Universidade Federal da Paraíba | Departamento de Informática
Volume 5 Número 1 - Jun//2021

Revista Comunicações em Informática

ISSN: 2595-0622

http://periodicos.ufpb.br/index.php/cei

Editorial

Temos a satisfação de trazer mais um número da Comunicações em Informática! Nele apresentamos cinco trabalhos que trazem a Computação e suas interfaces em diferentes contextos. Em um período de sobrecarga de trabalho remoto, ainda devido às restrições da pandemia de COVID-19, meu profundo agradecimento aos revisores destes trabalhos, que cederam parte de seu tempo para colaborar conosco.

O primeiro artigo, “Um novo olhar sobre o Fortran: da computação sequencial à paralela”, aborda uma linguagem de programação bastante conhecida e considera “antiga”: o Fortran. A partir de um olhar sobre o percurso evolutivo desta linguagem, o autor apresenta como ela vem se adaptando às novas necessidades computacionais, como a computação paralela e a programação de múltiplos núcleos em GPUs.

A representação de expressões faciais na tradução automática de Libras é abordada no artigo “Uma proposta de tradutor automático com análise de emoções de textos em português brasileiro para Libras”. Neste trabalho, os autores trazem um protótipo que mostrou a possibilidade de adaptação da *suite* Vlibras em realizar a representação de emoções em seu avatar.

O terceiro artigo, “Desenvolvimento de *dashboards* para análise de características geográficas e climáticas do estado da Paraíba (PB) - projeto plataforma multi-mapa PB” trás o uso de *dashboards* como um meio de representação e divulgação de informações por meio da visualização interativa. Os autores apresentam dois estudos de caso nos quais usuários sem experiência de programação utilizaram a solução desenvolvida para gerar *dashboards* com informações geográficas e climáticas do estado da Paraíba.

Os outros dois trabalhos desta edição fazem parte de uma chamada temática de Computação nas Engenharias, apresentados a seguir. Esta seção temática teve como objetivo expor esta forte conexão da Computação com as Engenharias e foi coordenada por uma união de pesquisadores destas duas áreas, aos quais agradeço a dedicação. Estes dois trabalhos estão descritos a seguir.

A todos que tem colaborado com esta revista: leitores, revisores, editores associados e editora universitária, agradeço pelo apoio e suporte. Convido-os, assim, a navegarem pelos temas e conteúdos dos trabalhos desta edição!

Liliane S. Machado
Editora-chefe

Seção Temática: A Computação nas Engenharias

A Ciência da Computação é uma área transversal que proporciona aplicações em diversas outras áreas do conhecimento, provavelmente em todas. A revista Comunicações em Informática apresenta a seção temática “A Computação nas Engenharias”, apresentando trabalhos em que o estado da arte é aplicado em aplicações práticas da engenharia.

Nesse contexto, o trabalho “Self-organized UAV flocking based only on relative range and bearing” apresenta uma aplicação de algoritmos de colaboração multiagentes na área de engenharia aeronáutica e robótica, em busca de melhores resultados para a formação de enxames de veículos aéreos autônomo. O trabalho apresenta métodos em que múltiplos veículos tomam decisões de forma independente e não centralizada, objetivando evitar colisões.

Uma outra aplicação que envolve Engenharia e Computação, mas agora em salas de cinema, é descrita no trabalho “Transmissão de áudio em tempo real sobre redes wireless IEEE 802.11 com foco em acessibilidade para o cinema digital”. Neste trabalho, são utilizadas técnicas de redes de computadores e de transmissão sem fio para a transmissão de áudio em tempo real de forma síncrona aos telespectadores. Por se tratar de um cenário complexo, pois há uma limitação de atraso máximo permitido entre a transmissão e a recepção pelos telespectadores, este trabalho apresenta uma solução capaz de transmitir áudio em tempo real sobre redes Wi-Fi entre um servidor e múltiplos clientes e ainda é proposto um algoritmo adaptativo que busca ajustar o buffer de playback com o propósito de atenuar o jitter da rede.

É com grande satisfação que apresentamos estes trabalhos e agradecemos aos autores envolvidos e aos demais que submeteram seus trabalhos para esta seção temática.

Alisson Vasconcelos de Brito
Cleonilson Protásio de Souza
Abel Cavalcante Lima Filho

Editores da seção temática

Um Novo Olhar Sobre o Fortran: da Computação Sequencial à Paralela

Omar Andres Carmona Cortes

Departamento de Computação (DComp)
Instituto Federal de Educação, Ciência e Tecnologia do Maranhão (IFMA)
omar@ifma.edu.br

Resumo: A Linguagem Fortran foi uma das primeiras linguagens de programação tendo sido criada ainda na década de 50. Quase 60 anos depois, a linguagem continua sua evolução e não há indícios que sua história irá se encerrar. Nesse contexto, este artigo tem como objetivo apresentar como a linguagem tem evoluído e como suas características têm permitido que a linguagem continue em uso, iniciando com a computação sequencial, perpassando pela sua adaptação à computação paralela, chegando ao seu auge na programação de muitos núcleos em GPUs. Assim, a ideia do artigo é lançar uma nova perspectiva sobre a linguagem, em especial para o público de ciência da computação, fazendo com que seu real potencial seja visível a toda comunidade.

Palavras-chave: *Fortran; computação paralela; evolução.*

1. Introdução

O título deste artigo pode parecer estranho para muitas pessoas, em especial para aqueles da área de ciência da computação que provavelmente só ouviram falar da linguagem com uma parte da história da computação.

A verdade é que devido à pressão do mercado de trabalho, os cursos de graduação em computação (Ciência da Computação, Engenharia da Computação, Sistemas de Informação e Engenharia de Software), pelo menos no Brasil, tem que dar ênfase às linguagens mais utilizadas no mercado, tais como C, C++, Java, Julia, Python, dentre outras, e seus respectivos ambientes de programação. Não há nada de errado nisso já que o egresso certamente buscará trabalho na área. No entanto, algumas oportunidades podem ser perdidas. Por exemplo, a *Indeed* (https://www.indeed.com/), versão americana, que é uma página de busca por emprego, retorna 104 ofertas de emprego com a busca "*Fortran Programmer*".

O fato é que muito embora pessoas sempre fizessem predições declarando a morte da linguagem Fortran, ela continua sendo utilizada por engenheiros e cientistas [1], podendo ainda ser fonte de trabalho como já apresentado. No contexto científico alguns trabalhos podem ser citados. Por exemplo, Maan e Sing [2] convertem código Fortran para SymPy que executa operações matemáticas simbólicas. Hsu e Asanza [3] investigam como a otimização paralela afeta a portabilidade de programas em Fortran. Barros e Casquillo [4] apresentam como combinar CPLEX e Fortran 90 na resolução de problemas lineares. E, Johnson et al. [5] apresenta uma interface em Fortran 2003 para utilizar bibliotecas numéricas implementadas em C e C++.

Nesse contexto, este artigo investiga o estado da arte da linguagem Fortran de modo a desmistificar sua morte, mostrando que a linguagem ainda vive e com grandes perspectivas de continuar vivendo, especialmente em se tratando de computação paralela.

Assim, este artigo está dividido da seguinte maneira: a Seção 2 mostra a evolução histórica do Fortran e quais características a tornam interessante em cada versão; a Seção 3 apresenta o estado-da-arte em programação sequencial e em programação paralela; a Seção 4 mostra uma discussão sobre como evoluiu o estado da arte ; finalmente, a Seção 5 apresenta as conclusões desta pesquisa.

2. A Linguagem Fortran

Fortran é o acrônimo para *FORmula TRANslator*, sendo que o impulso para o seu nascimento foi o surgimento do IBM 704 em 1954. No mesmo ano foi publicado o relatório intitulado "*The IBM Mathematical FORmula TRANslating System: FORTRAN*", sendo que o primeiro compilador, chamado de Fortran I, foi lançado no ano de 1957 [6].

Seguindo sua evolução, em 1966 surgiu o Fortran IV tornando-se um padrão que ficou conhecido como Fortran 66. Anos depois, em 1978, foi lançada uma revisão conhecida como Fortran 77 tornando-se o padrão conhecido como ANSI Fortran.

Em 1990, o Fortran 77 foi anexado ao Fortran 90 tornando-se um subconjunto deste [7]. Algumas características consideradas obsoletas foram retiradas da linguagem e foram inseridas características que antes só poderiam ser utilizadas em linguagens como o C, como por exemplo: (i) operações diretas com matrizes; (ii) aumentou-se a facilidade de se trabalhar com a computação numérica; (iii) possibilitou-se a definição de novos tipos pelos usuários; (iv) foram adicionadas novas estruturas de controle de fluxo, possibilitando a programação mais estruturada; (v) foram adicionadas facilidades na definição de módulos e procedimentos; e, (vi) incluiu-se a recursão, a alocação dinâmica e o uso de ponteiros.

Interessantemente, a partir do Fortran 90 também foi possível emular os principais conceitos de orientação a objeto encontrados em C++ tais como classes, herança, encapsulamento e sobrecarga [8][9]. Essas características foram incorporadas efetivamente no Fortran 2003 [10], sendo que também foi incorporada a linguagem a interoperabilidade com a linguagem C [6]. Na versão 2008 foi dada uma perspectiva mais de computação paralela à linguagem introduzindo características de programação paralela diretamente na linguagem, ou seja, sem a necessidade de extensões.

Até o presente momento o padrão é o Fortran 2018 cujas melhorias foram o aumento da interoperabilidade com a linguagem C e adição de mais características de paralelismo. Espera-se que após 2020 seja publicada a revisão do padrão Fortran 2018. O novo padrão ainda chamado de 202x está previsto para ser lançado em 2021 [11].

A seguir apresenta-se o estado da arte da linguagem Fortran indicando como a linguagem evoluiu naturalmente da versão sequencial para a versão paralela.

3. Evolução do Fortran: da programação sequencial à paralela

Como já mencionado, a partir do Fortran 90, características antes restritas a programadores de Linguagem C ficaram disponíveis para programadores Fortran. Com esse padrão deu-se a possibilidade de se trabalhar com matrizes dinâmicas. Além disso, a produtividade também foi aumentada introduzindo operações matriciais do tipo C=*A*+*B*, na qual *A*, *B* e *C* são matrizes. Em outras palavras, em linguagens como C ou mesmo Fortran 77, por exemplo, esse tipo de operação tinha que ser feita através de laços embutidos, ou seja, iterando-se dentro das matrizes executando a operação elemento por elemento. A partir do Fortran 90 a operação ocorre em apenas uma linha.

A partir do padrão 2003 incorporou-se a programação orientada a objetos em Fortran. Na verdade, introduziu-se a extensão de tipos definidos, *type extension*, que acabou por permitir a utilização de uma das características mais importantes da programação orientada a objetos, a herança. No mesmo padrão também se introduziram a utilização de polimorfismo, interfaces e encapsulamento, dando a linguagem a possibilidade de programação orientada a objetos e seu principal benefício, o reuso.

No padrão de 2008 foi adicionado a linguagem o que se chama de *coarray*, que é uma estrutura de dados que pode ser compartilhada entre diferentes *images*, sendo *images* cópias idênticas do executável. Essa característica permite que a linguagem se aproveite da computação paralela através de uma técnica de paralelismo chamada SPMD, do inglês *Single Program Multiple Data*. Em outras palavras, o objetivo era tornar a programação paralela de fácil implementação, fazendo com que várias *images* pudessem trabalham em diferentes dados ou partes de um mesmo *coarray*, fornecendo ao programador a possibilidade de trabalhar com programação de memória compartilhada. Além disso, foi introduzida também a estrutura *DO...CONCURRENT* que especifica laços sem interdependência, que é particularmente interessante para compiladores de paralelização automática.

O Fortran 2018 é uma extensão menor do Fortran 2008 [12]. Como já mencionado, suas principais melhorias se limitam a estender o trabalho com *coarrays* e tornar a linguagem mais interoperável com a linguagem C. A principal modificação foi a adição de *Teams*. A ideia principal de *Teams* é que conjuntos separados de *images* possam ser executadas de forma independente [12], fazendo possível programar no modelo de programação paralela chamado de *Multiple Program Multiple Data* (MPMD).

A seguir serão detalhadas as extensões paralelas do Fortran desde as primeiras extensões paralelas para linguagens sequenciais até a utilização de GPUs.

Extensões Paralelas

A partir da década de 1980 houve o crescimento exponencial da informática, o que levou a quebra do paradigma de von *Neumman* fazendo surgir às primeiras arquiteturas paralelas. Assim, a década seguinte, de 1990, foi bastante promissora para a linguagem Fortran, sendo que nessa década surgiram as primeiras extensões paralelas de linguagens sequenciais: o *Parallel Virtual Machie* (PVM) [13] e o *Message Passing Interface* (MPI) [14]. Ambas as extensões permitiram ao programador Fortran a programação paralela através de um paradigma chamado de programação por passagem de mensagem, sendo adequado para arquiteturas com memória distribuída. A grande vantagem advinda do uso das extensões paralelas de linguagens sequenciais é a curva de aprendizado, já que o programador não precisa aprender uma linguagem do zero e sim estender seu conhecimento.

Em 1993 surge o High Performance Fortran (HPF) [15] cuja meta era fornecer a possibilidade de trabalhar com paralelismo de dados, provendo como principal vantagem a identificação e paralelização de dados de forma automática pelo compilador através da instrução *forall*, além de proporcionar o controle da distribuição dos dados pelo programador. O objetivo do HPF era um modelo de programação comum que pudesse ser executado eficientemente em todos os tipos de arquiteturas paralelas [16].

Em seguida lançou-se o HPF+ [17] em 1998, que permitia um maior controle sobre a distribuição dos dados, podendo inclusive especificar a carga de trabalho em laços irregulares. Além disso, o HPF+ permitia a utilização de parâmetros para especificar redundância de dados, facilitando o trabalho do compilador. Ainda na década de 90, por volta de 1995, surgiu o Vienna Fortran [18] cujo objetivo é estender o Fortran para que programadores pudessem escrever código para arquiteturas com memória distribuída. De fato, o compilador Vienna Fortran é um sistema *source-to-source* que transforma programas em Fortran 95/HPF+ para Fortran 90/MPI [19].

Mais ou menos ao mesmo tempo que se começaram a popularizar os processadores com múltiplos núcleos no início dos anos 2000, surgiu também o OpenMP [20], que é uma API para programação paralela com memória compartilhada. Coincidentemente ou não, as arquiteturas com múltiplos núcleo são arquiteturas com memória compartilhada.

Já na década de 2010 surge o CUDA Fortran [21], que acaba por ganhar mais popularidade dado as *General Purpose Graphic Processor Unit* (GPGPU), que são placas gráficas que podem ser utilizadas para computação de propósito geral. CUDA Fortran é

essencialmente um Fortran 90 com algumas extensões que permite ao programador aproveitar o poder das GPUs em suas aplicações [22]. Finalmente, surge o OpenACC [23] que, de forma similar ao OpenMP, é uma biblioteca de diretivas que permite a paralelização automática de código em Fortran em GPUs.

4. Discussão

Até o Fortran 77, as operações com vetores e matrizes eram feitas de forma similar a outras linguagens, ou seja, dentro de laços. A partir do Fortran 90 as operações passaram a ser diretas, inclusive incluindo a capacidade de se trabalhar com subconjuntos dentro de uma mesma matriz ou vetor. Essa característica foi bastante visionária e viria a aparecer em linguagens como R, Julia e Python (usando *NumPy*), que atualmente são as principais linguagens *open source* para ciência dos dados e aprendizagem de máquina. Obviamente essas linguagens são mais produtivas que Fortran. Porém, devido à grande quantidade de código disponível para supercomputadores ainda estar em Fortran e de ter usuários para esses códigos, a possibilidade dessas novas linguagens substituírem o Fortran é pequena em curto e médio prazo.

Com o surgimento das arquiteturas paralelas, como já foi mencionado, Fortran foi uma das primeiras linguagens a receber extensões paralelas de linguagens sequenciais: PVM e MPI. Dessas duas extensões o MPI foi a que permaneceu.

De fato, a década de 90 foi rica em soluções para computação paralela, sendo uma delas o HPF, que apesar de inicialmente promissor se mostrou ineficiente. O problema com o HPF era que estruturas de dados irregulares podiam gerar uma sobrecarga na comunicação entre processos ou gerar um desbalanceamento de carga que inevitavelmente leva a um desempenho pobre da aplicação paralela [24]. Para solucionar esse problema foi criado o HPF+ e em seguida o Vienna Fortran para flexibilizar a programação paralela dada a quantidade de arquiteturas paralelas que vinham surgindo na época.

Os problemas criados pelas versões de HPF e Viena Fortran acabaram por ficar para trás quando surgiu o OpenMP, sendo que seu ponto forte é ser baseado em diretivas de compilação (*#pragma*), ou seja, em instruções especiais que indicam ao compilador quais regiões do programa devem ser paralelizadas pelo compilador. Assim, o programador também não tem que aprender uma nova linguagem e sim onde colocar as diretivas e algumas funções novas, diminuindo consideravelmente a curva de aprendizagem.

Ao mesmo tempo que soluções paralelas iam sendo desenvolvidas, a própria linguagem ia tendo sua capacidade de executar código paralelo sendo melhorada a cada versão. No Fortran 2018, por exemplo, *images* podem se comunicar em uma única direção, isto é, se uma chamada e feita da *image* A para a *image* B, não é necessário ter uma chamada correspondente em B para A como deve ser feito em MPI, o que simplifica consideravelmente a programação. Assim, possivelmente, devido às melhorias e facilidades de programação paralela nativa do Fortran aliado à facilidade disponibilizada pelo OpenMP na programação de memória compartilhada, levaram ao abandono das outras iniciativas como o HPF.

Por outro lado, o MPI permaneceu no mercado. Acredita-se que o MPI persistiu devido ao controle do paralelismo que a biblioteca fornece ao programador em sistemas de memória distribuída com passagem de mensagem. Inclusive o *gcc* [25], que é o compilador mais conhecido no mundo *Unix-alike*, tem suporte tanto para linguagem Fortran quanto para OpenMP sem a necessidade de instalação de bibliotecas adicionais. A partir da versão 8, o *gcc* possui inclusive suporte ao OpenACC para paralelização automática em GPU.

Para finalizar tem-se o CUDA Fortran, cuja capacidade de paralelismo ainda tem muito a ser explorado por dois motivos. O primeiro pela popularidade das GPGPUs que hoje estão disponíveis até em dispositivos móveis como celulares e o segundo pela construção dos supercomputadores baseados em GPUs. A Tabela 1 apresenta os seis supercomputadores baseados em GPU e sua posição no Top 500 [26] dos supercomputadores em junho de 2020.

Tabela 1 – Supercomputadores baseados em GPU.

Nome	Posição	Núcleos CUDA
Summit	2º	2.414.592
Sierra	3º	1.572.489
HPC5	6º	669.769
Selene	7º	272.800
Marconi-100	9º	347.776
Piz Daint	10º	387.872

Como se pode observar, dentre os 10 computadores mais rápidos do mundo, seis são baseados em GPU. Além dessa evolução do Fortran, a quantidade de código e bibliotecas científicas disponíveis na linguagem ainda a fazem atrativa tanto na computação científica quanto na engenharia e na supercomputação. Em outras palavras, ainda existe um poder computacional significativo a ser explorado, seja pela capacidade de computação paralela nativa da linguagem, seja pelo OpenMP, pelo MPI, pelo CUDA Fortran ou pelo OpenACC.

5. Conclusões

Este artigo apresentou uma evolução histórica do estado da arte da linguagem Fortran. Mostraram-se as características que foram evoluídas de modo a passar naturalmente da computação sequencial para computação paralela, sendo que o auge de novas tecnologias para a linguagem Fortran foi a década de 90, na qual surgiram tecnologias como o MPI, HPF e Vienna Fortran.

Atualmente a computação paralela em Fortran está focada na utilização de cinco tecnologias: computação paralela nativa do Fortran 2018, OpenMP, MPI, OpenACC e CUDA Fortran. Dentre essas destaca-se a possibilidade de exploração de OpenACC e CUDA Fortran já que, dentre as arquiteturas paralelas mais rápidas do mundo, seis são baseadas em GPU. Em outras palavras, Fortran ainda tem um longo caminho pela frente.

Bibliografia

[1] Kolakowski, N. (2019) Five Programming Languages that Refuse to Die, Dice, https://insights.dice.com/2019/10/03/5-programming-languages-refuse-die/, Acesso em 16/08/202.

[2] Maan, N.; Singh, P. (2020) Parsing C and Fortran code to SymPy Expressions, 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2020, pp. 592-597, DOI: 10.1109/Confluence47617.2020.9057820.

[3] Hsu, A.; Asanza, D. N. ; Schoonover, J. A.; Jibben, Z.; Carlson, N. N.; Robey, R. (2018) Performance Portability Challenges for Fortran Applications, IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC), Dallas, TX, USA, 2018, pp. 47-58, DOI: 10.1109/P3HPC.2018.00008.

[4] Barros, M.; Casquilho, M. (2019) Linear Programming with CPLEX: An Illustrative Application Over the Internet CPLEX in Fortran 90, 14th Iberian Conference on Information Systems and Technologies (CISTI), Coimbra, Portugal, pp. 1-6, DOI: 10.23919/CISTI.2019.8760632.

[5] Johnson, S. R.; Prokopenko, A.; Evans, K. J. (2020) Automated Fortran–C++ Bindings for Large-Scale Scientific Applications, *Computing in Science & Engineering*, vol. 22, no. 5, pp. 84-94, DOI: 10.1109/MCSE.2019.2924204.

[6] Sebesta, R. W. (2018) Conceitos de Linguagens de Programação, 11ª edição, Porto Alegre: Bookman.

[7] Reid, J. (1992) The advantages of Fortran 90, *Computing*, 48:219-238.

[8] Decyk, V., K.; Norton, C., D.; Szymansky. B., K. (1999) How to Express C++ Concepts in Fortran 90, Jet Propulsion Laboratory – California Institute of Tecnology, Passadena: California, Rensisselear Polytechnic Institute, Troy: N.Y.

[9] Decyk, V. K.; Norton, C. D..; Szymansky. B., K. (1999) Introduction to Object-Oriented Concepts Using Fortran, Jet Propulsion Laboratory – California Institute of Tecnology, Passadena: California AND Departament of Computer Science – Rensisselear Polytechnic Institute, Troy: N.Y.

[10] Metcalf, M.; Reid, J.; Cohen, M. (2004) Fortran 95/2003 Explained, 3e. Oxford University Press, Oxford, England.

[11] ISO, JTC1/SC22/WG5, Fortran 202x, https://wg5-fortran.org/f202x.html, Acesso em 16/08/2020.

[12] Metcalf, M.; Reid, J. and Cohen, M. (2018) Modern Fortran Explained, Oxford Press: UK.

[13] Index for PVM3 Library, http://www.netlib.org/pvm3/, Acesso em 16/08/2020

[14] Open MPI: Open Source High Performance ComputingHPF, https://www.open-mpi.org/, Acesso em 15/08/2020

[15] High Performance Fortran, http://hpff.rice.edu/index.htm, Acesso em 10/08/2020.

[16] Kennedy, K.; Koelbel, C.; Zima, H. (2007) The rise and fall of High Performance Fortran: an historical object lesson. In Proceedings of the third ACM SIGPLAN conference on History of programming languages (HOPL III). Association for Computing Machinery, New York, NY, USA, DOI: 10.1145/1238844.1238851.

[17] Optimizing HPF for Advanced Applications, http://www.par.univie.ac.at/project/hpf+/, Acesso em 01/08/2020.

[18] VFC, http://www.par.univie.ac.at/project/hpf+/vfcs.html, Acesso em 10/08/2020

[19] Benker, S. (1999) The Vienna Fortran Compiler, Scientific Programming, IOS Press, pp. 1058-9244.

[20] OpenMP: Enabling HPC since 1997, https://www.openmp.org/, Acesso em 10/08/2020

[21] Cuda Fortran, https://developer.nvidia.com/cuda-fortran, Acesso em 16/08/2020

[22] Ruetsch, G.; Fatica, M. (2014) CUDA Fortran for Scientistis and Engineers: Best Practices for Efficient CUDA Fortran Programming, Morgan Kaufman: San Francisco-USA.

[23] OpenACC: more science less programming, https://www.openacc.org/, Acesso 05/07/2020

[24] Ryne, R. D.; Habib, S. (1996) Beam Dynamics Calculation and Particle Tracking Using Massively Parallel Processors, Particle Accelerators, vol. 55, pp. 365-374.

[25] GCC the GNU Compiler, https://gcc.gnu.org/, Acesso em 01/08/2020

[26] Top 500, Lista Junho de 2020, https://www.top500.org/lists/top500/2020/06/, Acesso em 16/08/2020.

Uma Proposta de Tradutor Automático com Análise de Emoções de Textos em Português Brasileiro para Libras

Vinícius Matheus Veríssimo da Silva, Rostand Costa

LAVID/UFPB
Centro de Informática – Universidade Federal da Paraíba
{vinicius.matheus, rostand}@lavid.ufpb.br

Resumo: As áreas tecnológicas apresentam alternativas que auxiliam a comunidade surda na aquisição e compartilhamento de informações, como é o caso dos tradutores automáticos. Embora tal abordagem seja bem-vista pela comunidade surda, ela ainda sofre críticas frente a falta de humanidade dessas soluções. Diante disso, esse estudo tem por objetivo realizar a adaptação do tradutor automático de Português Brasileiro para Libras da Suíte VLibras, de modo que o avatar 3D da plataforma performe as emoções contidas na tradução, através de expressões faciais. Apesar de inicial, o estudo traz resultados promissores, os quais levam a crer que o uso do protótipo aqui desenvolvido pode vir a minimizar as críticas usuais por parte da comunidade surda ao utilizarem tradutores automáticos.

Palavras-chave: *acessibilidade; tradução automática; análise de emoções*.

1. Introdução

No Brasil, por volta de 10 milhões de pessoas possuem algum grau de perda auditiva grave (cerca de 5% da população), onde 2,7 milhões possuem um grau de surdez absoluta, conforme censo de 2010 do IBGE. Assim, uma fatia considerável da população brasileira está suscetível à total ou parcial privação de informações, visto que as mesmas são voltadas predominantemente para a população ouvinte.

Uma vez que a população surda tem como língua prioritária a língua de sinais, e estar em um meio onde o que predomina são as línguas orais faz com que isso acarrete resultados negativos em seus processos de desenvolvimento cognitivo e de aprendizado [2, 3], o que impacta diretamente em como uma pessoa surda se integra e participa de sua comunidade [4], uma vez que a sua capacidade de absorver e difundir informações é limitada.

Postas tais dificuldades, várias estratégias aparecem como forma de diminuir as barreiras que as pessoas surdas enfrentam, como a presença de intérpretes humanos em diversos ambientes. Porém, há casos onde a utilização de um intérprete humano não é possível, como na internet. Diante disso, uma solução é a utilização de tradutores automáticos de conteúdo multimídia em língua oral para língua de sinais, como os explorados nos trabalhos de Oliveira et al. [6], Stoll et al. [7] e Araújo [8].

O uso de tradutores automáticos é uma alternativa viável para que a população surda tenha maior acesso a informações, porém, alguns problemas são apontados no estudo de Rocha e Melgaço [5]. Ao realizarem uma pesquisa de opinião com usuários de aplicativos de tradução automática de Poruguês Brasileiro para Libras (Língua Brasileira de Sinais), os autores constaram que há aceitação dessas tecnologias por parte da comunidade surda, mas um ponto amplamente criticado foi a falta de expressões faciais e corporais adequadas no avatar 3D utilizado nessas ferramentas.

Nesse sentido, o objetivo deste trabalho é realizar a adaptação de um tradutor automático de língua oral para língua de sinais, visando possibilitar a expressão de emoções no avatar 3D, por meio de expressões faciais no tradutor da Suíte VLibras [8].

O restante do trabalho está organizado da seguinte forma: na Seção 2 será apresentada a metodologia de adaptação do tradutor automático; na Seção 3 serão apresentados os resultados dessa adaptação; na Seção 4 será realizada uma discussão acerca do trabalho apresentado; e, por fim, a Seção 5 trará a conclusão do trabalho.

2. Metodologia

O processo de adaptação do tradutor automático será realizado sobre o tradutor automático da Suíte VLibras. A Suíte VLibras é um conjunto de ferramentas para a tradução de conteúdo multimídia de Português Brasileiro para Libras, sinalizada por um avatar 3D. Esse sistema é composto de programas para computadores pessoais, aplicativos móveis, *plug-in web*, serviços de nuvem, entre outros. Além disso, ele pode ser aplicado em diversas esferas, sendo ele atualmente utilizado até mesmo por alguns *sites* públicos governamentais.

De forma mais específica, ressalta-se que a adaptação do tradutor automático será realizada no serviço de tradução em nuvem e na versão de *desktop* do sinalizador automático da Suíte VLibras. Processo que será descrito nas próximas seções.

2.1 Serviço de Tradução com Emoção

Para a criação do serviço de tradução com emoção, será utilizado o modelo de análise de emoção em textos Português Brasileiro proveniente do trabalho de Veríssimo e Costa [9]. O presente estudo utilizará o modelo que obteve melhores resultados computacionais. No trabalho de Veríssimo e Costa [9], os autores exploraram o aumento de uma base de dados de manchetes jornalísticas, e a criação de uma rede neural capaz de fazer a análise de emoções de textos em Português Brasileiro. As emoções compreendidas pelo

estudo são: alegria, raiva, tristeza, medo, nojo, surpresa e a ausência de emoção (neutra).

No que se refere ao tradutor automático, o serviço de tradução original da Suíte VLibras realiza a tradução de texto em Português Brasileiro para glosa Libras, uma forma de notação escrita das Línguas de Sinais. No que tange à adaptação desse serviço, será na forma de um *middleware*, que realizará as requisições de tradução para o serviço original e para o analisador de emoções simultaneamente.

Nesse serviço de tradução adaptada, o analisador de emoções e o tradutor automático original estão sendo executados em paralelo e em ambientes diferentes. O mesmo texto de entrada será enviado para ambos os algoritmos, ou seja, a análise de emoções e tradução serão realizadas sobre o mesmo texto em Português Brasileiro. Essa simultaneidade é positiva, pois a informação do texto não será modificada, sendo modificada apenas a sua forma de transmissão.

A Figura 1 mostra um exemplo genérico do fluxo de execução do serviço de tradução adaptado. É possível visualizar três módulos: o módulo de Classificação de Emoções, responsável por realizar a análise de emoções do texto; o módulo de Tradução, responsável por realizar a tradução do texto em glosa; e o Tradutor com Emoções, que se utiliza do resultado dos outros módulos para criar uma glosa com a informação de emoção.

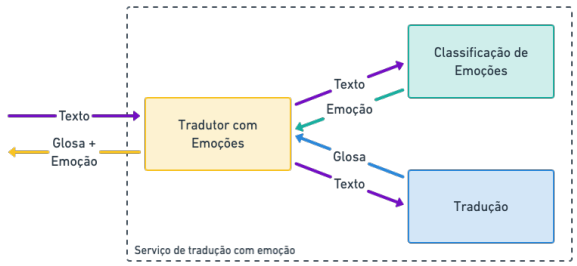

Figura 1. Exemplo genérico do fluxo do serviço de tradução adaptado.

Para a adição da emoção da glosa foi utilizada uma abordagem baseada em marcadores, como os utilizados nas linguagens de marcação HTML e XML. Os marcadores contendo os caracteres < e > indicam o início da emoção na porção do texto, e os marcadores contendo os caracteres <\ e >, denotam o fim da emoção.

A Figura 2 apresenta um exemplo de utilização dos marcadores na glosa. O exemplo “a)” é uma forma simples de utilização, com apenas uma emoção para todo o texto. Por outro lado, o exemplo “b)” apresenta uma forma mais complexa de utilização, com múltiplos marcadores, trazendo assim, múltiplas emoções em um mesmo texto.

a) <EMOÇÃO> TEXTO </EMOÇÃO>

b) <EMOÇÃO_1> TEXTO <EMOÇÃO_2> TEXTO </EMOÇÃO_2> TEXTO </EMOÇÃO_1>

Figura 2. Exemplo de utilização dos marcadores de emoções na glosa

Para essa adaptação, apenas o exemplo a) da Figura 2 será abordado, uma vez que o analisador de emoções de Veríssimo e Costa [9] realiza a identificação de uma emoção por texto.

O serviço de tradução adaptado foi criado utilizando a linguagem Python, e o *micro-framework* Flask para prover uma API de acesso. Além disso, foram utilizadas bibliotecas para auxiliar no paralelismo das operações, e o *framework* fastai, para o uso dos modelos de análise de emoções. O repositório do projeto pode ser acessado por esse link.

2.2 Sinalizador Automático com Emoções

A adaptação do sinalizador automático *desktop* da Suíte VLibras necessitará de modificação no código-fonte, que foi cedido pelo LAVID (Laboratório de Vídeo Digital), o qual desenvolve o projeto.

O sinalizador automático funciona realizando uma requisição ao serviço de tradução automática para que seja feita a tradução do texto em Português Brasileiro para glosa Libras. Para o sinalizador, cada palavra da glosa se refere a um *bundle* de animação, que descreve como o avatar 3D deve performar o sinal. Além disso, uma legenda acompanha a sinalização sincronizadamente.

A adaptação do sinalizador automático deverá interpretar a nova glosa fornecida pelo serviço de tradução adaptado que foi apresentado na Seção 2.1, de modo que ele deve apresentar a emoção contida na nova glosa, por meio das expressões faciais do avatar 3D, durante a sinalização das demais palavras da glosa. Além disso, essa informação de emoção não será apresentada na legenda que acompanha a sinalização, sendo apresentada apenas no avatar.

Em relação à definição das expressões faciais do avatar 3D, para cada uma das emoções abordadas, foi utilizado como base o estudo de Silva et al. [10], que descreve as expressões faciais da Libras na notação do sistema FACS, primariamente definido por Ekman e Friesen [11].

No FACS, as expressões faciais são definidas em termos de unidade de ação (AUs - *action units*), que representam os movimentos musculares necessários para a realização de tal ação na face. Além disso, essa notação também é composta da intensidade da AU, que pode ser representada por uma letra entre *A* (intensidade leve) e *E* (intensidade máxima).

A Tabela 1 traz uma adaptação da tabela apresentada no estudo de Silva et al. [10], que utilizou os valores definidos pela literatura para a expressões faciais das emoções, argumentando que as expressões faciais de emoções são amplamente utilizadas e compreendidas pela comunidade surda.

Cada AU se refere a um atuador para realizar uma determinada expressão facial. De forma resumida, *AU1* se refere ao atuador levantador de sobrancelha interna, *AU2* ao levantador de sobrancelha externa; *AU4* ao abaixador de sobrancelha; *AU5* ao levantador da pálpebra superior; *AU6* ao levantador da bochecha; *AU7* ao apertador de pálpebra; *AU9* ao enrugador de nariz; *AU12* ao puxador de canto dos lábios; *AU15* ao

depressor de canto dos lábios; *AU16* ao depressor do lábio inferior; *AU20* ao esticador de lábios; *AU23* ao endurecedor de lábio; e *AU26* ao atuador responsável pela queda do queixo.

Tabela 1. Emoções básicas na notação de FACS, adaptada de [10].

FACS	Descriçao
AU6 + AU12	Alegria
AU1 + AU4 + AU15	Tristeza
AU4 + AU5 + AU7 + AU23	Raiva
AU1 + AU2 + AU5B + AU26	Surpresa
AU1 + AU2 + AU5 + AU20 + AU26	Medo
AU1 + AU4 + AU5 + AU7	
AU9 + AU15 + AU16	Nojo

Para a reprodução dessas descrições das expressões faciais das emoções básicas no avatar 3D, foram utilizadas as animações de expressões faciais preestabelecidas no projeto, apesar das mesmas não possuírem paralelos exatos com todas as AUs, são elas: sorriso, fechar sobrancelhas, abrir sobrancelhas, abaixar sobrancelhas, subir sobrancelhas, franzir sobrancelhas, subir lábio superior, contrair lábios, baixar cantos da boca, subir cantos da boca, inflar bochechas, fazer bico, e contrair bochechas.

Em relação ao expressar das emoções, para expressar Alegria, foram modificados os valores das animações de sorriso e cantos da boca. Para expressar Tristeza, foram modificados os valores das animações das sobrancelhas, cantos da boca e contrair lábio. Para expressar Raiva, foram modificados os valores das animações das sobrancelhas, fazer bico e contrair os lábios. Para expressar Medo, foram modificadas as animações das sobrancelhas e bochechas, e também foi utilizada a animação de sorriso, sendo corrigido modificando o lábio superior e cantos da boca. Para expressar Nojo, foram modificados os valores das animações dos cantos da boca, contrair lábios, e também a animação de franzir as sobrancelhas, visando poder enrugar um pouco o nariz; sendo corrigida com a animação de abrir as sobrancelhas. Por fim, para expressar Surpresa, foram modificados os valores das animações das sobrancelhas, subir lábio superior. A expressão padrão do avatar 3D é considerada como a expressão do sentimento Neutro.

3. Resultados

A adaptação do tradutor e sinalizador automático da Suíte VLibras possibilitou a representação das emoções básicas, sendo elas: alegria, tristeza, raiva, medo, nojo e surpresa. Além da emoção neutra, a qual é performada pela ferramenta de forma padrão. Na Figura 3 é apresentado um *frame* da sinalização da palavra *ANDAR* nas 6 emoções básicas e na emoção neutra.

Para uma melhor avaliação da adaptação, também foi produzido um vídeo demonstrando a sinalização com emoção de textos contidos na base de dados de testes do analisador de emoções do estudo de Veríssimo e Costa [9]. O referido vídeo pode ser acessado por esse link.

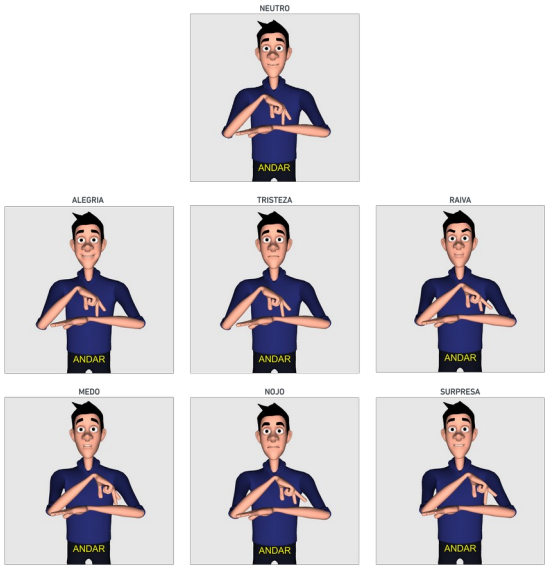

Figura 3. Exemplo da sinalização da palavra ANDAR na emoção neutra, alegria, tristeza, raiva, medo, nojo e surpresa.

Uma dificuldade encontrada pelo estudo foi que alguns movimentos importantes, como abrir a boca e controle da pálpebra não foram possíveis, frente às animações pré-definidas do avatar, o que prejudicou a definição de algumas expressões, como o medo e a surpresa. Além disso, ressalta-se que os testes foram realizados apenas com o avatar masculino, visto que, apesar de a ferramenta conter um avatar feminino, ele não apresenta as mesmas animações pré-definidas para expressões faciais, fazendo com que elas não sejam tratadas da mesma forma.

4. Discussão

A adaptação do tradutor automático de textos em Língua Oral para Língua de Sinais realizada a partir do fluxo genérico apresentado, e da sintaxe de construção da glosa com emoções, serve como um ponto inicial para as inúmeras possibilidades de implementação de um tradutor adaptado. A adaptação Suíte VLibras mostrou como a adaptação do tradutor automático pode ser feita com pouco impacto, mas em relação ao sinalizador, é necessária uma abordagem mais profunda.

Um ponto a ser mais trabalhado futuramente é a sintaxe de adição de emoções à glosa, que pode ser ampliada com a adição de metadados, ou atributos aos marcadores para uma melhor apresentação da emoção. Quanto ao sinalizador automático, aponta-se a possibilidade de ampliação das animações de expressões faciais do avatar 3D, para um melhor paralelo entre as expressões faciais das emoções básicas humanas. Por fim, é apontada a necessidade de testes com usuários reais para uma validação mais concisa do sistema como um todo.

5. Conclusões

Em um cenário onde uma parcela significativa da população, sendo essa a população de pessoas surdas, vivem à margem do acesso à ampla comunicação, visto

que a maioria das informações disponíveis são voltadas prioritariamente para pessoas ouvintes, esse trabalho surge como uma maneira de tentar auxiliar no aprimoramento de técnicas computacionais capazes de estreitar a comunicação entre pessoas ouvintes e não ouvintes.

Os resultados do presente estudo são promissores, de modo que levam a crer que esse tipo de adaptação pode ser bem-vista por parte dos usuários, podendo resolver alguns dos principais problemas referentes à expressividade dos tradutores automáticos.

Bibliografia

[1] Dizeu, L. C. T. D. B.; Caporali, S. A. (2005) A língua de sinais constituindo o surdo como sujeito. *Educação & Sociedade* 26(91): 583-597.

[2] Bai, Y.; Bruno, D. (2020) Addressing Communication Barriers Among Deaf Populations Who Use American Sign Language in Hearing-Centric Social Work Settings. *Columbia Social Work Review* 18(1): 37-50. DOI: 10.7916/cswr.v18i1.5928

[3] Ryan, C.; Johnson, P. (2019) Understanding language deprivation and its role in deaf mental health. *American Annals of the Deaf* 164(4): 519-524. DOI: 10.1353/aad.2019.0030

[4] Zucolotto, M.; Ruiz, L.; Pinheiro, N. (2019) Reflexões Sobre Linguagem, Sociedade e Surdez. *Revista Uniabeu* 12(30): 134-147.

[5] Rocha, C.; Melgaço, S. (2018) O uso de aplicativos para tradução de Libras. Anais do V Simpósio Internacional de Inovação em Mídias Interativas, 36-47.

[6] Oliveira, T.; Escudeiro, P.; Escudeiro, N.; Rocha, E.; Barbosa, F. (2019) Automatic sign language translation to improve communication. 2019 IEEE Global Engineering Education Conference (EDUCON), 937-942. DOI: 10.1109/EDUCON.2019.8725244

[7] Stoll, S.; Camgöz, N.; Hadfield, S.; Bowden, R. (2018) Sign language production using neural machine translation and generative adversarial networks. In: Proceedings of the 29th British Machine Vision Conference (BMVC 2018). British Machine Vision Association.

[8] Araújo, T. (2012) Uma solução para geração automática de trilhas em Língua Brasileira de Sinais em conteúdos multimídia. Tese. Programa de Pós-graduação em Engenharia Elétrica e da Computação. https://repositorio.ufrn.br/jspui/bitstream/123456789/15190/1/TiagoMUA_TESE.pdf

[9] Veríssimo, V.; Costa, R. (2020) Using Data Augmentation and Neural Networks to Improve the Emotion Analysis of Brazilian Portuguese Texts. In: Proceedings of the Brazilian Symposium on Multimedia and the Web, p. 13-20. DOI: 10.1145/3428658.3431080

[10] Silva, E.; Costa, P.; Kumada, K.; Martino, J.; Florentino, G. (2020) Recognition of affective and grammatical facial expressions: a study for Brazilian sign language. European Conference on Computer Vision, Springer, Cham, 218-236. DOI: 10.1007/978-3-030-66096-3_16

[11] Eckman, P., & Friesen, W. (1977) Manual for the facial action coding system. Consulting Psych. Press, Palo Alto.

Desenvolvimento de Dashboards para Análise de Características Geográficas e Climáticas do Estado da Paraíba (PB) - Projeto Plataforma Multi-Mapa PB

Miguel Marques Ferreira, Helon David Macêdo Braz, Flavio da Silva Vitorino Gomes, Joseíto de Oliveira Júnior

Centro de Energias Alternativas e Renováveis
Universidade Federal da Paraíba
{miguel.ferreira; hrlon; flavio; joseito.junior}@cear.ufpb.br

Resumo: O projeto Plataforma Multi-Mapa PB surgiu com o objetivo principal de possibilitar que usuários sem conhecimento profundo em programação, criem *dashboards* de vários tipos incluindo texto, imagem, listas suspensas, mapas georreferenciados, gráficos e tabelas para visualização de dados. Nesse sentido, um pacote nomeado de *json2dash* foi desenvolvido e dois *dashboards* foram elaborados para apresentação e validação da ferramenta utilizando dados obtidos através dos sistemas da *National Aeronautics and Space Administration* (NASA), da Agência Executiva de Gestão das Águas do Estado da Paraíba (AESA) e da Agência Nacional de Energia Elétrica (ANEEL).

Palavras-chave: *dashboard; Python; json2dash*.

1. Introdução

De acordo com dados do IBGE de 2018, o estado da Paraíba tem 1,9% da população do Brasil, enquanto o seu Produto Interno Bruto (PIB) corresponde a apenas 0,9% do PIB nacional [1]. Melhorar essa participação não é uma tarefa simples, mas certamente passa pela exploração inteligente de setores em que há um potencial maior inerente ao estado. A geração de energia por meio de fontes renováveis, tais como a solar fotovoltaica e a eólica, são exemplos de áreas com grande potencial de desenvolvimento.

De acordo com a Agência Internacional de Energia Renovável (IRENA), o Brasil é o segundo país no mundo que mais gera emprego no setor de geração de energia elétrica (hidrelétricas, usinas solares e eólicas) [2]. Tais fatores indicam crescimento consistente das fontes renováveis nos próximos anos, encorajando iniciativas no sentido de facilitar o planejamento e operação de novas usinas geradoras.

Em virtude das singularidades de cada região do Brasil, diversos fatores devem ser levados em conta para a exploração de uma fonte de energia. Fatores como a disponibilidade de recursos, interesses comerciais, domínio de tecnologias e a preservação do meio ambiente estão diretamente relacionados com a oferta de energia à população [3].

Dada a importância das características geográficas e climáticas, é imprescindível sua observação, análise e previsão. Nesse contexto, existem órgãos, empresas e agências como a Agência Executiva de Gestão das Águas do Estado da Paraíba (AESA) [4], a Agência Nacional de Energia Elétrica (ANEEL) [5] e a *National Aeronautics and Space Administration* (NASA) [6] que são responsáveis por mensurar as alterações vigentes e fornecer dados e informações aos mais diversos setores da sociedade.

As formas convencionais de visualização de dados utilizam gráficos, tabelas, diagramas, mapas, infográficos, painéis, etc. Contudo, o uso de *dashboards* (ou painéis interativos) surge com a proposta de sintetizar, uniformizar e detalhar as informações o máximo possível para evitar quaisquer problemas no entendimento, uma vez que compila os elementos gráficos comuns [7].

A visualização de dados fazendo uso de *dashboards* analíticos têm como meta a caracterização, predição, ganho de *insight* e o suporte nos processos de tomadas de decisões [8]. Ainda que existam informações e algumas bases de dados sobre as alterações climáticas, elas não estão disponíveis aos interessados utilizando uma grande variedade de representações gráficas [9].

O projeto Plataforma Multi-Mapa PB surgiu nesse ambiente, tendo como objetivo principal reunir informações relevantes sobre o estado da Paraíba (PB) e disponibilizá-las de forma amigável e simples para a sociedade. Trata-se de uma plataforma web que visa dar aos usuários a possibilidade de desenvolver seus próprios *dashboards* mesmo sem possuir conhecimento aprofundado de programação. O projeto está na sua primeira etapa, em que o foco é acessar e visualizar dados geográficos e climatológicos.

Em síntese, o presente trabalho expõe o processo de desenvolvimento de dois *dashboards*. Nessa primeira etapa foram usados dados abertos obtidos por meio dos sistemas da NASA, AESA e ANEEL referentes aos municípios do estado da Paraíba (PB). O trabalho foi organizado em mais quatro seções: Estado da Arte, Metodologia, Desenvolvimento e Resultados, Discussão e Conclusões.

2. Estado da Arte

No cenário atual, há vários projetos voltados para a permitir aos usuários a criação de *dashboards* interativos. Essas implementações se diferenciam pela facilidade de uso, pelo conhecimento de programação requerido e o mais importante, pelo nível de dependência criado entre a ferramenta e o projeto do usuário. Para exemplificar, algumas ferramentas com essa finalidade são Google Data Studio, Grafana, Voila e Panel.

O Google Data Studio é uma ferramenta *online* que possibilita a criação de *dashboards* sem a necessidade de utilizar uma linguagem de programação. É totalmente integrada ao ecossistema de aplicações da

empresa Google, entretanto não é possível implementar um *dashboard* que execute na máquina local [10].

O Grafana é uma ferramenta *online* de código aberto que possibilita a elaboração de *dashboards* sem a necessidade de utilizar uma linguagem de programação. Ao contrário da ferramenta da Google, o Grafana possibilita a integração com *APIs Rest*, mas também não possibilita implementação na máquina local e não permite a visualização dos *dashboards* sem um cadastro prévio [11].

Com o Voila é possível transformar de forma rápida e fácil *jupyter notebooks* em aplicativos web autônomos que podem rodar localmente [12]. O Voila é gratuito de código aberto, mas é necessário o interpretador Python para utilizá-lo.

O Panel é uma biblioteca Python de código aberto que permite criar aplicativos e *dashboards*, conectando *widgets* definidos pelo usuário a gráficos, imagens, tabelas ou texto [13]. É possível rodar localmente na máquina do usuário, mas é necessário utilizar o interpretador Python para utilizá-lo.

3. Metodologia

O projeto foi implementado a partir do uso da linguagem de programação Python, que é ideal para a concepção de aplicações em muitas áreas graças à sua relativamente fácil curva de aprendizado.

Os valores apresentados nos *dashboards* de exemplo mostrados neste trabalho foram retirados de uma base de dados local criada pelos integrantes do Projeto Multi-Mapa PB fazendo uso dos sistemas de dados abertos da AESA, ANEEL e IBGE.

Após a organização dos valores, surgiu a ideia de desenvolver um pacote Python que tivesse a capacidade de auxiliar a visualização web desses dados usando vários componentes. Esse pacote, constituído por um conjunto de módulos, foi nomeado de *json2dash*. Ele engloba algumas bibliotecas existentes dentro da linguagem Python, visando simplificar processos de programação e, principalmente, remover a necessidade de reescrever comandos comumente utilizados. As principais dependências do pacote *json2dash* são: Pandas, Folium e Dash.

O pacote Pandas é uma ferramenta desenvolvida para análise prática e manipulação de dados, flexível, de código aberto e baseada em estruturas nomeadas de *DataFrames* que possibilitam a leitura e escrita de valores tabulados em linhas e colunas, o que a torna bastante rápida e eficiente manipulação de dados [14].

A biblioteca Folium é um invólucro (do inglês, *wrapper*) de Leaflet.js, uma biblioteca *open-source* escrita em JavaScript para mapas interativos compatíveis com dispositivos móveis [15]. O Leaflet foi projetado visando a simplicidade, desempenho e usabilidade. Ele funciona com eficiência nas principais plataformas de desktop e móveis, podendo ser estendido com vários *plug-ins*. Como um *wrapper*, o Folium herda todas essas características.

O *framework* Dash, criado com base nas bibliotecas *Flask* e *Plotly*, é uma das ferramentas disponíveis para a construção de aplicações analíticas na internet. Renderizado no próprio navegador, sua utilização se torna bastante simples e consiste em uma interface composta por duas partes fundamentais: o *layout* e as *callbacks*. Essas partes, compreendem a construção visual com muitos componentes prontos para alocação na página web e funções de retorno para inserção de interatividade aos *sites* [16].

Dessa forma, a associação das bibliotecas citadas anteriormente originou o *json2dash*, pacote idealizado com o propósito de eliminar parcialmente a necessidade de utilizar programação para o desenvolvimento de aplicações web, uma vez que o usuário enviará informações por meio de um arquivo de texto no formato JSON (*JavaScript Object Notation*). As informações fornecidas como entrada são convertidas de forma automática em uma aplicação Dash que pode ser publicada na Internet.

No arquivo enviado pelo utilizador existe a opção de fornecer as chaves principais: *callback* e *layout* (Figura 1). Essas informações são suficientes para criação do *dashboard*.

```
{
    "callback": {
        "name_module": "",
        "file_path": "file.py"
    },
    "layout": {
        "rows": {
            "first_line": {
                "columns": {
                    "first_column": {
                    }
                }
            }
        }
    }
}
```

Figura 1. Estrutura de arquivo de texto enviado pelo usuário.

A chave *callback* é opcional, no entanto ela garante a atualização dinâmica do *layout* por meio de funcionalidades interativas para os componentes da página web. Quando utilizada, requer o envio do caminho até um arquivo Python com *callbacks*. A chave *layout* é obrigatória, sendo utilizada para a construção da parte visual da aplicação. O *layout* do *dashboard* é montado utilizando uma grade contendo linhas (*rows*) e colunas (*columns)*. Além disso, cada coluna pode conter um único componente visual escolhido dentro de uma variedade de elementos, tais como: texto com linguagem de marcação *markdown*, imagem, lista suspensa de seleção, mapa georreferenciado, gráfico, tabela, etc.

É possível instalar a ferramenta *json2dash* por meio do instalador de pacotes padrão do Python: o pip. Os membros do projeto direcionam o comando de instalação para o repositório do projeto na nuvem, podendo então desenvolver seus *dashboards* sem a necessidade de conhecer detalhes de implementação do pacote.

Para exemplificar as potencialidades do pacote *json2dash*, dois *dashboards* exemplo foram elaborados utilizando alguns dos componentes desenvolvidos, com destaque para mapas georreferenciados e gráficos dinâmicos.

4. Desenvolvimento e Resultados

Para evidenciar os resultados obtidos até agora, foi desenvolvido um protótipo de *dashboard*, em sua primeira versão, destinado ao estudo de grandezas climáticas dos municípios do estado da Paraíba utilizando dados de temperatura obtidos por meio do sistema da NASA compreendendo o intervalo de tempo de 1983 até 2018.

Essa aplicação, utilizando sobretudo o componente para geração de gráficos, entrega ao usuário a possibilidade de trabalhar com a seleção de até 13 (treze) tipos básicos de gráfico, incluindo gráficos de pontos, linhas, barras, distribuição (histogramas e *boxplots*), além de *heatmaps* (mapas de calor) e dispersão. Além disso, dentre as funcionalidades inseridas na aplicação, é possível manipular as grandezas para os eixos x e y das figuras, o intervalo de tempo de interesse e até a aparência da figura.

O painel de controle foi dividido em cinco seções: 1) *Ranking* de cidades por grandeza climática e mesorregião, 2) Gráficos por mesorregião, 3) Gráficos por intervalo de tempo, 4) Gráficos para comparação de grandezas entre cidades e 5) Mapa de calor (*HeatMap*) para uma cidade exclusiva.

Na Figura 2, é apresentada a segunda seção elaborada com as informações das temperaturas média, mínima e máxima a 2 metros do solo, em °C, filtradas a partir da escolha de uma cidade de interesse dentre as mesorregiões da Paraíba: agreste paraibano, Borborema, mata paraibana e sertão paraibano.

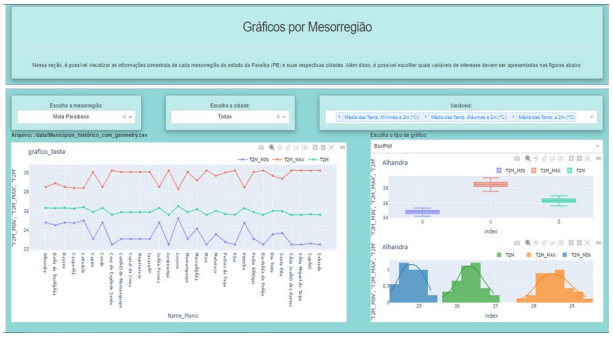

Figura 2. Seção do *dashboard*: Gráficos por Mesorregião.

Além disso, na maioria das seções, foi fixada a grandeza para o eixo das abscissas (eixo x) com valores relacionados ao intervalo de tempo em que aquela informação foi coletada. E assim, para o eixo das ordenadas (eixo y), múltiplas possibilidades de grandezas estão dispostas para seleção. Por exemplo:

- Precipitações média e acumulada;
- Umidade a 2 metros, específica e relativa;
- Pressão na superfície, em kPa;
- Temperaturas a 2 metros, em °C (mínima, máxima e média);
- Velocidades do Vento a 10 metros e 50 metros, em m/s (mínima, máxima e média).

Outro *dashboard* com informações georreferenciadas foi criado utilizando dados obtidos da AESA e da ANEEL, os quais dispõem da posição de aerogeradores, geração distribuída e usinas fotovoltaicas além da posição e geométrica de açudes da Paraíba (Figura 3).

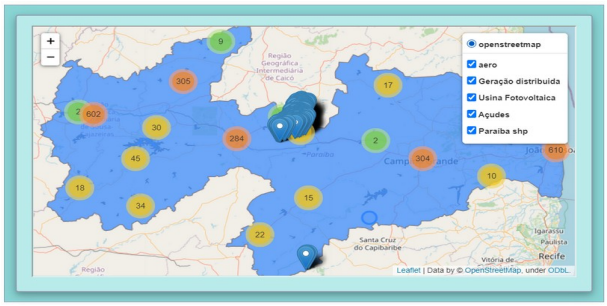

Figura 3. *Dashboard* com informações georreferenciadas.

5. Discussão

Informações a respeito do período de precipitação, incidência solar, regime dos ventos e temperatura são fatores diretamente relacionados com o desempenho de usinas geradoras. Por isso, os *dashboards* oferecem apoio àqueles que necessitam de informações para fundamentar decisões. O uso de *callbacks* para a elaboração do painel de controle facilita a realização de diferentes ajustes com complexidade mínima e, assim, a obtenção da visualização conforme desejada.

Na Figura 4, o *ranking* das 20 cidades da Paraíba com maior média de precipitação desde o ano de 1983 até 2018 é mostrado como um exemplo ilustrativo.

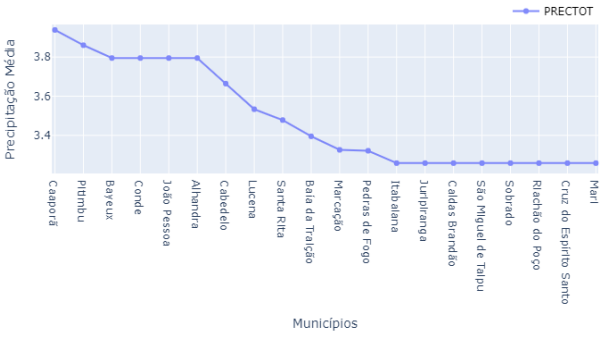

Figura 4. Ranking das cidades da Paraíba pela precipitação média.

Ainda sobre a Figura 4, é possível perceber que a maioria das cidades que apresentam uma maior precipitação média no intervalo analisado de 35 anos fazem parte da mesorregião da mata paraibana.

Na Figura 5, a temperatura média histórica de cada município, considerando uma distância de 2 metros do solo, é mostrada por meio de um mapa. Nele é possível perceber que a Borborema é a mesorregião em média mais fria, enquanto o sertão paraibano a mais quente.

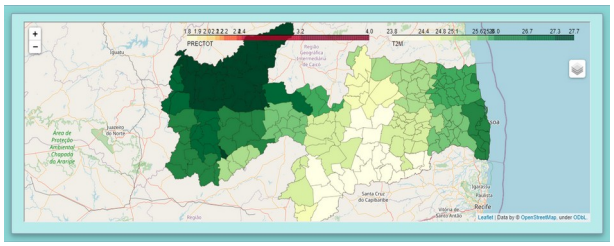

Figura 5. *Dashboard* com temperatura média a 2 metros do solo.

6. Conclusões

O projeto Plataforma Multi-Mapa PB foi apresentado do ponto de vista de sua motivação e objetivos gerais, tendo sido detalhada apenas sua primeira etapa, que consistiu no desenvolvimento de um pacote Python nomeado de *json2dash* e de dois *dashboards* de validação, voltadas para grandezas de interesse geográficas e climáticas do estado da Paraíba.

Foram utilizados dados abertos de diferentes fontes e apenas alguns dos componentes de visualização de informações disponíveis no pacote desenvolvido, mas considerando que esses resultados iniciais foram obtidos com um mínimo de programação por parte dos usuários, ficou demonstrado o potencial do projeto em termos de democratização do acesso a dados de forma genérica e também do ponto de vista das oportunidades saindo do contexto de visualização de dados e partindo para análises e previsões.

Bibliografia

[1] Coordenação de Contas Nacionais. (2020) Sistema de contas regionais: Brasil: 2018/IBGE, Coordenação de Contas Nacionais, 2018. IBGE.

[2] IRENA. International Renewable Energy Agency. (2020) Renewable Energy and Jobs – Annual Review, 2020. [s.n.]. Disponível em: <https://www.irena.org/publications/2020/Sep/Renewable-Energy-and-Jobs-Annual-Review-2020>. Acesso em 27/03/2021.

[3] Farias, M.; Sellitto, A. (2011) Uso da energia ao longo da história: evolução e perspectivas futuras. *Revista Liberato* 12(17): 7-16. DOI: 10.31514/rliberato.2011v12n17.p07

[4] AESA (2021) Informações Básicas – AESA. Disponível em: <http://www.aesa.pb.gov.br/>. Acesso em 27/03/2021.

[5] ANEEL (2021). Conheça a ANEEL. Disponível em: <https://www.aneel.gov.br/a-aneel>. Acesso em 27/03/21.

[6] NASA (2021) POWER Project. Disponível em: <https://power.larc.nasa.gov/>. Acesso em 27/03/21.

[7] Rolim, D. (2020) Dashboards para desenvolvimento de aplicações e visualização de dados para plataformas de cidades inteligentes. Dissertação de Mestrado. Universidade Federal do Rio Grande do Norte.

[8] Okoh, C.; Roy, R.; Mehnen, J. (2017) Maintenance informatics dashboard design for through-life engineering services. *Procedia CIRP* 59: 166-171.

[9] Sousa, M. (2019)Visualização de dados climáticos na Web. Tese de Doutorado. Universidade de Coimbra.

[10] Google (2021) Google Analytics Solutions. Disponível em: <https://datastudio.withgoogle.com/>. Acesso em 22/06/2021.

[11] Grafana (2021) Grafana: The open observability plataform. Projeto online. Disponível em: https://grafana.com/. Acesso em: 25/06/21.

[12] Voilá (2021) voila-dasboards/voila. Projeto online. Disponível em: <https://github.com/voila-dashboards/voila>. Acesso em 25/06/21.

[13] Panel (2021) holoviz/panel. Projeto online. Disponível em: <https://github.com/holoviz/panel>. Acesso em: 25/06/21.

[14] Pandas.org. (2021) About pandas. Disponível em: <https://pandas.pydata.org/about/>. Acesso em 27 março 2021.

[15] Leafletjs.org. (2021) Leaflet: An open-source JavaScript library for mobile-friendly interactive maps. Disponível em: <https://leafletjs.com/>. Acesso em 27/03/2021.

[16] Dash Enterprise (org.) (2021) Introduction to Dash. Disponível em: <https://dash.plotly.com/ introduction>. Acesso em 27/03/21.

Self-Organized UAV Flocking Based Only on Relative Range and Bearing

Thulio Guilherme Silva de Amorim, Tiago Pereira do Nascimento

Programa de Pós-Graduação em Informática
Centro de Informática - Universidade Federal da Paraíba
thulioguilherme53@gmail.com, tiagopn@ci.ufpb.br

Abstract: In this work, we address the problem of convergence and cohesiveness of an unmanned aerial vehicle (UAV) flocking. Thus, we propose a proximal control-based method for UAV self-organized flocking. Our method efficiently achieves flocking in the absence of alignment control and moves into an arbitrary direction without any direction control or informed robots. Robots use a Lennard-Jones potential function to maintain the cohesiveness of the flocking while avoiding collision within the teammates. We evaluate our method using the order metric, the steady-state value, and the settling time that can be used as a cohesiveness indicator.

Keywords: *unmanned aerial vehicles; flocking; swarm robotics; self-organization.*

1. Introduction

Flocking, also known as coordinated motion, is a behavior that consists of a large group of individuals moving together towards the same target direction. According to Ferrante et al. [1], flocking is traditionally realized using two main control rules: proximal control, which controls the cohesion of the swarm using local range-and bearing information about neighboring robots; and alignment control, which allows the robots to align in a common direction and uses more elaborate sensing mechanisms to obtain the orientation of neighboring robots.

Flocking can be achieved with the use of virtual physics-based design methods [2], where each robot is considered as a virtual particle that exerts virtual forces on other nearby robots. The most common methods for robot flocking usually use attraction-repulsion functions. Other works also add the alignment rule, e.g. the work of Ferrante et al. [1]. Some works even add information about a goal direction, e.g. the work of Tarcai et al. [3], and more recently Shirazi and Jin [4]. All of these methods have been applied to ground mobile robots, usually in an indoor environment, and some of them only in simulations.

UAV flocking algorithms are often tested only in simulations [5]. In contrast, recent fixed-wing UAV works have emerged with real robot experiments. Kownacki and Ołdziej [6], and Silic and Mohseni [7] have presented interesting studies, where the experiments rely on Global Positioning System (GPS) as the main positioning sensor and on communication between UAVs. Generally, UAV practical approaches use ranging systems based on radio signal transmissions [8]. Multi-rotor UAV flocking has also been tested in real conditions, the works of Virágh et al. [9] and Arul et al. [10] are example of contributions. Another recent work [11] has achieved flocking indoors with a swarm of up to 30 physical UAVs. This can be considered one of the most impressive achievements so far. Finally, all aerial mobile robot works cited uses an alignment control method which helps to achieve the flocking more quickly but requires more elaborate sensing mechanisms. In this work, we proposed a decentralized proximal control-based method for UAV self-organized flocking, which extends the work of Ferrante et al. [1].

2. Methodology

Flocking control methods are generally composed of three different term-functions: a proximal term, an alignment term, and an optional goal direction term, which is needed when the swarm is required to steer towards a specific target. Let us assume that in a flocking of n robots, a robot i, with $i \in \{1,\ldots, n\}$, is called the focal robot. The focal robot computes a flocking control function denoted by $\mathcal{F}$. Taking into account all flocking control terms, the flocking control function can be calculated as follows:

$$\mathcal{F} = \mathcal{P} + \mathcal{A} + \mathcal{G}, \quad (1)$$

where $\mathcal{P}$ is the proximal term, $\mathcal{A}$ is the alignment term, and $\mathcal{G}$ is the goal direction vector, which can only be used when there are informed robots (i.e. robots knowing the goal).

In this work, we propose a flocking control function $\mathcal{F}$, which needs only the proximal term to converge and move the UAVs into a unified direction. The equation (1) can therefore be rewritten as:

$$\mathcal{F}=\mathcal{P}. \quad (2)$$

The proximal term is used to make the UAV maintain the desired distance from other neighbor robots while keeping a cohesive formation. By using the proximal term, the focal robot can use the sensed range and bearing of its neighboring UAVs within a maximum interaction distance of D_p. For the model from our team's previous work [1] to be properly implemented on a flocking of UAVs, the distances between robots must be less than or equal to the maximum sensing range of the sensor. When the swarm converges into a stable formation, interactions between UAVs are limited to the first neighbors in the Voronoi sense, which in turn mandates that $D_p = \lambda d_{des}$, where d_{des} is the desired distance between UAVs, and λ is a positive gain that limits the maximum interaction distance to be less than

twice the desired distance between two robots. Therefore, we propose the proximal term function as:

$$\mathcal{P} = \sum_{i=1}^{m_p} p_i(d_i) e^{j\phi_i} \tag{3}$$

where m_p is the number of neighboring UAVs perceived by the focal robot within the maximum interaction distance, and d_i and ϕ_i denote the relative range and bearing, both expressed in the body frame of reference of the focal robot.

The term $p_i(d_i)$ is the magnitude of the proximal vector. In our case, we employ the Lennard-Jones potential used by our team for unmanned ground vehicles (UGVs) [1], and it can be calculated as:

$$p_i(d_i) = -\frac{4\alpha\epsilon}{d_i}\left[2\left(\frac{\sigma}{d_i}\right)^{2\alpha} - \left(\frac{\sigma}{d_i}\right)^{\alpha}\right], \tag{4}$$

where $P(d_i)$ is a virtual potential function, ϵ is the strength of the potential function which determines its depth, α is the steepness of the potential function, and σ is the amount of noise calculated by equation (5). Thus, the minimum of the potential function is when d_i=-ϵ, and it can be expressed as:

$$d_i = d_{des} = 2^{\frac{1}{\alpha}}\sigma \tag{5}$$

Magnitude-dependent Flocking Motion Control

The motion control is responsible for translates the output of the proximal control into a robot motion. The magnitude-dependent flocking motion control proposed by Ferrante et al. [1] makes possible the flocking in a random direction without an alignment control. In the following, we assume that the UAV has a translation axis x and a rotational axis y. The desired linear translational movement v is set as proportional to f_x, which is the projection of the flocking control vector $\mathcal{F}$ into the x axis of the body reference frame, where the translational motion is defined according to the applied non-holonomic constraints. Conversely, the desired angular movement ω is set as proportional to f_x , the projection of $\mathcal{F}$ into the y axis, where the rotational motion is defined according to the constraints.

We decompose the value of $\mathcal{F}$ into f_x and f_y by using the values of the bearing ϕ_i. We call f_x and f_y the projection of the flocking control vector $\mathcal{F}$ on the XY-plane of the body reference frame of the focal robot:

$$\begin{aligned} f_x &= \sum_{i=1}^{m_p} p_i(d_i)\cos\phi_i, \\ f_y &= \sum_{i=1}^{m_p} p_i(d_i)\sin\phi_i, \end{aligned} \tag{6}$$

where ϕ_i is the bearing.

Finally, the two movements can be calculated as:

$$\begin{aligned} v &= \kappa_1 f_x + B_s, \\ \omega &= \kappa_2 f_y, \end{aligned} \tag{7}$$

where B_s is the forward biasing movement, and κ_1 and κ_2 are the linear gains and the angular gains, respectively.

As the final step, we compute the desired pose for achieving or maintain the flocking using the linear movement v and angular movement ω as described in the equation (8). To achieve the advantages of MDFMC, the desired pose is generated satisfying non-holonomic constraints:

$$\begin{aligned} x_d(k) &= x + v * \cos\theta, \\ y_d(k) &= y + v * \sin\theta, \\ z_d(k) &= z_{coeff}, \\ \theta_d(k) &= \theta + \omega, \end{aligned} \tag{8}$$

where z_{coeff} is the desired height of the flock, $[x_d, y_d, z_d]^T = r_d$ is the desired position, and θ_d is the desired heading.

3. Experiments

In simulations, swarming algorithms usually are applied in a large group of robots ($\geq$ 100). When using ground robots, gathering this high number of robots is somehow feasible. However, the use of a large number of UAVs when simulating real disturbances from outdoor environments is not a simple task. Due to computational power limitations, we performed simulations with four robots but still succeeded in proving the efficiency of our proposed method.

3.1 Used Metrics

We use three metrics to analyze the effectiveness of our method. First, we use a metric that demonstrates how cohesive the swarm aligns in a common direction in a short period. This is the order metric ψ which measures the degree of agreement of the orientations of the UAVs within the swarm. Thus, the vectorial sum of the headings of all N robots is:

$$\mathbf{b} = \sum_{i=1}^{N} e^{j\phi_i}. \tag{9}$$

and the order can be calculated as:

$$\psi = \frac{1}{N}||\mathbf{b}||. \tag{10}$$

With this metric, the common heading is defined by a value of $\psi \approx 1$. When the UAVs point in different directions the value of $\psi \approx 0$. We also analyze the steady-state value that is reached for a given metric (in our case the order metric). The steady-state metric is the

asymptotic value reached by the order metric during the simulation. We can compute the value of the steady-state as:

$$\bar{\mu} = \frac{\sum_{(t=T-100)}^{T} \psi_t}{100}. \quad (11)$$

We also used the settling time which is the time needed to reach a steady-state in order. More precisely, the settling time t^* is defined as the time for which $\forall t \geq t$ we have $\mu \geq 0.95\bar{\mu}$. In other words, t^* is the time at which and after which the order stays above the 95% of the steady-state.

3.2 Results

All the control and estimation are supported by the Multi-robot Systems (MRS) UAV system [12]. The MRS UAV system is an open-source implementation system for supporting replicable research through realistic simulations. We build our method using the Robot Operating System (ROS).

The simulation environment was implemented on the MRS simulator [12] which is a simulation environment based on the open-source Gazebo simulator. Similar to the real robot, the simulated multirotor UAV can estimate its pose through a GPS localization system and communicate with the others through Wi-Fi. Each UAV runs an instance of our proposed method and obtains the range d and bearing ϕ of its neighboring by extracting it from the global position information of the others.

In the performed simulations, the initial position of each UAV was selected manually however the orientation was set randomly. After the take-off and hovering for a 10s period, the simulation starts. We summarize the value of the parameters used in the simulation in Table 1.

Since there is no goal position to reach or any other kind of stopping criteria, we allowed the swarm to try to achieve and maintain the flocking for 240 seconds. Once this experiment duration was reached, the robots are triggered to finish the mission and standstill in their last position.

Table 1. Parameters values.

Parameter	Description	Value
B_S	Maximum forward speed	0.3 m/s
κ_1	Linear gain	0.5
κ_2	Angular gain	0.2
α	Steepness of potential function	2
ϵ	Strength of potential function	6
d_{des}	Desired inter-robot distance	6 m
D_p	Maximum interaction range	10.8 m
λ	Interaction range gain	1.8
γ	Interpolation coefficient	0.95
z_{coeff}	Desired height	2.5 m

With only the proximal method proposed, the UAVs rapidly converged in range and bearing as can be observed in the settling time of 26 seconds (see Figure 1). During the simulation, the flocking moved towards an arbitrary direction, as intended (see Figure 2). The simulation revealed a steady-state value of μ =1.0097 as it indicates that the order assumed values close to 1 during the simulation.

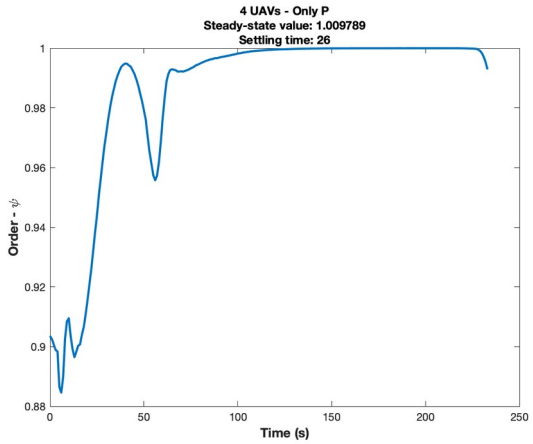

Figure 1. Order value in the experiment.

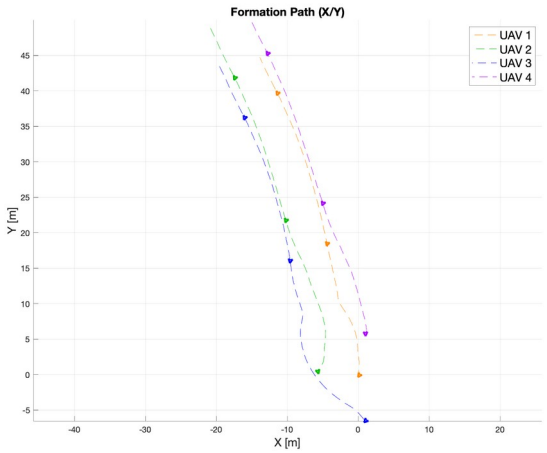

Figure 2. XY plot showing UAV headings during the simulation.

4. Discussion

The development of flocking algorithms in swarm robotics commonly uses ground robots in indoor environments mainly due to the simpleness of the environment setup. Basic research in the area of UAV flocking and formation flying was recently studied in [13]. Our results demonstrated that the proposed method works on UAVs. Furthermore, the simulations were performed in a distributed manner, where each robot has its control and flocking system. UAV swarm control is a relatively new field of research, and its applications are yet to be explored. Although, the adaptation of methods from ground robots to flying robots is usually mentioned as future work by many authors. We extend the work of Ferrante et al. [1] by adapting the decentralized proximal control-based method for UAV self-organized flocking.

5. Conclusion

We have presented an efficient and simple method for UAV flocking based only on a proximal function. This function has been used to enable the UAVs not only to maintain a cohesive flock but to move in an arbitrary direction. We performed experiments using GPS and communication devices to exchange UAV positions and estimate the range and bearing. Our method managed to enable the robots to converge towards the flock and maintain the cohesiveness of the group. It also enabled the UAVs to move in an arbitrary direction. In this way, we achieved self-organized flocking for aerial robots using only proximal control. Simulations with a large number of robots and real-world experiments in outdoor environments are examples of future works.

Bibliography

[1] Ferrante, E.; Turgut, A. E.; Huepe, C.; Stranieri, A.; Pinciroli, C.; Dorigo, M. (2012) Self-organized flocking with a mobile robot swarm: A novel motion control method. *Adaptive Behavior* 20(6): 460-477. DOI: 10.1177/1059712312462248

[2] Brambilla, M.; Ferrante, E.; Birattari, M.; Dorigo, M. (2013) Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence* 7:1-41. DOI: 10.1007/s11721-012-0075-2

[3] Tarcai, N.;Virágh, C.; Ábel, D.; Nagy, M.; Várkonyi, P. L.; Vásárhelyi, G.; Vicsek, T. (2011) Patterns, transitions and the role of leaders in the collective dynamics of a simple robotic flock. *Journal of Statistical Mechanics: Theory and Experiment* 2011(04): P04010. DOI: 10.1088/1742-5468/2011/04/p04010

[4] Shirazi, A. R.,; Jin, Y. (2020) Regulated morphogen gradients for target surrounding and adaptive shape formation. *IEEE Transactions on Cognitive and Developmental Systems*. DOI: 10.1109/TCDS.2020.2984087

[5] De Benedetti, M.; D'Urso, F.; Fortino, G.; Messina, F.; Pappalardo, G.; Santoro, C. (2017) A fault-tolerant self-organizing flocking approach for uav aerial survey. *Journal of Network and Computer Applications* 96: 14-30. DOI: 10.1016/j.jnca.2017.08.004

[6] Kownacki, C. & Ołdziej, D. (2016) Fixedwing uavs flock control through cohesion and repulsion behaviours combined with a leadership. International Journal of *Advanced Robotic Systems* 13(1): 36. DOI: 10.5772/62249

[7] Silic, M. & Mohseni, K. (2019) Field deployment of a plume monitoring uav flock. *IEEE Robotics and Automation Letters* 4(2): 769–775. DOI: 10.1109/LRA.2019.2893420

[8] Bhavana, T.; Nithya, M.; Rajesh, M. (2017) Leader-follower coordination of multiple robots with obstacle avoidance. SmartTechCon, 613-617. DOI: 10.1109/SmartTechCon.2017.8358444

[9] Virágh, C.; Vásárhelyi, G.; Tarcai, N.; Szörényi, T.; Somorjai, G.; Nepusz, T.; Vicsek, T. (2014) Flocking algorithm for autonomous flying robots. *Bioinspiration & Biomimetics* 9(2): 025012. DOI: 10.1088/1748-3182/9/2/025012

[10] Arul, S. H.; Sathyamoorthy, A. J.; Patel, S.; Otte, M.; Xu, H.; Lin, M. C.; Manocha, D. (2019) Lswarm: Efficient collision avoidance for large swarms with coverage constraints in complex urban scenes. I*EEE Robotics and Automation Letters* 4(4): 3940–3947. DOI: 10.1109/LRA.2019.2929981

[11] Vásárhelyi, G.; Virágh, C.; Somorjai, G.; Nepusz, T.; Eiben, A. E.; Vicsek, T. (2018) Optimized flocking of autonomous drones in confined environments. *Science Robotics* 3(20). DOI: 10.1126/scirobotics.aat3536

[12] Baca, T.; Petrlik, M.; Vrba, M.; Spurny, V.; Penicka, R.; Hert, D.; Saska, M. (2020) The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles [arXiv: 2008.08050]. Retrieved November 9, 2020, from http://arxiv.org/abs/2008.08050

[13] Saska, M.; Hert, D.; Baca, T.; Kratky, V.; Nascimento, T. (2020) Formation Control of Unmanned Micro Aerial Vehicles for Straitened Environments. *Autonomous Robots* 44: 1573–7527. DOI: 10.1007/s10514-020-09913-0

Transmissão de Áudio em Tempo Real sobre Redes Wireless IEEE 802.11 com Foco em Acessibilidade para o Cinema Digital

Caio Marcelo Campoy Guedes, Danilo Assis Nobre dos Santos Silva, Guido Lemos de Souza Filho

Programa de Pós-Graduação em Informática
Centro de Informática - Universidade Federal da Paraíba
{caio.marcelo, danilo, guido}@lavid.ufpb.br

Resumo: A transmissão de reforço de áudio acessível em salas de cinema é um exemplo em que o envio de conteúdo em tempo real é um fator limitante. Nessa aplicação, é desejável que o retardo entre a captura e exibição do áudio mantenha-se abaixo de um limiar de 40 ms. Este estudo tem como objetivo desenvolver uma solução capaz de transmitir áudio em tempo real sobre redes Wi-Fi entre um servidor e múltiplos clientes. Almejando o máximo de resiliência, a solução faz uso de *Forward Error Correction* e redundância temporal para mitigar os efeitos indesejáveis. Por fim, ainda é proposto um novo algoritmo adaptativo que busca ajustar o *buffer* de *playback* com o propósito de atenuar o *jitter* da rede.

Palavras-chave: *transmissão de áudio; áudio; FEC; algoritmo adaptativo; redundância;* smartphones*;* wireless*;* jitter.

1. Introdução

Com a constante luta para inclusão de deficientes visuais e auditivos em apresentações artísticas, observa-se o surgimento de diversas tecnologias desenvolvidas cujo propósito é auxiliar esses indivíduos a terem melhor compreensão sobre determinada apresentação, seja ela uma peça teatral ou uma exibição em salas de cinema. As técnicas presentes no mercado variam desde o uso de painéis textuais em que é realizada estenografia, ao uso de transmissores de áudio e a presença de intérpretes de línguas de sinal.

O reforço de áudio individual é um recurso bastante delicado, uma vez que variações no envio do conteúdo, seja por perda de pacotes ou latência, afetam diretamente a experiência do usuário. Dentre as tecnologias desenvolvidas para transmitir reforço de áudio em eventos de acessibilidade, destaca- se o uso de transmissão por FM (*Frequency Modulation*) e o uso de tecnologia IR (*infrared*). Embora ambas as soluções sejam maduras e possuam excelente desempenho no processamento e envio dos dados, existem problemas de alto custo e ajustes legais.

A alocação de frequências FM deve ser efetuada junto a um órgão governamental especializado, sendo necessário realizar um trâmite extenso em cada local onde se deseja implantar a solução. Ressalta-se também que a regulamentação depende dos órgãos nacionais especializados, que pode variar a depender do país em questão. Segundo Lan [1], o principal problema no acesso à tecnologia de radiofrequência está intrinsecamente relacionado à defasada regulamentação e à baixa efetividade nos padrões de alocação, o que limita o acesso aos recursos e vantagens oferecidas pela tecnologia. Outro fator relevante é o investimento necessário para a compra de equipamentos dedicados e aquisição da concessão pública para uso da frequência desejada que difere entre países e regiões.

Já as soluções que utilizam a tecnologia IR, lidam com um alcance muito menor quando comparada a solução que utiliza FM. Entretanto, há o benefício de não necessitar de homologação junto aos órgãos governamentais para atuação, o que a torna mais atrativa. Todavia, ainda há a utilização de um hardware dedicado, o que encarece a solução.

Dentre as alternativas baseadas em transmissão não guiada, a comunicação através de redes *wireless* tem-se difundido rapidamente por conta da sua praticidade e eficiência, beneficiando diferentes aplicações como, por exemplo, a transmissão de conteúdos audiovisuais, jogos e centros de entretenimento multimídia [2]. A transmissão sem fio também destaca-se pela flexibilidade no conjunto de soluções tecnológicas como as propostas nos padrões de comunicação elaborados pelo IEEE (*Institute of Electrical and Electronics Engineers*), que permitem o uso de diferentes frequências e larguras de banda [3]. Com a facilidade de implantação e baixo custo na instalação de bases *wireless*, serviços que utilizam transmissão de dados para múltiplos pontos podem ainda se beneficiar de protocolos que otimizam a transmissão de fluxos de áudio para grupos (*multicast*). Todavia, ao utilizar redes *wireless*, observa-se grande instabilidade quando realiza-se uma transmissão de baixa latência, o que é, em geral, proveniente de atraso no envio do conteúdo e a colisão entre os pacotes [4].

Embora a transmissão por modulação em frequência seja mais eficaz em transmitir o conteúdo, por possuir menor latência, o *tradeoff* entre alto custo e *delay* é extremamente crucial neste tipo de aplicação. Assim, diante dos resultados das análises realizadas sobre as tecnologias disponíveis, optou-se pela utilização de bases *wireless* IEEE 802.11.

Além do uso das bases *wireless* é importante definir o dispositivo que será utilizado para reproduzir os conteúdos transmitidos. Como um dos focos da solução é prover mobilidade ao usuário, já que a mesma será utilizada primordialmente em cinemas, decidiu-se utilizar *smartphones* como *hardware* de *playback*. *smartphones* modernos possuem alto poder computacional, permitindo flexibilidade durante o desenvolvimento do *software* de reprodução, assim como fácil substituição caso haja a necessidade de troca

do dispositivo ou caso se faça necessário um *upgrade* em algum dos aparelhos.

O grande desafio da escolha da tecnologia a ser utilizada na resolução do problema proposto é a latência, caracterizada como o intervalo entre o que está sendo emitido pelo locutor e o que está sendo escutado pelo cliente, pois não é desejado que haja a ocorrência de eco ou perda de sincronização labial, o que ocasiona uma experiência desagradável ao usuário. Eventos audiovisuais que necessitam de reforço de áudio são, em geral, limitados por um fator de tempo bem definido. Na literatura, um limiar onde não se percebe a divergência na apresentação entre as trilhas de áudio e vídeo, em televisores, deve ser um delta igual ou inferior a 40 ms entre o momento em que a imagem é exibida e o áudio é reproduzido [5][6]. Logo, se a transmissão do conteúdo não conseguir suprir a latência em valor máximo de 40 ms, ocorrerá um grande desconforto para o usuário, já que para deficientes auditivos o filme poderá ser exibido na frente da trilha de áudio, e para deficientes visuais a audiodescrição poderá sobrepor diálogos do filme.

Dessa forma, o escopo do trabalho aqui proposto é a transmissão de áudio com valores de latência da ordem de 40 ms, que é baixa o suficiente para atender requisitos de sincronização labial e transmissão de reforço de áudio em serviços de acessibilidade em salas de cinema.

2. Metodologia

Este trabalho busca desenvolver uma solução cliente-servidor, denominada FAST (*Fast Audio Streamer*), que gerencia a transmissão e reprodução de fluxos de áudio em tempo real com baixo retardo através do uso de redes Wi-Fi para múltiplos destinatários, cujo objetivo é melhorar a experiência de usuários em salas de cinema. Buscando resumidamente responder a seguinte questão de pesquisa: Como transmitir áudio para grupos de usuários utilizando dispositivos móveis como receptores com latência entre captura e exibição igual ou inferior a 40 ms?

Sendo assim, o objetivo geral deste trabalho é desenvolver uma solução de transmissão e recepção de áudio em tempo real de baixo retardo, a qual subdivide-se em dois componentes: o servidor, responsável pelo envio do conteúdo, e a aplicação cliente para dispositivos móveis, que reproduzirá a mídia transmitida. A comunicação entre módulos será realizada através do protocolo IP (*Internet Protocol*) e os dados serão transmitidos através de uma rede sem fio Wi-Fi.

Essa solução buscará enviar os dados da forma mais ágil possível sem comprometer a experiência do usuário, ou seja, o FAST atuará realizando uma transmissão de áudio com latência igual ou inferior a 40 ms almejando o mínimo de perda de pacotes. Dessa forma, uma revisão sistemática da literatura foi realizada e três estratégias foram selecionadas, são elas: *Forward Error Correction* [7], redundância temporal e o desenvolvimento de um algoritmo adaptativo que manipula a frequência das amostras. Essas estratégias foram selecionadas com base no uso das tecnologias em mercado e na sua presença em artigos científicos, cujos resultados foram os mais promissores.

A Figura 1 apresenta uma visão macro da arquitetura do FAST e nela é possível visualizar a comunicação entre a fonte e o pipeline de processamento do servidor.

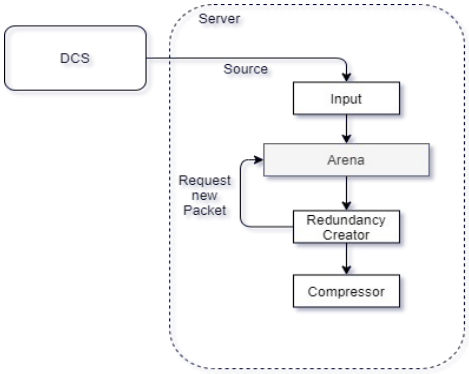

Figura 1. Arquitetura do FAST, visão do servidor.

Inicialmente, uma fonte de áudio, que neste trabalho é definida como DCS (*Digital Cinema Server*), transmite um fluxo de áudio em paralelo com a reprodução do vídeo referente a determinado filme. Em seguida, o módulo de captura do servidor, denominado *Input*, captura o áudio estéreo a uma taxa constante, para este trabalho, em que cada *sample* é definido como uma porção de áudio de 4 milissegundos. Os dados são então enviados a uma arena, que é uma região contínua de memória alocada junto ao *kernel* do sistema operacional. Essa região possui acesso privilegiado e é garantido que o sistema operacional não realizará paginação com essa porção. Em seguida, o módulo *Redundancy Creator* agrupa as amostras de áudio e procede com a criação de um pacote com 5 samples, das quais 4 são redundâncias, ou seja, 16 ms de conteúdo replicado. Esse é então enviado ao *Compressor*, módulo responsável por comprimir os dados e adicionar metadados para reconstrução das amostras através de FEC no lado do cliente. Assim que o processo de compressão é concluído, os dados são transmitidos através do protocolo UDP (*Multicast*) para os clientes conectados no grupo de recepção.

A Figura 2 exemplifica o funcionamento do FAST no lado do dispositivo móvel, neste trabalho, definido como um *smartphone*. Os dados que são recebidos na antena *wireless* do *smartphone* são enviados diretamente ao *Extractor*, módulo responsável pela decodificação dos pacotes, aplicação do FEC e pela inserção apropriada dos pacotes em uma nova *Arena*. É importante destacar que essa implementação da *Arena* não se beneficia da otimização do *kernel* e está sujeita a paginação. O *Extractor* tem visão direta sobre a região de memória onde os pacotes extraídos estão localizados, e, dessa forma, ele consegue identificar se é necessário utilizar alguma das redundâncias para corrigir uma falha na rede, que pode ser uma perda completa de um pacote ou um atraso na recepção. Assim, o módulo pode

corrigir até 4 pacotes perdidos na rede e em última hipótese ele consegue criar uma amostra de áudio, de pior qualidade, a partir dos metadados criados pelo algoritmo de FEC.

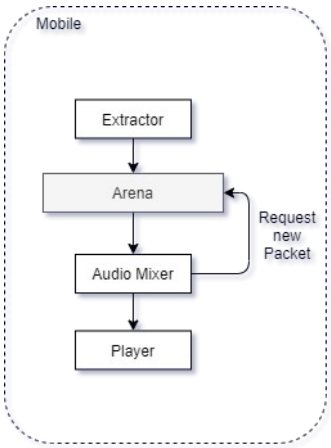

Figura 2. Arquitetura do FAST, visão do cliente.

Após a inserção na *Arena*, o *Audio Mixer* encarrega-se de preparar a trilha de áudio para ser executada para o ouvinte. Dentre as manipulações que o mixer pode realizar sobre o áudio original estão: amplificação do áudio, replicação de um dos canais e reamostragem, que é o algoritmo adaptativo proposto nesse trabalho. Ressalta-se que cada uma dessas opções pode ser desativada pelo usuário da aplicação.

A amplificação do áudio é o processo em que se aumenta o sinal referente ao áudio de entrada de forma uniforme. Esse processo é feito de forma linear onde um fator numérico é multiplicado a cada sample do áudio original. Porém, é importante definir um valor máximo para o processo de ganho, uma vez que existe a possibilidade de *clipping*, visto que o valor resultante da multiplicação pode exceder a precisão das amostras (*bit depth*) definido para o áudio em questão. A replicação dos canais é um requisito do uso da solução em salas de cinema, visto que no áudio estéreo que é transmitido aos espectadores são transmitidos o áudio amplificado da sala, lado esquerdo, que é utilizado por deficientes auditivos e a audiodescrição, lado direito, que é utilizada por deficientes visuais. O algoritmo de replicação de áudio tem complexidade linear, ou seja, dada uma entrada de áudio, é necessário a iteração sobre metade da informação para replicar os dados, e, dessa forma, o algoritmo não é impactante para a latência final da proposta. Além disso, o processo de replicação do conteúdo e a aplicação do ganho podem ser feitos concomitantemente, e, dessa forma, apenas uma iteração é necessária para aplicar ambos os algoritmos.

O *mixer* tem o papel de amenizar os efeitos de variação de retardo causado pela rede através do uso de um algoritmo adaptativo, que é responsável por expandir e contrair as amostras de áudio, assim como o funcionamento de um *buffer* elástico (uma espécie de sanfona), reproduzindo o áudio de forma mais lenta quando o *buffer* está se esvaziando e de forma mais rápida quando ele retorna a um nível aceitável. O áudio é manipulado através da adição ou remoção de dados de amostras do áudio, e, para isso, a ferramenta proposta utiliza a *libsamplerate*, um projeto *open source* para *upsampling* e *downsampling* de dados PCM (*Pulse Code Modulation*). A API (*Application Programming Interface*) fornece 5 mecanismos de *resampling*, variando entre algoritmos lineares e algoritmos por interpolação, e, dessa forma, possibilita a flexibilidade no ajuste da qualidade do algoritmo de *resampling* sem que se faça necessário a troca de bibliotecas. Como o FAST lida com oscilações na transmissão de dados muito rápidas, a correção efetuada pelo algoritmo precisa ocorrer instantaneamente, trazendo benefícios para a qualidade do serviço, uma vez que ao invés do espectador escutar um intervalo de silêncio que, em geral, é percebido como um estalo, o usuário escuta o áudio com uma pequena mudança de tonalidade.

Por fim, os dados são enviados ao *player* para que a informação seja reproduzida para o usuário.

3. Resultados

Para avaliar a solução, foram mapeados oito experimentos distintos utilizando a estratégia 2^kfatorial com o propósito de simular diferentes situações que podem ser encontradas em ambiente de cinema. Para estes experimentos foram mapeadas três variáveis independentes que são variadas com o propósito de avaliar se a aplicação consegue manter o *playback* em 40 ms e se o *buffer* da mesma permanece com ao menos uma amostra. Dessa forma, as variáveis independentes são:

1. Distância entre roteador e dispositivo (3.5 metros ou 7.3 metros);
2. Quantidade de redes na mesma faixa de canais (1 ou 3 roteadores);
3. Quantidade de dispositivos conectados na rede (1 ou 2 dispositivos).

Para cada execução do experimento, foram coletados a ocupação do *buffer*, os áudios da fonte e do dispositivo móvel, com o intuito de comparar a discrepância entre entrada e saída.

Dentre os resultados obtidos nos testes realizados, são apresentados a seguir os resultados para o primeiro experimento, roteador a 3.5 metros, 1 roteador na mesma faixa de canal e 1 dispositivo móvel. A partir de um histograma da ocupação do *buffer* foi possível identificar que a solução consegue manter uma média de alocação em *buffer* de 35 ms de áudio, sendo assim, a solução conseguiu manter-se abaixo do limite estipulado. Também foi possível notar que a solução oscilou a ocupação do *buffer* entre 4 ms de conteúdo e 40 ms, indicando que ela evitou a drenagem completa do *buffer* de *playback*. Ainda foi possível analisar a partir de um gráfico de *timeline* que o *resampling* realizado pelo algoritmo adaptativo auxilia a solução a manter-se em um ponto médio definido em 35 ms. No gráfico, foi visualizado um degrau onde amostras são

perdidas, esticadas por *upsampling*, até atingir um limiar estipulado e contraídas por *downsampling* para compensar a inserção de tempo.

Os demais experimentos realizados indicaram que a solução mantém valores muito similares com relação à ocupação de pacotes em *buffer* independentemente da variação das variáveis independentes, e, portanto, foram omitidos nesse trabalho.

4. Discussão

A solução deste trabalho destaca um serviço de transmissão de reforço de áudio acessível em salas de cinema. Todavia, esse não é o único cenário de uso para a solução. É possível utilizar o FAST em outras situações, como, por exemplo, visitas guiadas a museus, apresentações teatrais, partidas de futebol, dentre outros, trazendo tanto inclusão para os deficientes visuais e auditivos quanto novas formas de consumo de informação aos demais usuários.

5. Conclusões

Neste trabalho foi apresentada uma solução cliente-servidor denominada FAST, cujo foco é a transmissão de áudio em baixa latência sobre redes Wi-Fi. Ela trabalha com três estratégias distintas, que almejam baixa latência através do alívio de perda de pacotes e perturbações causadas sobre os sinais *wireless* pelo meio de transmissão. Desse modo, o uso de FEC, redundância temporal e controle adaptativo de consumo de amostras foram propostos como soluções, que melhoram significativamente a transmissão de fluxos de áudio. Ainda é possível afirmar que a solução apresenta diversos benefícios quando comparada às demais soluções do mercado e a outros trabalhos da literatura, que atingem latência média de 200 ms [7], como a baixíssima latência e o baixo custo de implantação. Esta solução é um dos poucos trabalhos que lida com reforço de áudio entre redes *wireless* e dispositivos móveis com latência inferior a 40 ms.

Bibliografia

[1] Lan, K. and Li, M. (2010) Feasibility study of using FM radio for data transmission in a vehicular network. International Computer Symposium (ICS2010) pp. 55-60. DOI: 10.1109/COMPSYM.2010.5685448.

[2] Kovacevic J.; Samardzija D.; Temerinac M. (2009) Joint coding rate control for audio streaming in short range wireless networks. *IEEE Transactions on Consumer Electronics* 55(2): 486-491. DOI: 10.1109/TCE.2009.5174411.

[3] Mangold, S.; Choi, S., Hiertz, G., Klein, O. and Walke, B. (2003) Analysis of IEEE 802.11e for QoS support in wireless LANs. *IEEE Wireless Communications* 10(6): 40-50. DOI: 10.1109/MWC.2003.1265851.

[4] Hardman V. et al. (1997) Reliable Audio for Use over the Internet. Proc. IINET'95 Conference pp. 171-178.

[5] Ravindran, K.; Bansal, V. (1993) Delay compensation protocols for synchronization of multimedia data streams. *IEEE Transactions on Knowledge and Data Engineering* 5(4): 574-589. DOI: 10.1109/69.234770.

[6] EBU Recommendation R37-2007 The relative timing of the sound and vision components of a television signal https://tech.ebu.ch/docs/r/r037.pdf. Acesso em 05/01/2020.

[7] McKinley, P. and Gaurav, S. (2000) Experimental evaluation of forward error correction on multicast audio streams in wireless LANs. In: Proc. eighth ACM International Conference on Multimedia (MULTIMEDIA '00). Association for Computing Machinery, New York, NY, USA, 416-418. DOI: 10.1145/354384.376304.

[8] Raju, G. et al. (2006) On Supporting Real-time Speech over Ad hoc Wireless Networks. 14th IEEE International Conference on Networks, pp. 1-6, DOI: 10.1109/ICON.2006.302667.