



Universidade Federal do Espírito Santo
Centro Tecnológico - Departamento de Engenharia Elétrica
Programa de Pós-Graduação em Engenharia Elétrica

**Uma Arquitetura de Microsserviços
centrada na Observabilidade Multinível para
Espaços Inteligentes baseados em Visão
Computacional**

Alexandre Pereira do Carmo

Vitória-ES, janeiro de 2021

Ficha catalográfica disponibilizada pelo Sistema Integrado de
Bibliotecas - SIBI/UFES e elaborada pelo autor

D631a do Carmo, Alexandre Pereira, 1973-
Uma Arquitetura de Microserviços centrada na
Observabilidade Multinível para Espaços Inteligentes baseados
em Visão Computacional / Alexandre Pereira do Carmo. - 2021.
181 f. : il.

Orientador: Anilton Salles Garcia.
Tese (Doutorado em Engenharia Elétrica) - Universidade
Federal do Espírito Santo, Centro Tecnológico.

1. Espaço Inteligente programável. 2. Arquitetura de
sistemas. 3. Orquestração hierárquica. 4. Observabilidade
multinível. 5. Microserviços. I. Garcia, Anilton Salles. II.
Universidade Federal do Espírito Santo. Centro Tecnológico. III.
Título.

CDU: 621.3

A minha família.

Agradecimentos

Agradeço a todos aqueles que me auxiliaram direta ou indiretamente nesse trabalho.

Resumo

Espaços Inteligentes são espaços físicos equipados com uma rede de sensores e atuadores, além de serviços de computação. Eles devem ser capazes de observar o ambiente e tomar decisões de forma a atender as necessidades dos seus usuários. Diferentes domínios podem ser atendidos pelos Espaços Inteligentes, dentre esses domínios destacam-se aqueles baseados em visão computacional. Estes últimos possuem câmeras como principal sensor, as quais coletam e processam uma variedade de informações. Devido ao grande volume de dados e a complexidade para seu processamento, aplicações de visão computacional possuem um conjunto de requisitos específicos e rigorosamente correlacionados que precisam ser atendidos. É comum que os requisitos de uma aplicação sejam definidos por elementos da infraestrutura. Porém, há requisitos específicos que somente a aplicação tem conhecimento e apenas ela poderá mensurá-los. Ainda que as informações sobre esses requisitos estejam no domínio da aplicação, eles podem ser diretamente impactados pelos recursos ofertados pela infraestrutura. Portanto, requisitos da infraestrutura tais como a taxa de transferência de dados, a capacidade de processamento ou o tempo de resposta estão rigorosamente correlacionados entre si e aos requisitos específicos das aplicações. Se por si só, atender simultaneamente a todos esses requisitos é um problema não trivial de ser resolvido, ele se agrava ainda mais ao se considerar algumas características dos espaços inteligentes baseados em visão computacional. Tais espaços devem ser escaláveis, capazes de abrigar múltiplas aplicações com requisitos que podem ou não ser dinâmicos. Por isso, a infraestrutura deve ser capaz de se modificar dinamicamente, de forma a atender continuamente aos requisitos, ao mesmo tempo que busca o uso racional dos recursos disponíveis, evitando sua sub ou super alocação. É nesse contexto que o trabalho apresentado nessa tese busca contribuir. O principal desafio é buscar uma forma de atender tanto aos requisitos específicos das aplicações quanto aos requisitos rigorosos e correlacionados da infraestrutura, garantindo o uso racional dos recursos disponíveis. Assim, a principal contribuição desse trabalho está em propor uma arquitetura para Espaços Inteligentes baseados em visão computacional com dois habilitadores essenciais: i) orquestração multinível centrada na observabilidade conjugada das aplicações e das camadas de infraestrutura; e ii) programabilidade granular da infraestrutura. A arquitetura proposta foi implementada em diferentes ambientes, com múltiplas aplicações e estudos de caso foram realizados como prova de conceito para sua validação.

Palavras-chaves: Espaço Inteligente programável. Arquitetura de sistemas. Orquestração hierárquica. Observabilidade multinível. Microserviços.

Abstract

Intelligent spaces are physical spaces equipped with a network of sensors and actuators, in addition to computing services. They must be able to observe the environment and make decisions in order to meet the needs of their users. Different domains can be served by Intelligent Spaces, among those domains stand out those based on computer vision. The latter have cameras as the main sensor, which collect and process a variety of information. Due to the large volume of data and the complexity for its processing, computer vision applications have a set of specific and strictly correlated requirements that need to be met. Commonly the requirements of many applications are defined by elements of the infrastructure. However, there are specific requirements that are known and measured only by the application. Although the information about these requirements is in the application domain, they can be directly impacted by the resources offered by the infrastructure. Therefore, infrastructure requirements such as data transfer rate, processing capacity or response time are closely correlated with each other and with many specific application requirements. Now, knowing that simultaneously meeting all these requirements is already a non-trivial problem to be solved, it gets even worse when considering intelligent spaces based on computer vision. Such spaces must be scalable, and capable of hosting multiple applications with requirements that may or may not be dynamic. Therefore, the infrastructure must be able to adapt dynamically, in order to continuously meet the requirements, while trying to keep a rational use of available resources, avoiding their sub or over-allocation. It is in this context that the work presented in this thesis aims to contribute. The main challenge is to find a way to meet both the specific requirements of the applications and the strict and correlated requirements of the infrastructure, keeping the rational use of the available resources. Thus, the main contribution of this work is to propose an architecture for Intelligent Spaces based on computer vision with two essential functionalities: i) multilevel orchestration centered on the combined observability of applications and infrastructure layers; and ii) granular programmability of the infrastructure. The proposed architecture was implemented in different environments, with multiple applications and case studies were performed as proof of concept for its validation.

Keywords: Programmable Intelligent Space. System architecture. Hierarchical orchestration. Multilevel observability. Microservices.

Lista de ilustrações

Figura 1 – Modelo de referência do ITU-T Y.2060/Y.4000.	32
Figura 2 – Visão funcional da arquitetura ARM.	33
Figura 3 – Arquitetura NFV do ETSI.	34
Figura 4 – Arquitetura do PIS.	35
Figura 5 – Processo de orquestração.	51
Figura 6 – Representação da implementação da Arquitetura do PIS.	53
Figura 7 – Representação da implementação da camada de suporte a aplicação do PIS.	54
Figura 8 – Mensagem protobuf.	55
Figura 9 – Dashboard para o experimento do estudo de caso 1.	56
Figura 10 – Representação da implementação da camada de serviços do PIS.	56
Figura 11 – Representação da implementação da camada de comunicação do PIS.	59
Figura 12 – Dashboard do RabbitMQ.	60
Figura 13 – Comunicação entre um produtor e um consumidor.	61
Figura 14 – Comunicação entre um produtor e dois consumidores.	61
Figura 15 – Exchange entre produtor e duas filas de consumidores.	62
Figura 16 – Relação entre exchange e routing_key.	62
Figura 17 – Implementação de RPC no broker.	63
Figura 18 – Dashboard do Weave.	65
Figura 19 – Representação da implementação da camada de virtualização do PIS.	65
Figura 20 – Entidade Virtual.	68
Figura 21 – Gateway da câmera.	69
Figura 22 – Representação da implementação da camada de gerenciamento do PIS.	70
Figura 23 – Exemplo de um rastreamento.	70
Figura 24 – Dashboard do Zipkin.	71
Figura 25 – Dashboard do Prometheus.	72
Figura 26 – Dashboad personalizado do Grafana.	74
Figura 27 – Integração Kubernetes e Prometheus.	76
Figura 28 – Abrangência do Espaço Inteligente da UFES.	80
Figura 29 – Abrangência do Espaço Inteligente do IFES Vitória.	80
Figura 30 – Abrangência do Espaço Inteligente da Universidade de Bristol.	82
Figura 31 – Robôs móveis na indústria 4.0.	86
Figura 32 – Robô móvel com o padrão Aruco	92
Figura 33 – Cadeia de Serviços que compõe a Aplicação.	93
Figura 34 – Arquitetura sem fio com Handover.	96

Figura 35 – Imagem capturada por uma das câmeras do PIS com os pontos marcados no chão	98
Figura 36 – Medições: a) Erro de localização nos 18 pontos marcados no chão b) CDF do erro de localização no PIS.	98
Figura 37 – Trajetória e velocidade do robô.	100
Figura 38 – Trajetória realizada e o erro para 2.5 FPS (a,b) e 10 FPS (c,d).	101
Figura 39 – Ilustração das medidas de tempo durante o loop de controle do robô.	102
Figura 40 – Cobertura das áreas das células no PIS da UFES.	104
Figura 41 – Tempo de comunicação (T_c): (a) 0 clientes, (b) 5 clientes, (c) 10 clientes and (d) PIS handover	105
Figura 42 – Erro de Trajetória: (a) 0 clientes, (b) 5 clientes, (c) 10 clientes e (d) PIS handover	106
Figura 43 – Cadeia de Serviços que compõe a aplicação.	112
Figura 44 – Fluxo de reconstrução 3D do esqueleto.	114
Figura 45 – Exemplo da reconstrução de um esqueleto 3D a partir de quatro imagens adquiridas no espaço inteligente da Ufes.	116
Figura 46 – Spam com gesture recogniser	117
Figura 47 – Matriz de confusão com os resultados do classificador de gestos.	119
Figura 48 – Exemplo de dois gestos que podem ser confundidos. a) gesto 7 e b) gesto 14. Nessas duas classes de gestos, o que as difere é a forma da mão.	120
Figura 49 – Fila única de mensagens na entrada do serviço de skeletons detector.	121
Figura 50 – Reconhecimento dos gestos em função do número de instâncias do serviço Skeletons Detector.	123
Figura 51 – Taxa de mensagens de entrada dos serviços SD, SG e GR em função do número de instâncias	124
Figura 52 – Incerteza do classificador do gesto em função da taxa de FPS das câmeras	127
Figura 53 – Resultado do experimento 5.	133
Figura 54 – Aplicação desenvolvida para visualização e envio de comandos.	163
Figura 55 – (a) Evolução das posições dos robôs e (b) erros de posição, orientação e distância da formação.	164
Figura 56 – Número de detecções simultâneas do padrão para cada robô.	164
Figura 57 – Cão-guia Robô Lysa	168
Figura 58 – Serviços que compõe a aplicação Mobilysa	170
Figura 59 – Aplicações: (a) Seguimento de pessoas (b) Desvio de pessoas (c) Mapa de Ocupação.	180
Figura 60 – Interrelação entre os serviços que compõem a aplicação.	182
Figura 61 – Robô com o padrão utilizado nos experimentos.	183
Figura 62 – Detecção de pessoas durante a tarefa de seguimento.	184
Figura 63 – Trajetória da pessoa e do robô durante a tarefa de seguir uma pessoa.	185

Figura 64 – Detecção de pessoas durante a tarefa de desvio.	186
Figura 65 – Trajetória realizada pelo robô e as posições das pessoas durante a navegação.	187
Figura 66 – Mapa de Ocupação.	188

Lista de tabelas

Tabela 1 – Comparação entre os trabalhos relacionados e o PIS.	30
Tabela 2 – Serviços de domínio.	57
Tabela 3 – Serviços de infraestrutura.	58
Tabela 4 – Lista de equipamentos do PIS da UFES	81
Tabela 5 – Lista de equipamentos do PIS do IFES	82
Tabela 6 – Lista de equipamentos do PIS de Bristol	83
Tabela 7 – Percentual do número de vezes que o sistema ultrapassa o requisito de tempo de resposta.	105
Tabela 8 – Lista dos gestos presentes no conjunto de dados utilizado. A classe 0 corresponde a um não-gesto, ou seja, quando nenhum gesto está sendo executado.	116
Tabela 9 – Média e desvio padrão do tempo de processamento de cada serviço para um número de 1000 amostras	118
Tabela 10 – Média e desvio padrão do tempo de processamento do Skeletons Detector para um número de 1000 amostras	118
Tabela 11 – Desempenho do Serviço Gesture Recognizer	120
Tabela 12 – Comparativo tecnológico	169
Tabela 13 – Análise da detecção de pessoas durante o experimento da aplicação de seguimento.	185
Tabela 14 – Tempo de resposta.	187

Lista de abreviaturas e siglas

2G	2th Generation mobile network
3G	3th Generation mobile network
3GPP	Third Generation Partnership Program
4G	4th Generation mobile network
5G	5th Generation mobile network
AI	Artificial Intelligence
AMQP	Advanced Message Queuing Protocol
AoA	Angle of Arrival
AP	Access Point
ARM	Architectural Reference Model
ARQ	Automatic Repeat Request
API	Application Programming Interface
BB	Bounding Box
BSSID	Basic Service Set Identifier
CDF	Cumulative Distribution Function
CF	Componente Funcional
CNN	Convolutional Neural Network
CPS	Cyber-Physical Systems
CPU	Central Process Unit
DC	Data Center
DCNN	Redes Neurais Convolucionais Profundas
DIND	Distributed Intelligent Network Devices
DNS	Domain Name System
E2E	End to End

ETSI	European Telecommunications Standards Institute
FI	Future Internet
FPGA	Field Programmable Gate Array
FPPI	False Positives Per Image
FPS	Frames Per Second
HARQ	Hybrid Automatic Repeat Request
HOG	Histogram of. Oriented Gradient
HTTP	Hypertext Transfer Protocol
HRI	Human Robot Interaction
GF	Grupos de Funcionalidades
GPS	Global Positioning System
GPU	Graphics Processing Unit
HPA	Horizontal Pod Autoscaler
HPN	High Performance Networks
ID	Identificação Única
IFES	Instituto Federal do Espírito Santo
IoT	Internet of Things
IP	Internet Protocol
ITU	International Telecommunication Union
ITU-T	Telecommunication Standardization Sector
KVM	Kernel-based VirtualMachine
LTE	Long Term Evolution
MR	Miss Rate
MQTT	Message Queuing Telemetry Transport
NAT	Network Address Translation
NFV	Network Functions Virtualization

OFDM	Orthogonal Frequency Division Multiplexing
OSI	Open System Interconnection
OVS	Open vSwitch
PIS	Programmable Intelligent Space
PUB	Publish
QoS	Quality of Service
RDN	Residues Defined Networks
REST	Representational State Transfer
RGB	Red, Green, Blue
RF	Radiofrequência
RFID	Radio Frequency IDentification
RGB-D	Red Green Blue Depth
ROI	Region of Interest
RPC	Remote Procedure Call
RSS	Received Signal Strength
SDN	Software Defined Network
SUB	Subscribe
SFC	Service Function Chain
SNR	Signal-to-Noise Ratio
TDoA	Time Difference Of Arrival
ToF	Time-of-Arrival
UFES	Universidade Federal do Espírito Santo
VLAN	Virtual Local Area Network
VPN	Virtual Private Network
VM	Virtual Machine
VxLAN	Virtual Extensible Local Area Network
YAML	YAML Ain't Markup Language

Sumário

1	INTRODUÇÃO	3
1.0.1	Estado da arte	6
1.0.2	Contribuição	7
1.1	Questão de Pesquisa	8
1.2	Hipóteses	8
1.3	Objetivos	9
1.4	Estrutura do trabalho	9
2	REFERENCIAL TEÓRICO E TRABALHOS RELACIONADOS	11
2.1	Arquitetura baseada em Microserviços	11
2.2	Características dos Espaços Inteligentes Atuais	16
2.2.1	Abrangência	16
2.2.2	Aplicações	17
2.2.3	Infraestrutura	20
2.3	Novas funcionalidades introduzidas pelo PIS	22
2.3.1	Orquestração multinível centrada na Observabilidade	22
2.3.2	Programabilidade Granular de Infraestrutura	26
2.4	Análise Comparativa dos Trabalhos Relacionados	28
3	ARQUITETURA DE MICROSERVIÇOS CENTRADA EM OBSERVABILIDADE	31
3.1	Modelos de Referência para o PIS	31
3.2	Arquitetura Proposta	34
3.2.1	Camada de Suporte a Aplicação	36
3.2.2	Camada de Serviços	37
3.2.3	Camada de Comunicação	38
3.2.4	Camada de Virtualização	39
3.2.5	Camada de Gerenciamento	41
3.2.6	Camada de Segurança	49
3.3	Processo de orquestração	49
4	IMPLEMENTAÇÃO	53
4.1	Arquitetura	53
4.1.1	Camada de suporte a aplicação	53
4.1.2	Camada de serviços	56

4.1.3	Camada de comunicação	59
4.1.4	Camada de virtualização	65
4.1.5	Camada de gerenciamento	69
4.2	Aplicações	78
4.3	Espaços Inteligentes Implantados	79
4.3.1	UFES	79
4.3.2	IFES - Vitória	80
4.3.3	HPN - Bristol	81
5	ESTUDO DE CASO 1: INDÚSTRIA 4.0	85
5.1	Cobertura sem fio e localização Indoor	88
5.2	Attocélulas e Localização por Visão Computacional	90
5.3	Aplicação de Controle Visual de Robôs Móveis em Tempo real	92
5.4	Arquitetura sem fio para implementação de attocélulas	95
5.5	Experimentos	97
5.5.1	Sistema de localização por visão computacional ofertado pelo PIS	97
5.5.2	Erro de trajetória x FPS	99
5.5.3	Entrega de pacotes para attocélulas com handover eficiente	103
5.6	Análise dos Resultados	106
6	ESTUDO DE CASO 2: RECONHECIMENTO DE GESTOS	109
6.1	Serviços e sua interação	112
6.2	Experimentos	114
6.2.1	Experimento 1: Tempo de processamento	117
6.2.2	Experimento 2: Taxas de Reconhecimento e Classificação	118
6.2.3	Experimento 3: Gestos reconhecidos e sua relação com o número de instâncias do serviço SD	120
6.2.4	Experimento 4: Incerteza e sua relação com a taxa de FPS	125
6.2.5	Experimento 5	128
7	CONCLUSÕES	135
7.1	Trabalhos Futuros	137
7.2	Contribuições	138
7.2.1	Publicações	138
7.2.2	Implementações	139
7.2.3	Projetos	140
	REFERÊNCIAS	143

ANEXOS	159
ANEXO A – APLICAÇÃO DE CONTROLE DE FORMAÇÃO DE ROBÔS MÓVEIS	161
ANEXO B – MOBILYSA - SISTEMA DE LOCALIZAÇÃO E CONTROLE DO CÃO-GUIA ROBÔ LYSA PARA AMBIENTES INTERNOS	167
ANEXO C – SERVIÇO DE DETECÇÃO DE PESSOAS . . .	175

1 Introdução

O conceito básico de Espaços Inteligentes é inspirado na ideia de computação pervasiva, em que a computação está imersa no ambiente frequentado ou ocupado livremente por pessoas e coisas (WEISER, 1999). A interação entre as diversas entidades acontece de forma natural, sem que haja um conhecimento explícito da tecnologia e computação embutidas no espaço. Assim, um espaço inteligente é capaz de oferecer novos ou melhores serviços através das interações que acontecem entre seus usuários e demais entidades existentes, ou ainda com o espaço propriamente dito.

Para ser considerado inteligente, o espaço deve ser capaz de observar o que está acontecendo no ambiente, processar as informações coletadas e tomar decisões e ações que atendam às necessidades dos usuários.

Neste trabalho, um espaço inteligente será definido como um espaço físico equipado com uma rede de sensores e atuadores, além de serviços de computação. Tais elementos devem ser gerenciados por uma infraestrutura de hardware e software responsável por coletar e analisar os dados, gerar as decisões e atuar quando necessário.

Um espaço inteligente pode ser composto por diferentes sensores e atuadores. Câmeras, microfones, ultrassom, termômetros e Global Positioning System (GPS) são exemplos de sensores que podem estar presentes em espaços inteligentes, enquanto robôs, telas de informação, altofalantes, chaves ou botões eletrônicos, sinalizações de trânsito e até mesmo carros ou drones, podem ser usados como atuadores para interagir com usuários ou modificar o ambiente.

Além de diferentes sensores e atuadores, uma ampla gama de aplicações voltadas para diferentes domínios podem fazer parte de um espaço inteligente. Como exemplo desses domínios podemos citar: automação residencial ou industrial, segurança pública, saúde, mobilidade urbana, controle de tráfego, gerenciamento de energia, logística e até mesmo finanças. Com domínios tão distintos, encontrar uma solução capaz de satisfazer os requisitos das aplicações de todos esses cenários representa um grande desafio.

Portanto, desenvolver uma arquitetura que atenda a um grande número de requisitos de aplicações de domínios tão diferentes é uma tarefa muito difícil. Desta forma, o problema é normalmente tratado de maneira específica, buscando criar arquiteturas de espaços inteligentes para domínios bem definidos.

Uma forma de se caracterizar um domínio específico pode ser conseguida quando se define um escopo físico para o espaço inteligente. Exemplos como salas especiais em aeroportos, casas inteligentes, prédios inteligentes ou cidades inteligentes são classificações

que levam em consideração o domínio caracterizado pela abrangência física. Outra forma de se classificar espaços inteligentes considera o tipo de sensor ou sensores predominantes no ambiente como o elemento principal que define o seu domínio.

Em alguns casos, é comum se utilizar um grande número de sensores com funções específicas mas que individualmente geram pequenos volumes de dados. Por exemplo, o trabalho (ILLINGWORTH et al., 2019) usa sensores terrestres como tectômetros, radiômetros de micro-ondas e radares de vento para monitoramento ambiental na Europa. Já o trabalho (AMOON; ALTAMEEM; ALTAMEEM, 2020) usa sensores de batimento cardíaco e movimento corporal em um domínio de *health care*.

Em contrapartida, há os casos em que os sensores são elementos mais complexos e coletam uma grande quantidade de dados. Um exemplo clássico, são espaços inteligentes baseados em visão computacional, os quais possuem câmeras como principal sensor. Tais espaços são particularmente interessantes devido à variedade de informação que pode-se extrair das imagens e a diversidade das aplicações desenvolvidas usando-se esses sensores. Outro aspecto importante é que, atualmente, câmeras são sensores presentes em um grande número de ambientes, normalmente voltadas para videomonitoramento e vigilância. Mesmo que ainda não sejam considerados inteligentes, tais ambientes podem ser transformados em espaços inteligentes baseados em visão computacional através de pequenas intervenções, aproveitando-se a estrutura já instalada.

Exemplos de tais espaços podem ser observados em diferentes trabalhos na literatura. Em (COSMA; RADOI; RADU, 2019), um rede de câmeras é usada para estimar a localização e a pose de pessoas em um ambiente. Outro exemplo pode ser visto em (SOMOV et al., 2019), onde uma plantação de tomates em uma estufa é monitorada e controlada por imagens de câmeras e outros sensores. O controle é realizado de forma a buscar as melhores condições de crescimento da plantação.

O estudo e caracterização de espaços inteligentes, que possuem câmeras como o seu principal sensor, se torna particularmente interessante devido ao fato de que esses espaços apresentam um conjunto de requisitos específicos e rigorosamente correlacionados para o seu pleno funcionamento.

É comum que os requisitos de uma aplicação sejam definidos por elementos da infraestrutura, tais como quantidade de memória para sua execução ou latência máxima permitida entre a aplicação e um atuador. Porém, o que nesse trabalho define-se por requisitos específicos são aqueles que somente a aplicação tem conhecimento e somente ela poderá mensurá-los. O erro da trajetória de um robô ou a incerteza no reconhecimento de um gesto são exemplos desse tipo de requisito. Em ambos os casos, somente a própria aplicação possui informações a seu respeito e somente ela poderá verificar se ele está sendo atendido ou não.

Ainda que as informações sobre esses requisitos estejam no domínio da aplicação, eles podem ser diretamente impactados pelos recursos ofertados pela infraestrutura. Considerando ainda os dois exemplos anteriores, a incerteza no reconhecimento de um gesto é diretamente impactada pela taxa de Frames Per Second (FPS) das câmeras. Da mesma forma, o número de câmeras influencia a precisão de localização de um robô no ambiente. Logo, uma alteração no requisito específico da aplicação gera um impacto considerável na infraestrutura.

No domínio de Espaços Inteligentes baseado em visão computacional, mesmo que poucas câmeras sejam utilizadas, o volume de dados gerado por esse sensores é muito grande. Fato que se torna mais preocupante quando o número de câmeras ou a taxa de FPS aumenta. Considere o exemplo da aplicação de reconhecimento de gestos, que para atingir o requisito específico de incerteza precise aumentar a taxa de FPS das câmeras. Essa aplicação deve então capturar cada imagem, processá-la e enviar comandos relativos aos gestos reconhecidos. Tudo isso dentro da janela de tempo definida pelo período de amostragem das imagens. Essa cadeia de ações implica em requisitos rigorosos e correlacionados da infraestrutura. Nesse caso o aumento da taxa de FPS se reflete no aumento do volume de dados trafegados, no aumento de poder computacional devido ao maior número de imagens e na diminuição do tempo de resposta, de forma que toda a cadeia de ações seja executada agora em uma janela de tempo menor. Atender a todos esses requisitos de forma simultânea é altamente complexo e se torna um grande desafio a ser enfrentado.

Todos esses requisitos geram um grande impacto na infraestrutura que suporta as aplicações que devem ser executadas nesses espaços inteligentes. Essa infraestrutura compartilhada, deve prover recursos suficientes para que todos esses requisitos sejam atendidos. Se por si só, esse é um problema não trivial de ser resolvido, ele se agrava ainda mais ao se considerar algumas características dos espaços inteligentes baseados em visão computacional.

Como primeira característica pode-se destacar as diferentes abrangências desses espaços que, como mencionado anteriormente, podem variar de pequenas salas a grandes cidades. Portanto, projetos de espaços inteligentes baseados em visão computacional devem ser escaláveis para atender a qualquer tamanho de ambiente. Uma segunda característica é que espaços como esse devem estar preparados para abrigar múltiplas aplicações, onde cada uma pode possuir um conjunto diferente de requisitos. Além disso, esses requisitos podem ser dinâmicos e podem ser alterados em tempo de execução a depender das condições do ambiente, exigindo maior ou menor volume de recursos.

Todas essas características de espaços inteligentes baseado em visão computacional, em conjunto com a necessidade de atender aos requisitos específicos e altamente rigorosos das aplicações de forma simultânea, fazem com que a infraestrutura deva ser capaz de se modificar dinamicamente, de forma a atender continuamente a esses requisitos e ao

mesmo tempo busque um melhor aproveitamento dos recursos disponíveis.

Desta forma, é possível perceber como o gerenciamento dos recursos pela infraestrutura é um ponto altamente relevante em uma arquitetura para Espaços Inteligentes baseados em visão computacional. O objetivo é a busca do uso racional dos recursos da infraestrutura. Por uso racional dos recursos entende-se pela alocação necessária para atendimento dos requisitos das aplicações sem que haja sub ou super alocação dos recursos, permitindo o correto atendimento ao maior número de demandas das aplicações possível.

Para propiciar o uso racional dos recursos é necessário que se estabeleça um gerenciamento granular da infraestrutura. Para isso, é necessário que cada elemento da arquitetura do espaço inteligente seja programável. Quer dizer, que cada elemento seja capaz de receber comandos que possam alterar o seu comportamento através de novas instruções ou novos parâmetros.

Considere como exemplo o requisito de tempo de resposta de uma tarefa de uma determinada aplicação. Essa tarefa é executada por dois serviços independentes de forma sequencial. Caso seja preciso diminuir o tempo de resposta, uma das ações possíveis é migrar um dos serviços para um servidor de maior capacidade de processamento. O problema é que ao fazer a migração, o tempo de comunicação entre os serviços pode ser aumentado. Nesse caso, a atuação apenas nos recursos de processamento mostra-se insuficiente para a solução do problema, sendo necessário a atuação também nos recursos de rede do ambiente. Esse exemplo ajuda a mostrar que quanto maior o nível de programabilidade do ambiente, maior o potencial para gerenciamento racional dos recursos.

1.0.1 Estado da arte

Na literatura encontram-se diferentes abordagens para a construção de espaços inteligentes. Em (CHEN et al., 2016) é apresentado um espaço inteligente para robótica em nuvem (Cloud Robotics). Esse espaço é composto por uma rede de sensores, como câmeras e ultrassom que são utilizados para capturar dados do ambiente e controlar robôs de serviço. Para isso, os autores propõem uma infraestrutura de nuvem híbrida para alocação das aplicações entre a borda e a nuvem. O objetivo principal é propor uma infraestrutura que seja escalável. Caso precise adicionar novas aplicações ou robôs, a plataforma tem como expandir os recursos para atender às novas demandas. Porém, nenhum requisito das aplicações é observado e o gerenciamento é rígido e feito apenas para os recursos de computação.

Uma abordagem mais comum para construção de espaços inteligentes está normalmente focada nas aplicações e nas redes de sensores e atuadores. Em (SPRUTE et al., 2018) foi criada uma aplicação de reconhecimento de gestos para controle de robôs móveis. Esse trabalho está centrado na computação sendo realizada nas próprias câmeras, tendo

por objetivo o cumprimento dos requisitos funcionais. A infraestrutura é simples e rígida e aspectos não funcionais das aplicações não são considerados.

Uma alternativa à utilização de plataformas desenvolvidas especificamente para espaços inteligentes, é a adaptação de arquiteturas projetadas para outros domínios. Em (ANTEQUERA et al., 2018) é realizada uma orquestração fim a fim de recursos em nuvem orientada à aplicação. A arquitetura prevê o controle granular da infraestrutura e com alocação dinâmica. Porém a orquestração é baseada em requisitos de infraestrutura descritos através de classes de aplicação, tais como aplicações de tempo real ou de transferência de dados intensiva. Requisitos específicos das aplicações não são observados e mesmo os requisitos de infraestrutura, apesar de rigorosos, não são correlacionados e simultâneos.

De maneira similar a arquitetura projetada para o projeto INITIATE (ALSHAER et al., 2020) propõe uma arquitetura que realiza a orquestração multidomínio em uma infraestrutura baseada em tecnologias como Software Defined Network (SDN), Network Functions Virtualization (NFV) e Internet of Things (IoT). A arquitetura gerencia os recursos de uma aplicação fim a fim e possui um alto nível de programabilidade. Mas tal como em (ANTEQUERA et al., 2018) a gerência dos recursos é realizada com uma visão apenas da própria infraestrutura.

Na maior parte dos trabalhos pesquisados, há uma visão isolada dos níveis de infraestrutura e de aplicação. Pouco se observa a relação entre os requisitos próprios da aplicação e os recursos da infraestrutura. E mesmo quando isso ocorre, a relação apresentada se mostra insuficiente para enfrentar os desafios do domínio de Espaços Inteligentes baseados em visão computacional.

É nesse contexto que o trabalho apresentado nessa tese busca contribuir. O principal desafio é como atender tanto aos requisitos específicos das aplicações quanto aos requisitos rigorosos e correlacionados da infraestrutura presentes no domínio de Espaços Inteligentes baseados em visão computacional, garantindo o uso racional dos recursos disponíveis na infraestrutura.

1.0.2 Contribuição

Para resolver esse desafio, a principal contribuição desse trabalho está em propor uma arquitetura para Espaços Inteligentes baseados em visão computacional com dois habilitadores essenciais: i) orquestração multinível centrada na observabilidade conjugada das aplicações e das camadas de infraestrutura; e ii) programabilidade granular da infraestrutura.

Até o momento, apesar de seu objetivo ser o de atender às aplicações, o processo de orquestração é normalmente todo concebido apenas na infraestrutura. Desta forma a sua

implementação, assim como o que a orquestração observa e onde atua estão concentrados na infraestrutura do sistema.

Comumente, todo o processo é realizado através de inferências de seu comportamento e não via informações fornecidas pelas próprias aplicações. Ainda que esse modelo seja adequado em muitas situações, ele não é suficiente para o domínio que trata esse trabalho. Desse modo, sem uma visão conjunta dos diferentes níveis da arquitetura, não é possível que tanto os requisitos específicos das aplicações, quanto os requisitos rigorosos e correlacionados da infraestrutura sejam atendidos.

Adicionalmente, a alocação de recursos dentro do processo de orquestração será tão preciso quanto a capacidade de controlar cada elemento dentro da arquitetura. A programabilidade granular da infraestrutura permite um controle detalhado dos componentes envolvidos no atendimento dos requisitos da aplicação. Dessa forma, previne-se o sub ou super dimensionamento habilitando o uso racional dos recursos.

Como resultado desse trabalho, foi proposta uma arquitetura para Espaços Inteligentes baseados em visão computacional. Essa arquitetura foi implementada em diferentes ambientes, com múltiplas aplicações e estudos de caso foram realizados como prova de conceito para sua validação.

1.1 Questão de Pesquisa

Este trabalho tem como objetivo responder a seguinte questão de pesquisa:

Como atender tanto aos *requisitos específicos* das aplicações quanto aos *requisitos rigorosos e correlacionados* da infraestrutura presentes no domínio de Espaços Inteligentes baseados em visão computacional, garantindo o *uso racional dos recursos* disponíveis na infraestrutura.

1.2 Hipóteses

Neste trabalho, propomos as seguintes hipóteses para endereçar a questão de pesquisa:

- Hipótese H1: O paradigma de orquestração precisa ser estendido para um modelo **multinível centrado na observabilidade** conjugada das aplicações e das camadas de infraestrutura.

Justificativa H1 : O processo de orquestração tem sido todo concebido na infraestrutura. Conforme já mencionado, apesar de seu objetivo ser atender às aplicações, a orquestração é implementada, observa e atua apenas na própria infraestrutura.

Ainda que esse modelo seja adequado em muitas situações, ele não é suficiente para o domínio que trata esse trabalho.

- Hipótese H2: Além das extensões funcionais propostas em H1, assumimos que a **programabilidade granular de infraestrutura** habilita o gerenciamento racional dos recursos, através do controle detalhado de cada componente da infraestrutura. **Justificativa H2** : Uma infraestrutura com capacidade de programação limitada, tende a levar a um sub ou super dimensionamento dos recursos. Em um contexto de alta dinamicidade dos requisitos de múltiplas aplicações de visão computacional, o problema da correta alocação de recursos de uma infraestrutura compartilhada torna-se ainda mais crucial.

1.3 Objetivos

Para responder os desafios apresentados, foram planejados os seguintes objetivos:

- Objetivo 1: Apresentar uma proposta de arquitetura projetada de forma a endereçar a questão de pesquisa apresentada.
- Objetivo 2: Demonstrar o funcionamento da arquitetura e sua generalização para o domínio de aplicação definido. Para isso, múltiplos espaços inteligentes foram implantados guiados pela arquitetura proposta, em diferentes localidades.
- Objetivo 3: Validar a arquitetura a partir de 2 estudos de casos utilizados como prova de conceito.

1.4 Estrutura do trabalho

Os demais capítulos dessa tese estão estruturados da forma descrita a seguir. O Capítulo 2 faz uma contextualização sobre Espaços Inteligentes e trabalhos relacionados. No Capítulo 3 é apresentada a arquitetura do Espaço Inteligente Programável baseado em visão computacional. O Capítulo 4 mostra as implementações realizadas baseadas na arquitetura proposta. A seguir, dois estudos de caso são apresentados nos Capítulos 5 e 6 como prova de conceito. E finalmente no Capítulo 7 são apresentadas as conclusões e contribuições do trabalho realizado.

2 Referencial Teórico e Trabalhos Relacionados

Os Espaços Inteligentes atualmente são muito diversos, podendo variar consideravelmente em sua finalidade e concepção. Porém algumas características são comumente encontradas nos Espaços Inteligentes atuais e nos ajudam a melhor classificá-los e entendê-los. Nesse contexto, podemos elencar a abrangência do Espaço Inteligente, a classe de aplicações que nele são executadas e a infraestrutura usada como suporte a essas aplicações como características importantes para o projeto e implementação de Espaços Inteligentes.

Porém, através da análise dessas três características será demonstrado que para superar os desafios específicos dos espaços inteligentes baseados em visão computacional, é necessário que novos habilitadores sejam incorporados.

Por esse motivo, este trabalho tem como objetivo apresentar a proposta de uma nova classe de Espaços Inteligentes, aqui chamado de Espaço Inteligente Programável (PIS, do inglês Programmable Intelligent Space). A diferença entre os Espaços Inteligentes atuais e o PIS é que à esse último foram incorporados dois novos habilitadores: a orquestração multinível centrada na observabilidade e programabilidade granular de infraestrutura.

Além disso, toda a arquitetura foi projetada utilizando um modelo baseado em microsserviços, que se mostra altamente aderente às características do PIS com foco em atender aos objetivos principais desse trabalho.

A seguir serão discutidos os conceitos de microsserviços e as características dos Espaços Inteligentes atuais.

2.1 Arquitetura baseada em Microsserviços

Serviço pode ser definido como uma funcionalidade de software reutilizável que pode ser usada por vários clientes para diferentes propósitos. Ele implementa um elemento específico do domínio, define sua interface e pode ser usado independentemente na rede (CERNY; DONAHOO; TRNKA, 2018).

Já segundo (FOWLER; LEWIS, 2014), o que define uma arquitetura baseada em microsserviço é uma abordagem onde cada bloco funcional da arquitetura é desenvolvido como um conjunto de pequenos serviços, cada um executando seu próprio processo de forma independente e se comunicando de forma leve com os demais serviços.

Essa abordagem tem como resultado uma arquitetura com um grande número de pequenos serviços independentes e fracamente acoplados, sendo sua granulação variável a

depende dos requisitos de cada bloco funcional.

Esses serviços são construídos em torno de uma função delimitada e auto contida dentro do escopo do domínio e funcionam através de mecanismos de implantação independentes totalmente automatizados. Há uma necessidade mínima de gerenciamento centralizado desses serviços, que podem ser escritos em diferentes linguagens de programação e utilizam diferentes tecnologias de armazenamento de dados. Algumas características são frequentemente encontradas em arquiteturas baseadas em microsserviços e estão presentes nessa arquitetura proposta.

Microsserviços são desenvolvidos para executarem uma tarefa no contexto de um domínio. Essa tarefa, em alguns casos conhecida como capacidade de negócio, é desenvolvida por uma equipe que assume responsabilidade por todo o seu desenvolvimento. Esse desenvolvimento pode incluir desde a interface com o usuário, armazenamento de dados até interações com outros serviços. A criação de equipes por capacidade de negócio ou tarefa, visa proporcionar a independência no desenvolvimento dos microsserviços. Uma pequena equipe pode escrever e manter esse microsserviço sem a necessidade de interações com outras equipes.

Com a tarefa ou capacidade de negócio bem delimitada, os microsserviços devem evitar a necessidade de qualquer outro elemento externo para cumprir a sua tarefa. Para isso, eles devem ter a inteligência necessária para, ao receber uma solicitação, aplicar a lógica definida conforme a regra de negócio e produzir uma resposta. Com a inteligência implementada dentro dos microsserviços, a comunicação entre eles deve ser simples, como por exemplo o modelo Hypertext Transfer Protocol (HTTP)/Representational State Transfer (REST) ou o modelo Publish (PUB)/Subscribe (SUB). Essa abordagem evita a necessidade de protocolos complexos ou ferramentas de orquestração centralizadas.

Os microsserviços bem delimitados e desacoplados permitem que seu desenvolvimento seja realizado utilizando diferentes tecnologias. Desde que se mantenham as interfaces de comunicação bem definidas os detalhes da implementação interna de cada serviço ficam ocultos de outros serviços. Por exemplo, a escolha pela utilização de uma determinada linguagem de programação não gera nenhum impacto em qualquer outro microsserviço. Da mesma forma que o uso de uma base de dados não afeta a decisão de armazenamento de um outro microsserviço.

A independência dos microsserviços resulta em uma outra característica para a arquitetura que é a facilidade de implantação e automação desse processo. Microsserviços devem ser capazes de ser iniciados e finalizados sem que afetem os demais serviços. A atuação em função de uma falha, ou mesmo uma atualização de um serviço pode ser feita sem a necessidade de coordenação entre as demais equipes.

Essa característica possibilita que sejam utilizadas técnicas de automação de

infraestrutura para todo o ciclo de vida do microsserviço. Particularmente, a evolução da computação em nuvem trouxe ganhos operacionais para a construção, implantação e operação dos microsserviços nesses ambientes.

Esse conjunto de características de uma arquitetura baseada em microsserviços fornece uma série de vantagens para o desenvolvimento do PIS. A agilidade de gerenciamento é uma delas.

O fato de microsserviços serem pequenos faz com que sua inicialização seja rápida. Consideremos o exemplo em que uma aplicação tem como requisito um limite para o tempo de resposta de uma tarefa. Em um determinado momento durante a execução da aplicação esse tempo de resposta foi excedido. Como forma de atender a esse requisito, o PIS realiza a migração de um dos microsserviços de um servidor para outro de maior capacidade. Desta forma o tempo de processamento do microsserviço em questão será diminuído permitindo o atendimento do requisito de tempo de resposta da aplicação.

Porém a execução desse tipo de ação requer um processo ágil de migração. Serviços muito grandes ou mesmo com um alto grau de dependência com outros serviços podem inviabilizar esse tipo de ação devido ao tempo necessário para finalização e inicialização do serviço.

Ainda que o processo de inicialização e finalização de microsserviços seja ágil, para algumas aplicações com requisitos ainda mais restritos, o tempo utilizado para a migração de serviços pode ser excessivo. Para esses casos, uma alternativa seria a inicialização de uma segunda instância do microsserviço no servidor de destino. Posteriormente seria feita a migração da comunicação para essa nova instância e só depois a instância anterior seria finalizada. Essa abordagem faz com que não haja interrupção no fluxo de execução da aplicação para os casos onde isso não é admissível.

O problema dessa abordagem é uma maior utilização dos recursos em função da duplicidade de instâncias executadas de forma simultânea durante o período de tempo da migração. Porém, a maior utilização dos recursos é minimizada pelo menor tempo de migração dos microsserviços.

Dessa forma, tanto o atendimento aos requisitos específicos da aplicação quanto o uso mais racional dos recursos da infraestrutura, que são os objetivos principais do PIS, são facilitados pela agilidade conferida pela arquitetura baseada em microsserviços.

Uma outra vantagem que uma arquitetura baseada em microsserviços fornece é a possibilidade de atender aos requisitos de uma aplicação de forma modular. Uma aplicação pode possuir requisitos de alto nível. Nesse caso, todos os componentes dessa aplicação tem um grau de responsabilidade para atendimento a esse requisito. Ao se decompor a aplicação, cada um dos microsserviços resultantes pode possuir suas próprias características e requisitos específicos. Como são independentes e desacoplados, há baixa interferência

entre eles.

Tomemos como exemplo uma aplicação composta por 2 microsserviços que rodam em um espaço inteligente composto por N câmeras. O primeiro microsserviço é responsável por receber as imagens em uma fila, localizar um determinado objeto em cada uma dessas imagens e gerar como saída a localização do objeto na imagem. Já o segundo serviço recebe a localização dos objetos nas imagens e faz o rastreamento desse objeto no ambiente. O primeiro serviço é altamente paralelizável, podendo ser iniciadas tantas instâncias quanto forem necessárias para processar as imagens dentro de um tempo limite máximo. Já o segundo serviço não é paralelizável, pois usa informações temporais para a tarefa de rastreamento. Caso as tarefas fossem executadas em um único serviço, a paralelização não seria possível.

Consideremos ainda que a tarefa de localização de objetos ocupa pouca memória e foi desenvolvido para ser executado em uma GPU. Já a tarefa de rastreamento não possui benefício em ser executado em GPU e tem um alto consumo de memória. Caso as tarefas estivessem agrupadas, o espaço inteligente teria que alocar GPUs mais robustas que possuísem mais memória ou um maior número de GPUs para possibilitar a execução da aplicação. Como resultado haveria um desperdício de recursos alocados com um desempenho possivelmente pior. Novamente aqui, a utilização de uma arquitetura baseada em microsserviços possibilitaria o gerenciamento mais granular dos recursos, tornando sua utilização mais adequada para alcançar os objetivos principais do PIS.

Um Espaço Inteligente baseado em Visão Computacional pode possuir abrangências diversas. A arquitetura do PIS deve ser capaz de lidar com ambientes de uma sala com poucas câmeras até ambientes maiores, como uma cidade. Um aumento na escala no ambiente gera uma série de dificultadores que devem ser tratados para o pleno funcionamento do Espaço Inteligente.

Considere como exemplo o controle semafórico para agilizar o trânsito de carros de serviço, como ambulâncias e viaturas de polícia. Para uma aplicação como essa, é necessário a captura e processamento das imagens da cidade. Em um modelo centralizado, todas as imagens seriam transferidas para um único ponto de alta capacidade, onde seriam processadas e o resultado seria enviado para diferentes semáforos espalhados pela cidade.

Esse modelo possui alguns problemas. Com o aumento do número de câmeras, irão aparecer congestionamentos na rede para transmissão de todas as imagens para um único ponto. Retransmissões, aumento da latência e perda de pacotes vão se tornando cada vez mais frequentes até um ponto que inviabilize o funcionamento do Espaço. O aumento da capacidade de processamento em um único ponto pode se tornar inviável a partir de determinados limites.

A divisão das tarefas em diferentes microsserviços possibilita a execução de múltiplas

instâncias em diferentes localidades. A medida em que se atinge o limite de um link ou de um servidor, novas instâncias podem ser alocadas em outros locais permitindo um aumento de escala do ambiente.

Apesar de uma arquitetura baseada em microsserviços trazer diversas vantagens, muitos desafios são introduzidos com a sua utilização. Talvez o principal problema seja a complexidade desse modelo.

Uma aplicação é composta em diferentes serviços independentes. Cada um desses microsserviços é responsável por sua própria tarefa e por isso não tem ciência sobre os demais. Devido a isso, a arquitetura deve possuir mecanismos que garantam o pleno funcionamento de toda a aplicação. Por exemplo, o monitoramento do tempo de resposta de uma tarefa da aplicação pode ser composta pelo tempo de processamento de diferentes microsserviços, mais o tempo de comunicação entre esses serviços. Essa métrica deve ser monitorada por um elemento externo aos serviços. Da mesma forma, outro bloco funcional deverá definir uma alocação dos recursos para que um terceiro bloco atue no gerenciamento dos microsserviços. A atuação conjunta desses blocos torna a arquitetura capaz de buscar o cumprimento desse requisito. Esses e outros blocos funcionais devem estar presentes na arquitetura de forma a atender aos objetivos principais do PIS.

A complexidade do modelo provoca também dificuldades para descoberta e correção de falhas da aplicação. Falhas do próprio microsserviço são mais simples de serem identificadas e corrigidas devido ao baixo acoplamento entre os microsserviços e sua divisão por capacidade de negócio. Porém, ainda que cada um dos microsserviços estejam funcionando corretamente, a aplicação como um todo pode não estar gerando os resultados esperados. Nesses casos, práticas como a correlação de informações através de logs e eventos de cada microsserviço, apresentam-se como ferramentas importantes para solução desse tipo de problema.

Outro desafio importante é a definição do tamanho dos microsserviços. A divisão de microsserviços de tamanho muito pequeno tem como consequência um aumento da comunicação entre serviços. Esse cenário gera longas cadeias de serviços onde o tempo total de comunicação pode inviabilizar o funcionamento do sistema.

Considere como exemplo uma aplicação de controle de um robô móvel. Na divisão da aplicação em microsserviços, um deles tem a tarefa de planejar uma trajetória, e o outro tem a tarefa de controle do robô. Diferentemente de serviços de processamento de imagens, ambos os microsserviços possuem o tempo de processamento muito baixo, porém o tempo de comunicação entre eles pode chegar a mais de uma ordem de grandeza maior. Nesse caso, se levar em consideração que ambos são desacoplados das demais tarefas, faria sentido a sua divisão. Porém, a excessiva sobrecarga de comunicação deve ser levada em consideração para a decisão de manter ambas as tarefas em um único microsserviço.

Para se diminuir a sobrecarga de comunicação, além da definição da granularidade dos microsserviços, é necessário cuidado no projeto das suas interfaces. As interfaces devem modelar o domínio e não a implementação interna do serviço. As interfaces devem ser simples e leves. A utilização padrões de serialização e comunicação assíncrona auxiliam no projeto de interfaces de comunicação com essas características.

2.2 Características dos Espaços Inteligentes Atuais

2.2.1 Abrangência

A abrangência de um Espaço Inteligente tem como principal delimitador o espaço físico por ele ocupado, podendo variar de uma pequena sala ([YAMAZAKI et al., 2020](#)) a uma grande região metropolitana ([MICHAELA; HORÁK, 2020](#)).

Além do tamanho de um espaço físico, eles podem variar também em função do tipo do espaço. Por exemplo, o fato de um espaço ser indoor ou outdoor leva a abordagens diferentes para torná-los inteligentes. Espaços indoor comumente são mais estruturados, permitindo uma flexibilidade maior para a sua instrumentação. Um prédio novo e inteligente a ser construído em uma grande cidade é muito mais simples de ser implementado do que um parque natural ou sítio arqueológico.

Além do espaço físico, um segundo delimitador da abrangência de um espaço Inteligente é o alcance de sua rede de sensores e atuadores. Tome como exemplo uma cidade inteligente que provê o serviço de controle de tráfego ([BHATT; TIWARI, 2019](#)). De acordo com as condições do trânsito em uma determinada região, os semáforos e placas de sinalização podem ser alterados, diminuindo o tempo médio de engarrafamentos. Porém a falta de sensores de velocidade ou semáforos programáveis em determinados pontos limitam a abrangência da cidade inteligente.

Tomando como base um espaço inteligente baseado em visão computacional, ainda que se possa utilizar diferentes sensores, câmeras são o principal dispositivo que fornece informações ao Espaço Inteligente. Considerando o grande volume de câmeras existentes nos mais variados ambientes nos dias atuais, conferir uma maior abrangência a tais espaços inteligentes requer um menor esforço. Apenas os pontos de interesse sem cobertura das câmeras precisariam de intervenção. E ainda assim, essa cobertura adicional poderia ser realizada em fases posteriores.

Como exemplo, podemos citar a cidade de Vitória (ES - Brasil). É uma cidade de 96.536KM² e que possui aproximadamente 350 câmeras operadas pelo setor de segurança municipal. Recentemente o governo municipal começou a implementar uma infraestrutura de suporte e diversos serviços à população usando as câmeras como principal sensor. Ainda que sua cobertura não seja completa, a ideia é transformar em um primeiro momento,

a parte da cidade com maior movimento de pessoas em uma área inteligente. Até que a cobertura completa seja alcançada em fases posteriores.

É importante ressaltar que a cobertura de câmeras em um ambiente pode não ser suficiente para o pleno funcionamento dos serviços a serem providos pelo espaço inteligente baseado em visão computacional. Problemas com interferência luminosa, qualidade das imagens ou a falta de intercessão na cobertura das câmeras são problemas comuns a este tipo de espaço que fazem com que na prática a sua abrangência seja reduzida.

Um terceiro delimitador que afeta a abrangência do espaço inteligente são as políticas definidas na utilização daquele espaço. Por exemplo, câmeras instaladas nas ruas de uma cidade podem ser utilizadas para reconhecimento de situações anômalas na área de segurança pública (SRIVASTAVA; BISHT; NARAYAN, 2017), porém caso a cobertura das câmeras alcance uma propriedade privada, essa parte da imagem deve ser descartada. Um outro caso pode ser observado em bancos. Apesar da ampla cobertura em agências, um cuidado deve ser tomado para evitar a captura de imagens dos clientes ao digitarem as suas senhas.

Todos esses aspectos devem ser levados em consideração para definição da abrangência dos espaços inteligentes. Considerando que esta definição é o primeiro e importante passo para o projeto desses espaços.

2.2.2 Aplicações

As aplicações que são executadas em um espaço inteligente devem fazer uso dos dados obtidos da rede de sensores instalados, processar esses dados para extrair informações, tomar decisões a partir dessas informações e atuar no ambiente via rede de atuadores. Existem inúmeras aplicações diferentes que podem ser executadas em espaços inteligentes. Cada uma delas possui características e requisitos específicos e podem ser classificadas de diferentes formas.

A classe de aplicações que esse trabalho tem por objeto de estudo é a classe de aplicações de visão computacional. Essa classe é importante por possibilitar a extração de uma grande quantidade de informação a partir de imagens que são utilizadas como dados de entrada.

Podemos observar diferentes aplicações dessa classe na literatura. Em (CHEN; CHEN; CHEN, 2018) os autores desenvolveram um sistema móvel de atendimento e orientação em um campus. Em (WANG et al., 2020) é apresentado um sistema de monitoramento de aprendizado e engajamento de estudantes em cursos on-line. Uma outra aplicação trabalha no reconhecimento de atividades de pessoas através da interação com robôs móveis (KUMRAI et al., 2020). Já em (RAZAVI; HAMIDKHANI; SADEGHI, 2019) é analisado o fluxo de carros em vias para controle de tráfego.

Imagens também podem ser usadas para realizar medidas indiretas de diferentes grandezas, como por exemplo: temperatura, velocidade ou distância. Ainda que seja possível a utilização de outros sensores para medição dessas grandezas, as câmeras possuem flexibilidade suficiente para poder suprir a ausência ou servir como redundância para essas medições.

Um outro tipo de aplicação que vem transformando os espaços cada vez mais inteligentes são aquelas que buscam analisar e prever o comportamento de pessoas ou coisas em um fluxo de imagens. Em (SANTOS et al., 2020) os autores buscam prever a ação executada por uma pessoa antes dela ser finalizada. Já em (BENTO; VASSALLO; SAMATELO, 2020) anomalias são detectadas podendo ser utilizadas em situações de prevenção a crimes em uma cidade.

Todas essas aplicações possuem um conjunto comum de características. Podemos começar citando o grande volume de dados gerados pelas câmeras que servem de entrada para essa classe de aplicações. Ainda que o volume de dados possa variar em função de sua resolução ou número de canais por exemplo, o volume de dados gerado por câmeras encontra-se algumas ordens de grandeza maior do que a maioria dos outros sensores. Enquanto sensores de presença ou de temperatura geram alguns bytes por amostra, as câmeras podem gerar comumente imagens com vários Mbytes.

O grande volume de dados, em conjunto com a alta complexidade dos algoritmos usados para análise e processamento desses dados, faz com que essa classe de aplicações demande um alto poder computacional. Em função dessa alta demanda, muitas dessas aplicações são desenvolvidas para tirar vantagem de hardware específico como GPUs ou FPGAs.

Esse tipo de hardware tira a vantagem de uma outra característica dessa classe de aplicações, que é o alto grau de paralelização. É comum que uma mesma instrução seja executada com um volume grande e independente de dados. Cálculos matriciais são um exemplo desse tipo de instrução. Nesses casos, não é necessário que um primeiro conjunto de dados seja executado, para que só depois de finalizado o processamento de um segundo conjunto seja iniciado. Vários conjuntos de dados podem ser processados simultaneamente, tornando o uso de placas do tipo Graphics Processing Unit (GPU) ou Field Programmable Gate Array (FPGA) altamente recomendável.

Além disso, é comum que aplicações de visão computacional que fazem parte de um espaço inteligente sejam executadas em tempo real. Quer dizer que para que essas aplicações sejam executadas corretamente, elas dependem não somente da lógica, mas também do tempo que demoram para serem executadas. A entrega de informação ou a execução de uma ação em um tempo pré estabelecido pode ser crítica podendo fazer com que a aplicação se torne inútil ou até mesmo perigosa caso esse tempo não seja cumprido.

Devido a essas características, um Espaço Inteligente baseado em visão computacional deve ser capaz de atender a um conjunto de requisitos rigorosos e correlacionados da infraestrutura. No contexto desse trabalho, serão destacados os requisitos de: taxa de transferência de dados, capacidade de processamento e tempo de resposta.

Taxa de transferência de dados

Em um espaço inteligente baseado em visão computacional, a maior parte dos dados são gerados por um conjunto de câmeras e enviados para a aplicação. O Espaço deve ser capaz de prover uma taxa de transferência adequada que evite atrasos ou perda desses dados por gargalos em sua infraestrutura.

Porém o fluxo de dados pode ser mais complexo do que pode parecer à primeira vista. Múltiplas aplicações podem fazer uso dos dados gerados pelas câmeras, fazendo com que se tenha múltiplas conexões multicast na infraestrutura. Além disso, aplicações podem gerar outras imagens que servirão como entrada para outras aplicações. Outro comportamento a ser considerado é a possibilidade de utilização de imagens não atuais obrigando a utilização de mecanismos de desacoplamento temporal, como a possibilidade de criação de filas e armazenamento de dados.

Todas essas características levam a um complexo fluxo de dados de alto volume. E o Espaço Inteligente deve ser capaz de atender às taxas de transferência de dados de cada conexão entre entidades, sejam elas sensores, aplicações ou qualquer outro serviço que faça parte do espaço.

Tempo de Resposta

Neste trabalho o tempo de resposta será definido como o tempo usado para execução de uma determinada tarefa da aplicação. Por exemplo, o estudo de caso do Capítulo 5 descreve uma aplicação de controle visual de robôs em tempo real. Nesta aplicação, o tempo de resposta é medido do momento entre a captura das imagens pelas câmeras até o envio da ação de controle ao robô. Esse tempo é composto pelo tempo de processamento de diferentes partes de uma aplicação somada ao tempo de comunicação utilizado para transferência de dados entre essas partes. Neste caso, o tempo de resposta é um requisito crítico da aplicação, devendo ser menor ou igual ao intervalo entre amostras. Caso isso não ocorra, o controle do robô pode ser prejudicado ou até mesmo inviabilizado.

O tempo de processamento é uma medida realizada pela própria aplicação. Já o tempo de comunicação é uma medida realizada pela infraestrutura. Ambas as medidas são necessárias para a composição e o cumprimento do requisito de tempo de resposta e o Espaço Inteligente deverá fornecer mecanismos para realizar essas medidas e completar a tarefa dentro do tempo demandado.

Capacidade de Processamento

Devido ao grande volume de dados gerados pelas câmeras, a dificuldade da análise das imagens e a restrição de tempo para seu processamento, exige-se do espaço inteligente um poder computacional extremamente elevado. Além disso, a alocação de recursos para prover a capacidade de processamento necessária pode se dar de diferentes formas.

Aplicações baseadas em múltiplas threads, podem se beneficiar com a alocação de múltiplos núcleos de processador. Já aplicações monolíticas altamente acopladas requerem um processador com maior capacidade. De forma semelhante, aplicações podem ser desenvolvidas para serem executadas em hardware específico, como mencionado anteriormente.

Essas diferentes opções tornam a tarefa de alocação de recursos para atender ao requisito de processamento, bastante complexa. Uma alocação inadequada dos recursos, pode ter como resultado o não cumprimento da capacidade de processamento requisitada pela aplicação. Um ponto importante a se ressaltar a respeito desses requisitos, é que todos eles podem ser alterados sob demanda. E essas variações de demanda podem ocorrer em diferentes aplicações simultaneamente.

Todas essas questões, elevam ainda mais a complexidade do ambiente, fazendo com que o desenvolvimento de um Espaço Inteligente que atenda a esses requisitos rigorosos e correlacionados se torne um grande desafio.

2.2.3 Infraestrutura

A infraestrutura de um Espaço Inteligente tem como função interconectar todos os sensores, atuadores e recursos computacionais que fazem parte do Espaço Inteligente. Para isso, são utilizadas diferentes tecnologias de rede que se mostrem mais adequadas para promover a conectividade necessária a cada um dos seus elementos.

Além disso, a infraestrutura deve servir de ambiente de armazenamento e execução das aplicações desenvolvidas para o ambiente, através do fornecimento de recursos de forma a atender aos requisitos dessas aplicações.

A infraestrutura é composta de elementos de hardware e software. Como elementos de hardware para comunicação, podemos citar o uso de switches, access points, antenas e até mesmo o cabeamento necessário para interligação desses dispositivos. Já para armazenamento e execução das aplicações, servidores, storages, placas GPU são exemplos de dispositivos de computação que podem ser utilizados como infraestrutura de um Espaço Inteligente.

Porém, apenas os dispositivos de hardware não são suficientes para que as aplicações funcionem adequadamente no ambiente. Existe toda uma infraestrutura de software que suporta às aplicações. Esses componentes de software desempenham as mais diferentes

funções, sem que necessariamente façam parte da aplicação. Na verdade, em muitos casos a aplicação nem mesmo tem conhecimento de sua utilização.

Sistemas operacionais são amplamente conhecidos e muitas vezes são parte do requisito de muitas aplicações. Já hipervisores e gerenciadores de containers, utilizados para virtualização, podem ser transparentes para as aplicações. Na linha de oferecer um ambiente funcional mas que ao mesmo tempo seja de fácil utilização pelos desenvolvedores das aplicações, a infraestrutura de um espaço inteligente pode prover desde load balance e barramento de mensagens até geradores de código automático para acesso ao Espaço Inteligente.

Um outro conjunto de ferramentas é aquele utilizado para o gerenciamento dos recursos computacionais e de computação, com o objetivo de atender aos requisitos das aplicações. Dentro desse conjunto se destacam os orquestradores, monitores e gerenciadores de recursos. Eles são responsáveis por coletar informações, analisar, tomar decisões e controlar os recursos computacionais do ambiente.

Em sua maioria, Espaços Inteligentes são construídos com foco nas aplicações. A infraestrutura é pensada apenas para atender ao requisito funcional da aplicação. Em (STOYANOV; OROZOV; POPCHEV, 2018) foi desenvolvida uma plataforma voltada para uma aplicação de monitoramento de qualidade da água do mar em regiões costeiras. Já em (YUAN, 2019), foi construído um estacionamento inteligente vertical. Na área de energia elétrica foram desenvolvidos sistemas de iluminação pública inteligente em (LEE; SHIN; LEE, 2019) e um redutor de perdas em linhas de energia em (TAUQIR; HABIB, 2019). Todos os trabalhos têm em comum uma infraestrutura extremamente rígida voltada a atender ao funcionamento de uma única aplicação sem preocupação com fatores como desempenho, escalabilidade ou racionalidade no uso de recursos computacionais, por exemplo.

Mesmo trabalhos que tratam de aplicações dentro do domínio de visão computacional, que possuem requisitos bastante rigorosos, a infraestrutura apresentada é bastante simples e rígida. Exemplos podem ser observados em (SPRUTE et al., 2018) e (KABUTAN; NISHIDA, 2016), onde ambos utilizam uma rede de câmeras para controle de robôs.

Porém recentemente, diferentes Espaços Inteligentes passaram a ter uma preocupação maior com o projeto de sua infraestrutura. Com o aumento desses ambientes, tanto em sua abrangência quanto no número de diversidade de aplicações, a escalabilidade se tornou uma característica necessária a esses espaços. (ABBAS et al., 2020) enumera as vantagens de unir computação em nuvem com computação pervasiva. Além da computação em nuvem como forma de permitir a escalabilidade, (BHARDWAJ et al., 2018) apresenta uma arquitetura que busca resolver a questão da interoperabilidade de dispositivos heterogêneos. A questão da interoperabilidade também é tratada em (TOSKOV et al., 2020), onde ele apresenta uma arquitetura baseada em entidades virtuais de forma a facilitar mudanças e

a interação entre os dispositivos. Da mesma forma que em (CARROZZO et al., 2018) que propõe uma arquitetura para federação entre diferentes espaços inteligentes.

A computação em nuvem busca resolver principalmente o problema de recursos computacionais necessários devido ao aumento dos Espaços Inteligentes. Porém, o aumento desses espaços tem como consequência, o aumento no volume de dados. Para que a conexão com a nuvem não se torne um gargalo do ambiente, algumas arquiteturas, tal como (CHEN et al., 2016) e (✓CUROVÁ et al., 2017) passaram a utilizar a combinação entre computação em nuvem e computação em borda. Além da questão do volume de dados, a latência é um outro requisito importante para aplicações de visão computacional. (Ben Sada et al., 2019) e (CICCONETTI; CONTI; PASSARELLA, 2019) buscam a diminuição da latência através da alocação dos recursos entre a borda e a nuvem.

Como apresentado anteriormente, o aumento do volume de dados trafegados, o aumento do poder computacional e a diminuição do tempo de resposta podem estar correlacionados. (WANG et al., 2020) realiza um levantamento sobre como esses requisitos rigorosamente correlacionados são tratados para aplicações que utilizam aprendizado profundo e mostra as lacunas ainda existentes. Já em (PAHL; CARLE; KLINKER, 2016) e (WANG; JIN, 2019) abordam a questão da necessidade de alocação dinâmica dos recursos de um Espaço Inteligente.

2.3 Novas funcionalidades introduzidas pelo PIS

Como comentado anteriormente, os Espaços Inteligentes atuais possuem como principais características que o definem: a abrangência, aplicações e infraestrutura. Porém, para superar os desafios apresentados pelos Espaços Inteligentes baseados em visão computacional, esse trabalho propõe a utilização do PIS. Além das características já discutidas, o PIS adiciona dois novos habilitadores: a orquestração multinível centrada na observabilidade e programabilidade granular de infraestrutura.

2.3.1 Orquestração multinível centrada na Observabilidade

Espaços Inteligentes possuem duas classes de usuários bem distintas. Uma classe é de desenvolvedores que atuam no nível de aplicação e a outra é de projetistas ou operadores de infraestrutura. Cada uma dessas classes possui uma visão distinta uma da outra.

Desenvolvedores de aplicações utilizam o Espaço Inteligente como uma plataforma para execução das suas aplicações. Como usuários, eles possuem uma visão limitada das características e detalhes de funcionamento da infraestrutura. A eles o que importa é que a aplicação seja executada da forma como foi concebida e que o resultado esperado seja alcançado.

Já o projetista ou operador da infraestrutura não conhece ou não entende o funcionamento das aplicações. Comumente não é passada nenhuma informação a respeito do seu funcionamento.

Para exemplificar esse problema imagine uma aplicação de controle de um robô móvel que segue uma pessoa para auxiliá-la em uma determinada tarefa como visto em (ALMONFREY et al., 2018). Neste caso, um dos requisitos da aplicação pode ser a distância entre o robô e a pessoa. Essa distância deverá estar entre determinados valores de forma a viabilizar a interação entre pessoa e máquina. Nesse caso, o controle do robô e o cálculo da distância são afetados pelos recursos disponíveis à aplicação.

É nesse ponto que se tem um problema. Os recursos são gerenciados pela infraestrutura. Porém, o requisito é muito específico e somente a aplicação tem conhecimento sobre ele. É justamente essa lacuna de informação e conhecimento que gera um problema para o gerenciamento dos recursos, bem como ao atendimento dos requisitos da aplicação.

Taxa de utilização do processador (CPU, do inglês Central Process Unit) e memória, latência de um pacote na rede ou falha de um link são exemplos do tipo de informação sob o domínio da infraestrutura. Porém a distância entre robô e pessoa não está entre essas informações.

Em casos como esse em que a infraestrutura não conhece o funcionamento da aplicação, a infraestrutura gerencia os recursos através de inferências. Por exemplo, se uma aplicação ou parte dela tem uma taxa de utilização de CPU acima de um determinado patamar, uma CPU de maior capacidade pode ser alocada para a aplicação em questão. Em muitos casos essa abordagem é suficiente e resolve o problema de alocação.

De uma forma geral, provedores de infraestrutura utilizam diferentes modelos de alocação onde tentam evitar uma dependência de informações compartilhadas pelas aplicações. Como esse compartilhamento não está sob o controle dos provedores, uma estratégia de alocação de recursos baseada nesse compartilhamento pode se mostrar ineficaz. Como resultado, se tem uma inadequação do serviço oferecido sendo o provedor responsável por esse resultado.

Ainda que em diferentes contextos o não compartilhamento de informações entre a infraestrutura e a aplicação possa se mostrar adequada, em alguns casos não é suficiente. Em (PICORETI et al., 2018) os autores obtiveram resultados melhores com o uso de métricas como tempo de resposta, medido pela aplicação, do que com a utilização da métrica de taxa de utilização de CPU. Nesse artigo pode-se observar a complexidade da definição do threshold dessa métrica de infraestrutura com o objetivo de tentar inferir o comportamento da aplicação. Como resultado, pode-se observar uma maior utilização dos recursos com uma pior taxa de atendimento do requisito da aplicação.

Ainda que com resultados inferiores, experimentos como esse mostram que é possível

gerenciar os recursos da infraestrutura através da inferência das necessidades da aplicação através da observação de métricas da própria infraestrutura. Porém em um segundo experimento neste mesmo artigo, os autores apresentam um cenário onde a alocação de recursos depende exclusivamente de informações da aplicação.

Nesse segundo experimento, além de sua função primária onde seu controle foi descrito no primeiro experimento, ele possui uma segunda função de detectar pessoas no ambiente. Neste caso, a alta taxa de utilização de CPU por uma única instância, não indica a necessidade de mais recursos. Essa necessidade só se faz presente no momento em que uma pessoa entra no ambiente. Note que nesse caso, somente a aplicação pode fornecer esse tipo de informação.

Ambos experimentos mostram que a integração de informações entre os níveis de aplicação e infraestrutura resultam em um melhor gerenciamento dos recursos da infraestrutura, bem como em um melhor atendimento aos requisitos das aplicações. Uma questão que se apresenta é que informação e como ela deve ser compartilhada.

Uma aplicação pode ser composta por diferentes serviços que podem ser executados de forma independente. Porém, cada serviço pode ter como entrada a saída de um outro serviço. Nesse caso, esse encadeamento de serviços (SFC, do inglês Service Function Chain) deve ser informado pela aplicação de forma que a infraestrutura possa fazer a sua alocação.

Caso algum desses serviços que compõem uma aplicação seja paralelizável, novas instâncias podem ser iniciadas de forma a diminuir o tempo de resposta de uma tarefa. Note que aumentar o número de instâncias de um serviço pode não refletir em uma melhoria no tempo de resposta. Isto pode ocorrer se ele não for paralelizável ou possua um limite de paralelização por serviço.

Além dessas, outras informações como a utilização de GPUs, quantidade de memória alocada ou uso intensivo de leitura e escrita em unidades de armazenamento também podem auxiliar a infraestrutura a gerenciar melhor os recursos e ao mesmo tempo atender aos requisitos da aplicação. A alocação de recursos, sem que ele seja necessário pela aplicação tem como consequência seu desperdício, e em alguns casos até um mau funcionamento da aplicação.

Uma das abordagens para se buscar atender aos requisitos das aplicações é o estabelecimento de níveis de qualidade de serviço (QoS, do inglês Quality of Service). Nesse caso, parâmetros de funcionamento da infraestrutura são definidos de acordo com a necessidade da aplicação. (EL-KASSABI et al., 2019) propõe um modelo de orquestração dinâmica para computação em nuvem. O modelo prevê que parâmetros de QoS, como latência ou percentual de utilização de processamento, sejam definidos para cada aplicação. Após a definição, a infraestrutura monitora continuamente esses parâmetros e aloca dinamicamente os recursos de forma a atendê-los. De forma similar, (DEBAUCHE et al.,

2020) faz a alocação de recursos baseado em QoS usando requisitos rigorosos de aplicações de visão computacional.

Um problema dessa abordagem é que as aplicações são normalmente compostas de diferentes serviços dispersos na infraestrutura. Nesses casos, é necessário que os níveis de QoS sejam estabelecidos, não apenas para aplicação como um todo, mas sim para cada um dos serviços de sua composição. Em (BLANCO et al., 2020) é realizada uma alocação dinâmica fim a fim (E2E, do inglês End to End) de serviços críticos que compõe uma aplicação. De forma mais específica, (ZHANG et al., 2019) realiza a orquestração E2E para a classe de serviços de visão computacional.

A utilização de QoS para alocação de recursos possui o limitador de que a observabilidade do processo ocorre apenas no nível de infraestrutura. Nesse caso é necessário que a aplicação tenha ciência dos parâmetros dos requisitos da infraestrutura que ela necessita. (DANG; KHAN; SIVRIKAYA, 2019) apresenta uma arquitetura para redes móveis de quinta geração (5G, do inglês 5th Generation mobile network) com alocação de recursos ciente da aplicação (do inglês application aware). A arquitetura provê um modelo de descrição dos estados da aplicação, que tentam traduzir os requisitos da aplicação para os requisitos de infraestrutura.

Na mesma linha de arquiteturas cientes da aplicação, (GHOSH; NGUYEN; KRISHNAMACHARI, 2019) propõe uma arquitetura com foco na orquestração de serviços baseada em padrões. Os padrões são descrições do funcionamento de classes de aplicações e seu respectivo mapeamento em requisitos da infraestrutura. Caso uma aplicação não se enquadre em um padrão pré existente, um novo padrão pode ser construído.

Um passo adiante no processo de atender aos requisitos das aplicações em um ambiente desafiador composto por aplicações de visão computacional foi apresentado em (SONKOLY et al., 2020). O trabalho apresenta uma arquitetura para redes 5G com orquestração hierárquica multinível e multidomínio. A arquitetura trabalha com padrões para classe de aplicações, mas adiciona o conceito de afinidade entre aplicações e recursos e utiliza essas informações para o controle do ciclo de vida dos serviços.

As arquiteturas apresentadas não foram projetadas para Espaços Inteligentes, mas dado suas características, poderiam ser adotadas para esses ambientes. Porém, ainda que tais arquiteturas busquem atender as necessidades das aplicações, sua visão continua voltada para observar os requisitos da infraestrutura e atuar sobre eles. Uma abordagem diferente é feita em (LEE et al., 2020), onde é definida uma relação direta entre os requisitos de infraestrutura e os requisitos específicos das aplicações de visão computacional. A medida que as condições da infraestrutura mudam, isso é passado para a aplicação de forma que ela se adapte às novas condições. A observação de ambos os níveis é realizada para orquestração. O problema nesse caso é que são as aplicações que devem se adaptar à infraestrutura e não o contrário. Nenhum controle racional dos recursos de infra é realizado.

2.3.2 Programabilidade Granular de Infraestrutura

Em um artigo de 2005 publicado no IEEE Pervasive Computing (HELAL, 2005), foi argumentado que, os Espaços Inteligentes não deveriam ser projetados como ambientes monolíticos, mas sim como uma entidade programável. A ideia foi criar um novo paradigma onde o mundo físico fosse projetado de forma independente ao desenvolvimento das aplicações. E os desenvolvedores dessas aplicações interagem com o Espaço Inteligente da mesma forma que uma biblioteca de uma linguagem.

Para atender aos requisitos das aplicações, e ao mesmo tempo ser capaz de um gerenciamento racional dos recursos, o PIS deve seguir esse paradigma e ser programável em sua concepção.

Existem diferentes definições para o termo programabilidade. No contexto desse trabalho, programabilidade será definida como a capacidade de um elemento na arquitetura de receber diferentes instruções para executar determinadas tarefas. O comportamento desse elemento arquitetural pode ser modificado através da alteração das instruções ou de seus parâmetros.

Porém, para atingir o objetivo desse trabalho, a programabilidade deverá estar presente em diferentes níveis de sua arquitetura. Quanto maior o nível de programabilidade, maior o potencial para atendimento a esses objetivos.

A programabilidade pode-se apresentar de diferentes formas. A padronização de interfaces entre diferentes elementos dentro do PIS é uma delas. Por exemplo, o PIS pode possuir câmeras de diferentes fabricantes como sensores no ambiente. Cada um desses fabricantes pode possuir uma forma de comunicação e serviços específicos. É importante para os desenvolvedores de aplicações que o acesso a esses dispositivos seja feito de forma única, não importando o seu tipo.

Além da padronização para utilização dos sensores e atuadores, é importante que existam interfaces de programação (API, do inglês Application Programming Interface) para acesso a todos os elementos do PIS, tais como serviços e recursos computacionais.

Além da padronização de acesso a diferentes entidades, a escolha de que recurso utilizar ou a forma de seu funcionamento deve estar presente no PIS. Máquinas virtuais podem ser utilizadas como método de virtualização para uma determinada aplicação, já containers podem ser usados para uma segunda aplicação. Essa escolha pode se dar em função dos requisitos de cada uma delas. Porém, esses métodos de virtualização não devem ser rígidos. Eles podem ser alterados ou até mesmo combinados em função de mudanças no ambiente.

Da mesma forma, a comunicação entre dois serviços pode ser alterada em função de sua necessidade de desempenho ou funcionalidades. A utilização de um proxy de aplicação pode ser interessante pela facilidade de implementação de um balanceamento de carga. Já

a utilização regras SDN pode se mostrar mais conveniente em situações onde o desempenho da comunicação seja mais importante.

Podemos perceber através desses exemplos que a programabilidade granular em diferentes níveis, tais como aplicação, recursos computacionais e comunicação provê uma grande flexibilidade. Porém uma flexibilidade ainda maior pode ser necessária.

Para isso é necessário que a programabilidade se estenda até outros níveis ou sub níveis. No Capítulo 5, poderá ser observada alterações em parâmetros nas camadas física e enlace nas redes wireless e cabeada em tempo de execução. Essa foi a forma encontrada para atender ao requisito de tempo de resposta de uma aplicação em um ambiente industrial.

É importante ressaltar que o PIS foi concebido para execução de aplicações em tempo real. E essas aplicações possuem demandas que podem se alterar em tempo de execução.

Além disso, características como a possibilidade de reutilização de aplicações ou serviços e a necessidade de escalabilidade para atendimento a ambientes cada vez maiores, expõe a necessidade de uma programabilidade granular em diferentes níveis do PIS.

(XU; HELAL, 2016) cria uma abstração para acesso aos sensores e atuadores em um Espaço Inteligente. Além disso, uma API é criada para acesso dos desenvolvedores ao Espaço.

Já a capacidade de programar o acesso a rede sem fio é apresentada em (KORZUN; GALOV; LOMOV, 2017). Outras arquiteturas apresentam uma capacidade de programação da rede no domínio ótico, como em (FICHERA et al., 2019).

Uma granularidade de programação da infraestrutura importante principalmente para Espaços Inteligentes baseados em visão computacional é apresentada em (THINAKARAN et al., 2019). A arquitetura virtualiza não só os nós de computação como os servidores, mas também as GPUs. Dessa forma, é possível gerenciar a alocação desse recurso comumente raro e de alto custo.

Arquiteturas para rede 5G, como (UNIYAL et al., 2020), (MANO et al., 2020) e (BLANCO et al., 2020) possuem um grau de programabilidade ainda mais amplo. Essas arquiteturas possuem uma capacidade de se programar desde a rede, em seus vários subdomínios, quanto a infraestrutura de computação entre centros de processamento de dados (DC, do inglês Data Center). Essas arquiteturas permitem a utilização de técnicas com Máquinas Virtuais e Containers ou a utilização de redes SDN para alocação de recursos para os serviços. Tudo de forma dinâmica e escalável.

Porém, essas arquiteturas têm uma forte visão da infraestrutura e não proveem funcionalidades para as aplicações, como a reutilização e compartilhamento de serviços, por exemplo. E mesmo com a visão voltada para a infraestrutura, a busca pelo gerenciamento

racional dos recursos é feito de forma simples e dissociada da aplicação.

2.4 Análise Comparativa dos Trabalhos Relacionados

Na seção anterior foram apresentadas diferentes arquiteturas que podem ser utilizadas para Espaços Inteligentes baseados em visão computacional e foi realizada uma discussão de suas características e funcionalidades. Nessa seção, será apresentada uma consolidação dessas arquiteturas e suas características. Dessa forma fica mais simples e clara uma análise comparativa entre elas.

Na primeira coluna da Tabela 1 está a lista de trabalhos que apresentam as arquiteturas. Nas demais colunas, estão listadas características relevantes presentes em cada uma delas. A seguir segue uma breve explicação de cada uma dessas características:

1. Facilidade de uso: Os desenvolvedores de aplicações representam uma das classes de usuários dos espaços inteligentes. Normalmente possuem conhecimento restrito da infraestrutura o que dificulta o seu uso. Portanto, a arquitetura deverá prover uma padronização de acesso aos elementos dessa arquitetura e transparência no uso de seus recursos, através de interfaces claras e simples.
2. Reusabilidade de serviços: Muitas aplicações de visão computacional possui partes semelhantes que podem ser compartilhadas entre si. Considerando que uma aplicação é a composição de serviços, a arquitetura deve promover mecanismos para a sua reutilização em diferentes aplicações.
3. Escalabilidade: Espaços inteligentes podem ter qualquer dimensão. Sua arquitetura deve ser escalável de forma a atender a um aumento de sua abrangência ou um maior número de aplicações.
4. Heterogeneidade: Aqui trata-se da utilização de diferentes tipos de equipamentos e dispositivos no espaço inteligente e sua integração. A capacidade de se utilizar múltiplos protocolos e interfaces afetam diretamente esse item.
5. Requisitos rigorosos: Referem-se aos rigorosos requisitos de infraestrutura que a classe de aplicações de visão computacional exige.
6. Requisitos correlacionados: Diz respeito à necessidade de se atender aos requisitos rigorosamente correlacionados simultaneamente.
7. Alocação dinâmica de recursos: As aplicações dessa classe mudam os seus requisitos em tempo de execução e a arquitetura deve ter a capacidade de alterar os recursos dinamicamente para continuar atendendo a esses requisitos.

8. Uso racional de recursos: Alocação apenas dos recursos necessários para atendimento dos requisitos das aplicações, evitando-se o sub ou super dimensionamento.
9. Requisitos da aplicação: Trata-se da capacidade da arquitetura de monitorar os requisitos das aplicações e atuar na infraestrutura para que esses requisitos sejam atendidos. Nesse item será avaliado se a arquitetura será capaz de atender a diferentes requisitos das aplicações: (I) requisitos de infraestrutura das aplicações, como por exemplo QoS; (T) padrões que mapeiam cada aplicação em uma classe de requisitos de infraestrutura comuns; (E) requisitos específicos das aplicações.
10. Programabilidade Granular: Promove o controle detalhado de cada elemento da infraestrutura. Caso a arquitetura apresente apenas uma programabilidade parcial, na tabela será marcada com a letra P.

Os trabalhos apresentados na tabela podem ser divididos em algumas classes. A primeira delas são trabalhos que apresentam facilidade de uso como uma de suas principais características. Tais trabalhos foram desenvolvidos especificamente para espaços inteligentes com foco na aplicação. Porém eles comumente possuem uma infraestrutura simples e muito rígida, onde a alteração na aplicação muitas vezes inviabiliza o seu funcionamento.

O segundo grupo de trabalhos apresenta arquiteturas para espaços inteligentes com foco na infraestrutura. Eles têm em comum as características de escalabilidade e a capacidade de atender aos requisitos rigorosos de aplicações de visão computacional.

O terceiro conjunto corresponde às arquiteturas que não foram desenvolvidas para espaços inteligentes, mas podem ser adaptadas para tal. Dentro desse conjunto estão os trabalhos que buscam atender aos requisitos das aplicações seja por mecanismos de QoS, alocação de recursos fim a fim ou arquiteturas cientes das aplicações. Ainda que mostrem uma evolução principalmente no que diz respeito ao foco na aplicação, essas arquiteturas ainda possuem alguns problemas. O principal deles é que elas não avaliam os requisitos específicos das aplicações. Tentam agrupar as aplicações com características similares e montam uma relação simplificada entre esse grupo de aplicações e requisitos da infraestrutura. Além disso, possuem uma programabilidade limitada e não possuem a preocupação com o gerenciamento racional dos recursos.

O quarto grupo contém as arquiteturas que podem ser adaptadas para utilização em espaços inteligentes e correspondem àquelas voltadas para redes 5G. O foco dessas arquiteturas normalmente está no gerenciamento de recursos da infraestrutura, e portanto, possuem uma programabilidade granular elevada. Destaque desse grupo é a arquitetura apresentada em (SONKOLY et al., 2020), que apesar de não tratar dos requisitos específicos das aplicações, possui uma visão tanto do nível de aplicação quanto do nível de infraestrutura e busca fazer o gerenciamento de recursos.

Tabela 1 – Comparação entre os trabalhos relacionados e o PIS.

Arquiteturas	1	2	3	4	5	6	7	8	9	10
(SPRUTE et al., 2018)	x				x					
(KABUTAN; NISHIDA, 2016)	x				x					
(ABBAS et al., 2020)			x							
(BHARDWAJ et al., 2018)			x	x			x			
(TOSKOV et al., 2020)	x		x	x						
(CARROZZO et al., 2018)				x						
(CHEN et al., 2016)			x		x					
(XCUROVÁ et al., 2017)			x		x					
(Ben Sada et al., 2019)			x		x					
(CICCONETTI; CONTI; PASSARELLA, 2019)			x		x					
(WANG et al., 2020)			x	x	x	x				
(PAHL; CARLE; KLINKER, 2016)		x	x		x		x			x
(WANG; JIN, 2019)			x		x		x			
(EL-KASSABI et al., 2019)		x	x		x		x		I	
(DEBAUCHE et al., 2020)		x	x		x	x	x		I	
(BLANCO et al., 2020)		x	x		x		x		I	P
(ZHANG et al., 2019)		x	x		x	x	x		I	P
(DANG; KHAN; SIVRIKAYA, 2019)		x	x		x		x		T	P
(GHOSH; NGUYEN; KRISHNAMACHARI, 2019)		x	x		x		x	x	T	x
(SONKOLY et al., 2020)		x	x		x	x	x		T	P
(LEE et al., 2020)		x	x		x	x	x		E*	P
(XU; HELAL, 2016)	x	x	x		x		x	x		P
(KORZUN; GALOV; LOMOV, 2017)			x		x		x	x		P
(FICHERA et al., 2019)			x		x		x	x	I	P
(THINAKARAN et al., 2019)			x		x		x	x		P
(UNIYAL et al., 2020)			x		x		x	x		x
(MANSO et al., 2020)			x		x		x	x		x
(BLANCO et al., 2020)			x		x		x	x		x
PIS	x	x	x	x	x	x	x	x	E	x

Fonte: Produção do próprio autor.

De todas as arquiteturas estudadas, uma única apresentou a capacidade de monitorar e avaliar os requisitos específicos das aplicações. Em (LEE et al., 2020), os requisitos específicos de aplicações de visão computacional são monitorados e usados para atuação no ambiente. Ele estabelece uma relação direta entre os requisitos específicos e os recursos da infraestrutura através de uma observabilidade multinível. O problema desse trabalho é que a observação não é utilizada para atender aos requisitos da aplicação e sim para adaptar a aplicação às condições da infraestrutura, atuando de forma oposta aos objetivos desse trabalho. Até por isso, não há um gerenciamento racional dos recursos da infraestrutura.

Portanto, até onde foi realizado o levantamento do estado da arte por esse autor, nenhuma das arquiteturas propostas na literatura apresenta as características necessárias para servir de base a Espaços Inteligentes baseados em visão computacional.

3 Arquitetura de Microserviços centrada em Observabilidade

3.1 Modelos de Referência para o PIS

A arquitetura do PIS não é uma arquitetura genérica e possui como domínio específico Espaços Inteligentes baseados em Visão Computacional. Para tanto, ela tem como seus objetivos principais, atender aos requisitos específicos de suas aplicações, atender aos requisitos rigorosamente correlacionados e ao mesmo tempo fazer uso racional dos recursos de sua infraestrutura.

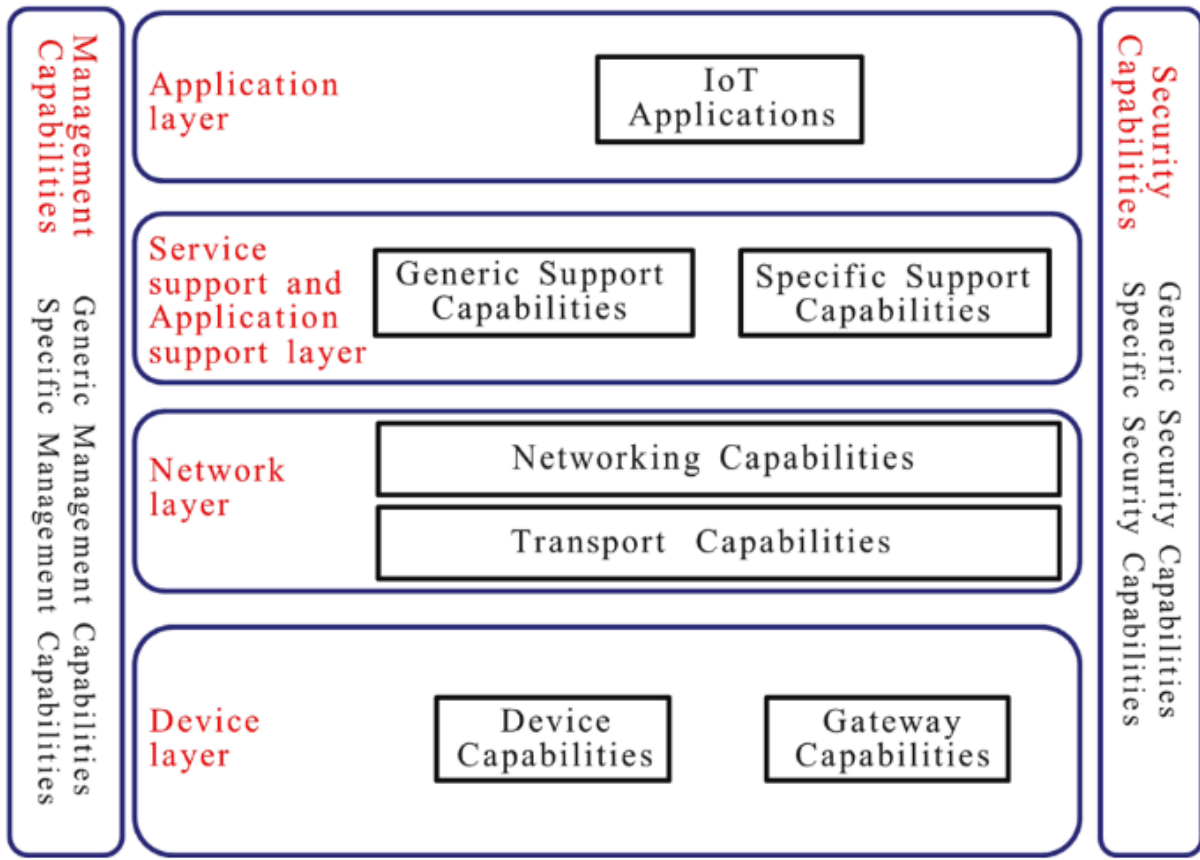
Além de atender aos objetivos principais, ou até mesmo como um caminho para atendê-los, a arquitetura do PIS deve buscar possuir também características como: facilidade de uso, interoperabilidade, disponibilidade, confiabilidade e escalabilidade.

Para tanto, a arquitetura do PIS foi projetada a partir de conceitos de diferentes modelos e arquiteturas que serviram de referência para a definição e de um conjunto inicial de seus blocos de construção.

O modelo de referência IoT ([ITU-T, 2012a](#)) do International Telecommunication Union (ITU), definido pelo seu setor de normatização das Telecomunicações (ITU-T, do inglês Telecommunication Standardization Sector), está representado na Figura 1 e serviu como primeira inspiração para arquitetura do PIS. O conceito de IoT se mostra altamente aderente ao de Espaços Inteligentes. Segundo o ITU, IoT pode ser definido como uma infraestrutura generalizada de objetos ou coisas, equipados com sensores, atuadores e elementos processadores, capazes de se comunicarem entre si e entre serviços pela Internet. Esses dispositivos adicionam à comunicação entre humanos, a comunicação entre humano e coisa e entre coisas ([CHEN et al., 2014](#)), permitindo o aparecimento de uma variedade de aplicações que poderão se beneficiar dos novos tipos de dados, serviços e operações disponíveis ([ZANELLA et al., 2014](#)).

Esse modelo de referência é composto por 4 camadas horizontais. A primeira é a camada de aplicação que contém todas as aplicações IoT. Em seguida, há a camada de serviços e suporte a aplicação que possui um conjunto de funções comuns e específicas de auxílio às aplicações. A terceira é a camada de rede, responsável pela comunicação entre as diferentes entidades do modelo. E por último a camada de dispositivos que é composta pelos diferentes elementos físicos e a descrição de sua interação com os demais elementos arquiteturais do modelo. Além dessas quatro camadas, o modelo prevê as camadas de gerenciamento e segurança. Essas camadas proveem funcionalidades associadas as 4 primeiras camadas horizontais do modelo.

Figura 1 – Modelo de referência do ITU-T Y.2060/Y.4000.



Fonte: (ITU-T, 2012a)

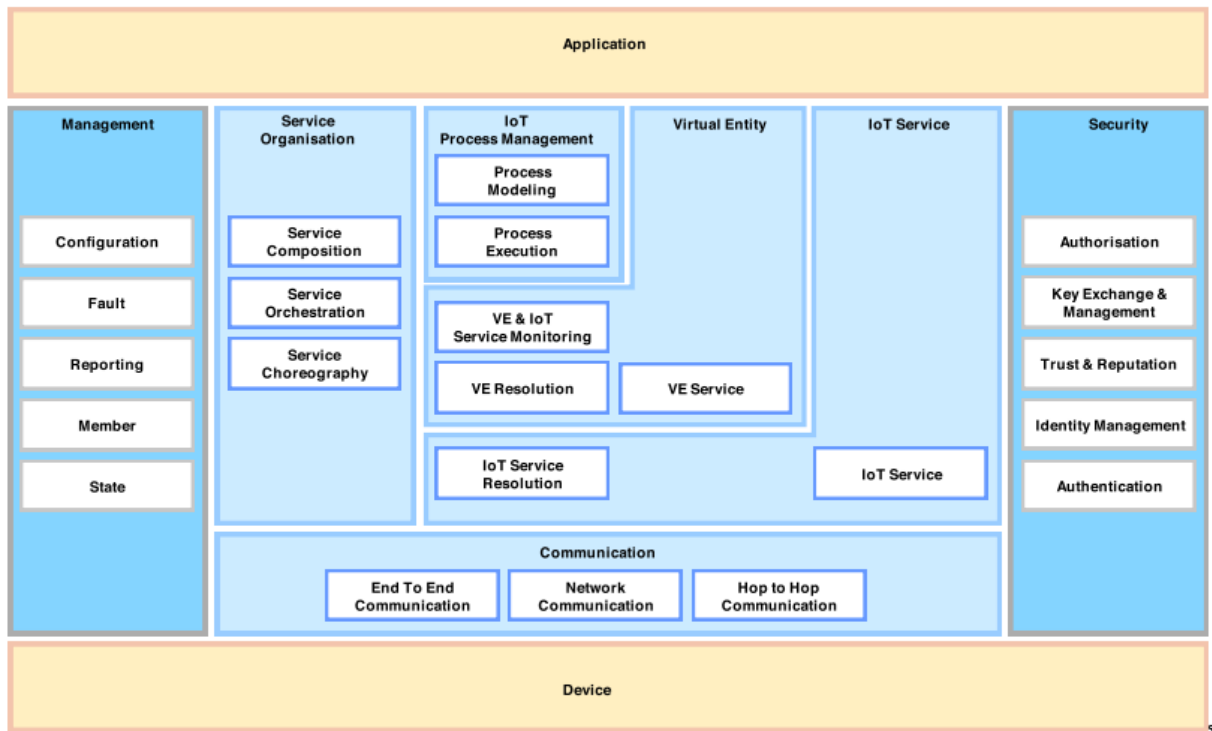
O modelo define diferentes funções para cada uma das camadas além de serviços ofertados entre elas. Dessa forma são criadas abstrações que facilitam a construção de arquiteturas específicas baseadas nesse modelo. O próprio ITU possui diversas recomendações de arquiteturas baseadas nesse modelo, tais como (ITU-T, 2012b) para o domínio de Web das coisas, (ITU-T, 2015) para E-Health e (ITU-T, 2018) transporte de segurança em gerenciamento de desastres.

Um segundo modelo usado como referência para o PIS é proposto no projeto IoT-A (BAUER et al., 2013). O projeto IoT-A propõe um modelo arquitetural de referência (ARM, do inglês Architectural Reference Model). Esse modelo é definido em um alto nível de abstração, fornecendo visões arquiteturais e perspectivas que são relevantes para a construção de diferentes arquiteturas para IoT. As visões do ARM fornecem descrições variadas que mostram a arquitetura sob diferentes ângulos e podem ser utilizadas durante as fases de projeto e implementação de uma arquitetura concreta. Exemplos de implementação podem ser vistos no projeto FIESTA-IoT (CARREZ et al., 2017) ou em (ELYAMANY; ALKHAIRI, 2015) que implementa uma arquitetura para ambientes acadêmicos.

Na visão funcional definida no ARM do IoT-A mostrada na Figura 2, são utilizados

nove grupos de funcionalidades (GF), a saber: aplicação, gerenciamento, organização de serviço, gerenciamento de processo de IoT, entidade virtual, serviço IoT, segurança, comunicação e dispositivo. Cada um desses GFs envolve um ou mais componentes funcionais (CF).

Figura 2 – Visão funcional da arquitetura ARM.



Fonte: (BAUER et al., 2013)

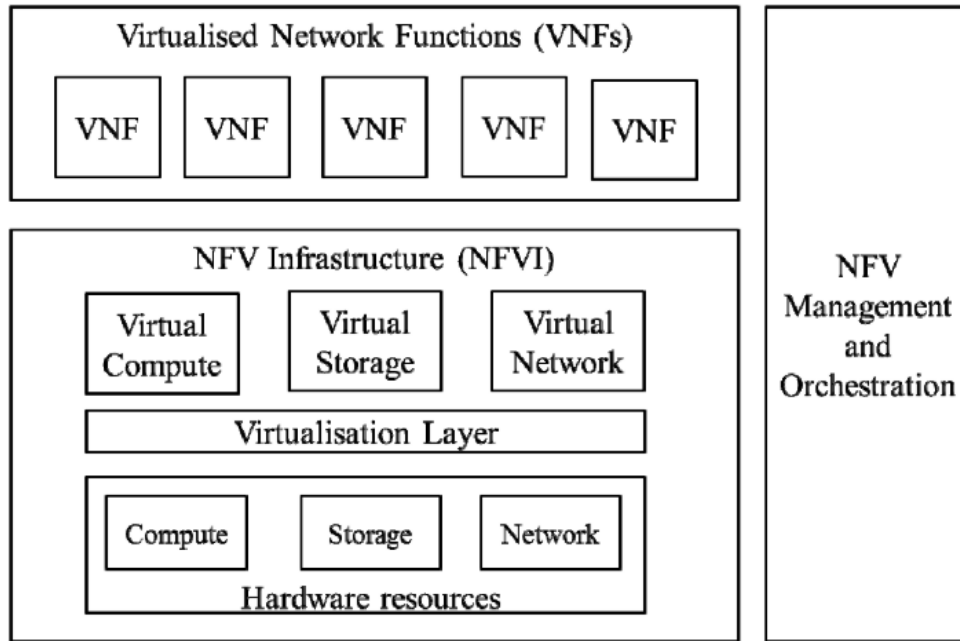
Todavia, apesar de a visão funcional descrever os CFs, ela não especifica as interações que ocorrem entre esses elementos pelo fato de tais interações serem tipicamente dependentes de escolhas de projetos, sendo portanto realizadas durante o desenvolvimento da arquitetura concreta. É importante observar que os GF de aplicação e de dispositivo estão fora do escopo da arquitetura de referência do IoT-A, enquanto que os GF de gerenciamento e de segurança são transversais aos demais GFs.

A utilização dos grupos funcionais de segurança e gerenciamento no ARM do projeto IoT-A, reforçam a necessidade dessas funções em todas as camadas de uma arquitetura. Uma outra inspiração desse modelo é a separação dos GFs de aplicação e dispositivo. Esses grupos interagem com o modelo mas não fazem parte dele. Por último, a utilização de entidades virtuais como contraparte digital de entidades físicas abstraem as suas especificidades facilitando a sua utilização.

Um terceiro modelo utilizado como base para o PIS é a arquitetura de referência de NFV do European Telecommunications Standards Institute (ETSI) (ETSI NFV ISG,

2014) mostrada na Figura 3. A virtualização de funções de rede tem por objetivo usar técnicas de virtualização para desacoplar as funções de rede de dispositivos de hardware dedicados. Dessa forma, as funções de rede passam a ser tratadas como serviços que podem ser alocados em diferentes equipamentos da infraestrutura, de acordo com as necessidades.

Figura 3 – Arquitetura NFV do ETSI.



Fonte: (ETSI NFV ISG, 2014)

A utilização de uma camada de virtualização como forma de abstrair os recursos físicos da infraestrutura é um ponto importante descrito na arquitetura de referência do ETSI. Outra decisão arquitetural importante é a inclusão das funções de orquestração e gerenciamento dos recursos da infraestrutura na camada de gerenciamento. Virtualização e orquestração são fatores importantes que proveem maior flexibilidade ao gerenciamento dos recursos. E por isso são características importantes adicionadas na arquitetura do PIS.

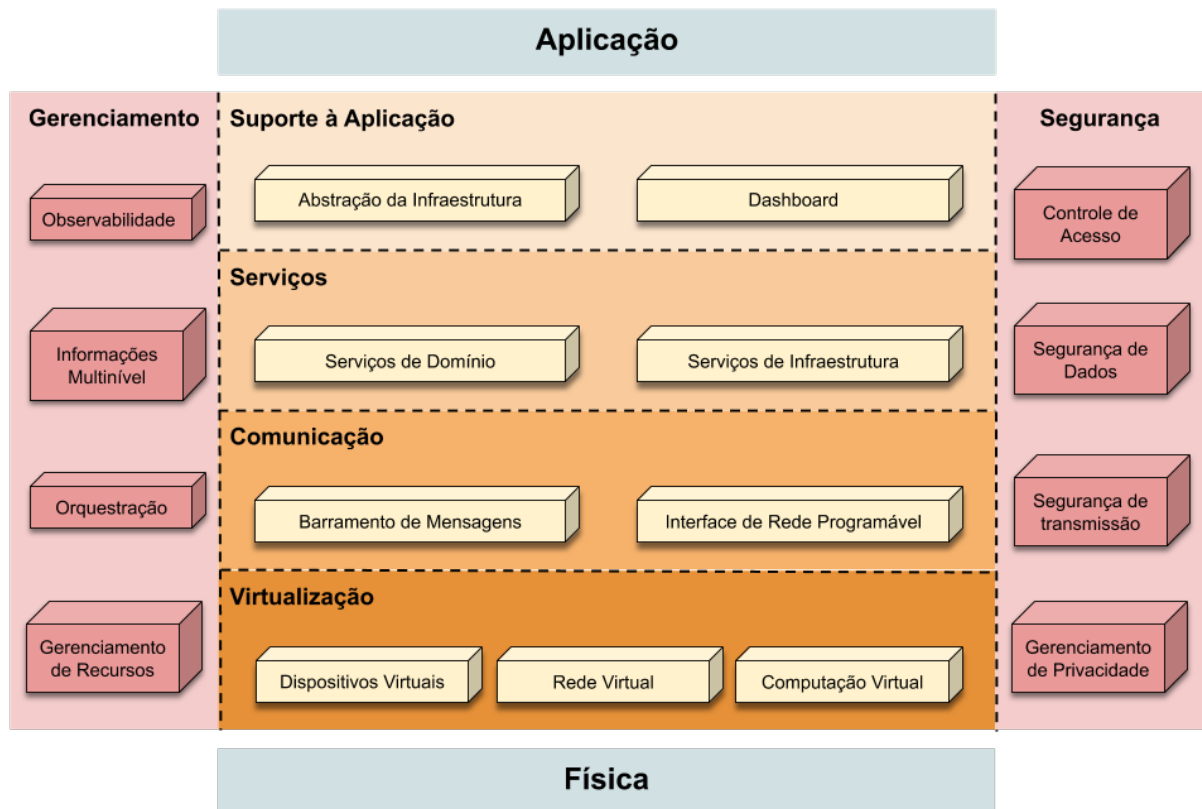
3.2 Arquitetura Proposta

Tomando como base os modelos de referência apresentados, a arquitetura proposta para o PIS é apresentada na Figura 4.

A arquitetura do PIS possui 4 camadas horizontais e duas camadas transversais, semelhante ao modelo do ITU. Porém as camadas se diferenciam do modelo de referência em alguns aspectos.

A camada de suporte a serviços e aplicação foi separada em duas camadas. A camada de serviços se mantém com as mesmas funcionalidades. Já a camada de suporte a aplicação serve basicamente como uma interface entre a arquitetura e as aplicações.

Figura 4 – Arquitetura do PIS.



Fonte: Produção do próprio autor.

No modelo de referência ARM, as camadas de aplicação e device interagem mas não fazem parte do modelo. Na arquitetura do PIS o mesmo ocorre com as camadas de aplicação e física. Porém, diferentemente do modelo de referência IoT, no PIS a camada física é composta não apenas por dispositivos IoT, mas também qualquer outro equipamento, tal como dispositivos de rede e servidores, por exemplo.

Para interação com esses equipamentos foi especificada a camada de virtualização, inspirada no modelo do ETSI. A virtualização pode ser feita através de uma relação de uma entidade virtual para cada entidade física ou pela virtualização de agrupamentos de dispositivos homogênicos ou mesmo heterogênicos.

Já as camadas de comunicação, segurança e gerenciamento seguem com funções semelhantes as suas contrapartes nos modelos de referência.

Toda a arquitetura foi projetada utilizando um modelo baseado em microsserviços, que se mostra altamente aderente as características do PIS com foco a atender aos objetivos principais desse trabalho.

3.2.1 Camada de Suporte a Aplicação

A camada de suporte a aplicação da arquitetura do PIS provê mecanismos que permitem a interação entre os desenvolvedores com o Espaço Inteligente. Isso inclui a interação e comunicação com os diferentes dispositivos e serviços que compõe o Espaço Inteligente. Quanto mais heterogêneo for a sua composição, mais complexo será para o desenvolvedor utilizá-los.

Cada um desses dispositivos pode possuir um padrão de acesso diferente. Por exemplo, fabricantes de câmeras podem utilizar diferentes protocolos para acesso as imagens que são implementados através de drivers diferentes. Essa variedade de protocolos de acesso pode estar presente inclusive para dispositivos de mesmo fabricante.

Da mesma forma, a mesma classe de dispositivos podem oferecer diferentes recursos. Um modelo de câmera pode fornecer o recurso de Region of Interest (ROI) por exemplo. Já um segundo modelo pode não possuir esse recurso.

Para ambos os casos, a arquitetura deve prover mecanismos de padronização de acesso aos recursos. No primeiro caso, pode ser realizada a conversão dos diferentes protocolos para um único a ser usado dentro do PIS. Já no segundo caso o função ROI pode ser implementada como um serviço externo à camera. Tanto os recursos disponibilizados diretamente pelo dispositivo quanto aqueles implementados externamente, estarão disponíveis através de uma entidade virtual que servirá de padrão de acesso ao dispositivo em questão.

A função de padronização e abstração da infraestrutura é realizada pela camada de suporte a aplicação. Essa camada serve como interface entre as aplicações dos desenvolvedores e a infraestrutura do PIS facilitando a sua interação. O objetivo dessa camada é prover o PIS da maior transparência possível.

Como consequência dessa transparência é gerada uma independência das ferramentas e implementações utilizadas no PIS. Pegue como exemplo a utilização de uma ferramenta de monitoramento do tempo de execução de um determinado serviço. Atualizações nessa ferramenta impactariam na implementação dos serviços que compõe as aplicações, caso não houvesse uma padronização via API da camada de suporte a aplicação. Além de uma API que expõe dispositivos e serviços para os desenvolvedores, a camada de suporte a aplicação provê também um dashboard como outra forma de interação entre a infraestrutura do espaço Inteligente e seus usuários.

Um PIS tem nas imagens das câmeras, sua principal fonte de dados para extração de informações. Por tanto, a visualização dessas imagens se mostra uma recurso relevante de ser fornecido pelo Dashboard. É importante ressaltar que é necessária, não só a visualização das imagens diretamente providas pelas câmeras, como também as imagens resultantes de diferentes etapas de uma aplicação.

Além das imagens, outras informações podem ser visualizadas no Dashboard. As

informações compartilhadas pela aplicação para atendimento dos seus requisitos ou parâmetros dos diferentes elementos que compõe a infraestrutura podem ser disponibilizadas pelo mesmo meio. Isso permite uma visualização ampla de todo o ambiente.

Dessa forma, as funções realizadas pelo Dashboard e pela API, se transformam em ferramentas importantes tanto para desenvolvedores de aplicação, quanto para os responsáveis pela infraestrutura possam usar para atingir aos objetivos do PIS.

3.2.2 Camada de Serviços

Como dito anteriormente, o PIS é projetado segundo um modelo baseado em microsserviços. Nesse caso, todas as funções da arquitetura do PIS, as aplicações que nele são executadas e até mesmo os seus dispositivos, são representados por serviços. Essa camada é composta por esse conjunto de serviços que podem ser divididos em duas classes: serviços de domínio e serviços de infraestrutura.

Os serviços de domínio são aqueles que são utilizados para compor as aplicações do Espaço Inteligente. No contexto desse trabalho, eles são serviços específicos do domínio de visão computacional e processamento de imagens. Fazem parte dessa classe, serviços de localização de objetos, rastreamento ou de conversão de referenciais, por exemplo.

Além dos serviços específicos de visão computacional, outros serviços da aplicação também fazem parte da mesma classe. Como exemplo podemos citar os serviços de planejamento de rota e controle de robôs.

Essa camada mantém um conjunto desses serviços que podem ser reutilizados por diferentes aplicações de forma concomitante e independente. Da mesma forma um determinado serviço pode ser substituído sem interferência aos demais que fazem parte da composição de uma aplicação.

A outra classe diz respeito aos serviços de infraestrutura. Fazem parte dessa classe os serviços de que implementam as funções das diferentes camadas do PIS. Os serviços de orquestração de recursos e de barramento de mensagens, que fazem parte das camadas de gerenciamento e comunicação respectivamente, são exemplos desse classe de serviços.

Mesmo os equipamentos físicos são representados por serviços através de processos de virtualização e abstrações via entidades virtuais. Para isso, a camada de serviços interage com os dispositivos do Espaço Inteligente usando as funcionalidades prevista na camada de virtualização. De forma similar, os serviços são expostos para os desenvolvedores através das funcionalidades da camada de suporte a aplicação.

Nem todos os serviços estão disponíveis para todos os usuários. Desenvolvedores podem ter acesso a apenas um subconjunto de serviços de domínio. Já com relação aos serviços de infraestrutura, apenas seus responsáveis devem ter acesso a eles. Exceção feita

a serviços que podem ser necessários para que a aplicação interaja com a infraestrutura, como por exemplo um serviço de sincronismo de câmeras ou serviços relativos a função de gerenciamento de informações multinível.

As características e funções dessa camada, e o fato dela ser projetada segundo um modelo baseado em microsserviços, permitem um melhor gerenciamento de toda a arquitetura. Isso contribui com o objetivo de se atender aos requisitos específicos das aplicações, os requisitos rigorosamente correlacionados da infraestrutura e ao mesmo tempo, realizar a alocação mais racional dos recursos.

3.2.3 Camada de Comunicação

A camada de comunicação é uma abstração que modela a variedade de esquemas de interação derivados das diferentes tecnologias de rede.

A abstração engloba toda a forma de comunicação presente em diferentes modelos, como por exemplo o modelo Open System Interconnection (OSI), desde as camadas inferiores como física e enlace, até as camadas superiores como apresentação e aplicação. Para isso essa camada cria uma interface padronizada de comunicação entre os serviços e dispositivos.

Endereçamento, encaminhamento e roteamento estão entre as funções básicas da camada de comunicação. Essas funções podem ser usadas por diferentes tecnologias e protocolos. A possibilidade de escolha do protocolo ou da tecnologia disponível mais adequada à comunicação entre duas ou mais entidades, é uma característica importante presente nessa camada.

Por exemplo, pode-se utilizar o endereçamento do protocolo IP (do inglês Internet Protocol) clássico para facilitar a integração entre diferentes dispositivos ou mesmo entre data centers. Já a utilização de redes definidas por resíduos (RDN, do inglês Residues Defined Networks) ([MARTINELLO et al., 2017](#)) pode ser mais adequada para situações onde a aplicação necessita de um melhor desempenho da rede para diminuição do tempo de resposta de uma determinada tarefa.

Existem diferentes modelos de comunicação que podem ser utilizados dentro da arquitetura do PIS. Por exemplo, em Espaços Inteligentes, os dados dos sensores são essencialmente do tipo push-based. O que se encaixa muito bem com um padrão de streaming. As soluções de streaming geralmente incluem um broker como intermediário central, responsável pela distribuição de mensagens para os clientes usando um modelo PUB/SUB.

Uma das grandes vantagens desse modelo é possibilitar que um sensor possa publicar seus dados para inúmeros clientes realizando apenas uma única conexão ao broker. O broker então, é responsável por receber a mensagem contendo esses dados e gerenciar as

conexões com os múltiplos clientes. Essa abordagem se vantajosa principalmente ao se levar em consideração que sensores comumente possuem restrições com relação a capacidade computacional e consumo de energia.

Outra vantagem da utilização de broker como elemento central de comunicação, é o desacoplamento temporal entre o produtor da mensagem e o cliente. O sensor pode enviar continuamente as mensagens sem que todos ou alguns dos clientes estejam prontos para recebê-las. Nesse caso, o broker armazena as mensagens e gerencia a entrega para cada cliente.

Um dos grandes problemas na utilização dos brokers como elemento intermediário é o sobrecarga na comunicação. O custo de se levar para as camadas mais altas a decisão de roteamento de cada mensagem pode inviabilizar o funcionamento de determinadas aplicações com requisitos mais restritivos. Esse problema é ainda maior ao se utilizar câmeras como principal sensor do ambiente.

Para esses casos, uma abordagem onde seja possível programar diretamente os dispositivos de rede para encaminhamento direto entre os sensores e os clientes pode se mostrar mais adequada. Redes SDN ou a programação direta de interfaces de rede inteligentes são exemplos dessa abordagem.

É justamente essa capacidade de programação da camada de comunicação que provê a arquitetura com a capacidade de atender aos requisitos específicos das aplicações e os requisitos rigorosamente correlacionados do Espaço Inteligente e ao mesmo tempo permitir a melhor utilização dos recursos possível. Isso é possível desde que sejam mantidas as interfaces de comunicação padronizadas.

3.2.4 Camada de Virtualização

A camada de virtualização tem como função abstrair os recursos de hardware para utilização pelas demais camadas da arquitetura. Do ponto de vista dos serviços que necessitam desses recursos a camada de virtualização e os recursos de hardware são a mesma entidade.

A camada de virtualização garante que o software necessário para acesso aos recursos da infraestrutura sejam dissociados dos recursos de hardware e, portanto, o software pode ser implantado em diferentes recursos físicos.

A estrutura arquitetônica do PIS não se restringe ao uso de qualquer implementação específica da camada de virtualização. Acessos diretamente ao hardware ou múltiplas camadas de virtualização podem ser usadas, desde que as interfaces da camada sejam respeitadas. Para garantir transparência operacional, o acesso aos recursos de hardware da infraestrutura deve ser independente de seu cenário de implantação.

Quando a virtualização é usada no domínio de recursos de rede, o hardware de rede é abstraído pela camada de virtualização para realizar caminhos de rede virtualizados que fornecem conectividade entre serviços. Várias técnicas permitem isso, incluindo camadas de abstração de rede que isolam recursos via redes virtuais e sobreposições de rede, incluindo Virtual Local Area Network (VLAN), Virtual Private Network (VPN), Virtual Extensible Local Area Network (VxLAN), etc. Switches e roteadores virtuais também podem ser utilizados.

Da mesma forma que a rede, recursos computacionais também são virtualizados por essa camada. Servidores e storages podem ser compartimentados de diferentes formas e utilizados por múltiplos serviços simultaneamente. Máquinas virtuais, containers ou mesmo o unikernel são exemplos de tecnologia utilizadas para esse fim.

Dentro da classe de recursos computacionais, dispositivos específicos também podem ser virtualizados. Como exemplo desse tipo de dispositivo podemos citar as GPUs. A virtualização desses dispositivos permitem a divisão do dispositivo em múltiplas GPUs de menor capacidade, permitindo a sua utilização simultânea por diferentes serviços.

Mesmo dispositivos como sensores e atuadores podem ser virtualizados. Nesse caso, além do compartilhamento dos dispositivos, a virtualização pode proporcionar a oferta de novos recursos que não estejam disponíveis no hardware. Além disso, a heterogeneidade desses dispositivos tornam ainda mais importante a sua padronização. Atender a essas necessidades faz parte das funções da camada de virtualização.

A inclusão de uma camada de virtualização na arquitetura do PIS possibilita a utilização de uma infraestrutura eficiente permitindo que usuários e aplicações possam utilizá-la simultaneamente. Permitindo inclusive a coexistência de diferentes versões dos mesmos serviços.

A possibilidade de manter ambientes de desenvolvimento, teste e produção na mesma infraestrutura permite um ciclo de desenvolvimento muito mais eficiente e integrado, reduzindo os custos de desenvolvimento e o tempo de disponibilização da aplicação.

Outra vantagem importante é a diminuição do consumo de energia utilizada pela infraestrutura. Isto é possível explorando os recursos de gerenciamento de energia em servidores e storages, bem como consolidação de carga de trabalho. Por exemplo, contando com técnicas de virtualização, seria possível concentrar a carga de trabalho em um número menor de servidores fora do horário de pico. Dessa forma, todos os outros servidores poderiam ser desligados ou colocados no modo de economia de energia.

Apesar das diversas vantagens, a incorporação da camada de virtualização na arquitetura do PIS traz também uma série de desafios que devem ser avaliados e soluções devem ser buscadas para o projeto da arquitetura do PIS. Dentre esses desafios devemos citar o compromisso com o desempenho e o gerenciamento de informações e recursos.

Ao se utilizar a camada de virtualização, é provável que uma redução do desempenho deva ser levada em consideração. O desafio é como manter a degradação de desempenho dentro dos limites aceitáveis para se cumprir os requisitos das aplicações.

A arquitetura deverá prover funcionalidades que possibilitem ao desenvolvedor da aplicação expressar claramente os seus requisitos. Além disso, o desempenho disponível da infraestrutura subjacente precisa ser claramente indicado, para que os dispositivos virtuais saibam o que podem obter do hardware.

De posse dessa informação a infraestrutura deverá definir a abordagem que minimize os efeitos na latência, taxa de transferência e sobrecarga de processamento, mas que ao mesmo tempo faça uma melhor compartilhamento e alocação dos recursos.

Tomemos como exemplo uma GPU que é compartilhada através de sua virtualização em N GPUs de menor capacidade. Essas GPUs virtualizadas podem atender até M serviços que necessitem desse tipo de recurso. Porém, essa virtualização pode implicar em uma diminuição de desempenho. Apesar de para maioria dos serviços essa diminuição seja aceitável, os requisitos de um dos serviços podem não ser atendidos. Nesse caso, uma segunda técnica de acesso direto ao hardware pode ser empregada, porém com o custo de não ser mais possível o compartilhamento dessa GPU.

Para tomada dessa decisão, a arquitetura deve prover mecanismos que através das informações disponíveis, possa tomar uma decisão de alocação mais racional dos recursos e ao mesmo tempo possa atender aos requisitos das aplicações do PIS.

As funcionalidades necessárias para isso estão na camada de gerenciamento da arquitetura do PIS e serão apresentadas a seguir.

3.2.5 Camada de Gerenciamento

Essa camada é responsável pelo gerenciamento dos elementos de todas as camadas da arquitetura do PIS. Ela possui 4 blocos funcionais principais que serão detalhados a seguir: Observabilidade, Informações Multinível, Orquestração e Gerenciamento de Recursos.

Observabilidade

Embora possa haver muitas definições para o termo observabilidade, geralmente se refere à capacidade de visualizar e compreender o comportamento de um sistema coletando, processando dados de um sistema em tempo de execução e apresentando-os de maneira adequada para os diferentes tipos de usuários, como desenvolvedores de aplicações, administrador de sistemas e sistemas de orquestração. Esse processo permite não apenas a análise do comportamento atual, mas também a inferência do comportamento futuro de um sistema.

A observabilidade de um sistema distribuído pode ser aprimorada empregando o que é comumente chamado de três pilares da observabilidade: logs, métricas e rastreamento.

Um log é um registro imutável de um evento discreto que ocorreu em algum momento. Por exemplo, um evento descrevendo que um servidor começou a processar uma solicitação. Os logs podem ser classificados em não estruturados (texto de forma livre sem um esquema específico) e estruturados (seguindo um esquema, binário ou texto específico). Eles também podem ser produzidos por diferentes fontes, como aplicações, sistemas operacionais, etc. Os logs geralmente são coletados por uma camada unificada de agregação de logs para serem pré-processados e depois persistidos em um armazenamento de dados para posterior processamento e análise.

Os logs podem transportar um grande volume de dados de todas as entidades do sistema e, portanto, podem ser usados para identificar quase tudo o que é considerado útil. Por esse motivo, eles são frequentemente usados para auditoria ou solução de problemas. No entanto, isso tem um custo no desempenho do próprio serviço. Quanto maior a quantidade de logs e mais detalhados, maior o impacto no desempenho do serviço sob alta demanda. Em outras palavras, a sobrecarga do log aumenta linearmente com o número de eventos.

Métricas são números coletados periodicamente, formando uma série temporal. As métricas são produzidas agregando eventos de uma entidade. Portanto, elas podem mostrar tendências sobre o comportamento de um sistema. Por exemplo, com as métricas, podemos ver o histograma das latências de uma Remote Procedure Call (RPC) para um serviço específico ou a quantidade de solicitações que o serviço processou até o momento. As métricas são comumente usadas para gerar alertas.

Alertas são ações acionadas por eventos quando uma condição predeterminada é atendida. Por exemplo, se um sistema está produzindo muitas respostas de erro no último minuto, podemos gerar um alerta em um painel ou até executar alguma ação predefinida para corrigi-lo. Para esse fim, os alertas permitem que situações que exijam intervenção imediata sejam monitoradas, automaticamente ou através de um aviso de intervenção manual.

Como os logs, as métricas são coletadas periodicamente para uma entidade central a ser processada e persistida. Porém, a coleta dessas métricas gera uma sobrecarga constante com a atividade do sistema, que depende apenas da quantidade de métricas.

Portanto, diferentemente dos logs, a sobrecarga só aumenta se o número de elementos monitorados for aumentado, por exemplo, com a inclusão de novos serviços ou novas métricas nos serviços existentes. Além disso, as métricas geram um volume muito menor de dados em comparação com os logs.

Métricas e logs fornecem informações sobre serviços individualmente, mas não são suficientes para aplicativos com vários serviços distribuídos por uma infraestrutura. Para

identificar o fluxo de informações de um aplicativo com serviços distribuídos e entender completamente seu comportamento, é necessário usar o rastreamento.

Os rastreamentos podem mostrar a causalidade dos eventos em um sistema e, portanto, ver como um evento interage com outro. A relação causa-efeito de um rastreamento é muito importante para ajudar a entender como um evento se propaga pelo sistema.

Para inferir a causalidade dos eventos em um rastreamento, cada serviço deve contribuir propagando um contexto de rastreamento junto com a carga útil usual. Esse contexto também é enviado diretamente a uma ferramenta que mais tarde será usada para correlacionar a cadeia de eventos. Esse último contexto também pode conter dados úteis para solução de problemas, como pilha de rastreios e outras anotações específicas da aplicação (SIGELMAN et al., 2010).

A sobrecarga do rastreamento é semelhante à dos logs, ou seja, o custo do rastreamento aumenta linearmente com o número de eventos sendo produzidos.

Os três pilares da observabilidade podem ser empregados em dois modelos distintos de monitoramento: caixa branca e caixa preta. No monitoramento de caixa branca, temos acesso ao estado interno do sistema por meio de informações fornecidas por ele. Em outras palavras, o modelo de caixa branca permite o uso de métricas, logs e rastreio fornecidos pelo aplicativo.

Por outro lado, no monitoramento de caixa preta, o sistema não fornece nenhuma informação sobre seu estado interno. O estado do aplicativo é definido pela observação da relação entre as entradas e saídas do elemento observado por meio de uma entidade externa, semelhante à experiência do usuário. Portanto, para monitorar esse tipo de sistema seria necessário, por exemplo, a geração de tráfego estruturado ou a coleta e análise dos dados de entrada e saída. Para dar uma ilustração, o tempo médio de resposta de um servidor pode ser deduzido por verificações regulares. Outra solução seria interceptar e analisar todo o tráfego que vai para o servidor.

Uma outra abordagem recorrentemente usada em aplicações em nuvem é o uso do padrão sidecar. Nesse padrão, um elemento sidecar é implementado junto com um aplicativo que compartilha o mesmo ciclo de vida. Este elemento fornece recursos extras para o aplicativo (BURNS; GOOGLE, 2016). Por exemplo, no contexto da observabilidade, um proxy de sidecar pode ser usado para instrumentar um serviço com funcionalidades de monitoramento no modelo caixa preta. Assim a função de monitoramento seria transparente para cada um dos serviços do ambiente (KLEIN, 2017).

As aplicações desenvolvidas para um ambiente em nuvem tal como o PIS, podem ser bastante complexas devido à interação entre seus vários componentes e a infraestrutura em que estão hospedados.

Como a infraestrutura do PIS pode hospedar um grande número de aplicações

compostas por vários serviços, a implementação da observabilidade em tal ambiente requer uma análise cuidadosa de seus vários elementos, para que não afete o desempenho geral do sistema.

Um sistema de monitoramento deve poder receber dados de todos os elementos participantes do ambiente em nuvem, tanto no nível da aplicação quanto da infraestrutura. Esses dados devem ser tão variados e tão completos quanto possível, a fim de permitir uma compreensão completa do ambiente e dos possíveis comportamentos, dependendo de sua interação. Portanto, nenhuma falha ou problema será negligenciado devido à falta de dados.

No entanto, a coleta de um grande volume de dados pode ser inviável devido a limites de rede ou armazenamento. Além disso, a coleta de uma enorme quantidade de dados sem limitações pode levar a problemas no desempenho do sistema, além de ter dados coletados desnecessariamente.

Assim, a coleta de um grande volume de dados pode ter o efeito oposto, ou seja, a observabilidade do sistema pode ser reduzida em vez de aumentada. Isso pode ocorrer devido ao atraso na perda de informações ou na coleta de dados. Além disso, informações relevantes podem ser ignoradas, dada a complexidade de analisar um volume maior de dados.

Além da redução da observabilidade, os problemas de desempenho mencionados também podem levar a um aumento de falsos positivos ou falsos negativos. Tanto um quanto outro, podem afetar a confiabilidade do sistema de monitoramento e atingir um nível no qual ele se torna inútil.

Para reduzir a sobrecarga de observabilidade nesses cenários, técnicas de amostragem adaptativa podem ser usadas. Serviços com alta taxa de transferência podem ter uma baixa taxa amostral sem grande perda de informações. Por outro lado, os serviços de baixa taxa de transferência não exigem amostragem e todas as solicitações podem ser monitoradas sem afetar o desempenho do sistema (SIGELMAN et al., 2010).

Dito isto, um sistema de monitoramento deve ser o mais simples e preciso possível. Seu desenvolvimento e operação devem ser projetados para facilitar sua manutenção e uso.

Portanto, durante o projeto, os objetivos do sistema de monitoramento devem ser muito claros para responder a perguntas como: O que será coletado? Como será coletado? E com que frequência será coletado?

Uma vez definido, cada usuário deve ter acesso às informações mais relevantes para eles. Onde os usuários podem variar de desenvolvedores e operadores a serviços da arquitetura como os orquestradores.

Compartilhamento de Informações Multinível

O PIS tem como um de seus objetivos, atender a requisitos específicos das aplicações. Isso é realizado através do controle dos recursos pela infraestrutura do Espaço Inteligente.

Na maior parte das arquiteturas de Espaços Inteligentes ou mesmo naquelas que podem ser adaptadas para esses espaços, os níveis de aplicação e de infraestrutura possuem baixa integração entre si. Isso ocorre devido a esses níveis possuírem visões muito diferentes. À aplicação importa o seu correto funcionamento, não sendo de seu interesse nem onde ou mesmo como ela será executada. Muitas vezes a infraestrutura é tratada como uma fonte inesgotável de recursos que estarão disponíveis no momento e na forma exata de suas necessidades. Ainda que essas necessidades não estejam claras para nenhum dos níveis.

Já a infraestrutura comumente é projetada com uma visão sobre si própria para suportar aplicações genéricas. Em alguns casos são gerados padrões gerais de comportamento de grupos de aplicações, porém para alguns domínios de aplicações isso não é o suficiente.

Para domínios, tal como o de Espaços Inteligentes baseados em Visão Computacional, existem requisitos específicos que precisam ser atendidos que a abordagem tradicional não suporta. Para esses casos é necessário uma maior integração que possibilite uma visão conjunta desses diferentes níveis. Esse é o papel do bloco funcional de Informações Multinível.

Muitas vezes existe uma relação clara, do ponto de vista da aplicação, entre os requisitos específicos da aplicação e os recursos da infraestrutura. Por exemplo, Um aumento do FPS impacta diretamente no requisito de precisão da trajetória percorrida por um robô. Essa é uma informação relevante que deve ser passada para a infraestrutura.

Além desse tipo de informação, esse bloco funcional permite não só que a aplicação possa compartilhar seus requisitos específicos e sua relação com os recursos da infraestrutura, mas também que eles sejam descritos de forma granular para cada um dos serviços que a compõem.

Por exemplo, para uma aplicação de interação entre pessoa e robô, um serviço de localização de pessoas pode precisar de um mínimo de 2 imagens de câmeras simultâneas para o seu correto funcionamento, já o serviço de controle do robô exige que sejam usadas câmeras posicionadas a uma altura mínima do chão. Nesse caso, a aplicação possui múltiplos requisitos que devem ser atendidos para o seu correto funcionamento, porém isso pode ser feito de forma independente, facilitando o gerenciamento dos recursos.

Adicionalmente a isso, também existem características da aplicação que somente ela conhece, mas que possuem impacto na alocação dos recursos. Utilização de GPU ou nível de paralelização dos serviços são exemplos dessas características. Portanto, para uma melhor alocação de recursos é preciso não só passar as informações de requisitos, mas

também suas características e correlações entre serviços e recursos da infraestrutura.

O bloco de Informações Multinível prevê a utilização de formatos de descrição padronizados, para que essa passagem de informações entre o nível de aplicação e infraestrutura ocorra. Esse formato deve ser genérico o suficiente para que seja possível passar expressar todas as informações que precisam ser compartilhadas, mas ao mesmo tempo, deve ser precisa o suficiente para que não haja dúvida em sua interpretação.

De posse dessas informações compartilhadas, a infraestrutura inicia o processo inicial de configuração e alocação de recursos. Posteriormente, essas informações também são utilizadas para determinar o que será observado de ambos os níveis, para dinamicamente, alterar os recursos da infraestrutura. Assim, a arquitetura do PIS pode atender tanto às aplicações quanto realizar o controle racional dos recursos da infraestrutura.

Orquestração

Segundo ([Saraiva de Sousa et al., 2019](#)), orquestração se refere à ideia de selecionar e controlar automaticamente vários recursos, serviços e sistemas para atender a certos objetivos. Para isso, os orquestradores devem realizar um conjunto coordenado de atividades que abstraem e automatizam o controle de recursos ofertados aos serviços que compõe as aplicações.

A orquestração no ambiente de nuvem foi primeiramente tratada em ([GALIS et al., 2009](#)) no contexto da Internet do Futuro (FI, do inglês Future Internet) e se refere à localização, coordenação e seleção de recursos, incluindo computação, armazenamento e redes virtuais para atender aos requisitos desejados. Portanto, para os autores, a função de orquestração coordena o comportamento integrado e as operações para adaptar e otimizar dinamicamente os recursos em resposta ao contexto em mudança de acordo com os objetivos e políticas do negócio.

No cenário SDN, orquestração se refere a uma função abrangente para gerenciar e automatizar o comportamento da rede ([VAUGHAN-NICHOLS, 2011](#)). De acordo com ([MAYER; MANSFIELD, 2015](#)), orquestração de serviço refere-se ao controle programático da infraestrutura subjacente, incluindo redes existentes e tecnologias habilitadoras, como SDN e NFV. Com base nessas definições, nesse trabalho o bloco funcional de orquestração será responsável por todo o ciclo de vida dos serviços que compõe as aplicações. Para isso será realizado o controle racional dos recursos da infraestrutura, de forma que os requisitos específicos das aplicações e os requisitos rigorosos e correlacionados na infraestrutura sejam atendidos.

O ciclo de vida dos serviços de uma aplicação passa por sua criação, execução até sua finalização. Para sua criação, o orquestrador utiliza as informações compartilhadas pela aplicação. Essas informações servem de base para a alocação inicial de recursos.

Durante a execução, para que o orquestrador verifique se os requisitos estão sendo atendidos, são utilizadas métricas para definir se a aplicação precisa ou não de mais recursos. Se as métricas satisfizerem alguma condição predeterminada, como atingir certos limites, o orquestrador poderá agir fornecendo mais ou menos recursos. Essas alterações ocorrerem de forma dinâmica durante todo o período de execução. Quando a aplicação finalizar, o orquestrador precisa então liberar os recursos alocados à aplicação.

Entre as ações mais comuns realizadas pelo orquestrador estão o escalamento, a migração de serviços e a mudança de caminho de comunicação. O escalamento pode ser do tipo vertical ou horizontal, que consiste em aumentar a quantidade de recursos e o número de réplicas de uma entidade, respectivamente. Já a migração ocorre quando um serviço é movido de um nó para outro. Ao realizar a ação de migração é necessário reconfigurar o encaminhamento da comunicação entre os serviços envolvidos. A mudança de caminho de comunicação ocorre quando há múltiplas opções de conexão entre dois serviços que se comunicam, mesmo que não haja uma migração entre serviços.

Porém, é necessário um conjunto mais ampliado e granular de ações do orquestrador, para que os requisitos exigidos no domínio de Espaços Inteligentes baseados em Visão Computacional sejam atendidos. Alterações no modelo de virtualização, mudança de parâmetros nas camadas mais baixas de rede ou mesmo mudanças nos próprios sensores e atuadores do ambiente são exemplos dessa granularidade necessária.

Todas as ações do orquestrador são realizadas através de instruções enviadas aos gerenciadores de cada tipo de recurso da infraestrutura. Logo quanto mais granular o conjunto de ações, mais complexo ou maior o número de gerenciadores de recurso da infraestrutura.

O orquestrador possui um domínio de atuação que determina o que ele observa e qual o alcance dessa atuação. Recursos computacionais, rede de comunicação e sensores e atuadores são exemplos de domínios de atuação. Ciente dos requisitos, ele vai atuar no seu domínio de forma a atender aos objetivos definidos. Porém, em um ambiente de requisitos rigorosamente correlacionados, eles só poderão ser atendidos com uma atuação conjunta entre diferentes domínios.

Podemos citar como exemplo o requisito de tempo de resposta de uma aplicação. Para sua diminuição, uma ação de migração de um serviço que estivesse na borda para a nuvem poderia ser realizada. Nesse caso, o tempo de processamento seria reduzido, porém o tempo de comunicação aumentaria podendo impactar negativamente o tempo de resposta.

Para casos como esse é necessário que o processo de orquestração tenha ciência dos diferentes níveis ou domínios de atuação. Com esse objetivo, o PIS prevê um modelo de orquestração multinível, onde o orquestrador deve levar em consideração as especificidades

de cada um desses níveis, principalmente num contexto de tecnologias heterogêneas, mas deve atuar de forma coordenada para atender ao objetivo comum.

Os níveis ou domínios de atuação a que se referem a orquestração multinível devem levar em consideração que o PIS é um ambiente compartilhado entre múltiplos usuários e aplicações. Logo, a orquestração por aplicação deve ser prevista. Nesse caso, cada aplicação poderá ter seu próprio orquestrador que é responsável pela sua própria alocação de recursos. Todos esses orquestradores de aplicação possuem uma autonomia limitada, sendo controlados por um orquestrador geral de mais alto nível.

Ainda que modelos hierárquicos e multi domínios sejam previstos e utilizados na literatura ([Saraiva de Sousa et al., 2019](#)), eles dizem respeito a uma infraestrutura hierárquica e heterogênea e uma divisão entre domínios administrativos. Mas a falta de compartilhamento das visões dos níveis de aplicação e infraestrutura dificultam o atendimento dos objetivos do PIS.

Gerenciamento de recursos

O bloco funcional de gerenciamento de recursos é responsável pela interação com os dispositivos físicos do ambiente através de suas contrapartes virtuais. Por isso, ele é altamente dependente das capacidades da infraestrutura de cada Espaço Inteligente.

Os gerenciadores de recursos são utilizados pelos orquestradores como forma de atuação no ambiente. Os orquestradores enviam comandos padronizados de mais alto nível, solicitando uma determinada alocação de recursos. Os gerenciadores recebem esses comandos e os traduzem para os elementos específicos de cada ambiente.

A hierarquização dos orquestradores também se reflete nos gerenciadores de recursos. Um orquestrador pode controlar um ou mais gerenciadores de recursos, que por sua vez também podem gerenciar vários elementos da físicos da infraestrutura.

Tomemos como exemplo dessa cadeia hierárquica, o OpenStack ([OPENSTACK FOUNDATION, 2020a](#)) amplamente utilizado em sistemas de computação em nuvem. Nesse sistema o Heat ([OPENSTACK FOUNDATION, 2020b](#)) pode ser utilizado como orquestrador e este pode utilizar o Nova ([OPENSTACK FOUNDATION, 2020d](#)) como gerenciador de recursos de computação e o Neutron ([OPENSTACK FOUNDATION, 2020c](#)) como gerenciador de rede.

Os gerenciadores devem prover, não só a capacidade de controlar os dispositivos como nós computacionais, rede e sensores e atuadores, mas também que esse controle atinga um maior granularidade possível. Essa é uma das características que auxiliará o PIS a melhor atingir aos seus objetivos.

3.2.6 Camada de Segurança

Segurança é um aspecto que deve ser considerado na concepção de uma nova arquitetura para espaços inteligentes. Funções como controle de acesso, segurança dos dados, segurança da comunicação e gerenciamento de privacidade devem ser projetadas visando todas as camadas da arquitetura.

Um requisito essencial para um Espaço Inteligente multiusuário é que ele tenha a capacidade de ocultar detalhes específicos de cada uma das aplicações. Isso garante a privacidade e a confidencialidade entre os usuários, preservando capacidades e recursos para demandante (FRANCESCON et al., 2017).

Apesar da importância do tópico, e estar previsto na arquitetura, a segurança não faz parte do escopo desse trabalho e seus detalhes não serão abordados.

3.3 Processo de orquestração

Aplicações desenvolvidas para nuvem geralmente são compostas de muitos componentes menores, formando uma arquitetura de sistema distribuído, conhecida como arquitetura orientada a microsserviços. Essa arquitetura tem muitas vantagens sobre uma arquitetura monolítica. Os serviços agora são pequenos, têm uma responsabilidade única e, portanto, são mais fáceis de entender. Além disso, os serviços não são mais fortemente acoplados e podem ser facilmente alterados, redimensionados e implantados independentemente do restante da aplicação. Portanto, os orquestradores podem escalar uma aplicação com mais eficiência escalando os microsserviços que estão sobrecarregados, em vez de precisar escalar a aplicação como um todo, mesmo quando um de seus componentes ou microsserviços não exige escalonamento. No entanto, um problema que se apresenta é que, ao decompor uma aplicação complexa em muitos microsserviços distribuídos, a capacidade de solucionar problemas facilmente é normalmente perdida (ESPOSITO; CASTIGLIONE; CHOO, 2016), (BALALAIE; HEYDARNOORI; JAMSHIDI, 2016), (OLIVEIRA et al., 2016).

Por exemplo, uma solicitação feita para um serviço de front-end pode acionar solicitações para muitos outros serviços de back-end. Nesse cenário, o tempo de resposta do serviço de front-end será fortemente associada ao tempo de resposta dos serviços de back-end. Portanto, para se ter uma maior visibilidade e realmente entender o comportamento do sistema, a interação entre esses serviços deve ser monitorada e analisada em vários níveis. Dessa forma, é possível identificar gargalos e atuar para melhoria do funcionamento das aplicações. Porém, sem ferramentas adequadas, pode ser inviável a realização dessa tarefa.

Consequentemente, para que uma arquitetura baseada em microsserviços seja

realisticamente usada em um ambiente de produção, é fundamental que desenvolvedores e operadores possam entender como se comportam essas complexas aplicações distribuídas (SIGELMAN et al., 2010). Com esse propósito a camada de gerenciamento da arquitetura do PIS provê funções que promovem a observabilidade de todo o ambiente.

Outro aspecto que pode afetar o comportamento de uma aplicação em um ambiente distribuído é a infraestrutura em que está hospedado. Uma das principais vantagens da computação em nuvem é a capacidade de provisionar infraestrutura sob demanda. A infraestrutura do PIS funciona da mesma forma, alocando mais recursos quando as aplicações precisam e liberando recursos quando estes não são mais necessários.

Nesse contexto, é possível perceber a interdependência entre infraestrutura e aplicação. Por um lado, a infraestrutura requer métricas das aplicações para a alocação racional de recursos. Por outro lado, as aplicações dependem fortemente da infraestrutura subjacente para atender a seus requisitos específicos. Portanto, a observabilidade de todo o ambiente pode ser mutuamente benéfica tanto para infraestrutura quanto para as aplicações. Melhorando não apenas a capacidade dos desenvolvedores de entender suas aplicações em um ambiente distribuído, mas também para ser usada para um melhor gerenciamento dos recursos da infraestrutura.

Infelizmente, a observabilidade geralmente é implementada apenas na camada de infraestrutura e, em geral, a observação da camada de infraestrutura pode não capturar aspectos importantes da camada de aplicação. Como consequência, a alocação de recursos é feita de forma ineficiente, geralmente resultando em um cenário de sub ou super provisionamento.

No PIS, o processo de orquestração realizado na camada de gerenciamento é responsável pela alocação racional dos recursos e ao mesmo tempo atender aos requisitos específicos e rigorosos das aplicações. Esse processo é baseado em uma orquestração multinível centrada na observabilidade e uma programabilidade granular de toda a infraestrutura.

O processo de orquestração é implementado através dos principais blocos funcionais da camada de gerenciamento. Esses blocos utilizam as demais camadas da arquitetura de forma a interagir com a camada de aplicação e com a camada física, como mostrado na Figura 5. Essas outras camadas não estão sendo mostradas para facilitar o entendimento das interações.

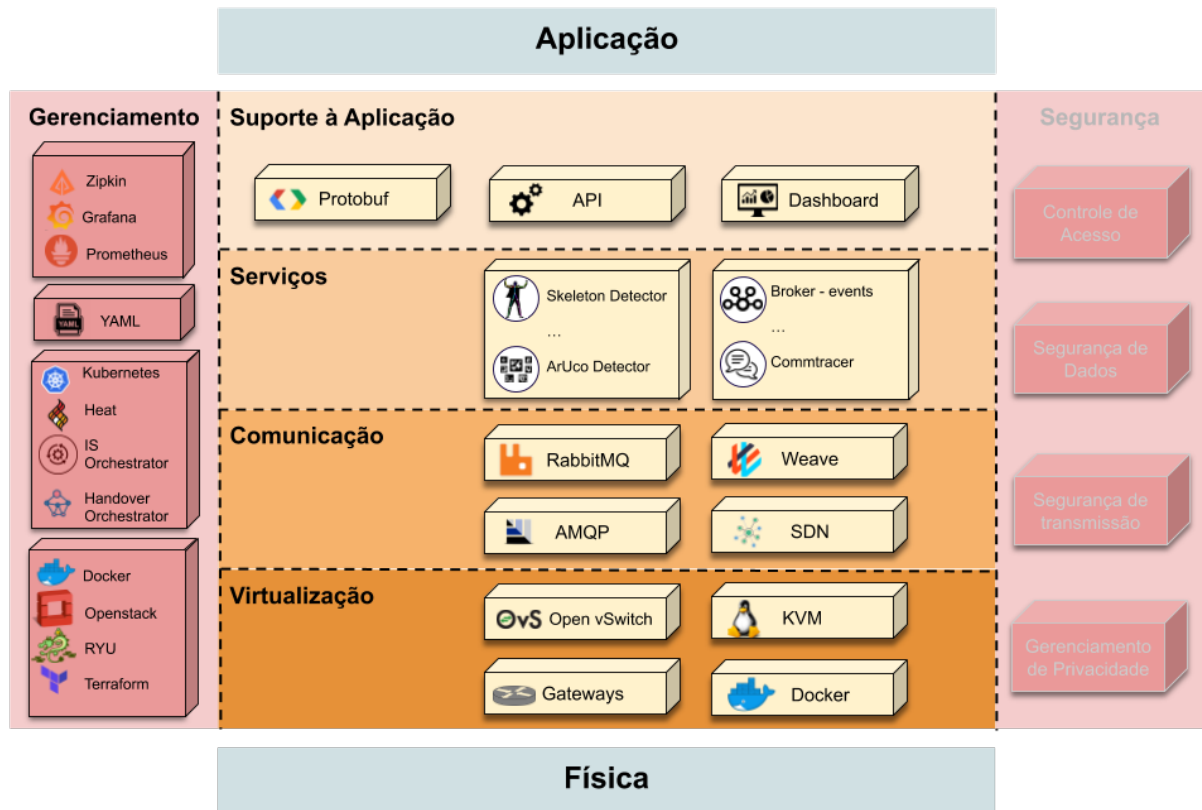
Para que uma aplicação seja inicializada, ela deve passar um conjunto de informações relevantes ao PIS, tais como seus requisitos, características e as relações entre os serviços e recursos. Essas informações são enviadas ao bloco funcional responsável pelo compartilhamento de informações multinível através de um descritor. Esse bloco é responsável por correlacionar os requisitos de diferentes níveis e configurar os parâmetros da função de orquestração através de uma interface padronizada.

Esse processo de orquestração, que incorpora os habilitadores da arquitetura, se apresenta como um elemento fundamental para que o PIS possa atingir os seus objetivos tão desafiadores: atender aos requisitos específicos e rigorosamente correlacionados das aplicações e ao mesmo tempo gerenciar racionalmente os recursos da infraestrutura.

4 Implementação

Nesse capítulo, serão apresentadas as implementações da arquitetura proposta. Uma representação do que foi implementado da arquitetura pode ser visto na Figura 6. Foram implantados até o momento, 3 espaços inteligentes em localidades diferentes. Cada um desses espaços possui sua própria infraestrutura de hardware e variações na implementação da arquitetura proposta. Além disso, serão descritas múltiplas aplicações foram implementadas para serem executadas nesses Espaços Inteligentes.

Figura 6 – Representação da implementação da Arquitetura do PIS.



Fonte: Produção do próprio autor.

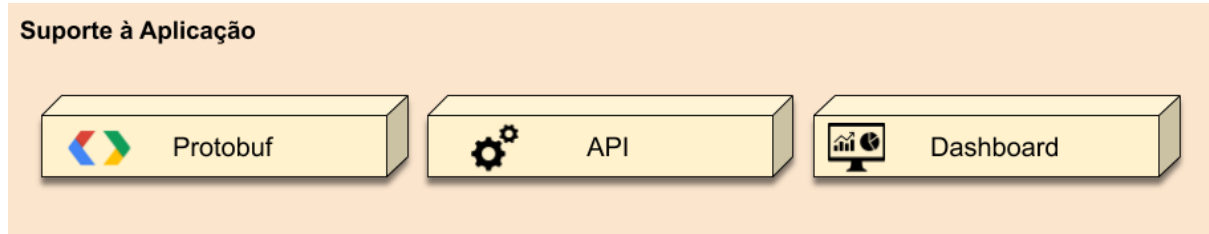
4.1 Arquitetura

4.1.1 Camada de suporte a aplicação

A camada de suporte a aplicação serve de interface entre as aplicações e o Espaço Inteligente. Ela deve prover mecanismos de padronização de acesso as entidades que compõe o PIS, promovendo transparência e facilidade de uso aos desenvolvedores. Para

isso, foram desenvolvidas uma API e Dashboards como interfaces de acesso, representadas na Figura 7.

Figura 7 – Representação da implementação da camada de suporte a aplicação do PIS.



Fonte: Produção do próprio autor.

API

A API é composta por um conjunto de bibliotecas que devem ser importadas nas aplicações e serviços que utilizem o PIS. As principais bibliotecas criadas são is-wire e is-msgs.

A biblioteca is-wire foi criada inicialmente com uma abstração da camada de comunicação do PIS. Posteriormente, foram adicionadas novas abstrações para acesso a diferentes ferramentas e funções utilizadas no Espaço Inteligente tais como rastreamento e geração de métricas. Alterações em versões ou mesmo a troca de ferramentas que impactem em sua interação com outros componentes da arquitetura são tratadas aqui. Dessa forma, as aplicações e serviços já desenvolvidos não precisam ser reescritos e mantêm-se a transparência no acesso ao PIS.

Foram desenvolvidas duas versões da biblioteca is-wire, uma em C++ e a outra versão em python. Essas linguagens foram escolhidas em função de serem as linguagens mais utilizadas para o domínio de aplicações para Espaços Inteligentes baseados em visão computacional. Caso necessário, versões em outras linguagens podem ser desenvolvidas.

A is-msgs é uma biblioteca de mensagens utilizada para comunicação no PIS. A biblioteca é baseada no protobuf desenvolvido pelo Google. O protobuf é um mecanismo extensível de linguagem neutra para serialização de dados estruturados. Segundo (GOOGLE LLC, 2020), tem semelhanças com o XML, porém é menor, mais simples e mais rápido. Com ele, é possível definir a estrutura dos seus dados nas mensagens e gerar código fonte em diferentes linguagens, que posteriormente podem ser incorporado nos serviços.

Na construção da biblioteca is-msgs foram definidas um grande conjunto de mensagens que são comumente utilizadas por diferentes serviços do PIS. Ainda que o conjunto de mensagens seja bastante abrangente e que sua reutilização seja incentivada, novas mensagens podem ser criadas de forma a estender a biblioteca.

O processo de criação passa pela definição do formato da mensagem, a especificação dos tipos dos campos, definição dos identificadores e regras de uso de cada um dos campos. Várias mensagens podem ser agrupadas e posterior geração de código que permita a sua utilização. Um exemplo da definição de mensagens pode ser visto na Figura 8

Figura 8 – Mensagem protobuf.

```
1 message ImageSettings {
2   // Image resolution (height, width).
3   Resolution resolution = 1;
4
5   // Image compression details. e.g: PNG.
6   ImageFormat format = 2;
7
8   // Color space.
9   ColorSpace color_space = 3;
10
11  /* Bounding poly defining the region of interest in the image.
12   This region is usually represented as a rectangle modelled by
13   the TopLeft and BottomRight vertices. */
14  BoundingPoly region = 4;
15 }
16
17 // Models the resolution of an image, that is the number of pixels
18 message Resolution {
19   // Number of vertical pixels.
20   uint32 height = 1;
21   // Number of horizontal pixels.
22   uint32 width = 2;
23 }
```

Fonte: Produção do próprio autor.

Dashboard

Dashboard é uma interface gráfica de visualização de dados e parâmetros de ferramentas ou sistemas, que tem por objetivo facilitar a sua utilização pelos usuários. O PIS é composto por diferentes ferramentas de terceiros integradas e adaptadas com outras desenvolvidas especificamente para esse trabalho. Muitas das ferramentas possuem um dashboard próprio que são utilizadas para para seu gerenciamento. Alguns desses dashboards serão apresentados durante a descrição da implementação das demais camadas da arquitetura.

Além disso, dashboards específicos foram desenvolvidos para servirem para visualização e controle determinadas aplicações. Na figura 9 pode-se ver um exemplo de um dos dashboards desenvolvido especificamente para a aplicação de controle de robôs descrita no capítulo 5.

Nesse dashboard é possível, por exemplo, configurar parâmetros da rede sem fio, velocidade e trajetória do robô. Também é possível visualizar métricas como o erro de trajetória do robô ou a célula sem fio ativa em cada instante. Esse tipo de interface busca facilitar a execução de uma determinada aplicação e a análise de seus resultados.

Dashboards específicos fornecem um amplo controle de uma determinada aplicação. Porém, essa abordagem pode ser inviável em um ambiente com múltiplas aplicações. Além

Figura 9 – Dashboard para o experimento do estudo de caso 1.



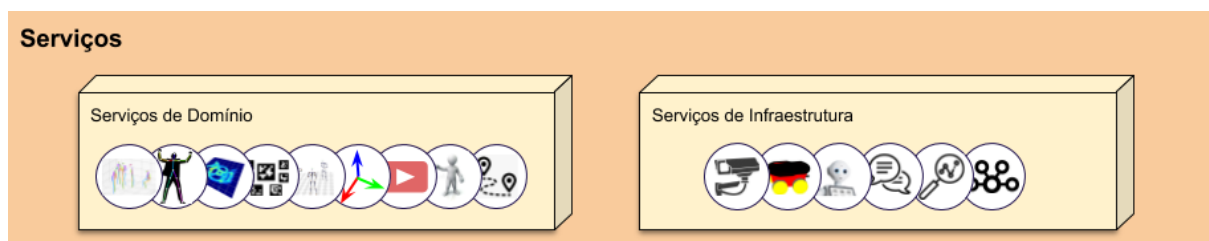
Fonte: Produção do próprio autor.

disso, é necessário o gerenciamento do próprio PIS de forma unificada. Como forma de facilitar a utilização do PIS por desenvolvedores e administradores, uma interface gráfica geral de todo o PIS se encontra atualmente em desenvolvimento.

4.1.2 Camada de serviços

Todas as funções da arquitetura do PIS são implementadas como microsserviços. Como pode ser visto na Figura 10, existem duas classes principais de serviços: serviços de domínio e serviços de infraestrutura.

Figura 10 – Representação da implementação da camada de serviços do PIS.



Fonte: Produção do próprio autor.

Aplicações no espaço inteligente são composições de serviços de domínio. Durante o projeto dessas aplicações, suas funções são separadas em serviços independentes que se

comunicam entre si através de interfaces bem estabelecidas.

Como exemplo, a aplicação de controle de trajetória de robôs utilizada no Capítulo 5 utiliza o detector do marcador visual aruco e o controlador do robô como serviços de domínio, além de outros serviços de infraestrutura para seu funcionamento.

Foram desenvolvidos um grande conjunto de serviços de domínio para o Espaço Inteligente. A lista dos principais serviços e uma breve descrição, pode ser vista na Tabela 2. Nela também está indicado se o serviço todo desenvolvido localmente (L), se houve desenvolvimento para adaptação de um serviço existente (A) ou é um serviço de terceiro (T).

Tabela 2 – Serviços de domínio.

Serviço	Desen.	Descrição
is-face-detector	A	Detector de faces
is-aruco-detector	A	Deteção do marcador visual aruco
is-skeletons-detector	L	Detector de esqueleto 2D em imagens
is-skeletons-grouper	L	Localizador 3D de esqueletos
is-skeletons-heatmap	L	Mapa de ocupação
is-skeletons-3d	A	Visualizador 3D dos esqueletos
is-frame-transformation	L	Converte coordenadas entre referenciais
is-robot-controller	L	Controlador do robô
is-tracker	A	Rastreador de objetos
is-gesture-recognizer	L	Reconhecedor e classificador de gestos
is-pedestrian	L	Detector de Pedestres
is-aruco-calib	L	Calibração das câmeras usando aruco
speech-recognition-web-service	A	Reconhecimento de voz
path-planning-service	L	Planejamento de trajetória
position-interpreter-service	L	Converte uma palavra em uma posição no ambiente
3d-reconstruction	L	Reconstrução 3D

Fonte: Produção do próprio autor

Em muitos dos serviços listados, diferentes versões foram desenvolvidas. Seja para melhoria do serviço, para utilização de algoritmos diferentes ou generalização do serviço. Em outros casos, como o Robot Controller, dois serviços foram fundidos após a verificação de que o tempo de comunicação entre eles era muito maior do que os tempos de processamento somados.

Um esforço foi feito no sentido de desenvolver os serviços de forma que pudessem ser reaproveitados na composição de diferentes aplicações. Um serviço como o Skeleton Detector é utilizado tanto para uma aplicação de mapa de ocupação quanto para um reconhecedor de gestos, por exemplo. Além disso, os mesmos serviços puderam ser utilizados em múltiplos espaços com dispositivos diferentes, sem que tenha sido necessário adaptá-los.

A outra classe de serviços dessa camada diz respeito aos serviços de infraestrutura. Nessa classe estão incluídas ferramentas já existentes que foram adaptadas ou configuradas para que exercessem as funções previstas na arquitetura do PIS. Além disso, outras ferramentas de infraestrutura foram desenvolvidas para suprir as carências existentes nas ferramentas atuais. Uma lista dos principais serviços de infraestrutura utilizados no PIS estão listados na Tabela 3.

Tabela 3 – Serviços de infraestrutura.

Serviço	Desen.	Descrição
openstack*	T	Infraestrutura de nuvem
kubernetes*	T	Orquestrador de Containers
gpustake-scheduler-extender	T	Virtualização de GPU
weave	T	Virtualização de rede
is-mjpeg-server	L	Servidor para visualização de imagens
is-broker-events	L	Publicação de eventos do broker
rabbitmq	T	Broker
zipkin	T	Rastreamento distribuído
is-camera-gateway	L	Gateway das câmeras
is-robots-gateways	L	Gateway dos robôs
is-orchestrator	L	Orquestrador de primeiro nível
handover-orchestrator	L	Orquestrador de rede
is-commtracer*	L	Medição de tempo de comunicação
prometheus*	T	Servidor de métricas
grafana	T	Visualizador de dados
ryu	T	Controlador SDN
is-sync	L	Sincronismo de dispositivos
is-service-discovery	L	Descoberta e apresentação de serviços

* Correspondem a um agrupamento de serviços

Fonte: Produção do próprio autor

Em nosso Espaço Inteligente, usamos Docker Containers para gerenciar nossos serviços. Docker é uma plataforma para construir, executar e compartilhar aplicativos com contêineres. Um contêiner nada mais é do que um processo em execução, com alguns recursos de encapsulamento adicionais aplicado a ele para mantê-lo isolado do host e de outros containers. Uma imagem de um container, inclui tudo o que é necessário para executar um serviço, como código, dependências ou quaisquer outros objetos de sistema de arquivos necessários (DOCKER, 2020).

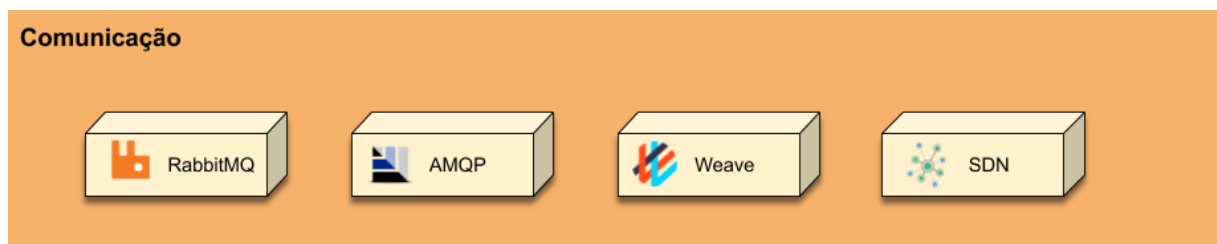
Dessa forma, todas as entidades do PIS são implementados através de microsserviços. E todos os microsserviços são encapsulados em Docker Containers. Portanto, o gerenciamento do PIS passa necessariamente pelo gerenciamento desses containers. Ainda que o PIS não se restrinja a containers como única forma de encapsulamento de serviços,

Docker é a forma mais utilizada e o seu gerenciamento está disponível na camada de gerenciamento e será descrito em mais detalhes posteriormente.

4.1.3 Camada de comunicação

A camada de comunicação é responsável por fornecer conectividade e interação entre as entidades no Espaço Inteligente, e os componentes utilizados em sua implementação podem ser vistos na Figura 11.

Figura 11 – Representação da implementação da camada de comunicação do PIS.



Fonte: Produção do próprio autor.

No PIS, diferentes protocolos podem ser utilizados para comunicação, como por exemplo o HTTP. O HTTP é um protocolo amplamente difundido e facilmente implementável nos serviços. Ele utiliza o padrão request/response onde um servidor aguarda uma solicitação de qualquer cliente, que em seguida enviará uma resposta de retorno. Porém o PIS, tem mensagens como base para comunicação entre serviços, e portanto, utiliza um protocolo desse tipo como padrão de comunicação.

O protocolo Advanced Message Queuing Protocol (AMQP) foi escolhido para o padrão de comunicação. Esse é um protocolo de mensagens assíncrono, ou seja, depois que uma mensagem é enviada, não se espera uma resposta imediata. Uma vantagem de se utilizar um protocolo assíncrono é que o destinatário da mensagem não precisa estar disponível no instante de envio da mensagem.

Para utilizar esse protocolo, deve-se escolher também um broker. O broker é um elemento intermediário na comunicação que tem a função de receber e encaminhar mensagens. Essas mensagens podem ser modificadas antes de serem encaminhadas e, através do broker, é possível monitorar o fluxo de mensagens. Uma vantagem de utilizar uma infraestrutura com um broker é que as aplicações só precisam saber o seu endereço para se comunicar com outras aplicações. Além disso, ele oferece a possibilidade de persistir mensagens, ou seja, se o destinatário não estiver disponível, a mensagem fica armazenada até que este fique disponível.

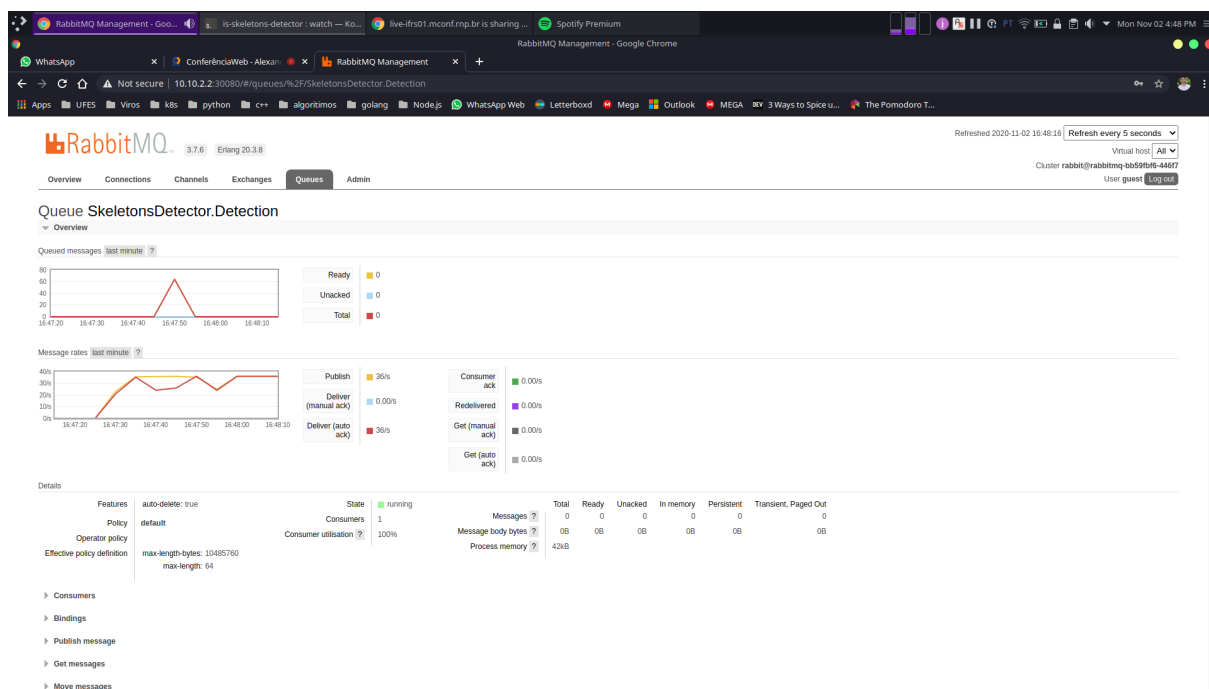
Uma desvantagem de se ter um elemento centralizador é a convergência do fluxo de mensagens para um único ponto. Contudo, é possível possuir múltiplas instâncias de

brokers em pontos diferentes da rede, o que ajuda a minimizar esse problema.

Um dos motivos para a escolha do AMQP foi a sua interoperabilidade. Ele possui implementações em diversas linguagens como Java, Python, PHP, JavaScript, C/C++, entre outras. Além disso, possui toda sua especificação aberta, o que facilita a solução de eventuais problemas na fase de desenvolvimento. Para utilizar o AMQP, deve-se escolher uma de suas inúmeras implementações. Usualmente utiliza-se implementações do protocolo associadas a um broker.

Para o AMQP, existem várias possibilidades de broker: Apache ActiveMQ, FFMQ, HornetQ, RabbitMQ entre outros. Basicamente, todos eles possuem as características que se espera de um broker, como escalabilidade e robustez. Para essa implementação foi escolhido o RabbitMQ. Ele é um broker amplamente utilizado em grandes projetos (RABBITMQ, 2020). Possui um dashboard que facilitam o seu monitoramento e configuração, como mostrado na Figura 12. Além disso, ele possui implementações do cliente AMQP em diversas linguagens. Além disso, o RabbitMQ possui extensões para operar com outros protocolos como o Message Queuing Telemetry Transport (MQTT) e HTTP.

Figura 12 – Dashboard do RabbitMQ.



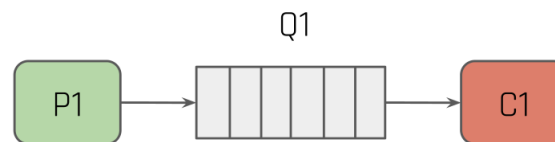
Fonte: Produção do próprio autor.

Funcionamento interno do RabbitMQ

A função do broker é receber e encaminhar mensagens. Para isso, o RabbitMQ implementa filas onde essas mensagens ficam armazenadas para que possam posteriormente

ser encaminhadas ao destinatário. A Figura 13, mostra um produtor, P1, uma fila, Q1 e um consumidor, C1. Essa fila pode ser configurada para ser persistente e ter tamanho fixo, por exemplo. Se uma mensagem for entregue à fila e ninguém consumi-la, ela ficará lá até que o próximo consumidor se conecte e consuma essa mensagem.

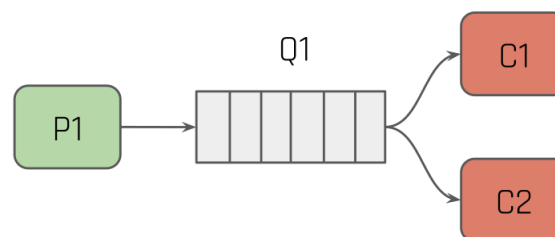
Figura 13 – Comunicação entre um produtor e um consumidor.



Fonte: (QUEIROZ, 2016)

Uma mesma fila pode ser utilizada por mais de um consumidor, como mostra a Figura 14. Nesse caso, quando uma mensagem chegar à fila, ela será entregue a um dos consumidores com o critério de alternância baseado em round-robin, onde cada consumidor recebe uma mensagem por vez em sequência e de forma cíclica. Esse modelo é muito utilizado para serviços com múltiplas instâncias, onde várias requisições são enviadas, e então o broker distribui as tarefas para cada consumidor disponível. Caso a taxa de entrada de mensagens seja maior do que a taxa de consumo em qualquer instante, as mensagens ficarão aguardando na fila.

Figura 14 – Comunicação entre um produtor e dois consumidores.

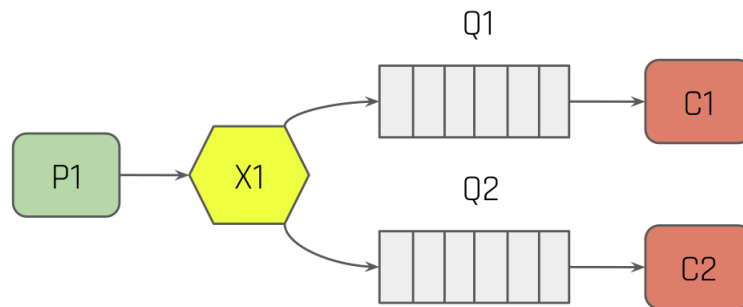


Fonte: (QUEIROZ, 2016)

Em uma outra situação ocorre quando dois consumidores precisam receber o mesmo dado do produtor de forma independente, como representado na Figura 17. A utilização simultânea das imagens dos gateways das câmeras pelos serviços Skeleton Detector e Aruco Detector, é um exemplo desse cenário.

Para que isso funcione, o broker precisa de algo a mais do que filas. Surgem então os exchanges, que são posicionados antes das filas, e tem a função de encaminhar as mesmas mensagens para uma ou várias filas. A Figura 15 ilustra um exemplo do produtor P1 enviando uma mensagem que chega no exchange X1, que, por sua vez, decide para qual fila irá enviá-la, podendo assim, a mensagem chegar para C1, para C2, ou para ambos.

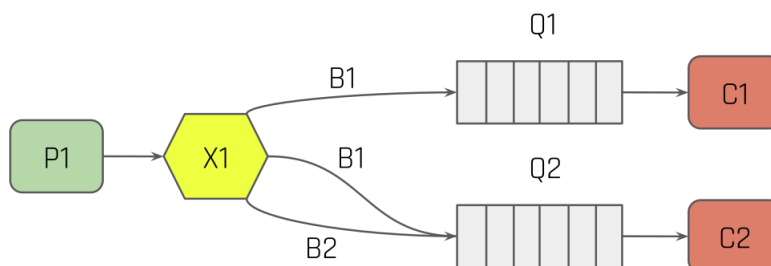
Figura 15 – Exchange entre produtor e duas filas de consumidores.



Fonte: (QUEIROZ, 2016)

A decisão de encaminhamento depende, não só dos exchanges, mas também dos bindings. Os bindings podem ser entendidos como regras de roteamento. Uma mesma fila pode ter mais de uma regra, ou ainda, diferentes filas podem possuir a mesma regra de roteamento, fazendo com que a mensagem seja replicada.

Portanto, uma mensagem quando enviada deve possuir um exchange de destino, bem como sua identificação de roteamento, o `routing_key`. Assim, o broker saberá para onde enviá-la. Por exemplo, na Figura 16, a fila Q2 possui dois bindings, B1 e B2 com o exchange X1, bem como a fila Q1 possui um binding com X1. Dessa maneira, uma mensagem enviada ao broker, com o exchange X1 de destino e o `routing_key` B1, será enviada para as filas Q1 e Q2, e portanto, chegarão aos consumidores C1 e C2. Já uma segunda mensagem com o exchange X1, mas com o `routing_key` B2, será encaminhada apenas para C2.

Figura 16 – Relação entre exchange e `routing_key`.

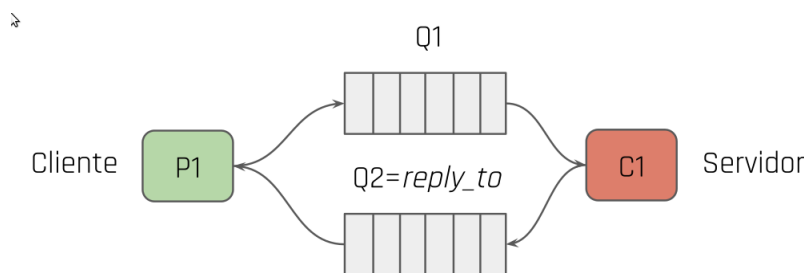
Fonte: (QUEIROZ, 2016)

Existem vários tipos de exchanges. Até então, nos exemplos mencionados, eles eram do tipo `direct`, ou seja, a mensagem era encaminhada para uma fila apenas de acordo com o binding. Entretanto, se for necessário mais critérios para encaminhar as mensagens, um exchange do tipo `topic` deverá ser utilizado, por exemplo. As mensagens entregues para um exchange desse tipo devem possuir `routing_keys` construídos por uma lista de palavras separadas por pontos, por exemplo “operacoes.soma” e “operacoes.subtracao”.

Normalmente, utiliza-se essa lógica para separar filas por tipos e subtipos. Além disso, pode-se utilizar os caracteres especiais “” e “#” para realizar um grupo de bindings, por exemplo se fizermos o binding: “operacoes.#”, as mensagens com “operacoes.soma” e “operacoes.subtracao” serão enviadas para a mesma fila.

Além das configurações acima apresentadas, pode ser implementado um modelo baseado em RPC, que é útil para a criação de serviços, onde envia-se uma mensagem com algum conteúdo que será processado, ou utilizado para alterar algum parâmetro da entidade, e uma resposta será enviada de volta para o remetente da mensagem. A Figura 17 mostra uma representação desse modelo, no qual foi omitido o exchange para simplificar a explicação. Nesse exemplo, o cliente C1 deve enviar o seu pedido para a fila que o servidor está escutando, Q1. Além disso, deve enviar o nome da fila para a qual a resposta será enviada, Q2. Essa fila usualmente não possui um nome conhecido, ficando à cargo do broker gerar um nome único para ela. O cliente fica então escutando essa fila, e quando a mensagem de resposta chega, é feita a correlação com a identificação colocada na mensagem de pedido.

Figura 17 – Implementação de RPC no broker.



Fonte: (QUEIROZ, 2016)

Como o protocolo de mensagens utilizado é assíncrono, se forem feitos vários pedidos pela mesma fonte, as respostas podem não chegar na ordem que os pedidos foram feitos. Para isso, junto da mensagem é enviado um ID (`correlation_ID`), para que seja feita a correlação com a resposta recebida. Tanto o `correlation_ID` quanto a fila na qual será recebida a resposta (`reply_to`) são inseridos em campos do cabeçalho do protocolo AMQP.

Outros modos de comunicação

A comunicação no PIS é baseada em um barramento de mensagens implementado por um broker. Porém, outras formas de comunicação são utilizadas e gerenciadas pelo PIS para promover a comunicação entre as entidades e prover a granularidade necessária.

Uma dessas formas é a utilização de redes definidas por software. O SDN permite o gerenciamento unificado de dispositivos de rede distribuídos pela sua infraestrutura. Esses dispositivos de rede podem ser tanto físico quanto virtuais.

No PIS, o SDN é utilizado para encaminhamento de fluxo de dados baseado em regras regidas centralizadamente por um controlador SDN. O protocolo utilizado entre o controlador SDN e os dispositivos é o Openflow.

Um exemplo de utilização dessa tecnologia no PIS pode ser observada no Estudo de caso apresentado no Capítulo 5. Nesse caso, um switch altera o fluxo de dados, escolhendo qual ponto de acesso sem fio será utilizado pelo robô móvel. As regras para encaminhamento são baseadas em função da localização do robô dentro do ambiente.

Enquanto a comunicação entre o serviço localizado no robô e os demais serviços na infraestrutura é feita via broker, pode-se observar que o controle SDN atua de forma independente e complementar. Ambos os modos impactam no desempenho da comunicação. Por isso, ter o controle de diferentes modos de comunicação permitem o controle mais granular e consequentemente facilitam atingir os objetivos do PIS.

Um outro modo de comunicação do PIS é implementado pelo Weave. O Weave cria uma rede virtual sobreposta que conecta contêineres Docker em vários hosts e permite sua descoberta automática (WEAVERWORKS, 2020). Com o Weave, as aplicações baseadas em microsserviços que consistem em vários contêineres podem ser executados em hosts em diferentes localidades. Os serviços usam a rede como se os containers estivessem todos plugados no mesmo switch de rede, sem ter que configurar mapeamentos de portas ou links, mas também podem ser expostos externamente.

O Weave cria uma nova rede Layer 2 usando recursos do kernel Linux; um daemon configura essa rede e gerencia o roteamento entre as máquinas. Ele próprio é implementado através de um serviço que deve ser instalado em cada host do espaço.

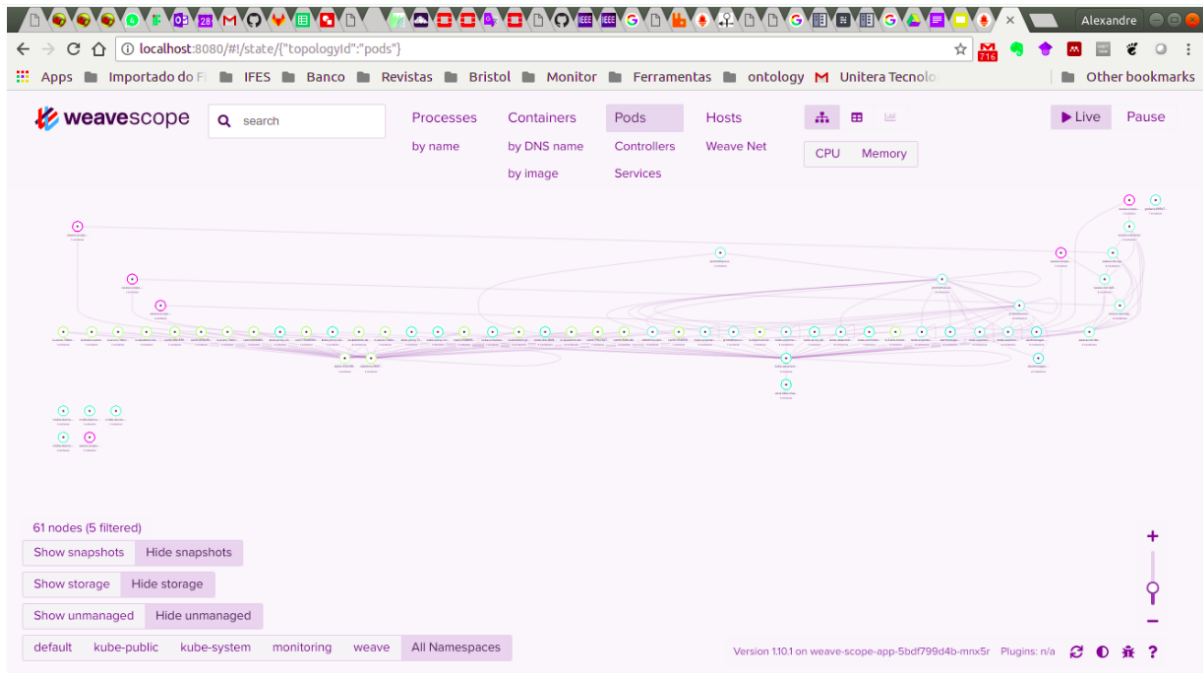
O Weave implementa a descoberta de serviços, fornecendo um servidor “micro DNS” em cada nó. Assim, os serviços podem ser acessados através de nomes sem que seja necessária configurações de rede, incluindo balanceamento de carga entre várias instâncias do mesmo serviço.

O objetivo do Weave é simplificar a configuração de uma rede de contêineres. Cada contêiner pode encontrar o IP de qualquer outro contêiner usando uma simples consulta ao sistema de nome de domínio (DNS, do inglês Domain Name System) no nome do contêiner, e também pode se comunicar diretamente sem uma tradução de endereços de rede (NAT, do inglês Network Address Translation), sem usar mapeamentos de portas ou reconfiguração de links. Além disso, a utilização do Weave não requer alterações no código dos serviços.

Além disso, o weave fornece seu próprio dashboard. A Figura 18 mostra uma janela do dashboard com todos os serviços do PIS utilizados no Espaço Inteligente implantado no laboratório High Performance Networks (HPN) na Universidade de Bristol. São 62 containers de serviços conectados pelo Weave, divididos entre serviços de uma aplicação e

serviços de infraestrutura.

Figura 18 – Dashboard do Weave.

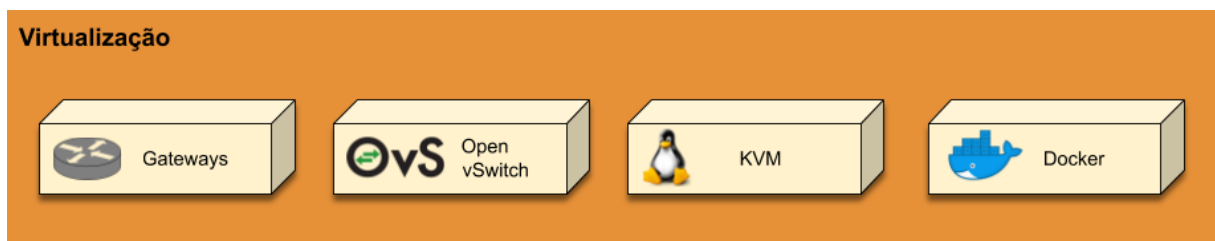


Fonte: Produção do próprio autor

4.1.4 Camada de virtualização

A camada de virtualização tem como função abstrair os recursos de hardware para utilização pelas demais camadas da arquitetura. Os sensores e atuadores, rede e dispositivos computacionais fazem parte do conjunto de elementos que são virtualizados no PIS, como pode ser visto na Figura 19.

Figura 19 – Representação da implementação da camada de virtualização do PIS.



Fonte: Produção do próprio autor.

A virtualização dos recursos computacionais é realizada em três níveis no PIS. O primeiro nível é realizado por máquinas virtuais (VM, do inglês Virtual Machine) e é utilizado para alocação de recursos de longo prazo. Nesse caso, um conjunto de VMs

são alocadas para um projeto ou usuário, e estarão disponíveis para que uma ou mais aplicações sejam executadas. VMs de usuários ou projetos diferentes são virtualmente separadas e o excesso de carga em um projeto não tem influência em um outro e vice versa, proporcionando um alto grau de isolamento. Para implementação das máquinas virtuais foi utilizado o Kernel-based Virtual Machine (KVM) e o Linux é o Sistema Operacional em todos os hosts e guests.

Containers são usados como o segundo nível de virtualização de recursos computacionais. Os containers fornecem desempenho e agilidade na utilização de microsserviços e por isso são utilizados para alocação de recursos de curto prazo. É possível determinar a quantidade de memória alocada ou partes dos núcleos de uma CPU, por container. Além disso, alterações na quantidade de recursos alocados podem ser facilmente realizados em tempo de execução.

Docker é a implementação de containers utilizada no PIS. Na arquitetura, o docker se comporta tanto como um elemento de virtualização de recursos computacionais, quanto como uma forma de empacotamento de serviços.

Tanto a virtualização por máquinas virtuais quanto por containers podem ser utilizados no PIS de forma isolada ou conjunta. O Espaço Inteligente implantando no laboratório HPN da Universidade de Bristol usa as duas formas de virtualização, principalmente em função da necessidade de isolar projetos simultâneos e completamente independentes. Já os Espaços Inteligentes do Instituto Federal do Espírito Santo (IFES) - Campus Vitória e da Universidade Federal do Espírito Santo (UFES), utilizam apenas a virtualização por containers.

A GPU é o terceiro nível de virtualização de recursos computacionais. Foram utilizadas duas maneiras de virtualizar as GPUs. A tecnologia de GPU passthrough é uma delas. Nesse caso, a GPU é toda disponibilizada para uma única máquina virtual, sem que haja compartilhamento simultâneo com outra VMs. Nesse caso, o host fornece a VM um endereço para acesso direto ao hardware. Os drives do dispositivo são instalados na VM. Esse modelo de virtualização é restritivo pois impede o compartilhamento entre múltiplas VMs, porém é o que apresenta melhor desempenho.

Uma segunda forma de virtualização de GPU é feita através da virtualização da memória do dispositivo. Um serviço requisita o recurso de uma GPU informando a quantidade de memória necessária para sua execução. Dessa forma, múltiplos serviços podem compartilhar a mesma GPU até o limite de sua memória total. Essa virtualização é realizada pelo serviço `gpushare-scheduler-extender`. Dessa forma, o orquestrador tem a possibilidade de alocar os serviços de acordo com a disponibilidade de um conjunto de GPUs.

Há a possibilidade também que o desenvolvedor tenha como requisito um determi-

nado modelo de GPU. Nesse caso, o orquestrador poderá organizar as GPUs em subgrupos e realizar a alocação de acordo com os requisitos e disponibilidade desses recursos.

A virtualização de rede é implementada através da ferramenta Open vSwitch (OVS). Open vSwitch é uma implementação de software de um switch virtual multicamada, projetado para permitir a automação de rede eficaz por meio de extensões programáticas (LINUX FOUNDATION, 2020).

O Open vSwitch opera tanto como um switch de rede baseado em software rodando em uma máquina virtual, como em um hardware de switch dedicado. No PIS, ambas as situações são utilizadas. O OVS é utilizado tanto pelo KVM, quanto o Openstack. Ferramentas essas utilizadas na implementação da arquitetura do PIS.

No caso da execução em hardware, diferentes modelos de switch são utilizados. Além disso, o OVS permite a divisão de um equipamento em múltiplos switches virtuais com gerenciamento independente. Dessa forma, em algumas implantações foi disponibilizado apenas parte do equipamento para gerenciamento pelo PIS.

Gateways de dispositivos

Sensores e atuadores utilizam o conceito de entidades virtuais, que são as representações das entidades físicas no mundo digital. Essas entidades tem a função de simplificar o uso de dispositivos heterogêneos através de uma padronização. Além disso deve promover o seu compartilhamento sempre que possível.

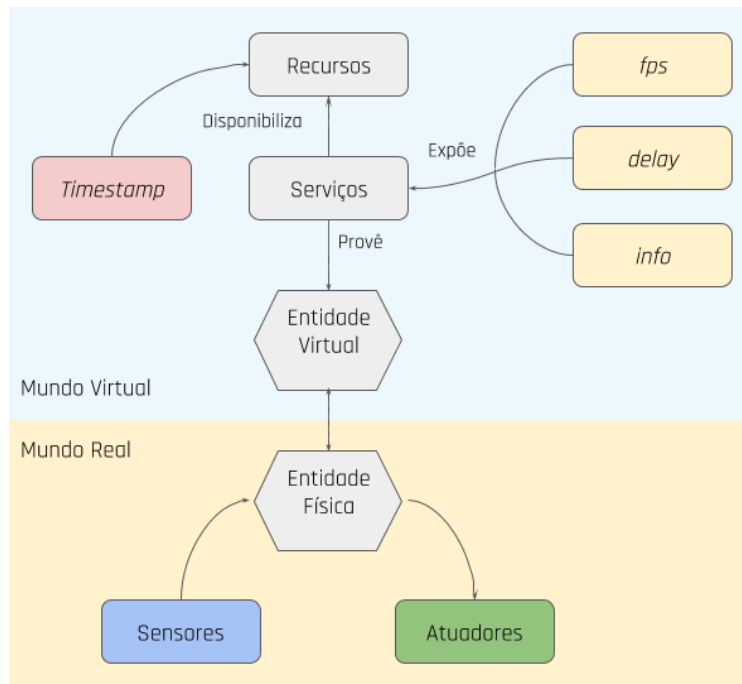
Em (CARMO et al., 2016) foi apresentado um modelo ontológico conceitual para os dispositivos do espaço inteligente, onde é mostrada a relação entre cada entidade. A representação desse modelo utilizada para sua implementação é mostrada na Figura 20.

A entidade virtual deve prover serviços, e através desses, disponibilizar recursos. Além disso, deve expor serviços que sejam capazes de alterar parâmetros da entidade física. Cada entidade possui um nome, que é constituído pela classe à qual ela pertence, seguida de um ID único, da seguinte maneira: “classe.ID”.

Foi definido um conjunto mínimo de recursos e serviços que toda entidade virtual deve conter para interação com as demais entidades do PIS:

- o recurso de timestamp, que indica o instante no qual algum dado do dispositivo foi capturado, ou ainda, o instante em que um dado foi inserido em um banco de dados;
- o serviço de fps, para que possa alterar a taxa de aquisição de dados;
- o serviço de delay, que deverá aplicar um atraso de até um período em apenas um período de amostragem, logo em seguida que o serviço for requisitado;

Figura 20 – Entidade Virtual.



Fonte: Adaptado de (QUEIROZ, 2016)

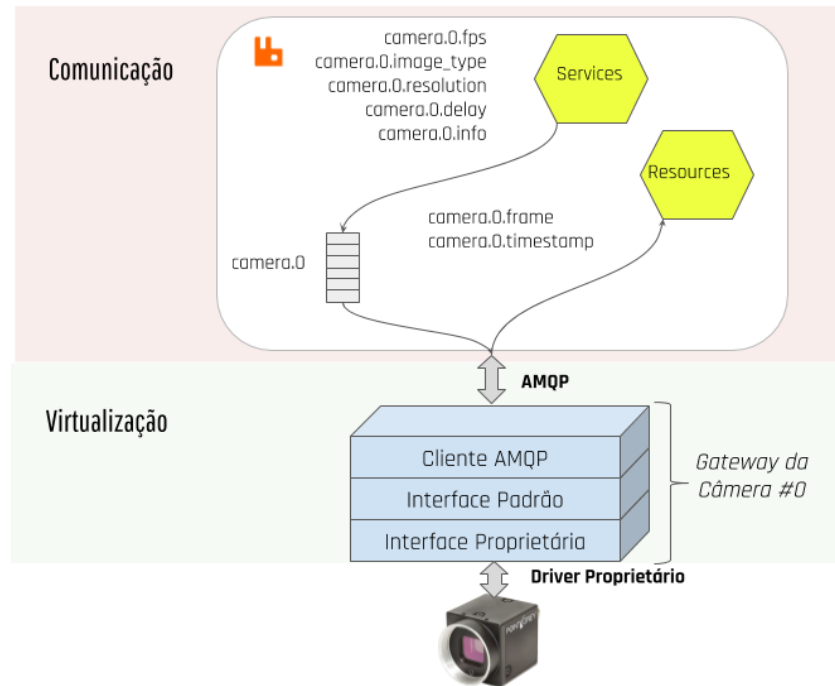
- o serviço de info, que deverá fornecer uma estrutura de dados com a identificação da entidade virtual na infraestrutura, bem como os seus recursos e serviços oferecidos.

A implementação das entidades virtuais é feita através de serviços de gateway para cada tipo de dispositivo. Como exemplo, a Figura 21 mostra a representação do gateway de uma das câmeras utilizadas no Espaço Inteligente da UFES e sua relação com a camada de comunicação.

O gateway da câmera está subdividido em três partes: interface proprietária, que corresponde as bibliotecas fornecidas pelo fabricante; interface padrão, que são as padronizações feitas que devem ser seguidas por qualquer câmera inserida à infraestrutura; e o cliente AMQP, responsável por fazer a interface com a camada de comunicação. Cada serviço de gateway de câmera possui uma fila no broker com o seu nome, seguindo o padrão já apresentado: "camera.ID". Como pode ser visto na camada de comunicação da Figura 21, a fila de nome "camera.0" possui os seguintes bindings ligados ao exchange de serviços:

- FPS: "camera.0.fps";
- Tipo de imagem: "camera.0.image_type";
- Resolução: "camera.0.resolution";

Figura 21 – Gateway da câmera.



Fonte: Adaptado de (QUEIROZ, 2016)

- Delay: “camera.0.delay”;
- Informação: “camera.0.info”.

Além disso, os recursos oferecidos por cada câmera são enviados ao exchange de recursos com os seguintes `routing_keys`.

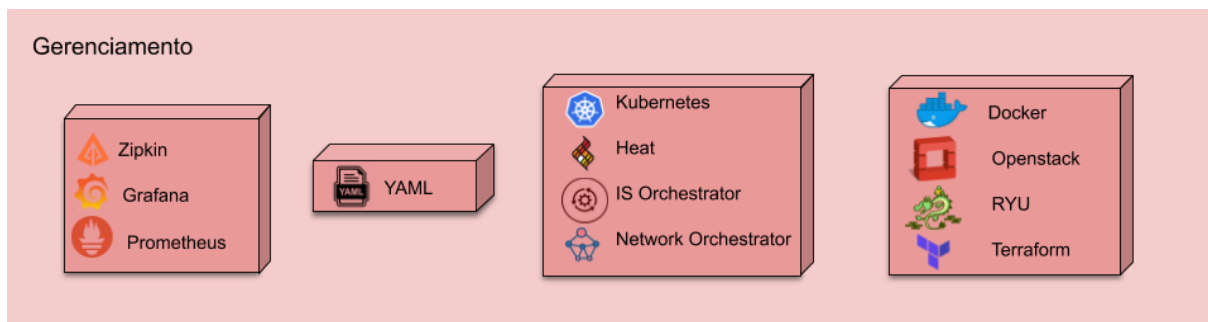
- Image: “camera.0.frame”;
- Timestamp: “camera.0.timestamp”.

Foram desenvolvidos gateways para cada tipo de sensor ou atuador utilizado nos Espaços Inteligentes implantados. Ainda que um gateway desenvolvido para um modelo de câmera possa servir de base para um segundo modelo, as especificidades desse segundo modelo em alguns casos exigem alterações no gateway inicial. Porém, o acesso pelas demais entidades da arquitetura é padronizada e transparente para os desenvolvedores.

4.1.5 Camada de gerenciamento

Na camada de gerenciamento, foram utilizadas várias ferramentas que implementam as funções descritas nessa camada. Sua representação pode ser vista na Figura

Figura 22 – Representação da implementação da camada de gerenciamento do PIS.



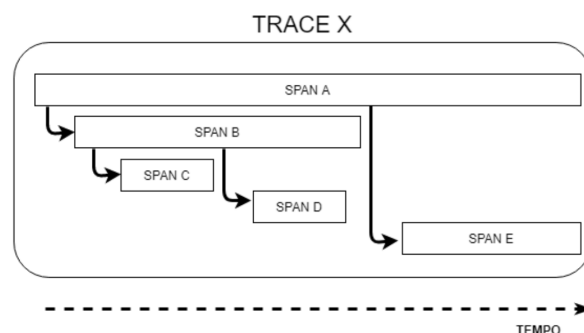
Fonte: Produção do próprio autor.

Observabilidade

Dentro de observabilidade, a função de rastreo é definida como um método para monitorar aplicações distribuídas, principalmente em uma arquitetura de microsserviços ([OPENTRACING, 2020b](#)). Os objetivos principais são a descoberta de falhas e análise de desempenho.

O padrão utilizado para rastreamento é o OpenTracing ([OPENTRACING, 2020a](#)). Tal padrão utiliza metadados dinâmicos para propagar a relação de causalidade entre os spans. Span pode ser definido como um conjunto de informações específicas de uma aplicação e infraestrutura, tal como registros de tempo e tarefa. O padrão define o formato para os spans e a convenção semântica para seu conteúdo. A Figura 23 proporciona uma visão clara de um rastreamento, isto é, como os spans estão relacionados no tempo e a relação de causalidade entre si.

Figura 23 – Exemplo de um rastreamento.



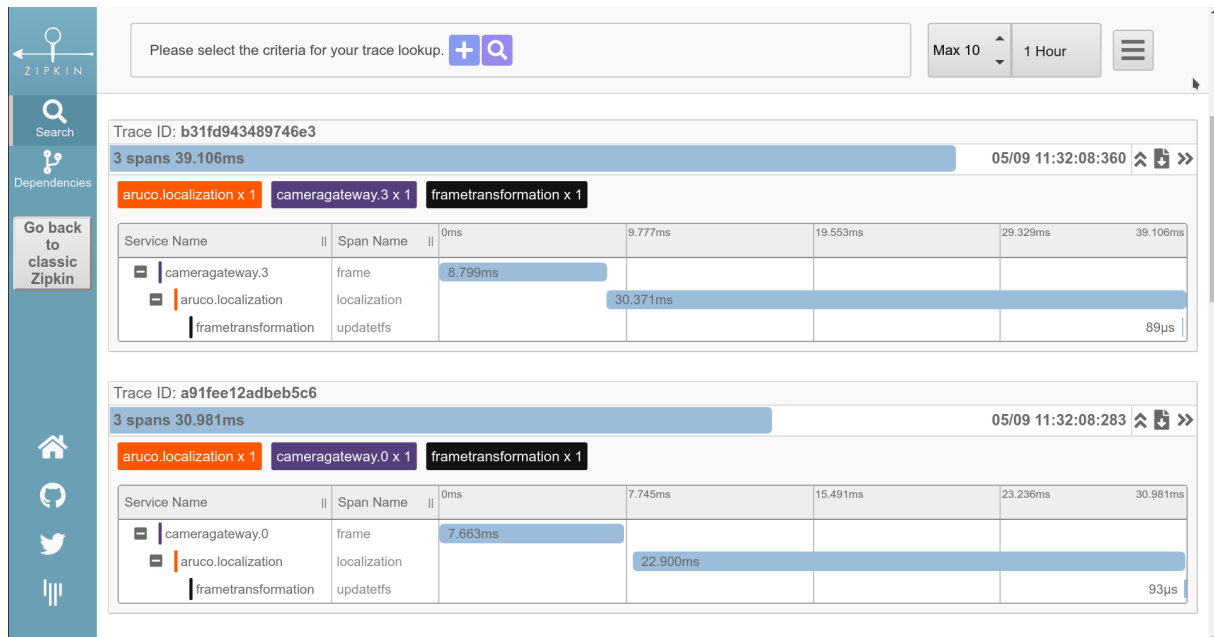
Fonte: ([COTTA, 2020](#))

Para se utilizar o rastreamento, é necessário a inserção de fragmento de códigos dentro do código base do serviço, mais especificamente no início e no final da parte que se deseja coletar métricas. Esse processo de adição do código extra, que será executado quando o sistema estiver realizando suas funções, é chamado de instrumentação ([GEIMER](#)

et al., 2009).

A ferramenta de rastreamento utilizada no PIS que implementa o padrão OpenTracing é o Zipkin (ZIPKIN, 2020). Uma de suas vantagens é o suporte a múltiplos protocolos de comunicação, incluindo o AMQP e um dashboard de fácil utilização como mostrado na Figura 24.

Figura 24 – Dashboard do Zipkin.



Fonte: Produção do próprio autor.

O Zipkin é focado na coleta e exibição dos traces para o usuário, o que o torna uma importante ferramenta para avaliação de desempenho de uma aplicação. Com ela, por exemplo, é possível realizar a medição dos tempos de processamento de cada serviço que compõe a aplicação, bastando para isso, a instrumentação dos serviços pelo desenvolvedor. Porém, tal como outras ferramentas similares, o Zipkin não possui a opção de medição dos tempos de comunicação entre serviços.

Para preencher essa lacuna de informação foi desenvolvida o serviço CommTracing para o PIS (COTTA, 2020). O serviço, em conjunto com alterações na biblioteca is-wire da camada de suporte a aplicação, agrega informações sobre o tempo de comunicação entre serviços e as envia para o Zipkin.

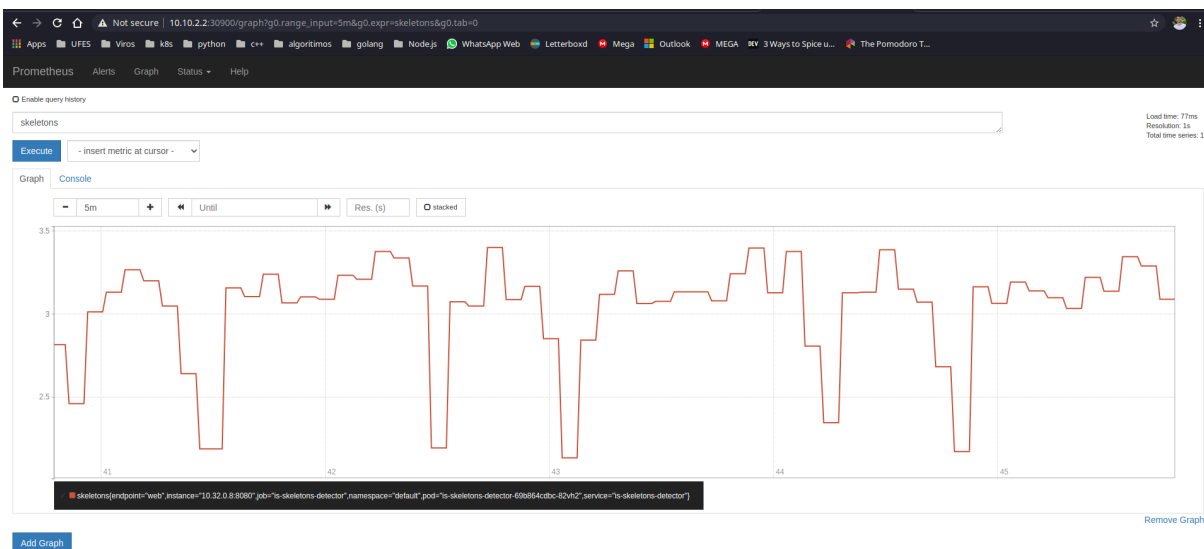
Uma das vantagens dessa abordagem é que não é necessário que o desenvolvedor adicione novas porções de código nos seus serviços. A medição do tempo de comunicação é realizada de forma independente. Além disso, é possível observar de forma unificada as medições de tempo de processamento, tempo de comunicação e o tempo de resposta de qualquer tarefa de uma aplicação. Além do mais, foi possível aproveitar toda a infraestrutura

da ferramenta Zipkin, sem que tenha sido necessário a sua modificação.

O Prometheus é um sistema de monitoramento para serviços e aplicações. Ele coleta as métricas de seus alvos em determinados intervalos, avalia expressões de regras, exibe os resultados e também pode acionar alertas se alguma condição for observada como verdadeira. Ele utiliza um modelo de dados multidimensional como uma série temporal identificados pelo nome da métrica e pares chave/valor (PROMETHEUS, 2020).

Os principais componentes do Prometheus incluem o servidor que lida com a coleta e armazenamento das métricas, um gateway que recebe as métricas das aplicações e encaminha ao servidor, um gerenciador de alertas e um dashboard para consulta dos dados e gerenciamento da ferramenta. Uma tela do Dashboard pode ser vista na Figura 25.

Figura 25 – Dashboard do Prometheus.



Fonte: Produção do próprio autor.

Para o monitoramento das aplicações, o Prometheus captura indicadores e métricas por meio de dois métodos:

- Instrumentação: adicionando código de bibliotecas ao código-fonte do serviço monitorada.
- Agentes: São utilizados para coletar métricas de sistemas de terceiros. Como exemplo, podemos citar o node-exporter que monitora o hardware do host e as métricas do kernel.

O Prometheus coleta métricas de seus alvos através do protocolo HTTP. As métricas são armazenadas num banco de dados de séries temporais para posteriormente serem

consultadas. Os dados coletados podem ser consultados através de seu dashboard ou através de consultas por ferramentas externas. A consulta externa é realizada utilizando-se a linguagem PromQL. Utilizando esta linguagem, é possível consultar os detalhes das métricas em um dado instante da linha de tempo

No PIS, o principal ciclo de controle do processo de orquestração passa pelo Prometheus. As métricas são coletadas dos serviços através de sua instrumentação, e da infraestrutura através dos agentes. As métricas coletadas são enviadas para o servidor Prometheus para armazenamento. Orquestrador então consulta o servidor para obtenção das métricas e as utiliza para definir a alocação de recursos.

Ainda que seja possível utilizar o dashboard do Prometheus para visualização dos dados, o Grafana se mostra uma ferramenta mais adequada para esse fim. O Grafana permite que você consulte, visualize, alerte e entenda as métricas, independentemente de onde estejam armazenadas ([GRAFANA LABS, 2020](#)).

O Grafana possui integração nativa com o Prometheus facilitando a importação e visualização dos dados. Além das métricas, é possível criar visualizadores de logs e outros dados não estruturados. Os logs no PIS são gerados pelos serviços e expostos através do sistema de logs do Docker. Esses logs são então consultados e apresentados pelo Grafana.

O Grafana também possibilita a criação de painéis dinâmicos e personalizados de forma simples e rápida. A Figura 26 mostra um segundo dashboard personalizado, construído com o Grafana para acompanhamento e análise do experimento executado no estudo de caso do Capítulo 5.

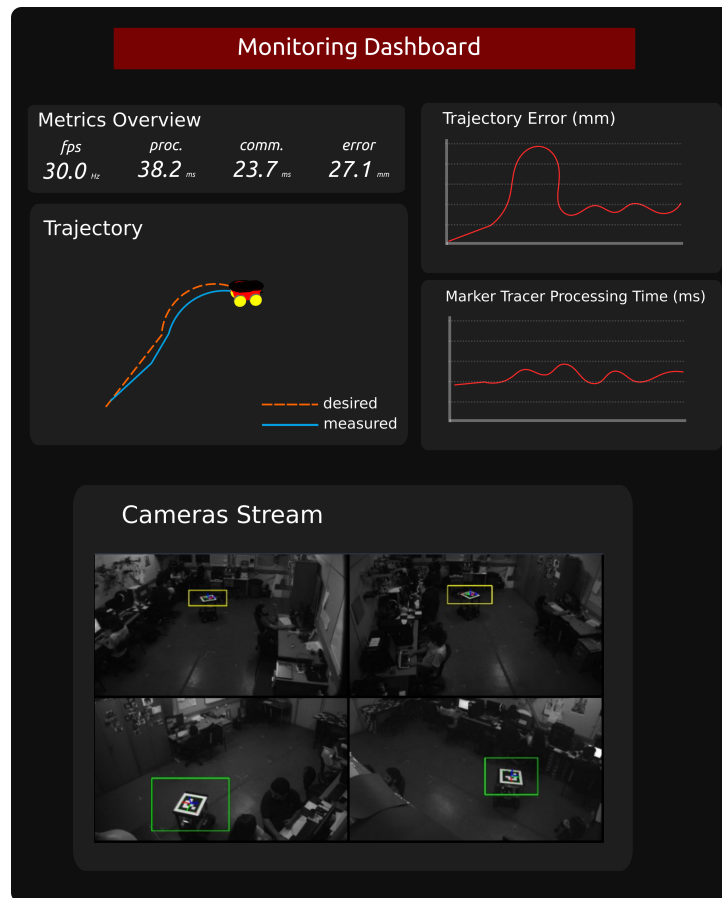
Orquestração e Compartilhamento de Informações Multinível

O Kubernetes é o principal orquestrador de containers no PIS. Ele é um sistema de código aberto para automatizar a implantação, dimensionamento e gerenciamento de serviços em contêineres ([KUBERNETES, 2020](#)). Foi baseado em dois sistemas de gerenciamento de containers usados internamente no Google: Borg e Omega e anos de experiência operando contêineres em grande escala no Google ([BURNS et al., 2016](#)).

O Kubernetes é composto por um conjunto de ferramentas que são utilizadas para executar aplicações baseadas em serviços distribuídos. As principais funcionalidades utilizadas no PIS são:

- Descoberta de serviço e balanceamento de carga: O Kubernetes pode expor um contêiner usando o nome DNS ou o próprio endereço IP com o suporte do Weave. Se o tráfego para um contêiner for alto, o Kubernetes pode balancear a carga e distribuir o tráfego de rede para que a implantação da aplicação seja estável.
- Gerenciamento de armazenamento: Permite a montagem automática de sistema de

Figura 26 – Dashboard personalizado do Grafana.



Fonte: Produção do próprio autor

armazenamento, sejam eles armazenamentos locais ou provedores de nuvem pública, por exemplo.

- Implementações, migrações e reversões automatizadas: É possível descrever o estado desejado para seus contêineres em implantação usando o Kubernetes. Ele pode alterar o estado real para o estado desejado em uma taxa controlada. Assim, em um processo de migração, novos containers são criados e removidos de forma a manter o estado predefinido e a aplicação funcionando durante todo o processo.
- Alocação automática de containers: Os recursos de computação disponíveis, são definidos através de um cluster de nós para o Kubernetes. De posse dos requisitos de CPU e memória de cada container, o Kubernetes realiza a sua alocação buscando o menor desperdício dos recursos.
- Autocorreção: O sistema de autocorreção do Kubernetes reinicia os contêineres que falham, substitui os contêineres, elimina os contêineres que não respondem à verificação de integridade definida pelo usuário. Um novo container somente é anunciado aos clientes quando ele está no estado pronto.

O kubernetes utiliza diferentes objetos como abstrações dentro de seu conjunto de ferramentas. Dentre esses objetos destacam-se o pod, cluster, e Deployments

A unidade de gerenciamento do Kubernetes é o Pod, que são conjuntos de um ou mais containers. Pods com mais de um container devem existir quando há uma dependência entre eles. É o caso de sidecars que proveem comunicação ou monitoramento a um serviço, por exemplo.

Um cluster Kubernetes é um agrupamento de nós de computação, sejam eles físicos ou virtuais. Do ponto de vista das aplicações, o cluster se comporta como um único servidor com a soma de todos os recursos individuais dos nós. Ao Kubernetes cabe a orquestração desses recursos distribuídos. Nós computacionais podem ser adicionados e retirados, mesmo em tempo de execução. Nesses casos, o Kubernetes fará a realocação de recursos de forma a atender aos serviços em execução.

Um Deployment é um objeto do Kubernetes que pode representar uma aplicação em execução em seu cluster. Ele permite que você descreva o ciclo de vida de uma aplicação, especificando seus componentes e recursos. De acordo com a descrição, o Kubernetes irá monitorar o estado atual da aplicação e se necessário alterar para o estado desejado.

Deployments são criados escrevendo um manifesto. Esse manifesto é descrito através de arquivos de configuração padronizados do tipo na linguagem YAML (do inglês, YAML Ain't Markup Language). YAML é um formato baseado em texto legível para especificar informações de tipo de configuração. É utilizado em diferentes ferramentas como Openstack, controladores SDN e o próprio Kubernetes.

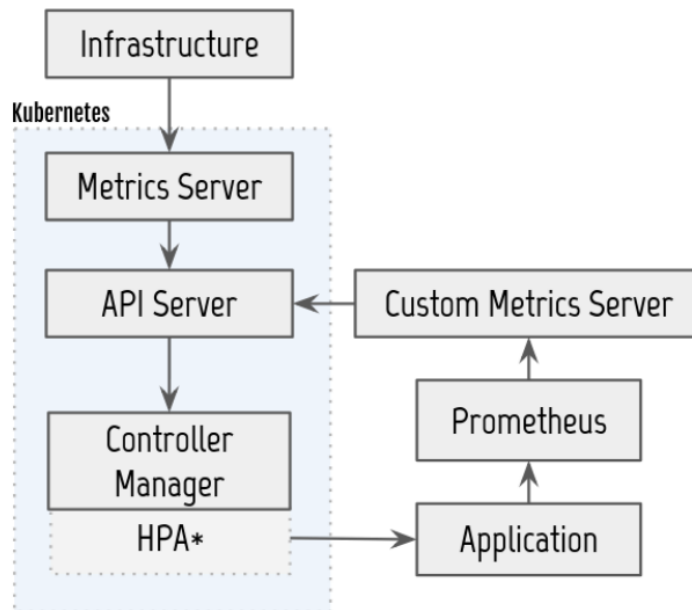
Os Deployments já permitem fazer a descrição de uma aplicação de forma que os recursos possam ser gerenciados pelo Kubernetes. Mas as informações contidas nessa descrição ainda são insuficientes para o propósito do PIS.

Dessa forma, a função de compartilhamento de informações multinível do PIS foi implementada de forma a aproveitar o modelo de descrição utilizado no Kubernetes. Nesse modelo, foram criadas extensões para que um orquestrador de mais alto nível faça a sua leitura e envie comandos os orquestradores de nível inferior, como o próprio Kubernetes, ou diretamente para os gerenciadores de recursos.

O Kubernetes tem suporte nativo para orquestração automática com base em métricas de infraestrutura, como CPU e uso de memória. Ele possui uma API que permite que sua funcionalidade principal de orquestração seja estendida. Esta API pode ser utilizada para informar o Kubernetes sobre outros tipos de métricas. A Figura 27 mostra como o Kubernetes foi estendido para realizar a orquestração usando métricas no nível de aplicação e também da infraestrutura.

O custom metric server é responsável para coletar métricas de serviços do Prometheus e converter em um formato para que o escalador automático horizontal de Pods

Figura 27 – Integração Kubernetes e Prometheus.



Fonte: (PICORETI et al., 2018)

(HPA, do inglês Horizontal Pod Autoscaler) do Kubernetes seja capaz consumir. O HPA então, dimensiona automaticamente o número de pods dos serviços baseado nessas métricas. Com métricas de ambas as camadas, o Kubernetes pode realizar a orquestração dos serviços no PIS.

O Kubernetes é um orquestrador de containers. Apesar de containers ser a principal forma escolhida para implementação dos serviços, é necessário a utilização de orquestradores que atuam em outros níveis.

Para o nível de máquinas virtuais foi implantado o orquestrador Heat. Ele é o principal projeto para orquestração no OpenStack. Ele implementa um mecanismo de orquestração em nuvem com base em modelos no formato de arquivos de texto que podem ser tratados como código. Os modelos permitem a criação da maioria dos tipos de recursos, como instâncias ou volumes, bem como algumas funcionalidades mais avançadas, como escalamento de instâncias de VMs, por exemplo.

Para o nível de rede, foi desenvolvido o serviço de Handover Orchestration. Esse serviço faz a orquestração da rede em dois subníveis a rede de acesso de rádio baseada no padrão 802.11n e a rede de switches SDN. Ele utiliza informações de localização dos dispositivos por imagem para ativação da célula de comunicação sem fim, bem como para definição de encaminhamento na rede de switches.

O sistema de orquestração do PIS é realizado por múltiplos orquestradores organizados de forma hierárquica. Cada orquestrador dos níveis inferiores possui uma visão local, e pode tomar decisões dentro do seu escopo de atuação. Porém, o seu alcance de

atuação limitado.

Como o orquestrador de nível mais alto na hierarquia foi desenvolvido o IS Orchestrator. Esse serviço lê o arquivo com a descrição das informações multinível para inicialização das aplicações, e configura os orquestradores dos níveis inferiores. Durante todo o período de execução, ele continuamente coleta as informações observadas das camadas de aplicação e infraestrutura e reconfigura os demais orquestradores. Além disso ele é capaz também de acionar diretamente os gerenciadores de recursos.

Gerenciamento de Recursos

Os gerenciadores de recursos criam uma camada de abstração que facilitam a atuação nos equipamentos e dispositivos físicos através da camada de virtualização. No PIS são utilizados gerenciadores em três níveis diferentes: Máquinas virtuais, Containers e Rede. Para os sensores e atuadores, sua abstração de gerenciamento está implementada diretamente nos gateways de cada dispositivo.

O OpenStack é utilizado no PIS como infraestrutura de nuvem privada. Tal como o Heat para orquestração, o NOVA é o principal projeto para gerenciamento de máquinas virtuais ([OPENSTACK FOUNDATION, 2020d](#)) e por isso foi escolhido para ser o gerenciador desse tipo de recurso.

Outro gerenciador de máquinas virtuais utilizado foi o Terraform. O Terraform é uma ferramenta para criar, alterar e gerenciar versões de infraestrutura com segurança e eficiência ([HASHICORP, 2020](#)). Ele é uma ferramenta agnóstica capaz se integrar com gerenciadores de diferentes provedores de serviço. No PIS ele é utilizado para alocação das VMs que compõem o clusters Kubernetes dentro do OpenStack.

A função de gerenciamento de container é realizada principalmente pelo próprio Docker. O Docker prevê tanto um padrão para empacotamento dos serviços, quanto ferramentas para gerenciamento dos containers. Essas ferramentas são inclusive, utilizadas pelo Kubernetes para orquestração dos serviços.

Já para o gerenciamento da rede foi escolhido o RYU. Ele é um controlador SDN com uma API bem definida que permite a sua integração com novas aplicações. O RYU suporta diferentes protocolos de gerenciamento de dispositivos de rede, incluindo o OpenFlow ([RYU, 2020](#)).

Todos os gerenciadores de recursos dos diferentes níveis recebem comandos dos orquestradores e são utilizados para atuarem no ambiente de forma simples e padronizada.

4.2 Aplicações

O PIS foi concebido para o domínio de aplicações de robótica e visão computacional com requisitos altamente exigentes do ponto de vista de recursos da infraestrutura. Além disso, o PIS deve prover acesso padronizado a recursos heterogêneos, reusabilidade de serviços, escalabilidade, entre outras características que facilitem a utilização do PIS por um amplo conjunto de aplicações.

De forma a mostrar como essas características podem ser aplicadas a uma diversidade de aplicações, um conjunto dessas aplicações foi implementado e experimentos foram executados no PIS. Dentro desse conjunto destacam-se as aplicações para controle de dispositivos em tempo real e interação com o ambiente.

Um exemplo é a aplicação de controle de formação de robôs móveis. O objetivo principal é mostrar como a odometria visual fornecida pelo PIS pode ser utilizada em tarefas robóticas, de maneira distribuída e com o erro mínimo de estimativa da postura. Para isso foi desenvolvido um controlador de formação, que em conjunto com outros serviços já existentes no Espaço Inteligente, foi capaz de atingir o objetivo traçado. Detalhes dessa aplicação podem ser vistas no Anexo [A](#).

Uma vez que a infraestrutura de espaço inteligente utilizada é baseada em serviços, novas aplicações poderão ser implementadas a partir da composição destes e de outros serviços. Essa característica de reutilização de serviços transforma o PIS em uma promissora plataforma que servirá de base para novos trabalhos na área de robótica.

Um exemplo da flexibilidade e reaproveitamento de serviços do PIS pode ser observado com a aplicação do MobiLysa descrita no Anexo [B](#). O MobiLysa é uma aplicação de controle do cão guia robô Lysa. Através da aplicação, um usuário deficiente visual poderá indicar o destino desejado em um ambiente indoor, seja por voz ou outra forma de interação, e ser guiado até o local de maneira independente e segura.

A técnica de localização utilizada é a fusão de visão computacional com a odometria do robô. Dessa forma busca-se uma maior precisão proporcionada pela visão computacional, em conjunto com uma maior robustez proporcionada pela odometria. Já que, mesmo no caso de falha da cobertura visual o robô continuará tendo informações de localização para a sua navegação.

Nesse caso, a aplicação possui dois grandes requisitos que devem ser atendidos: A precisão de localização e o tempo de resposta ao usuário. Portanto, além da característica de reusabilidade de serviços, o PIS deverá promover a gerência dos recursos de forma que os requisitos da aplicação sejam atendidos.

Como último exemplo, podemos citar o serviço de detecção de pessoas desenvolvido para o PIS e que foi utilizado em diferentes aplicações. O serviço utiliza-se de uma rede

multi câmeras, e busca atender aos requisitos de tempo das aplicações considerando uma capacidade computacional limitada e compartilhada. Para validação do serviço implementou-se 3 diferentes aplicações de tempo real para serem usadas como prova de conceito. O detalhamento completo do serviço e suas aplicações pode ser visto no Anexo C.

Através dos experimentos realizados pode ser demonstrada e validada a eficácia do serviço de detecção humana no contexto do PIS com base em uma rede com várias câmeras. É importante enfatizar que a eficácia não é apenas propriedade da detecção de pessoas, mas também tem relação com o atendimento de requisitos de tempo e sua capacidade de se utilizar características presentes no PIS para que esse requisito seja atendido.

4.3 Espaços Inteligentes Implantados

Foram implantados três Espaços Inteligentes baseado em visão computacional. Esses espaços estão localizados na UFES, IFES Vitória e na Universidade de Bristol. Todos os espaços utilizam a arquitetura e implementações descritas nos capítulos e seções anteriores.

Ainda que o software tenha a mesma base, eles se diferem em relação ao espaço físico e hardware utilizado. A implantação de diferentes Espaços Inteligentes com equipamentos heterogêneos, mostra a flexibilidade da arquitetura.

A adoção da arquitetura aqui desenvolvida, proporciona aos usuários desses diferentes espaços a possibilidade de desenvolver e testar aplicações do domínio de visão computacional e robótica. O objetivo é propiciar um ambiente que atenda aos requisitos específicos e rigorosamente correlacionados dessas aplicações e ao mesmo tempo faça uso racional dos recursos da infraestrutura.

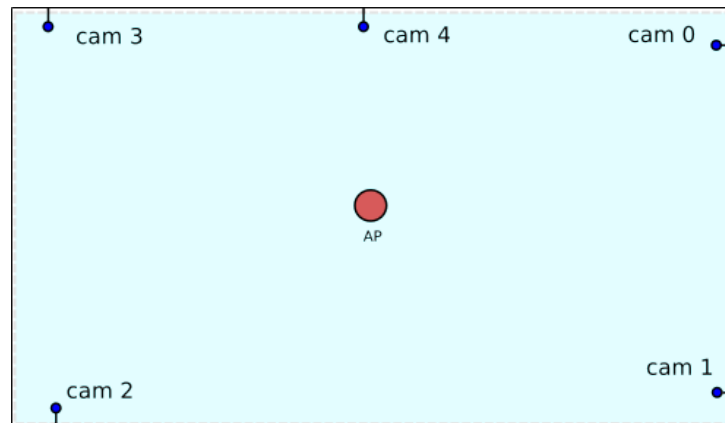
A seguir será feita a descrição do ambiente físico desses espaços e a infraestrutura de hardware utilizada em cada um.

4.3.1 UFES

O primeiro Espaço Inteligente foi implantado no laboratório de robótica e visão computacional da UFES. Ele possui uma área retangular de aproximadamente 30m². O ambiente possui 5 câmeras fixas com suas localizações indicadas na Figura 28. A cobertura visual das câmeras abrangem todo o laboratório, bem como a cobertura de rede sem fio.

Todos os serviços e ferramentas desenvolvidos foram testados nesse laboratório, exceto aqueles que interagem com equipamentos específicos de outro espaço, como por exemplo, o gateway do robô Pepper disponível na Universidade de Bristol .

Figura 28 – Abrangência do Espaço Inteligente da UFES.



Fonte: Produção do próprio autor.

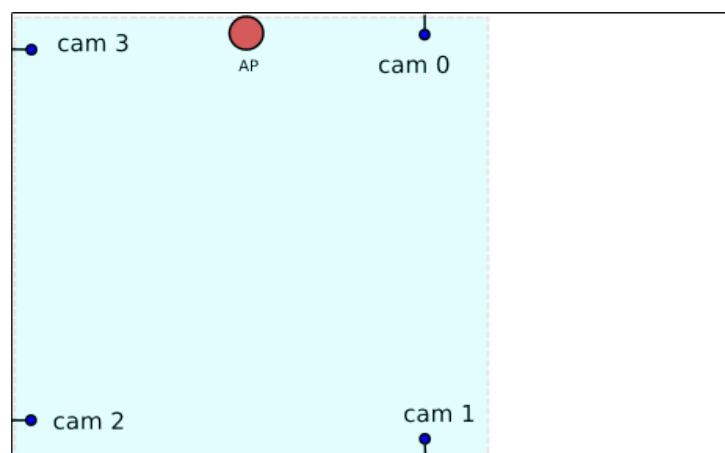
Os equipamentos de computação se localizam em um datacenter em um outro prédio a aproximadamente 300m do laboratório. A interligação entre os dois ambientes é realizada através de cabeamento ótico.

Durante o trabalho, o espaço sofreu diversas modificações. A lista de equipamentos de sua configuração atual pode ser vista na Tabela 4.

4.3.2 IFES - Vitória

O Espaço Inteligente do IFES Vitória foi implantado no laboratório de Tecnologias do Futuro. Ele possui uma área retangular de aproximadamente $54m^2$, porém as câmeras cobrem uma área quadrada de $36m^2$. Na Figura 29 a área de cobertura das 4 câmeras é representada pela área em azul. Todo o laboratório tem cobertura de rede sem fio.

Figura 29 – Abrangência do Espaço Inteligente do IFES Vitória.



Fonte: Produção do próprio autor.

A lista de equipamentos de sua configuração atual pode ser vista na Tabela 5.

Tabela 4 – Lista de equipamentos do PIS da UFES

Quant.	Equipamento	Descrição/Modelo
4	Câmera	Flir Blackfly GigE
1	Câmera	Intelbras VIP 3230 VF
1	Robô	Lysa
1	Robô	Pioneer 3AT
1	Servidor	CPU: 2 x Intel(R) Xeon(R) Silver 4214 (12C/24T); GPU: 3 x Nvidia Titan V e 1 x Nvidia Titan Xp; Mem: 64 GB de RAM; SSD: 256 GB Rede: 1 x GigE
1	Servidor	CPU: Intel(R) Core(TM) i7-7700 4.20GHz (4C/8T); GPU: 1 x Nvidia GeForce GTX 1080; Mem: 32 GB de RAM; HD: 1 TB; Rede: 1 x GigE
1	Servidor	CPU: Intel(R) Core(TM) i7-7700K 4.20GHz (4C/8T); GPU: Nvidia GeForce GTX 1070; Mem: 48 GB de RAM; HD: 1 TB; Rede: 1 x GigE
1	Servidor	CPU: Intel(R) Core(TM) i7-6850K 3.60GHz (6C/12T); GPU: Nvidia GeForce GTX 1080 Ti; Mem: 64 GB de RAM; HD: 1 TB e 2 TB; Rede: 1 x GigE
1	Storage	Modelo: Qnap TS-832XU HD: 8TB
1	Switch	Mikrotik Crs328-24p-4s+rm L5
1	Switch	Pica8 P-3297
1	Acess Point	CISCO WAP371-A-K9

Fonte: Produção do próprio autor.

4.3.3 HPN - Bristol

O terceiro Espaço Inteligente foi implantado no laboratório do grupo HPN (High Performance Networks) na Universidade de Bristol. Ele possui uma área aproximada de 80m². As 8 câmeras instaladas no laboratório proporcionam uma cobertura visual de aproximadamente 50m², representada na Figura 30 pela área do retângulo azul.

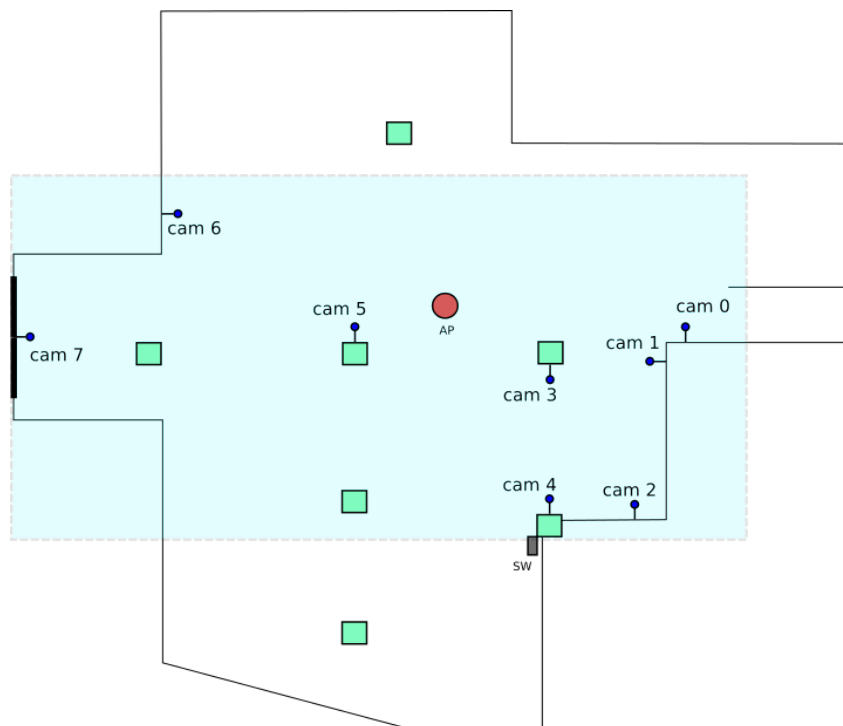
O Datacenter do grupo fica ao lado do laboratório logo após a porta localizada junto a câmera 7. A lista de equipamentos de sua configuração atual pode ser vista na Tabela 6.

Tabela 5 – Lista de equipamentos do PIS do IFES

Quant.	Equipamento	Descrição/Modelo
4	Câmera	Flir Blackfly GigE
1	Servidor	CPU: 1 x Core i5-7400 (4C/4T); GPU: 1 x GeForce GT 730; Mem: 8 GB de RAM; HD: 1 TB Rede: 4 x GigE
1	Switch	CISCO SG100D-08P
1	Acess Point	Linksys EA2700

Fonte: Produção do próprio autor.

Figura 30 – Abrangência do Espaço Inteligente da Universidade de Bristol.



Fonte: Produção do próprio autor.

Tabela 6 – Lista de equipamentos do PIS de Bristol

Quant.	Equipamento	Descrição/Modelo
8	Câmera	Flir Blackfly GigE
1	Robô	Pepper - SoftBank Robotics
1	Servidor	CPU: Intel Xeon Gold 5120 2.2G, 14C/28T; GPU: 2 x Nvidia V100; Mem: 192 GB de RAM; SSD: 960 GB HD: 2TB Rede: 4 x GigE, 2 x 10GigE
1	Servidor	CPU: Intel Xeon Gold 5120 2.2G, 14C/28T; Mem: 128 GB de RAM; SSD: 960 GB; HD: 2TB; Rede: 4 x GigE
1	Switch	Corsa DP2400
1	Switch	Pica8 P-3297
1	Acess Point	CISCO WAP371-A-K9

Fonte: Produção do próprio autor.

5 Estudo de Caso 1: Indústria 4.0

Estamos no início de uma nova revolução industrial, na qual a Internet móvel, sensores vestíveis, Internet das Coisas, inteligência artificial (AI, do inglês Artificial Intelligence), computação em nuvem e robôs autônomos serão combinados em sistemas ciber-físicos (CPS, do inglês Cyber-Physical Systems). Os efeitos dessa revolução nas fábricas e outras instalações de fabricação são geralmente referidos como Indústria 4.0.

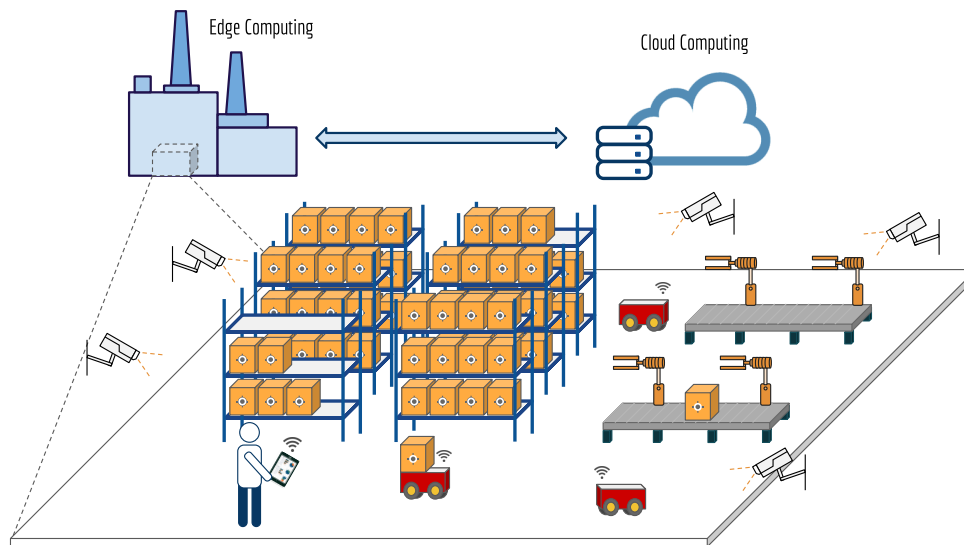
A robótica móvel é possivelmente uma das principais forças motrizes por trás dessa revolução (LU, 2017). Prevê-se que os processos da Indústria 4.0 adotem maciçamente robôs móveis (e até autônomos) de diferentes formas e tamanhos, para revolucionar a fabricação e a logística, tirando proveito de seus graus mais altos de liberdade em comparação com seus antecessores. As baterias evoluíram imensamente nas últimas décadas e os cabos de energia não são mais necessários. Assim, a conectividade sem fio limitada, confiável e onipresente com latência liberará os robôs para se movimentarem mais livremente.

O paradigma da robótica em nuvem também trará requisitos de latência importantes (KEHOE et al., 2015), à medida que leva a capacidade de processamento e os loops de controle dos robôs físicos para instalações de computação remota em outros lugares. Mesmo com o advento da computação de borda (ou seja, mais próxima das instalações do que a nuvem principal), a latência E2E (por exemplo, abaixo de milissegundo (AL., 2017)) permanece uma restrição fundamental para garantir a estabilidade dos loops de controle, onde os algoritmos são executados remotamente.

Como exemplo do cenário da Indústria 4.0 descrito acima, a Figura 31 mostra uma representação de uma linha de produção com um armazém automatizado, robôs móveis e usuários. Há também a necessidade de soluções escalonáveis e simplificadas para a próxima evolução da Internet das Coisas. Como consequência do uso generalizado da IoT na indústria, o número de dispositivos por metro quadrado (e possivelmente a demanda de largura de banda) em breve disparará (WOLLSCHLAEGER; SAUTER; JASPERNEITE, 2017). Em relação à comunicação sem fio, esse fato pode levar os planos atuais para a implantação de células pico e femto em direção à célula atto (isto é, células sub-quadradas) em instalações industriais.

Portanto, para enfrentar os desafios mencionados, são esperadas redes mais densas e ágeis. E as informações de contexto, especialmente as informações de localização, podem ser uma estratégia revolucionária para lançar aplicações confiáveis e sem fio em tempo real (e de missão crítica) no contexto da Indústria 4.0. Como mencionado em (YASSIN et al., 2017a), as informações de localização tornaram-se um fator chave nos sistemas de comunicação atuais, permitindo que serviços específicos sejam oferecidos ou executados de

Figura 31 – Robôs móveis na indústria 4.0.



Fonte: (CARMO et al., 2019).

acordo com a localização do o dispositivo ou do usuário.

Abordagens inovadoras que exploram a cooperação e a coordenação com mecanismos de localização para tecnologias moveis ou sem fio são certamente uma tendência a ser seguida nas áreas de rede e robótica. Abordagens como as apresentadas em (CONTI et al., 2012) e (WIN; GIFFORD; THOMAS, 2011) mostram como o uso de modelos de erro de alcance e a cooperação entre dispositivos podem melhorar a localização em redes de comunicação sem fio. Também (WIN et al., 2018) os autores demonstram que a fusão multi sensor e a cooperação espacial podem aumentar significativamente a localização em um cenário de grande escala com centenas de agentes móveis.

Por exemplo, o reconhecimento preciso da localização permite que o sistema de comunicação execute o direcionamento de feixe e o controle dinâmico de potência, o que pode resultar em símbolos mais curtos para comunicações de menor latência. Além disso, a transferência preditiva e preventiva em uma cobertura extremamente densa baseada em células pode ser alimentada por mecanismos de rastreamento de robôs. Por último, mas não menos importante, os sistemas de localização também podem ajudar os protocolos da camada superior. Por exemplo, reconfigurações, como roteamento dinâmico na rede de backhaul (LIBERATO et al., 2016), precisam ser implementadas de forma proativa, a fim de fornecer uma experiência de serviço sem problemas para usuários 5G (Di Taranto et al., 2014) .

No entanto, ambientes industriais geralmente impedem o uso de métodos tradicionais de localização baseados em radiofrequência, pois comumente esses ambientes encontram-se cheios de obstáculos de diferentes materiais, formas e tamanhos que podem obstruir os sinais entre emissores e receptores. Muitos métodos de localização baseados em

RF exigem um caminho não obstruído entre um transmissor e um receptor. A força do sinal, o atraso do sinal e o ângulo de chegada da propagação de ondas eletromagnéticas serão provavelmente prejudicados por níveis severos de interferências em instalações industriais internas (BRENA et al., 2017).

Uma abordagem promissora para obter precisão de localização em torno de alguns centímetros ou menos é usar métodos de visão computacional em ambientes industriais (POSADA et al., 2015; BRENA et al., 2017). Geralmente, esses ambientes apresentam muitas câmeras que são usadas para fins de monitoramento e vigilância. Contudo, extrair informações precisas de localização, a partir de imagens capturadas por uma rede de câmeras, geralmente requer uma largura de banda grande e algoritmos complexos para serem executados em tempo real. Além de robôs móveis e outras máquinas, os ambientes da Indústria 4.0 também estarão lotados por telefones celulares, tablets e outros dispositivos IoT que exigem serviços com rápido feedback, como por exemplo, para evitar acidentes ou atrasos na produção.

Considerando todas essas questões, o PIS foi utilizado para permitir a cooperação entre um sistema de localização visual e a infraestrutura de comunicação de um ambiente industrial. Mais precisamente, foi implementada uma cobertura de attocélulas interna integrada a um método de localização de dezenas de centímetros de visão computacional. Essa estratégia permite atender ao requisito de erro de trajetória de um robô de uma aplicação em tempo real no contexto da Indústria 4.0. Será mostrada a correlação entre esse requisito específico da aplicação e o tempo de resposta para execução da tarefa de controle do robô. Serviços de nuvem em borda e a rede SDN foram usados para reduzir o tempo de resposta. As experiências realizadas mostram que o tempo de processamento do serviço de localização e o tempo de comunicação da rede sem fio são os componentes mais significativos do tempo de resposta. Assim, foi empregada a orquestração de recursos de computação para reduzir o tempo de processamento de serviços e um gerenciamento granular da rede através de uma arquitetura sem fio programável baseada em SDN, para reduzir o atraso na entrega enquanto um robô se move no ambiente.

O que será demonstrado com esse estudo de caso:

- O PIS como um facilitador para atender aos rigorosos e específicos requisitos de aplicações de robótica no ambiente da Indústria 4.0.
- A programabilidade granular da infraestrutura proporciona a criação de attocélulas baseadas em serviços de localização por visão computacional. A utilização dessas attocélulas permitem que a densidade de dados por área seja reduzida e, como consequência, os requisitos específicos de aplicações de robótica em tempo real sejam atendidos.

- A integração dos níveis de aplicação e infraestrutura por meio do PIS pode gerar benefícios mútuos. A aplicação pode ter os seus requisitos atendidos, mesmo em um ambiente de dados de alta densidade por área, enquanto a infraestrutura assume uma nova funcionalidade que pode ser oferecida a outras aplicações.

5.1 Cobertura sem fio e localização Indoor

Atualmente as operadoras de rede móvel implantam estações base com áreas de cobertura muito heterogêneas. Eles variam de algumas macrocélulas com vários metros quadrados (para áreas rurais escassamente povoadas) a um grande número de pequenas antenas (geralmente para espaços internos e densamente povoados, como estações de metrô), cobrindo apenas alguns metros quadrados. Este último é conhecido como femtocélula. Existe até a iniciativa de usar sistemas de luz visível (TSONEV; VIDEV; HAAS, 2014) e antenas no chão (THIELENS et al., 2017) para criar femtocélulas cobrindo apenas alguns dm^2 . No entanto, problemas de propagação eletromagnética em instalações industriais podem exigir soluções inovadoras, uma vez que as soluções atuais para fornecer altas taxas de dados ainda estão vinculadas a menos dispositivos que compartilham um ponto de acesso (AP, do inglês Access Point) específico.

Em relação ao tempo de comunicação para dispositivos móveis, a camada física sem fio é o primeiro link. A latência da camada física possui três elementos básicos: codificação de símbolos e processamento de envio; transmissão; processamento de recepção e decodificação de símbolos. O primeiro e o último são fortemente orientados para a tecnologia (software/hardware). A transmissão, no entanto, será definida pela estrutura de quadro escolhida, da qual a duração do símbolo é uma parte importante. Para começar, a duração atual do símbolo da multiplexação ortogonal por divisão de frequência (OFDM, do inglês Orthogonal Frequency Division Multiplexing) é excessivamente longa (cerca de 1 ms) para garantir metas de baixa latência. Uma estratégia é diminuir o espaçamento entre subportadoras. Isso reduzirá a duração do símbolo OFDM e, como resultado, o intervalo de tempo de transmissão (AL., 2017).

Na Indústria 4.0, há vários dispositivos simultaneamente exigindo serviços de comunicação com alta confiabilidade e baixa latência (TARNEBERG et al., 2017). Infelizmente, porém, os mecanismos de controle de link que fornecem confiabilidade no LTE (do inglês Long Term Evolution) podem não ser viáveis em uma latência de transmissão tão baixa usando o tradicional HARQ (do inglês Hybrid Automatic Repeat Request) e ARQ (do inglês Automatic Repeat Request) (AL., 2017).

Portanto, métodos de alocação de recursos reativos, que integram métodos de localização e conectividade sem fio, foram empregados pelo sistema de comunicação para resolver o problema de mobilidade e os requisitos de baixa latência nas redes sem fio Indoor

(MUPPIRISETTY; YIU; WYMEERSCH, 2016).

No caso da localização baseada em RF, existem métodos projetados desde as primeiras gerações de sistemas de rádio móveis celulares. O posicionamento do identificador celular estava presente na segunda geração (2G). Na terceira geração (3G), o sistema de posicionamento baseado em tempo usando sinais de sincronização relevantes à comunicação foi o mecanismo empregado. Sinais de referência de posicionamento dedicados também foram empregados na quarta geração (4G). Dessa forma, a precisão alcançável no 4G é de cerca de dezenas a alguns metros, ou até um pouco menos (XU; CHOU, 2017; ZHOU et al., 2017).

Além disso, como já mencionado na seção anterior, existem métodos que usam uma estratégia de cooperação entre dispositivos sem fio para reduzir o erro e obter melhor precisão de localização (WIN; GIFFORD; THOMAS, 2011). Mas, embora esse tipo de abordagem possa atingir uma precisão inferior a 15 cm (WIN et al., 2018), ainda pode enfrentar problemas de latência em ambientes de alta densidade. Atrasos e falhas na comunicação podem ocorrer devido ao alto número de dispositivos e é provável que problemas como falhas de conectividade, causando latência na transferência, possam aparecer nessas situações.

Atualmente, os métodos de localização e rastreamento baseados em RF dependem da qualidade das medições coletadas e da metodologia aplicada. A maioria deles usa a chamada multilateração, que basicamente usa geometria para combinar as estimativas de alcance de diferentes dispositivos. Essas estimativas podem ser provenientes de diferentes medidas, como RSS (do inglês Received Signal Strength), ToF (do inglês Time-of-Arrival), TDoA (do inglês Time Difference Of Arrival) e AoA (do inglês Angle of Arrival) (BRENA et al., 2017). Geralmente, a precisão submetro requer um oscilador de precisão de nanossegundos e medições precisas.

Além disso, a localização baseada em RF apresenta algumas limitações intrínsecas, como baixa precisão do relógio devido a osciladores de baixo custo, sincronização, atenuação e estabilidade. Métodos de variação baseados no tempo, como ToF e TDoA, dependem de medições precisas de tempo de transmissão. Esse tempo de transmissão está sujeito a desvios devido à presença de muitas estruturas metálicas que causam perda direta, atraso e múltiplos caminhos do sinal (YASSIN et al., 2017a). A força do sinal recebido (RSS) e a correlação de ToF com a distância são severamente afetados devido à interferência destrutiva e construtiva dos componentes que gera efeitos do Small scale fading (DARDARI; CLOSAS; DJURIĆ, 2015). E mesmo quando o método não depende da sincronização, como o AoA, requer uma linha de visão direta para realizar medições (BRENA et al., 2017). Como resultado, as reivindicações do sistema de localização baseado em RF de precisão de 1 m podem não se aplicar à maioria dos ambientes industriais.

Portanto, para algumas aplicações que precisam de sistemas de localização, como

rastreamento visual de robótica e máquinas, essa precisão é insuficiente. A precisão buscada deve ser uma ordem inferior a 1 m, enquanto a comunicação deve ocorrer com baixo tempo de comunicação, movendo-se na direção 5G (Di Taranto et al., 2014).

5.2 Attocélulas e Localização por Visão Computacional

Em relação à latência da camada física, quando se trata da interface sem fio, as attocélulas podem ser aplicadas para reduzir as condições de propagação de múltiplos caminhos, permitindo o uso de símbolos mais curtos (AL., 2017). No entanto, nesse caso, são necessárias mais reconfigurações no backhaul e, portanto, o sistema de localização pode ser explorado como um método para alocar proativamente recursos de rede (LEE; YOO, 2017). Como a configuração da rede pode depender das atualizações da tabela em um número grande e variável de equipamentos (LIBERATO et al., 2016), uma melhor agilidade da rede pode ser alcançada se a previsão de mobilidade for usada para antecipar alterações na topologia e executar o reencaminhamento antes das interrupções da rota.

De fato, o sistema de localização pode ajudar a reduzir o overhead geral e a latência nas camadas física e de rede, movendo-se na direção de enfrentar os desafios 5G. Sob essa perspectiva, os métodos de localização baseados na visão computacional e projetados para dispositivos móveis e robótica (CLARK; TRIGONI; MARKHAM, 2015; TSAI; CHANG; CHEN, 2016; RAMPINELLI et al., 2014; ALMONFREY et al., 2018) podem fornecer um sistema de localização com precisão suficiente para suportar as attocélulas.

Os autores de (CLARK; TRIGONI; MARKHAM, 2015) discutem alguns dos problemas ao usar métodos baseados em visão computacional para localização indoor, mas também propõem meios de contornar esses problemas e implementar um sistema de odometria inercial visual para um dispositivo móvel. Sua abordagem alcança precisão de posicionamento submétrica e, com esse resultado, os autores afirmam que a odometria visual inercial pode fornecer os níveis de precisão de posicionamento necessários para uma adoção generalizada.

Para ajudar na tarefa de monitorar câmeras de vigilância, é proposto um sistema de posicionamento interno baseado em visão para edifícios inteligentes (TSAI; CHANG; CHEN, 2016). Os autores aplicam a subtração dos elementos não essenciais da imagem para separar o objeto em primeiro plano. Depois disso, eles rastreiam esses objetos e usam a transformação linear direta para integrar informações da câmera a um mapa panorâmico e localizar os objetos em movimento.

Em (RAMPINELLI et al., 2014), o sistema de localização visual usa uma matriz de leds infravermelhos em um quadro negro para estimar a pose do robô. Essas informações são usadas para calibrar as quatro câmeras que pertencem ao espaço inteligente implantado no laboratório. A precisão alcançada é suficiente para calibrar as câmeras, corrigir a odometria

do robô e controlar o robô durante aplicações em tempo real no ambiente.

Outros marcadores foram utilizados como método de localização de robôs durante a evolução do PIS. Um exemplo mostrado no Capítulo 4, um serviço foi desenvolvido para localização de um padrão preto e branco fixado em cima de um robô móvel. Mais uma vez a precisão alcançada foi da ordem de alguns centímetros e todos os experimentos foram executados em tempo real.

Portanto, se considerarmos a implantação do PIS em um ambiente de Indústria 4.0, o conceito de "localização como um serviço" pode ser oferecido e usado não apenas para fornecer recursos de navegação aos robôs, mas também para ajudar a definir attocélulas para atender à comunicação. Portanto, é importante a criação de infraestruturas de comunicação programáveis que possibilitem a utilização de sistemas de localização baseados em visão computacional.

Oportunidades de utilização de sistemas de localização multicâmeras

Atualmente, muitos ambientes industriais possuem uma rede de câmeras instaladas para monitorar o ambiente devido a razões de segurança. Mas esse sistema com várias câmeras também pode ser usado para localizar dispositivos móveis. As câmeras são sensores com um amplo campo de visão, capazes de fornecer uma grande quantidade de informações sobre objetos e usuários no local de trabalho sem interferir diretamente no próprio ambiente. Além da localização de robôs móveis, as câmeras em ambientes industriais também podem ser usadas para reconhecimento de objetos e gestos, além de situações de risco.

No entanto, existem alguns desafios em relação ao uso da visão computacional e do processamento de imagens para localizar dispositivos móveis. Um desses desafios é lidar com condições adversas de iluminação, especialmente em ambientes externos. Embora em ambientes internos as condições de iluminação sejam mais constantes, ainda pode haver uma influência significativa sobre os algoritmos de extração de dados e a precisão de localização pode ser alterada.

A precisão do sistema de localização também é altamente dependente do processo de calibração. Se as câmeras estiverem calibradas corretamente e sua posição relativa for conhecida, é possível obter uma precisão de alguns centímetros ao localizar qualquer tipo de objeto ou indivíduo. Quando uma das dimensões do objeto é fornecida, estimar sua localização pode ser simplificado, mas ainda é fortemente dependente da qualidade da calibração. No entanto, uma vez concluída a calibração, não há necessidade de recalibração, a menos que as câmeras tenham sua pose (posição e orientação) alterada. Além disso, não há erro acumulativo no processo de localização da visão por computador, ou seja, mesmo que o robô se mova pelo local de trabalho, o erro de localização é constante e não muda com o deslocamento.

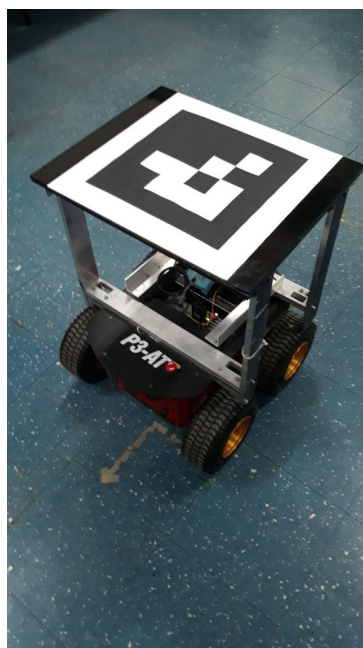
Além disso, normalmente em ambientes com várias câmeras, quanto mais câmeras houver com mais áreas sobrepostas, melhor será a precisão da localização. Também é importante garantir que todos os quadros sejam capturados ao mesmo tempo, ou seja, as câmeras devem ser sincronizadas, para que a posição atual dos objetos em movimento possa ser determinada com o menor erro possível.

Considerando a grande quantidade de dados, o uso da visão computacional também exige alta largura de banda e baixa latência entre os nós de processamento. Mas se a infraestrutura permitir que o processamento pesado seja paralelizado e executado na nuvem, a visão computacional se tornará uma solução estratégica para localização e controle. Por isso, o PIS equipado com uma rede de câmeras e equipamentos de comunicação sem fio, fornecendo programabilidade adequada no nível de aplicação e infraestrutura, é um facilitador estratégico para implementar esta solução.

5.3 Aplicação de Controle Visual de Robôs Móveis em Tempo real

Como exemplo para o ambiente da Indústria 4.0, será utilizada uma aplicação que comandará um robô móvel indoor em tempo real por meio de uma estratégia de controle com feedback visual. As informações visuais são usadas para recuperar a pose do robô para controlar sua velocidade e trajetória. Para facilitar sua detecção, o robô tem um padrão visual conectado à sua plataforma (Figura 32).

Figura 32 – Robô móvel com o padrão Aruco



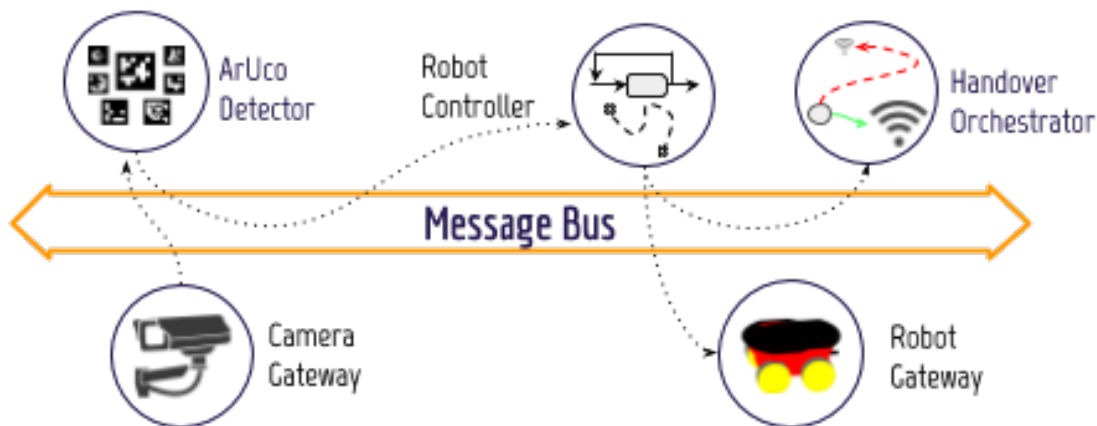
Fonte: (CARMO et al., 2019).

Neste estudo de caso, o padrão adotado como o marcador anexado ao robô é conhecido como ArUco ([GARRIDO-JURADO et al., 2014](#)). Este tipo de marcador pode ser detectado muito rapidamente e permite recuperar sua pose sem ambiguidades.

ArUco é um padrão binário e apresenta um algoritmo de detecção e correção automática de erros. Além disso, cada marcador ArUco corresponde a um código de identificação única (ID). Devido aos recursos mencionados, muitos marcadores podem ser detectados ao mesmo tempo na mesma imagem com um custo computacional fixo. Os marcadores usados em nosso teste tiveram suas dimensões registradas e essas informações foram usadas para recuperar a pose tridimensional (3D) do robô, que tem um marcador ArUco anexado à sua plataforma. Como as dimensões desse marcador são conhecidas, o robô pode ser localizado mesmo que seja visto por apenas uma das câmeras instaladas no PIS.

Esse estudo de caso foi realizado na implementação do PIS da UFES. A Figura 33 mostra os principais serviços que compõe a aplicação em questão. Toda a comunicação entre os componentes da aplicação passa pelo serviço Message Bus. O Message Bus, os gateways e o Handover Orchestrator, são classificados como serviços de infraestrutura. Já o Aruco Detector e Robot Controller são exemplos de serviços de domínio.

Figura 33 – Cadeia de Serviços que compõe a Aplicação.



Fonte: ([CARMO et al., 2019](#)).

Todos os serviços estão em execução na nuvem, exceto o gateway do robô que está sendo executado dentro do robô. Como dito anteriormente, os gateways expõem recursos físicos dos dispositivos como serviços. Para isso, eles devem traduzir a forma de interação de cada um desses dispositivos (câmeras, robôs etc.) em um modelo padronizado na arquitetura do PIS.

Os principais serviços que compõe a aplicação de controle de robô em tempo real estão descritos a seguir:

- **Camera Gateway:** existe um gateway associado a cada câmera e ele é o responsável por estabelecer uma conexão com a câmera usando o protocolo específico. Após a conexão, o gateway passa a capturar imagens RAW e faz a conversão para um formato padrão (neste caso, JPEG), além de disponibilizar essas imagens para todos os outros serviços. Um exemplo de serviço que recebe imagens do Camera Gateway é o Aruco Detector. Além disso, o Gateway da câmera pode receber comandos para ajustar parâmetros como espaço de cores, FPS e resolução da imagem.
- **Aruco Detector:** recebe as imagens capturadas pelas câmeras e executa o algoritmo de detecção ArUco. Se o ID de um ArUco detectado tiver uma dimensão associada, o serviço calcula sua posição e orientação 3D no referencial do mundo.
- **Robot Controller:** recebe a pose do robô do serviço Aruco Detector e atualiza as informações necessárias para o controlador do robô. A pose do robô é acumulada ao longo do tempo e, em seguida, usada pelo controlador, que aplica uma restrição temporal para definir as velocidades lineares e angulares que serão enviadas ao robô a cada instante. Além de gerar os sinais de controle para o robô, este serviço também retorna sua pose atual no referencial do mundo.
- **Robot Gateway:** semelhante ao Camera Gateway, este serviço cria a conexão com o robô e pode ser usado para configurar alguns dos parâmetros do robô. Além disso, este serviço é responsável por receber os comandos do Robot Controller e transmiti-los ao robô, mas não antes de converter os comandos no protocolo proprietário específico do robô.
- **Handover Orchestrator:** usa a pose do robô, retornada pelo Robot Controller, para orquestrar a transferência entre as células de comunicação sem fio. O Handover Orchestrator recebe informações sobre a localização do robô através de serviços do Espaço Inteligente, em sua concepção. No entanto, o Handover Orchestrator também está controlando outros dois blocos fundamentais: i) uma rede de acesso via rádio baseada no padrão 802.11n usando uma nova solução sem fio definida por software e ii) uma rede de retorno composta por switches SDN para fornecer comunicação entre o ambiente de trabalho do robô, as câmeras do Espaço Inteligente e a nuvem para acessar os demais serviços necessários à aplicação (MARTINEZ et al., 2018). Em vez de usar as informações do SNR (do inglês signal-to-noise ratio) para fazer uma operação de transferência, o Handover Orchestrator utiliza as informações fornecidas pelos serviços de domínio do Espaço Inteligente.
- **Message Bus:** permite que serviços separados trabalhem juntos, mas em um modo dissociado, de forma que novos serviços possam ser facilmente adicionados ou removidos sem afetar os demais. O Message Bus roteia todos os dados entre os serviços por meio de mensagens. Na arquitetura do PIS, ele usa o modelo de

comunicação de publish/subscribe com o AMQP como protocolo de mensagens. As mensagens são publicadas no barramento e qualquer serviço que se inscreveu nesse tipo de mensagem o receberá. Esse modelo promove agilidade e flexibilidade em relação à comunicação de protocolo de alto nível comumente utilizada entre aplicações.

Apenas para ilustrar como esses elementos interagem, consideremos um loop de controle para mover o robô com feedback visual. A taxa do loop de controle está associada ao FPS configurado através do Gateway de cada câmera. A cada ciclo de controle, o serviço Aruco Detector consome as imagens fornecidas pelos Gateways da Câmera e tenta detectar o marcador do robô. Se o marcador inteiro for encontrado pelo menos em uma das imagens, sua pose em 3D é estimada.

A posição e a orientação do robô são assumidas como iguais ao centro do padrão. Depois disso, essas informações são usadas pelo serviço Robot Controller para produzir um comando, que é transmitido ao robô pelo Robot Gateway, usando redes com e sem fio no PIS. Além disso, a localização do robô também é usada pelo serviço Handover Orchestrator para reprogramar a rede sem fio sempre que necessário.

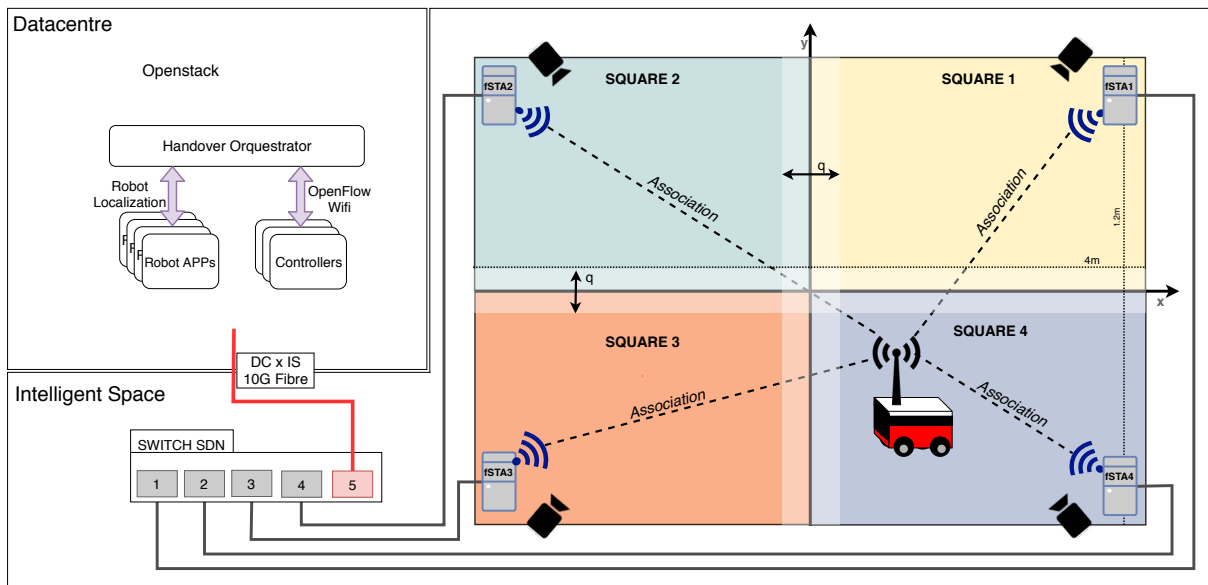
5.4 Arquitetura sem fio para implementação de attocélulas

O objetivo dessa abordagem para implementação de attocélulas é obter transferência de baixa latência em redes sem fio para suportar a mobilidade do robô. Para esse fim, a arquitetura foi estendida ([MARTINEZ et al., 2018](#)), e múltiplos AP são configurados como clientes sem fio IEEE 802.11 (WiFi), instalados no ambiente e conectados à infraestrutura SDN. Como parte do módulo de comunicação WiFi foi instalado um daemon hostapd no robô, para que ele funcione como um AP móvel.

Em nossa ambiente de teste do PIS, cada zona possui 1,2 m² de área e é coberta por um ponto de acesso (célula), conforme ilustrado na Figura 34. Esse tamanho é definido como uma célula, considerando as metas estabelecidas pelo 3GPP (Third Generation Partnership Program) ([3GPP, 2018](#)). O objetivo é obter uma densidade de tráfego superior a 1 Mbps/m² (ou 1 Tbps/km²) para fornecer uma taxa de dados ao cliente superior a 10 Mbps a um usuário por 10 m². Obviamente, adicionar mais células por m² fornece mais largura de banda por área. Porém um maior número de células aumento o número de vezes que será necessário realizar o handover entre as células. Por isso, para aproveitar a capacidade total da taxa de transferência de dados é necessário que o handover seja realizado o mais rapidamente possível.

Na Figura 34, existem 4 attocélulas trabalhando como servidores físicos com o Linux Operating System e o Open vSwitch instalados, os servidores são utilizados como

Figura 34 – Arquitetura sem fio com Handover.



Fonte: (CARMO et al., 2019).

bridges e estão equipados com uma interface WiFi e outra Ethernet. A interface WiFi é usada para associação no BSSID (do inglês Basic Service Set Identifier) criado pelo AP. Dessa maneira, todos os clientes no espaço inteligente são associados ao AP móvel, criando um esquema de multi-conectividade. As interfaces Ethernet nos clientes são usadas para a arquitetura de backhaul. O objetivo é explorar a vantagem de um AP móvel com vários clientes associados a ele. A operação de handover é realizada no backhaul, selecionando o cliente pelo qual rotear o tráfego usando o protocolo OpenFlow. Com essa operação única, a entrega e a atualização das rotas no backhaul são realizadas ao mesmo tempo, sem a necessidade de mecanismos adicionais de sincronismo entre eles.

Inicialmente, todas as attocélulas são associadas ao robô e, com base na posição do robô, o controlador SDN adiciona uma regra OpenFlow no switch de backhaul para rotear o tráfego pela attocélula correspondente. Isso permite testar diferentes técnicas de handover para definir o ponto de acesso adequado. O ponto de acesso é escolhido através das informações fornecidas pelo serviço Aruco Detector, dependendo da localização física do robô. As regras dos switches Openflow definirão o caminho para alcançar a porta do switch à qual o robô está conectado. O cenário é mostrado na Figura 34, com todas as attocélulas associadas e o tráfego passando pela attocélula 4 após a gravação da regra do OpenFlow para enviar tráfego através do switch SDN na porta 5.

5.5 Experimentos

Para avaliar a viabilidade de se utilizar o PIS para atingir os objetivos apresentados no início do capítulo, os experimentos foram separados em dois grupos e estruturados da seguinte forma:

- No primeiro grupo, o primeiro aspecto a ser investigado é a precisão da localização baseada na visão computacional. Será mostrado que com valores mais altos de FPS obtém-se maior precisão. Os valores de FPS são parâmetros configuráveis da rede multicâmeras do PIS. No entanto, quanto maior a precisão, maior a quantidade de recursos para atender aos rigorosos requisitos da aplicação em tempo real. Desta forma, deve haver um balanceamento entre precisão e uso dos recursos da infraestrutura que deve ser avaliada.
- O segundo aspecto que é investigado é o estabelecimento de attocélulas baseado em um serviço de localização visual. Uma avaliação do tempo de comunicação é realizada quando vários dispositivos (robôs ou outros) compartilham a mesma infraestrutura sem fio. Avaliamos o tempo de comunicação à medida que o número de dispositivos aumenta por m^2 , e comparamos com a solução aqui apresentada de attocélulas com handover eficiente.

5.5.1 Sistema de localização por visão computacional ofertado pelo PIS

Para avaliar o erro de localização no PIS com base na visão computacional, foram marcados 18 pontos igualmente espaçados no chão para recuperar sua posição a partir das imagens das câmeras. A Figura 35 mostra um frame de imagem capturado por uma das câmeras no PIS, onde podem ser vistos 15 de 18 pontos.

Como os pontos são visualizados por mais de uma câmera, todas as localizações estimadas são comparadas com a medida realizada in loco que é utilizada como referência. Essas medidas são usadas para calcular o erro médio relacionado a essa posição. As medidas de referência dos pontos no chão foram realizadas considerando a indicação do eixo no piso, que também pode ser vista na Figura 35. Os resultados são apresentados na Figura 36a, onde o raio dos círculos representa os erros médios de localização obtidos para cada posição e as cores dos círculos indicam quantas câmeras são capazes de ver cada posição em particular.

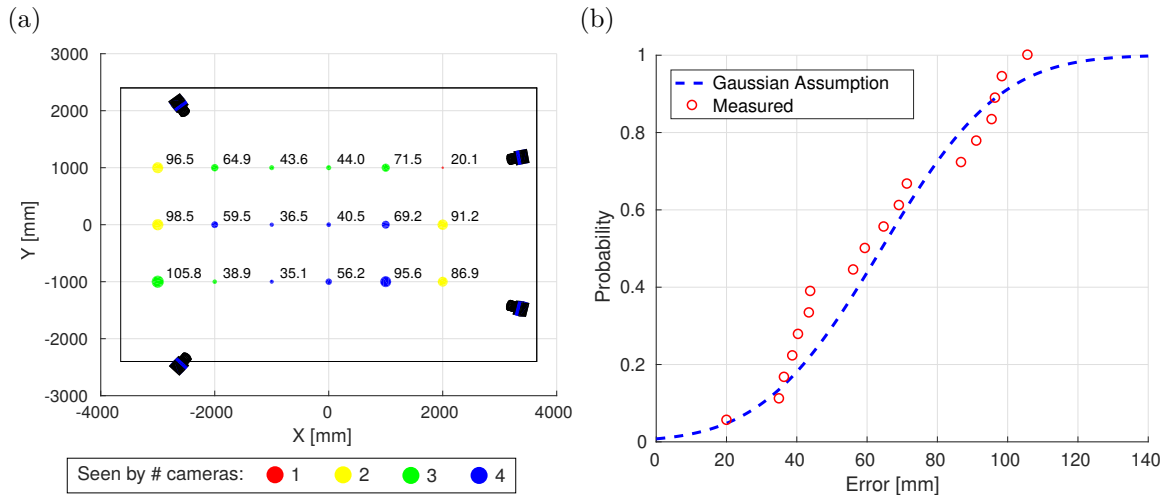
Considerando todo o espaço de trabalho, a Figura 36b mostra uma função de distribuição cumulativa (CDF, do inglês Cumulative Distribution Function) do erro de localização (junto com a curva gaussiana) em nosso protótipo PIS. Observe que o erro de localização permanece abaixo de 9 cm por 90 % de todo o espaço mapeado.

Figura 35 – Imagem capturada por uma das câmeras do PIS com os pontos marcados no chão



Fonte: (CARMO et al., 2019).

Figura 36 – Medições: a) Erro de localização nos 18 pontos marcados no chão b) CDF do erro de localização no PIS.



Fonte: (CARMO et al., 2019).

Normalmente, o número mínimo de câmeras necessárias para a reconstrução da pose do robô são duas, pois cada câmera que observa o robô fornece duas equações para um sistema de três variáveis (a coordenada 3D do robô). No entanto, se o espaço de trabalho do robô for plano, como em muitos ambientes industriais, uma única câmera poderá ser suficiente se uma das coordenadas de pose do robô for conhecida, por exemplo, sua altura. Apesar da possibilidade de localização com uma câmera, o espaço de trabalho do robô móvel pode conter oclusões e, portanto, são necessárias mais câmeras para cobrir todo o

ambiente.

Pode-se observar que o erro de reconstrução da pose do robô depende de muitas variáveis. O primeiro é a qualidade da calibração das câmeras: quanto melhor a qualidade da calibração, menor o erro de reconstrução. No entanto, o processo de calibração é complexo, pois cada câmera possui 12 parâmetros que relacionam um ponto 3D no ambiente a um pixel bidimensional (2D) na imagem. Além disso, geralmente a precisão da localização se torna melhor à medida que o número de câmeras, que vêem o ponto a ser reconstruído, aumenta. Mas isso deve ser um ponto de atenção, pois as câmeras que estão mais distantes do ponto reconstruído também podem adicionar erros e prejudicar a estimativa da posição. Como podemos observar na Figura 36a, embora a maioria dos pontos vistos por mais de uma câmera apresente melhor localização, o ponto visualizado por apenas uma câmera foi o que obteve o menor erro de reconstrução. A razão disso é que elementos como resolução de imagem, algoritmo de detecção, qualidade e características da lente também influenciam diretamente a qualidade da reconstrução. Essa dependência de muitas variáveis inviabiliza uma modelagem matemática do erro de localização.

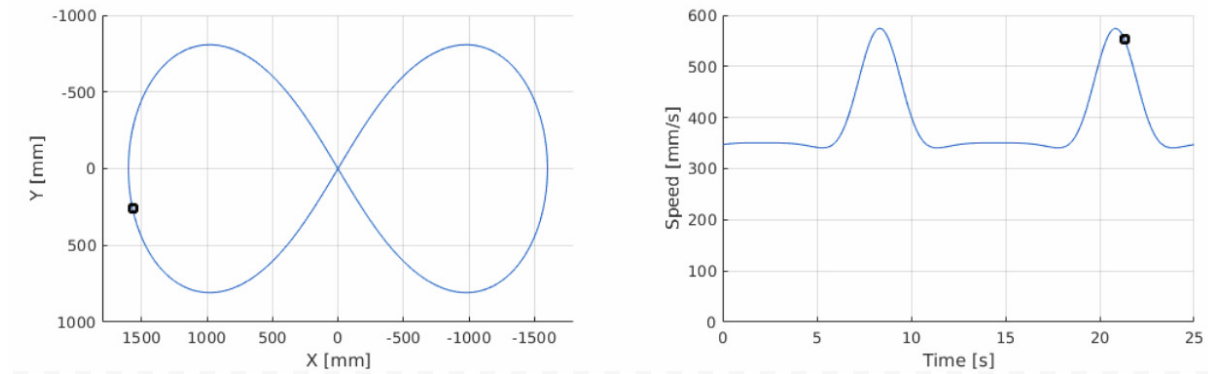
Embora esses primeiros resultados sejam medições estáticas das câmeras, eles demonstram que o PIS pode suportar facilmente um sistema de localização para utilização em attocélulas com incerteza abaixo de 10 cm. Além disso, a localização do robô pode ser identificada com precisão semelhante. Portanto, podemos construir uma estratégia de localização para detectar a migração entre as células usando o serviço de localização visual oferecido pelo PIS, em vez de usar a força do sinal convencional (e imprecisa) ou outros métodos baseados em RF.

5.5.2 Erro de trajetória x FPS

A aplicação usada para o experimento consiste em controlar um robô móvel por uma trajetória predefinida. As velocidades do robô são definidas de acordo com o tempo desejado por volta para executar todo o caminho. Nesse caso, em cada loop de controle, o robô deve estar em uma determinada posição com determinadas velocidades lineares e angulares para garantir que o caminho seja executado no tempo desejado. A forma escolhida para o caminho da trajetória é a Lemunscata de Bernoulli. Tanto o caminho quanto a velocidade linear esperada para um desempenho de 25 segundos por volta são ilustrados na Figura 37.

Essa trajetória foi escolhida devido à sua variação na velocidade e complexidade do caminho. Podemos considerar a trajetória dividida em duas partes: trajetória simples — (linha reta com velocidade constante) e trajetória complexa (caminho curvo com acelerações positivas e negativas). A diferença entre a trajetória desejada e a realizada pelo robô é definida como o erro da trajetória. Espera-se que esse erro esteja diretamente associado à complexidade do caminho e à velocidade do robô.

Figura 37 – Trajetória e velocidade do robô.



Fonte: (CARMO et al., 2019).

Para entender o impacto do FPS no erro de trajetória, considerando um tempo fixo por volta de 25 segundos, realizamos dois experimentos. Para o primeiro, a taxa de quadros da câmera foi definida como 2,5 FPS, o que significa que a janela de prazo para os comandos chegarem ao robô, ou seja, o requisito de tempo de resposta, é de $400 \text{ ms} = (1/2,5)$. Então, para o segundo caso, a taxa de quadros da câmera foi aumentada para 10 FPS, estreitando a janela do prazo para $100 \text{ ms} = (1/10)$.

Como pode ser visto na Figura 38a, para 2,5 FPS, o robô tenta seguir a trajetória desejada, não executando corretamente as velocidades esperadas para cobrir todo o caminho em 25 segundos. O erro médio para essa trajetória é de cerca de 100 mm, chegando a 330 mm na parte mais difícil (Figura 38b).

No entanto, quando a taxa de quadros é alterada para 10 FPS, o robô executa a trajetória sem problemas. O erro médio é inferior a 80 mm, mesmo na parte mais difícil da trajetória. Tanto a trajetória como o erro são mostrados respectivamente nas Figuras 38c e 38d.

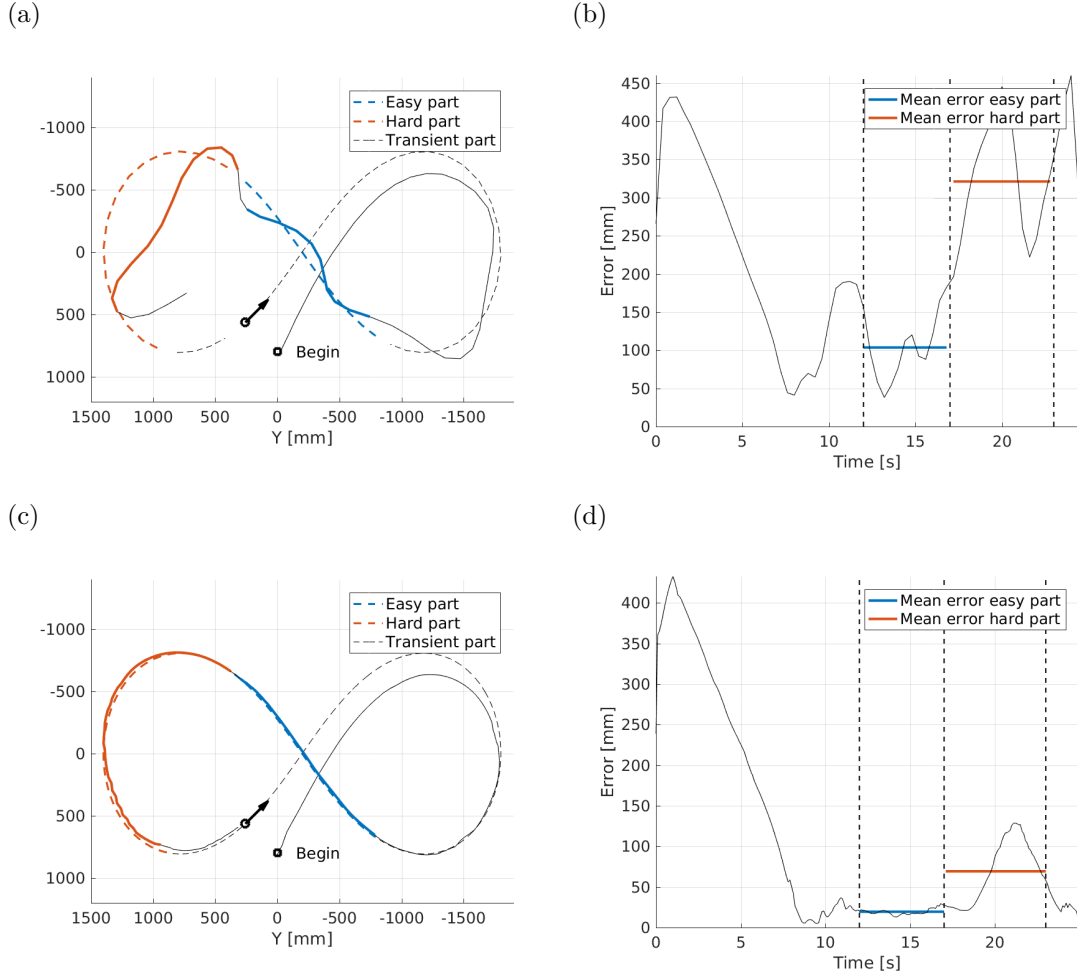
Claramente, quanto maior o FPS, menor o erro de trajetória¹. O aumento do FPS permitiu uma melhoria na precisão em relação as partes complexa e simples da trajetórias, reduzindo os erros médios de 330 para 80 mm e de 100 para 15 mm, respectivamente. Vale ressaltar que a faixa de melhoria está associada ao tempo adotado por volta, 25 segundos por volta neste caso. Além disso, com um FPS mais alto, uma amostragem mais fina é aplicada e, portanto, é obtido um melhor feedback visual.

Portanto, para atender a um requisito máximo de erro de trajetória, o aplicação precisaria alterar o FPS. No entanto, o desafio aqui é que aumentar o FPS (2,5 a 10 FPS) implica em uma redução proporcional para o tempo de resposta necessário (400 ms a 100

¹ Um vídeo de uma das experiências pode ser encontrado na seguinte URL: <https://youtu.be/5r6p0ucESTw>.

ms), pois o período entre quadros determina a janela de tempo para o controle de feedback visual.

Figura 38 – Trajetória realizada e o erro para 2.5 FPS (a,b) e 10 FPS (c,d).



Fonte: (CARMO et al., 2019).

Como atender o prazo da janela de tempo do loop de controle?

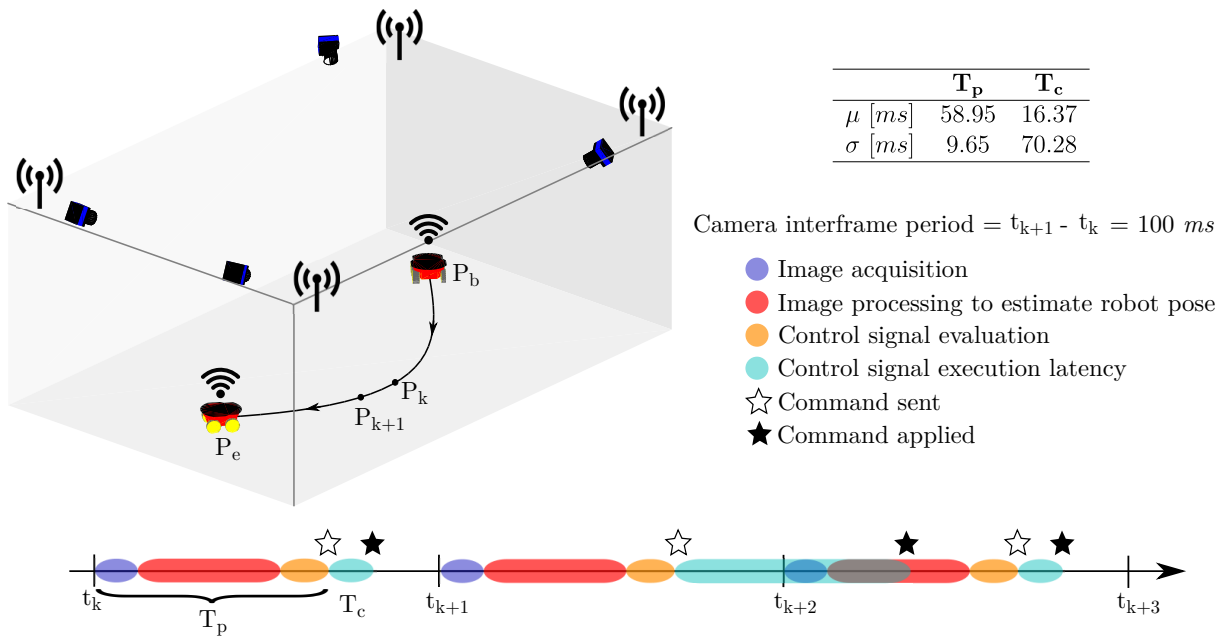
Um dos desafios a serem enfrentados pelo PIS é que existem muitos elementos que podem impor gargalos em termos de taxa de transferência e latência à tarefa de controlar um robô. Assim, os indicadores de desempenho clássicos da rede também são métricas importantes a serem avaliadas juntamente com o erro de localização discutido anteriormente.

Para esta aplicação, como mostrado na Figura 39, o período entre quadros da câmera define uma janela de tempo para todo o ciclo de controle. Basicamente, todas as etapas devem ocorrer dentro desse intervalo, por exemplo $t_{k+1} - t_k = 100$ ms, definido pela taxa de quadros de 10 FPS. Em outras palavras, depois que um quadro de imagem é capturado, ele deve ser enviado para a infraestrutura em nuvem, os serviços de localização

e controle devem ser executados e o sinal de controle deve chegar ao robô antes que o próximo quadro de imagem seja capturado. Caso contrário, se o comando de controle não chegar ao robô a tempo, as informações desatualizadas não serão mais precisas, fazendo com que o robô se desvie da trajetória planejada, o que pode levar à instabilidade geral do robô ou colisão em casos extremos.

A Figura 39 ilustra a janela de tempo em que o loop de controle do robô deve ocorrer. Para esse experimento, essa janela de tempo corresponde ao tempo de resposta da tarefa de controle dessa aplicação. Os comandos resultantes do loop de controle atingem o robô dependendo de: i) T_p , o tempo de processamento; mais ii) T_c , o tempo de comunicação. Os valores desses intervalos de tempo correspondem à média das medições obtidas durante as experiências apresentadas na próxima seção. O intervalo T_p inclui o tempo necessário para realizar a aquisição da imagem, o processamento da imagem para obter a estimativa da pose do robô e a geração dos comandos de controle. Por sua vez, T_c representa o tempo para transmitir o sinal de controle ao robô. Observe que T_c inclui o tempo para atravessar várias redes (rede DC, rede com e sem fio), mas a conexão sem fio é o componente mais dominante do tempo de comunicação.

Figura 39 – Ilustração das medidas de tempo durante o loop de controle do robô.



Fonte: (CARMO et al., 2019).

Como forma de atingir o menor valor possível para T_p nos testes, realizou-se a orquestração dos serviços que compõem o aplicativo de maneira semelhante à (PICORETI et al., 2018). Nesse caso, o orquestrador observa a utilização da CPU do serviço com maior tempo consumido no loop de controle e executa um escalamento horizontal desse serviço. O processo de escalamento horizontal consiste em monitorar a utilização da CPU como

uma métrica para o serviço Aruco Detector. Quando essa métrica excede um limite, o orquestrador inicia novas instâncias desse serviço até o máximo permitido pelo aplicativo. Dessa forma, a carga de processamento por instância diminui e, conseqüentemente, T_p diminui.

Outra ação executada pelo orquestrador é fixar as CPUs virtuais que executam o serviço Aruco Detector para garantir o tempo de processamento da CPU alocado para esse processo, a idéia é que T_p não mude muito ($\sigma_{T_p} = 9,65$ ms). Em relação ao impacto de T_p e T_c no atendimento ao requisito do tempo de resposta, apesar do tempo médio de comunicação $T_c = 16,37$ ms ser substancialmente menor em comparação ao tempo de processamento $T_p = 58,95$ ms, a latência de comunicação T_c pode variar significativamente ($\sigma_{T_c} = 70,28$ ms), principalmente dependendo de quantos dispositivos compartilham a mesma rede sem fio.

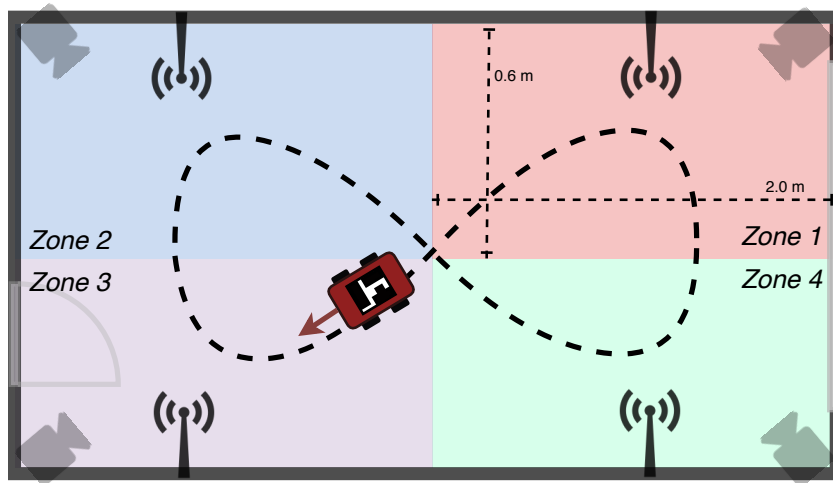
Em um caso ideal, T_p e T_c cabem dentro da janela de prazo, conforme ilustrado no primeiro intervalo de tempo $[t_k, t_{k+1}]$. Mas se T_c aumentar demais, como no intervalo de tempo $[t_{k+1}, t_{k+2}]$, os comandos enviados não serão aplicados a tempo, ou seja, antes de que um novo quadro seja recebido. Nesse caso, o robô manterá o último comando de controle por muito mais que um intervalo de tempo, fazendo com que o robô se desvie da trajetória desejada e, portanto, aumente o erro geral. Se esses prazos perdidos ocorrerem raramente, não haverá impacto na execução da trajetória. Mas se isso acontecer com frequência, eles potencialmente prejudicarão a capacidade de controlar adequadamente o robô.

Portanto, fica claro que as especificações de rede de baixa latência devem ser aprimoradas, uma vez que são necessárias taxas de FPS mais altas para lidar com robôs móveis mais rápidos. Mais importante, à medida que o número de robôs no PIS aumenta, aumenta o compartilhamento da rede sem fio pelos robôs e por qualquer outro dispositivo sem fio. Dado que a rede sem fio atual é composta por apenas um ponto de acesso WiFi, funcionando como uma grande célula que cobre toda a área do PIS, essa variabilidade de T_c seria afetada crucialmente. Este é o tópico abordado por nossa abordagem de célula sem fio no próximo experimento.

5.5.3 Entrega de pacotes para attocélulas com handover eficiente

No contexto de ambientes indoor industriais, diferentes dispositivos móveis, como robôs industriais, estarão em movimento em todo o espaço de produção. Portanto, eles nem sempre podem ser atendidos com qualidade pela mesma estação base ou ponto de acesso devido a restrições da área de cobertura ou concorrência entre os clientes de uma mesma área, a depender do seu número. Para esse experimento foram implantadas quatro pequenas células (chamadas attocélulas) para cobrir toda a área útil do PIS de aproximadamente $4,8$ m², na qual o robô pode executar suas tarefas de navegação.

Figura 40 – Cobertura das áreas das células no PIS da UFES.



Fonte: (CARMO et al., 2019).

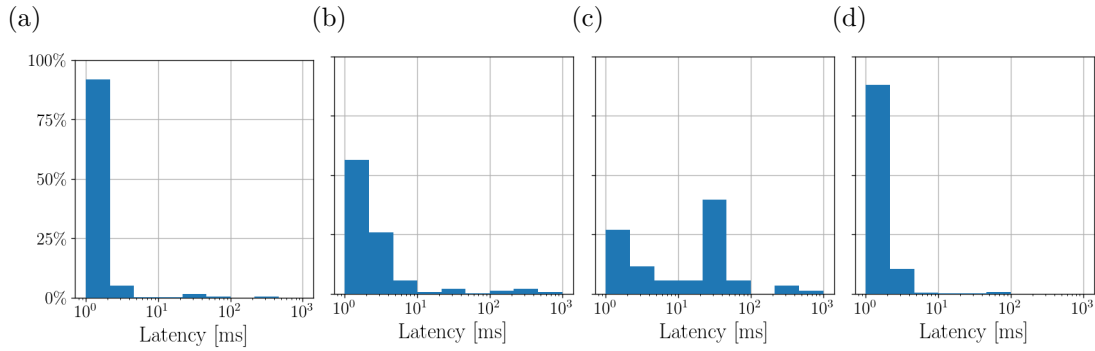
O objetivo deste experimento é mostrar o efeito do compartilhamento de uma célula wifi no controle do robô. Além disso, será mostrada uma análise do desempenho do sistema durante o processo de entrega, observando o tempo de comunicação T_c entre a nuvem e o robô. Conforme mostrado na Figura 40, o robô se move por diferentes zonas durante sua trajetória, cada uma delas atendida por uma respectiva attocélula, servindo a $zone_i$. Durante a transição do robô de uma área para outra, o orquestrador usa o serviço de localização como acionador do processo de entrega, toda vez que o robô entra em uma nova zona.

Os experimentos são conduzidos no WiFi operando na banda de 5 GHz. O objetivo é avaliar a taxa de transferência (entrega de pacotes) do nosso sistema de handover eficiente para attocélulas, sustentada pelas informações de localização fornecidas pelo serviço de visão computacional. Para todos os cenários de experimentação, o FPS é definido como 10 (janela do prazo final = 100 ms) e a velocidade do robô é definida como 25 segundos por volta. Para testar nossa entrega, executamos quatro cenários diferentes. O primeiro cenário **a** será utilizada como referência onde há apenas uma célula WiFi cobrindo toda a área do PIS, e consequentemente não ocorre o handover entre células. Então, para o cenário **b** e **c**, variamos o número de clientes que competem pela célula WiFi sem transferência, medindo o tempo de comunicação T_c . Finalmente, o cenário **d** lida com nossa abordagem de handover eficiente para attocélulas.

A Figura 41 mostra o histograma do tempo de comunicação para todos os cenários. O cenário **a** (Figura 41a) é a referência que será utilizada como base para comparação, pois o robô é o único cliente conectado a apenas uma célula WiFi que cobre toda a área e não há compartilhamento nessa rede. Na verdade, esse é um caso altamente improvável, porque a rede sem fio geralmente é compartilhada por muitos dispositivos, principalmente

no contexto da Indústria 4.0. Mesmo assim, observe que cerca de 90 % dos pacotes são entregues dentro de alguns milissegundos.

Figura 41 – Tempo de comunicação (T_c): (a) 0 clientes, (b) 5 clientes, (c) 10 clientes and (d) PIS handover



Fonte: (CARMO et al., 2019).

As medidas de latência mostradas nas Figuras 41b e 41c representam cenários mais realistas nos quais há uma carga de trabalho de 5 clientes gerando tráfego simultâneo ao robô a 10 Mbps cada um na Figura 41b e 10 clientes na Figura 41c. Como pode ser visto, em ambos os casos a latência aumentou, mas aumentou drasticamente no caso de 10 clientes, ficando mais próxima de 100 ms para 40% das amostras. Esse resultado tem um impacto direto na frequência com que o sistema ultrapassa o requisito de tempo de resposta (15,9% de acordo com a Tabela 7) e é crucial no erro de trajetória (consulte a Figura 42), que pode ser 5 vezes pior do que nos outros casos.

Tabela 7 – Percentual do número de vezes que o sistema ultrapassa o requisito de tempo de resposta.

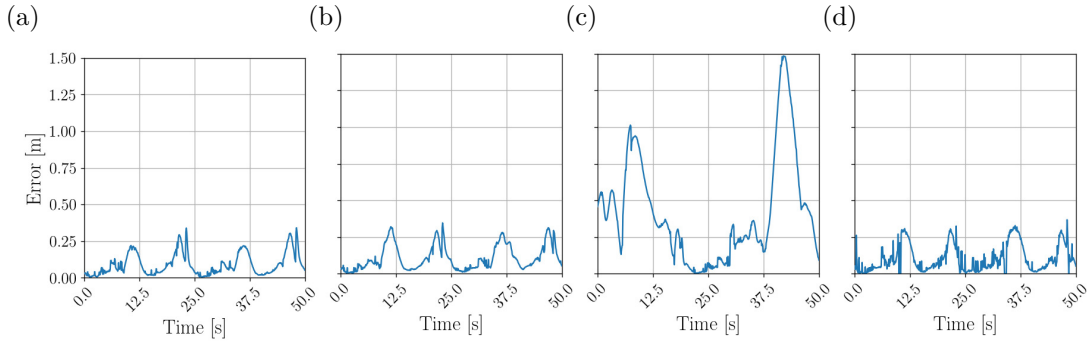
Scenario	0 client (a)	5 clients (c)	10 clients (c)	PIS handover (d)
Exceeded count (%)	0.81	4.43	15.97	1.38

Fonte: (CARMO et al., 2019).

O resultado pode ser observado na Figura 41d, que mostra o histograma do tempo de comunicação para o caso da solução de handover eficiente do PIS. Comparando com os casos anteriores, podemos ver que o tempo de comunicação obtido é muito próximo do primeiro caso de referência, embora haja 10 clientes simultâneos gerando tráfego. Isso é explicado pela integração entre a visão computacional e o controlador SDN que guia o serviço Handover Orchestrator do PIS. Quando o robô se move de uma zona para outra, a pose do robô é calculada pelo serviço de localização, e o orquestrador altera a célula que serve o robô. Apesar de todo o tráfego simultâneo ser transmitido na mesma rede sem fio, eles são atendidos pelas outras attocélulas fora da zona atual. Portanto, a abordagem de handover para attocélulas do PIS se mostrou viável para prover uma comunicação

móvel atendendo ao requisito de entrega de pacotes dentro da janela de tempo do loop de controle, com o tempo de comunicação e erro de trajetória semelhante à referência (consulte Figura 42d).

Figura 42 – Erro de Trajetória: (a) 0 clientes, (b) 5 clientes, (c) 10 clientes e (d) PIS handover



Fonte: (CARMO et al., 2019).

5.6 Análise dos Resultados

As próximas aplicações para a Indústria 4.0 trarão a necessidade de comunicação rápida com dispositivos móveis para serem usados no ambiente industrial. Esse requisito impõe baixa latência e o uso de pequenas células sem fio, como uma maneira de reduzir a densidade de dados por área e obter execução em tempo real. Embora células menores possam causar um aumento na frequência de handover, se um método de localização preciso for aplicado para rastrear os dispositivos móveis, as restrições para o tempo de comunicação podem ser atendidas e as aplicações em tempo real podem ser implantados com sucesso.

Nesse contexto, este estudo de caso propõe o uso do PIS como forma de fornecer attocélulas baseadas em visão computacional que permitem a implantação de aplicações com feedback em tempo real. Esse tipo de aplicação possui requisitos específicos rígidos para sua operação completa e um método de localização preciso pode ajudar a atender a esses requisitos. O PIS usado como ambiente de teste fornece um serviço de localização de visão computacional, baseado em uma rede multicâmeras, que permite o uso de attocélulas e uma arquitetura programável.

Para isso foi desenvolvida uma aplicação para robôs móveis cujo objetivo é controlar a trajetória do robô considerando um tempo fixo de 25 segundos por volta. Para executar esta tarefa, é executado um loop de controle em tempo real com base em um feedback visual. Esta aplicação permitiu abordar muitos aspectos envolvidos, como localização, tempo de comunicação e largura de banda.

Geralmente, um sistema baseado em visão requer uma grande quantidade de recursos de infraestrutura e afeta diretamente o tempo de resposta da aplicação. Nesta aplicação, para controlar o robô dentro de um erro de trajetória aceitável, experimentos preliminares mostram que o comando de controle deve chegar dentro de uma janela de tempo, definida pelo FPS da câmera.

Nos primeiros experimentos, pode-se observar que é possível controlar o robô usando WiFi sem compartilhar o canal com outros dispositivos. No entanto, este é um cenário improvável, especialmente no contexto da Indústria 4.0, onde o aumento no uso de dispositivos sem fio é exatamente o que se busca. Nos testes a seguir, pode-se observar que aumentando o número de conexões simultâneas usando o mesmo canal do robô, o tempo de comunicação T_c também aumenta até que o controle do robô se torne inviável.

Assim, uma alternativa é diminuir o tamanho da célula, criando as chamadas attocélulas. Suas áreas são reduzidas até que normalmente apenas um dispositivo seja coberto por cada célula, o que acabaria por impedir o compartilhamento de canais com outros dispositivos ou, pelo menos, não com muitos deles. O problema com essa abordagem é que o uso de attocélulas aumentaria a necessidade do número de vezes que seria necessário a realização de handover entre células. Como o tempo de médio de handover é extremamente alto (aproximadamente 4s) (ZEHL; ZUBOW; WOLISZ, 2016), o controle do robô seria inviável.

Através dos últimos experimentos, mostramos que, se o sistema de localização visual do PIS for usado, é possível obter um cenário viável em que T_c possa ser semelhante ao caso improvável de usar uma única célula sem compartilhamento. O uso do PIS permite a criação de attocélulas e também uma arquitetura sem fio programável para fornecer um handover mais eficiente.

Portanto, esse estudo de caso demonstra a viabilidade do uso do serviço de visão computacional baseada no PIS como sistema de localização, um dos requisitos necessários para aplicações indoor de baixa latência, no contexto da Indústria 4.0. O tempo de comunicação da rede sem fio pode ser tratada no PIS: alterando os parâmetros da camada física para reduzir o tempo de transmissão e o tempo de entrega com base no sistema de localização. O handover geralmente se refere à tarefa de conectar o objeto móvel que estava em uma célula à uma célula vizinha de forma a manter a conectividade do robô. Conhecendo a localização do robô, um controlador SDN pode definir os canais sem fio de maneira proativa para manter essa conectividade sempre que o robô sair de sua attocélula. Isso evita a dependência de ações reativas baseadas, por exemplo, na força do sinal RF.

6 Estudo de Caso 2: Reconhecimento de Gestos

Uma importante função planejada para um Espaço Inteligente, é proporcionar meios de interagir com seus usuários. Para isso, seus sensores, atuadores e serviços devem ser escolhidos e projetados para incluírem essa capacidade. Nesse caso, um conjunto de recursos fundamentais deve ser considerado, tais como a definição de formas de interação, interfaces adequadas (Vega-Barbas et al., 2018) e métodos de reconhecimento de gestos e/ou ações (ZHANG et al., 2019; MITRA; ACHARYA, 2007). É por meio desses recursos e uma infraestrutura de hardware e software, que dê suporte a sua realização, que se pode agregar a devida interatividade ao espaço inteligente.

Dentre as interfaces de interação, os gestos dinâmicos são considerados uma das mais intuitivas (SANTOS, 2016). Um gesto dinâmico pode ser definido como um conjunto de pequenos movimentos, comumente realizados pelos braços, mas não restritos a eles, que visam a comunicação entre indivíduos. Um gesto pode ser compreendido como sendo a evolução da pose (configuração das juntas) de um indivíduo em um intervalo de tempo. Sendo assim, um sistema automático que objetive o reconhecimento de gestos deve considerar essa definição e utilizar aqueles sensores que melhor capture esses tipos de características.

Desde o advento do *Microsoft Kinect 360*, em meados de 2011, o reconhecimento de gestos dinâmicos pode ser executado de maneira mais efetiva. Isso é devido ao fato de que o sensor de profundidade presente no kinect possibilita a extração dos esqueletos de pessoas presentes em seu campo visual. Dessa forma, é possível utilizar modelos relativamente simples para reconhecer gestos dinâmicos, uma vez que o esqueleto fornece uma representação compacta do movimento corporal de uma pessoa. O problema com o uso desse tipo de sensor é que mesmo sendo de fácil aquisição e com um custo acessível, a quase totalidade dos ambientes interacionais onde um reconhecedor de gestos pode ser utilizado possuem câmeras RGB (do inglês Red, Green, Blue) como principais sensores. Com isso, modelos dependentes do kinect possuem uma restrição natural de uso, uma vez que tais ambientes com câmeras convencionais têm apenas as imagens RGB como a principal fonte de informação para o reconhecimento de gestos.

Adaptando a definição anterior para a área de visão computacional, um gesto dinâmico pode ser entendido como a evolução da pose de uma pessoa em uma sequência de imagens. Dessa maneira, um modelo que pretende resolver esse problema enfrenta diferentes desafios, tanto na extração de características espaciais (obtenção da pose e/ou movimento em cada imagem), quanto na análise da evolução de cada característica (relacionar as

características temporalmente). Isso levanta uma barreira nas pesquisas de reconhecimento de gestos em vídeos, e direciona esforços para modelos baseados em esqueletos.

O problema com o reconhecimento de gestos dinâmicos é ainda maior dentro do escopo dos Espaços Inteligentes multicâmeras. Mesmo que as várias câmeras possam ajudar na diminuição de características oclusas, elas aumentam significativamente a necessidade de processamento do classificador e ainda geram um problema referente à decisão de que câmera deve ser utilizada no reconhecimento.

Em uma situação de reconhecimento *online* esses problemas são ainda mais presentes, uma vez que não se tem a informação do tamanho de cada sequência que representa um gesto. Além disso, um mesmo gesto pode ser realizado com mais ou menos imagens a depender do usuário. Ou seja, por um lado mais câmeras aumenta a área monitorada pelo espaço ao mesmo tempo que reduz os problemas com oclusão, por outro lado, quanto mais câmeras, mais complexo pode se tornar o processo de reconhecimento de gestos. Sendo assim, objetivando o uso de ambientes multicâmeras para a interação natural, (QUEIROZ et al., 2018) propôs um método que usa as imagens de N câmeras de um espaço inteligente calibrado para reconstruir o esqueleto tridimensional de cada indivíduo presente no mesmo. Com essa solução, é possível utilizar o potencial dos reconhecedores de gestos baseados no Kinect mediante múltiplas câmeras.

Mesmo com a possibilidade de utilização de esqueletos 3D, alguns trabalhos utilizam modelos baseados em sequências fixas de imagens (ESCOBEDO-CARDENAS; CAMARA-CHAVEZ, 2015; LIU et al., 2019), o que acaba gerando uma dependência de como os gestos devem ser executados, e por conseguinte limita a sua naturalidade de execução. Um problema similar ocorre com as abordagens baseadas em regras (SANTOS; NETO; SALEME, 2015), que apresentam uma forte dependência da percepção do projetista. Uma ligeira mudança na realização de um gesto pode prejudicar o seu reconhecimento. Sendo assim, uma abordagem baseada em um modelo de aprendizado automático, orientado à tarefa de classificação de gestos e que não limitasse o tamanho da sequência de entrada, superaria as restrições dos modelos de sequências fixas e os baseados em regras.

Dessa maneira, este estudo de caso utiliza um reconhecedor de gestos *online* que possa ser utilizado em um espaço interacional multicâmera. A proposta consiste em utilizar um serviço baseado no método apresentado em (QUEIROZ et al., 2018) para extrair o esqueleto 3D de um usuário presente na cena, fornecer o esqueleto para um modelo que o reconhecerá como sendo gesto ou não-gesto. Isto possibilitará a segmentação temporal de sequências contendo gestos, as quais deverão alimentar um outro modelo responsável por classificá-las em um dos gestos possíveis. O reconhecimento e a classificação dos gestos foram implementados em um único serviço dentro do Espaço Inteligente.

Porém, para validar a adequação do serviço de reconhecimento e classificação de gestos à uma aplicação do Espaço Inteligente, é necessário mensurar o seu desempenho

e verificar se esse desempenho atende aos requisitos desta aplicação. O desempenho do serviço pode ser medido através de três fatores: a taxa de reconhecimento dos gestos, a taxa de classificação correta dos gestos e o tempo de execução dos serviços.

A taxa de reconhecimento dos gestos mede o percentual de gestos reconhecidos em relação ao número total de gestos executados. Quanto maior a taxa melhor o desempenho do reconhecedor. Já para avaliar a taxa de classificação correta dos gestos, é considerado que o gesto foi reconhecido e sua classificação é realizada entre um conjunto de gestos conhecidos. O percentual de gestos corretamente classificados em relação ao total de gestos, corresponde a taxa de classificação.

Por fim, o tempo de processamento dos serviços de reconhecimento e classificação de gestos é um fator importante que pode determinar a viabilidade de sua utilização em uma aplicação. Principalmente em um ambiente interacional onde as aplicações possuem como requisito o funcionamento em tempo real.

Existe uma relação direta entre o desempenho de um sistema de reconhecimento de gestos e o número de imagens geradas para processamento. O processo de reconhecimento de gestos precisa de imagens de pelo menos duas fontes diferentes capturadas dentro da mesma janela de tempo. Porém, as taxas podem ser melhoradas caso haja um número maior de câmeras. Por outro lado, o não processamento de imagens, seja por falta de capacidade de processamento ou a perda dessas imagens em sua transmissão, reduz o desempenho do reconhecedor. De forma similar, o desempenho do classificador de gestos é fortemente impactado pelo número de imagens utilizadas dentro de um período de tempo. Logo, alterações da taxa de FPS influenciam diretamente na taxa de classificação. Quanto maior a taxa de FPS melhor é a classificação dos gestos e vice versa.

Essas relações se mostram o desafio de se utilizar reconhecimento e classificação de gestos baseado em uma rede multi câmeras do tipo RGB. Quanto maior o desempenho que se busca da aplicação, maior é quantidade de recursos necessária. Um aumento do número de imagens, seja pelo aumento do número de câmeras ou pelo aumento da taxa de FPS, impactam diretamente na capacidade de transmissão da rede. Da mesma forma, esse aumento do número de imagens exige um aumento proporcional da capacidade de processamento da infraestrutura.

Dessa forma, aplicações que utilizam o reconhecimento e classificação de gestos tem o desempenho como um de seus requisitos. Tal como as demais aplicações da classe de visão computacional apresentadas nesse trabalho, esse é um requisito específico e altamente exigente em relação aos recursos da infraestrutura. Para atender a esses requisitos e ao mesmo tempo fazer um uso racional dos recursos disponíveis o PIS se mostra como uma alternativa viável a implantação de um ambiente interacional baseado em gestos.

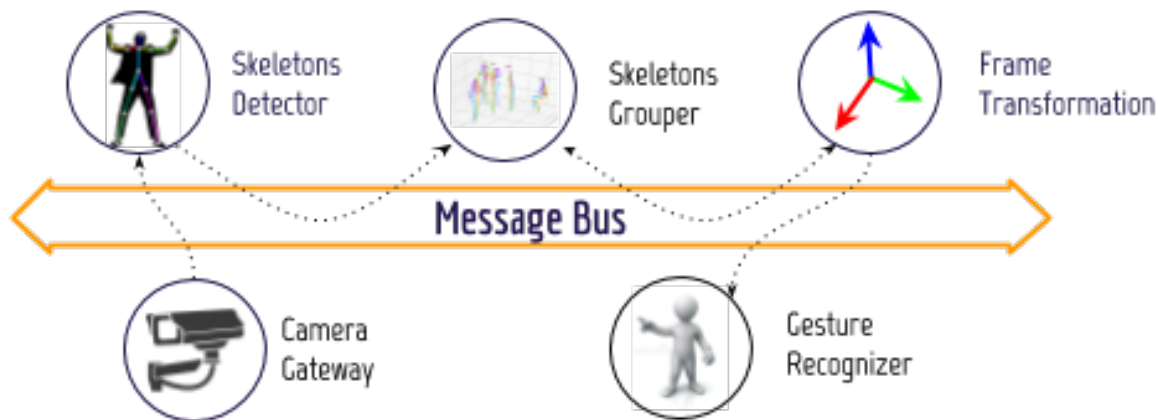
O que será demonstrado com esse estudo de caso:

- A orquestração hierárquica multinível centrada na observabilidade dos níveis de aplicação e infraestrutura do PIS, permitem que os múltiplos requisitos específicos das aplicações sejam atendidos de forma simultânea.
- A utilização racional dos recursos da infraestrutura é conseguida através do modelo de orquestração empregado no PIS. Nesse modelo, os recursos são fornecidos ou liberados somente na medida necessária para atender aos requisitos rigorosos da aplicação.

6.1 Serviços e sua interação

Como exemplo para um ambiente interacional, será utilizada uma aplicação composta por diferentes serviços com o objetivo de se identificar a execução de gestos por pessoas dentro do PIS. Os principais serviços que compõem a aplicação estão mostrados na figura 43.

Figura 43 – Cadeia de Serviços que compõe a aplicação.



Fonte: Produção do próprio autor.

- **Camera Gateway (CG):** o serviço de camera gateway é o mesmo utilizado nas diferentes aplicações já descritas anteriormente nesse trabalho.
- **Skeletons Detector (SD):** O serviço de detecção de esqueletos foi desenvolvido com base em (QUEIROZ et al., 2018), onde se realiza a detecção 2D das juntas de esqueletos presentes em uma imagem. Cada esqueleto é constituído por um conjunto de até 18 juntas representadas como coordenadas (u, v) na imagem da câmera na qual foram detectadas. Tal serviço é baseado no detector OpenPose (WEI et al., 2016) e é capaz de detectar múltiplos esqueletos na mesma imagem.

Considerando um espaço inteligente com N câmeras, a extração de esqueletos utilizada será a abordada em (QUEIROZ et al., 2018), como mencionado anteriormente. Dessa forma, a cada instante de tempo as imagens capturas pelas N câmeras são passadas para o reconstrutor de esqueletos que fornece um esqueleto 3D para cada usuário presente na cena. Cada coordenada 3D das juntas está no sistema de coordenadas no qual as câmeras foram calibradas.

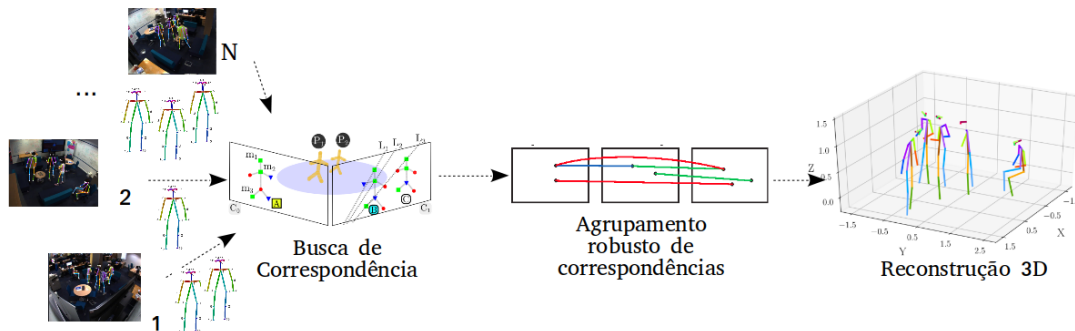
- **Frame Transformation (FT):** Em uma rede de câmeras, cada uma possui um referencial em relação ao referencial global do ambiente. Logo, um serviço para realizar a conversão de coordenadas entre referenciais se faz necessário para que as informações adquiridas, em cada uma das câmeras, possam ser representadas em um referencial global comum.
- **Skeletons Grouper (SG):** O serviço de reconstrução 3D das juntas também foi desenvolvido com base em (QUEIROZ et al., 2018) e utiliza os esqueletos detectados pelo serviço Skeletons Detector. A cada instante de tempo, os esqueletos detectados numa mesma imagem recebem um identificador único, relacionado à câmera na qual foram identificados. Note que, para que a reconstrução 3D seja possível, as imagens capturadas devem pertencer a um sistema de câmeras calibrado.

O serviço possui basicamente três etapas: (1) busca de correspondências, (2) agrupamento robusto de correspondências e (3) reconstrução 3D das juntas (vide Figura 44). Inicialmente, todas as imagens são analisadas duas a duas, usando-se geometria epipolar, para se encontrar as possíveis correspondências entre as juntas detectadas. Isso gera uma lista de correspondências, a qual é passada para a segunda etapa, onde o intuito é percorrer e verificar essa lista, eliminando as correspondências errôneas e agrupando todas aquelas associadas à mesma junta. Finalmente, a partir das correspondências consideradas corretas, realiza-se a reconstrução 3D através de um sistema de equações, que relaciona as coordenadas das juntas correspondentes.

- **Gesture Recogniser (GR):** Em uma aplicação online, as imagens são fornecidas ao modelo uma a uma, seguindo a ordem de captura. Assim, para que o serviço possa identificar uma sequência de imagens como pertencente a uma das classes de gestos, é necessário que duas etapas sejam realizadas: o reconhecimento do gesto e sua classificação.

A etapa de reconhecimento tem por objetivo determinar quando um possível gesto inicia e termina. Para isso, o serviço recebe como entrada uma sequência de esqueletos 3D reconstruídos e extraem deles características específicas, como angulações, velocidades e acelerações, a fim de melhorar a representação do movimento das juntas. Considerando a dependência temporal dos esqueletos entre o momento atual e nos

Figura 44 – Fluxo de reconstrução 3D do esqueleto.



Fonte: Produção do próprio autor.

instantes de tempo anteriores, busca-se reconhecer o esqueleto como pertencente a um gesto ou a um não-gesto.

A segunda etapa do serviço tem por objetivo classificar o gesto como pertencente a uma das possíveis classes de gestos. Para isso utiliza a saída do reconhecedor para filtrar quais os esqueletos 3D reconstruídos que serão utilizados para a classificação.

6.2 Experimentos

Na seção anterior foi apresentada uma aplicação que possui em sua composição diferentes serviços utilizados para reconhecimento e classificação de gestos em um ambiente interacional. Para o seu funcionamento adequado, a aplicação possui requisitos de desempenho que precisam ser cumpridos. Nesse caso os requisitos são taxa de reconhecimento de gestos e o valor de incerteza de sua classificação.

Nessa seção serão realizados diferentes experimentos que mostram a relação entre o desempenho da aplicação e os recursos disponibilizados. O objetivo é demonstrar que o adequado compartilhamento de informações entre os níveis de aplicação e infraestrutura possibilita que recursos sejam disponibilizados sempre que necessário para atendimento aos requisitos da aplicação. E de forma similar, recursos devem ser liberados sempre que não forem mais necessários.

Foram realizados 5 experimentos divididos em 3 partes. Na primeira parte serão mostrados 2 experimentos que medem as taxas de desempenho da aplicação. Na segunda parte, os experimentos 3 e 4 relacionarão as taxas de desempenho com a alocação de recursos. Por último será mostrado o experimento 5, onde os diferentes mecanismos de orquestração foram implementados de forma integrada, de forma a atender ao requisito da aplicação e realizar o controle racional dos recursos da infraestrutura.

Estrutura do experimento

Os experimentos desse estudo de caso foram realizados no PIS implementado na UFES. Já o PIS do IFES de Vitória foi utilizado para captura de imagens para treinamento dos modelos utilizados no serviço Gesture Recognizer.

Todos as imagens foram gravadas de forma a possibilitar que o experimento possa ser replicado e os resultados fossem determinísticos. Para se utilizar as imagens gravadas pelas câmeras, foi desenvolvido um novo serviço de camera gateway que realiza a aquisição dessas imagens e as disponibiliza da mesma forma como o serviço de camera gateway original. Todos os demais serviços do Espaço Inteligente não sofreram nenhuma alteração e foram utilizados independentemente do serviço de gateway utilizado. É importante frisar que todas as imagens utilizadas no experimento foram geradas pelo próprio PIS e não alteram a estrutura da aplicação, sendo a gravação e leitura dessas imagens uma capacidade a mais adicionada ao ambiente.

Conjunto de dados

O conjunto de dados utilizado neste trabalho é composto por 2.400 gestos distribuídos entre 15 classes distintas. Cada vídeo corresponde a um gesto realizado oito vezes por um mesmo voluntário. Como o espaço utilizado possui 4 câmeras, os voluntários foram instruídos a realizar cada um dos 15 gestos 8 vezes: de frente para cada câmera e de frente para um ponto entre elas. No total, foram realizadas 20 aquisições de 20 voluntários distintos em dois espaços inteligentes diferentes. No primeiro espaço inteligente do IFES de Vitória, foram realizadas 18 aquisições (2.160 gestos), as quais serão utilizadas para o treinamento dos modelos propostos. No segundo espaço, localizado na Universidade Federal do Espírito Santo, campus Goiabeiras, (UFES - Lab Viro), foram realizadas mais duas aquisições (240 gestos), as quais serão utilizadas como conjunto de teste. Usar um conjunto de testes capturado em um outro espaço ajudará na validação da capacidade de generalização da proposta. Deve-se salientar que, antes de cada aquisição de dados, os voluntários assistiram a um vídeo demonstrativo de como cada gesto deveria ser executado. No entanto, mesmo executando um pouco diferente do indicado, as aquisições não foram interrompidas.

Todos os vídeos capturados foram rotulados de maneira que cada *frame* contenha o rótulo do gesto ao qual ele pertence: não-gesto (classe 0) ou gesto (classes de 1 a 15). Uma lista do vocabulário de gestos presentes no conjunto de dados é apresentada na Tabela 8.

Pré-processamento dos dados

As imagens capturadas pelas quatro câmeras num mesmo instante de tempo passaram pelo processo de extração de esqueleto realizado pelo serviço de Skeletons

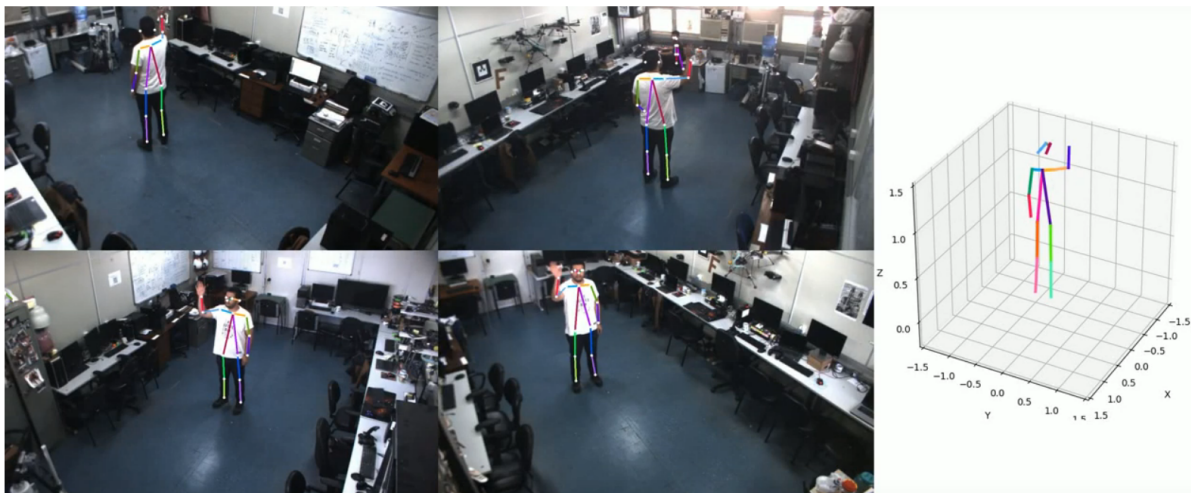
Tabela 8 – Lista dos gestos presentes no conjunto de dados utilizado. A classe 0 corresponde a um não-gesto, ou seja, quando nenhum gesto está sendo executado.

Classe	Gesto	Classe	Gesto
0	Não-gesto	8	Não (permissão)
1	Pedido de ajuda	9	Ruim (<i>feedback</i>)
2	Venha aqui	10	Dar passagem
3	Pode sair	11	Apontar
4	Siga-me	12	Dúvida
5	Pare	13	Mais alto (volume)
6	Abortar (missão/tarefa)	14	Mais baixo (volume)
7	Bom (<i>feedback</i>)	15	Silêncio

Fonte: Produção do próprio autor.

Detector e reconstruídos pelo serviço de Skeletons Grouper. Sendo assim, foram gerados 300 sequências de esqueletos 3D correspondentes aos 2.400 gestos. Considerando os ruídos que podem ser gerados na etapa de extração, foi aplicada uma rotina de supressão de não máximos a fim de eliminar os esqueletos com juntas menos prováveis. Na Figura 45, pode ser vista a reconstrução de um esqueleto tridimensional a partir das imagens e dos esqueletos bidimensionais nelas detectados.

Figura 45 – Exemplo da reconstrução de um esqueleto 3D a partir de quatro imagens adquiridas no espaço inteligente da Ufes.



Fonte: Produção do próprio autor.

Como o modelo de esqueleto do *openpose* não possui a junta que representa o centro do torso (tratada aqui apenas como junta do torso), esta foi criada como sendo o ponto central do triângulo formado pelas juntas do tórax e dos quadris. Dessa forma, cada esqueleto foi centralizado na junta do torso e tiveram os comprimentos entre juntas normalizados de maneira a terem norma unitária. Em seguida, como as juntas dos membros inferiores comumente não trazem informações relevantes para a diferenciação dos gestos utilizados, optou-se por selecionar apenas sete juntas correspondente aos punhos, cotovelos,

ombros e cabeça.

Treinamento

Os dois modelos propostos, o reconhecedor e o classificador de gestos, foram treinados separadamente. Para isso, foi necessário utilizar todo o conjunto de treino, de forma que: (i) para o treinamento do reconhecedor (classificador binário), as observações com rótulo 0 são consideradas não-gestos e as com rótulo maior que 0 são assumidas como gestos; (ii) para o treinamento do classificador, são utilizadas apenas as sequências segmentadas (aquelas que contêm gestos) e os correspondentes rótulos de classe.

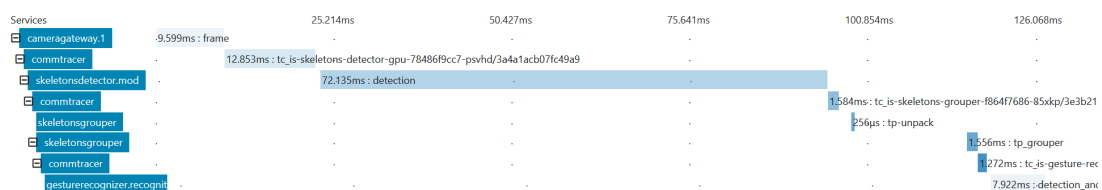
Para a predição, utilizaram-se as sequências completas do conjunto de teste, possuindo gestos e não-gestos. Desse modo, sempre que o modelo do reconhecedor marcava uma observação como sendo gesto, as próximas observações eram armazenadas até que a primeira predição de um não-gesto ocorresse. Nesse momento, a sequência armazenada era passada para o modelo de classificação que determinava a qual das 15 classes de gestos a sequência pertencia.

É importante salientar que o modelo do reconhecedor pode gerar vários ruídos na predição, o que prejudica a segmentação da sequência. Neste sentido, aplicou-se sobre a sua predição um filtro de média móvel com janela de tamanho 3 e sobreposição 2. Com isso, o problema pôde ser atenuado, uma vez que agora o reconhecedor muda a sua predição de gesto para não-gesto quando a média é igual a 0, ou de não-gesto para gesto quando a média é igual a 1.

6.2.1 Experimento 1: Tempo de processamento

O objetivo desse primeiro experimento é verificar o desempenho da aplicação baseado na medição de seu tempo de processamento. A observabilidade projetada para o PIS permite que seja possível medir o tempo de processamento de cada serviço e o tempo de comunicação entre eles. O resultado de uma medida da cadeia completa de serviços que compõe a aplicação pode ser vista na Figura 46.

Figura 46 – Spam com gesture recogniser



Fonte: Produção do próprio autor.

Pode-se observar que o tempo de processamento do serviço Skeletons Detector é o de maior contribuição ao tempo de resposta total da cadeia. Na Tabela 9 é possível verificar que esse comportamento se apresenta mesmo considerando um número maior de amostras.

Tabela 9 – Média e desvio padrão do tempo de processamento de cada serviço para um número de 1000 amostras

Serviços	CG	SD	SG	GR
Média (ms)	12.77	72.58	1.02	9.27
Desvio Padrão (ms)	14.54	14.79	0.90	4.82

Fonte: Produção do próprio autor.

Esses valores foram medidos usando a versão do serviço SD desenvolvido para GPU. Devido ao tipo de operação realizado, esse serviço em muito se beneficia da GPU como recurso de processamento. Ainda que o mesmo serviço possa utilizar o processamento baseado em CPU, o tempo de processamento necessário para a sua execução é muito superior. Para verificar o impacto da utilização das duas tecnologias, foram desenvolvidas duas versões do mesmo serviço e medido o seu tempo de processamento. Na tabela 10 pode-se observar que a versão para GPU chega a ser duas ordens de grandeza mais rápida do que a versão para CPU com um core alocado. Ao aumentar o número de cores o tempo diminui, porém essa não é uma solução que se mostre escalável.

Tabela 10 – Média e desvio padrão do tempo de processamento do Skeletons Detector para um número de 1000 amostras

Skeletons Detector	CPU 1 core	CPU 2 core	CPU 3 core	CPU 4 core	GPU
Média (ms)	5.187,09	2.375,95	1.579,80	1.169,59	72,58
Desvio Padrão (ms)	400,51	192,18	145,64	117,50	14,79

Fonte: Produção do próprio autor.

Esses resultados mostram que a utilização da versão em CPU é inviável na maioria das aplicações práticas, principalmente se levar em consideração sua execução em tempo real. Considerando que GPU é um recurso com menor disponibilidade e maior custo, a gerência desse recurso se mostra essencial na infraestrutura do PIS.

6.2.2 Experimento 2: Taxas de Reconhecimento e Classificação

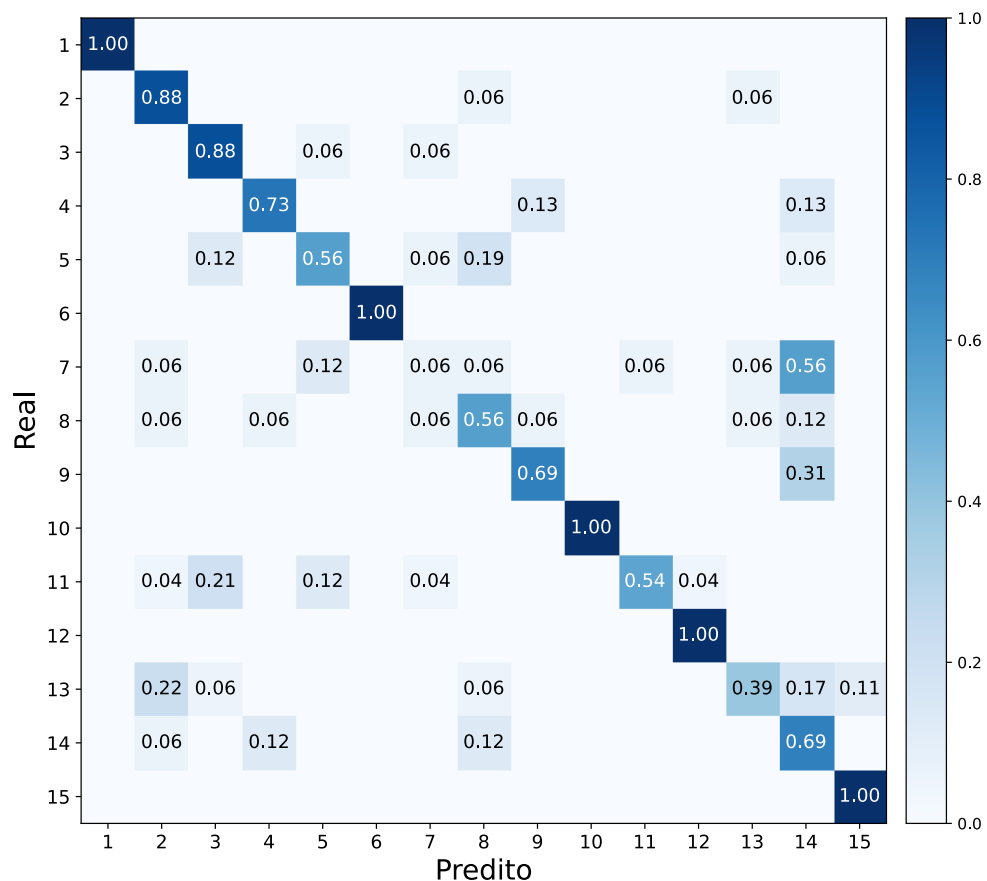
Nesse segundo experimento foi medido o desempenho das taxas de reconhecimento e classificação, separadamente e de forma conjunta. Os resultados foram analisados

comparando-se com os vídeos rotulados na base de dados. Assim, é possível analisar cada uma das taxas de forma independente.

O reconhecedor alcançou uma taxa média de 82% de reconhecimento sobre a base de testes. Considerando a média móvel utilizada, o segmentador pode estar realizando a predição três *frames* após o início e o fim do gesto, o que gera um erro de pelo menos seis observações por gesto. Dessa maneira, em um cálculo grosseiro, como o conjunto de teste possui 240 gestos e 18.932 observações (somadas as dos gestos e as dos não-gestos), tem-se um erro esperado de pelo menos 7%. Fazendo com que os 82% de taxa de reconhecimento alcançado seja ainda mais significativo. Isso mostra a efetividade do modelo, mesmo em um ambiente distinto e com usuários nunca vistos durante o treinamento.

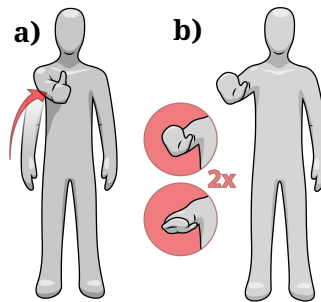
Na Figura 47, é possível ver a matriz de confusão com os resultados do classificador. A taxa média obtida na classificação dos gestos foi de 72,40%. Note que os gestos 1, 6, 10, 12 e 15 alcançaram a taxa máxima de 100%. Os gestos 2 e 3 alcançaram quase 90%, enquanto que o pior resultado foi para o gesto 7, que alcançou apenas 6%. Mesmo com o baixo valor na taxa para algumas classes, pode-se considerar que a proposta alcançou resultados satisfatórios.

Figura 47 – Matriz de confusão com os resultados do classificador de gestos.



Fonte: Produção do próprio autor.

Figura 48 – Exemplo de dois gestos que podem ser confundidos. a) gesto 7 e b) gesto 14. Nessas duas classes de gestos, o que as difere é a forma da mão.



Fonte: Produção do próprio autor.

O vocabulário de gestos utilizado possui gestos ambíguos que dependem não só do movimento dos braços, mas também da forma da mão. Dessa maneira, os gestos que podem ser diferenciados apenas com o movimento alcançaram a taxa máxima de classificação. Por outro lado, aqueles mais dependentes da forma da mão foram confundidos uns com os outros, como já era de se esperar.

Como exemplo, o que diferencia o gesto 7 do gesto 14 (Figura 48) é a forma das mãos, uma vez que eles possuem o mesmo movimento. No gesto 7, a mão está fechada e com o polegar apontando para cima, enquanto que no gesto 14 a mão está aberta com palma inicialmente para cima e depois para baixo.

Considerando a operação conjunta de ambos os modelos, a taxa obtida foi de 76%. O desempenho do serviço pode ser observado de forma consolidada na tabela 11.

Tabela 11 – Desempenho do Serviço Gesture Recognizer

	Reconhecedor	Classificador	Operação conjunta
Desempenho	82,0%	72,4%	76,0%

Fonte: Produção do próprio autor.

Ao se levar em conta todas as restrições intrínsecas ao vocabulário de gestos utilizado, julga-se que esse valor também é satisfatório. Implicando que o reconhecedor e o classificador possam ser utilizados em um ambiente interacional *online*.

6.2.3 Experimento 3: Gestos reconhecidos e sua relação com o número de instâncias do serviço SD

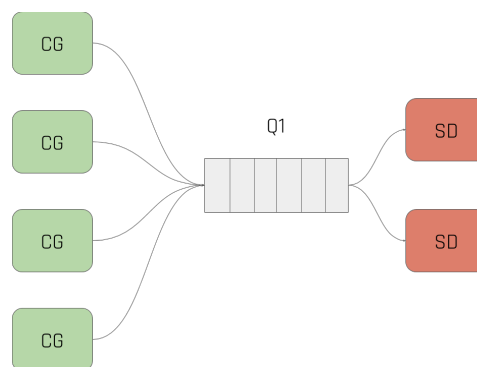
Vimos através do experimento 1 que o serviço de Skeletons Detector é aquele que mais contribui com o tempo de resposta da aplicação. Além disso, ele utiliza o recurso GPU para processamento, recurso esse que possui disponibilidade limitada na infraestrutura.

Essas características mostram a necessidade de gerenciamento mais apurado desse recurso em especial.

O objetivo desse experimento é relacionar a taxa de reconhecimento de gestos com os recursos ofertados ao serviço de Skeletons Detector.

Como dito anteriormente, o serviço de Skeletons Detector faz a leitura das imagens de uma fila localizada no barramento de mensagens, processa as imagens e gera na saída um vetor com os esqueletos 2D extraídos. Como pode ser visto na Figura 49, todos os CG enviam as imagens relativas às câmeras do ambiente para essa fila única.

Figura 49 – Fila única de mensagens na entrada do serviço de skeletons detector.



Fonte: Produção do próprio autor.

Caso a taxa de geração de imagens pelas câmeras aumente, o serviço de SD deve ter a capacidade de processá-las para evitar um aumento da fila e consequentemente um descarte das imagens. Esse aumento da capacidade de processamento pode se dar de duas formas: com o escalonamento vertical ou horizontal do serviço de SD.

No escalonamento vertical deve-se aumentar o poder computacional dos recursos ofertados ao serviço. Porém, como o SD roda essencialmente em GPU, aumentar esse poder computacional implica em ofertar uma outra GPU de maior capacidade ou aplicar técnicas que possibilitem o trabalho conjunto de múltiplas GPU para um mesmo serviço. Ainda que seja possível aplicar o escalonamento vertical, a alocação de múltiplas GPUs ou mesmo de uma GPU de maior capacidade são pouco viáveis para um tipo de recurso com disponibilidade limitada e custo elevado.

Já a técnica de escalonamento horizontal busca-se aumentar o número de instâncias de um serviço em execução simultânea. O aumento do número de instâncias do serviço SD implica em um aumento da taxa de processamento das imagens. Assim com o controle do número de instâncias busca-se o equilíbrio nas taxas de geração e processamento das imagens de forma a minimizar o descarte das imagens e seu impacto no reconhecimento dos gestos da aplicação.

O escalonamento horizontal é uma técnica simples e de fácil implementação em

infraestruturas de nuvem tal como o PIS. Porém, ela não é aplicável em qualquer situação. Tomemos como exemplo o serviço *Skeletons Grouper*. Esse serviço recebe como entrada os esqueletos 2D provenientes das múltiplas câmeras do ambiente dentro da janela de tempo definida pela taxa de FPS. O serviço precisa de pelo menos 2 esqueletos 2D para reconstruir o esqueleto 3D. Caso tenha mais entradas a reconstrução pode ser melhor. Se novas instâncias forem executadas cada uma delas trabalhará com um subconjunto dos esqueletos 2D e o resultado será pior em termos de localização e ainda seria necessário o tratamento da duplicação dos resultados de esqueletos 3D. Esse é um exemplo em que podemos observar que o aumento do número de instâncias de um serviço teria o efeito inverso ao desejado.

No caso do serviço SD ele é considerado um serviço sem estados, no sentido em que o processamento de cada imagem independe de qualquer outra imagem do ambiente, sejam imagens anteriores ou simultâneas. Essa característica possibilita que diferentes instâncias do SD possam ser executadas simultaneamente sem interferência mútua ou impactem no funcionamento da aplicação como um todo.

Nesse caso, um aumento do número de instâncias gera um aumento da capacidade de processamento, e como consequência impede o descarte de imagens em função do aumento da fila de mensagens. Com uma baixa taxa de descartes de imagens, melhores são as taxas de reconhecimento e classificação dos gestos.

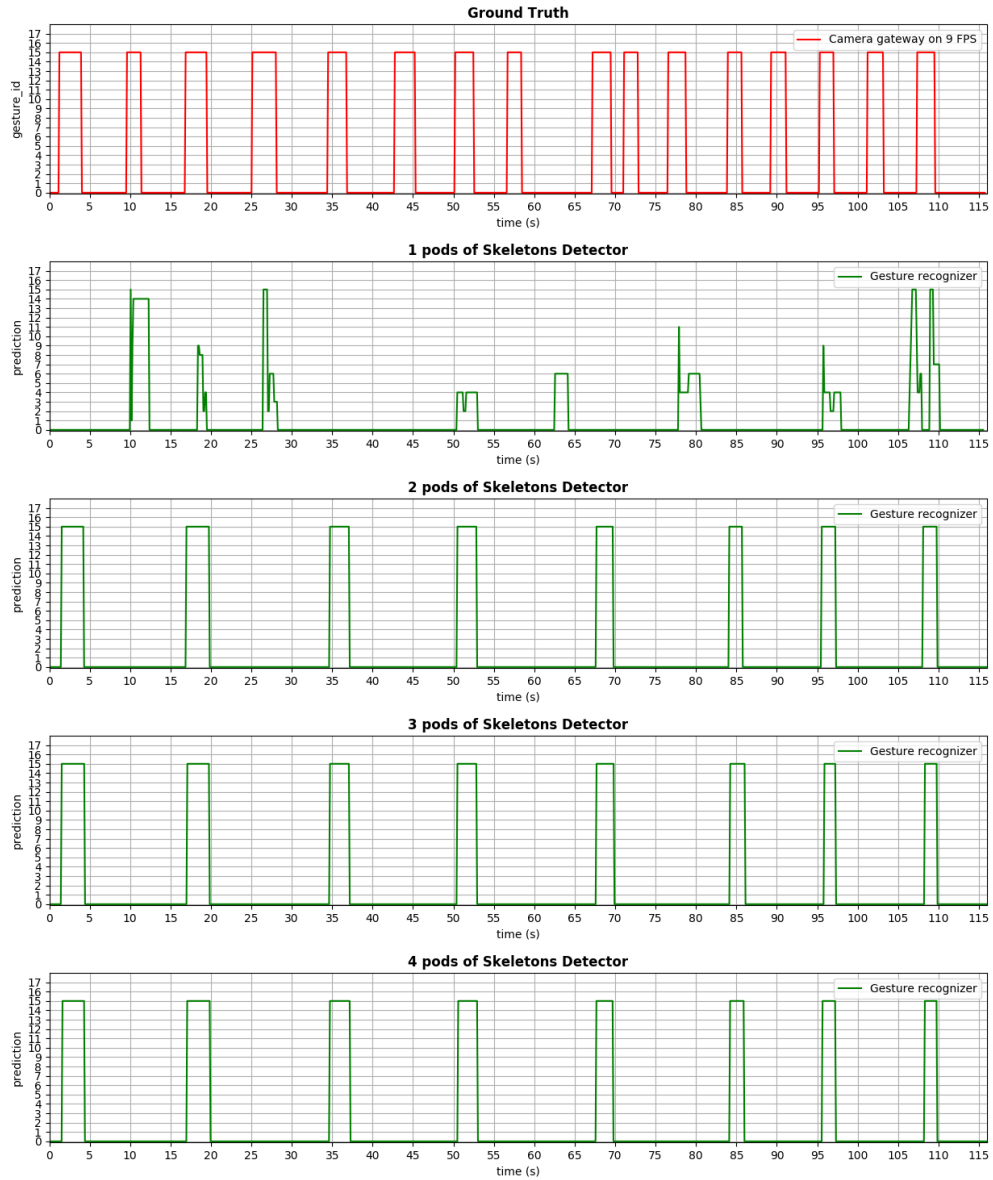
A relação entre o número de instâncias com a taxa de reconhecimento e classificação de gestos pode ser observada na figura 50. Ela mostra 5 gráficos que relacionam a classificação do gesto pelo tempo do experimento.

No experimento, cada esqueleto 3D de entrada do serviço GR é classificado em uma classe de gestos (classes de 1 a 15) ou não gesto (classe 0) de acordo com o vocabulário de gestos apresentado na Tabela 8. Para o experimento foi escolhida a classe 15, por ser uma das que atingiu 100% de taxa de classificação quando o classificador foi analisado de forma independente. O gráfico "Ground Truth" ilustra a rotulação das imagens da base de dados. Como resultado podemos observar a realização de 16 gestos da classe 15, bem como o tempo de duração de cada gesto. Para o experimento foi utilizada uma taxa fixa de 9 FPS para todas as câmeras. Essa taxa foi escolhida tomando como base um parâmetro máximo definido pela aplicação.

Podemos observar que ao se utilizar uma única instância de SD, o desempenho tanto do reconhecedor, quanto do classificador, é muito baixo. Ocorrem no reconhecedor tanto falsos positivos, como por exemplo entre 60s e 65s, como falsos negativos, como o gesto que ocorre no instante 35s. Já o classificador erra a classificação de todos os gestos, inclusive alterando a sua classe antes mesmo de sua finalização.

Já para 2 instâncias de SD tem-se um desempenho melhor. O reconhecedor não

Figura 50 – Reconhecimento dos gestos em função do número de instâncias do serviço Skeletons Detector.



Fonte: Produção do próprio autor.

apresenta mais nenhum falso positivo e consegue reconhecer 8 dos 16 gestos realizados. Já o classificador atinge 100% de todos os gestos reconhecidos.

É importante salientar aqui que a aplicação possui como meta a minimização do número de falsos positivos. Cada gesto reconhecido deve resultar em uma ação correspon-

dente. O objetivo é que não seja executada nenhuma ação sem uma solicitação. Já no caso de um gesto executado não ser reconhecido, ele poderá ser executado novamente sem prejuízo ao funcionamento da aplicação.

Ao se analisar os gráficos para 3 e 4 instâncias de SD, pode-se observar que os resultados possuem uma grande similaridade quando são utilizados apenas 2 instâncias. Para explicar esse resultado, foram feitas medidas da taxa de mensagens por segundo que servem de entrada para os serviços SD, SG e GR em função da variação do número de instâncias do serviço SD. Essas medições são apresentadas na figura 51.

Figura 51 – Taxa de mensagens de entrada dos serviços SD, SG e GR em função do número de instâncias



Fonte: Produção do próprio autor.

É possível verificar no gráfico a taxa de mensagens por segundo que servem de entrada para os serviços SD, SG e GR em função da variação do número de instâncias do serviço SD. Considerando que o experimento foi realizado utilizando-se 4 câmeras a 9 FPS, obteve-se uma taxa muito próxima a taxa máxima de 36 mensagens/s na entrada do serviço SD. A variação do número de instâncias do serviço SD não modifica significativamente a taxa de mensagens de entrada do próprio serviço.

As mensagens recebidas pelo serviço SD contém as imagens que serão analisadas. Os esqueletos 2D são detectados e o resultado é enviado através de mensagens para o serviço de SG. Cada mensagem analisada deve gerar uma mensagem correspondente na saída, exceto quando a imagem é descartada. O descarte ocorre no serviço SD sempre que

o tempo de resposta (1/FPS) é excedido.

Pode-se observar no gráfico que a quantidade de mensagens na entrada de SG, enviadas por SD aumenta na medida que o número de instâncias de SD também aumenta. Com o aumento do número de instâncias, aumenta-se a capacidade de processamento das imagens e um número menor delas excede o tempo de resposta.

Porém, o número de mensagens na entrada do serviço GR não segue linearmente o incremento na taxa de mensagens da entrada do SG. Observa-se que a partir de 2 instâncias, a taxa de entrada em GR se mantém próxima ao limite de 9 mensagens/s. Esse resultado se justifica em função de que com duas imagens simultâneas já é possível reconstruir um esqueleto 3D. Novos esqueletos 2D podem apresentar melhorias na reconstrução 3D, mas essas melhorias são menos significativas a partir desse valor. Logo, para essa aplicação, é obrigatório o aumento do número de instâncias até que seja possível a reconstrução dos esqueletos 3D aproximadamente na mesma taxa de captura de imagens das câmeras. Após esse valor, o controle do número de instancias deve ser feito de forma a avaliar a sua contribuição no reconhecimento e classificação dos gestos.

É possível observar por esse experimento que existe uma relação direta entre os recursos alocados para um serviço, nesse caso um maior número de instâncias, e o resultado esperado do serviço. Percebe-se isso, com o incremento do serviço SD de 1 para 2 instâncias. Porém, essa melhoria não é linear, como pode ser observado com o aumento de um número maior de instâncias.

Esse comportamento mostra a importância do compartilhamento de informações entre os níveis de aplicação e o de infraestrutura existente no PIS. Dessa forma, a taxa de esqueletos 2D se torna uma importante métrica que a infraestrutura pode utilizar para gerenciar racionalmente os seus recursos.

6.2.4 Experimento 4: Incerteza e sua relação com a taxa de FPS

Como o modelo utilizado é temporal, existe uma relação direta entre a taxa de classificação de gestos e a taxa de FPS disponibilizada para o serviço GR. Quanto maior maior o número de esqueletos 3D gerados durante a execução de um gesto maior é a assertividade no seu reconhecimento e classificação.

Para que se possa medir a taxa de desempenho da classificação do serviço GR, é necessário que se tenha previamente cadastrados todos os gestos executados no ambiente e se meça todos os gestos efetivamente reconhecidos pelo serviço. Em um ambiente de testes e treinamento, isso é feito através de uma base de dados rotulada, como visto nos experimentos anteriores.

Porém, em um ambiente de execução em tempo real, não existe essa rotulação prévia e por consequência não é possível medir a taxa de desempenho na classificação do

gesto. Isso se apresenta como um problema para mensurar o desempenho da aplicação, e por consequência, o gerenciamento dos recursos pela infraestrutura como forma de atender a esse requisito de desempenho.

Apesar de não ser possível a medição direta da taxa de acertos, a aplicação fornece uma alternativa para avaliar o seu próprio desempenho. Essa forma alternativa é realizada através do cálculo de um índice de incerteza no reconhecimento de cada gesto.

Um classificador comumente estima a probabilidade da sua entrada pertencer a uma das classes para as quais foi treinado. Nesse sentido, poderia-se utilizar esse valor de probabilidade como sendo o grau de certeza do classificador. Porém, devido à falta de base de dados representativas para o problema a ser resolvido, os classificadores tendem a serem treinados de maneira a tornarem-se altamente confiantes em suas predições. Dessa maneira, mesmo uma entrada que não pertença ao domínio do problema (um ruído), poderá ser classificada com alto valor de probabilidade. Isso poderia prejudicar significativamente a tomada de decisão quanto a alocação de recursos em função da qualidade de predição do modelo. Sendo assim, é necessário um modelo que forneça um valor de incerteza mais sensível a esse excesso de confiança.

Nesse sentido, o serviço utiliza um modelo de classificação estocástico que oferece além da probabilidade, a incerteza da classificação para cada entrada recebida. Em um modelo estocástico, a mesma entrada sempre produz valores distintos na saída, mesmo que muito próximos. Dessa maneira, repetindo a mesma entrada N vezes, ter-se-ão N saídas distintas (amostras). Com isso, uma possível medida de incerteza pode ser a variância das N amostras. Uma variância pequena, significa que o modelo está de acordo que a entrada pertence a classe com maior probabilidade média, uma variância maior, significa que o modelo não tem certeza sobre a sua predição.

O problema no uso da variância na medição da incerteza do modelo é que ela pode ser pequena, mesmo que a probabilidade média também seja. Por exemplo, uma variância $\sigma^2 = 0.01$ com uma probabilidade média $\mu = 0.5$ não traz muita confiança sobre a assertividade do modelo. Dessa forma, uma métrica de incerteza que leva em conta tanto o valor de probabilidade quanto a variação das N amostras é a entropia.

Sendo assim, a incerteza fornecida por esse serviço refere-se ao valor de entropia calculado entre 100 amostras fornecidas pelo modelo. Em complemento, a classe predita é aquela que possui o maior valor médio de probabilidade calculado entre as 100 amostras. Em termos práticos, o número de 100 amostras foi utilizado tanto para o reconhecedor, como para o classificador de gestos.

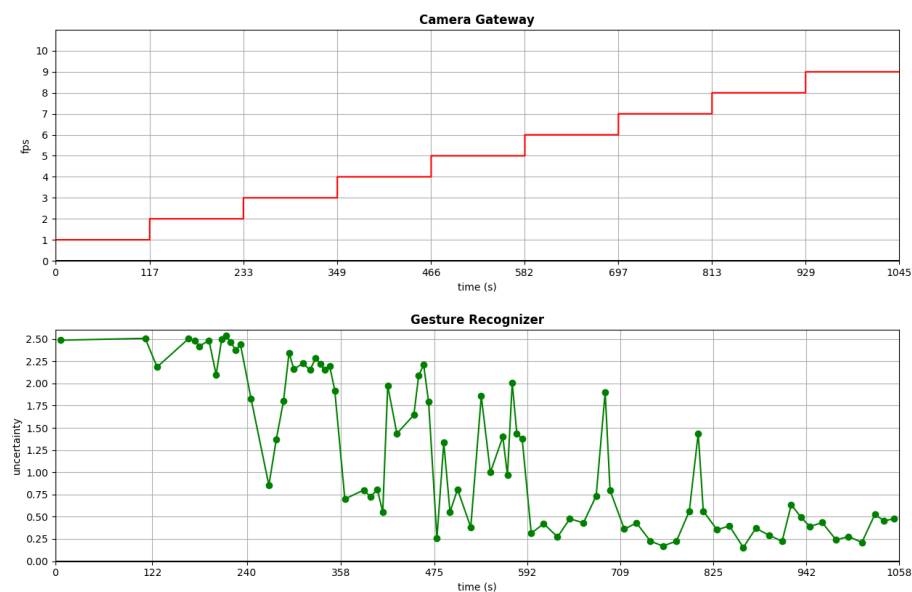
Portanto, o serviço de GR tem como saída a classificação de cada gesto dentro do vocábulo de gestos apresentados na Tabela 8, em conjunto com o valor da incerteza dessa mesma classificação. Do ponto de vista da aplicação, a incerteza se torna um requisito

essencial do serviço GR. Ainda que ela não represente a taxa de acerto do classificador, ela substitui essa taxa e passa a ser utilizada como método de avaliação de seu desempenho.

Dada essa correlação existente entre a taxa de desempenho na classificação dos gestos e o valor de sua incerteza, é possível então analisar a variação dessa incerteza em relação com a taxa de FPS utilizada pelas câmeras.

A Figura 52 mostra o resultado de um experimento com a relação entre FPS e a incerteza. Nesse experimento foi utilizado o mesmo conjunto de imagens do experimento 3 com 2 instancias do serviço SD. Para cada valor de FPS, 16 gestos da classe 15 são realizados. Após a execução dos 16 gestos, a taxa de FPS é incrementada e o mesmo conjunto de imagens é analisado novamente com a nova taxa. Esse ciclo se repete até uma taxa limite de 9 FPS imposta pela aplicação.

Figura 52 – Incerteza do classificador do gesto em função da taxa de FPS das câmeras



Fonte: Produção do próprio autor.

Cada ponto do gráfico representa o valor de incerteza de um gesto reconhecido. Pode-se observar que para 1 FPS, somente 2 gestos foram reconhecidos. Já para a taxa de 2 FPS ou mais, oito reconhecimentos foram realizados, igualando a taxa obtida no experimento anterior.

Com relação à incerteza da classificação calculada pelo serviço, ela se mantém alta para valores de 1 e 2 FPS, mesmo com um incremento no número de reconhecimentos. A medida que a taxa de FPS é aumentada, os valores de incerteza diminuem.

Um ponto importante a ser observado é a variação do valor da incerteza dentro

de uma mesma taxa de FPS. Considere por exemplo os valores obtidos para a taxa de 3 FPS. Pode-se observar uma variação de 0,85 a 2,35. Essa variação ocorre principalmente devido às condições de execução de cada um dos gestos. Um gesto executado de forma muito diferente do padrão proposto, dificulta o seu reconhecimento, e consequentemente aumenta a incerteza de sua classificação. Porém, é possível observar que a medida que a taxa de FPS aumenta, a incerteza dos pontos diminui. Como resultado, para uma taxa de 8 ou 9 FPS todos os pontos possuem uma incerteza abaixo de 0,75.

Com os resultados, fica clara a relação direta entre os recursos ofertados através do aumento da taxa de FPS das câmeras, com o valor da incerteza na classificação dos gestos.

6.2.5 Experimento 5

Nós vimos através dos experimentos anteriores que a aplicação possui um conjunto de requisitos específicos e eles possuem uma relação com a alocação de recursos da infraestrutura. Esses requisitos podem ser representados pelas taxas de desempenho da aplicação ou através de um conjunto de métricas que possuem relação direta com o desempenho.

Neste último experimento, a aplicação será executada e o gerenciamento dos recursos será realizado. O objetivo é mostrar que o PIS é capaz de atender aos requisitos específicos de uma aplicação como essa, realizando um controle racional dos recursos da infraestrutura.

A aplicação, ao ser executada, deve ser primeiro capaz de detectar se há pessoas no ambiente. Caso existam pessoas, seus esqueletos deverão ser identificados nas imagens e sua localização será encontrada. De posse da localização dos esqueletos é possível reconhecer se a pessoa está executando um gesto e classificá-lo.

Para o seu pleno funcionamento, a aplicação possui dois requisitos principais: a incerteza da classificação dos gestos e a taxa de esqueletos 2D na entrada do serviço SG. Como visto no experimento 4, não é possível medir diretamente o desempenho do classificador em uma aplicação em tempo real. Por conta disso, a aplicação calcula o valor da incerteza da classificação. Quanto menor o valor de incerteza, melhor é o desempenho do classificador. Portanto, a depender da situação em que a aplicação será executada, um valor máximo de incerteza é estabelecido.

O segundo requisito diz respeito ao correto funcionamento do serviço SG que compõe a aplicação. O serviço SG precisa que o esqueleto de uma pessoa seja detectado em pelo duas imagens capturadas simultaneamente. Portanto, o serviço SG deve ter em sua entrada uma taxa mínima de mensagens de 2 vezes a taxa de FPS das câmeras. Além disso, aumentar ainda mais essa taxa de mensagens contribui para que o valor da incerteza seja ainda menor.

Foi possível observar através dos experimentos anteriores que esses requisitos estão

diretamente relacionados com os recursos alocados para a aplicação. Dessa forma, além de seus requisitos, a aplicação precisa fornecer à infraestrutura um conjunto de informações que relacionem os requisitos com os recursos. Esse conjunto de informações deve incluir ainda características da aplicação que auxiliem a infraestrutura no gerenciamento desses mesmos recursos.

Informações compartilhadas pelas aplicações

Além dos requisitos, uma informação básica a ser compartilhada é a cadeia de serviços que compõe a própria aplicação. Os serviços que compõe a cadeia para execução desse experimento podem ser observados na Figura 43. Para cada serviço é feito um descrição de seus parâmetros e características específicas, como por exemplo a imagem base do container a ser executada ou as configurações iniciais para sua inicialização.

Uma outra informação relevante que a aplicação deve informar à infraestrutura é sobre quais serviços foram desenvolvidos especificamente para serem executados em GPU. Os serviços de SD e GR foram desenvolvidos de forma a se aproveitar do alto grau de paralelismo oferecido por esse dispositivo. Porém o serviço de SD também possui uma versão para CPU. Ainda que o tempo de execução do serviço da versão em CPU seja muito superior à versão em GPU, como visto no experimento 1, a versão em CPU consome menos recursos e será utilizada pela aplicação em uma situação bem específica para detecção de pessoas no ambiente. Os demais serviços usam a CPU como unidade de processamento.

Também é passado para a infraestrutura o grau de paralelismo admitido por cada serviço. Por exemplo, uma única instância do serviço de CG pode ser utilizado para interfacear todas as câmeras do ambiente. Porém o tratamento das imagens realizado pelo CG impõe um tempo de execução considerável. Se um único Gayeway for instanciado, ele terá que tratar as imagens provenientes das 4 câmeras sequencialmente, quadruplicando o tempo de execução do serviço. Um aumento do número de instâncias desse serviço reduz o tempo total de execução. Porém, esse aumento deve ser limitado ao número de câmeras do ambiente. Um número de instâncias de CG superior ao número de câmeras, implicaria em instâncias ociosas e um consumo desnecessário de recursos.

Da mesma forma que o serviço de CG, o serviço de SD pode ser instanciado até o limite de câmeras do ambiente. Já para os serviços de SG, como explicado no experimento 4, deve ter apenas uma única instância em execução, e por consequência o mesmo se aplica ao serviço de GR.

Alguns parâmetros específicos do funcionamento das câmeras também devem ser fornecidos pela aplicação. Informações como resolução e canais de cores das imagens são essenciais ao seu correto funcionamento. Para esse experimento específico esses parâmetros são passados à infraestrutura em sua inicialização e não são mais alterados durante a execução. Já a taxa de FPS, deve ser alterada. Esse é o principal parâmetro que será

modificado pela infraestrutura para que o requisito de incerteza na classificação dos gestos seja atingido. O segundo fator controlado pela infraestrutura para atendimento ao requisito máximo de incerteza e o número de instâncias de SD.

A incerteza na classificação dos gestos é influenciada por diferentes fatores. Mudanças na iluminação do ambiente ou posicionamento das pessoas podem dificultar a sua classificação. Uma compensação desses fatores pode ser realizado parcialmente pelo aumento da taxa de FPS das câmeras ou o número de instâncias de SD. Da mesma forma, em situações mais favoráveis de classificação, a taxa de FPS ou o número de instâncias poderão ser diminuídos e ainda assim o requisito de incerteza continuará sendo atendido. Isso reforça o caráter dinâmico do controle dessa taxa pelo orquestrador.

O orquestrador então, de posse dessas informações, inicializa a aplicação alocando os recursos e configurando os dispositivos adequadamente. Além disso, parte das informações são atualizadas constantemente através de métricas geradas pela própria aplicação para subsidiar o orquestrador em sua tomada de decisões. Permitindo assim, uma melhor alocação dos recursos e o atendimento aos requisitos da aplicação.

As métricas utilizadas nesse experimento serão o valor da incerteza na classificação dos gestos e a taxa de descartes de mensagens pelo serviço SD. esta última métrica é calculada pela divisão da taxa de mensagens na entrada do serviço SD pela sua taxa da saída.

Além dos próprios requisitos, o número de esqueletos nas imagens será utilizado como uma terceira métrica como forma de detecção de pessoas no ambiente.

Funcionamento da Aplicação

Para inicializar a aplicação, o orquestrador primeiramente executa os serviços de CG e SD na versão desenvolvida para CPU. Ambos os serviços são utilizados para detectar se há alguma pessoa na sala. Não havendo nenhuma pessoa na sala, não há necessidade dos outros serviços estarem em execução. O tempo de execução do serviço SD para CPU é muito elevado para ser usado para o reconhecimento de gestos. Esse tempo de processamento elevado faz com que o serviço descarte múltiplas imagens na fila de entrada inviabilizando o reconhecimento e classificação dos gestos. Porém, ao ser usado com a finalidade de apenas detectar pessoas no ambiente, esse descarte é aceitável, dado que a detecção não necessita de múltiplas imagens sequencias e o atraso de alguns segundos na detecção de pessoas é suportado.

Nesse experimento foi alocado 4 cores de CPU para uma única instância de SD, de forma que o tempo de processamento do serviço fique próximo a 1 segundo. Como consequência, as câmeras foram configuradas para iniciarem com uma taxa de FPS igual a 2. Considerando que o serviço não é capaz de processar uma taxa de imagens superior

a essa, a taxa reduzida de FPS tem como resultado uma diminuição da utilização dos recursos da infraestrutura, principalmente com relação ao tráfego na rede.

Ao detectar uma pessoa, o serviço SD informa ao orquestrador através da métrica de esqueletos 2D. O orquestrador então, troca a versão do serviço de SD da versão em CPU para a versão em GPU. Além disso, o orquestrador inicializa os serviços de SG e GR.

A partir da inicialização de todos os serviços, o orquestrador começa a monitorar as duas outras métricas: o valor da incerteza na classificação dos gestos e a taxa de descartes de mensagens pelo serviço SD.

A taxa de descartes de mensagens é monitorada pela infraestrutura através do número de mensagens nas filas de entrada dos serviços SD e SG. Ambas as filas ficam localizadas no broker que envia ao servidor de métricas do PIS.

O serviço IS Orchestrator funciona como o orquestrador de mais alto nível hierárquico. Ele é responsável por ler as informações multinível compartilhadas e acionar diretamente os gerenciadores de recursos e também os orquestradores de nível mais baixo. Nesse experimento, o IS Orchestrator configura o HPA do Kubernetes, de forma a utilizar a métrica de taxa de descarte de mensagens por SD.

O HPA é configurado na inicialização do serviço SG. A partir desse ponto, ele monitora essa métrica de forma a mantê-la sempre em um valor igual ou superior a 1.2, mesmo que a taxa de FPS seja alterada em tempo de execução.

Para manter a métrica nesse patamar o HPA aumenta ou diminui o número de instâncias do serviço SD. Quanto maior o número de instâncias, maior a capacidade de processamento das imagens e menor o número de descartes.

A outra métrica monitorada pelo IS Orquestrador é a incerteza na classificação dos gestos. Essa métrica é calculada pelo serviço GR e passada para a infraestrutura. O orquestrador monitora essa métrica controlando os recursos da infraestrutura com o objetivo de mantê-la dentro de uma faixa de valores definida pela aplicação. Para esse experimento, o valor da incerteza usado foi de 0,6 com uma tolerância de 0,3.

Tanto a métrica de detecção de esqueletos, quanto a métrica de incerteza possuem um grau de variabilidade alto em função das condições do ambiente, como a forma de execução dos gestos e posicionamento das pessoas nas imagens. Como forma de diminuir o ruído e consequentemente a oscilação no controle dos recursos, o orquestrador monitora a média dos valores das métricas. A detecção de esqueletos é realizada a cada frame e foi utilizada a média móvel de janela tamanho 10 e sobreposição 9. Já a incerteza só é calculada ao final da execução de um gesto reconhecido. Dado o número menor de métricas gerados por tempo, foi usada uma média móvel de janela tamanho 5 e sobreposição 4.

O controle realizado pelo IS Orquestrador é realizado através da configuração da

taxa de FPS das câmeras e pelo escalamento horizontal do número de instâncias de SD. Enquanto o valor da incerteza estiver acima da faixa estipulada, o orquestrador pode atuar elevando um ou outro parâmetro, ou mesmo ambos simultaneamente. Porém, caso as condições para reconhecimento dos gestos se alterem e o valor da incerteza reduza muito abaixo da faixa especificada pela aplicação, o orquestrador atua no sentido contrário e diminui a taxa de FPS e/ou o número de instâncias, fazendo com que a incerteza volte ao patamar desejado. Assim, o requisito da aplicação continua atendido e os recursos são economizados.

Esse experimento 5 inicia-se com ninguém no ambiente. Depois, os mesmos vídeos dos experimentos anteriores são utilizados. Esses vídeos mostram duas pessoas em sequencia realizando o gesto da classe 15. Esse vídeo se repete continuamente e termina com o mesmo vídeo sem ninguém do início do experimento.

Análise dos resultados

O resultado do experimento é apresentado na Figura 53 através de 5 gráficos. O primeiro gráfico mostra o número de pods do serviço de Skeletons Detector desenvolvidos para CPU e GPU. O número de pods representa o número de instâncias de cada serviço. O segundo gráfico representa a métrica de taxa de mensagens descartadas utilizada pelo HPA do Kubernetes. O terceiro gráfico mostra a taxa de FPS das câmeras. A incerteza na classificação dos gestos pode ser observada no quarto gráfico. E por último é mostrado a métrica de número de esqueletos detectados no ambiente.

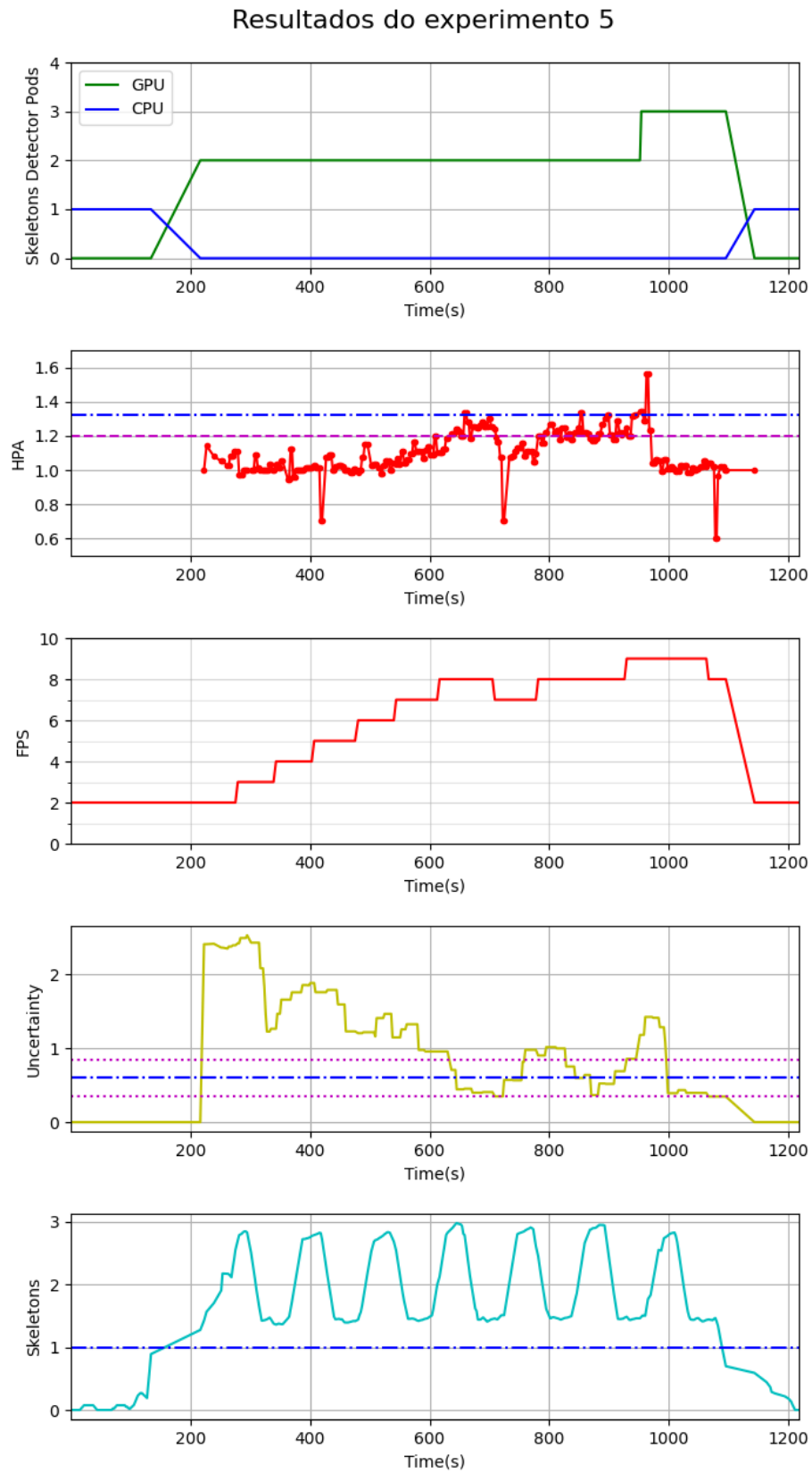
No início do experimento somente um pod do serviço SD para CPU está em execução. Como não há ninguém no ambiente, os serviços de SG e GR não foram inicializados e o HPA não é ainda necessário.

Ao identificar alguém na sala o orquestrador inicializa o serviço SD para GPU e todos os demais serviços que compõe a aplicação. Além disso o serviço SD para CPU é finalizado. A partir desse momento é possível medir os valores de incerteza e a taxa de descartes de mensagens pelo serviço SD.

Com a taxa de FPS em 2 a incerteza é muito alta. Por isso, o orquestrador começa a elevar essa taxa de forma que o valor de incerteza diminua. Com alguma oscilação inerente a precisão no reconhecimento e classificação dos gestos, a incerteza vai reduzindo até alcançar a faixa de incerteza definida pela aplicação. Isso ocorre a uma taxa de 8 FPS.

A medida que a taxa de FPS aumenta pode-se perceber através do gráfico do HPA um aumento no descarte de mensagens. Valores acima de 1 indicam que descartes ou ao menos atrasos na saída do serviço SD estão ocorrendo. O HPA foi configurado pelo IS Orchestrator para atuar em 1.2 com uma tolerância de 0,1. Até a taxa de 8 FPS, a métrica continua dentro da tolerância e por isso não há atuação do HPA.

Figura 53 – Resultado do experimento 5.



Fonte: Produção do próprio autor.

Após algum tempo na taxa de 8 FPS, o valor de incerteza, em função das condições de execução dos gestos, cai abaixo da faixa pré estabelecida. Nesse momento o orquestrador diminui o FPS para 7 como forma de diminuir a utilização dos recursos da infraestrutura.

Novas variações ocorrem até que a taxa de FPS chegue a 9. O problema nesse ponto é que o aumento do FPS começa a causar o efeito inverso ao desejado. O valor da incerteza aumenta ainda mais. Isso ocorre devido ao aumento de descartes estar muito elevado dificultando o reconhecimento e classificação dos gestos.

Para solucionar esse problema o HPA atua, inicializando uma nova instancia de SD. Essa alteração faz com que a métrica do HPA retorne para um valor próximo de 1. Como consequência da diminuição do número de descartes, o valor da incerteza cai a taxa de FPS pode ser reduzida novamente. Dessa forma, chega-se a um novo ponto de equilíbrio até que não tenha mais ninguém no ambiente e o orquestrador finalize os serviços não necessários e inicialize novamente o serviço SD para CPU para que ele possa monitorar a presença de novas pessoas no ambiente.

Pode-se observar através desse estudo de caso, como os recursos gerenciados pelo PIS permitem que os requisitos específicos de uma aplicação de visão computacional sejam atendidos. Percebe-se que os recursos são alocados e liberados apenas na medida necessária evitando-se o seu desperdício.

Para isso, o PIS utiliza um modelo de orquestração baseado em informações fornecidas tanto pela aplicação, quanto informações coletadas da infraestrutura. A orquestração é realizada de forma hierárquica, onde o orquestrador local, aqui representado pelo HPA do Kubernetes, possui autonomia limitada para alocação de instâncias do serviço SD. Esse orquestrador é configurado por um outro orquestrador de mais alto nível, aqui representado pelo serviço IS Orchestrator, que possui uma visão mais abrangente de todo o ambiente.

Com isso, os objetivos do PIS são alcançados, tanto no atendimento dos requisitos específicos da aplicação quanto ao uso racional dos recursos da infraestrutura.

7 Conclusões

Espaços Inteligentes baseados em visão computacional apresentam importantes desafios para o seu pleno funcionamento. Suas aplicações demandam um grande volume de dados, e a alta complexidade de seus algoritmos exigem um alto poder computacional para extrair as informações desses dados. Ao se considerar ainda, que muitas dessas aplicações são executadas em tempo real, o tempo máximo de resposta também se torna uma importante característica desses espaços.

Além dessas características, tais espaços precisam ser escaláveis de forma que possam ser utilizados em espaços de diferentes abrangências. Devem ser capazes de executar múltiplas aplicações com requisitos distintos simultaneamente. E tais requisitos podem ser dinâmicos, alterando-se em tempo de execução.

É comum que os requisitos de uma aplicação sejam definidos por elementos da infraestrutura. Porém, essa classe de aplicações possuem requisitos específicos que somente a aplicação tem conhecimento e somente ela poderá mensurá-los. Podemos citar o erro de trajetória de um robô ou a incerteza no reconhecimento de gestos, apresentados nos dois estudos de caso, como exemplos de requisitos específicos das aplicações. Pudemos observar através de diferentes experimentos, uma relação direta entre esses requisitos específicos e os recursos disponibilizados pela infraestrutura.

Considerando o Espaço Inteligente baseado em visão computacional é um ambiente com infraestrutura compartilhada entre múltiplas aplicações com diferentes requisitos. É necessário que seja estabelecido um controle adequado, para que os requisitos específicos das aplicações sejam atendidos, e o uso racional dos recursos seja realizado.

Além disso, pôde-se observar que o atendimento aos requisitos específicos das aplicações implica que requisitos rigorosamente correlacionados da infraestrutura também devam ser atendidos. Pôde-se observar nos experimentos que um aumento do número de imagens implica em: um aumento da capacidade de processamento, um aumento do tráfego na rede e uma diminuição do tempo de resposta. Atender a todos esses requisitos simultaneamente em um ambiente com infraestrutura compartilhada foi possível através da utilização da arquitetura proposta nessa tese.

A arquitetura proposta para Espaços Inteligentes baseados em visão computacional possui diferentes características que facilitam resolver os desafios apresentados. Porém dois habilitadores se mostram essenciais: i) orquestração multinível centrada na observabilidade conjugada das aplicações e das camadas de infraestrutura; e ii) programabilidade granular da infraestrutura.

Comumente, os níveis de aplicação e infraestrutura são completamente dissociados e requisitos específicos não são considerados. O controle dos recursos pela infraestrutura é feito por inferência das necessidades das aplicações. A correlação entre os diferentes requisitos e sua relação com os recursos disponibilizados não é estabelecido, o que dificulta ainda mais o gerenciamento dos recursos.

A proposta apresentada estabelece mecanismos de intercambio de informações dos diferentes níveis. Com isso, é possível definir o que deve ser observado, os seus relacionamentos e em que pontos deve-se especificamente atuar para atender aos objetivos propostos.

Dois estudos de caso foram apresentados para validar a proposta. No primeiro estudo de caso foi realizado o controle visual de um robô móvel no contexto da Indústria 4.0. A aplicação desse estudo de caso possui um requisito específico de erro de trajetória do robô. Também é apresentada a relação entre esse requisito específico com o tempo de resposta da tarefa de controle.

A observação dos níveis de aplicação e infraestrutura permitiu a medição do tempo de resposta da tarefa de controle da aplicação. Através desse monitoramento, pode-se perceber um alto tempo de comunicação com o robô pela rede sem fio. A programação granular da infraestrutura de rede sem fio, permitiu que as antenas fossem reprogramadas de forma a diminuir o tempo de resposta da aplicação e o requisito de erro de trajetória do robô fosse atendido.

No segundo estudo de caso foi utilizada uma aplicação de reconhecimento de gestos em um ambiente interacional. Um requisito específico apresentado nesse estudo de caso é a incerteza da classificação do gesto. Diversas informações foram compartilhadas entre a aplicação e a infraestrutura, incluindo a relação entre a incerteza e número de câmeras e sua taxa de FPS. Todas essas informações foram utilizadas para monitoramento e controle dos recursos pelo orquestrador hierárquico. Nesse caso, o orquestrador de mais alto nível possui uma visão mais abrangente e controlava diretamente alguns dos recursos da infraestrutura, mas também configurava o orquestrador de mais baixo nível para controle de recursos específicos.

É importante frisar que nesse experimento o valor da incerteza varia não só pela alocação dos recursos da infraestrutura, mas também é fortemente influenciada por fatores do ambiente. Pode-se observar pelos resultados que a observação de métricas da própria aplicação permitem que a infraestrutura compense, esses fatores externos, mantendo a incerteza dentro do parâmetro pré estabelecido.

Outro resultado observado é o uso racional dos recursos pela infraestrutura. O orquestrador foi capaz de alocar e desalocar os recursos de acordo com a necessidade da aplicação, evitando sua sub ou super alocação.

Outro ponto importante a ser enfatizado, é que a arquitetura aqui proposta foi implementada e utilizada em diferentes ambientes com múltiplas aplicações. Além disso, esses Espaços Inteligentes Programáveis foram utilizados como base para vários projetos de pesquisa. Desta forma percebe-se que a arquitetura aqui apresentada é genérica o suficiente para ser usada em diferentes contextos e cenários.

7.1 Trabalhos Futuros

A arquitetura proposta prevê mecanismos para o compartilhamento de informações multinível. Porém a definição das relações entre os requisitos específicos das aplicações e os recursos da infraestrutura dependem de um conhecimento prévio que nem sempre está disponível. Para o estabelecimento dessas relações ainda é necessário a atuação de um especialista com uma visão, mesmo que parcial, dos níveis de aplicação e infraestrutura.

Um possível caminho para solução desse problema seria o estabelecimento de um modelo ontológico que estabelecesse as relações necessárias entre as entidades existentes dos diferentes níveis do modelo. Dessa forma, possibilitaria inclusive uma tradução entre os requisitos dos diferentes níveis e diminuiria a necessidade de uma visão conjugada nem sempre disponível.

Uma outra abordagem para definição e controle dos recursos da infraestrutura e seu impacto nos requisitos específicos das aplicações, seria a implantação de aprendizado automático pelo orquestrador. Nesse caso, a medida que o orquestrador altere a alocação de recursos, seria observado o impacto positivo ou negativo nos requisitos específicos da aplicação. Cada mudança geraria um maior volume de informação de forma a contribuir com o processo de aprendizado do orquestrador. A expectativa é que essa abordagem diminua a necessidade de uma descrição rígida das relações entre os requisitos e a alocação de recursos.

O PIS provê os mecanismos para atender aos requisitos das aplicações e, ao mesmo tempo, fazer o uso racional dos recursos da infraestrutura. Porém, uma funcionalidade a ser incorporada é a busca pela melhor alocação possível de recursos. Portanto, uma evolução natural ao PIS seria a incorporação de mecanismos de otimização do processo de alocação de recursos. Os benefícios serão ainda mais relevantes em um cenário de maior disputa de recursos causado por múltiplas aplicações diferentes.

Outra futura contribuição seria na melhoria de desempenho na comunicação entre os dispositivos e serviços que transmitem imagens pela infraestrutura. O volume de dados por imagem transmitida é muito grande, e gera um impacto considerável no tempo de comunicação, podendo em alguns casos inviabilizar o atendimento aos requisitos das aplicações.

O principal modelo de comunicação é baseado no broker, que possui vantagens no controle das comunicações entre os serviços mais em contrapartida gera um impacto no desempenho. Está previsto no PIS a utilização de outros modelos de comunicação, porém a utilização de um modelo híbrido de comunicação que alie as vantagens de gerenciamento da comunicação do broker com um outro mecanismo de encaminhamento mais eficiente não foi pesquisado.

Por fim, não fez parte do escopo dessa tese as questões relativas a segurança. Porém é um item essencial é precisa ser investigado e incorporado à arquitetura.

7.2 Contribuições

7.2.1 Publicações

O desenvolvimento dessa pesquisa gerou as seguintes publicações:

- Ano de 2016:**
- DO CARMO, A. P.; ZAMPERINI, T. M.; MELLO, M. R. S.; LEAL, A. L. C.; GARCIA, A. S.. Ontologia das coisas para espaços inteligentes baseados em visão computacional. Brazilian Ontology Research Seminar - ONTOBRAS, 2016, p. 181–186
- Ano de 2017:**
- SANTOS, C. C.; PICORETI, R.; CARMO, A. P.; VASSALLO, R. F.; GARCIA, A. Modelagem de um Comportamento Interacional Entre Homen e Robô para um Espaço Inteligente Baseado em Visão Computacional. XIII Simpósio Brasileiro de Automação Inteligente - SBAI, 2017, p. 2265-2270.
 - GOMES, R. L.; MARTINELLO, M.; VASSALLO, R. F.; DO CARMO, A. P.; GARCIA, A. S.; SIMEONIDOU, D. et. al. How can emerging applications benefit from EaaS in open programmable infrastructures?. IEEE Summer School on Smart Cities 2017 - IEEE S3C2017, 2017.
- Ano de 2018:**
- ALMONFREY, D.; DO CARMO, A. P.; DE QUEIROZ, F. M.; PICORETI, R.; VASSALLO, R. F.; SALLES, E. O. T. A flexible human detection service suitable for Intelligent Spaces based on a multi-camera network. International Journal of Distributed Sensor Networks, 2018, <https://doi.org/10.1177/1550147718763550>.
 - PICORETI, R.; DO CARMO, A. P.; DE QUEIROZ, F. M.; GARCIA, A. S.; VASSALLO, R. F.; SIMEONIDOU, D. Multi-

level Observability in Cloud Orchestration. 16th Int. Conf. on Pervasive Intelligence & Comp., 2018, <https://10.1109/DASC/PiCom/Data>

- Marquez, P.; MARTINELLO, M.; VASSALLO, R. F.; DO CARMO, A. P.; SIMEONIDOU, D. et. al. Optical and wireless network convergence in 5G systems – an experimental approach. IEEE Computer-Aided Modeling Analysis and Design of Communication Links and Networks. 2018.
- SANTOS, P. B.; DO CARMO, A. P.; VASSALLO, R. F. et. al. Monitoramento de Recursos para Aplicações de Robótica em Espaços Inteligentes baseados em uma Nuvem OpenStack. XVI Workshop em Clouds e Aplicações, 2018.

Ano de 2019: • DO CARMO, A. P.; GARCIA, A. S.; VASSALLO, R. F.; SIMEONIDOU, D. et. al. Programmable intelligent spaces for Industry 4.0: Indoor visual localization driving attocell networks. Transactions on Emerging Telecommunications Technologies, 2019, <https://10.1002/ett.3610>.

- COTTA, W. A. A.; DO CARMO, MACHADO, F.; VASSALLO, R. F.; A. P., DE QUEIROZ, F. M.; GARCIA, A. S.; VASSALLO, R. F. MobiLysa - Sistema de Localização e Controle do Cão-guia Robô Lysa para Ambientes Internos baseado em Visão Computacional. Anais Estendidos do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia), 2019.

Ano de 2020: • DO CARMO, A. P.; DE QUEIROZ, F. M.; SANTOS, C. C.; SILVA, L. A.; VASSALLO, R. F.; . Uso de um espaço inteligente baseado em visão computacional para o controle de formação de robôs móveis. Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP), 2020.

- CUSTODIO, P. V. S.; DE QUEIROZ, F. M.; ALMONFREY, D.; COTTA, W. A. A.; DO CARMO, A. P.; VASSALLO, R. F.; Proposta de um Ambiente Interacional baseado em um Espaço Inteligente Programável. Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP), 2020.

7.2.2 Implementações

Como resultado deste trabalho foram realizadas 3 implantações do PIS:

- Laboratório VIROS - UFES - Vitória, ES, Brasil

- Laboratório de Tecnologias do Futuro - IFES, Vitória, ES, Brasil
- High performance Networks (HPN) - University of Bristol, Bristol, UK

7.2.3 Projetos

O Espaço Inteligente proposto nesse trabalho foi ou está sendo utilizado nos seguintes projetos de pesquisa:

- Construção da Infraestrutura de um Espaço Inteligente para Posicionamento e Controle de Dispositivos Robóticos

Financiador: Fundação de Amparo à Pesquisa e Inovação do Espírito Santo

Período: 2015 - 2017

- FUTEBOL: Federated Union of Telecommunications Research Facilities for an EU-Brasil Open Laboratory

Financiadores: European Union's Horizon 2020 for research, technological development, and demonstration; Ministério da Ciência, Tecnologia, Inovações e Comunicações; Rede Nacional de Ensino e Pesquisa;

Período: 2016 - 2019

- 5GCity - A Distributed Cloud & Radio Platform for 5G Neutral Hosts

Financiador: European Union's Horizon 2020 for research, technological development, and demonstration

Período: 2017 - 2019

- MOBILYSA - Sistema de localização e controle do cão-guia robô Lysa para ambientes internos baseado em visão computacional

Financiador: Rede Nacional de Ensino e Pesquisa

Período: 2019 - 2020

- Cidades Inteligentes e Segurança Pública no Espírito Santo

Financiador: Fundação de Amparo à Pesquisa e Inovação do Espírito Santo

Período: 2019 - 2020

- Videomonitoramento Inteligente Aplicado à Segurança Pública

Financiador: Emenda parlamentar

Período: 2020 - 2021

- Kit de Transformação Digital para prédios Inteligentes

Financiador: SETEC/MEC

Período: 2020 - 2022

- IdCar - Sistemas de identificação única de veículos

Financiador: Fundação de Amparo à Pesquisa e Inovação do Espírito Santo

Período: 2021 - 2022

- Controlador semafórico de grande porte com capacidade para processamento de vídeo embarcado

Financiador: Fundação de Amparo à Pesquisa e Inovação do Espírito Santo

Período: 2021 - 2022

Referências

- 3GPP. *Service requirements for next generation new services and markets*. [S.l.], 2018. Rev. 16.3.0. Citado na página 95.
- ABBAS, I. et al. Issues and Challenges of Cloud Computing in Performance Augmentation for Pervasive Computing. In: *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*. IEEE, 2020. p. 1–7. ISBN 978-1-7281-7116-6. Disponível em: <<https://ieeexplore.ieee.org/document/9179462/>>. Citado na página 21.
- Adduci, M.; Amplianitis, K.; Reulke, R. A quality evaluation of single and multiple camera calibration approaches for an indoor multi camera tracking system. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, p. 9–15, 2014. Citado na página 179.
- AL., I. P. et. A survey on low latency towards 5g: Ran, core network and caching solutions. *Submitted in IEEE Communications Surveys and Tutorials*, <https://arxiv.org/abs/1708.02562v1>, 2017. Citado 3 vezes nas páginas 85, 88 e 90.
- ALBAWENDI, S. et al. Overview of behavioural understanding system with filtered vision sensor. In: *2015 International Conference on Interactive Technologies and Games*. [S.l.: s.n.], 2015. p. 90–95. Citado na página 179.
- ALMONFREY, D. et al. A flexible human detection service suitable for Intelligent Spaces based on a multi-camera network. *International Journal of Distributed Sensor Networks*, SAGE PublicationsSage UK: London, England, v. 14, n. 3, p. 155014771876355, mar 2018. ISSN 1550-1477. Disponível em: <<http://journals.sagepub.com/doi/10.1177/1550147718763550>>. Citado 10 vezes nas páginas 23, 90, 180, 182, 183, 184, 185, 186, 187 e 188.
- ALMONFREY, D. et al. Neural cells insights on pedestrian detection. In: *CBA 2016 - XXI Brazilian Conference of Automation*. [S.l.: s.n.], 2016. <<http://www.swge.inf.br/proceedings/paper/?P=CBA2016-0590>>. Citado na página 183.
- ALSHAER, H. et al. The UK Programmable Fixed and Mobile Internet Infrastructure : Overview , capabilities and use cases deployment. *IEEE Access*, v. 4, p. 1–12, 2020. Citado na página 7.
- AMOON, M.; ALTAMEEM, T.; ALTAMEEM, A. Internet of things sensor assisted security and quality analysis for health care data sets using artificial intelligent based heuristic health management system. *Measurement*, v. 161, p. 107861, 2020. ISSN 0263-2241. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0263224120303997>>. Citado na página 4.
- ANTEQUERA, R. B. et al. ADON: Application-Driven Overlay Network-as-a-Service for Data-Intensive Science. *IEEE Transactions on Cloud Computing*, IEEE, v. 6, n. 3, p. 640–655, 2018. ISSN 2168-7161. Disponível em: <<https://ieeexplore.ieee.org/document/7364221/>>. Citado na página 7.

- ASSIS, L. de. *Cão-Guia treinado no Brasil já é uma realidade*. 2019. Disponível em: <<https://emails.estadao.com.br/blogs/comportamento-animal/cao-guia-treinado-no-brasil-ja-e-uma-realidade/>>. Acesso em: 04 nov. 2020. Citado na página 167.
- ATIS. *ATIS Telecom Glossary*. 2020. Disponível em: <<http://www.atis.org/glossary/>>. Acesso em: 04 nov. 2020. Citado na página 176.
- BALALAIE, A.; HEYDARNOORI, A.; JAMSHIDI, P. Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. *IEEE Software*, v. 33, n. 3, p. 42–52, may 2016. ISSN 1937-4194. Citado na página 49.
- BAUER, M. et al. *Internet of Things-Architecture IoT-A Final architectural reference model for the IoT v3.0*. [S.l.], 2013. Citado 2 vezes nas páginas 32 e 33.
- Ben Sada, A. et al. A Distributed Video Analytics Architecture Based on Edge-Computing and Federated Learning. In: *2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCCom/CyberSciTech)*. [S.l.: s.n.], 2019. p. 215–220. Citado na página 22.
- BENENSON, R. et al. Ten years of pedestrian detection, what have we learned? In: *ECCV*. [S.l.]: Springer, 2015. (Lecture Notes in Computer Science, v. 8926). Citado na página 177.
- BENTO, F. R. O.; VASSALLO, R. F.; SAMATELO, J. L. A. Detecção de Anomalias em Vias Públicas Usando Características Espaciais e um Classificador Sequencial Bidirecional. In: *Anais do XXIII Congresso Brasileiro de Automática*. Santa Maria, RS, Brasil: [s.n.], 2020. Citado na página 18.
- BHARDWAJ, S. et al. Smart space concepts, properties and architectures. *IEEE Access*, Institute of Electrical and Electronics Engineers Inc., v. 6, p. 70088–70112, 2018. ISSN 21693536. Citado na página 21.
- BHATT, D. P.; TIWARI, M. Smart traffic sign boards (STSB) for smart cities. In: *2nd Smart Cities Symposium (SCS 2019)*. [S.l.: s.n.], 2019. p. 1–4. Citado na página 16.
- BLANCO, B. et al. Intelligent Orchestration of End-to-End Network Slices for the Allocation of Mission Critical Services over NFV Architectures. In: MAGLOGIANNIS, I.; ILIADIS, L.; PIMENIDIS, E. (Ed.). *Artificial Intelligence Applications and Innovations. AIAI 2020 IFIP WG 12.5 International Workshops*. Cham: Springer International Publishing, 2020. p. 74–83. ISBN 978-3-030-49190-1. Citado 2 vezes nas páginas 25 e 27.
- BRENA, R. F. et al. Evolution of Indoor Positioning Technologies: A Survey. *Journal of Sensors*, v. 2017, 2017. ISSN 16877268. Citado 3 vezes nas páginas 87, 89 e 169.
- BRSSCIĆ, D. Social robots in smart public environments. In: *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*. [S.l.: s.n.], 2014. p. 651–653. Citado na página 175.
- BURNS, B.; GOOGLE, D. O. *Design patterns for container-based distributed systems*. [S.l.], 2016. Acesso em: 11 out. 2020. Citado na página 43.

- BURNS, B. et al. Borg, omega, and kubernetes. *ACM Queue*, v. 14, p. 70–93, 2016. Disponível em: <<http://queue.acm.org/detail.cfm?id=2898444>>. Citado na página 73.
- CAI, Z. et al. A unified multi-scale deep convolutional neural network for fast object detection. In: *ECCV*. [S.l.: s.n.], 2016. Citado na página 177.
- CARMO, A. do et al. Uso de um Espaço Inteligente Baseado em Visão Computacional para o Controle de Formação de Robôs Móveis. In: *Anais do XII Simpósio Brasileiro de Computação Ubíqua e Pervasiva*. Porto Alegre, RS, Brasil: SBC, 2020. p. 171–180. ISSN 2595-6183. Disponível em: <<https://sol.sbc.org.br/index.php/sbcup/article/view/11223>>. Citado 2 vezes nas páginas 163 e 164.
- CARMO, A. P. et al. Programmable intelligent spaces for Industry 4.0: Indoor visual localization driving attocell networks. *Transactions on Emerging Telecommunications Technologies*, John Wiley & Sons, Ltd, p. e3610, apr 2019. ISSN 2161-3915. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3610>>. Citado 11 vezes nas páginas 86, 92, 93, 96, 98, 100, 101, 102, 104, 105 e 106.
- CARMO, A. P. do et al. Ontologia das coisas para espaços inteligentes baseados em visão computacional. In: BARACHO, R. M. A.; ISOTANI, S.; ALMEIDA, M. B. (Ed.). *ONTOBRAS – Brazilian Ontology Research Seminar (ONTOBRAS)*. Aachen: [s.n.], 2016. (CEUR Workshop Proceedings, 1862), p. 181–186. ISSN 1613-0073. Disponível em: <<http://ceur-ws.org/Vol-1862/#paper-18>>. Citado na página 67.
- CARREZ, F. et al. A Reference Architecture for federating IoT infrastructures supporting semantic interoperability. In: *2017 European Conference on Networks and Communications (EuCNC)*. [S.l.: s.n.], 2017. p. 1–6. Citado na página 32.
- CARROZZO, G. et al. Interoperation of IoT platforms in confined smart spaces: The SymbIoTe smart space architecture. In: *2018 5th International Conference on Internet of Things: Systems, Management and Security, IoTSMMS 2018*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2018. p. 38–45. ISBN 9781538695852. Citado na página 22.
- CERNY, T.; DONAHOO, M. J.; TRNKA, M. Contextual Understanding of Microservice Architecture: Current and Future Directions. *SIGAPP Appl. Comput. Rev.*, Association for Computing Machinery, New York, NY, USA, v. 17, n. 4, p. 29–45, jan 2018. ISSN 1559-6915. Disponível em: <<https://doi.org/10.1145/3183628.3183631>>. Citado na página 11.
- CHEN, H. et al. A hybrid cloud robot framework based on intelligent space. In: *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*. IEEE, 2016. v. 2016-Septe, p. 2996–3001. ISBN 9781467384148. Disponível em: <<http://ieeexplore.ieee.org/document/7578487/>>. Citado 2 vezes nas páginas 6 e 22.
- CHEN, L.; CHEN, T.; CHEN, D. iGuiding: A Mobile Campus Care and Guidance System Based on Internet of Things Technologies. In: *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. [S.l.: s.n.], 2018. p. 436–438. Citado na página 17.
- CHEN, S. et al. A Vision of IoT : Applications , Challenges , and Opportunities With China Perspective. *Internet of Things Journal, IEEE*, v. 1, n. 4, p. 349–359, 2014. ISSN 2327-4662.

Disponível em: <<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6851114>>. Citado na página 31.

CICCONETTI, C.; CONTI, M.; PASSARELLA, A. Low-latency distributed computation offloading for pervasive environments. In: *2019 IEEE International Conference on Pervasive Computing and Communications, PerCom 2019*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2019. ISBN 9781538691489. Citado na página 22.

CLARK, R.; TRIGONI, N.; MARKHAM, A. Robust vision-based indoor localization. In: *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*. New York, NY, USA: ACM, 2015. (IPSN '15), p. 378–379. ISBN 978-1-4503-3475-4. Disponível em: <<http://doi.acm.org/10.1145/2737095.2742929>>. Citado na página 90.

CONTI, A. et al. Network Experimentation for Cooperative Localization. v. 30, n. 2, p. 467–475, 2012. Citado 2 vezes nas páginas 86 e 168.

COOK, D. J. et al. Detection of social interaction in smart spaces. *Cybernetics and systems*, v. 41, n. 2, p. 90–104, 2010. Citado na página 175.

COOK, D. J. et al. Casas: A smart home in a box. *Computer*, v. 46, n. 7, p. 62–69, 2013. Citado na página 175.

COSMA, A.; RADOI, I. E.; RADU, V. CamLoc: Pedestrian Location Estimation through Body Pose Estimation on Smart Cameras. In: *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. [S.l.: s.n.], 2019. p. 1–8. ISSN 2471-917X. Citado na página 4.

COTTA, W. A. A. *Medição de tempo de comunicação e exibição de tempo de resposta para Espaços Inteligentes Programáveis*. 88 p. Dissertação (Mestrado em Engenharia Elétrica) — Engenharia Elétrica, Universidade Federal do Espírito Santo, Vitória, 2020. Citado 2 vezes nas páginas 70 e 71.

✓CUROVÁ, D. et al. Intelligent space at center for intelligent technologies - System proposal. *SAMI 2017 - IEEE 15th International Symposium on Applied Machine Intelligence and Informatics, Proceedings*, IEEE, p. 191–195, 2017. Citado na página 22.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2005. v. 1, p. 886–893 vol. 1. Citado na página 183.

DANG, X. T.; KHAN, M. A.; SIVRIKAYA, F. An Autonomous Service-Oriented Orchestration Framework for Software Defined Mobile Networks. In: *Proceedings of the 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops, ICIN 2019*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2019. p. 277–284. ISBN 9781538683361. Citado na página 25.

DARDARI, D.; CLOSAS, P.; DJURIĆ, P. M. Indoor tracking: Theory, methods, and technologies. *IEEE Transactions on Vehicular Technology*, v. 64, n. 4, p. 1263–1278, April 2015. ISSN 0018-9545. Citado na página 89.

Dardari, D.; Closas, P.; Djurić, P. M. Indoor tracking: Theory, methods, and technologies. *IEEE Transactions on Vehicular Technology*, v. 64, n. 4, p. 1263–1278, 2015. Citado na página 169.

- DEBAUCHE, O. et al. A new Edge Architecture for AI-IoT services deployment. *Procedia Computer Science*, Elsevier BV, v. 175, p. 10–19, jan 2020. ISSN 18770509. Citado na página 25.
- Di Taranto, R. et al. Location-aware communications for 5g networks: How location information can improve scalability, latency, and robustness of 5g. *IEEE Signal Processing Magazine*, v. 31, n. 6, p. 102–112, 2014. Citado 2 vezes nas páginas 86 e 90.
- DOCKER. *Orientation and setup / Docker Documentation*. 2020. Disponível em: <<https://docs.docker.com/get-started/>>. Acesso em: 02 nov. 2020. Citado na página 58.
- DOLLÁR, P. et al. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 36, p. 1532–1545, 2014. Citado na página 177.
- DU, X. et al. Fused dnn: A deep neural network fusion approach to fast and robust pedestrian detection. In: *arXiv*. [S.l.: s.n.], 2016. Citado na página 177.
- EL-KASSABI, H. T. et al. Trust enforcement through self-adapting cloud workflow orchestration. *Future Generation Computer Systems*, Elsevier B.V., v. 97, p. 462–481, aug 2019. ISSN 0167739X. Citado na página 24.
- ELYAMANY, H. F.; ALKHAIRI, A. H. IoT-academia architecture: A profound approach. In: *2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. IEEE, 2015. p. 1–5. ISBN 978-1-4799-8676-7. Disponível em: <<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7176275>>. Citado na página 32.
- ESCOBEDO-CARDENAS, E.; CAMARA-CHAVEZ, G. A robust gesture recognition using hand local data and skeleton trajectory. In: IEEE. *2015 IEEE International Conference on Image Processing (ICIP)*. [S.l.], 2015. p. 1240–1244. Citado na página 110.
- ESPOSITO, C.; CASTIGLIONE, A.; CHOO, K. R. Challenges in Delivering Software in the Cloud as Microservices. *IEEE Cloud Computing*, v. 3, n. 5, p. 10–14, 2016. ISSN 2325-6095. Citado na página 49.
- ETSI NFV ISG. *GS NFV 002 - V1.2.1 - Network Functions Virtualisation (NFV); Architectural Framework*. [S.l.], 2014. v. 1, 1–21 p. Citado na página 34.
- FICHERA, S. et al. Latency-Aware Resource Orchestration in SDN-Based Packet Over Optical Flexi-Grid Transport Networks. *J. Opt. Commun. Netw.*, OSA, v. 11, n. 4, p. B83–B96, apr 2019. Disponível em: <<http://jocn.osa.org/abstract.cfm?URI=jocn-11-4-B83>>. Citado na página 27.
- FOWLER, M.; LEWIS, J. *Microservices a definition of this new architectural term*. MartinFowler.com, 2014. Disponível em: <<http://martinfowler.com/articles/microservices.html>>. Acesso em: 06 out. 2020. Citado na página 11.
- FRANCESCON, A. et al. X-MANO: Cross-domain management and orchestration of network services. In: *2017 IEEE Conference on Network Softwarization (NetSoft)*. [S.l.: s.n.], 2017. p. 1–5. Citado na página 49.

FRAUNDORFER, F.; SCARAMUZZA, D. Visual odometry: Part ii: Matching, robustness, optimization, and applications. *IEEE Robotics & Automation Magazine*, IEEE, v. 19, n. 2, p. 78–90, 2012. Citado na página 161.

GALIS, A. et al. Management and Service-aware Networking Architectures (MANA) for future internet: Position paper: System functions, capabilities and requirements. In: *2009 4th International Conference on Communications and Networking in China, CHINACOM 2009*. [S.l.: s.n.], 2009. p. 28–40. ISBN 9781424443376. Citado na página 46.

GANGANATH, N.; LEUNG, H. Mobile robot localization using odometry and kinect sensor. In: *2012 IEEE International Conference on Emerging Signal Processing Applications*. [S.l.]: IEEE, 2012. p. 91–94. ISBN 978-1-4673-0898-4. Citado na página 161.

GARRIDO-JURADO, S. et al. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, v. 47, n. 6, p. 2280 – 2292, 2014. ISSN 0031-3203. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0031320314000235>>. Citado na página 93.

GEIMER, M. et al. A generic and configurable source-code instrumentation component. In: ALLEN, G. et al. (Ed.). *Computational Science – ICCS 2009*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 696–705. ISBN 978-3-642-01973-9. Citado na página 71.

GHOSH, P.; NGUYEN, Q.; KRISHNAMACHARI, B. Container orchestration for dispersed computing. In: *WOC 2019 - Proceedings of the 2019 5th International Workshop on Container Technologies and Container Clouds, Part of Middleware 2019*. New York, New York, USA: Association for Computing Machinery, Inc, 2019. p. 19–24. ISBN 9781450370332. Disponível em: <<http://dl.acm.org/citation.cfm?doid=3366615.3368354>>. Citado na página 25.

GIRSHICK, R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.]: IEEE Computer Society, 2014. p. 580–587. Citado 2 vezes nas páginas 175 e 177.

GLAS., D. F. et al. Human-robot interaction in public and smart spaces. *Intelligent Assistive Robots: Recent Advances in Assistive Robotics for Everyday Activities*, Springer International Publishing, v. 106, p. 235–273, 2015. Citado 2 vezes nas páginas 175 e 178.

GLAS, D. F. et al. The network robot system: Enabling social human-robot interaction in public spaces. *Journal of Human-Robot Interaction.*, Journal of Human-Robot Interaction Steering Committee, v. 1, n. 2, p. 5–32, 2013. Citado na página 175.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <<http://www.deeplearningbook.org>>. Citado na página 171.

GOOGLE LLC. *Protocol Buffers* / Google Developers. 2020. Disponível em: <<https://developers.google.com/protocol-buffers>>. Acesso em: 02 nov. 2020. Citado na página 54.

GRAFANA LABS. *Grafana Features* / Grafana Labs. 2020. Disponível em: <<https://grafana.com/grafana/>>. Acesso em: 02 nov. 2020. Citado na página 73.

- HASHICORP. *Introduction - Terraform by HashiCorp*. 2020. Disponível em: <<https://www.terraform.io/intro/index.html>>. Acesso em: 02 nov. 2020. Citado na página 77.
- HE, K. et al. Mask r-cnn. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2017. Citado na página 171.
- HELAL, S. *Programming pervasive spaces*. 2005. 84–87 p. Citado na página 26.
- HOSANG, J. et al. Taking a deeper look at pedestrians. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2015. p. 4073–4082. Citado na página 177.
- IBGE. *Características gerais da população, religião e pessoas com deficiência*. [S.l.], 2010. Disponível em: <https://biblioteca.ibge.gov.br/visualizacao/periodicos/94/cd_2010_religiao_deficiencia.pdf>. Acesso em: 04 nov. 2020. Citado na página 167.
- ILLINGWORTH, A. J. et al. How Can Existing Ground-Based Profiling Instruments Improve European Weather Forecasts?. *Bulletin of the American Meteorological Society*, v. 100, n. 4, p. 605–619, 2019. ISSN 00030007. Citado na página 4.
- ITU-T. *ITU-T Recommendation Y.2060 : Overview of the Internet of things*. [S.l.], 2012. Citado 2 vezes nas páginas 31 e 32.
- ITU-T. *ITU-T Recommendation Y.2063 : Framework of the web of things*. [S.l.], 2012. Citado na página 32.
- ITU-T. *ITU-T Recommendation Y.2075 : Capability framework for e-health monitoring services*. [S.l.], 2015. Citado na página 32.
- ITU-T. *ITU-T Recommendation Y.4457 : Architectural framework for transportation safety services*. [S.l.], 2018. Citado na página 32.
- KABUTAN, R.; NISHIDA, T. Development of robotic intelligent space using multiple RGB-D cameras for industrial robots. n. September, 2016. Disponível em: <<https://www.researchgate.net/publication/308413148>>. Citado na página 21.
- KEHOE, B. et al. A survey of research on cloud robotics and automation. *IEEE Transactions on automation science and engineering*, IEEE, v. 12, n. 2, p. 398–409, 2015. Citado na página 85.
- KLEIN, M. *Lyft's Envoy: Experiences Operating a Large Service Mesh*. 2017. Acesso em: 11 out. 2020. Citado na página 43.
- KORZUN, D. G.; GALOV, I. V.; LOMOV, A. A. Smart space deployment in wireless and mobile settings of the Internet of Things. *2016 IEEE 3rd International Symposium on Wireless Systems within the IEEE International Conferences on Intelligent Data Acquisition and Advanced Computing Systems, IDAACS-SWS 2016 - Proceedings*, IEEE, n. September, p. 86–91, 2017. Citado na página 27.
- KUBERNETES. *Production-Grade Container Orchestration*. 2020. Disponível em: <<https://kubernetes.io/>>. Acesso em: 02 nov. 2020. Citado na página 73.

KUMRAI, T. et al. Human Activity Recognition with Deep Reinforcement Learning using the Camera of a Mobile Robot. In: *2020 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. [S.l.: s.n.], 2020. p. 1–10. ISSN 2474-249X. Citado na página 17.

LEE, H. et al. *BumbleBee: Application-aware adaptation for container orchestration*. 2020. Citado 2 vezes nas páginas 25 e 30.

LEE, J.; YOO, Y. Handover cell selection using user mobility information in a 5g sdn-based network. In: *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*. [S.l.: s.n.], 2017. p. 697–702. Citado na página 90.

LEE, J.-E. et al. Human and robot localization using histogram of oriented gradients (hog) feature for an active information display in intelligent space. *Advanced Science Letters*, v. 5, p. 1–8, 2012. Citado 2 vezes nas páginas 175 e 179.

LEE, S.; SHIN, I.; LEE, N. Development of Intelligent Space Lighting System. In: *ICTC 2019 - 10th International Conference on ICT Convergence: ICT Convergence Leading the Autonomous Future*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2019. p. 1030–1032. ISBN 9781728108926. Citado na página 21.

LI, B.; YAO, Q.; WANG, K. A review on vision-based pedestrian detection in intelligent transportation systems. In: *Proceedings of 2012 9th IEEE International Conference on Networking, Sensing and Control*. [S.l.: s.n.], 2012. p. 393–398. Citado 2 vezes nas páginas 175 e 177.

LIBERATO, A. et al. Dynamic backhauling within converged networks. In: *Proceedings of the 2016 Workshop on Fostering Latin-American Research in Data Communication Networks*. New York, NY, USA: Association for Computing Machinery, 2016. (LANCOMM '16), p. 31–33. ISBN 9781450344265. Disponível em: <<https://doi.org/10.1145/2940116.2940122>>. Citado 2 vezes nas páginas 86 e 90.

LINUX FOUNDATION. *Open vSwitch*. 2020. Disponível em: <<https://www.openvswitch.org/>>. Acesso em: 02 nov. 2020. Citado na página 67.

LIU, X. et al. Hidden states exploration for 3d skeleton-based gesture recognition. In: *IEEE. 2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. [S.l.], 2019. p. 1846–1855. Citado na página 110.

LU, Y. Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, Elsevier, v. 6, p. 1–10, 2017. Citado na página 85.

MANSO, C. et al. End-to-end SDN / NFV orchestration of multi-domain transport networks and distributed computing infrastructure for beyond-5G services. *IEICE Transactions on Communications*, 2020. Citado na página 27.

MAO, J. et al. What can help pedestrian detection? In: *CVPR*. [S.l.: s.n.], 2017. Citado na página 177.

MARTINELLO, M. et al. Programmable residues defined networks for edge data centres. In: *2017 13th International Conference on Network and Service Management, CNSM 2017*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2017. v. 2018-January, p. 1–9. ISBN 9783901882982. Citado na página 38.

- MARTINEZ, V. M. G. et al. Ultra reliable communication for robot mobility enabled by sdn splitting of wifi functions. In: *IEEE ISCC*. [S.l.: s.n.], 2018. Citado 2 vezes nas páginas 94 e 95.
- MATSUHIRA, N. et al. Development of robotic transportation system - shopping support system collaborating with environmental cameras and mobile robots -. In: *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*. [S.l.: s.n.], 2010. p. 1–6. Citado na página 178.
- MAYER, A.; MANSFIELD, S. *The Third Network: Lifecycle Service Orchestration Vision, Technical Report*. [S.l.], 2015. Disponível em: <http://www.mef.net/Assets/White_Papers/MEF_Third_Network_LSO_Vision_FINAL.pdf>. Citado na página 46.
- MI, J.; TAKAHASHI, Y. Performance analysis of mobile robot self-localization based on different configurations of RFID system. In: *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. [S.l.]: IEEE, 2015. p. 1591–1596. ISBN 978-1-4673-9107-8. Citado na página 161.
- MICHAELA, Z.; HORÁK, T. Smart Cities and Quality of Life perception in the Czech Republic. In: *2020 Smart City Symposium Prague (SCSP)*. [S.l.: s.n.], 2020. p. 1–5. Citado na página 16.
- MITRA, S.; ACHARYA, T. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, IEEE, v. 37, n. 3, p. 311–324, 2007. Citado na página 109.
- MORIOKA, K.; HASHIKAWA, F.; TAKIGAWA, T. Human identification based on walking detection with acceleration sensor and networked laser range sensors in intelligent space. *International Journal on Smart Sensing and Intelligent Systems*, v. 6, n. 5, p. 2040–2054, 2013. Citado na página 175.
- MRAZOVAC, B. et al. System design for passive human detection using principal components of the signal strength space. In: *2012 IEEE 19th International Conference and Workshops on Engineering of Computer-Based Systems*. [S.l.: s.n.], 2012. p. 164–172. Citado na página 175.
- MUPPIRISETTY, L. S.; YIU, S.; WYMEERSCH, H. Lapra: Location-aware proactive resource allocation. In: *2016 IEEE Global Communications Conference (GLOBECOM)*. [S.l.: s.n.], 2016. p. 1–6. Citado na página 89.
- O’KANE, J. M. Global localization using odometry. In: IEEE. *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. [S.l.], 2006. p. 37–42. Citado na página 161.
- OLIVEIRA, F. et al. Delivering software with agility and quality in a cloud environment. *IBM Journal of Research and Development*, v. 60, n. 2-3, p. 10:1–10:11, mar 2016. ISSN 0018-8646. Citado na página 49.
- OPENSTACK FOUNDATION. *Open Source Cloud Computing Infrastructure - OpenStack*. 2020. Disponível em: <<https://www.openstack.org/>>. Acesso em: 11 out. 2020. Citado na página 48.

- OPENSTACK FOUNDATION. *OpenStack Docs: OpenStack Compute (heat)*. 2020. Disponível em: <<https://docs.openstack.org/heat/latest/>>. Acesso em: 11 out. 2020. Citado na página 48.
- OPENSTACK FOUNDATION. *OpenStack Docs: OpenStack Compute (neutron)*. 2020. Disponível em: <<https://docs.openstack.org/neutron/latest/>>. Acesso em: 11 out. 2020. Citado na página 48.
- OPENSTACK FOUNDATION. *OpenStack Docs: OpenStack Compute (nova)*. 2020. Disponível em: <<https://docs.openstack.org/nova/latest/>>. Acesso em: 11 out. 2020. Citado 2 vezes nas páginas 48 e 77.
- OPENTRACING. *The OpenTracing data model specification*. 2020. Disponível em: <<https://github.com/opentracing/specification/blob/master/specification.md>>. Acesso em: 02 nov. 2020. Citado na página 70.
- OPENTRACING. *What is Distributed Tracing?* 2020. Disponível em: <<https://opentracing.io/docs/overview/what-is-tracing/>>. Acesso em: 02 nov. 2020. Citado na página 70.
- PAHL, M. O.; CARLE, G.; KLINKER, G. Distributed smart space orchestration. In: *Proceedings of the NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2016. p. 979–984. ISBN 9781509002238. Citado na página 22.
- PICORETI, R. et al. Multilevel Observability in Cloud Orchestration. In: *16th Int. Conf. on Pervasive Intelligence & Comp.* [S.l.: s.n.], 2018. p. 776–784. ISBN 9781538675182. Citado 3 vezes nas páginas 23, 76 e 102.
- POSADA, J. et al. Visual computing as a key enabling technology for industrie 4.0 and industrial internet. *IEEE computer graphics and applications*, IEEE, v. 35, n. 2, p. 26–40, 2015. Citado na página 87.
- PROMETHEUS. *Overview / Prometheus*. 2020. Disponível em: <<https://prometheus.io/docs/introduction/overview/>>. Acesso em: 02 nov. 2020. Citado na página 72.
- QUEIROZ, F. M. et al. Estimating tridimensional coordinates of skeleton joints in a multicamera system. In: *Anais do XIV Workshop de Visão Computacional - WVC 2018*. [S.l.: s.n.], 2018. p. 108–114. Citado 3 vezes nas páginas 110, 112 e 113.
- QUEIROZ, F. M. de. *Desenvolvimento da Infraestrutura de um Espaço Inteligente baseado em Visão Computacional e IoT*. 80 p. Monografia (Graduação) — Engenharia elétrica, Universidade Federal do Espírito Santo, Vitória, 2016. Citado 5 vezes nas páginas 61, 62, 63, 68 e 69.
- RABBITMQ. *Messaging that just works — RabbitMQ*. 2020. Disponível em: <<https://www.rabbitmq.com/>>. Acesso em: 02 nov. 2020. Citado na página 60.
- RAMPINELLI, M. et al. An intelligent space for mobile robot localization using a multi-camera system. *Sensors (Basel, Switzerland)*, v. 14, n. 8, p. 15039–15064, 2014. ISSN 14248220. Citado 2 vezes nas páginas 90 e 162.

- RAZAVI, M.; HAMIDKHANI, M.; SADEGHI, R. Smart Traffic Light Scheduling in Smart City Using Image and Video Processing. In: *2019 3rd International Conference on Internet of Things and Applications (IoT)*. [S.l.: s.n.], 2019. p. 1–4. Citado na página 17.
- REDMON, J.; FARHADI, A. *YOLOv3: An Incremental Improvement*. 2018. Citado 2 vezes nas páginas 171 e 177.
- Ren, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 39, n. 6, p. 1137–1149, 2017. Citado 2 vezes nas páginas 171 e 177.
- RIBEIRO, D. et al. A real-time deep learning pedestrian detector for robot navigation. In: *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. [S.l.: s.n.], 2017. p. 165–171. Citado na página 177.
- RYU. *Ryu SDN Framework*. 2020. Disponível em: <<https://ryu-sdn.org/>>. Acesso em: 02 nov. 2020. Citado na página 77.
- SANTOS, C. A. S.; NETO, A. N. R.; SALEME, E. B. An event driven approach for integrating multi-sensory effects to interactive environments. In: IEEE. *2015 IEEE International Conference on Systems, Man, and Cybernetics*. [S.l.], 2015. p. 981–986. Citado na página 110.
- SANTOS, C. C. dos. *Proposta de uma metodologia para a obtenção de vocabulários de gestos intuitivos para a interação homem-robô*. 164 f. Dissertação (Mestrado) — (Pós-Graduação em Ciência da Computação) - Universidade Federal de Sergipe, 2016. Citado na página 109.
- SANTOS, C. C. dos et al. Action Anticipation for Collaborative Environments: The Impact of Contextual Information and Uncertainty-Based Prediction. *NEUROCOMPUTING*, 2020. Citado na página 18.
- Saraiva de Sousa, N. F. et al. Network Service Orchestration: A survey. *Computer Communications*, v. 142-143, p. 69–94, 2019. ISSN 0140-3664. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0140366418309502>>. Citado 2 vezes nas páginas 46 e 48.
- SCARAMUZZA, D.; FRAUNDORFER, F. Visual odometry [tutorial]. *IEEE robotics & automation magazine*, IEEE, v. 18, n. 4, p. 80–92, 2011. Citado na página 161.
- SHANMUGAMANI, R. *Deep Learning for Computer Vision: Expert Techniques to Train Advanced Neural Networks Using TensorFlow and Keras*. Packt Publishing, 2018. ISBN 9781523116751. Disponível em: <<https://books.google.com.br/books?id=nlifwgEACAAJ>>. Citado na página 171.
- SIGELMAN, B. H. et al. *Dapper , a Large-Scale Distributed Systems Tracing Infrastructure*. [S.l.], 2010. 14 p. Disponível em: <<https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/36356.pdf>>. Acesso em: 11 out. 2020. Citado 3 vezes nas páginas 43, 44 e 50.
- SOMOV, A. et al. Pervasive Agriculture: IoT-Enabled Greenhouse for Plant Growth Control. *IEEE Pervasive Computing*, v. 17, n. 4, p. 65–75, 2019. ISSN 1536-1268. Citado na página 4.

SONKOLY, B. et al. 5G Applications from Vision to Reality: Multi-Operator Orchestration. *IEEE Journal on Selected Areas in Communications*, Institute of Electrical and Electronics Engineers Inc., v. 38, n. 7, p. 1401–1416, jul 2020. ISSN 15580008. Citado 2 vezes nas páginas 25 e 29.

SPRUTE, D. et al. Gesture-Based Object Localization for Robot Applications in Intelligent Environments. In: *Proceedings - 2018 International Conference on Intelligent Environments, IE 2018*. [S.l.: s.n.], 2018. p. 48–55. ISBN 9781538668443. Citado 2 vezes nas páginas 6 e 21.

SRIVASTAVA, S.; BISHT, A.; NARAYAN, N. Safety and security in smart cities using artificial intelligence — A review. In: *2017 7th International Conference on Cloud Computing, Data Science Engineering - Confluence*. [S.l.: s.n.], 2017. p. 130–133. Citado na página 17.

STOYANOV, S.; OROZOVA, D.; POPCHEV, I. Internet of things water monitoring for a smart seaside city. In: *2018 20th International Symposium on Electrical Apparatus and Technologies, SIELA 2018 - Proceedings*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2018. ISBN 9781538634196. Citado na página 21.

ŞUCAN, I. A.; MOLL, M.; KAVRAKI, L. E. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, v. 19, n. 4, p. 72–82, December 2012. <<http://ompl.kavrakilab.org>>. Citado na página 186.

SURIE, D.; PARTONIA, S.; LINDGREN, H. Human sensing using computer vision for personalized smart spaces. In: *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*. [S.l.: s.n.], 2013. p. 487–494. Citado 3 vezes nas páginas 175, 178 e 188.

TARNEBERG, W. et al. Utilizing massive mimo for the tactile internet: Advantages and trade-offs. In: *2017 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*. [S.l.: s.n.], 2017. p. 1–6. Citado na página 88.

TAUQIR, H. P.; HABIB, A. Integration of IoT and smart grid to reduce line losses. In: *2019 2nd International Conference on Computing, Mathematics and Engineering Technologies, iCoMET 2019*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2019. ISBN 9781538695098. Citado na página 21.

THIELENS, A. et al. Radiofrequency exposure near an attocell as part of an ultra-high density access network. *Bioelectromagnetics*, v. 38, n. 4, p. 295–306, 2017. ISSN 1521-186X. Disponível em: <<http://dx.doi.org/10.1002/bem.22045>>. Citado na página 88.

THINAKARAN, P. et al. Kube-Knots: Resource Harvesting through Dynamic Container Orchestration in GPU-based Datacenters. In: *Proceedings - IEEE International Conference on Cluster Computing, ICC*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2019. v. 2019-Sept. ISBN 9781728147345. ISSN 15525244. Citado na página 27.

TOSKOV, B. et al. Architecture of Intelligent Guard System in the Virtual Physical Space. In: *2020 IEEE 10th International Conference on Intelligent Systems (IS)*. IEEE, 2020. p. 265–269. ISBN 978-1-7281-5456-5. Disponível em: <<https://ieeexplore.ieee.org/document/9200177/>>. Citado na página 21.

TSAI, T. H.; CHANG, C. H.; CHEN, S. W. Vision based indoor positioning for intelligent buildings. In: *2016 2nd International Conference on Intelligent Green Building and Smart Grid (IGBSG)*. [S.l.: s.n.], 2016. p. 1–4. Citado na página 90.

TSONEV, D.; VIDEV, S.; HAAS, H. Light fidelity (Li-Fi): towards all-optical networking. In: DINGEL, B. B.; TSUKAMOTO, K. (Ed.). *Broadband Access Communication Technologies VIII*. SPIE, 2014. v. 9007, p. 1 – 10. Disponível em: <<https://doi.org/10.1117/12.2044649>>. Citado na página 88.

UNIYAL, N. et al. 5GUK Exchange: Towards sustainable end-to-end multi-domain orchestration of softwarized 5G networks. *Computer Networks*, v. 178, p. 107297, 2020. ISSN 1389-1286. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1389128619316287>>. Citado na página 27.

VAUGHAN-NICHOLS, S. J. OpenFlow: The Next Generation of the Network? *Computer, Institute of Electrical and Electronics Engineers (IEEE)*, v. 44, n. 8, p. 13–15, aug 2011. ISSN 0018-9162. Citado na página 46.

Vega-Barbas, M. et al. Interaction patterns for smart spaces: A confident interaction design solution for pervasive sensitive iot services. *IEEE Access*, v. 6, p. 1126–1136, 2018. Citado na página 109.

VIXSYSTEM. *Quem é Lysa?* 2020. Disponível em: <<http://www.caoguiarobo.com.br/>> Acesso em: 04 nov. 2020. Citado na página 168.

WANG, X. Intelligent multi-camera video surveillance: A review. *Pattern Recognition Letters*, v. 34, n. 1, p. 3 – 19, 2013. ISSN 0167-8655. Extracting Semantics from Multi-Spectrum Video. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S016786551200219X>>. Citado na página 162.

WANG, X. et al. *Convergence of Edge Computing and Deep Learning: A Comprehensive Survey*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2020. 869–904 p. Citado na página 22.

WANG, X.; JIN, Z. *An Overview of Mobile Cloud Computing for Pervasive Healthcare*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2019. 66774–66791 p. Citado na página 22.

WANG, Y. et al. Automated Student Engagement Monitoring and Evaluation during Learning in the Wild. In: *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. [S.l.: s.n.], 2020. p. 270–275. Citado na página 17.

WEAVEWORKS. *Introducing Weave Net*. 2020. Disponível em: <<https://www.weave.works/docs/net/latest/overview/>>. Acesso em: 02 nov. 2020. Citado na página 64.

WEI, S.-E. et al. Convolutional pose machines. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 4724–4732, 2016. Citado na página 112.

WEISER, M. The Computer for the 21st Century. *SIGMOBILE Mob. Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 3, n. 3, p. 3–11, 1999. ISSN 1559-1662. Disponível em: <<http://doi.acm.org/10.1145/329124.329126>>. Citado na página 3.

WIN, M. Z.; GIFFORD, W. M.; THOMAS, I. B. M. Network Localization and Navigation via Cooperation. n. May, p. 56–62, 2011. Citado 3 vezes nas páginas 86, 89 e 168.

WIN, M. Z. et al. Efficient Multisensor Localization for the Internet of Things: Exploring a New Class of Scalable Localization Algorithms. *IEEE Signal Processing Magazine*, IEEE, v. 35, n. 5, p. 153–167, 2018. ISSN 15580792. Citado 2 vezes nas páginas 86 e 89.

Win, M. Z. et al. Efficient multisensor localization for the internet of things: Exploring a new class of scalable localization algorithms. *IEEE Signal Processing Magazine*, v. 35, n. 5, p. 153–167, 2018. Citado na página 168.

WOLLSCHLAEGER, M.; SAUTER, T.; JASPERNEITE, J. The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0. *IEEE Industrial Electronics Magazine*, IEEE, v. 11, n. 1, p. 17–27, 2017. Citado na página 85.

XU, S.; CHOU, W. An Improved Indoor Localization Method for Mobile Robot Based on WiFi Fingerprint and AMCL. *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, p. 324–329, 2017. Disponível em: <http://ieeexplore.ieee.org/document/8275781/>. Citado na página 89.

XU, Y.; HELAL, A. Scalable Cloud–Sensor Architecture for the Internet of Things. *Internet of Things Journal*, IEEE, v. 3, n. 3, p. 285–298, 2016. Citado na página 27.

YAMAZAKI, Y. et al. Home Atmosphere Visualization Based on Motion Sensing in Smart Living Room. In: *2020 IEEE 2nd Global Conference on Life Sciences and Technologies (LifeTech)*. [S.l.: s.n.], 2020. p. 280–281. Citado na página 16.

YASSIN, A. et al. Recent Advances in Indoor Localization: A Survey on Theoretical Approaches and Applications. *IEEE Communications Surveys & Tutorials*, v. 19, n. 2, p. 1327–1346, 2017. ISSN 1553-877X. Disponível em: <http://ieeexplore.ieee.org/document/7762095/>. Citado 2 vezes nas páginas 85 e 89.

YASSIN, A. et al. Recent Advances in Indoor Localization: A Survey on Theoretical Approaches and Applications. *IEEE Communications Surveys & Tutorials*, v. 19, n. 2, p. 1327–1346, 2017. ISSN 1553-877X. Disponível em: <http://ieeexplore.ieee.org/document/7762095/>. Citado na página 168.

YUAN, L. Study on the design of car-sharing parking space in the background of smart city-a case study of suzhou. In: *Proceedings - 2019 International Conference on Robots and Intelligent System, ICRIS 2019*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2019. p. 382–385. ISBN 9781728126326. Citado na página 21.

ZABULIS, X. et al. Multicamera human detection and tracking supporting natural interaction with large-scale displays. *Machine Vision and Applications*, v. 24, n. 2, p. 319–336, Feb 2013. Citado na página 175.

ZANELLA, a. et al. Internet of Things for Smart Cities. *IEEE Internet of Things Journal*, v. 1, n. 1, p. 22–32, 2014. ISSN 2327-4662. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6740844>. Citado na página 31.

ZEHL, S.; ZUBOW, A.; WOLISZ, A. BIGAP – A Seamless Handover Scheme for High Performance Enterprise IEEE 802.11 Networks. In: *15th IEEE/IFIP Network Operations and Management Symposium*. [S.l.: s.n.], 2016. p. 1015–1016. ISBN 9781509002238.

Citado na página 107.

ZHANG, H.-B. et al. A comprehensive survey of vision-based human action recognition methods. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 19, n. 5, p. 1005, 2019.

Citado na página 109.

ZHANG, L. et al. Is faster r-cnn doing well for pedestrian detection? In: *ECCV*. [S.l.: Springer, 2016. (Lecture Notes in Computer Science, v. 9906), p. 443–457. Citado na página 177.

ZHANG, S.; BENENSON, R.; SCHIELE, B. Filtered channel features for pedestrian detection. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2015. p. 1751–1760. Citado na página 177.

ZHANG, S.; BENENSON, R.; SCHIELE, B. Citypersons: A diverse dataset for pedestrian detection. In: *CVPR*. [S.l.: s.n.], 2017. Citado na página 177.

ZHANG, W. et al. Hetero-Edge: Orchestration of Real-time Vision Applications on Heterogeneous Edge Clouds. In: *Proceedings - IEEE INFOCOM*. [S.l.: Institute of Electrical and Electronics Engineers Inc., 2019. v. 2019-April, p. 1270–1278. ISBN 9781728105154. ISSN 0743166X. Citado na página 25.

ZHOU, M. et al. Robust Neighborhood Graphing for Semi-supervised Indoor Localization with Light-loaded Location Fingerprinting. *IEEE Internet of Things Journal*, v. 4662, n. c, p. 1–1, 2017. ISSN 2327-4662. Disponível em: <http://ieeexplore.ieee.org/document/8115100/>. Citado na página 89.

ZHOU, Z. et al. Towards omnidirectional passive human detection. In: *2013 Proceedings IEEE INFOCOM*. [S.l.: s.n.], 2013. p. 3057–3065. Citado na página 175.

ZIPKIN. *OpenZipkin · A distributed tracing system*. 2020. Disponível em: <https://zipkin.io/>. Acesso em: 02 nov. 2020. Citado na página 71.

Anexos

ANEXO A – Aplicação de Controle de Formação de Robôs Móveis

Saber a postura precisa de um robô em seu ambiente de trabalho é um dos problemas centrais da robótica móvel, pois a tarefa a ser realizada pelo mesmo pode ser melhor executada caso tais informações sejam conhecidas.

A fim de solucionar esse problema, comumente utiliza-se a odometria, método que calcula a posição e orientação de um robô móvel por meio da integração no tempo dos giros de suas rodas (O’KANE, 2006). Apesar de ser muito utilizada em várias aplicações robóticas, a odometria acaba acumulando erros ao decorrer do tempo. Isso ocorre devido à imprecisão dos sensores utilizados na medição e a situações que causam deslizamentos das rodas, como a falta de atrito com o solo ou superfícies irregulares, por exemplo, em que a localização é atualizada sem que o robô tenha se movido. Dessa maneira, quanto mais tempo o robô estiver movimentando-se, maior será o erro em sua postura.

Para diminuir esse erro e melhorar a estimativa da postura de robôs móveis, além dos odômetros, geralmente a informação de outros sensores é utilizada. Em (MI; TAKAHASHI, 2015), por exemplo, foi utilizado um sistema composto por vários leitores de identificadores de rádio frequência (RFID, do inglês Radio Frequency IDentification) colocados em posições conhecidas do ambiente, e por uma etiqueta RFID sobre o robô. Esse sistema permite determinar sua postura do robô com grande precisão. Em (GANGANATH; LEUNG, 2012), um sensor de imagem e profundidade (RGB-D, do inglês Red Green Blue Depth) foi adicionado ao robô para identificar *landmarks* no ambiente, e, com essa informação, minimizou-se a incerteza na localização do robô. Contudo, ambos os trabalhos exigem que sejam adicionados novos dispositivos ao ambiente e ao próprio robô. Isso pode tornar-se um problema quando se tem um grande número de robôs trabalhando no ambiente, ou ainda, quando a plataforma robótica não possibilita a inclusão de novos dispositivos.

Dessa forma, é necessário utilizar uma abordagem que ofereça a possibilidade de localizar dispositivos robóticos no ambiente, de maneira a diminuir a dependência de configuração do espaço físico e/ou do próprio dispositivo. Assim, por se basear em informações fornecidas por uma ou mais câmeras presentes no ambiente para estimar a posição e orientação do robô (FRAUNDORFER; SCARAMUZZA, 2012), a odometria visual pode ser uma candidata à resolução do problema de localização. Uma das vantagens na utilização dessa abordagem é que o erro na estimativa da postura é relativo apenas à qualidade da calibração de cada câmera e, conseqüentemente, não se acumula no tempo (SCARAMUZZA; FRAUNDORFER, 2011), o que é importante para a realização de tarefas

complexas. Por outro lado, se a calibração do sistema visual não for bem estimada, o erro pode ser consideravelmente grande para a realização de atividades que exigem mais precisão de posicionamento.

Além disso, nos últimos anos o videomonitoramento vem sendo utilizado em diversas áreas, como segurança residencial, monitoramento de tráfego, de pacientes hospitalizados, de estações de trem, dentre outros (WANG, 2013). Com isso, em ambientes que já possuem uma infraestrutura de câmeras instaladas com a finalidade de videomonitoramento, é possível utilizar tal infraestrutura para a execução de tarefas que requerem a localização de dispositivos sem muitas modificações. No entanto, para que a odometria visual possa ser melhor empregada, é necessário que as câmeras estejam bem calibradas e que o sistema como um todo realize o processamento das imagens e identificação das características relevantes.

Em Espaços Inteligentes, os robôs podem ser considerados sensores do ambiente e, por isso, podem coletar dados que ajudam nas tomadas de decisão. Seguindo uma linha diferente sobre o papel dos robôs nesses tipos de espaços, (RAMPINELLI et al., 2014) descreve que eles, além de serem sensores presentes no ambiente, também podem ser atuadores no mesmo. Dessa forma, o espaço é capaz de não só adquirir informações fornecidas pelos robôs, mas também de gerenciá-los na realização de tarefas.

No PIS implementado no UFES, o robô trabalha apenas em uma superfície plana, devido a isso, a odometria visual pode ser estimada por uma única câmera. No entanto, quanto maior for o número de câmeras, melhor será a estimativa da postura do robô e maior será a área de cobertura onde o mesmo poderá atuar. Por esse motivo, para o controle dos robôs serão utilizadas as quatro câmeras monoculares do ambiente devidamente calibradas que fornecem dados visuais para vários serviços distribuídos em rede. Esses serviços são responsáveis por processar as imagens, calcular a localização visual do robô, dentre outras tarefas, possibilitando assim, a obtenção de informações precisas sobre a posição e orientação do robô no ambiente.

Com tudo o que foi posto, o objetivo principal é mostrar como a odometria visual fornecida pelo PIS pode ser utilizada em tarefas robóticas, de maneira distribuída e com o mínimo de erro na estimação da postura. Para isso foi desenvolvida uma aplicação composta por diferentes serviços distribuídos pelo PIS.

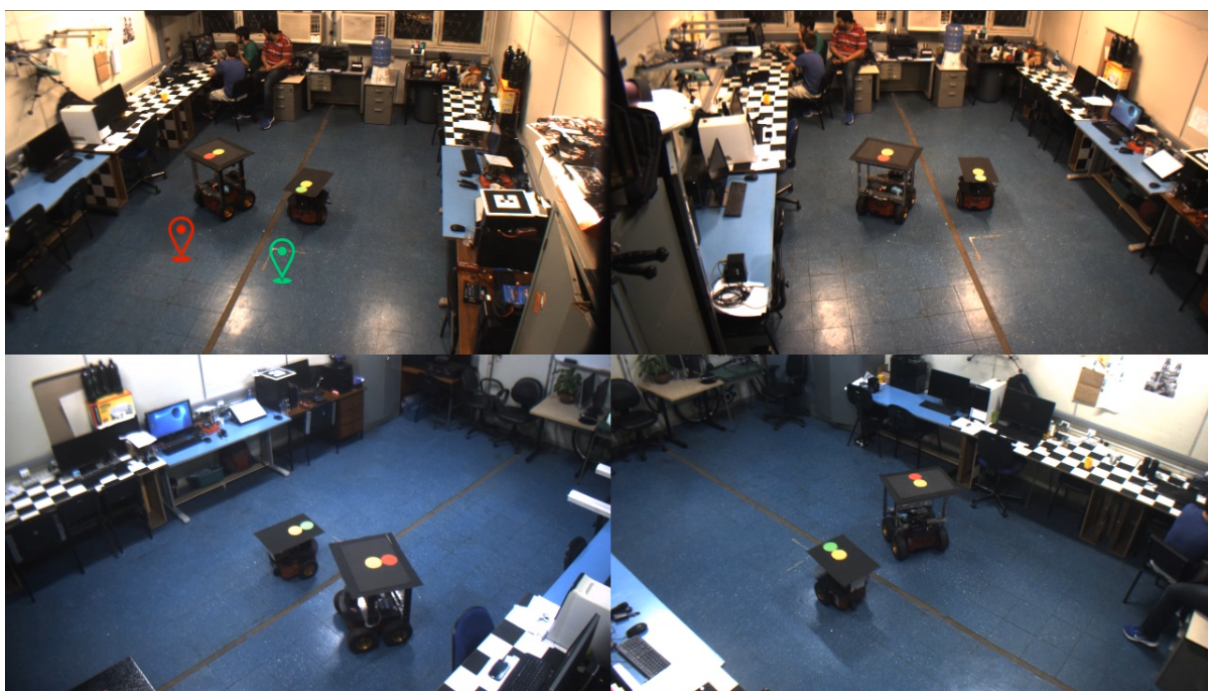
Experimentos

Para realização dos experimentos foi desenvolvida uma aplicação de visualização e controle de formação de robôs. Através dela é possível selecionar a posição final desejada de cada robô móvel na tela de visualização das imagens das câmeras. Tais posições além de indicar a posição final de cada robô, definem o aspecto desejado para a formação, i.e.,

a distância entre robôs e a orientação da reta que os une.

A Figura 54 mostra a tela do usuário após a marcação das posições finais desejadas através dos marcadores vermelho e verde. Os pontos devem ser selecionados na imagem de apenas uma das câmeras e as coordenadas na imagem são convertidas para o referencial do mundo. Vale ressaltar que é possível realizar a reconstrução do ponto selecionado na imagem de uma câmera, resultando em um ponto no referencial do mundo porque é definido *a priori* que o ponto selecionado está localizado sempre no piso do laboratório.

Figura 54 – Aplicação desenvolvida para visualização e envio de comandos.



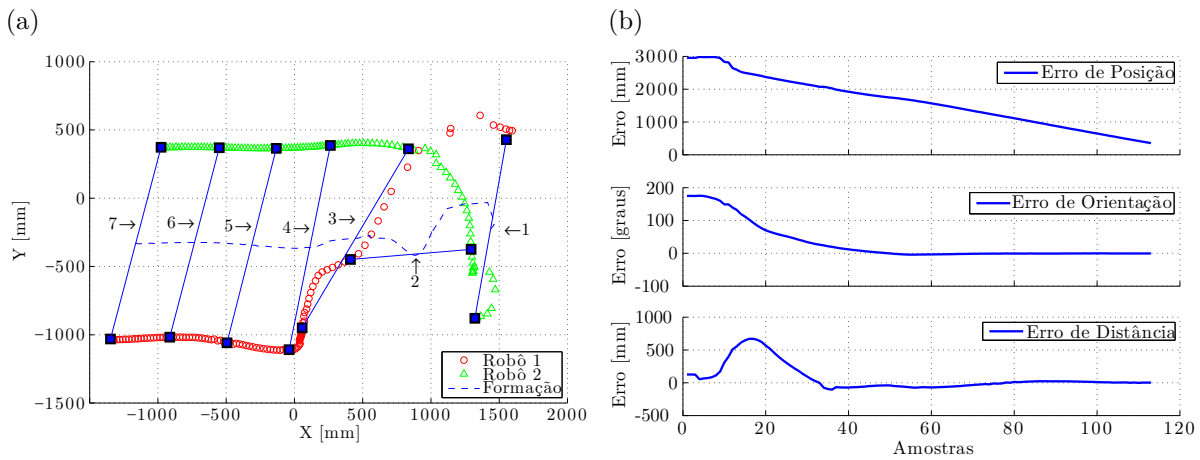
Fonte: (CARMO et al., 2020)

Para o experimento em questão, o controlador recebe o comando de posição final da formação contendo as coordenadas do ponto central da formação, a distância entre os centros dos robôs em milímetros, e a orientação da formação em graus. Nesse experimento, a orientação final do robô estava a 180° da orientação inicial, de modo que toda a formação era obrigada a rotacionar, aumentando a complexidade do processo de controle e de identificação dos mesmos pelo sistema visual.

A Figura 55a mostra as posições dos robôs durante a execução da tarefa de controle de formação, além da indicação de sete instantes da execução da tarefa, a fim de mostrar a evolução da formação. Além disso, são apresentados os erros de posição, orientação e distância da formação na Figura 55b, na qual verifica-se que todos os erros das variáveis de formação convergiram assintoticamente para zero.

Além disso, na Figura 56, está apresentado o número de detecções simultâneas do

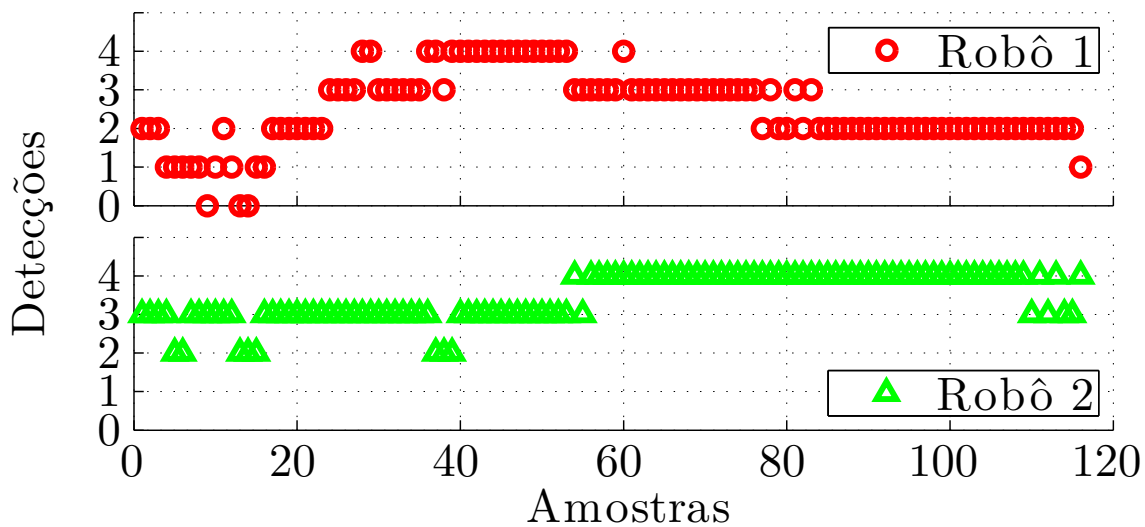
Figura 55 – (a) Evolução das posições dos robôs e (b) erros de posição, orientação e distância da formação.



Fonte: (CARMO et al., 2020)

padrão para cada robô. Nota-se que existem instantes nos quais o robô 1 é detectado apenas por uma câmera. Mesmo assim, foi possível determinar sua postura. A não detecção feita pelas outras câmeras pode ter vários motivos, como oclusão ou condições de iluminação que impossibilitem a segmentação. Portanto, isso mostra a vantagem de se utilizar mais de uma câmera com sobreposição de vistas para localização dos robôs no espaço inteligente.

Figura 56 – Número de detecções simultâneas do padrão para cada robô.



Fonte: (CARMO et al., 2020)

A partir do experimento descrito, observa-se que o desempenho do controlador utilizado foi satisfatório. Dessa forma, é possível observar que a utilização do espaço inteligente como provedor e gerenciador de aplicações que necessitam de odometria visual para realizar tarefas traz bons resultados.

Várias vantagens podem ser citadas no uso de câmeras para localização de robôs móveis, como o erro fixo, redundância de sensores, evitando assim problemas de oclusão, e diminuição do sensoriamento embarcado nos dispositivos robóticos. Além disso, câmeras permitem diversas aplicações dentro da área de espaços e cidades inteligentes, como detecção de pessoas, gestos, segurança pública, entre outros.

Uma vez que a infraestrutura de espaço inteligente utilizada é baseada em serviços, novas aplicações poderão ser implementadas a partir da composição destes e de outros serviços. Essa característica de reutilização de serviços transforma o PIS em uma promissora plataforma que servirá de base para novos trabalhos na área de robótica.

ANEXO B – MobiLysa - Sistema de localização e controle do cão-guia robô Lysa para ambientes internos

Um exemplo da flexibilidade e reaproveitamento de serviços do PIS pode ser observado com a aplicação do MobiLysa.

No Brasil, estima-se que mais de seis milhões e meio de pessoas sejam portadoras de deficiência visual (IBGE, 2010) e, embora não existam estatísticas oficiais, estima-se que existam menos de duzentos cães-guias no país.

O treinamento de um cão-guia é diferenciado e custa, em média, R\$ 50.000,00. Além disso, o treinamento é realizado somente em alguns lugares do Brasil, gerando assim uma longa fila para conseguir um cão treinado. O longo tempo de espera e o alto custo reduzem significativamente as possibilidades de pessoas com deficiência visual terem esse recurso.

O maior desafio é oferecer acessibilidade às pessoas com deficiência visual, garantindo a igualdade de condições com os outros. No entanto, as limitações do indivíduo com deficiência tendem a se tornar uma barreira para esse aprendizado. Desenvolver recursos de acessibilidade seria uma forma concreta de neutralizar as barreiras causadas pela deficiência e inserir esses indivíduos em diferentes ambientes.

Outro desafio está relacionado diretamente aos órgãos públicos e privados, espaços que precisam melhorar a acessibilidade, como shoppings, escolas, hospitais, aeroportos, instituições bancárias e empresas. Todos estes espaços precisam de tecnologias para auxiliar na locomoção das pessoas com deficiência.

O cão-guia robô Lysa (ASSIS, 2019), mostrado na Figura 57, tem funções semelhantes às de um cão-guia convencional. É equipado com dois motores e cinco sensores que avisam aos deficientes visuais por meio de mensagens de voz gravadas, quando há buracos, obstáculos e riscos de colisões em altura. Para dar maior autonomia a esses, estão sendo desenvolvidos algoritmos de navegação com o uso de GPS, onde o usuário pode informar sua rota e a Lysa o levará até o seu destino. Entretanto, o uso de GPS é capaz de auxiliar apenas na localização e navegação da Lysa em ambientes externos, já que tal sensor não apresenta funcionamento adequado em ambientes internos. A localização e controle em ambientes internos pode ser conseguida através do PIS.

O PIS pode ser usado para localizar e controlar o cão-guia robô Lysa, contribuindo

Figura 57 – Cão-guia Robô Lysa



Fonte: Adaptado de ([VIXSYSTEM, 2020](#))

assim, para melhorar o atendimento aos seus usuários. O sensoriamento distribuído e a inteligência presente no ambiente permitem que informações mais amplas, dadas por câmeras por exemplo, e não apenas por sensores do próprio robô, sejam usadas para tarefas mais complexas. Assim a localização e navegação do robô podem ser realizados de forma mais fácil e robusta, aumentando a confiabilidade do sistema e conforto do usuário.

Logo, o principal objetivo do MobiLysa é desenvolver uma aplicação de controle do robô Lysa. O robô guiará uma pessoa até um local desejado dentro de um ambiente indoor, como por exemplo um prédio público. Através da aplicação, o usuário poderá indicar o destino desejado, seja por voz ou outra forma de interação, e ser guiado até o local de maneira independente e segura.

Existem diferentes tecnologias disponíveis para localização de dispositivos móveis. Por exemplo, o trabalho apresentado em ([YASSIN et al., 2017b](#)), mostra como a localização se tornou um fator chave para serviços específicos que são oferecidos ou executados de acordo com o dispositivo ou localização do usuário.

É notório que atualmente, abordagens que exploram cooperação e coordenação com mecanismos de localização, para tecnologias sem fio/móveis, são certamente uma tendência a ser seguida em pesquisas nas áreas de redes e robótica. Abordagens como as apresentadas nos trabalhos ([CONTI et al., 2012](#))([WIN; GIFFORD; THOMAS, 2011](#)) mostram como o uso de modelos de erro de alcance e a cooperação entre dispositivos podem melhorar a localização em redes de comunicação sem fio. Além disso, no trabalho ([Win et al., 2018](#)), os autores demonstram que a fusão multissensorial e a cooperação espacial podem aumentar significativamente a localização em um cenário de grande escala com centenas de agentes móveis.

No entanto, problemas como obstrução de sinais entre emissores e receptores geral-

Tabela 12 – Comparativo tecnológico

Tecnologia	Precisão Aproximada	Desvantagem
Tecnologias com codificação de sinal		
Infravermelho	57cm ~23cm	Interferência do sol
VLC	10cm	Alto custo
Ultrassom	1cm ~2m	Interferências
WiFi	1,5m	Vulnerável a mudanças no AP
Bluetooth	30cm ~10m	Necessita mapeamento de sinal
ZigBee	25cm	Necessita equipamento específico
RFID	1m ~5m	Precisão muito baixa
UWB	15cm	Alto custo
Tecnologias passivas sem codificação de sinal		
Geomagnético	2m	Necessita de mapeamento
Inercial	2m	Acumula erro
Som ambiente	-	Sem precisão
Luz ambiente	10cm ~alguns metros	Sensibilidade a luminosidade
Visão computacional	1cm ~1m	Sensibilidade a luminosidade

Fonte: Adaptado de (BRENA et al., 2017)

mente dificultam o uso de métodos tradicionais de localização baseados em radiofrequência (RF). A intensidade do sinal, o atraso do sinal e o ângulo de chegada da propagação da onda eletromagnética podem ser prejudicados por níveis severos de impedimentos e interferências em determinados ambientes (BRENA et al., 2017).

Dessa forma, uma abordagem promissora para obter precisão de localização na ordem de poucos centímetros é usar métodos de visão computacional em ambientes inteligentes (BRENA et al., 2017), (Dardari; Closas; Djurić, 2015). Imagens podem fornecer uma riqueza de informações que, a partir do devido processamento, são capazes de gerar estimativas de localização bastante precisas.

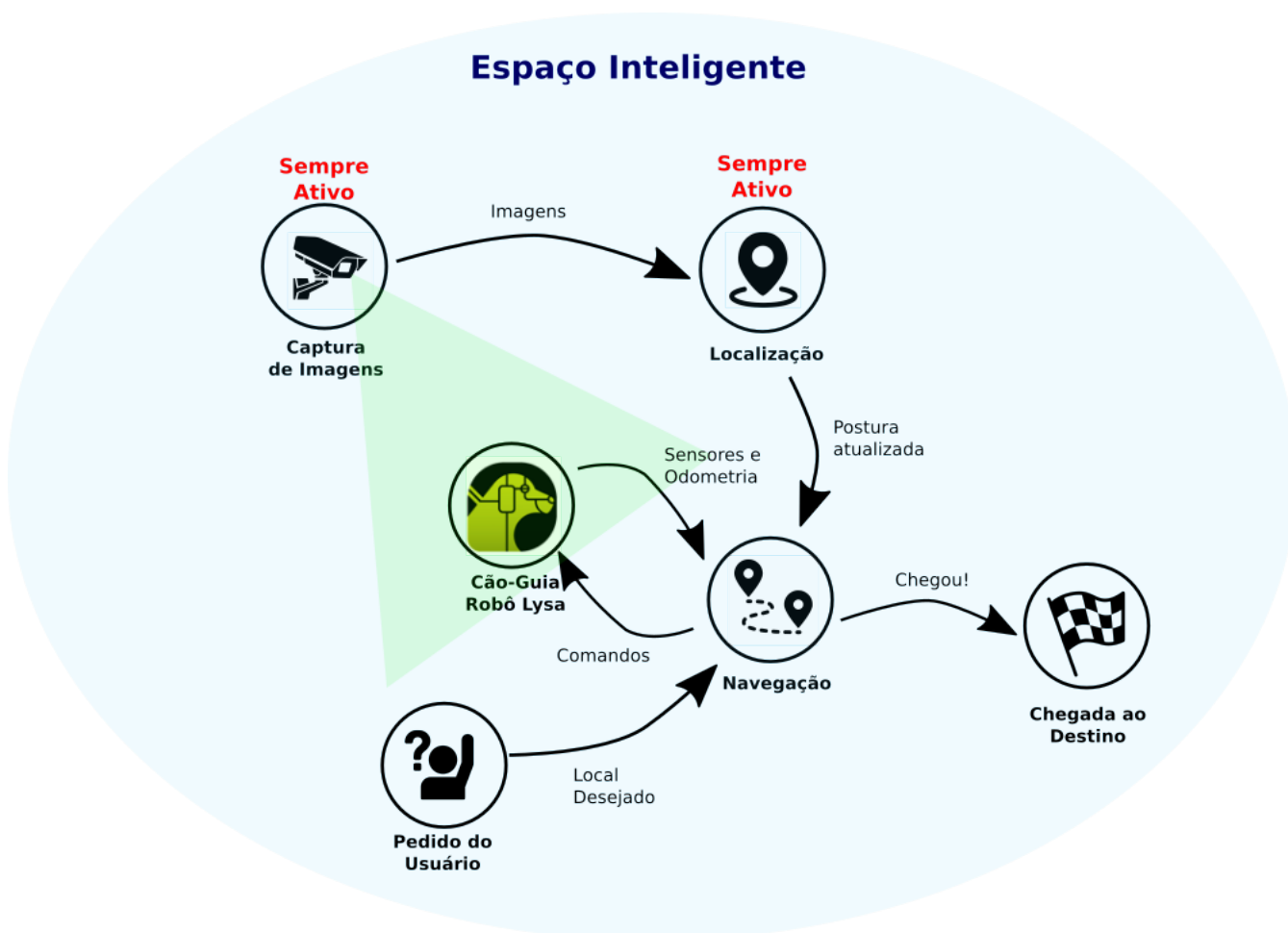
Um comparativo entre as principais técnicas de localização em ambientes internos é mostrado na Tabela 12, onde pode-se ver que técnicas de localização por visão computacional possuem uma melhor precisão quando comparadas com outras técnicas tradicionais de localização em ambientes internos.

Apesar de sistemas de localização baseados em visão computacional fornecerem informações precisas, extrair dados a partir de um conjunto de imagens exigem uma alta utilização de recursos. A gerência desses recursos e ao mesmo tempo a necessidade de atender aos requisitos da aplicação em tempo real é o desafio que o PIS deve enfrentar.

Componentes da Aplicação

A aplicação MobiLysa é composta por diferentes serviços. A interação entre os serviços e o espaço é mostrado na Figura 58.

Figura 58 – Serviços que compõe a aplicação Mobilysa



Fonte: Produção do próprio autor

A Mobilysa usará a rede de câmeras do PIS para localizar o robô no ambiente e realizar o controle da Lysa principalmente por realimentação visual. Como as câmeras utilizadas no ambiente não foram alteradas, o mesmo serviço de gateway será reutilizado. Já o robô é diferente, e por isso o gateway teve que ser adaptado para gerar um novo serviço para o esse dispositivo específico.

A aplicação inicia a partir de um comando do usuário. O comando é capturado através de um serviço de interface com o usuário que receberá os comandos de voz e extrairá a localização do destino. O destino é enviada então para o serviço de navegação que em conjunto com a localização atual do robô definirá uma trajetória da origem até o ponto de destino.

A localização do robô será feita em tempo real usando-se um método de reconhecimento de objetos. Dada a localização, a navegação do robô Lysa poderá ser controlada fazendo-se uso de um mapa do ambiente. Tal mapa será previamente construído e conterá todas as informações relevantes para condução dos usuários aos locais de destino disponíveis. Esse mapa deverá ser atualizado sempre que alterações significativas acontecerem no ambiente.

Apesar de se considerar o uso de um Espaço Inteligente Provagramável baseado em visão computacional para o desenvolvimento do serviço MobiLysa, o mapa a ser utilizado não é formado pelas imagens capturadas pelas câmeras. Esse mapa será do tipo híbrido (topológico e métrico). Dessa forma, mesmo que as câmeras do ambiente não apresentem sobreposição em todas as regiões, ou seja, apresentem zonas mortas, a navegação será possível devido aos sensores embarcados no robô e estratégia de navegação adotada.

Vale ressaltar que as tarefas de detecção e desvio de obstáculos são funcionalidades embarcadas no próprio robô e, portanto, não representam uma preocupação para o serviço de localização e navegação proposto. Caso algum desvio aconteça por causa de um obstáculo, assim que possível o robô será redirecionado à trajetória planejada.

Além de imagens, dados de odometria ou de outros sensores locais do robô podem ser usados para melhorar o desempenho do controlador, principalmente nas zonas mortas mencionadas anteriormente. É importante que a navegação no ambiente possa ocorrer de forma suave e estável, uma vez que o robô estará guiando uma pessoa com deficiência visual através de seu movimento.

Diferentemente do serviço de localização utilizado anteriormente pela aplicação de controle de formação de robôs, o serviço de localização do Mobilysa, utiliza técnicas mais robustas de reconhecimento de objetos. Essas técnicas fazem uso de processamento através de várias camadas de uma rede neural convolucional (CNN, do inglês Convolutional Neural Network)([GOODFELLOW; BENGIO; COURVILLE, 2016](#)). A camada final da CNN produz um valor probabilístico, correspondente a cada uma das possíveis categorias a qual o objeto pode ser classificado. Dentre os algoritmos mais utilizados para detecção de objetos pode-se listar o YOLOv3 ([REDMON; FARHADI, 2018](#)), Mask R-CNN ([HE et al., 2017](#)) e Faster R-CNN ([Ren et al., 2017](#)), os quais são capazes de realizar detecção de objetos em tempo real, de forma rápida e precisa. Técnicas como transfer learning ([SHANMUGAMANI, 2018](#)) podem ainda ser aplicadas para adaptar reconhecedores de objetos como YOLO, para um problema mais específico como a detecção do robô Lysa.

Dessa forma, será gerado um banco de imagens do robô Lysa, onde o robô deverá aparecer em diferentes posições e sob diferentes pontos de vista, tanto isoladamente quanto sendo utilizado por usuários. Essas imagens serão utilizadas para adaptação e treinamento de uma rede neural, a fim de que a detecção e reconhecimento da plataforma possa acontecer satisfatoriamente.

Com exceção do gateway do robô que é executado nele próprio, todos os outros serviços que compõe a aplicação MobiLysa serão executados na infraestrutura computacional em nuvem do PIS. Isso dará ao sistema modularidade, escalabilidade e versatilidade, pois novos serviços poderão ser adicionados ao sistema sempre que necessário, além da reutilização imediata de serviços já existentes na arquitetura. Essa característica se mostra ainda mais importante em um contexto em que múltiplas Lysas estarão funcionando em diferentes espaços simultaneamente.

Estágio atual e perspectivas

O sistema MobiLysa, traz uma proposta de solução para localização e navegação em ambiente internos que melhora a acessibilidade para as pessoas com deficiência visual, permitindo a sua condução a locais desejados por meio do uso do robô Lysa. Tal condição proporciona uma maior independência de locomoção para o usuário.

Diferentes espaços, públicos e privados, buscam melhorar a acessibilidade e mobilidade de usuários com deficiência visual. Entretanto, há sempre o desejo que o robô seja capaz de guiar a pessoa até um determinado local, o que só é conseguido através de um sistema de localização e navegação.

Dessa forma, o MobiLysa se apresenta como uma aplicação interessante a esses espaços, com grande circulação de pessoas, com boas perspectivas de aceitação pelo público alvo e pela sociedade.

O MobiLysa encontra-se atualmente em desenvolvimento. Os serviços da aplicação de controle de formação de robôs apresentado anteriormente, serviram de base para o desenvolvimento dessa aplicação.

O serviço de gateway das câmeras é exatamente o mesmo e foi reaproveitado. Já para o gateway do robô foi reutilizado utilizado gateway anterior e alterado apenas a parte do driver para comunicação com a Lysa.

O serviço de localização utilizado até o momento é similar ao utilizado na aplicação de controle de formação. A localização também é baseada em marcadores, mas de um tipo diferente chamado de Aruco. Esse serviço de localização é o mesmo utilizado no estudo de caso mostrado no capítulo 5. O novo serviço de localização desenvolvido especificamente para localização da Lysa está em desenvolvimento. A expectativa é eliminar o uso de um marcador visual atrelado ao robô Lysa. Afinal, ter um marcador visual acoplado à plataforma pode ser esteticamente indesejável ou até mesmo comprometer o processo de localização do dispositivo.

Um desafio importante nessa mudança diz respeito ao desempenho desse serviço. A expectativa é que o desempenho em termos de tempo de processamento desse serviço seja

muito superior ao serviço de localização atual. Caso isso se confirme, aumentará ainda mais a necessidade de gerenciamento dos recursos.

O serviço de navegação da Lysa teve que ser bastante alterado. O serviço de controle foi dividido em dois: o serviço path planning e o serviço robot controller.

O path planning deve possuir o mapa do ambiente. Ao receber a localização de destino pelo usuário ele irá calcular o melhor caminho de acordo com a localização atual da Lysa. Esse caminho pode ser alterado caso o usuário queira alterar o destino ou mesmo parar a navegação a qualquer momento. Já o serviço robot controller teve que ser adequado para se ajustar aos parâmetros específicos do novo robô.

Por último, como serviço de interface com o usuário, foi reutilizado o serviço em que se indicava o destino final através do clique na imagem de uma das câmeras do ambiente. Esse serviço será substituído por outro em que a posição final será definida por um comando de voz.

Além do desenvolvimento técnico do sistema, tem-se também como passo futuro tornar esse sistema economicamente viável para sua comercialização.

ANEXO C – Serviço de Detecção de Pessoas

Atualmente, muitas aplicações desenvolvidas para um espaço inteligente precisam detectar pessoas para promover a interação homem-máquina ou homem-robô, oferecer serviços ou até inferir contextos observando o movimento humano (GLAS *et al.*, 2015; GLAS *et al.*, 2013; ZHOU *et al.*, 2013; MRAZOVAC *et al.*, 2012; BRSSCIÉ, 2014).

Portanto, é comum ver redes com múltiplos sensores sendo usadas para extrair informações úteis do ambiente e detectar a presença humana (COOK *et al.*, 2013; MRAZOVAC *et al.*, 2012). Alguns trabalhos evitam o uso de câmeras devido à complexidade dos algoritmos envolvidos e aos problemas com variações de oclusão, cor e brilho, além de alterações na aparência dos objetos devido ao ponto de vista adotado. Nesses casos, sensores como lasers, ultrassons e sensores de movimento são geralmente usados (MORIOKA; HASHIKAWA; TAKIGAWA, 2013; SURIE; PARTONIA; LINDGREN, 2013; COOK *et al.*, 2010).

No entanto, quando se trata de interação direta com seres humanos, não apenas é necessária a detecção de presença, mas também uma estimativa mais precisa da posição das pessoas e, às vezes, reconhecimento de rosto ou gesto. As redes com várias câmeras são preferidas, já que as câmeras fornecem uma rica fonte de informações sobre o meio ambiente e a presença de objetos na cena (LEE *et al.*, 2012; ZABULIS *et al.*, 2013).

No caso específico da detecção de pessoas, a maioria dos métodos é baseada em modelo e avaliada apenas em poucos conjuntos de dados públicos, após horas ou dias de ajuste de hiperparâmetros. Como mencionado em (LI; YAO; WANG, 2012), esses tipos de detectores buscam uma solução genérica, tentando reduzir o máximo de erros possível e, com muito poucas exceções, apenas testes offline são realizados.

Apesar do progresso na detecção de objetos liderado pela família R-CNN (GIRSHICK *et al.*, 2014), a eficácia da detecção de objetos e pessoas em aplicações em tempo real, conforme exigido pelos Espaços Inteligentes, ainda é um problema não resolvido. Se a detecção de pessoas com uma câmera já exige alta capacidade computacional, é ainda mais difícil realizar isso com uma rede de câmeras.

Portanto, um dos problemas atuais para se aplicar a detecção humana em um Espaço Inteligente que possui câmeras como os principais sensores no ambiente é desenvolver um detector leve que não sobrecarregue a infraestrutura, economizando recursos computacionais para outros serviços ou aplicações.

Assim, os detectores de pessoas podem se beneficiar das vantagens oferecidas pela rede multi câmeras e dos recursos disponíveis devido à arquitetura do PIS. A detecção de pessoas deve ser fornecida como um serviço, que pode ser usado por diferentes aplicações e ser executado em diferentes tipos de infraestruturas, sendo ao mesmo tempo leve e flexível.

Com isso em mente, desenvolveu-se um serviço leve e flexível que fornece detecção de pessoas para o Espaço Inteligente baseado em uma rede de várias câmeras, como o PIS. As informações visuais do ambiente são fornecidas usando apenas a rede de câmeras, enquanto o detector de pessoas é oferecido como um serviço para diferentes aplicações. Além de realizar a detecção humana, o serviço proposto se aproveita e utiliza importantes propriedades do PIS, como paralelismo, escalabilidade e alocação dinâmica para atender aos requisitos específicos do serviço e ao mesmo tempo fazer um uso mais racional dos recursos da infraestrutura.

São dois os objetivos principais do serviço de detecção humana: (i) que possua características como flexibilidade, escalabilidade, paralelismo e confiabilidade; e também (ii) realizar a localização humana, fornecida por uma rede multi câmeras, que nos permita evitar alguns problemas que fazem com que os detectores de pessoas com uma única câmera falhem em aplicações desenvolvidas para o mundo real. O inter-relacionamento do serviço de detecção de pessoas e da rede de câmeras é tratado para atender aos requisitos das aplicações que são executadas em tempo real.

Quanto ao termo tempo real, de acordo com (ATIS, 2020), a distinção entre tempo real e tempo quase real não é muito clara. Assim, no contexto deste serviço e as aplicações que o utilizam, será adotado o termo tempo real quando a taxa de processamento de dados alcançada não inserir atrasos que não permitam atender aos requisitos de tempo das aplicações. Nesse caso, a experiência dos usuários na interação com o PIS não deverá ser afetada pelas técnicas utilizadas para o processamento dos dados ou para alocação de recursos. Um exemplo é o tempo de resposta do robô em relação ao movimento do ser humano nas aplicações propostas neste trabalho. Idealmente, o tempo de resposta não pode ser afetado pelo tempo de processamento do detector de pessoas.

Portanto, aqui será apresentado um serviço de detecção de pessoas baseado em uma rede multi câmeras, que possa atender aos requisitos de tempo das aplicações considerando uma capacidade computacional limitada e compartilhada. Para validação do serviço implementou-se 3 diferentes aplicações de tempo real para serem usadas como prova de conceito, sendo que duas envolvem a interação homem robô.

Detectores de Pessoas e sua relação com Espaços Inteligentes Atuais

A detecção de pessoas é uma área de pesquisa ativa. Mesmo com o sucesso da detecção geral de objetos ([GIRSHICK et al., 2014](#); [CAI et al., 2016](#); [Ren et al., 2017](#); [REDMON; FARHADI, 2018](#)), a detecção humana foi tratada como um campo de interesse exclusivo. Isso ocorre principalmente porque os seres humanos são componentes essenciais para muitas aplicações atuais, como assistência ao motorista e sistemas de vigilância inteligentes ([ZHANG et al., 2016](#)). Além disso, os detectores de pessoas apresentam problemas específicos ao tratar a sua segmentação dos elementos em segundo plano ([MAO et al., 2017](#)).

O exemplo mais estudado de detecção humana é a área de detecção de pedestres. Muitas soluções já foram propostas na literatura ([DOLLÁR et al., 2014](#); [BENENSON et al., 2015](#); [ZHANG; BENENSON; SCHIELE, 2015](#); [HOSANG et al., 2015](#); [DU et al., 2016](#); [ZHANG et al., 2016](#)). Apesar do fato de as Redes Neurais Convolucionais Profundas (DCNN) terem impulsionado o estado da arte da detecção de pedestres, mesmo os detectores genéricos mais bem treinados têm desempenho fraco ao serem avaliados em diferentes datasets ([ZHANG; BENENSON; SCHIELE, 2017](#)). Isso indica que, em alguns casos, o uso de informações adicionais da cena ainda pode ser inevitável ([LI; YAO; WANG, 2012](#)).

Apesar de apresentar bons resultados relacionados a taxa de acertos e precisão, detectores de pessoas baseados em DCNN necessitam de GPUs para suprir uma alta demanda de capacidade de processamento. Mesmo quando a infraestrutura do Espaço Inteligente possua esse tipo de recurso específico ele comumente é limitado, e por isso é interessante ter serviços e aplicações leves para que eles não consumam todos os recursos disponíveis. Portanto, sempre que possível, a implementação de um detector de pessoas que não utiliza GPU também é desejável.

Portanto, o fato de o serviço de detecção de pessoas não necessitar do uso de GPU, mesmo para aplicações reais, traz flexibilidade para as aplicações que o utilizarem em sua composição. ([RIBEIRO et al., 2017](#)), por exemplo, diferentemente do serviço aqui proposto, o uso de GPU é obrigatório para realizar a detecção de pessoas. Além disso, seu método de detecção humana não é entregue como um serviço que pode ser usado de forma flexível por diferentes aplicações. A solução proposta em ([RIBEIRO et al., 2017](#)) não foi projetada para ser distribuída pelos nós da infraestrutura, portanto, não está relacionada a características como sincronismo, escalabilidade e confiabilidade. Eles usam apenas um nó com alto poder computacional baseado em múltiplas GPUs para ter uma baixa taxa de falsos positivos, enquanto usamos técnicas simples e as informações de redundância fornecidas por uma rede de câmeras para atingir esse objetivo. Nesse caso, temos um detector leve que pode ser distribuído pela infraestrutura.

Existem diferentes trabalhos sobre Espaços Inteligentes que realizam detecção de pessoas usando redes de sensores. Alguns sensores diferentes podem ser abordados, mas serão focados os trabalhos que usam principalmente sensores visuais ou sua combinação com alguns outros tipos de sensores. A ideia é destacar as diferenças entre esses trabalhos e o serviço de detecção aqui apresentado.

Em (SURIE; PARTONIA; LINDGREN, 2013), um Kinect montado na parede é usado para detectar pessoas usando o reconhecimento de esqueletos e faces, extraído de imagens e campos de profundidade. Já no caso do detector aqui apresentado, o serviço implementado é baseado em modelo e depende apenas de imagens RGB (vermelho-verde-azul), enquanto nenhum sensor de campo de profundidade é usado para obter as informações tridimensionais. Os autores de (SURIE; PARTONIA; LINDGREN, 2013) também mencionam que seu espaço de trabalho é restrito devido ao campo de visão limitado da câmera do kinect, exigindo uma configuração mais estruturada. Por outro lado, o serviço aqui apresentado se beneficia de diferentes pontos de vista fornecidos por uma rede multi câmera, identificando pessoas com várias poses em relação a esses sensores. Além disso, não há restrições quanto à posição da câmera na área de trabalho, sendo a única configuração prévia necessária é a calibração da câmera.

Os autores de (GLAS. et al., 2015) desenvolveram um "uma rede de robôs" para implantar robôs sociais em aplicações práticas como shopping centers e outros espaços comerciais. No referido trabalho, aplicações de inteligência ambiental e interação homem-máquina são implementadas usando dados obtidos dos sensores instalados no ambiente. Os sensores utilizados são lasers de campo de profundidade e um supervisor humano também é empregado quando o sistema enfrenta dificuldades nas tarefas de reconhecimento e planejamento. No detector aqui proposto, nenhum supervisor humano é utilizado e o sistema é construído usando uma filosofia de não intervenção humana durante a oferta de serviços, além disso, o detector do PIS utiliza uma rede de câmeras, que geralmente já estão instaladas na maioria dos edifícios comerciais, exigindo assim, um menor número de modificações físicas em comparação à instalação de lasers no ambiente. Além disso, em (GLAS. et al., 2015), a rede de robôs é o único serviço fornecido. No PIS, robôs (atuadores) e câmeras (sensores) por exemplo, são tratados como dispositivos que facilmente podem ser adicionados a rede de sensores e atuadores. Além disso, o detector de pessoas é apenas mais um dos muitos serviços fornecidos a diferentes aplicações.

Um sistema de transporte robótico para assistência às compras é desenvolvido em (MATSUHIRA et al., 2010). Porém, o foco não está em um sistema com serviços distribuídos. A aplicação parece funcionar bem, mas a abordagem de detecção humana é mais simples e depende da fusão de sensores de vídeo e lasers de campo de profundidade. Apesar de usar um conjunto de câmeras, as informações tridimensionais não são totalmente exploradas.

O objetivo em (ALBAWENDI et al., 2015) é investigar um sistema de monitoramento baseado em câmera que possua um desempenho aceitável e de baixo custo para os idosos. A ideia é limitar a quantidade de informações transmitidas pelo sensor visual, reduzindo o nível de intrusão da câmera na rotina dos idosos. Comparando com o detector do PIS, o sistema deles é mais um aplicação e a detecção de objetos depende muito da subtração de elementos em segundo plano, exigindo um espaço de trabalho mais estruturado e, portanto, mais restritivo.

No (Adduci; Amplianitis; Reulke, 2014), uma aplicação de rastreamento com várias câmeras é criado. No entanto, de acordo com os autores, a aplicação depende da estimativa da nuvem de pontos, e exige que modificações de hardware e software sejam preparadas para tarefas em tempo real. Já os serviços de detecção de pessoas e robôs, aqui apresentados, são adequados para aplicações em tempo real, tal como um serviço para controlar o robô. Diferentemente dos trabalhos mencionados anteriormente, um dos objetivos do serviço aqui apresentado é que ele possa ser executado em um ambiente distribuído e com isso possa atender aos requisitos de tempo e confiabilidade das aplicações.

Como última comparação, os autores de (LEE et al., 2012) desenvolvem um método baseado em visão para localização de humanos e robôs, a fim de serem usados em um serviço ativo de exibição de informações em um Espaço Inteligente. Para isso, eles configuram uma rede com várias câmeras para estimar as posições 3D de um humano e de um robô, que são detectados usando o método de recurso Histograma de gradientes orientados (HOG, do inglês Histogram of Oriented Gradient). Uma das principais diferenças é que o sistema não é baseado em uma arquitetura de serviço, com uma configuração fixa. Embora sejam usados vários dispositivos de rede inteligentes distribuídos (DIND, do inglês Distributed Intelligent Network Devices), que consistem em uma câmera e um dispositivo de rede, cada DIND é conectado a desktops e atuam como um nó de processamento individual que pode se comunicar com outros através da rede local. Portanto, nenhum serviço, como detecção de pessoas e robôs ou controle de robôs, é fornecido por sua arquitetura de sistema. Também para estimar a posição 3D de uma pessoa, é necessário combinar pelo menos duas detecções fornecidas por câmeras diferentes, para que seu sistema possa realizar triangulação. Sua função de localização pode enfrentar alguns problemas se muitas pessoas estiverem presentes no espaço de trabalho, pois é obrigatório ter duas detecções da mesma pessoa. No detector aqui apresentado, a detecção de pessoas e robôs pode ser realizada mesmo que apenas uma câmera seja capaz de observar o robô e a pessoa. Como assumimos que humanos e robôs estão sempre de pé no chão da sala, nosso sistema pode estimar as posições 3D com apenas uma imagem. Detecções adicionais são usadas para eliminar falsos positivos que possam ocorrer e melhorar a localização 3D estimada. Por fim, o ambiente apresentado pelos autores é muito mais estruturado do que o necessário para funcionamento do detector no PIS, não havendo objetos no espaço de trabalho definidos para os experimentos e um ambiente leve e homogêneo, o que facilita a detecção de pessoas

e robôs.

Aplicações que utilizam o serviço de detecção de Pessoas

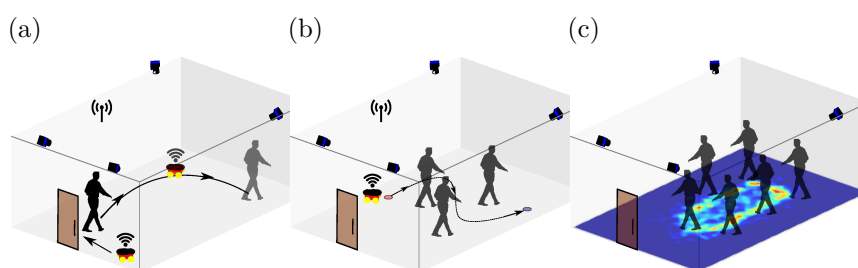
Foram escolhidos 3 aplicações que possuem o serviço de detecção de pessoas em sua composição. Essas aplicações foram desenvolvidas para servirem de Prova de Conceito para mostrar a eficácia do serviço de detecção humana aqui proposto dentro do PIS. A Figura 59 ilustra as tarefas executadas em cada aplicação desenvolvida.

A primeira aplicação é uma tarefa de acompanhamento de uma pessoa realizada por um robô (Figura 59a). Nesse caso, o robô deve seguir um humano que acabou de entrar na sala. Essa é uma estratégia muito comum, porque os humanos que acabaram de chegar à sala podem precisar de algum serviço ofertado pelo robô de atendimento.

Já a Figura 59b mostra outra tarefa, na qual o robô precisa navegar no Espaço Inteligente, desviando-se dos humanos presentes no ambiente. Isso é importante, por exemplo, para ajudar na primeira tarefa, onde o robô deve chegar a alguém, mas não deve atingir outras pessoas enquanto se move. Essas aplicações podem ser consideradas com Provas de conceito para muitas tarefas reais do dia-a-dia que podem ser executadas em ambientes onde uma rede com várias câmeras é fornecida, como bancos, museus, shopping centers e praças.

Por fim, a Figura 59c mostra o mapa de ocupação acumulativa do Espaço Inteligente. Esse mapa é referido como acumulativo, pois tem o objetivo de consolidar os locais mais visitados na sala ao longo do tempo. Portanto, é uma análise temporal. Isso é muito útil, por exemplo, para determinar dinamicamente os melhores locais para colocar anúncios ou para determinar o preço do aluguel de uma loja em um shopping center. Os locais mais visitados são geralmente aqueles em que os produtos ou anúncios são mais expostos e vistos pelos consumidores. Consequentemente, salas comerciais são compreensivelmente mais caras nesses locais.

Figura 59 – Aplicações: (a) Seguimento de pessoas (b) Desvio de pessoas (c) Mapa de Ocupação.



Fonte: (ALMONFREY et al., 2018)

A Figura 60 mostra os serviços específicos de visão computacional que compõem as aplicações e o fluxo de mensagens entre eles. Serviços de suporte, como serviço de calibração e serviço de sincronização, não são mostrados no diagrama para melhor visualização.

O serviço Pattern Tracker é responsável por recuperar e publicar a posição 3D do robô, usando um padrão geométrico reconhecido pela rede com várias câmeras. Com essa odometria visual, o robô pode ser controlado pelo serviço Robot Control. Os serviços Human Localization e Pattern Tracker consomem os frames publicados pelos gateways da câmera. O serviço Human Localization é responsável por fornecer bounding box (BB) ao serviço de filtragem de BB. BB são anotações com a localização das pessoas nas imagens. O serviço BB Filtering recebe as imagens anotadas e aplica filtros com o objetivo de diminuir o número de falsos positivos e melhorar o desempenho de detecção. Já o serviço BB Frame Conversion, é responsável por projetar os BBs publicados pelo serviço BB Filtering na coordenada 3D do mundo. Uma vez que as informações em 3D das pessoas e do robô estejam disponíveis, as aplicações poderão ser executadas.

A única aplicação que não usa o serviço Pattern Tracker é o Mapa de Ocupação. É importante notar que todas as aplicações mostradas na Figura 60 são independentes e podem ser executadas ao mesmo tempo no Espaço Inteligente. Cada fluxo da Figura 60 é apenas uma ocorrência do loop principal, e esses fluxos são sistematicamente repetidos durante a execução das aplicações.

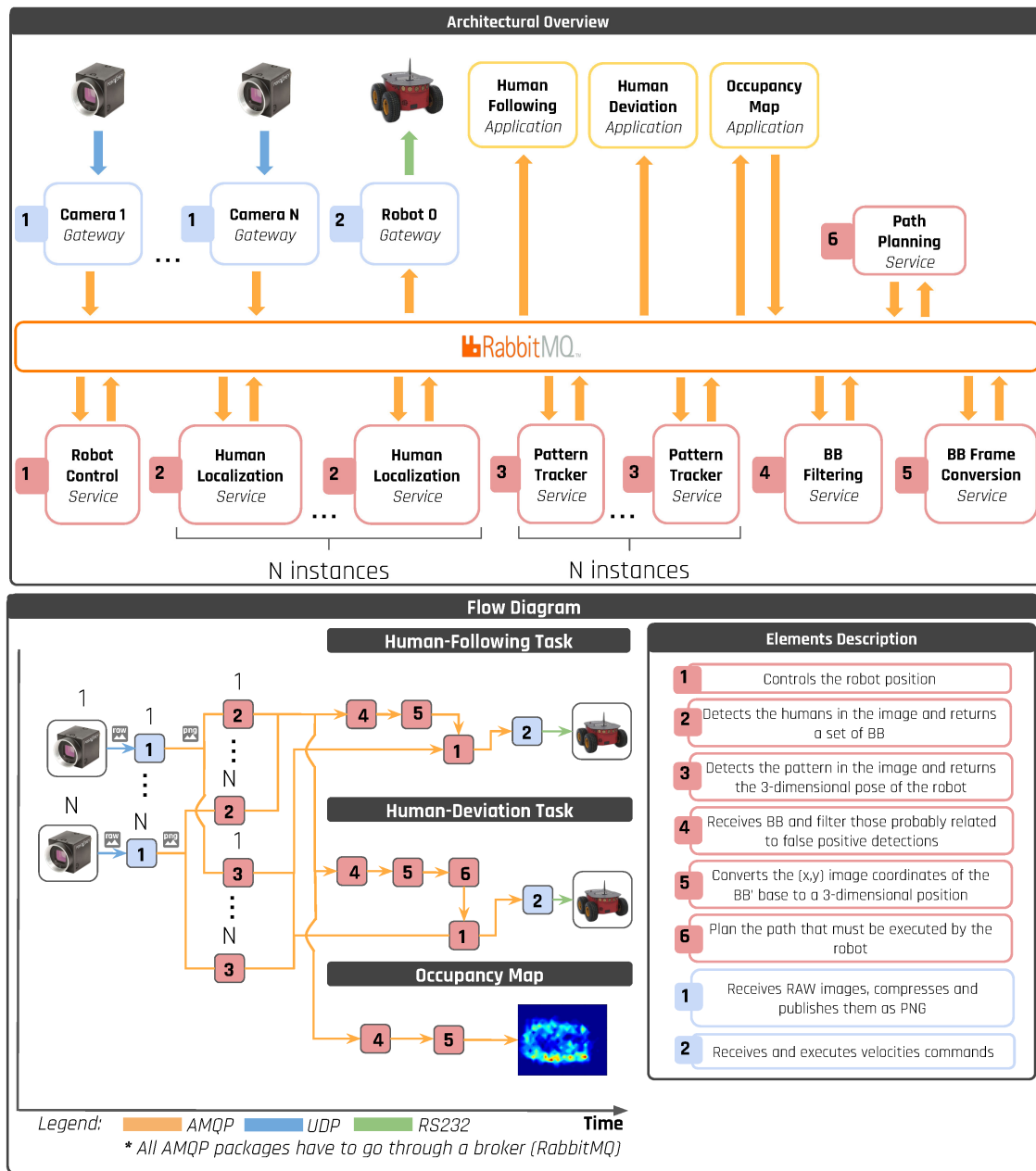
Os fluxos originados dos serviços de Human Localization e Pattern Tracker são assíncronos e executados em paralelo. Nas aplicações de acompanhamento de pessoas e de desvio de pessoas, esses fluxos convergem para o serviço de controle de robô, pois fornecem as informações necessárias para executar as aplicações propostas. Cada uma das N instâncias dos serviços Human Localization e Pattern Tracker também são executadas em paralelo. Isso mostra o paralelismo intrínseco do nosso sistema. Esse recurso é de particular importância, devido ao fato de que detectores de pessoas e de objetos normalmente são tarefas com alto tempo de processamento. Nesse caso, esses processos podem ser distribuídos pela infraestrutura do PIS, usando os nós com mais recursos disponíveis.

Experimentos

Nesta seção, será demonstrada e validada a eficácia do nosso serviço de detecção humana no contexto do PIS com base em uma rede com várias câmeras. É importante enfatizar que a eficácia não é apenas propriedade da detecção de pessoas, mas também tem relação com o atendimento de requisitos de tempo e sua capacidade de se utilizar características presentes no PIS para que esse requisito seja atendido.

Entre outras, as principais contribuições validadas durante os experimentos são:

Figura 60 – Interrelação entre os serviços que compõem a aplicação.



Fonte: (ALMONFREY et al., 2018).

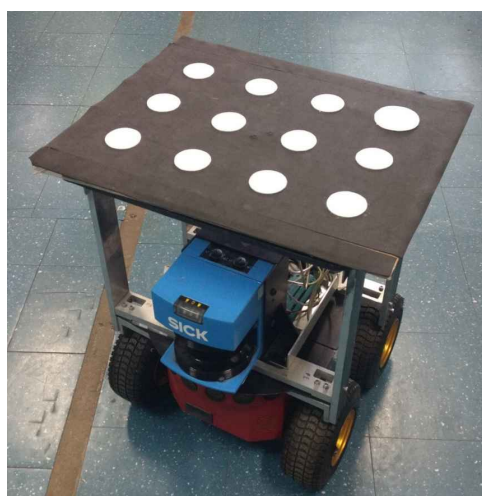
- a eficácia do serviço de detecção humana no atendimento dos requisitos das aplicações por meio do PIS baseado em uma rede de múltiplas câmeras;
- o desenvolvimento de um serviço de detecção de pessoas capaz de cooperar com outros serviços da arquitetura do PIS;
- o atendimento dos requisitos das aplicações em tempo real devido à cooperação adequada entre a detecção de pessoas e outros serviços.

Materiais e Métodos

Para esse experimento foi utilizado o PIS implementado na UFES. As quatro câmeras são os únicos sensores empregados. Não há limite específico no número de câmeras que podem ser usadas pelo serviço de detecção de pessoas e por qualquer outro serviço do Espaço Inteligente. Porém quanto maior o número de câmeras utilizado melhor o resultado do detector

Como mencionado anteriormente, o robô é rastreado usando um serviço de odometria visual que detecta um padrão anexado a ele, como mostra a Figura 61.

Figura 61 – Robô com o padrão utilizado nos experimentos.



Fonte: (ALMONFREY et al., 2018).

O serviço de detecção de pessoas foi treinado usando o conjunto de dados INRIA (DALAL; TRIGGS, 2005) conforme descrito em (ALMONFREY et al., 2016). Nenhuma amostra com imagem do nosso Espaço inteligente foi usada para treinamento.

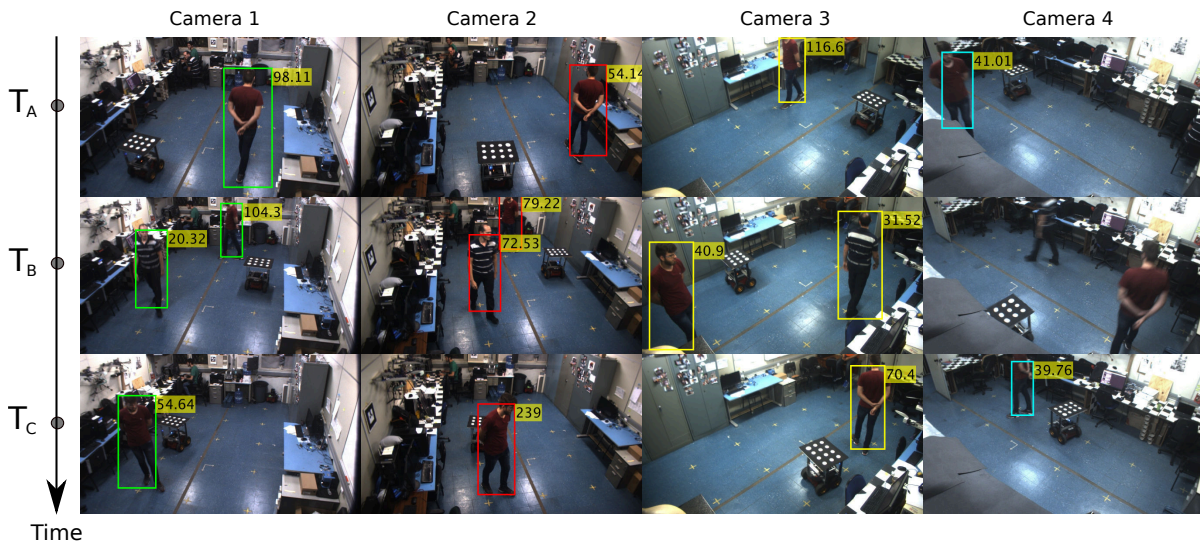
Embora este não seja o foco do presente trabalho, será apresentada uma análise da precisão do serviço de detecção humana no primeiro experimento. Serão utilizadas as métricas de taxa de erros (MR, do inglês Miss Rate) e falsos positivos por imagem (FPPI, do inglês False Positives Per Image). MR é a fração do total de pessoas não identificados pelo método durante o experimento. FPPI é o número de amostras não humanas detectadas como pessoas dividido pelo número de imagens do experimento.

Seguimento de pessoas

Neste experimento, o serviço de detecção de pessoas é usado para gerar pontos de destino para o controlador do robô durante a tarefa de seguimento de pessoas. A Figura 62 mostra o processo de detecção de pessoas em três momentos diferentes (T_A , T_B e T_C)

durante a navegação do robô. Os números mostrados nas figuras representam o valor de confiança da detecção.

Figura 62 – Detecção de pessoas durante a tarefa de seguimento.



Fonte: (ALMONFREY et al., 2018).

Observe que no momento T_B da Figura 62 outra pessoa está presente na área de trabalho apenas para mostrar que o sistema pode detectar mais de um pedestre por vez durante o experimento. No entanto, o robô está configurado para continuar seguindo a primeira pessoa. Apesar de nem todas as pessoas serem detectadas por todas as câmeras, elas são detectadas pela maioria das câmeras na maior parte das vezes. Portanto, o uso de uma rede de câmeras aumenta a taxa de detecção humana. Vale ressaltar que nosso detector não foi treinado usando imagens do nosso Espaço Inteligente e, por isso, são esperados alguns erros na detecção considerando apenas uma das câmeras.

Para completar, a Tabela 13 mostra uma breve análise da precisão do serviço de detecção humana. A ideia não é reivindicar a detecção do estado da arte, porque esse não é o foco deste trabalho. Queremos apenas avaliar a detecção no contexto dos requisitos da aplicação. A partir da tabela referida, é possível confirmar que o MR para a detecção de visão única é maior que para a visão múltipla. Isso mostra que as detecções da câmera são complementares e isso é mais evidente para a câmera 4.

O serviço de detecção humana possui um MR que atende às demandas da aplicação. Como pode ser visto na Tabela 13, um FPPI baixo é alcançado devido ao modelo utilizado no serviço de detecção de pessoas que prioriza essa métrica. A Figura 63 mostra a trajetória descrita pela pessoa e pelo robô durante o experimento. É possível ver pela figura que o serviço de detecção humana fornece informações adequadas para a tarefa a ser realizada. Os instantes T_A , T_B e T_C mostrados na Figura 62 são destacados na trajetória. Apenas

Tabela 13 – Análise da detecção de pessoas durante o experimento da aplicação de seguimento.

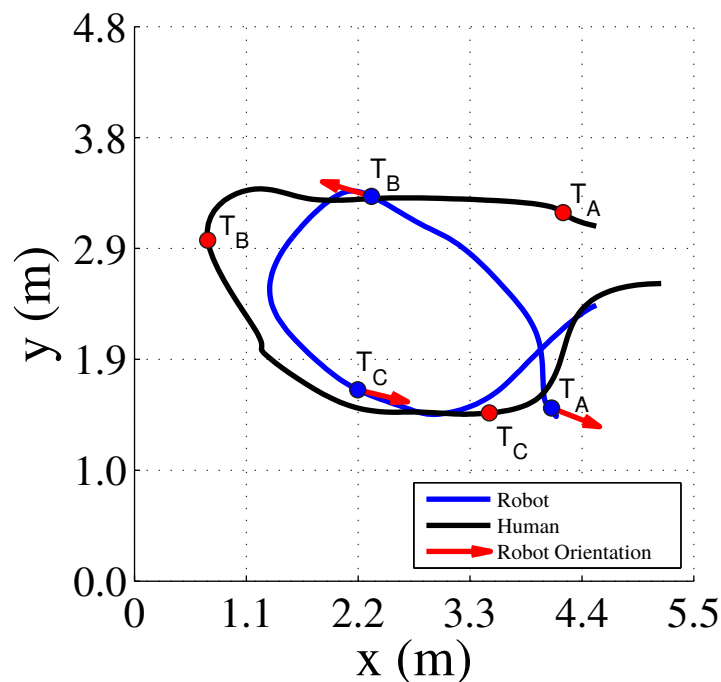
Device	MR (%)	FPPI
Camera 1	14.0	5.0×10^{-3}
Camera 2	26.4	2.7×10^{-2}
Camera 3	38.7	4.9×10^{-2}
Camera 4	62.3	3.8×10^{-2}
All Cameras	30.3	3.0×10^{-2}
Camera 1*	11.2	0
Camera 2*	11.2	2.2×10^{-2}
Camera 3*	14.2	3.8×10^{-2}
Camera 4*	21.3	4.4×10^{-2}
All Cameras*	13.3	2.6×10^{-2}

*A BB das imagens de outras câmeras é projetado na imagem da câmera atual. Isso mostra a vantagem de usar uma rede de câmeras.

Fonte: (ALMONFREY et al., 2018).

a trajetória da pessoa seguida pelo robô é mostrada para uma melhor compreensão. No início do experimento (instante T_A), o robô estava orientado a 330° no sentido anti-horário. Nesse caso, ele teve que girar aproximadamente 120° em seu eixo para iniciar a tarefa a seguimento de pessoa.

Figura 63 – Trajetória da pessoa e do robô durante a tarefa de seguir uma pessoa.



Fonte: (ALMONFREY et al., 2018).

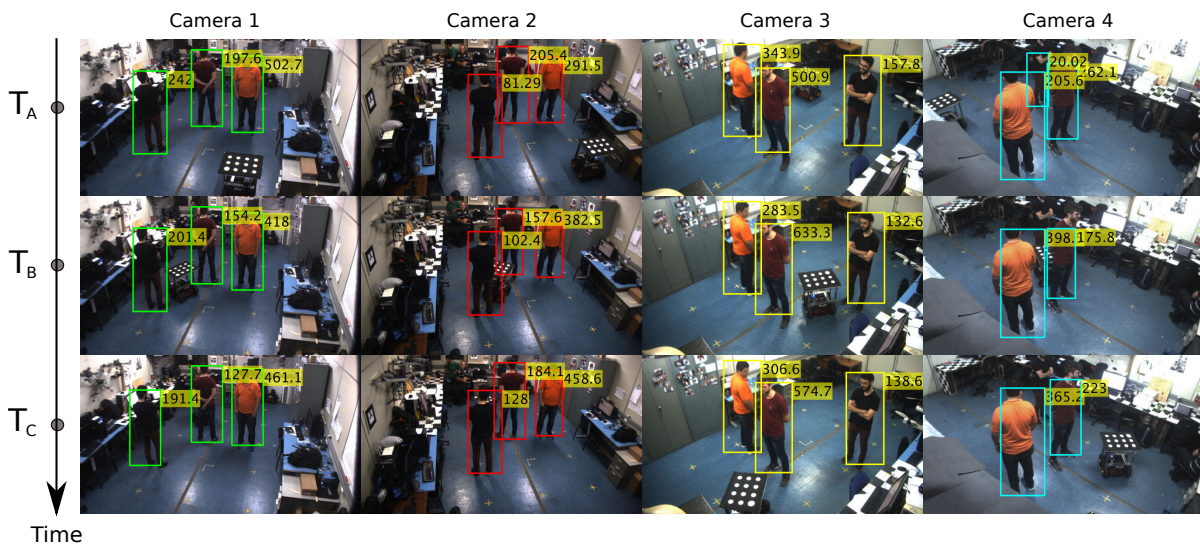
Como mencionado anteriormente, as posições das pessoas no mundo (3D) são

obtidas do BB nas imagens capturadas durante a mesma janela de amostragem. Para qualquer pessoa, várias posições podem ser obtidas, pois cada câmera pode fornecer um BB detectado. A posição 3D das pessoas é então estimada a partir de todas as imagens daquela pessoa dentro da janela. Em todas as experiências deste trabalho, as posições humanas são agrupadas se estiverem dentro de uma região de 0,5 m de diâmetro (geralmente o espaço médio no plano terrestre ocupado pelo corpo humano). Esta região é usada para representar a localização da pessoa no espaço tridimensional. Durante a navegação do robô, para melhorar a robustez contra falhas pontuais de detecções, um buffer é usado para acumular as dez últimas posições do ser humano seguidas pelo robô. A média desse buffer é adotada como a posição real.

Desvio de pessoas

Nesse experimento, o controlador do robô usa o serviço de detecção de pessoas para executar a tarefa de desvio. Tratamos as pessoas como obstáculos e planejamos o caminho que deve ser executado pelo robô usando uma estratégia de planejamento de caminho apresentada em (ŞUCAN; MOLL; KAVRAKI, 2012). A Figura 64 mostra o processo de detecção de pessoas durante a navegação do robô em três momentos diferentes (T_A , T_B e T_C).

Figura 64 – Detecção de pessoas durante a tarefa de desvio.

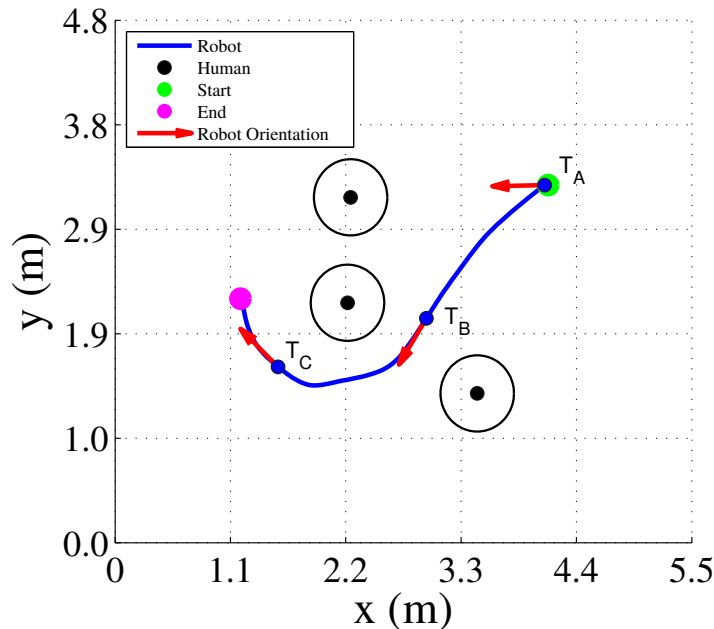


Fonte: (ALMONFREY et al., 2018).

A figura 65 mostra a trajetória descrita pelo robô e as posições das pessoas durante a navegação do robô. Usando as informações fornecidas pelo serviço de detecção, o robô pode navegar sem atingir as pessoas presentes no ambiente. Os instantes T_A , T_B e T_C

mostrados na Figura 64 são destacados na trajetória. A estratégia de planejamento do caminho procura o caminho com menos possibilidade de colisão.

Figura 65 – Trajetória realizada pelo robô e as posições das pessoas durante a navegação.



Fonte: (ALMONFREY et al., 2018).

A tabela 14 mostra uma análise de tempo de resposta para a aplicação de desvio de pessoas. Um tempo T_{HDS} superior a T_{RC} significa que o requisito da aplicação não está sendo atendido. Considerando que o tempo de processamento do serviço de detecção de pessoas é o que mais contribui para o tempo de resposta, isso significa que ele não está sendo capaz de processar as imagens no tempo exigido pela aplicação. T_{HDS} menor ou igual a T_{RC} implica que o tempo de resposta da tarefa está de acordo com os requisitos de tempo. Na prática, o valor de T_{HDS} deverá oscilar em torno de um valor próximo a T_{RC} .

Tabela 14 – Tempo de resposta.

$T_{RC}(s)$	$T_{HDS}(s)$
0.250	0.195

T_{RC} - é o requisito de tempo de resposta para a tarefa de controle do robô; é o tempo entre a captura das imagens e o recebimento do comando do robô.

T_{HDS} - é o tempo de resposta médio medido pela aplicação. Foram utilizadas 180 amostras para o cálculo da média.

Fonte: (ALMONFREY et al., 2018).

Mapa de ocupação do ambiente

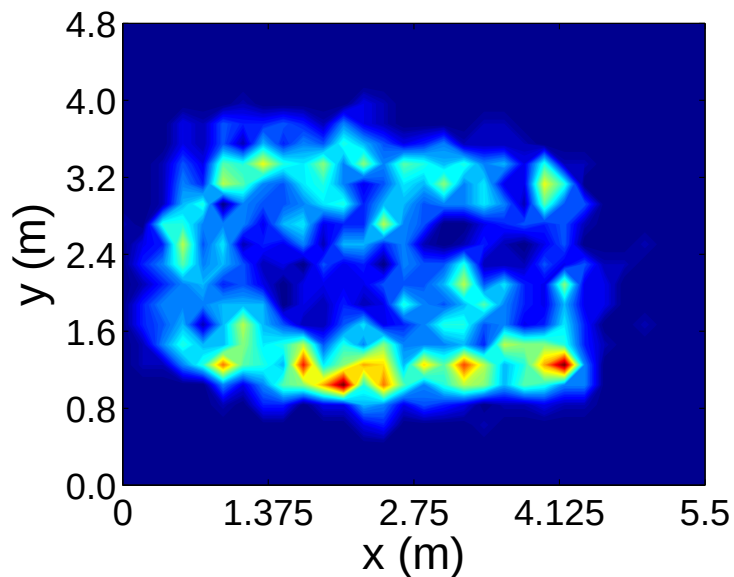
Neste último experimento, foi construído um mapa cumulativo de ocupação do Espaço Inteligente. O objetivo é mostrar os locais mais visitados do ambiente ao longo do

tempo.

A Figura 66 mostra o mapa de ocupação cumulativa. O Espaço Inteligente mede 4,8m x 7,3m. No entanto, devido ao posicionamento do sistema de câmeras, parte dos corpos humanos fica fora da imagem ou distorce a maioria das câmeras nos últimos 1,8 m do eixo horizontal. Como nenhuma pessoa é detectada na região referida, será mostrado apenas 4,8m x 5,5m de área. Esta é a área operacional do serviço de detecção. As pessoas foram orientadas a executar principalmente uma trajetória elíptica neste experimento e isso pode ser confirmado na Figura 66.

Em (SURIE; PARTONIA; LINDGREN, 2013), onde a solução é altamente estruturada, espera-se que um campo de visão maior prejudique a detecção. Como nesse trabalho pode-se lidar com áreas de trabalho menos estruturadas, um campo de visão maior beneficiará a detecção devido a uma área operacional maior.

Figura 66 – Mapa de Ocupação.



Fonte: (ALMONFREY et al., 2018).