UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Um simulador para robótica social aplicado a ambientes internos

José Pedro Ribeiro Belo

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

José Pedro Ribeiro Belo

Um simulador para robótica social aplicado a ambientes internos

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Prof.^a Dr.^a Roseli Aparecida Francelin Romero

USP – São Carlos
Maio de 2018

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

B452s Belo, José Pedro Ribeiro
Um simulador para robótica social voltado para
ambientes internos / José Pedro Ribeiro Belo;
orientadora Roseli Aparecida Francelin Romero. --
São Carlos, 2018.
106 p.

Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática
Computacional) -- Instituto de Ciências Matemáticas
e de Computação, Universidade de São Paulo, 2018.

1. Simulador Robótico. 2. Interação Humano-Robô. 3.
Robótica Social. 4. Ontologia Robótica. 5. Sistema
Cognitivo. I. Romero, Roseli Aparecida Francelin,
orient. II. Título.

José Pedro Ribeiro Belo

A simulator for social robotics applied to indoor environments

Master dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof.^a Dr.^a Roseli Aparecida
Francelin Romero

USP – São Carlos
May 2018

*Dedico este trabalho aos meus pais,
que sempre me deram apoio, força e incentivo
para a realização dos meus sonhos.*

AGRADECIMENTOS

Primeiramente agradeço aos meus pais, Aparecida e José Carlos, que me deram base e força para a conclusão desta etapa em minha vida. Agradeço também à toda minha família e à minha namorada, Ana Paula, por todo apoio e carinho dado.

Agradeço, especialmente, a Prof.^a Dr.^a Roseli Romero por ter me orientado, pelo incentivo e por ter confiado no meu trabalho ao longo do desenvolvimento desta Dissertação de Mestrado.

Aos professores do ICMC, que sempre estiveram dispostos a ensinar, contribuindo para a minha formação e de certa forma para o projeto. Ao Prof.^o Dr.^o Fernando Osório e ao pesquisador Dr.^o Josué Ramos, do CTI Renato Archer, pelas sugestões, correções e orientações durante o exame de qualificação.

Agradeço também aos colegas do grupo de Computação Bioinspirada (Biocom), em especial do Laboratório de Aprendizado de Robôs (LAR), pela disposição em ajudar e discutir ideias. Em particular, agradeço ao colega Helio Azevedo, que direcionou pensamentos e ideias por longas horas de discussão e reflexão sobre o tema da pesquisa. Também aos colegas do Laboratório de Estatística, pelos esclarecimentos prestados.

Agradeço a todos os colegas dentro e fora do ICMC, com os quais tive a oportunidade de conviver e trocar experiências.

Por fim, agradeço ao CNPq, à CAPES e à FAPESP, pelo financiamento total e parcial desta dissertação. Muito obrigado!

*“O primeiro pecado da humanidade foi a fé;
a primeira virtude foi a dúvida.”
(Carl Sagan)*

RESUMO

BELO, J. P. R. **Um simulador para robótica social aplicado a ambientes internos**. 2018. 106 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

A robótica social representa um ramo da interação humano-robô que visa desenvolver robôs para atuar em ambientes não estruturados em parceria direta com seres humanos. O relatório *A Roadmap for U.S. Robotics From Internet to Robotics*, de 2013, preconiza a obtenção de resultados promissores em 12 anos desde que condições apropriadas sejam disponibilizadas para a área. Uma das condições envolve a utilização de ambiente de referência para desenvolver, avaliar e comparar o desempenho de sistemas cognitivos. Este ambiente é denominado *Robot City* com atores, cenários (casas, ruas, cidade) e auditores. Até o momento esse complexo ambiente não se concretizou, possivelmente devido ao elevado custo de implantação e manutenção de uma instalação desse porte. Nesta dissertação é proposto um caminho alternativo através da definição e implementação do simulador de sistemas cognitivos denominado *Robot House Simulator* (RHS). O simulador RHS tem como objetivo disponibilizar um ambiente residencial composto por sala e cozinha, no qual convivem dois agentes, um robô humanoide e um avatar humano. O agente humano é controlado pelo usuário do sistema e o robô é controlado por uma arquitetura cognitiva que determina o comportamento do robô. A arquitetura cognitiva estabelece sua percepção do ambiente através de informações sensoriais supridas pelo RHS e modeladas por uma ontologia denominada *OntSense*. A utilização de uma ontologia garante rigidez formal aos dados sensoriais além de viabilizar um alto nível de abstração. O RHS tem como base a ferramenta de desenvolvimento de jogos Unity sendo aderente ao conceito de código aberto com disponibilização pelo repositório *online* GitHub. A validação do sistema foi realizada através de experimentos que demonstraram a capacidade do simulador em prover um ambiente de validação para arquiteturas cognitivas voltadas à robótica social. O RHS é pioneiro na integração de um simulador e uma arquitetura cognitiva, além disto, é um dos poucos direcionados para robótica social provendo rica informação sensorial, destacando-se o modelamento inédito disponibilizado para os sentidos de olfato e paladar.

Palavras-chave: simulador robótico, interação humano-robô, robótica social, ontologia robótica, sistema cognitivo.

ABSTRACT

BELO, J. P. R. **A simulator for social robotics applied to indoor environments**. 2018. 106 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

Social robotics represents a branch of human-robot interaction that aims to develop robots to work in unstructured environments in direct partnership with humans. The *Roadmap for Robotics from the Internet to Robotics*, 2013, predicts achieving promising results in 12 years as long as appropriate conditions are made available to the area. One of the conditions involves the use of a reference environment to develop, evaluate and compare the performance of cognitive systems. This environment is called *Robot City* with actors, scenarios (houses, streets, city) and auditors. To date, this complex environment has not been materialized, possibly due to its high cost of installing and maintaining. In this dissertation an alternative way is proposed through the definition and implementation of the simulator of cognitive systems called *Robot House Simulator* (RHS). The RHS simulator aims to provide a residential environment composed of living room and kitchen, in which two agents live together, a humanoid robot and a human avatar. The human avatar is controlled by the user of the system and the robot is controlled by a cognitive architecture that determines the behavior of the robot. The cognitive architecture establishes its perception of the environment through sensorial information supplied by the RHS and modeled by an ontology called *OntSense*. The use of an ontology guarantees formal rigidity to the sensory data in addition to enabling a high level of abstraction. The RHS simulator is based on the Unity game engine and is adheres to the open source concept, available on the GitHub online repository. The validation of the system was performed through experiments that demonstrated the simulator's ability to provide a validation environment for cognitive architectures aimed at social robotics. The RHS simulator is a pioneer in the integration of a simulator and a cognitive architecture. In addition, it is one of the few for social robotics to provide rich sensory information where it is worth noting the unprecedented modeling available to the senses of smell and taste.

Keywords: robotic simulator, human-robot interaction, social robotic, robotic ontology, cognitive system.

LISTA DE ILUSTRAÇÕES

Figura 1 – SimRobot: ambiente de simulação com uma cadeira de rodas autônoma . . .	30
Figura 2 – Webots: robô humanoide disposto em um campo de futebol simulado	31
Figura 3 – Stage: percurso de múltiplos robôs	32
Figura 4 – USARSim: robô aéreo explorando escombros	33
Figura 5 – Marilou: cadeira de rodas robô em um ambiente residencial	34
Figura 6 – Klamp't: robô humanoide configurado no ambiente de simulação	35
Figura 7 – SIGVerse: robô humanoide preparando <i>Okonomiyaki</i> para um avatar humano	36
Figura 8 – V-REP: diversidade de tipos de robôs que podem ser simulados e configurados	38
Figura 9 – DRCSim: robô ATLAS em um ambiente de testes	39
Figura 10 – Linha do tempo de arquiteturas cognitivas. As cores indicam o tipo da arquitetura	44
Figura 11 – CMDE	48
Figura 12 – <i>OntSense</i> e ontologias superiores	49
Figura 13 – Olfato modelado no <i>OntSense</i>	51
Figura 14 – Visão Geral do simulador RHS	56
Figura 15 – Hierarquia dos principais componentes do cenário de atuação, divididos em duas categorias, componentes concretos (verde) e abstratos (azul)	57
Figura 16 – Sala de Estar	58
Figura 17 – Cozinha	59
Figura 18 – Avatar humano e agente robótico	61
Figura 19 – Interface do usuário no modo robô	63
Figura 20 – Interface do usuário no modo de controle do avatar humano	65
Figura 21 – Interface do usuário no <i>God Mode</i>	66
Figura 22 – Hierarquia de controle do <i>Simulator Manager</i>	67
Figura 23 – Hierarquia de controle do <i>Canvas</i>	69
Figura 24 – Em azul, raios que auxiliam na detecção de objetos no ambiente	74
Figura 25 – Diagrama de sequência para a execução do pedido: fechar torneira	82
Figura 26 – Exemplo de mensagens do <i>Log</i>	84
Figura 27 – Configuração inicial do ambiente para o primeiro experimento	88
Figura 28 – Interface de usuário apresentando os dados capturados pelos sentidos do robô em diferentes etapas	89
Figura 29 – Avatar humano pegando o pacote de bolachas da mão do robô	93

LISTA DE QUADROS

Quadro 1 – Simuladores robóticos e algumas de suas propriedades, ordenados por ano de desenvolvimento	40
Quadro 2 – Classes-chave da ontologia <i>OntSense</i>	50
Quadro 3 – Comandos que a arquitetura cognitiva pode enviar ao RHS em sua versão atual	53
Quadro 4 – Exemplos de objetos e elementos do ambiente simulado	60
Quadro 5 – Comandos disponíveis no simulador	72
Quadro 6 – Propriedades básicas associadas a objetos e agentes no simulador	76
Quadro 7 – Propriedades perceptíveis pela audição do robô simulado	77
Quadro 8 – Propriedades perceptíveis pelo tato do robô simulado	78
Quadro 9 – Propriedades perceptíveis pelo olfato do robô simulado	79
Quadro 10 – Propriedades perceptíveis pelo paladar do robô simulado	82
Quadro 11 – Experimento 1: comandos e parâmetros utilizados para atender a requisição de pegar um pacote de bolachas pelo avatar humano	90
Quadro 12 – Experimento 2: comandos e parâmetros utilizados para atender o pedido de fechar a torneira	92
Quadro 13 – Experimento 2: comandos e parâmetros utilizados para atender o pedido de pegar um pacote de bolachas	92

LISTA DE ABREVIATURAS E SIGLAS

2D	duas dimensões
3D	três dimensões
CAD	<i>Computer Aided Design</i>
CLR	<i>Common Language Runtime</i>
CMDE	<i>Cognitive Model Development Environment</i>
CST	<i>Cognitive Systems Toolkit</i>
DARPA	<i>Defense Advanced Research Projects Agency</i>
DRC	<i>DARPA Robotics Challenge</i>
DRCSim	<i>DRC Simulator</i>
GNU LGPL	<i>GNU Lesser General Public License</i>
HRP	<i>Humanoid Robotics Project</i>
IA	Inteligência Artificial
ICMC	Instituto de Ciências Matemáticas e de Computação
IHR	Interação Humano-Robô
IK	<i>Inverse Kinematics</i>
Klamp't	<i>Kris' Locomotion and Manipulation Planning Toolbox</i>
LAR	Laboratório de Aprendizado de Robôs
MECA	<i>Multi-purpose Enhanced Cognitive Architecture</i>
MORSE	<i>Modular Open Robots Simulation Engine</i>
MRDS	<i>Microsoft Robotics Developer Studio</i>
OpenHRP	<i>Open Architecture Humanoid Robotics Platform</i>
OpenRAVE	<i>Open Robotics and Animation Virtual Environment</i>
RDF	<i>Resource Description Framework</i>
RHS	<i>Robot House Simulator</i>
SARGE	<i>Search and Rescue Game Environment</i>
STDR Simulator	<i>Simple Two Dimensional Robot Simulator</i>
URI	<i>Uniform Resource Identifier</i>
USARSim	<i>Unified System for Automation and Robotics Simulation</i>
USP	Universidade de São Paulo
V-REP	<i>Virtual Robot Experimentation Platform</i>
VR	<i>Virtual Reality</i>

SUMÁRIO

1	INTRODUÇÃO	23
1.1	Objetivos	25
1.2	Justificativa	26
1.3	Contribuições, evolução e estrutura do documento	26
2	REVISÃO BIBLIOGRÁFICA	29
2.1	Simuladores robóticos	29
2.2	Motores de jogos para simulação	41
2.3	Arquiteturas cognitivas	42
2.4	Considerações finais	45
3	AMBIENTE CMDE	47
3.1	Descrição do ambiente	47
3.1.1	<i>OntSense: uma ontologia para os sentidos</i>	48
3.2	Simulador robótico RHS	52
3.3	Considerações finais	54
4	SIMULADOR RHS	55
4.1	Organização e componentes do sistema	56
4.1.1	<i>Robô</i>	61
4.1.2	<i>Avatar humano</i>	63
4.1.3	<i>God Mode</i>	64
4.2	Comunicação com o usuário e com a arquitetura cognitiva	66
4.2.1	<i>Simulator Manager</i>	66
4.2.2	<i>Canvas</i>	68
4.3	Comandos internos	68
4.4	Percepção do ambiente	73
4.4.1	<i>Visão</i>	74
4.4.2	<i>Audição</i>	77
4.4.3	<i>Tato</i>	78
4.4.4	<i>Olfato</i>	79
4.4.5	<i>Paladar</i>	80
4.5	Interação do simulador RHS com arquitetura cognitiva	82
4.6	<i>Log de dados</i>	83

4.7	Primeira versão e distribuição do software	84
4.8	Discussão	85
4.9	Considerações finais	86
5	VALIDAÇÃO DO RHS	87
5.1	Experimentos realizados	87
5.1.1	<i>Experimento inicial utilizando um driver</i>	87
5.1.2	<i>Experimento com o simulador integrado ao CMDE</i>	91
5.2	Considerações finais	94
6	CONCLUSÃO E TRABALHOS FUTUROS	95
	REFERÊNCIAS	99

INTRODUÇÃO

Os sistemas robóticos, inicialmente restritos a ambientes estruturados e atividades repetitivas em ambiente fabril, estão evoluindo rapidamente para atuar também em ambientes não estruturados em parceria com seres humanos. A evolução natural da robótica visa uma maior aproximação com os seres humanos, tanto na robótica social quanto na industrial. A robótica social tem com objetivo desenvolver robôs que interajam de forma natural com humanos, fisicamente ou socialmente, permitindo uma vida confortável e autônoma. Por outro lado, a indústria vem integrando humanos e robôs em suas linhas de produção direcionando para um futuro onde ambos irão compartilhar espaço para realizar, em conjunto, tarefas envolvidas em processos fabris.

Este cenário constituído por robôs onipresentes demanda o equacionamento de diversos desafios. Parte desses desafios encontram-se em um campo crescente da robótica conhecido como Interação Humano-Robô (IHR) (GOODRICH; SCHULTZ, 2007). Em particular, na subárea definida por Fong, Nourbakhsh e Dautenhahn (2003) como *Socially Interactive Robots* ou “Robótica Social”, termo utilizado nesta dissertação.

As aplicações da robótica social abrangem desde tarefas que envolvem busca e resgate, até as áreas da educação e robótica assistiva, além de várias outras aplicações que requeiram algum tipo de habilidade social.

Os ambientes estruturados são tradicionalmente modelados por eventos e sequências de ações coordenadas por máquina de estados finitos. Entretanto, ambientes não estruturados requerem uma estratégia de processamento diferente¹. Neste contexto, a cognição assume um papel preponderante e o uso de arquiteturas cognitivas torna-se uma opção interessante como

¹ Nesta dissertação, são adotados os conceitos de ambientes não estruturados e estruturados definidos por Oliver, Jeaneung e Marc (2016). Ambientes estruturados dizem respeito a ambientes controlados, como chão de fábrica e instalações experimentais específicas. Ambientes não estruturados se referem a ambientes que não foram modificados especificadamente para facilitar a execução de tarefas por parte do robô.

apoio para tarefas que envolvam robótica social (KOTSERUBA; TSOTSOS, 2017). A interação do robô com um ambiente não estruturado representa um desafio em termos de reconhecimento e de inferência sobre as características e funções de centenas de objetos presentes no ambiente. Paralelamente, a necessidade de identificar sons, sabores, texturas, etc., torna-se crítica em atividades que exigem forte interação com os seres humanos.

Para que robôs possam interagir diretamente com seres humanos, é desejável que a comunicação seja realizada no mesmo nível cognitivo. Para alcançar este objetivo é necessário se apoiar na ciência cognitiva. A ciência cognitiva investiga os processos associados ao processamento do conhecimento realizado pela mente humana (REISBERG, 2013). A percepção do ambiente representa o ponto inicial do aprendizado, a partir deste ponto modelos cognitivos são construídos e refinados a medida que o ser humano amadurece e adquire conhecimentos adicionais. Dentre os tópicos de interesse da área, temos: percepção visual, atenção e conscientização, memória, representação mental, linguagem, emoção, raciocínio, criatividade, cultura e sociedade (REISBERG, 2013).

Atualmente, os sistemas cognitivos aplicados na robótica interagem diretamente com os sensores e atuadores presentes no robô. Essa abordagem possui as seguintes desvantagens:

- O especialista em modelo cognitivo é obrigado a se envolver em questões como: movimento do robô, síntese de voz, captura de sons, visão, controle dos graus de liberdade, etc. Estas questões desviam o pesquisador do foco principal, que é capacitar o robô a interagir socialmente através da construção de modelos para as atividades a serem executadas no ambiente alvo.
- A existência de múltiplas plataformas de robôs, muitas das quais proprietárias e com interfaces distintas, minimiza a troca de conhecimentos e habilidades entre os pesquisadores.
- A reprodução dos experimentos pode ser de alto custo devido ao fato que quanto mais recursos um robô possui maior será o recurso financeiro dispendido.
- A comparação entre implementações de sistemas cognitivos distintos torna-se complexa exatamente pela dificuldade em reproduzir os experimentos realizados.

Todos estes fatores podem influenciar negativamente no desenvolvimento de um determinado projeto e postergar a obtenção de resultados. A robótica social, em geral, e sistemas cognitivos, em particular, ainda sofrem de um sério problema que atrasa sua evolução: a ausência de *benchmarks* e simuladores apropriados para validação de modelos. Esse aspecto foi ressaltado no relatório *Roadmap for U.S. Robotics: From Internet to Robotics* (Robotics VO, 2013):

”... foram sugeridos ambientes de referência completos para desenvolver, avaliar e comparar o desempenho com relação a uma aplicação ou implementação particular. Esses ambientes podem variar em tamanho

e complexidade, de um espaço de trabalho simples (uma mesa de escritório ou de uma bancada de cozinha) para uma sala, uma casa, ou um quarteirão inteiro. Neste contexto, a noção de uma cidade de robôs (*Robot City*) foi mencionada. Trata-se de um ambiente urbano comum, em que todos os habitantes fazem parte da experiência e ajudam no processo de avaliação, bem como na definição de requisitos adequados para ambientes de aplicação cotidiana.“

Um ambiente semelhante foi criado pela Google para testar seus carros autônomos (AUSTIN, 2017). Neste caso, foi construída uma cidade no deserto da Califórnia, onde são desenvolvidos cenários que desafiam os carros autônomos à responderem de forma aceitável situações elaboradas em ambientes compostos por cones, sinalização, manequins, outros carros e até mesmo pedestres humanos, participantes do experimento. Outros ambientes para testes englobam os criados pelo *Defense Advanced Research Projects Agency* (DARPA) para o *Grand Challenge* (DARPA, 2014) e *Robotics Challenge* (DARPA, 2015). Entretanto, todos eles apresentam um custo elevado e inacessível para a maioria dos grupos de pesquisa da área. Uma estratégia alternativa é o uso de simuladores em lugar de cidades emuladas.

Os simuladores robóticos permitem o desenvolvimento e validação de sistemas antes de serem utilizados em robôs reais, evitando riscos além do desgaste destes robôs desnecessariamente. Além disto, os simuladores facilitam a implementação de algoritmos para sensores ainda indisponíveis no mercado. Uma estratégia para o desenvolvimento de simuladores é a utilização de motores de jogos. Isto porque, segundo Alexander *et al.* (2005), os motores de jogos têm grande potencial em oferecer um ambiente bastante fiel para simulações computacionais.

Está em desenvolvimento no Laboratório de Aprendizado de Robôs - LAR/ICMC/USP um ambiente para validação de sistemas cognitivos aplicados em robótica social. Esta dissertação apresenta o componente principal deste ambiente que é um simulador cognitivo, denominado *Robot House Simulator* (RHS). O simulador RHS tem sido desenvolvido como o apoio da ferramenta de desenvolvimento de jogos *Unity*. Distinto dos simuladores robóticos clássicos, o foco é garantir a interação social do robô, ou seja, interação com um avatar humano, reação a recepção de comandos e atuação em um ambiente não estruturado.

1.1 Objetivos

O objetivo geral desta dissertação é contribuir na área de robôs socialmente interativos através das seguintes ações:

- acelerar o processo de implementação de sistemas cognitivos em robôs;
- viabilizar a reprodução de experimentos associados a sistemas cognitivos;
- permitir a comparação entre implementações distintas;

- apoiar a instalação de cursos na área de IHR, com foco em robótica social, pela viabilização de simuladores cognitivos nos experimentos práticos (BERRY, 2015); e
- reduzir o esforço do especialista em sistemas cognitivos no desenvolvimento de sistemas voltados para a robótica social.

O objetivo específico consiste no desenvolvimento de um simulador, RHS, que modele as ações necessárias para exercitar sistemas cognitivos utilizados em robótica social. A elaboração do simulador também envolve como premissa básica ser aderente ao conceito de código aberto viabilizando sua utilização, manutenção e evolução por outros grupos.

1.2 Justificativa

A utilização de robôs físicos atuando em ambientes complexos pode demandar um custo inviável, principalmente para pequenos laboratórios. A hipótese deste trabalho é que a utilização de um simulador com ambientes definidos, objetos, obstáculos, e agentes (humano e robótico) permita exercitar o uso de sistemas cognitivos voltados à robótica social, com baixo custo de utilização.

A escolha do desenvolvimento de um simulador robótico, com ambientes para modelos cognitivos, apresenta-se como uma solução eficiente para contornar os problemas de custo de implementação. Algumas vantagens podem ser apontadas na utilização desta abordagem:

- o ambiente de simulação pode ser copiado e disseminado facilmente através da web;
- não existe a necessidade da utilização de um ambiente físico para testes com o robô; somente um computador configurado apropriadamente;
- a utilização de plataformas, voltadas para o desenvolvimento de jogos, facilita o desenvolvimento e acelera a geração de resultados;
- o desenvolvedor de sistemas cognitivos não terá que se envolver diretamente com a complexidade de sensores e atuadores, pois o simulador realiza esse controle automaticamente; e
- aumenta a vida útil dos robôs reais minimizando seus desgastes com testes em ambientes reais.

1.3 Contribuições, evolução e estrutura do documento

A principal contribuição deste projeto é facilitar o desenvolvimento de sistemas cognitivos voltados para robótica social. Não menos importante, os resultados alcançados contribuem diretamente com uma tese de doutorado (AZEVEDO, 2016), em andamento no Instituto de

Ciências Matemáticas e de Computação (ICMC) da Universidade de São Paulo (USP), Campus de São Carlos, junto ao Laboratório de Aprendizado de Robôs (LAR). Vale ressaltar que o laboratório e a orientação utilizados na tese de doutorado são os mesmos associados a esta dissertação. Essa integração garante uma evolução consistente na medida que viabiliza a interação de pesquisadores no desenvolvimento de um ambiente complexo, envolvendo cognição e robótica, em evolução no laboratório.

O escopo inicial do projeto envolve elaborar um ambiente domiciliar, especificadamente, sala e cozinha. A evolução natural do simulador abrange o modelamento de outros ambientes internos como banheiro, quarto, lavanderia e etc., ou mesmo ambientes externos, tais como, casas vizinhas, supermercados, hospitais e farmácias. Naturalmente, a definição de novos cenários demanda a concepção de missões e ações adicionais, enriquecendo o simulador proposto. Outra evolução que a princípio pode ser identificada, é a utilização exclusiva de avatar adulto. A inclusão de avatares representando crianças, idosos e pessoas com dificuldades físicas no ambiente de simulação, viabilizará a realização de experimentos com uma população mais fidedigna de seres humanos com potencial de interação com robôs em um ambiente residencial.

Esta dissertação está estruturada da seguinte forma. No [Capítulo 2](#), é apresentado o estado da arte de pesquisas em simuladores robóticos e arquitetura cognitivas. Também são tratados alguns trabalhos que utilizam motores de desenvolvimento de jogos para a simulação de robôs. No [Capítulo 3](#), a pesquisa proposta é detalhada, posicionando o simulador robótico no ambiente em desenvolvimento no LAR e alguns aspectos de sua implementação são descritos.

No [Capítulo 4](#), o simulador desenvolvido é apresentado, levando em consideração os materiais e métodos utilizados, além de conceitos definidos com o intuito de alcançar os objetivos do trabalho. Os experimentos e estudos de casos são tratados no [Capítulo 5](#), bem como uma discussão sobre os resultados. No [Capítulo 6](#), são apresentadas as conclusões e trabalhos futuros.

REVISÃO BIBLIOGRÁFICA

O foco principal deste trabalho envolve o uso de simuladores para agentes robóticos, em particular seu uso em robótica social. Existem diversos simuladores robóticos que visam dar suporte a diversos tipos de robôs (humanoides, móveis, industriais, etc) e outros em apenas um tipo ou modelo, tais como robôs industriais ou cirúrgicos de determinada companhia.

Neste capítulo, é apresentada uma visão geral dos simuladores robóticos, ressaltando os simuladores associados de alguma forma com IHR tais como robôs móveis, humanoides e simuladores genéricos. Será apresentada também uma abordagem que utiliza motores de desenvolvimento de jogos para a simulação e posteriormente a área das arquiteturas cognitivas. O objetivo deste capítulo é estabelecer o estado da arte em simuladores robóticos, além de contribuir para determinar conceitos e atributos necessários para a construção do simulador proposto levando em consideração aspectos cognitivos.

2.1 Simuladores robóticos

Simulações são essenciais pois permitem validar diversas implementações antes de integrar o sistema com um robô real. Além disso, é interessante evitar a utilização de robôs desnecessariamente, pois a vida útil dos mesmos decai lentamente com o uso. Isto faz com que os movimentos mudem de acordo com os desgastes dos robôs, as vezes prejudicando o algoritmo ou técnica testada (ECHEVERRIA *et al.*, 2012; KUO *et al.*, 2013).

Desde o início da pesquisa em robótica existe a necessidade de simular ações e reações de diferentes máquinas autônomas, e por isso a utilização de simuladores robóticos tem crescido com o campo da robótica (HARRIS; CONRAD, 2011). Os primeiros esforços no desenvolvimento de ferramentas de simulação voltadas a robótica são datados da década de 1990. Em 1994, Siems, Herwig e Röfer (1994) afirmaram que a pesquisa com robôs autônomos é muito dependente de um hardware robusto, muitas vezes inacessível para alguns grupos de pesquisas. Desta forma, os

autores propuseram o *SimRobot*, como uma alternativa para a realização de experimentos com robôs.

Figura 1 – SimRobot: ambiente de simulação com uma cadeira de rodas autônoma

Fonte: adaptada de Röfer (1998).

Na época, o SimRobot permitia a definição de objetos hierárquicos, articulações rotativas e telescópicas, sensores de direção, distância, cor e luz, bem como câmeras bidimensionais. Além disso, era possível utilizar a linguagem C para definir e configurar objetos e adicioná-los ao ambiente de simulação. Mesmo sendo um dos pioneiros, este simulador oferecia um ambiente com elementos em três dimensões (3D), dando maior realidade aos experimentos, como apresentado na Figura 1.

Outro pioneiro entre os simuladores robóticos é o *Khepera Simulator* (MICHEL, 1996), disponibilizado livremente na internet em 1995. Foi proposto para oferecer suporte ao robô Khepera em um ambiente de duas dimensões (2D). Em 1998, Michel (1998) propôs o Webots, com o objetivo preliminar de melhorar o simulador Khepera, adicionando mecanismos realistas de renderização em 3D, disponibilizando suporte para diversas arquiteturas robóticas (robôs móveis,

humanoides, etc.). Atualmente o Webots é um software comercial mantido pela Cyberbotics (CYBERBOTICS, 2018), sendo utilizado em mais de 1300 universidades e centros de pesquisa de todo o mundo. Na Figura 2, é ilustrado o ambiente de simulação do Webots apresentando um robô humanoide disposto em um campo de futebol.

Figura 2 – Webots: robô humanoide disposto em um campo de futebol simulado

Fonte: Cyberbotics (2018).

Um ano depois da proposta do Webots, no Laboratório *USC Robotics Research Laboratory* foi dado início ao desenvolvimento do projeto *Player/Stage* devido a necessidade de um simulador para sistemas multi-robôs (PLAYER, 2014). O Player fornece uma interface para robôs e sensores em um modelo cliente/servidor. Sua versatilidade permite que programas escritos em qualquer linguagem se comuniquem via rede com múltiplos robôs clientes. Já o Stage é um simulador 2D de baixa fidelidade para uma grande população de robôs. O conjunto Player/Stage foi adotado por inúmeros laboratórios totalizando 310,536 downloads no final de 2017, desde sua liberação no Source Forge¹. Entretanto, seu uso generalizado vem gradativamente diminuindo de um pico máximo de 5165 downloads em nov/2009 para 241 downloads em dez/2017, possivelmente pela liberação de ambientes mais versáteis, como por exemplo ROS (ROS, 2018).

O Stage é normalmente utilizado como um *plugin* do Player fornecendo dispositivos virtuais para simulações. A implementação cliente/servidor do conjunto garante que a troca do mundo virtual para o real ocorra sem alterações severas de código. O simulador oferece uma

¹ <sourceforge.net/projects/playerstage>

Figura 3 – Stage: percurso de múltiplos robôs

Fonte: Player (2014)

visão 2D (Figura 3) do ambiente e do robô simulado, sendo possível acompanhar as leituras dos sensores instanciados no robô em tempo real de simulação.

Outro produto originado do projeto Player foi o *Gazebo* (Open Source Robotics Foundation, 2014a). Este é um simulador 3D de alta fidelidade (KOENIG; HOWARD, 2004), para uma pequena população de robôs. Foi desenvolvido em 2002, inicialmente liberado como complemento do Player e do Stage, mas acabou ganhando autonomia em 2011, sendo hoje utilizado com outras ferramentas.

O Gazebo é um simulador de código aberto que permite testar rapidamente algoritmos e projetos de robôs utilizando cenários realistas em ambientes internos e externos. Ele fornece aos usuários um mecanismo robusto para simular eventos físicos com representações gráficas de alta qualidade através de uma interface de programação.

Apesar do Player tradicionalmente utilizar os simuladores Stage/Gazebo, é também possível utilizar o Player com outros simuladores. O *Unified System for Automation and Robotics Simulation* (USARSim) (CARPIN *et al.*, 2007) é um exemplo. O desenvolvimento do USARSim ocorreu também em 2002, com o intuito de oferecer suporte a aplicações de pesquisa e resgate urbano. Ele evoluiu para atender diversos propósitos, suportando uma variedade de robôs distintos, incluindo robôs com rodas, pernas e até mesmo robôs voadores, como apresentado na Figura 4. O USARSim é um simulador gratuito baseado na plataforma de desenvolvimento de jogos Unreal Engine e atualmente é utilizado nas competições *RoboCup* (ROBOCUP, 2016).

Em 1998, no Japão, iniciou-se um projeto de robótica com o patrocínio da Honda intitulado *Humanoid Robotics Project* (HRP) com o objetivo de desenvolver diversos tipos de robôs humanoides (INOUE; HIRUKAWA, 2000). Ao longo dos anos, foram desenvolvidas ferramentas com o intuito de auxiliar no desenvolvimento destes robôs, incluindo o simulador *Open*

Figura 4 – USARSim: robô aéreo explorando escombros

Fonte: [USARSim \(2015\)](#)

Architecture Humanoid Robotics Platform (OpenHRP) disponibilizado em 2003. Atualmente, o OpenHRP está na versão 3 e é um software de código aberto. Ele é dotado de um ambiente complexo baseado em um motor de física, o qual permite a simulação dinâmica de robôs humanoides. Além disto, o OpenHRP provê bibliotecas de controle de movimento, compatível com robôs reais. De forma resumida, este simulador oferece um ambiente de validação e testes para robôs humanoides, levando-se em consideração aspectos físicos do ambiente.

Vários simuladores têm como objetivo modelar robôs de forma mais real possível, devido a necessidade de validar aspectos físicos destes agentes. Porém, outros têm como objetivo prover suporte a validação de técnicas de Inteligência Artificial (IA). Em 2005, o simulador 3D *Simbad* ([SIMBAD, 2011](#)) foi desenvolvido em Java, voltado para propósitos científicos e educacionais. Segundo [Hugues, Bredeche e Futurs \(2006\)](#), o simulador é dedicado a pesquisadores que necessitam de uma base simples para estudar IA e aprendizado de máquina, no contexto de robôs e agentes autônomos. O sistema ainda provê bibliotecas para algoritmos evolutivos e redes neurais artificiais.

No mesmo ano, a Energid, durante um contrato de trabalho com um centro espacial da NASA, desenvolveu o ambiente de simulação e de controle Actin® ([ENERGID TECHNOLOGIES CORPORATION, 2015](#)). Trata-se de uma ferramenta comercial voltada a demonstração, teste e validação de diversos modelos de robôs, incluindo humanoides, manipuladores, veículos terrestres, aéreos, aquáticos, etc. Além da NASA, o Actin® é também utilizado pela DARPA,

uma agência norte americana responsável por desenvolver tecnologias para uso militar(DARPA, 2018).

Em 2006, a Microsoft lançou o *Microsoft Robotics Developer Studio* (MRDS) (JACKSON, 2007), voltado para hobistas, desenvolvedores acadêmicos e profissionais, viabilizando a criação de aplicativos de robótica em uma ampla variedade de cenários. Este é um simulador 3D que utiliza bibliotecas avançadas de física e renderização. Ele oferece também suporte a diversas plataformas robóticas, sendo possível controlar o robô real através dela. Sua última versão, versão 4, foi lançada em 2012 e oferece diversos ambientes, tais como apartamentos, fábricas, casas, ambientes externos e urbanos.

Figura 5 – Marilou: cadeira de rodas robô em um ambiente residencial

Fonte: anyCode (2016)

Diankov e Kuffner (2008) apontaram que um dos desafios em desenvolver robôs autônomos é a necessidade de integrar e testar rigorosamente *scripts* de alto nível, tais como algoritmos de navegação, de percepção e de controle. Desta forma, os autores propuseram o simulador *Open Robotics and Animation Virtual Environment* (OpenRAVE). Este tem como objetivo oferecer um ambiente para testar, desenvolver e implementar algoritmos de planejamento de movimento, onde o foco principal é a simulação e análise de informações cinemáticas e geométricas relacionadas a algoritmos de planejamento. O desenvolvimento do OpenRAVE teve início em 2006, sendo lançado em 2008 com a licença de uso GNU Lesser General Public License (GNU LGPL), que garante inclusive seu uso em aplicações comerciais.

Como apresentado anteriormente, alguns simuladores oferecem uma gama de agentes

idênticos a modelos de robôs existentes. Porém, outros disponibilizam ferramentas para que usuários criem seus próprios robôs, utilizando técnicas de CAD². O simulador *Marilou Robotics Studio* (ANYCODE, 2016), ou *Marilou anyKode*, é um exemplo de simulador que oferece CAD para a modelagem de robôs. Tendo seu desenvolvimento sido iniciado em 2008, o Marilou tem como objetivo prover um ambiente de simulação de robôs móveis, humanoides, robôs manipuladores e multi-robôs, utilizando bibliotecas que dão fidelidade às forças físicas do ambiente (Figura 5). Contudo, é uma ferramenta de código fechado, tendo diversas licenças que podem ser adquiridas para cada tipo de necessidade, incluindo uma licença gratuita que pode ser usada somente como *hobby* e não para uso comercial ou educacional.

Figura 6 – Klamp't: robô humanoide configurado no ambiente de simulação

Fonte: INTELLIGENT MOTION LAB (2017)

A necessidade de utilizar técnicas alternativas ao uso de robôs reais é evidente. A utilização de robôs reais as vezes é inviável para vários laboratórios, devido principalmente ao alto custo de aquisição ou pela complexidade do ambiente de utilização do agente robótico. Em 2009, na Universidade de Indiana, pesquisadores desenvolveram o *Kris' Locomotion and Manipulation Planning Toolbox* (Klamp't) (INTELLIGENT MOTION LAB, 2017), inicialmente, como uma plataforma de pesquisa robótica, porém sendo utilizado como ferramenta educacional ao longo dos anos. O Klamp't destaca-se pelo suporte a aspectos robóticos de locomoção e manipulação, fornecendo mecanismos para analisar e programar robôs, permitindo a prototipagem de comportamentos inteligentes. Nos últimos anos, foi utilizado em diversos projetos tais como, *Amazon Picking Challenge* (ROBOCUP2016, 2016), *DARPA Robotics Challenge* (DRC) (DARPA, 2015)

² Ferramenta digital para desenvolvimento de projetos e desenhos técnicos.

e o *IROS 2016 Robot Grasping and Manipulation Challenge* (HAUSER, 2016). Na Figura 6 é apresentado o ambiente de simulação com a presença de um robô humanoide.

Outra ferramenta direcionada a propósitos educacionais e de pesquisa é o *Lpzrobots* (DER; MARTIUS, 2011). Desenvolvido em 2009, na Universidade de Leipzig, Alemanha, este simulador tem como foco prover um ambiente 3D com forças físicas e um controlador que permite o desenvolvimento e testes de algoritmos para robôs reais e simulados. O simulador de física pode manipular corpos rígidos, sendo possível simular o comportamento de juntas, fricção, elasticidade, deslizamento, etc.

Segundo Inamura *et al.* (2010), a compreensão dos mecanismos da inteligência dos seres vivos é uma das mais importantes abordagens para desenvolvimento de um robô inteligente. Para tal compreensão, é necessário o envolvimento de diversas áreas, porém, a pesquisa colaborativa é um processo demorado e trabalhoso, isto porque as ferramentas e experiências de cada área são muito diferentes. Nesse contexto, em 2010, Inamura *et al.* (2010) propuseram um ambiente denominado *SIGVerse*, que integra simulações físicas e de comunicação social com o objetivo de prover uma abordagem para o estudo sobre a inteligência social juntamente com a robótica.

Figura 7 – SIGVerse: robô humanoide preparando *Okonomiyaki* para um avatar humano

Fonte: Inamura (2011)

O SIGVerse permite que os experimentos de IHR sejam validados disponibilizando um avatar humano que pode interagir com o robô no ambiente simulado. Na Figura 7, é apresentado um exemplo desta interação, onde um robô cozinheiro prepara um prato enquanto o avatar

humano (controlado pelo usuário) guia o robô com ingredientes que devem compor a receita, no caso *Okonomiyaki*³. Versões atuais deste simulador permitem que o usuário tenha uma imersão maior na simulação através de óculos de *Virtual Reality* (VR)⁴ e do sensor Microsoft Kinect⁵. A tecnologia VR também está se desenvolvendo rapidamente nos últimos anos. Ela viabiliza a interação humano-computador e permite interação com robôs de forma semelhante ao que ocorre em um ambiente real (LI *et al.*, 2015).

Em Novikova, Watts e Inamura (2015), foi apresentado um experimento envolvendo aspectos cognitivos utilizando SIGVerse (INAMURA; MIZUCHI, 2017), com o intuito de explorar o potencial do simulador para a IHR. Neste experimento, um avatar humano é controlado pelo usuário através de controles imersivos onde ele interage com um robô simulado. Os aspectos cognitivos envolvem a percepção de emoções do avatar por parte do robô, sendo duas as emoções: surpresa e alegria. Apesar de tudo, este é um trabalho inicial, no qual o aspecto cognitivo é minimizado pela captura de somente duas emoções, sendo o foco do trabalho direcionado para demonstrar a presença de dois agentes (humano e robô) no mesmo ambiente simulado.

De acordo com Echeverria *et al.* (2011), para as validações em robôs serem úteis, as simulações devem prover fidelidade suficiente ao robô real. Com base nesta premissa, foi proposto o *Modular Open Robots Simulation Engine* (MORSE), um simulador de código aberto voltado para a pesquisa robótica. Ele oferece um ambiente 3D, no qual robôs virtuais podem interagir usando sensores e atuadores, os quais funcionam da mesma maneira de seus correspondentes do mundo real. Em Echeverria *et al.* (2012), o MORSE é voltado para a área de IHR, no qual foi adicionado um avatar humano e uma interface de usuário para que o mesmo tenha uma imersão maior na simulação. As interfaces utilizadas são o controle Nintendo WiiMote e o Microsoft Kinect.

Outro simulador que preza pela fidelidade na representação virtual dos robôs e das forças físicas do ambiente é o *Virtual Robot Experimentation Platform* (V-REP) (ROHMER; SINGH; FREESE, 2013). Ainda em 2010, o V-REP foi desenvolvido com o objetivo de tornar as simulações e modelos mais acessíveis, reduzir o custo das implementações de sistemas robóticos, aumentar a produtividade, facilitar a verificação de sistemas e agilizar a prototipagem. O V-REP oferece um ambiente 3D, onde cada objeto pode ser controlado individualmente através de um *script* ou de um nó ROS (ROS, 2018), além disto ele disponibiliza um componente (*Remote API*) que permite que a simulação seja controlada de forma remota. Todas estas características tornam o V-REP versátil e ideal para aplicações multi-robôs. Como ilustrado na Figura 8, este simulador permite a modelagem e simulação de uma variedade de robôs.

A maioria dos simuladores apresentados tem como objetivo contribuir para áreas de pesquisa, da educação, comercial, entre outras. Porém, a simulação pode ser utilizada com

³ Comida japonesa, uma espécie de panqueca com ingredientes diversos.

⁴ Realidade Virtual, em português.

⁵ Sensor de movimentos que permite o usuário interagir com os jogos eletrônicos sem a necessidade de utilizar um controle manual.

Figura 8 – V-REP: diversidade de tipos de robôs que podem ser simulados e configurados

Fonte: Rohmer, Singh e Freese (2013)

outros propósitos, como por exemplo competições. O USARSim e o Klamp't, apresentados anteriormente, são alguns exemplos de simuladores utilizados nesta modalidade. Contudo, outros são criados justamente para atender as necessidades de determinada competição, como é o caso do *DRC Simulator* (DRCSim) (OPENSOURCEROBOTICSFOUNDATION, 2014). Esta ferramenta surgiu em 2013, com o propósito de dar suporte ao desafio DRC, que por sua vez, tem como objetivo catalisar o desenvolvimento de tecnologias para operações de contingência em instalações, que sofreram danos devido a ocorrência de desastres (DARPA, 2015).

A DRC foi uma competição realizada em junho de 2015, cujas equipes de pesquisadores competiram por um prêmio de \$2M de dólares. Nesta competição, os robôs deveriam executar uma missão em um ambiente real. A missão consistiu em entrar, dirigir e sair de um veículo, caminhar em um terreno acidentado, entrar por uma porta, subir escada, atravessar uma passarela, usar uma ferramenta para abrir passagem através de uma parede, abrir uma válvula e por fim, conectar uma mangueira de incêndio. Neste contexto, o DRCSim foi utilizado para a classificação das equipes, onde estas participavam de provas eliminatórias. O DRCSim (Figura 9) é uma plataforma de código aberto, que calcula e exibe comportamentos físicos e sensoriais dos robôs em um espaço virtual 3D. Esse simulador representa uma contratação especial do grupo que desenvolve o simulador Gazebo para simular o ambiente da competição utilizando o robô *ATLAS*, da Boston Dynamics Robotics (Boston Dynamics, 2017).

Durantes os anos, vários simuladores surgiram com o objetivo de prover um ambiente de simulação realista com uma grande quantidade de serviços e parâmetros de configuração. Contudo, estas características trazem uma maior complexidade a ferramenta, consequentemente, aumentando sua curva de aprendizado e utilização. Em Tsardoulis, Zalidis e Thallas (2014),

Figura 9 – DRCSim: robô ATLAS em um ambiente de testes

Fonte: [OpenSourceRoboticsFoundation \(2014\)](#)

foi apresentado um simulador com a finalidade de oferecer um ambiente para simulação de um robô, ou de um enxame de robôs, da forma mais simples possível. Denominado como *Simple Two Dimensional Robot Simulator* (STDR Simulator), ele oferece um ambiente 2D, no qual é possível validar algoritmos sem a utilização do ambiente gráfico, isto através de conexões *ssh*. Ele é totalmente compatível com o ROS, o que facilita a utilização dos algoritmos em robôs reais ou em outros simuladores.

No [Quadro 1](#) é compilado o conjunto de simuladores apresentados e que de alguma forma estão relacionados com a área do trabalho proposto. No quadro, os simuladores estão ordenados pelo ano de sua concepção, seguidos por uma breve descrição, suas licenças de utilização, a última versão lançada (até janeiro de 2018), o ano de liberação da última versão de cada um e se oferecem suporte a simulação em 3D. Considerando a data de liberação da última versão, pode-se identificar um grupo de simuladores que se encontram em atividade maior com liberações em 2017: Webots, Stage, Gazebo, USARSim, Klamp't, SIGVerse e V-REP.

Como já dito, o foco da revisão foi dado em simuladores genéricos (que modelam vários tipos de robôs e ambientes), simuladores para robótica móvel, para humanoides, além é claro, para a área de IHR. Apesar dos serviços de robôs na indústria impulsionarem a investigação da área de IHR ([TAN; INAMURA, 2012](#)), os simuladores robóticos voltados para a indústria não foram abordados devido a esta área ser caracterizada por robôs fixos, manipuladores e braços robóticos, fugindo dessa forma do foco dos robôs socialmente interativos. Contudo, alguns exemplos destes simuladores são: *MotoSim* ([Yaskawa America, Inc., 2017](#)), *ROBOGUIDE* ([FANUC America Corporation, 2017](#)), *Robologix* ([Logic Desing Inc., 2018](#)), *RobotExpert* ([Siemens Product Lifecycle Management Software Inc., 2017](#)), *RobotStudio* ®([ABB, 2018](#)),

Quadro 1 – Simuladores robóticos e algumas de suas propriedades, ordenados por ano de desenvolvimento

Ano	Simulador	Descrição	Licença	Última Versão	Ano Últ. Ver	3D
1994	SimRobot	Alguns tipos de robôs. Voltado para a pesquisa.	Código aberto	-	2014	Sim
1995	Khepera Simulator	Robótica Móvel, voltado para o robô Khepera. Ambiente simplista.	Freeware	2.0*	1996	Não
1998	Webots	Multi-robôs, utilizado em pesquisas e no ensino.	Proprietário	R2018a	2017	Sim
1999	Stage	Multi-robôs, baixa fidelidade. Simulação em 2D com elementos 3D.	GNU GPL v2	4.3.0	2017	Não
2002	Gazebo	Multi-robôs, vários tipos de robôs. Voltado para ambientes externos.	Apache 2.0	8.0.0	2017	Sim
2002	USARSim	Vários tipos de robôs. Voltado para pesquisas de resgate. Utilizado no RoboCup.	GNU GPL v2	1.2	2017	Sim
2003	OpenHRP	Voltado para humanoides.	Eclipse (EPL)	3.1.4	2012	Sim
2005	Simbad	Voltado para educação e pesquisa. Simplista.	GNU GPL	1.4	2007	Sim
2005	Actin®	Pesquisa, prototipagem, treinamento e ferramenta de vendas.	Proprietário	4.1.6	2015	Sim
2006	MRDS	Vários tipo de robôs. Voltado para hobistas, desenvolvedores acadêmicos e profissionais.	Proprietário	4.0	2012	Sim
2006	OpenRAVE	Vários tipos de robôs. Voltado para validação e teste de algoritmos de planejamento.	GNU Lesser GPL	0.8.2	2012	Sim
2008	Marilou	Vários tipos de robôs. Ambiente de simulação e modelagem.	Comercial	2010	2010	Sim
2009	Klamp't	Vários tipos de robôs. Utilizado em pesquisas, salas de aulas e competições.	BSD	0.7	2017	Sim
2009	Lpzrobots	Vários tipos de robôs, com foco em robôs com juntas e membros. Voltado para a pesquisa e educação.	GPL/ CC BY-NC-SA 2.5	0.7.2	2012	Sim
2010	SigVerse	Multi-agentes, IHR, comp. na nuvem, VR.	Código aberto/ EULA	3.0	2017	Sim
2010	V-REP	Multi-robôs, vários tipos de robôs, controle distribuído.	Proprietário/ GNU GPL	3.4.0	2017	Sim
2010	Morse	Multi-robôs, vários tipos de robôs, IHR, voltado para pesquisa.	BSD	1.4	2016	Sim
2013	DRCSim	Competição DARPA. Utiliza humanoide ATLAS. Baseado no Gazebo.	Apache 2.0	7.0.0	2016	Sim
2014	STDR Simulator	Pesquisa, robótica móvel. Ambiente simplista de fácil configuração.	GNU GPL v3	0.1	2014	Não

Fonte: Dados da pesquisa.

Nota – Dados capturados em Janeiro de 2018.

Nota – A documentação do SimRobot não atribui versões ao simulador.

Nota – Última versão do simulador Khepera está atualmente indisponível para download.

Nota – EULA: acordo de licença de usuário final.

WorkcellSimulator (IT+Robotics, 2018), entre outros.

Paralelamente, também existem na literatura simuladores direcionados para uma aplicação específica. Como exemplo, na área de exames de robôs pode-se citar o *ARGoS* (PINCIROLI *et al.*, 2012), na robótica modular o *Robot3D* (WINKLER *et al.*, 2012), na área de robôs cirúrgicos o *da Vinci Skills Simulator* (Intuitive Surgical, Inc., 2018) e para robôs aquáticos o *UWSim* (PRATS *et al.*, 2012).

Nesta seção, apresentou-se os principais simuladores robóticos relacionados ao tema de pesquisa em ordem cronológica, além de algumas ferramentas para áreas específicas da robótica. Uma outra alternativa utilizada na simulação de robôs (e também no desenvolvimento de simuladores) são as *game engines* (motores de jogos). Como apresentado no Capítulo 1, os motores

de jogos têm grande potencial para oferecer um ambiente fiel para simulações computacionais, sendo possível utilizá-los na pesquisa robótica. Na próxima seção, serão apresentados alguns trabalhos que utilizam tais ferramentas.

2.2 Motores de jogos para simulação

Uma simulação computacional capaz de modelar e renderizar um ambiente com objetos e agentes, muitas vezes utiliza bibliotecas para a simulação física entre objetos e renderização. Por exemplo, o Gazebo utiliza a biblioteca *OGRE* (Ogre3D, 2018) para renderização e as bibliotecas *ODE*, *Bullet*, *DART* e *Simbody*, que modelam o comportamento físico de corpos (Open Source Robotics Foundation, 2014b). Uma outra abordagem para implementação de ferramentas de simulação consiste na utilização de motores de jogos, originalmente, utilizados para o desenvolvimento de jogos. A Unity (Unity Technologies, 2018a) é uma ferramenta para desenvolvimento de jogos, que oferece recursos para modelar ambiente reais com efeitos físicos. Estes recursos requerem uma infra-estrutura básica para edição de ambiente, renderização 3D, detecção de colisão, dinâmica de corpo rígido, animação e elaboração de *scripts*. Muitos desses recursos também são encontrados em simuladores robóticos, como o Webots e V-REP, mencionados anteriormente.

Mattingly *et al.* (2012) apresentaram uma abordagem alternativa para a simulação de ambientes e agentes voltados a área da robótica ao utilizar uma biblioteca desenvolvida na Unity para simulação. Os autores optaram pela Unity devido a sua flexibilidade e facilidade de uso.

A Unity disponibiliza uma vasta documentação em seu site oficial⁶ e uma plataforma para aquisição de pacotes prontos que podem ser incorporados ao projeto, denominada *Asset Store*⁷. Nesta plataforma, os desenvolvedores disponibilizam modelos, sons, texturas, animações e etc.. Muitos desses pacotes são gratuitos e desenvolvidos pela própria Unity. O download da ferramenta é gratuito para iniciantes, estudantes e hobistas (Unity Technologies, 2018b).

A versão 3.0 do SIGVerse (INAMURA; MIZUCHI, 2017) visa dar uma maior imersão para o usuário com os robôs simulados, isto através da utilização de óculos VR. Desta forma, a partir dessa versão o simulador foi remodelado utilizando a Unity, que disponibiliza este recurso. Desta forma, a Unity fica responsável pela imersão do usuário, através do óculos VR, com o avatar humano, dando ao usuário a sensação de estar “dentro” do ambiente de simulação.

Outro exemplo de ferramenta que utiliza um *game engine* é o *Search and Rescue Game Environment* (SARGE). O foco principal do SARGE é prover um videogame voltado para treinar operadores de diversos modelos de robô de resgate (CRAIGHEAD, 2008). Em sua última atualização o SARGE utiliza a Unity, mas inicialmente o projeto utilizou outro *game engine*, denominado Unreal Engine 2 (Epic Games, 2018). Segundo Craighead, Burke e Murphy (2007)

⁶ <unity3d.com/pt/learn/tutorials>

⁷ <www.assetstore.unity3d.com>

esta migração foi necessária devido aos vários *bugs*⁸ presentes e pela falta de documentação da Unreal2, o que dificultava o desenvolvimento e manutenção do SARGE.

O USARSim, apresentado anteriormente, também utilizava o Unreal2. Apesar de ser um simulador de código livre, era necessário na época, que o usuário adquirisse uma licença paga do Unreal2. Nesta mesma época, esta versão do motor de jogo utilizava uma biblioteca simples de física, porém versões mais recentes utilizam bibliotecas de física mais robustas e realistas.

A Unreal Engine é desenvolvido pela Epic Games e atualmente está na versão 4. Ela é uma ferramenta com grande capacidade gráfica, sendo possível modelar cenários com muito realismo, incluindo técnicas avançadas de iluminação dinâmica e um sistema de partículas, permitindo tratar com milhares partículas ao mesmo tempo. A última versão foca em dar suporte a jogos da última geração de consoles existentes, disponibilizando liberações para diversas plataformas incluindo Microsoft Windows, Linux, Mac OS e Android (Epic Games, 2018).

Tanto a Unity quanto a Unreal Engine oferecem ótimos recursos para a simulação de agentes levando em consideração aspectos físicos e gráficos. Os trabalhos citados mostram que é possível utilizar tais ferramentas para a pesquisa científica, em particular no desenvolvimento de simuladores para a robótica.

É importante ressaltar que as técnicas utilizadas para os simuladores robóticos clássicos e as ferramentas de desenvolvimento de jogos não devem ser consideradas ferramentas similares. Os simuladores clássicos tem um ambiente de configuração mais restrito e direcionado ao foco da aplicação, oferecendo suporte para modelos e abordagens de controle similares as presentes em robôs reais. Por outro lado, as ferramentas de desenvolvimento de jogos oferecem a possibilidade de construção de ambientes elaborados, utilização de avatares humanos, definição de sensores ainda não disponibilizados em robôs comerciais e implementação de elementos de interação necessários em robótica social.

2.3 Arquiteturas cognitivas

A partir do momento que robôs passam a interagir diretamente com seres humanos é desejável que a comunicação seja realizada em um nível cognitivo apropriado para atender os objetivos da interação. A ciência cognitiva é um estudo interdisciplinar da mente e da inteligência englobando: filosofia, psicologia, inteligência artificial, neurociência, linguística e antropologia (THAGARD, 2014).

As arquiteturas cognitivas representam uma ferramenta da ciência cognitiva que tem como objetivo criar sistemas computacionais que possam raciocinar sobre diversos domínios de problemas, almejando chegar ao nível da inteligência humana (KOTSERUBA; TSOTSOS, 2017). Com a utilização de técnicas de IA, as arquiteturas cognitivas modelam os mecanismos e

⁸ Erros, falhas ou defeitos de software.

estruturas que fundamentam a cognição humana (LEHMAN; LAIRD; ROSENBLOOM, 2007).

Segundo Laird (2008), uma arquitetura cognitiva consiste em memórias para armazenar conhecimento; unidades de processamento que extraem, selecionam, combinam e armazenam conhecimento e linguagens de representação do conhecimento.

As arquiteturas cognitivas podem ser classificadas em três tipos (KOTSERUBA; TSOTSOS, 2017): conexionistas, simbólicas e híbridas. A hipótese da abordagem conexionista é que abstrações, modelos mentais e comportamentos podem ser mapeadas através de redes interligadas por unidades simples (neurônios), porém esta abordagem não é adequada para realizar deduções e processos de raciocínio. As simbólicas são baseadas em sistemas de produção utilizando regras explícitas para direcionar o comportamento, mas não é capaz de aprender e trabalhar com dados ruidosos. As híbridas mesclam as duas hipóteses com o intuito de suprir suas limitações e combinar seus pontos fortes.

Segundo Taatgen e Anderson (2010), o futuro das arquitetura cognitivas não é da unificação das diversas arquiteturas propostas, mas sim o surgimento de mecanismos e princípios compartilhados que gradualmente tendam a padronizar a área. Em Kotseruba e Tsotsos (2017), é estimado que existam mais de 300 arquiteturas cognitivas na literatura. No entanto, apenas um terço destes projetos estão atualmente ativos. Dentre eles, o *ACT-R*, *Clarion*, *Soar*, *EPIC* e *LIDA* são os mais citados na literatura. Na Figura 10, é apresentada uma linha do tempo com 85 arquiteturas cognitivas, ordenadas pela data de surgimento de cada uma. As datas de término são baseadas nas publicações, atividades na página da web do projeto ou do repositório *on-line*. A Figura 10 também apresenta o tipo da arquitetura. De acordo com esse levantamento observa-se que as arquiteturas simbólicas tiveram uma atenção especial desde meados da década de 1980 até o início dos anos 90. Após o ano 2000, ocorreu um interesse maior na abordagem híbrida. Já as arquiteturas conexionistas, apesar de serem um grupo pequeno, estão distribuídas uniformemente no decorrer de todos esses anos.

No contexto nacional, destaca-se o trabalho de Paraense *et al.* (2016), no qual é proposto um *toolkit*, denominado *Cognitive Systems Toolkit* (CST), para a construção de arquiteturas cognitivas genéricas. Uma derivação deste trabalho é apresentada em Gudwin *et al.* (2018). Neste trabalho, foi apresentada a arquitetura cognitiva *Multi-purpose Enhanced Cognitive Architecture* (MECA), construída a partir do CST e do SOAR. Esta arquitetura implementa processamento baseado em regras e exploração de espaço de estados em módulos, com um sistema motivacional utilizando subsunção dinâmica.

Baseados na atividade do projeto e no suporte oferecido à interação social e às emoções humanas, duas arquiteturas podem ser destacadas: a Soar e a ACT-R.

A arquitetura Soar procura modelar a cognição através da representação e processamento de informações simbólicas, modeladas como um conjunto de regras de produção, enquanto a ACT-R realiza associação de seus módulos com áreas específicas do cérebro humano adotando

Figura 10 – Linha do tempo de arquiteturas cognitivas. As cores indicam o tipo da arquitetura

uma postura da neurociência com ênfase em aprendizado.

Ambas arquiteturas são consideradas abordagens híbridas, pois combinam conceitos e regras simbólicas com elementos sub-simbólicos (conexionista), tais como valores de ativação, aprendizado por reforço, processo de seleção estocástica, etc. (ACT-R Research Group, 2013a; KOTSERUBA; TSOTSOS, 2017).

A arquitetura Soar teve início na segunda metade da década de 80 e se consolidou como uma das mais relevantes arquiteturas cognitivas disponíveis. Com o objetivo de criar um sistema computacional com as mesmas habilidades cognitivas dos seres humanos, a Soar integra o raciocínio baseado em conhecimento, execução reativa, raciocínio hierárquico, planejamento, aprendizado por experiência, aprendizado por reforço, memória semântica (armazena e recupera fatos sobre o mundo), memória episódica (armazena e recupera experiências) e um modelo

baseado na avaliação de emoção, visando criar um sistema computacional genérico, com as mesmas habilidades cognitivas dos seres humanos(LAIRD, 2012).

A arquitetura ACT-R (ACT-R Research Group, 2013b) está organizada numa série de módulos, cada um dos quais processa um tipo diferente de informação que permite associação com regiões do cérebro humano. Os módulos viabilizam o processamento visual e motor, alcançar objetivos e manter conhecimento de longo prazo. Cada módulo possui um *buffer*⁹, que quando considerados em conjunto representam o estado atual do sistema. A partir do estado atual é possível realizar buscas no módulo “memória procedural”, para obter produções que efetivamente implementam as atividades realizadas pela ACT-R. Cada produção possui um custo (em termos de tempo necessário para atingir as metas) e probabilidade de sucesso. Em cada ciclo, a ACT-R determina quais produções combinam com o estado atual do sistema, executando aquela com menor custo. Os resultados alcançados pelo sistema podem ser comparados com a atividade humana através de métricas obtidas da psicologia cognitiva, tais como: o tempo para executar uma atividade, precisão na execução e dados neurológicos para determinar quais áreas do cérebro foram ativadas. Os dados neurológicos são obtidos através da análise de imagem funcional de ressonância magnética.

No presente trabalho, foi escolhida a arquitetura Soar para validação do simulador RHS devido a integração com outros trabalhos e pesquisas do grupo (LAR). No Capítulo 5, serão apresentados experimentos utilizando a Soar para validação do simulador e do ambiente de desenvolvimento no qual este simulador é integrante.

2.4 Considerações finais

Neste Capítulo, uma revisão bibliográfica sobre simuladores robóticos foi apresentada. Essa visão contempla o primeiro objetivo apresentado no início do capítulo: estabelecer o estado da arte em simuladores robóticos. Na análise dos simuladores apresentados, observou-se a maioria deles requer que o usuário interaja diretamente nos sensores e atuadores presentes no robô simulado, o que pode exigir um esforço maior de aprendizado. A revisão bibliográfica revelou também trabalhos que tratam somente marginalmente de aspectos cognitivos necessários à robótica social. Considerando o tratamento dado as informações sensoriais, o sentido de visão é o mais atendido. Além da visão, o nosso objetivo com o simulador RHS é cobrir os demais sentidos tais como, audição, tato, paladar e olfato, sendo estes dois últimos inéditos em simuladores robóticos.

Durante a evolução dessa revisão outra abordagem foi identificada o uso de ferramentas de desenvolvimento de jogos para a simulação. A utilização de uma *game engine* pode simplificar o desenvolvimento, isto devido as bibliotecas de renderização e física, além de outras técnicas e

⁹ Região de memória física utilizada para armazenar temporariamente os dados enquanto eles estão sendo movidos de um lugar para outro.

métodos providos pelo motor de jogo. Por outro lado, a utilização de uma interface aderente a um protocolo formal garante clareza na comunicação entre o simulador e seus clientes. No próximo Capítulo, será apresentado este protocolo juntamente com o ambiente de desenvolvimento do qual o simulador RHS é parte integrante.

AMBIENTE CMDE

O simulador apresentado neste trabalho é parte integrante de um ambiente em desenvolvimento no LAR, denominado *Cognitive Model Development Environment* (CMDE). O CMDE tem como objetivo minimizar o esforço na fase de validação de sistemas que atuem na área de robótica social. O simulador, por sua vez, modela as ações necessárias para validar sistemas cognitivos.

O CMDE é composto basicamente por três módulos: a arquitetura cognitiva, a ontologia *OntSense* e o simulador RHS. A seguir, serão apresentados os três módulos mencionados.

3.1 Descrição do ambiente

O objetivo na elaboração e implementação do CMDE é minimizar o esforço na fase de validação de sistemas que atuem na área de robótica social. Este ambiente deve definir um conjunto de missões e ações exercitados como parte da validação cognitiva.

O ambiente CMDE, apresentado na [Figura 11](#), consiste de três nós de processamento: *Cognition*, *Simulation* e *SPARQL server*. O primeiro nó, RHS, representa o simulador de modelos cognitivos, o segundo nó implementa o “Sistema Cognitivo” ([LANGLEY; LAIRD; ROGERS, 2009](#)) que será testado no ambiente. O terceiro nó, *SPARQL server*, estabelece uma *Resource Description Framework* (RDF) *triple store* ([RDF Working Group, 2014](#)) responsável por manter informações (triplas RDF) relacionadas aos sentidos. A RDF é um modelo leve para tráfego de informações que permite o uso de *Uniform Resource Identifier* (URI) para mapear relacionamentos entre nós presentes na Web, esta estrutura é definida como uma *triple RDF*. Por sua vez, a linguagem SPARQL ([W3C, 2013](#)) fica responsável por padronizar as consultas destes dados, os quais estão armazenados na base *Fuzeki* ([APACHE, 2017](#)).

Naturalmente, após o desenvolvedor concluir os testes utilizando o simulador, o sistema definitivo deve migrar para um robô real com o mínimo de alterações desde que o acesso seja

Figura 11 – CMDE

Fonte: Elaborada pelo autor.

realizado utilizando o mesmo protocolo de comunicação.

Para facilitar o acesso à *triple store* foram implementadas duas APIs (Interface de Programação de Aplicação¹). A primeira API, desenvolvida na linguagem Java utiliza o *framework Apache Jena* para a interação da arquitetura cognitiva com a *triplestore*.

A segunda API, desenvolvida na linguagem C# utiliza o *framework dotNetRDF* (DOT-NETRDF, 2017), para interação do RHS com a *triplestore*. O motor de jogos Unity utiliza a linguagem C# para o desenvolvimento de seus scripts de operação. A disponibilização de uma API nessa linguagem facilita a integração do simulador RHS e evita a utilização de mecanismos para executar a máquina virtual Java, necessária para *framework Apache Jena*, em conjunto com a máquina virtual *Common Language Runtime* (CLR), necessária para execução de programas desenvolvidos na linguagem C#.

Dois elementos presentes no CMDE merecem atenção especial: a ontologia *OntSense* (AZEVEDO; ROMERO, 2016) apresentada a seguir, e o simulador RHS, essência desta dissertação, detalhado na Seção 3.2.

3.1.1 OntSense: uma ontologia para os sentidos

Atualmente, arquiteturas cognitivas aplicadas em robótica interagem diretamente nos sensores e atuadores presentes no robô. Conforme detalhado no Capítulo 1, essa abordagem possui algumas desvantagens que acabam por postergar a obtenção dos resultados. Uma estratégia para abordar essa questão é manter a comunicação entre os nós de processamento robótico e cognitivo aderentes a um padrão com mensagens de alto nível semântico, utilizando conceitos formalmente descritos e relacionados, ou seja, uma ontologia.

¹ Do inglês *Application Programming Interface*.

Essa estratégia cria uma separação clara, mas fácil de ser transposta, entre a arquitetura cognitiva e o sistema robótico, reduzindo a complexidade no desenvolvimento de agentes robóticos voltados para robótica social.

Nesse contexto, a ontologia *OntSense* tem como objetivo modelar as informações sensoriais do ambiente, necessárias para realizar as atividades envolvidas nas tarefas da IHR. As principais características desta ontologia são:

- padronizar a percepção do meio ambiente pela arquitetura cognitiva;
- ser aderente ao modelo de código aberto com a disponibilidade de versões através do repositório GitHub². Esta disponibilidade incentiva a colaboração na evolução da ontologia e APIs associadas.
- facilitar o acesso à enorme quantidade de informações disponíveis na Web, com o uso da tecnologia de web semântica;

Até onde pesquisou-se, não há uma ontologia com essas características na literatura.

Figura 12 – *OntSense* e ontologias superiores

Fonte: Elaborada pelo autor.

Na Figura 12, são mostradas as principais classes da *OntSense*. No bloco superior desta figura, são apresentadas as ontologias superiores usadas como base para a *OntSense*: SUMO

² <github.com/helioaz/ontSense/>

(PEASE, 2016) e IEEE 1872/2015 (IEEE, 2015). No bloco inferior, as classes básicas da *OntSense* são apresentadas.

O conceito *RobotPerceptionEvent* representa a superclasse usada como base para definir todos os sentidos presentes no ambiente. Três associações da classe *RobotPerceptionEvent*, não apresentadas na Figura 12 para manter a clareza, merecem ser relatadas: o instante de captura do evento (*hasCaptureTime*), o gerador de objetos do evento (*generatedBy*) e o *isSenseOf* relacionamento com o agente robótico (*Robot*), responsável por receber e processar a informação.

Quadro 2 – Classes-chave da ontologia *OntSense*

Classe	Descrição	Pos	Obj
<i>RobotPerceptionEvent</i>	Representa a superclasse associada com a percepção do robô sobre o ambiente.	-	-
<i>RobotSmell</i>	Gera informações sensoriais sobre o olfato. Atributo principal: tipo de odor.	X	X
<i>RobotHear</i>	Gera informações sensoriais sobre a audição. Atributo principal: tipo de som.	X	X
<i>RobotTouch</i>	Gera informações sensoriais sobre o tato. Atributos principais: rugosidade, umidade, pressão, temperatura e dureza.	X	X
<i>RobotTaste</i>	Gera informações sensoriais sobre o paladar. Atributos principais: doçura, acidez, salinidade, amargor e umami.		X
<i>RobotVision</i>	Gera informações sensoriais sobre o olfato. Atributo principal: objeto sendo observado.		X

Nota – X indica o tipo de associação.

Na Quadro 2, são detalhadas as classes apresentadas na Figura 12. Um aspecto chave da *OntSense* está associado ao elemento responsável pelo disparo do evento de percepção. Na Quadro 2, as duas últimas colunas são utilizadas para identificar se o evento está associado a uma posição (*Pos*) ou a um objeto (*Obj*).

Dependendo do tipo de sentido, podem ocorrer duas situações. Na primeira, apenas a posição aproximada do evento gerador é identificada, por exemplo, olfato e audição. Na segunda situação, o objeto gerador é conhecido e, conseqüentemente, sua posição exata, por exemplo, os sentidos de visão e paladar. Vale ressaltar que o mesmo sentido pode ter ambas as formas de identificação, como exemplo, considerando o sentido de olfato, é comum a necessidade de identificar o odor de um objeto cujas partículas odoríferas são insuficientes para excitar os sensores olfativos. Neste caso, o objeto deve ser conhecido e pode ser colocado na proximidade de sensores olfativos para reconhecimento de odor.

Cada sentido na ontologia usa conceitos das ontologias superiores, juntamente com as novas classes estabelecidas para representar a informação relacionada a uma determinada percepção. Na Figura 13 é apresentado um exemplo envolvendo a captura de informação

3.2 Simulador robótico RHS

Os simuladores robóticos clássicos não viabilizam a captura de todos os sentidos e de aspectos cognitivos presentes no ambiente. Longe de ser uma deficiência, esse requisito não é atendido simplesmente pelo fato dos simuladores tradicionais serem fidedignos a determinado modelo de robô, considerando o comportamento físico de cada sensor e atuador, bem como, a atualização sincronizada do ambiente de acordo com a evolução da simulação. No contexto do CMDE, o simulador cognitivo RHS possui as seguintes características:

Comandos em alto nível: os comandos fornecidos ao simulador possuem um alto grau de abstração. Uma sintaxe de ações e respostas de alto nível é utilizada para acompanhar as atividades do robô. Exemplos de ações são: falar, pegar, mover, olhar, soltar, etc. Aspectos como comandos para controle de velocidade ou posicionamento físico de atuadores não fazem parte do protocolo utilizado na interface com o simulador.

Informações sensoriais em alto nível: as informações dos cinco sentidos do robô, assim como os comandos, são modelados em alto nível de abstração. Os dados visuais, por exemplo, não precisam ser refinados pela arquitetura cognitiva, isto porque o simulador prioriza o envio de informações sobre objetos e elementos percebidos no ambiente. O mesmo acontece com os outros sentidos.

Protocolo aderente a norma 1872-2015 (IEEE, 2015): a utilização de um padrão consolidado cria uma hierarquia de conceitos de alto nível (robô, sistema robótico, etc.), que garante a compreensão por outros pesquisadores interessados no ambiente.

Ferramentas comerciais de desenvolvimento: utilização de ferramenta comercial para a elaboração de cenários e ações necessários para uma execução consistente do simulador.

Os comandos são fornecidos ao simulador diretamente pela arquitetura cognitiva. Os possíveis comandos que ela pode enviar estão listados no [Quadro 3](#), onde são apresentados os parâmetros que acompanham os comandos e uma breve descrição. Com este conjunto, a arquitetura cognitiva é capaz de excitar o robô simulado a fim de executar tarefas básicas sem se envolver com questões de navegação, configuração de motores e atuadores e movimentação física para a execução destas tarefas. Estas funções se resumem na interação com objetos e elementos do ambiente (pegar objetos, abrir portas, etc), movimentação (girar, movimentar, focar o olhar, etc.) e comunicação (falar).

A comunicação entre arquitetura e o simulador é realizada via *socket*, viabilizando a presença de mais do que um nó de processamento. Como os comandos são dados estruturados, utiliza-se *Protocol Buffers* ([Google Developers, 2017](#))(conhecido também como *protobuf*) para a serialização dos dados entre os dois nós de processamento. Isto simplifica o envio e recepção de dados pelo *socket*. Cada comando enviado pela arquitetura cognitiva é analisado e processado

Quadro 3 – Comandos que a arquitetura cognitiva pode enviar ao RHS em sua versão atual

Comando	Parâmetros	Descrição
<i>ActivateLeft</i>	ID	Liga (abre) determinado elemento do ambiente com a mão esquerda.
<i>ActivateRight</i>	ID	Liga (abre) determinado elemento do ambiente com a mão direita.
<i>CancelCommands</i>	Sem parâmetros.	Cancela todos os comandos enviados com conclusão pendente.
<i>DeactivateLeft</i>	ID	Desliga (fecha) determinado elemento do ambiente com a mão esquerda.
<i>DeactivateRight</i>	ID	Desliga (fecha) determinado elemento do ambiente com a mão direita.
<i>HeadReset</i>	Sem parâmetros	Reinicia o foco da visão do robô.
<i>LeaveLeft</i>	ID ou Posição	Deixa objeto da mão esquerda em determinada posição.
<i>LeaveRight</i>	ID ou Posição	Deixa objeto da mão direita em determinada posição.
<i>LookAt</i>	ID ou Posição	Direciona o foco da visão do robô para determinada posição ou objeto.
<i>LookFor</i>	Texto	Procura, através da visão, elemento no ambiente dado parte de seu nome.
<i>Move</i>	ID ou Posição	Movimenta o robô até uma posição ou objeto.
<i>Rotate</i>	Ângulo (em graus)	Rotaciona o robô dado um ângulo.
<i>SmellLeft</i>	Sem Parâmetros	Sente o cheiro do objeto da mão esquerda.
<i>SmellRight</i>	Sem Parâmetros	Sente o cheiro do objeto da mão direita.
<i>Speech</i>	Texto	Envia uma mensagem que o robô deve transmitir através da fala.
<i>TakeLeft</i>	ID	Pega objeto com a mão esquerda.
<i>TakeRight</i>	ID	Pega objeto com a mão direita.
<i>TasteLeft</i>	Sem Parâmetros	Prova o sabor do objeto contido na mão esquerda.
<i>TasteRight</i>	Sem Parâmetros	Prova o sabor do objeto contido na mão direita.
<i>Turn</i>	ID ou Posição	Rotaciona o robô para que ele encare determinado objeto ou posição.

pelo simulador. Ao final da execução, uma resposta é gerada e enviada pelo simulador atualizando a arquitetura sobre o estado de execução dos comandos recebidos.

Cada comando enviado possui uma identificação única, que é utilizada na composição do comando de resposta. A resposta então segue o formato $\langle ID, status \rangle$, sendo o primeiro parâmetro o identificador e o *status*, o estado final do comando com aquele ID. Somente dois tipos de *status* são enviados para arquitetura: *success* e *fail*. O primeiro indica se determinado comando foi executado com sucesso, já o segundo, indica se não foi possível executá-lo. Neste último caso, a arquitetura cognitiva receberá as informações sensoriais do robô (as quais são atualizadas periodicamente), possibilitando que ela reflita sobre o estado dos elementos do ambiente e assim adapte os comandos para contornar o problema. No próximo capítulo, estas questões serão abordadas com maiores detalhes, bem como a estrutura geral do simulador RHS.

3.3 Considerações finais

Neste capítulo, foi apresentada a interação do RHS com o ambiente CMDE e com a ontologia *OntSense*. Esses elementos visam minimizar esforços no desenvolvimento de agentes robóticos voltados para robótica social.

No próximo capítulo, serão descritas as ferramentas, métodos e estratégias utilizadas no desenvolvimento do simulador RHS. Além disto, serão apresentadas as estruturas, ambientes, objetos, agentes e tarefas implementadas na ferramenta, bem como a organização do sistema, os comandos internos e os sentidos associados ao robô.

SIMULADOR RHS

Neste capítulo, é apresentada a descrição dos procedimentos adotados para o desenvolvimento do projeto. Considerando a revisão bibliográfica apresentada, quando consideramos sistemas cognitivos, a ausência de simuladores apropriados torna-se evidente. Este projeto minimiza esse problema contribuindo no desenvolvimento do ambiente CMDE ([Figura 11](#)), em particular na construção do nó de processamento do simulador RHS.

A implementação do simulador utiliza a ferramenta de desenvolvimento Unity, a qual foi apresentada na [Seção 2.2](#). Levando em consideração que um dos requisitos dos simuladores robóticos é modelar ambientes de forma realística ([MATTINGLY et al., 2012](#)), as estruturas disponibilizadas pela Unity se adequam perfeitamente ao problema proposto, principalmente pelo fato da nossa abordagem não ter objetivo de interagir diretamente sensores e atuadores.

Apesar de utilizarmos uma ferramenta para desenvolvimento de jogos, o objetivo não foi desenvolver um sistema que se encaixe nesta categoria. Existem diversos trabalhos científicos que propõe *serious game* (jogo sério, em português) como uma ferramenta de incentivo. As aplicações nesse contexto envolvem: ensino ([WOUTERS; OOSTENDORP, 2017](#)), treinamento ([GAMITO et al., 2017](#)), terapia ([BONNECHERE et al., 2017](#)), área da saúde ([FLEMING et al., 2017](#)), entre outras áreas.

Na [Figura 14](#) é apresentada uma visão geral do simulador RHS, ressaltando os principais módulos do simulador e a comunicação com o sistema cognitivo e usuário. O módulo “Sistema Cognitivo” envolve tanto o sistema propriamente dito quanto a base de triplas apresentada no [Capítulo 3](#).

A comunicação entre sistema cognitivo e o simulador é dada via *socket*, onde o núcleo do simulador (*Simulator Manager*, a ser visto em [Subseção 4.2.1](#)) gerencia a maioria dos recursos do simulador, incluindo o processamento dos comandos recebidos via *socket* quanto os dados sensoriais do robô a serem enviados à arquitetura. Este módulo repassa os comandos ao robô humanoide, que irá atuar no ambiente de acordo com as requisições do usuário.

Figura 14 – Visão Geral do simulador RHS

Fonte: Elaborada pelo autor.

A interação do usuário com o simulador é dada via mouse, teclado e interface gráfica (Interface Humano-Computador), onde ele pode controlar diretamente o avatar humano, inclusive interagir com o ambiente e realizar pedidos ao robô. O usuário não é capaz de controlar o robô diretamente, para isto ele deve interagir com o robô (comunicação verbal, por exemplo) para influenciar as ações do agente.

Nas próximas seções serão apresentadas a organização, os relacionamentos e componentes do sistema, que inclui os seguintes elementos: ambiente, objetos, o avatar humano e o robô. Na sequência, serão apresentados os comandos envolvidos na simulação, seus estados de execução e também os sentidos modelados no robô virtual. Por fim, questões de distribuição do software serão tratadas.

4.1 Organização e componentes do sistema

O simulador é formado por diversos componentes responsáveis por agrupar e construir elementos físicos (objetos, móveis, paredes, etc), agentes (robô e avatar humano), elementos gráficos (interface gráfica), módulos de gerenciamento e outros aspectos de configuração e simulação do ambiente.

Na Unity, cada entidade tem como base a classe *GameObject*. A utilização de *scripts*, materiais, texturas e animações depende de uma ou mais entidades *GameObject* para existirem. Cada objeto desta classe possui um componente do tipo *Transform* que armazena sua posição, rotação e escala. A Unity agrupa os objetos em uma estrutura hierárquica que permite que um *GameObject* tenha diversos filhos, cada um com uma configuração de posição, rotação e escala independente de seu pai, mas que utiliza a posição deste como referência. Por exemplo, uma mão possui cinco dedos (filhos) que se movimentam livremente, porém presos ao pai (mão). Caso a mão gire, a rotação local dos dedos permanece a mesma em relação ao pai, porém sua rotação global muda em relação ao mundo.

Figura 15 – Hierarquia dos principais componentes do cenário de atuação, divididos em duas categorias, componentes concretos (verde) e abstratos (azul)

Fonte: Elaborada pelo autor.

A hierarquia dos principais componentes modelados no simulador é apresentada na Figura 15, onde estes podem ser concretos ou abstratos. Os componentes concretos dependem somente de seus atributos internos (*scripts*, animações, texturas, etc) para existir ou formar algo que seja significativo no ambiente, já os abstratos são constituídos de um ou mais elementos concretos ou abstratos. Por exemplo, uma sala de estar (componente abstrato) depende das paredes e móveis (conjuntos que agrupam componentes concretos) para existir. Um copo, uma mesa, uma xícara ou uma parede são exemplos de componentes concretos.

Na nossa representação hierárquica, um elemento que possui uma entidade filha também

pode ser considerada concreta, desde que ela possua suas próprias propriedades para existir “fisicamente” ou executar alguma função no ambiente simulado. Na Figura 15 o componente concreto *Head*, o qual representa a cabeça do robô, possui uma câmera (componente concreto) como filha. Porém a cabeça do robô não necessita da câmera para existir, sendo que este elemento é somente um componente adicionado devido à necessidade da câmera se mover junto com a cabeça do agente.

Na hierarquia apresentada, existem dois componente que são responsáveis por aspectos operacionais e de gerência do simulador. O componente *Simulator Manager* tem como tarefa a gerência do robô e dos IDs dos objetos, além de configurar a comunicação via *Socket* e *API OntSense* com a arquitetura cognitiva. O outro componente é o *Canvas*, que por padrão da Unity, agrupa elementos de interface gráfica e permite a interação do usuário com o simulador. Ambos os módulos serão abordados com mais profundidade subsequentemente.

Figura 16 – Sala de Estar

Fonte: Elaborada pelo autor.

O ambiente utilizado pela simulação é composto por dois cômodos, sala e cozinha, apresentados na Figura 16 e Figura 17. Esses elementos são representados na hierarquia dentro do componente *Rooms* como *Living Room* e *Kitchen*, respectivamente. Cada um destes cômodos são formados por um conjunto de elementos abstratos que por sua vez são formado por elementos concretos, os quais estão implícitos na Figura 15. Tanto os ambientes quanto os elementos que os compõe são parte do pacote *Complete Home Interior Pack* (NEKOBOLT, 2015), disponível na *Asset Store* da Unity. O módulo *Directional Light*, apresentado na hierarquia, é responsável pela luz natural e representa o sol, sendo possível simular períodos do dia em versões futuras.

A entidade *Objects* agrupa diversos objetos que tanto o robô quanto o avatar humano podem manipular e transportar. Como forma de simular as propriedades perceptíveis através dos sentidos do robô, tais como cor, rugosidade, odor, sabor, etc., foram utilizados *scripts* que

Figura 17 – Cozinha

Fonte: Elaborada pelo autor.

armazenam estas propriedades, particulares de cada objeto. Estes possuem propriedades táteis e visuais, porém não as demais. Por exemplo, um copo no simulador possui propriedades de tato e visão, porém não emite som, não tem sabor e nem odor, desta forma ele não é configurado com as respectivas propriedades. Outro exemplo, é a torta de cereja, ela possui propriedades de visão, tato, paladar e olfato, porém não emite som. Na [Seção 4.4](#) serão apresentados cada um dos sentidos e as respectivas propriedades que podem ser abstraídas de dado objeto.

Além dos objetos, outros elementos do ambiente merecem destaque por suas propriedades. No [Quadro 4](#) são apresentados alguns objetos que possuem propriedades que vão além de tato e visão, tais como odor, sabor, emissão sonora ou possuem estados.

Os estados dizem respeito a modos que alguns objetos podem assumir de acordo com sua finalidade, e algumas destas finalidades exigem que o objeto altere seu estado. Por exemplo, uma porta pode estar fechada ou aberta, uma lâmpada pode estar acesa ou apagada, um copo pode estar cheio ou vazio. Atualmente no simulador, somente as portas, gavetas e lâmpadas estão configuradas para terem seus estados alterados. Uma porta pode ter como estado *Opened* (aberta) ou *Closed* (Fechada), bem como as gavetas. Já as lâmpadas estão associadas a interruptores que podem ter como estados *On* (ligado) e *Off* (desligado).

Outra propriedade que pode ser atribuída aos elementos do ambiente simulado é a URI. Uma das motivações deste trabalho é o envolvimento com ontologias para facilitar o acesso às informações adicionais de objetos através da web semântica. Por exemplo o objeto “caneca”(mug), tem como URI: www.wikidata.org/wiki/Q386215. Nesta URI, pode-se ter informações que o objeto em questão é um tipo de copo, geralmente fabricado de cerâmica com formato cilíndrico. Esse relacionamento oferece um valioso conjunto de informações que podem ser utilizadas pela arquitetura cognitiva.

Quadro 4 – Exemplos de objetos e elementos do ambiente simulado

Objeto	Propriedades em Destaque	Localização
Bolinhos	Paladar.	Sem local fixo.
Torta (Pie)	Paladar e olfato, com emissão de partículas de cheiro.	Sem local fixo.
Carne (<i>Meat Pack</i>)	Paladar e olfato, com emissão de partículas de cheiro.	Sem local fixo.
Salmão (<i>Salmon Pack</i>)	Paladar e olfato, com emissão de partículas de cheiro.	Sem local fixo.
Porta (<i>Door</i>)	Pode ser aberta ou fechada.	Entre a cozinha e a sala.
Gavetas	Podem ser abertas ou fechadas.	Balcão da cozinha
Portas do armário	Podem ser abertas ou fechadas.	Armário da cozinha.
Torneira (<i>Tap</i>)	Audição, através da água saindo. Pode ser aberta ou fechada.	Pia da cozinha.
Interruptor (<i>Switch 01</i>)	Liga ou desliga todas as luzes da cozinha.	Cozinha.
Interruptor (<i>Switch 02</i>)	Liga ou desliga todas as luzes da sala de estar.	Sala.
Geladeira (<i>Fridge</i>)	Portas deste objeto podem ser abertas ou fechadas.	Cozinha.

Outros dois componentes presentes simulador são o robô e o avatar humano. O robô foi instanciado a partir do modelo disponibilizado pela [Unity Technologies \(2012\)](#) gratuitamente. O avatar humano foi modelado utilizando o Adobe Fuse ([Adobe, 2018](#)) e configurado através do Adobe Mixamo ([ADOBE, 2018](#)). Na [Figura 18](#) são apresentadas a aparência do agente humano e do robótico.

Como ambos os agentes são humanoides, suas configurações são semelhantes, ou seja, ambos são dotados de esqueletos e juntas (implícitos em *Body Parts* na hierarquia de componentes) que permitem a utilização de animações, as quais oferecem maior naturalidade na movimentação. Cada agente possui duas câmeras, uma acoplada a suas cabeças na altura dos olhos e outra que “acompanha” os agentes a uma certa distância. A primeira permite o usuário visualizar o que o agente está “vendo” e a segunda permite visualização mais ampla dele inserido no ambiente e das ações sendo executadas.

Além dos dois agentes, um terceiro componente facilita a interação com o ambiente, o módulo *God Mode*. Ele permite o usuário manipular objetos livremente durante a simulação, trocando-os de posição e alterando seus estados. O usuário tem o poder de movimentar a câmera pelo ambiente sem colidir com outras estruturas, isto por não possuir um “corpo físico”. Esta característica faz com que as operações físicas inerentes ao simulador não atuem sobre *God Mode*, tal como a gravidade por exemplo. A seguir, os dois agentes e o módulo *God Mode* são apresentados com mais detalhes.

Figura 18 – Avatar humano e agente robótico

Fonte: Elaborada pelo autor.

4.1.1 Robô

O robô é controlado pela arquitetura cognitiva via componente *Simulator Manager*, recebendo e executando comandos. Para tornar isto possível, o robô foi dotado de diversas ações e operações corporais simples, que em conjunto, permitem a realização de tarefas mais elaboradas. As movimentações e interações corporais do robô são dadas através de três elementos principais: animações, *Inverse Kinematics* (IK) e *scripts*.

As animações, com o apoio da classe *NavMesh*, é utilizado para suprir movimentos de locomoção e rotação do robô. A *NavMesh* é uma classe de IA da Unity que oferece suporte para testes de viabilidade de trajetos e localização, calcula custos para caminhos específicos de áreas acessíveis e oferece ajustes no comportamento do agente a fim de buscar caminhos e desviar de obstáculos.

As animações são utilizadas para dar naturalidade na movimentação do robô pelo ambiente. Seria possível utilizar animações para todas as ações do robô pelo ambiente, porém para cada ação seria necessária uma animação diferente, pois elas não oferecem flexibilidade de acordo com a situação e tarefa a ser executada. Por exemplo, suponha que o robô disponha de somente uma animação para pegar um objeto, no momento em que for necessário pegar o objeto em uma posição pouco favorável a captura seria feita de forma não natural resultando em uma transferência brusca do objeto para a mão do robô. A alternativa utilizada para estes casos de interação foi a utilização da técnica IK.

A técnica IK tem como objetivo movimentar determinadas articulações para que um membro alcance uma dada posição, como por exemplo, configurar as posições e rotações do ombro, braço e antebraço para que a mão alcance um objeto. Como a estrutura utilizada na Unity

é hierárquica, um nó filho se movimenta automaticamente com seu pai, mas não o contrário. Assim, a Unity disponibiliza métodos que realizam operações para a movimentação de mãos, pés e cabeça, através dos braços pernas e pescoço, respectivamente.

No simulador, a IK é utilizada nas interações que envolvem interações manuais e da cabeça. A utilização da técnica deixa as interações do robô com os elementos do ambiente muito mais naturais e dinâmicas. Apesar destas vantagens, o suporte da IK se limita em movimentar os membros do humanoide, não sendo possível agachar ou curvar sua espinha para alcançar determinado posição. Desta forma, não seria possível interagir com objetos no chão ou em cima de mesas, por exemplo. Esta limitação não é desejável, pois assim o ambiente deveria ser estruturado, ou pelo menos semi-estruturado, para viabilizar a atuação do robô.

A espinha do robô é um nó da hierarquia (implícito em *Body Parts* da [Figura 15](#)) como qualquer outro, desta forma é possível manipulá-lo através de *scripts*. O desejável é que o robô incline o suficiente para que alcance o objeto. A utilização de funções trigonométricas, com o objetivo de determinar o ângulo ideal de inclinação da espinha do robô, resolve o problema perfeitamente. Através desta abordagem, é possível rotacionar a espinha nos três eixos, permitindo que o robô incline em direção ao objeto, tanto frontal quanto lateralmente. A ação de agachar também foi resolvida com ajuda de funções trigonométricas, porém com auxílio da IK.

Objetos na Unity utilizam de componentes denominados *Colliders*, os quais permitem corpos colidirem e não transpassarem um ao outro, como exemplo, impede que o robô atravessasse o piso por ação da gravidade. O *collider* está ancorado junto aos pés do robô, ou seja, o robô é sustentado por eles. A IK é capaz de configurar a rotação das juntas das pernas automaticamente a fim de simular o movimento de agachar, porém, primeiro é necessário encontrar a posição em que os pés devem ficar em relação ao quadril, isto é feito a partir da trigonometria, assim como foi feito na espinha. Desta forma, as pernas são flexionadas e o *collider* se move em conjunto, dando a impressão de que o robô está agachando. Tanto a inclinação da espinha quanto a ação de agachar, são intrínsecos aos comandos de interações com objetos, não sendo necessário a arquitetura cognitiva detalhar essas composições antes de interagir com um objeto.

Como dito previamente, o robô é controlado pela arquitetura cognitiva que deve atuar considerando as informações de percepção recebidas através do servidor e *SPARQL server* ([Seção 3.1](#)). O avatar humano representa uma importante fonte de informações sensoriais, ou seja, parte do comportamento do robô é resultado dos comandos dados pelo usuário do simulador atuando sobre o avatar humano. Para ampliar a perspectiva do usuário, o simulador disponibiliza duas câmeras acopladas ao robô, além de apresentar uma lista de objetos capturados pelos canais sensoriais.

A [Figura 19](#) mostra a tela de interface apresentada ao usuário do simulador. A tela principal apresenta a câmera que acompanha o robô e, no canto superior esquerdo, a câmera que representa sua visão. Logo abaixo, está disposto um painel com cinco abas, cada uma listando os elementos percebidos por determinado sentido. Abaixo desse painel está localizado um *info*bar

Figura 19 – Interface do usuário no modo robô

Fonte: Elaborada pelo autor.

responsável por apresentar possíveis frases ditas pelo robô.

Os comandos e ações mencionadas nesta seção são detalhados na [Seção 4.3](#). Os detalhes associados as informações sensoriais associadas a percepção do ambiente são apresentados na [Seção 4.4](#).

4.1.2 Avatar humano

No simulador, o avatar é representado por uma mulher adulta, nomeada como “Mariana”. O objetivo desta personagem no simulador é a interação direta com o robô. De forma distinta do robô, o avatar humano é controlado diretamente pelo usuário do simulador, isto é, os comandos são fornecidos pelas interfaces de entrada.

As interfaces de entrada utilizadas são o teclado e o mouse. Através do teclado, o usuário consegue realizar a movimentação do agente humano pelo ambiente e também mover seu foco de visão. As animações de movimento são ativadas através do acionamento das teclas, onde o usuário tem a liberdade de movimentar o avatar pelo ambiente sem definir um ponto específico no plano. Consequentemente, esse cenário dispensa a utilização do técnicas de navegação inteligente (*NavMesh*).

Com o mouse, o usuário é capaz de mover a câmera que segue o avatar, sendo possível aplicar zoom e reposicioná-la. Além desta interação, o usuário pode enviar comandos para o avatar humano, de forma semelhante aos comandos enviados pela arquitetura cognitiva para o robô. A gama de comandos para o avatar é menor que para o robô, isto pelo fato do usuário poder controlá-lo diretamente através do teclado. Estes comandos disponibilizados são: pegar e largar objetos, ativar (abrir) e desativar (fechar) elementos do ambiente e falar.

A interface responsável pelo envio de comandos, apresenta uma lista de objetos passíveis de manipulação. Com o objetivo de limitar esta lista, somente elementos que visíveis pelo avatar humano são apresentados ao usuário. Desta forma, considerando nossa abordagem sobre os sentidos, a visão é o único sensor que o avatar humano possui que é capaz de capturar informações do ambiente. Porém, através da tela do computador e das saídas de áudio, o usuário tem acesso aos elementos visuais e auditivos da simulação. Foi implantado no simulador uma funcionalidade em que as partículas de cheiro se tornem visíveis ao usuário, o que permite que este tenha uma compreensão melhor do comportamento desses elementos no ambiente.

O avatar humano ainda possui a representação de emoções, dadas através de um *script*. São utilizadas as seis emoções básicas definidas por Ekman e Friesen (1971), *Happiness* (felicidade), *Sadness* (tristeza), *Anger* (raiva), *Surprise* (surpresa), *Disgust* (desgosto, nojo) e *Fear* (medo).

Em 1994, Ekman e Davidson (1994) definem mais onze emoções, *Amusement* (diversão), *Contempt* (desprezo), *Contentment* (contentamento, satisfação), *Embarrassment* (embaraço, constrangimento), *Excitement* (excitação), *Guilt* (culpa), *Pride in achievement* (orgulho na conquista de algo), *Relief* (alívio), *Satisfaction* (satisfação), *Sensory Pleasure* (prazer sensorial) e *Shame* (vergonha). Porém os autores afirmam que nem todas as emoções deste segundo grupo podem ser codificadas através de expressões faciais, o que dificulta a detecção delas por sistemas computacionais. A utilização das seis emoções básicas foi então adotada com o objetivo de simplificar o sistema, sem perda de conteúdo. Uma sétima classificação foi acrescentada, a classe *Neutral*, que é atribuída ao avatar caso não esteja emitindo emoção alguma.

Atualmente, o avatar humano não reflete sua emoção através de expressões faciais e nem movimenta a boca ao emitir sons. Em relação à emissão de sons, a simulação da fala é realizada através de um áudio genérico, que carrega a *string* com a mensagem a ser capturada pelo sentido de audição do robô. As questões de detecção e reconhecimento também serão abordadas na Seção 4.4.

A Figura 20 apresenta o modo avatar, onde o usuário tem acesso ao agente e suas câmeras. Note que diferente do modo robô, na Figura 19, o usuário tem acesso a um painel, onde é possível enviar comandos ao avatar humano e configurar sua emoção.

4.1.3 God Mode

Na etapa de desenvolvimento, o editor Unity permite que o programador faça modificações e configure objetos presentes na cena. Contudo, na versão compilada, essa liberdade de configuração não está mais acessível uma vez que o executável é autônomo e independente da ferramenta Unity. Desta forma, o modo *God Mode* foi implementado com o intuito de dar ao usuário controle sobre os elementos do ambiente durante a simulação.

O *God Mode* possui basicamente duas funcionalidades: movimentar a câmera livremente

Figura 20 – Interface do usuário no modo de controle do avatar humano

Fonte: Elaborada pelo autor.

e manipular objetos. Na primeira, a movimentação da câmera é dada através da interação com os recursos do mouse. Com o botão esquerdo a câmera pode ser arrastada, com o direito ela é rotacionada na direção do movimento do mouse e com a rolagem do *scroll* é possível movimentar a câmera para frente e para trás.

A manipulação do objeto diz respeito às operações de reposicionamento e alteração de seu estado. Além disso, é possível obter informações das propriedades do objeto. Essa operação inicia com a seleção de um objeto do ambiente através do acionamento do botão esquerdo do mouse com o cursor sobre ponto desejado. Ao fazer isto, o elemento é realçado e suas propriedades são apresentadas em um painel no canto inferior direito da interface.

O reposicionamento do objeto pode ser dado de duas formas: a primeira por meio da inserção das coordenadas x , y e z e a segunda por meio da seleção da posição desejada diretamente no ambiente. Quando esta segunda opção é adotada, um cursor vermelho é exibido habilitando o usuário a demarcar a nova posição para o objeto.

Na [Figura 21](#) é apresentado este modo, onde o copo foi selecionado e foi realçado (através do aumento de sua luminosidade). No painel do canto inferior direito, são apresentadas as informações referentes ao copo, além dos campos de reposicionamento do objeto. Na figura ainda é possível visualizar a demarcação (cor vermelha) que auxilia no reposicionamento do objeto selecionado.

No [Quadro 4](#) foram apresentados alguns objetos que podem ter seus estados alterados, por exemplo, uma porta pode abrir e fechar, um interruptor pode acender ou apagar um conjunto de lâmpadas, entre outros. O usuário através de operações sobre o avatar humano pode alterar o estado dos objetos perfeitamente, porém esta forma exige que o avatar humano seja conduzido

Figura 21 – Interface do usuário no *God Mode*

Fonte: Elaborada pelo autor.

até o objeto para executar a ação. O modo *God Mode* oferece uma alternativa mais versátil para o reposicionamento e mudança de estados dos objetos.

Observe que durante uma sessão de simulação o modo *God Mode* pode ser utilizado para trocar um objeto de local, fechar uma porta ou abrir uma torneira. Essas ações inserem uma imprevisibilidade no ambiente que pode ser utilizada para validar a robustez de aspectos cognitivos implementados através da arquitetura cognitiva.

4.2 Comunicação com o usuário e com a arquitetura cognitiva

Na seção anterior apresentou-se os componentes do simulador RHS. Nesta seção, o objetivo é detalhar como estes componentes interagem, focando na comunicação estabelecida com a arquitetura cognitiva e na interação do usuário (através da interface) com o avatar humano. Na hierarquia apresentada na [Figura 15 \(Seção 4.1\)](#), existem dois elementos principais responsáveis pela gerência do simulador, o *Simulator Manager* e o *Canvas*. Esses dois componentes são detalhados nos próximos parágrafos.

4.2.1 Simulator Manager

A principal funcionalidade do *Simulator Manager* é prover recursos para a comunicação da arquitetura cognitiva com o robô simulado através da recepção de comandos e envio de informações da percepção do ambiente.

Figura 22 – Hierarquia de controle do *Simulator Manager*

Fonte: Elaborada pelo autor.

Na Figura 22 é apresentada a hierarquia do *Simulator Manager*, que tem quatro módulos sob o seu controle: *Robot*, *Unique ID Distributor*, *Socket* e *API OntSense*. O primeiro módulo diz respeito ao robô. Ele contém diversos *scripts* que permitem a execução de comandos (*Agent Interaction*, *Movement*, *Speech* e *Head Movement*) e percepção do ambiente através dos sentidos (*Vision*, *Hearing*, *Touch*, *Taste* e *Smell Manager*). Na hierarquia de controle (Figura 22), cada módulo utiliza elementos adicionais para o processamento de tarefas. Por exemplo, os sentidos utilizam os sensores e as ações (execução de comandos) utilizam membros do corpo, componentes da Unity (*NavMesh*, *Animator*), etc.

O segundo módulo é o *Unique ID Distributor*. Cada elemento no ambiente possui um ID (identificador) único associado gerado através do módulo em questão. O *Unique ID Distributor* também por armazenar e disponibiliza o identificador do objeto quando solicitado. Atualmente o ID é gerado utilizando o número da instancia do elemento no ambiente Unity, esse número é gerado pela Unity no início da simulação e permanece o mesmo até o final da mesma.

Através do módulo *Socket* o simulador realiza a comunicação com a arquitetura cognitiva utilizando o protocolo TCP/IP com o apoio da biblioteca de serialização *protobuf*. Esse nó de comunicação é responsável pelo recebimento de comandos (*Receiver*) e envio de respostas (*Sender*) sobre o estado de execução destes comandos.

Finalmente, no módulo *API OntSense*, o simulador utiliza um servidor SPARQL para atualizar a *triple store* com as informações de percepção do ambiente, conforme apresentado no [Capítulo 3](#).

De forma geral, o *Simulator Manager* recebe informações de sentidos do robô e as envia até uma base de dados via *API OntSense*. Ao mesmo tempo, recebe comandos da arquitetura via *socket*, processa e encaminha para execução pelo robô simulado. Ao final da execução do comando, o *Simulator Manager* envia uma resposta informando sucesso ou falha.

4.2.2 Canvas

Na Unity, o módulo *Canvas* tem como principal funcionalidade agrupar os elementos de interface gráfica. Como o usuário interage com o sistema através de elementos desta interface (tais como botões, caixas de texto, etc) o *Canvas* é responsável pelo controle do avatar humano e do modo editor. A hierarquia de controle do *Canvas* é apresentada na [Figura 23](#). Nela estão presentes três módulos: *User Canvas Manager*, *God Mode* e *Canvas Manager*.

O módulo *User Canvas Manager* tem como objetivo apresentar ao usuário um painel para a construção de comandos a serem dados ao avatar humano. É possível também definir a emoção do humano através desta interface. Como apresentado na [Subseção 4.1.2](#), o avatar humano tem propriedades em comum com o robô, porém com menor quantidade de funcionalidades, como pode ser verificado na [Figura 23](#).

Assim como com o avatar, a interação com o *God Mode* é dada através do mouse e da interface gráfica. As operações envolvidas e as funcionalidades disponíveis neste modo foram apresentadas em detalhes na [Subseção 4.1.3](#).

O *Canvas Manager* possibilita pausar, reiniciar ou alterar velocidade de execução da simulação. O componente é responsável por apresentar as imagens geradas pela visão (câmeras) do robô do avatar humano e também por apresentar as informações dos sentidos do robô, sendo possível desativar sua visualização. O *Canvas Manager* tem como função apresentar o último comando que o robô recebeu, junto com seu estado de execução. Esta informação é apresentada em uma barra que tem sua cor alterada de acordo com o estado do comando, azul para “executando”, verde para “sucesso” e vermelho para “falha”.

4.3 Comandos internos

O simulador RHS trabalha com comandos que atuam no robô simulado, determinado sua interação com o ambiente. Na [Seção 3.2](#), foram apresentados os comandos disponibilizados para a arquitetura cognitiva, definidos como externos. Para simplificar sua utilização por parte da arquitetura cognitiva, os comandos externos possuem um alto nível de abstração com um número reduzido de parâmetros.

Figura 23 – Hierarquia de controle do *Canvas*

Fonte: Elaborada pelo autor.

O mapeamento dos comandos externos realizado pelo simulador define os “comandos internos”. Esses comandos possuem uma sintaxe mais elaborada com a inclusão de novos parâmetros, sendo mais próxima do ambiente de desenvolvimento do simulador. De forma resumida, os dois tipos se diferem em poucos aspectos, sendo que os internos priorizam uma padronização da implementação do simulador e os externos facilitam a comunicação com a arquitetura cognitiva.

Os comandos internos são agrupados em cinco categorias, “Interação”, “Movimentação”, “Foco de Visão”, “Comunicação” e “Operacional”. A categoria “Interação” agrupa comandos

que tem como objetivo a interação manual com objetos. Os comandos deste grupo são: “*Take*”, “*Leave*”, “*Activate*”, “*Deactivate*”, “*Taste*” e “*Smell*”.

O comando “*Take*” refere-se a captura de objetos, onde é possível selecionar a mão de interação. A execução do comando consiste em direcionar a mão até o objeto, sendo possível agachar e inclinar caso necessário, agarrar o objeto e posicionar a mão com o objeto próximo ao tronco. O comando “*Release*” realiza a operação inversa mas, segue o mesmo princípio.

O comando “*Activate*” é responsável por ligar ou abrir objetos, por exemplo, interruptores ou portas. Sua operação é semelhante ao comando “*Take*”, porém ao invés de agarrar um objeto, o contato com a mão altera seu estado. O comando “*Deactivate*” realiza a operação inversa, sendo utilizado para desligar ou fechar um objeto no ambiente.

Os últimos comandos da categoria “Interação” são “*Taste*” e “*Smell*”. Esses comandos são utilizados, respectivamente, para que o robô prove o sabor e determine o odor de um objeto. Em cada um dos comandos da “Interação” é possível configurar qual mão deve executar a tarefa, essa liberdade que possibilita que o robô interaja com o ambiente mesmo que uma das mãos esteja ocupada.

Os comandos da categoria “Movimentação” são: “*Move*”, “*Turn*” e “*Rotate*”. Em “*Move*”, o robô recebe uma posição ou objeto e se locomove até lá. “*Turn*” é um comando para que o robô gire em determinada direção, usando um objeto ou local como referência. “*Rotate*” é correlato ao comando anterior, mas ao invés de uma referência ele recebe um ângulo de rotação.

A movimentação realizada ao receber o comando “*Move*” envolve a classe *NavMesh* da Unity, apresentada na [Subseção 4.1.1](#). No simulador RHS esta técnica foi utilizada justamente para realizar a movimentação do robô no ambiente levando em consideração possíveis obstáculos.

No caso do caminho estar obstruído por uma porta fechada, por exemplo, o robô vai até a ela mas, o algoritmo *NavMesh* interrompe o processo de navegação devido a inexistência de um caminho até o local desejado. No contexto do CMDE isto é plausível, já que é papel da arquitetura cognitiva detectar uma porta fechada, e assim, excitar o simulador apropriadamente para que o robô consiga abrir a porta, possibilitando que a navegação continue.

Na categoria “Foco da Visão” são agrupados comandos que visam direcionar a câmera do robô (no caso a visão) para determinado objeto ou posição. Atualmente existem três comandos nesta categoria: “*Look At*”, “*Look For*” e “*Head Reset*”.

O comando “*Look At*” faz com que o Robô foque seu olhar para um objeto ou posição de interesse. “*Look For*” tem como objetivo fazer com que o robô procure determinado elemento no ambiente através do movimento do pescoço. Os movimentos consistem em olhar para os lados, para baixo, para cima e nas diagonais. Ao encontrar o objeto desejado, o robô foca sua visão neste objeto, da mesma forma como é feito em “*Look At*”. Este comando é importante quando o robô se aproxima demais de um objeto e o mesmo desaparece de sua área de visão, ou quando se deseja verificar se um elemento está presente no ambiente. O terceiro comando,

“*Head Reset*”, tem objetivo fazer com que o robô desfaça o atual foco da visão, ou seja, faz com que ele olhe para frente, sem ter um foco de atenção em particular.

A categoria “Comunicação” tem como objetivo agrupar comandos que permitam ao robô comunicar com o avatar humano. Atualmente, somente um comando compõe esta classe. Através de “*Speech*”, o robô verbaliza o texto fornecido como meio de comunicação. Isto é simulado através de *strings*, onde a informação é fornecida ao usuário através da interface gráfica.

Por fim, comandos da categoria “Operacional” estão voltados para aspectos operacionais da execução dos comandos. No momento, o único comando que compõe esta classe é “*Cancel*”. Este comando é responsável por cancelar o comando atualmente ativo e demais comandos na fila de espera. Implementações futuras podem contemplar comandos para inverter as prioridades de outros comandos na fila ou cancelá-los individualmente.

O [Quadro 5](#) traz um resumo dos comandos apresentados agrupados pelas categorias citadas. A separação em categorias tem como vantagem a execução “simultânea” de comandos, isto porque algumas ações realizadas através destes comandos tem a necessidade de serem preservadas mesmo depois de sua conclusão. Por exemplo, quando o robô está com seu foco de visão direcionado a um objeto, é necessário atualizar constantemente a rotação da cabeça do robô para que ele continue olhando para o ponto de interesse, isto porque o robô ou o objeto podem se mover. Outro exemplo é quando o robô está segurando um copo, ele precisa atualizar constantemente a posição de repouso da mão ocupada para que demais movimentos do corpo não interfiram nesta posição. Se não houvesse esta separação em categorias, não seria possível o robô carregar um copo e se movimentar ao mesmo tempo. Isto porque um comando iria encerrar o anterior, fazendo com que o robô abandone o copo capturado, por exemplo.

Cada categoria de comandos é executada em *scripts* diferentes, isto faz com que o robô consiga manter ações de comandos de categorias distintas em paralelo. Por exemplo, o robô pode segurar um copo (interação), enquanto se locomove (movimentação) e olha para um humano sem perder o foco (foco de visão). A única categoria que permite individualmente duas ações ao mesmo tempo é “Interação”, isto graças as duas mãos do robô.

O nível de abstração dos comandos foi definido seguindo o conceito de *Affordance*. O termo *affordance* foi inicialmente introduzido por psicólogos na década de 60 e não tem uma tradução para a língua portuguesa, mas pode ser definido como a representação das conexões entre objetos do ambiente e uma sequência de passos memorizados no sistema motor para manipulá-los de forma automática e independente da representação abstrata do objeto ([SHAW; BRANSFORD, 2017](#)). Exemplos de *affordance*: porta: abrir/fechar; caneca: pegar; botão: apertar; etc. De forma resumida, tudo que é feito de forma automática por um ser humano adulto não será detalhado em sub-comandos pelo simulador.

A utilização de *affordance* libera a arquitetura cognitiva de modelar em detalhes aspectos sensorio/motor do robô, tais como rotação dos eixos dos membros, equilíbrio, interpretação do

Quadro 5 – Comandos disponíveis no simulador

Categoria	Comando	Descrição
Interação	<i>Take</i>	Pega objeto com determinada mão.
	<i>Leave</i>	Deixa objeto de determinada mão em uma dada localização.
	<i>Activate</i>	Altera o estado de um objeto para aberto/ligado.
	<i>Deactivate</i>	Altera o estado de um objeto para fechado/desligado.
	<i>Taste</i>	Prova o “sabor” de um objeto.
	<i>Smell</i>	Sente o “odor” de um objeto.
Movimentação	<i>Move</i>	Movimenta o robô para um objeto ou posição.
	<i>Rotate</i>	Rotaciona o robô em seu próprio eixo dado o ângulo (em graus).
	<i>Turn</i>	Rotaciona o robô para que ele encare determinado objeto ou posição.
Foco de Visão	<i>Look At</i>	Rotaciona a cabeça de forma que o robô foque em um objeto ou posição.
	<i>Look For</i>	Realiza uma varredura com a cabeça, considerando a visão, a procura de um objeto.
	<i>Head Reset</i>	Reenceta o foco da visão.
Comunicação	<i>Speech</i>	Simula a fala no formato de <i>string</i> para o usuário.
Operacional	<i>Cancel</i>	Cancela todos os comandos da fila, inclusive o em execução, caso exista.

dados sensoriais, processamento gráfico de imagens, etc.

Os comandos internos possuem estados de execução que permitem sua gerência. Outra característica destes estados é permitir com que a arquitetura cognitiva tenha informações sobre os comandos enviados. Os comandos internos são implementados em dois estágios, executando e finalizado. Porém o comando pode ser finalizado com sucesso, quando o objetivo é alcançado, ou com falha, caso contrário. Desta forma foram definidos três estados que auxiliam na gerência e atualização de comandos: *Running* (rodando), *Success* (sucesso) ou *Fail* (falha). Assim que um comando é criado, ele recebe o estado *Running*. Comandos recebidos pelo simulador são armazenados em fila, sendo executados somente quando o estado do comando atual é diferente de *Running*.

O estado *Fail* indica que um comando não pode ser executado, seja porque o objeto não é alcançável ou a mão solicitada para a tarefa esteja ocupada. Além disso, este estado é utilizado para indicar uma resposta negativa para a arquitetura cognitiva. Por exemplo, a arquitetura envia o comando para o robô procurar (“*Look For*”) o avatar humano ao seu redor, mas ele não o encontra, retornando o estado *Fail*. Ou seja, a resposta dada pode ser interpretada como “o avatar humano não está presente na área analisada”, talvez porque ele tenha mudado sua posição ou está se escondendo atrás de um objeto.

Um exemplo simples de falha que pode ser resolvido utilizando as informações dos sentidos é o comando “*Move*”. Supondo que o robô receba um comando para mover até um

local e encontre uma porta fechada em seu trajeto. Neste caso, a navegação é encerrada e o comando é definido como *Fail*. Através das informações dos sentidos, a arquitetura cognitiva terá conhecimento de que a porta está fechada e deverá excitar o simulador com comandos para contornar o problema, no caso enviar um comando para aproximar-se da porta e abri-la.

Por fim, vale a pena ressaltar que um comando não pode ser executado caso o robô não tenha conhecimento do objeto ou da localização desejada, ou seja, que não esteja em seu campo visual. Isto porque o robô não tem memória e nem capacidade de armazenar objetos e suas propriedades, sendo a arquitetura cognitiva responsável por tratar com estas questões. Os únicos elementos que o robô tem conhecimento de antemão são os cômodos da casa. Caso a arquitetura queira que o robô vá para um determinado local desconhecido por ele, ela deve enviar coordenadas do local em questão. No caso de locais ou objetos sendo vistos pelo robô (naquele instante de tempo) a arquitetura pode enviar o ID daquele elemento.

A seguir serão apresentados os sentidos utilizados no robô que habilitam a percepção do ambiente. Os objetos presentes em um dado cenários são passíveis de percepção pelos sentidos de acordo com suas propriedades internas.

4.4 Percepção do ambiente

A robótica social busca desenvolver técnicas para uma maior aproximação entre humanos e robôs. O simulador proposto juntamente com o CMDE, oferece suporte a realização de tarefas em ambientes domésticos interagindo diretamente com seres humanos. A execução de tarefas pelos humanos é auxiliada pelos seus sentidos, que fornecem informações do ambiente utilizadas para tomadas de decisão. Por exemplo, a visão pode ser utilizada para a localização de objetos, a audição para ouvir um alarme, o tato para conferir se uma roupa secou, o olfato para verificar se um alimento no forno não queimou e o paladar para examinar se um alimento está estragado ou não, além de várias outras atividades. Desta forma, nada mais natural do que utilizar sentidos similares aos dos seres humanos no agente robótico simulado.

Claro que a complexidade biológica dos seres humanos extrapola os cinco sentidos, gerando informações como equilíbrio, aceleração e propriocepção. Além disso, temos sensores não humanos capazes de medir radiação, campos magnéticos, luz infravermelha, som refletido, etc. os quais poderiam compor os sentidos do robô. A restrição da modelagem de cinco sentidos apresentada neste trabalho representa apenas um ciclo de interação no processo de desenvolvimento. Versões futuras da ontologia *OntSense* podem e devem aumentar o alcance das opções dos sensores, extrapolando a classificação tradicional dos sentidos.

Os sensores disponíveis no robô simulado fornecem informações de percepção processadas, ou seja, os dados fornecidos não necessitam de processamento adicional para serem utilizadas como por exemplo, utilizando técnicas de reconhecimento ou de visão computacional. Isto porque o nosso objetivo não é trabalhar com tais técnicas e nem disponibilizá-las para as

arquiteturas cognitivas. Como se trata de um ambiente simulado, tem-se o controle sobre os objetos que o robô percebe e assim fornecer informações processadas com pouco esforço. Ao operar com robôs reais a proposta é que o fabricante do robô adote a mesma postura, ou seja, realize o processamento dos dados antes da inserção na *triple store*.

O **Quadro 6** apresenta as propriedades associadas aos objetos, tais como, ID (identificador único), *tag*(identificador de tipo), nome, posição, tipo, cor, material e estado (aberto, fechado, etc.). O avatar humano possui todas estas características, porém ele vai além do conceito de objeto devido ao fato de ser um agente que modifica e sente o ambiente, além de possuir uma propriedade particular que é a emoção.

As propriedades representam a essência do objeto, sua posição e aparência. O sentido da visão habilita a percepção da “aparência” e da posição do objeto. Propriedades adicionais dos objetos são detectadas através dos outros sentidos, por exemplo, a rugosidade é fornecida pelo tato, a salinidade através do paladar, o odor através do olfato, e assim por diante. A seguir, cada um dos sentidos bem como as propriedades perceptíveis por cada um são apresentados.

4.4.1 Visão

A visão do robô é representada por uma câmera. Porém, no contexto do CMDE não é interessante e nem desejável enviar imagens do ambiente para a arquitetura cognitiva associada. Como dito anteriormente, são enviadas informações já processadas sobre os elementos do ambiente. A captura dos objetos é utilizada raios do tipo *UnityEngine.Physics.Raycast* como apresentado na **Figura 24**. São enviados um número limitado de raios em direção ao foco da visão do robô que, ao colidirem com um elemento do ambiente caracterizam um objeto que é acrescentado a uma lista.

Figura 24 – Em azul, raios que auxiliam na detecção de objetos no ambiente

Fonte: Elaborada pelo autor.

Como afirmado anteriormente, o sentido da visão permite identificar efetivamente quais são os elementos percebidos no ambiente, isto graças as propriedades detectadas por este sentido. Cada uma das propriedades é apresentada no [Quadro 6](#), seguidas de uma breve descrição e possíveis valores que podem compor a propriedade.

Geralmente, um objeto apresenta mais do que uma cor mas, por simplicidade somente a cor predominante do objeto é recuperada. Esta propriedade é representada pelo sistema de cores aditivas vermelho (R), verde (G) e azul (B), cada uma variando entre 0 e 1. Assim como a cor, um objeto pode conter mais de um material sua composição, mas aqui é utilizado somente a informação do material externo predominante. Neste trabalho, os objetos podem ser classificados nos seguintes materiais:

UnknownMaterial: desconhecido. Quando o material é desconhecido ou não se encaixa nas demais classes.

WoodMaterial: madeira e derivados. São objetos feitos de madeira, por exemplo, cadeira, mesa, porta retrato, etc. Note que porta retrato geralmente também é formado por vidro e papel, porém é levado em consideração somente a madeira, pois esta compõe a maior parte deste objeto.

MetalMaterial: cobre, alumínio, ferro, etc. Exemplos: geladeira, fogão, garfo, faca, etc.

PlasticMaterial: plásticos, polímeros e derivados. Exemplos: interruptor, tomadas elétricas, embalagens, relógio, etc. No caso do relógio, sua composição também leva metal por conta de suas engrenagens e mecanismos, porém a classificação é feita dada a sua parte visível, neste caso o plástico.

RubberMaterial: borracha. Exemplos: Bola, elástico, etc.

GlassMaterial: vidro. Exemplos: copo, caneca, janela, etc.

RockMaterial: pedra, concreto, cimento, porcelana, mármore e derivado de minérios. Exemplos, paredes, chão, canecas de porcelana, estátuas, etc.

OrganicMaterial: compostos orgânicos. É uma classe um pouco genérica pois engloba tanto alimentos quanto seres vivos e até mesmo derivados do petróleo que não se encaixam nas demais categorias. Alguns exemplos são, tortas, carnes, humanos, doces, grama, vela, cremes, etc.

PaperMaterial: papel e papelão. Exemplos: foto, jornal, caixas, livros, etc.

ClothMaterial: tecidos e panos. Esta categoria abrange qualquer tipo de tecido, tanto de origem vegetal quanto sintética. Exemplos: cortinas, roupas, tapete, almofada, sofá, etc.

Quadro 6 – Propriedades básicas associadas a objetos e agentes no simulador

Propriedade	Descrição	Valores
ID	Armazena um ID único do elemento.	Números inteiros.
<i>Tag</i>	Tipo do objeto. Ex. parede, porta, gaveta, chão, torneira, interruptor, etc.	Valores do tipo <i>String</i> .
Nome	Nome do objeto associado. Por padrão da Unity, todos os elementos tem um nome associado.	Valores do tipo <i>string</i> .
Posição	Posição do elemento no mundo simulado. São dadas através da classe <i>UnityEngine.Vector3</i> .	Coordenadas cartesianas.
Material Predominante	Tipo de material predominante que o objeto é composto.	<ul style="list-style-type: none"> • <i>UnknownMaterial</i> • <i>WoodMaterial</i> • <i>MetalMaterial</i> • <i>PlasticMaterial</i> • <i>RubberMaterial</i> • <i>GlassMaterial</i> • <i>RockMaterial</i> • <i>OrganicMaterial</i> • <i>PaperMaterial</i> • <i>ClothMaterial</i>
Cor Predominante	Armazena a cor predominante do objeto utilizando o sistema de cores aditivas, RGB.	Três valores (RGB) entre 0 e 1.
Estado	Tipo do objeto. Ex. parede, porta, gaveta, chão, torneira, interruptor, etc.	<ul style="list-style-type: none"> • <i>NoState</i> • <i>OnState</i> • <i>OffState</i> • <i>OpenedState</i> • <i>ClosedState</i>
Emoção	Armazena emoções humanas. Objetos e agentes robóticos não possuem este atributo.	<ul style="list-style-type: none"> • <i>NoEmotion</i> • <i>AngerEmotion</i> • <i>DisgustEmotion</i> • <i>FearEmotion</i> • <i>HappinessEmotion</i> • <i>SadnessEmotion</i> • <i>SurpriseEmotion</i>
URI	Armazena um localizador ou nome do objeto associando-o à Web Semântica.	<i>Link</i> , endereço web.

A classificação dos materiais foi feita de acordo com os objetos disponíveis no ambiente simulado. A partir do momento que outros cenários forem criados haverá a necessidade de criar novas categorias para materiais ou até mesmo reformular a classificação. Entretanto, para os ambientes atualmente utilizados, a classificação se adéqua perfeitamente.

As demais propriedades associadas a visão são: a emoção e o URI, abordadas na [Seção 4.1](#). No quadro apresentado, são utilizadas seis emoções básicas para classificação, além da emoção *NoEmotion* que representa agentes e objetos que não estão demonstrando emoção no

momento da captura.

4.4.2 Audição

Segundo [Schacter, Gilbert e Wegner \(2011\)](#) o som possui três propriedades: frequência, amplitude e timbre. A frequência é o comprimento de onda do som, medida em *hertz*. A amplitude diz respeito a intensidade do som, medida em decibéis. Uma mistura de frequências ou diferenças na complexidade das ondas sonoras corresponde ao timbre, que diz respeito a natureza do som. Todas estas propriedades em conjunto são essenciais para que os animais consigam interpretar sons no ambiente. No caso do simulador, as informações sonoras disponibilizadas pela audição não necessitam de processamento adicional para classificação pois indicam claramente o som capturado.

As seguintes propriedades estão associadas a captura de som: posição, descrição, volume e tipo ([Quadro 7](#)). O tipo de som identifica sua espécie, como por exemplo, sons de pássaros (*BirdsSound*), campainha (*BellSound*), motor (*MotorSound*), água escorrendo da torneira (*LiquidFlowingSound*), voz humana (*HumanVoiceSound*) e voz do robô (*RobotVoiceSound*).

A propriedade “descrição” diz respeito a informação que o som está transmitindo, como exemplo, no caso de um avatar humano, este campo fornece a frase dita pelo humano que será encaminhada para a arquitetura cognitiva.

Quadro 7 – Propriedades perceptíveis pela audição do robô simulado

Propriedade	Descrição	Valores
Posição	Local de origem do som. É substituído pelo ID, caso o objeto seja conhecido.	Coordenadas cartesianas ou identificador do objeto.
Tipo	Tipo do som emitido. Por exemplo, campainha, motor, música, etc.	<ul style="list-style-type: none"> • <i>UnknownSound</i> • <i>BarkingSound</i> • <i>BellSound</i> • <i>BirdsSound</i> • <i>LiquidFlowingSound</i> • <i>MotorSound</i> • <i>MusicSound</i> • <i>TVSound</i> • <i>HumanVoiceSound</i> • <i>RobotVoiceSound</i>
Descrição	Descrição ou informação do som. Por exemplo, quando o avatar humano fala, este campo armazena a frase dita.	Valores do tipo <i>string</i> .
Volume	Volume do som no ambiente.	Entre 0 e 1.

Através da audição, o robô tem conhecimento de que algum som está sendo emitido em determinado ponto do ambiente, mas nem sempre é capaz de estabelecer o objeto gerador

deste estímulo. Contudo, ele é capaz de identificar a posição exata de origem do som, o que permite a arquitetura cognitiva envie comandos para mover o robô até aquele ponto. A partir do momento que o elemento responsável pelo som é avistado (sentido da visão), o robô passa a enviar o identificador do objeto ao invés da posição. Note que somente o ID é necessário para identificar um objeto já que as demais propriedades, como a *Tag* por exemplo, são fornecidas pelo canal da visão.

4.4.3 Tato

O tato é o sentido que permite “sentir” o mundo ao nosso redor. Através de receptores dispostos na pele, é possível identificar pressão, textura, padrões e vibrações contra a pele (SCHACTER; GILBERT; WEGNER, 2011). Além disto, a pele dispõe de receptores referentes a dor, coceira, temperatura, propriocepção¹ e mecanorrecepção² (JOHNSON, 2002).

Os sensores de tato foram dispostos em cada uma das mãos do robô, capturando todo tipo de contato realizado. É claro que pode-se configurar sensores em todo o corpo do robô, porém outras áreas iriam gerar estímulos e informações de pouca valia para a atual nesta versão do CMDE.

Os sensores captam informações do objeto desde que exista um contato entre a mão do robô e o objeto. Estas informações dizem respeito a temperatura, pressão, rugosidade, umidade e dureza do objeto sendo apresentadas no Quadro 8. Estímulos de dor, coceira, percepção de movimento e deformações na pele não foram modelados no robô por não estarem ligadas diretamente as ações modeladas no sistema e não influenciarem de forma direta na tomada de decisão pela arquitetura cognitiva.

Quadro 8 – Propriedades perceptíveis pelo tato do robô simulado

Propriedade	Descrição	Valores
Posição	Local do objeto sentido pelo tato. É substituído pelo ID, caso o objeto seja conhecido.	Coordenadas cartesianas ou identificador do objeto.
Temperatura	Temperatura em graus Célsius.	Decimais (<i>float</i>) positivos e negativos (°C).
Pressão	Pressão exercida pelo objeto.	Entre 0 e 1.
Rugosidade	Nível de rugosidade do objeto.	Entre 0 e 1.
Umidade	Umidade presente no objeto.	Entre 0 e 1.
Dureza	Resistência do material do objeto.	Entre 0 e 1.

A propriedade Temperatura é dada em graus Celsius. Um objeto sem esta informação tem como valor a temperatura ambiente, a qual é uma constante definida no simulador. A pressão

¹ Percepção sobre movimento e força muscular, ângulo articular, deslocamento, equilíbrio, peso e distribuição do próprio corpo e das suas partes.

² Associado à deformações na pele.

diz respeito a força que o objeto exerce sobre o sensor. Nesta implementação, a pressão é dada levando em consideração a força que a massa do objeto exerce sobre a área do sensor. Este valor só é obtido quando o objeto é capturado pelo robô. Caso o robô somente encoste no objeto, não será possível calcular sua pressão.

Outras propriedades são a rugosidade, que diz respeito ao nível de aspereza do objeto, a dureza, que mede a resistência do material quando uma força é aplicada em sua superfície, e a umidade. Todas as propriedades precisam ser configuradas no objeto para definir o sentido do tato. Com exceção da propriedade temperatura, todas as propriedades recebem valores parametrizados entre 0 e 1.

Assim como o sentido da audição, o tato é capaz de detectar a posição do objeto em contato com as mãos do robô e ainda, quando o mesmo é conhecido fornecer seu ID. Vale lembrar que o objeto é conhecido quando ele está presente no campo de visão do robô ou está em sua posse, ou seja, em uma das mãos.

4.4.4 Olfato

O olfato é um mecanismo que permite a detecção e processamento de partículas odoríferas presentes no ar, seja para reconhecimento, direcionamento, acasalamento ou caça. Nos humanos, o olfato proporciona complexas informações que orientam o comportamento ingestivo, para orientação no ambiente e avaliação de risco em determinadas situações (DALTON, 2002).

Considerando sua utilização em agentes robóticos, o olfato pode ser utilizado em segurança (vazamento de gás ou detecção de fumaça), alimentos (identificação e preparo) e resgate (pessoas ou animais perdidos ou soterrados) (VILLARREAL; OLAGUE; GORDILLO, 2016).

Quadro 9 – Propriedades perceptíveis pelo olfato do robô simulado

Propriedade	Descrição	Valores
Posição	Local de origem do odor. É substituído pelo ID, caso o objeto seja conhecido.	Coordenadas cartesianas ou identificador do objeto.
Tipo	Classificação do odor detectado. Objetos com odor emitem partículas odoríferas no ambiente, no contexto do simulador, se o objeto não emite estas partículas então será classificado como <i>NoSmell</i> .	<ul style="list-style-type: none"> • <i>NoSmell</i> • <i>FragrantSmell</i> • <i>WoodySmell</i> • <i>FruitySmell</i> • <i>ChemicalSmell</i> • <i>DecayedSmell</i> • <i>MintySmell</i> • <i>SweetSmell</i> • <i>PopcornSmell</i> • <i>PungentSmell</i> • <i>LemonSmell</i>

O odor, no ambiente simulado, é modelado a partir da emissão de partículas que se

propagam no ambiente à partir de um objeto. Através de um sensor disposto na cabeça, o robô consegue capturar estas partículas no momento que elas entram em contato com o sensor. O objetivo dessas partículas é modelar parcialmente a distribuição de odores em um ambiente real. Na implementação atual algumas características inerentes a dispersão de partículas não são consideradas como por exemplo, movimentação de ar ou fluxo de geração.

As propriedades associadas ao olfato são apresentadas no [Quadro 9](#). A propriedade “Tipo” define o tipo do odor do objeto seguindo a classificação criada por [Castro, Ramanathan e Chennubhotla \(2013\)](#) e é detalhada no [Quadro 9](#). Objetos que não possuem odor não liberam partículas no ambiente, consequentemente não são percebidos pelo sentido do olfato. Neste caso, quando o robô executa o comando “*Smell*”, a informação obtida é *NoSmell*. As classes de cheiro utilizadas foram:

NoSmell: sem odor.

FragantSmell: perfumado. Exemplos: perfume, floral, rosa, violeta, etc.

WoodySmell: amaderado. Exemplos: cedro, ervas, grama cortada, terra, mofo, etc.

FruitySmell: frutado (não cítrico). Exemplos: abacaxi, cereja, banana, morango, etc.

ChemicalSmell: químico. Exemplos: medicinal, anestésico, desinfetante, ácido, leite azedo, etc.

DecayedSmell: fétido. Exemplos: putrefato, sujeira, rançoso, fezes, etc.

MintySmell: mentolado. Exemplos: Fresco, aromático, anis, alcaçuz, etc.

SweetSmell: doce. Exemplos: baunilha, chocolate, amêndoa, caramelo, etc.

PopcornSmell: pipoca. Exemplos: amanteigado, pasta de amendoim, oleoso, noz, amêndoa, etc.

PungentSmell: acre. Exemplos: alho, cebola, queimado, fumaça, etc.

LemonSmell: limão. Exemplos: cítrico, laranja, frutado (cítrico), etc.

Diferente dos outros sentidos apresentados anteriormente, não é possível identificar o objeto emissor de odor simplesmente através da visão. De fato, a identificação é realizada através do comando “*Smell*” que aproximação do objeto ao sensor de olfato presente na cabeça do robô.

4.4.5 *Paladar*

O paladar permite reconhecer o sabor de substâncias colocadas sobre a língua. Os animais utilizam o paladar para detectar substâncias que sejam comestíveis. Assim, sabores adocicados característicos de substâncias que produzem energia causam uma sensação de prazer identificados pelo paladar. Por outro lado, sabores amargos são associados a substância perigosas e geram uma

sensação de desconforto (SCHACTER; GILBERT; WEGNER, 2011). A língua é o principal órgão do sistema gustativo, onde os sabores são reconhecidos quando substâncias entram em contato com as papilas gustativas.

No robô simulado a posição do sensor gustativo está posicionado em local similar à língua nos mamíferos, ou seja, posicionada na cabeça do agente. Naturalmente, o sensor de reconhecimento poderia ser instalado em qualquer outra posição (nos dedos do robô, por exemplo), porém a posição escolhida deixa mais evidente e intuitiva a ação do robô quando ele experimenta uma substância. A partir deste sensor, o robô é capaz de detectar cinco propriedades do associadas a sensação de paladar de um objeto: doçura, acidez, salinidade, amargor e *umami*, os quais representam os cinco sabores básicos (HUANG *et al.*, 2006). O Quadro 10 lista estas propriedades. Abaixo segue uma breve descrição de cada um dos sabores e seus significados na percepção do paladar:

Doçura: evocada principalmente por açúcares. É um indicador de fontes de nutrientes.

Acidez: sabor produzido por ácidos. Pode ser considerado um sinal de alimentos decompostos e impróprios para consumo humano.

Salinidade: causada por materiais iônicos incluindo cloreto de sódio e de potássio. Indica um bom equilíbrio eletrolítico nos alimentos.

Amargor: sabor produzido por alcaloides. Considerado como um sinal de material venenoso e ajuda a impedir que os seres humanos ingiram tais materiais.

Umami: evocado principalmente por aminoácidos, geralmente encontrados em tomates maduros, carne, queijo e espargos, por exemplo.

Segundo Kobayashi e Ikezaki (2013) uma das dificuldades presentes na pesquisa sobre o paladar é a definição de uma unidade apropriada para as informações geradas pelos sensores gustativos. Existem diversas abordagens para apresentar a concentração de um sabor em uma amostra, tais como molaridade, percentual, título em massa, partes por milhão, etc. Estas representações são utilizadas nos processos químicos e naturalmente poderiam ser nossa unidade de medida para representar as informações do sensor gustativo do robô. Porém a interação entre substâncias faz com que as intensidades dos sabores percebidos pelos seres humanos mudem. Por exemplo a amargura do café pode ser reduzida com a adição de açúcar.

No contexto do simulador, a intensidade de cada propriedade é representada por valores parametrizados entre 0 e 1, sendo 1 o valor de maior concentração. O Quadro 10 traz um resumo sobre as propriedades do objetos relativas ao sabor.

O paladar é um sentido particular, pois necessita de um comando (“*Taste*”) para que seja ativado. Assim, para que o robô reconheça o sabor de um objeto é necessário que ele seja levado

Quadro 10 – Propriedades perceptíveis pelo paladar do robô simulado

Propriedade	Descrição	Valores
ID	Identidade do objeto.	Número inteiro.
Doçura	Concentração do gosto doce.	Entre 0 e 1.
Acidez	Concentração do gosto ácido.	Entre 0 e 1.
Salinidade	Concentração do gosto salgado.	Entre 0 e 1.
Amargor	Concentração do gosto amargo.	Entre 0 e 1.
Umami	Concentração do gosto umami.	Entre 0 e 1.

até o sensor gustativo posicionado na cabeça. Nesse contexto, a identificação do objeto deve ser conhecida isto é, tem-se o seu ID. Para “saborear” um objeto presente no ambiente é necessário pegá-lo e antes disto é preciso vê-lo.

4.5 Interação do simulador RHS com arquitetura cognitiva

Como citado anteriormente, o objetivo da *OntSense* é definir, para cada um dos sentidos, um conjunto de características mínimas que viabilize à arquitetura cognitiva reconstruir internamente o ambiente para execução de tarefas. Assim, o RHS (aderente à *OntSense*) minimiza os aspectos particulares associados a movimentação física do agente robótico para priorizar características essenciais associadas a cada sentido.

Figura 25 – Diagrama de sequência para a execução do pedido: fechar torneira

Fonte: Elaborada pelo autor.

Na [Figura 25](#) é apresentado um diagrama de sequência com um exemplo de interação do simulador RHS (representado por “Robô” e “Humano”) com a arquitetura cognitiva (ator “Arquitetura”). Este diagrama tem como objetivo elucidar a relação do simulador com dois componentes do CMDE, ontologia *OntSense* e arquitetura cognitiva.

No exemplo apresentado na figura, a interação se inicia quando o agente humano enuncia: “fechar torneira”. Desta forma o robô envia esta informação (aderente à *OntSense*) à arquitetura cognitiva via *API OntSense* (Ver [Figura 11](#)). Além desta informação, são enviados dados capturados pelos cinco sentidos do robô, inclusive a informação do ruído de água proveniente de determinada posição.

Baseada nessas informações, a arquitetura envia uma série de comandos em alto nível ao RHS a fim de atender o pedido do avatar humano. Logo após encontrar o objeto “torneira” através da visão, o sistema cognitivo extrai o identificador deste objeto e formula o comando *Move* “Ident Torneira” com objetivo de mover o robô até ele e, por fim, é enviado o comando *Close* “Ident Torneira” para fechar a objeto alvo.

4.6 Log de dados

O simulador RHS é uma ferramenta de suporte ao desenvolvimento e validação de arquiteturas cognitivas. Desta forma, é de grande importância apresentar ao usuário do simulador informações relativas a execução do sistema. Como a Unity permite que o desenvolvedor utilize o console para imprimir mensagens personalizadas, nada mais apropriado do que utilizar esta ferramenta para a apresentação de informações relativas ao simulador RHS.

As mensagens de *Log* do simulador tem como objetivo a apresentar informações de inicialização de módulos, recebimento e execução de comandos, erros de configuração e erros de sistema. Desta forma, as mensagens foram categorizadas em três classes: “*Command*”, “*RHS*” e “*System*”.

Na classe “*Command*”, as mensagens tem caráter de informar as etapas de execução de dado comando, isto desde que ele é recebido pelo simulador até o momento em que sua execução é finalizada. Quando um comando é finalizado com falha, não é justificada a causa desta condição à arquitetura cognitiva, porém esta informação é apresentada no relatório, onde o usuário pode ter acesso e assim ajustar a arquitetura apropriadamente.

Os comandos do tipo “*RHS*” tem objetivo de prover informações sobre o simulador de uma forma geral, tais como a inicialização de módulos, mensagens recebidas e enviadas a arquitetura, erros e ausência de módulos, etc. Os comandos do tipo “*System*” são responsáveis por apresentar mensagens de configuração e de erro relacionadas a bibliotecas do sistema, como por exemplo, erros de conexão via *socket*. Demais erros da Unity também são apresentados no relatório, porém sem uma classificação atribuída.

Figura 26 – Exemplo de mensagens do Log

```
RHS>>> simulation paused.  
RHS>>> socket incoming data.  
Command>>> Robot Kyle received command ID: d21d6897-73aa-4a5d-8382-b8d73f6c55f8 LookFor Mariana.  
Command>>> Robot Kyle command d21d6897-73aa-4a5d-8382-b8d73f6c55f8 Started!  
RHS>>> simulation speed changed: 1.  
RHS>>> simulation started.  
Command>>> Robot Kyle command d21d6897-73aa-4a5d-8382-b8d73f6c55f8 Success!
```

Fonte: Elaborada pelo autor.

Na [Figura 26](#) são exemplificados algumas mensagens do tipo “*Command*” e “*RHS*”. Nestas mensagens são apresentadas informações sobre o recebimento de um comando (*LookFor Mariana*) e sua execução, além de informar que a simulação foi pausada e posteriormente continuada com velocidade igual a 1.

As informações geradas pelo “Log de Dados” podem ser acessadas pelo usuário de duas formas, dependendo da estratégia usada para ativação do simulador:

Versão compilada: Nesse caso o simulador é ativado através da chamada de executável em modo *standalone*. O usuário pode acessar as mensagens, no sistema Windows, através do arquivo `RHS_DATA\output_log.txt` (diretório do executável).

Com suporte Unity: Nesse caso o simulador é ativado através da ferramenta Unity. A Unity possui um console que apresenta informações internas de erro, de aviso e as mensagens geradas pelos *scripts* do simulador.

4.7 Primeira versão e distribuição do software

A Unity permite a compilação do projeto para vários sistemas operacionais, incluindo Windows, Linux e Mac OS, tanto arquiteturas 32-bits quanto 64-bits. Porém, a primeira versão compilada do simulador RHS foi gerada para *Windows* 64-bits.

Como forma de distribuir a ferramenta desenvolvida à comunidade, o projeto integral foi disponibilizado no repositório GitHub através do endereço github.com/JPedroRBelo/RHS. Tanto o projeto (arquivos, modelos, *scripts*, etc.) quanto a versão compilada estão disponíveis no repositório. A licença atribuída ao projeto é a GNU *General Public License* v3.0 ([Free Software Foundation, 2018](#)), a qual habilita o usuário:

- usar o software para qualquer finalidade;
- mudar e adaptar o software de acordo com as necessidades;
- compartilhar o software com demais pessoas; e
- compartilhar as mudanças realizadas.

Alguns pacotes utilizados no desenvolvimento foram obtidos na *Asset Store*, sendo que um deles pago, especificamente o pacote que disponibiliza os elementos que compõe a sala, a cozinha e os objetos. Desta forma, o projeto aberto foi modificado para substituir o pacote mencionado, deixando os elementos do ambiente menos fiéis a realidade. Caso o utilizador necessite dispor de objetos e ambientes utilizados neste projeto, é possível os adquirir o *asset* no site oficial³. Contudo, a versão compilada oferece todos os recursos e elementos apresentados anteriormente, inclusive as funcionalidades do pacote pago.

4.8 Discussão

Para a execução das tarefas, o robô foi dotado dos cinco sentidos básicos dos humanos, visão, audição, tato, olfato e paladar, sendo estes dois últimos pouco explorados em simuladores robóticos. Como o foco do simulador RHS é prover a interação social do robô, além da execução de tarefas comuns a pessoas, nada mais conveniente do que utilizar os mesmos sensores (sentidos) dos seres humanos. Os humanos possuem outros “sensores”, tais como equilíbrio ou nível de oxigênio no sangue, porém a utilização dos cinco sentidos visa limitar o escopo do problema, dando ênfase nas ações modeladas e seus requisitos.

Uma questão que pode ser levantada está associada à validade de também incluir sensores para odor e sabor. Os robôs humanoides disponíveis no mercado não oferecem sensores para esses sentidos e a perspectiva de ser oferecida como um produto no curto prazo é remota. A inclusão desses sentidos foi realizada considerando dois aspectos. O primeiro está diretamente associado à robótica social que insere os robôs em atividades domésticas que exigem esse tipo de funcionalidade, em particular, manuseio e limpeza de alimentos ou detecção de vazamento de gás ou fumaça. O segundo aspecto está associado a uma característica básica dos simuladores, que é a capacidade de habilitar testes com dispositivos que ainda não estão disponíveis ou têm um alto custo de aquisição e operação. Assim, a inclusão desses sentidos viabiliza a pesquisa associada ao seu uso em sistemas robóticos futuros.

É interessante ressaltar que a estrutura proposta pode ser utilizada em ambientes não estruturados (veja [Capítulo 3](#)). Esta característica é alcançada pela renovação constante das informações sensoriais enviadas ao sistema cognitivo. Na versão atual, o período de atualização foi configurado como sendo 400ms. Isso significa que a arquitetura cognitiva terá constantemente atualizações do ambiente viabilizando a tomada de decisões em ambientes dinâmicos comumente encontrado em residências. Assim, a mudança da posição de objetos, a alteração de odor, humor, temperatura dentre outras, serão rapidamente assimilados e processados pelo robô.

³ <www.assetstore.unity3d.com/en/#!/content/31049>

4.9 Considerações finais

Neste Capítulo, foi apresentado o simulador RHS, o qual é uma ferramenta para a validação de arquiteturas cognitivas voltadas para a robótica, em particular para a robótica social. Distinto dos simuladores clássicos, o foco do simulador RHS é a interação social do robô com um humano sem haver a preocupação em configurar sensores e atuadores do agente robótico.

No próximo capítulo serão apresentados alguns experimentos envolvendo o simulador RHS integrado ao CMDE, o que abrange o *OntSense* e a arquitetura cognitiva Soar.

VALIDAÇÃO DO RHS

Neste capítulo, são apresentados dois experimentos realizados visando verificar se o simulador proposto modela as ações necessárias para exercitar sistemas cognitivos. Estes experimentos focam em demonstrar o ambiente para a execução de tarefas, dadas por comandos de alto nível, os dados sensoriais capturados utilizando os cinco sentidos e também a interação do usuário com o sistema, realizada através da fala do avatar simulado.

5.1 Experimentos realizados

O primeiro experimento consiste em utilizar um *driver*, que possibilita o envio de comandos viabilizando testes de integração do simulador. O segundo experimento utiliza todos os elementos presentes no ambiente CMDE inclusive as APIs, o servidor SPARQL e a arquitetura cognitiva Soar.

Todos os experimentos do simulador foram realizados em um computador com sistema operacional Microsoft Windows 10 64-bit, processador Intel®Core™ i7 CPU @ 2.80GHz × 8, placa gráfica GeForce GTX 480/PCIe/SSE2 e memória RAM de 7,8 GiB. A versão Unity utilizada é a 2017.4.0.

A arquitetura cognitiva foi executada em um segundo computador na mesma rede, porém com sistema operacional Ubuntu 16.04 LTS 64-bit, Intel®Core™ i7-6500U CPU @ 2.50GHz × 4, placa gráfica Intel®HD Graphics 520 (Skylake GT2) e memória RAM de 15,6 GiB.

5.1.1 Experimento inicial utilizando um driver

Para viabilizar os testes de integração da ferramenta, foi desenvolvido um *driver* com o intuito de auxiliar na validação dos comandos desenvolvidos para o simulador. O propósito do *driver* é substituir temporariamente uma arquitetura cognitiva, viabilizando o envio de comandos

ao simulador antes da implementação completa do comportamento do robô na arquitetura cognitiva.

A missão associada a este experimento consiste em requisitar ao robô para pegar um pacote de bolachas (“*Crackers*”). Desta forma, a execução do pedido se resume em ir até a cozinha, pegar o pacote de bolachas, levá-lo até o avatar humano e avisá-lo sobre a finalização da tarefa. Ao final, o usuário comanda o avatar humano para pegar o pacote da mão do robô utilizando os recursos da interface gráfica do simulador.

Figura 27 – Configuração inicial do ambiente para o primeiro experimento

Fonte: Elaborada pelo autor.

O ambiente foi configurado da seguinte forma: avatar humano disposto no centro da sala em pé; robô posicionado na sala de estar, no campo de visão do avatar humano; pacote de bolachas em cima da bancada da cozinha; porta entre a sala e a cozinha fechada; torneira aberta e salmão exalando odor próximo ao pacote de bolachas. Na Figura 27 é apresentada uma captura de tela com estas especificações. Através desta missão, é possível exercitar a maioria das tarefas que o robô é capaz de realizar, bem como a percepção das propriedades dos elementos através dos sentidos.

Inicialmente, o usuário insere o texto “Bring Crackers” (traga bolachas), onde o avatar simula a fala com esta informação. O robô, através da audição, captura o som da fala e o simulador apresenta este evento na interface do usuário. No contexto do CMDE, esta informação (entre outras relacionadas aos sentidos) seria enviada para a *triplestore*, porém neste experimento as informações serão somente apresentadas na tela. Desta forma, o usuário deve consultar a interface para planejar os comandos a serem gerados no *driver*. Na Figura 28, é mostrada a interface com cada um dos sentidos disponibilizados.

No Quadro 11, é mostrada uma lista com os comandos fornecidos através do *driver*, seus parâmetros e o estado final alcançado ao final da etapa associada àquele comando. Note que este

Figura 28 – Interface de usuário apresentando os dados capturados pelos sentidos do robô em diferentes etapas

Fonte: Elaborada pelo autor.

Nota – Em 28a, dados ao final da etapa 0. Em 28b, ao avatar humano enviar comando verbal. Em 28c, etapa 7. Em 28d etapa 6. E em 28e na etapa 8.

Quadro 11 – Experimento 1: comandos e parâmetros utilizados para atender a requisição de pegar um pacote de bolachas pelo avatar humano

Etapa	Comando	Parâmetro	Estado Final
0	<i>Move</i>	Position Kitchen	<i>Fail</i>
1	<i>Move</i>	ID Door_kitchen_living	<i>Success</i>
2	<i>ActivateLeft</i>	ID Door_kitchen_living	<i>Success</i>
3	<i>Move</i>	Position Kitchen	<i>Success</i>
4	<i>LookAt</i>	Position (-7.3,1.1,1.4)	<i>Success</i>
5	<i>Move</i>	ID Tap	<i>Success</i>
6	<i>DeactivateRight</i>	ID Tap hearing	<i>Success</i>
7	<i>TakeRight</i>	ID Salmon	<i>Success</i>
8	<i>TasteRight</i>	(null)	<i>Success</i>
9	<i>LeaveRight</i>	ID Counter	<i>Success</i>
10	<i>LookFor</i>	“Crackers”	<i>Success</i>
11	<i>Move</i>	ID Crackers	<i>Success</i>
12	<i>TakeLeft</i>	ID Crackers	<i>Success</i>
13	<i>Move</i>	Position LivingRoom	<i>Success</i>
14	<i>LookFor</i>	“Mariana”	<i>Success</i>
15	<i>Move</i>	ID Mariana	<i>Success</i>
16	<i>Speech</i>	“Order completed”	<i>Success</i>

estado não indica uma falha de execução, mas uma impossibilidade de executar dado comando, com as configurações atuais do ambiente. Os valores dos parâmetros apresentados neste quadro não representam o valor real passado ao robô.

Na etapa 0, é dado o comando para o robô se mover até a posição da cozinha. Mas, durante a execução da tarefa, o robô se depara com uma obstrução no caminho. Através das informações sensoriais, especificadamente a visão, é possível detectar o problema pela informação:

ID: 2896
Door: Door_kitchen_living
Position: (-4.2,1.0,-1.3)
RGB: 0.267, 0.337, 0.431
Type: Wood
Status: Closed

onde o valor de *Status* é *closed*, indicando que há uma porta fechada em seu campo de visão (Figura 28a). Após executar comandos para abrir a porta, é enviado novamente o comando para que o robô se mova até a cozinha. Neste caso, a ação foi concluída com sucesso.

Na etapa 3, é dada atenção ao sentido da audição, que está capturando determinado som na posição “-7.3, 1.1, 1.4”. Veja na Figura 28b (momento em que robô recebia comando do avatar), que existe um evento de percepção relacionado a esta posição do tipo *LiquidFlowing-Sound*. Contudo, outras características que identificam o objeto produtor do estímulo não são apresentadas. Isto porque o robô não tem conhecimento deste objeto, ou seja, ele ainda não está

visível. A partir do momento que o robô vê o objeto, dado pelo comando da etapa 4, é possível obter seu ID e assim dar comandos utilizando este parâmetro. Nas etapas 5 e 6, decidimos que o robô deveria fechar a torneira.

Durante a execução dos comandos anteriores, o robô capturou através do olfato (ver [Figura 28d](#)) partículas de cheiro. Neste caso, tem-se também somente a posição de origem do odor, sendo responsabilidade do agente externo (arquitetura ou usuário) pedir ao robô experimentar suposto objeto a fim de verificar a origem deste estímulo. Nas etapas 7 e 8, foi determinado que o robô deveria experimentar o sabor do objeto *Salmon*, que estava em seu campo de visão. Na [Figura 28e](#) são apresentadas as propriedades gustativas do objeto em questão e na [Figura 28c](#) as relativas ao tato.

Logo após, foi dada continuidade na execução das tarefas relacionadas ao pedido do pacote de bolachas pelo avatar humano. O robô então, realiza uma busca visual pela bolacha, vai até ela e a captura com a mão esquerda (etapas 10, 11 e 12). Logo após, ele é comandado a retornar à sala. Chegando lá, recebe outro comando para procurar o avatar e posteriormente mover-se até ele (etapas 14 e 15). Por fim, é enviada a mensagem “Order completed” (pedido concluído) e apresentada na tela do usuário. O usuário então controla o avatar para pegar o pacote de bolachas da mão do robô.

Neste estudo de caso, todas as ações geradas pelo robô foram executadas com sucesso (sem erros de execução ou erros imprevistos). O robô conseguiu caminhar, pegar e deixar objetos, ativar e desativar elementos do ambiente e interagir com estes. É claro que se for enviado um comando ao robô que ele não é capaz de executar (pegar um copo fora do campo de visão ou de alcance) irá gerar uma falha. No entanto, isto não é um erro de implementação ou de execução do simulador, mas sim um comando equivocado ou precoce dado por parte do usuário ou da arquitetura. O robô (no simulador RHS) é capaz de executar tarefas quando estas estão ao seu alcance, sendo responsabilidade da arquitetura prepará-lo previamente para tal execução. Por exemplo, mover-se até determinado objeto para pegá-lo, pegar um objeto antes de prová-lo, ou como visto anteriormente, abrir uma porta antes de mudar de cômodo.

Com o intuito de ilustrar melhor a execução desse experimento, foi disponibilizado no Youtube¹ o vídeo que apresenta partes etapas da execução dos passos presentes no [Quadro 11](#).

5.1.2 Experimento com o simulador integrado ao CMDE

Para a realização do segundo experimento, o simulador foi integrado ao ambiente CMDE, conforme apresentado na [Figura 10](#) na [Seção 3.1](#). Esse experimento utiliza as APIs responsáveis pela troca de informações com o servidor SPARQL de forma aderente a ontologia *OntSense*. Além disto, neste caso não se utiliza o *driver* do experimento anterior. Em seu lugar, é utilizada a arquitetura Soar, a qual interpreta as informações de percepção obtidas do servidor SPARQL e

¹ www.youtube.com/watch?v=36kP8SNGqcQ

envia comandos para o simulador.

Neste experimento, dois cenários são exercitados. No primeiro, o robô percebe o som de água fluindo de determinada posição, compartilha essa informação com o avatar humano. O avatar humano ordena o fechamento da torneira apropriada, o robô fecha a torneira e retorna a sua posição original. No segundo cenário, o avatar humano pede ao robô um pacote de bolachas, de forma similar ao cenário explorado pelo experimento da [Subseção 5.1.1](#), porém a porta entre a sala e a cozinha está aberta e o odor do salmão não está presente.

Quadro 12 – Experimento 2: comandos e parâmetros utilizados para atender o pedido de fechar a torneira

Etapa	Comando	Parâmetro	Estado Final
0	<i>Speech</i>	“Sound of flowing liquid”	<i>Success</i>
1	<i>Move</i>	Kitchen	<i>Success</i>
2	<i>LookAt</i>	Position (3.2,1.2,-4.3)	<i>Success</i>
3	<i>Move</i>	ID Tap	<i>Success</i>
4	<i>DeactivateRight</i>	ID Tap	<i>Success</i>
5	<i>Move</i>	Position LivingRoom	<i>Success</i>
6	<i>LookFor</i>	“Mariana”	<i>Success</i>
7	<i>Move</i>	ID Mariana	<i>Success</i>
8	<i>Speech</i>	“Order Completed”	<i>Success</i>

Quadro 13 – Experimento 2: comandos e parâmetros utilizados para atender o pedido de pegar um pacote de bolachas

Etapa	Comando	Parâmetro	Estado Final
9	<i>Move</i>	Position Kitchen	<i>Success</i>
10	<i>LookFor</i>	“Crackers”	<i>Success</i>
11	<i>Move</i>	ID Crackers	<i>Success</i>
12	<i>TakeLeft</i>	ID Crackers	<i>Success</i>
13	<i>Move</i>	Position LivingRoom	<i>Success</i>
14	<i>LookFor</i>	“Mariana”	<i>Success</i>
15	<i>Move</i>	ID Mariana	<i>Success</i>
16	<i>Speech</i>	“Order Completed”	<i>Success</i>

As duas partes do experimento são dadas na mesma seção de simulação, assim que ele cumpre a primeira tarefa, ele recebe a segunda. Os comandos recebidos pela arquitetura estão disponíveis no [Quadro 12](#) e no [Quadro 13](#).

Ao iniciar a simulação, o robô avisa o avatar sobre o som de um líquido fluindo, por sua vez, o usuário solicita ao robô, através do avatar: “Close Tap”. Neste experimento, o robô é guiado a ir até a origem do som de água fluindo (etapas 1 e 2 do [Quadro 12](#)). Seria perfeitamente aceitável que a arquitetura soubesse que a torneira fica na cozinha (levando em consideração os cômodos disponíveis na simulação) e assim guiar o robô até o local a procura de uma torneira. Na etapa 3, o robô recebe o comando para fechar a torneira. Logo depois ele é guiado a ir até

a posição inicial, a qual indica o ponto onde a tarefa foi dada a ele (etapa 4). Finalmente, a arquitetura decide mover o robô ao avatar e informar a conclusão da tarefa (etapas 5,6 e 7).

Figura 29 – Avatar humano pegando o pacote de bolachas da mão do robô

Fonte: Elaborada pelo autor.

A próxima interação é dada pelo usuário, que decide pedir ao robô para pegar o pacote de bolachas. Desta vez, a arquitetura assume que o pacote está na cozinha, movendo o robô até lá e o fazendo procurar o objeto (etapas 8 e 9 do [Quadro 13](#)). Da mesma forma do Experimento 1, o robô move-se até o objeto e o captura (etapas 10 e 11). A arquitetura o move para a posição inicial e executa os procedimentos para entregar pacote ao avatar humano (etapas 12, 13, 14 e 15). A interação termina com o avatar capturando o pacote na mão do robô, ilustrada pela [Figura 29](#).

Neste experimento, todos os comandos dados pela arquitetura foram concluídos com sucesso (*Success*) e todas as ações executadas pelo robô não apresentaram erros de execução. A ocorrência de falhas da execução de comandos exige que a arquitetura cognitiva associada tenha uma capacidade maior da análise dos elementos recebidos pelos sentidos. Por exemplo, no caso da porta fechada do experimento anterior, a arquitetura teria que induzir que o problema é dado pela porta fechada ou por um obstáculo colocado no caminho do robô. Outra situação que pode ocorrer é quando a arquitetura envia o comando “*LookFor*” para procurar o avatar, por exemplo, caso o robô não o encontre, é retornado uma resposta negativa (*Fail*). Neste caso, é esperado que arquitetura envie o robô a outro local ou o rotacione para realizar a busca em outra área do cômodo.

Através deste experimento, a arquitetura conseguiu manipular um robô a fim de executar tarefas interagindo com um humano. Cada uma das duas tarefas foram executadas utilizando um conjunto de comandos de alto nível (definidas por *affordance*) baseadas nos cinco sentidos

definidos. Desta forma, o usuário (no experimento anterior), a arquitetura cognitiva e o protocolo *OntSense* não tiveram que trabalhar com dados brutos de sensores e atuadores.

5.2 Considerações finais

Foram apresentados dois experimentos envolvendo o simulador proposto. A primeira utilizando o *driver* que permite o usuário enviar comandos ao simulador e a segunda utilizando o CMDE, onde os comandos são gerados pela arquitetura cognitiva.

Experimentos mais elaborados, com nível de complexidade crescente entre as missões, com certeza iriam enriquecer a demonstração e validação do simulador. Porém, quanto mais elaboradas as missões, maior a capacidade de planejamento e aprendizado que a arquitetura cognitiva deve prover. Apesar de estimulante, é importante ressaltar que essa atividade transcende o escopo inicial do projeto. Contudo, os experimentos realizados mostram que o simulador, junto ao protocolo *OntSense*, é capaz de prover um ambiente para validação de arquiteturas cognitivas, atendendo ao objetivo específico deste projeto.

Dado o simulador RHS e a ontologia *OntSense*, arquiteturas cognitivas voltadas para área de IHR podem ser testadas sem a preocupação de trabalharem diretamente com sensores e atuadores. Esta característica permite que desenvolvedores de arquiteturas cognitivas trabalhem com uma linguagem mais próxima a humana do que a da robótica, o que de certa forma contribui na minimização da lacuna existente na comunicação entre humanos e robôs.

CONCLUSÃO E TRABALHOS FUTUROS

Nesta dissertação, foi proposto um simulador para robótica social, denominado *Robot House Simulator*, RHS, voltado para ambientes internos. A aplicação foi direcionada para ambientes de uma residência: sala e cozinha. O foco desse simulador é capturar informações de percepção do ambiente disponibilizados para o sistema cognitivo, via sentidos do robô. Esse simulador é parte integrante do ambiente para avaliação de sistemas cognitivos, CMDE, em desenvolvimento no Laboratório de Aprendizado de Robôs da USP de São Carlos.

O objetivo principal do RHS é viabilizar experimentos que envolvam robótica social e cognição com o uso de simulador aderente ao conceito de software livre. As vantagens de utilização do ambiente CMDE e em particular o RHS incluem: acelerar o processo de implementação de sistemas cognitivos em robôs, viabilizar a reprodução de experimentos associados a sistemas cognitivos, permitir a comparação entre implementações distintas e apoiar a instalação de cursos de IHR com foco em robótica social.

Apesar da utilização de uma arquitetura cognitiva, o simulador RHS poderá ser utilizado com outra técnica. Ele poderá ter como centro de controle qualquer arquitetura de tomada de decisão, tais como: algoritmos de aprendizado de máquina, redes neurais, autômatos finitos, etc.

O simulador RHS possui três características importantes que não foram identificadas em trabalhos similares:

Sensores para odor e sabor: os simuladores disponíveis na literatura não oferecem sensores para os sentidos de olfato e paladar. A inclusão da percepção desses sentidos no simulador RHS está diretamente associada à robótica social, que insere os robôs em atividades que exigem essa funcionalidade, em particular, limpeza doméstica e manuseio de alimentos. Outro aspecto está associado a uma característica básica dos simuladores: capacidade de viabilizar testes com dispositivos que ainda não estão disponíveis ou tem alto custo de aquisição e operação.

Hierarquia de comandos: a associação dos comandos com *affordance* (SHAW; BRANSFORD, 2017) representa uma abordagem promissora para modelar comandos gerados por uma arquitetura cognitiva atuando sobre um agente robótico. Essa estratégia permite que ações repetitivas com modelamento complexo mas muito bem mapeadas na literatura (definição de trajetórias por exemplo), sejam atribuídas ao agente robótico, liberando a arquitetura cognitiva para atividade mais relevantes considerando cognição.

Acesso Web: a definição da ontologia *OntSense* para refletir as informações de percepção possibilita acesso a tecnologia de *Web Semântica*, viabilizando o acesso ao gigantesco provedor de informações oferecido pela Web. Essas informações completam os dados de percepção do ambiente produzindo um processamento mais rico por parte da arquitetura cognitiva.

O trabalho desenvolvido nesta dissertação contribuiu para a publicação de 4 artigos:

- BELO, J. P. R.; ROMERO, R. A. F.; AZEVEDO, H. Simulador para sistemas cognitivos voltado para robótica social. In: **XIII Simpósio Brasileiro de Automação Inteligente - SBAI2017**. Porto Alegre, Brazil: SBA, 2017.
- BELO, J. P. R.; AZEVEDO, H.; ROMERO, R. A. F. RHS Simulator for Robotic Cognitive Systems. In: **14th Latin American Robotics Symposium(LARS 2017) and 5th Brazilian Symposium on Robotics (SBR 2017)**. Curitiba, Brazil: IEEE, 2017.
- AZEVEDO, H.; ROMERO, R. A. F.; BELO, J. P. R. Reducing the gap between cognitive and robotic systems. In: **2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)**. 2017.
- AZEVEDO, H.; BELO, J. P. R.; ROMERO, R. A. F. Cognitive and robotic systems: Speeding up integration and results. In: **14th Latin American Robotics Symposium(LARS 2017) and 5th Brazilian Symposium on Robotics (SBR 2017)**. 2017.

Além dos artigos publicados, foi também submetido para uma revista o seguinte artigo:

- AZEVEDO, H.; BELO, J. P. R.; ROMERO, R. A. F. Cognition, Affordance and Robotics: using ontology to formalize the sensory information. In: **Integrated Computer-Aided Engineering**. Submetido em: julho, 2017.

Através dos experimentos realizados, o simulador se mostrou eficiente em prover um ambiente de validação para arquiteturas cognitivas voltadas à robótica social. As etapas futuras de desenvolvido envolvem:

Inclusão de outros cômodos: nesta primeira versão do RHS, o simulador contempla o modelamento de ambientes residenciais, tais como, sala e cozinha. A implementação de banheiros, quartos, corredores, e outros tipos de cômodos, que tradicionalmente constituem uma casa, permitirá uma dimensão maior de problemas a serem validados.

Inclusão de outros objetos: a utilização de outros objetos enriquecerá os ambientes e viabilizará a execução de tarefas mais elaboradas.

Extensão para ambientes externos: quintais, casas vizinhas, farmácias, supermercados, hospitais, etc. Esta etapa aproximará o simulador da proposta da geração de cenários complexos para testes de inserção de agentes robóticos em uma área mais extensa.

Inclusão de outros sentidos: tais como sensores de aceleração, radioatividade ou electromagnetismo, equilíbrio ou nível de oxigênio no sangue.

Definição de comandos adicionais: engloba o desenvolvimento de ações que permitem a arquitetura cognitiva aumentar o grau de envolvimento com o ambiente. Por exemplo, sentar, pular, movimentos gestuais, carregar pessoas, etc.

Prover imersão ao usuário: isto é, através do óculos VR e do Kinect prover uma interação maior do usuário com o ambiente e com o robô simulado.

Melhoria do *GodMode*: aprimorar este modo de interação para que o usuário seja capaz de configurar propriedades de audição, olfato, paladar, tato e visão.

Permitir a interação de vários usuários: envolve a utilização de vários avatares instanciados no simulador, cada um controlado por um usuário diferente, através do mesmo nó de processamento ou em outros nós conectados na mesma rede.

Integração completa com o ROS: utilizar o *framework* ROS com o intuito de dar uma maior versalidade ao simulador RHS.

Estas etapas buscam enriquecer o simulador RHS de forma que contribuam para o desenvolvimento de arquiteturas cognitivas e também para que participantes de experimentos possam ter uma maior interação e experiência com o sistema a ser validado.

REFERÊNCIAS

ABB. **RobotStudio - ABB Robotics**. 2018. Disponível em: <<http://new.abb.com/products/robotics/robotstudio>>. Acesso em: 02 jan. 2018. Citado na página 39.

ACT-R Research Group. **ACT-R**. 2013. Disponível em: <<http://act-r.psy.cmu.edu/about/>>. Acesso em: 08 fev. 2018. Citado na página 44.

_____. **ACT-R**. 2013. Disponível em: <<http://act-r.psy.cmu.edu/>>. Acesso em: 05 jan. 2018. Citado na página 45.

ADOBE. **3D Animation Online Services, 3D Characters, and Character Rigging - Mixamo**. 2018. Disponível em: <<https://www.mixamo.com/>>. Acesso em: 5 jan. 2018. Citado na página 60.

Adobe. **Criar modelos e personagens 3D**. 2018. Disponível em: <<http://www.adobe.com/br/products/fuse.html>>. Acesso em: 5 jan. 2018. Citado na página 60.

ALEXANDER, A. L.; BRUNYÉ, T.; SIDMAN, J.; WEIL, S. A. From gaming to training: A review of studies on fidelity, immersion, presence, and buy-in and their effects on transfer in pc-based simulations and games. **DARWARS Training Impact Group**, v. 5, p. 1–14, 2005. Citado na página 25.

ANYKODE. **Marilou: the universal mechatronic software**. 2016. Disponível em: <<http://www.anycode.com/marilou.php>>. Acesso em: 02 jan. 2018. Citado nas páginas 34 e 35.

APACHE, S. F. **Apache Jena Fuseki**. 2017. Disponível em: <<https://jena.apache.org/documentation/fuseki2/>>. Acesso em: 20 jan. 2018. Citado na página 47.

AUSTIN, M. **Google built an entire fake city to test the AI of its driverless cars**. 2017. Disponível em: <<https://www.digitaltrends.com/cars/google-fake-city/>>. Citado na página 25.

AZEVEDO, H. **Protocolo de Comunicação entre Sistemas Cognitivo e Robótico por meio de Ontologia Cognitiva**. 2016. Monografia (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP. Citado na página 26.

AZEVEDO, H.; ROMERO, R. A. F. Interface for communication between the robotic and cognitive systems through the use of a cognitive ontology. In: **COGNITIVE 2016, The Eighth International Conference on Advanced Cognitive Technologies and Applications**. Roma, Italy: Curran Associates, Inc., 2016. p. 1–3. Citado nas páginas 48 e 51.

AZEVEDO, H.; ROMERO, R. A. F.; BELO, J. P. R. Reducing the gap between cognitive and robotic systems. In: **2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)**. [S.l.: s.n.], 2017. p. 1049–1054. Citado na página 51.

BELO, J. P. R.; AZEVEDO, H.; ROMERO, R. A. F. Rhs simulator for robotic cognitive systems. In: **14rd Latin American Robotics Symposium - LARS'2017**. Curitiba, Brazil: IEEE, 2017. ISBN: 978-1-5090-3656-1. Citado na página 51.

BERRY, C. A. Teaching a First Course in Human-Robot Interaction. In: **2015 ASEE Annual Conference and Exposition**. Seattle, Washington: ASEE Conferences, 2015. <https://peer.asee.org/24799>. Citado na página 26.

BONNECHERE, B.; OMELINA, L.; JANSEN, B.; JAN, S. V. S. Balance improvement after physical therapy training using specially developed serious games for cerebral palsy children: preliminary results. **Disability and rehabilitation**, Taylor & Francis, v. 39, n. 4, p. 403–406, 2017. Citado na página 55.

Boston Dynamics. **Atlas - The World's Most Dynamic Humanoid**. 2017. Disponível em: <http://www.bostondynamics.com/atlas>. Acesso em: 02 jan. 2018. Citado na página 38.

CARPIN, S.; LEWIS, M.; WANG, J.; BALAKIRSKY, S.; SCRAPPER, C. Usarsim: a robot simulator for research and education. In: IEEE. **Robotics and Automation, 2007 IEEE International Conference on**. [S.l.], 2007. p. 1400–1405. Citado na página 32.

CASTRO, J. B.; RAMANATHAN, A.; CHENNUBHOTLA, C. S. Categorical Dimensions of Human Odor Descriptor Space Revealed by Non-Negative Matrix Factorization. **PLoS ONE**, Public Library of Science, San Francisco, USA, v. 8, n. 9, p. e73289, sep 2013. ISSN 1932-6203. Disponível em: <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3776812/>. Citado nas páginas 51 e 80.

CRAIGHEAD, J. **About SARGE**. 2008. Disponível em: <http://sarge.sourceforge.net/page11/page11.html>. Acesso em: 03 jan. 2018. Citado na página 41.

CRAIGHEAD, J.; BURKE, J.; MURPHY, R. Using the unity game engine to develop sarge: A case study. v. 4552, 01 2007. Citado na página 41.

CYBERBOTICS. **Webots robot simulator**. 2018. Disponível em: <https://www.cyberbotics.com/>. Acesso em: 02 jan. 2018. Citado na página 31.

DALTON, P. Olfaction. **Stevens' handbook of experimental psychology**, Wiley Online Library, 2002. Citado na página 79.

DARPA. **The DARPA Grand Challenge: Ten Years Later**. 2014. Disponível em: <http://www.darpa.mil/news-events/2014-03-13>. Acesso em: 24 jan. 2017. Citado na página 25.

_____. **What is the DARPA Robotics Challenge (DRC)?** 2015. Disponível em: <http://archive.darpa.mil/roboticschallenge/trialsarchive/about/index.html>. Acesso em: 24 jan. 2017. Citado nas páginas 25, 35 e 38.

_____. **About DARPA**. 2018. Disponível em: <https://www.darpa.mil/about-us/about-darpa>. Acesso em: 02 jan. 2018. Citado na página 34.

DER, R.; MARTIUS, G. The lpzrobots simulator. In: **The Playful Machine**. [S.l.]: Springer, 2011. p. 293–308. Citado na página 36.

DIANKOV, R.; KUFFNER, J. Openrave: A planning architecture for autonomous robotics. **Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34**, v. 79, 2008. Citado na página 34.

DOTNETRDF, P. **dotNetRDF: An Open Source .NET Library for RDF**. 2017. Disponível em: <<http://www.dotnetrdf.org/>>. Acesso em: 20 jan. 2018. Citado na página 48.

ECHEVERRIA, G.; LASSABE, N.; DEGROOTE, A.; LEMAIGNAN, S. Modular open robots simulation engine: Morse. In: IEEE. **Robotics and Automation (ICRA), 2011 IEEE International Conference on**. [S.l.], 2011. p. 46–51. Citado na página 37.

ECHEVERRIA, G. b.; LEMAIGNAN, S. b. c.; DEGROOTE, A. b.; LACROIX, S. b.; KARG, M.; KOCH, P.; LESIRE, C.; STINCKWICH, S. f. Simulating complex robotic scenarios with morse. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 7628 LNAI, p. 197–208, 2012. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84868031048&doi=10.1007%2f978-3-642-34327-8_20&partnerID=40&md5=67bff3a53ae2e2d35582679f473e2efe>. Citado nas páginas 29 e 37.

EKMAN, P.; FRIESEN, W. V. Constants across cultures in the face and emotion. **Journal of personality and social psychology**, American Psychological Association, v. 17, n. 2, p. 124, 1971. Citado na página 64.

EKMAN, P. E.; DAVIDSON, R. J. **The nature of emotion: Fundamental questions**. [S.l.]: Oxford University Press, 1994. Citado na página 64.

ENERGID TECHNOLOGIES CORPORATION. **Actin Toolkit Released By Energid Robotics**. 2015. Disponível em: <<http://www.energid.com/about-us/news-item/energid-releases-actin-toolkit/>>. Acesso em: 02 jan. 2018. Citado na página 33.

Epic Games. **Game Engine Technology by Unreal**. 2018. Disponível em: <<https://www.unrealengine.com/what-is-unreal-engine-4>>. Acesso em: 03 jan. 2018. Citado nas páginas 41 e 42.

FANUC America Corporation. **ROBOGUIDE - FANUC Robot Simulation Software | FANUC America**. 2017. Disponível em: <<http://fanucamerica.com/home/products-services/robots/robot-simulation-software-FANUC-ROBOGUIDE>>. Acesso em: 02 jan. 2018. Citado na página 39.

FLEMING, T. M.; BAVIN, L.; STASIAK, K.; HERMANSSON-WEBB, E.; MERRY, S. N.; CHEEK, C.; LUCASSEN, M.; LAU, H. M.; POLLMULLER, B.; HETRICK, S. Serious games and gamification for mental health: current status and promising directions. **Frontiers in psychiatry**, Frontiers, v. 7, p. 215, 2017. Citado na página 55.

FONG, T.; NOURBAKHSH, I.; DAUTENHAHN, K. A survey of socially interactive robots. **Robotics and Autonomous Systems**, v. 42, n. 3-4, p. 143–166, 2003. Citado na página 23.

Free Software Foundation. **The GNU General Public License v3.0 - GNU Project - Free Software Foundation**. 2018. Disponível em: <<https://www.gnu.org/licenses/gpl-3.0.en.html>>. Acesso em: 16 jan. 2018. Citado na página 84.

GAMITO, P.; OLIVEIRA, J.; COELHO, C.; MORAIS, D.; LOPES, P.; PACHECO, J.; BRITO, R.; SOARES, F.; SANTOS, N.; BARATA, A. F. Cognitive training on stroke patients via virtual reality-based serious games. **Disability and rehabilitation**, Taylor & Francis, v. 39, n. 4, p. 385–388, 2017. Citado na página 55.

GOODRICH, M. A.; SCHULTZ, A. C. Human-robot interaction: a survey. **Foundations and trends in human-computer interaction**, Now Publishers Inc., v. 1, n. 3, p. 203–275, 2007. Citado na página 23.

Google Developers. **Protocol Buffers**. 2017. Disponível em: <<https://developers.google.com/protocol-buffers/docs/overview>>. Acesso em: 10 jan. 2018. Citado na página 52.

GUDWIN, R.; PARAENSE, A.; PAULA, S. de; FRÓES, E.; GIBAUT, W.; CASTRO, E.; FIGUEIREDO, V.; RAIZER, K. An overview of the multipurpose enhanced cognitive architecture (meca). **Procedia Computer Science**, Elsevier, v. 123, p. 155–160, 2018. Citado na página 43.

HARRIS, A.; CONRAD, J. Survey of popular robotics simulators, frameworks, and toolkits. In: . [S.l.: s.n.], 2011. p. 243–249. Cited By 12. Citado na página 29.

HAUSER, K. **IROS 2016 Grasping and Manipulation Competition Simulation Framework**. 2016. Disponível em: <http://www.rhgm.org/activities/competition_iros2016/simulation.pdf>. Acesso em: 02 jan. 2018. Citado na página 36.

HUANG, A. L.; CHEN, X.; HOON, M. A.; CHANDRASHEKAR, J.; GUO, W.; TRÄNKNER, D.; RYBA, N. J.; ZUKER, C. S. The cells and logic for mammalian sour taste detection. **Nature**, Nature Publishing Group, v. 442, n. 7105, p. 934–938, 2006. Citado na página 81.

HUGUES, L.; BREDECHE, N.; FUTURS, T. Simbad: an autonomous robot simulation package for education and research. In: SPRINGER. **SAB**. [S.l.], 2006. v. 4095, p. 831–842. Citado na página 33.

IEEE. **IEEE Standard Otologies for Robotics and Automation**. New York, NY, 2015. Citado nas páginas 50 e 52.

INAMURA, T. **Okonomiyaki Collaborative Cooking Agent**. 2011. Disponível em: <<http://www.sigverse.org/wiki/en/?Okonomiyaki%20Collaborative%20Cooking%20Agent>>. Acesso em: 02 jan. 2018. Citado na página 36.

INAMURA, T.; MIZUCHI, Y. Competition design to evaluate cognitive functions in human-robot interaction based on immersive vr. In: **RoboCup International Symposium**. [S.l.: s.n.], 2017. Citado nas páginas 37 e 41.

INAMURA, T.; SHIBATA, T.; SENA, H.; HASHIMOTO, T.; KAWAI, N.; MIYASHITA, T.; SAKURAI, Y.; SHIMIZU, M.; OTAKE, M.; HOSODA, K. *et al.* Simulator platform that enables social interaction simulation—sigverse: Sociointelligence simulator. In: IEEE. **System Integration (SII), 2010 IEEE/SICE International Symposium on**. [S.l.], 2010. p. 212–217. Citado na página 36.

INOUE, H.; HIRUKAWA, H. Hrp: Humanoid robotics project of miti. **Journal of the Robotics Society of Japan**, The Robotics Society of Japan, v. 18, n. 8, p. 1089–1092, 2000. Citado na página 32.

INTELLIGENT MOTION LAB. **Intelligent Motion Laboratory at Duke University**. 2017. Disponível em: <<http://motion.pratt.duke.edu/klampt/>>. Acesso em: 02 jan. 2018. Citado na página 35.

Intuitive Surgical, Inc. **Intuitive Surgical - da Vinci Si Surgical System - Skills Simulator**. 2018. Disponível em: <https://www.intuitivesurgical.com/products/skills_simulator/>. Acesso em: 02 jan. 2018. Citado na página 40.

- IT+Robotics. **WorkcellSimulator - IT+Robotics**. 2018. Disponível em: <<http://www.it-robotics.it/products/3d-simulation/workcellsimulator/?lang=en#>>. Acesso em: 02 jan. 2018. Citado na página 40.
- JACKSON, J. Microsoft robotics studio: A technical introduction. **IEEE Robotics & Automation Magazine**, IEEE, v. 14, n. 4, 2007. Citado na página 34.
- JOHNSON, K. Neural basis of haptic perception. **Stevens' handbook of experimental psychology**, Wiley Online Library, 2002. Citado na página 78.
- KOBAYASHI, Y.; IKEZAKI, H. Advanced taste sensors based on artificial lipid membrane. **Biochemical Sensors: Mimicking Gustatory and Olfactory Senses**, CRC Press, p. 5, 2013. Citado na página 81.
- KOENIG, N.; HOWARD, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In: IEEE. **Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on**. [S.l.], 2004. v. 3, p. 2149–2154. Citado na página 32.
- KOTSERUBA, I.; TSOTSOS, J. K. 40 years of cognitive architectures core cognitive abilities and practical applications. **arXiv preprint arXiv:arXiv:1610.08602v2**, 2017. Citado nas páginas 24, 42, 43 e 44.
- KUO, P.-H.; HO, Y.-F.; LEE, K.-F.; TAI, L.-H.; LI, T.-H. Development of humanoid robot simulator for gait learning by using particle swarm optimization. In: . [s.n.], 2013. p. 2683–2688. Disponível em: <<https://www.scopus.com/inward/record.uri?eid=2-s2.0-84893606684&doi=10.1109%2fSMC.2013.457&partnerID=40&md5=13b9498b17bcea8289eac27dc4a570cb>>. Citado na página 29.
- LAIRD, J. E. Extending the soar cognitive architecture. **Frontiers in Artificial Intelligence and Applications**, IOS Press, v. 171, p. 224, 2008. Citado na página 43.
- _____. **The Soar cognitive architecture**. [S.l.]: MIT press, 2012. Citado na página 45.
- LANGLEY, P.; LAIRD, J. E.; ROGERS, S. Cognitive architectures: Research issues and challenges. **Cognitive Systems Research**, Elsevier, v. 10, n. 2, p. 141–160, 2009. Citado na página 47.
- LEHMAN, J.; LAIRD, J.; ROSENBLOOM, P. **A gentle introduction to soar, an architecture for human cognition: 2006 update**. [S.l.]: Retrieved, 2007. Citado na página 43.
- LI, W.; SONG, P.; TAN, J.; ZHU, C.; DUAN, F. Verification the feasibility of sigverse for human-robot interaction simulation through following task. In: . [s.n.], 2015. p. 1189–1194. Disponível em: <<https://www.scopus.com/inward/record.uri?eid=2-s2.0-84964526742&doi=10.1109%2fROBIO.2015.7418933&partnerID=40&md5=334bf8872a831a62221adebd2f96617e>>. Citado na página 37.
- Logic Desing Inc. **Robologix » RoboLogix Overview**. 2018. Disponível em: <https://www.robologix.com/robologix_overview.php>. Acesso em: 02 jan. 2018. Citado na página 39.
- MATTINGLY, W.; CHANG, D.-J.; PARIS, R.; SMITH, N.; BLEVINS, J.; OUYANG, M. Robot design using unity for computer games and robotic simulations. In: . [s.n.], 2012. p. 56–59. Disponível em: <<https://www.scopus.com/inward/record.uri?eid=>

- 2-s2.0-84869068961&doi=10.1109%2fCGames.2012.6314552&partnerID=40&md5=dc328ff379db2d2265c76c65faf3a36b>. Citado nas páginas 41 e 55.
- MICHEL, O. Khepera simulator version 2.0 user manual. **Université de Nice–Sophia Antipolis. Laboratoire I3S-CNRS, France/EPFL–Lausanne, Swiss**, 1996. Citado na página 30.
- _____. Webots: Symbiosis between virtual and real mobile robots. In: SPRINGER. **International Conference on Virtual Worlds**. [S.l.], 1998. p. 254–263. Citado na página 30.
- NEKOBOLT. **Complete Home Interior Pack**. 2015. Disponível em: <<https://www.assetstore.unity3d.com/en/#!/content/31049>>. Acesso em: 05 jan. 2018. Citado na página 58.
- NOVIKOVA, J.; WATTS, L.; INAMURA, T. Modeling human-robot collaboration in a simulated environment. In: ACM. **Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts**. [S.l.], 2015. p. 181–182. Citado na página 37.
- Ogre3D. **OGRE - Open Source 3D Graphics Engine | Home of a marvelous rendering engine**. 2018. Disponível em: <<https://www.ogre3d.org/#>>. Acesso em: 02 jan. 2018. Citado na página 41.
- OLIVER, B.; JEAHEUNG, P.; MARC, T. Mobility and Manipulation. In: SICILIANO, B.; KHATIB, O. (Ed.). **Springer Handbook of Robotics**. 2. ed. Berlin, Heidelberg: Springer International Publishing, 2016. cap. 40, p. 1007–1036. ISBN 978-3-319-32550-7. Disponível em: <<http://link.springer.com/10.1007/978-3-319-32552-1>>. Citado na página 23.
- Open Source Robotics Foundation. **Gazebo**. 2014. Disponível em: <<http://gazebosim.org>>. Acesso em: 18 jan. 2018. Citado na página 32.
- _____. **Gazebo: Blog: New Feature Highlight: Multiple Physics Engines**. 2014. Disponível em: <http://gazebosim.org/blog/feature_physics>. Acesso em: 02 jan. 2018. Citado na página 41.
- OPENSOURCEROBOTICSFOUNDATION. 2014. Disponível em: <<https://bitbucket.org/osrf/drcsim/wiki/Home>>. Acesso em: 27 dez. 2017. Citado nas páginas 38 e 39.
- PARAENSE, A. L.; RAIZER, K.; PAULA, S. M. de; ROHMER, E.; GUDWIN, R. R. The cognitive systems toolkit and the cst reference cognitive architecture. **Biologically Inspired Cognitive Architectures**, v. 17, p. 32 – 48, 2016. ISSN 2212-683X. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S2212683X1630038X>>. Citado na página 43.
- PEASE, A. **Suggested Upper Merged Ontology (SUMO)**. 2016. Disponível em: <<http://www.adampease.org/OP/>>. Acesso em: 10 dez. 2016. Citado na página 50.
- PINCIROLI, C.; TRIANNI, V.; O'GRADY, R.; PINI, G.; BRUTSCHY, A.; BRAMBILLA, M.; MATHEWS, N.; FERRANTE, E.; Di Caro, G.; DUCATELLE, F.; BIRATTARI, M.; GAMBARDELLA, L. M.; DORIGO, M. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. **Swarm Intelligence**, Springer, Berlin, Germany, v. 6, n. 4, p. 271–295, 2012. Citado na página 40.
- PLAYER. **The Player Project**. 2014. Disponível em: <<http://playerstage.sourceforge.net/>>. Acesso em: 28 dez. 2017. Citado nas páginas 31 e 32.

PRATS, M.; PÉREZ, J.; FERNÁNDEZ, J. J.; SANZ, P. J. An open source tool for simulation and supervision of underwater intervention missions. In: IEEE. **Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on**. [S.l.], 2012. p. 2577–2582. Citado na página 40.

RDF Working Group. **RDF - Semantic Web Standards**. 2014. Disponível em: <<https://www.w3.org/RDF/>>. Acesso em: 9 mai. 2018. Citado na página 47.

REISBERG, D. (Ed.). **The Oxford Handbook of Cognitive Psychology**. [S.l.]: Oxford University Press, 2013. ISBN 9780195376746. Citado na página 24.

ROBOCUP. **RoboCup - Objective**. 2016. Disponível em: <<http://www.robocup.org/objective/>>. Acesso em: 02 jan. 2018. Citado na página 32.

ROBOCUP2016. **Amazon Picking Challenge RoboCup**. 2016. Disponível em: <<http://www.robocup2016.org/en/events/amazon-picking-challenge/>>. Acesso em: 02 jan. 2018. Citado na página 35.

Robotics VO. A roadmap for us robotics: from internet to robotics. **Robotics Virtual Organization**, 2013. Citado na página 24.

RÖFER, T. Strategies for using a simulation in the development of the bremen autonomous wheelchair. In: CITESEER. **Society for Computer Simulation International**. [S.l.], 1998. Citado na página 30.

ROHMER, E.; SINGH, S.; FREESE, M. V-rep: A versatile and scalable robot simulation framework. In: . [s.n.], 2013. p. 1321–1326. Disponível em: <<https://www.scopus.com/inward/record.uri?eid=2-s2.0-84893724133&doi=10.1109%2fIROS.2013.6696520&partnerID=40&md5=4aa1243e9c20f345a2f2d6ec16ce5490>>. Citado nas páginas 37 e 38.

ROS. **About ROS**. 2018. Disponível em: <<http://www.ros.org/about-ros/>>. Acesso em: 15 jan. 2018. Citado nas páginas 31 e 37.

SCHACTER, D.; GILBERT, D. T.; WEGNER, D. M. **Psychology (2nd Edition)**. New York: Worth, 2011. Disponível em: <http://www.amazon.com/Psychology-Daniel-L-Schacter/dp/1429237198/ref=sr_1_1?s=books&ie=UTF8&qid=1313937150&sr=1-1>. Citado nas páginas 77, 78 e 81.

SHAW, R.; BRANSFORD, J. **Perceiving, acting and knowing: Toward an ecological psychology**. [S.l.]: Routledge, 2017. v. 27. Citado nas páginas 71 e 96.

Siemens Product Lifecycle Management Software Inc. **RobotExpert: Siemens PLM Software**. 2017. Disponível em: <<https://www.plm.automation.siemens.com/en/products/tecnomatix/manufacturing-simulation/robotics/robotexpert.shtml>>. Acesso em: 02 jan. 2018. Citado na página 39.

SIEMS, U.; HERWIG, C.; RÖFER, T. **SimRobot: ein Programm zur Simulation sensorbestückter Agenten in einer dreidimensionalen Umwelt**. [S.l.]: ZKW, 1994. Citado na página 29.

SIMBAD. **Simbad Project Home**. 2011. Disponível em: <<http://simbad.sourceforge.net/>>. Acesso em: 02 jan. 2018. Citado na página 33.

TAATGEN, N.; ANDERSON, J. R. The past, present, and future of cognitive architectures. **Topics in Cognitive Science**, Wiley Online Library, v. 2, n. 4, p. 693–704, 2010. Citado na página 43.

TAN, J.; INAMURA, T. b. Sigverse - a cloud computing architecture simulation platform for social human-robot interaction. In: . [s.n.], 2012. p. 1310–1315. Disponível em: <<https://www.scopus.com/inward/record.uri?eid=2-s2.0-84864448971&doi=10.1109%2FICRA.2012.6225359&partnerID=40&md5=7e6f9f3a82cc6881c28194ffe2d6f4d0>>. Citado na página 39.

THAGARD, P. Cognitive science. In: ZALTA, E. N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Fall 2014. [S.l.]: Metaphysics Research Lab, Stanford University, 2014. Citado na página 42.

TSARDOULIAS, M.; ZALIDIS, C.; THALLAS, A. **STDR Simulator (Simple Two Dimensional Robot Simulator) - ROS robotics news**. 2014. Disponível em: <<http://www.ros.org/news/2014/02/stdr-simulator-simple-two-dimensional-robot-simulator.html>>. Acesso em: 02 jan. 2018. Citado na página 38.

Unity Technologies. **Space Robot Kyle**. 2012. Disponível em: <<https://www.assetstore.unity3d.com/en/#!/content/4696>>. Acesso em: 5 jan. 2018. Citado na página 60.

_____. **Unity**. 2018. Disponível em: <<https://unity3d.com/pt>>. Acesso em: 2 jan. 2018. Citado na página 41.

_____. **Unity - Products**. 2018. Disponível em: <<https://unity3d.com/pt/unity>>. Acesso em: 3 jan. 2018. Citado na página 41.

USARSim. **USARSim**. 2015. Disponível em: <<https://sourceforge.net/projects/usarsim/>>. Acesso em: 19 dez. 2017. Citado na página 33.

VILLARREAL, B. L.; OLAGUE, G.; GORDILLO, J. Synthesis of odor tracking algorithms with genetic programming. **Neurocomputing**, Center for Robotics and Intelligent Systems, Tecnológico de Monterrey, Monterrey, N.L., Mexico, v. 175, p. 1019–1032, jan 2016. ISSN 09252312. Citado na página 79.

W3C. **SPARQL 1.1 Protocol**. 2013. Disponível em: <<https://www.w3.org/TR/sparql11-protocol/>>. Acesso em: 20 jan. 2018. Citado na página 47.

WINKLER, L.; VONÁSEK, V.; WÖRN, H.; PŘEUCIL, L. Robot3d—a simulator for mobile modular self-reconfigurable robots. In: IEEE. **Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on**. [S.l.], 2012. p. 464–469. Citado na página 40.

WOUTERS, P.; OOSTENDORP, H. V. **Instructional Techniques to Facilitate Learning and Motivation of Serious Games**. [S.l.]: Springer, 2017. Citado na página 55.

Yaskawa America, Inc. **MotoSim EG, Motoman Simulator Enhanced Graphics**. 2017. Disponível em: <<https://www.motoman.com/hubfs/PDFs/MotoSimEG.pdf?t=1514527424551>>. Acesso em: 02 jan. 2018. Citado na página 39.