A Machine Learning Approach to Interpolating Indoors Trajectories

Daniel Carvalho, Daniel Sullivan, Rafael Almeida, Carlos Caminha

Programa de Pós Graduação em Informática Aplicada - Universidade de Fortaleza, Brazil
danielrotheia@edu.unifor.br
daniel.sullivan@edu.unifor.br
rafael.alm@edu.unifor.br
caminha@unifor.br

Abstract. In this article we propose a machine learning-based modeling to solve network overload problems caused by continuous monitoring of the trajectories of multiple tracked devices indoors. The proposed modeling was evaluated with hundreds of object coordinate locations tracked in three synthetic environments and one real environment. We show that it is possible to solve the problem of network overload increasing latency in sending data and predicting as server-side trajectories with *ensemble* models, such as the *Random Forest*, and using Artificial Neural Networks. We also show that it is possible to predict at least fifteen intermediate coordinates of the paths of the tracked objects with R^2 greater than 0.95.

CCS Concepts: • **Computing methodologies** → **Machine learning algorithms**.

Keywords: Data Mining, Internet of Things, Machine Learning

1. INTRODUÇÃO

A tecnologia tem estado cada vez mais presente no cotidiano das pessoas e o crescimento de aplicações de *IoT* (*Internet of Things*) é um dos aceleradores desse processo. Atualmente muitos objetos são capazes de se interconectar, transmitindo e recebendo dados da nuvem, possibilitando a comunicação entre pessoas, processos e ambientes [Andreev et al. 2015]. Essa popularização do *IoT* se deve especialmente ao intenso avanço tecnológico na área de sistemas embarcados, onde surgem inúmeras oportunidades a serem exploradas comercialmente, como no contexto das *smart homes*, ocorrendo uma inovação dos utensílios domésticos [Ponte et al. 2019], e no contexto da assistência pessoal, surgindo itens pessoais "inteligentes", contribuindo no conforto ou no auxílio de deficientes auditivos, por exemplo [Samie et al. 2016]. Em 2019 existiam cerca de 36 bilhões dispositivos de *IoT*, no entanto, com o crescimento de 12% anualmente desse número, é esperado um total de 125 bilhões em 2030 [Campbell 2019].

No contexto específico de monitoramento de objetos em ambientes fechados, empresas têm oferecido serviços de rastreamento que podem utilizar inúmeras tecnologias tais como, *Bluetooth Low Energy (BLE)* [Faragher and Harle 2014], *Ultra-Wideband (UWB)* [Ruiz and Granja 2017] e giroscópio e acelerômetro [Coronel et al. 2008]. Desta forma, podem existir centenas ou até milhares de objetos rastreados simultaneamente em um único ambiente, o que pode levar a uma sobrecarga da rede onde está acontecendo essa transmissão de dados [Lee et al. 2017; Li et al. 2019].

A solução natural para aliviar a carga de tráfego na rede é aumentar a cadência de envio de dados, em outras palavras, aumentar latência. Por exemplo, se os objetos enviam suas coordenadas pela rede a cada dois segundos, a latência poderia ser aumentada para intervalos de cinco segundos para proporcionar alívio no tráfego. No entanto, aumentar a latência tem como consequência direta a redução da precisão dos trajetos armazenados do lado do servidor, o que pode dificultar que análises futuras caracterizem padrões de movimentação que acontecem dentro desses ambientes.

Na literatura é possível encontrar inúmeros artigos que buscam interpolar/prever coordenadas para descobrir pontos em um trajeto [Wiest et al. 2012; Hunter et al. 2009; Pecher et al. 2016]. Esses artigos se concentram em prever trajetórias em ambientes abertos, sendo os objetos rastreados normalmente veículos que perdem o sinal do *GPS* em túneis ou outras áreas que restringem a comunicação via satélite. Dentre as técnicas mais utilizadas podemos citar a utilização de interpolação[Liu et al. 2012] assumindo pontos médios, ou também aplicações com o filtro de Kalman em conjunto com modelos *Constant Turn Rate and Acceleration (CTRA)* [Mancini 2021] como estimadores de trajetória. A eficácia dessas técnicas se dá especialmente porque só há necessidade de prever trajetórias a nível macro, por exemplo, não é um problema se o trajeto interpolado passar por cima de uma esquina, o mais importante é saber por quais ruas o veículo passou.

Apesar da eficácia dessas técnicas, em ambientes fechados, é maior a exigência pela compreensão de micro padrões de movimentação. No caso específico de um supermercado, utilizar interpolação linear implicaria em trajetos previstos cruzando gôndolas, o que poderia confundir a análise posterior das trajetórias dos objetos/pessoas rastreados. Por exemplo, não seria precisa a resposta para a seguintes questões: Por quais seções ou produtos um determinado funcionário, que utiliza uma pulseira rastreável, passou? Quantos metros quadrados foram limpos com um aspirador de pó rastreado?

Neste artigo propomos uma modelagem de Aprendizado de Máquina (AM) para o problema de interpolar coordenadas de objetos rastreados em ambientes fechados. Utilizaremos dados reais coletados por pesquisadores da Universidade de Guelph [Kennedy et al. 2019]. Para permitir uma avaliação mais completa da modelagem proposta, também serão utilizados dados sintéticos gerados para três ambientes distintos. Temos como objetivo responder as seguintes questões de pesquisa:

—PQ1 - A modelagem utilizada seria capaz de prever trajetos que desviam de obstáculos em ambientes fechados?
—PQ2 - Quanto a latência pode ser aumentada sem perda de desempenho dos algoritmos de Aprendizado de Máquina?

Este artigo está organizado da seguinte maneira: Na Seção 2 serão apresentados alguns artigos relacionados e como pretendemos contribuir com este artigo. Na Seção 3 será mostrado como foram construídos os ambientes sintéticos e como ocorreu o processo de geração de dados. A Seção 4 apresenta em detalhes a modelagem utilizada para os modelos de AM. Na Seção 5 serão apresentados os resultados desta pesquisa e serão respondidas as perguntas de pesquisa. Na Seção 6 serão feitas as considerações finais, considerando os resultados alcançados.

2. ESTADO DA ARTE

Uma das soluções mais utilizadas para prever pontos intermediários é o método de Interpolação Linear [Akima 1970]. Esse método constrói uma função contínua a partir de dados discretos conectando dois pontos interpolados [Meijering 2002]. [Wu et al. 2020] afirma que a Interpolação Linear é um método que possui diversas variações e é utilizado para solucionar problemas em diversas áreas como visão computacional [Upchurch et al. 2017], fotografia digital [Petschnigg et al. 2004], computação gráfica [Joblove and Greenberg 1978] e calibração de imagens [Li et al. 2008]. Em seu artigo é apresentado um método que melhora a performance da Interpolação Linear, além disso é descrito um método para avaliar a qualidade do interpolador [Wu et al. 2020].

Em 1960, Kalman desenvolveu uma solução recursiva para o problema de filtragem linear de dados discretos [Kalman 1960]. Seu método ficou bastante famoso e foi utilizado em diversas aplicações [Vikranth et al. 2016; Patel and Thakore 2013; Seng et al. 2016]. Com essa popularização o algoritmo passou a ser chamado de filtro de Kalman (FK) em homenagem a seu criador.

[Lam et al. 2018] utiliza o filtro de Kalman combinado com um modelo de AM, *O-SVM (Optimized Support Vector Machine)* com a finalidade de corrigir coordenadas em uma amostra de dados coletados.

Os autores aplicaram o filtro de Kalman para suavizar ruídos nas trajetórias. Em uma segunda etapa, já no servidor, foi utilizado o modelo *O-SVM*, treinado nestes dados higienizados para corrigi-los. Por fim, este método foi comparado com outros já conhecidos, *CoreLocation Framework (CL)*, *Open ALTBeacon Standard (ALTPLM)*, *Linear Regression (LRM)*, *Non-Linear Regression (PLRM)*. O que se obteve foi um erro médio inferior a estes outros métodos.

[Li et al. 2018] realiza o mesmo procedimento inicial, utiliza o *FK* para suavizar os dados provindos de rastreadores *BLE* para em seguida ser aplicado o *PSO-BPNN (Back Propagation Neural Network optimized by Particle Swarm Optimization)*, reposicionando assim a coordenada que será enviada para o servidor.

[Hirakawa et al. 2018] desenvolvem um método baseado em aprendizagem por reforço para preencher coordenadas faltantes em trajetos de animais na natureza. Em teoria o *GPS (Global Positioning System)* deveria registrá-las a cada minuto, porém por razões variadas, isso não ocorria. Então foi construído um espaço-recompensa baseado nas preferências ambientais da espécie estudada, uma ave marinha. Nesse método, os traçados feitos entre pontos de preferência são retas, a fim de tornar a previsão mais suave, visto que estas aves podem tomar caminhos indiretos.

Este artigo avança o estado da arte ao solucionar o problema de sobrecarga de rede provocado por inúmeros dispositivos rastreadores tentando acessá-la simultaneamente. Tem-se como objetivo reduzir a quantidade de dados enviados pela rede, utilizando modelos de Aprendizado de Máquina para interpolar coordenadas das trajetórias do lado do servidor.

3. CONJUNTO DE DADOS

Para realização desta pesquisa são necessários dados de localização em duas dimensões com um intervalo de tempo entre os pontos onde o conjunto representa a movimentação de um objeto em um ambiente fechado. Foi criado um gerador de dados utilizando a *engine* de desenvolvimento *UNITY* [Kucera et al. 2018]. Essa *engine* foi utilizada para modelar ambientes e gerar trajetórias de objetos dentro destes. Foi criada uma entidade que representa um objeto se movimentando e outra que aleatoriamente seleciona novos destinos para esse objeto. A cada novo destino, o objeto rastreado realiza um novo caminho ótimo em relação a distância percorrida dentro do ambiente. Foi acoplado um rastreador para retornar a posição do objeto rastreado a cada quadro por segundo do simulador e registrar sua coordenada X e Y em um arquivo. Foram criados três ambientes, denominados Ambiente 1, Ambiente 2 e Ambiente 3, com obstáculos posicionados manualmente.

Foi utilizado também um conjunto com dados reais, obtido em [Kennedy et al. 2019]. Esse conjunto de dados foi populado a partir do rastreado da locomoção de indivíduos dentro de um escritório durante vinte dias. Cada indivíduo rastreado foi identificado unicamente e foram utilizados dados de apenas uma pessoa, resultando em 17.050 registros. Para realizar o rastreamento, foram utilizados *Raspberry Pi's* e rastreadores *BLE*. O dado foi gravado com o uso do *software Beaconpi* [Kennedy et al. 2018]. Esse ambiente será chamado de Ambiente 4 neste artigo.

Os quatro ambientes podem ser observados na Figura 1. Nela, cada ponto azul representa um local onde o objeto esteve durante seu rastreio. Os itens A, C, E e G ilustram os obstáculos dos ambientes, já os itens B, D, F e H ilustram as coordenadas rastreadas nesses ambientes. Os itens A e B correspondem ao Ambiente 1, os itens C e D ao Ambiente 2 ,os itens E e F ao Ambiente 3 e, por fim, os itens G e H representam o Ambiente 4. O processo de simulação de trajetos em ambientes fechados pode ser visualizado no link *https://youtu.be/pDGFbHTDNkE*.

4. METODOLOGIA

Com o objetivo de responder as questões de pesquisa, serão utilizados dois modelos de previsão baseados em algoritmos de aprendizado de máquina (*Random Forest* e Rede Neural Artificial - *RNA*),

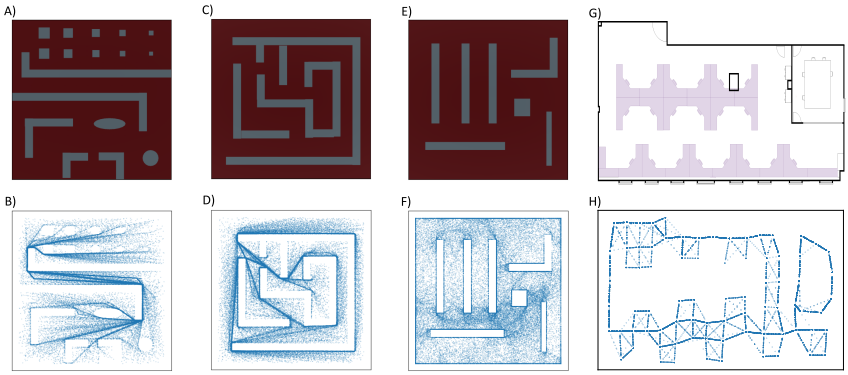


Fig. 1. Ambientes utilizados na pesquisa.

além disso o método de interpolação linear será utilizado como caso base para se comparar a eficiência das previsões em ambientes fechados onde é necessário que se tenha precisão da rota feita pelos dispositivos. A qualidade da previsão será comparada em um ambiente real com diversos obstáculos no caminho dos dispositivos.

Em seguida será simulado o efeito que a variação no tempo de latência poderá causar na qualidade das rotas previstas. Isso será feito variando a quantidade de pontos a serem previstos entre um ponto inicial e final.

4.1 Modelagem de *features*

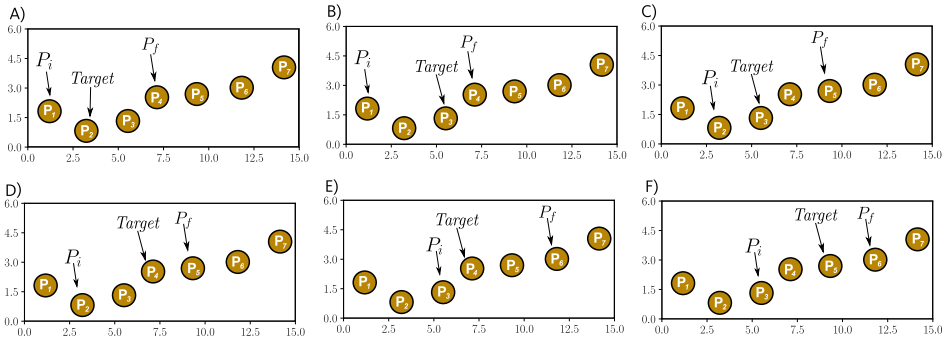
As *features* foram extraídas dos dados descritos na Seção 3 e todos os algoritmos utilizados neste artigo utilizam a mesma modelagem onde cada exemplo possui as seguintes *features*:

—1 - Ponto de início do trajeto ou de um trecho do trajeto(P_i);
—2 - Ponto final do trajeto ou de um trecho do trajeto(P_f);
—3 - Tempo relativo em que foi registrado o ponto final (T_f). O valor de T_f é calculado somando a quantidade de pontos até o P_f multiplicado pela latência. É importante citar que o tempo de P_i é um valor referência, então é sempre 0, não sendo necessário assim utilizá-lo como *feature*;
—4 - Tempo relativo em que se quer prever o ponto intermediário (T_n) sendo n o indicador de ordem cronológica do ponto. Por exemplo, se é desejado prever apenas um ponto entre P_i e P_f, então existirá uma observação com T_1 (tempo em que o primeiro ponto a ser previsto foi ou seria registrado). Caso queira se prever mais pontos, então teremos mais observações uma com T_2, outra com T_3 até por fim chegar ao último ponto a ser previsto T_n.

Os *targets* de cada observação são as coordenadas X e Y de cada ponto localizado entre P_i e P_f. Além disso, para que sejam construídos vários exemplos, foi utilizado o conceito de janela deslizante que está ilustrado na Figura 2. Nessa imagem, cada círculo amarelo representa um ponto (que tem as coordenadas X, Y, e o tempo que foi registrado) d é um parâmetro que indica a quantidade de pontos intermediários entre P_i e P_f(para o exemplo $d = 2$). Todas as janelas são de tamanho 3, pois são compostas por P_i,P_f e um $Target$.

A partir da Figura 2 A é criado o primeiro exemplo que tem $P_i = P_1$, $P_f = P_4$ e $Target = P_2$. Em seguida, Figura 2 B é feito um $shift$ no $target$ e um novo exemplo é formado. Como não há mais possibilidades de construção de exemplo com essa janela, pois o $target$ passou por todos os pontos entre P_i e P_f, então a janela é deslizada para a direita, ou seja, $P_i = P_2$, $P_f = P_5$ e $Target = P_3$, dessa forma um novo exemplo é criado em C. O processo continua até que não seja mais possível criar novos exemplos. Todas as observações possíveis estão ilustradas na tabela presente na Figura 2,

onde XP_i, YP_i são as coordenadas do ponto inicial, XP_f, YP_f e TP_f são as coordenadas e o tempo de registro do ponto final, TP_{int} e $Target$ são o tempo de registro e as coordenadas do ponto a ser previsto.



Exemplos

	XP_i	YP_i	XP_f	YP_f	TP_f	TP_{int}	$Target$	
A)	1	2	7.3	2.5	3	1	3	0.7
B)	1	2	7.3	2.5	3	2	5.5	1.2
C)	3	0.7	9.5	2.7	3	1	5.5	1.2
D)	3	0.7	9.5	2.7	3	2	7.3	2.5
E)	5.5	1.2	12	3	3	1	7.3	2.5
F)	5.5	1.2	12	3	3	2	9.5	2.7

Fig. 2. Modelagem dos exemplos para os modelos de Aprendizado de Máquina.

4.2 Modelagem e execução dos algoritmos de aprendizado de máquina

O algoritmo *Random Forest* [Breiman 2001] foi modelado utilizando o *RandomForestRegressor* em conjunto ao *MultiOutputRegressor*, ambos componentes do módulo *scikit-learn* [Pedregosa et al. 2011], usando apenas a parametrização *default*, sem qualquer tipo de ponderação [Ponte et al. 2020]. O algoritmo foi executado dez vezes, devido a sua característica estocástica, e o dado foi dividido de forma que 80% fosse usado para treino, e 20% para teste.

A RNA utilizada foi do tipo *MultiLayer Perceptron (MLP)* [Silva et al. 2016; Bonaccorso 2017]. Utilizou-se a implementação do *tensorflow* [Abadi et al. 2016] para realizar todos os testes deste artigo.

A rede neural construída possui quatro camadas, a primeira com 2000 neurônios, a segunda com 500 neurônios, a terceira com 20 neurônios e 2 neurônios na última. Com relação as funções de ativação, nas três primeiras, foi utilizado *ReLu* e na ultima, tangente hiperbólica. O otimizador utilizado foi o *Adam* com *learning rate* de 0.0001.

Similarmente ao que foi feito no algoritmo *Random Forest*, foram realizadas dez execuções para todos os testes, dividindo os dados em 64% para treino, 20% para validação e 16% para teste. Além disso, foram utilizadas 50 épocas para a fase de treinamento.

Foram coletadas duas métricas para cada combinação de d, ambiente e modelo utilizado. São elas o R^2 e *MAE (Mean Absolute Error)*. Todo código necessário para realizar os experimentos descritos neste artigo, está disponível em *https://github.com/ddrc1/indoors-trajectory-prediction*.

Estes resultados são comparados e discutidos na sessão seguinte (5).

5. RESULTADOS

A Figura 3 ilustra o resultado obtido no primeiro experimento, onde os modelos de previsão foram executados com dados de um ambiente real. É possível observar três trechos de rota onde a interpolação

linear não desvia dos obstáculos, comportamento já esperado. Entretanto os modelos de aprendizado de máquina foram capazes de fazer curvas, desviando de obstáculos, respondendo então a $PQ1$. Em outras palavras, o que se observa é a capacidade dos modelos de AM de aprender onde estão os obstáculos no ambiente. Esse aprendizado se dá sem a utilização de qualquer informação da planta do ambiente, utilizando apenas as coordenadas dos objetos rastreados.

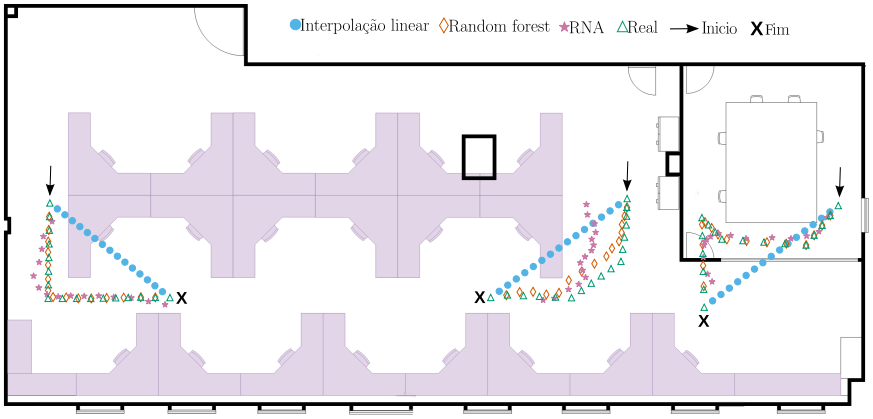


Fig. 3. Amostragens de rotas interpoladas dentro do Ambiente 4.

Na Figura 4 é ilustrada a variação de $d = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$ simulando ocasiões de baixa e alta latência. Observa-se que há uma perda de desempenho à medida que d aumenta. Isso ocorre devido ao aumento da distância entre as coordenadas interpoladas e consequentemente a maior dificuldade de prever a trajetória realizada. Entretanto, mesmo para valores de $d = 15$, os modelos de aprendizado de máquina mantém $R^2 > 0.95$. Esse resultado responde a $PQ2$, mostrando que é possível aumentar a latência até 15 vezes mantendo boas previsões. Quanto ao método de Interpolação Linear, é possível observar que os valores das métricas pioram de maneira mais intensa na medida em que o valor de d é aumenta. Isso ocorre fundamentalmente devido à característica do método de assumir um trajeto em linha reta entre os dois pontos interpolados. Quando d aumenta, é mais improvável que o trajeto tenha sido feito em linha reta. Pode-se perceber que nos itens G e H da Figura 4, referentes ao Ambiente 4, os valores da interpolação linear para o eixo X apresentaram um desempenho melhor, isto se deve à grande quantidade de caminhos horizontais retilíneos no ambiente.

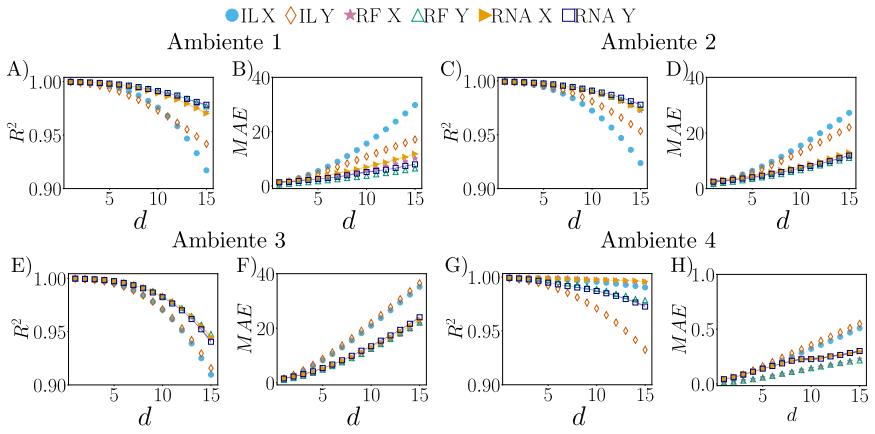


Fig. 4. Comparação de métricas para diferentes valores de d nos ambientes

6. CONCLUSÃO

Neste artigo mostramos que é factível resolver o problema de sobrecarga de rede provocado por uma grande quantidade de dispositivos de *IoT* enviando coordenadas simultaneamente para a nuvem. Ao modelar esse problema como um problema de Aprendizado de Máquina, mostramos que é possível prever com, boa taxa de acerto, as trajetórias realizadas por objetos rastreados em ambientes fechados. A modelagem proposta possibilita que algoritmos de Aprendizado de Máquina prevejam trajetórias que desviam de obstáculos ou passam por portas e corredores. Foi observado ainda que essa modelagem permite prever até quinze coordenadas intermediárias de uma trajetória com $R^2 > 0.95$.

Essas previsões levantam a possibilidade de aumentar a latência de coleta desses dados, fazendo a previsão dos trajetos realizados do lado do servidor. Para isso seria necessário que dados fossem coletados com a latência mínima durante um pequeno período. A partir daí os dados coletados podem ser utilizados para alimentar modelos de Aprendizado de Máquina, permitindo que estes aprendam como as trajetórias acontecem no ambiente monitorado. Nossos resultados mostram que, indiretamente, os algoritmos aprendem a localização de portas, paredes e obstáculos, mesmo sem qualquer acesso a planta do ambiente.

Como trabalhos futuros destacamos a necessidade de estudar o impacto que a quantidade de dados tem na qualidade das previsões. Devido os modelos de Redes Neurais Artificiais serem conhecidos por alcançar melhores resultados quando treinados com grandes volumes de dados, acreditamos que seria interessante observar o impacto que essa quantidade de dados tem nos resultados. Acreditamos também que adaptar as arquiteturas de *RNA* utilizadas recentemente na literatura para previsões de series temporais pode agregar aos resultados, especialmente utilizando o mecanismo de *Attention* e *Transformer* [Miao et al. 2020].

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16). pp. 265–283, 2016.

Akima, H. A new method of interpolation and smooth curve fitting based on local procedures. Journal of the ACM (JACM) 17 (4): 589–602, 1970.

Andreev, S., Galinina, O., Pyattaev, A., Gerasimenko, M., Tirronen, T., Torsner, J., Sachs, J., Dohler, M., and Koucheryavy, Y. Understanding the iot connectivity landscape: a contemporary m2m radio technology roadmap. IEEE Communications Magazine 53 (9): 32–40, 2015.

Bonaccorso, G. Machine learning algorithms. Packt Publishing Ltd, 2017.

Breiman, L. Random forests. Machine learning 45 (1): 5–32, 2001.

Campbell, M. Smart edge: The effects of shifting the center of data gravity out of the cloud. Computer 52 (12): 99–102, 2019.

Coronel, P., Furrer, S., Schott, W., and Weiss, B. Indoor location tracking using inertial navigation sensors and radio beacons. In The Internet of Things. Springer, pp. 325–340, 2008.

Faragher, R. and Harle, R. An analysis of the accuracy of bluetooth low energy for indoor positioning applications. In Proceedings of the 27th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2014). pp. 201–210, 2014.

Hirakawa, T., Yamashita, T., Tamaki, T., Fujiyoshi, H., Umezu, Y., Takeuchi, I., Matsumoto, S., and Yoda, K. Can ai predict animal movements? filling gaps in animal trajectories using inverse reinforcement learning. Ecosphere 9 (10): e02447, 2018.

Hunter, T., Herring, R., Abbeel, P., and Bayen, A. Path and travel time inference from gps probe vehicle data. NIPS Analyzing Networks and Learning with Graphs 12 (1): 2, 2009.

Joblove, G. H. and Greenberg, D. Color spaces for computer graphics. In Proceedings of the 5th annual conference on Computer graphics and interactive techniques. pp. 20–25, 1978.

Kalman, R. E. A new approach to linear filtering and prediction problems, 1960.

Kennedy, B., Taylor, G. W., and Spachos, P. Ble beacon based patient tracking in smart care facilities. In 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). IEEE, pp. 439–441, 2018.

Kennedy, M., Spachos, P., and Taylor, G. W. Ble beacon indoor localization dataset, 2019.

Kucera, E., Haffner, O., and Leskovskỳ, R. Interactive and virtual/mixed reality applications for mechatronics education developed in unity engine. In 2018 Cybernetics & Informatics (K&I). IEEE, pp. 1–5, 2018.

Lam, C. H., Ng, P. C., and She, J. Improved distance estimation with ble beacon using kalman filter and svm. In 2018 IEEE International Conference on Communications (ICC). IEEE, pp. 1–6, 2018.

Lee, S. K., Bae, M., and Kim, H. Future of iot networks: A survey. Applied Sciences 7 (10): 1072, 2017.

Li, G., Geng, E., Ye, Z., Xu, Y., Lin, J., and Pang, Y. Indoor positioning algorithm based on the improved rssi distance model. Sensors 18 (9): 2820, 2018.

Li, X., Liu, Y., Ji, H., Zhang, H., and Leung, V. C. Optimizing resources allocation for fog computing-based internet of things networks. IEEE Access vol. 7, pp. 64907–64922, 2019.

Li, Z., Shi, Y., Wang, C., and Wang, Y. Accurate calibration method for a structured light system. Optical Engineering 47 (5): 053604, 2008.

Liu, S., Liu, C., Luo, Q., Ni, L. M., and Krishnan, R. Calibrating large scale vehicle trajectory data. In 2012 IEEE 13th International Conference on Mobile Data Management. pp. 222–231, 2012.

Mancini, A. Vehicle path prediction for safety enhancement of autonomous driving. Ph.D. thesis, Politecnico di Torino, 2021.

Meijering, E. A chronology of interpolation: from ancient astronomy to modern signal and image processing. Proceedings of the IEEE 90 (3): 319–342, 2002.

Miao, H., Cheng, G., Gao, C., Zhang, P., and Yan, Y. Transformer-based online ctc/attention end-to-end speech recognition architecture. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 6084–6088, 2020.

Patel, H. A. and Thakore, D. G. Moving object tracking using kalman filter. International Journal of Computer Science and Mobile Computing 2 (4): 326–332, 2013.

Pecher, P., Hunter, M., and Fujimoto, R. Data-driven vehicle trajectory prediction. In Proceedings of the 2016 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation. SIGSIM-PADS '16. Association for Computing Machinery, New York, NY, USA, pp. 13–22, 2016.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. the Journal of machine Learning research vol. 12, pp. 2825–2830, 2011.

Petschnigg, G., Szeliski, R., Agrawala, M., Cohen, M., Hoppe, H., and Toyama, K. Digital photography with flash and no-flash image pairs. ACM transactions on graphics (TOG) 23 (3): 664–672, 2004.

Ponte, C., Caminha, C., Bomfim, R., Moreira, R., and Furtado, V. A temporal clustering algorithm for achieving the trade-off between the user experience and the equipment economy in the context of iot. In 2019 8th Brazilian Conference on Intelligent Systems (BRACIS). IEEE, pp. 604–609, 2019.

Ponte, C., Caminha, C., and Furtado, V. Otimização de florestas aleatórias através de ponderação de folhas em árvore de regressão. In Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional. SBC, pp. 698–708, 2020.

Ruiz, A. R. J. and Granja, F. S. Comparing ubisense, bespoon, and decawave uwb location systems: Indoor performance analysis. IEEE Transactions on instrumentation and Measurement 66 (8): 2106–2117, 2017.

Samie, F., Bauer, L., and Henkel, J. Iot technologies for embedded computing: A survey. In 2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS). IEEE, pp. 1–10, 2016.

Seng, K.-Y., Chen, Y., Chai, K. M. A., Wang, T., Fun, D. C. Y., Teo, Y. S., Tan, P. M. S., Ang, W. H., and Lee, J. K. W. Tracking body core temperature in military thermal environments: An extended kalman filter approach. In 2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN). IEEE, pp. 296–299, 2016.

Silva, I. D., Spatti, D. H., and Flauzino, R. A. Redes Neurais Artificiais para engenharia e ciências aplicadas. Arliber Editora Ltda., 2016.

Upchurch, P., Gardner, J., Pleiss, G., Pless, R., Snavely, N., Bala, K., and Weinberger, K. Deep feature interpolation for image content changes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

Vikranth, S., Sudheesh, P., and Jayakumar, M. Nonlinear tracking of target submarine using extended kalman filter (ekf). In International Symposium on Security in Computing and Communication. Springer, pp. 258–268, 2016.

Wiest, J., Hoffken, M., Kresel, U., and Dietmayer, K. Probabilistic trajectory prediction with gaussian mixture models. In 2012 IEEE Intelligent Vehicles Symposium. IEEE, 2012.

Wu, Y.-C., Hsu, K.-L., Liu, Y., Hong, C.-Y., Chow, C.-W., Yeh, C.-H., Liao, X.-L., Lin, K.-H., and Chen, Y.-Y. Using linear interpolation to reduce the training samples for regression based visible light positioning system. IEEE Photonics Journal 12 (2): 1–5, 2020.