Localização e Navegação de um Robô Móvel Omnidirecional: Caso de Estudo da Competição Robot@Factory

Paulo José Costa, Nuno Moreira, Daniel Campos, José Gonçalves, José Lima, e Pedro Luís Costa

***Title* - Localization and Navigation of an Omnidirectional Mobile Robot: The Robot@Factory Case Study.**

***Abstract*—The Robot@Factory competition, was recently included in Robotica, the main Robotics Portuguese Competition. This robot competition takes place in an emulated factory plant, where Automatic Guided Vehicles (AGVs) must cooperate to perform tasks. To accomplish their goals the AGVs must deal with localization, navigation, scheduling and cooperation problems, that must be solved autonomously. The presented robot competition can play an important role in education due to the inherent multi-disciplinary concepts that are involved, motivating students to technological areas. It also plays an important role in research and development, because it is expected that the outcomes that will emerge here, will later be transfered to other application areas, such as service robots and manufacturing.**

By presenting a scaled down factory shop floor, this competition creates a benchmark that can be used to compare different approaches to the problems that arise on this kind of environments. Also, the ability, in some restricted areas, to alter the environment, can promote the test and evaluation of different localization mechanisms, something that is usually, more restricted in other competitions, opening this area to be explored and benchmarked. In this paper it is discussed one of the possible approaches, that can be applied to the robot competition, serving as reference to the actual and new potential participating teams.

***Index Terms*—Robotics, Education, Localization, Navigation, Prototyping**

I. Introdução

HOJE em dia a indústria procura criar fábricas cada vez mais flexíveis, por exemplo no transporte de matéria prima entre postos de trabalho, usando AGVs, por forma a otimizar os tempos e permitir a rápida reconfigurabilidade de *layouts*. Estes transportadores atuam num ambiente de trabalho dinâmico em que podem surgir obstáculos, tal como os trabalhadores a cruzar com o robô, material caído ou armazenado no percurso e mesmo outros robôs a circularem em trabalho [1]-[6].

A procura pelo aumento de eficiência num ambiente fabril é uma problemática cada vez mais recorrente nos dias de hoje, tornando a temática de robôs móveis um motivo de intensa investigação científico-tecnológica. Este tema é constituído por três grandes áreas de estudo: controlo, localização e navegação.

Tendo em conta os paradigmas referidos, usou-se a competição Robot@Factory como plataforma de teste com o objetivo de que os resultados obtidos possam mais tarde ser utilizados para a resolução de problemas em fábricas reais. A competição Robot@Factory procura recriar um problema inspirado nos desafios que um robô autónomo terá de enfrentar durante a sua utilização numa fábrica. Esta fábrica é constituída por um armazém de aprovisionamento, um armazém de produto final e oito máquinas de processamento. A arena da competição encontra-se exemplificada na Figura 1.

A tarefa dos robôs consiste em transportar o material entre armazéns e máquinas. Para isso, estes deverão apresentar um mínimo de capacidades que incluem recolher, transportar e posicionar os materiais, localizar-se e navegar no ambiente fornecido, assim como evitar choques com paredes, obstáculos e outros robôs. A competição decorrerá em três mangas que apresentam desafios de dificuldade crescente. Esta pretende ser uma prova que permita a transição gradual ao nível de complexidade e exigência técnica entre as ligas juniores e seniores.

De acordo com a regulamentação da competição o simulador *SimTwo* surge como referência para teste do *software*. Assim, por forma a validar todos os algoritmos, para além da prototipagem do robô, foi desenvolvido o seu modelo simulado em *SimTwo*, acelerando desta forma o desenvolvimento do *software* para o AGV, garantindo um ambiente de implementação complementar e reduzindo a fadiga causada no *hardware* do robô [7] [8].

Na prototipagem do robô para a competição, a nível da locomoção existem diversas topologias, nomeadamente difer-

Fig. 1. Arena da competição Robot@Factory

P. J. Costa pertence ao Departamento de Engenharia Eletrotécnica e de Computadores, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal e-mail: paco@fe.up.pt.

N. Moreira e D. Campos pertencem ao Departamento de Engenharia Eletrotécnica e de Computadores, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal.

J. Gonçalves e J. Lima pertencem ao Departamento de Eletrotecnia, Instituto Politécnico de Bragança, Bragança, Portugal.

P. L. Costa pertence ao Departamento de Engenharia Eletrotécnica e de Computadores, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal.

encial, omnidirecional ou de *Ackerman*. No caso a aplicar foi escolhida a locomoção omnidirecional dado que ao contrário da de *Ackerman* não necessita de sistemas mecânicos auxiliares, e além disso permite efetuar movimentos de translação e rotação simultâneos, o que não se verifica nas outras topologias, permitindo aumentar sua competitividade.

Para colmatar a problemática da localização, é usual o recurso a sistemas de localização relativa, tais como odometria e sensores inerciais, bem como o recurso a sistemas de localização absoluta, sistemas esses que, com recurso a uma câmara de vídeo ou *laser range finders*, permitem a identificação de obstáculos e marcadores no meio envolvente. As soluções mais comuns são baseadas em sistemas de faixas, sistemas de triangulação e trilateração. No entanto, por forma a causar o menor impacto possível no meio fabril onde o robô se encontra inserido, foi adotado um sistema de localização absoluta baseado em dados de um *laser range finder*, o *Perfect Match*. Juntamente com a odometria e com o recurso ao Filtro de Kalman Estendido a efetuar a fusão sensorial, implementou-se um sistema de localização eficaz e independente de auxiliares externos para o seu funcionamento.

Após saber a localização torna-se imperativo definir os sistemas de navegação a usar, sendo que os mais frequentemente adotados são os de seguimento de linhas brancas, pois apresenta uma baixa complexidade de implementação, ou então o seguimento de trajetórias pré-definidas com o recurso as *waypoints* conhecidos no mapa, podendo também recorrer-se a métodos mais complexos de planeamento de trajetórias dinâmicas.

Uma vez que num ambiente industrial existem diversos fatores que variam dinamicamente, que levam a ser necessário replanear a trajetória em tempo real, sem restringir os movimentos possíveis, foi desenvolvido um algoritmo dinâmico baseado na expansão do mapa e pesquisa em grafo com a possibilidade de modificar o caminho com a presença de vários robôs [9]-[14].

O artigo está estruturado da seguinte forma: em primeiro lugar é apresentado o *SimTwo*, sendo o simulador oficial da competição, depois é apresentado o protótipo realizado, de seguida o seu sistema de localização e o planeamento de trajetórias. Finalmente são apresentados os resultados e a conclusão.

II. O SimTwo

O simulador *SimTwo* surge como o simulador oficial da competição *Robot@Factory* [15], representando o campo e todos os elementos da competição de forma fiel e numa escala muito próxima da real, sendo apenas necessário modelar o robô para o pretendido, existindo inclusive diversos sensores, como por exemplo de linha branca, LIDAR, câmara e sensores infravermelhos. Tal como se pode ver na Figura 2 a representação do campo simulado assemelha-se ao real.

O *Simtwo* [16] trata-se de um sistema de simulação e teste com um ambiente 3D, que permite recriar vários ambientes, onde se podem implementar diferentes tipos de robôs, como por exemplo com configurações omnidirecionais ou diferenciais. Além disso dispõe de uma aproximação realista, permitindo tomar considerações físicas como a forma, a massa, os atritos das superfícies, entre outros, e ainda dispõe de modelos que visam aproximar e capturar os elementos não lineares presentes nos motores que atuam nos robôs.

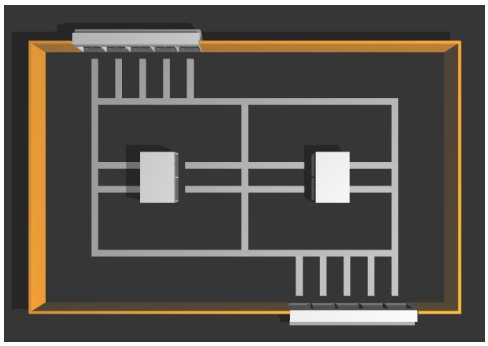

Fig. 2. Ambiente do *Robot@Factory* representado em *SimTwo*

O ambiente de representação é modular, como se vê na Figura 3, tendo uma secção para impressão de valores e controlo usando uma folha de cálculo, outra janela para a criação do ambiente gráfico utilizando *XML - eXtensible Markup Language*, ainda tem uma janela para fazer a componente de controlo dos robôs, programada em Pascal, e por fim tem uma janela de configuração do simulador, configurando a vista a usar, as portas de comunicação, entre outros parâmetros.

Para além disto é dada a possibilidade de comunicação por diversos meios, nomeadamente protocolo Modbus, UDP e porta série virtual.

III. Sistema robótico prototipado

A. Mecânica do sistema

O robô foi projetado e prototipado por forma a ser possível participar no *Robot@Factory* de forma competitiva, não tendo restrições ao nível da topologia de locomoção, sendo apenas limitado no tamanho a $45x40$ cm e 35 cm de altura [15], foi, ainda, adotada uma topologia omnidirecional com três rodas, desfasadas 120^{o} entre elas. Esta escolha deve-se ao facto de não apresentar restrições de movimentos, o que se torna benéfico para a prova e, ao contrário do de quatro rodas, não é necessário um sistema mecânico extra para realizar suspensão.

Nas Figuras 4, 5 e 6 é possível ver o protótipo construído pela equipa por forma a participar no *Robot@Factory*.

B. Cinemática do sistema

Como se pode observar pela configuração do robô omidirecional de 3 rodas, Figura 7, as velocidades V_x, V_y e ω

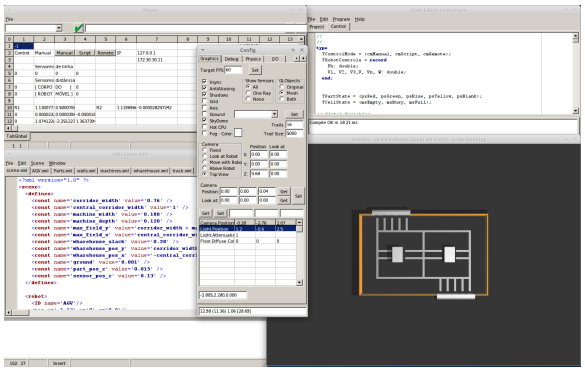

Fig. 3. Ambiente do *Simtwo*

Fig. 4. Vista de cima do robô prototipado

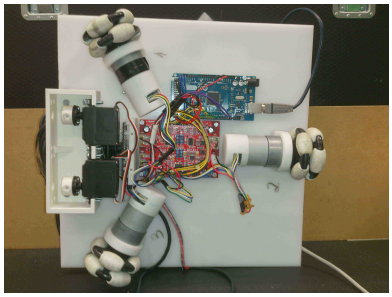

Fig. 5. Vista de baixo do robô prototipado

variam com as velocidades lineares V_1, V_2 e V_3, segundo a equação 1 [17].

$$\begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \quad (1)$$

onde

$$a_{11} = \frac{\sqrt{3} * cos(\theta) + sin(\theta)}{3}$$
$$a_{12} = -\frac{2 * sin(\theta)}{3}$$
$$a_{13} = -\frac{\sqrt{3} * cos(\theta) + sin(\theta)}{3}$$
$$a_{21} = \frac{\sqrt{3} * sin(\theta) - cos(\theta)}{3}$$
$$a_{22} = \frac{2 * cos(\theta)}{3}$$
$$a_{23} = -\frac{\sqrt{3} * sin(\theta) - cos(\theta)}{3}$$
$$a_{31} = a_{32} = a_{33} = -\frac{1}{3 * L}$$

O projeto do controlador de seguimento de segmentos de reta tem como parâmetros de saída os valores de velocidade

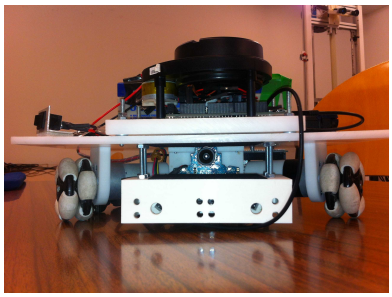

Fig. 6. Vista frontal do robô prototipado

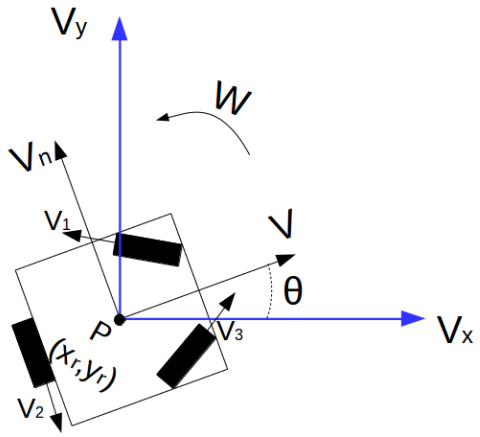

Fig. 7. Geometria de um robô omnidirecional de três rodas

no formato V, V_n e ω. Estas variáveis são obtidas com recurso à equação 2.

$$\begin{bmatrix} V \\ V_n \\ \omega \end{bmatrix} = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} \quad (2)$$

Para atuar os motores calcula-se a velocidade a aplicar a cada roda, (V_1, V_2, V_3), através de (V, V_n, ω), através da equação 3 [18] , em que L é distância das rodas ao eixo de rotação.

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} -sin(\pi/3) & cos(\pi/3) & L \\ 0 & -1 & L \\ sin(\pi/3) & cos(\pi/3) & L \end{bmatrix} \cdot \begin{bmatrix} V \\ V_n \\ \omega \end{bmatrix} \quad (3)$$

A velocidade linear de cada roda, durante um período de amostragem, é determinada a partir do número de transições geradas pelo correspondente *encoder*. Definindo o deslocamento entre cada transição e multiplicando-o pelo número de transições contadas entre amostragens, obtém-se a velocidade angular de cada roda (ω_1, ω_2 e ω_3). Cada uma das velocidades lineares é obtida resolvendo a equação 4.

$$V_i = \omega_i * r_{r_i}, \ i = 1, \ 2, \ 3 \quad (4)$$

sendo r_{r_i} o raio de cada uma das rodas.

C. Hardware do sistema

Por forma a efetuar a localização, foi utilizado um *laser range finder*, proveniente do aspirador robótico *Neato XV-11*, que possui uma resolução angular de aproximadamente 1^o, frequência de aquisição de 5 Hz e um ângulo de leitura de 360^o. Este possui também um alcance de $0.06 - 5$ metros. Foi também utilizada, com o objetivo de identificar o estado das peças a transportar/maquinar, uma câmara de vídeo, a *PlayStation Eye*.

A nível de atuadores foram escolhidos para efetuar a tração do sistema robótico motores DC da *Pololu* de 12 V, com caixa redutora de $30:1$ e velocidade máxima de 350 RPM. Outro dos motivos para a escolha destes motores relaciona-se com o facto de serem providos de *enconders* em quadratura, característica importante pois permite o cálculo da odometria e o controlo em malha fechada do motor. Para a carga e

descarga das peças foi desenvolvido um sistema de duas garras controladas através de dois servomotores standard, *Futaba S3003*, para efetuar a rotação de cada uma.

IV. Sistema de Localização

A. Localização relativa

A estimação da localização relativa do robô foi feita através do cálculo da odometria, recorrendo à integração numérica das equações apresentadas em III-B. A estimativa da posição e orientação do robô pode ser calculada, aplicando-se uma aproximação de primeira ordem, como exemplificado nas equações 5, 6 e 7 [19].

$$x(k) = x(k-1) + V_x T \tag{5}$$

$$y(k) = y(k-1) + V_y T \tag{6}$$

$$\theta(k) = \theta(k-1) + \omega T \tag{7}$$

onde T é o período de amostragem.

B. Localização absoluta

O algoritmo implementado por forma a efetuar o cálculo da localização absoluta do sistema robótico foi o *Perfect Match*, com recurso a dados de um *laser range finder*, sendo um algoritmo com uma abordagem ao problema baseado no erro associado a cada medição, minimizando-o. É uma metodologia caracterizada por ser de elevada eficiência e de requerer baixo poder computacional [20].

1) Mapa: Para ser possível a localização utilizando o algoritmo *Perfect Match* é necessário o robô conhecer o mapa do local por onde se vai movimentar. O mapa do campo da competição *Robot@Factory*, é conhecido, permitindo a representação do mesmo.

O mapa criado para interpretação do robô é uma matriz onde se encontram representadas as paredes e máquinas do campo, com uma resolução de 1 cm.

2) Mapeamento das leituras: Primeiramente é necessário enquadrar cada uma das medições feitas pelo *range finder* no mapa onde o robô se encontra. Seja $s_1 ... s_n$ o vetor de medições, relativas ao robô, proveniente da interface de comunicação com o *laser range finder* (Figura 8), e α o ângulo dessa medição relativa à orientação do robô. É possível então determinar a posição de cada ponto lido, em relação ao sistema robótico no mundo (ξ_{xy}) , segundo a equação [8].

$$P_{l_i} = \xi_{xy} + \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} \cdot s_i \begin{bmatrix} cos(\alpha) \\ sin(\alpha) \end{bmatrix} \tag{8}$$

Obtém-se desta forma as coordenadas x e y, no mundo, de cada medição feita pelo *range finder*.

3) Minimização do erro: Como já referido, a estimação da posição do robô, feita segundo o *Perfect Match*, baseia-se na minimização do erro entre o mapa do campo e o mapeamento feito. Recorrendo a um mapa de distâncias, é possível definir o erro para cada um dos pontos lidos.

Um mapa de distâncias é uma matriz que contém em cada célula um valor numérico representativo da distância ao objeto mais próximo. Por forma a obter o máximo de precisão nos valores calculados, recorreu-se a uma máscara $3x3$, como visto na matriz a seguir representada, que faz o varrimento por todo o mapa. No final do varrimento, todos os valores da matriz são divididos por 2.

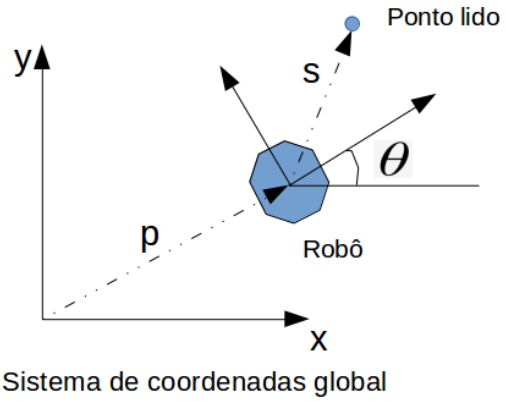

Fig. 8. Representação de robô em referencial do mundo e de ponto lido relativo a robô

$$\begin{bmatrix} 3 & 2 & 3 \\ 2 & 0 & 2 \\ 3 & 2 & 3 \end{bmatrix} \tag{9}$$

Segundo o teorema de Pitágoras, a distância entre um ponto em $P_{l-1,c-1}$ e $P_{l,c}$ será $\sqrt{1^2 + 1^2} = 1.4$, no entanto aplicando a metodologia previamente referida este valor será igual a 1.5, tendo apenas um erro de 1 mm, considerando-se ser uma boa aproximação. O resultado da aplicação desta máscara ao mapa da Figura 9 encontra-se representado na Figura 10.

Em seguida é necessário sobrepor as duas matrizes e avaliar o erro de cada leitura. Considera-se como o erro da leitura o valor presente na célula da matriz de distâncias na qual o ponto P_{l_i} pousar.

A função do erro é determinar o quão desalinhados estão os dois mapas. Quanto maior o erro, menos coincidentes estes são. Segundo [20], a melhor função para avaliar este tipo de erros é apresentada na equação 10.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0														0
0														0
0						0	0	0						0
0						0	0	0						0
0						0	0	0						0
0														0
0														0
0														0
0						0	0	0						0
0						0	0	0						0
0						0	0	0						0
0														0
0														0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 9. Exemplo de mapa

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	1	2	2	2	1.5	1	1	1	1.5	2	2	2	1	0
0	1	2	3	2	1	0	0	0	1	2	3	2	1	0
0	1	2	3	2	1	0	0	0	1	2	3	2	1	0
0	1	2	3	2	1	0	0	0	1	2	3	2	1	0
0	1	2	3	2.5	1.5	1	1	1	1.5	2.5	3	2	1	0
0	1	2	3	3	2.5	2	2	2	2.5	3	3	2	1	0
0	1	2	3	2.5	1.5	1	1	1	1.5	2.5	3	2	1	0
0	1	2	3	2	1	0	0	0	1	2	3	2	1	0
0	1	2	3	2	1	0	0	0	1	2	3	2	1	0
0	1	2	3	2	1	0	0	0	1	2	3	2	1	0
0	1	2	2	2	1.5	1	1	1	1.5	2	2	2	1	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 10. Exemplo de mapa de distâncias

$$erro = 1 - \frac{c^2}{c^2 + e^2} \tag{10}$$

Esta função de erro tem como vantagem em relação à do erro quadrático médio o facto de ser mais robusta aquando da ocorrência de medidas discrepantes. Desta forma, se um determinado erro de medição for muito elevado, não vai afetar de forma significativa a correção efetuada.

Por forma a minimizar o erro calculado, como sugerido por [20], recorreu-se ao algoritmo *resilient backpropagation* (RPROP) [21] [22]. Este algoritmo tem em conta apenas o sinal da derivada parcial sobre todos os pontos, e atua de forma independente sobre o erro de cada ponto.

Dado que a estimação da posição do robô é feita tanto para x, y e θ, é necessário calcular o gradiente para cada uma destas variáveis. O sinal do gradiente calculado tem como função indicar a direção da atualização do erro.

Primeiramente é necessário definir um valor de atualização, Δ_{ij}, que define o valor do deslocamento que o erro vai ter, consoante o sinal do somatório dos gradientes segundo determinada direção, o valor da correção segundo essa mesma direção vai variar segundo 11.

$$\Delta w_{ij} = \begin{cases} -\Delta_{ij}, & se \text{ somatório } de\ gradiente > 0 \\ \Delta_{ij}, & se \text{ somatório } de\ gradiente < 0 \\ 0, & se\ outros\ valores \end{cases} \tag{11}$$

Posteriormente atualiza-se Δ_{ij}, consoante o sinal do gradiente se mantenha constante ou mude. A mudança do sinal do gradiente indica que a última correção foi demasiado grande, levando o algoritmo a passar pelo objeto. Caso isto se verifique, o valor de atualização é multiplicado por uma constante de valor inferior a 1, η^-, de forma a diminuir o avanço e se conseguir aproximar do mínimo local. Contrariamente, se o valor do gradiente mantiver o seu sinal, Δ_{ij} é multiplicado por uma constante superior a 1, η^+, de forma a ocorrer uma convergência mais rápida. No caso de mudança de sinal do gradiente, [21] [22] sugerem que não haja atualização de Δw_{ij}. Isto é conseguido igualando o valor do gradiente anterior a zero.

Os valores de Δw são somados e divididos pelo número total de pontos avaliados, obtendo desta forma a média do deslocamento que o robô terá que efetuar segundo x, y e θ. Este processo é repetido até se verificarem critérios de paragem previamente estipulados. O número de interações do algoritmo RPROP, o erro total e a variação do erro total foram os critérios adotados.

Por forma a simplificar a implementação do algoritmo, foram efetuadas umas mudanças ao mesmo. Estas alterações estão relacionadas com a atualização do peso na célula a ser analisada, onde $\Delta w_{ij}(t)$ passou a ser o salto que o ponto dá segundo a coordenada a ser analisada. Desta forma, se o ponto analisado se encontrar a uma elevada distância do mínimo local, este é transposto $\Delta w_{ij}(t)$ células na sua direção.

V. Fusão sensorial

Para a realização da fusão sensorial foi utilizado o Filtro de Kalman Estendido. Este algoritmo é constituído por dois passos, a previsão e a correção, sendo que a sua equação de estado é representada por:

$$\frac{dX(t)}{dt} = f(X(t),\ u(t),\ t) \tag{12}$$

onde $u(t)$ são os parâmetros de entrada que, neste caso particular, é composto pelas velocidades lineares do ponto de contacto com a superfície de cada roda [23].

A. Previsão

1) Estimação de estado: A estimação do estado no instante t_k requer o conhecimento do estado em t_{k-1} e é feita por integração numérica, como demonstrado nas equações 5, 6 e 7.

2) Propagação da covariância: Por forma a calcular a propagação da covariância, é necessário ter definida as equações que definem a transição do estado. Este sistema de equações, definido na equação 1, deve ser linearizado em torno de $X(t) = X(t_k)$, $u(t) = u(t_k)$ e $t = t_k$, resultando em:

$$A^*(t_k) = \begin{bmatrix} 0 & 0 & b_{13} \\ 0 & 0 & b_{23} \\ 0 & 0 & 0 \end{bmatrix} \tag{13}$$

Onde:

$b_{13} = \frac{\sqrt{3}*cos(\theta)+sin(\theta)}{3} V_1 - \frac{2*sin(\theta)}{3} V_2 - \frac{\sqrt{3}*cos(\theta)-sin(\theta)}{3} V_3$

$b_{23} = \frac{\sqrt{3}*sin(\theta)+cos(\theta)}{3} V_1 + \frac{2*cos(\theta)}{3} V_2 - \frac{\sqrt{3}*sin(\theta)+cos(\theta)}{3} V_3$

A matriz transição de estado (ϕ) é então a apresentada na equação 14:

$$\phi^*(t_k) = \begin{bmatrix} 1 & 0 & b_{13} \cdot T \\ 0 & 1 & b_{23} \cdot T \\ 0 & 0 & 1 \end{bmatrix} \tag{14}$$

A propagação da covariância, $P(t_k^-)$, é calculada através da equação 15,

$$P(t_k^-) = \phi^*(t_k) P(t_{k-1}) \phi^*(t_k)^T + Q(t_k) \tag{15}$$

onde $Q(t_k)$ define a covariância do erro. Esta matriz estipula o rigor das medições feitas pela odometria.

B. Correção

1) Ganho do Filtro de Kalman: O ganho do Filtro de *Kalman* tem como objetivo pesar quais os valores mais confiáveis, entre as diversas fontes de medidas. Este é calculado pela equação que se segue:

$$K(t_k) = P(t_k^-)H(t_k)^T[H(t_k)P(t_k^-)H(t_k)^T + R(t_k)]^{-1} \quad (16)$$

Onde $R(t_k)$ representa a covariância do erro das medições. Esta matriz, à semelhança de $Q(t_k)$, define o rigor das medições feitas, neste caso das do *laser range finder*.

Como as medidas efetuadas pelo *laser range finder* são processadas fora do filtro de *Kalman* estendido, isto é, as medições que entram nos parâmetros do filtro são coincidentes com o referencial do mundo, outro aspeto a salientar é o facto da matriz $H(t_k)$ ser a identidade. Desta forma, a equação que define o ganho do filtro é reduzida à equação 17.

$$K(t_k) = P(t_k^-)[P(t_k^-) + R(t_k)]^{-1} \quad (17)$$

2) Atualização da estimação de estado: A atualização do estado é feita recorrendo a equação 18.

$$X(t_k) = X(t_k^-) + K(t_k) * [z(t_k) - X(t_k^-)] \quad (18)$$

Onde $z(t_k)$ representa o estado calculado obtido através das medições do *laser range finder*.

3) Atualização da covariância: O último passo do cálculo do Filtro de *Kalman* Estendido é a atualização da covariância, necessário para as iterações seguintes do algoritmo.

$$P(t_k) = [I - K(t_k)H(t_k)]P(t_{k-1}) \quad (19)$$

Como já verificado, a matriz $H(t_k)$ é a matriz identidade, simplificando 19 a 20.

$$P(t_k) = [I - K(t_k)]P(t_{k-1}) \quad (20)$$

VI. Planeamento de trajetórias

Após se efetuar a localização podem-se planear as trajetórias a efetuar pelo robô para se deslocar entre dois pontos. Para o planeamento existe uma elevada panóplia de métodos passíveis de utilização, como seguimento de linha [24] [25] e deslocamento entre *waypoints*, até métodos de complexidade mais elevada que permitam o cálculo de percursos que não sejam restritivos planeando trajetórias, como algoritmos *bug*, de *roadmap* e de decomposição em células que se encontram expostos em [26].

A. Expansão do mapa

Antes de expandir o mapa é necessário fazer a representação do ambiente. Assim para o mapeamento do espaço recorre-se à utilização de ferramentas de geração de *Bitmap*, atribuindo diferentes códigos de cor consoante a ocupação das áreas, verde equivale a livre, vermelho a ocupado e amarelo a próximo de obstáculo.

Assim, o mapa após construido é expandido, considerando o robô como um círculo de raio (R) igual à maior distância do centro à extremidade, e de seguida considerar todas as posições (X e Y) que se encontrem a uma distância inferior a R como locais com o valor ocupado, desta forma ao efetuar a expansão garante-se que nunca haverá contacto independentemente da orientação que o agente tome.

Este método consiste em três fases, primeiro converter o *Bitmap* em matriz, de seguida calcular a matriz de distância, por forma a obter a distância do robô aos obstáculos, e finalmente atribuir um estado às posições no mapa consoante as distâncias.

Inicialmente percorre-se o mapa construido em *Bitmap* e é criada uma matriz na qual os obstáculos são representados como 1 e o espaço livre como 0.

Para determinar a distância aos obstáculos é aplicada a transformada introduzida por *Borgefors* em 1984 [27], que consiste em realizar dois varrimentos, representados na Figura 11, um da esquerda para a direita e de cima para baixo e outro da direita para a esquerda e de baixo para cima nos quais se aplica uma transformada de distâncias, relativamente a pontos com o valor de obstáculo (1), sendo que no primeiro varrimento tudo o que não for obstáculo terá um peso infinito.

Por fim após obter a matriz de distância, o algoritmo percorre a matriz e, consoante o valor encontrado, define o estado nessa posição, ou seja se a distância for inferior ao raio do robô é tida como ocupada, caso esteja no intervalo de $]Raio + n * Tamanho\ célula, Raio + (n+1) * Tamanho\ célula]$, com $n = 0, 1, 2$, insere uma camada protetora que provoca um custo acrescido no planeamento do A^*, para garantir que a trajetória é segura e que só entra nessa área caso seja proveitoso. Estes valores são alterados na matriz que representa o mapa para depois poderem ser usados como custos adicionais (K_e) para a pesquisa.

1) Alteração para múltiplos robôs: As alterações passam primeiramente por iniciar uma comunicação onde se transmitem as posições dos outros robôs, para saber onde o inserir no mapa, de seguida calcula-se a distância entre eles, caso esta seja inferior a um *threshold* predefinido considera-se obstáculo, caso contrário não é tido em conta, já que se encontra demasiado afastado. O passo seguinte é de inserir um círculo com o raio do segundo robô no *Bitmap* semelhante ao obtido após a construção do mapa e, por fim proceder à expansão, usando o método descrito anteriormente.

B. Algoritmo de pesquisa A^*

O algoritmo A^* é designado como o melhor primeiro, [28], e consiste na pesquisa num grafo do caminho mais curto que une os nós inicial e final, obtendo o percurso ótimo. Uma vez que este se trata de um método informado

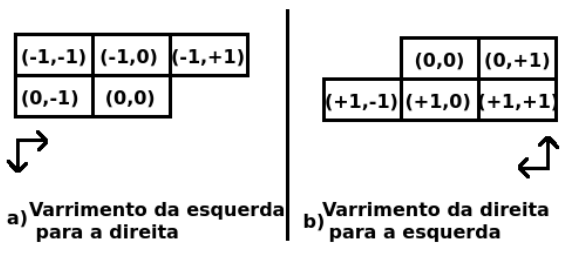

Fig. 11. Transformadas dos varrimentos

utiliza uma função heurística, $f(n)$, para estimar o caminho com o menor custo estimado desde o nó n até ao final e é obtido através da soma de duas funções $g(n)$ e $h(n)$, isto é $f(n) = g(n) + h(n)$. A função $g(n)$ representa o custo desde o nó inicial até ao nó atual (n), podendo ou não ser heurística, e a $h(n)$ é a estimativa, através de uma heurística, do custo do percurso desde n até ao nó final, devendo esta ser subestimada, podendo-se ver a representação na Figura 12.

O algoritmo utilizado consistiu numa implementação semelhante à definida por *Costa et al.* [29], onde após a representação dos nós, sofreu modificações para analisar os nós vizinhos, na estrutura usada para a definição da lista aberta e na heurística usada. Outra das alterações a realizar é retirar a condição de revisitar nós da lista fechada para verificar se sofrem alterações, pois reduz o número de visitas aos nós sem afetar a admissibilidade, pois, utilizando uma heurística consistente, a estimativa será sempre menor que a estimativa do nó adjacente mais o custo de se movimentar de um nó para o seguinte.

1) Construção dos nós vizinhos: Para a construção dos nós vizinhos foi realizada uma pesquisa com conetividade 8 em torno do n_{melhor}, nó com menor $f(n)$. Ao visitar cada nó atualizam-se os valores da posição, (X, Y), o custo de ir do n_{melhor} para lá, $c(n_{melhor}, n)$, o apontador do nó pai e se está livre ou não. Para os que se encontram numa posição vertical ou horizontal o $c(n_{melhor}, n)$ será obtido através da equação 21 e para os da diagonal o custo será dado por 22, em que $Sizec$ é o tamanho definido para as células e K_e é o custo extra que depende do valor que se encontra no mapa para definir as camadas de proteção ou se está totalmente livre, dependente do código de cores usado no mapa.

$$c(n_{melhor}, n) = Sizec + K_e * Sizec \tag{21}$$

$$c(n_{melhor}, n) = Sizec * \sqrt{2} + K_e * Sizec * \sqrt{2} \tag{22}$$

2) Estrutura utilizada para listas: O algoritmo durante a sua execução despende uma parte considerável de tempo a adicionar valores por ordem crescente de $f(n)$ e a remover o melhor nó das listas aberta e fechada, por isso é necessário escolher uma estrutura de dados com uma complexidade temporal baixa, isto é que seja rápida de executar principalmente para este tipo de atuações. Para tal recorre-se a *binary heaps* que são estruturas com uma complexidade temporal para os métodos de inserção e remoção baixa, O(log n), o que permite trabalhar facilmente com uma quantidade elevada de nós de forma mais eficaz.

3) Heurística: A heurística escolhida para o problema foi a da distância euclidiana, uma vez que permite mover-se em qualquer sentido, ou seja permite calcular a distância em reta e em diagonal. Esta heurística, $h(n)$, é calculada através da equação 23, em que K é um ganho, utilizado por *Costa* em 2011 [26] , que tem como objetivo dar maior peso aos nós que estão mais próximos da meta. D pode ser definido como o custo mínimo de ir de um ponto até ao adjacente, podendo ser também definido como 1. Já n_x e n_y são, respetivamente, as coordenadas em XX e YY do nó atual e d_x e d_y são, respetivamente, as coordenadas em XX e YY do nó final.

$$h(n) = K * D * \sqrt{(n_x - d_x)^2 + (n_y - d_y)^2} \tag{23}$$

C. Sistema de Navegação

Para efetuar as trajetórias é necessário definir um sistema de navegação que efetue o controlo de movimentos e, de seguida, um método para o controlo de ações que alterne entre modos de funcionamento normal com o planeamento criado e com funcionamento especial.

1) Controlo de movimento: Para executar as trajetórias o robô necessita de ter um controlador que indique as velocidades (V, V_n, ω) que são necessárias para o seu movimento seguir o pedido pelo planeamento efetuado. Uma vez que os movimentos realizados vão ser de dimensões reduzidas, as manobras podem ser aproximadas por linhas retas mantendo na mesma uma certa suavidade nos movimentos, que se aproximam de curvas.

Para o controlador de seguimento de linhas foi implementado um método baseado no apresentado por *Conceição et al.* em 2006 [30] . Este método define os vetores de velocidade através da posição do robô e de um segmento de reta que une o ponto inicial ao final. Efetuando uma mudança de referencial para facilitar a implementação, passando o segmento a estar alinhado com o eixo XX e o ponto final passar a ser o (0,0), tal como representado na Figura 13, em que X_{act} é a distância ao ponto final e Y_{act} é a distância do robô ao segmento de reta.

2) Controlo de ações: Uma vez que é utilizado um método de expansão do mapa que considera todo o comprimento do robô, incluindo o sistema de empilhamento de peças, existem situações em que é impossível calcular as trajetórias recorrendo ao método de planeamento sugerido devido a que a posição do posto de trabalho se encontra numa zona ocupada. Assim foi implementada uma infraestrutura constituída por máquinas de estado hierárquicas que permitem decompor os percursos para funcionar em três modos

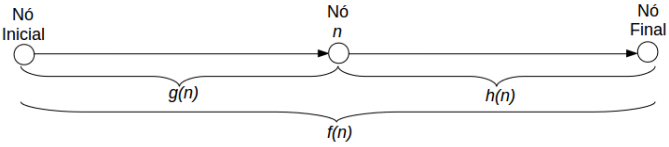

Fig. 12. Funções $f(n)$,$g(n)$ e $h(n)$

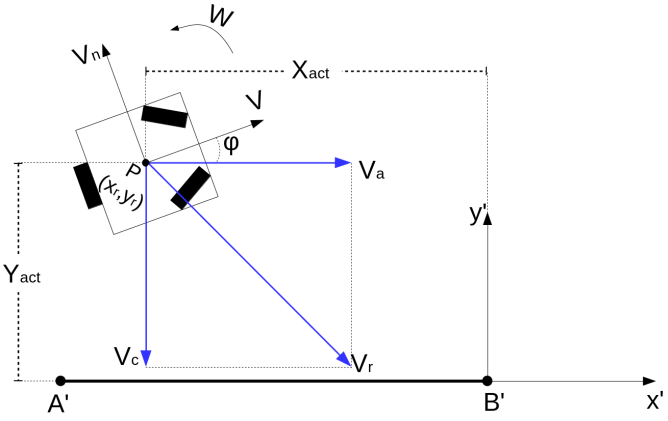

Fig. 13. Esquema do controlo após mudança de coordenadas

entrada e saída dos postos de trabalho, planeamento com recurso ao A^* para os percursos interiores e carga e descarga de peças. Em que o primeiro modo realiza uma reta entre dois *waypoints* de uma forma pré-definida, o segundo modo calcula o percurso a efetuar e realiza um seguimento das retas que efetuam a união entre o nós, usando o método de controlo de movimento, e o último realiza um controlo de posição, estabilizando na posição em que as garras se encontram dentro da máquina, capturando ou libertando a peça.

3) Controlo de ações com múltiplos robôs: No caso de existirem múltiplos robôs o primeiro passo a tomar é verificar se o ponto final, que é o ponto da máquina na zona marcada como livre no mapa, está livre ou se existe lá algum obstáculo, nomeadamente outro robô. Caso não exista nenhuma impossibilidade o ponto definido como meta será mantido, senão, se porventura houver um obstáculo nessa localização, calcula-se o ponto mais próximo do fina e o robô desloca-se para lá até o destino estar disponível.

VII. Resultados

A. Localização

Nesta secção são apresentados e debatidos os resultados provenientes dos testes efetuado no robô prototipado. O teste de validação será segundo uma trajetória que tem como objetivo o movimento do robô por várias partes do campo, validando a localização em toda a extensão do campo.

Esta trajetória, de 3.2 metros, encontra-se representada na Figura 14, sendo uma sequência de retas com a seguinte ordem: A - B - C - D - E.

Neste teste, o robô começa na posição $(-1.13, -0.5, 90^o)$ (em coordenadas no formato (x, y, θ)) e dirige-se para a posição $(1.13, -0.5, 0^o)$, definida como o ponto B. Ao chegar perto de B, este começa a dirigir-se para o ponto C, caracterizado pelas coordenadas $(1.13, 0.5, 90^o)$ e de seguida dirige-se até $(0, 0.5, 270^o)$ (ponto D). Para terminar, segue para o ponto E, nas coordenadas $(0, -0.5, 180^o)$, onde estabiliza.

Ao longo de vários testes, o valor para as matrizes R e Q que se verificou ser mais eficaz para a estimação do estado foi $R_{11} = R_{22} = R_{33} = 100$ e $Q_{11} = Q_{22} = 1000$ e $Q_{33} = 100000$ (*teste 2C*).

A notória ondulação ao longo do movimento do robô deve-se às pequenas correções da posição do robô para a trajetória estipulada, induzidos pelo controlo dos motores e de erros inerentes à localização. Uma comparação entre a localização obtida por fusão sensorial e à calculada pelo sistema baseado no *laser range finder* é visível na Figura 18.

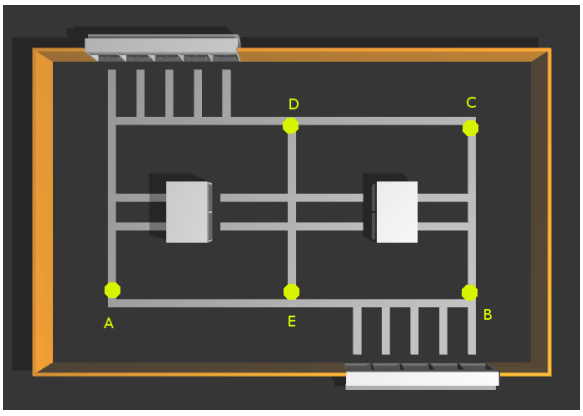

Fig. 14. Pontos de referência da trajetória para teste e validação de algoritmos

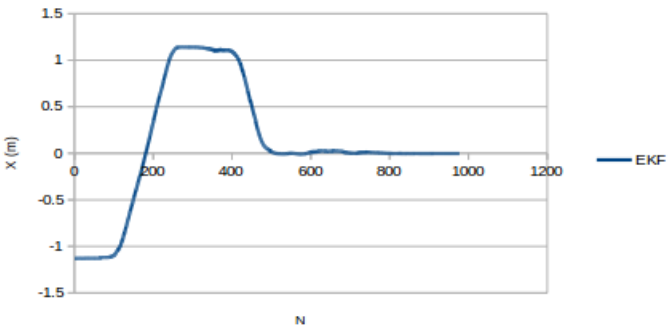

Fig. 15. Evolução da coordenada x ao longo das iterações N

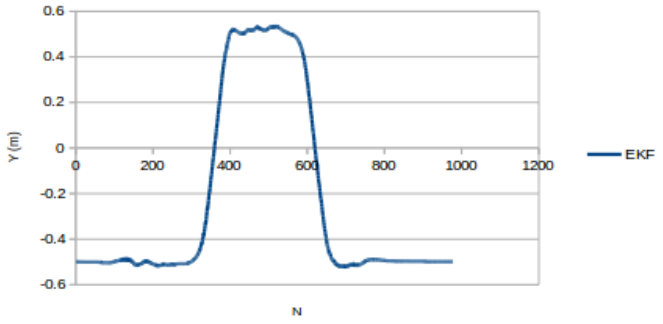

Fig. 16. Evolução da coordenada y ao longo das iterações N

B. Trajetórias

Nesta secção são apresentados os resultados obtidos com o planeamento de trajetórias desenvolvido e pelo sistema de navegação utilizado.

Desta forma, inicialmente ao expandir-se obtém-se um mapa do campo similar ao representado na Figura 19, o qual depende em parte do tamanho das células a usar, as quais quanto maior forem menor será a resolução.

Já no caso de existir outro robô no mapa, nomeadamente na posição (X,Y)=(0,0) é adicionado mais um elemento ao mapa como se pode observar na Figura 20.

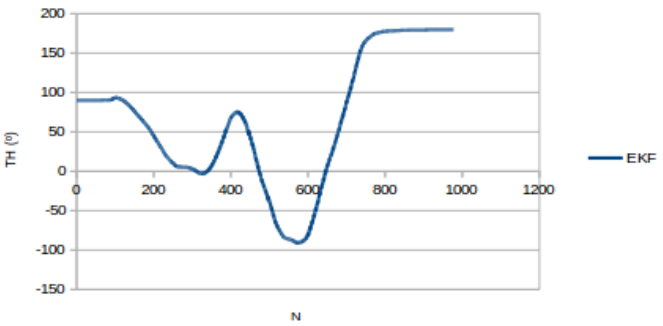

Fig. 17. Evolução da coordenada θ ao longo das iterações N

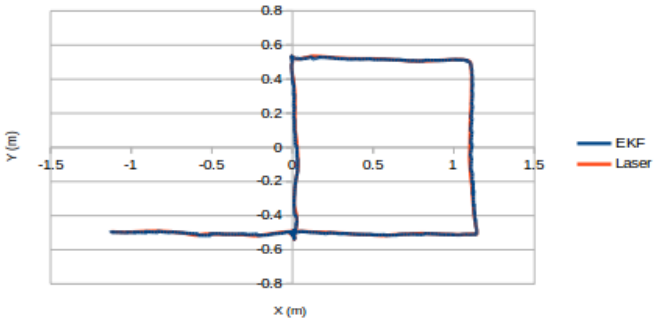

Fig. 18. EKF: Comparação da trajetória para teste de validação de algoritmos com localização baseada em *laser range finder* - teste 2C

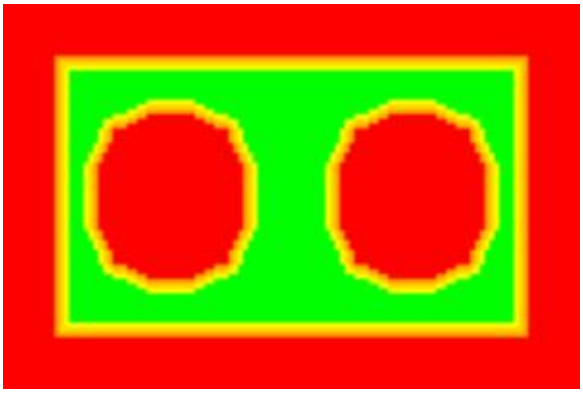

Fig. 19. Representação do mapa expandido

O variar o tamanho das células apresenta vantagens relativamente ao tempo de processamento, diminui com o aumento das células, e ainda as camadas de proteção ideais variam, sendo cada vez menos necessárias. Contudo uma resolução baixa implica a perda do percurso ótimo, a deformação dos objetos e, ainda, a possível perda de percursos viáveis.

Após expandir o mapa aplica-se o algoritmo A^* para gerar a trajetória. Os resultados obtidos diminuem o tempo de processamento quanto maior for o tamanho da célula usado, pois menos nós a ser pesquisados serão gerados. Contudo a variação do valor de K na heurística faz com que se perca o resultado ótimo, uma vez que se pode sobrestimar a distância, mas apresenta a vantagem de reduzir drasticamente os tempos de processamento, com um peso mínimo no comprimento da trajetória, tal como se vê na Figura 21 e

Fig. 20. Representação do mapa expandido com dois robôs

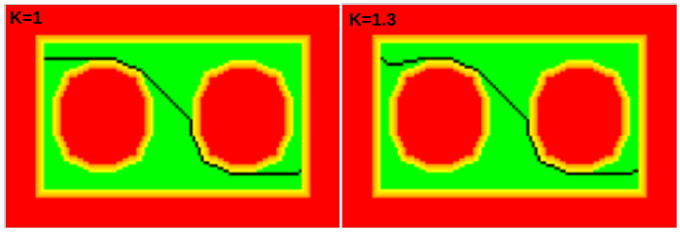

Fig. 21. Planeamento de trajetórias com dois valores de K

TABLE I
EFEITOS CAUSADOS PELA VARIAÇÃO DO VALOR K NO ALGORITMO A^*

	Comprimento da Trajetória (cm)	**Tempo de Processamento** (ms)
$K = 1$	288.6	30
$K = 1.3$	292.8	4
$K = 1.7$	299.0	1
$K = 1.9$	301.1	1

representados as variações na tabela I.

Caso exista outro robô em campo, a trajetória calculada será a representada na Figura 22 dependendo do ganho da heurística.

Após se executar o sistema de navegação, juntado todos os modos de funcionamento, obtém-se o caminho que une os armazéns e que permite desde o deslocamento entre máquinas até à entrada e saída nos postos de trabalho.

Os resultados obtidos no simulador com um e dois robôs podem ser consultados nas Figuras 23 e 24, respetivamente.

VIII. CONCLUSÃO

Neste artigo foram apresentados métodos de controlo, localização e navegação para um robô omnidirecional num ambiente industrial. A abordagem adotada teve em consideração a minimização da alteração do meio envolvente bem como um sistema de planeamento de trajetórias dinâmico e não restritivo que visa encontrar o caminho mais curto entre dois pontos sem comprometer a segurança do AGV. A abordagem foi implementada e testada num ambiente simulado e num robô real, inserida na competição *Robot@Factory*, tendo-se comprovado a viabilidade dos algoritmos e do protótipo desenvolvido. Pretende-se que a abordagem proposta sirva de referência para as equipas participantes na competição e para potenciais novas equipas.

As equipas participantes nas diversas edições desta competição contribuiram de uma forma positiva para este desafio, tornando claro que a sua configuração foi bem escolhida, sendo possível a utilização de protótipos com abordagens simples (do ponto de vista tecnológico e científico), sem

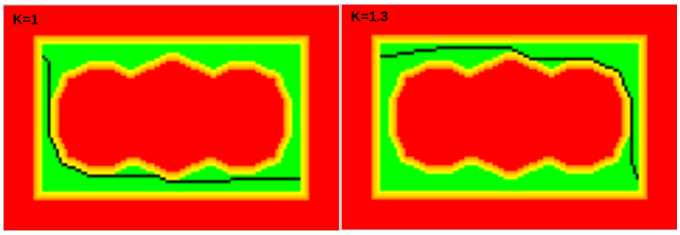

Fig. 22. Planeamento de trajetórias com dois robôs, variando valores de K

Fig. 23. Navegação de um armazém para o outro

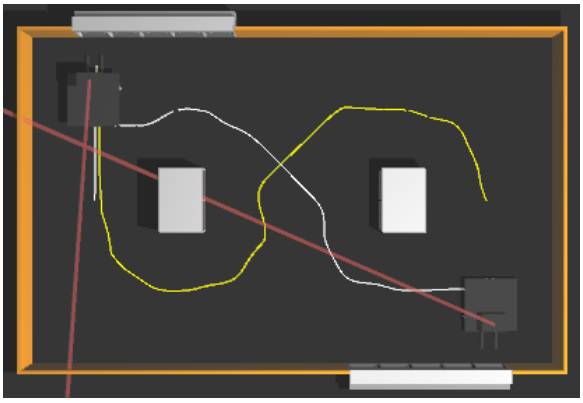

Fig. 24. Navegação de um armazém para o outro com dois robôs

comprometer a utilização de abordagens mais complexas, as quais permitem melhores desempenhos.

Referências

[1] Pinto, M., Moreira, A. P., & Matos, A. (2012). Localization of mobile robots using an extended kalman filter in a LEGO NXT. *IEEE Transactions on Education, 55*, 135–144. doi:10.1109/TE.2011.2155066

[2] Yu, J., Liu, K. H., & Luo, P. (2011). A mobile RFID localization algorithm based on instantaneous frequency estimation. *2011 6th International Conference on Computer Science and Education (ICCSE),* 525–530. doi:10.1109/ICCSE.2011.6028694

[3] Ji, C., Wang, H., & Sun, Q. (2010). Improved particle filter algorithm for robot localization. *2nd International Conference on Education Technology and Computer (ICETC), 4*, 171–174. doi:doi: 10.1109/ICETC.2010.5529710

[4] Lee, S. C., Choi, J. S., & Lee, D. H. (2012). Trilateration based multi-robot localization under anchor-less outdoor environment. In *ICCSE 2012 - Proceedings of 2012 7th International Conference on Computer Science and Education* (pp. 958–961). doi:10.1109/ICCSE.2012.6295224

[5] Choi, B.-S. C. B.-S., & Lee, J.-J. L. J.-J. (2007). The Position Estimation of Mobile Robot Under Dynamic Environment. *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*. doi:10.1109/IECON.2007.4460261

[6] Prestes, E. (2007). Monte Carlo Localization driven by BVP. *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society,* 2724–2729. doi:10.1109/IECON.2007.4460326

[7] Lima, J., Gonçalves, J., & Costa, P. J. (2015). Modeling of a Low Cost Laser Scanner Sensor. In A. P. Moreira, A. Matos, & G. Veiga (Eds.), *CONTROLO'2014 – Proceedings of the 11th Portuguese Conference on Automatic Control* (pp. 697–705). Springer International Publishing. doi:10.1007/978-3-319-10380-8_67

[8] Gonçalves, J., Lima, J., & Costa, P. J. (2015). DC Motors Modeling Resorting to a Simple Setup and Estimation Procedure. In A. P. Moreira, A. Matos, & G. Veiga (Eds.), *CONTROLO'2014 – Proceedings of the 11th Portuguese Conference on Automatic Control* (pp. 441–447). Springer International Publishing. doi:10.1007/978-3-319-10380-8_42

[9] Randria, I., Khelifa, M. M. Ben, Bouchouicha, M., & Abellard, P. (2007). A Comparative Study of Six Basic Approaches for Path Planning Towards an Autonomous Navigation. *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society.* doi:10.1109/IECON.2007.4460164

[10] Tsai, C.-C., Huang, H.-C., & Chan, C.-K. (2011). Parallel Elite Genetic Algorithm and Its Application to Global Path Planning for Autonomous Robot Navigation. *IEEE Transactions on Industrial Electronics, 58,* 4813–4821. doi:10.1109/TIE.2011.2109332

[11] Hui, N. B. (2010). Coordinated motion planning of multiple mobile robots using potential field method. *2010 International Conference on Industrial Electronics, Control and Robotics,* 6–11. doi:10.1109/IECR.2010.5720131

[12] Francis, S. L. X., Anavatti, S. G., & Garratt, M. (2011). Online incremental and heuristic path planning for Autonomous Ground Vehicle. In *IECON Proceedings (Industrial Electronics Conference)* (pp. 233–239). doi:10.1109/IECON.2011.6119317

[13] Engin, M., & Engin, D. (2012). Path Planning of Line Follower Robot. *2012 5th European DSP Education and Research Conference (EDERC),* 1–5. doi:doi: 10.1109/EDERC.2012.6532213

[14] Kim, H., & Kim, B. K. (2014). Online Minimum-Energy Trajectory Planning and Control on a Straight-Line Path for Three-Wheeled Omnidirectional Mobile Robots. *IEEE Transactions on Industrial Electronics, 61(9)*, 4771–4779. doi:doi: 10.1109/TIE.2013.2293706

[15] Comissão Especializada para o Festival Nacional de Robótica. (2013). Robot@Factory - Competition rules and technical specifications.

[16] Costa, P. (2014). paco/Wiki—Main/SimTwo. Retrieved from http://paginas.fe.up.pt/∼paco/wiki/index.php?n=Main.SimTwo

[17] Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011). Introduction to Autonomous Mobile Robots. MIT Press.

[18] Oliveira, H. P., Sousa, A. J., Moreira, A. P., & Costa, P. (2009). Modeling and Assessing of Omni-directional Robots with Three and Four Wheels. In *CONTEMPORARY ROBOTICS - Challenges and Solutions* (pp. 109–138).

[19] Gonçalves, J. (2005). *Controlo de robots omnidireccionais*.

[20] Lauer, M., Lange, S., & Riedmiller, M. (2006). Calculating the Perfect Match: An Efficient and Accurate Approach for Robot Self-localization. *Robocup 2005: Robot Soccer World Cup, 4020*, 142–153.

[21] Riedmiller, M. (1994). *Rprop-description and implementation details. Report* (pp. 5–6).

[22] Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: the RPROP algorithm. *IEEE International Conference on Neural Networks.* doi:10.1109/ICNN.1993.298623

[23] Gonçalves, J. (2009). *Modelação e simulação realista de sistemas no domínio da robótica móvel.*

[24] Makrodimitris, M., Nikolakakis, A., & Papadopoulos, E. (2011). Semi-autonomous color line-following educational robots: Design and implementation. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM* (pp. 1052–1057). doi:10.1109/AIM.2011.6027098

[25] Li, Y., Wu, X., Shin, D., Wang, W., Bai, J., He, Q., Luo, F., & Zheng, W. (2012). An Improved Line Following Optimization Algorithm for Mobile Robot. 2012 7th International Conference on Computing and Convergence Technology (ICCCT), 84–87.

[26] Costa, P. L. (2011). *Planeamento Cooperativo de Tarefas e Trajectórias em Múltiplos Robôs*.

[27] Borgefors, G. (1984). Distance transformations in arbitrary dimensions. *Computer Vision, Graphics, and Image Processing, 26*, 270. doi:10.1016/0734-189X(84)90194-4

[28] Russell, S., & Norvig, P. (2009). Artificial Intelligence: A Modern Approach. *Prentice Hall*. doi:10.1017/S0269888900007724

[29] Costa, P., Moreira, A. P., & Costa, P. (2009). Real-time path planning using a modified A* algorithm. In P. J. S. Gonçalves, P. J. D. Torres, & C. M. O. Alves (Eds.), *ROBOTICA 2009 -Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions* (pp. 147–152).

[30] A. S. Conceição, Moreira, A. P., & Costa, P. J. (2006). Controller optimization and modelling of an omni-directional mobile robot. *7th Portuguese Conference on Automatic Control.*

Paulo Costa , Porto, Portugal, 1968, Licenciado em Engenharia electrotécnica e de computadores opção de Informática e Sistemas pela Faculdade de Engenharia da Universidade do Porto (FEUP) em 1991, obteve o Mestrado em Engenharia Electrotécnica e de Computadores pela FEUP em 1995. Obteve o Doutoramento na FEUP na área de Controlo e Robótica, tendo realizado uma tese de dissertação intitulada "Localização em Tempo Real de Múltiplos Robots num Ambiente Dinâmico" em 2000. É professor na Faculdade de Engenharia da Universidade do Porto nas áreas de automação, robótica e sistemas digitais. É investigador sénior no INESC-TEC (Portugal), na unidade de Robótica e Sistemas Inteligentes, sendo os seus principais interesses a modelação, localização, navegação, controlo e projeto de robôs móveis. Tendo participado e organizado, desde 1998, inúmeras competições relacionadas com a robótica, é um dos proponentes da competição robot@factory.

Nuno Moreira , Aveiro, Portugal, 1989. Estudante em Engenharia Eletrotécnica e de Computadores ramo de Automação pela Faculdade de Engenharia da Universidade do Porto, a realizar uma dissertação de mestrado intitulada: "Localização de Robôs Autónomos em Ambiente Industrial". É membro da equipa FeupFactory, participante no festival nacional de robótica na competição Robot@Factory.

Daniel Campos , Porto, Portugal, 1990. Estudante em Engenharia Eletrotécnica e de Computadores ramo de Automação pela Faculdade de Engenharia da Universidade do Porto, a realizar uma dissertação de mestrado intitulada: "Planeamento Simultâneo de Trajetórias para Múltiplos Robôs Autónomos num Ambiente Industrial". É membro da equipa FeupFactory, participante no festival nacional de robótica na competição Robot@Factory.

José Gonçalves , Bragança, Portugal, 1976, Licenciado em Engenharia electrotécnica e de computadores ramo de Automação, produção e electrónica industrial pela Faculdade de Engenharia da Universidade do Porto (FEUP) em 2000, obteve o Mestrado em Engenharia electrotécnica e de computadores pela FEUP em 2005 no ramo de Informática e automação, tendo realizado uma tese de dissertação intitulada: "Controlo de robôs omnidireccionais". Obteve o Doutoramento na FEUP na área de Controlo e Robótica, tendo realizado uma tese de dissertação intitulada "Modelação e simulação realista no domínio da robótica móvel". É professor no Instituto Politécnico de Bragança nas áreas de automação, robótica, sistemas embebidos, eletrónica e instrumentação. É investigador sénior no INESC-TEC (Portugal), na unidade de Robótica e sistemas inteligentes, sendo os seus principais interesses localização, navegação, controlo e prototipagem de robôs móveis. É membro da equipa IPB@Factory, participante no festival nacional de robótica na competição Robot@Factory.

José Lima , Porto, Portugal, 1978. Licenciado em Engenharia Eletrotécnica e de computadores ramo de Automação, produção e Eletrónica Industrial pela Faculdade de Engenharia da Universidade do Porto (FEUP) em 2001, obteve o grau de Mestre em Engenharia Eletrotécnica e de Computadores pela FEUP em 2004 no ramo de Informática e Automação, tendo realizado uma tese de dissertação intitulada: "Sistema de Aquisição de Imagem Via Ethernet e Processamento em Tempo Real". Obteve o grau de Doutor na FEUP na área de Controlo e Robótica, tendo realizado uma tese de dissertação intitulada "Construção de um Modelo Realista e Controlo de um Robô Humanóide". É professor no Instituto Politécnico de Bragança nas áreas de automação, robótica, sistemas embebidos e eletrónica de potência desde 2001. É investigador sénior no INESC-TEC (Portugal), na unidade de Robótica e sistemas inteligentes, sendo os seus principais interesses localização, navegação, controlo e robótica móvel. É responsável pela equipa IPB@Factory, que participa no Festival Nacional de Robótica na competição Robot@Factory.

Pedro Costa , Porto, Portugal, 1973, Licenciado em Engenharia electrotécnica e de computadores ramo de Informática e Sistemas pela Faculdade de Engenharia da Universidade do Porto (FEUP) em 1996, obteve o Mestrado em Engenharia electrotécnica e de computadores pela FEUP em 1999 no ramo Sistemas, tendo realizado uma tese de dissertação intitulada: "Controlo de uma equipa de robots móveis". Obteve o Doutoramento na FEUP na área de Controlo e Robótica, tendo realizado uma tese de dissertação intitulada "Planeamento Cooperativo de tarefas e trajectórias em Múltiplos Robôs". É professor na FEUP nas áreas de robótica e programação. É investigador sénior no INESC-TEC (Portugal), na unidade de Robótica e sistemas inteligentes, sendo os seus principais interesses controlo, planeamento de trajetórias e manipuladores.