UNIVERSIDADE CÂNDIDO MENDES

CHRYSTIANO BARBOSA DE SOUZA ARAÚJO
LEANDRO MORAES VALE CRUZ
LUCAS CARVALHO
THIAGO RIBEIRO NUNES

DEFINIR O TÍTULO

Campos dos Goytacazes - RJ
Junho - 2009

JÔNATAS OLIVEIRA LOPES SOARES
MAYCON BARRETO LOPES
WALLACE GOMES DE SOUZA

MÓDULO DE MAPEAMENTO DO TOOLKIT HÓRUS

Monografia apresentada à Universidade Cândido Mendes como requisito obrigatório para a obtenção do grau de Bacharel em Cinências da Computação.

ORIENTADOR: Prof. D.Sc. Ítalo Matias

CO-ORIENTADOR: Prof. D.Sc. Dalessandro Soares

Campos dos Goytacazes-RJ
2009

JÔNATAS OLIVEIRA LOPES SOARES
MAYCON BARRETO LOPES
WALLACE GOMES DE SOUZA

MÓDULO DE MAPEAMENTO DO TOOLKIT HÓRUS

Monografia apresentada à Universidade Cândido Mendes como requisito obrigatório para a obtenção do grau de Bacharel em Cinências da Computação.

Aprovada em ____ de _____ de 2009.

BANCA EXAMINADORA

Prof. D.Sc. Ítalo Matias - Orientador
Doutor pela UFRJ

Prof. D.Sc. Dalessandro Soares
Doutor pela PUC-Rio

Prof. BLABLABLA
Univeridade de Londres

Agradecimentos

Resumo

Palavras-chave: MAPEAMENTO, LOCALIZAÇÃO, SIMULTANEOUS LOCALIZATION AND MAPPING, SLAM, TOOLKIT, AGENTE INTELIGENTE.

Abstract

Keywords: MAPPING, LOCALIZATION, SIMULTANEOUS LOCALIZATION AND MAPPING, SLAM, TOOLKIT, HÓRUS, INTELIGENT AGENT.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Existem alguns tipos de ambiente que são inóspitos ao homem. Nesses casos é comum utilizar um robô para explorar e atuar em tais locais. A movimentação desses robôs pode ser automática (agentes autônomos), semi-automática agentes semi-autônomos) ou manuais. Neste trabalho, serão apresentados os passos para a construção de um toolkit utilizado para o desenvolvimento de aplicações que envolvam agentes inteligentes, com foco em dois problemas centrais. O primeiro problema refere-se a movimentação autônoma de um agente inteligente em ambientes desconhecidos. O segundo problema refere-se á visão computacional, onde o agente deve ser capaz de extrair informações do ambiente através da utilização de câmeras virtuais ou reais.

Um Agente, por definição, é todo elemento ou entidade autônoma que pode perceber seu ambiente por algum meio cognitivo ou sensorial e de agir sobre esse ambiente por intermédio de atuadores. Pode-se citar como exemplos de agentes inteligentes, além de um robô autônomo ou semi-autônomo, personagens de um jogo, agentes de busca e recuperação de informação, entre outros.

Para que um agente autônomo seja capaz de atuar em um ambiente desconhecido é necessário anteriormente explorar esse local. Essa exploração pode ser feita através de um mapeamento desse ambiente. A forma como mapeia-se o ambiente internamente no sistema é determinante na sua precisão e performance. As diferentes abordagens para controle de agentes móveis autônomos interagem fortemente com a representação do ambiente. Uma proposta para mapeamento do ambiente, ainda não implementada no Horus, é construir um ambiente virtual 3D associado a um ambiente real no qual um robô real está explorando. Essa abordagem exige que o agente reconheça padrões no ambiente explorado e represente-os no ambiente virtual.

Durante a exploração do ambiente, o agente deverá ser capaz de estimar sua posição local para localizar-se globalmente e se recuperar de possíveis erros de localização. Um correto mapeamento do ambiente junto a aplicação correta das leis da cinemática podem resolver tal problema. Uma proposta para a localização de um agente no ambiente é a utilização do método Monte Carlo [5] ou do método SLAM (*Simultaneous Location and Mapping*) [6], [7]. O método selecionado para ser implementado no toolkit horus foi o SLAM.

Um agente explora um ambiente através de sensores. O sensoriamento provê ao robô as informações necessárias para a construção de uma representação do ambiente onde está inserido e para uma interação com os elementos contidos nesse. Sistemas com uma variedade de sensores tendem a obter resultados mais precisos. A fusão de dados de sensores, ou como é mais conhecida, fusão de sensores, é o processo de combinação de dados de múltiplos sensores para estimar ou predizer estados dos elementos da cena. Neste trabalho, foram utilizados lasers, câmeras e odômetro como sensores.

Para simular a visão de um agente inteligente, são utilizadas câmeras virtuais. Na abordagem desse trabalho, a visão é a principal forma de percepção

do ambiente. A visão possibilita reconhecer padrões e classifcar obstáculos. Existem diferentes tipos de obstáculos. Estes podem se classifcados como transponível (aquele que não interrompe a trajetória), intransponível (aquele que o exigirá recalcula a trajetória por outro caminho) e redutor (aquele que permite ao robô seguir pela trajetória, porém a uma velocidade mais lenta). Mesmo mediante a obstáculos transponíveis e redutores, pode ser conveniente recalcular o caminho devido ao aumento do custo do percurso. Uma proposta para o desvio de trajetória é o modelo baseado em Campos Potenciais proposto por [8]. A classifcação de um obstáculo ocorre mediante a algum método de reconhecimento de padrões, baseado em visão computacional.

Com isso, o toolkit Horus propõe uma coleção de classes e algoritmos voltados a resolução de problemas pertencentes as áreas de visão computacional e mapeamento automático de ambientes. Nessa monografia, será dado foco à parte de visão computacional do Horus. De forma a validar a implementação do toolkit e demonstrar a sua utilidade, foram desenvolvidas três aplicações distintas: Os simuladores Teseu e Ariadnes e o ANPR Django. Neste trabalho, serão apresentados a parte de visão computacional do simulador Ariadnes e a aplicação para reconhecimento automático de placas de automóveis ANPR Django.

Capítulo 2

Inteligência Computacional

2.1 Agentes Inteligentes

A Inteligência Computacional (IA) é uma área de estudo da ciência da computação que procura desenvolver sistemas computacionais capazes de ter ações e reações similares as capacidades humanas, tais como: pensar, criar, solucionar problemas entre outros. Um Agente, por definição, é todo elemento ou entidade autônoma que pode perceber seu ambiente por algum meio cognitivo ou sensorial e de agir sobre esse ambiente por intermédio de atuadores. Pode-se citar como exemplos de agentes inteligentes, além de um robô autônomo ou semi-autônomo, personagens de um jogo, agentes de busca e recuperação de informação e agentes de chats.

Existem diferentes definições para a arquitetura de um robô presentes na literatura. Este trabalho considera a definição abordada por Arkin [**??**] a qual considera que uma arquitetura de um robô está mais relacionada com os aspectos de software que os de hardware. Apesar de algumas diferenças, os modelos de arquitetura para robôs móveis descrevem um mecanismo para construção de um sistema de desenvolvimento e controle de agentes inteligentes,

apresentando, principalmente, quais os módulos presentes e como estes interagem.

De uma forma geral, os módulos de uma arquitetura de um agente inteligente se preocupam com aspectos como percepção, planejamento e atuação/execução. A percepção refere-se à compreensão do ambiente e dos elementos nele contido. Denomina-se *sequência de percepções*, a história completa de tudo que o agente já percebeu, ou seja, o conjunto de todas as percepções do agente até um dado momento; o planejamento à inteligência do robô; e a atuação, ou execução, é o modo como o robô procede no ambiente, ou seja, movimentos, captura de informações, etc.

Existem diversos modelos de arquitetura para robótica entre os quais ressaltamos os modelos de três camadas como: SSS [**??**], Atlantis [**??**], 3T [**??**]. Todas as arquiteturas se dividem semelhantemente da seguinte forma:

- camada reativa: orienta os sensores e atuadores, além de tomar decisões de baixo nível, como visão e movimento;
- camada deliberativa: responsável pela inteligência do robô, aspectos mais globais, que não são alterados a cada iteração;
- camada de execução: intermedia essas duas outras camadas.

As camadas reativa e deliberativa provêem processos denominados de comportamentos. Em termos matemáticos, o comportamento do agente á a função que mapeia qualquer percecpção ou sequência de percepções para uma ação específica[LivroWallace] essa função é conhecida como *função de agente*.

Com base nesses conceitos, definiu-se neste trabalho um agente como uma entidade composta de comportamentos, dispositivos e um programa

de agente. Os dispositivos do agente, utilizados no sistema em questão, são de sensoriamento (lasers e câmeras) e de movimentação (rodas). Os comportamentos do agente são: mapeamento, navegação e reconhecimento de objetos. O programa de agente é responsável pelo controle da execução de todos os comportamentos supracitados.

Os comportamentos são procedimentos implementados para representar ações e reações dos agentes. As ações são comportamentos ativos, ou seja, procedimentos que visam realizar um objetivo previamente definido. Por outro lado, reações são procedimentos realizados mediante a estimulos externos. Exemplos de ação e reação ocorrem no deslocamento de um agente de uma posição a outra. Para realizar o deslocamento é necessário traçar uma rota. Tal comportamento é definido como uma ação. Levando em consideração que durante o percurso, esse agente se deparou com algum obstáculo. Isso deve gerar um comportamento de replanejamento da rota para alcançar o objetivo inical sem colidir com o obstáculo. A esse replanejamento, denomina-se reação.

Comportamentos podem ser divididos em duas categorias. Os Comportamentos Primários: parar, reduzir velocidade, acelerar, desviar de obstáculos, virar, inverter direção, dirigir-se a meta, fotografar, disparar lasers e etc; e Comportamentos Inteligentes: mapear, reconhecer objeto, navegar e executar uma tarefa específica. Um Comportamento Inteligente executa um conjunto de comportamentos primários para atingir seu objetivo. Os comportamentos primários ocorrem na camada reativa, enquanto que, os inteligentes ocorrem na camada deliberativa.

Entende-se por *Programa de Agente* o programa que recebe as percepções do ambiente como entrada e as mapeia para uma determinada ação através da função de agente. Além da estrutura citada acima, o programa de agente

pode ser estruturado de outras maneiras. Um exemplo de estrutura é construir o programa de agente como um conjunto de sub-rotinas que serão executadas de forma assíncrona em relação ao ambiente. O programa permanece em *loop*, recebendo todas as percepções geradas pelo ambiente, e repassa cada percepção para uma sub-rotina que irá tratá-la.

Os programas de agente podem ser classificados em quatro categorias principais:

- Agentes reativos simples: esse tipo de programa de agente se basea apenas na percepção atual para executar as suas ações, onde a sequência de percepções até o momento é ignorada.

- Agentes reativos baseados em modelo: esse programa de agente armazena uma sequência de percepções e toma suas decisões levando em consideração as percepções dessa sequência.

- Agentes baseados em objetivos: esse programa de agente possui informações sobre o objetivo que deve alcançar. Logo, suas decisões são tomadas com base na combinação das percepções do ambiente e nas informações do objetivo.

- Agentes baseados na utilidade: esse tipo de programa de agente tem a preocupação, não só de alcançar o seu objetivo final, como também em determinar a melhor forma possível de alcançá-lo.

Em duas das aplicações desenvolvidas neste trabalho, os programas de agente utilizados se enquadram na categoria de agentes baseados na utilidade.

2.2 Reconhecimento de Padrões

Reconhecimento de padrões é um atividade que os humanos fazem a todo tempo e, normalmente, sem um esforço consciente. Seres humanos recebem informações através de vários sensores orgânicos, as quais são processadas instantaneamente pelo cérebro. Essa habilidade é ainda mais impressionante no que diz respeito a assertividade do processo de reconhecimento mesmo quando as informações não se encontram em condições ideais, como por exemplo, em situações onde as informações são vagas, imprecisas, ou até mesmo, incompletas. A área de Reconhecimento de Padrões é responsável por projetar algoritmos e abordagens que procuram aproximar as tarefas realizadas computacionalmente das habilidades humanas. Esse processo consiste em classificar e descrever objetos através de um conjunto de características ou propriedades. Um dos principais conceitos dentro de reconhecimento de padrões é o discriminante. Tal conceito consiste em medir uma distância de um determinado padrão para cada outro previamente conhecido. Logo, a classe de um determinado padrão será a mesma do seu vizinho mais próximo [Simp 92], [Simp 93], de menor distância ou o protótipo mais parecido [Torb 98]. Espera-se de um sistema de reconhecimento de padrões que este seja capaz de aprender de uma forma adaptativa e dinâmica. Em sistemas de reconhecimento de padrões automáticos, as etapas de aprendizagem e reconhecimento são combinados a fim de atingir um objetivo desejado [Cagn 93] [Valli 98]. Portanto, redes neurais artificiais é uma das principais técnicas utilizadas nesse sentido [Nigr 93] e será apresentada na seção seguinte. Um típico sistema de reconhecimento de padrão consiste em três partes: aquisição de dados, seleção/extração de características e classificação/clustering [Pattern recognition book].

- Aquisição de dados: é o processo de seleção dos dados que serão usados como entrada no processo de reconhecimento. Tais dados podem ser qualitativos, quantitativos ou ambos. Podendo ser numéricos, linguísticos, entre outros.

- Seleção/Extração de características: o objetivo principal desta etapa consiste em gerar o melhor conjunto de características necessárias para o processo de reconhecimento, de modo a maximizar a eficácia do sistema. A grande dificuldade dessa etapa está na determinação de um critério adequado para a escolha de um bom conjunto de características. Um bom critério é aquele que é imutável para qualquer variação possível dentro de uma classe, todavia, deve ser capaz de destacar as diferenças importantes a fim de discriminar entre diferentes tipos de padrões.

- Classificação/clustering: classificação é o processo de definição de qual classe um determinado padrão de entrada pertence. Essa classificação pode ser feita utilizando técnicas determinísticas e probabilísticas. As classes são definidas a partir de um conjunto de amostas apresentadas na etapa de aprendizagem.

A técnica de reconhecimento de padrões possui uma grande variedade de aplicações. Dentre elas, pode-se citar: reconhecimento de faces, leitura biométrica, identificação de circuitos impressos defeituosos, reconhecimento óptico de caracteres, reconhecimento de fala e de escrita cursiva, entre outros.

2.3 Redes Neurais Artificiais

Redes Neurais Artificiais (RNAs) são estruturas computacionais que visam imitar a forma com que o cérebro humano processa as informações. O cérebro

possui a capacidade de organizar e encadear seus elementos estruturais, conhecidos como neurônios, para realizar o processamento das informações de forma não linear e paralela. Dessa forma, as principais características das redes neurais são a habilidade de aprender relações complexas e não lineares entre os padrões de entrada e as saídas, utilizarem procedimentos de treinamento seqüencial e se auto-adaptar aos dados [2].

Na sua forma geral, uma rede neural é uma máquina projetada para imitar a forma com que o cérebro realiza uma tarefa particular ou uma função de interesse, podendo ser implementada em hardware ou simulada através de software. Para atingir uma performance razoável, as redes neurais empregam uma massiva interconexão de unidades de processamento simples, denominadas neurônios [1].

As redes neurais possuem a capacidade de aprendizado e, como conseqüência, de generalização. Generalização refere-se à produção de saídas racionais para entradas que não foram apresentadas a rede durante a fase de treinamento ou aprendizado. As capacidades de aprendizado e de generalização tornam as redes neurais capazes de resolver problemas complexos e não-lineares. A seguir, serão explicadas algumas características do neurônio biológico, fundamentais para o entendimento das RNAs.

2.3.1 Neurônio Biológico

O sistema nervoso é formado por um conjunto extremamente complexo de neurônios. Os neurônios estão conectados uns aos outros através de sinapses. Nos neurônios a comunicação é realizada através de impulsos elétricos, quando um impulso é recebido, o neurônio o processa, e passado um limite de ação, dispara um segundo impulso o qual flui do corpo celular para o axônio que, por sua vez, pode ou não estar conectado a um dendrito de outra célula. A

Figura 1 apresenta uma representação de um neurônio.

Os principais componentes de um neurônio são:

- Os dendritos: membrana que recebe os estímulos gerados por outras células. Os dendritos são as entradas do neurônio.
- Soma: é o corpo do neurônio, que é responsável por coletar e combinar informações vindas de outros neurônios;
- O axônio: membrana constituída de uma fibra tubular que é responsável por transmitir os estímulos para outras células. O axônio representa a saída do neurônio.

O neurônio biológico é constituído de um corpo celular denominado soma. Nesse Local ocorre o processamento metabólico da célula nervosa ou neurônio. A partir da soma, projetam-se extensões filamentares denominadas dendritos, e o axônio. Este modelo anatômico foi identificado por Ramon Cajal em 1894. Com base nas pesquisas de Erlanger e Gasser, em 1920, e outras posteriores, passou-se a entender o comportamento do neurônio biológico como sendo o dispositivo computacional do sistema nervoso, o qual possui muitas entradas e uma única saída.

As entradas ocorrem através das conexões sinápticas, que conectam a árvore dendrital aos axônios de outras células nervosas. Os sinais que chegam pelos dendritos são pulsos elétricos conhecidos como impulsos nervosos ou potenciais de ação, e constituem a informação que o neurônio processará de alguma forma para produzir como saída um impulso nervoso no seu axônio.

Sinapse é o nome dado ao ponto de contato entre a terminação axônica de um neurônio e o dendrito de outro. É pelas sinapses que os nodos se unem funcionalmente, formando a rede neural. As sinapses funcionam como

válvulas, e são capazes de controlar a transmissão de impulsos entre os nodos na rede [3] e estão compreendidas entre duas membranas celulares: a membrana pré-sináptica, que recebe o estímulo vindo de uma célula, e a membrana pós-sináptica, que é a do dendrito. Na região pré-sináptica, se o estímulo nervoso recebido atinge um determinado limiar em um espaço curto de tempo, a célula "dispara", produzindo um impulso que é transferido para outras células através de neurotransmissores presentes na membrana dendrital. Dependendo do neurotransmissor, a conexão sináptica é excitatória ou inibitória. A conexão excitatória provoca uma alteração no potencial da membrana que contribui para formação do impulso nervoso no axônio de saída, enquanto que a conexão inibitória age no sentido contrário.

O mecanismo como é criado o potencial de ação ou impulso nervoso é o seguinte: quando o potencial da membrana está menos eletronegativo do que o potencial de repouso, diz-se que a membrana está despolarizada e quando está mais negativo, diz-se que ela está hiperpolarizada. O impulso nervoso ou potencial de ação é uma onda de despolarização de certa duração de tempo, que se propaga ao longo da membrana. A formação de um potencial de ação na membrana axonal ocorre quando essa membrana sofre despolarização suficientemente acentuada para cruzar um determinado valor conhecido como limiar de disparo. Quando esse limiar é superado, os estímulos são passados para outras células através das ligações sinápticas.

2.3.2 O Neurônio Artificial MCP

McCulloch e Pits [4] propuseram um modelo matemático do neurônio artificial MCP. Esse modelo foi proposto com base nos principais conceitos do neurônio biológico. O modelo matemático do neurônio proposto por McCulloch e Pits apresenta n terminais de entrada x1, x2, ..., xn (representando

os dendritos) e apenas um terminal de saída y (representando o axônio). Os terminais de entrada do neurônio têm pesos acoplados w1,w2, ..., wn cujos valores podendo ser positivos ou negativos dependendo de as sinapses correspondentes serem inibitórias ou excitatórias. O efeito de uma sinapse particular i no neurônio pós-sináptico é dado por xiwi. Os pesos determinam o nível em que o neurônio deve considerar sinais de disparo que ocorrem naquela conexão. Uma descrição do modelo está ilustrada na Figura 2.

O corpo do neurônio realiza um simples somatório dos valores xiwi que chegam a ele. Se o somatório dos valores ultrapassa o limiar de ativação ou threshold, o neurônio dispara (valor 1 na saída), caso contrário o neurônio permanece inativo (valor 0 na saída). A ativação do neurônio é realizada através de uma função de ativação, que dispara ou não o neurônio dependendo do valor da soma ponderada das suas entradas. No modelo MCP original, a função de ativação ativará a saída quando:

Na equação acima, n é a quantidade de entradas do neurônio, wi é o peso associado à entrada xi e ? é o limiar do neurônio.

2.3.3 Funções de Ativação

A partir do modelo apresentado por McCulloch e Pits, surgiram vários outros modelos que permitem a produção de qualquer saída, não apenas zero ou um, e com várias funções de ativação. As funções de ativação mais comuns são:

- Função Limiar: função utilizada no modelo de McCulloch e Pits, caracterizada por "tudo ou nada", representada da seguinte forma:

  Onde v é igual ao valor produzido pelo somatório das entradas do neurônio.

- Função Sigmóide: essa é uma função semilinear, limitada e monotônica

que pode assumir valores entre 0 e 1. Existem várias funções sigmodais, porém, a mais usada é a função logística definida pela equação abaixo:

Onde a é o parâmetro de inclinação da função sigmóide e v é o valor de ativação do neurônio.

- Função Signum: essa função apresenta as mesmas características da função limiar, porém, se limita ao intervalo entre 1 e -1. Essa função é representada por:

  Onde b são os limites inferiores e superiores (b = —1— no gráfico) e v é o valor de ativação.

- Tangente Hiperbólica: seu gráfico é parecido com o da Função Sigmóide, assumindo valores entre 1 e -1, sendo representada por:

  Onde a é o parâmetro de inclinação da curva, b são os limites inferiores e superiores (b = —1— no gráfico) e v o valor de ativação.

2.3.4 Arquiteturas de Redes Neurais

A definição da arquitetura de uma RNA define a maneira com que os neurônios são estruturados na rede [1]. A arquitetura restringe o tipo de problema que pode ser tratado pela rede. Redes com uma camada única de nodos MCP, por exemplo, só conseguem resolver problemas linearmente separáveis. Redes recorrentes, por sua vez, são mais apropriadas para resolver problemas que envolvem processamento temporal [3]. A arquitetura de uma rede é definida pelos seguintes parâmetros: número de camadas da rede, número de neurônios em cada camada, tipo de conexão entre os neurônios e topologia da rede.

Em geral, as redes neurais podem ser classificadas quanto ao número

de camadas, quanto ao tipo de conexão e quanto à conectividade entre os neurônios. Quanto ao número de camadas têm-se:

- Redes de uma única camada: existe apenas um nó entre uma entrada e uma saída da rede. A Figura 3 apresenta dois exemplos de redes de única camada.
- Redes de múltiplas camadas: existe mais de um neurônio entre alguma entrada e alguma saída (Figura 4).

Quanto ao tipo de conexão têm-se:

- Feedforward, ou acíclica: a saída de um neurônio na i-ésima camada da rede não pode ser usada como entrada de nodos em camadas de índice menos ou igual a i (Figura 4).
- Feedback, ou cíclica: a saída de algum neurônio na i-ésima camada da rede é utilizada como entrada em neurônios da camada de índice menor ou igual a i (Figura 5).

  Quanto à conectividade têm-se:

  - Redes parcialmente conectadas (Figura 4).
  - Redes completamente conectadas (Figura 3).

2.3.5 Processo de aprendizado

A principal propriedade de uma rede neural é a sua habilidade de aprender sobre o ambiente no qual está inserida de forma a melhorar a sua performance na resolução de problemas complexos. Essa melhora na performance é adquirida a cada instante do processo de aprendizado

de acordo com uma forma de medição pré-estabelecida. Com isso, uma rede neural aprende sobre seu ambiente através de um processo iterativo de ajuste dos pesos sinápticos adquirindo mais conhecimento sobre o problema após cada iteração do processo de aprendizado. Segundo Mendel e McLaren [4], no contexto de redes neurais, aprendizado é o processo pelo qual os parâmetros de uma rede neural são ajustados através de estímulos produzidos pelo ambiente no qual a rede está inserida. O tipo de aprendizado é determinado pela maneira particular com que os parâmetros são modificados. Foram desenvolvidos diversos métodos de treinamento de redes, podendo ser agrupados em dois paradigmas principais: aprendizado supervisionado e aprendizado não-supervisionado. No entanto, outros dois paradigmas bastante conhecidos são os de aprendizado por reforço (que é um caso particular de aprendizado supervisionado) e aprendizado por competição (que é um caso particular de aprendizado não supervisionado).

- Arendizado Supervisionado Esse tipo de aprendizado é o mais utilizado em RNAs. Esse aprendizado é dito supervisionado porque as entrada e as respectivas saídas desejadas são fornecidas por um supervisor externo, normalmente chamado de "professor. Seu objetivo é ajustar os pesos da rede, de forma a encontrar uma ligação entre os pares de entrada e saída fornecidos pelo professor. O professor é responsável por direcionar o processo de aprendizado fornecendo os padrões de entrada, as saídas desejadas e a taxa de erro desejada. A cada padrão de entrada submetido à rede pelo professor, compara-se a resposta desejada (que representa uma solução ótima para aquele padrão de entrada) com a resposta calculada, ajustando-se os pesos das conexões para minimizar o erro.

A minimização da diferença é incremental, já que pequenos ajustes são feitos nos pesos a cada iteração do aprendizado. A soma dos erros quadráticos de todas as saídas é normalmente utilizada como medida de desempenho da rede e também como função de custo a ser minimizada pelo algoritmo de treinamento. A Figura 6 apresenta um esquema básico de aprendizado supervisionado.

Nesse trabalho, o algoritmo de aprendizado utilizado nesse trabalho foi o backpropagation.

– Aprendizado Não Supervisionado. No aprendizado não supervisionado não há um professor que oriente o processo de aprendizado. Ao contrário do aprendizado supervisionado, que possui pares de entrada e saída, para esses algoritmos, somente são disponibilizados os padrões de entrada. De acordo com as regularidades estatísticas com que as entradas ocorrem, a rede cria classes de acordo com as características extraídas de cada padrão de entrada. Dessa forma, para cada entrada fornecida a rede, a saída será a classe a qual a entrada pertence. Caso a entrada não pertença a nenhuma classe pré-existente, a rede cria uma nova classe para essa entrada. A base para a utilização desse tipo de aprendizado é a redundância nos dados, sem redundância, seria praticamente impossível a utilização bem sucedida do aprendizado não supervisionado. A Figura 7 apresenta uma esquema de um sistema de aprendizado não supervisionado.

  A estrutura do sistema de aprendizado não supervisionado pode adquirir uma variedade de formas diferentes. Como exemplo de arquitetura, pode-se considerar uma rede como uma camada de entrada, uma camada de saída, conexões feedforward da entrada

para a saída e conexões laterais dos neurônios da camada de saída. Outro exemplo é uma rede feedforward com múltiplas camadas, em que a livre organização procede na base de camada por camada. Nestes dois exemplos, o processo de aprendizado consiste em modificar repetidamente o peso sináptico de todas as conexões do sistema em resposta às entradas.

2.4 Visão Computacional

Esse módulo tem como principal objetivo o reconhecimento de padrões. No Ariadnes o padrão a ser reconhecido é uma placa com o nome dos locais do ambiente e setas que indicam as direções dos mesmos. Para reconhecimento de uma placa é necessário identificar algumas características de uma imagem, que servirão de padrões de entrada para uma rede neural.

2.4.1 Extração de características

No campo de reconhecimento de padrões, extrair caracerísticas significa extrair medidas associadas ao objeto que se deseja reconhecer, de forma que essas medidas sejam semelhantes para objetos semelhantes e diferentes para objetos distintos [Santos 2007]. Definir vetores de características é o método para representação de dados mais comum e conveniente para problemas de classificação e reconhecimento. Cada característica resulta de uma medição qualitativa ou quantitativa, que é uma variável ou um atributo do objeto [Guyon et al., 2006] . Para reconhecer um caractere de uma representação bitmap, há a necessidade

de extrair características do mesmo para descrevê-lo de uma forma mais apropriada para o seu processamento computacional e reconhecimento. Como o método de extração de características afeta significantemente a qualidade de todo o processo de reconhecimento de padrões, é muito importante extrair características de modo que elas sejam invariantes no que diz respeito às várias condições de iluminação, tipo de fonte e possíveis deformações dos caracteres causadas, por exemplo, pela inclinação da imagem.

Geralmente, a descrição de uma região de uma imagem é baseada em suas representações interna e externa. A representação interna de uma imagem é baseada em suas propriedades regionais, como cor ou textura. A representação externa é selecionada quando se deseja dar ênfase nas características da forma do objeto. Logo, o vetor de características de uma representação externa inclui características como o número de linhas, a quantidade de arestas horizontais, verticais e diagonais, etc.

O conjunto de vetores de características forma um espaço vetorial. Cada caractere representa uma determinada classe, e todas as formas de representação desse caractere definem as instâncias dessa classe. Todas as instâncias do mesmo caractere devem ter uma descrição similar através de vetores numéricos chamados de "'descritores"', ou "'padrões"'. Logo, vetores suficientemente próximos representam o mesmo caractere. Essa é a premissa básica para que o processo de reconhecimento de padrões seja bem sucedido.

No capítulo que trata do toolkit Horus, serão explicados alguns métodos de extração de características implementados no módulo de visão computacional.

2.4.2 Reconhecimento Óptico de Caracteres

Reconhecimento Óptico de Caracteres ou OCR (*Optical character recoginition*) é um campo de pesquisa nas áreas de reconhecimento de padrões, inteligência artificial e visão computacional. Em computação, OCR é o processo de tradução eletrônica de imagens de textos para textos que possam ser editados computacionalmente, permitindo assim, a realização de operações que seriam inviáveis de serem realizadas sobre o texto em formato de imagem. Sistemas OCR, ou de reconhecimento de caracteres, datam do final dos anos 50 e têm sido amplamente utilizados em computadores desktop desde os anos 90. Esses sistemas disponibilizam textos contidos em imagens, capturadas por dispositivos ou geradas computacionalmente, em textos editáveis por computador. Os textos gerados por sistemas OCR são, normalmente, utilizados por outras ferramentas que permitem operações que seriam impossíveis de serem realizadas sobre imagens, como por exemplo, a busca de determinado conteúdo de interesse. Apesar de mais de 40 anos de pesquisa, sistemas OCR ainda estão muito longe de alcançar a eficácia de um ser humano. A eficácia desses sistemas está fortemente ligada à qualidade d as imagens. Imagens limpas e de alta qualidade levam esses sistemas a atingirem uma taxa de aproximadamente 99% de eficácia [1]. Porém, imagens com baixa resolução, com ruído ou com diferenças de luminosidade, por exemplo, podem levar esses sistemas a cometerem erros grosseiros e confundirem diversos tipos de caracteres. Caracteres como "6"e "9", "B"e "8"e "o"e "0", são facilmente confundidos em imagens imperfeitas. Nesse sentido, atualmente, boa parte das pesquisas em OCR está focada na melhoria da sua eficácia no que diz respeito à extração de textos de imagens que não se encontram em condições

ideais. Além disso, o reconhecimento de textos escritos a mão em linguagem cursiva ainda são uma área de pesquisa muito ativa. Há vários sistemas OCR, livres e proprietários, presentes no mercado hoje. Dentre eles temos: O processo de OCR é constituído de várias etapas, com responsabilidades bem definidas, que ao final apresentam o texto editável. A figura 1 apresenta um esquema básico do processo de OCR.

Figura 1: Etapas do processo de OCR

Na figura 1, temos as várias etapas de um processo clássico de OCR. Inicialmente, é necessário tornar a imagem binária, isto é, transformar a imagem, que inicialmente se encontra em escala de cinza, em uma imagem com apenas duas cores: preto e branco, cores essas representadas respectivamente pelos inteiros 0 e 255. Na etapa de segmentação, cada caractere presente na imagem é recortado da imagem binária para ser tratado individualmente. Após a segmentação, cada caractere é passado individualmente para a etapa de extração de características, onde um vetor numérico é extraído a partir das características desse caractere. Por último, é realizada a etapa de reconhecimento do vetor de características, que finalmente deverá apontar o caractere correto. Cada etapa desse processo pode ser implementada de diversas maneiras e por vários algoritmos diferentes. Nas sessões seguintes serão apresentadas algumas formas de implementação dessas etapas juntamente com os algoritmos implementados no toolkit horus utilizados no processo de OCR.

Capítulo 3

Horus

Horus é um toolkit de desenvolvimento e controle de agentes inteligentes, desenvolvido na linguagem de programação Python. Outros exemplos de aplicações desenvolvidas com o Horus são: um sistema de Reconhecimento Automatico de Placas de Automóveis, um sistema de Processamento de Imagens e Extração de Características e um sistema de mapeamento autônomo de ambientes chamado Teseu.

O toolkit Horus fornece os módulos Core, Mapeamento, Visão e Util. O módulo Core apresenta as abstrações que devem ser implementadas pelas aplicações para construir um agente inteligente. O módulo Mapeamento fornece algoritmos de localização, mapeamento e navegação para um agente. O módulo Visão fornece os algoritmos de visão computacional necessários na etapa de reconhecimento de padrões. Por último, o módulo Util fornece um conjunto de funções utilitárias que podem ser usadas tanto no toolkit Horus quanto em qualquer outra aplicação. Cada um desses módulos será explicado nas subseções seguintes.

3.1 Core

O horus pode ser utilizado de dois modos. A primeira forma é como uma coleção de algoritmos, que podem ser utilizados de forma independente. A segunda forma consiste na extensão das abstrações fornecidas pelo módulo core do Horus. Nesse módulo existem abstrações para implementação de agentes, dispositivos, comportamentos e programas de agente. As principais abstrações e seus relacionamentos são apresentados na Figura x.

Arquitetura do Core

Na arquitetura acima, toda a implementação da inteligência do agente deve estar localizada em subclasses da classe *Brain* (cérebro). Essa classe representa o programa de agente. Como a implementaç programa de agente depende de

A ordem de execução dos comportamentos define a máquina de estados de cada agente de acordo com cada aplicação. O toolkit Horus, desenvolvido nesse projeto, permite a configuração dessa máquina de estados.

Todo comportamento deve ser subclasse da abstração *Behavior* presente no horus. Instâncias da classe *Behavior* recebem

3.2 Processamento de Imagem

3.3 Visão

O Sistema de Visão é um dos mais complexos e completos do ser humano, pois fornece um conjunto de informações necessárias à interação do homem com o ambiente. Inicia-se com a captação dos estímulos luminosos do ambiente formando uma imagem, que juntamente aos outros estímulos captados por demais sensores do corpo (som, temperatura, pressão, umidade, cheiro, etc) e as informações contidas na memória, compõem uma cena compreendida pelo cérebro.

Esse módulo tem como principal objetivo o reconhecimento de padrões. No Ariadnes o padrão a ser reconhecido é uma placa com o nome dos locais do ambiente e setas que indicam as direções dos mesmos. Para reconhecimento de uma placa é necessário identificar algumas características de uma imagem, que servirão de padrões de entrada para uma rede neural.

3.3.1 Extração de características

Para realizar o reconhecimento de objetos em uma cena, é necessário extrair características das imagens desse objeto, de forma a identificá-lo, independentemente das variações com que ele possa ocorrer na imagem. O toolkit Horus apresenta três algoritmos para extração de características. Cada um deles será explicado nos itens abaixo.

- Matriz de Pixel

A maneira mais simples de extrair características de um bitmap é associar a luminância de cada pixel com um valor numérico correspondente no vetor de características.

Esse método, apesar de simples, possui alguns problemas que podem torná-lo inadequado para o reconhecimento de caracteres. O tamanho do vetor é igual à altura do bitmap multiplicado pela sua largura, portanto, bitmaps grandes produzem vetores de características muito longos, o que não é muito adequado para o reconhecimento. Logo, o tamanho do bitmap é uma restrição para esse método. Além disso, este método não considera a proximidade geométrica dos pixels, bem como suas relações com a sua vizinhança. No entanto, este método pode ser adequado em situações onde o bitmap do caractere se encontra muito opaco ou muito pequeno para a detecção de arestas.

– Histograma de Arestas por Regiões

Esse método extrai o número de ocorrências de determinados tipos de arestas em uma região específica do bitmap. Isso torna o vetor de características desse método invariante com relação à disposição das arestas em uma região e a pequenas deformações do caractere. Sendo o bitmap representado pela função discreta $f(x, y)$, largura w e altura h, onde $0 \leq x < w$ e $0 \leq y < h$. Primeiramente é realizada a divisão do bitmap em seis regiões $(r0, r1, ..., r5)$ organizadas em três linhas e duas colunas. Quatro layouts podem ser utilizados para a divisão do bitmap em regiões. Definindo a aresta de um caractere como uma matriz 2X2 de transições de branco para preto nos valores dos pixels, tem-se quatorze diferentes tipos de arestas, como ilustrado na figura 4.

O vetor de ocorrências de cada tipo de aresta em cada sub-região da imagem é normalmente muito longo o que não é uma boa prática em reconhecimento de padrões, onde o vetor de características deve ser tão menor quanto possível. Com isso, pode-se agrupar tipos de arestas semelhantes para reduzir o tamanho do vetor de características. Por questões de simplicidade, o agrupamento dos tipos de aresta será desconsiderado no algoritmo de extração de características. Sendo n igual ao número de tipos de arestas diferentes, onde h_i é uma matriz 2×2 que corresponde ao tipo específico de aresta, e p igual ao número de regiões retangulares em um caractere têm-se:

O vetor de características de saída é ilustrado pelo padrão abaixo. A notação $h_j@r_i$ siginifica "número de ocorrências de um tipo de aresta representado pela matriz h_j na região r_i":

– Intensidade de blocos falta falar

3.3.2 Reconhecimento de Objetos

Reconhecimento de objetos é o processo de identificar um determinado objeto através de suas características. Normalmente, esse processo se inicia com a captura de informações sobre o objeto através de câmeras ou outros tipos de sensores, como sonares por exemplo. Em seguida, essas informações passam pelo processo de extração de características com a finalidade de se extrair um vetor de informações que identifiquem unicamente o objeto independente das variações com que ele se apresente. Por fim, esse vetor de características é passado para o processo de reconhecimento, o qual identifica o objeto através de suas

características.

Para tarefas de reconhecimento, o Horus disponibiliza funções para construção e treinamento de redes neurais através da utilização de uma biblioteca denominada FANN (*Fast Artificial Neural Network*). O FANN é uma biblioteca de código aberto implementada em linguagem C que fornece conectores para diversas linguagems de alto nível, dentre elas pode-se citar: Java, C++, Python e Ruby.

Outra funcionalidade disponibilizada pelo horus para reconhecimento de objetos é o módulo de OCR. Esse módulo é utilizado em aplicações em que haja a necessidade de se reonhecer textos existentes em imagens.

falar do tesseract

O módulo OCR é utilizado nas aplicações ANPR e Ariadnes. No ANPR, o módulo OCR é utilizado para reconhecer o texto que se encontra nas placas dos automóveis. Já na aplicação Ariadnes, o módulo OCR é utilizado pelo agente inteligente para reconhecer os textos que se encontram nas placas informativas presentes no ambiente.

3.4 Mapeamento e Navegação

Chamamos de mapeamento ao processo de identificar locais no ambiente do simulador e representa-los em um grafo. O mapeamento no horus utiliza uma técnica genérica denominada SLAM. Nessa técnica, um agente consegue realizar o mapeamento e a localização no ambiente de forma simultânea. Os dispositivos utilizados pela implementação da técnica SLAM são lasers, para identificar obstáculos, e um odômetro, para medir distâncias percorridas.

O SLAM é composto por vários procedimentos interligados. Cada um desses procedimentos pode ser implementado de diversas formas. Dentre os procedimentos implementados no Horus, podemos citar:

1. Landmark Extraction: procedimento responsável pela extração de marcos no ambiente.
2. Data Association: procedimento que associa os dados extraídos de um mesmo marco por diferentes leituras de lasers.
3. State Estimation: procedimento responsável por estimar a posição atual do robô com base em seu odômetro e nas extrações de marcos no ambiente.
4. State Update: procedimento que atualiza o estado atual do agente.
5. Landmark Update: procedimento que atualiza as posições dos marcos no ambiente em relação ao agente.

Neste trabalho, a proposta utilizada é mapear o ambiente através de um grafo conexo, cujos nós referem-se a: entradas/saídas do ambiente, acessos aos cômodos, obstáculos fixos e esquinas. O peso das arestas será calculado de acordo com o custo de processamento no deslocamento entre a posição de um nó ao outro.

O problema de navegação consiste na localização e definição do caminho que o agente deve seguir. Após a construção de uma representação do ambiente em forma de um grafo, o agente é capaz de se localizar e se movimentar pelo ambiente através dos vértices e arestas, previamente mapeados no grafo. Para a utilização de grafos, o Horus fornece classes para su

Capítulo 4

Conclusões e Trabalhos Futuros