Algoritmos de Coordenação para Enxames de Robôs

Leandro Soriano Marcolino e Luiz Chaimowicz

[1]VeRLab - Laboratório de Visão Computacional e Robótica
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, MG – Brasil

{soriano,chaimo}@dcc.ufmg.br

Abstract. *In this paper, we present coordination algorithms that allow a swarm of robots to work in a distributed fashion to navigate. We present algorithms to solve two main problems: (i) Navigate in an environment with unknown obstacles; (ii) Navigate in a more coordinated and efficient fashion, avoiding congestion situations. We present simulation results, including experimental analisys, and results using a group of real robots, showing the viability of the proposed algorithms.*

Resumo. *Neste trabalho, são apresentados algoritmos de coordenação que permitem a um enxame de robôs trabalhar de forma distribuída durante a navegação. São apresentados algoritmos para resolver dois problemas principais: (i) Navegar em um ambiente contendo obstáculos desconhecidos; (ii) Navegar de forma mais coordenada e eficiente, evitando congestionamentos. São apresentados resultados em simulação, incluindo análises experimentais, e resultados utilizando um conjunto de robôs reais, demonstrando a viabilidade das propostas.*

1. Introdução

Grandes grupos de robôs têm recebido muita atenção recentemente. Geralmente chamados de *enxames*, esses sistemas utilizam um grande número de agentes simples para realizar diversas tarefas. Em geral, enxames de robôs devem trabalhar de forma distribuída e usar recursos limitados de processamento e comunicação. Devido a essas características, novos algoritmos para controlar e coordenar esses grandes grupos de robôs têm sido desenvolvidos.

Durante a realização de uma determinada tarefa, um robô deve navegar no ambiente, ou seja, se movimentar de forma a atingir um determinado alvo, enquanto evita colisões com obstáculos e outros robôs. Normalmente, deseja-se que a navegação seja o mais eficiente e o mais confiável possível. Existem dois problemas principais: (i) Como encontrar um caminho viável até o alvo, em um ambiente contendo obstáculos desconhecidos? (ii) Como evitar congestionamentos quando um grande número de robôs se dirige à mesma região do ambiente? O objetivo do trabalho de iniciação científica, portanto, foi desenvolver soluções distribuídas para esses problemas, tornando a navegação do enxame mais suave, robusta e eficiente. As soluções desenvolvidas foram analisadas e avaliadas através de simulações e experimentos reais. Neste artigo é apresentada uma visão geral do trabalho que foi desenvolvido, mais detalhes podem ser encontrados nas referências indicadas.

2. Navegação em Ambientes com Obstáculos

Uma problema importante ao se utilizar grandes grupos de robôs é a navegação. Uma abordagem comum é controlar os robôs de forma descentralizada, misturando a descida do gradiente com forças locais de repulsão. Porém, como no método convencional de campos potenciais [Khatib 1986], a presença de obstáculos e forças locais de repulsão entre os robôs pode prejudicar a convergência devido aos mínimos locais, regiões onde a força resultante aplicada sobre o robô se anula ou o padrão de forças leva a movimentos repetitivos. Alguns trabalhos, como [Hsieh and Kumar 2006], provam a ausência de mínimos locais para tipos específicos de ambiente, enquanto outros desenvolvem funções de navegação para ambientes conhecidos, como por exemplo [Pimenta et al. 2005]. Mas esses métodos podem ser difíceis de calcular em tempo real e podem não ser aplicáveis a todos os tipos de ambientes. Outros trabalhos tratam o enxame como uma entidade mais simples, ou utilizam hierarquias, para trabalhar com um menor número de graus de liberdade ([Kloetzer and Belta 2006]). Neste trabalho, ao invés de restringir o ambiente ou desenvolver controladores e funções de navegação complexas, é utilizada a composição de controladores simples e coordenação descentralizada para superar o mínimo local em ambientes contendo obstáculos desconhecidos.

Foi utilizada uma estratégia de coordenação que permite aos robôs encontrar um caminho viável até o alvo com a ajuda dos outros robôs. Esse algoritmo será chamado Algoritmo Resgate (AR). A idéia básica é que alguns robôs que atingiram o alvo sejam realocados como robôs de resgate. Esses robôs vão refazer o caminho percorrido procurando por robôs que possam estar presos em mínimos locais. Será apresentada aqui apenas uma idéia geral do algoritmo desenvolvido. Mais detalhes podem ser encontrados em [Marcolino and Chaimowicz 2008c].

Os robôs do enxame podem estar em um de cinco diferentes estados durante a execução da tarefa: *normal*, *preso*, *resgate*, *anexado* e *completo*. Todos os robôs começam no estado *normal*. Se eles caírem em uma região de mínimo local, eles mudam o estado para *preso*. Quando um robô chega no alvo ele pode se tornar um robô de *resgate*. Basicamente, enquanto se move em direção ao alvo, um robô salva uma seqüência de pontos que é usada para marcar o seu caminho. Se ele se tornar um robô de *resgate*, irá refazer o caminho percorrido de trás para frente, procurando por robôs no estado *preso*. Após percorrer o caminho ao contrário, o robô move-se novamente para o alvo seguindo o caminho na direção correta. Quando um robô de *resgate* detecta um robô *preso* em sua vizinhança, realiza um *broadcast* de sua posição corrente e do seu caminho. Qualquer robô *preso* que esteja a uma certa distância do robô de *resgate* e que possua uma linha de visão direta com ele receberá o *broadcast*. Após recebê-lo, o robô *preso* muda o seu estado para *anexado*. Um robô *anexado* irá mover para a posição recebida e depois seguirá o caminho até o alvo. Um robô *anexado* também pode se comunicar com outros robôs no estado *preso*, espalhando a informação sobre o caminho possível até o alvo. Nessa situação, os robôs no estado *preso* mudam o seu estado para *anexado* e vão também poder transmitir a informação a seus vizinhos, criando uma poderosa corrente de comunicação. Finalmente, um robô irá trocar seu estado para *completo* quando atingir o alvo.

Foram realizados diversos testes, tanto em simulação quanto com robôs reais [Marcolino and Chaimowicz 2008a, Marcolino and Chaimowicz 2008b]. Serão apresentados aqui os principais resultados. Na Figura 1 pode ser vista uma execução em

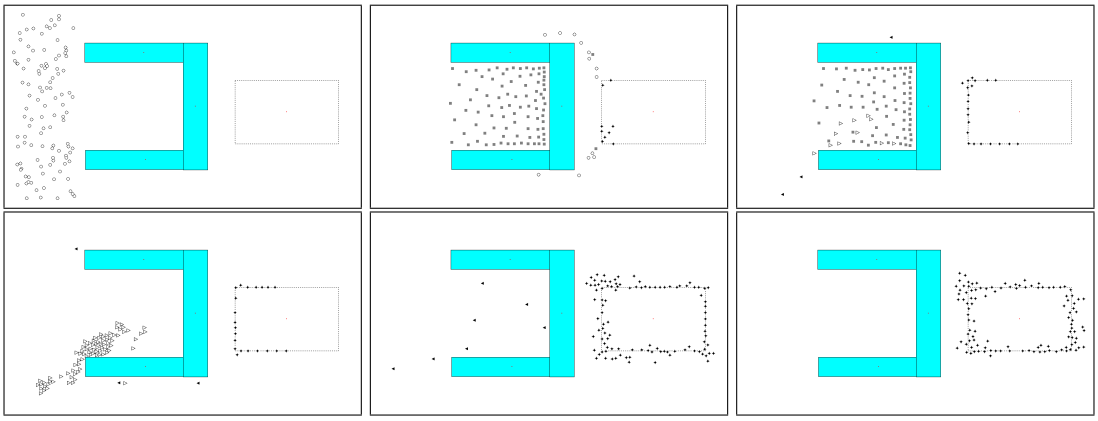


Figura 1. Execução em simulação do AR.

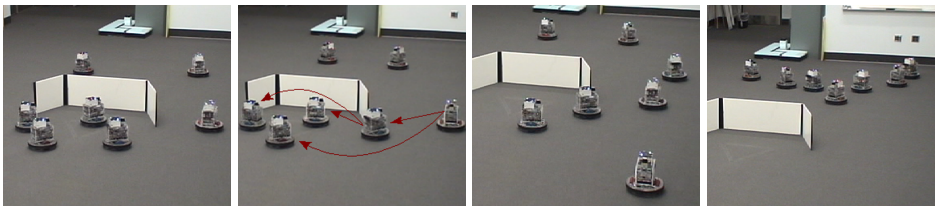
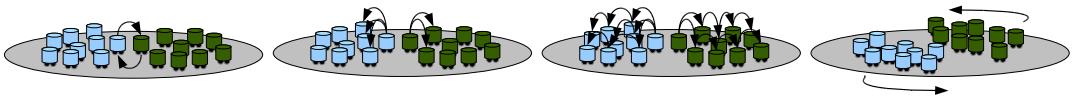


Figura 2. Execução real utilizando o AR.

simulação desse algoritmo, em um cenário clássico de mínimo local: um obstáculo em forma de U formando um beco sem saída. Foram simulados 110 robôs nesse cenário. Os robôs iniciam na esquerda, no meio há um obstáculo e na direita pode ser visto o alvo (quadrado sublinhado). Os estados dos robôs são representados pelos diferentes formatos: *normal* (círculos brancos), *preso* (quadrados cinzas), *anexado* (triângulos brancos apontando para a direita), *resgate* (triângulos pretos apontando para a esquerda), *completo* (diamantes pretos). Como pode ser observado, devido aos robôs de resgate, um grande número de robôs que estavam presos na região de mínimo local receberam um caminho viável até o alvo e foram capazes de convergir de forma apropriada. Também foi realizada uma análise experimental em simulação, onde comparou-se o algoritmo com uma execução convencional, utilizando apenas forças de repulsão. A análise mostrou que, com mais do que cem robôs, a quantidade que não consegue chegar ao alvo tornou-se próxima de 0, enquanto que em uma execução convencional manteve-se alta. A análise foi realizada com um intervalo de confiança de 95%. As simulações foram executadas utilizando o MuRoS, um simulador desenvolvido no VeRLab que permite testar diferentes controladores e algoritmos de coordenação em tempo real.

Realizou-se, então, uma execução real do algoritmo, utilizando sete robôs *scarab*. Os *scarabs* são robôs diferenciais, controlados cinematicamente [Michael et al. 2008]. Os robôs iniciam na parte inferior do cenário e devem convergir para o alvo que está depois do obstáculo em forma de U. Os snapshots da execução podem ser vistos na Figura 2. Como pode ser observado, essa prova de conceito mostra que o algoritmo pode ser utilizado em um grupo de robôs reais, permitindo-os convergir para um alvo em um ambiente contendo obstáculos desconhecidos.



(a) (b) (c) (d)

Figura 3. Passos da execução do algoritmo de coordenação proposto ACG.

3. Evitando Congestionamentos

Outro problema importante que ocorre durante a navegação de um enxame são os congestionamentos, que acontecem quando um grande número de robôs se dirige à mesma região do ambiente no mesmo intervalo de tempo. Esses problemas podem acontecer, por exemplo, quando grupos de robôs navegam em direções opostas ou quando um grande número de robôs se dirige a um mesmo objetivo. Trabalhos sobre controle de tráfego podem ser encontrados tanto na área de robótica cooperativa como na área de sistemas multiagentes. Alguns trabalhos utilizam um agente para gerenciar o tráfego nas interseções onde pode acontecer um congestionamento, como [Dresner and Stone 2005]. Uma abordagem semelhante, na área da robótica, pode ser vista em [Viswanath and Krishna 1997], onde uma rede de sensores é utilizada para coordenar o tráfego dos robôs. Já em [Treuille et al. 2006] é proposto um mecanismo para evitar o congestionamento ao ser simulada uma multidão de pessoas. O método, porém, é muito centralizado para ser utilizado em um enxame de robôs. No presente trabalho, são propostos métodos de coordenação descentralizada que permitem a um enxame de robôs evitar situações de congestionamento. Os algoritmos funcionam sem assumir que os robôs navegam em faixas delimitadas, nem necessitam de meios externos ao enxame, como redes de sensores ou agentes colocados nas interseções.

3.1. Algoritmo para Conflitos de Grupos

Primeiro será apresentado o mecanismo utilizado para o caso em que grupos de robôs se movimentam em direções contrárias, que será chamado Algoritmo para Conflitos de Grupos (ACG). A idéia geral do algoritmo é que os primeiros robôs a perceberem o risco de um congestionamento avisem os outros, permitindo-os atualizar dinamicamente a trajetória de forma a evitar o problema. Mais detalhes sobre esse algoritmo podem ser encontrados em [Marcolino and Chaimowicz 2009a].

Sempre que um robô detecta a presença de outro, ele envia uma mensagem avisando qual é sua direção de destino. Assim, caso as direções de destino sejam diferentes, os robôs são capazes de perceber o risco iminente de um congestionamento. Essa etapa inicial do algoritmo pode ser vista na Figura 3(a). Os robôs que perceberam o risco de um congestionamento enviam uma mensagem aos seus vizinhos, como pode ser visto na Figura 3(b). Cada robô, ao receber essa mensagem, a retransmite para seus respectivos vizinhos, como mostra a Figura 3(c). Dessa forma, a informação do risco de um congestionamento é transmitida pelo enxame e cada grupo poderá desviar de forma apropriada, como mostrado na Figura 3(d). Assim que um robô percebe o risco de um congestionamento, seja encontrando um robô do outro grupo, seja pelo aviso de outros robôs de seu próprio grupo, ele desvia do local onde aconteceria o problema. Para realizar o desvio, o robô utiliza como base a direção de seu destino. Pode-se especificar, por exemplo, que cada robô desviará no sentido anti-horário.

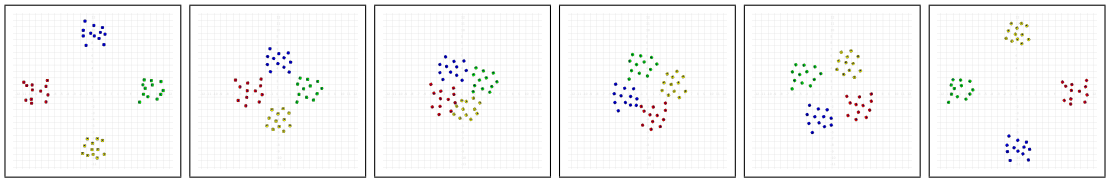


Figura 4. Execução em simulação do algoritmo proposto ACG.

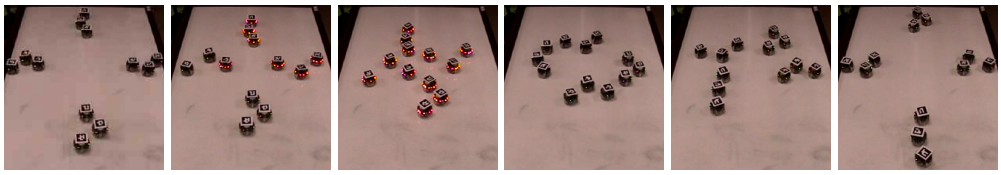


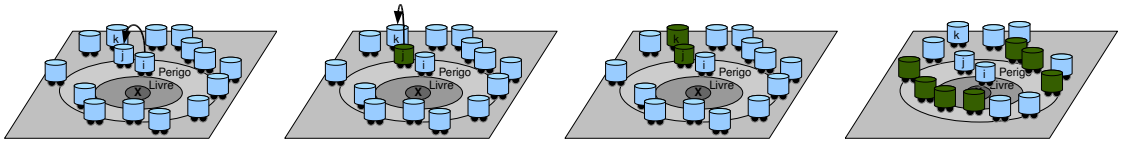
Figura 5. Execução real do algoritmo proposto ACG.

Foram realizados experimentos em simulação e com robôs reais, apresentados em [Marcolino and Chaimowicz 2009a]. Uma execução em simulação desse algoritmo pode ser vista na Figura 4. Quatro grupos de robôs devem navegar em direções opostas no cenário. Como pode ser observado, os grupos circularam em torno da região onde aconteceria o congestionamento, e todos conseguiram chegar de uma forma suave e eficiente no alvo proposto. Também foi realizada uma análise experimental em simulação, onde comparou-se o algoritmo com uma execução convencional, utilizando apenas forças de repulsão. Essa análise foi realizada com um intervalo de confiança de 95% e mostrou um ganho no tempo de convergência de 40%. As simulações foram realizadas utilizando o *Player/Stage*, uma plataforma livre que provê uma interface simples aos sensores e atuadores dos robôs através de uma rede IP.

Foram realizadas diversas execuções reais desse algoritmo, utilizando a plataforma de experimentação com enxames de robôs que vêm sendo desenvolvida no VeRLab. Essa plataforma é composta por doze robôs *e-puck* [Cianci et al. 2007], um arcabouço de programação integrado ao *Player* além de mecanismos de comunicação e localização. Os snapshots de uma execução com quatro grupos podem ser vistos na Figura 5. E-pucks com os leds acesos estão desviando da região de congestionamento. Essa prova de conceito mostra que o algoritmo é viável em um grupo de robôs reais, permitindo-os navegar de uma forma suave e eficiente mesmo quando se movimentam em direções opostas.

3.2. Algoritmo para Conflitos de Alvo

Para lidar com o caso em que um grande número de robôs se dirige para um mesmo objetivo, é utilizado o Algoritmo para Conflitos de Alvo (ACA). A idéia geral do algoritmo é obrigar alguns robôs a esperar enquanto outros se dirigem ao alvo comum. Assim, um número menor de robôs tenta chegar ao alvo no mesmo intervalo de tempo, diminuindo o problema do congestionamento. Mais detalhes sobre esse algoritmo podem ser encontrados em [Marcolino and Chaimowicz 2009b]. A solução é modelada como uma *Máquina de Estados Finitos Probabilística*, onde as arestas são marcadas com probabilidades que definem a transição que será tomada. Os robôs podem estar em um de quatro diferentes estados: *normal*, *esperando*, *preso* e *impaciente*. À partir do estado *esperando*, o robô pode trocar para o estado *impaciente* com uma probabilidade ρ ou pode permanecer no mesmo estado com uma probabilidade $1 - \rho$.



(a) (b) (c) (d)

Figura 6. Passos da execução do algoritmo de coordenação proposto ACA. A cor verde identifica robôs no estado *esperando* ou *preso*.

É definida como *perigo* uma região em torno do alvo. Dentro dessa região, é definida uma região *livre*. A idéia geral é que os robôs que chegam na região *perigo* irão se coordenar de forma que apenas alguns deles entrem na região *livre* no mesmo intervalo de tempo. Ao chegar na região *livre*, os robôs se movimentam diretamente para o alvo. Também é definida uma sub-área na região de sensoriamento do robô como uma área-α. Considerando um sistema de coordenadas centralizado na posição do robô, com o eixo y apontando em direção ao alvo, a área-α é definida como o arco $[-\alpha, \alpha]$ com centro em y e raio δ. Essa área-α será utilizada para detectar outros robôs que possam interferir com a navegação em direção ao alvo.

Caso um robô *normal* esteja na região *perigo* e detecte outro robô em sua área-α com o mesmo alvo, ele irá mudar o seu estado para *esperando* (Figura 6(a)). Um robô *esperando* não irá se movimentar em direção ao alvo. A cada iteração, irá verificar se pode trocar o seu estado. Como mencionado, o robô irá trocar o seu estado para *impaciente* com uma probabilidade ρ e irá manter o seu estado em *esperando* com uma probabilidade $1 - \rho$. Um robô *normal* também pode trocar seu estado para *preso*. Isso acontece quando detecta um robô *esperando* ou *preso* com o mesmo alvo que ele. Dessa vez, essa troca de estado pode acontecer mesmo fora da região *perigo* (Figura 6(b) e (c)). No estado *preso*, o robô se comporta da mesma forma que um robô *esperando*, não irá se mover na direção do alvo. Porém, a transição à partir desse estado não depende de probabilidades. Um robô *preso* irá retornar ao estado *normal* quando não houver mais robôs no estado *esperando* ou *preso* em sua área-α. Finalmente, um robô *impaciente* se movimenta em direção ao alvo, da mesma maneira que um robô *normal*. Porém, um robô *impaciente* não irá mais parar de se movimentar, isto é, não pode trocar seu estado para *esperando* nem *preso*. Essa situação pode ser vista na Figura 6(d), onde o robô j retoma o seu movimento em direção ao alvo no estado *impaciente*. Além disso, o robô k altera o seu estado para *normal* e também volta a se movimentar em direção ao alvo. Pode-se ver na figura que os outros robôs trocaram seus estados para *esperando* ao entrarem na região *perigo* e, portanto, não irão impor dificuldades para o robô j chegar ao alvo e deixar essa região, permitindo uma navegação mais suave.

Uma execução em simulação desse algoritmo pode ser vista na Figura 7, onde 48 robôs foram posicionados aleatoriamente fora da região *perigo* e da região *livre*. Todos os robôs possuíam um alvo no centro do cenário. Os robôs são representados por diferentes formas de acordo com seus estados: *normal* ($+$), *esperando* ($\circ$), *preso* ($\triangle$) e *impaciente* ($\times$). Robôs no estado *normal* que já chegaram no alvo proposto e estão se movimentando em direção ao próximo alvo são representados pelo símbolo (*). O círculo externo representa a região *perigo*, enquanto o interno representa a região *livre*. Como pode ser observado, os robôs *esperando* formam uma barreira na região *perigo*, enquanto os robôs

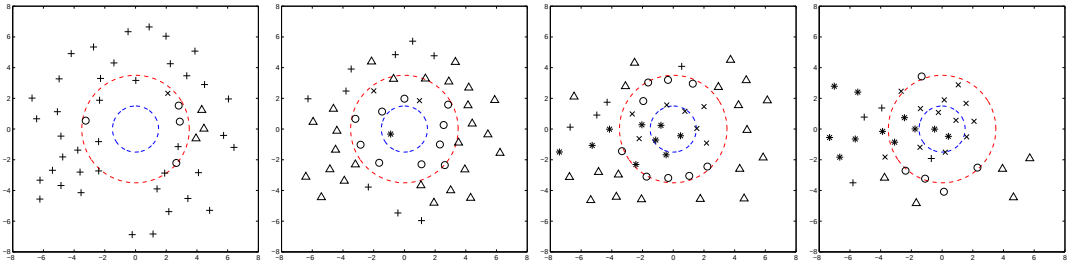


Figura 7. Execuções em simulação do algoritmo proposto ACA.

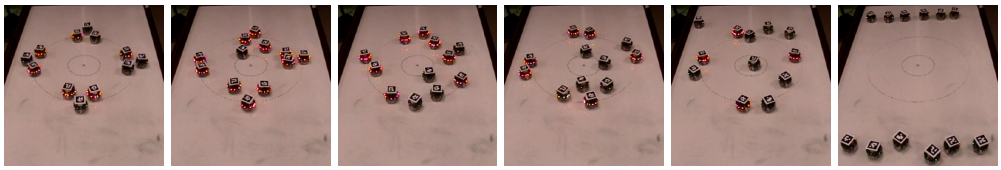


Figura 8. Execução real do algoritmo proposto ACA.

no estado *preso* tendem a esperar fora dessa região. Isso permite que todos os robôs cheguem no alvo de uma forma mais suave, já que o número de disputas é bem menor. Foi realizada uma análise experimental, com um intervalo de confiança de 95%, onde foi obtido um ganho no tempo de convergência de 20% em relação a uma execução utilizando apenas forças de repulsão. Essas simulações foram executadas utilizando o Player/Stage.

Também foi realizada uma execução real desse algoritmo, utilizando doze robôs *e-puck*, que pode ser vista na Figura 8. E-pucks com os leds acesos estão no estado *esperando* ou *preso*, enquanto e-pucks com os leds apagados estão no estado *normal* ou *impaciente*. Doze robôs são distribuídos em torno da região do alvo (indicada por uma pequena marca nas imagens) em grupos de três. Após chegar no alvo comum, cada robô deve se movimentar para seu alvo individual na região superior ou inferior do cenário. Como pode ser observado, os robôs conseguiram completar a tarefa de forma suave. Assim, essa prova de conceito mostra que o algoritmo é viável para um grupo de robôs reais, permitindo-os navegar com mais eficiência. Mais detalhes sobre os experimentos realizados, incluindo provas de convergência, podem ser vistos em [Marcolino and Chaimowicz 2009b].

4. Conclusão

Neste trabalho[1], foram apresentadas soluções para que um enxame de robôs navegue de forma mais eficaz e eficiente. Uma delas utiliza o trabalho coordenado dos robôs para que eles consigam navegar em direção a um alvo em um cenário contendo obstáculos desconhecidos. As outras apresentam algoritmos de coordenação distribuída para evitar o problema do congestionamento quando grupos de robôs navegam em direções opostas ou possuem um alvo em comum. Foram realizadas simulações, análises experimentais e execuções reais para avaliar os algoritmos, onde se mostrou que eles são viáveis e permitem ao enxame ter uma navegação mais suave e eficiente. Como trabalho futuro pretende-se investigar mais profundamente as situações apresentadas, para atingir maiores ganhos de eficiência e, assim, permitir uma navegação ainda melhor de um enxame de robôs.

[1]Este trabalho foi parcialmente financiado pelo CNPq e Fapemig. Os autores gostariam de agradecer Renato Garcia do VeRLab - UFMG e Nathan Michael, Jonh Fink e Vijay Kumar do Grasp Lab. University of Pennsylvania pelo apoio nos experimentos.

Referências

Cianci, C. M., Raemy, X., Pugh, J., and Martinoli, A. (2007). Communication in a Swarm of Miniature Robots: The e-Puck as an Educational Tool for Swarm Robotics. In *Simulation of Adaptive Behavior (SAB-2006), Swarm Robotics Workshop*, Lecture Notes in Computer Science (LNCS), pages 103–115.

Dresner, K. and Stone, P. (2005). Multiagent traffic management: an improved intersection control mechanism. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 471–477.

Hsieh, M. A. and Kumar, V. (2006). Pattern generation with multiple robots. In *Proceedings of the 2006 International Conference on Robotics and Automation (ICRA)*, pages 2442–2447, Orlando, FL.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Rob. Res.*, 5(1):90–98.

Kloetzer, M. and Belta, C. (2006). Hierarchical abstractions for robotic swarms. In *Proceedings of the 2006 International Conference on Robotics and Automation (ICRA)*, pages 952–957, Orlando, FL.

Marcolino, L. S. and Chaimowicz, L. (2008a). A coordination mechanism for swarm navigation: Experiments and analysis (short paper). In *Proc. of the Seventh Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 1203–1206.

Marcolino, L. S. and Chaimowicz, L. (2008b). Experiments in the coordination of large groups of robots. In *Proceedings of the Brazilian Symposium on Artificial Intelligence*, pages 268–277.

Marcolino, L. S. and Chaimowicz, L. (2008c). No robot left behind: Coordination to overcome local minima in swarm navigation. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, pages 1904–1909.

Marcolino, L. S. and Chaimowicz, L. (2009a). Traffic control for a swarm of robots: Avoiding group conflicts. Submitted to the 2009 IEEE International Conference on Intelligent Robots and Systems.

Marcolino, L. S. and Chaimowicz, L. (2009b). Traffic control for a swarm of robots: Avoiding target congestion. Submitted to the 2009 IEEE International Conference on Intelligent Robots and Systems.

Michael, N., Fink, J., and Kumar, V. (2008). Experimental testbed for large multi-robot teams: Verification and validation. *IEEE Robotics and Automation Mag.*, 15(1):53–61.

Pimenta, L. C. A., Fonseca, A. R., Pereira, G. A. S., Mesquita, R. C., Silva, E. J., Caminhas, W. M., , and Campos, M. F. M. (2005). On computing complex navigation functions. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3463–3468.

Treuille, A., Cooper, S., and Popović, Z. (2006). Continuum crowds. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 1160–1168, New York, NY, USA. ACM.

Viswanath, K. and Krishna, K. M. (1997). Sensor network mediated multi robotic traffic control in indoor environments. In *Proc. of the Int. Conf. on Advanced Robotics*.