Universidade da Beira Interior
Departamento de Informática

Inteligência Artificial: *Projeto Prático*

Elaborado por:

41358 — Beatriz Tavares da Costa
41381 — Igor Cordeiro Bordalo Nunes

Docente:

Professor Doutor Luís Filipe Barbosa de Almeida Alexandre

7 de Janeiro de 2021

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

IA Inteligência Artificial

UC Unidade Curricular

SWOT *Strength, Weakness, Opportunity and Threat*

Capítulo 1

Introdução

O presente Capítulo descreve os objetivos delineados para a implementação deste projeto, bem como a organização do documento.

O projeto doravante apresentado foi desenvolvido no âmbito da Unidade Curricular (UC) de Inteligência Artificial (IA) como um projeto introdutório a esta área.

1.1 Constituição do grupo

O presente projeto foi realizado pelos elementos listados na Tabela 1.1.

Nº	**Nome**
41358	Beatriz Tavares da Costa
41381	Igor Cordeiro Bordalo Nunes

Tabela 1.1: Constituição do grupo de trabalho.

1.2 Objetivos

Este projeto prático tem como principal objetivo criar a inteligência de um *robot* que tem como seu mundo virtual o piso de um hospital. Este dispositivo percorre o mapa sob o controlo de certos *inputs* do teclado. A fim de explorar a programação da inteligência do *robot*, deve também ser possível fazer perguntas ao mesmo, e este terá de ser capaz de responder às mesmas em qualquer momento da simulação, pelo que é necessário implementar funções que sejam capazes de obedecer a tais requisitos.

1.3 Organização do Documento

De modo a refletir o trabalho que foi feito, este documento encontra-se estruturado da seguinte forma:

1. No primeiro capítulo — **Introdução** — é apresentado o projeto, o enquadramento do mesmo, a constituição do grupo de trabalho, a enumeração dos objetivos delineados para a conclusão do mesmo e a respetiva organização do documento.

2. No segundo capítulo — **Tecnologias Utilizadas** — são descritas as ferramentas e bibliotecas utilizadas no desenvolvimento da inteligência do *robot* no mundo virtual disponibilizado.

3. No terceiro capítulo — **Desenvolvimento e Implementação** — são apresentadas e descritas as escolhas, os algoritmos pensados e métodos utilizados na implementação dos mesmos.

4. No quarto capítulo — **Reflexão Crítica e Problemas Encontrados** — denota-se a divisão de tarefas pelos elementos do grupo e expõe uma análise *Strength, Weakness, Opportunity and Threat* (SWOT) onde se expõem os pontos fortes, fracos, oportunidades e ameaças ao trabalho desenvolvido.

5. No quinto capítulo — **Conclusões e Trabalho Futuro** — , apresenta-se uma reflexão do trabalho e conhecimentos adquiridos ao longo do desenvolvimento do projeto prático e um contrabalanço com a possibilidade de existirem objetivos não alcançados e que se podem explorar no futuro.

Capítulo 2

Tecnologias e Ferramentas Utilizadas

2.1 Introdução

Este capítulo enumera as tecnologias e ferramentas utilizadas para o desenvolvimento da parte de inteligência de um *robot* num mundo virtual, projeto descrito neste documento.

2.2 Ferramentas Utilizadas

São descritas em seguida, primariamente, as ferramentas e tecnologias utilizadas, acrescentando também a razão pelas quais as mesmas foram escolhidas.

1. Biblioteca *NetworkX* [1] — biblioteca do *Python* [2] utilizada para criar e manipular as estruturas de dados utilizadas (grafos);
2. *Git* — sistema de controlo de versões no qual foi gerido o repositório do código-fonte do projeto;
3. *Python* [2] — linguagem de programação utilizada para o desenvolvimento do projeto.

2.3 Conclusões

Neste capítulo foram descritas as tecnologias e ferramentas utilizadas na realização e desenvolvimento do projeto referido neste documento. As mesmas são referidas nos próximos capítulos para efeitos de explicação de como as mesmas foram aplicadas.

Capítulo 3

Desenvolvimento e Implementação

3.1 Introdução

Este capítulo explora o percurso realizado aquando do desenvolvimento do projeto descrito neste documento, em particular as escolhas e detalhes de implementação.

3.2 Escolhas de Implementação

Antes da implementação dos algoritmos para satisfazer as respostas às perguntas colocadas, foi necessário definir a estrutura de dados utilizada para a manipulação do mundo virtual e dos objetos nele contidos.

Desta forma, além da escolha do uso de listas para guardar os objetos que o dispositivo robótico encontra no seu mundo virtual, foi utilizada numa primeira abordagem uma matriz 800×600 para efeitos de aplicação de um algoritmo de *path-finding* (necessário para as respostas aos enunciados descritos nas subsecções **??**, **??** e **??**). Esta abordagem revelou alguns problemas de eficiência, conforme é refletido na secção 4.4). A solução passou então pela substituição da matriz por uma estrutura de dados que permite representar de forma bastante eficiente os dados recolhidos sobre o mundo virtual: grafos. Para este fim, foi utilizada a biblioteca *NetworkX* [1] para a criação e gestão dos mesmos. Dois grafos foram utilizados neste âmbito:

1. Grafo *floor* (Figura 3.1) — armazena informações sobre as salas visitadas, a sua ligação e os objetos nelas contidos;

2. Grafo *map* (Figura 3.2) — realiza o mapeamento do mundo, ao registar em detalhe todos os caminhos possíveis entre salas e as portas que as conectam, para efeitos de execução do algoritmo de *path-finding* A*.

Figura 3.1:

Por análise do grafo da Figura 3.1, podemos constatar que se trata de um grafo não dirigido, composto por nodos e arestas, sendo que os nodos guardam informações acerca das divisões e como seus atributos constam os objetos contidos nessas mesmas divisões. Os atributos estão definidos como um dicionário, onde a chave do mesmo é a categoria de objeto e o valor é a lista de nomes dos objetos dessa categoria encontrados pelo *robot*.

Figura 3.2:

Por seu turno, o grafo da Figura 3.2, sendo na mesma um grafo não dirigido, armazena em cada nodo a posição das salas visitadas (i.e. o ponto médio de cada sala) e das portas que conectam as salas em tuplos do tipo (x, y), e em cada aresta é dado como atributo a distância euclidiana entre os dois nodos que esta conecta.

Em termos de organização do código-fonte do trabalho prático, optou-se por se utilizar classes, as quais se enumeram seguidamente:

1. *Log* — útil para efeitos de *debugging* de forma a identificar mais facilmente possíveis *bugs*;
2. *LinearFunction* — permite criar instâncias de funções lineares necessárias para a resposta às perguntas 5 e 6 (subsecções 3.3.5 e 3.3.6, respetivamente);
3. *Things* — armazena listas de objetos e pessoas, tratando diretamente da resposta à pergunta 1 (subsecção 3.3.1) e auxiliando as restantes classes a gerir os seus dados;
4. *Robot* — realiza a gestão dos dados inerentes ao *robot*, em particular a velocidade e a bateria, bem como a sua relação com o tempo;
5. *Hospital* — principal classe do programa na qual a informação relativa ao piso do hospital é atualizada conforme as informações dadas pelo *robot*;
6. *Utils* — coleta um conjunto de funções auxiliares, como por exemplo o cálculo de distâncias, a troca de variáveis e a descrição textual de caminhos.

3.3 Detalhes de Implementação

Nesta secção são descritos os detalhes de implementação para cada pergunta proposta no enunciado do projeto prático, através da explicação dos respetivos algoritmos.

3.3.1 Pergunta 1

Enunciado: *Qual foi a penúltima pessoa que viste?*

A classe *Things* fornece o método `getLastButOnePerson()`, o qual devolve o nome da penúltima pessoa que o *robot* encontrou. Este método depende da prévia invocação da função `add()` por parte da classe *Hospital* a fim de atualizar corretamente as pessoas encontradas.

Caso o *robot* não tenha encontrado pelo menos duas pessoas no momento da pergunta, é devolvida uma mensagem pré-definida de erro pela seguinte constante:

```
ERROR_NOT_ENOUGH_PEOPLE = "Não foram encontradas pelo menos 2 pessoas até ao momento"
```

3.3.2 Pergunta 2

Enunciado: *Em que tipo de sala estás agora?*

A classe *Hospital* providencia a função `getCurrentTypeOfRoom()`, o qual devolve um código que indica o tipo da sala onde o *robot* se encontra no momento. Esta função encapsula a utilização de outra: a função `getTypeOfRoom()`, a qual permite determinar o tipo de qualquer sala a qualquer momento. Neste caso, a primeira função encontra-se definida da seguinte forma:

```
1 @staticmethod
2 def getCurrentTypeOfRoom():
3     return
    ↪  Hospital.getTypeOfRoom(Hospital._currentRoom)
```

O método `roomDescription()` converte o código devolvido pelas funções anteriores em descrições textuais a fim de produzir o *output* pretendido.

De notar que a função `getTypeOfRoom()` determina o tipo da sala com o seguinte algoritmo:

1. Se o número da sala está no intervalo $[1, 4]$, então devolve o código correspondente a um corredor;
2. Para cada atributo do nodo da sala atual, é feita a contagem de cada tipo de objeto que seja mobília (cama, cadeira e mesa);
3. A contagem dos objetos de cada categoria indica o respetivo tipo de sala:
   - Quarto: ≥ 1 cama;
   - Sala de enfermeiros: 0 camas, ≥ 1 cadeiras **e** ≥ 1 mesas;
   - Sala de espera: > 2 cadeiras, 0 mesas, 0 camas.

3.3.3 Pergunta 3

Enunciado: *Qual o caminho até à sala de enfermeiros mais próxima?*

Para esta pergunta, o método `getPathToNearestNurseOffice()` da classe *Hospital* devolve uma lista com o nome dos nodos do grafo *map* correspondentes ao caminho mais curto desde a posição atual do *robot* até à sala de enfermeiros mais próxima.

Para este fim, o *robot* é temporariamente adicionado ao grafo *map* e são feitas as ligações por arestas a todos os vizinhos do nodo correspondente à sala onde o *robot* se encontra. De seguida, o algoritmo A* é aplicado sobre o grafo recorrendo aos métodos `astar_path` e `astar_path_length` da biblioteca *NetworkX* para todas as salas de enfermeiros conhecidas até ao momento. O caminho mais curto de todos os caminhos determinados corresponde ao caminho para a sala de enfermeiros mais próxima, sendo então este caminho codificado numa lista, a qual é devolvida.

O método `pathDescription()` da classe *Utils* converte esta lista numa série de instruções em português para fácil leitura do utilizador.

3.3.4 Pergunta 4

Enunciado: *Qual é a distância até ao médico mais próximo?*

Para esta questão, não tendo sido pedido o caminho, optou-se por se procurar o médico mais próximo em linha reta. A função `getDistanceToNearestDoctor()` da classe *Hospital* providencia a resposta.

Desta forma, o grafo *floor* é filtrado de forma a recolher os médicos até então encontrados e as respetivas posições. A lista resultante é então mapeada com a função `distance()` da classe *Utils* de forma a converter as posições dos médicos em distâncias euclidianas em relação ao *robot*. A lista é ordenada pela distância, sendo então devolvido o primeiro elemento desta, correspondente ao médico mais próximo.

A informação é devolvida com o seguinte formato: "Médico <*nome*> na sala <*nº da sala*> a uma distância de <*distância*>."

3.3.5 Pergunta 5

Enunciado: *Quanto tempo achas que demoras a ir de onde estás até às escadas?*

A resposta a esta pergunta assemelha-se à da pergunta 3 (secção 3.3.3), sendo tratada pela função `getTimeToStairs()` da classe *Hospital*. O *robot* é adicionado temporariamente ao grafo *map* e é determinado o caminho mais curto até às escadas (codificada internamente como sendo a sala 0 (zero)).

Com base na distância determinada, é estimado o tempo que demorará a chegar às escadas com recurso ao método `predictTimeFromDistance()` da classe *Robot*. Esta classe calcula a cada nova posição do *robot* a variação da velocidade ao longo do tempo sob a forma de uma função linear a fim de se poder extrapolar o tempo necessário a percorrer uma certa distância.

A estimativa é feita com a seguinte fórmula:

$$\Delta t = \frac{2d}{v_f + v_i} \tag{3.1}$$

Esta é determinada com base no seguinte facto físico:

$$d = \int_{t_i}^{t_f} v(t)dt \tag{3.2}$$

A distância é, portanto, a área sob a curva da função $v(t)$ (velocidade em função do tempo). Todavia, no nosso caso, a "curva" é uma reta uma vez que se optou por usar funções lineares para maior eficiência.

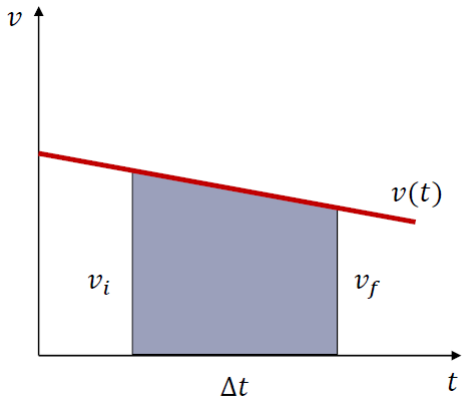


Figura 3.3: Estimativa da velocidade do *robot*: função $v(t)$
Estimativa da velocidade do *robot*: função $v(t)$.

Desta forma, a área sob a curva é a área de um trapézio (Figura 3.3), tal que:

$$d = \frac{v_f + v_i}{2}\Delta t \tag{3.3}$$

Por rearranjo da igualdade, chega-se à equação (3.1).

O tempo determinado é formatado pela função `timeToStr()` da classe *Utils* de forma a apresentar a estimativa em segundos e milissegundos.

3.3.6 Pergunta 6

Enunciado: *Quanto tempo achas que falta até ficares sem bateria?*

À semelhança da questão 5 (secção 3.3.5), a classe *Robot* avalia a variação da bateria ao longo do tempo com uma função linear.

O método `getTimeToDie()` da classe *Hospital* encapsula o método `predictTimeFromBa` da classe *Robot*, pedindo o tempo que falta até a bateria chegar a $0\,\%$.

Novamente, o método `timeToStr()` da classe *Utils* é utilizado para converter o número obtido em segundos e milissegundos.

3.3.7 Pergunta 7

Enunciado: *Qual a probabilidade de encontrar um livro numa divisão se já encontraste uma cadeira?*

Para responder a esta pergunta, tomámos particular atenção ao seguinte excerto do enunciado do projeto:

> *A existência de uma cama, torna mais provável que exista um livro na mesma divisão. O mesmo se passa com a existência de uma cadeira: aumenta a probabilidade de existirem livros na mesma divisão.*

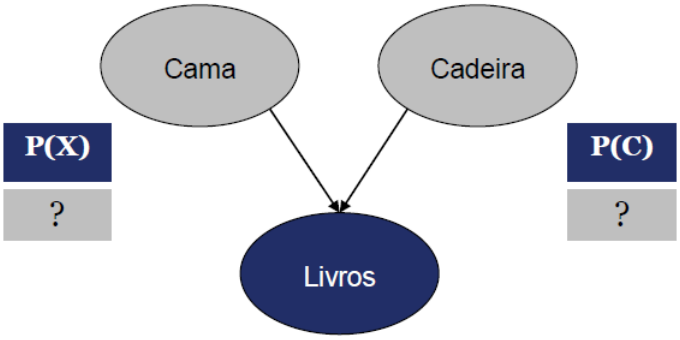
Tal significa que os eventos **não são independentes**, o que invalida por princípio a aplicação direta de uma probabilidade condicionada. Neste sentido, optou-se por uma **probabilidade condicionada baseada numa Rede *Bayesiana*** (Figura 3.4).

Seja então:

- L: Livro
- C: Cadeira
- X: Cama

Com base nesta Rede *Bayesiana*, sabemos que a probabilidade procurada é a seguinte:

$$P(L|C) = \frac{P(L,C)}{P(C)} \tag{3.4}$$



Cama (X)	Cadeira (C)	P(Livro| Cama, Cadeira) P(L|X,C)
V	V	P(L|X,C)
F	V	P(L|¬X,C)

Figura 3.4: Estimativa da velocidade do *robot*: função $v(t)$
Estimativa da velocidade do *robot*: função $v(t)$.

A probabilidade $P(C)$ pode ser determinada diretamente pela divisão entre o número de salas com cadeiras e o número total de salas (que não corredores). Contudo, a probabilidade $P(L, C)$ é dada pela equação (3.5).

$$P(L, C) = \sum_{X \in \{V,F\}} P(C, X, L) = P(C, X, L) + P(C, \neg X, L) \tag{3.5}$$

Por conseguinte, as duas probabilidades necessárias são dadas pela respetiva marginalização:

- $P(C, X, L) = P(X)P(C)P(L|X, C)$;
- $P(C, \neg X, L) = P(\neg X)P(C)P(L|\neg X, C)$.

Onde $P(\neg X) = 1 - P(X)$.
Por fim, temos que:

- $P(L|X, C) = \frac{P(L \wedge X \wedge C)}{P(X \wedge C)}$;
- $P(L|\neg X, C) = \frac{P(L \wedge \neg X \wedge C)}{P(\neg X \wedge C)}$.

Estas últimas podem ser determinadas diretamente contabilizando os números das salas totais e das salas que cumprem cada uma das condições pretendidas (por exemplo, $P(X \wedge C) = (\text{salas com X e C})/(\text{total salas})$).

O método `getProbabilityOfBookIfChairFound()` da classe *Hospital* realiza esta série de cálculos a fim de encontrar uma solução à equação (3.5).

3.3.8 Pergunta 8

> **Enunciado:** *Se encontrares um enfermeiro numa divisão, qual é a probabilidade de estar lá um doente?*

Por falta de indicação em contrário no enunciado do projeto, consideramos que estes são eventos **independentes**, pelo que pode ser aplicada a **probabilidade condicionada**.

O método `getProbabilityOfPatientKnowingNurses()` da classe *Hospital* realiza este cálculo, respondendo à equação (3.6):

$$P(D|E) = \frac{P(D \wedge E)}{P(E)} \tag{3.6}$$

Onde:

- $P(D)$: probabilidade de um doente estar numa sala;
- $P(E)$: probabilidade de um enfermeiro estar numa sala.

Tendo em conta que, por exemplo, $P(E) = (\text{salas com enfermeiros})/(\text{total salas})$, o cálculo destas probabilidades é direto, sendo por fim retornado o resultado da divisão da equação (3.6).

3.4 Conclusões

No presente capítulo foram apresentados os passos e os métodos necessários ao desenvolvimento do projeto prático e do funcionamento do mesmo.

Desta forma, através do conteúdo exposto neste capítulo, encontra-se a apresentação do projeto desenvolvido e o funcionamento do mesmo, mas também a contextualização das ferramentas utilizadas conforme listadas no Capítulo 2.

Capítulo 4

Reflexão Crítica e Problemas Encontrados

4.1 Introdução

Neste Capítulo são explorados os seguintes tópicos:

- Objetivos propostos vs. alcançados (Secção 4.2): compara os objetivos inicialmente propostos com aqueles que foram concluídos no projeto prático;
- Divisão de trabalho pelos elementos do grupo (Secção 4.3): lista as tarefas divididas por cada elemento constituinte.
- Problemas encontrados (Secção 4.4): na sequência da Secção 4.2, explora os problemas encontrados durante a implementação dos algoritmos;
- Reflexão crítica (Secção 4.5): é feita uma SWOT em retrospetiva pelo grupo acerca do projeto.

4.2 Objetivos Propostos vs. Alcançados

A Tabela 4.1 expõe os objetivos propostos inicialmente para o projeto e identifica quais foram alcançados totalmente ou parcialmente, e quais não foram bem sucedidos.

4.3 Divisão de Tarefas

Para a gestão e divisão das tarefas que delineiam o projeto, foi realizada uma reunião inicial onde o foco incidiu na definição das metas por cada membro do

Objetivo proposto	**Alcançado?**
Pergunta 1	•
Pergunta 2	•
Pergunta 3	•
Pergunta 4	•
Pergunta 5	•
Pergunta 6	•
Pergunta 7	•
Pergunta 8	•
Documentação do código	•

Tabela 4.1: Objetivos propostos e respetiva indicação de sucesso.
Legenda. • Totalmente alcançado; ○ Parcialmente alcançado. – Não alcançado.

grupo de forma balanceada. A referida gestão é apresentada na tabela 4.2. A escolha baseou-se no equilíbrio da dificuldade e temas das questões.

Não obstante, apesar da divisão de tarefas, houve uma essencial cooperação inicial entre os membros para definição das estruturas de dados e classes que se consideraram necessárias para a consolidação do projeto. Naturalmente houve de igual forma colaboração pontual no esclarecimento de dúvidas entre os membros.

A concretização do projeto foi conseguida com a realização de reuniões semanais, com *deadlines* bem definidas para cada tarefa, tendo igualmente sido realizadas algumas reuniões extraordinárias em momentos críticos do desenvolvimento.

Por fim, o relatório e a apresentação foram divididos pelas diferentes partes que cada membro implementou, assim como pelos restantes Capítulos adicionais. A revisão final do relatório foi, contudo, conjunta a fim de garantir a sua coerência.

4.4 Problemas Encontrados

A implementação dos diferentes algoritmos necessários para responder às diversas perguntas colocadas levou a que fossem encontrados alguns problemas, os quais tiveram de ser ultrapassados a fim de terminar o projeto prático. Os problemas mais notáveis são resumidos na Tabela 4.3, incluindo as soluções encontradas para os ultrapassar.

De notar que, já na fase final do projeto, percebemos que existe a possibilidade de as relações entre a bateria e a velocidade com o tempo serem funções exponenciais. A sua implementação não seria complexa uma vez que a derivada e a primitiva de uma função exponencial do tipo ke^x é a própria função exponencial (propriedade importante para a pergunta 5 (secção 3.3.5)):

Tarefas	**Beatriz Costa**	**Igor Nunes**
Pergunta 1	•	
Pergunta 2		•
Pergunta 3	•	
Pergunta 4		•
Pergunta 5	•	
Pergunta 6		•
Pergunta 7	•	
Pergunta 8		•
Relatório	•	•
Apresentação	•	•

Tabela 4.2: Distribuição de tarefas pelos elementos do grupo.

Problema	**Solução**
Ineficiência do algoritmo A* com recurso a uma matriz 800×600	Implementação de dois grafos no lugar da matriz
Determinação da relação entre a bateria, velocidade e tempo	Utilização de funções lineares para as respetivas estimativas
Localização imprecisa dos objetos	Determinar a direção do *robot* de forma a estimar a localização real do objeto

Tabela 4.3: Problemas encontrados durante o desenvolvimento do projeto e respetivas soluções.

4.5 Reflexão Crítica

É proposto expor a reflexão crítica face ao trabalho realizado para o desenvolvimento deste projeto através de uma análise SWOT.

4.5.1 Pontos Fortes

1. Forte estruturação dos tipos de dados com recurso a classes;
2. Utilização dos grafos para aumento significativo da eficiência;
3. Implementação de um método altamente eficiente para estimar a duração da bateria e prever a velocidade.

4.5.2 Pontos Fracos

1. O código-fonte final, incluindo documentação, é denso e de difícil navegação;
2. As funções lineares que relacionam a bateria e a velocidade com o tempo não representam fielmente a variação que ocorre ao longo do tempo;
3. Várias funções não foram otimizadas a fim de melhorar o consumo de recursos e/ou de aumentar a eficiência temporal.

4.5.3 Ameaças

1. Em efeitos de expansão do número de *robots* a circular, as classes com métodos estáticos não permitem a criação de instâncias independentes para cada *robot*, ficando assim um piso limitado apenas a um *robot*;
2. O uso de grafos não permite obter o caminho exato que o *robot* deve fazer pelo mundo até determinada sala, sendo por conseguinte imprecisa a distância calculada para lá chegar;
3. O método de estimação da bateria do *robot* é impreciso uma vez que o comportamento real destas funções não se revela linear.

4.5.4 Oportunidades

1. Reformular a organização do código de forma a permitir a divisão de ficheiros por classes;

2. Estudar melhor a variação da bateria e da velocidade em relação ao tempo a fim de auferir se se tratam de funções exponenciais;

3. Estudar os melhores métodos de otimização de cada função a fim de evitar processos potencialmente redundantes.

4.6 Conclusões

Esta fase de reflexão permitiu analisar o trabalho levado ao longo das semanas de planeamento e implementação dos algoritmos. Com esta análise, o grupo pôde tirar conclusões acerca das estratégias utilizadas, as quais serão expostas no Capítulo seguinte.

Capítulo 5

Conclusões e Trabalho Futuro

5.1 Conclusões Principais

Este projeto permitiu-nos adquirir um melhor conhecimento acerca da linguagem *Python*, como também nos permitiu evoluir em temas recorrentes na Unidade Curricular de Inteligência Artificial. Além disso, facilitou imenso o estudo da parte teórica desta cadeira, principalmente na parte dos algoritmos de pesquisa de caminhos e nas probabilidades relacionadas com a construção de redes *Bayesianas*.

Este projeto revelou como a IA tem atualmente aplicações interessantes na área dos *robots* domésticos de limpeza, os quais têm tido uma procura crescente no mercado.

5.2 Trabalho Futuro

Numa perspetiva futura, a nossa abordagem perante o projeto passaria primariamente por separar as classes em vários ficheiros para efeitos de organização do código, e por procurar eliminar as imprecisões na parte da estimação da bateria do dispositivo robótico.

Bibliografia

[1] N. developers, “NetworkX NetworkX documentation,” 2020, [Online] https://networkx.org/. Último acesso a 3 de janeiro de 2021.

[2] P. S. Foundation, “Welcome to Python.org,” 2021, [Online] https://www.python.org/. Último acesso a 21 de dezembro de 2020.