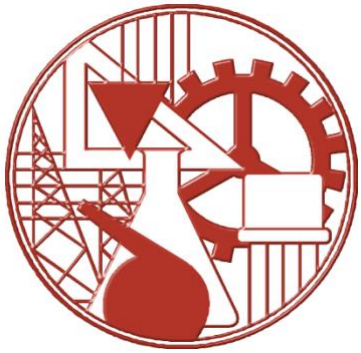
INSTITUTO POLITÉCNICO DE LISBOA
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de
Eletrónica e Telecomunicações e de Computadores



Biblioteca de Navegação Autónoma Baseada em Campos de Potencial

Filipe Ventura

(Licenciado)

Trabalho Final de Mestrado para obtenção do Grau de Mestre em Engenharia Informática e de Computadores

Júri:

Presidente: Doutor Hélder Jorge Pinheiro Pita, ISEL

Vogal – Arguente: Mestre Paulo Alexandre Medeiros de Araújo, ISEL

Vogal – Orientador: Doutor Luís Filipe Graça Morgado, ISEL

Novembro de 2013

Resumo

No contexto da inteligência artificial tem havido uma crescente ênfase no desenvolvimento de sistemas autónomos em particular para aplicações em ambientes reais, por exemplo sistemas robóticos, e para aplicações em ambientes virtuais, por exemplo jogos de computador.

Dependendo das tarefas às quais um sistema autónomo se destina pode surgir a necessidade de contemplar deslocações entre dois pontos. Esta capacidade é denominada de navegação autónoma e tem sido um tema bastante abordado no âmbito da inteligência artificial, na medida em que é uma das caraterísticas que mais contribui para a autonomia de uma solução.

Uma das opções para abordar o problema da navegação autónoma é a utilização de métodos baseados em campos de potencial. A principal caraterística desta abordagem consiste na atribuição de um valor de potencial aos elementos do ambiente e este ser propagado pelo ambiente gerando um campo de potencial. Este campo de potencial posteriormente possibilita a orientação do sistema autónomo no sentido de atingir os objetivos especificados.

Com este projeto pretende-se o desenvolvimento de um conjunto de métodos que sigam esta abordagem e a realização de testes que evidenciem as vantagens e desvantagens de cada método. O resultado deste trabalho será a disponibilização de uma biblioteca de métodos de navegação autónoma baseados em campos de potencial.

Palavras-chave: sistemas autónomos, navegação autónoma, campos de potencial, potencial, ambiente, agente

Abstract

In the context of artificial intelligence, the development of autonomous systems in particular for applications in real environments, e.g. robotic systems, and for applications in virtual environments, e.g. computer games, has been growing over the years.

Depending on the tasks that can be assigned to an autonomous system it could be necessary perform displacements between two points. This ability is called autonomous navigation and has been studied in the context of artificial intelligence and it is one of the features that most contributes to the autonomy of a solution.

One option to address the problem of autonomous navigation is the use of methods based on potential fields. The main characteristic of this approach consists in assigning a potential value to the elements of the environment and generating a potential field by propagating the potential over the environment. This potential field allows autonomous system achieve specified objectives.

This project aims to develop a set of methods that follow this approach and testing it. In the end it could be possible verify the advantages and disadvantages of each method. The result of this work is to deliver a library of autonomous navigation methods based on potential fields.

Keywords: autonomous solutions, autonomous navigation, potential fields, potential, environment, agent

Agradecimentos

Foram várias as pessoas que me acompanharam ao longo desta caminhada, desde o meu primeiro dia de "caloiro" até ao dia de hoje e que contribuíram de forma inquestionável para o meu crescimento pessoal e profissional.

Começo assim por agradecer a todos os meus colegas e amigos que trilharam este caminho que nem sempre foi uma linha recta. Ficam recordados os momentos de pressão mas também as vitórias alcançadas e as lições aprendidas.

Pela concretização deste trabalho quero expressar o meu sinal de apreço e agradecimento ao meu orientador, Professor Luís Morgado. Pelas segundas oportunidades que me proporcionou e pela dedicação e conselhos que me passou ao longo destes dois anos de trabalho. Acredito que muitas vezes foi quase impossível vislumbrar a finalização deste trabalho.

Quero também agradecer ao meu irmão. Tenho a noção da disparidade dos nossos pontos de vista em muitos temas mas sei que estás disponível sempre que for preciso. Fica este trabalho, também como o mote para que também tu não desistas e que persistas no cumprimento dos teus objetivos.

Por fim quero agradecer a quem foi mais importante, aos meus pais. Durante estes anos sempre se colocaram em último para que fosse possível oferecer aos filhos a perspetiva de um futuro risonho e melhor que o seu presente. Espero acima de tudo que sintam orgulho e um sentido de dever cumprido agora que chega ao término esta etapa da minha vida que ao mesmo tempo é também a sua.

Índice Geral

Índice de figuras

Índice de diagramas

1. Introdução

O desenvolvimento de sistemas autónomos tem ganho nos últimos anos um maior interesse devido às suas utilizações. Exemplos de soluções deste género que surgiram são o *Google Driverless Car* (1), *robots* criados para a realização de tarefas diárias (2) e *robots* para utilização em fábricas (3).

Transversalmente existem associadas a este tipo de sistemas diversas problemáticas, sendo a navegação autónoma uma delas. A sua resolução implica o desenvolvimento de sistemas capazes de se deslocarem entre dois pontos sem intervenção humana, ou seja, sistemas capazes de navegar autonomamente.

Neste tipo de sistemas deve existir uma forma de verificar o seu grau de sucesso e a sua viabilidade de utilização em ambientes reais. Para isso são normalmente utilizadas como métricas o tempo de resposta, a distância percorrida ou o número de passos executados entre o ponto inicial e o ponto final.

As soluções desenvolvidas para abordar o problema da navegação autónoma podem ser divididas em três grupos (4):

- Planeamento global do percurso;
- Métodos de navegação local;
- Métodos híbridos.

No primeiro grupo incluem-se algoritmos que realizam um planeamento do percurso entre os dois pontos e só posteriormente realizam a navegação planeada. O percurso gerado evita todos os obstáculos, mas caso seja necessário recalcular o percurso, como pode acontecer em ambientes dinâmicos, o desempenho do método degrada-se.

No segundo grupo incluem-se métodos que não necessitam de conhecer o ambiente previamente, baseando a navegação na informação local percecionada pelos sensores. Desta forma o agente possui uma maior reatividade ao ambiente sendo passível a sua utilização em ambientes dinâmicos. No entanto, os métodos de navegação local apresentam problemas com a existência de ótimos locais que invalidam a navegação.

Por fim os métodos híbridos tentam conjugar as vantagens dos grupos mencionados acima e eliminar as suas limitações. Pode ser utilizado um método de planeamento para a obtenção de um plano global sendo a navegação detalhada realizada através de um método de navegação

local. Apesar desta conjugação, esta abordagem pode também apresentar limitações ao nível do seu desempenho, caso se considere, a necessidade de replanear percursos.

O presente trabalho tem por âmbito os métodos baseados em campos de potencial, os quais pertencem ao segundo grupo. Esta opção prende-se com a capacidade de fornecer ao agente mecanismos de navegação autónoma associados a baixos consumos de recursos computacionais.

1.1. Motivação

A procura de sistemas inteligentes com capacidade de navegar autonomamente levou investigadores de diversas áreas a estudar e a desenvolver soluções capazes de resolver os problemas que enfrentam. Um exemplo é a indústria dos jogos que pretende capacitar as personagens existentes nos jogos com mais inteligência (5), (6), (7).

Atualmente encontra-se disponibilizado um conjunto de soluções que pode ser utilizado para a resolução da problemática da navegação autónoma. As caraterísticas existentes no ambiente são os principais fatores que parametrizam a decisão da solução. No extremo, este ponto pode levar a casos em que uma solução com um bom desempenho num ambiente A pode não ser a melhor opção para um ambiente B.

No âmbito da procura de soluções capazes de superar esta problemática, durante cerca de três décadas foi desenvolvido um conjunto de algoritmos que acabaria por ser designado como métodos baseados em campos de potencial.

Com a utilização de métodos baseados em campos de potencial pretende-se o desenvolvimento de soluções simples e com um baixo consumo de recursos computacionais, sendo que a maior contribuição vem da não necessidade de calcular todo o percurso no início da navegação mas sim reagir de uma forma adequada aos estímulos gerados pelos elementos presentes no ambiente e que surgem ao longo do percurso.

Tendo em conta estas características a utilização desta família de métodos pode permitir o cumprimento de duas características importantes para a navegação autónoma, a geração automática de percursos e um baixo consumo de recursos.

1.2. Objetivos

No presente trabalho pretende-se o desenvolvimento de uma biblioteca de métodos de navegação autónoma baseados em campos de potencial e de uma plataforma de teste de forma a validar a sua execução.

Em relação à biblioteca de métodos de navegação autónoma baseados em campos de potencial pretende-se a implementação de algoritmos baseados em campos de potencial previamente selecionados. Este conjunto de implementações reflete concretizações já documentadas e devidamente testadas.

Em relação à plataforma de simulação, inclui-se o desenho e implementação de uma plataforma que deverá ser o mais genérica possível, sendo que deverá ser cumprido um conjunto de requisitos do ponto de vista da engenheira de *software* de forma a tornar possível a realização de testes para os diferentes métodos presentes na biblioteca.

1.3. Organização do documento

Capítulo 1 – Introdução	Capítulo introdutório acerca da navegação autónoma onde é também apresentada a motivação e os objetivos deste trabalho.
Capítulo 2 – Navegação autónoma baseada em campos de potencial	Capítulo introdutório dos principais conceitos associados ao desenvolvimento de métodos baseados em campos de potencial, os seus problemas e possíveis soluções.
Capítulo 3 – Métodos de navegação com base em campos de potencial	Apresentação de trabalhos desenvolvidos por outros autores e que são utilizados como base para os métodos de navegação autónoma com base em campos de potencial desenvolvidos neste trabalho.
Capítulo 4 – Biblioteca de métodos de navegação autónoma baseados em campos de potencial	Descrição do enquadramento geral da solução e descrição dos principais aspetos referentes à génese do desenvolvimento da biblioteca de método de navegação.

Capítulo 5 – Plataforma de teste	Descrição dos principais pontos associados ao desenvolvimento da plataforma de testes para validação da execução dos métodos de navegação autónoma desenvolvidos.
Capítulo 6 – Resultados experimentais	Capítulo dedicado à apresentação dos resultados obtidos e análise dos mesmos tendo em conta as expectativas e características associadas a cada método de navegação autónoma.
Capítulo 7 – Conclusão	Apresentação das conclusões retiradas do presente trabalho e análise dos pontos que futuramente podem ser contemplados no seu seguimento.

2. Navegação autónoma baseada em campos de potencial

A navegação autónoma baseada em campos de potencial tem associada a si um conjunto de conceitos base os quais serão apresentados seguidamente. A sua definição pretende facilitar a compreensão dos pontos transversais às abordagens que serão posteriormente apresentadas.

2.1. Conceito de agente

Um agente é uma entidade autónoma presente no ambiente, sendo este capaz de navegar até ao objetivo. Na literatura *Maes* (8), define um agente da seguinte forma.

"Um agente é um sistema que tenta cumprir um conjunto de objetivos e se encontra inserido num ambiente complexo e dinâmico sendo que o perceciona através de sensores e atua sobre ele através de atuadores." (8)

Considerando o contexto deste trabalho, a navegação baseada em campos de potencial, o agente pode também ser definido como um sistema sob o efeito de um ou mais campos de potencial.

Uma característica importante associada à noção de agente é a noção de autonomia que na literatura *Maes* define da seguinte forma.

"Um agente é designado de autónomo se a sua operação for completamente autónoma, ou seja, se decidir por si próprio como relacionar a informação obtida através de sensores de forma a cumprir os objetivos" (8)

Estes objetivos são cumpridos através da geração de ações, sendo estas parametrizadas pelos estímulos percecionados pelo agente através dos vários sensores. Exemplos de ações são "Mover" ou "Agarrar". A primeira é responsável pela deslocação do agente ao longo do ambiente e a segunda só surge caso o agente se encontro próximo do objetivo.

2.2. A noção de campo de potencial

2.2.1. Potenciais e forças

Os campos de potencial são gerados pela propagação do potencial associado aos elementos existentes no ambiente. Esta dispersão leva a que cada ponto do ambiente tenha associado um valor de potencial resultante da soma das intensidades dos campos de potencial que afetam esse ponto. Desta forma é possível definir um campo de potencial como um campo escalar (Figura 1), ou seja, através de associação entre um valor escalar a cada ponto do espaço (9).

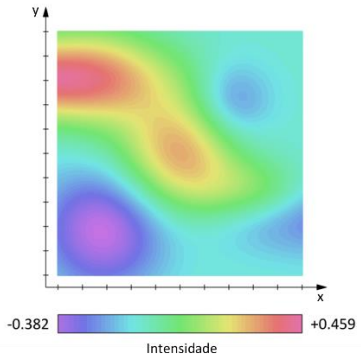


Figura 1 – Exemplo de um campo escalar *in* (10)

O potencial gerador destes campos permite caraterizar os elementos existentes no ambiente através da atribuição de potenciais diferenciados tendo em conta a natureza do elemento, através de um valor escalar. Daqui pode-se denominar este potencial como potencial escalar.

Através da caraterização dos elementos é possível relacionar cada uma das posições e derivar outras características como a diferença de potencial. Tendo em conta que o potencial aumenta quanto mais próxima da origem do potencial uma posição se encontrar. Uma forma de definir esta variação é através de uma direção e uma magnitude. A utilização destes dois conceitos permite a definição de um campo vetorial (Figura 2).

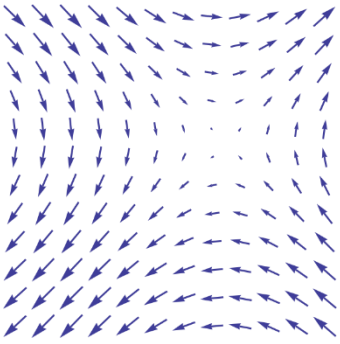


Figura 2 – Exemplo de um campo vetorial in (11)

Considerando o vetor associado a cada posição do ambiente é possível perceber que existem pontos em que o vetor associado é igual. A conjunção destes pontos geram uma superfície equipotencial e é o conjunto de várias superfícies equipotenciais desde a origem do potencial até às posições em que o potencial é zero que formam os campos de potencial (Figura 3).

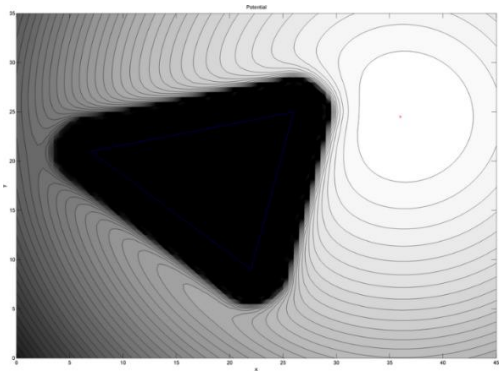


Figura 3 – Superfícies equipotenciais que definem um campo de campo de potencial tendo em conta um obstáculo *in* (12)

Considerando a existência de um agente no ambiente, o qual pretende navegar a partir de um qualquer ponto até a uma posição que seja considerada como objetivo, deve utilizar os campos de potencial gerados pelos restantes elementos presentes no ambiente através dos vetores de gradiente associados a cada posição.

Portanto, o campo de potencial exerce um efeito sobre o agente e que o orienta ao longo do ambiente desde o ponto inicial até ao ponto final da sua navegação. Este efeito pode ser definido como uma força exercida pelo campo de potencial sob o agente sendo que a sua representação pode consistir num vetor composto por duas componentes, magnitude e direção.

Em última instância, a força mencionada acima pode ser definida como a capacidade que um campo de potencial tem em exercer o seu efeito repulsivo ou atrativo sobre um agente.

A relação entre a força gerada e um campo de potencial permite a classificação deste em termos do seu tipo. Dentro desta classificação é possível identificar dois tipos de campo de potencial base, atrativos e repulsivos. Visualmente um campo de potencial atrativo pode ser representado tal como apresentado na Figura 4.

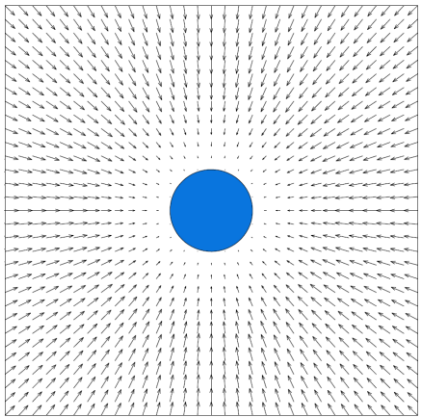


Figura 4 – Campo de potencial atrativo *in* (13)

As setas apresentadas determinam a direção do campo sendo este mais forte quanto mais perto do centro o agente se encontrar. Sendo assim um agente dentro desta área apresenta um comportamento que o leva a aproximar-se do elemento representado no centro da figura acima.

No caso de um campo de potencial repulsivo a representação é diferente (Figura 5):

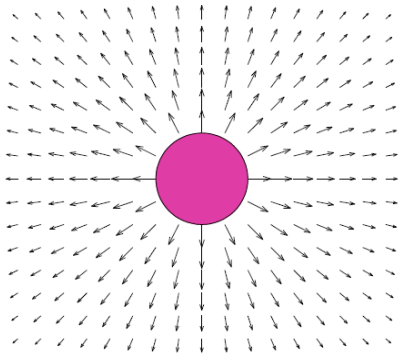


Figura 5 – Campo de potencial repulsivo *in* (13)

Como é possível observar, a direção da força exercida pelo campo de potencial associado ao elemento, leva a que um agente sob o efeito deste campo de potencial se afaste do elemento representado no centro da figura.

2.2.2. Função de campo de potencial

Como indicado anteriormente existe associada a cada posição uma força que resulta do somatório das forças exercidas pelos vários campos de potencial que exercem um efeito sob essa posição. O cálculo deste vetor é realizado matematicamente através de uma função denominada de função de campo de potencial, a qual define um mapeamento entre um valor escalar e uma coordenada possível num espaço (14), ou seja, uma função de campo de potencial f pode ser definida genericamente da seguinte forma.

$$f: \mathbb{R}^n \rightarrow \mathbb{R}. \qquad (1)$$

É possível obter a força resultante exercida pelos campos de potencial associados aos elementos presentes no ambiente para uma determinada posição através de uma função de campo de potencial definida, por exemplo, com base na distância dessa posição à origem do potencial. Esta situação é ilustrada na figura seguinte (Figura 6).

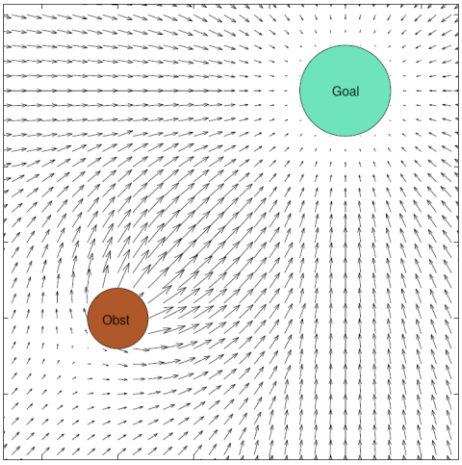


Figura 6 – Conjugação de um potencial atrativo e um potencial repulsivo *in* (13)

2.2.3. Perfil de magnitude

A variação da componente magnitude dos vetores que representam a força exercida por um campo de potencial é designada de perfil de magnitude. Esta variação pode ser dependente da distância a que o agente se encontra do ponto objetivo e dos obstáculos que se encontram dentro do seu alcance.

Um exemplo da importância da definição dos perfis de magnitude passa pela forma como o agente se desloca ao longo do ambiente. Considerando um campo de potencial repulsivo como o exemplificado na Figura 5, os vetores de força poderiam ter uma componente de direção que afaste o agente e uma componente de magnitude constante independente da distância a que o

agente se encontre da origem do potencial. Este tipo de perfil é denominado de perfil de constante magnitude (15), sendo a sua representação a seguinte.

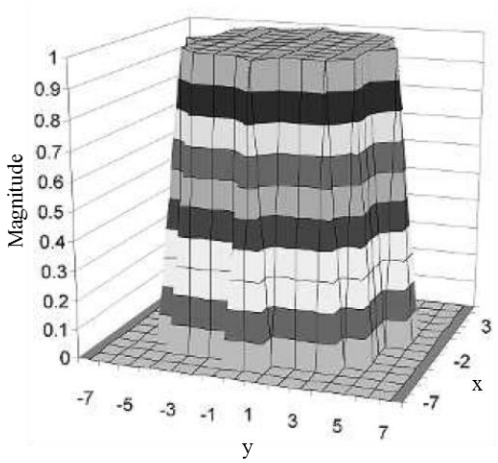


Figura 7 – Perfil de magnitude constante *in* (15)

Tendo em conta este perfil o agente irá deslocar-se a uma velocidade constante enquanto se encontrar sob efeito do campo de potencial ao qual o perfil se encontra associado, mas iria parar bruscamente assim que estivesse fora da área pela qual o campo de potencial se propaga.

Um comportamento mais próximo daquele que se pretende observar em ambientes reais, seria o agente, à medida que se afasta da origem do campo de potencial, diminuir a sua velocidade. Ou seja, o agente responderia duma forma proporcional aos estímulos que recebe do ambiente, conceito esse designado de reflexividade.

Tendo em conta o comportamento que se pretende, *Murphy* (15) indica dois perfis possíveis.

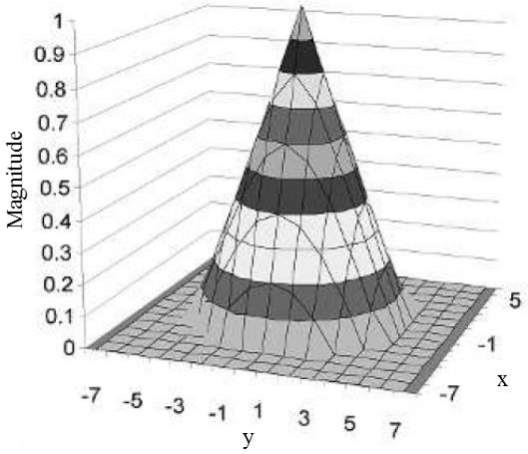


Figura 8 – Perfil de decaimento linear *in* (15)

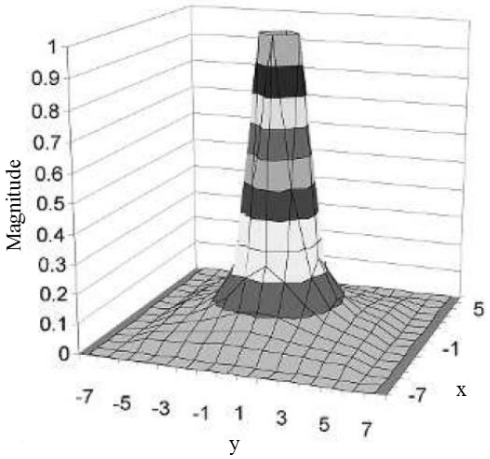


Figura 9 – Perfil de decaimento exponencial *in* (15)

Ao analisar estas duas possibilidades compreende-se que quanto mais o agente se aproxima da origem do campo de potencial, maior será o seu grau de reação, sendo este refletido, por exemplo, na velocidade que o agente demonstra. A diferença entre estes perfis encontra-se na forma como o agente reage ao estímulo, já que no primeiro caso a reação é proporcional ao longo do tempo e no segundo caso o agente pode reagir de forma abrupta.

Independentemente do perfil escolhido, o principal aspeto a reter recai no comportamento do agente. Graças ao perfil é possível definir um agente com reações mais ou menos abruptas dependendo, por exemplo, do tipo de elemento que o agente detete ao longo do seu percurso.

Na prática, tendo em conta os dois perfis apresentados, podem ser ambos utilizados. Uma situação possível passa pelo agente aproximar-se dos objetivos duma forma proporcional, apresentando um perfil linear, e afastar-se dos obstáculos duma forma mais abrupta, apresentando um perfil exponencial.

2.3. Navegação com base no gradiente do campo

Na secção anterior foi apresentado um conjunto de noções transversais à caraterização de métodos de navegação com base em campos de potencial. Duma forma geral o ambiente no qual o agente opera é modelado por campos de potencial gerados pelos potenciais associados aos elementos existentes no ambiente. A força exercida por estes campos influenciam a navegação do agente a partir do ponto inicial até ao objetivo.

A navegação com base no gradiente do campo (método do gradiente) surge como uma forma de utilizar as características apresentadas pelos campos de potencial. A principal característica a realçar é a possibilidade de, através de um campo de potencial, definir forças que contribuem para a definição da ação a realizar por parte do agente, sendo esta contribuição maior quanto mais próximo este se encontra da origem do campo de potencial.

Este aspeto permite a definição de um gradiente (Figura 10) composto por vetores com a direção do gradiente do campo de potencial e com uma magnitude decrescente em função da distância do agente ao centro do campo de potencial.

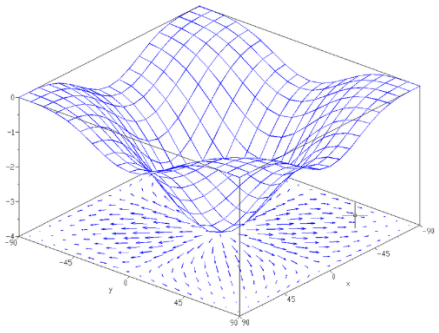


Figura 10 – Representação gráfica de um gradiente associado a um campo de potencial *in* (9)

A noção de gradiente pode ser definida como um mapeamento entre cada coordenada possível pertencente a um espaço *n* dimensional com um vetor (14).

De forma a permitir a navegação do agente é necessário que o agente se afaste de obstáculos e se aproxime dos objetivos. De forma a cumprir este ponto é necessário associar aos elementos existentes no ambiente potenciais de sinal inverso. Desta forma o gradiente, no caso dos obstáculos, é composto por um conjunto de vetores com uma direção oposta ao centro do campo de potencial.

O agente encontra-se desta forma sob efeito de dois tipos de forças, repulsivas e atrativas. Deste somatório resulta uma força resultante definida formalmente da seguinte forma.

$$F_{res} = \sum F_{act} + \sum F_{rep} \qquad (2)$$

Em que F_{res} representa a força resultante, $\sum F_{act}$ representa o somatório das forças atrativas e $\sum F_{rep}$ representa o somatório das forças repulsivas.

Esta força é também ela definida por uma direção e uma magnitude, que posteriormente irá determinar a ação tomada por parte do agente (Figura 11).

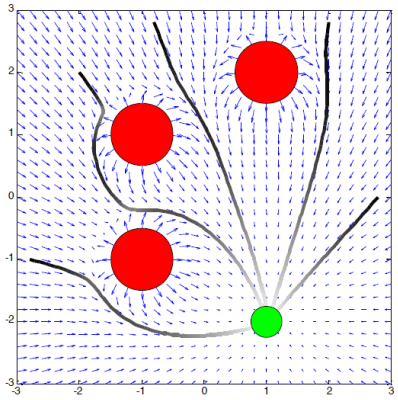


Figura 11 – Somatório de campos repulsivos com um campo atrativo. As linhas representadas a preto permitem observar os vários percursos que o agente pode tomar dependendo do ponto inicial *in* (16)

Partindo destas premissas foram desenvolvidas diversas técnicas de navegação autónoma sendo que um conjunto destas será abordado mais em rigor posteriormente neste trabalho.

Numa primeira análise a utilização de um método de navegação baseado no gradiente parece ser suficiente para o sucesso da navegação de um agente entre dois pontos. O problema dessa utilização surge da possibilidade de encontrar pontos onde as forças repulsivas e atrativas se anulam ou a força resultante não é suficiente para promover a deslocação do agente para outras posições. As posições onde existe este equilíbrio são denominadas de ótimos locais e podem degenerar em zonas compostas por várias posições.

2.4. O problema dos ótimos locais

A principal desvantagem na utilização de métodos baseados em campos de potencial está relacionado com a existência de ótimos locais, sendo que a sua resolução não é trivial. Este tipo de configurações são normalmente caracterizadas por pontos ou zonas que impedem ou dificultam a navegação do agente. A representação de ótimos locais é ilustrada na Figura 12 e na Figura 13.

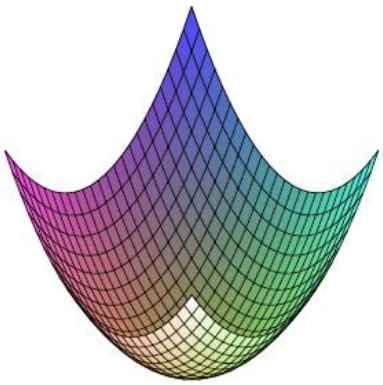


Figura 12 – Superfície demonstrativa da existência de um ótimo local *in* (14)

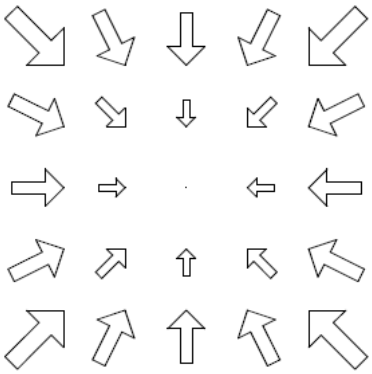


Figura 13 – Campo vetorial correspondente *in* (14)

Em ambas as figuras é possível verificar que a direção da força em qualquer ponto tem uma direção para um determinado ponto. Este tipo de representação seria aceitável se no centro de cada uma destas representações residisse o ponto caraterizado pelo potencial mais favorável, ou seja, uma posição objetivo. O problema surge quando do somatório das forças exercidas pelos campos de potencial resulta uma força nula ou que leva o agente para uma posição que não corresponde ao potencial mais favorável. Um exemplo prático associado a este tipo de situações é apresentado em seguida (Figura 14).

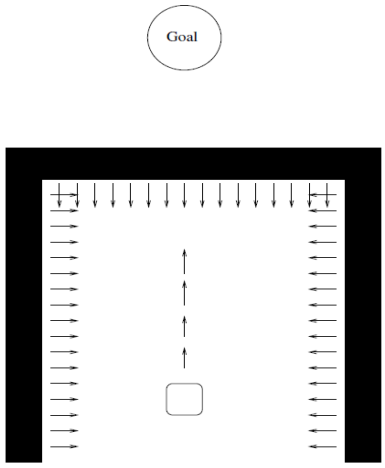


Figura 14 – Beco, normalmente conhecido por problema *Box Canyon in* (13)

Neste tipo de configuração o agente desloca-se ao longo do beco atraído pelo campo de potencial gerado pelo elemento objetivo colocado após a parede existente ao fim do beco. Num determinado momento o agente estancará a sua navegação num ponto próximo da parede colocada ao fim do beco. Devido ao equilíbrio entre as forças exercidas pelos campos de potencial repulsivos gerados pelas paredes e a força exercida pelo campo de potencial atrativo gerado pelo alvo, o agente não conseguirá sair dessa zona.

Além desta situação existem outras que não permitem ao agente concluir com sucesso a navegação no sentido de atingir uma posição objetivo. Dependendo da sua configuração, estas situações podem levar a uma classificação com diferentes tipos de ótimos locais. São exemplos disso as seguintes designações (14):

- Ravinas, nos quais o agente não consegue seguir o potencial mais favorável devido a ao efeito exercido por outros potenciais;
- Planaltos, uma área na qual as respetivas posições possuem um valor de potencial próximo;
- Pontos "sela", em que existem diversas direções com um valor de potencial próximo e sem direção definida.

2.4.1. Ravinas

Este tipo de ótimos locais, Figura 15 e Figura 16, pode não inviabilizar a navegação do agente mas leva à geração de percursos mais ineficientes. Conceptualmente são caracterizados por potenciais simétricos sendo que entre eles existe uma zona caracterizada por forças de valor inferior.

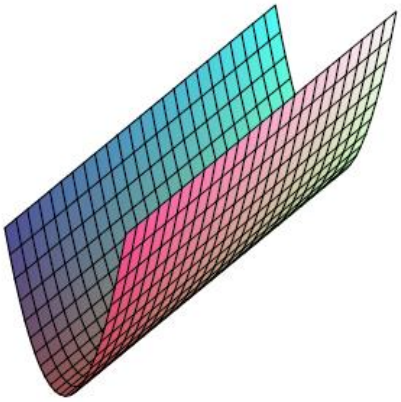


Figura 15 – Superfície demonstrativa da existência de um ótimo local do tipo ravina *in* (14)

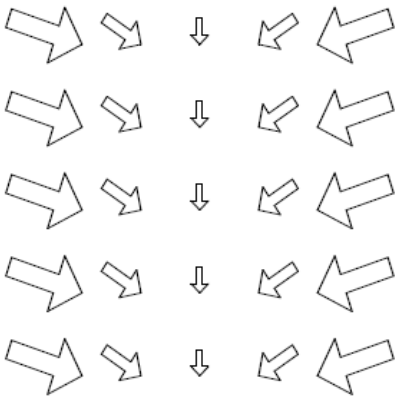


Figura 16 – Campo vetorial correspondente a uma ravina *in* (14)

Um dos exemplos práticos deste tipo de problema passa pela navegação do agente entre duas paredes, por exemplo um túnel. O percurso ótimo seria composto por deslocações retilíneas até ao fim dessa zona. O problema surge quando o agente deteta uma das paredes sendo corretamente afastado graças a uma força repulsiva mas que, ao ser exercida, vai empurrar o agente na direção da parede oposta, repetindo-se a situação. Desta forma o agente cria um percurso ineficiente (Figura 17).

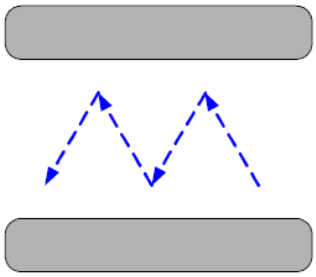


Figura 17 – Exemplo prático de ótimo local do tipo ravina

2.4.2. Planaltos

Este tipo de situações surge no momento em o potencial calculado para um conjunto de pontos é igual ou próximo entre si. A nível da navegação este problema faz-se sentir no tempo que o agente demora a percorrer uma região composta por posições com estas características, Figura 18 e Figura 19.

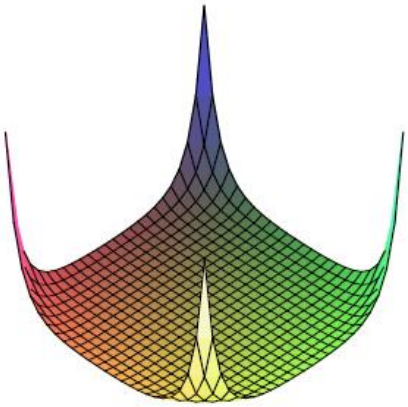


Figura 18 – Superfície demonstrativa da existência de um mínimo local do tipo planalto *in* (14)

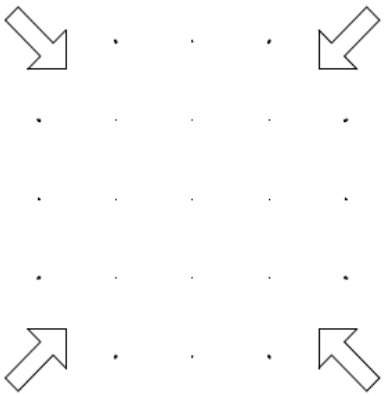


Figura 19 – Campo vetorial correspondente a um planalto *in* (14)

Esta situação pode ocorrer se o agente se encontrar numa zona onde a magnitude da força exercida não é suficiente para promover a deslocação do agente. Isto pode acontecer por exemplo em algoritmos que adicionam aleatoriedade ao deslocamento do agente para evitar ótimos locais, mas que pode em última instância afastar o agente para zonas onde o campo de potencial atrativo é demasiado fraco.

De notar que a mesma situação pode acontecer também para potenciais atrativos elevados. Podem existir pontos ou zonas associados a um valor de potencial atrativo elevado e que o agente ao deslocar-se entre elas apresenta o mesmo comportamento. Um exemplo deste tipo de situação sucede se o agente se encontrar a uma distância igual entre dois alvos (Figura 20).

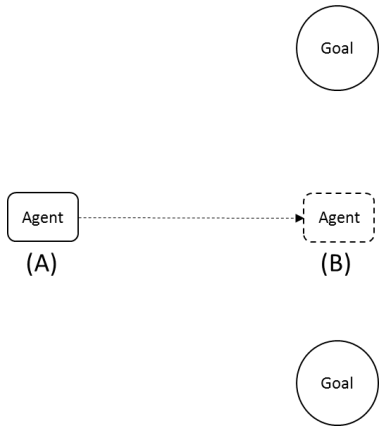


Figura 20 – Exemplo prático de um ótimo local do tipo planalto

Na situação apresentada acima a navegação do agente inicia-se no ponto A e desloca-se tendo em conta os campos de potencial até ao ponto B. A partir deste ponto o agente não consegue continuar. Esta ocorrência deve-se ao equilíbrio que existe entre as forças atrativas exercidas pelos campos de potencial associados aos alvos apresentados.

2.4.3. Selas

Este tipo de ótimos locais surgem quando as forças resultantes calculadas para um conjunto de pontos não permitem ao agente seguir uma direção de uma forma consistente. Devido à existência de vários percursos possíveis podem existir pontos em que a força resultante não permita ao agente sair de uma determinada zona ou posição. As Figura 21 e Figura 22 representam este tipo de ótimo local.

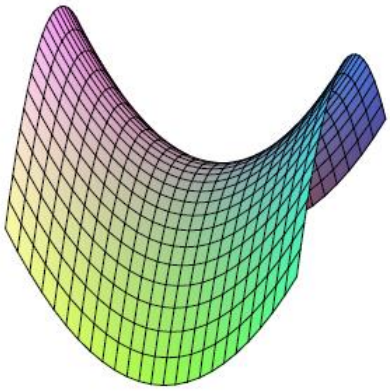


Figura 21 – Superfície demonstrativa da existência de um ótimo local do tipo sela *in* (14)

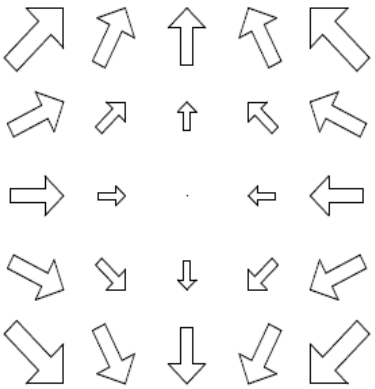


Figura 22 – Campo vetorial correspondente a uma sela *in* (14)

Como exemplo prático iremos considerar a Figura 23.

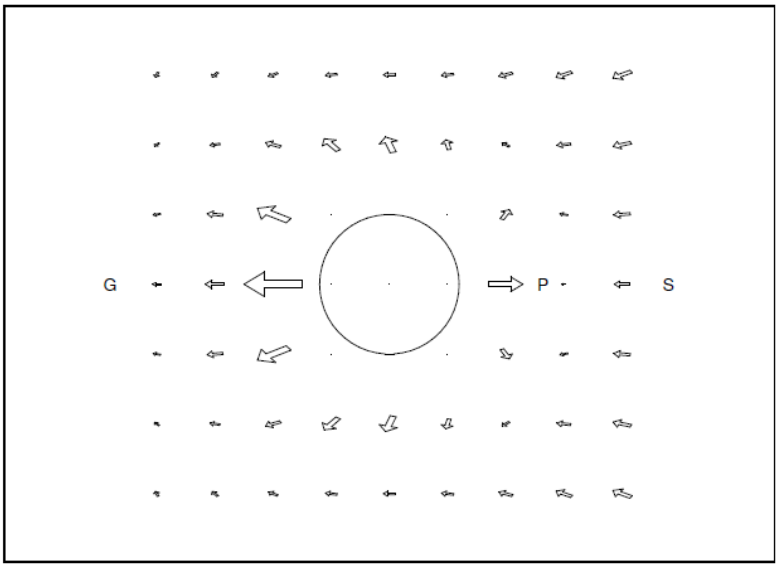


Figura 23 – Exemplo de um ponto sela, representado por P *in* (17)

O cenário proposto é o seguinte. Pretende-se que o agente se desloque do ponto S para o ponto G e existindo um obstáculo circular no centro da figura. Neste cenário as forças resultantes exercidas sob o agente irão promover deslocações até junto do ponto P. Neste ponto o agente

passa a estar sob o efeito do campo de potencial gerado pelo potencial associado ao obstáculo e que em caso de equilíbrio, tal como acontece nesta situação, origina a estagnação de movimentos por parte do agente.

2.5. Eliminação de ótimos locais

Para a resolução dos problemas associados aos ótimos locais uma das abordagens propostas passou pela criação de soluções capazes de eliminar do ambiente configurações que levam à sua existência.

A utilização de funções harmónicas é um dos exemplos mais referenciados dentro desta abordagem tendo sido *Connolly* um dos seus principais percursores (18), (19), (20).

A principal vantagem apresentada pela utilização de funções harmónicas diz respeito a "não permitirem a criação de ótimos locais numa região" (19). A única ocorrência de ótimos locais considerada nesta abordagem refere-se concretamente a pontos únicos do tipo sela e dos quais, segundo o autor, seria possível o agente deslocar-se desse ponto através da análise dos pontos vizinhos.

A aplicação de funções harmónicas levanta no entanto problemas, nomeadamente em ambientes dinâmicos, situação evidenciada por *Connolly* e *Burns* (19), visto não garantir que o agente consiga evitar os obstáculos. Isto resulta do facto do potencial ser calculado tendo em conta todos os obstáculos e alvos existentes no ambiente e não apenas os que se encontram dentro duma área delimitada e próxima do agente. Devido a este facto o cálculo pode ser demorado e durante esse tempo a configuração do ambiente pode alterar-se, pelo que não existe a sua contabilização no cálculo.

A resolução deste problema passa pela utilização de métodos numéricos que permitem a integração de um conjunto de condições, neste caso os limites dos elementos mais próximo da posição atual do agente.

Como foi referido esta abordagem tem como principal vantagem a eliminação dos ótimos locais. Apesar disso não é uma solução muito utilizada devido ao consumo de recursos computacionais que tem associado (21).

2.6. Deteção de ótimos locais

Tendo em conta que a eliminação de ótimos locais pode contribuir negativamente para o desempenho do agente, outra opção aceitável passa pela deteção dos ótimos locais. Desta forma o agente sabe que não se encontra no seu destino e pode atuar em conformidade. No trabalho apresentado por *Bell* (14) as formas de detetar ótimos locais são normalmente divididas em dois tipos:

- Matemáticas;
- Heurísticas.

Na primeira estratégia, o agente tenta através de funções matemáticas analisar o terreno e perceber se existe um ótimo local no que se encontra (22). *Bell* dá como exemplo a utilização da matriz *Hessiana* devido às propriedades que esta apresenta.

Caso o cálculo do seu determinante para uma determinada posição seja igual 0 é possível obter uma das seguintes conclusões:

- Se todos os valores próprios são negativos então o ponto é um máximo local;
- Se todos os valores próprios são positivos então o ponto é um mínimo local;
- Se existirem valores próprios positivos e negativos então trata-se de um ponto sela.

Apesar deste aspeto, esta solução pode não ser interessante tendo em conta que o agente pode não se encontrar num ponto em concreto mas sim numa zona próxima sendo que dessa forma se torna mais complexa a análise.

Em relação à deteção de ótimos locais baseada em funções heurísticas, esta é normalmente caraterizada pela utilização e parametrização de métricas. Por exemplo se a distância para o objetivo diminui ao longo do tempo, a velocidade ao longo do tempo ou se estas variáveis se encontram abaixo de um determinado valor tomado como referência.

Um exemplo de utilização deste tipo de abordagem pode ser encontrado no trabalho de *Juliá*, *Gil*, *Payá* e *Reinoso* (23) que numa primeira instância consideraram a existência de um ótimo local caso o somatório das várias forças exercidas se aproxime de zero. Numa segunda instância foi possível concluir que esta validação não permite cobrir todos os casos como por exemplo o agente navegar dentro de uma determinada zona.

Os autores propuseram assim uma segunda solução baseada no potencial associado aos pontos vizinhos. Esta abordagem tomada considera que o ambiente em redor do agente passa a ser representado de uma forma discreta, sendo calculada a força resultante sobre cada uma dessas zonas.

Assim o desempenho desta solução depende da área para qual é calculado o efeito dos campos de potencial. Quanto menor for a área mais rápido o cálculo é realizado.

Outro exemplo de utilização deste tipo de soluções pode ser encontrado no trabalho apresentado por *Park* e *Lee* (24). De um modo geral este trabalho contempla a adição de obstáculos virtuais caso o agente se encontre num ótimo local. De forma a verificar esta premissa o agente deteta os ótimos locais através do deslocamento observado num dado intervalo de tempo:

$$|x_c(t) - x_c(t - T_a)| \leq S_a, t \geq T_a \quad (3)$$

Sendo $x_c(t)$ a posição atual do agente, $x_c(t - T_a)$ uma posição anterior do agente distanciada por um intervalo de tempo T_a, e S_a a distância mínima que o agente deveria percorrer dentro de desse intervalo de tempo. Caso se verifique esta condição considera-se que o agente se encontra num ótimo local.

2.7. Fuga de ótimos locais

A partir dos exemplos anteriormente apresentados é possível observar que a deteção de ótimos locais por si só não é suficiente para a resolução de ótimos locais, visto ser necessário a realização de uma ação ou conjunto de ações para que o agente se afaste dessa zona. Algumas das hipóteses que podem ser utilizadas passam pela adição de aleatoriedade, adição de forças, preenchimento de ótimos locais ou realização de movimentos de retrocesso.

2.7.1. Adição de aleatoriedade

A adição de aleatoriedade após o agente detetar que se encontra num ótimo local é uma das abordagens mais utilizada. Este facto resulta normalmente da relativa simplicidade com que se implementam este tipo de soluções. Infelizmente os problemas que podem advir deste tipo de solução também resultam desta relativa simplicidade, sendo um exemplo o caso da não garantia de repetição ótimos locais.

Apesar deste aspeto, foram vários os autores que se dedicaram de forma a melhorar o desempenho dos algoritmos que podem ser incluídos neste tipo de abordagem. *Latombe* e *Barraquand* são dos autores mais referenciados e os seus trabalhos têm muitas vezes servido como base para outros.

Além destes, também *LaValle* (21) abordou soluções deste tipo sendo disso exemplo um sistema composto por três modos de funcionamento, *Random Walk*, *Best First* e *Backtrack* (Figura 24).

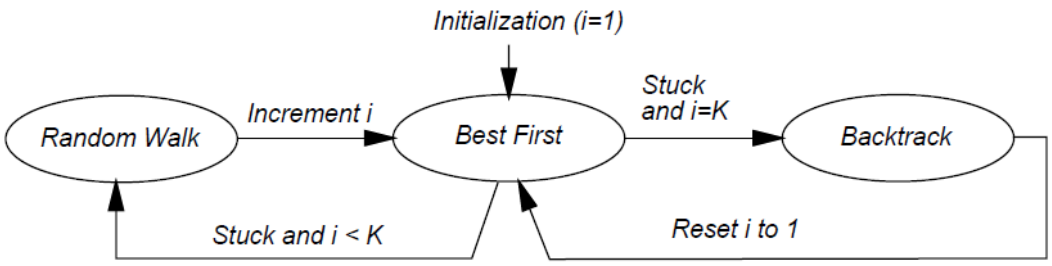


Figura 24 – Sistema proposto por *LaValle* composto por três modos de execução *in* (21)

A nível da execução este sistema inicia-se no modo *Best-First* no qual o agente assegura a navegação através das forças exercidas pelos campos de potencial. Se em algum momento a progressão do agente for considerada nula devido a um ótimo local o agente comuta para o modo *Random Walk*.

Neste segundo modo o agente considera um conjunto de posições e realiza uma série de deslocamentos aleatórios. Apesar da aleatoriedade, existe, tal como indicado um conjunto de soluções e pressupostos que podem ser tidos em conta para a implementação deste modo.

Neste modo o próprio conceito de aleatoriedade deve ser definido, já que pode ser interpretado como a realização de um conjunto de aleatório de movimentos por parte do agente mas também pode ser interpretado como a seleção aleatória de um conjunto limitado de ações.

Após um número *n* de iterações neste modo o agente volta ao modo inicial.

Caso o agente volte a deparar-se com um ótimo local comuta então para o modo *Backtrack*. Neste modo o agente seleciona uma das posições considerada anteriormente durante o modo *Random Walk* comutando novamente para o modo inicial.

Segundo o autor, a utilização de aleatoriedade considerando as várias abordagens apresentadas de geração de deslocamentos aleatórios e geração de posições aleatórias, permite a obtenção de soluções capazes de suportar o dinamismo associado ao ambiente. Como desvantagem o agente pode gerar percursos pouco eficientes.

Outro caso no âmbito do desenvolvimento de soluções baseadas na adição de aleatoriedade passa pelo trabalho desenvolvido por *Casselli*, *Reggiani* e *Rocchi* (25) denominado de *StraightLine*. De um modo geral o algoritmo realiza uma série de deslocações aleatórias sendo cada movimentação caraterizada pela seleção aleatória de uma direção que o agente segue até encontrar um obstáculo ou uma configuração em que a posição atual se encontre associada a um potencial mais favorável que o ótimo local do qual o agente tenta escapar. Caso o agente encontre um obstáculo gera uma nova direção aleatória e repete o mesmo processamento.

Além deste método, o trabalho apresentado por *Caselli*, *Reggiani* e *Rocchi* apresenta mais duas variações deste método de forma a melhorar o seu desempenho.

Outro trabalho divulgado baseado neste tipo de abordagem é apresentado por *Kirkpatrick*, *Gelatt* e *Vecchi* (26) no qual apresentam a sua solução denominada de *Simulated Annealing* (26).

De um modo geral este algoritmo permite aleatoriedade através da realização de movimentos aleatórios caso sejam cumpridos determinados requisitos. A probabilidade de exploração aleatória é definida por um parâmetro designado "temperatura" sendo que este parâmetro sofre um decaimento ao longo da execução do algoritmo.

Na génese desta abordagem encontra-se a premissa que o agente pode efetuar movimentos aleatórios dependendo de uma probabilidade à qual se encontra associada o parâmetro "temperatura", sendo esta probabilidade proporcional a este parâmetro.

A nível do comportamento demonstrado pelo agente, e tendo em conta que no início da navegação a probabilidade de realização de movimentos aleatórios é superior, é possível verificar numa primeira instância um comportamento exploratório. À medida que a probabilidade decresce o agente comporta-se de forma cada vez mais semelhante a um método baseado no gradiente.

Com isto podemos concluir que esta solução nem sempre segue a direção mais favorável do campo de potencial já que por vezes o agente desloca-se num sentido mais desfavorável. Além disto e segundo os autores, este algoritmo tem garantias de o ponto objetivo ser sempre alcançado já que mesmo com a existência de ótimos locais e com a temperatura a um valor baixo existe a possibilidade de realizar movimentos aleatórios que afastam o agente desse ponto ou zona. Apesar desta garantia de completude não existe garantia a nível temporal de que a solução é atingida em tempo útil.

2.7.2. Adição de forças

No seu trabalho, *Safadi* (27) apresenta uma proposta de solução adicionando ao somatório das forças atrativas e repulsivas uma terceira componente a qual se encontrada designada como força virtual. Segundo a autora, é possível concluir que esta força é condicionada pela quantidade de espaço livre disponível, ou seja, a força exercida pelo campo de potencial gerado pelo espaço livre seria tão intensa quanto mais espaço livre existir e seria somada com as restantes forças atrativas e repulsivas sempre que o agente detete que se encontre num ótimo local. Matematicamente *Safadi* define a esta força da seguinte forma:

$$F_f = F_{cf}\left(\cos(\theta)\, e_x + \sin(\theta)\, e_y\right) \qquad (4)$$

Onde θ é designado de *free space orientation*, F_{cf} é uma força constante, e_x.indica a componente da força na dimensão *x*, e e_y indica a componente da força na dimensão *y* Visualmente a relação entre as três forças pode ser representada da seguinte forma (Figura 25):

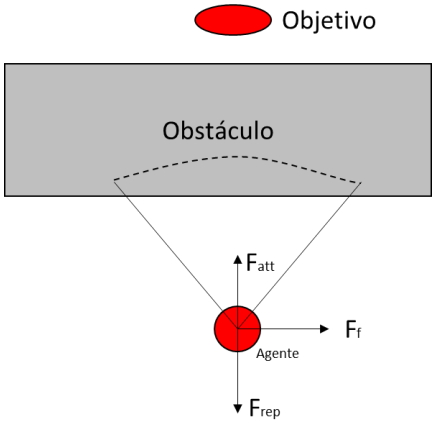


Figura 25 - Relação das várias forças envolvidas no cálculo utilizado por *Safadi in* (27)

O cálculo da força resultante do somatório das forças exercidas por cada um dos campos de potencial é formalmente definida do seguinte modo:

$$F = F_{att} + F_{rep} + F_f \qquad (5)$$

Apesar dos resultados obtidos pela aplicação desta técnica, segundo *Safadi*, existem pontos que devem ser vistos como desvantagens. Destes podem referir-se, por exemplo, o cálculo da força virtual, a não eliminação do problema dos ótimos locais e o comportamento demonstrado pelo agente.

Também *Park* e *Lee* (24) utilizam esta técnica como forma de ultrapassar a problemática associada aos ótimos locais. Neste trabalho existe um aspeto interessante a reter, a representação do agente. O agente pode ser visto como um bloco único, mas por vezes o *robot* pode ser

constituído por vários elementos ou ter uma dimensão razoável. Neste caso, considerar o agente como um ponto único pode não ser suficiente para que todo o agente evite obstáculos. Devido a isto os autores, tal como anteriormente *Khatib*, consideram a existência de vários pontos no agente sobre a ação de forças exercidas por campos de potencial.

A adição de forças é concretizada após a deteção de um ótimo local e através da contemplação de um obstáculo virtual. O agente deve posteriormente contemplar a existência deste obstáculo e do campo de potencial por ele gerado. Este campo segue as mesmas primitivas que caracterizam os restantes campos, nomeadamente a propagação do potencial ao longo do ambiente. Assim é possível afastar o agente para áreas onde seja possível o agente seguir um percurso até ao objetivo.

Este técnica apresenta também ela apresenta desvantagens nomeadamente a gestão dos obstáculos virtuais. A sua adição elimina ótimos locais mas ao mesmo tempo pode levar à criação de novos ótimos locais. Daqui retira-se que a ação desse campo de potencial sobre o agente deverá ser limitada e corretamente parametrizada.

2.7.3. Mecanismos de preenchimento

Este tipo de abordagem é caraterizada pela criação de um método para não considerar posições associadas a ótimos locais. Dois exemplos possíveis associados a este tipo de abordagem passam pela associação dessas posições a potenciais repulsivos e que ao longo do tempo podem ser incrementados, ou considerar a posição como previamente atingida e não a considerar novamente para fins de deslocação.

Dentro do segundo exemplo encontra-se a estratégia apresentada por *Baert* (28). Neste trabalho, o ambiente é definido de uma forma discreta e não continua, ao contrário dos trabalhos apresentados anteriormente. Sendo assim, *Baert* define o ambiente no qual o agente navega através de uma grelha e define o campo de potencial da seguinte forma.

"Na sua essência um campo de potencial pode ser pensado como uma matriz. Cada *pixel* existente na grelha é representado por um número que nos indica algo acerca do estado daquele pixel dependendo do contexto atual. Tendo em conta que o objetivo é encontrar um percurso entre o ponto inicial e o ponto final é lógico assumir que serão esses números os pontos fulcrais para o agente se mover ao longo da grelha" (28).

A forma utilizada por *Baert* para permitir ao agente navegar até à posição considerada como objetivo passa pela atribuição de um valor a cada um dos elementos da grelha. Este valor é baseado na distância desse elemento até à posição considerada como objetivo. Após esta

primeira fase o agente analisa os pontos vizinhos e desloca-se para o vizinho com o potencial mais favorável.

Tal como em abordagens que consideram o ambiente contínuo, esta solução apresenta também ela a probabilidade de existência de ótimos locais. A estratégia de preenchimento de mínimos é concretizada pela integração do algoritmo de pesquisa *best-first*.

A implementação deste algoritmo contempla a utilização de uma árvore que vai sendo construída ao longo da execução do agente e que tem como base a posição inicial do agente. Para cada posição são consideradas oito posições vizinhas as quais são adicionadas à árvore todas aquelas que não são repetidas. Após isto o agente deve deslocar-se para a posição com o potencial mais favorável dentro das posições vizinhas à posição atual.

Considerando como exemplo a situação apresentada na Figura 26.

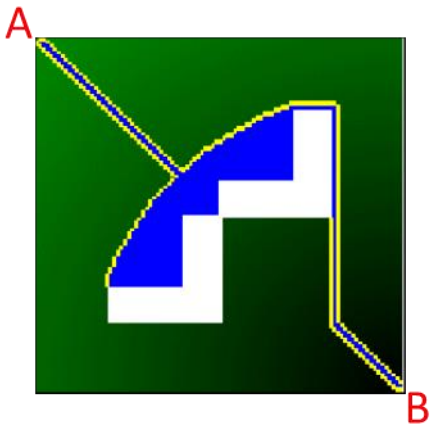


Figura 26 - Percurso gerado entre A e B com aplicação do algoritmo *Best-First in* (28)

O agente entre os dois pontos encontra um obstáculo, representado a branco, o qual deverá ultrapassar. As primeiras posições a ser geradas, representadas a azul, são aquelas que tem associada uma menor distância até ao ponto B, e que neste caso se encontram no cento do obstáculo. À medida que o agente gera e descarta essas posições o agente irá aproximar-se da extremidade do obstáculo até ser capaz de o ultrapassar.

Apesar da utilização desta técnica ser capaz de ultrapassar os problemas associados com ótimos locais possui características menos vantajosas a nível da manutenção da árvore de gestão das posições. Este aspeto leva a um maior consumo de memória e que conduz a uma degradação do desempenho do agente.

2.7.4. Mecanismos de retrocesso

Este tipo de abordagem também denominada por *backup* ou *backtracking*, é caraterizada pelo regresso do agente a pontos do percurso nos quais já se encontrou. Sendo assim, o percurso demonstra pontos em que o agente ao não conseguir ultrapassar determinadas zonas regressa a uma posição anterior de forma a gerar um novo percurso.

No trabalho apresentado por *Gambardella* e *Versino* (29) é considerado o desenvolvimento de uma solução com conhecimento global do ambiente no qual o agente se encontra. Da sua análise conclui-se que uma solução desse tipo não conseguiria oferecer um desempenho aceitável em ambientes dinâmicos. Como exemplo os autores propõem o ambiente apresentado na figura seguinte, Figura 27.

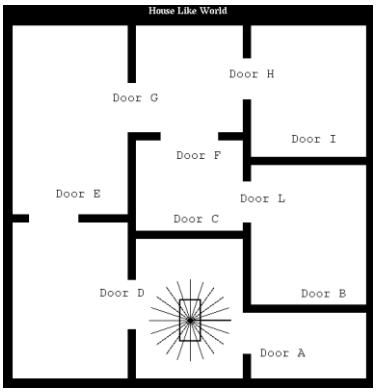


Figura 27 - Ambiente apresentado por *Gambardella* e *Versino in* (29)

O dinamismo apresentado neste ambiente encontra-se associado às portas, que ao longo do tempo podem ser abertas ou fechadas. De forma a contemplar este tipo de situação e tornar o desempenho do agente mais aceitável, os autores consideram que o agente deve, durante as suas deslocações, criar informação que o ajude a navegar ao longo do ambiente e a evitar ótimos locais.

A solução proposta divide-se nos seguintes pontos:

- Planeamento;
- Ótimos locais;
- Aprendizagem.

A relação destes três pontos com a utilização de mecanismos de retrocesso surge da forma como é realizada a modelação do ambiente. Durante a execução da navegação o agente explora os pontos existentes no ambiente sendo que dependendo das caraterísticas analisadas classifica determinados pontos como:

- Nós associados a ótimos locais;
- Nós associados a objetivos intermédios;
- Nós de retrocesso.

A classificação é realizada através da relação entre um determinado ponto e a sua distância para o ponto considerado como objetivo. Desta classificação individual pode ser retirada informação como a classificação de nó associado a ótimo local caso nenhum dos seus vizinhos não se encontre a uma distância inferior do ponto considerado como objetivo.

A utilização do mecanismo de retrocesso é evidenciada aquando do cumprimento de determinadas condições em que no momento da deteção de um ótimo local o agente traça um percurso para o nó de retrocesso.

A utilização deste tipo de estratégia requere que durante a navegação o agente seja capaz de analisar um determinado ponto e perceber se esse ponto pode ser considerado como um ponto de retrocesso aceitável. Ou seja, a partir desse ponto o agente pode gerar outros percursos diferentes daquele que o levaram a ter de iniciar o processo de retrocesso. Independentemente da estratégia de seleção de pontos de retrocesso este tipo de abordagem não garante, só por si, a não repetição de ótimos locais.

2.8. Resumo

Os algoritmos de navegação baseados em campos de potencial surgiram como forma de abordar a problemática da navegação autónoma. De forma a cumprir este requisito é associado um potencial aos elementos presentes no ambiente, sendo este potencial dependente da natureza do elemento. Considerando a propagação deste potencial ao longo do ambiente é possível definir campos de potencial, os quais exercem uma força sobre outros elementos. Para navegar os agentes utilizam as forças exercidas sob eles de forma a atingir os objetivos.

Esta solução permite o desenvolvimento de agentes que podem ser utilizados em ambientes reais devido ao seu baixo tempo de resposta e ao baixo consumo de recursos computacionais. Como compromisso as soluções baseadas nestes algoritmos podem apresentar a existência de ótimos locais, caraterizados pelo equilíbrio entre forças numa determinada posição ou zona e que em última instância inviabilizam a navegação.

De forma a contornar os ótimos locais foram surgindo ao longo do tempo diversas soluções como por exemplo:

- Eliminação de ótimos locais;
- Deteção de ótimos locais;
- Fuga de ótimos locais.

Infelizmente nenhuma destas técnicas resolve de forma eficaz e concreta o problema, sendo que todas elas apresentam desvantagens quer sejam os percursos gerados pelo agente, a não existência de garantia de não repetição de ótimos locais ou a necessidade de consumir mais recursos computacionais.

De um modo geral a utilização deste tipo de abordagem não oferece soluções perfeitas para a resolução da navegação entre pontos mas oferece soluções com compromissos interessantes, na medida em que é possível um agente realizar um percurso com sucesso evitando os obstáculos que vão surgindo sem que para isso exista um consumo de recursos demasiado elevado e o mantendo um tempo de resposta que permite a sua utilização em ambientes reais.

De seguida serão apresentados os métodos de navegação com base em campos de potencial selecionados para implementação, os quais utilizam diferentes aspetos dos temas abordados neste capítulo.

3. Métodos de navegação com base em campos de potencial

3.1. Método do gradiente

Este primeiro método baseado em campos de potencial remonta aos anos 80, tendo sido *Khatib,* (30) um dos seus autores. Até esse ponto a navegação no ambiente na qual se inclui a capacidade de evitar colisões entre um agente e os restantes elementos do ambiente era da responsabilidade de um controlo de alto nível.

Este tipo de soluções obrigam a um planeamento prévio sendo necessário o cálculo de todo o percurso antes da realização da navegação até ao objetivo. O problema desta abordagem decorre do tempo de planeamento do percurso que pode ser superior ao pretendido, sendo tanto pior quantas mais vezes existir a necessidade de replanear o percurso. Um exemplo deste tipo de situação seria o agente deparar-se com um obstáculo inesperado. De forma a colmatar este problema, o método do gradiente foi apresentado como uma forma de melhorar o desempenho de métodos de navegação em tempo real (30).

A forma como *Khatib* resume a filosofia associada a este método é a seguinte:

"O agente (manipulador) move-se num campo de forças. A posição tida como destino é considerada um pólo atrativo e os elementos tidos como obstáculos são considerados como pólos repulsivos." (30)

Seguindo esta filosofia deve-se considerar que este método é baseado na caracterização de cada elemento inserido no ambiente. Esta caracterização é concretizada através da associação de diferentes potenciais aos elementos. Considerando um ambiente composto por um objetivo e um obstáculo o potencial resultante pode ser definido da seguinte forma (30):

$$P_{res}(x) = P_{atr}(x) + P_{rep}(x) \tag{6}$$

Em que o potencial resultante, $P_{res}(x)$, é definido pelo somatório do potencial associado com o objetivo, $P_{atr}(x)$, com o potencial associado com o obstáculo, $P_{rep}(x)$ e sendo x a posição atual.

Não sendo o campo de potencial uniforme em todo o ambiente também a força por ele exercida é diferente dependendo do ponto em que o agente se encontre. Tomando como exemplo um campo de potencial atrativo deve-se considerar que quanto mais próximo o agente se encontrar do alvo maior será a força exercida sobre ele. O campo de potencial pode ser visto então como

originando um gradiente, tal como descrito anteriormente o que permite definir a força resultante da seguinte forma (30):

$$F_{atr}(x) = grad[P_{atr}(x)] \quad (7)$$

Sendo esta expressão válida também para a força gerada pelo campo de potencial repulsivo associado ao obstáculo. Somando estas duas forças é possível obter a força resultante que atua sobre o agente da seguinte forma (30):

$$F_{res} = F_{\text{atr}} + F_{rep} \quad (8)$$

Em que a força resultante (F_{res}) resulta do somatório da força atrativa gerada pelo potencial associado ao objetivo (F_{atr}) com a força repulsiva gerada pelo potencial associado ao obstáculo (F_{rep}).

Nesta abordagem o conhecimento a que o agente tem acesso é maioritariamente local à exceção da informação associada com os alvos. Em relação aos obstáculos o grau de conhecimento pode ser mais baixo podendo o agente só ser afetado pelo campo de potencial por eles gerados caso se aproxime demais, ou seja, o campo de potencial associado aos obstáculos só afeta a navegação do agente mediante o cumprimento de determinados critérios, neste caso a distância o que leva à definição de um perfil de magnitude semelhante aos que vimos anteriormente.

A atribuição desta caraterística aos potenciais repulsivos permite que numa primeira instância a navegação do agente seja apenas afetada pelos obstáculos que se encontrem numa área próxima do agente e numa segunda instância permite ao agente navegar numa área maior. Esta caraterística pode também ser vista como uma barreira criada para que o agente não colida com os obstáculos que lhe surjam.

A nível de resultados obtidos verificou-se que o nível de sucesso deste método é limitado. Isto acontece visto que em ambientes onde existam muitos obstáculos a existência de ótimos locais leva o agente a estabilizar em pontos que não são objetivos. Mesmo com a adição de técnicas ou procedimentos que possam atenuar o número de ocorrências em que o agente não consegue atingir o seu objetivo existe sempre a probabilidade da navegação falhar visto que a execução do método ocorre num contexto local.

Segundo o autor, este problema poderia ser colmatado pela adição de elementos que permitiriam a obtenção de conhecimento global do ambiente e que poderiam originar diversos pontos de orientação do percurso. Após este primeiro passo o agente ligaria cada um dos pontos baseando-se no método do gradiente. O problema desta abordagem passa pela dificuldade de

implementação de mecanismos que deverão executar em tempo real e pelo consumo elevado de recursos computacionais que podem apresentar.

3.2. Método de deteção e fuga baseada em *wall following*

No capítulo anterior foram apresentadas diversas opções que podem ser tomadas em consideração de forma a lidar com o problema dos ótimos locais. Duas das soluções possíveis passam pela utilização de estratégias de deteção de ótimos locais e pela utilização de estratégias de fuga de ótimos locais. Apesar desta separação são várias as soluções existentes que tiram proveito da conjugação de ambas. O método de deteção e fuga baseado em *wall following* é uma das soluções existentes onde esta situação se verifica.

Este tipo de estratégia foi abordado por diversos autores entre eles *Borenstein* e *Koren* (31), *Yun* e *Tan* (32) e *Zhu, Zhang* e *Song* (33).

O ponto transversal a estas soluções passa pela existência de dois modos de navegação que comutam entre si dependendo dos estímulos recebidos pelo agente. Num primeiro modo a navegação é baseada em campos de potencial como por exemplo o método do gradiente, e num segundo modo a navegação é baseada no algoritmo de *wall following*.

A comutação entre cada um dos modos seria realizada através da deteção de um ótimo local e a deteção do término da fuga desse ótimo local. A arquitetura representativa deste modelo é a seguinte (Figura 28).

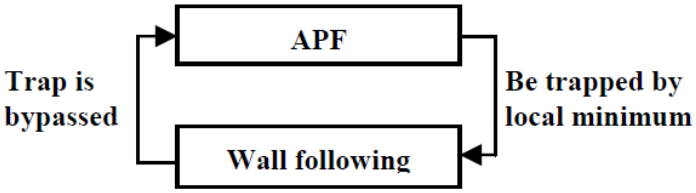


Figura 28 – Arquitetura do modelo de comutação de modo *in* (33)

Para a implementação desta arquitetura é necessário a implementação de três pontos:

- Método baseado em campos de potencial;
- Algoritmo de *wall following*;
- Estratégia de comutação entre modos.

Borenstein e *Koren* (31) começaram por implementar um método baseado na implementação sugerida por *Khatib* com a diferença a incidir na representação do ambiente. Nesta implementação o agente representa o ambiente baseado numa grelha sendo que a cada célula

existe associado um grau de certeza da probabilidade da existência de um obstáculo. Este grau de certeza seria obtido pelos sensores existentes no agente.

Com isto o agente é repelido de conjuntos de células que se encontrem preenchidas de obstáculos e as restantes são ignoradas. A Figura 29 mostra o agente desenvolvido e o efeito das várias forças existentes.

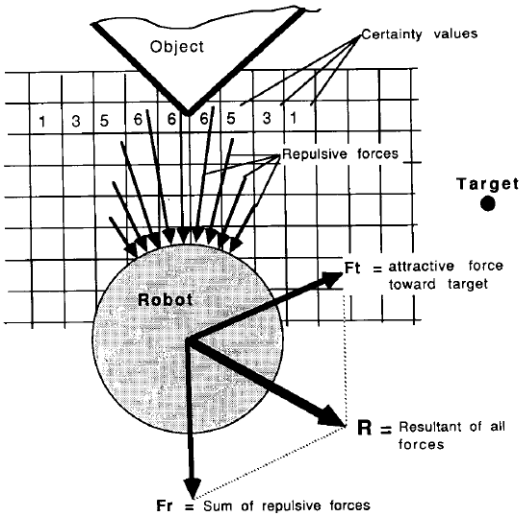


Figura 29 – Exemplo de execução do agente de *Borestein* e *Koren in* (31)

A nível da comutação de modos foi considerada a direção do agente. É considerado que o agente utiliza o algoritmo baseado em campos de potencial desde que se mantenha alinhado com o alvo, caso contrário deverá comutar de modo. Este pressuposto é definido através de duas expressões.

$$|\theta_t - \theta_0| > 90^0 \quad (9)$$

$$|\theta_t - \theta_0| \leq 90^0 \quad (10)$$

Sendo θ_t o ângulo entre o agente e o objetivo e θ_0 o ângulo no qual o agente navega atualmente. A primeira expressão (9) provoca a comutação para o modo *wall following* e a segunda expressão (10) provoca a comutação para o modo baseado em campos de potencial.

Apesar de funcional e simples esta estratégia pode exibir ciclos, situação que obriga à adição de novas validações. Além deste aspeto a própria representação do ambiente em grelha exige uma análise cuidada em relação à dimensão das células da grelha.

O trabalho apresentado por *Yun* e *Tan* (32) altera a formulação dos campos de potencial devido às caraterísticas físicas associadas ao agente por exemplo número de sensores, orientação dos sensores, sonares entre outros.

A nível do algoritmo de *wall following* o método utilizado contempla uma distância que o agente deve manter entre si e a parede.

A nível da comutação foram contempladas a velocidade e a distância para o objetivo. Caso a velocidade em determinado ponto fosse inferior a uma velocidade mínima pré-definida, equação (11), o agente comuta para o modo *wall following*. Quando a distância para o obstáculo é inferior a uma distância pré-definida o agente comuta de volta à navegação baseada em campos de potencial, equação (12). Este ponto pode ser expresso pelas seguintes formulações (33):

$$v\,(t) \;< v_{lim} \wedge \; x \;\neq x_d \tag{13}$$

$$\Delta\,|x - x_d| \;< 0 \tag{14}$$

Sendo $v\,(t)$ a velocidade do agente num determinado momento, v_{lim} a velocidade limite, x a posição do agente e x_d a posição do obstáculo.

A nível de resultados e segundo os autores esta nova implementação permitia um bom desempenho em ambientes onde a configuração contemplasse obstáculos que os autores classificaram como *bench type*, *corner* e *dead-end channel*. Devido a esta falta de generalização a utilização deste algoritmo pode não ser aceitável em outros ambientes.

Figura 30 – Exemplos de obstáculos do tipo *bench, corner* e *dead-end channel in* (32)

Além deste aspeto os autores evidenciam um problema que não é possível resolver com a utilização desta técnica, Figura 31.

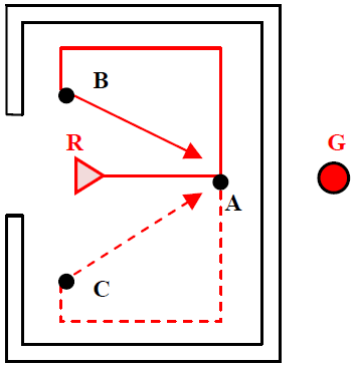


Figura 31 - Situação não resolvida pelo método de *Yun* e *Tan in* (32)

Esta situação seria posteriormente resolvida no trabalho de *Zhu*, *Zhang* e *Song* (33). Estes autores consideravam a imperfeição das condições de comutação e a falta de conhecimento global do agente como as razões para a impossibilidade de resolução de um conjunto de situações, nas quais se incluía o exemplo acima referido (Figura 31). Isto levou à adição de uma componente constituída pela memória sendo este aspeto a principal divergência entre esta implementação e as indicadas anteriormente as quais consideram um conjunto de informação local.

Deste trabalho resultaram as seguintes formulações. Para a comutação do mecanismo de navegação para o modo *wall following* contribui o cumprimento de qualquer uma das seguintes formulações.

$$\left|F_{att}(x) + F_{rep}(x)\right| < \varepsilon \quad (15)$$

$$|x\,(t+T) - x\,(t)| \leq s_1 \quad (16)$$

$$|x_a - x_b| < \alpha . s_{ab} \quad (17)$$

A primeira expressão indica que o somatório da força atrativa, $F_{att}(x)$, e a força repulsiva, $F_{rep}(x)$, não pode ser inferior a um valor ε. A segunda expressão diz respeito à distância percorrida pelo agente num determinado intervalo de tempo T não pode ser inferior a um determinado valor s_1. Por fim, a terceira expressão diz respeito à distância percorrida entre dois pontos que não deverá ser inferior a uma percentagem α da distância entre dois pontos s_{ab}.

A nível da comutação para o modo de navegação baseado em campos de potencial existe a seguinte formulação:

$$-90^0 \leq \theta_t \leq \beta, \quad if\ anticlockwise\ direction, \quad (a) \quad (18)$$

$$\beta \leq \theta_t \leq -90^0, \quad if\ clockwise\ direction, \quad (b) \quad (19)$$

$$crossjudge\left(l_{rg}, PathMap\right) = false, \quad (c) \quad (20)$$

$$\left|x - x_{front-obstacle}\right| > s_2\,, \quad (d) \quad (21)$$

$$|x - x_d| < \rho, \quad (e) \quad (22)$$

O cumprimento das condições (a) ou (b) em conjunto com (c) e (d) ou o cumprimento da condição (e) promovem a comutação.

Neste contexto as condições (a) e (b) dizem respeito ao alinhamento do agente com o objetivo, sendo que o ângulo θ_t se deverá encontrar entre o intervalo -90^0 e o ângulo definido por β. A condição (d) diz respeito à distância entre o agente e obstáculo e que caso seja superior a s_2 valida a comutação. A condição (e) diz respeito à distância entre o agente e o objetivo o qual deverá ser inferior à distância entre o agente e o obstáculo.

Por fim a expressão (c) apresenta o ponto diferenciador entre esta e outras implementações. Nesta implementação os autores contemplam a adição de memória sendo esta representada na expressão por $PathMap$ na qual constam os pontos anteriormente percorridos pelo agente. A variável l_{rg} representa uma linha contínua a partir do agente até ao obstáculo. Para a validação desta expressão os autores consideram que nenhuma reta definida por dois pontos anteriores contínuos intersecta a recta l_{rg}.

Em suma, os exemplos mencionados acima indicam que o principal ponto para uma estratégia baseada na deteção e fuga de ótimos locais não se resume apenas ao desenvolvimento de um algoritmo baseado em campos de potencial que não seja afetado por ótimos locais ou ao desenvolvimento de um mecanismo de fuga que não repita ótimos locais anteriores mas que deve ser dada a adequada relevância ao mecanismo de comutação entre os dois modos de funcionamento.

3.3. Algoritmo *avoiding-past*

A não utilização ou pouca utilização de memória é uma das características normalmente apresentadas por métodos de navegação autónoma baseados em campos de potencial. Apesar das vantagens apresentadas por este tipo de abordagem existem situações que levam à necessidade de utilização da memória por forma a ser possível o agente navegar até ao objetivo.

Um exemplo deste tipo de situação já foi identificado neste trabalho, Figura 31, onde a utilização de memória é utilizada para a comutação entre um método baseado em campos de potencial e um algoritmo de *wall following*.

Mais uma vez o recurso à memória foi a solução encontrada para resolver as limitações associadas com os métodos de navegação baseados em campos de potencial, desta feita por *Balch* e *Arkin* (34) no desenvolvimento do método de navegação baseado no algoritmo *avoiding-past* (34). Este trabalho demonstrou que com a aplicação deste conceito é possível obter resultados interessantes tanto a nível do desempenho como a nível da qualidade dos percursos gerados.

Este método tal como os restantes baseados em arquiteturas reativas tem na sua organização um conjunto de comportamentos que juntos podem formar comportamentos mais complexos. Este aspeto encontra-se evidenciado mais uma vez neste método, sendo que na sua génese considera-se a existência de três comportamentos base.

Estes comportamentos consistem em primeiro lugar em "Evitar obstáculo" o qual se encontra definido pela seguinte expressão de perfil de potencial (34):

$$V_{magnitude} = \begin{cases} 0, & d > S \\ \dfrac{S-d}{S-R} * G, & R < d \leq S \\ \infty, & d \leq R \end{cases} \quad (23)$$

Sendo, S designado de esfera de influência ou zona na qual o agente pode sentir o efeito da força repulsiva exercida pelo campo de potencial gerado pelo obstáculo, R o raio do obstáculo, G um ganho e d a distância até ao centro do obstáculo.

O segundo comportamento pretende a aproximação ao objetivo, "Mover para objetivo", sendo expresso pelos autores da seguinte forma (34):

$$V_{magnitude} = valor\ fixo$$
$$V_{direction} = direcção\ do\ objectivo$$

Por fim o terceiro comportamento, "Ruído", tem como função evitar alguns problemas que possam interferir com a navegação do agente. Os autores definiram este comportamento da seguinte forma:

$$V_{magnitude} = valor\ fixo$$
$$V_{direction} = direcção\ aleatória\ durante\ algum\ tempo$$

Apesar de este comportamento ser capaz de evitar alguns dos problemas associados com a navegação baseada em campos de potencial, este comportamento pode não atingir um desempenho aceitável quando deparado, por exemplo, com configurações do tipo *box canyon* como anteriormente ilustrada na Figura 14.

De forma a resolver problemas como este e outros de maior complexidade, *Balch* e *Arkin* (34) adicionaram uma quarta componente, força repulsiva exercida por um campo de potencial associado a posições anteriores.

Esta abordagem é inspirada pela forma como determinados insetos, por exemplo, formigas conseguem orientar-se entre as fontes de alimento e o ninho. No caso das formigas existe a libertação de um tipo de feromona durante o percurso, o qual as restantes formigas seguem e que servirá de referência no regresso. À medida que mais formigas vão utilizando este percurso a presença de feromonas vai tornando-se mais forte e mais fácil de seguir.

Transportando este conceito para a navegação autónoma em agentes pode-se substituir-se este conceito de feromonas por memória espacial. Esta é concretizada através da memória de

posições anteriores do agente, as quais tem associado um potencial repulsivo de forma a afastar o agente de posições já percorridas. Da mesma forma que a presença de feromonas se torna mais forte à medida que o percurso é repetido por outras formigas, no algoritmo *avoiding-past* existe um fator multiplicativo que reforça a força exercida pelo campo de potencial gerado por uma posição anterior. Tendo isto em conta, a adição desta quarta componente às três previamente mencionadas dá origem à arquitetura representada na Figura 32.

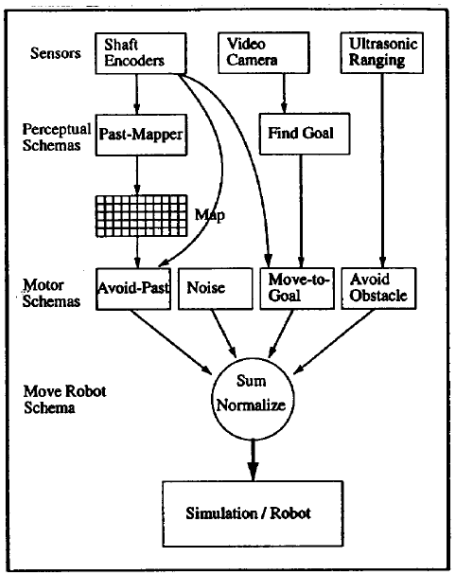


Figura 32 – Diagrama de blocos utilizado no algoritmo *avoiding-past in* (34)

Para a construção da componente de força de memória devem ser tidos em conta dois aspetos, a gestão da memória e a forma de como esta é refletida na restante implementação. Isto é conseguido através do conceito de visita, o que na prática se reflete no número de vezes que o agente se encontra numa determinada posição.

Numa primeira abordagem válida pode considerar-se que seria gerada uma memória para cada ponto e que essa memória contribuiria para que o agente se afastasse. A questão lançada tendo em conta este pressuposto relaciona-se com o facto de em vez de se considerar um ponto, considerar-se uma área. Apesar de ser necessário a análise de critérios para decidir a dimensão da área, segundo os autores essa solução apresenta um melhor desempenho. Este facto deve-se à criação de uma memória mais esparsa mas suficiente para que o agente se afaste de zonas anteriormente percorridas.

Desta discussão surgiram dois parâmetros que refletem este conceito de área quando da geração da memória, sendo a dimensão da área marcada como "visitada" definida pelo parâmetro PAST-MARK, e da geração da força repulsiva associada à memória, sendo a dimensão da área considerada para efeitos de repulsão definida por PAST-HORIZON.

Considerando que o agente se encontra a navegar pelo ambiente, cada posição percorrida é analisada verificando-se se pertence a uma área já considerada o que leva à incrementação do número de visitas dessa área ou se pertence a uma nova área.

```
for i = (X – PAST-MARK) to (X + PAST-MARK):
        for j = (Y – PAST-MARK) to (Y + PAST-MARK):
                if A[i,j] < MAX:
                        A[i,j] = A[i,j] + 1
```

Sendo que A[] é a memória espacial indexada por X,Y que correspondem à posição do agente, MAX o número máximo de visitas para uma determinada posição e PAST-MARK a dimensão da área a recordar.

A nível da geração da força repulsiva exercida a partir do campo de potencial gerado pela memória é contabilizada tendo em conta a posição atual e as áreas anteriormente visitadas.

```
xvec.mag = 0, xvec.dir = 90, yvec.mag = 0, yvec.dir = 0, count = 0
for k = (X - PAST-HORIZON) to (X + PAST-HORIZON):
        for l = (Y - PAST-HORIZON) to (Y + PAST-HORIZON):
                if ( k < X ): xvec.mag = xvec.mag + A[k,l]
                if ( k > X ): xvec.mag = xvec.mag - A[k,l]
                if ( l < X ): yvec.mag = yvec.mag + A[k,l]
                if ( l > X ): yvec.mag = yvec.mag - A[k,l]
                count = count + A[k,l]
tempvec = sum-vector ( xvec, yvec)
pastvec.dir = tempvec.dir
pastvec.mag = past-gain * (count / ((PAST-HORIZON * 2) ^ 2 * MAX))
return (pastvec)
```

Dum modo geral, quantas mais vezes uma determinada área for visitada pelo agente maior será a força repulsiva exercida por essa área. Além desta caraterística outras podem ser adicionadas como por exemplo a diminuição do efeito de repulsão de uma determinada área. Por exemplo se uma área foi visitada fora de um intervalo de tempo pode ser descartada, permitindo assim que no futuro volte a ser analisada.

Por fim há que indicar que o sucesso desta solução também depende dos valores atribuídos aos parâmetros PAST-MARK e PAST-HORIZON. Segundo o indicado no trabalho, caso o parâmetro PAST-MARK seja um valor demasiado baixo o agente vai apresentar um comportamento em que parece manter-se na mesma zona. Se o valor for demasiado elevado o

agente poderá não explorar todas as zonas. O mesmo acontece para o parâmetro PAST-HORIZON.

3.4. Preenchimento de mínimos

Esta técnica é caraterizada pelo desenvolvimento de mecanismos com capacidade para descartar deslocamentos do agente para uma determinada posição ou zona do ambiente. Exemplos deste tipo de técnica são a incrementação do potencial repulsivo associado a uma posição ou zona do ambiente ou pela utilização de técnicas que excluem uma posição ou zona previamente percorrida pelo agente.

No trabalho apresentado por *Masehian* e *Amin-Naseri* (35) encontram-se referenciadas três soluções de navegação sendo que em duas delas existe a integração de um módulo baseado em campos de potencial. Cada um destes módulos soluciona a problemática dos ótimos locais de forma diferente.

Numa primeira abordagem, por forma a evitar que o agente retorne a posições percorridas anteriormente é atribuído a cada posição um potencial repulsivo. O valor associado ao potencial deve ser parametrizado dentro de um intervalo e cumprir um conjunto de pressupostos. Com isto o agente afasta-se da posição ou zona já que deixa de ser um ótimo local.

Durante a execução do processo de navegação existe um processo de caraterização das posições anteriores. Nesta implementação este processo é concretizado através da marcação das posições anteriores como “vistas”. Com isto, estas posições serão futuramente excluídas sempre que se pretenda realizar um deslocamento.

3.5. Algoritmo *wavefront*

A utilização deste algoritmo é sugerida, por exemplo, no trabalho apresentado por *Baert* (28) e tem como principal caraterística a não geração de ótimos locais. Esta caraterística sucede da própria execução do algoritmo, sendo a sua especificação a seguinte:

```
1  Execução é iniciada a partir de um ponto inicial considerado
   como objetivo
2  Atribuir a esse ponto o valor 0
3  Adicionar este ponto à lista de pontos seguintes
4  Iniciar ciclo
5        Retirar primeiro elemento da lista de pontos seguintes
6        Gerar pontos sucessores
7        Atribuir a cada um dos pontos sucessores o valor um
         valor incrementado ao valor da posição atual
8        Adicionar os pontos sucessores à lista de pontos
         seguintes
9        Se a lista de pontos seguintes se encontrar vazia
         terminar, caso contrário continuar
```

Executando este processo consegue-se a obtenção de uma representação do ambiente tal como a apresentada em baixo (Figura 33).

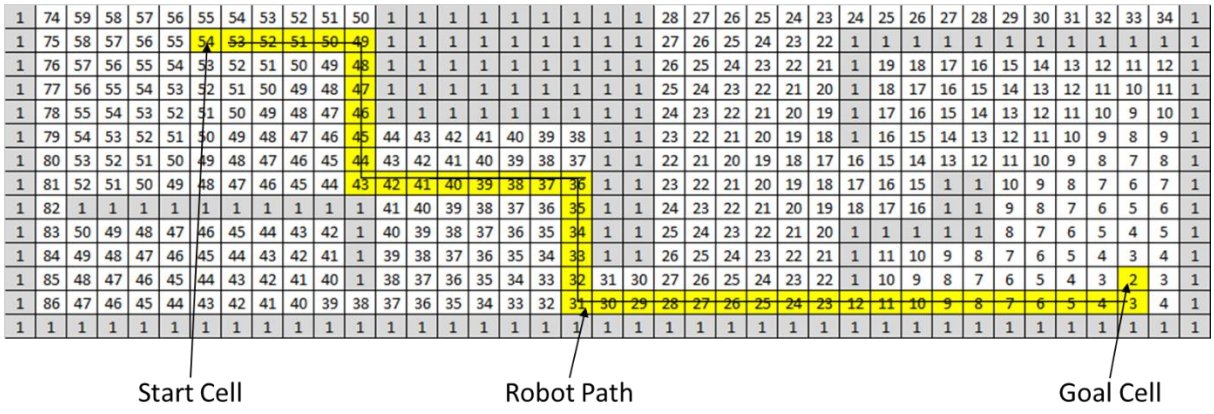


Figura 33 - Representação do ambiente considerando o algoritmo *wavefront in* (36)

Após a execução do algoritmo e tomando como exemplo a figura acima pode considerar-se a seguinte navegação por parte do agente. Início na célula indicada como *"Start Cell"*, analisar os pontos vizinhos e verificar qual possui o valor mais baixo. Após esta análise efetuar o deslocamento para esse ponto vizinho e repetir estes passos até chegar à célula identificada

como "*Goal Cell*". Esta célula possui como caraterística principal possuir o valor mais baixo em todo o ambiente.

De uma forma geral a execução deste algoritmo é composta por dois passos, a geração da representação do ambiente e a navegação até ao objetivo. Desta representação do ambiente nasce uma representação de um campo de potencial virtual que o agente utiliza para a realização da navegação.

Disto podem advir vantagens como a não criação de pontos ou zonas considerados como ótimos locais e desvantagens como o pré-processamento necessário para gerar a representação do ambiente.

3.6. Resumo

Estes algoritmos são algumas das propostas existentes baseadas em campos de potencial. Como foi possível verificar a maioria destes algoritmos possuem uma origem comum, o método do gradiente desenvolvido por *Khatib* (30).

Devido às limitações apresentadas pela primeira versão apresentada por *Khatib* outras foram desenvolvidas sendo que destas existem casos que até contrariam as premissas da implementação base como é o caso da utilização de memória.

Outros optaram por alterações na forma como é representado o ambiente, nomeadamente na utilização de ambientes discretos ao invés dos ambientes contínuos.

Apesar das várias alterações introduzidas o conceito base segundo o qual o agente é visto como um elemento sob a ação de vários campos de potencial, mantém-se, sendo que da sua utilização é possível desenvolver soluções com capacidade de resposta em tempo real.

4. Biblioteca de métodos de navegação com base em campos de potencial

Antes de abordar o desenvolvimento da biblioteca em concreto é interessante compreender a forma como esta se integra em relação a toda a solução desenvolvida. Deste modo a próxima secção é dedicada a este tema.

4.1. Visão

A biblioteca de métodos de navegação baseados em campos de potencial contempla um conjunto de algoritmos que podem ser usados para resolver a problemática da navegação autónoma.

Pretende-se que este conjunto de métodos seja passível de utilização em testes ou no desenvolvimento de futuras aplicações promovendo desta forma o reaproveitamento de componentes e a possibilidade de acelerar o desenvolvimento dessas soluções.

Outro ponto interessante passa pela contribuição que poderá ser realizada por parte dos utilizadores tendo em conta que com a identificação de erros e adição de novos métodos podem valorizar o trabalho até ao momento desenvolvido.

4.2. Arquitetura geral da solução

A concretização do presente trabalho é composta pelo desenvolvimento de duas peças de *software*. Previamente à especificação de cada uma das componentes é importante ter uma visão geral sobre a forma como estas componentes se relacionam entre si e com componentes de terceiros tendo em conta a proposta de arquitetura.

4.2.1. Componentes identificadas

Como foi anteriormente mencionado este trabalho é concretizado através do desenvolvimento de duas componentes, uma biblioteca de métodos de navegação autónoma baseada em campos de potencial e uma plataforma de teste para validar a execução dos métodos implementados.

A nível da biblioteca, esta apresenta-se como um componente único não sendo composta por outras componentes. No caso da plataforma de teste o mesmo não sucede sendo composta por um conjunto de subcomponentes.

Desta forma e a este nível torna-se interessante abordar a organização da plataforma de teste. A divisão da plataforma de teste permite a identificação de três componentes sendo as suas responsabilidades as seguintes:

- Visualização – a componente associada é responsável pela gestão da interface de utilização;
- Controlo – as componentes associadas são responsáveis pela representação das entidades ambiente e agente;
- Orquestração – a componente associada é responsável pela ligação entre as componentes anteriormente mencionadas.

A divisão da plataforma de teste pelas três componentes é uma tentativa de diminuir o impacto caso se pretendam efetuar alterações como por exemplo a adição ou remoção de elementos.

4.2.2. Outras componentes

O desenvolvimento de determinados pontos associados às componentes identificadas acima levou à introdução de outras componentes sendo de destacar a introdução da biblioteca *pygame*.

A seleção desta biblioteca deveu-se à necessidade de desenvolver determinadas funcionalidades tendo como requisito a sua facilidade de integração. Tendo em conta que esse desenvolvimento se encontra a ser realizado com a utilização de linguagem *python* esta seleção permitia um desenvolvimento mais rápido não sendo necessária a criação de pontos de integração entre tecnologias.

Em relação à sua utilização, a biblioteca *pygame* é usada em dois pontos, no desenvolvimento da interface gráfica e no encapsulamento do ambiente no qual o agente navega.

A nível de desenvolvimento da interface gráfica a sua utilização permite mostrar um ecrã ao utilizador com todos os elementos necessários para a validação do método de navegação tendo em conta um determinado ambiente.

A nível do encapsulamento permite a obtenção de informação acerca do ambiente enquanto o agente navega, nomeadamente os elementos detetados pelo agente tendo em conta uma determinada posição.

4.2.3. Organização global

Após a identificação das componentes é importante verificar como estas se encontram compostas entre si. A seguinte arquitetura (Diagrama 1) encontra-se como proposta para o desenvolvimento das várias componentes.

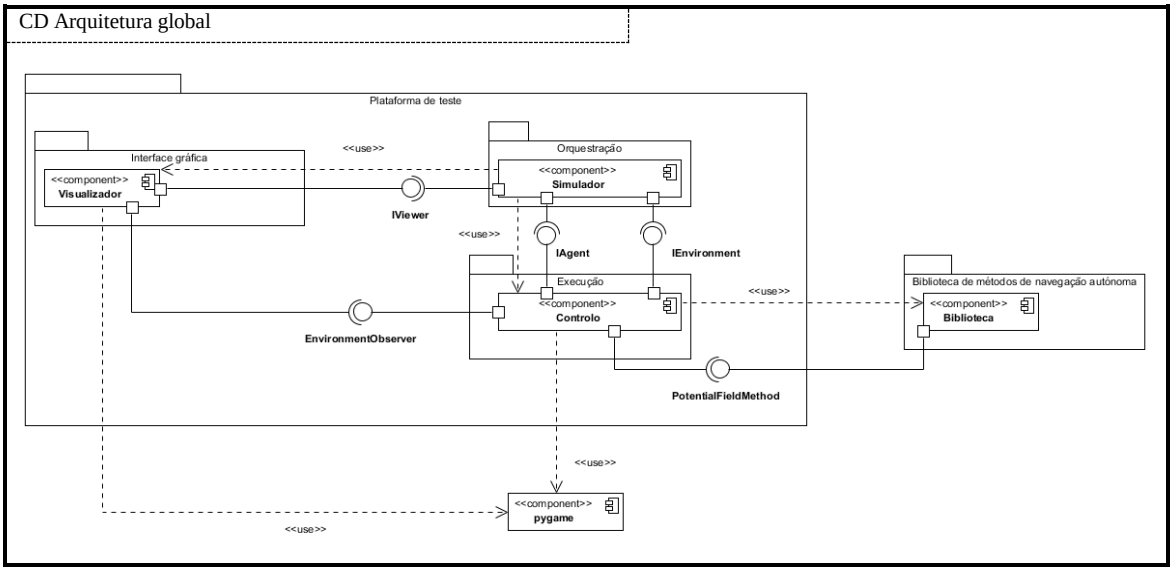


Diagrama 1 – Arquitetura global proposta

Como é possível verificar existe um conjunto de contratos entre os vários componentes especificados sob a forma de interfaces. São estes contratos os responsáveis pelo suporte da modularidade necessária para um menor impacto sempre que se pretendam realizar alterações.

A nível destes contratos existem a realçar os seguintes pontos:

- Interface *EnvironmentObserver* utilizada no âmbito da atualização da interface gráfica sempre que existam alterações no ambiente e baseada no padrão *Observer*;
- Interface *IViewer* responsável pela criação de independência entre o simulador e o visualizador permitindo que a execução dos algoritmos seja visualizada através de uma nova interface gráfica;
- Interface *IEnvironment* responsável pelo desacoplamento entre os objetos responsáveis pela representação do ambiente no qual o agente se encontra e o simulador;
- Interface *IAgent* responsável pelas mesmas funções que a interface acima, mas desta feita em relação aos agentes;
- Interface *PotentialFieldMethod* utilizada para definir os pontos comuns em relação à implementação específica de cada um dos métodos.

A abordagem aprofundada da biblioteca de métodos de navegação autónoma baseados em campos de potencial será realizada seguidamente. Quanto à plataforma de teste tem como

principal função permitir a validação da execução dos algoritmos definidos nesta fase sendo que será abordada posteriormente no capítulo correspondente.

4.3. Seleção de métodos de navegação baseados em campos de potencial

Como foi possível verificar anteriormente, existem diversas implementações de métodos baseados em campos de potencial. Apesar da existência de pontos comuns entre elas cada implementação sustenta-se em premissas diferentes o que lhe atribui vantagens e desvantagens a nível da sua utilização em ambientes reais.

A biblioteca a desenvolver pretende incluir um conjunto de concretizações relevantes tendo em conta as abordagens apresentadas de forma a demonstrar os problemas associados a estas abordagens e a forma como as diferentes concretizações os superam.

O primeiro método de navegação sugerido é o método do gradiente. Esta seleção dá-se devido a este ser o algoritmo normalmente utilizado para o desenvolvimento de outros. A principal vantagem que se pode retirar desta abordagem é que para ambientes mais simples, ou seja, com uma probabilidade reduzida de existência de ótimos locais, apresente uma percentagem de sucesso aceitável. Além disto, este algoritmo demonstra a simplicidade associada a algoritmos baseados em campos de potencial.

Tal como identificado anteriormente este algoritmo tem como principal desvantagem não realizar uma navegação com sucesso caso o agente se depare com um ótimo local.

Das várias abordagens propostas existe a conjugação de duas abordagens que é interessante sendo ela a deteção e fuga de ótimos locais. A versão apresentada aqui vai ser semelhante à proposta abordada anteriormente na secção 3.2, sendo que vai ser desenvolvido um algoritmo *wall following* para realizar a fuga em relação aos ótimos locais e a deteção será realizada através da comparação entre a força resultante e um valor α considerado como referência.

Outras das abordagens que vai ser considerada tem como principal característica a adição de memória. Tal como sugerido por outros autores, este facto pode ajudar o agente no sentido de recordar as zonas pelas quais já se encontrou. Tendo em conta estes dois factos vai ser implementado método baseado no algoritmo *avoiding-past*, sendo que as memórias existentes mantidas pelo agente vão ser utilizadas para gerar uma força de repulsão que o afasta de zonas anteriores.

As abordagens até aqui apresentadas são caraterizadas pela sua representação contínua do espaço onde se integram. Apesar disso existem soluções nesta área que caraterizam o espaço de forma discreta, normalmente sobre a representação duma grelha e estando cada uma das células associadas a um potencial resultante da ação exercida pelos campos de potencial presentes no ambiente.

Neste âmbito são disponibilizados dois algoritmos, *minimum filling* e *wavefront*. Em relação ao primeiro a sua utilização é interessante na medida em que possui formas de não considerar as posições anteriormente percorridas pelo agente. Em relação ao algoritmo *wavefront* a sua seleção deve-se ao facto a execução deste algoritmo não apresentar a existência de ótimos locais.

4.4. Especificação dos métodos de navegação autónoma implementados

Antes da apresentação da arquitetura proposta para a biblioteca de métodos de navegação autónoma baseada em campos de potencial é importante compreender a execução dos algoritmos contemplados. Só assim será possível compreender os pontos comuns e os pontos diferenciadores existentes entre as várias abordagens e a forma como estes contribuíram para a definição da arquitetura. Com isto é apresentado seguidamente as linhas gerais de execução dos algoritmos implementados.

4.4.1. Método do gradiente

A execução deste método, tal como foi enunciado previamente neste trabalho, tem como principal base a caraterização dos vários elementos através de um potencial. Este potencial irá posteriormente gerar um campo de potencial que exerce uma força sobre o agente.

Sendo assim este método deverá ter em conta os elementos detetados por parte do agente e calcular a força resultante do somatório de cada uma das forças exercidas pelos campos de potencial que atuam sobre aquela posição. No fim o algoritmo deve retornar uma ação que deverá ser realizada pelo agente. Seguindo esta linha podemos definir a execução da seguinte forma.

```
1  A execução inicia-se a partir de uma posição inicial (x, y)
2  Enquanto a posição objetivo não é atingida
3        Considerar um conjunto de elementos do ambiente
         previamente percecionados
4        Calcular somatório das forças exercidas pelos campos de
         potencial gerado pelos elementos percecionados
5        Retornar uma ação definida pela magnitude e ângulo da
         próxima deslocação que o agente deve realizar
```

4.4.2. Deteção e fuga com utilização do algoritmo *wall following*

Este método, referido anteriormente, é composto por dois modos de execução. Numa primeira instância o modo utilizado segue a mesma execução que o método apresentado acima e numa segunda instância utiliza um algoritmo de *wall following*. A decisão de qual o modo em execução recai sobre o cumprimento de premissas previamente definidas.

A nível do módulo *wall following* este foi implementando utilizando os conceitos associados aos campos de potencial. De forma concreta, as forças consideradas dizem apenas respeito aos obstáculos próximos do agente sendo que à componente direção da força exercida pelos campos de potencial repulsivo é adicionada uma direção que mantém o agente a deslocar-se paralelamente ao obstáculo.

Sendo assim é possível partir da definição de execução mencionada acima, a qual vai ser denominada de modo campo de potencial, e adicionando dois novos pontos, a comutação de modo campos de potencial e o modo *wall following*. A execução pode assim ser definida da seguinte forma.

```
A execução inicia-se a partir de uma posição inicial (x, y)
Enquanto a posição objetivo não é atingida
      Validar qual é o modo ativo
            Caso o modo atual seja o modo campo de potencial
                  Validar se o agente se encontra num ótimo
                  local
            Caso o modo atual seja o modo wall following
                  Verificar se o agente já não se encontra na
                  zona associada ao ótimo local
      Se necessário promover a comutação de modo
      Retornar uma ação, tendo em conta o modo ativo, definida
      pela magnitude e ângulo da próxima deslocação que o
      agente deve realizar
```

4.4.3. Algoritmo *avoiding-past*

Como foi evidenciado anteriormente este algoritmo difere dos restantes na medida em que para superar a problemática dos ótimos locais utiliza uma componente de memória. Esta componente é utilizada para afastar o agente de posições previamente percorridas. Além desta, uma quarta força é adicionada, mais concretamente uma força aleatória.

Partindo da definição de execução do método do gradiente definido acima e adicionando esta componente de memória, é possível definir a execução deste método da seguinte forma.

```
A execução inicia-se a partir de uma posição inicial (x, y)
Enquanto a posição objetivo não é atingida
      Guardar em memória a posição atual
      Considerar um conjunto de elementos do ambiente
      Calcular somatório das forças exercidas pelos campos de
      potencial gerado pelos elementos percecionados
      Calcular o somatório das forças exercidas pelos campos
      de potencial virtuais gerados pelas posições anteriores
      Gerar uma força aleatória
      Calcular o somatório de todas as forças
      Retornar uma ação definida pela magnitude e ângulo da
      próxima deslocação que o agente deve realizar, baseada
      na força resultante calculada
```

4.4.4. Método *minimum filling*

Na medida em que a memória pode ser uma solução para a resolução da problemática dos ótimos locais pode recorrer-se a técnicas utilizadas em outras áreas. Um destes caso passa pela implementação do método *minimum filling*.

Como foi mencionado anteriormente este método representa o ambiente duma forma discreta através de uma grelha em que cada célula representa uma posição do ambiente. A nível de pontos comuns, o cálculo do potencial é realizado da mesma forma que o método do gradiente mencionado anteriormente. A nível de aspetos diferenciadores resume-se ao descartar de posições anteriores, não permitindo que estas sejam consideradas no momento da decisão da próxima deslocação. Sendo assim é possível definir a execução deste método da seguinte forma.

```
1  A execução inicia-se a partir de uma posição inicial (x, y)
2  Enquanto a posição objetivo não é atingida
3        Considerar um conjunto de posições composto pelas
         posições vizinhas à posição atual
3        Caso neste conjunto existam posições anteriormente
         percorridas, estas deixam de ser consideradas
4        Calcular somatório das forças exercidas pelos campos de
         potencial que atuam sobre as posições vizinhas
5        Selecionar a posição seguinte considerando o potencial
         mais favorável
6        Gerar uma força que promova a deslocação entre a posição
         atual e a posição selecionada no passo anterior
7        Retornar uma ação definida pela magnitude e ângulo da
         próxima deslocação que o agente deve realizar baseada na
         força gerada no passo anterior
```

4.4.5. Algoritmo *wavefront*

A principal caraterística deste método em relação aos restantes e já mencionada anteriormente, é a não geração de ótimos locais. Tal como o método apresentado acima a representação do ambiente é realizada duma forma discreta através de uma grelha.

A diferença entre este e o método anterior passa pela fórmula de cálculo do potencial associado a cada uma das posições. Neste caso o cálculo é realizado tendo em conta informação global acerca do ambiente e é realizado partindo-se das posições consideradas como objetivo até ao

ponto no qual o agente se encontra. A definição de execução pode ser assim apresentada da seguinte forma.

```
1  Criar representação do global do ambiente
2  Partir da posição considerada como objetivo
3  Enquanto o potencial não é calculado para todas as posições
   do ambiente
4        Considerar um conjunto de posições composto pelas
         posições vizinhas à posição atual
5        Atribuir um valor a cada uma das células vizinhas um
         valor superior ao da posição atual
6  Gerar um percurso desde a posição atual até à posição
   considerada como objetivo
7  Enquanto o percurso gerado não for terminado
8        Gerar uma força que promova a deslocação entre a posição
         atual e a posição seguinte definida no percurso
9        Retornar uma ação definida pela magnitude e ângulo da
         próxima deslocação que o agente deve realizar baseada na
         força gerada no passo anterior
```

4.5. Arquitetura

Após a análise anteriormente apresentada é possível apresentar uma proposta de arquitetura mais informada na qual deverão ser refletidos os pontos comuns e os pontos diferenciadores de entre cada um dos métodos apresentados anteriormente.

Os pontos evidenciados dessa análise demonstram que existem relações entre os algoritmos que podem ser transpostas para a arquitetura, nomeadamente a transversalidade do método do gradiente já que os restantes algoritmos usam este método como base da sua execução. Partindo desta premissa é proposta a seguinte arquitetura (Diagrama 2).

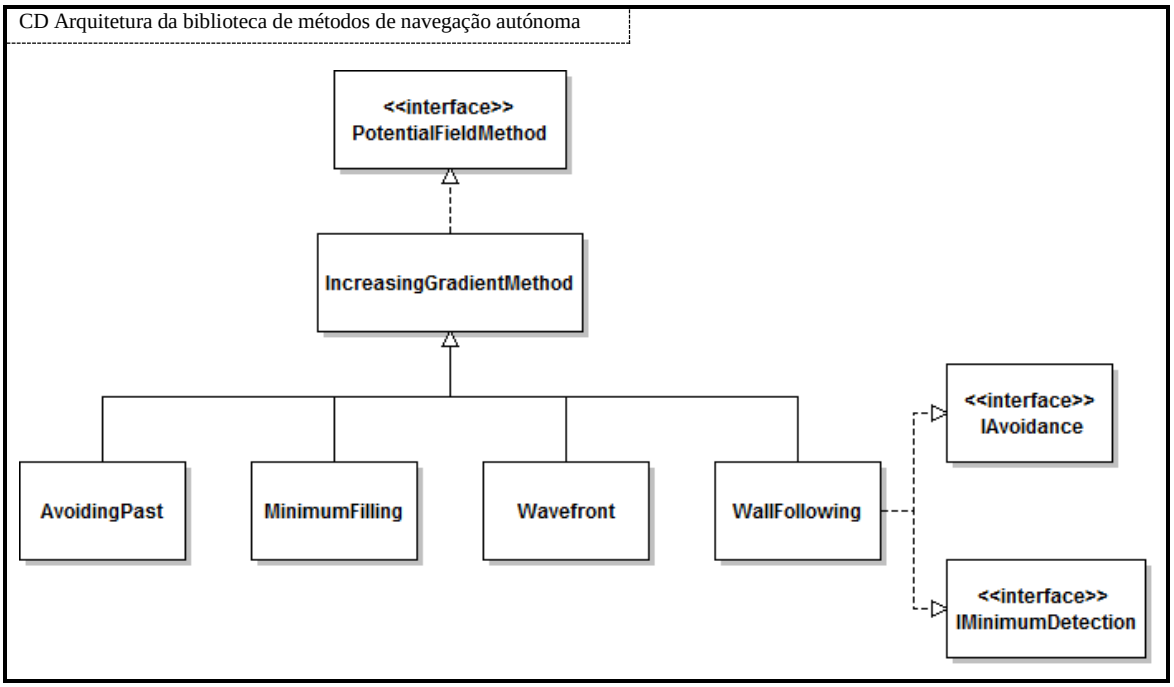


Diagrama 2 – Arquitetura da biblioteca de métodos de navegação autónoma baseados em campos de potencial

O ponto de integração exposto por esta arquitetura incide na definição da interface *PotentialFieldMethod*. Esta é definida apenas por um método, *navigate*, o qual deve ser implementado em concreto pelas restantes classes. Do ponto de vista do agente que encapsula o método em cada iteração apenas tem de indicar o contexto de execução e invocar este método.

Um ponto não evidenciado acima passa pela definição de algoritmos baseados em deteção e fuga de ótimos locais como é o caso do método no qual existe a utilização do algoritmo *wall following*. Caso se pretendam adicionar novos algoritmos que sigam esta abordagem eles devem estender das interfaces *IAvoidance* e *IMinimumDetection*.

Com esta proposta de arquitetura é pretende-se manter a modularidade necessária e o suporte à mudança necessário para que a adição e/ou alteração de novos métodos causem o menor número de impactos possível em relação ao que se encontra desenvolvido até ao momento.

4.6. Implementação

4.6.1. Método do gradiente

A implementação deste algoritmo visa refletir a solução proposta por *Khatib* (30). Duma forma geral o agente segue os potenciais que considera mais favoráveis. No presente trabalho este caraterística é concretizada através de potenciais crescentes, sendo a posição objetivo detentora do potencial mais elevado do ambiente.

Tal como vimos nos capítulos anteriores, a ação realizada pelo agente depende das forças exercidas sob ele pelos campos de potencial gerados pelos potenciais associados aos elementos do ambiente, sendo aqui definidas por um vetor com a representação $(direcção, magnitude)$.

A direção encontra-se definida como o ângulo α, entre o agente e o elemento a ser considerado. Caso o elemento em questão seja um obstáculo é adicionado ao ângulo α o valor π, de forma a força ser repulsiva.

$$\text{Dir} = \begin{cases} \alpha & , força\ atrativa \\ \alpha + \pi & , força\ repulsiva \end{cases} \tag{24}$$

A formulação usada para definir a magnitude das forças atrativas e das forças repulsivas é a seguinte:

$$\text{Mag} = \begin{cases} \dfrac{i_{max} * (d_{max} - d)}{d_{max}} & , d < d_{max} \\ 0 & , d \geq d_{max} \end{cases} \tag{25}$$

Sendo Mag a magnitude da força, i_{max} a intensidade máxima da força, d_{max} a distância máxima entre o agente e o elemento, o que define também a distância a que um campo de potencial pode exercer uma força, e d a distância entre a posição atual do agente e o elemento. Esta definição corresponde a um perfil de magnitude linear.

4.6.2. Deteção e fuga com utilização do algoritmo *wall following*

Como evidenciado anteriormente, o método apresentado acima apresenta ótimos locais o que leva ao insucesso da navegação. Devido a isto outras soluções foram apresentadas entre as quais a deteção de ótimos locais e a fuga de ótimos locais. Apesar de serem abordagens separadas a sua conjugação, como é este o caso, leva à criação de soluções capazes de ultrapassar o problema dos ótimos locais.

O desenvolvimento de um algoritmo baseado em deteção e fuga apresenta, normalmente, a implementação de três pontos base:

- Modo de navegação baseado em campos de potencial;
- Modo de realização de fuga de ótimos locais;
- Método de gestão das comutações entre modos.

Nesta implementação o primeiro modo é concretizado através da utilização do método do gradiente definido anteriormente. Em relação ao segundo modo será implementado através de um algoritmo *wall following*.

Em relação ao mecanismo de comutação implementado é baseado nas seguintes verificações:

- Se o agente se encontrar no modo de navegação baseado em campos de potencial
    - Caso a componente magnitude do vetor representativo da força for inferior a um valor α é realizada uma comutação para o modo de realização de fuga de ótimos locais.
- Se o agente se encontrar no modo de realização de fuga de ótimos locais
    - Caso um conjunto β de iterações, em que cada iteração apresenta um valor da componente magnitude do vetor representativo da força sucessivamente superior à iteração anterior é realizada a comutação para o modo de navegação baseado em campos de potencial

4.6.3. Algoritmo *avoiding-past*

A utilização de memória acabou por ter um papel importante no desenvolvimento de métodos de navegação baseados em campos de potencial. Apesar de numa primeira fase este tipo de solução fosse de evitar pelo consumo de recursos computacionais que poderiam ter de ser usados, foi possível mais tarde demonstrar que a sua utilização poderia possibilitar diversas soluções e sem um uso extensivo de memória.

Na implementação, esta utilização recai na memória de posições anteriormente percorridas pelo agente. Estas memórias vão ser utilizadas mais tarde para que o agente se afaste de zonas anteriormente percorridas. Esta componente é posteriormente adicionada à força resultante.

Além da força gerada pela memória existe uma outra força que é adicionada à força resultante que é utilizada para a navegação do agente. Esta componente é uma força aleatória que pode ser usada para desbloquear situações geradas pelas outras forças. Caso o agente se encontre a colidir, esta força pode sofrer um incremento que afasta o agente da zona problemática.

Sendo assim o cálculo da força resultante é formalmente definido da seguinte forma:

$$\mathrm{F_{res}} = \sum F_{atr} + \sum F_{rep} + F_{mem} + F_{alt} \quad (26)$$

Sendo $\mathrm{F_{res}}$ a força resultante, $\sum F_{atr}$ representa o somatório das forças atrativas, $\sum F_{rep}$ representa o somatório das forças repulsivas, F_{mem} a força gerada pela memória e F_{alt} a força aleatória.

4.6.4. Método *minimum filling*

O preenchimento de ótimos locais é uma das formas de ultrapassar os problemas existentes na navegação baseada em campos de potencial. Duas implementações possíveis são a atribuição de potenciais repulsivos a posições consideradas como ótimos locais e a exclusão dessas posições mediante serem marcadas como analisadas.

A opção implementada passa pela segunda hipótese. Na solução desenvolvida o ambiente passa a ser considerado como discreto passando a ser visto como uma grelha. O principal aspeto desta abordagem relaciona-se com o facto de que as posições são analisadas uma única vez.

Caso o agente se encontre num beco a execução do algoritmo vai levar à exclusão de todos os pontos que fazem parte dessa zona. Após isso o agente sairá dessa zona e vai continuar o seu percurso até ao objetivo.

4.6.5. Algoritmo *wavefront*

Este algoritmo tem como principal caraterística a não geração de ótimos locais, mas obriga à necessidade de processar o ambiente previamente.

Este processamento é realizado partindo da posição tida como objetivo e vão sendo atribuídos valores a todas as posições do ambiente. De notar que, tal como na abordagem anterior, o ambiente é discreto. À posição do objetivo é atribuído o valor zero, sendo atribuído às posições vizinhas um valor incrementado em relação ao da posição anterior e assim sucessivamente até todo o ambiente se encontrar considerado.

Apesar da criação de uma representação do ambiente livre de ótimos locais o agente pode não conseguir passar em alguns pontos e pode encontrar dificuldades nos cantos dos obstáculos. Sendo assim é necessário que nos pontos considerados o agente tenha espaço suficiente para passar ou manobrar. Nesta implementação isto é conseguido através da análise ao redor da posição sendo que deve existir uma área de dimensão a livre de obstáculos.

Um dos passos associado com este algoritmo incide na geração das posições vizinhas. Tendo em conta que que o ambiente é representado duma forma discreta as posições vizinhas a considerar poderiam ser aquelas que se encontram imediatamente ao lado. Por exemplo, considerando uma posição (x, y) as posições vizinhas seriam $(x-1, y)$, (x, y), $(x+1, y)$ e assim sucessivamente.

Na presente implementação é possível considerar áreas através da geração de posições vizinhas a uma maior distância através da adição de uma constante c à componente. Exemplo, considerando uma posição (x, y) as posições vizinhas seriam $(x-c, y)$, (x, y), $(x+c, y)$ e assim sucessivamente.

4.6.6. Concretização da arquitetura

Tendo em conta a arquitetura apresentada acima, Diagrama 3, e aspetos enunciados na secção anterior, 4.6, é possível apresentar um modelo que agrupa esta informação (Diagrama 3).

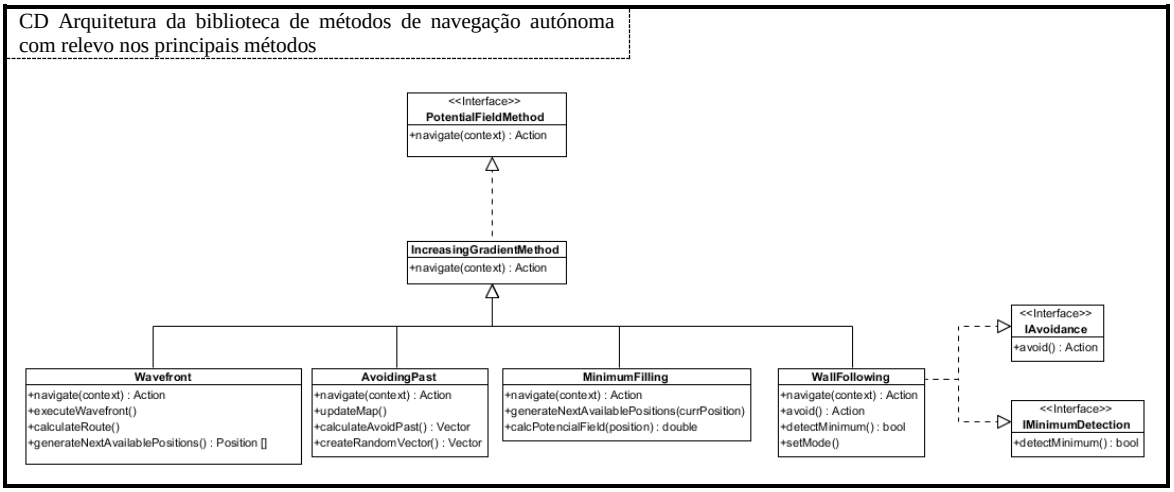


Diagrama 3 – Diagrama de classes da biblioteca de métodos baseados em campos de potencial, sendo relevante a indicação dos principais métodos de cada implementação

Analisando o diagrama, o aspeto central passa pela definição do método *navigate*, o qual é comum a todas as classes definidas. A sua definição, tal como mencionado anteriormente, que permite a implementação dos métodos de navegação baseados em campos de potencial selecionados.

Segue-se a especificação dos métodos definidos na arquitetura.

Nome do método de navegação	Método do gradiente
Classe	*IncreasingGradientMethod*
Método	Descrição
navigate	É realizada a soma das forças que atuam sobre o agente, sendo retornada uma ação

Nome do método de navegação	Método de deteção e fuga baseada no algoritmo *wall following*
Classe	*WallFollowing*
Método	Descrição
navigate	Executa o método responsável pela avaliação do modo em que o agente deve navegar e executa o método de navegação associado a esse modo
avoid	Método que executa a navegação do agente baseada no método *wall following*
detectMinium	Verifica se o agente se encontra num ótimo local
terminateWallFollowing	Verifica se as últimas deslocações do agente já permitem terminar o método de *wall following*
setMode	Analisa qual o modo de navegação a utilizar

Nome do método de navegação	Método baseado no algoritmo *avoiding-past*
Classe	*AvoidingPast*
Método	Descrição
navigate	Executa o método que atualiza a informação acerca das posições anteriores, o método que calcula a repulsão da exercida pela memória e executa o método que gera a força aleatória
updateMap	Atualiza a informação relacionada com as posições anteriores
calculateAvoidPast	Calcula a força exercida pela memória associada às posições anteriores
generateRandomVector	Geração do vetor que define uma força aleatória, sendo multiplicada por um fator caso o agente se encontre a colidir com um elemento do ambiente

Nome do método de navegação	Preenchimento de mínimos
Classe	*MinimumFilling*
Método	Descrição
navigate	Executa o método responsável pela geração das posições seguintes e o método responsável pelo cálculo da força exercida sob cada uma das posições. No fim calcula o a força necessária para que o agente se desloque entre a posição atual e a próxima
generateNextAvailablePositions	Geração das posições seguintes
calculatePotentialField	Cálculo da força exercida sob uma determinada posição

Nome do método de navegação	Método baseado no algoritmo *wavefront*
Classe	*Wavefront*
Método	Descrição
navigate	Caso o percurso não se encontre definido executa o método que implementa o algoritmo *wavefront* e o método que gera o percurso. Após esta fase o método gera as ações necessárias para seguir o percurso gerado
executeWavefront	Implementação do algoritmo w*avefront*
calculateRoute	Geração do percurso tendo em conta o resultado da execução do algoritmo *wavefront* sob o ambiente
generateNextAvailablePositions	Método utilizado pelo algoritmo *wavefront* para obtenção das posições vizinhas à posição atual

4.7. Resumo

A biblioteca de métodos baseados em campos de potencial pretende oferecer um conjunto de algoritmos baseados em campos de potencial sendo que os algoritmos que a constituem pretendem demonstrar um conjunto das abordagens encontradas nesta área.

De forma a permitir a adição de novos métodos e a alteração dos atuais a biblioteca deve oferecer uma estrutura que seja capaz de suportar a mudança ao longo do tempo, logo é necessário compreender os pontos transversais aos algoritmos.

De forma a validar os algoritmos é necessária uma plataforma de testes, a qual vai ser apresentada no capítulo seguinte.

5. Plataforma de teste

5.1. Visão

O presente trabalho é constituído por duas componentes. A primeira componente é a biblioteca de métodos de navegação com base em campos de potencial, apresentada no capítulo anterior, e a segunda componente consiste numa plataforma de testes.

Esta segunda componente deve ser utilizada para validação da utilização dos algoritmos. Dentro das suas funcionalidades inclui-se a execução do agente selecionado, integrado num ambiente obtido a partir de um ficheiro fornecido pelo utilizador.

Desta forma, a utilização desta plataforma permite ao utilizador avaliar a execução dos métodos, verificando o resultado de navegação obtido para os ambientes de teste fornecidos.

5.2. Especificação de requisitos

A plataforma de teste é composta por um conjunto de funcionalidades, as quais oferecem aos utilizadores a possibilidade de validar as abordagens utilizadas para a solução da problemática da navegação autónoma através da aplicação de métodos baseados em campos de potencial. Este conjunto de funcionalidades é disponibilizado ao utilizador dependendo do papel que se lhe encontra atribuído.

A utilização da plataforma pode ser dividida em duas fases:

- Fase de testes;
- Fase de utilização final.

Na primeira fase é realizada a validação efetiva de todos os desenvolvimentos, sendo necessário que sejam realizados testes sobre a plataforma. Deste ponto é possível identificar o primeiro ator:

- Utilizador de teste

Na fase de utilização final a plataforma será disponibilizada para outros utilizadores, sendo estes atores identificados como:

- Utilizador final

Ambos os atores identificados têm como principal responsabilidade reportar erros que se encontram na aplicação ou indicar alterações que visam o desenvolvimento de melhorias.

5.2.1. Modelo de casos de utilização

De uma forma geral, os atores antes de realizar a execução do agente devem selecionar o ambiente e selecionar o agente. A execução do agente selecionado irá refletir o algoritmo que se encontra na sua génese.

Disto podemos retirar o seguinte modelo (Figura 34) no qual é possível mapear as relações entre os utilizadores e o sistema e as relações entre os casos de utilização.

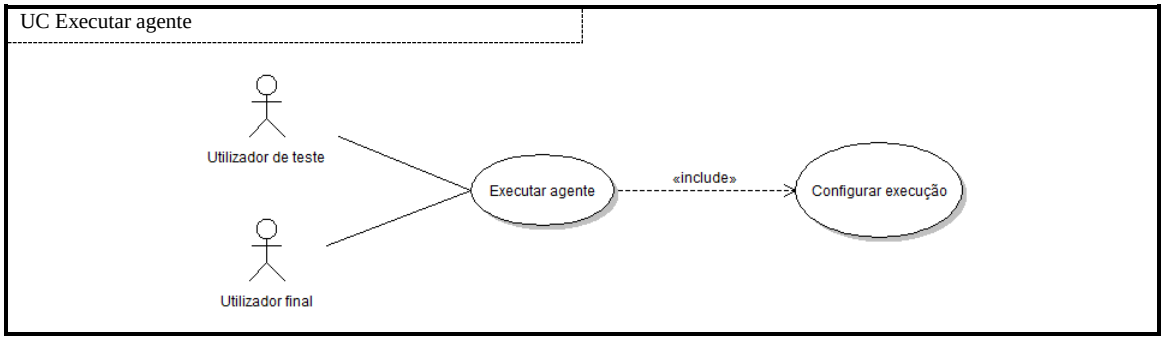


Figura 34 – Modelo de casos de utilização

Os casos de utilização permitem descrever as funcionalidades disponibilizadas para os atores identificados. Esta descrição visa sobretudo a análise das várias interações entre o utilizador e o sistema.

5.2.1.1. Executar agente

Resumo		
	O utilizador pretende validar a execução de um agente dentro de um determinado ambiente	
Ator		
	Utilizador de teste ou utilizador final	
Cenário principal		
	Ação do ator	Ação do sistema
1	Indica o ambiente através do ficheiro de configuração	
2	Indica o agente a utilizar na execução através do ficheiro de configuração	
3	Utilizador executa ficheiro main.exe	
4		O sistema apresenta um ecrã onde decorre a execução do agente integrado no ambiente

5.2.1.2. Configurar execução

Resumo		
	O utilizador pretende efetuar as configurações para a execução do agente	
Ator		
	Utilizador de teste ou utilizador final	
Cenário principal		
	Ação do ator	Ação do sistema
1	Utilizador acede ao ficheiro de configuração	
2	Utilizador configura ambiente	
3	Utilizador configura agente	

5.3. Arquitetura

5.3.1. Organização Geral

A plataforma de teste é essencialmente composta por quatro elementos sendo o simulador responsável pela coordenação dos restantes elementos presentes. A plataforma além de constituída por estes elementos incorpora a utilização de duas bibliotecas sendo que a primeira foi desenvolvida durante este trabalho, a biblioteca de métodos de navegação autónoma baseados em campos de potencial, e a segunda é a biblioteca *pygame* utilizada para a apresentação da execução do algoritmo ao utilizador. Tendo em conta os pontos mencionados a plataforma de teste é representada da seguinte forma (Diagrama 4).

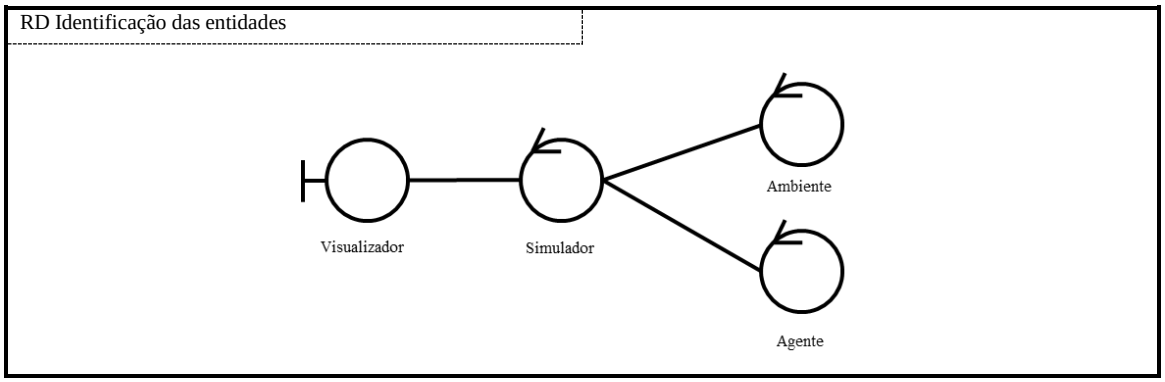


Diagrama 4 – Diagrama representativo dos papéis atribuídos a cada elemento

Partindo do diagrama de robustez apresentado anteriormente e transpondo as principais ideias evidenciadas na fase de análise para o modelo de projeto, a implementação da plataforma de teste pode ser então representada através das classes base (Diagrama 5).

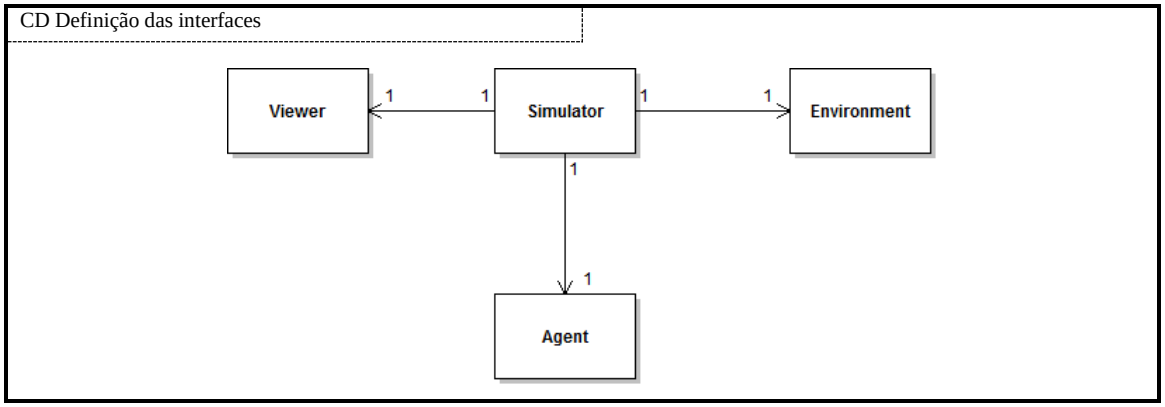


Diagrama 5 – Diagrama de classes das componentes pertencentes à plataforma de teste

5.3.2. Concretização dos casos de utilização e interação entre componentes

Partindo dos casos de utilização identificados e as componentes que servem de suporte às funcionalidades que permitem a concretização dos casos de utilização é necessário compreender a forma como estas interagem entre si.

Do ponto de vista do utilizador a execução é iniciada sendo-lhe apresentado um ecrã onde é possível visualizar a navegação do agente selecionado tendo em conta o ambiente no qual este se integra.

Do ponto de vista do sistema existe uma primeira fase, a qual pode ser denominada de fase de parametrização, em que existe a configuração de todos os elementos necessários para a execução do agente. Posteriormente existe uma segunda fase, a qual pode ser denominada de fase de execução, em que os diversos componentes interagem para cumprir a execução do algoritmo.

Estes pontos podem então ser descritos da seguinte forma (Diagrama 6).

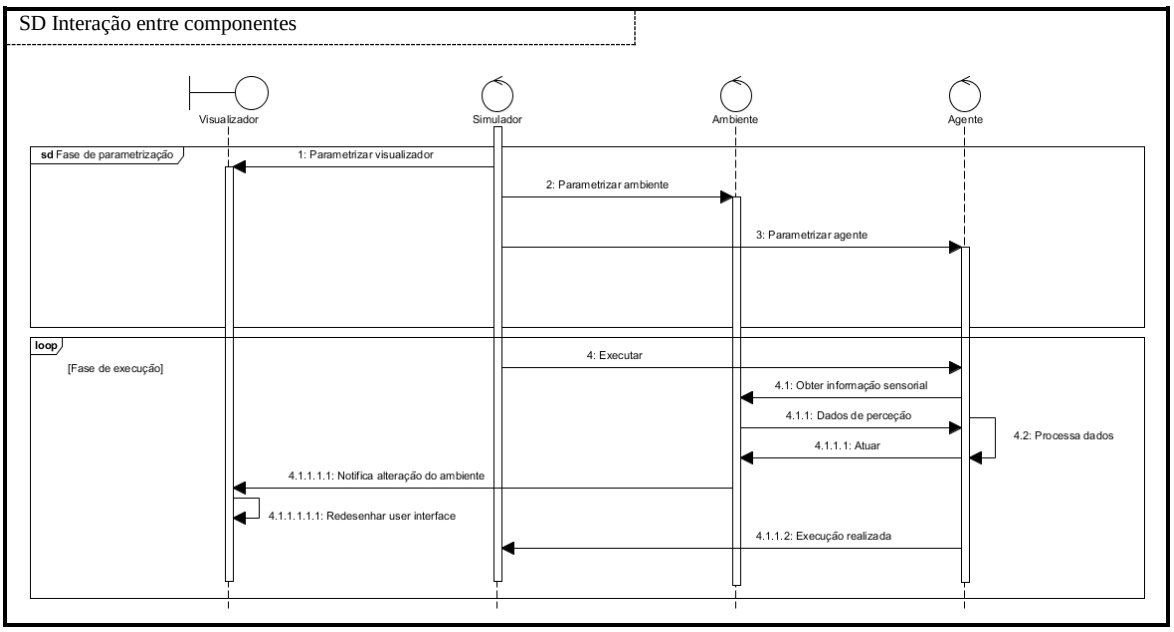


Diagrama 6 – Interação entre as várias componentes para a concretização dos casos de utilização

Da análise deste diagrama é de destacar a centralidade que é dada ao simulador visto ser este controlo o responsável pela parametrização das restantes entidades e controlo da execução. Apesar disso as restantes interações não necessitam da intervenção do simulador sendo que as entidades nelas envolvidas interagem de forma independente.

5.4. Implementação dos métodos de navegação autónoma

5.4.1. Simulador

Como evidenciado anteriormente a entidade Simulador é responsável pela orquestração entre as restantes entidades identificadas sendo o ponto centralizador da execução de toda a plataforma.

A sua concretização deve incluir os seguintes pontos:

- Parametrização do ambiente;
- Parametrização dos agentes;
- Parametrização do visualizador;
- Iniciar a execução.

Sendo assim e partindo do modelo anteriormente apresentado (Diagrama 7) a seguinte hierarquia pode ser apresentada como forma de cumprir os requisitos indicados acima.

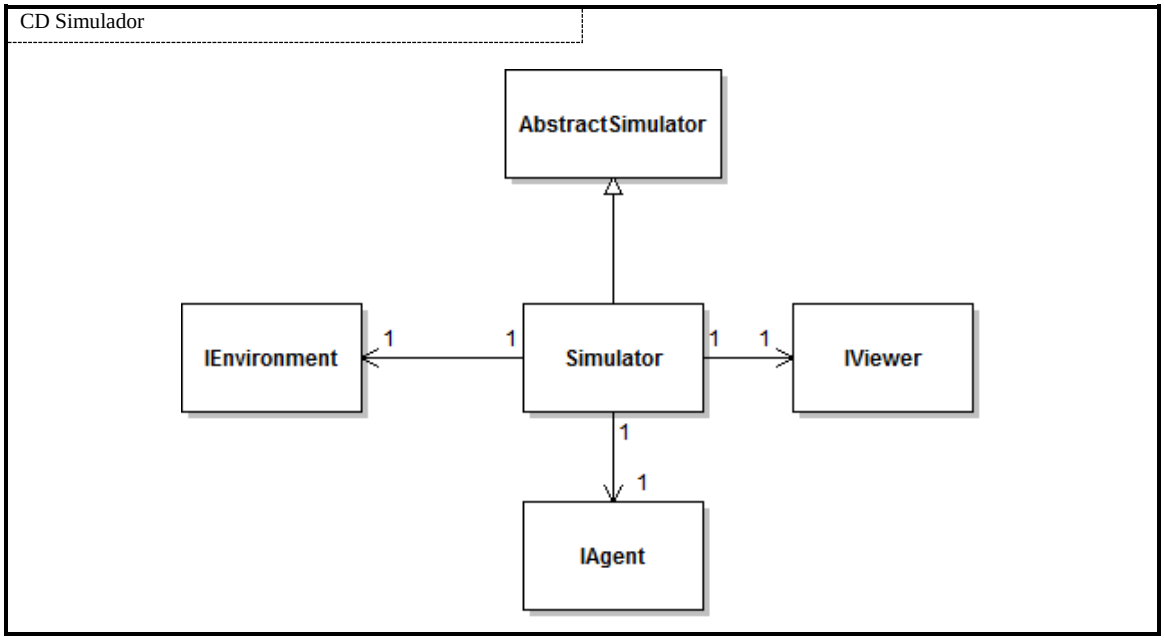


Diagrama 7 – Diagrama de classes para a classe *Simulator* incluindo as componentes com as quais se relaciona

Sendo a modularidade um ponto transversal no desenvolvimento do presente projeto o simulador, concretizado através das classes *AbstractSimulator* e *Simulator*, deverá apresentar características que facilitem a sua alteração ou substituição. Devido a esta necessidade a classe *AbstractSimulator* encontra-se responsável pela definição do conjunto de funções necessárias para a ligação entre os vários elementos. Por sua vez a classe *Simulator* encontra-se responsável pela implementação dos métodos.

5.4.2. Ambiente

Esta componente é responsável pela representação do ambiente e pela concretização do ambiente no qual o agente se integra. Sendo assim deve encapsular formas de obter a partir do ambiente a informação necessária.

Neste trabalho o ambiente é fornecido como uma imagem sendo depois tratada através da biblioteca *pygame*. A representação do ambiente deve encapsular um conjunto de métodos que tiram partido da biblioteca *pygame* para obtenção de informação acerca do ambiente.

Para o cumprimento destes pontos é proposta a seguinte arquitetura (Diagrama 8).

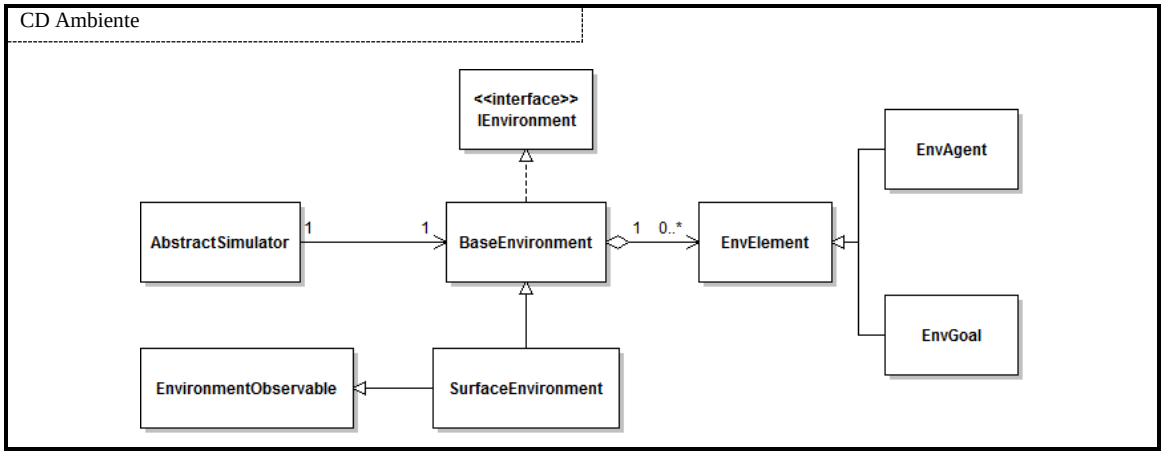


Diagrama 8 – Diagrama de classes associado à entidade Ambiente, concretizada na classe *BaseEnvironment*, e relações com as restantes componentes existentes

A classe *BaseEnvironment* é responsável pela definição de todas funções associadas com o ambiente. No âmbito do presente projeto as funções aqui definidas dividem-se essencialmente em dois grupos. O primeiro é composto por funções que deverão possibilitar a extração de informação incluída no ambiente e o segundo deverá representar a realização das ações sobre o ambiente. Nestas funções incluem-se:

- Obter informação relacionada com o agente;
- Obter informação relacionada com elementos do ambiente;
- Obter informação relacionada com os objetivos;
- Mover o agente.

De forma a manter características de modularidade para suportar eventuais alterações ao longo do tempo a implementação desta definição encontra-se a cargo das classes que encapsulam os ambientes. Neste caso esta responsabilidade recai na classe *SurfaceEnvironment*.

De forma a representar os elementos existentes no ambiente recorre à sua representação através da classe *EnvElement* sendo esta por sua vez concretizada pelos objetos do tipo *EnvGoal* e

EnvAgent que visam representar, respetivamente, os objetivos e os agentes existentes no ambiente.

De notar que falta abordar a componente *EnvironmentObservable* sendo esta componente responsável pela notificação de entidades que dependam das alterações do estado do ambiente. Este tema será novamente abordado posteriormente.

5.4.3. Agente

Esta entidade visa o encapsulamento das várias implementações de agentes sobre algoritmos baseados em campos de potencial. Logo esta componente terá de providenciar todos os meios necessários para que o algoritmo execute corretamente. Dentro destes é necessário contemplar os sensores e atuadores que, respetivamente, providenciam informação do ambiente e permitem atuar sobre o ambiente.

A nível da arquitetura, a concretização desta entidade pode ser realizada através de uma interface e uma classe base a qual define os pontos comuns a todos os algoritmos desenvolvidos. Esta classe base será denominada *AbstractAgent* e será dela que as restantes implementações vão derivar.

De forma a cumprir os requisitos de interação com o ambiente e requisitos de modularidade é proposta a seguinte hierarquia (Diagrama 9).

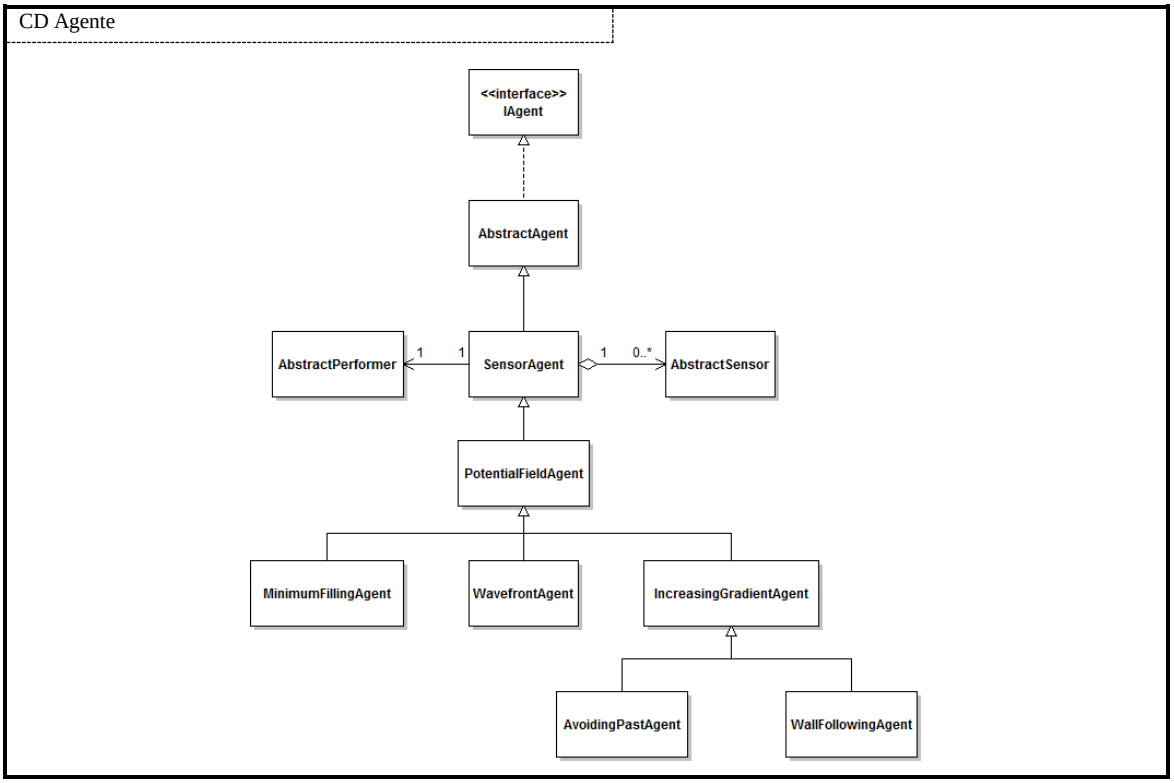


Diagrama 9 – Diagrama de classes associado ao agente e relações com outras entidades

Duma forma geral apresentam-se aqui dois tipos de entidades. As entidades *AbstractAgent* e *SensorAgent* como pontos agregadores da definição das funcionalidades entre os vários agentes desenvolvidos e as restantes entidades que visam o encapsulamento dos vários algoritmos selecionados, presentes na biblioteca de algoritmos baseados em campos de potencial.

O ponto comum entre estes agentes é a sua execução. Esta é definida por um conjunto de fases desde a obtenção da informação até à realização da ação. Isto levou à formalização do seguinte diagrama de estados (Diagrama 10).

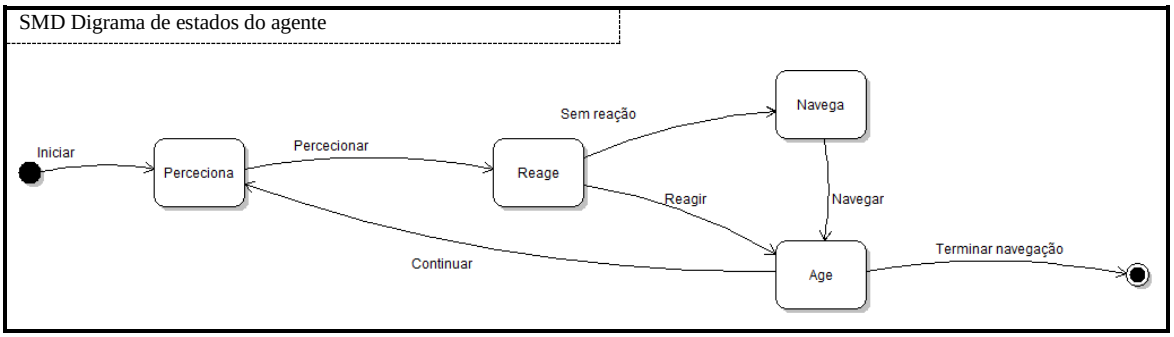


Diagrama 10 – Diagrama de estados representativo da execução do agente

Com isto entende-se que a execução do agente é suportada pelas seguintes fases:

- Perceciona – nesta fase o agente deverá efetuar a recolha da informação a partir do ambiente sendo que esta recolha pode ser, por exemplo, efetuada através de sensores.
- Reage – nesta fase o agente deve gerar uma ação caso se encontre numa situação contemplada. É disto exemplo a proximidade ao objetivo que caso o agente se encontre próximo é gerada uma ação que lhe permita terminar a navegação.
- Navega – nesta fase só é executada se na fase reagir não tiver sido gerada uma ação. Nessa situação o algoritmo que se encontra na génese do agente deve ser executado de forma a produzir uma ação.
- Age – nesta fase o agente deve executar a ação gerada na fase de processamento. De forma a cumprir este objetivo podem ser utilizados atuadores.

De notar que nesta seção não foram abordadas as entidades *AbstractPerformer* e *AbstractSensor* as quais serão abordadas seguidamente.

5.4.4. Integração entre agente e ambiente

A obtenção de informação para utilização durante a execução da navegação e a realização de ações sobre o ambiente são os dois pontos de interação entre o ambiente e o agente. Daqui compreende-se a importância de identificar as entidades responsáveis pela concretização destas funcionalidades no agente.

5.4.4.1. Visão global

As entidades responsáveis pelas funcionalidades mencionadas acima são concretamente sensores e atuadores. Tendo como ponto de partida as entidades agente e ambiente especificadas acima e de forma a permitir a integração dos sensores e atuadores é proposta a seguinte arquitetura (Diagrama 11).

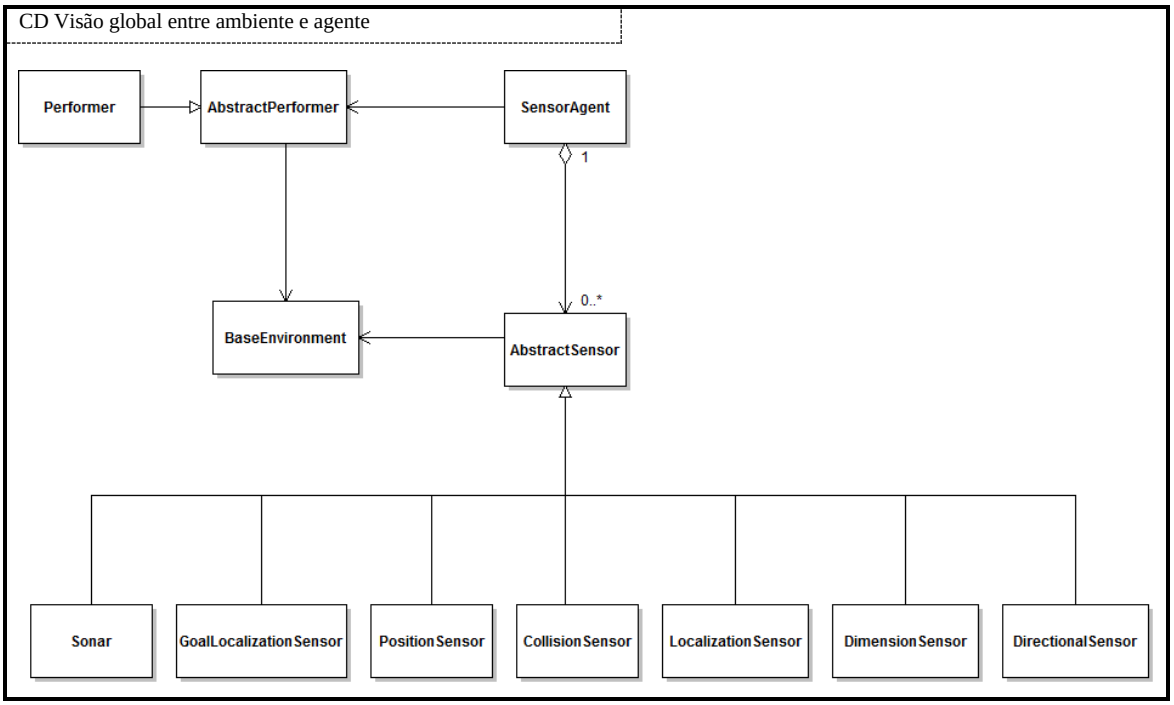


Diagrama 11 – Arquitetura geral de integração entre as entidades agente e ambiente e relações entre si

Nesta arquitetura é de notar que a definição das funcionalidades associadas aos atuadores centram-se na classe base *AbstractPerformer* e definição das funcionalidades associadas aos sensores derivam da classe base *AbstractSensor*. As concretizações associadas a estas classes base permitem capacitar o agente de formas de interagir com o ambiente.

Com esta arquitetura pretende-se modelar as várias componentes e fornecer um grau de abstração de forma a minimizar as alterações que possam vir a ser efetuadas. Segue-se a especificação de cada uma das entidades.

5.4.4.2. Sensor

Por forma a gerar ações o agente tem de possuir uma forma de obter informação proveniente do ambiente. Caso este agente se concretizasse num *robot* real tal necessidade seria suprimida, por exemplo, pela inclusão de sensores. Sendo assim é também importante a adição desse conceito à arquitetura desenvolvida até ao momento.

Dependendo da informação que pode ser necessária obter por parte do agente a partir do ambiente, pode existir a necessidade do agente incorporar diversos tipos de sensores. Isto leva a necessidade de modelar estes elementos de uma forma que o impacto da sua substituição ou alteração seja menor.

Para o cumprimento dos requisitos identificados acima é proposta a seguinte hierarquia (Diagrama 12) tendo como base a entidade *AbstractSensor* a qual define todos os pontos transversais aos sensores desenvolvidos.

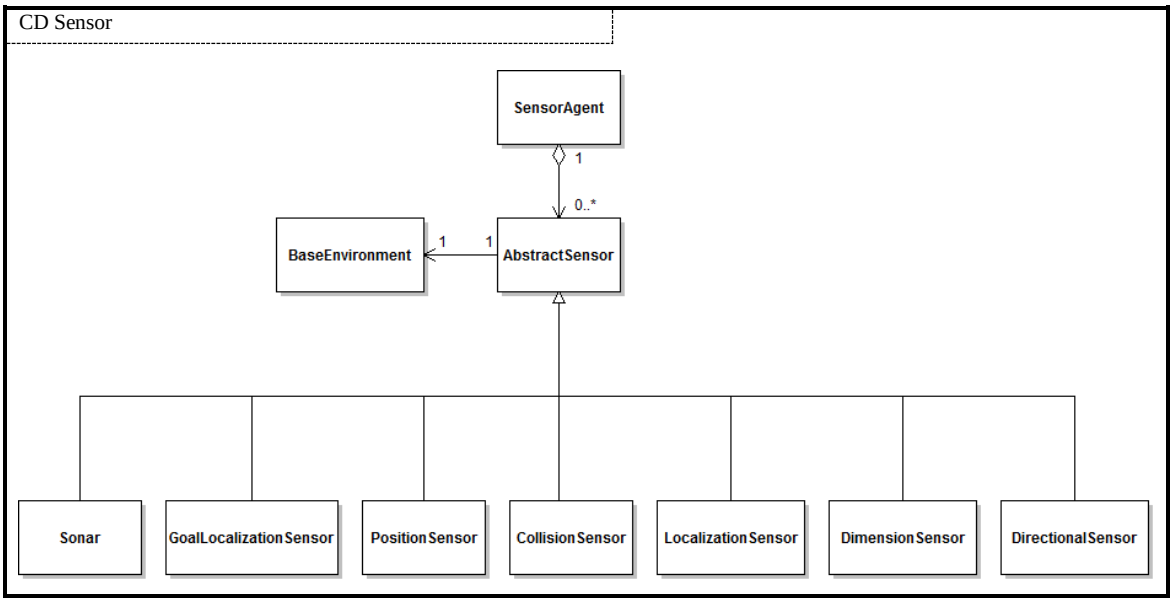


Diagrama 12 – Diagrama de classes associadas com a obtenção de informação a partir do ambiente

O diagrama acima permite observar as relações entre os vários componentes responsáveis pela obtenção de informação a partir do ambiente e que o agente vai utilizar posteriormente. Para os algoritmos desenvolvidos foi necessário a concretização dos seguintes sensores.

- *Sonar* – permite a obtenção de informação tendo em conta o ambiente na sua globalidade;
- *GoalLocalizationSensor* – permite a obtenção de informação relacionada com a localização do objetivo;
- *PositionSensor* – obtenção de informação tendo em conta uma determinada posição;
- *CollisionSensor* – verifica se o agente se encontra a colidir com outro elemento do ambiente;
- *LocalizationSensor* – permite a obtenção de informação acerca da posição atual do agente;
- *DimensionSensor* – permite a obtenção da dimensão do ambiente;
- *DirectionalSensor* – permite a obtenção de informação relacionada com a direção que o agente se encontra a seguir.

Estes componentes podem ser adicionados ao agente não existindo uma obrigatoriedade em relação a cada um deles. Desta forma durante a implementação do agente deve ser analisada qual a informação necessária para o sucesso da execução do agente.

5.4.4.3. Atuador

Tal como foi mencionado acima, a realização de ações sobre o ambiente é outro dos pontos de interação entre as componentes agente e ambiente. Esta responsabilidade é atribuída à entidade atuador que perante o cenário atual apenas necessita de providenciar a concretização das seguintes ações:

- Mover agente;
- Agarrar alvo.

A realização destas ações pode depender do agente e do ambiente. Sendo assim é necessária criar abstração a nível do atuador criando uma classe para a definição das funções associadas com este tipo de objetos e uma segunda classe onde são implementadas as funções definidas acima. A hierarquia proposta abaixo (Diagrama 13) visa o cumprimento dos requisitos definidos.

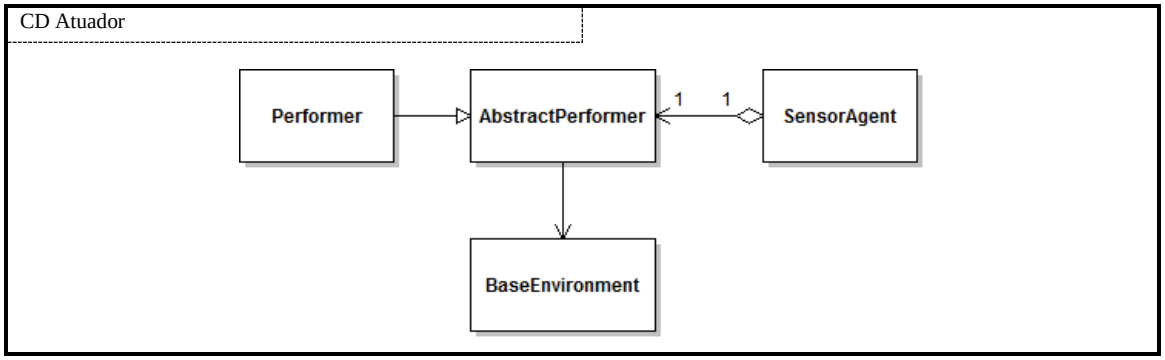


Diagrama 13 – Diagramas de classes relacionado com a entidade atuador e relações com as restantes entidades

5.4.5. Visualizador

Esta componente tem como responsabilidade fornecer ao utilizador informação visual acerca da execução do agente selecionado tendo em conta a parametrização realizada. De forma a cumprir este requisito é necessário existir a representação dos elementos do ambiente a nível da componente gráfica. Sendo assim é proposta a seguinte arquitetura (Diagrama 14).

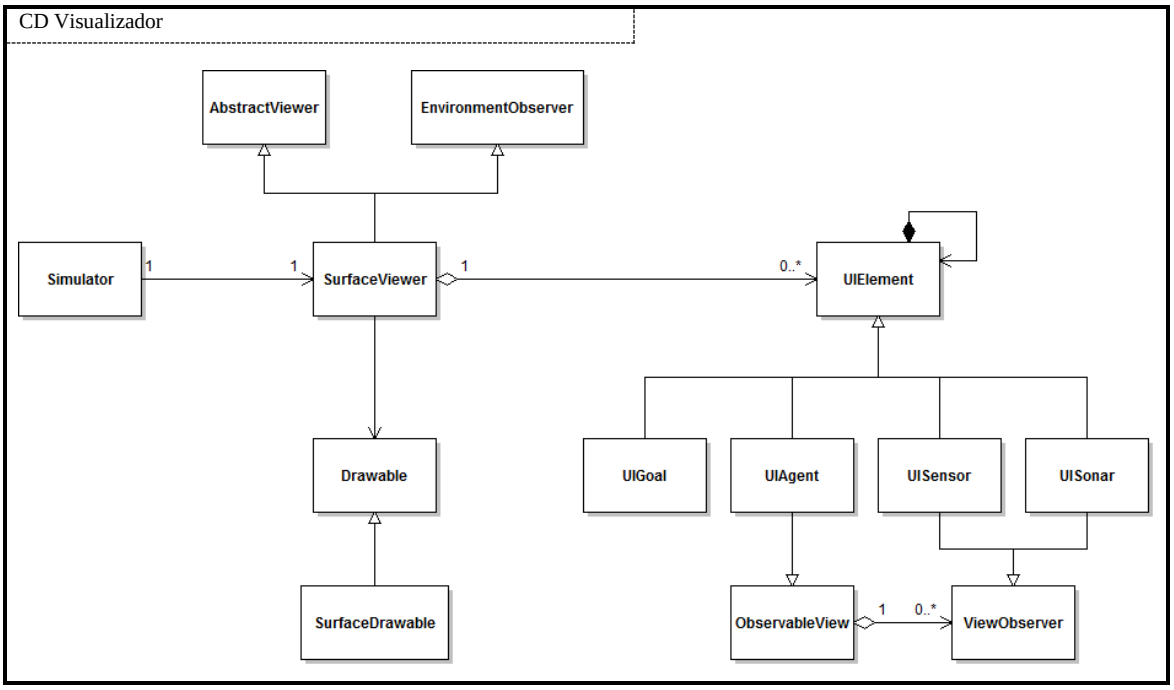


Diagrama 14 – Diagrama de classes associado ao visualizador e relações com as restantes componentes

Tal como nos restantes elementos também o visualizador deve constituir um módulo que permita facilitar eventuais alterações. De forma a cumprir este objetivo também aqui deve existir uma classe que defina um conjunto suficiente de funções para que seja possível a apresentação da execução do agente ao utilizador sendo este ponto concretizado através da classe *AbstractViewer*. A implementação desta interface é posteriormente da responsabilidade da classe *SurfaceViewer* a qual possui na sua génese a biblioteca *pygame*.

De forma a representar o ambiente existe a definição de objetos definidos pela classe *UIElement*. Estas entidades devem representar todos os elementos existentes no ambiente e elementos que fazem parte deles. Sendo assim existe a definição dos objetos objetivo, agente, sensor e sonar.

Outro ponto que há que considerar é a relação entre os elementos indicados acima, nomeadamente a dependência entre eles. Esta situação acontece para os objetos que representam o agente, o sensor e o sonar visto, que sempre que o agente se move os restantes elementos devem desenhar-se novamente. Daqui definiu-se a utilização de um padrão *observer* para que sempre que o agente altere a sua posição os elementos dependentes recebam essa notificação e atualizem os dados necessários para que se voltem a desenhar no ecrã.

A entidade visualizador é tida como o gestor de toda a componente gráfica o que lhe incute a responsabilidade de indicar onde os elementos representativos do ambiente se devem apresentar.

Deste ponto nasceu a definição das classes *Drawable* e *SurfaceDrawable* as quais fornecem acesso ao ecrã na qual os restantes elementos se apresentam.

Em relação à classe *EnvironmentObserver*, a qual permite obter notificações acerca da alteração de estado provenientes do ambiente, será um tema posteriormente abordado.

5.4.6. Integração entre ambiente e visualizador

Como já foi evidenciado existem pontos na plataforma onde é necessária uma troca de informação entre as partes para atualização de dados. Uma destas situações foi anteriormente evidenciada em relação aos elementos gráficas que representam o ambiente.

Durante a execução do algoritmo o agente desloca-se através do ambiente e estas deslocações devem ser apresentadas ao utilizador. É assim necessário existir algum tipo de sincronização entre os dados existentes nas entidades de controlo e os dados apresentados na entidade de fronteira concretizada no visualizador.

Sendo assim, recorreu-se à aplicação do padrão *observer* como forma de cumprir este requisito. O ambiente, entidade controlo, deve estender uma classe que permite o registo de eventuais interessados em alterações de dados. Encontrando-se o visualizador nesta situação, este deve estender a classe que o identifica como interessado nas alterações. De forma a cumprir estes pontos é proposta a seguinte arquitetura (Diagrama 15).

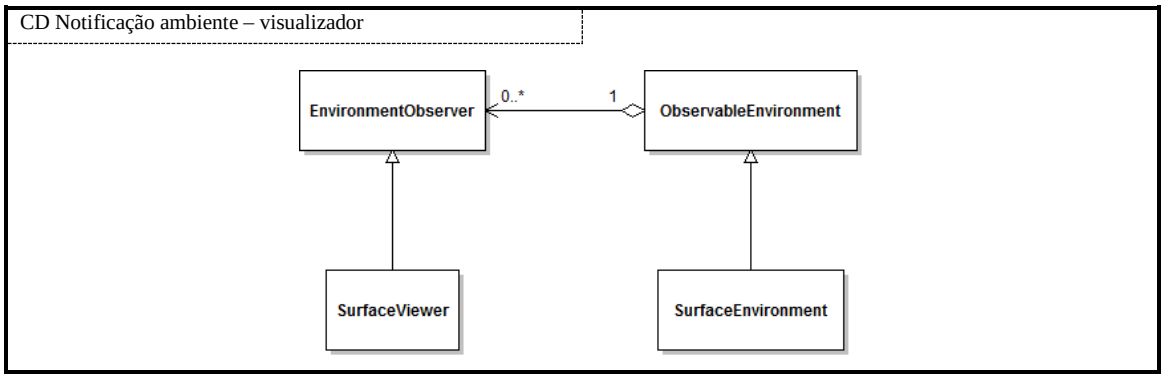


Diagrama 15 – Diagrama de classes de integração entre o ambiente e o visualizador

5.5. Resumo

A realização desta plataforma tem como principal objetivo fornecer uma forma de testar e validar o comportamento dos métodos de navegação baseados em campos de potencial selecionados anteriormente.

Apesar da incidência deste trabalho recair na biblioteca, é também importante que a plataforma de testes cumpra os requisitos que lhe são propostos duma forma forma modelar para que os desenvolvimentos futuros desenvolvimentos causem menor impacto.

É graças a esta plataforma que é possível a obtenção dos resultados de teste que serão analisados no capítulo seguinte.

6. Resultados experimentais

Tendo em conta que no âmbito do presente trabalho se pretende o desenvolvimento de uma biblioteca composta por um conjunto de métodos de navegação autónoma baseados em campos de potencial e de uma plataforma de teste de forma a validar a execução desses mesmos algoritmos, os resultados que devem ser demonstrados deverão permitir a comparação entre os diversos métodos e a confirmação das expetativas até este ponto mencionadas.

6.1. Ambientes considerados

Numa primeira instância para a validação e comparação dos resultados é necessário criar um conjunto de ambientes que possam evidenciar os comportamentos dos vários métodos. Assim são propostos os seguintes ambientes base de seguida apresentados Figura 35, Figura 36, Figura 37, Figura 38 e Figura 39.

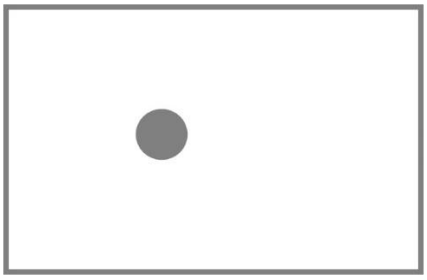


Figura 35 – Ambiente 1

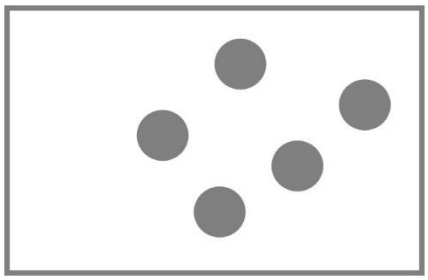


Figura 36 – Ambiente 2

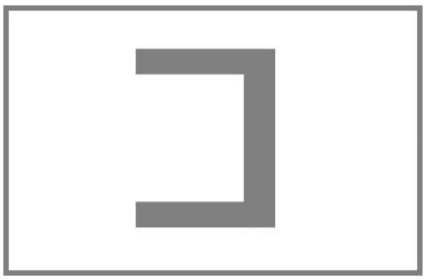


Figura 37 – Ambiente 3

Figura 38 – Ambiente 4

Figura 39 – Ambiente 5

A proposta destes ambientes surge da sua identificação como configurações em que a navegação falha ou como configurações normalmente consideradas para fins de teste deste tipo de métodos.

6.2. Parâmetros globais

Para que a análise dos resultados seja correta, além da definição dos ambientes de teste é necessário definir um conjunto de premissas passível de utilização para todos os agentes. Posto isto, são definidos os seguintes valores para os parâmetros de teste:

- Pontos iniciais – estes pontos são configuráveis e representam os pontos onde o agente inicia a sua navegação. Nos testes realizados encontram-se configurados cinco pontos de partida para cada ambiente;
- Pontos objetivo – estes pontos são configuráveis e representam os pontos nos quais o agente deverá terminar a sua navegação. Nos testes realizados encontram-se configurados cinco pontos objetivo para cada ambiente;
- Tempo máximo de resolução – este tempo corresponde ao tempo máximo que o agente tem para navegar desde um ponto inicial até um ponto objetivo. Nos testes realizados encontram-se configurados 30 segundos;
- Execução – o agente inicia a sua execução num ponto inicial e tenta navegar até cada ponto objetivo no tempo máximo de resolução. A execução termina após serem considerados todos os pontos de partida.

Com a definição dos parâmetros de execução do agente falta definir quais serão os parâmetros de comparação que serão utilizados. Tendo em conta o que se pretende para este trabalho, são utilizados os seguintes parâmetros:

- Percentagem de sucesso – este valor é obtido tendo em conta o número total de testes e o número de vezes que o agente conseguir navegar desde o ponto de partida até ao ponto objetivo dentro do tempo máximo de resolução;
- Numero médio de passos – este valor considera o número de iterações que o agente realiza até atingir o objetivo. Doutra forma pode ser visto como o número de ações que o agente tem de realizar para cumprir o objetivo;
- Distância média percorrida – esta distância corresponde ao somatório da distância percorrida entre cada iteração sendo expressa em unidades de movimentação do agente;
- Tempo médio de convergência – este valor corresponde ao tempo utilizado pelo algoritmo desde que inicia a sua execução até atingir a posição objetivo.

6.3. Resultados obtidos

6.3.1. Método do gradiente

	Ambiente 1	Ambiente 2	Ambiente 3	Ambiente 4	Ambiente 5
Percentagem de sucesso (%)	88%	64%	36%	36%	48%
Número médio de passos	115	115	38	97	79
Distância média percorrida (pixel)	251	223	81	182	166
Tempo médio de convergência (s)	4.6	5.9	1.16	2.0	4.0

6.3.2. Método de deteção e fuga baseada em *wall following*

	Ambiente 1	Ambiente 2	Ambiente 3	Ambiente 4	Ambiente 5
Percentagem de sucesso (%)	92%	88%	52%	44%	52%
Número médio de passos	116	149	121	69	93
Distância média percorrida (pixel)	245	316	175	129	166
Tempo médio de convergência (s)	4	6.1	3.8	2.9	4.0

6.3.3. Algoritmo de *avoiding-past*

	Ambiente 1	Ambiente 2	Ambiente 3	Ambiente 4	Ambiente 5
Percentagem de sucesso (%)	100%	100%	80%	84%	76%
Número médio de passos	65	90	120	199	277
Distância média percorrida (pixel)	235	316	354	543	189
Tempo médio de convergência (s)	4.3	8.9	5.0	5.1	4.9

6.3.4. Método de *minimum filling*

	Ambiente 1	Ambiente 2	Ambiente 3	Ambiente 4	Ambiente 5
Percentagem de sucesso (%)	100%	96%	92%	68%	76%
Número médio de passos	201	306	429	235	276
Distância média percorrida (pixel)	235	323	360	207	265
Tempo médio de convergência (s)	4.8	8.9	3.9	4.1	7.9

6.3.5. Algoritmo *wavefront*

	Ambiente 1	Ambiente 2	Ambiente 3	Ambiente 4	Ambiente 5
Percentagem de sucesso (%)	100%	100%	100%	100%	100%
Número médio de passos	13	15	15	15	17
Distância média percorrida (pixel)	280	337	315	321	371
Tempo médio de convergência (s)	5.0	5.8	5.9	5.6	5.2

6.4. Análise dos resultados e conclusões

6.4.1. Método do gradiente

Os resultados apresentados por este método correspondem ao que seria expectável. O método demonstrou que quando não existem ótimos locais oferece um método de fácil implementação para a resolução da problemática da navegação autónoma.

Quando confrontado com a presença de ótimos locais o agente demonstrou que não consegue progredir, o que permite afirmar que a utilização deste método é aceitável caso o ambiente seja composto por poucos obstáculos e a probabilidade da existência de ótimos locais seja quase nula.

Apesar disto, este algoritmo pode ser interessante na medida em que pode ser integrado em abordagens híbridas em que existe um planeamento prévio do percurso o qual é dividido em etapas sendo que o agente utiliza este método para ligar cada uma das etapas.

6.4.2. Deteção e fuga com utilização do algoritmo *wall following*

Durante a apresentação deste algoritmo foi indicado que a implementação do algoritmo *wall following* é baseado na aplicação dos conceitos associados a outros métodos baseados em campos de potencial. A ideia passava pelo aproveitamento desses conceitos e tornar o *wall following* mais dinâmico.

Após a análise dos resultados obtidos verificou-se que em comparação com o método do gradiente a percentagem de sucesso é superior. Apesar disso, não consegue demonstrar resultados equiparáveis aos demonstrados pelos restantes métodos, sendo que pioram à mesma taxa que os resultados demonstrados pelo método do gradiente.

Apesar de na sua essência o desenvolvimento de um método baseado em campos de potencial poder ser interessante, não demonstrou uma melhoria significativa nos resultados mas levanta a questão de qual será o comportamento apresentado pelo agente se ao invés deste método um outro seja selecionado.

6.4.3. Algoritmo *avoiding-past*

Os resultados da execução deste método demonstram que com a adição das duas componentes, memória e aleatoriedade, é possível obter um bom desempenho por parte do algoritmo.

Um ponto relevante passa pelo caráter arbitrário da componente aleatória. Mesmo em situações em que existe a criação de ótimos locais por parte das restantes componentes da força resultante, a adição da aleatoriedade permite desbloquear o agente.

Ainda dentro da execução há que evidenciar a relevância da memória que "empurra" o agente evitando que este percorra zonas previamente visitadas permitindo ao agente escapar de determinadas zonas com mais eficácia.

6.4.4. Método de *minimum filling*

O ponto mais importante evidenciado por este algoritmo surge da sua execução no ambiente 4. Este ambiente é caraterizado por uma zona central semelhante a um beco mas em que existe uma saída/entrada delimitada.

Apesar de este ambiente ser semelhante ao ambiente 3 notou-se uma redução considerável do desempenho deste algoritmo, situação essa também evidenciada se comparada a percentagem de sucesso com os restantes ambientes.

Esta situação decorre da execução deste método que percorre todas as posições até as excluir. Tendo em conta a zona central do ambiente sempre que o agente se encontre dentro nessa zona antes de sair existe um consumo elevado de tempo o que não permite ao agente terminar a navegação dentro do tempo considerado.

6.4.5. Algoritmo *wavefront*

Além dos aspetos evidenciados pela análise dos resultados apresentados por este método existem outros pontos que devem ser analisados, nomeadamente características associadas ao algoritmo.

Um dos pontos previamente mencionados acerca deste método refere que a principal desvantagem incidia no tempo e recursos consumidos para a geração de um percurso. Como foi indicado na especificação do algoritmo a geração de posições vizinhas é realizada tendo em conta uma constante *c*. Sendo assim é possível representar o ambiente em zonas de maior ou menor dimensão tendo em conta o valor desta constante.

Isto implica também o tempo de execução do algoritmo, na medida em que quanto maior for a zona menor será o número de posições a considerar e menor será o tempo consumido pelo algoritmo na geração do percurso.

Posto isto, é necessário compreender as implicações que associadas a este ponto. Caso o valor da constante *c* seja elevado pode contribuir para que determinadas zonas não sejam consideradas, o que pode dificultar ou invalidar a navegação em ambientes com demasiados obstáculos. Caso o valor da constante *c* seja baixo aumenta o tempo de execução do algoritmo.

Este ponto demonstra que o conhecimento acerca de determinadas caraterísticas do ambiente é importante para a configuração dos métodos utilizados pelo agente para atingir os objetivos.

6.4.6. Análise global

Partindo dos resultados apresentados anteriormente, é possível compreender que existe uma divisão em dois grupos. Um primeiro grupo no qual se insere o método do gradiente e o método de deteção e fuga e um segundo grupo composto pelos restantes três métodos.

O primeiro grupo é caraterizado por algoritmos que demonstram um desempenho pior já que nos mapas utilizados sempre que se deparam com ótimos locais não conseguem terminar a navegação com sucesso. Se a nível do método do gradiente esta situação já seria espectável o mesmo não acontece com o método de deteção e fuga, mas derivado da sua implementação este algoritmo não demonstrou um melhor desempenho ao longo dos ambientes.

Relativamente ao segundo grupo, os métodos nele integrados demonstram que conseguem com bastante eficácia resolver o problema dos ótimos locais. Apesar do método *minium filling* apresentar um desempenho mais baixo, tal pode dever-se ao tempo que o algoritmo demora quando tem de sair de zonas como as existentes no ambiente 4, mas ao considerarmos um tempo máximo mais alto de execução o agente apresentará melhores resultados a nível de concretização de navegações com sucesso.

O mesmo tipo de problema afeta o método baseado no algoritmo *avoiding-past*, que no ambiente 5 demonstrou que em determinadas situações não consegue sair da zona de bloqueio. Para o agente conseguir sair desta zona necessita de guardar um maior número de posições em memória de forma a evitar as posições anteriormente percorridas, e aumentar o tempo máximo de resolução do problema.

Por fim conclui-se que pela análise dos resultados apresentados o método baseado em campos de potencial que apresentou os melhores resultados foi o método baseado no algoritmo *wavefront*.

7. Conclusão

7.1. Trabalho realizado

7.1.1. Métodos de navegação autónoma baseados em campos de potencial

A utilização de métodos de navegação autónoma baseados em campos de potencial pode ser vista como uma opção válida e eficaz de resolver a problemática da navegação autónoma para agentes autónomos.

As principais razões que podem levar à utilização dos métodos desenvolvidos prendem-se sobretudo com a sua simplicidade e consumo de recursos. Isto é particularmente interessante em ambientes em que não existam ótimos locais e o agente tenha de reagir rapidamente caso se depare com um obstáculo imprevisto.

Apesar deste facto, foi possível verificar que dos métodos desenvolvidos existem soluções capazes de superar ambientes onde existam ótimos locais. Isto sendo conseguido através da adição de memória, representação do ambiente de uma forma discreta, ou através de pré-processamento do ambiente.

Este ponto só demonstra o carácter evolutivo que pode ser adicionado a estes métodos, sendo dessa forma possível a sua utilização através da adaptação do método ao ambiente no qual o agente será integrado.

Relativamente a este último aspeto, é de notar que é necessário ter cuidado com as adaptações ou configurações já que estas podem degenerar em soluções demasiado complexas do ponto de vista da sua compreensão e implementação, ou soluções computacionalmente dispendiosas e que não podem ser utilizadas em ambientes reais.

7.1.2. Linguagem *python*

Durante a implementação da biblioteca e da plataforma de teste, a utilização da linguagem *python* tornou possível verificar algumas das vantagens que existem na modelação deste tipo de problemas tendo em conta linguagens mais dinâmicas.

Este facto é particularmente interessante na forma como a informação é trocada entre os vários componentes. Tal como em outras linguagens interpretadas não existe uma tipificação rígida dos objetos, apesar de ser possível desenvolver classes e instanciar objetos, logo é possível alterar as mensagens com um menor impacto.

Outra vantagem passa pelas estruturas nativas. O suporte nativo de dicionários e a integração do conceito de tuplo, por exemplo, são dois tipos de estrutura nativos que simplificam toda a aplicação e que evitam a definição de tipos desnecessariamente.

Apesar disto, foi notório que a modelação das arquiteturas e a transposição destas para implementação não é direta. Um destes pontos recai na definição de contratos através de interfaces, já que este conceito não se encontra totalmente mapeado na linguagem *python*.

Tendo em conta as vantagens e desvantagens apresentadas, o balanço é positivo e demonstra que este tipo de linguagens podem ser utilizadas neste contexto de desenvolvimento.

7.2. Trabalho futuro

7.2.1. Caraterísticas físicas dos agentes

No trabalho aqui apresentado considerou-se o agente como um objeto uniforme. Este ponto é correto na medida em que ao ser aplicada uma força no agente que o afaste de colisões todo o agente evita colisões.

Caso seja considerado um agente de uma dimensão superior ou um agente composto por vários elementos é necessário a introdução de conceitos como *point subjected to the potential* (PSP), referidos, por exemplo, no trabalho apresentado por *Khatib* (30).

Nestes casos, o agente deve ser dividido, por exemplo através de secções, sendo a força resultante calculada para cada uma dessas secções. A força resultante total será resultado do somatório destas.

A adição desta caraterística implicaria a adição de outros mecanismos mas contribuiria para uma melhor representação de situações reais.

7.2.2. Múltiplos agentes

Durante a análise das soluções baseadas em campos de potencial verificou-se a existência de diversos trabalhos nos quais eram considerados múltiplos agentes. De entre estes trabalhos incluíam-se soluções nas quais os ótimos locais eram solucionados através da própria repulsão e atração entre os agentes, os quais eram classificados como enxames.

Dentro do enquadramento atual não seria um requisito a implementação de uma forma de suporte para este tipo de característica, mas seria aceitável o desenvolvimento de suporte para múltiplos agentes individuais.

Um dos pontos que poderia merecer a atenção é a possibilidade de criar ótimos locais dinâmicos. Isto pode suceder da execução de cada um dos agentes ao deslocar-se para a zona onde se encontra a posição considerada como alvo. Ao longo do percurso caso, o agente se aproxime de outros, a força repulsiva entre eles teria de ser somada com as restantes forças que atuam sobre o agente. Isto pode provocar a criação de um ótimo local que numa primeira instância não existia.

Um exemplo interessante seria a resolução de uma situação em que dois agentes tenham de passar por um corredor estreito. Neste caso pode existir a possibilidade de ambos se encontrarem no início desse corredor e nenhum deles prosseguir a navegação.

7.2.3. Dinamismo

A adição de dinamismo e a contemplação de múltiplos agentes são dois pontos aos quais deve ser dada a sua devida importância. Isto sucede visto que cada agente além de evitar obstáculos estáticos pode ter de evitar obstáculos móveis, outros agentes, ou até mesmo atingir objetivos móveis.

Compreender a forma como estes métodos se comportam tendo em conta estas caraterísticas pode contribuir para a sua maior utilização.

7.2.4. Interface gráfica

A importância de uma boa interface gráfica é muitas vezes deixada de parte sendo que existe uma maior preocupação em relação ao núcleo da execução e todos os mecanismos que se lhe encontram associados.

Também neste trabalho esta situação refletiu-se numa interface gráfica que não permite ao utilizador facilmente configurar a execução e o ambiente associados a cada agente.

Numa próxima versão este deve ser um dos pontos a melhorar. Primeiro pela mais-valia a nível da apresentação da própria biblioteca e segundo pelo tempo despendido na realização de testes e validação de execução dos métodos de navegação autónoma.

7.3. Considerações finais

O desenvolvimento de agentes autónomos e inteligentes tenta, partindo da análise de caraterísticas humanas ou demonstradas por outros seres vivos, a criação de soluções capazes de realizar determinadas tarefas.

Se considerarmos a forma como a navegação é realizada pelos seres humanos, existem casos em que esta é numa primeira fase planeada selecionando o melhor percurso. Se durante este percurso surgirem imprevistos normalmente existe a realização de uma alternativa.

Ao transpor esta situação para a navegação autónoma em agentes, considero que um bom caso de estudo seria a realização de uma análise em relação a métodos híbridos. Estes são caraterizados por uma primeira fase em que existe o cálculo de um percurso ótimo. Caso surja um imprevisto deverão agir em conformidade sendo que essa ação pode ser realizada através de métodos como os apresentados neste trabalho.

Bibliografia

(1) **Google Driveless Car**. *Wikipedia*. http://en.wikipedia.org/wiki/Google_driverless_car. 2013

(2) **iRobot**. *iRobot*.http://store.irobot.com/shop/index.jsp?categoryId=2804605. 2013

(3) **Automated Guided Vehicle**.*Wikipedia*. http://en.wikipedia.org/wiki/Automated_guided_vehicle. 2013

(4) **Hee, Lee Gim and H. Ang Jr., Marcelo**. *Mobile robots navigation, mapping and localization : Part I.* Singapore: National University of Singapore, 2009. 2013

(5) **Hagelbäck, Johan**. *Using Potential Fields In Real-Time Strategy Game Scenario*. AIGameDev. 31 Janeiro 2009. http://aigamedev.com/open/tutorials/potential-fields/. 2012

(6) **Hagelbäck, Johan and Johansson, Stefan J**. *Using Multi-agent Potential Fields in Real-time Strategy Games.* International Foundation for Autonomous Agents and Multiagent Systems, 2008. 2012

(7) **Johansson , Stefan J. and Hagelbäck, Johan**. *The Rise of Potential Fields in Real Time Strategy Bots.* Ronneby, Sweden: Department of Software and Systems Engineering Blekinge Institute of Technology. 2012

(8) **Maes, Pattie**. *Modeling Adaptive Autonomous Agents.* Cambridge: MIT Media Laboratory. 2013

(9) **Wikipedia**. *Wikipedia*. http://en.wikipedia.org/wiki/Gradient. 2012

(10) **Scalar Field**. *Wikipedia*. http://en.wikipedia.org/wiki/Scalar_field. 2012

(11) **Vector field**. *Wikipedia*. http://en.wikipedia.org/wiki/Vector_field. 2012

(12) **Cyberbotics' Robot Curriculum/Advanced Programming Exercises.** *Wikibooks*. http://en.wikibooks.org/wiki/Cyberbotics'_Robot_Curriculum/Advanced_Programming_Exercises. 2013

(13) **Goodrich, Michael A**. *Potential Fields Tutorial.* 2013

(14) **Bell, Graeme**. *Forward Chaining for Potential Field Based Navigation.* Tese de Doutoramento da University of St Andrews, 2005. 2012

(15) **Murphy, Robin R**. *Introduction to AI Robotics.* Cambridge, Massachusetts, 2000. 2013

(16) **Hellström, Thomas**. *umu.se*. 19 Dezembro 2011. 2013

(17) **Livesey, Mike and Bell, Graeme**. *The Existence of Local Minima in Local-Minimum-Free Potential Surfaces.* North Haugh, St Andrews: University of St Andrews. 2013

(18) **Connolly, Christopher I**. *Harmonic Functions and Collision Probabilities.* Ravenswood Ave., Menlo Park: SRI International, Inc. EK 266. 2013

(19) **Connolly, C. I., Burns, J. B. and Weiss, R**. *Path planning using Laplace's Equation.* Amherst: University of Massachussets, 1994. 2013

(20) **Connolly, Christopher I. and Grupen, Roderic A**. *Applications of Harmonic Functions to Robotics.* Amherst: University of Massachussets, 1992. 2013

(21) **LaValle, Steven M**. *Planning Algorithms.* Cambridge. 2013

(22) **Lewis, P. Jonathan and Wier, K. Michael**. *Subgoal chaining and the local minimum problem.* St Andrews. 2013

(23) **Juliá, Miguel, et al**. Local minima detection in potential field based cooperative multi-robot exploration. Elche: Miguel Hernández University. 2013

(24) **Park, Min Gyu and Lee, Min Cheol**. A New Technique to Escape Local Minimum in Artificial Potential Field Based Path Planning. Pusan: Pusan National University, 2003. 2013

(25) **Caselli, Stefano, Reggiani, Monica and Rocchi, Roberto**. *Heuristic Methods for Randomized Path Planning in Potential Fields.* Parma: Universita di Parm, 2001. 2013

(26) **Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P**. *Optimization by Simulated Annealing.* JSTOR.ORG, 1983. 2012

(27) **Safadi, Hani**. *Local Path Planning Using Virtual Potential Field.*McGill University, 18 April 2007. http://www.cs.mcgill.ca/~hsafad/robotics/. 2012

(28) **Baert, Stefan**. *Motion Planning Using Potential Fields* .gamedev.net, 15 July 2000. http://www.gamedev.net/page/resources/_/technical/artificial-intelligence/motion-planning-using-potential-fields-r1125. 2012

(29) **Gambardella, Luca Maria and Versino, Cristina**. *Robot Motion Planning Integrating Planning Strategies and Learning Methods.* Lugano: IDSIA - Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, 1994. 2013

(30) **Khatib, Oussama**.*Real-time obstacle avoidance for manipulators and mobile robots.*. 1986, The International Journal of Robotics Research. 2012

(31) **Borenstein, Johann and Koren, Yorem**. *Real-Time Obstacle Avoidance for Fast Mobile Robots*. IEEE Transaciton on Systems, Man and Cybernitcs. 1989, Vol. 19.

(32) **Yun, Xiaoping and Tan, Ko-Cheng**. A wall-following method for escaping local minima in potential field based motion planning. Monterey: ICAR97, 1997. 2012

(33) **Zhu, Yi, Zhang, Tao and Song, Jingyan**. *An improved wall following method for escaping from local minimum in artificial potential field based path planning.* Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference, Shanghai, 2009. 2012

(34) **Balch, Tucker and Arkin, Ronal**. *Avoiding the Past: A simple but effective strategy for reactive navigation.* Atlanta, Georgia: Georgia Institute of Technology, 1993. 1050-4729/93. 2013

(35) **Masehian, Ellips and Amin-Naseri, M. R**. *Composite Models for Mobile Robot Offline Path Planning.* Iran: Tarbiat Modares University, 2007. 2013

(36) **Nattharith, P**. *Mobile Robot Navigation using a Behavioural Strategy.* Newcastle: School of Mechanical and Systems Engineering Newcastle University. 2013

(37) **Technologies, Network of Excellence Information Society.** *Euron.*July 2005. http://www.euron.org/miscdocs/docs/euron2/year2/dr-14-1-industry.pdf. 2013

(38) **Mabrouk, Mohamed M. and McInnes, C.R**. *Solving the potential field local minimum problem using internal agent states*. Strathprints: University of Strathclyde Institutional Repository.2008. http://strathprints.strath.ac.uk/8107/1/strathprints008107.pdf. 2013

(39) **Thurau, Christian and Bauckhage, Christian.** *Learning human-like Movement Behavior for Computer Games.* Bielefeld, Germany: Bielefeld University. 2012

(40) **Kuipers, Benjamin**. *Lecture 7: Potential Fields and Model Predictive Control*. http://www.google.com.br/url?sa=t&rct=j&q=potential%20field%20robotics&source=web&cd=6&cad=rja&ved=0CGwQFjAF&url=http%3A%2F%2Fwww.cs.utexas.edu%2F~pstone%2FCourses%2F395Tfall05%2Fresources%2Fweek9b-ben-potential-fields.ppt&ei=CawVUerFN4HQhAeZhoDoAw&usg=AF. 2013

(41) **McInnes, Colin R. and Mabrouk, Mohamed H**. *An Emergent Wall Following Behaviour to Escape Local Minima for Swarms of Agents.* Glasgow: Department of Mechanical Engineering, University of Strathclyde, 2008. 2013

(42) **Rocchi, Roberto, Reggiani, Monica and Caselli, Stefano**. *Heuristic Methods for Randomized Path Planning in Potential Fields.* Parma: Dipartimento di Ingegneria dell'Informazione, Universita di Parma, 2001. 2013

(43) **Carpin, Stefano and Pilloneto, Gianluigi**. *Merging the adaptive random walks planner with the randomized potential fields planner.* Bremen and Padova: International University Bremen and University of Padova, 2005. 2013

(44) **Lamiraux, F. and Laumond, J. P**. *On the Expected Complexity of Random Path Planning.* Toulousse, 1996. 2013

(45) **Scheutz, Matthias and Logan, Brian**. *Affective vs Deliberative Agent Control.* Nottingham: University of Nottingham, 2001. 2013

(46) **Vascák, Ján**. *Navigation of mobile robots using potential fields and computational intelligene means.* Kosice, Slovakia: Faculty of Electrical Engineering and Informatics, Technical University in Kosice, 2007. 2013

Apêndices

Apêndice 1 – Instruções de instalação dos componentes

De forma a fornecer o ambiente de desenvolvimento e testes, encontra-se no suporte informático fornecido à parte um conjunto de elementos que devem ser referenciados. A sua instalação, deve ser também ela realizada pela ordem de apresentação.

- Instalação para suporte a *python* – na pasta "/Python(x,y)" reside o executável necessário para a instalação de suporte a *python*. Além disto inclui um conjunto que de outras funcionalidades que podem ser instaladas nomeadamente o ambiente desenvolvimento *sypder*.
- Instalação da biblioteca *pygame* – na pasta "/Pygame" reside o executável necessário para instalar esta biblioteca
- Instalação da biblioteca desenvolvida – na pasta "/Biblioteca" reside o executável responsável pela instalação da biblioteca desenvolvida no âmbito deste projeto
- Execução – deve ser executado o ficheiro main.exe que se encontra na pasta "/Plataforma de testes/dist"

Os pontos configuráveis em relação à execução da plataforma são abordados no apêndice seguinte.

Apêndice 2 – Configuração de parâmetros da plataforma de teste – Simulador

Para testar os algoritmos desenvolvidos além de uma demonstração da execução de um algoritmo é possível parametrizar um conjunto de parâmetros. Na pasta "/Plataforma de testes/dist/configuration_folder" existe uma estrutura de forma configurar os seguintes parâmetros.

Simulador	Localização	/simulador_configuration/exe_conf/execution_config.xml		
	Nome	Tag	Valor	Descrição
	Ambiente Posições agente	environment/path starts/position	path	Caminho absoluto para o ficheiro de teste. Este deve ser baseado nos ficheiros de teste indicados no suporte informático
				Coordenadas x, y para o agente
	Posições objetivo	objectives/position		Coordenadas x, y para os objetivos
	Tempo esgotado	timeout		Intervalo de tempo máximo em segundos para o agente terminar a navegação
	Tempo esgotado	timeout		Intervalo de tempo dado ao agente para finalizar a navegação
	Agente selecionado	agent/type	iga	Agente implementado através do método do gradiente
			wfa	Agente implemento através do método de deteção e fuga
			apa	Agente implementado através do método baseado no algoritmo *avoiding-past*
			mfa	Agente implementado através do método *minimum filling*
			wa	Agente implementado através do algoritmo *wavefront*

Apêndice 3 – Configuração de parâmetros da plataforma de teste – Agentes

Existe associado a cada algoritmo um conjunto de parâmetros configurável. Na pasta "/Plataforma de testes/dist/configuration_folder/agente_configuration" encontram-se os ficheiros onde se encontram as variáveis de configuração dos vários agentes. Segue-se a identificação da localização e parâmetros de configuração que se encontram definidos.

- Método do gradiente
    - Localização: \cfg_iga\exe_conf\execution_config.xml
    - Parâmetros de configuração:
        - obstacleMaxIntensity – intensidade de repulsão exercida por um campo de potencial repulsivo
        - obstacleMaxField – alcance máximo do alcance associado a um campo de potencial repulsivo
        - goalMaxIntensity – intensidade de atração exercida por um campo de potencial atrativo
        - goalMaxField – alcance máximo associado a um campo de potencial atrativo
- Método de deteção e fuga baseado em *wall following*
    - Localização: \cfg_wfa\exe_conf\execution_config.xml
    - Parâmetros de configuração:
        - obstacleMaxIntensity – intensidade de repulsão exercida por um campo de potencial repulsivo
        - obstacleMaxField – alcance máximo do alcance associado a um campo de potencial repulsivo
        - goalMaxIntensity – intensidade de atração exercida por um campo de potencial atrativo
        - goalMaxField – alcance máximo associado a um campo de potencial atrativo
        - wallFollowingMaxNrOfBetterPotencials – número de potenciais favoráveis para terminar a fuga baseada no método *wall following*
        - wallFollowingMaxMemory – dimensão da memória para verificação de potenciais favoráveis
        - wallFollowingMinimunDetectionPotencial – valor de magnitude que o agente utiliza para considerar que se encontra num ótimo local

- Método baseado no algoritmo *avoiding-pat*
    - Localização: \cfg_apa\exe_conf\execution_config.xml
    - Parâmetros de configuração:
        - obstacleMaxIntensity – intensidade de repulsão exercida por um campo de potencial repulsivo
        - obstacleMaxField – alcance máximo do alcance associado a um campo de potencial repulsivo
        - goalMaxIntensity – intensidade de atração exercida por um campo de potencial atrativo
        - goalMaxField – alcance máximo associado a um campo de potencial atrativo
        - avoidPastPastMaxField – intensidade de repulsão exercida pelo campo de potencial repulsivo associado a uma posição anterior
        - avoidPastPastMaxIntensity – alcance máximo do campo de potencial associado a uma posição anterior
        - avoidPastRandomMagnitudeMultiplier – fator multiplicativo da magnitude da força aleatória caso o agente se encontre a colidir com um elemento presente no ambiente
        - avoidPastMaxMemory – dimensão da memória utilizada
        - avoidPastMaxVisitsPerPosition – número máximo vezes que o agente pode encontrar-se numa determinada posição/área
        - avoidPastPastMark – dimensão da área considerada visitada pelo agente sempre que se encontre numa posição
        - avoidPastPastHorizon – não se encontra a ser utilizado
        - avoidPastPastGain – não se encontra a ser utilizado
- Método minium filling
    - Localização: \cfg_mfa\exe_conf\execution_config.xml
    - Parâmetros de configuração:
        - obstacleMaxIntensity – intensidade de repulsão exercida por um campo de potencial repulsivo
        - obstacleMaxField – alcance máximo do alcance associado a um campo de potencial repulsivo
        - goalMaxIntensity – intensidade de atração exercida por um campo de potencial atrativo
        - goalMaxField – alcance máximo associado a um campo de potencial atrativo

- Método wavefront
    - Localização: \cfg_wa\exe_conf\execution_config.xml
    - Parâmetros de configuração:
        - obstacleMaxIntensity – intensidade de repulsão exercida por um campo de potencial repulsivo
        - obstacleMaxField – alcance máximo do alcance associado a um campo de potencial repulsivo
        - goalMaxIntensity – intensidade de atração exercida por um campo de potencial atrativo
        - goalMaxField – alcance máximo associado a um campo de potencial atrativo
        - wavefrontAreaAnalysis – área ocupada pelo agente
        - wavefrontAreaGeneration – área coberta por cada posição gerada