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 distributed coordination algorithms that allow a swarm of robots to navigate in complex scenarios. 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 distribuída que permitem a um enxame de robôs navegar em cenários complexos. 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. O uso de enxames pode trazer diversas vantagens. A divisão do trabalho entre os membros do time em geral aumenta a eficiência do sistema. Além disso, também se consegue mais robustez, pois com um número maior de robôs é mais fácil ter redundância e, portanto, desenvolver sistemas com tolerância a falhas. Porém, há muitos desafios ao se trabalhar com enxames. Em geral, eles devem trabalhar de forma distribuída e usar recursos limitados de processamento e comunicação. Devido à 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. Experimentos reais são muito importantes em robótica para mostrar a viabilidade de um algoritmo em cenários reais, com todos os problemas causados pelas incertezas dos sensores e atuadores, falhas de comunicação, restrições de tempo real, etc. Porém, apenas alguns trabalhos em enxames utilizam robôs reais para validar os algoritmos, como por exemplo [Pimenta et al. 2008, Correll et al. 2006, McLurkin and Smith 2004]. 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 [Chaimowicz et al. 2005, Bachmayer and Leonard 2002]. 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 ([Kamphuis and Overmars 2004], [Kloetzer and Belta 2006], [Li and Chou 2003]). 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 são 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*. A Figura 1 mostra a máquina de estados finitos utilizada no mecanismo de coordenação deste trabalho.

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.

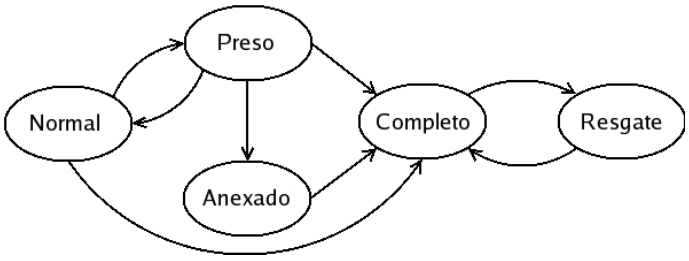

Figura 1. Máquina de estados finitos mostrando os estados possíveis e as transições para cada membro do enxame.

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. Além disso, quando um robô *anexado* envia uma mensagem a um robô *preso*, ele adiciona sua posição atual como um novo ponto na informação de caminho. 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. Assim, robôs que não teriam uma linha de visão direta com nenhum robô de *resgate* podem facilmente escapar do mínimo local graças ao novos pontos que são adicionados ao caminho por esse processo. 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. As simulações foram executadas utilizando o MuRoS, um simulador de múltiplos robôs que permite implementar diversas tarefas, testar diferentes controladores e observar os robôs em tempo real [Chaimowicz et al. 2001]. Na Figura 2 pode ser vista uma execução em 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 desse algoritmo. Dois cenários foram utilizados para realizar essa análise. O primeiro é composto por um alvo quadrado rodeado por quatro obstáculos em forma de `U`. O segundo cenário é o mesmo apresentado na Figura 2. Cada execução foi realizada 10 vezes e a média aritmética dos resultados foi calculada. Comparou-se a performance do algoritmo proposto com dois outros: um Algoritmo Simples, onde os robôs não desenvolvem nenhuma estratégia para escapar do mínimo local e um Algoritmo Aleatório, onde forças aleatórias

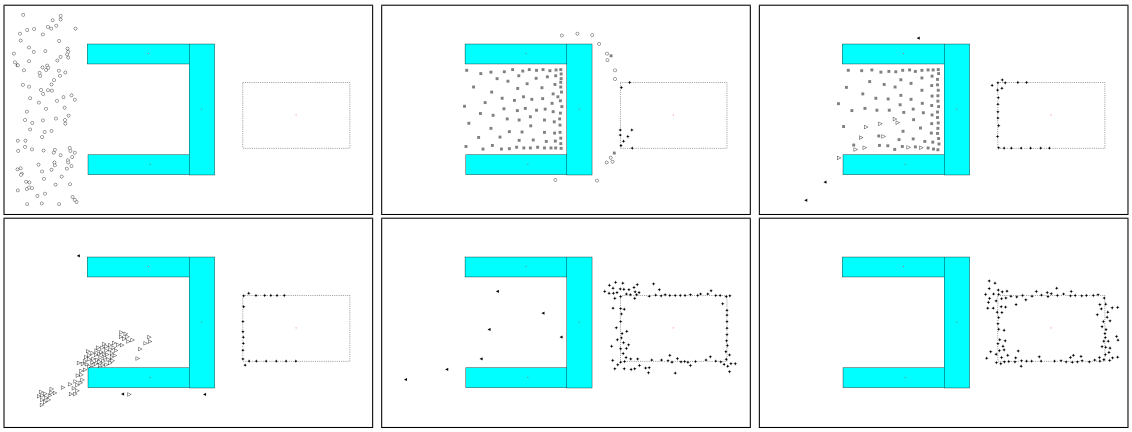

Figura 2. Execução em simulação do AR.

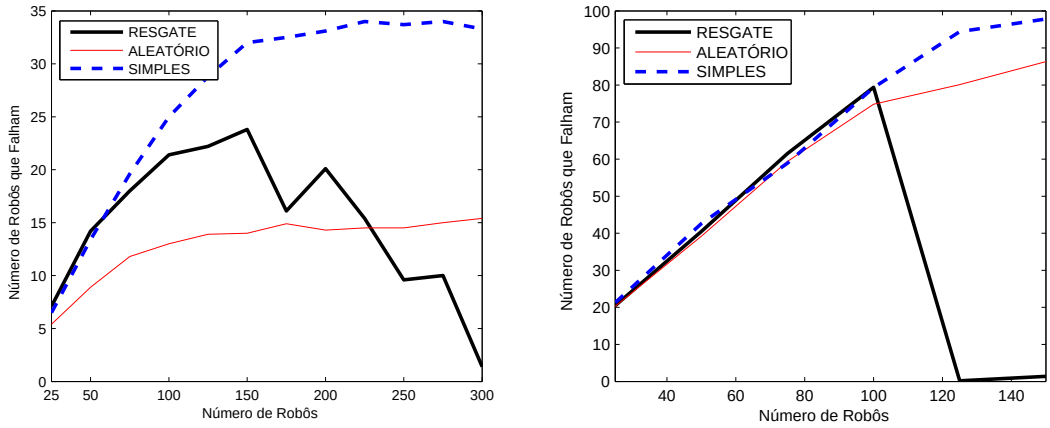

Figura 3. Número de robôs que falham para o primeiro (esquerda) e para o segundo cenário (direita).

são aplicadas nos robôs presos. Basicamente, no Algoritmo Aleatório, se um robô estiver no estado *preso*, ele fica sujeito a uma força aleatória (em termos de direção e tempo) para tentar escapar do mínimo local. A Figura 3 mostra o número de robôs que falharam, ou seja, que não conseguiram chegar ao alvo. Como pode ser observado, acima de um determinado número de robôs o algoritmo proposto obteve melhores resultados do que os outros algoritmos. O número de robôs que falharam chegou próximo de zero no segundo cenário. A porcentagem de falha tende a diminuir quando a quantidade de robôs torna-se alta, indicando um aumento na qualidade da convergência quando aumenta-se o número de robôs.

Também foram realizadas simulações medindo o número total de mensagens trocadas com o aumento do número de robôs. Foram executadas simulações no primeiro cenário com o raio de comunicação configurado para 20 vezes o raio dos robôs. Os resultados podem ser vistos na Figura 4. Basicamente, existem dois tipos de mensagens no algoritmo: "mensagens de estado", que são enviadas pelos robôs presos pedindo por ajuda, e "mensagens de caminho", que são enviadas pelo robô de *resgate* ao receber uma mensagem de estado. Como as mensagens de estado são enviadas por robôs presos pedindo por ajuda, sua variação é proporcional ao número de robôs que falham (mostrado

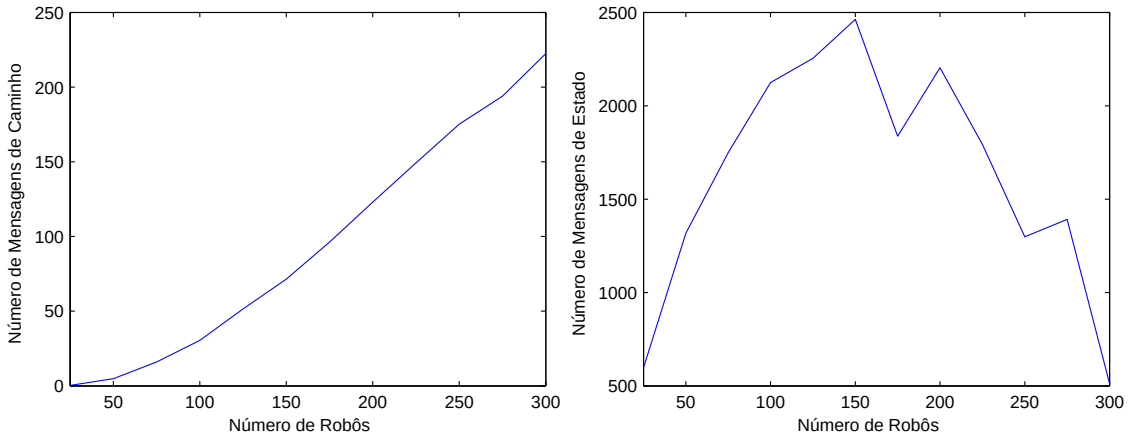

Figura 4. Variação do número de mensagens de caminho (esquerda) e de estado (direita) com o aumento do número de robôs.

na Figura 3). A troca de informações de caminho depende do número de robôs no estado *resgate* e no estado *preso*. Como pode ser observado na Figura 4, o número de mensagens de caminho tende a aumentar linearmente com o número de robôs. Além disso, o número de mensagens de estado é bem maior do que o número de mensagens com informações de caminho. Mensagens de estado geralmente são muito pequenas, com apenas alguns bytes representando o estado do robô. Por outro lado, o tamanho das mensagens de caminho depende do número de pontos armazenados pelo robô. Alguns experimentos realizados nesse cenário mostraram que esse número não é muito alto (aproximadamente 230 pontos) e decresce com o aumento do número de robôs. Além disso, após eliminar pontos redundantes, o número total torna-se menor do que 20, o que não compromete a largura de banda. Atualmente estamos trabalhando em uma análise experimental mais detalhada deste algoritmo, incluindo análise de intervalo de confiança, regressão linear, etc.

O algoritmo também foi validado em experimentos reais utilizando sete robôs *scarab* do *Grasp Lab. - University of Pennsylvania*, dentro de um projeto de cooperação internacional NSF-CNPq. Os *scarabs* (Figura 5) são robôs diferenciais, controlados cinematicamente. Eles são equipados com computador on-board, sensor laser Hokuyo URG, comunicação wireless (802.11) e dois motores de passo para a atuação. Cada robô tem acesso a sua posição absoluta, utilizando um conjunto de câmeras fixadas no teto. Mais detalhes sobre os *scarabs* e o sistema de localização podem ser vistos em [Michael et al. 2008]. Os robôs não têm acesso a nenhum mapa do ambiente, e devem perceber os obstáculos e os outros robôs localmente, utilizando os sensores laser. O laser também é utilizado para verificar a existência de uma linha de visão direta entre um robô *preso* e um robô de *resgate*.

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 6. Na Figura 6(a), três robôs conseguem mover em direção ao alvo, enquanto os outros estão presos em uma região de mínimo local, em frente ao obstáculo. Os robôs presos estão espalhados nessa região, por causa das forças locais de repulsão. A Figura 6(b) mostra um robô de *resgate* na direita do obstáculo, enviando um caminho viável para o alvo. Os robôs que têm uma linha de visão direta com o robô de *resgate* aceitam essa mensagem e mudam de estado para *anexado*. Um deles retransmite a informação para

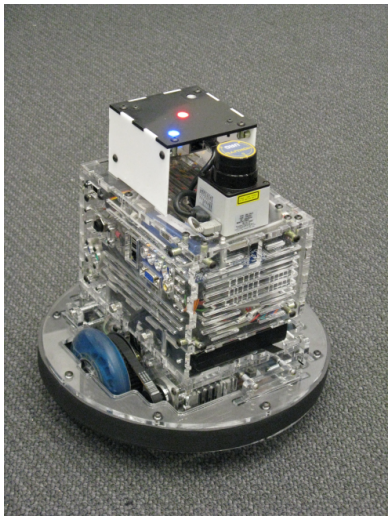

Figura 5. Um dos sete robôs scarab utilizados nos experimentos.

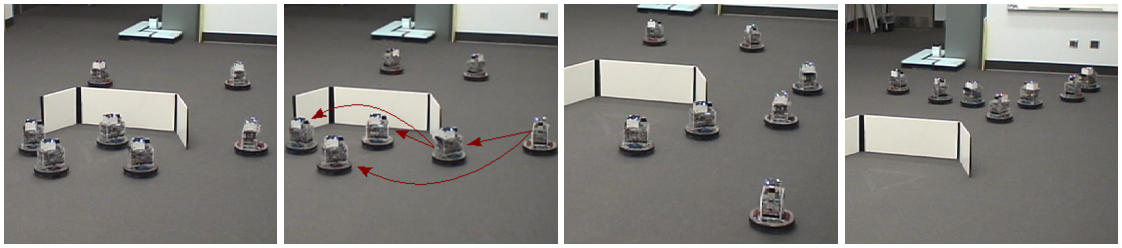

Figura 6. Execução real utilizando o AR. As setas indicam transmissão de mensagem.

dois outros robôs no estado *preso* que não possuíam uma linha de visão direta, permitindo a todos escapar da região de mínimo local, como pode ser visto na Figura 6(c). O estado na Figura 6(d) é logo alcançado, onde quase todos os robôs já atingiram o alvo. 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.

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 multi-agentes. 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. Ao invés de lidar com controle de tráfego, existem métodos que procuram encontrar formas mais eficientes para evitar colisões do que forças locais de repulsão. Em [Krishna and Hexmoor 2004] é proposto um algoritmo onde os robôs coordenam suas velocidades de forma a evitar uma colisão. A coordenação pode envolver não só os robôs diretamente envolvidos na provável colisão, como também os robôs vizinhos, que podem ter que alterar suas velocidades para auxiliar os robôs envolvidos. Outros trabalhos que lidam com evitar colisões são [Jager and Nabel 2001, Yasuaki and Yoshiki 2001, Cai et al. 2007]. Porém, evitar colisões não significa necessariamente evitar congestionamentos. Mesmo com um bom método para evitar colisões o sistema pode ficar aglomerado e pouco eficiente. Além disso, em geral esses trabalhos não mostram casos com um grande número 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 7(a). Os robôs que perceberam o risco de um congestionamento enviam uma mensagem aos seus vizinhos, como pode ser visto na Figura 7(b). Cada robô, ao receber essa mensagem, a retransmite para seus respectivos vizinhos, como mostra a Figura 7(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 7(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. Uma representação gráfica do controlador de desvio pode ser vista na Figura 8, onde a constante ϕ controla a distância que os robôs irão percorrer no eixo do desvio.

Foram realizados experimentos em simulação e com robôs reais, apresentados em [Marcolino and Chaimowicz 2009a]. 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. Na Figura 9 pode ser visto o resultado utilizando apenas forças de repulsão locais. Na Figura 10 o resultado utilizando o ACG pode ser visto. Como pode ser observado visualmente, o comportamento dos robôs foi

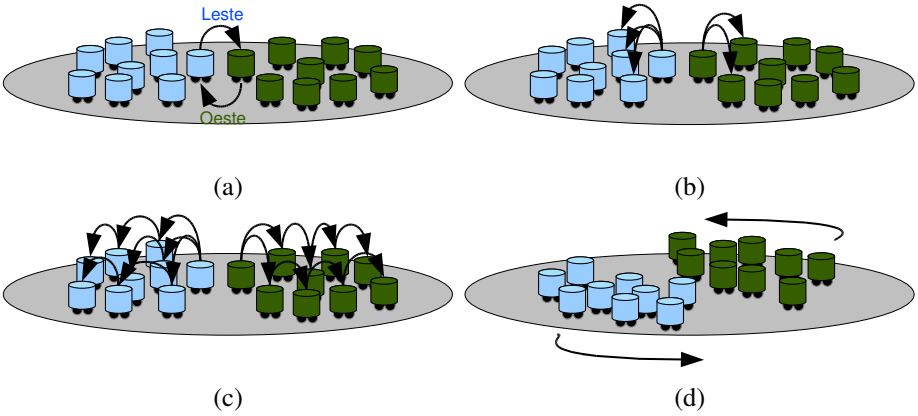

Figura 7. Passos da execução do algoritmo de coordenação proposto ACG.

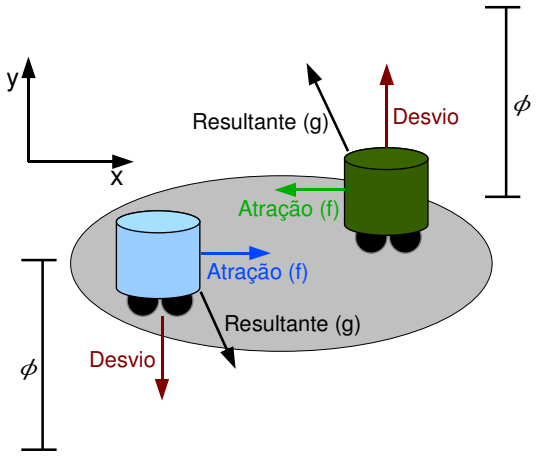

Figura 8. Controlador utilizado pelo robô para evitar um congestionamento no ACG.

melhor utilizando o algoritmo proposto. A navegação dos robôs foi mais suave, os grupos mantiveram-se coesos e o risco de colisões foi baixo. Já na navegação utilizando apenas forças locais de repulsão, os grupos se misturaram, ocorrendo um alto risco de colisões. Pode-se perceber uma aglomeração dos robôs na região central, caracterizando uma situação de congestionamento. Pode-se observar também que enquanto alguns robôs já conseguiram atingir o alvo, outros continuam presos na região do congestionamento, desperdiçando recursos. Percebe-se, assim, que a navegação utilizando o método de coordenação proposto é mais eficiente e confiável. Uma execução com quatro grupos de robôs pode ser vista na Figura 11. 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 (não coordenado). Utilizou-se na avaliação o cenário em que dois grupos de robôs se movimentam em direções opostas. Variou-se o número de robôs por grupo, e mediu-se o tempo de execução e o número de mensagens utilizadas. Para um determinado número de robôs, rodou-se a simulação 20 vezes e calculou-se a média aritmética do resultado. Na

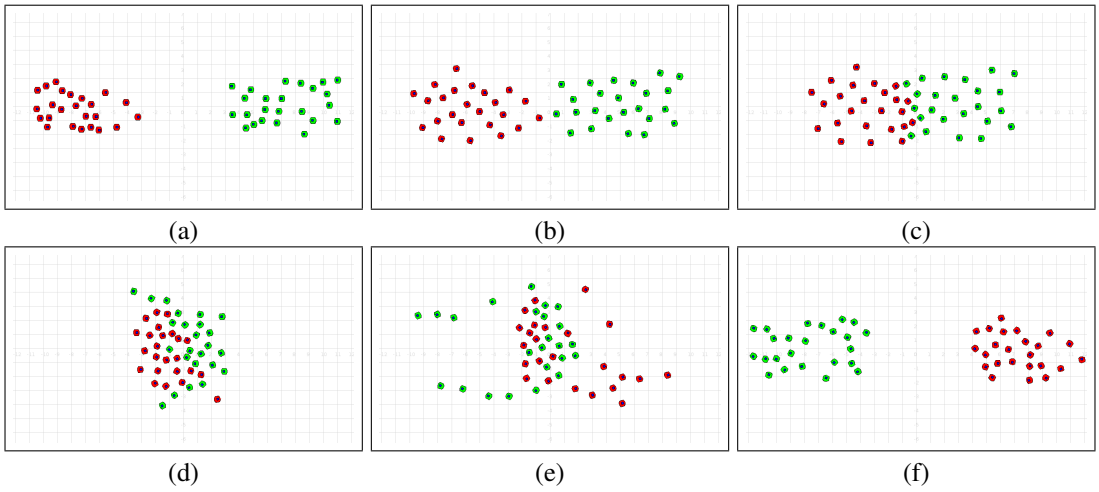

Figura 9. Execução com dois grupos utilizando apenas forças de repulsão locais.

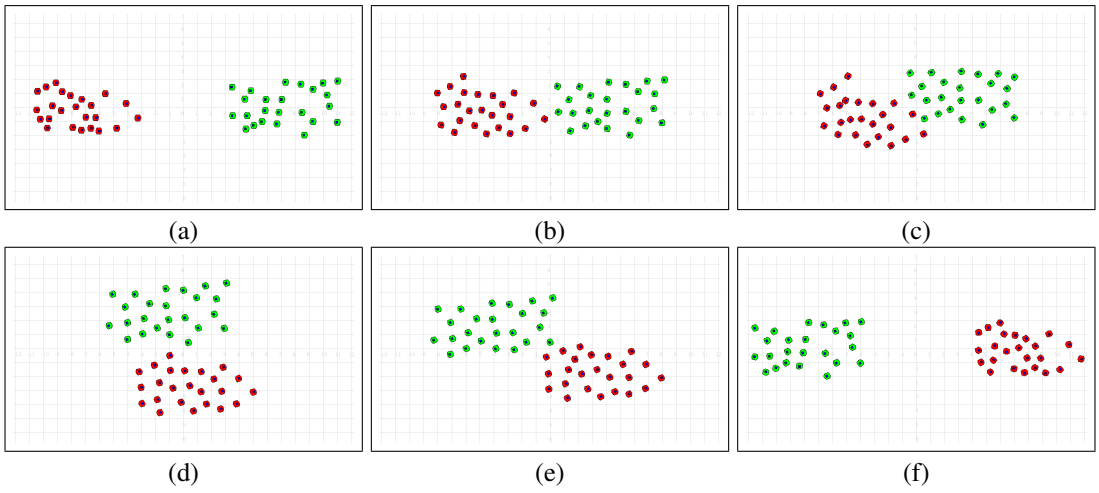

Figura 10. Execução com dois grupos utilizando o algoritmo ACG.

Figura 12 pode ser visto o tempo utilizado pelos dois algoritmos. Como pode ser observado, o tempo de convergência do algoritmo não coordenado aumenta monotonicamente com o aumento do número de robôs, enquanto o algoritmo proposto se aproximou de um número fixo de iterações. Também pode ser observado que o ACG teve um tempo de convergência menor do que o algoritmo não coordenado. Foi executado um *teste-t* que mostrou que o ACG foi melhor em todos os pontos analisados com 95% de confiança. Também foi calculado o desvio padrão dos resultados, o que mostrou que o algoritmo proposto tem um menor desvio em relação à média. O ganho no tempo de convergência chegou a 40% utilizando o algoritmo proposto.

Na Figura 13 pode ser visto o número de mensagens utilizadas pelo algoritmo proposto para um número variável de robôs. O melhor modelo linear encontrado foi $y = 34,0875x + 6,7446$, com um coeficiente de determinação (R^2) de $0,9978$. Já o melhor modelo quadrático, mostrado na figura, foi $y = -0,2969x^2+42,9942x-49,3679$, com um coeficiente de determinação de $0,9998$. Apesar do melhor modelo ter sido uma função quadrática, o termo quadrático é pequeno e negativo. Portanto, esse resultado mostra que o algoritmo escala bem e é adequado para um grande número de robôs.

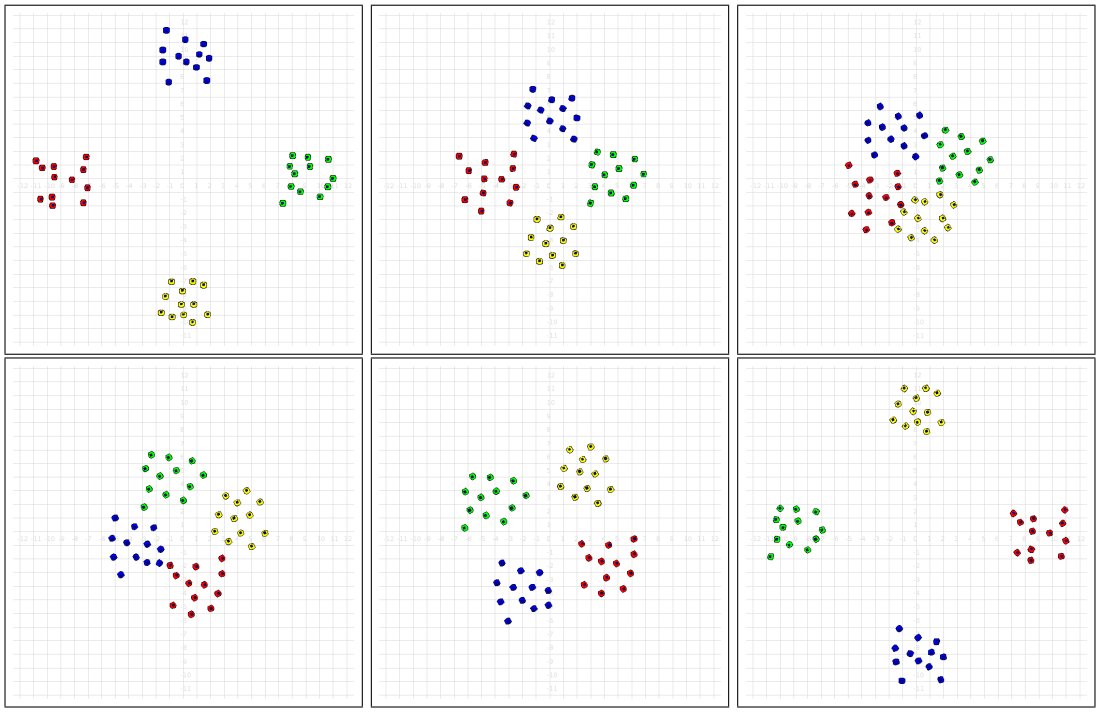

Figura 11. Execução com quatro grupos do algoritmo proposto ACG.

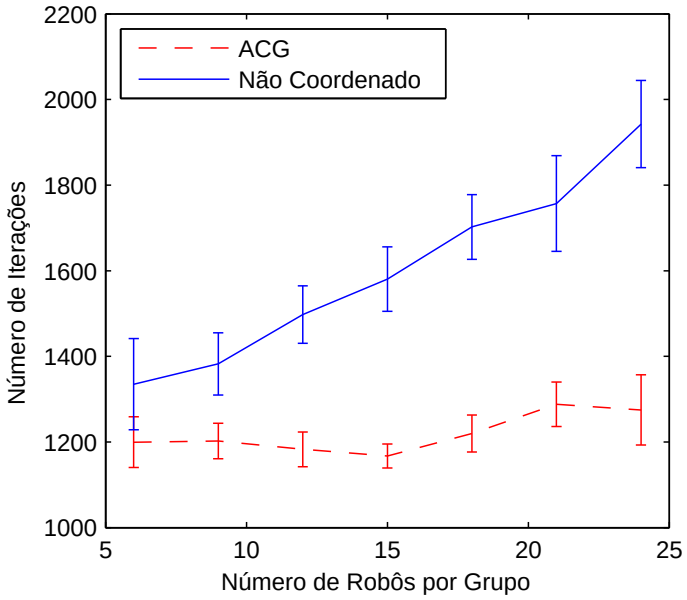

Figura 12. Tempo utilizado pelos dois algoritmos. As barras indicam o intervalo de confiança dos resultados, com 95% de confiança.

Foram realizadas diversas execuções reais desse algoritmo, utilizando o arcabouço de experimentação desenvolvido no VeRLab - Laboratório de Visão Computacional e Robótica do DCC – UFMG [Garcia and Chaimowicz 2009]. Esse arcabouço é composto por doze robôs e-puck, uma plataforma de programação integrada ao arcabouço Player/Stage/Gazebo [Gerkey et al. 2003] além de mecanismos de comunicação e localização. O e-puck é um robô diferencial de pequeno porte (7cm de diâmetro) que é muito apropriado para experimentações com enxames [Cianci et al. 2007]. Cada robô

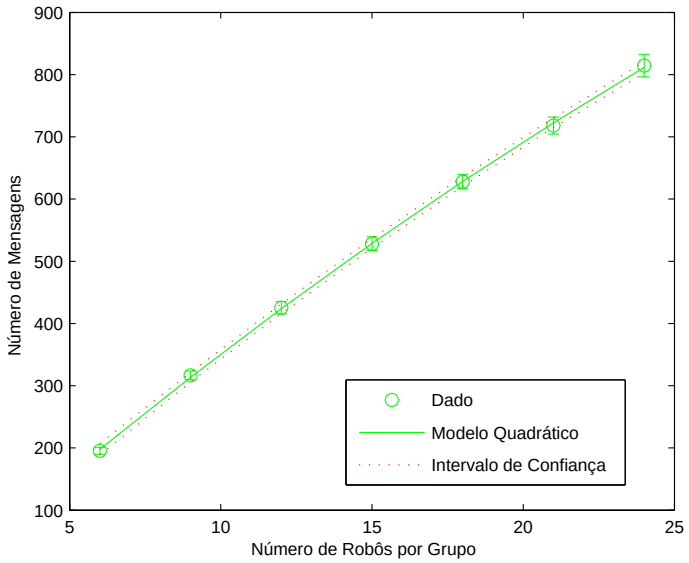

Figura 13. Número de mensagens enviadas, utilizando o ACG, para um número crescente de robôs. Os intervalos de confiança correspondem a um nível de confiança de 95%.

é equipado com um anel de 8 sensores infra-vermelho que permitem a percepção de obstáculos nas proximidades e um grupo de leds coloridos que indicam o estado do robô. O processamento local é realizado com um microprocessador dsPIC e uma interface sem fio bluetooth permite comunicação entre os robôs e a execução de controle remoto. O robô também possui uma micro-camera, um acelerômetro 3D, alto-falante e microfones. A Figura 14 mostra o "enxame" de robôs do VeRLab.

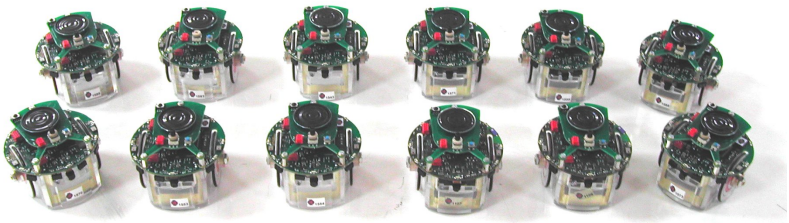

Figura 14. Doze robôs *e-puck* utilizados nos experimentos.

Os snapshots de uma execução com dois grupos podem ser vistos na Figura 15, enquanto com quatro grupos podem ser vistos na Figura 16. Os gráficos indicam a posição dos robôs e os respectivos estados: robôs seguindo o controlador normal ($+$); robôs seguindo o controlador de desvio ($\circ$), e robôs que, após desviar, retornaram ao controlador normal ($\times$). E-pucks com os leds acesos estão desviando da região de congestionamento. Como pode ser observado, utilizando o algoritmo proposto os robôs completaram a tarefa de uma forma suave nos dois cenários. Nas duas execuções o tempo total foi aproximadamente 2 minutos. Também foram rodados os mesmos cenários utilizando apenas forças locais de repulsão, onde foi necessário aproximadamente 4,5 minutos no caso com dois grupos e 6 minutos no caso com quatro grupos. O ganho no tempo de convergência foi de 55% no primeiro caso e de 66% no segundo. Portanto, essas provas de conceito mostram que o algoritmo é viável em um grupo de robôs reais, permitindo-os navegar de

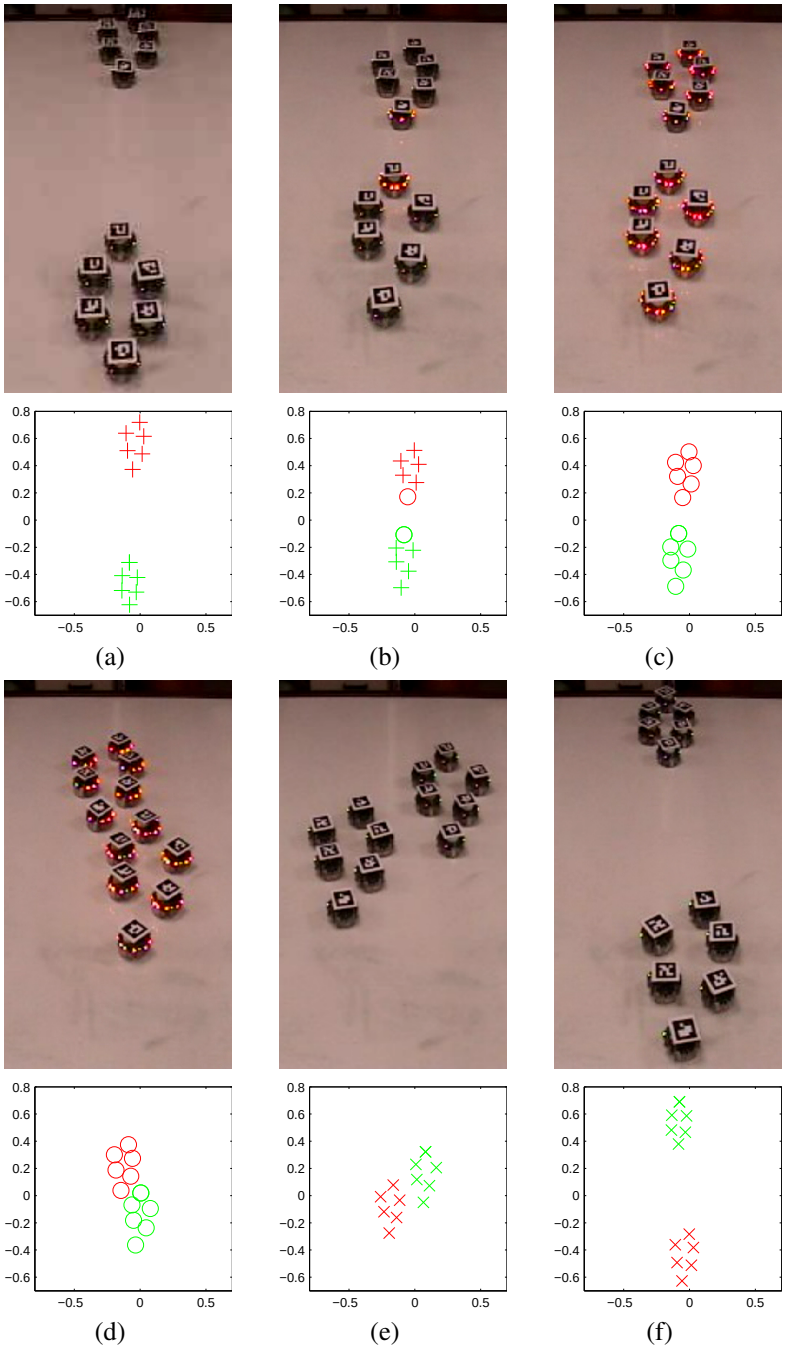

Figura 15. Execução real com dois grupos utilizando o algoritmo ACG.

uma forma suave e eficiente mesmo quando grupos de robôs 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 encontra-

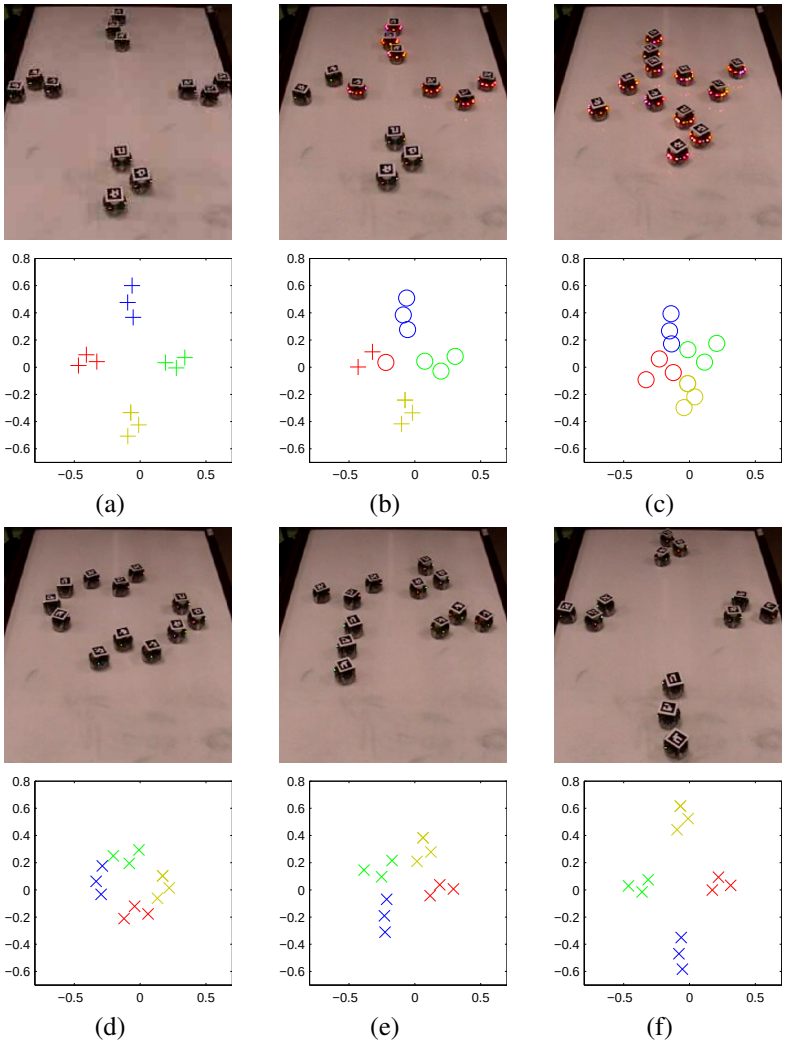

Figura 16. Execução real com quatro grupos utilizando o algoritmo ACG.

dos em [Marcolino and Chaimowicz 2009b]. A solução é modelada como uma *Máquina de Estados Finitos Probabilística* [Vidal et al. 2005], onde as arestas são marcadas com probabilidades que definem a transição que será tomada. A Figura 17 mostra a máquina de estados finitos probabilística utilizada neste projeto. Como pode ser visto, 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 $1 - \rho$.

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 (veja a Figura 18). Essa área-α será utilizada para detectar outros robôs que possam interferir com a navegação em direção ao alvo.

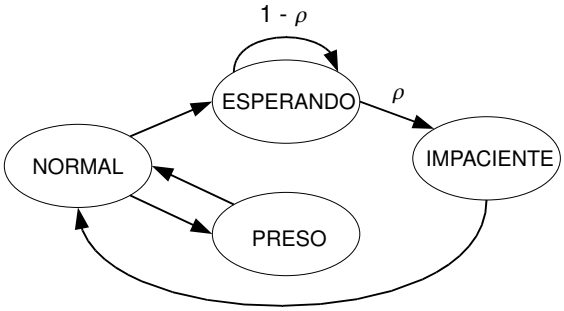

Figura 17. Máquina de estados finitos probabilística do ACA mostrando os estados possíveis e as transições para cada membro do enxame.

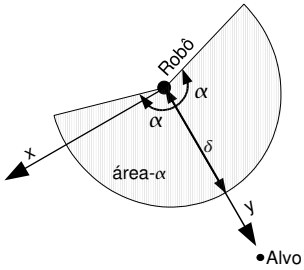

Figura 18. Área de sensoriamento (área-α) considerada por um robô para trocar o seu estado.

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 19(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 19(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 19(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 20, 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*

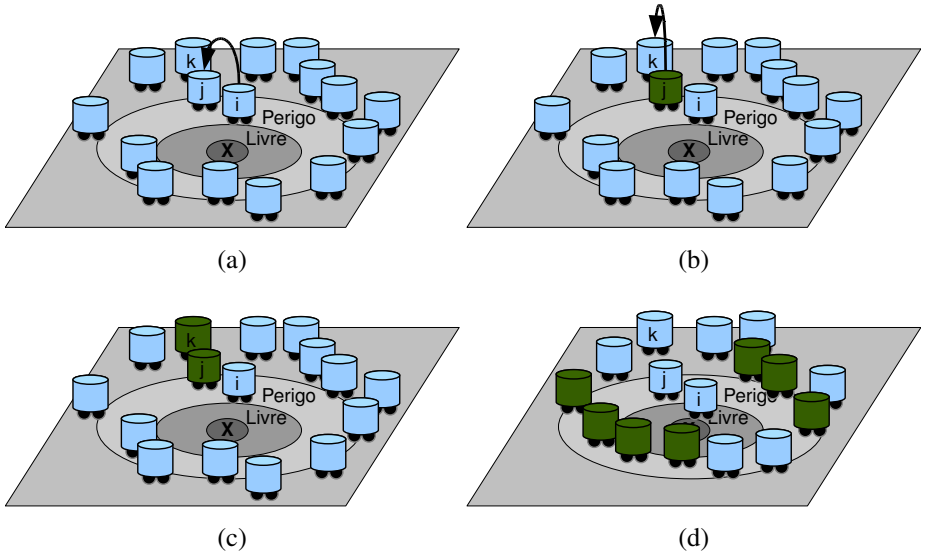

Figura 19. Passos da execução do algoritmo de coordenação proposto ACA. A cor verde identifica robôs no estado *esperando* ou *preso*.

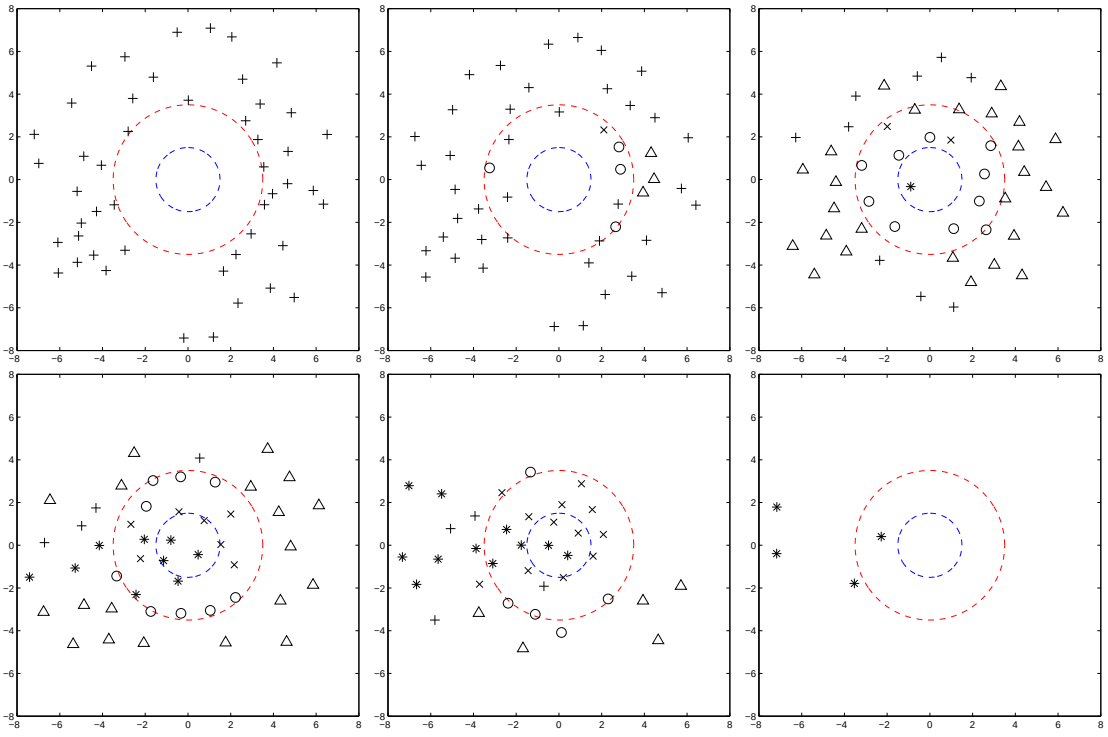

Figura 20. Execuções em simulação do algoritmo proposto ACA.

(×). 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 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. As simulações foram executadas utilizando o Player/Stage.

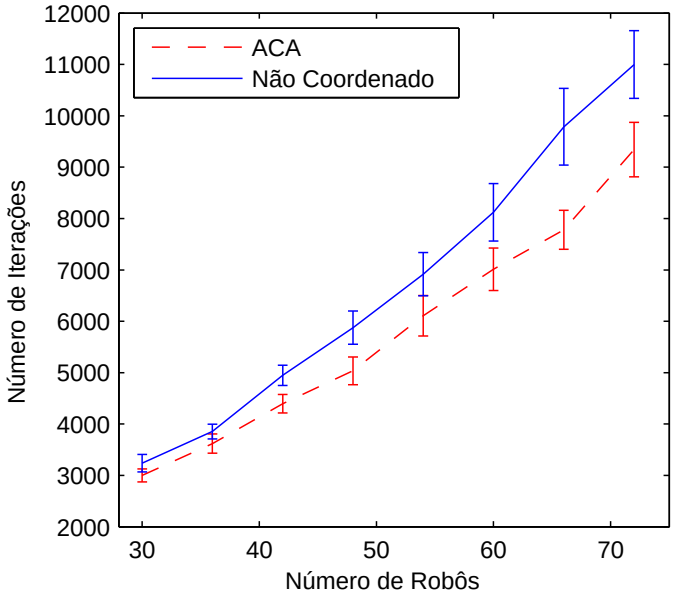

Figura 21. Tempo utilizado pelos dois algoritmos. As barras indicam o intervalo de confiança dos resultados, com 95% de confiança.

Foi executada uma série de simulações para avaliar a eficiência do ACA em comparação com o uso de apenas forças locais de repulsão. A cada execução, os robôs foram posicionados aleatoriamente no cenário, fora da região *perigo*. Variou-se o número de robôs e mediu-se o tempo de execução e o número de mensagens enviadas. Cada simulação foi executada 20 vezes e a média dos resultados foi considerada. A Figura 21 mostra o tempo de execução para um número variável de robôs. Como pode ser observado, o algoritmo proposto obteve uma melhor eficiência com o aumento do número de robôs. Foi executado um *teste-t* que mostrou que o ACA foi melhor em todos os pontos analisados com mais de 42 robôs com 95% de confiança. Além disso, o ganho de tempo de convergência chegou a 20% com o uso do algoritmo proposto. Também foi calculado o desvio padrão dos resultados, o que mostrou que com mais de 42 robôs o ACA possui um menor desvio em relação à média.

Na Figura 22 pode ser visto o número de mensagens utilizadas pelo ACA para um número variável de robôs. O melhor modelo encontrado para a curva foi a quadrática $y = 0,5107x^2 + 7,4987x - 30,2645$, com um coeficiente de determinação (R^2) de $0,9964$. Apesar do modelo ser uma função quadrática, o termo quadrático é pequeno. Esse resultado mostra que o algoritmo escala bem e é adequado para um grande número de robôs.

Também foi realizada uma execução real desse algoritmo, utilizando doze robôs *e-puck*, que pode ser vista na Figura 23. 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. Na Figura 23(a) pode-se ver a posição inicial dos robôs (numerados de 1 a 12). Ao entrar na região *perigo*, os robôs mudam de estado para *esperando* ($\circ$) assim que detectam outro robô com o mesmo alvo na área-α (Figura 23(b)). O robô 10 muda seu estado para

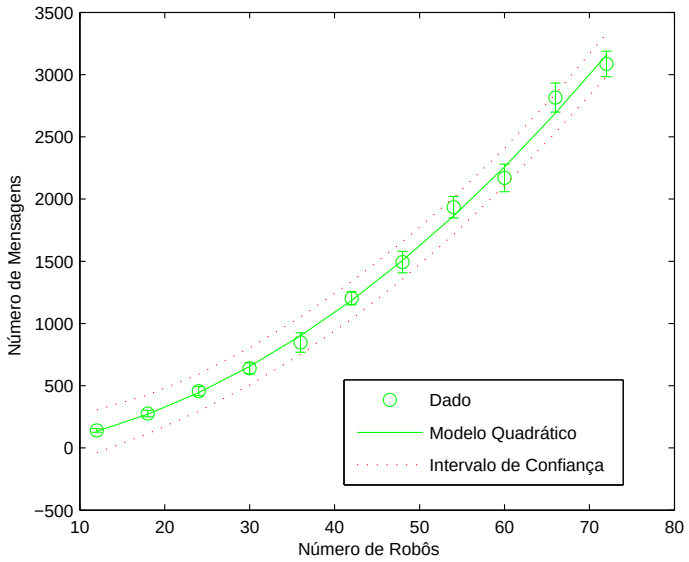

Figura 22. Número de mensagens enviadas, utilizando o ACA, para um número crescente de robôs. Os intervalos de confiança correspondem a um nível de confiança de 95%.

impaciente ($\times$), e começa a se movimentar em direção ao alvo (Figura 23(c)). Os robôs 3 e 6, que estão fora da região de *perigo*, ao detectar robôs no modo *esperando* na área-α, mudam seu estado para *locked* ($\triangle$) (Figura 23(d)). Eles retornam ao estado *normal* ($+$) apenas quando detectam que não há mais robôs no modo *esperando* em sua área-α (Figura 23(e)). Com o passar do tempo, os robôs entram no modo *impaciente* e se aproximam do alvo (Figura 23(f)). Logo, muitos são bem-sucedidos em atingir o alvo comum (*) e se dirigem ao seu próximo objetivo (Figura 23(g)), levando ao estado final em que todos os robôs conseguem completar a tarefa com sucesso (Figura 23(h)).

Como pode ser observado, os robôs conseguiram completar a tarefa de forma suave e eficiente. O tempo total dessa execução foi 7 minutos. Também foi rodado o mesmo cenário utilizando apenas forças locais de repulsão, que precisou de 9 minutos para uma execução completa da tarefa. Portanto, o ganho no tempo de convergência foi de 22%, um resultado melhor do que o que foi encontrado nas simulações, pois foi conseguido com um número pequeno de robôs. Com mais robôs, é esperado que o ganho no tempo de convergência seja ainda melhor. 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, 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 permi-

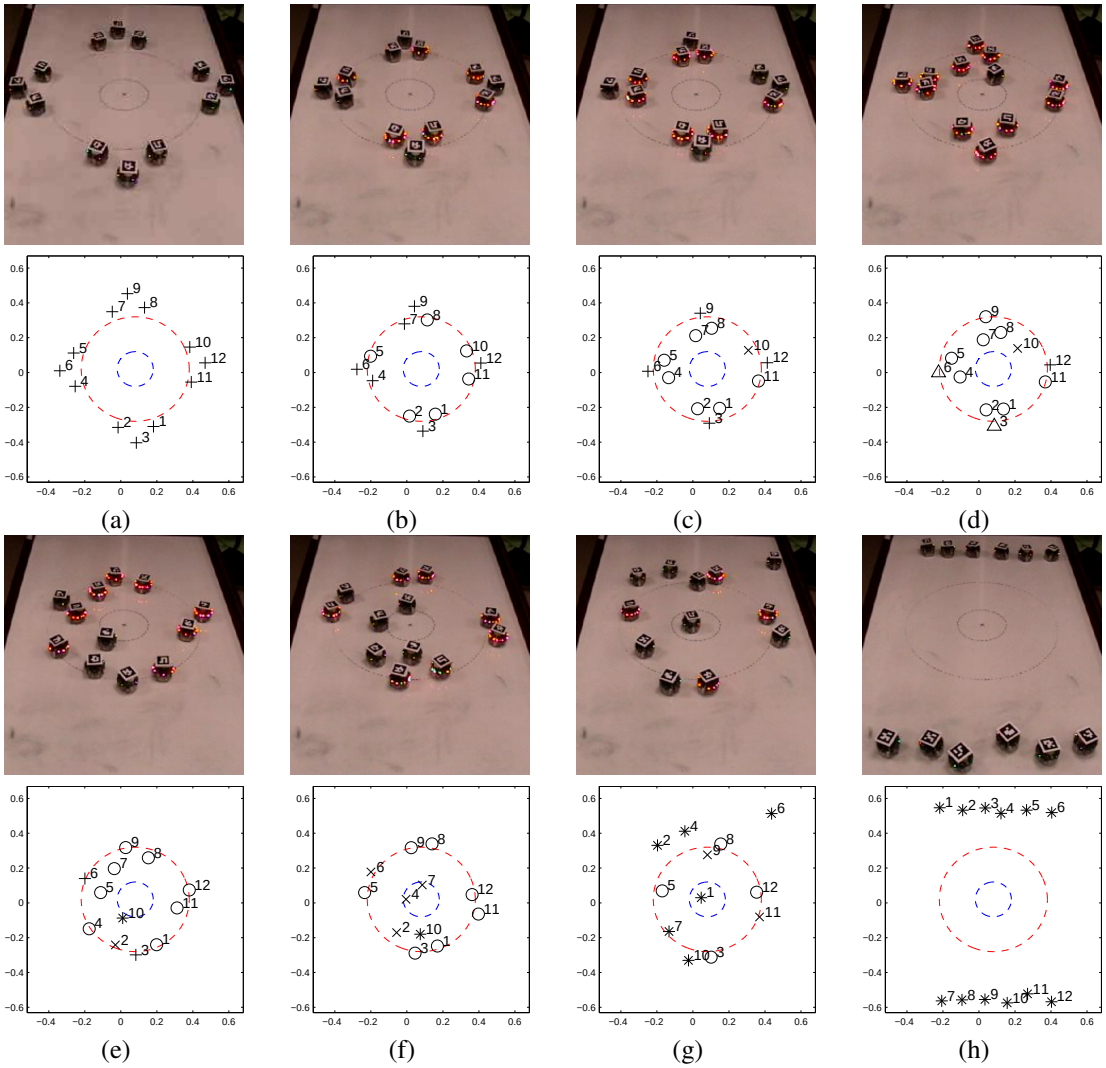

Figura 23. Execução real utilizando o algoritmo de coordenação ACA.

tem 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.

Agradecimentos

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

Bachmayer, R. and Leonard, N. E. (2002). Vehicle networks for gradient descent in a sampled environment. In *Proc. of the IEEE Conf. on Decision and Control*, pages 112–117.

Cai, C., Yang, C., Zhu, Q., and Liang, Y. (2007). Collision avoidance in multi-robot systems. In *Proceedings of the 2007 IEEE International Conference on Mechatronics and Automation*, pages 2795–2800, Harbin, China.

Chaimowicz, L., Campos, M., and Kumar, V. (2001). Simulating loosely and tightly coupled multi-robot cooperation. In *Proc. of V SBAI – Brazilian Symposium on Intelligent Automation*.

Chaimowicz, L., Michael, N., and Kumar, V. (2005). Controlling swarms of robots using interpolated implicit functions. In *Proc. of the 2005 IEEE Int. Conf. on Robotics and Automation*, pages 2498–2503.

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.

Correll, N., Rutishauser, S., and Martinoli, A. (2006). Comparing coordination schemes for miniature robotic swarms: A case study in boundary coverage of regular structures. In *Proc. of the Int. Symposium on Experimental Robotics (ISER)*.

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, New York, NY, USA. ACM.

Garcia, R. and Chaimowicz, L. (2009). Uma infra-estrutura para experimentação com enxames de robôs. In *Anais do IX Simpósio Brasileiro de Automação Inteligente*.

Gerkey, B., Vaughan, R. T., and Howard., A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics*, pages 317–323, Coimbra, Portugal.

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.

Jager, M. and Nabel, B. (2001). Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots. In *Proceedings of the 2001 IEEE International Conference on Intelligent Robots and Systems, IROS 2001*, pages 1213–1219, Maui, USA.

Kamphuis, A. and Overmars, M. (2004). Motion planning for coherent groups of entities. In *Proc. of the 2004 IEEE Int. Conf. on Robotics and Automation*, pages 3815–3821, New Orleans, Louisiana.

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.

Krishna, K. M. and Hexmoor, H. (2004). Reactive collision avoidance of multiple moving agents by cooperation and conflict propagation. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 2141–2146.

Li, T. Y. and Chou, H. C. (2003). Motion planning for a crowd of robots. In *Proc. of the 2003 IEEE Int. Conf. on Robotics and Automation*, pages 4215–4221.

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. In *Proceedings of 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. In *Proceedings of the 2009 IEEE International Conference on Intelligent Robots and Systems*.

McLurkin, J. and Smith, J. (2004). Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. In *Proc. of the 7th Int. Symposium on Distributed Autonomous Robotic Systems*.

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

Pimenta, L., Michael, N., Mesquita, R. C., Pereira, G. A. S., and Kumar, V. (2008). Control of swarms based on hydrodynamic models. In *Proc. of the 2008 IEEE Int. Conf. on Robotics and Automation*, pages 1948–1953.

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.

Vidal, E., Thollard, F., de la Higuera, C., Casacuberta, F., and Carrasco, R. C. (2005). Probabilistic finite-state machines-part i. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(7):1013–1025.

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

Yasuaki, A. and Yoshiki, M. (2001). Collision avoidance method for multiple autonomous mobile agents by implicit cooperation. In *Proceedings of the 2001 IEEE International Conference on Intelligent Robots and Systems, IROS 2001*, pages 1207–1212, Maui, USA.