar a posição x. Utilizando a posição x e os dados do laser z, é possível integrar estes dados no tempo para obter uma estimativa de cada célula do mapa, i.e. para todas as células do mapa é necessário resolver um problema binárioa resolver por um filtro binário de Bayes. Sabendo todas as leituras e o percurso do robô, o melhor mapa é:

$$\hat{m}_i = arg_{m_i} max\{p(m_i|x_{1:t}, z_{1:t}\} \tag{2}$$

onde $p(m_i)$ é a probabilidade de a célula m_i estar ocupada. Para evitar instabilidades quando as probabilidades estão próximas de zero, usa-se a representação de logaritmica de possibilidades (*log odds*):

$$l_{t,i} = log\frac{p(m_i|x_{1:t}, z_{1:t})}{1 - p(m_i|x_{1:t}, z_{1:t})} \tag{3}$$

com condição inicial

$$l_{0,i} = log\frac{p(m_i)}{1 - p(m_i)} \tag{4}$$

A probabilidade pode ser extraída através de

$$p(m_i|x_{1:t}, z_{1:t}) = 1 - \frac{1}{1 + exp(l_{i,t})} \tag{5}$$

O algoritmo *occupancy_grid_mapping*, representado na Figura 1 percorre todas as células m_i da grelha e muda as que estão dentro do cone do sensor laser na amostra z_t.

Algorithm occupancy_grid_mapping($\{l_{t-1,i}\}, x_t, z_t$)**:**
  for all cells $\mathbf{m}_i$ do
    if $\mathbf{m}_i$ in perceptual field of z_t then
      $l_{t,i} = l_{t-1,i} + \mathbf{inverse\_sensor\_model}(\mathbf{m}_i, x_t, z_t) - l_0$
    else
      $l_{t,i} = l_{t-1,i}$
    endif
  endfor
  return $\{l_{t,i}\}$

Fig. 1. Algoritmo da grelha de ocupação para mapeamento.

A função *inverse_sensor_model* implementa $p(m_i|x_t, z_t)$ na representação logaritmica de possibilidades.

O pacote que usámos para o mapeamento foi o *gmapping*. O *gmapping* estimou a posição através dos dados de odometria recebidos do *RosAria*. O laser utilizado foi o Hokuyo URG-04LX-UG01 de curto alcance e utilizámos o pacote *hojuyo_node* para a leitura de dados.

B. Localização

C. Navegação

For doing the Navigation the used commands gived by the user

Odometry:

The pose have tree parameters: x, y , θ

Algorithm:

We use Rosaria to obtain odometry information for doing the localization in the map, the odometry information obtain in topic pose published by Rosaria. Another thing we use Rosaria is for reading the sonar, give by the topic sonar.

D. Execução do Plano Coordenado

Para conseguir coordenar as acções do robô de maneira a executar a tarefa proposta, é necessário ter uma representação abstracta dessa mesma tarefa. O robô recepcionista é um sistema sequencial, ou seja, executa uma acção de cada vez – ou mover-se para uma posição-alvo ou interagir com o utilizador para adquirir esse objectivo. Para tais sistemas, o fluxo de controlo é usualmente especificado recorrendo a uma Máquina de Estados Finitos (MEF). Uma MEF é então implementada no nó *p3dx_smach*, usando o pacote *smach*. A tarefa do robô é dividida em três estados de execução: INITIAL, TO_GOAL e GET_GOAL. A Figura 2, gerada pelo nó *smach_viewer*, ilustra a forma como o controlo flui entre os estados. De seguida explica-se o funcionamento de cada estado.

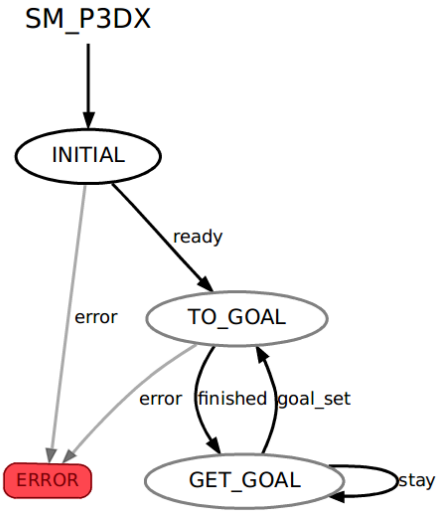

Fig. 2. Fluxo de execução da máquina de estados.

1) INITIAL: Estado inicial. O nó verifica se os ficheiros relativos ao mapa (*.pmg* e *.yaml*) estão presentes e lê o ficheiro de texto com as especificações dos diferentes locais de interesse (*setpoints*), guardando-os num dicionário. Caso ocorra alguma erro neste processo , retorna-se *error* e a execução é interrompida. Caso contrário, restorna-se *ready* e o controlo é passado ao estado *TO_GOAL*, com instruções para levar o robô até à base. Neste projecto, a posição base é em frente aos elevadores. Numa aplicação geral, esta seria, por exemplo, a entrada de um edifício.

2) TO_GOAL: Este estado recebe como argumento do estado anterior o identificador da posiçao-alvo. De seguida procura esse identificador no dicionário de *setpoints* para obter as coordenadas e orientação correspondentes. Por último, uma ordem é enviada para o nó de navegação *move_base*, usando um cliente de acção criado com o pacote *actionlib*. O servidor de acção encontra-se já implementado na *stack* de navegação, pelo nó *move_base_msgs*. O controlo é mantido até que o objectivo seja alcançado ou cancelado pelo utilizador, sendo então passado para o estado *GET_GOAL*, com vista a determinar a próxima directiva. São também passados argumentos que indicam qual foi o objectivo perseguido e se este foi alcançado ou cancelado. Mais uma vez, caso ocorra algum erro no processo, como o robô se perca ou o caminho se encontre bloqueado, este estado retorna *error* e a execução é interrompida.

3) GET_GOAL: À excepção do primeiro objectivo, que é invariavelmente a base, as posições alvo têm de ser determinadas pro interacção com o utilizador. *GET_GOAL* encarrega-se dessa tarefa, recorrendo aos nós de reconhecimento de fala (*pocketsphinx*) e de síntese de voz (*say*). Caso o robô se encontre na base – o que pode ser determinado pelos argumentos fornecidos por *TO_GOAL* – o robô simplesmente aguarda que um utilizador inicie a interacção, voltando a executar o mesmo estado se nenhum objectivo for transmitido. Caso contrário, o próprio robô pede ao utilizador um novo objectivo, ordenando o retorno à base se este recusar ou o ignorar durante um certo período de tempo.

E. Interacção com o utilizador

A interacção com utilizadores foi feita com recurso a reconhecimento e síntese de voz. Para o primeiro, foram testados duas soluções com características muito distintas. Para o último, foi utilizado um síntetizador de som com capacidade de síntetizar voz a partir de um texto recebido.

1) Reconhecimento de voz: Para compreender como é que o reconhecimento de voz é realizado é necessário introduzir alguns conceitos importantes. A unidade básica da fala é entendida como um *fone*. Contudo, as características acústicas correspondentes a um fone variam conforme o contexto em que esse fone aparece, a pessoa, etc. Devido a estas variações utilizam-se subestados dentro de um fone para melhorar o reconhecimento. Usa-se, também, o contexto em que os fones aparecem, o que se irá traduzir num problema de procura do contexto que melhor se aproxima dos dados recebidos. Os fones constróem sílabas. Dependendo das condições de fala, a mesma sílaba correponde a diferenes fones. Palavras restringem considerávelmente a quantidade de fones que se têm de comparar e quanto menor for o dicionário mais rápido será o reconhecimento.

Uma das soluções testadas foi a livraria *Pocketsphinx* da plataforma Sphinx. Esta livraria, disponível num pacote do ROS com o mesmo nome, está optimizada para portabilidade que é exactamente o que pretendemos na implementação num robô. A plataforma Sphinx utiliza algoritmos extensivamente utilizados em investigação baseados em *Hidden Markov Models* (HMM). Os principais componentes do processo de reconhecimento de voz esão representados na Figura 3.

O audio recebido pelo microfone é convertido em numa sequência de vectores acústicos $Y_{1:T} = y_1, ..., y_T$, num

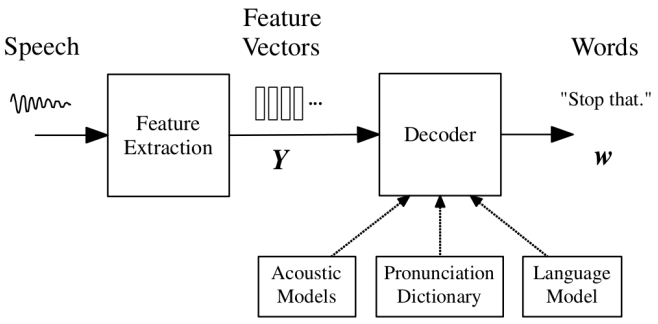

Fig. 3. Principais componentes do processo de reconhecimento de fala.

processo denominado de extração de *features*. Os números dos vectores são baseados nas propriedades acústicas da fração correspondente na sequência. Depois desta extração, o descodificador tenta encontrar a sequência de palavras $w_{1:L} = w_1, ..., w_L$ que é mais provável ter estado na origem de Y, ou seja:

$$\hat{w} = arg_w max\{P(w|Y)\} \tag{6}$$

No entanto, $P(w|Y)$ é difícil de modelar directamente, pelo que se utiliza a regra de Bayes para transformar (6) no equivalente:

$$\hat{w} = arg_w max\{p(Y|w)P(w)\} \tag{7}$$

A verossimilhança $p(Y|w)$ é determinada pelo *modelo acústico* e a probabilidade $P(w)$ é determinada pelo *modelo de linguagem*. Para qualquer palavra w, é criado um modelo acústico pela concatenação de modelos fonéticos para criar palavras como elas são definidas num dicionário de pronunciação. Os modelos fonéticos podem ser treinados especificamente para dicionários, ambientes ou contextos específicos. O modelo de lingagem é um modelo em que a probabilidade da palavra N é dependente apenas nos seus $N-1$ antecessores. Os parâmetros no modelo de línguagem são determinados através da contagem da ocorrência das palavras no $corpus$ do texto.

Assim, para o reconhecimento de voz é feito pelo nó *pocketsphinx*, recebendo como argumentos um dicionário de pronunciações e um modelo de linguagem. Para o nosso projecto modificámos o código original para também aceitar outros modelos acústicos que não o original. Desta forma, podemos carregar um modelo que melhor se adapte à nossa utilização.

Apesar de não o termos feito, o desempenho do reconhecimento de voz poderia ter sido melhorado se se tivesse treinado o modelo acústico ou adaptado ao dicionário utilizado.

2) Síntese de voz: A síntese de voz foi feita com recurso ao pacote *sound_play* da *stack audio_common*. O algoritmo trata qualquer som (ficheiro *WAV* ou *OGG* ou texto sintetisado) como algo que pode ser reproduzido ou parado. O estado da reprodução de um som pode ser mudado através da publicação para um tópico específico *robotsound*. O nó *soundplay_node* é o nó responsável pela reprodução do som. Os restantes nós do pacote servem para fornecer o som a ser reproduzido. O nó que nos dá acesso à síntese de voz é o *say* que recebe texto por argumento da linha de comandos ou de um texto. O código deste nó foi alterado por forma a receber texto por subscrição a um tópico. A síntese de voz é feita através do *Festival*.

F. Visualização

The project was developed on the Robot Operating System framework, which is open-source and contains a myriad of different off-the-shelf packages ready for use.

III. Resultados

A. Mapeamento

Quando mapeámos apenas dentro da sala o mapa resultante estava de acordo com a realidade. Contudo, quando passámos para o mapeamento dos corredores do quinto piso da Torre Norte do IST, o mapa resultante não correspondia à realidade, como se pode observar na Figura 4.

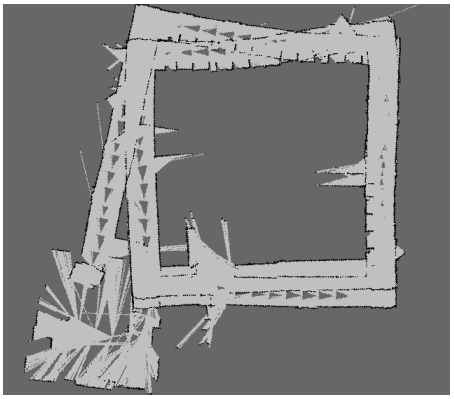

Fig. 4. Mapa resultante do mapeamento do quinto piso da Torre Norte.

B. Reconhecimento de voz

Tendo em conta a nossa utilização do robô como recepcionista, o dicionário de pronunciações foi limitado a um pequeno número de palavras. Foram introduzidas algumas palavras de comandos para o robô e as restantes estavam associadas a localizações. Os modelos acústico e fonético utilizados estão adaptados apenas para utilização com língua Inglesa, pelo que os comandos e nomes de localizações escolhidos foram, de igual forma, em língua Inglesa.

O microfone utilizado foi o imbutido num Os resultados que obtivemos foram bastante satisfatórios num ambiente com pouco ruido, mesmo com a utilização de um modelo acústico não adaptado. As palavras *hello*, *yes* e *no* foram quase sempre reconhecidas à primeira e raramente mal intrepretadas por outras palavras. Os nomes das localizações eram mais frequentemente confundidos com outras palavras, mas geralmente não com nomes de outras localizações, pelo que não afectava muito o desempenho do sistema.

C. Síntese de voz

Os nós da síntese de voz reproduziram com sucesso as frases enviadas. O discurso, apesar de não natural como o discurso humano, foi perceptível e compreensível. As frases síntetizadas foram *"hello"* quando o robô iniciava ou estava na base, *"where to go"* depois do utilizador expressar intenção em ser guiado pelo robô, *"repeat please"* quando o robô não percebia o que era dito ou não encontrava correspondente nas localizações guardadas, *"want to go anywhere else"* quando o robô chega a um destino que não a base e *"goal was canceled"* quando um objectivo é cancelado.

IV. Conclusão

O mapeamento foi algo que não correu como esperado. O mapeamento, por si, é um processo complicado pois estamos a mapear e a localizar simultâneamente. A localização é feita somente através de odometria.

Appendix A
Title of Appendix A

Appendix A text goes here. desired:

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.