FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Sistemas de Cooperação Multi-robô para assemblagem em indústria automóvel

Joana Raquel Ramiro Santos

PREPARAÇÃO DA DISSERTAÇÃO

PREPARAÇÃO DA DISSERTAÇÃO

Orientador: António Paulo Moreira

Co-orientador: Germano Veiga

16 de Fevereiro de 2014

Sistemas de Cooperação Multi-robô para assemblagem em indústria automóvel

Joana Raquel Ramiro Santos

PREPARAÇÃO DA DISSERTAÇÃO

16 de Fevereiro de 2014

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

ADT	Abstract Data Type
ANDF	Architecture-Neutral Distribution Format
API	Application Programming Interface
CAD	Computer-Aided Design
CASE	Computer-Aided Software Engineering
CORBA	Common Object Request Broker Architecture
UNCOL	UNiversal COmpiler-oriented Language
Loren	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed vehicula lorem commodo dui
WWW	*World Wide Web*

Capítulo 1

Introdução

Este documento foi desenvolvido no âmbito da unidade curricular Preparação da Dissertação, e visa apresentar o trabalho realizado ao longo do primeiro semestre, nomeadamente Revisão de Literatura, e planeamento de tarefas futuras. Assim, no capitulo 2 será apresentado o Estado de Arte, focado em dois tópicos essenciais no desenvolvimento do projeto. São eles:

- **Simuladores:** levantamento de alguns simuladores usados em Robótica. Comparação das caracteristicas de cada um deles, e correspondente análise para o caso do projeto STAMINA;
- **Algoritmos Coordenação Multi-robô:** estudo de algoritmos de planeamento de trajetórias e coordenação multi-robô;

1.1 Objetivos

Os objetivos desta tese são:

- Desenvolver um simulador realista do sistema, e que inclua a planta da fábrica, sensores, robôs, e outros elementos dinâmicos que possam influenciar o sistema;
- Estudo e implementação de um algoritmo de planeamento de trajetória e coordenação multi-robô;
- Teste no ambiente de simulação previamente desenvolvido;

1.2 Estrutura do Documento

No capítulo 2 é apresentado a Revisão de Literatura e Levantamento do estado de arte, segundo dois tópicos essenciais descritos acima. No capítulo 3 encontra-se a definição do problema que motivou esta dissertação, incluindo as ferramentas que irão ser usadas para a sua concretização. E finalmente, no capitulo 4 são apresentadas as ferramentas usadas para o planeamento do projeto, bem como as tarefas futuras já delineadas.

Capítulo 2

Estado de Arte

Neste capítulo será realizado um levantamento teórico sobre os principais tópicos que irão fundamentar todo o tema de tese. Inicialmente serão abordados alguns simuladores existentes no mercado, e a possibilidade de serem usados ou não, no desenvolvimento do trabalho. Na secção 2.2 será apresentado um estudo teórico de algoritmos quer de planeamento de trajetórias, quer de coordenação multi-robô, e a forma como estes dois tipos de implementação se relacionam.

2.1 Simuladores

2.1.1 Contexto

Em robótica a simulação assume particular importância em situações onde robôs e humanos partilham espaços, ou quando existe cooperação entre robôs. A fase de simulação desempenha um papel muito importante no projeto STAMINA, na medida em que nos permitirá testar algoritmos que serão usados em várias fases do projeto, incluindo os algoritmos de coordenação multi-robô. O principal objetivo da simulação é fornecer um ambiente simulado com as principais características da planta da fábrica da PSA, onde os AGV's irão operar. Esse ambiente de simulação será usado para comparar novos algoritmos, considerando a sua robustez, eficiência e segurança. Nas secções seguintes será realizada a escolha do simulador. Serão apresentados 4 softwares de simulação open-source, existentes no mercado. Considerando as características mais relevantes e que satisfaçam os requisitos do projeto, será realizada uma comparação entre os 4 softwares, e tomada uma decisão.

2.1.2 Comparação

Atualmente existem muitos simuladores disponíveis para auxiliar no estudo da Robótica. Contudo, considerando as características do projeto, foi possível selecionar 4 (Gazebo, V-Rep, SimTwo, USAR-Sim). Será realizada uma pequena descrição de cada simulador, e de seguida serão comparados entre si, considerando alguns requisitos essenciais, Staranowicz e Mariottini.

Gazebo é um simulador 3-D que permite a integração com o ROS. Este simulador usa duas bibliotecas: OGRE(object-oriented graphics rendering engine) and ODE. Estas permitem que o Gazebo apresente ambientes 3-D dinâmicos com precisão. Outra característica é a possibilidade de simular ambientes variados na medida em que este oferece diferentes combinações de robôs, a nível de articulações e formas. Tudo isto permite obter uma simulação realista, avaliando também todos os objetos considerando as suas características físicas (por exemplo, massa, fricção). Contudo, o Gazebo não permite alterar o ambiente de simulação durante o tempo de execução.

Outro simulador interessante é o **USAR-Sim**, Unified System for Automation and Robot Simulation. Tal como o Gazebo, este também oferece ambientes 3-D e que são suportados tanto em Windows como Linux. Este simulador é essencialmente usado em situações onde há interação entre humanos e robôs, ou multi-robô. USAR usa três protocolos principais para o código de controlo, sendo eles, Unreal Engine proprietary communication Protocol, MOAST (Mobility Open Architecture and tools engine) and USAR-Sim Matlab Toolbox, Petry e Mokaram et al.. Contudo, este simulador não fornece um tutorial claro, com exemplos e detalhes de implementação e funcionamento.

SimTwo é um software versátil que permite simular diferentes tipos de robôs e diferentes configurações. Este simulador é usado principalmente em investigação e educação. Neste contexto, fornece uma interface simples que permite efetuar, de forma rápida, a construção de um ambiente de simulação e testá-lo, Costa e Costa et al. (2011). Permite integração com o ROS, embora seja necessária uma conexão UDP, e a configuração não é ainda, simples. Outra desvantagem é a falta de documentação.

V-Rep é um software versátil, e com um conjunto de caracteristicas que o tornam adequado a aplicações multi-robô. Baseia-se numa arquitetura de controlo distribuída, o que permite subdividir um sistema complexo, em vários módulos mais simples, e programá-los de forma independente. Em termos de customização, V-Rep fornece a possibilidade de controlar os objetos individualmente. Cada objeto ou modelo pode ser controlado por 5 formas diferentes: um script, um plugin, um nó ROS, um cliente remoto ou uma solução customizada. A fim de proporcionar, uma simulação mais realista, este software admite vários níveis de precisão, rapidez, e outras características que podem ser ajustadas pelo utilizador. Uma propriedade interessante é a existência de partículas customizáveis, que poderão ser usadas para simular água ou ar, jatos, motores a jato... Finalmente, V-Rep oferece a possibilidade de aceder à simulação através de uma aplicação externa, ou um dispositivo hardware remoto. As suas propriedades, são disponibilizadas num manual de utilizador completo, e com tutoriais de fácil compreensão, coppeliarobotics.

Tal como já foi referido, um dos objetivos desta tese é testar algoritmos de coordenação multi-robô e desenvolver um simulador que permite a integração de várias partes do projeto STAMINA. V-Rep satisfaz os requisitos do projeto (ver tabela 2.1). Por isso, foi o simulador escolhido para implementar o ambiente de simulação.

Antes de apresentar a tabela comparativa, serão descritos os critérios que permitiram selecionar o V-Rep.

- Linguagem de Programação: Neste critério serão consideradas todas as linguagens de programação que são suportadas pelo simulador. Este deve admitir pelo menos, uma linguagem comum que possa ser integrada em várias plataformas.

- Suporte Multi-Thread: Neste ponto é considerada a capacidade de simular mais do que uma tarefa em simultâneo. Este requisito é muito importante para testar os algoritmos que serão implementados nesta tese, e outros. Esta característica torna o simulador mais eficiente e robusto.

- Sensores: Serão apresentados os principais sensores que o simulador inclui.

- Deteção de Colisões: Esta característica está presente na maioria dos simuladores. Estes devem detectar a possibilidade de colisão de robôs, paredes, e outros objetos.

- Integração com ROS: Tal como foi referido, o projeto STAMINA irá fazer a integração de vários nós, que serão agregados com ROS. Por isso, o software deve comtemplar essa possibilidade.

- Comunicação: Neste critério será considerado que protocolos serão usados na comunicação entre nós.

- Formato dos ficheiros de importação

Segundo Staranowicz e Mariottini, Petry, coppeliarobotics, Costa, Mokaram et al., Costa et al. (2011).

Linguagem programação	Suporte Multi-thread	Sensores	Deteção Colisões	Integração ROS	Comunicação	Ficheiros importação
			Gazebo			
C, C++, Java, Python	Sim	Odometria, Alcance, Visão	Sim	Sim	TCP/IP	?
			V-Rep			
C, C++, Python, Java, Lua, Matlab, Urbi	Sim	Proximidade, Visão, Odometria	Sim	Sim	TCP/UDP	OBJ, DXF, 3DS, STL, COLLADA, URDF
			USAR Sim			
C, C++, Java	Sim, mas usa Java Virtual machine(JVM)	Som, Visão, Odometria, Toque, Alcance	Sim	Sim	TCP/IP	?
			SimTwo			
Pascal	Não	Visão, Laser, Alcance, Luminosidade	Sim	Sim, mas não é direto	UDP ou Porta Série	3DS

Tabela 2.1: Tabela Comparativa simuladores

2.2 Algoritmos Coordenação Multi-robô

Em sistemas multi-robô, o planeamento do caminho que um robô deve seguir e a coordenação entre os diferentes robôs são dois problemas fundamentais. Ambos devem ser implementados e abordados simultaneamente, ao invés de serem tratados como problemas independentes. Nesta secção vamos estudar algoritmos existentes para planeamento de trajetórias e algumas metodologias usadas para coordenação multi-robô. Para a implementação de coordenação multi-robô é necessária uma etapa de planeamento do caminho, desde um ponto inicial ao destino, para cada robô (secção 2.2.1).

2.2.1 Path Planning

O planeamento do caminho consiste em descrever o caminho desde o ponto inicial até ao destino, evitando colisões com objetos ou outros robôs. Esse planeamento pode ser realizado pelos métodos clássicos (em termos matemáticos ou geométricos), ou métodos probabilísticos.

2.2.1.1 Métodos Probabilísticos

Em espaços dimensionais grandes, os métodos geométricos não são viáveis, enquanto que os métodos probabilisticos, neste caso, conseguem planear de forma eficiente o caminho. Serão apresentados o PRM (Probabilistic Roadmap) e RRT (Rapidly-Exploring Random Tree).

O método **PRM** consiste em tomar amostras aleatórias pertencentes ao espaço e verificar se elas pertencem ou não ao espaço livre. O método divide-se em duas fases, segundo Choset e Kavraki et al. (1996):

- Fase de aprendizagem: o roadmap é construído e armazenado num grafo, em que os nós correspondem a configurações livres de colisão, e as arestas a caminhos entre nós, sem colisões.
- Fase de consulta: todos os nós de início e de fim, são conectados a dois nós.

A Fase de aprendizagem/Construção do Roadmap, consiste nos seguintes passos:

1. Inicializar o grafo G(V, E). V é o conjunto dos nós, gerados aleatoriamente como configurações livres do robô; e E é o conjunto dos caminhos livres que conectam dois nós.
2. Uma configuração/nó q, é escolhida aleatoriamente.
3. Verificar se essa configuração/nó pertence ao espaço livre. Se sim, adicionar ao grafo G.
4. Repetir para N nós escolhidos.
5. Para cada nó q, seleccionar k nós vizinhos.
6. Usando um método local, unir os nós q e q', sendo q' um vizinho de q.
7. Se a conexão consistir num caminho pertencente ao espaço livre, então adicioná-lo ao grafo G.

A fase de consulta consiste em:

1. Definir os pontos de início e de fim no grafo.
2. Encontrar os k vizinhos do nó de partida e do nó fim, no grafo G.
3. Contruir o caminho usando um método de planeamento, por exemplo Dijkstra.

Outro método probabilístico é o **RRT**. O objetivo é, por amostragem de pontos pertencentes ao espaço livre, mapear um caminho, desde um ponto de início até um ponto de fim. Segundo LaValle (1998), o RRT possui a maioria das propriedades do 'Probabilistic Roadmap', sendo a única vantagem a possibilidade de poder ser aplicado ao planeamento de sistemas não-holonómicos, e que considerem a dinâmica e as restrições cinemáticas do robô. Um sistema não-holonómico, é um sistema cuja dimensão do espaço é superior aos graus de controlo possíveis ou desejáveis do

sistema. A árvore é contruída incrementalmente através de amostras aleatórias do espaço. São usadas duas árvores, uma com a 'raiz' no ponto inicial, e outra com 'raiz' no ponto final.

O método consiste em, segundo Choset:

1. Tendo as configurações q_{init} e q_{goal}, inicializar duas árvores com origem nesse ponto, respetivamente T_{init} e T_{goal}.
2. Cada árvore é expandida, fazendo:
   (a) q_{rand} é obtido por uma distribuição normal. q_{rand} é um ponto aleatório do espaço.
   (b) é determinado o q_{near}, que corresponde ao nó pertencente à árvore, mais próximo de q_{rand}.
   (c) acrescentar um ramo em q_{near}, na direção de q_{rand}, com o tamanho definido para cada ramificação. Este passo, é realizado se não houver colisão. Caso contrário, será escolhido outro nó.

2.2.1.2 Métodos Clássicos

Nos métodos clássicos, segundo Siegwart et al. (2011) é possivel distinguir duas estratégias:

- Pesquisa em grafo: é construído um mapa composto por vários nós e respetivas ligações. Neste caso, o planeamento do caminho resume-se a um problema de pesquisa em grafo.
- Campos de Potencial: é descrita uma função matemática associada ao espaço livre, e o gradiente dessa função pode ser seguido até ao ponto de destino.

Comecemos por analisar o método **Campos de Potencial**, segundo Costa (2011). Este consiste na criação de um gradiente ao longo do mapa do robô, que o vai conduzindo até ao destino. O ponto final é representado como um conjunto de forças atrativas e os obstáculos, forças repulsivas. Neste método, o robô é tratado como um ponto sujeito ao campo potencial artificial, campo este, criado pelo conjunto dessas forças atrativas e repulsivas. Não é necessária a construção de um mapa, que represente o ambiente onde o robô vai operar.

A **Pesquisa em Grafo**, contrariamente ao método anterior, necessita de um primeiro passo que consiste em transformar o modelo do ambiente num mapa discreto e que se adeque ao algoritmo que será implementado. Os algoritmos para o planeamento diferem na forma como usam esse mapa discretizado. Para a **construção do mapa**, e iniciar a representação do espaço livre e ocupado, existem várias abordagens, embora apenas sejam apresentadas aqui as principais.

- **Roadmap:** consiste em representar o espaço fisico como um conjunto de nós e ligações entre esses nós. Esses nós podem corresponder a localizações reais, e as ligações a caminhos entre esses nós. O roadmap pode ser construído de várias formas. No caso de **"visibility graph"** todos os pares de nós que são visiveis de um para o outro são conectados por um segmento. Os nós correspondem aos vértices dos obstáculos. Neste método, o número

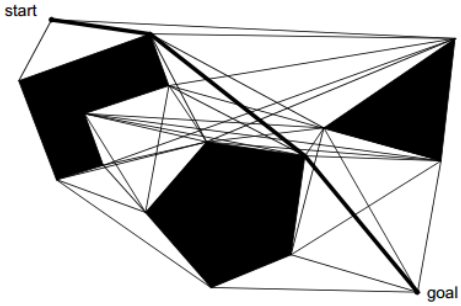


Figura 2.1: Visibility Graph (Siegwart et al. (2011))

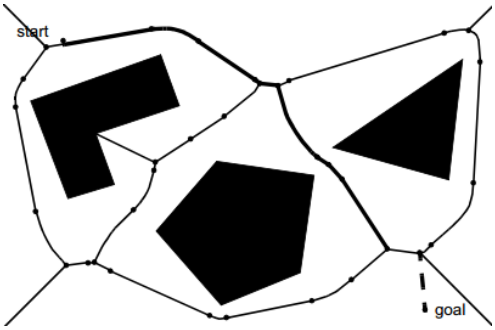


Figura 2.2: Voronoi Diagram (Siegwart et al. (2011))

de nós, e segmentos incrementa com o número de polígonos dos obstáculos. Por isso, pode-se tornar lento e ineficiente comparado com outras técnicas quando o ambiente tem muitos elementos. Outro método é pelo **diagrama Voronoi** (DV). Este tende a maximizar a distância entre o robô e os obstáculos do mapa. O espaço é dividido em regiões e em cada região só existe um obstáculo. O diagrama é o conjunto de pontos equidistantes a dois ou mais obstáculos. O algoritmo DV apresenta uma fraqueza, em relação ao alcance limitado dos sensores de localização. Sendo que o diagrama maximiza a distância aos obstáculos, qualquer sensor de curto alcance pode não conseguir identificar a sua localização.

- **Decomposição em Células:**Podemos distinguir decomposição em células exatas ou aproximadas. Na **decomposição em células exatas**, cada célula corresponde ao espaço livre ou ocupado do ambiente real. Não importa a posição do robô dentro da célula, apenas é relevante a capacidade do robô atravessar de uma célula livre para outra célula livre adjacente. Tal como na construção do roadmap, também neste caso, a complexidade do ambiente conduz a um rápido aumento do número de células, podendo tornar o algoritmo computacionalmente pesado. A **decomposição em células aproximadas**, não corresponde exatamente ao mapa do ambiente, pois existem células livres, ocupadas e parcialmente ocupadas. Esta técnica é das mais usadas em Robótica, essencialmente por conduzir a algoritmos de planeamento de trajetórias computacionalmente menos pesados. Dependendo do espaço, pode-se usar a decomposição em células de tamanho fixo, em que o tamanho da célula não depende dos ojetos pertencentes ao ambiente, ou "quadtree"em que, sempre que uma célula esteja parcialmente ocupada, é decomposta recursivamente em 4 retângulos idênticos, até um limite minimo de decomposição especificado (K* tamanho do robô).

Após a construção do mapa, e tendo a sua representação num grafo baseado em conexões entre nós, existem alguns algoritmos que permitem realizar a pesquisa do caminho entre um nó inicial e um nó final. Os mais relevantes são Pesquisa em Largura, Pesquisa em Profundidade, Dijkstra, A* e D*. Ambos, começam a sua pesquisa no nó inicio e exploram todos os seus nós vizinhos. Para cada um desses nós, são considerados os seus vizinhos não explorados, e assim sucessivamente. Iremos considerar para um dado nó, n, g(n) o custo acumulado desde o nó início até ao nó n, h(n)

o custo esperado desde o nó n ao nó fim (função heuristica). O custo total será representado por f(n)= g(n)+h(n).

Na **Pesquisa em Largura**, os nós são explorados considerando sempre a sua proximidade com o nó origem, isto é, explora todas as ligações de um nó, antes de passar para outro. A computação da solução é rápida, e se se considerar o custo individual de cada transição constante em todo o grafo, então esta pesquisa é ótima, dando sempre o caminho de custo mínimo. No entanto, se os custos das transições não forem todas iguais, não é garantido que o algoritmo forneça a solução ótima.

Na **Pesquisa em Profundidade** cada nó é explorado até ao nível mais profundo do grafo. Um inconveniente deste algoritmo é a redundância de caminhos, e o facto de revisitar nós já explorados. No entanto, apresenta vantagem ao nível da complexidade de espaço em memória. Para cada nó, apenas é armazenado um caminho desde o nó início até ao nó fim, e todos os nós vizinhos ainda não-explorados. Depois de um nó ter sido expandido, e respetivamente todos os seus nós-filhos explorados, esse caminho pode ser eliminado da memória.

O **algoritmo de Dijkstra** é semelhante à pesquisa em largura, mas considera que os custos de cada iteração podem assumir qualquer valor positivo. O algoritmo expande os nós, de forma semelhante à pesquisa em largura, no entanto os nós vizinhos do nó expandido são ordenados de acordo com f(n), que corresponde a g(n), já que neste método não é usada qualquer heuristica (h(n)=0).

O **algoritmo A*** é dos mais usados em Robótica para Path Planning, em grande parte devido às seguintes propriedades:

- Ótimo;
- Consistente, significa que entre nós com ligação entre si, a estimativa de chegar ao destino, é sempre menor ou igual ao custo de chegar ao nó vizinho mais o custo desde esse nó ao destino;
- Completo, isto é, apresenta sempre solução quando esta existe;
- Admissível.

Será apresentado o pseudocódigo do A* em 1, no entanto para a sua compreensão são necessárias as definições seguintes, Moreira et al. (2009):

- O: lista aberta que contém os nós ainda não foram analisados e que poderão vir a ser escolhidos.
- C: lista fechada que contém os nós já processados.
- Star(n): conjunto de nós vizinhos/adjacentes ao nó n.
- c(n1, n2): custo de ir desde o nó n1 ao nó n2.
- n_{init}: Nó Inicio.

- n_{end}: Nó Fim.

Algorithm 1 A*

$O \leftarrow n_{init}$ {A lista aberta é inicializada com o nó início}
for $O \neq NULL$ **do**
  1.Escolher o nó com função f(n) menor, n_{melhor}
  2. $C \leftarrow n_{melhor}$ {Retirar o nó melhor da lista aberta}
  if n_{melhor} == n_{end} **then**
    return Termina algoritmo
  end if
  for Star(n_{melhor}) **do**
    if ($ni \notin O$) and ($ni \notin C$) **then**
      (Se não estiver na lista aberta nem na lista fechada)
      $O \leftarrow ni$
    else if $ni \in O$ **then**
      (Se estiver na lista aberta)
      Verificar se seguindo este caminho, o custo é menor. Se sim, então alterar o pai do nó x para o n_{melhor}
    else if $ni \in C$ **then**
      (Se estiver na lista fechada)
      Verificar se f(n) é melhor do que quando foi processado. Se sim, então alterar o pai do nó x para o n_{melhor}
      $O \leftarrow ni$
    end if
  end for
  return Termina algoritmo
end for

Este algoritmo usa uma função heuristica para estimar o custo desde um dado nó n ao destino. Existem várias heuristicas que se podem aplicar, e tentando sempre que estas sejam o mais próximo possivel da realidade. Geralmente em Robótica, o algoritmo A* é aplicado sobre um mapa em grelha, e uma das heuristicas mais seleccionada é a distância entre uma célula e a célula de destino, ignorando a existência de obstáculos. É a chamada distância euclidiana, o caminho em linha reta até ao destino.

O algoritmo A* , como vimos, planeia o caminho inicial entre dois pontos, usando a informação do mapa; move o robô ao longo do trajeto até que este, alcance o ponto final ou detecte uma discrepância entre o mapa e o ambiente real. Nesse caso, reconstrói o mapa e recalcula o caminho. Esta solução pode tornar-se pouco eficiente principalmente em ambientes dinâmicos, isto é, que sofram muitas alterações, e em que essas alterações ocorram longe do ponto de destino ou ainda existam poucas informações do mapa. Assim, surge o D*, como sendo uma generalização do A*, e aplicado a este tipo de ambientes. O D* é um algoritmo incremental que mantém o custo ótimo parcial dessas localizações, atualizando o mapa, e permitindo reduzir os custos computacionais.

2.2.2 Bibliotecas Path Planning

Com a evolução da Robótica, têm sido criados recursos que permitem a partilha e troca de ideias entre a comunidade de Robótica. Assim, é possivel encontrar bibliotecas no contexto dos nossos trabalhos, cujo problema já foi estudado por alguém, e que disponibilizou a sua implementação. Isto permite reutilizar código, poupar tempo, e focalizar a atenção noutros problemas mais complexos. Nesse sentido, importa referenciar aqui, algumas bibliotecas existentes que implementem algoritmos de path planning. Serão apresentadas também as principais utilizações a que essas bibliotecas se destinam, e a sua adequação ou não, a este projeto.

Existem dois softwares que permitem a integração de vários módulos/bibliotecas para path planning. Farei referência aqui a dois:

- **MoveIt**: De acordo com moveit.ros.org (a) é um dos principais softwares em Robótica para manipulação móvel. Inclui as versões mais recentes de manipulação, perceção 3D, cinemática, controlo e navegação. No que diz respeito ao planeamento de caminhos e movimento do robô, este software disponibiliza a integração de vários métodos de path planning e de várias bibliotecas, através de um serviço ROS. Na imagem seguinte está representada a arquitectura de alto nível para o nó primário move_group, nó este fornecido pelo MoveIt. Este nó recebe todos os componentes individuais do sistema (dados dos sensores, exemplo) e fornece um conjunto de serviços ROS que poderão ser usados pelos utilizadores.

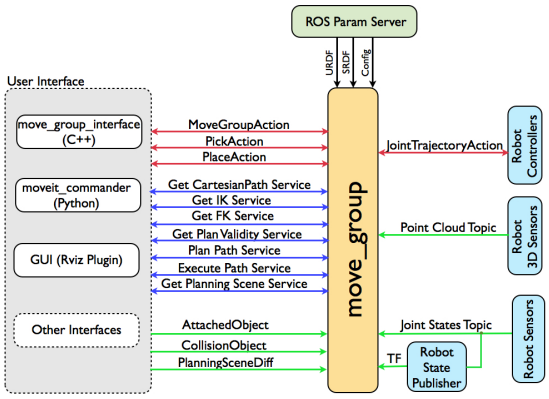


Figura 2.3: System Architecture for the primery node move_group (moveit.ros.org (a))

A interface para os vários algoritmos de path planning são fornecidos através de um serviço ROS, dado pelo nó move_group. Geralmente, e por definição, o MoveIt recorre à biblioteca **OMPL**, que será desenvolvida a seguir, para a implementação dos algoritmos de planeamento de caminhos.

- **KineoWorks**: É um software comercializado pela Siemens que permite a computação de trajetórias evitando colisões, para qualquer sistema cinemático, embora pela pesquisa realizada seja mais indicado para braços robóticos. Este software controla as caracteristicas necessárias para um controlo de movimentos seguros, e confiáveis: (siemens (2014))

  - Cinemática: o robô pode ser decomposto como um conjunto de articulações podendo ser introduzidas restrições conforme as caracteristicas principais do sistema.
  - Deteção de Colisões: com métodos rápidos e exatos, detecta colisões ao longo da trajetória.

Além destes softwares importa analisar também a biblioteca **OMPL** já referida. Esta biblioteca fornecida pelo ROS, reúne um conjunto de algoritmos path planning, não considerando deteção de colisões nem visualização. No entanto, o que se procura nesta fase, é apenas uma biblioteca que permita implementar o algoritmo A*. OMPL apresenta duas variantes que poderiam ser interessantes para este projeto. São elas:

- **PRMstar**: Executa o algoritmo PRM (Probabilistic Roadmap), usando a estratégia "star". Consiste na construção do roadmap e planeamento do caminho usando métodos probabilisticos. Considerando o espaço do ambiente de operação do robô, este algoritmo executa automaticamente o número de vizinhos para cada nó, tendo em conta o número de células, em que o espaço é subdividido. ompl (a)

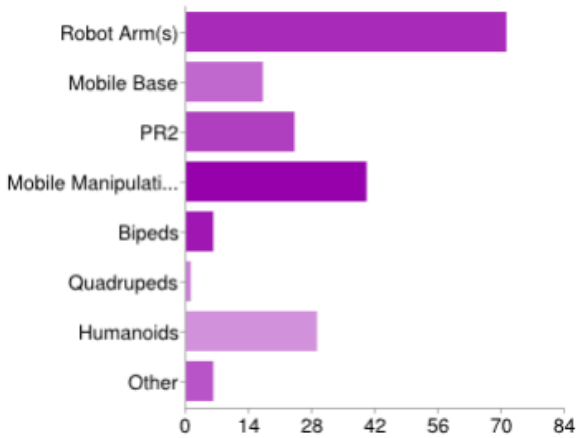
- **RRTstar**: Implementa o algoritmo incremental, RRT (Rapidly-exploring Random Trees), com garantia de optimabilidade.A noção de optimabilidade, depende da função heuristica da distância, definida no Espaço de Estados onde se é operado. ompl (b)

Ambos os algortimos PRM* e RRT* são baseados em amostragem. Nas duas aproximações são conetados pontos amostrados aleatoriamente, pertencentes ao espaço, apenas diferem na forma como o gráfico coneta esses pontos (Karaman e Frazzoli).

PRM começa por construir o Roadmap que representa um conjunto de trajetórias livres de colisão, e posteriormente fornece o caminho mais curto entre um estado inicial e um estado final, ao longo do mapa. Este método é probabilisticamente completo, de tal forma que a probabilidade de falhar decai exponencialmente para zero com o aumento do número de amostras. Em muitas aplicações, a construção do roadmap à priori é um desafio, ou inviável, ou porque o ambiente é demasiado estruturado, ou o ambiente é alterado muitas vezes. Nesse sentido, surge o RRT, um algoritmo incremental. Este tipo de algoritmos evita a necessidade de definir o número de amostras à priori, e retorna uma solução assim que considerar suficiente o conjunto de trajetórias construídas.

Apesar destas soluções serem interessantes, a ideia para a implementação deste projeto era recorrer ao algoritmo A*, pois lida com ambientes dinâmicos, com sistemas com muitas restrições, e já foi experimentado em várias plataformas.

Em relação aos softwares aqui mencionados, verificou-se serem mais adequados a aplicações com manipuladores robóticos. A imagem 2.4, foi retirada de um relatório que considera as respostas a algumas perguntas sobre o MoveIt, aos seus utilizadores, moveit.ros.org (b). Neste caso, foi questionado em que tipos de robôs é que o MoveIt está a ser utilizado. E como podemos verificar, a percentagem maior corresponde aos braços robóticos.



Robot Arm(s)	71	37%
Mobile Base	17	9%
PR2	24	12%
Mobile Manipulation Robots (Other than PR2)	40	21%
Bipeds	6	3%
Quadrupeds	1	1%
Humanoids	29	15%
Other: (UAVs, Lego NXT, Quadrotor)	6	3%

Figura 2.4: Survey of what kind of robots using MoveIt (moveit.ros.org (b))

Assim, foi decidido não usar nenhuma destas bibliotecas, e construir um nó ROS que execute o algoritmo A*. Dessa forma, pode ser sempre customizado. Isto é, existe sempre a possibilidade de acrescentar algumas melhorias ao algoritmo, nomeadamente na deteção de colisões, ou adaptação do movimento do AGV consoante a velocidade do obstáculo.

2.2.3 Metodologias Coordenação Multi-Robô

Novos desafios têm sido propostos à comunidade da Robótica, e a resolução de problemas cada vez mais complexos, é exigida. São múltiplas as situações onde um robô a operar sozinho, não atinge a performance requerida. Assim, a utilização de múltiplos robôs a cooperarem entre si, permite obter melhores resultados, por diferentes razões:

- A distribuição de tarefas por múltiplos robôs cooperando entre si, permite maior rapidez na sua execução, procurando igualmente obter a solução de custo mínimo. Permite subdividir uma tarefa em sub-tarefas, e executá-las de forma concorrente.

- O facto de existir redundância de sensores, permitir obter uma localização mais precisa.

- A solução obtida é mais robusta, na medida em que existe redundância de informação, e portanto maior certeza nos dados obtidos. Um exemplo, é na construção de mapas.

- Outro aspeto é a robustez no que diz respeito a falhas. Caso algum robô avarie, a falha é comunicada à equipa e portanto, pode ser compensada, atribuindo as tarefas desse robô, aos restantes.

Existem muitas áreas onde os sistemas multi-robô podem ser aplicados. Algumas delas, (Dias (2004)) são: gestão automática de armazéns, exploração de ambientes, linhas de montagem, agricultura, ambientes inteligentes e muitas outras.

Obstante estas vantagens, um sistema deste tipo apresenta sempre restrições e exigências de performance, quer a nível de rapidez na execução de tarefas, segurança, comunicação, capacidade de identificar informação inconsistente. Por isso, a implementação de um algoritmo de coordenação multi-robô é complexo e exigente. Muitos autores, têm estudado este problema, e já existem muito exemplos de implementação. Aqui tentarei generalizar as resoluções encontradas, e apresentar algumas.

Segundo Dias (2004), é possivel distinguir 3 abordagens:

- Centralizada;
- Distribuída;
- Baseada na Economia;

Uma abordagem **centralizada** considera o sistema multi-robô como um sistema de um único robô com vários graus de liberdade. É seleccionado um líder, que distribuirá funções a cada robô. Esta metodologia apresenta principal vantagem, em ambientes estáticos (que não sofram alterações) e em que o sistema seja constituído por um número reduzido de robôs. Nestas condições, este método tem fortes probabilidades de conduzir a planeamentos ótimos. No entanto, esta abordagem apresenta também desvantagens. No caso de ocorrer uma falha no líder, o funcionamento de todo o sistema fica comprometido, levando a um algoritmo de selecção de outro líder, que é igualmente complexo e conduzindo a instantes de tempo, em que o sistema fica desativo, sem sistema de gestão. A nível de requisitos computacionais, é exigente, na medida em que a informação relevante de todos os sensores e mapa, são enviados para uma única central para processamento.

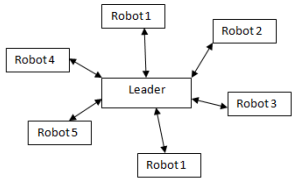


Figura 2.5: Centralized Architecture

A abordagem **distribuída** usa métodos que permitem distribuir responsabilidades de planeamento por todos os membros da equipa. Cada robô opera em grande parte de forma independente, mediante a informação disponível dos sensores e planeia o seu percurso com base nas informações

locais. A nível de comunicação não é tão restritivo, já que os robôs podem comunicar apenas com os da sua periferia. Em relação à abordagem centralizada, apresenta a vantagem de menor poder computacional, maior robustez e maior rapidez na execução de tarefas. Neste caso, nenhum robô fica dependente de um líder, logo já não existe um ponto de falha.

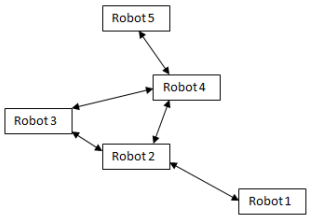


Figura 2.6: Distributed Architecture

A abordagem **baseada na economia** é usada para o controlo de sistemas multi-agentes. Num sistema multi-agente, cada robô-agente tem uma existência própria, é uma entidade computacional que toma as suas decisões baseadas nas alterações do que o rodeia. Dias (2004) faz a distinção entre controlo multi-agente e controlo multi-robô. Na primeira, as restrições mais frequentes relacionam-se com recursos, enquanto que na segunda, consideram-se as restrições de comunicação. O seu controlo é bem diferente, na medida em que os sistemas robóticos devem permitir acomodar intervalos de erros provenientes das falhas dos sensores, e da sua interação com o ambiente, tal não acontece com os robôs-agentes. Além disso, é expectável que os sistemas robóticos estejam mais sujeitos a falhas, e estas sejam de naturezas mais variadas.

Até agora foram sintetizadas as abordagens possíveis para a resolução de um problema multi-robô. Para clarificar, serão apresentados 3 metodologias práticas, resultantes da pesquisa efectuada. É de realçar, que sendo um problema complexo, existem muitos artigos, com exemplos práticos, e que diferem uns dos outros. No entanto, em todos os eles, existem duas fases que compõem um problema de coordenação multi-robô: a alocação de tarefas, e a coordenação propriamente dita. Esta dissertação foca-se na coordenação, considerando que as tarefas já foram atribuídas, no entanto, para clarificação das metodologias apresentadas, será referida por vezes, a forma como é feita a alocação das tarefas.

Serão apresentados os 3 trabalhos seguintes:

1. **CoMutaR;** Este trabalho teve como objetivo a distribuição de tarefas por equipas de robôs moveis, garantindo a coordenação entre as equipas formadas. M.Shiroma e Campos (2009) A sua descrição encontra-se na sub-secção 2.2.3.1.

2. **Trafcon;** Esta metodologia considera um sistema de AGV's real baseado num armazém automático e o objetivo seria controlar o tráfego explorando os diagramas de coordenação, e tendo em conta todas as restrições do ambiente real. Na secção 2.2.3.2 é possivel ver a descrição desta metodologia.

3. **TraderBots;** O principal objetivo é o desenvolvimento de um método baseado na Economia que solucione o problema de coordenação multi-robô, em ambientes dinâmicos, Dias (2004). Na sub-secção 2.2.3.3 encontra-se uma breve descrição desta implementação.

2.2.3.1 CoMutaR

A metodologia **CoMutaR(Coalition formation based on Multi-tasking robots)**, é usada tanto para a distribuição de tarefas por equipas de robôs como para garantir a coordenação entre as equipas. Esta abordagem difere das restantes, na medida em que, um robô pode ter de executar várias tarefas em simultâneo, robô multi-task. Inicialmente começam por definir um conjunto de robôs, R, e um conjunto de tarefas, T, sendo m o número de robôs, e p o número de tarefas, tal que:

$$R = r_1, r_2, ..., r_m$$

$$T = t^1, t^2, ..., t^p$$

São definidos dois problemas:

1. Alocação de tarefas: encontrar a função A:T $\mapsto$R, tal que $A(t^k)$ é a equipa de robôs capazes de satisfazer a tarefa k.

2. Coordenação equipa: Coordenar as ações, tal que sejam capazes de cumprir a tarefa t^i.

Será apenas analisado aqui o problema 2. Consideremos que temos os seguintes dados:

- Um conjunto de robôs, R.
- Um conjunto de tarefas, T.
- A alocação de tarefas, A.

Sendo $a_{i,j} \longrightarrow$Ação/tarefa j no robô i, podemos definir um conjunto de recursos limitados, $X^{i,k}$, no que concerne à partilha do respetivo recurso. Um recurso restrito pode pertencer a um robô (energia, posição) ou ser intrínseco ao ambiente (configuração do espaço):

$X^i = \{X^{i,1}, X^{i,2}, \ldots, \} \longrightarrow$ Recursos de participação restrita, se i=0 pertencente ao ambiente, se i$\geq$0, pertencente ao robot (i=1, 2, ..., m).

Para cada recurso $X^{i,k}$ é verificada a sua disponibilidade, associando a este um Domínio, $C^{i,k}$. Para cada ação $a_{i,j}$ é definida uma função, dada por:

$$\alpha_{i,j}^{l,k} : R \mapsto C^{l,k}$$

Esta função de restrição é função do tempo, e mede a quantidade de recursos partilhados, $X^{l,k}$ do robô, requeridos pela ação $a_{i,j}$.

O espaço $C^{l,k}$ admite dois operadores que irão definir as restrições, que condicionará a tomada de decisão de um robô. Esses operadores são, o operador composição $\oplus$ (a soma das restrições impostas por um dado número de funções restrição) e o operador comparação $\prec$, que toma os valores $\{true, false\}$, e verifica se a soma das funções restrição excedem uma dada capacidade máxima, $\alpha_{max}^{l,k}$.

Desta forma, usando os operadores, podemos definir,

$$\sum_{\mathbf{j=1}}^{\mathbf{n_i}} \varphi_{i,j}^{l,k} \cong \varphi_{i,1}^{l,k} \oplus \varphi_{i,2}^{l,k} \oplus \ldots \oplus \varphi_{i,n_i}^{l,k}$$

como sendo a **restrição imposta por todas as ações** em execução no robô R_l sobre o recurso partilhado $X^{l,k}$. Da mesma forma, temos de somar para todos os robôs:

$$\sum_{\mathbf{i=1}}^{\mathbf{m}} \varphi_{i,j}^{l,k} \cong \varphi_{1,j}^{l,k} \oplus \varphi_{2,j}^{l,k} \oplus \ldots \oplus \varphi_{m,j}^{l,k}$$

Esta expressão representa a **restrição imposta por todos os robôs**, sobre o recurso partilhado $X^{l,k}$.

Concluindo, podemos verificar se a soma de todas as funções restrição impostas por todos as robôs, excede a capacidade máxima de um dado recurso partilhado $X^{l,k}$, pela seguinte expressão:

$$\sum_{i=1}^{m} \sum_{j=1}^{n_i} \varphi_{i,j}^{l,k} \prec \varphi_{max}^{l,k}$$

Nesta metodologia, descrita em M.Shiroma e Campos (2009), a primeira fase é definir o domínio φ_{max}, e os operadores $\oplus$ e $\prec$. Por exemplo, para um barramento de comunicação, especialmente por wireless, muitas vezes usada entre robôs para comunicarem, apresenta uma largura de banda limitada. Assim, neste caso, φ_{max} correspondia à máxima largura de banda. O domínio seria um conjunto de números reais e a função de restrição $\varphi_{i,j}$ correspondia à largura de banda requerida pela ação $a_{i,j}$, necessária para que esta executasse as suas operações. O operador $\oplus$ seria a soma algébrica e o operador $\prec$ seria 'menor do que, ou igual'. A análise efetuada consistiria em somar as necessidades em termos de largura de banda para cada ação, e verificar se era inferior ou igual ao limite máximo.

2.2.3.2 TRAFCON

O algoritmo proposto em Secchi et al. (2014), é baseado num CD(Coordination Diagram), que permite mapear um problema de coordenação num problema de planeamento. No entanto, usando apenas essa metodologia, o sistema apresentaria algumas desvantagens tais como: poder computacional elevado, logo não seria aplicável a sistemas reais de AGV's com muitos veículos. Além disso, sendo completo, a coordenação seria computada antes dos robôs iniciarem a marcha, e no caso do veículo ter de alterar a sua trajetória, devido a um obstáculo inesperado, por exemplo,

a computação devia ser repetida do zero. Por isso, o que o algoritmo **TRAFCON** apresenta é uma versão extendida da ferramenta CD proposta em sistemas multi-robô. Baseia-se numa estratégia de coordenação acoplada a um algoritmo de routing dinâmico. Neste artigo, a metodologia foi implementada num sistema real de AGV's. Cada AGV foi dotado de um laser de segurança na frente, e a navegação é realizada por triangulação, através de refletores distribuídos pelas paredes da fábrica. O roadmap é dividido em segmentos e cada segmento é ocupado apenas por 1 AGV. Por razões de segurança, os AGV's não podem andar para trás. Assim, o método TRAFCON apresenta estas limitações que podem ser relevantes e desfavoráveis, dependendo da aplicação. A arquitetura de controlo, pode ser resumida na seguinte imagem, Secchi et al. (2014).

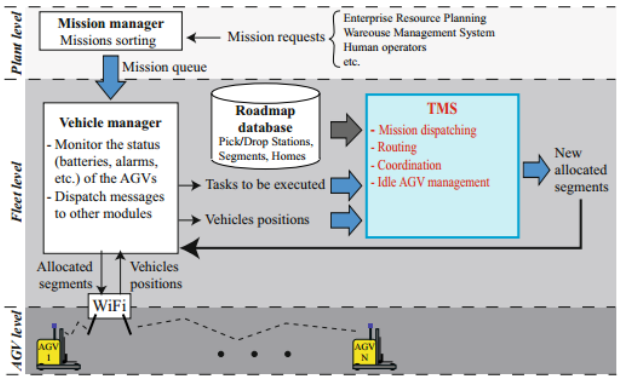


Figura 2.7: Metodologia para o sistema de controlo, para o projeto TRAFCON

A arquitetura de controlo consiste nos seguintes sistemas:

- ERP System: Recebe um conjunto de tarefas e gera um conjunto de 'missões' para os AGV's.
- A unidade central é constituída por:
  - Vehicle Manager: Dá a posição do AGV, nomeadamente o segmento da fábrica que ocupa.
  - Traffic Manager System: Controla um grupo de AGV's, evitando colisões. Este define quantos e quais os segmentos que estão associados a um AGV, durante o seu trajeto, usando uma estratégia de gestão.
- Sistema de Baixo nível, para controlo dos comandos para cada AGV.

Para a definição do algoritmo, comecemos por definir os seguintes parâmetros:

- $\Re$: Roadmap constituído por várias curvas/segmentos τ.
- p_i: caminho definido como uma sequência de segmentos adjacentes, n_i.
- A_i: veículo i atribuido ao caminho p_i.

- $A(x)$: Volume ocupado por um AGV em $\Re$.

Antes de um veículo se mover de um segmento para outro (de τ para τ'), é verificado se esse deslocamento não irá provocar colisão, através da seguinte condição:

$$A(\tau(\alpha)) \cap A(\tau'(\beta)) \neq 0$$

$$(\alpha, \beta) \in [0, l_\tau] \times [0, l_{\tau'}]$$

Se esta condição for verdadeira, significa que existe colisão. Assim, sempre que é atribuída uma missão a um AGV, o caminho é planeado (entre $x_i^{init} ex_i^{goal}$). É explorado o CD e analisadas possíveis colisões, pela expressão acima. Posteriormente para cada veículo é disponibilizada uma lista dos segmentos que não irão colidir com os segmentos já reservados. Mediante essa lista é possivel gerir as trajetórias para cada robô.

2.2.3.3 TraderBots

Esta metodologia, descrita em Dias (2004), difere das anteriores, na medida em que usa a analogia de um sistema económico para a coordenação entre os robôs. Isto é, não existe um sistema de gestão de alto nível, cada veículo é considerado uma entidade independente que toma as suas decisões, e negoceia com os outros elementos da equipa. Para que o 'bem comum' seja alcançado, máximo lucro, dois robôs negoceiam de forma a tomarem a melhor decisão para os dois. Assim, o objetivo da equipa é completar as tarefas que lhe foram atribuídas, com sucesso e maximizando os lucros totais (diferença entre receita e custos) enquanto cada robô maximiza o seu lucro individual. Será apresentada, de forma breve, a forma como a equipa gere os seus custos, receitas e lucros.

Considere-se:

- Função trev: corresponde ao mapeamento de possíveis resultados de uma tarefa, em termos de valores da receita. Por exemplo, pode ser uma combinação do tempo gasto, com o combustível gasto.

- Função tcost: corresponde ao mapeamento de possíveis esquemas/alternativas para executar a tarefa, em ordem ao custo. Por exemplo, quantificação da prioridade dada à tarefa pelo operador, e o risco envolvido na execução da tarefa.

O objetivo da equipa é executar um plano, cujo lucro, dado por, $trev(P) - tcost(P)$ seja maximizado.

Cada robô, apresenta:

- Função rrev: corresponde à receita para cada robô, por exemplo, número de objetos transportados.

- Função rcost: corresponde aos custos para cada robô, exemplo, em ordem à energia gasta.

Se uma tarefa está em oferta a um preço máximo r, e o custo para realizar a tarefa é c, resulta num lucro máximo p, dado por $p = r - c$. Cada robô, calcula a sua proposta b, $b = \alpha p + c$. α depende de quão competitivo o robô licitante quer ser. Se α alto, então o licitante tem a oportunidade de aumentar o seu lucro, mas corre um risco maior de perder a tarefa para outro robô, que apresente oferta menor. Esta negociação entre robôs, é realizada de forma, a que o negócio seja vantajoso para ambas as partes.

O papel que cada robô desempenha na equipa, pode variar consoante as tarefas que este queira licitar.

A grande vantagem desta metodologia, é a possibilidade de poder ser integrada em sistemas dinâmicos.

Capítulo 3

Problema a tratar

3.1 Definição do Problema

O problema central desta dissertação consiste em desenvolver um algoritmo que permita a coordenação de uma equipa de AGV's, de forma a evitar colisões e otimizar o caminho percorrido por cada um deles. Dessa forma, o problema poderá ser decomposto em várias fases, por forma, a simplificá-lo. Numa fase inicial, será implementado um algoritmo que determine de forma ótima, o melhor caminho, para apenas um robô, entre um ponto inicial e um final, evitando obstáculos. Posteriormente é que serão integrados vários robôs e delineada uma estratégia para que eles cooperem entre si. Esta dissertação contempla também o desenvolvimento de um ambiente de simulação, próximo da fábrica da PSA, daí a necessidade de escolher um software para a simulação.

3.2 Descrição das plataformas a utilizar

Para o desenvolvimento desta dissertação serão usadas duas ferramentas/softwares. Tal como vimos no capitulo 2, foi escolhido o V-Rep como software para o desenvolvimento do ambiente de simulação. Por outro lado, e tendo em conta que o trabalho resultante desta dissertação, será integrado no projeto STAMINA, considerou-se também interessante usar a ferramenta ROS. Estes dois módulos serão explicados a seguir.

3.2.1 Simulador Escolhido

Tal como justificado no capítulo 2, foi seleccionado o V-Rep. Considerou-se que para os requisitos do projeto STAMINA, este seria o software mais adequado, para a realização da simulação.

3.2.2 Integração dos vários módulos do projeto

O ROS (Robotic Operating System) é uma framework open-source adequada ao desenvolvimento do código para um robô. Qualquer sistema robótico comporta um conjunto de módulos extenso, que poderá ser complicado de gerir quando se trabalha em equipa, ou quando esses módulos são complexos. O ROS oferece um conjunto de bibliotecas, e ferramentas de forma a simplificar a construção de um robô complexo e robusto (ros.org). Assim, é possivel subdividir tarefas pelos elementos da equipa, e no fim, fazer a sua integração, de forma simples, numa arquitectura de desenvolvimento cooperativo.

Segundo Martinez e Fernández (2013), ROS foi originalmente desenvolvido em 2007 pelo Laboratório de Inteligência Artificial de Stanford (SAIL). Em 2008, o desenvolvimento prosseguiu com o Instituto Willow Garage e mais 20 instituições. A partir daí várias unidades de investigação começaram a usar o ROS nos seus projetos, e a partilhar o seu código. Em simultâneo, muitas empresas começaram a adaptar os seus produtos para serem usados com ROS. Apesar de não ser um sistema operativo, fornece algumas funcionalidades standard de um SO, tais como controlo de dispositivos de baixo-nível, envio de mensagens entre processos, e gestão de pacotes.

É baseado numa topologia centralizada, onde o processamento é realizado em nós, que recebem ou enviam dados. Esses dados podem ser provenientes de sensores, variáveis de controlo, planeamento e/ou atuadores. A biblioteca é direccionada para Unix.

A ideia fundamental é a reutilização de código, por isso os seus utilizadores, usam o código já existente, modificam-no de acordo com a sua aplicação e voltam a partilhar.

Desta forma, considerou-se o ROS uma plataforma propícia a esta dissertação, para facilitar a integração dos diversos módulos.

Capítulo 4

Planeamento

O planemento é uma das etapas mais importantes de um projeto, sendo decisivo para o seu sucesso. Este permite estabelecer objetivos iniciais, a longo ou curto prazo, estabelecer metas e datas limites. Desta forma, é possivel gerir melhor os recursos, e organizar o tempo, de forma a atingir os objetivos. Neste capítulo são apresentadas as tarefas do projeto, bem como as datas previstas para o início e conclusão de cada uma delas, secção 4.1. Na secção 4.2, encontra-se a metodologia adotada para a realização do trabalho.

4.1 Calendarização das tarefas do projeto

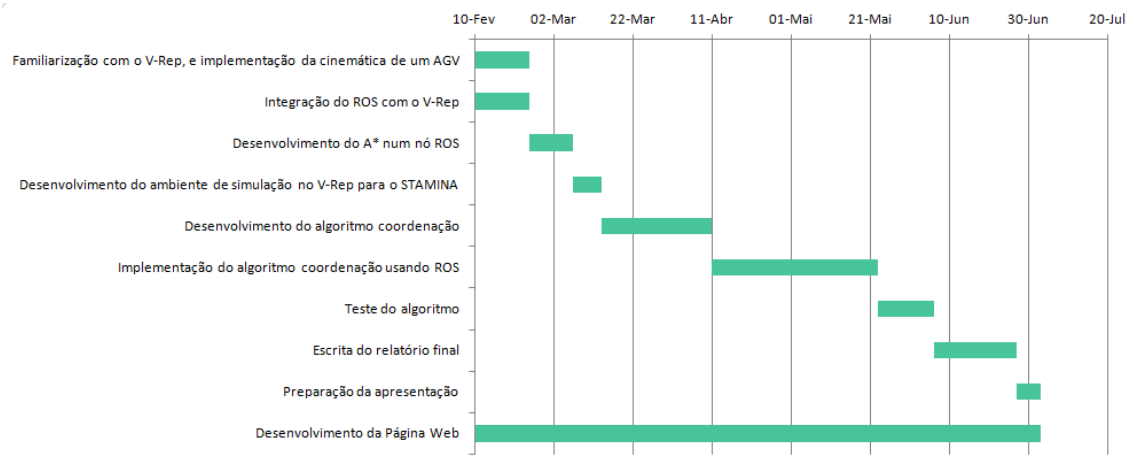


Figura 4.1: Diagrama de Gantt com as tarefas principais

Na figura 4.1 encontra-se o diagrama de Gantt correspondente às principais tarefas, a realizar. Este tipo de diagramas permite uma representação visual das tarefas concluídas, não concluídas e não iniciadas. Neste caso, a verde encontram-se as tarefas ainda não-iniciadas. A distribuição temporal adotada é em dias, assim como a duração. Tal distribuição pode ser verificada na tabela 4.1.

	Start	Duration	Finish
Familiarização com o V-Rep, e implementação da cinemática de um AGV	10-02-2014	14	24-02-2014
Integração do ROS com o V-Rep	10-02-2014	14	24-02-2014
Desenvolvimento do A* num nó ROS	24-02-2014	18	07-03-2014
Desenvolvimento do ambiente de simulação no V-Rep para o STAMINA	07-03-2014	7	14-03-2014
Desenvolvimento do algoritmo coordenação	14-03-2014	28	11-04-2014
Implementação do algoritmo coordenação usando ROS	11-04-2014	42	23-05-2014
Teste do algoritmo	23-05-2014	14	06-06-2014
Escrita do relatório final	06-06-2014	21	27-06-2014
Preparação da apresentação	27-06-2014	6	03-07-2014
Desenvolvimento da Página Web	03-02-2014	150	03-07-2014

Tabela 4.1: Distribuição temporal das tarefas

- **Desenvolvimento do A* num nó ROS;** O algoritmo A* será implementado, num nó ROS permitindo maior modularidade e fácil integração com outras partes do projeto. O objetivo da implementação do A* é o planeamento do caminho até um ponto de destino, para cada robô.

- **Integração do ROS com o V-Rep;** Como já referido, será usado ROS para integrar os algoritmos para o controlo dos AGV's. Nesse sentido, é necessário realizar a integração dos nós ROS que contêm cada um dos algoritmos com o simulador escolhido, V-Rep.

- **Desenvolvimento do ambiente de simulação no V-Rep para o STAMINA;** Os testes anteriores serão realizados usando um ambiente de simulação simples, com poucas estações de inicio e fim. Por isso, nesta fase, será construído o ambiente de simulação o mais fiel possível ao ambiente da PSA, onde os robôs irão operar.

- **Desenvolvimento do algoritmo coordenação;** Formular algoritmo de coordenação multi-robô.

- **Implementação do algoritmo coordenação usando ROS;** Tal como no algoritmo A*, também este algoritmo será programado em C, e implementado num nó ROS.

- **Teste do algoritmo;** Testar robustez do algoritmo, e acrescentar melhorias, ou restrições.

- **Escrita do relatório final;** Será realizado um relatório com o desenvolvimento deste projeto e respetivas conclusões. Ele refletirá todo o trabalho desenvolvido ao longo do semestre.

- **Preparação da apresentação;** Na distribuição das tarefas, foi também contemplado o tempo necessário para a preparação da apresentação oral do projeto.

- **Desenvolvimento da Página Web;** Ao longo do semestre será realizada uma página web que será constantemente atualizada, conforme a evolução do trabalho.

4.2 Metodologia

A metodologia adotada consiste nos seguintes passos:

- Levantamento dos simuladores existentes no mercado, e que cumpram os requisitos do projeto. Comparação de 4 softwares de simulação para fundamentação da escolha.
- Levantamento de algoritmos de path planning e de coordenação multi-robô.
- Familiarização com o ambiente ROS, e implementação do A* num nó ROS.
- Integração do nó A* no simulador V-Rep com um ambiente de simulação simples.
- Desenvolvimento do ambiente de simulação, próximo do ambiente onde os AGV's irão operar.
- Desenvolvimento de um algoritmo para a coordenação multi-robô.
- Integração do algoritmo no V-Rep.
- Realização de testes.
- Escrita do relatório final, que reflita o trabalho desenvolvido.

4.3 Ferramentas para o desenvolvimento do trabalho

No desenvolvimento da dissertação, serão usadas diversas ferramentas, que permitirão executar e testar os algoritmos implementados. Assim, será usado ROS, versão Fuerte, por permitir modularidade e partilha de módulos. Posteriormente, foi seleccionado o software de simulação V-Rep pelo seu ambiente user-friendly, documentação completa e características que cumprem os requisitos do projeto. Finalmente, para a escrita da dissertação será usada a ferramenta LaTeX, não só por contemplar as formatações pré-definidas para a construção de documentos técnicos e científicos, mas também pela sua facilidade em corrigir e formatar, em qualquer fase do projeto.

Referências

H. Choset. Probabilistic roadmap path planning.

coppeliarobotics. URL http://www.coppeliarobotics.com/features.html.

Paulo Costa, José Gonçalves, José Lime e Paulo Malheiros. Simtwo realistic simulator: A Tool for the Development and Validation of Robot Software. *Theory and Applications of Mathematics & Computer Science*, 1:17–33, 2011. URL http://uav.ro/applications/se/journal/index.php/TAMCS/article/view/3.

Paulo José Cerqueira Gomes da Costa. Simtwo. URL http://paginas.fe.up.pt/~paco/wiki/index.php?n=Main.SimTwo.

Pedro Luís Cerqueira Gomes da Costa. *Planeamento Cooperativo de tarefas e trajectórias em múltiplos robôs*. Thesis, 2011.

M. B. Dias. Traderbots: A new paradigm for robust and efficient multirobot coordination in dynamic environments, 2004. URL http://search.proquest.com/docview/305207631?accountid=43623. Copyright - Copyright UMI - Dissertations Publishing 2004; Last updated - 2014-01-08; First page - n/a; M3: Ph.D.

Sertac Karaman e Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. Report, Massachusetts Institute of Technology, Cambridge.

Lydia Kavraki, Petr Svestka, Jean-Claude Latombe e Mark Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *Robot. Autom. IEEE*, 1996.

Steven M. LaValle. Rapidly-exploring random trees: A new tool for path planning, 1998.

Aaron Martinez e Enrique Fernández. *Learning ROS for Robotics Programming*. 2013. ISBN 1782161449.

Saeid Mokaram, Khairulmizam Samsudin, Abdul Rahman Ramli e Hamideh Kerdegari. Mobile robots communication and control framework for usarsim.

António Paulo Moreira, Paulo J.Costa e Pedro Costa. Real-time path planning using a modified a* algorithm, 2009.

moveit.ros.org. Moveit, a. URL http://moveit.ros.org/.

moveit.ros.org. Moveit! survey results, b. URL http://moveit.ros.org/data/surveys/MoveIt!-2013-Survey.pdf.

Pedro M.Shiroma e Mario F. M. Campos. Comutar: A framework for multi-robot coordination and task allocation, 2009.

ompl. Prmstar class reference, a. URL http://ompl.kavrakilab.org/classompl_1_1geometric_1_1PRMstar.html.

ompl. Rrtstar class reference, b. URL http://ompl.kavrakilab.org/classompl_1_1geometric_1_1RRTstar.html#gRRTstar.

Marcelo Roberto Petry. *A Vision-based Approach Towards Robust Localization for Intelligent Wheelchairs*. Thesis.

ros.org. About ros. URL http://www.ros.org/about-ros/.

Cristian Secchi, Roberto Olmi, Cesare Fantuzzi e Marco Casarini. Trafcon–traffic control of agvs in automatic warehouses. In *Gearing Up and Accelerating Cross-fertilization between Academic and Industrial Robotics Research in Europe:*, páginas 85–105. Springer, 2014.

Roland Siegwart, Illah R. LinkNourbakhsh e Davide Scaramuzza. *Introduction to Autonomous Mobile Robots*. 2nd ed edição, 2011.

siemens. Kineoworks, 2014. URL http://www.plm.automation.siemens.com/en_us/products/open/kineo/kineoworks/#lightview-close.

Aaron Staranowicz e Gian Luca Mariottini. A survey and comparison of comercial and open-source robotic simulator software. Report, University of Texas at Arlington.