Localização Tolerante a Falhas de Usuários Móveis em Ambientes Fechados

Alessandro Copetti, J.C.B. Leite, Orlando Loques

Instituto de Computação, Universidade Federal Fluminense
Niterói, RJ, Brasil

{acopetti, julius, loques}@ic.uff.br

Abstract. *The knowledge of a mobile device location is essential for the development of context-aware applications. This paper investigates solutions for the location problem, in indoor environments, that cater for low cost, precision, and fault tolerance requirements. In addition, the feasibility of execution in devices with restricted resources and the processing time required to execute the location function are also of concern. Specifically, we evaluate two solutions using Neural Networks and the k-nearest neighbor's algorithm, which use radio signals generated by access points to mobile networks. A real physical scenario was used to implement and evaluate the techniques.*

Resumo. *O conhecimento da localização de um dispositivo móvel é essencial para o desenvolvimento de aplicações sensíveis ao contexto. Este trabalho investiga técnicas de localização de dispositivos móveis em ambientes fechados, que atendam aos requisitos de baixo custo, precisão e tolerância a falhas. Como preocupações adicionais, colocam-se a viabilidade de execução em dispositivos com restrições de recursos e o tempo gasto na execução da função de localização. Especificamente, avaliamos soluções baseadas em redes neurais e no algoritmo dos k-vizinhos mais próximos, as quais utilizam os sinais de rádio emitidos por pontos de acesso de redes sem fio. Um cenário real foi utilizado para implementação e avaliação das técnicas.*

1. Introdução

A localização de um dispositivo móvel é uma informação essencial para o desenvolvimento de aplicações sensíveis ao contexto. Num cenário básico de computação móvel usuários deslocam-se portando dispositivos dotados de comunicação sem fio, entrando e saindo de ambientes, onde a aplicação avalia o contexto para tomar as ações corretas. Por exemplo, em um ambiente hospitalar, pode ser importante executar uma determinada ação quando um médico sai de uma sala de cirurgia, ou localizar o médico mais próximo a um determinado local no hospital.

Para ambientes internos (*indoor*), onde o uso de localização via GPS (*Global Positioning System*) não é viável, diversas outras tecnologias e propostas têm sido investigadas. Algumas alternativas são baseadas em sensores especiais que possibilitam alta precisão de localização a um custo relativamente alto. Por outro lado, redes sem fio baseadas em pontos de acesso (PAs) interligados entre si são cada vez mais comuns. Isso possibilita a implementação de técnicas de localização de baixo custo, nas quais os PAs exercem a função dos sensores especiais.

Neste cenário, um dispositivo móvel pode obter os valores das intensidades dos sinais emitidos pelos PAs que estão sendo percebidos naquele lugar, e usar alguma técnica para transformá-los em uma localização física. Numa situação ideal, sem interferências na recepção do sinal causadas por paredes, pessoas ou objetos, a solução seria trivial. Contudo, ambientes reais requerem a adoção de modelos empíricos, que geralmente utilizam medidas reais submetidas a um processo de calibração[1], de modo a obter um mapa de sinais (*radio map* ou *radio fingerprint*) mais representativo, contendo menos imprecisões. O processo de calibração é demorado e trabalhoso, sendo necessário coletar um conjunto expressivo de dados em vários horários dentro do ambiente.

Existem várias abordagens que visam reduzir a imprecisão da função de localização (cf. Seção 2). Contudo, essas soluções não contemplam de forma integrada a tolerância a falhas de PAs, nem uma implementação leve adequada às limitações da computação móvel. O impacto da ausência do sinal de um ponto de acesso devido a falha ou oscilações na visibilidade deve ser minimizado pelo sistema. Este tratamento exige recursos computacionais, que já são escassos em determinados dispositivos.

Este trabalho objetiva investigar soluções para o problema da localização de dispositivos móveis em ambientes fechados que atendam aos requisitos de custo, precisão e tolerância a falhas. Como preocupações adicionais, colocam-se a viabilidade de execução em dispositivos com restrições de recursos (processamento, comunicação, memória e energia) e o tempo gasto na execução da função de localização. Mais especificamente, avaliamos soluções baseadas em redes neurais (RN) e no algoritmo dos k-vizinhos mais próximos (k-VMP), as quais utilizam os sinais de rádio freqüência emitidos pelos pontos de acesso de uma rede infra-estruturada. Um cenário de salas e um dispositivo móvel foram utilizados para executar e avaliar diversas configurações comuns em ambiente reais.

As seções estão estruturadas da seguinte forma: inicialmente, na Seção 2, são comentados alguns trabalhos relacionados; na Seção 3 são discutidas questões de projeto, o tratamento de tolerância a falhas e uma abordagem leve para o serviço de localização, sendo também mostradas as duas técnicas acima referidas. Na Seção 4 são apresentados testes de desempenho realizados com base nas duas técnicas referidas e um exemplo de aplicação real é brevemente descrito. Finalmente, na Seção 5 são apresentadas as conclusões.

2. Trabalhos Relacionados

Algumas abordagens têm sido estudadas para tratar do problema da localização de dispositivos móveis em ambientes internos, dentre elas as baseadas em modelos determinísticos e probabilísticos. Os modelos determinísticos são pouco precisos em razão da influência dos obstáculos na propagação do sinal. Os modelos probabilísticos descrevem a dependência das propriedades do sinal observado em relação à localização e movimentação do terminal [Kontkanen et al. 2004], incluindo suposições subjetivas nesses ambientes. Na mesma direção da abordagem probabilística, existem algoritmos que fazem comparações com a base de dados dos sinais observados ou a utilizam para treinar uma rede. Essa técnica, chamada de análise de cenas, tem mostrado bons resultados. Ela faz um casamento (*pattern matching*) entre os valores das intensidades dos sinais obtidos

[1]A calibração objetiva gerar exemplos de localização com os associados sinais dos pontos de acesso (sPA) no formato: (x, y, sPA1, sPA2, ..., sPAn).

e os valores pré-armazenados obtidos por uma calibração. Dentre as técnicas que buscam aprender de uma melhor forma o mapa de sinais estão as redes neurais e a dos k-vizinhos mais próximos. Outras, podem ser encontradas em [Brunato e Battiti 2005].

Em [Rodriguez et al. 2004] é descrita uma implementação, em um dispositivo *hand-held*, de um sistema baseado em redes neurais, onde algumas técnicas de aprendizagem são testadas. A calibração envolve 154 pontos em uma área de 800m^2. Em [Brunato e Battiti 2005] são apresentadas comparações com outras abordagens e também uma implementação baseada em um *hand-held*. A área do teste realizado é de 750m^2, onde capturam-se 257 medidas. Ambos os trabalhos não se preocupam com a tolerância a falhas.

O trabalho descrito em [Prasithsangaree et al. 2002] avalia uma situação de falha em um PA no processo de localização. Usando o algoritmo dos k-VMP, quando o PA de sinal mais potente é desligado verifica-se um aumento de cerca de 3m no erro de localização, em relação ao erro em situações sem falhas, que é aproximadamente de 13m para 95 por cento dos casos. Diferentemente, em nossa abordagem, são feitas injeções de falhas em uma base de treinamento e avaliado o impacto dessa ação na eficiência das técnicas RN e k-VMP.

Um ponto importante a considerar em sistemas *indoor* de localização é o emprego de dispositivos com restrições de capacidade computacional. Em trabalhos como [Ito e Kawaguchi 2005] e [Bahl e Padmanabhan 2000] as propostas são implementadas com base em *notebooks*. Além disso, na maioria dos trabalhos existe uma dependência entre o dispositivo móvel e um servidor, normalmente encarregado de executar remotamente o algoritmo de localização. Dessa forma, o dispositivo móvel não só consome energia para efetuar a comunicação, mas também pode enfrentar problemas de desconexão com o servidor. Em nosso trabalho, uma implementação centrada num dispositivo móvel, com grandes restrições de processamento e memória, é apresentada.

Independentemente da abordagem empregada para o problema da localização baseada em potência de sinal, sabe-se que existe um limite na precisão, que se situa entre 2m e 12m [Ito e Kawaguchi 2005]. Há, portanto, uma incerteza na estimativa da posição que não pode ser eliminada nessas técnicas. Um problema ainda maior na questão da precisão ocorre quando acontecem falhas nos PAs, tópico que é investigado no presente artigo.

3. Requisitos e Técnicas de Suporte à Localização

Localização é uma informação fundamental para a implementação de aplicações sensíveis ao contexto. Essas aplicações, em função da necessidade de adaptabilidade, necessitam de graus variáveis de precisão nessa informação. Nessa seção, discutimos questões que afetam a eficiência de um sistema de localização, enfatizando o tratamento de falhas e uma abordagem leve para a implementação de um sistema de localização.

3.1. Questões de Projeto

O desenvolvimento de um sistema de localização compreende vários aspectos, que suscitam várias opções de projeto, as quais têm impacto sobre algumas de suas características de funcionamento, bem como sobre detalhes de implementação. Esses aspectos, opções e impactos estão sintetizados na tabela 1.

Tabela 1. Aspectos no projeto de um sistema de localização e seus impactos

Aspecto	Opções	Impactos
algoritmo de localização	comparação (k-VMP) ou rede (neural)	precisão, tempo de execução, recursos consumidos
resolução	área ou coordenadas	nível de detalhamento
execução do algoritmo	local ou remota	recursos locais, comunicação com servidor
consistência na detecção de falhas	tratamento da variação no sinal	localização errônea, precisão
limitações da computação móvel	em vários recursos (e.g., processador)	concorrência com outras aplicações, redução *throughput*
lidar com grandes ambientes	cenário único ou múltiplo	tratamento da base de dados
posicionamento dos PAs	política da instituição, melhor localização	capacidade de generalização

Nesse trabalho decidimos concentrar nossa avaliação nos dois principais algoritmos que usam mapa de sinais: o k-VMP e as redes neurais. A escolha do primeiro se deu por sua simplicidade de implementação e a do segundo por sua capacidade de treinamento, que potencialmente permite adaptar-se a situações de falha. Ambos utilizam um mesmo banco de dados gerado através de amostragens dos sinais dos PAs, realizadas em diferentes pontos de uma área pré-definida. A variação do sinal em diferentes momentos faz com que diversas amostragens sejam necessárias para enriquecer o mapa de sinais e, assim, formar os dados para o treinamento e teste do sistema.

Os dados obtidos como resposta no problema de localização podem ser de dois tipos: coordenadas (x, y), definindo a exata localização do usuário, ou por área, indicando, por exemplo, uma sala de um determinado prédio. A abordagem por coordenadas é interessante por atender a requisitos de aplicações envolvendo grandes salas, como em aeroportos ou museus. Contudo, sua implementação exige o uso de sensores especiais de alto custo.

Ainda sobre o espaço físico, a implementação do sistema pode ser influenciada pela sua utilização em grandes áreas ou andares de um prédio, quando diferentes subconjuntos de PAs podem ser visíveis. Assim, a questão do uso de uma banco de dados único ou particionado deve ser considerada. Por exemplo, se o dispositivo móvel for um *handheld*, pode ser impraticável o armazenamento nesse dispositivo de uma base única. Adicionalmente, mesmo no caso de um dispositivo móvel com maior capacidade de recursos, o tempo de pesquisa na base de dados pode ser razoavelmente longo, ocupando o processador e prejudicando a realização de outras tarefas. O particionamento do banco de dados por áreas, e a obtenção sob demanda de uma partição específica a um servidor central, evitaria esse problema. Uma solução adaptativa para a seleção da partição poderia basear-se no histórico de movimentos do usuário.

Outro ponto importante que afeta a precisão da localização é o posicionamento dos PAs. Por exemplo, desconsiderando-se a ocorrência de falhas, dois PAs muito próximos pouco ajudam a um algoritmo de localização. No caso, pode-se obter amostras ruins

que dificultam a obtenção da localização correta. Além disso, perde-se a capacidade de generalização pois o aprendizado é baseado em falsas interpretações.

3.2. Tolerando Falhas de PAs no Serviço de Localização

As redes sem fio estão propensas a uma série de problemas, tais como a perda de conectividade e falhas de infra-estrutura [Gandhi 2003]. Na maioria dos sistemas de localização, duas situações geralmente não são tratadas: PA com sinal muito fraco e que às vezes não é audível, e a total inoperância de um PA, que pode ser ocasionada por diversos motivos, tais como o desligamento do equipamento, o seu mal funcionamento, manutenção ou mesmo falta de energia. A ausência do sinal de um PA compromete o desempenho do sistema de localização, causando uma variação muito grande na sua precisão.

A ausência do sinal de um PA implica na falta de um dado de entrada no sistema. Em [Yu et al. 2006] métodos que lidam com situações de perda de entradas tentando substituí-las por valores tão próximos quanto possível aos valores originais são investigados. No entanto, devido à grande variabilidade e independência dos valores dos sinais de entrada ser uma característica inerente ao problema da localização aqui discutido, esses métodos não são aplicáveis, pois torna-se praticamente impossível a recomposição do valor de algum sinal perdido. Dessa forma, o ponto a ser investigado é se a função de localização tem a habilidade de reconhecer padrões, mesmo com as informações de entrada contendo ruídos ou ainda sendo apresentadas de forma incompleta, proporcionando assim uma solução com tolerância a falhas, robusta e com capacidade de generalização.

No caso do sistema de localização aqui proposto, utilizamos um processo de injeção de falhas para a geração de uma base de dados de referência (BD), a ser usada tanto no treinamento da rede neural, quanto na escolha dos pontos a serem utilizados pelo método k-VMP. Em qualquer caso, admitiremos somente falhas simples de PAs.

3.3. Abordagem Leve para o Sistema de Localização

Usualmente os trabalhos que lidam com o problema da localização focalizam somente na precisão da solução para a avaliação de seu desempenho. Assim, deixam de lado questões importantes para ambientes móveis, tais como os tempos de execução, a energia gasta no processo e a possibilidade de desconexão. O tempo de execução é influenciado tanto pelo algoritmo de localização quanto pela arquitetura do sistema de localização. Questões relativas ao tempo de execução do algoritmo serão discutidas na Seção 4.

Quanto à arquitetura, uma opção para o sistema de localização seria o dispositivo móvel obter os valores das intensidades dos sinais dos PAs e repassá-los a um servidor para o cálculo. No entanto, devido a restrições desses dispositivos móveis, como o uso de bateria, e ainda devido à possibilidade de desconexões no enlace, a utilização de comunicação intensiva não é uma prática adequada. Uma arquitetura que executa a função de localização localmente no cliente resolve este problema, desde que não gaste muito tempo de UCP e que também não tenha uma grande ocupação de memória. A solução local também apresenta vantagens em caso de falha no próprio servidor.

3.4. As Redes Neurais

A aplicação de redes neurais requer um processo de desenvolvimento envolvendo a preparação dos dados, a modelagem da rede neural e a análise dos dados [Yu et al. 2006].

Junto a isso, o ajuste de diversos parâmetros de treinamento e a implementação determinarão a qualidade da solução que, no caso, aplica-se para um serviço de localização de usuários de dispositivos móveis em ambientes internos.

O primeiro passo na modelagem de uma rede neural é a definição de sua arquitetura, que determina como os neurônios estão conectados. Este trabalho tem interesse nas redes diretas (*feedforward*), que não possuem laços de conexões (Figura 1). Outros tipos de arquitetura e conceitos básicos podem ser encontrados em [Kröse 1996].

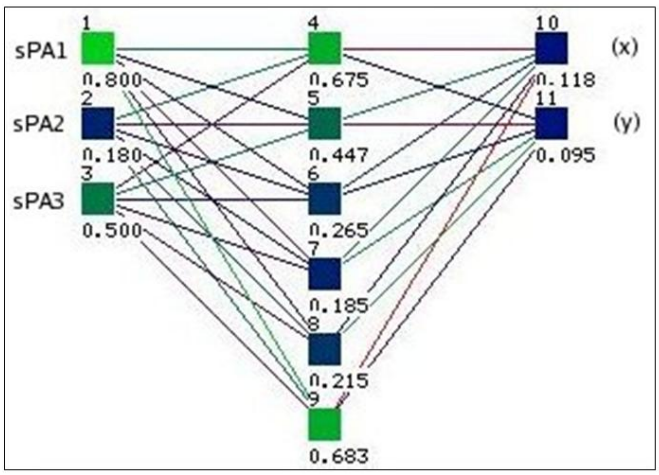

Figura 1. Exemplo de rede neural com topologia 3-6-2

Nessa arquitetura, um algoritmo utilizado para o treinamento (fase de aprendizagem, na qual são feitos ajustes nos pesos sinápticos e *thresholds*) é o de retropropagação (*backpropagation*). Após a apresentação de um exemplo à rede, é feito o cálculo de erro (saída da rede menos a saída desejada), que se propaga em direção aos neurônios de entrada, ajustando os valores dos pesos das sinapses.

Um dos parâmetros da rede é o número de ciclos que serve para ajustá-la, confirmar soluções e corrigir caminhos de soluções que possivelmente foram sobrescritos por um outro exemplo. No entanto, estimar um valor alto para o número de ciclos pode não melhorar o desempenho, ou, até pior, causar o *overtraining*. Nesse caso, a rede torna-se sintonizada em ruídos particulares, isto é, falsas propriedades do conjunto de treinamento. Outro caso que pode ocorrer é o *overfitting*. Ele também está relacionado com a capacidade de generalização da rede, ou seja, se ela pode resolver conjuntos de dados que estejam fora do conjunto de treinamento. Nesse caso, o modelo de aprendizado torna-se sintonizado com as características específicas do conjunto de treinamento.

O software empregado neste trabalho para criar e treinar as redes neurais foi o SNNS [Universidade de Stuttgart 2005]. A arquitetura da rede é *feedforward*, com treinamento em *backpropagation*. A taxa de aprendizado, usada para a modificação dos pesos nos nodos intermediários, foi $0,2$. A iniciação da rede utilizou pesos randomicamente distribuídos entre -1 e 1 e, após vários testes com os parâmetros da rede e com diferentes números de nós na camada oculta, a arquitetura 3-12-2 mostrou-se a com melhor desempenho. Pela característica do problema tratado ser não linear, foi utilizada a função de ativação sigmoidal. Para esse efeito, duas funções foram testadas, a logística (com saída

entre $[0, +1]$) e a tangente hiperbólica (com saída entre $[-1, +1]$), sendo a melhor precisão obtida através da tangente hiperbólica. O número de ciclos de treinamento utilizado nesse trabalho foi de 2000.

3.5. O Método dos k-Vizinhos mais Próximos

k-VMP é um método de classificação que, a partir da leitura dos sinais no ponto para o qual se deseja obter a localização, procura em sua base de amostras de referência as k mais próximas. O centróide obtido a partir dessas k amostras fornecerá a indicação da posição desejada. De acordo com [Brunato e Battiti 2005] o k-VMP é um método que apresenta baixos erros na estimativa da posição. Em [Prasithsangaree et al. 2002] são testadas variações desse método, concluindo que a seleção de 3 vizinhos (3-VMP) sem a inclusão de pesos apresenta o melhor resultado. Segundo experimentos mostrados em [Bahl e Padmanabhan 2000], três PAs são suficientes para obter a localização, não trazendo uma melhora significativa na precisão a adição de novos PAs.

4. Implementação e Testes

Os estudos realizados neste trabalho nos ajudam a definir os pontos fundamentais a serem testados na direção de um sistema de localização tolerante a falhas. Esta seção realiza simulações com RN e k-VMP e mostra uma implementação simples de uma aplicação de localização. O objetivo final é executar essa aplicação num dispositivo móvel com restrições na quantidade de memória e processamento, que seja tolerante a falhas de um PA por vez e obtenha a localização do dispositivo com uma boa precisão. Uma boa precisão média depende da aplicação e do ambiente, mas assume-se aqui que seja no máximo de 5m para tentar evitar que o serviço responda com uma sala diferente da que realmente se encontra. Isso atende a diversas aplicações sensíveis ao contexto, como por exemplo as hospitalares.

4.1. Ambiente de Execução dos Testes

A coleta dos dados para o experimento realizou-se em 11 salas do Instituto de Computação da UFF (ver figura 2). Em cada sala foram feitas medições nos quatro cantos e no centro, para cada medição foram capturadas 29 amostras durante 1min, e a cada 15s foi feita uma mudança de direção (Norte, Sul, Leste e Oeste). O processo incluiu 56 pontos de calibração e 5 PAs, dando um total de 1624 exemplos. Numa fase de preparação dos dados, os dados repetidos foram excluídos e separados na proporção $1/3$ para testes e $2/3$ para treinamento. O espaço de teste possui 25m de comprimento e 14m de largura.

A Figura 2 mostra a tela de execução da aplicação de localização num dispositivo *handheld* que será descrita na subseção 4.3. As salas F e G são na verdade corredores. Os quadrados pequenos representam os cinco PAs disponíveis para os testes, enquanto o círculo na sala ”C” mostra a localização atual do dispositivo móvel. Nos testes da subseção 4.2, utilizou-se um PC com processador Athlon de 2GHz e Sistema Operacional Linux com kernel 2.6.9.

4.2. Testes de Desempenho

Os testes realizados visaram avaliar as técnicas k-VMP e RN sob vários aspectos: tempo de execução, pegada (*footprint*: área de memória requerida para código e dados), precisão

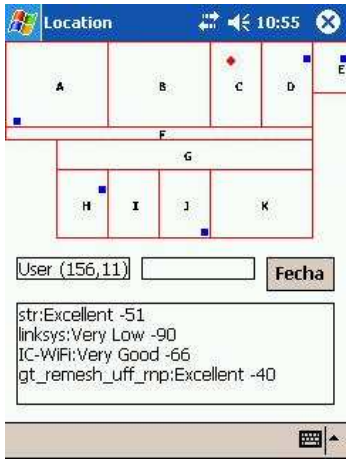

Figura 2. Tela da aplicação de localização

(em termos de coordenadas cartesianas) e tolerância a falhas. Para tanto, foram criadas quatro bases de dados a serem utilizadas nas fases distintas do processo de avaliação. Essas bases são divididas em duas classes: a primeira, de sufixo sf, não leva em consideração PAs falhos; a segunda, de sufixo cf, além dos mesmos exemplos anteriores, onde os PAs funcionam corretamente, contém casos de falhas simples, ou seja, situações em que o sinal de um dos PAs é perdido. Para efeito da fase de calibração, definimos duas bases, BDsf e BDcf. Na fase de testes utilizamos as bases Tsf e Tcf. A título de ilustração, para 3 PAs, o número de exemplos nas diversas bases de dados BDsf, BDcf, Tsf e Tcf, foi, respectivamente, 236, 907, 91 e 395; para 4 e 5 PAs esses números aumentam. Cada exemplo em uma dessas bases inclui o valor do sinal para até 5 PAs e o valor real das coordenadas (x,y) do ponto em questão.

Um teste realizado no *desktop* comparou o tempo de execução da RN com o k-VMP. Esse último algoritmo precisa processar uma base de dados (ordenação), que pode estar armazenada em memória ou em disco. Mesmo quando armazenada em disco, os exemplos são carregados para memória no momento da execução. A tabela 2 apresenta os resultados da média de 50 execuções de amostras retiradas das bases T, para o algoritmo k-VMP. Como esperado, os tempos de execução em situações sem falha são menores do que aqueles em situações com falha. Vale lembrar que num *handheld* os tempos de execução do k-VMP devem ser bem maiores, dada a diferença entre as plataformas computacionais. Considerando a ordem de grandeza desses tempos, esse aumento pode não causar diretamente um efeito perceptível ao usuário. Contudo, durante a execução do algoritmo outras funções essenciais poderão ser prejudicadas, e.g., recepção e envio de mensagens, o que poderá perturbar indiretamente funções que estiverem em realização, e.g., recepção de um vídeo. Por outro lado, como uma rede neural já treinada não necessita processar uma base dados para produzir sua resposta, os tempos de execução ficaram sempre abaixo dos $6\mu s$, o que reduz as possíveis interferências com outras funções da aplicação.

Um segundo ponto avaliado foi a quantidade de memória utilizada para a implementação dos métodos. Isso torna-se um ponto crítico quando o algoritmo de localização tem de executar em um *handheld*, como parte de um sistema maior de computação sensível ao contexto. No caso do k-VMP, a parte relevante é o tamanho da

Tabela 2. Tempo de execução dos k-VMP (em microsegundos)

	Somente memória		**Disco e memória**	
N° **PAs e** n° **exemplos**	sf	cf	sf	cf
3 PAs; BDsf=236 BDcf=897	123	562	934	3446
4 PAs; BDsf=236 BDcf=1137	130	779	1085	5134
5 PAs; BDsf=236 BDcf=1377	133	981	1212	6967

base de dados. No caso dessa aplicação, a base BDcf ocupou aproximadamente 57KB, o que já é um tamanho significativo, anda mais se considerarmos que em um ambiente real a área a ser ocupada pode ser muito maior do que os 350m^2 aqui assumidos. Uma área maior exige mais exemplos, e a solução pode apresentar problemas de escalabilidade. Já no caso da RN, como o programa final contém somente os pesos já ajustados (i.e., não há necessidade de BD), a pegada é muito pequena. No caso de nosso exemplo, a memória necessária foi cerca de 5KB, incluindo o código.

Os testes a seguir avaliam a precisão alcançada pelos dois métodos. As figuras 3-a e 4-a, e o primeiro grupo de colunas da figura 5 (BDsf, Tsf), mostram o resultado obtido com as técnicas, sem qualquer previsão para o tratamento de falhas. O impacto da ocorrência de uma falha em sistemas treinados sem essa hipótese é enorme (ver figuras 3-b, 4-b, e 5, colunas BDsf, Tcf). Sistemas treinados assumindo-se a hipótese de falhas, em funcionamento normal ou com um PA falho (ver figuras 3-c, 3-d, 4-c, 4-d e os dois últimos grupos de colunas da figura 5), apresentam resultados bastante razoáveis, comparáveis àqueles de sistemas treinados e executados sem falhas. Obviamente, há um preço a ser pago em tempo de execução e em pegada no sistema. Finalmente, deve ser notado que nesses experimentos o algoritmo k-VMP apresentou um desempenho médio superior ao da RN, em termos de precisão.

4.3. Aplicação de Localização

Com base nos estudos e resultados obtidos e pela complexidade de desenvolvimento das RN, esta subseção descreve brevemente a validação desta solução através de uma aplicação de localização.

O desenvolvimento da aplicação inicia com uma fase de preparação dos dados (como já visto anteriormente) que exclui os dados repetidos, injeta as falhas (gerando novos exemplos simulando a ausência de sinal) e, finalmente, separa $1/3$ de dados para teste e $2/3$ para treinamento. Após a rede estar treinada, utilizou-se o utilitário *snns2c* (do próprio simulador SNNS), que converte a rede para um programa em C. O ambiente EMbedded Visual C++ (EVC) serviu para gerar uma DLL a partir do código em C. A aplicação de localização, que chama constantemente a rede neural, foi implementada em C# usando o Visual Studio .NET com a biblioteca Compact Framework. A DLL com a rede neural resultou num código de 5KB. A aplicação de localização chama a dll a cada 2s e a tela do dispositivo é atualizada (Figura 2).

O dispositivo móvel utilizado é um Pocket da HP com processador Intel PXA255 de 400MHz, memória de 64MB SDRAM (56MB acessíveis pelo usuário), ROM de 32MB e Sistema Operacional Microsoft Mobile Pocket PC 2003. Os pontos de acessos são da Linksys, modelo WRT54G, padrão 802.11g, $2,4$GHz.

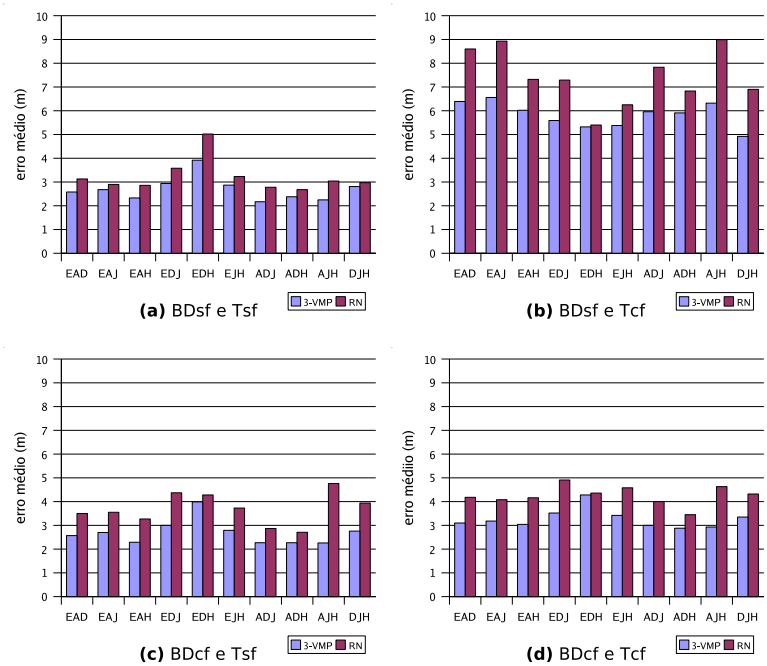

Figura 3. Média do erro para 3 PAs, usando 3-VMP e RN: (a) BD sf e testes sem falhas; (b) BD sf e testes com falhas; (c) BD cf e testes sem falhas; e, (d) BD cf e testes com falhas

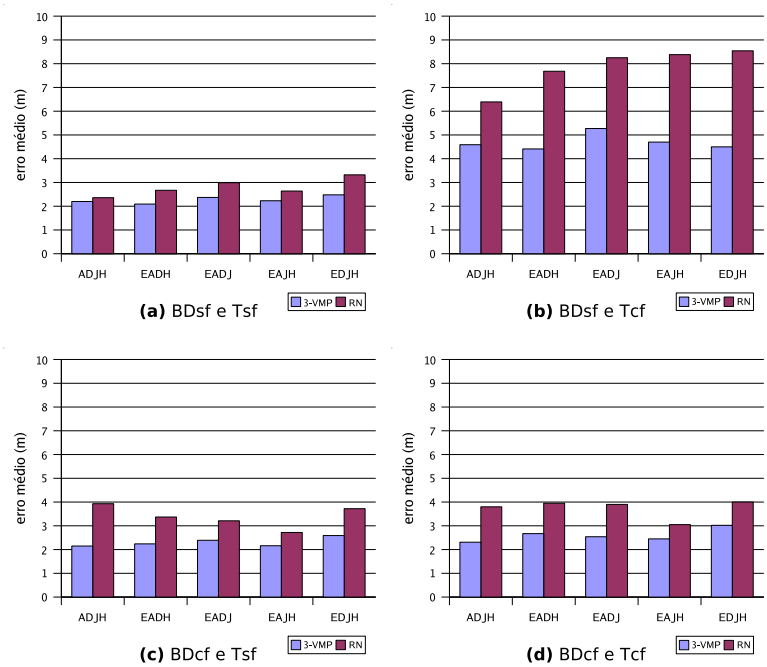

Figura 4. Média do erro para 4 PAs, usando 3-VMP e RN: (a) BD sf e testes sem falhas; (b) BD sf e testes com falhas; (c) BD cf e testes sem falhas; e, (d) BD cf e testes com falhas

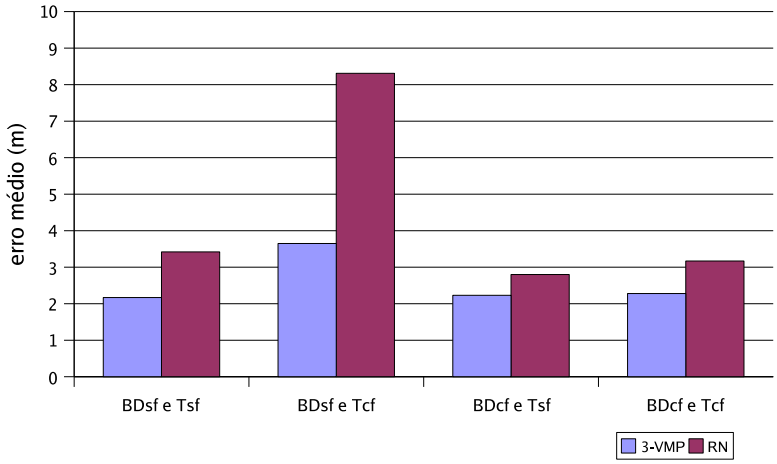

Figura 5. Média do erro para 5 PAs usando, 3-VMP e RN

O teste resumiu-se unicamente na movimentação entre as salas desligando-se apenas 1 PA. A permanência no mesmo local por no máximo 5s já converge para a sala correta, o que pode determinar uma espera de $(5+x)$s para lançar uma adaptação. Ainda mais testes são necessários para determinar a taxa de acerto, mas os resultados até o momento são animadores. Uma implementação com k-VMP também é interessante para ampliar os aspectos de comparação.

5. Conclusões

Este trabalho analisou duas técnicas de localização de dispositivos móveis em ambientes fechados. O principal objetivo foi contemplar requisitos de tolerância a falhas, com custo baixo, considerando as limitações de dispositivos móveis. Demonstrou-se que as duas técnicas permitem obter uma boa precisão em presença de falhas a partir de uma base de dados gerada com injeção de falhas.

Embora tenha apresentada precisão ligeiramente inferior, a técnica baseada em RN destacou-se em aspectos, tais como a pequena pegada requerida e o menor tempo de execução, características que podem facilitar sua utilização em dispositivos móveis com restrições de capacidade. Além disso, ela apresenta um diferencial no contexto de grandes espaços físicos, que poderiam ser divididos em sub-áreas, que poderiam ser tratadas pela simples mudança de pesos da rede. Nesse contexto, a utilização da técnica k-VMP implicaria numa troca de dados (base de dados) consideravelmente maior, gastando mais recursos de comunicação.

Agradecimentos

Os autores agradecem ao CNPq e a Faperj pelo financiamento parcial deste trabalho.

Referências

Bahl, P. e Padmanabhan, V. (2000). RADAR: An in-building RF-based user location and tracking system. *IEEE INFOCOM*, volume 2, pp. 775–784, Tel-Aviv, Israel.

Brunato, M. e Battiti, R. (2005). Statistical learning theory for location fingerprinting in wireless LANs. *Computer Networks*, 47(6):825–845.

Gandhi, R. (2003). Tolerance to access-point failures in dependable wireless local-area networks. *IEEE Workshop on Object-Oriented Real-Time Dependable Systems*, pp. 136–143, Nápoles, Itália.

Ito, S. e Kawaguchi, N. (2005). Bayesian based location estimation system using wireless LAN. *IEEE Workshop on Object-Oriented Real-Time Dependable Systems*, pp. 273–278, Kauai Island, HI, EUA.

Kontkanen, P., Myllymäki, P., Roos, T., Tirri, H., Valtonen, K., e Wettig, H. (2004). Chap. 11: Probabilistic methods for location estimation in wireless networks. Ganesh, R., Kota, S., Pahlavan, K., e Agustí, R., editores, *Emerging Location Aware Broadband Wireless Adhoc Networks*, pp. 173–187. Kluwer Academic Publishers.

Kröse, B. (1996). An introduction to neural networks. Universidade de Amsterdam. http://citeseer.ist.psu.edu/ose96introduction.html.

Prasithsangaree, P., Krishnamurthy, P., e Chrysanthis, P. K. (2002). On indoor position location with wireless LANs. *IEEE Symposium on Personal, Indoor, and Mobile Radio Communications*, pp. 273–278, Lisboa, Portugal.

Rodriguez, M., Favela, J., Martínez, E., e Muñoz, M. (2004). Location-aware access to hospital information and services. *IEEE Transactions on Information Technology in Biomedicine*, 8(4):448–455.

Universidade de Stuttgart (2005). Stuttgart neural network simulator. http://www-ra.informatik.uni-tuebingen.de/SNNS/.

Yu, L., Wang, S., e Lai, K. (2006). An integrated data preparation scheme for neural network data analysis. *IEEE Transactions on Knowledge and Data Engineering*, 18(2):217–230.