

# Máquinas de Estados Finitas Estatísticas aplicada a Robótica

Eliton Luiz Scardin Perin

Universidade Federal de Mato Grosso do Sul

Faculdade de Computação

Bacharelado em Ciência da Computação

Trabalho de Conclusão de Curso

Campo Grande, MS, September 16, 2014

## Abstract

Using robots to estimate and identify locations regardless of where they are, being your source of information its sensors. The robot will use a probabilistic finite state machine to reach its conclusions.

## 1 Introdução

Este trabalho visa a implementação de máquinas de estados finitos estatísticas aplicada à robótica. Até o momento uma aplicação usando a máquina de estados finitos para robôs foi implementado, e a partir desses exemplos, com a mesma ideia de autômato determinístico finito associado a teoria de probabilidade objetivamos um robô obter sua localização aproximada.

## 2 Proposta

Utilizar máquinas de estados finitos junto com probabilidade e estatística para desenvolver aplicações em robôs. As máquinas de estados modelam o comportamento dos robôs, unindo a probabilidade e estatística realizam decisões para que estados prosseguir, tornando o sistema **mais inteligente** para fazer suas tarefas, com cálculos de probabilidade e estatística.

O trabalho a ser realizado pelo robô é **chagar** a um destino determinado pela faixa preta e também desviar de obstáculos que podem estar no seu caminho. A aplicação também irá tratar a busca pela faixa preta, pois inicialmente o robô não conhece nada sobre o ambiente em que está. No meio do caminho existem outras cores que significam mudança de rotas ou

## 2.1 Máquinas de Estado Finitos

Máquinas de estados finitos (FSM do inglês **finit-state machine**) ou ainda autômato de estados finitos, é um modelo matemático **para computação, que consegue descrever programas em computadores quanto sequência lógica de circuitos.** Máquinas de estados finitos aceitam padrões de linguagem gerados por um comportamento, característica dos robôs.

Uma FSM pode ser descrita por uma quintupla. Seja  $M$  uma FSM qualquer.

$$M = ( \Sigma, S, s_i, \delta, F )$$

- $\Sigma$  é o conjunto de símbolos de entrada.
- $S$  é o conjunto dos estados.
- $s_i$  é o estado inicial.
- $\delta$  é a função de transição entre os estados.
- $F$  é o conjunto dos estados finais ou de aceitação.

Toda máquina de estados finitos possui apenas um estado inicial ( $s_i$ ), contido no conjunto dos estados ( $S$ ), onde cada estado representa um comportamento do sistema. A máquina possui outro conjunto para determinar todas as transições ( $\delta$ ) possíveis entre os estados da máquina.

Para as transições ocorrerem é preciso ler um símbolo de entrada do conjunto de símbolos ( $\Sigma$ ). Baseado nesse valor e dependendo do estado em que o processamento se encontra, a máquina transita, realizando uma ação ou não. A cada passo de transição pode-se estar em apenas um estado. Terminado a leitura da entrada e suas respectivas transições, a máquina para. Se o processo parar em um estado final do conjunto de estados finais ( $F$ ) então a máquina aceitou os dados de entrada. **Na transição de um estado para outro é carregada as informações dos estados anteriores, desde o estado inicial.**

### 2.1.1 Máquinas de Moore e Mealy

As máquinas de Moore são autômatos de estados finitos onde as **saídas ou resultados** são determinadas pelo estado que atingem. Essa máquina é representada por uma sêxtupla:

$$M = ( \Sigma, \Phi, S, s_i, \delta, \gamma )$$

- $\Sigma$  é o conjunto de símbolos de entrada.
- $\Phi$  é o conjunto de símbolos de saída.
- $S$  é o conjunto dos estados.
- $s_i$  é o estado inicial.
- $\delta$  é a função de transição entre os estados.

- $\gamma$  é a função de saída de dados.

A função de transição entre os estados da máquina Moore é realizado através da leitura de um símbolo do conjunto de entrada e dependendo do estado que está ela transita. A função de saída expressa um símbolo do conjunto de símbolos de saída dependendo do estado em que se encontra.

As máquinas de Mealy são autômatos de estados finitos onde as saídas **ou resultados** são determinadas tanto pelo estado que atingem quanto pelo símbolo corrente do conjunto de símbolos de entrada.

A máquina de Mealy pode ser representada pela mesma notação de uma máquina de Moore, porém sua função de saída de dados muda. A função de saída expressa um símbolo do conjunto de símbolos de saída dependendo do estado que se encontra e do símbolo corrente lido do conjunto de símbolos de entrada.

### 2.1.2 Máquinas de Estados Finitos aplicada à Robótica

O modelo de máquina estado finito especifica um programa para robôs, já que dependem de comandos gerados pelos seus sensores servindo como entrada para a máquina.

Numa máquina de estados finitas para robôs seguimos o mesmo formalismo. Agora as entradas são dadas pelos sensores e as ações são executadas pelos atuadores do robô. Dentre os sensores uso de distância, luz e cor. **Os atuadores serão 3 motores.**

## 2.2 Probabilidade e Estatística

Probabilidade é uma área da matemática que estuda a previsão de respostas num determinado fenômeno ou experimento. Ao realizar um experimento, qualquer resultado possível faz parte de um conjunto, denominado espaço amostral e simbolizado por  $U$ . Um resultado ou subconjunto desse espaço amostral chamamos de evento.

A probabilidade e estatística **constroem** as regras que calculam a escolha do próximo estado, dado o estado em que se encontra. **Utilizando os conceitos de probabilidade condicional, variáveis aleatórias, marginalização, Teorema de Bayes, distribuição conjunta e condicional[2].**

### 2.2.1 Probabilidade

A definição de probabilidade está na chance de ocorrer um evento ou eventos dentro do espaço amostral:  $P(n) = \frac{n}{c}$ , onde  $n$  é o elemento ou quantidade de elementos do evento e  $c$  é a quantidade de elementos dentro conjunto do espaço amostral. Desta definição temos:

$P(U) = 1$ , onde  $U$  é o conjunto do espaço amostral. A chance de ocorrer algum elemento desse evento que é o próprio espaço amostral é de 1 (ou 100 %).

$P(\{\}) = 0$ , onde  $\{\}$  representa um conjunto vazio.

### 2.2.2 Variáveis Aleatórias

É uma função que definimos sobre os eventos que podem ocorrer dentro do espaço amostral.

### 2.2.3 Probabilidade Condicional

A probabilidade condicional é calculada quando sabemos que uma parte do evento aconteceu ou sabe que vai acontecer. Então, com um subconjunto do conjunto total para nosso novo evento, temos:

$$P(E_1|E_2) = \frac{P(E_1 \cap E_2)}{P(E_2)}.$$

### 2.2.4 Distribuição Conjunta

A distribuição conjunta de um conjunto de variáveis aleatórias é uma função a partir do produto dos elementos do espaço amostral com valores de probabilidade que somam 1 sobre todo o espaço amostral[2].

### 2.2.5 Distribuição Condicional

Quando temos um conjunto de variáveis aleatórias e também sabemos que ocorreu um evento envolvendo uma dessas variáveis aleatórias, nós poderíamos usar a probabilidade condicional sobre essa variável. Formulando:

$$P(X = x|Y = y) = \frac{P(X=x,Y=y)}{P(Y=y)} \text{ para } P(Y = y) > 0, \text{ ou ainda,}$$
$$P(X = x|Y = y) = \frac{P(X=x,Y=y)}{P(X=x)} \text{ para } P(X = x) > 0$$

## 2.3 Máquinas de Estados Finitos Probabilísticas

Em uma Máquina de Estados Probabilística, **tenta-se** modelar o tempo como uma discreta sequência de passos, e podemos pensar no estado, entrada para o autômato e a observação em cada passo. Então temos variáveis aleatórias  $S_0, S_1, S_2, \dots$  para modelar os estados em cada passo de tempo as variáveis aleatórias  $O_0, O_1, O_2, \dots$  para modelar a observação em cada passo de tempo, e a variáveis aleatórias  $I_0, I_1, I_2, \dots$  para modelar as entradas do autômato em cada passo de tempo[2].

## 2.4 Robô da LEGO Mindstorms NXT


Este robô faz parte de uma linha de **brinquedos** da Lego, voltado para educação tecnológica. Lançado em agosto de 2006 o Mindstorms NXT possui um processador **potente**, com software próprio e sensores de luz, de toque e de som, permitindo a criação, programação e montagem de robôs com noções de distância, capazes de reagir a movimentos, ruídos e cores.

### 2.4.1 Características do Robô

Este robô possui a seguinte configuração e dispositivos:

- Processador Atmel 32-bit ARM;
- Três portas de saída digital;
- Quatro portas de entrada (uma IEC 61158, tipo 4);
- Display tipo matriz;
- Alto-falante;
- Pilha no formato AAA;
- Bluetooth;
- Porta de Comunicação USB 2.0;
- Três servo-motores interativos (com encoder acoplado);
- Quatro sensores: ultra-som, som, luz, cor e contato;
- Programa de computador intuitivo com uma versão LEGO do LabVIEW;
- Compatível com PCs e com MACs.

## 2.5 Sistemas Operacionais para robôs (ROS)

**Sistemas Operacionais ou Operativos**  softwares responsáveis por gerenciar os recursos do computador, tais como os dispositivos e programas pertencentes ao computador. Os SO's também oferecem uma interface para o usuário. Quando não existiam, cada programa era carregado para a memória do computador manualmente. Na década de 60 surgiram os primeiros sistemas operacionais, com capacidade de executar uma sequência de programas sem que o programador tivesse que carregá-los para a memória.

Os sistemas operacionais para robôs são dedicados ao gerenciamento dos recursos disponíveis no robô, como atuadores, sensores, motores e unidades de processamento, e também a comunicação entre eles. Um dos sistemas abordados nesse projeto será o ROS (Robot Operating System), entretanto ele não é um sistema operacional para robô.

O ROS fornece bibliotecas e ferramentas para ajudar os desenvolvedores de software criar aplicações para robôs. Ele fornece abstração de hardware, dispositivo de drivers, bibliotecas, visualizadores, transmissão de mensagens, gerenciamento de pacotes, e muito mais. ROS está licenciada sob uma fonte aberta, a licença BSD( Berkeley Software Distribution ). Embora simplifique a programação o ROS não é um sistema de tempo real.

Um sistemas **de tempo real** são dedicados a obter respostas das suas tarefas em um tempo determinado e quando o sistema não respeita esse prazo, é considerado que houve **uma falha no sistema**. Esse tipo de sistema se adapta a realidade de um robô, pois esperamos que ele **reaja rapidamente** em várias situações.

### 2.5.1 Vantagens do ROS

Fornecer um conjunto de softwares que abstraem os drivers de controle do robô, oferecendo uma interface **simplificada**. Cada dispositivo ou mecanismo é tratado como um nó de uma rede peer-to-peer, para que realizem troca de mensagens. A comunidade que utiliza o ROS tem recursos funcionais como distribuições, repositórios e Wiki do ROS.

### 2.5.2 Desvantagens do ROS

Podem ser mais lentos que um software implementado pelo próprio programador, **programando até rotinas que um driver de controle do robô**. Sua forma de comunicar entre processos e dispositivos é através de um método de sistemas distribuídos chamado de publisher/subscriber onde o acoplamento de informação **é fraco**, ou seja, um processo ou dispositivo pode pegar ou fornecer uma informação desatualizada para os outros.

## 2.6 O BricksOs

O BricksOs é um sistema operacional e compilador de código aberto [1], onde o usuário possui maior controle sobre o robô, bem como um uso completo de sua memória. Existe um ambiente para desenvolvimento e execução de programas RCX, baseado na Linguagem C/C++ e mais algumas ferramentas para envio dos programas ao RCX. O BrickOS é um sistema operacional alternativo para o RCX da Lego Mindstorms. Foi desenvolvido originalmente para Linux, mas é possível programar em qualquer editor de texto. Porém, existem alguns programas específicos como o BricxCC (NQC, BrickOS e outros) [Bri 2008] e o NQCEdit (NQC) [NQC 2008] que permitem editar, testar o programa, e carregar o programa no robô via torre infravermelha (num ambiente integrado).

## 2.7 Tutorial de instalação do BrickOs no Windows

A instalação será feita através de um programa que simula um SO Linux em um terminal, chamado Cygwin. O Cygwin pode ser obtido nesse link: <http://www.cygwin.com/>. No momento de instalação selecione Install from Internet e no diretório c:\cygwin\.

Após a instalação execute o terminal do Cygwin.

No terminal aberto cria uma nova pasta:

```
$ mkdir /build
```

Faça o download dos seguintes pacotes, salvando em c:\cygwin\build:

- <ftp://ftp.gnu.org/pub/gnu/gcc/gcc-2.95.2.tar.gz>
- <ftp://ftp.gnu.org/pub/gnu/binutils/binutils-2.10.1.tar.gz>
- <http://legos.sourceforge.net/cygwin/download/legos-buildgcc.zip>, sendo que este você deve extrair o conteúdo.

Agora construa o compilador com o seguinte comando no terminal:

```
$ cd /build
```

```
$ ./buildgcc.sh
```

Em seguida faça outro download, deste link <http://sourceforge.net/projects/brickos> e coloque em `c:\cygwin\`. E logo digite:

```
$ cd /  
$ tar xvfz brickos-versão-corrente.tar.gz
```

Neste momentos vamos configurar o SO:

```
$ ln -s brickOS-0.2.6.10 brickOS  
$ cd /brickOS  
$ ./configure; make  
$ cd util  
$ make strip
```

Ligue o RCX. Selecione o IR na porta COM1.

```
$ ./firmdl3 ../boot/brickOS.srec  
$ ./dll ../demo/helloworld.lx
```

Pressione Run no RCX e veja se funcionou.

Mais informações podem ser encontradas no <http://brickos.sourceforge.net/docs/INSTALL-cygwin.html>.

## 2.8 O BricxCC

O BricxCC foi usado inicialmente para aprimorar o aprendizado da programação para robô. Ele é um software para programar os robôs da LEGO MINDSTORMS e sua família, incluindo a terceira geração EV3.

## 3 Resultados

Com a aplicação faz movimentos para buscar a linha ou faixa preta onde se deve caminhar. Ela também desvia de objetos que estejam no caminho, tanto sobre a faixa quanto fora dela. Para realizar os movimentos existem dois estados principais, um para mover o robô para frente e outro para girar o robô em torno do próprio eixo.

Inicialmente o robô procura uma faixa para seguir um destino determinado pela faixa preta. Para procurar a faixa trata o problema, fazendo o robô andar em espiral, com os movimentos de seguir em frente e girar 90°, sempre dobrando a distância que ele anda para frente. Se em qualquer momento encontrar uma faixa preta, passa para outro estado que só ira ajeitar o robô sobre a linha, com o movimento de andar para frente e girar em torno do próprio eixo em  $Z^\circ$ .

Segue um automato abaixo, que descreve o comportamento da implementação para o robô, sem o tratamento de desvio:

$a_1$  : andar para frente  
 $a_2$  : andar para frente  
 $a_3$  : andar para frente  
 $g_1$  : girar  $Y^\circ$  para direita  
 $g_1$  : girar  $Z^\circ$  para esquerda

## 4 Conclusões

Máquinas de estados finitos modelam problemas envolvendo robótica, pois conseguem gerenciar o comportamento deles. Entretanto gera mais processamento para o robô, dependendo das ações e quantidade dos estados que possui a máquina.

## Referências

- [1] ‘an open source embedded operating system and provides a ‘c’ and ‘c++’ programming environment for the lego mindstorms robotics kits’. <http://brickos.sourceforge.net>. Last Access in 30/04/2008, 2008.
- [2] Jacob White Harold Abelson Dennis Freeman Tomás Lozano-Pérez Kaelbling, Leslie and Isaac Chuang. 6.01sc introduction to electrical engineering and computer science i, spring 2011. (mit opencourseware: Massachusetts institute of technology). <http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-01sc-introduction-to-electrical-engineering-and-computer-science-i-spring-2011>, 2011. (Accessed 12 May, 2014). License: Creative Commons BY-NC-SA.