



**Beatriz  
Saraiva Simões**

**Gémeo Digital de um laboratório vivo com  
Simulador**

**Digital Twin of Living Lab with Simulator**



**Beatriz  
Saraiva Simões**

**Gémeo Digital de um laboratório vivo com  
Simulador**

**Digital Twin of Living Lab with Simulator**

Projeto apresentado à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Licenciatura em Engenharia Computacional, realizada sob a orientação científica da Doutora Susana Isabel Barreto de Miranda Sargento, Professora catedrática do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

**o júri / the jury**

presidente / president

**Prof. Doutor Carlos António Delgado Sousa Brites**

professor associado do Departamento de Física da Universidade de Aveiro

vogais / examiners committee

**Prof. Doutor António José Ribeiro Neves**

professor associado c/ agregação do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro (Arguente)

**Prof. Doutor Susana Isabel Barreto de Miranda Sargento**

professora catedrática do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro (Orientadora)

**Inv. Marcos André Lopes Mendes**

investigador do Instituto de Telecomunicações da Universidade de Aveiro (Co-orientador)

## **agradecimentos / acknowledgements**

Quero expressar o meu mais profundo agradecimento à minha Mãe, Fátima, por me deixar crescer, aprender, experimentar e brincar. Por ter sempre uma palavra amiga e de carinho. Guiou-me ao que sou hoje.

Aos meus amigos, que são parte de mim. Todo o meu percurso foi mais leve graças a vós, obrigada por me deixarem partilhar e por me ajudarem a carregar a bagagem. Em especial à Mafalda, que entrou umas paragens à frente mas sentou-se em primeira classe.

À Sumeja, à Maria e à Mariana por me ajudarem a fazer o meu caminho mesmo quando eu não conseguia caminhar. Por me mostrarem que mesmo sem um joelho, existe sempre solução para chegar ao nosso objetivo. Ao João Gonçalo que desafiou a matemática e me mostrou que, de facto,  $dois + 2 = 5$ , obrigada por seres o irmão que nunca tive. Sem individualidades, o meu mais sincero obrigado à Equipa.

À Mariana, por não me deixar desistir e por ser sempre o meu conforto nos momentos mais difíceis. Obrigada por estares sempre aqui e por me mostrares que o mundo não é só isto.

À minha equipa de orientação, quero agradecer à catedrática Susana Sargento e ao Investigador Pedro Rito por me proporcionarem a oportunidade de trabalhar nesta área e me desafiarem a procurar novos conhecimentos.

Um agradecimento especial ao Investigador Marcos Mendes, por toda a orientação, conhecimento, aprendizagem e pela disponibilidade às piores horas do dia. Obrigada por me mostrar qual a direção a seguir.

**palavras-chave**

Gémeo Digital, Cidades Inteligentes, Plataformas com integração de dados em tempo real, Laboratório vivo, Deteção de objetos.

**resumo**

Este trabalho propõe o desenvolvimento de um Digital Twin (DT) de um *Living Lab*, com integração de dados do Aveiro Tech City Living Lab (ATCLL), centrando-se na representação visual de objetos detetados por câmaras e sensores, através da plataforma Unity.

A arquitetura proposta assenta em três blocos principais: (i) aquisição de dados em tempo real através de sistemas de Visão por computador, nomeadamente o modelo YOLOv11; (ii) transmissão dos dados anonimizados utilizando os protocolos MQTT e TCP; e (iii) visualização e instanciamento dos modelos no ambiente virtual do Unity, com mapeamento geográfico do terreno obtido via *OpenStreetMap*.

Os resultados demonstram uma correta correspondência entre objetos físicos e suas instâncias digitais, com tempos de processamento médios de 0,123 ms por objeto para uma única câmara, de 0,187 ms em cenários com quatro câmaras, e de 0,180 ms para todas as câmaras do sistema.

**keywords**

Digital Twin, Smart Cities, Real-time data platform, Living Lab, Object tracking.

**abstract**

This work proposes the development of a Digital Twin (DT) of a Living Lab, integrating data from the Aveiro Tech City Living Lab (ATCLL), with a focus on the visual representation of objects detected by cameras and sensors, through the Unity platform.

The proposed architecture is structured into three main modules: (i) real-time data acquisition through computer vision (CV) systems, particularly using the YOLOv11 model; (ii) transmission of anonymized data using MQTT and TCP communication protocols; and (iii) visualization and instantiation of the models within the Unity virtual environment, using geographical terrain mapping obtained via OpenStreetMap.

The results demonstrate a consistent correspondence between physical objects and their digital instances, with average processing times of 0.123 ms per object for a single camera, 0.187 ms in four-camera scenarios, and 0.180 ms when all system cameras are active.

**reconhecimento da  
utilização de ferramentas de  
IA**

**Reconhecimento da utilização de tecnologias e ferramentas de  
Inteligência Artificial generativa, software e outras ferramentas de  
apoio.**

Reconheço a utilização do ChatGPT (Open AI, <https://chat.openai.com>) para aprimorar a linguagem científica do meu trabalho e a utilização do tradutor automático DeepL (DeepL SE, <https://www.deepl.com/translator>) para traduzir de Inglês para Português o texto de artigos.

# Conteúdo

<b>Conteúdo</b>	<b>i</b>
<b>Lista de Figuras</b>	<b>iii</b>
<b>Lista de Tabelas</b>	<b>iv</b>
<b>Glossário</b>	<b>v</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Contexto e Motivação . . . . .	1
1.2 Objetivos . . . . .	2
1.3 Estrutura do Documento . . . . .	2
<b>2 Conceitos Teóricos</b>	<b>3</b>
2.1 Visão por Computador . . . . .	3
2.2 Cidades Inteligentes . . . . .	4
2.3 Gémeo Digital . . . . .	6
2.3.1 Origem dos Gémeos Digitais . . . . .	6
2.3.2 Definição de Gémeo Digital . . . . .	7
2.3.3 Dimensões dos Gémeos Digitais . . . . .	7
2.4 Tempo Real . . . . .	9
2.5 Unity como plataforma de desenvolvimento . . . . .	10
<b>3 Arquitetura</b>	<b>11</b>
3.1 Implementação . . . . .	11
3.2 Visão Geral da Arquitetura . . . . .	13
3.3 Especificações da máquina utilizada . . . . .	17
<b>4 Resultados</b>	<b>18</b>
4.1 Visão por Computador . . . . .	18
4.2 Representação geográfica do terreno . . . . .	19

4.3	Representação dos modelos . . . . .	19
4.4	Análise da visualização no sistema físico vs. digital . . . . .	20
4.5	Registo de tempos . . . . .	23
4.6	Considerações finais . . . . .	25
<b>5</b>	<b>Conclusão</b>	<b>26</b>
	<b>Referências</b>	<b>28</b>

# Lista de Figuras

2.1	À esquerda da Figura está representado o mapa ATCCL de Aveiro com a localização dos <i>smartlamp posts</i> existentes na cidade. À direita da Figura apresenta-se uma representação gráfica de um <i>smartlamp post</i> . Adaptado de [7], [10]. . . . .	5
2.2	Sensores disponíveis num <i>smartlamp post</i> . Adaptado de [11]. . . . .	6
2.3	Classificações para os UDT associadas às várias camadas existentes. Adaptado de [2]. . .	8
2.4	Comparação das performances dos modelos YOLO. Adaptado de [17]. . . . .	10
3.1	Comunicação entre dispositivos IoT via MQTT. Adaptado de [21]. . . . .	12
3.2	Comunicação entre duas aplicações via TCP. Adaptado de [22]. . . . .	13
3.3	<i>Pipeline</i> para construção do DT no Unity. . . . .	14
3.4	Arquitetura do sistema para representação dos dados em tempo real no DT. . . . .	15
3.5	Em 3.5a apresenta-se a funcionalização do algoritmo <i>ray-casting</i> e em 3.5b a implementação deste algoritmo no DT desenvolvido. . . . .	17
4.1	Representação da cidade . . . . .	19
4.2	Representação dos modelos obtidos e utilizados como instâncias no Unity. . . . .	20
4.3	DT do P35-1. . . . .	21
4.4	DT do P22. . . . .	21
4.5	Objetos detetados pelo NAP-VISION do P35. Quatro câmaras na mesma localização que apresentam orientações diferentes. . . . .	22
4.6	DT do P35 com integração das quatro câmaras. . . . .	23
4.7	Comparação dos tempos de processamento no Unity para diferentes configurações de câmara no sistema P35. . . . .	23
4.8	Tempo de processamento no Unity em função do número de objetos para todas as câmaras implementadas no sistema. . . . .	24
4.9	Tempo de processamento no Unity para instanciar cada tipo de modelo. . . . .	24

# **Lista de Tabelas**

4.1 Correspondência de coordenadas no sistema Unity às coordenadas reais, com legenda de cores para a visualização da Figura 4.1. . . . .	19
---	----

# Glossário

<b>ATCLL</b>	Aveiro Tech City Living Lab	<b>NAP</b>	<i>Network Architectures and Protocols</i>
<b>COCO</b>	<i>Common Objects in Context</i>	<b>NASA</b>	<i>National Aeronautics and Space Administration</i>
<b>DT</b>	<i>Digital Twin</i>	<b>OSM</b>	<i>Open Street Map</i>
<b>FPS</b>	frames per second	<b>SC</b>	<i>Smart Cities</i>
<b>IA</b>	Inteligência Artificial	<b>SCP</b>	<i>Smart City Platforms</i>
<b>IoT</b>	<i>Internet of Things</i>	<b>SLP</b>	smartlamp posts
<b>IT</b>	Instituto de Telecomunicações	<b>TCP</b>	<i>Transmission Control Protocol</i>
<b>mAP</b>	<i>Mean Average Precision</i>	<b>UDT</b>	<i>Urban Digital Twin</i>
<b>MEC</b>	<i>Multi-access Edge Computing</i>	<b>VC</b>	Visão por Computador
<b>ML</b>	<i>Machine Learning</i>	<b>YOLO</b>	<i>You Only Look Once</i>
<b>MQTT</b>	<i>Message Queuing Telemetry Transport</i>		

# Introdução

As cidades são das mais antigas criações da humanidade. Em contrapartida, a tecnologia representa uma das áreas mais marcadas pelo progresso constante e acelerado. A integração da tecnologia nas cidades atuais pretende torná-las mais habitáveis, sustentáveis e resilientes, dando assim origem ao conceito de *Smart Cities* (SC) [1]. Contudo, o desenvolvimento urbano exige um planeamento estratégico e rigoroso, uma vez que as decisões tomadas afetam diretamente a qualidade de vida dos cidadãos. Neste contexto, a tecnologia *Digital Twin* (DT) surge como um modelo virtual preciso de um sistema físico, permitindo que seja realizada uma monitorização das cidades em *real-time* [2].

## 1.1 CONTEXTO E MOTIVAÇÃO

Nos últimos anos, as crescentes exigências associadas à complexidade dos sistemas urbanos tornaram evidente a necessidade de abordagens mais integradas, inteligentes e adaptativas. A globalização, a pressão para a redução de custos, bem como a urgência de responder a mudanças ambientais, sociais e tecnológicas vieram expor algumas fragilidades dos atuais processos de planeamento, monitorização, e tomada de decisão. As limitações identificadas têm impulsionado a adoção de ferramentas tecnológicas, entre as quais se destaca o DT. Esta permite a criação de réplicas virtuais de um produto, processo ou sistema físico, capazes de refletir o estado atual e os seus comportamentos, e ajudar no apoio à decisão. A integração de dados provenientes de sensores *Internet of Things* (IoT) concede aos DT a capacidade para simular cenários em *real-time*, avaliar cenários *what-if* e identificar falhas no planeamento estratégico urbano implementado [2]. A sua aplicação em contexto urbano tornou-se particularmente relevante face aos desafios que as grandes áreas metropolitanas têm enfrentado resultantes do rápido crescimento populacional, das desigualdades socioeconómicas e também das alterações climáticas. Neste contexto, as SC podem ser vistas como uma resposta a estes obstáculos, orientadas para a melhoria da qualidade de vida dos seus habitantes, através da integração de novas tecnologias e também de Inteligência Artificial (IA) [3].

Na pesquisa do termo “digital twins” na base de dados Scopus, identificaram-se mais de vinte e nove mil (29 000) publicações até 29 de fevereiro de 2025, considerando os campos do título, resumo e palavras-chave, revelando um crescente interesse por parte da comunidade científica nesta tecnologia. Este crescimento reflete a aposta de várias cidades em processos de digitalização, com significativos investimentos relacionados ao desenvolvimento e implementação dos DT [4].

A motivação subjacente a este trabalho relaciona-se com o elevado potencial dos DT. Em oposição às simulações tradicionais, os DT permitem a integração de dados em *real-time*, permitindo uma comunicação direta com o mundo físico através da articulação com diversas tecnologias, como é o caso dos sensores IoT. Desta forma, representam uma ferramenta importante no apoio à tomada de decisão do planeamento urbano, favorecendo a exploração de diversos cenários, a antecipação de eventos e a simulação de outros. Acrescem ainda a possibilidade de permitirem análises detalhadas que permitam responder à crescente complexidade destes ambientes [4].

## 1.2 OBJETIVOS

O objetivo principal deste trabalho é o desenvolvimento de um DT para uma cidade, com integração de dados em *real-time*. Os dados serão provenientes de sistemas físicos de monitorização, como câmeras e algoritmos de Visão por Computador (VC) para a deteção automática de objetos. Para tal, é necessário a construção prévia de um sistema que espelhe virtualmente o comportamento de um sistema físico.

## 1.3 ESTRUTURA DO DOCUMENTO

O presente documento está estruturado em cinco capítulos, cada um dos quais descreve de forma pormenorizada as diferentes etapas do trabalho desenvolvido. O Capítulo 1 apresenta uma breve introdução ao tema e apresenta os objetivos do projeto. No Capítulo 2 efetua-se uma análise aprofundada, com destaque para os conceitos teóricos que estão na base deste projeto. O Capítulo 3 descreve a arquitetura do sistema proposto. O Capítulo 4 apresenta os resultados obtidos bem como uma análise crítica e discussão dos mesmos. E por fim, o Capítulo 5 sumariza as principais conclusões do trabalho e propõe perspetivas futuras, tendo em consideração o estado atual do projeto.

# CAPÍTULO 2

## Conceitos Teóricos

Se a IA funciona como um cérebro humano, pode-se entender a VC como os olhos, fornecendo a percepção visual para identificação de objetos, formas, cores e padrões de movimento. O conceito de DT pretende espelhar as características físicas de algo, num mundo digital, onde as possibilidades de visualização, teste e simulações são extensas. O ponto de convergência dos conceitos de VC e SC, é o ponto ideal para o desenvolvimento de um *Urban Digital Twin* (UDT), que se traduz por uma representação digital da cidade em estudo [5]. A compreensão de um UDT como um mundo extenso de possibilidades, permite alargar a visão a todas as áreas que fazem parte desta tecnologia, incentivando uma abordagem crítica e criativa que pode impulsionar o desenvolvimento de novas soluções, permitindo assim, que cada tecnologia cresça individualmente. À medida que cada tecnologia associada aos UDT se desenvolve de forma autónoma, consolida-se o seu contributo específico. Quando integradas, essas tecnologias complementam-se e reforçam-se mutuamente, potenciando a eficácia e a abrangência dos UDT [5].

### 2.1 VISÃO POR COMPUTADOR

Uma das subáreas da IA profundamente relacionada com a aprendizagem automática *Machine Learning* (ML) e que tem vindo a registar um aumento crescente nos últimos anos, é a VC. Esta dedica-se à aquisição, processamento e interpretação de dados, como imagens e vídeos, tendo como principal objetivo funções como a deteção, reconhecimento e a classificação de objetos [6]. Para tal, as câmaras representam a principal fonte de aquisição de dados permitindo capturar informações que serão posteriormente transmitidas aos algoritmos de VC [6]. A eficiência dos sistemas que integram esta tecnologia dependem, em larga escala, do desempenho dos algoritmos implementados, os quais permitem extrair padrões e identificar anomalias, conferindo aos modelos a capacidade de generalização de forma autónoma e sem necessidade de supervisão direta [6].

No contexto das SC, a VC desempenha um papel fundamental para a monitorização e gestão urbana. Através da análise contínua de imagens captadas por câmaras distribuídas no

espaço da cidade, é possível implementar soluções de controlo de tráfego, deteção automática de acidentes, análises de padrões de mobilidade e apoio à segurança pública [7]. Por exemplo, estes sistemas podem identificar em tempo real zonas com maior fluxo de tráfego ou períodos de maior afluência de pessoas em áreas específicas. Com base nestes dados, as entidades responsáveis podem atuar de forma consciente e fazer uma gestão adaptada da cidade, contribuindo para decisões mais informadas, sustentáveis e eficientes.

No entanto, como referido em [8], apesar das inúmeras vantagens associadas à integração da VC, é importante reconhecer os desafios éticos e legais que surgem da sua utilização, nomeadamente ao nível da proteção de dados. A recolha de dados em espaços públicos, quando não devidamente regularizada, pode conduzir à construção de perfis de determinados indivíduos, como as suas rotinas e atividades, comprometendo a confidencialidade e a segurança da informação pessoal. Para mitigar estes riscos, têm vindo a ser implementadas normas por parte de entidades públicas e privadas quanto à utilização responsável desta tecnologia. Em particular, o Regulamento Geral de Proteção de Dados, em vigor na União Europeia, estabelece um conjunto de normas específicas aplicáveis aos sistemas de VC [8]. Na literatura, Das *et al.* identificam três tipos de privacidade a considerar na aplicação de técnicas de VC: a privacidade dos dados, a privacidade dos modelos e a privacidade na fase de implementação. A consideração destes três níveis é essencial para garantir que os avanços científicos e tecnológicos se alinham com os princípios fundamentais de proteção dos direitos individuais no espaço urbano digital [9].

A integração da VC com outras tecnologias, como a IoT, *cloud computing*, o *edge computing* e o *Multi-access Edge Computing* (MEC), tem vindo a reforçar a sua utilidade em ambientes complexos, viabilizando respostas em tempo quase real [7]. Este progresso é especialmente significativo quando os sistemas de VC são integrados com a tecnologia de DT, permitindo que a informação visual captada no espaço físico seja diretamente incorporada nos modelos virtuais da cidade. Esta ligação entre a percepção visual e a modelação digital permite não só o espelhamento do estado atual do modelo físico, mas também a antecipação e a previsão de cenários, e a simulação de eventos futuros [2].

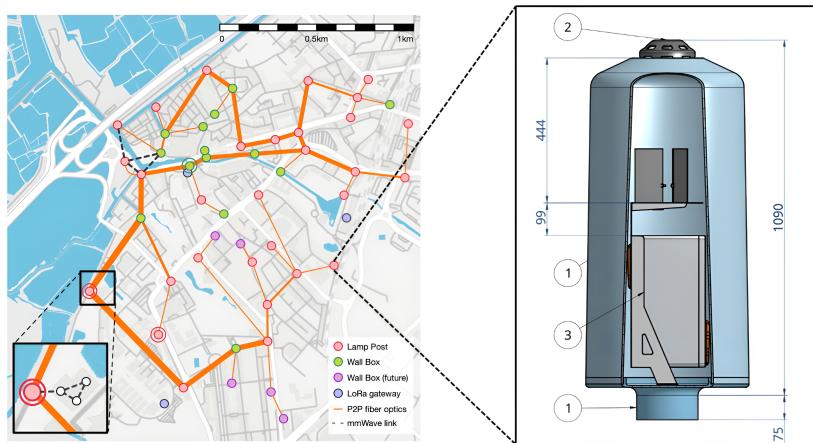
## 2.2 CIDADES INTELIGENTES

Apesar de existirem várias definições para o conceito de SC, os autores têm uma compreensão geral que se baseia nos avanços tecnológicos em torno da transformação do ambiente urbano. Segundo Caprari *et al.* “o conceito de SC está indiscutivelmente associado a processos de otimização e de modernização tecnológica do ambiente urbano, ou seja, tornar uma cidade “inteligente” significa criar condições para melhorar a qualidade de vida dos seus habitantes, valorizando locais (de trabalho, comércio, lazer, ...) e facilitando o acesso aos serviços (saúde, cultura, mobilidade, energia,...)”. Assim, o processo de digitalização das cidades surge fortemente relacionado à questão dos DT, sendo, por vezes, confundido com o conceito de Plataformas para Cidades Inteligentes *Smart City Platforms* (SCP). As SCP são infraestruturas de dados que tem como objetivo recolher, integrar e disponibilizar dados provenientes de sensores e outros sistemas digitais. Estas plataformas constituem a base sobre

a qual os DT podem ser implementados, uma vez que fornecem os dados e os mecanismos necessários para a sua criação e funcionamento [1]. Paralelamente à evolução do conceito das SC regista-se também uma tendência crescente para a implementação de *Living Labs* urbanos. Estes espaços caracterizam-se como ecossistemas abertos que permitem o desenvolvimento, teste, implementação e avaliação de soluções tecnológicas em contexto real, permitindo às cidades avaliar o impacto das novas soluções aplicadas a diferentes áreas, como a mobilidade, energia, segurança, entre outros [10].

Neste contexto, a integração de UDT em *Living Labs* surgem com particular relevância dado o seu potencial para simular, antecipar e otimizar processos. Os UDT funcionam como uma réplica virtual da cidade, alimentados por um fluxo contínuo de dados recolhidos por infraestruturas físicas e sensores. Como referido em [3], os *Living Labs* fornecem um ambiente ideal para recolher dados com precisão e em tempo real, ou seja, dados contextualizados e atualizados para alimentar o DT, viabilizando simulações e análises fáceis ao contexto real da cidade. Assim, o sincronismo do conceito de UDT e *Living Labs*, permite reduzir custos, testar a implementação de soluções antes da sua implementação definitiva e assistir as cidades numa melhor tomada de decisão [3].

Aveiro Tech City Living Lab (ATCLL) é um projeto desenvolvido pelo Instituto de Telecomunicações (IT) em parceria com a Câmara Municipal de Aveiro. Rito *et al.* apresentam uma descrição completa da arquitetura e implementação de serviços testados no ATCLL. A plataforma inclui dispositivos IoT, como radares, LiDAR e câmaras. Integra várias redes e protocolos de comunicação, espalhados por quarenta e quatro (44) pontos no espaço da cidade. Na Figura 2.1 podem ser observadas as suas localizações, e, a estrutura física que integra os elementos mencionados, a esta dá-se o nome de *smartlamp posts* (SLP). Os SLP incluem ainda tecnologias como *edge computing*, *cloud management* e MEC, [10].



**Figura 2.1:** À esquerda da Figura está representado o mapa ATCCL de Aveiro com a localização dos *smartlamp posts* existentes na cidade. À direita da Figura apresenta-se uma representação gráfica de um *smartlamp post*. Adaptado de [7], [10].

No âmbito deste projeto, focam-se dois sistemas de captação de dados fundamentais para o DT desenvolvido, as câmaras e os radares. Estes, incorporados nos SLP, estão destacados na Figura 2.2, em (1) um exemplo das câmaras utilizadas, e em (2) um exemplo dos radares.



**Figura 2.2:** Sensores disponíveis num *smartlamp post*. Adaptado de [11].

### 2.3 GÉMEO DIGITAL

O número de artigos publicados sobre o tema DT nas mais diversas áreas aumentou drasticamente na última década, registando-se sobretudo publicações relacionadas com a integração de DT nas SC e em questões de planeamento urbano [2], [5]. Embora a existência de uma representação virtual da cidade não implique, por si só, a resolução dos desafios associados ao espaço da cidade, um UDT pode constituir uma ferramenta com potencial para integrar princípios de segurança, sustentabilidade e equidade. Neste contexto, coloca-se a questão: “O que é um DT e de que forma pode contribuir para o desenvolvimento das cidades?”

#### 2.3.1 Origem dos Gémeos Digitais

A origem do conceito de DT está intrinsecamente associada à evolução da engenharia aeroespacial e à indústria de fabrico, destacando-se em particular, o papel fundamental desempenhado pelo programa *Apollo* da Administração Nacional da Aeronáutica e Espaço *National Aeronautics and Space Administration* (NASA), em 1969. Embora o termo DT não tivesse sido formalmente introduzido na altura, a missão já evidenciava os princípios fundamentais desta tecnologia. Foram construídas duas naves espaciais idênticas: uma foi lançada para o espaço com fins operacionais, enquanto que a outra permaneceu em terra num ambiente simulado que replicava as condições da nave em missão, permitindo acompanhar em tempo real o seu funcionamento. Anos mais tarde, em 2003, Michael Grieves apresentou um modelo conceptual no âmbito da gestão do ciclo de vida do produto (*Product Lifecycle Management*). Este modelo previa a existência de um espaço físico e de um espaço virtual interligados, permitindo o fluxo bidimensional de dados e informação entre ambos. Inicialmente designado por “Modelo de Espelho de Informação”, este conceito viria mais tarde a ser reconhecido como o antecessor do atual conceito de DT [12]. Embora esteja presente na literatura há cerca de duas décadas, o primeiro artigo a explorar esta tecnologia com mais detalhe foi publicado apenas em 2012, num estudo da NASA, introduzindo formalmente o termo DT [13]. Desde então o desenvolvimento desta tecnologia tem-se intensificado, impulsionado pelos avanços científicos e tecnológicos em áreas como as comunicações, IoT, *cloud computing*, *edge computing*, a análise de grande volume de dados (*big data analytics*) e a

IA. Este progresso acelerado resultou num aumento significativo tanto da literatura bem como das aplicações práticas associadas ao conceito de DT [14]. Como consequência, observou-se uma adaptação progressiva do DT a novos domínios científicos e industriais, estendendo-se à saúde, às SC, à robótica, aos sistemas de transporte, entre outras [1].

### 2.3.2 Definição de Gémeo Digital

Na literatura são apresentadas diversas conceptualizações de DT, refletindo a sua crescente aplicação em múltiplos domínios do conhecimento. Esta diversidade de conceptualizações resulta da adequação do conceito às especificidades de cada área. No entanto, pode generalizar-se a definição de um DT como uma réplica virtual e dinâmica de um objeto, processo, produto ou de um sistema físico, caracterizada pela integração de dados em tempo real de diferentes fontes [15]. No caso dos sistemas físicos, o DT assume-se com uma representação digital que incorpora as propriedades estruturais, o estado operacional e o comportamento do seu gémeo físico, utilizando modelos computacionais e dados continuamente atualizados. Através do DT é possível simular o desempenho do sistema, integrando os vários fatores do ambiente em que está inserido. A criação de um DT exige a recolha e integração de dados provenientes de diversos sistemas de informação, incluindo informações sobre as características de fabrico, condições do ambiente em que atuam, propriedades intrínsecas à entidade e até resultados de análise de dados anteriores. Esta informação é processada e incorporada num modelo digital que espelha o comportamento do sistema físico, permitindo a sua monitorização, diagnóstico e otimização [15]. Os sensores IoT desempenham um papel fundamental neste processo, garantindo a recolha de dados diretamente a partir da entidade física. Estes dados são transmitidos para o DT, em tempo real, que os utiliza para atualizar e ajustar a sua representação. Através desta ligação permanente entre os domínios físico e virtual, é possível não apenas a observação passiva, mas também a projeção de cenários futuros com base em simulações estocásticas. Esta capacidade sustenta a tomada de decisão pró-ativa, a otimização de operações, a mitigação de riscos e a redução de custos associados [15].

### 2.3.3 Dimensões dos Gémeos Digitais

Os DT no contexto das SC podem ser considerados como UDT. Com os avanços das tecnologias que integram as SC, posicionam os UDT na vanguarda da modelação, simulação e análise urbana sendo altamente precisos [5]. Na literatura, este tipo específico de DT pode ser classificado em quatro categorias de acordo com o seu nível de funcionalidade e capacidade de interação com o sistema físico. A distinção entre elas baseia-se no nível de integração dos dados, na capacidade analítica e no grau de autonomia. Acresce-se ainda que cada categoria pode agrupar uma ou mais camadas hierárquicas de especialização.

Desta forma apresenta-se de seguida, e na Figura 2.3, uma visão geral da classificação e das respetivas camadas, baseado em [2]:

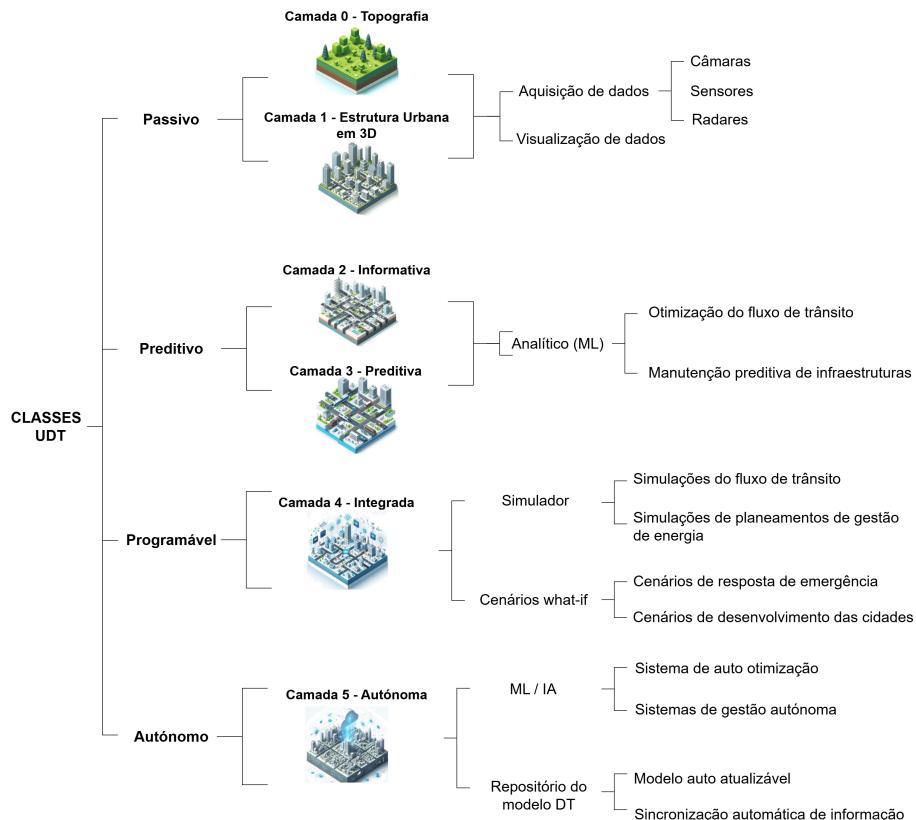
**UDT passivo** : Os UDT do tipo passivo são os mais rudimentares e estruturais de todos, em que o seu objetivo é mapear o terreno da cidade. Compreende a “**Camada 0 - Topografia**”, que representa apenas os elementos geográficos estáticos como canais,

lagos e zonas de vegetação; e a “**Camada 1 - Estrutura urbana em 3D**” que representa tridimensionalmente as infraestruturas, bens e serviços.

**UDT preditivo** : Os UDT preditivos permitem uma interação entre o ambiente físico e o ambiente virtual. Para além de representarem a cartografia da cidade, acompanham também os dados dos sensores em tempo real e fazem uma análise analítica, permitindo às entidades responsáveis antecipar e planear cenários. Inclui a “**Camada 2 - Informativa**”, esta verifica a exatidão da camada anterior através da recolha de dados em tempo real para monitorização; e a “**Camada 3 - Preditiva**”, que aplica modelos analíticos e algoritmos de ML para a antecipação de eventos e, assim, testar e programar diferentes cenários.

**UDT programável** : Correspondem à “**Camada 4 - Integrada**”, a qual incorpora um *software* dedicado a realizar simulações de diferentes cenários *what-if* com total controlo do ambiente virtual. Isto permite otimizar operações e distribuir/utilizar os recursos da cidade da melhor forma.

**UDT autónomo** : Os UDT com *feedback* inteligente são o último nível de desenvolvimento dos DT existentes até à data. Também com uma única camada, a “**Camada 5 - Autónoma**”, utilizam ML para tomar decisões estratégicas a cada passo do planeamento e das simulações para viabilizar soluções para qualquer tipo de problemas que possam surgir dentro da cidade.



**Figura 2.3:** Classificações para os UDT associadas às várias camadas existentes. Adaptado de [2].

## 2.4 TEMPO REAL

A deteção de objetos em tempo real é um dos tópicos mais importantes na área de VC. Estes sistemas utilizam algoritmos de processamento de imagem e ML para funções como a deteção de vários objetos, a condução autónoma, a robótica, entre outros. Os sistemas de VC são capazes de identificar e categorizar múltiplos objetos, mesmo que distintos [16].

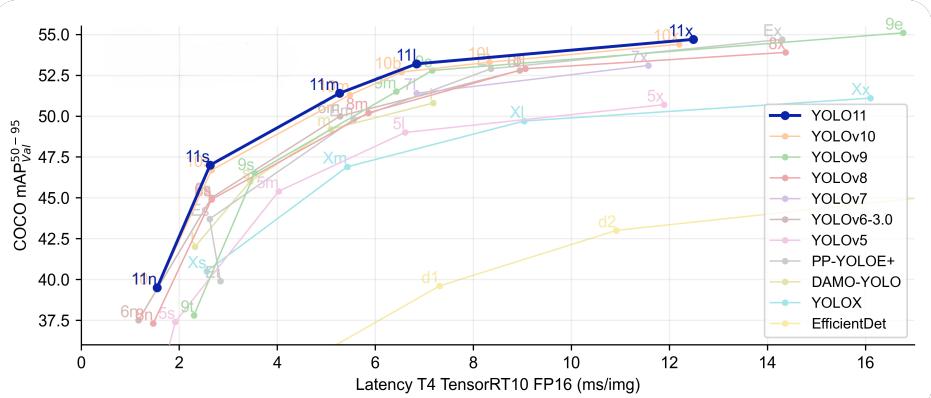
O trabalho desenvolvido por Mendes *et al.* propõe uma nova abordagem para a deteção e monitorização de objetos em tempo real, no contexto da mobilidade e segurança rodoviária. Utiliza algoritmos de VC e *edge computing*, com recurso a câmeras estáticas e infraestruturas de baixo custo, estes sistemas fazem parte da implementação de vários testes em contexto de uma SC, ATCLL. O procedimento apresentado baseia-se na utilização de uma única câmara (estática) para detetar, classificar e localizar objetos com movimento, como veículos, bicicletas e peões, permitindo rastreá-los, estimar a distância entre eles e a inferência de potenciais colisões. Destaca-se este trabalho na área da deteção de objetos em tempo real uma vez que apresenta a capacidade de transformar dados visuais brutos, em informação que pode ser consumida e interpretada por sistemas informáticos, como por exemplo, a velocidade, a aceleração, vetores de movimento e a distância de travagem. Para os sistemas de apoio à decisão em tempo real e para a prevenção de acidentes esta informação é fundamental, especialmente em zonas urbanas com elevada presença de piões [8].

Existem vários algoritmos para a deteção de objetos, entre os quais, o *You Only Look Once* (YOLO), *Region Based Convolutional Neural Networks* (R-CNN), *Fuzzy-Based Convolutional Neural Network* (F-CNN). A principal diferença entre eles é a otimização do desempenho computacional. Os diferentes modelos do YOLO centram-se na melhoria da velocidade de inferência de várias GPUs, ou seja, são uma melhor escolha para cenários que apresentam um grande volume de dados e exigem precisão em tempo real, enquanto que os outros mencionados focam-se em cenários com restrições de hardware, nomeadamente em sistemas de *edge computing*, onde a inferência é realizada em dispositivos com capacidade computacional limitada, geralmente baseados em CPU [16]. Em [7], Mendes, apresenta uma visão geral dos principais algoritmos para a deteção de objetos e é também fundamentado o porquê da escolha do YOLO como método mais adequado para o desenvolvimento do trabalho proposto. De uma forma geral, o YOLO é um algoritmo de deteção de objetos em tempo real, rápido, preciso e capaz de funcionar em vários dispositivos, incluindo *smartphones*, drones e dispositivos incorporados *embedded devices*.

Até à data, o YOLO11 é o último modelo lançado, pela equipa *Ultralytics*, a 30 de setembro de 2024. Esta versão suporta várias tarefas, como a deteção de objetos, a segmentação e classificação de imagens, e a estimativa da posição e orientações específicas dos objetos. Entre as principais melhorias introduzidas, destaca-se a reformulação dos hiperparâmetros de treino, o que permitiu otimizar a forma como as características dos objetos são identificadas e, consequentemente, aumentar a velocidade de inferência do modelo, mantendo elevados níveis de precisão [17].

Os últimos modelos do YOLO foram testados para um mesmo conjunto de dados, o

*Common Objects in Context* (COCO), [18], é um dos conjuntos de dados mais utilizados para treinar e testar aplicações de VC, possui cerca de trezentas e trinta mil (330 000) imagens sendo que mais de duzentas mil (200 000) são *labeled*. Após o teste dos modelos para o conjunto de dados COCO, o YOLOv11 atinge uma precisão média *Mean Average Precision* (mAP) mais elevada utilizando menos recursos computacionais comparativamente às versões anteriores, o que o torna computacionalmente eficiente sem comprometer a exatidão, como mostra a Figura 2.4, [19].



**Figura 2.4:** Comparação das performances dos modelos YOLO. Adaptado de [17].

## 2.5 UNITY COMO PLATAFORMA DE DESENVOLVIMENTO

O Unity é um ambiente de desenvolvimento integrado (IDE) que permite aos utilizadores criar jogos, animações 3D, entre outras ferramentas. O Unity Engine foi desenvolvido em C++ e em C#, com suporte para código em C# e JavaScript. Utiliza um editor que compila automaticamente o código e injeta as modificações para o ambiente virtual. O processo de compilação utiliza os compiladores de C#. Em relação ao ambiente virtual possui o seu próprio sistema de eixos e as respetivas coordenadas a partir de um ponto (0,0,0). Esta plataforma permite ainda o desenvolvimento de aplicações com a componente tempo real. É também uma ferramenta que tem vários protocolos com instituições de ensino e permite que estudantes, investigadores ou qualquer pessoa associada à instituição possa desenvolver projetos sem custos associados, como é o caso deste projeto [20].

# CAPÍTULO 3

## Arquitetura

Os conceitos analisados no Capítulo 2 fornecem uma base teórica importante para a compreensão da estrutura proposta do DT desenvolvido. A utilização da VC neste projeto tem como objetivo a extração e interpretação de dados provenientes de um ambiente urbano, recorrendo à infraestrutura da SC em estudo é possível a utilização de dados em tempo real. Estes conceitos são fundamentais para identificar padrões e retirar informações importantes, posteriormente utilizadas pelos UDT para os mais diferentes objetivos.

Este capítulo pretende descrever a arquitetura do sistema desenvolvido, os seus componentes e todos os softwares e protocolos utilizados, tendo em conta o fluxo e o armazenamento de dados. Descreve também como é que os diferentes tipos de dados são recolhidos e como é feito o processamento para a visualização dos mesmos.

### 3.1 IMPLEMENTAÇÃO

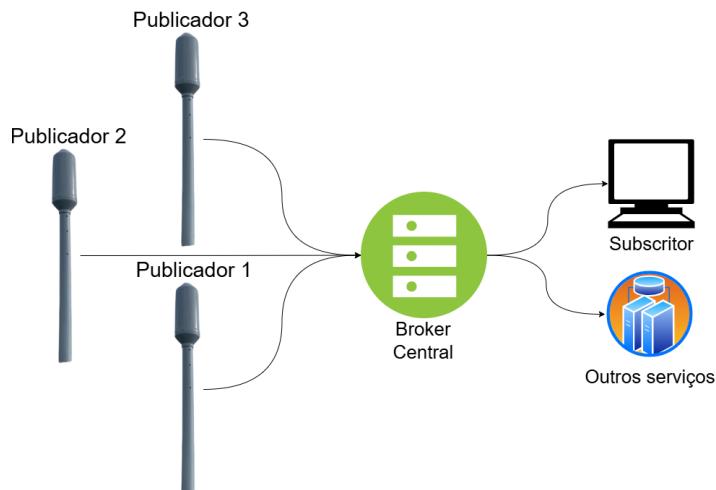
Através dos algoritmos de VC é possível fazer a deteção, o *tracking* e a classificação dos diferentes objetos em tempo real, tornando possível a visualização da realidade da cidade em ambiente virtual. Deste modo, para o desenvolvimento deste projeto consideram-se três grandes blocos: **(i)** deteção de objetos e aquisição de dados em tempo real; **(ii)** transmissão e partilha dos dados anonimizados através de vários protocolos, e, transformação dos dados generalizados para um formato que o Unity compreenda; **(iii)** visualização e representação digital num ambiente virtual. Os três blocos possuem características diferentes, o que implica abordagens diferentes para cada um.

Para **(i)**, a aquisição de dados é feita essencialmente através de câmaras e radares, estas infraestruturas fazem parte dos serviços incluídos no NAP-VISION, um projeto desenvolvido pelo grupo *Network Architectures and Protocols* (NAP) do IT de Aveiro, que para além dos referidos disponibiliza mais serviços de aquisição de dados. Assim, seguiu-se uma abordagem muito semelhante à adotada por Mendes, seguindo-se uma descrição de acordo com a do autor, [7]:

1. Deteção de objetos: Tem como objetivo identificar a presença e o posicionamento dos objetos numa cena.
2. *Tracking* dos objetos: É uma técnica aplicada numa sequência de *frames*, em que identifica o mesmo objeto em todos os *frames*, que podem ou não estar em posições diferentes, e mantém a sua identidade quando deteta que a sua posição foi alterada.
3. Localização dos objetos: É o processo de identificação da localização do objeto, e as respetivas margens, numa imagem ou vídeo. De momento, quando localizado é apresentado uma caixa, por exemplo um retângulo, à volta do objeto.

Em relação a (ii), apresenta-se de seguida uma visão geral dos protocolos que permitem a transmissão de dados: *Message Queuing Telemetry Transport* (MQTT) e *Transmission Control Protocol* (TCP).

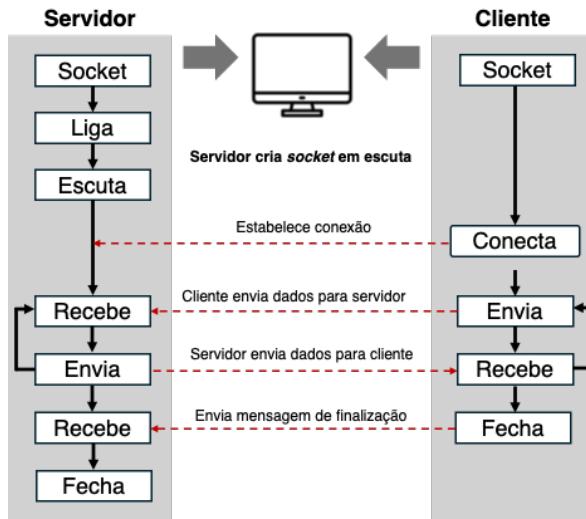
O **MQTT** é um protocolo de comunicação constituído por quatro componentes, o *broker* que é um dispositivo central, os clientes (nós da IoT), tópicos e mensagens. O MQTT segue o método de publicar-subscrever. Os clientes são nós da IoT, isto é, são dispositivos físicos interligados que podem ser sensores, atuadores, ou qualquer dispositivo que interaja com o sistema, em que podem funcionar como publicadores (do inglês, *publishers*), subscritores (do inglês, *subscriber*) ou ambos. Os clientes têm associadas a si três funções principais: seleção do tópico, publicação e subscrição do tópico. Os clientes comunicam entre si através de um cliente central (nó central) denominado *broker*, este permite que os nós da IoT publiquem ou subscrevam tópicos, ou ambos ao mesmo tempo, como representado na Figura 3.1. Assim os nós da IoT não comunicam diretamente entre si, a informação é sempre recebida e enviada pelo *broker*. O tópico é semelhante a um caminho (do inglês, *path*), é delimitado por “/” e permite aceder às mensagens através do *broker*, o tópico também contém informação sobre os nós de origem e de destino das mensagens transmitidas na rede [21].



**Figura 3.1:** Comunicação entre dispositivos IoT via MQTT. Adaptado de [21].

O protocolo TCP permite uma transmissão de dados entre vários *hosts* através da uma rede IP. Segue um modelo cliente-servidor no qual, tipicamente, o cliente inicia a comunicação

enquanto o servidor aguarda a resposta do pedido do cliente. Este fluxo de dados é representado na Figura 3.2, [22].



**Figura 3.2:** Comunicação entre duas aplicações via TCP. Adaptado de [22].

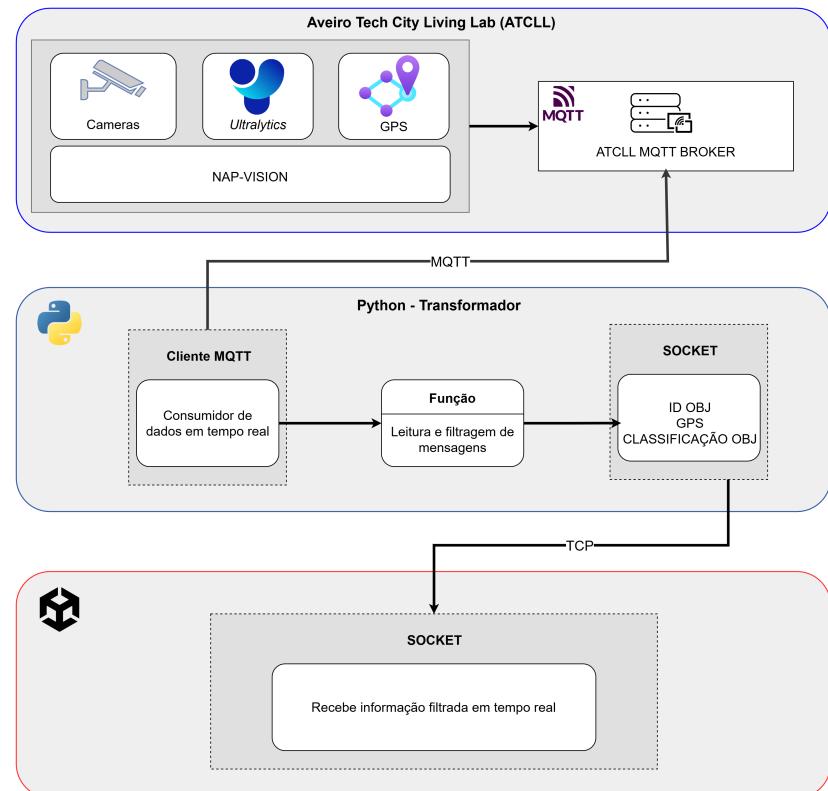
Considerando (iii), procedeu-se à seguinte sequência de etapas:

1. Mapeamento da cidade: Exportou-se o terreno da zona de interesse, através das coordenadas correspondentes.
2. Representação dos modelos: Obteve-se os modelos necessários para representação dos objetos através do site *open source* “Free 3D” (<http://free3d.com>). De momento estão implementados seis (6) modelos reconhecíveis pelas câmaras: pessoa, bicicleta, carro, autocarro, camião (representado por um tipo de carro diferente) e barco. Além destes, existem também outros dois (2) - sinal STOP e uma boca de incêndio - de forma a poder mostrar que o DT desenvolvido, permite não só a representação automática baseada em reconhecimento visual, mas também a personalização manual da cidade com elementos que ainda não são detetados automaticamente, como é possível observar na Figura 4.2 da secção 4.3 do Capítulo 4.
3. Desenvolvimento de funções: Implementou-se algumas funções essenciais para que o gémeo virtual represente com precisão o gémeo físico.

### 3.2 VISÃO GERAL DA ARQUITETURA

O sistema desenvolvido apresenta uma grande dependência do NAP-VISION visto que é este serviço que representa a captação dos dados em tempo real. Caso o NAP-VISION apresente alguma instabilidade ou problema que o impossibilite de funcionar, o sistema não tem injeção de dados, o que significa que, apesar de funcional desvirtua o seu propósito uma vez que não lhe é possível espelhar o comportamento do mundo físico. Acrescenta-se ainda, e tal como discutido no Capítulo 2, dos vários modelos YOLO disponíveis o que apresenta melhores resultados é o YOLOv11 e, por esta razão, é o modelo implementado no ATCLL e, consequentemente, utilizado neste trabalho.

Tendo em conta estes aspetos, apresenta-se de seguida a arquitetura do sistema. Divide-se em dois diagramas uma vez que a Figura 3.3 surge no enquadramento dos dois primeiros blocos, (i) e (ii), referidos na secção 3.1 - recolha de dados e comunicação - ao passo que a Figura 3.4 contextualiza-se no terceiro bloco, (iii).



**Figura 3.3:** Pipeline para construção do DT no Unity.

A análise do diagrama da Figura 3.3 permite a compreensão do fluxo dos dados e como o sistema está idealizado. Inicialmente, os dados são recolhidos de várias fontes disponíveis no NAP-VISION e continuamente transmitidos e armazenados no *broker* do MQTT, ambos os serviços foram desenvolvidos pelo NAP e facultados ao presente projeto. De seguida, através de um *script python*, e através do MQTT, subscreve-se os dados que foram publicados no *broker*. Enquanto o *script* está ativo é permitida, através do terminal, a leitura de todas as mensagens que estão a ser publicadas em tempo real no *broker*, sem perda de informação. No mesmo *script* existe um método que procede à filtragem das mensagens para retirar apenas a informação relevante para o DT, no caso, filtra os seguintes parâmetros: **objectID, latitude, longitude, confidence, classification**.

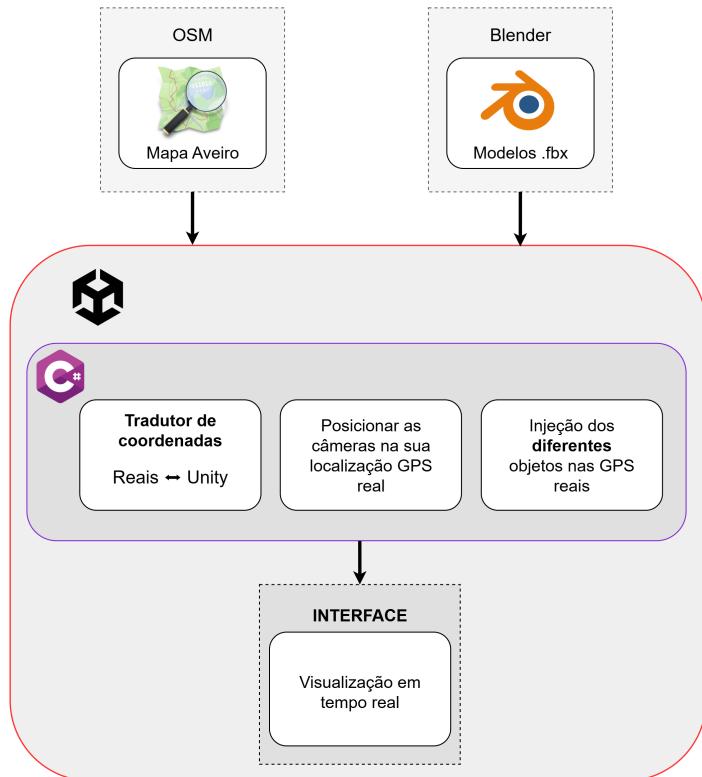
**objectID** : Identificador único de cada objeto detetado.

**latitude e longitude** : Coordenadas GPS reais da localização do objeto detetado no sistema físico.

**confidence** : Métrica de 0 a 1 que indica a confiança do modelo no reconhecimento do objeto.  
**classification** : Distingue e classifica os diferentes objetos.

Ainda no mesmo ambiente, é criada uma *socket* - um tipo de comunicação TCP - que irá receber os dados processados e filtrados do módulo anterior, esta socket funciona como cliente e é responsável pela transmissão de dados para o Unity que por sua vez assume o papel de servidor. É importante referir que para a transmissão ser bem sucedida é necessário paralelizar este processo com os da Figura 3.4, visto que como o Unity representa o servidor da ligação TCP, é necessário estar em *runtime* previamente, para ter permissões de escuta, antes do cliente enviar os dados, como apresentado no esquema da Figura 3.2.

A Figura 3.4 apresenta a sucessão de etapas necessárias para a visualização em tempo real.



**Figura 3.4:** Arquitetura do sistema para representação dos dados em tempo real no DT.

Numa fase inicial, são criadas as condições para a representação no ambiente gráfico do Unity. Através do *Open Street Map* (OSM), [23], uma ferramenta *open source* que permite a qualquer utilizador ter uma representação geográfica do terreno, é importada a zona de interesse de Aveiro, onde existem dados a ser recolhidos. De seguida, foram criados os modelos correspondentes às diferentes classificações de objetos, os quais serão utilizados para representar no ambiente virtual os objetos detetados no mundo físico.

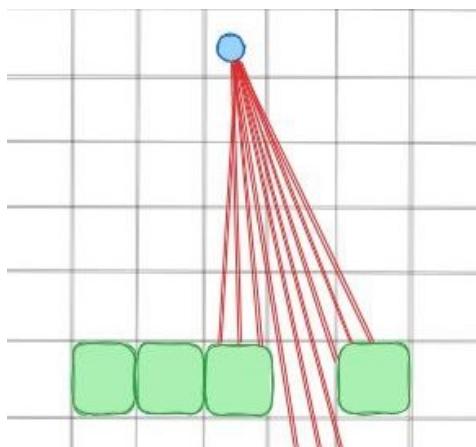
Do ponto de vista da programação, foram desenvolvidos alguns *scripts* em C# com o objetivo de otimizar o DT, nomeadamente:

- **GPSMapper.cs:** Responsável pela conversão de coordenadas entre o gémeo físico e o gémeo virtual, tendo em conta que, conforme descrito na Secção 2.5, o Unity utiliza um sistema de coordenadas próprio. Este componente é essencial para assegurar que qualquer objeto detetado no ATCLL seja corretamente localizado no ambiente virtual.

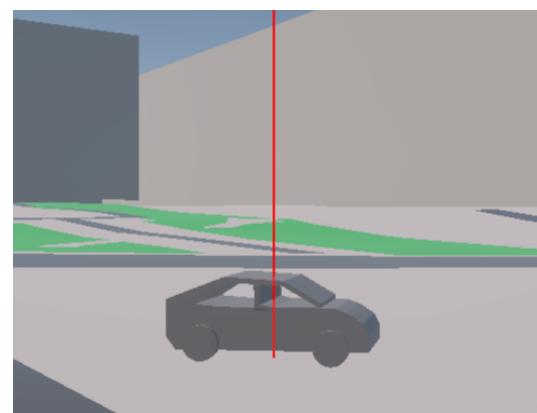
- **ObjectType.cs**: Funciona como um classificador de objetos para os modelos. Lê o valor do parâmetro **classification** do MQTT e associa o modelo que o representa dentro do Unity.
- **CameraPlacer.cs**: Posiciona câmaras virtuais nas exatas coordenadas e com a mesma orientação das várias câmaras físicas existentes no ATCLL.

Para além dos referidos, existe ainda o **SocketReceiver.cs** que é destacado por ser responsável pela comunicação em tempo real entre os sistemas físico e virtual. É nesta componente do sistema que está implementado o servidor da comunicação TCP, quando executado cria uma *socket* que fica à espera que lhe sejam transmitidas informações, ou seja fica à escuta. Quando os dados são recebidos são convertidos para um formato comprehensível pelo Unity. Importa distinguir que o termo “objeto” refere-se ao elemento detetado fisicamente no ambiente real, enquanto “instância” refere-se à representação virtual desse objeto.

Através da utilização da função *GPSMapper.cs*, cada coordenada GPS recebida, é processada e convertida numa coordenada do sistema Unity. Para determinar a componente vertical da instância, foi implementado o algoritmo ***ray-casting***, que de um modo geral consiste na projeção de um raio vertical descendente a partir de um ponto elevado, invisível ao utilizador. O ponto de interseção da instância com a superfície do terreno é considerada como posição final da instância. Desta forma, garante-se que a instância é posicionada corretamente, tanto nas coordenadas GPS como é ajustada à topografia da cidade. A Figura 3.5a apresenta, de maneira simplificada, o princípio do algoritmo *ray-casting*. O ponto azul representa a origem dos raios, este ponto é definido arbitrariamente pelo utilizador, desde que esteja acima da superfície. As linhas vermelhas representam os raios que são lançados na direção do terreno, sendo o ponto de colisão com a superfície, o ponto utilizado como referência para o posicionamento vertical do objeto. Na Figura 3.5b é apresentado um exemplo da aplicação do *ray-casting* no sistema para posicionamento vertical de um modelo de um carro. Neste caso, o raio é representado a vermelho exclusivamente para efeitos de visualização da implementação do algoritmo. Consoante a classificação dos objetos, é instanciado o modelo correspondente, isto é, para cada novo objeto identificado (o parâmetro **objectId** é único) é injetado no sistema uma nova instância, caso já exista e o objeto tenha alterado a sua localização, a posição da instância é atualizada. Ainda neste tópico, foi implementado um mecanismo de *tracking* temporal que regista a última deteção de cada objeto. Quando um objeto não é detetado num intervalo de tempo superior ao tempo de tolerância definido, neste caso definiu-se dois segundos, a instância é removida do ambiente virtual, evitando que os objetos que já saíram de cena não sejam acumulados no DT e, consequentemente, sejam eliminados da visualização virtual.



(a) Exemplo gráfico.



(b) Exemplo prático.

**Figura 3.5:** Em 3.5a apresenta-se a funcionalização do algoritmo *ray-casting* e em 3.5b a implementação deste algoritmo no DT desenvolvido.

### 3.3 ESPECIFICAÇÕES DA MÁQUINA UTILIZADA

Para o desenvolvimento do projeto utilizou-se uma máquina DELL Precision 3591 com um processador Intel Core Ultra 7 165H, com frequência base de 1.40 GHz, capacidade da memória RAM de 32 GB com possibilidade de utilização de 31,5 GB do total. Uma placa gráfica NVIDIA RTX 2000 Ada Generation, com 8 GB de memória. O sistema operativo é o Windows 11 Home, versão 24H2, arquitetura de 64 bits.

Todos os resultados apresentados estão limitados à capacidade da máquina descrita.

# CAPÍTULO 4

## Resultados

O presente capítulo apresenta os resultados obtidos no decurso do trabalho, destacando tanto os pontos positivos como os pontos negativos, acompanhado de uma análise crítica dos mesmos. Os resultados obtidos incluem médias relativamente ao numero de mensagens e métricas relativas à interpretação e atualização dos objetos simulados.

### 4.1 VISÃO POR COMPUTADOR

Inicialmente realizou-se um estudo para a compreensão do algoritmo YOLO e para perceber de que forma é que este permite realizar a deteção de objetos. Para o teste utilizou-se o modelo YOLOv11 aplicado a vídeos *open source* disponíveis no site “Pexels” (<https://www.pexels.com/search/videos>). Os resultados demonstram que para um vídeo de 20 s que apresente resolução Full HD a 30 frames per second (FPS) e cerca de 900 frames, a utilização da CPU resultou num tempo de processamento médio de 100 ms por frame. Comparativamente, o uso da GPU realizou a mesma tarefa em 30 ms, com um tempo médio por frame de 18 ms evidenciando um desempenho significativamente superior. Apesar da vantagem em utilizar a GPU, utilizá-la para múltiplas tarefas em simultâneo, como a identificação de objetos e representação dos mesmos num ambiente virtual irá condicionar o desempenho da mesma. Esta limitação levantou a necessidade de explorar métodos alternativos que permitem a separação das tarefas. Neste contexto, verificou-se que o NAP-VISION utiliza o mesmo modelo, YOLOv11 e apresenta resultados de teste em contexto real no ATCLL, apresentando ainda a vantagem de integrar argumentos relativos às coordenadas GPS dos objetos detetados. Assim, optou-se por utilizar os dados provenientes deste sistema, solucionando a questão da tarefa para a deteção.

Desta forma, tornou-se necessário compreender o fluxo de dados que integra o NAP-VISION, analisar de que forma é que estes são processados e identificar as capacidades e limitações do sistema, esta etapa é essencial para que os próximos passos possam ser planeados, assegurando um alinhamento com a arquitetura já implementada no NAP-VISION.

## 4.2 REPRESENTAÇÃO GEOGRÁFICA DO TERRENO

As coordenadas GPS utilizadas para limitar o terreno foram determinadas por observação direta com base na comparação com a área do ATCLL que apresenta recolha de dados. Obtiveram-se dois pontos de referência, sendo que os outros dois foram obtidos através da interseção de linhas retas ortogonais traçadas a partir dos pontos de referência.

Na Tabela 4.1 apresentam-se as coordenadas GPS selecionadas, bem como a respetiva correspondência no sistema de coordenadas do Unity. A Figura 4.1 mostra uma vista superior do levantamento do território e já com a integração de edifícios em 3D no Unity. Para facilitar a interpretação das coordenadas escolhidas, foram atribuídas cinco cores distintas aos diferentes pontos de referência, o código de cores é também apresentado na Tabela 4.1.

**Tabela 4.1:** Correspondência de coordenadas no sistema Unity às coordenadas reais, com legenda de cores para a visualização da Figura 4.1.

	Localização na Figura 4.1	GPS (lat, long)	Sistema Unity (x,z)
Vermelho	Canto superior esquerdo	(40.623700, -8.661910)	(-4807.29, -6999.17)
Verde	Canto superior direito	-----	(-4805.47, 7070.37)
Azul	Canto inferior esquerdo	-----	(4815.39, -6999.17)
Amarelo	Canto inferior direito	(40.648700, -8.639220)	(4813.57, 7070.37)
Magenta	Centro	-----	(0,0)



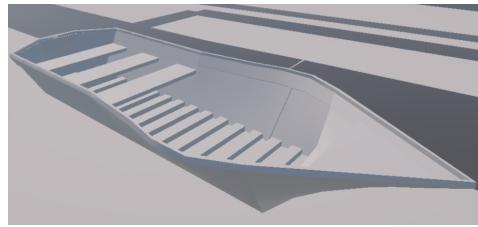
**Figura 4.1:** Representação da cidade

## 4.3 REPRESENTAÇÃO DOS MODELOS

Para instanciar os objetos detetados obtiveram-se modelos representativos, apresentados na Figura 4.2.



(a) Representação dos modelos utilizados e dos dois modelos adicionais.



(b) Representação do modelo do barco.

**Figura 4.2:** Representação dos modelos obtidos e utilizados como instâncias no Unity.

Os modelos dos objetos são instanciados após o processamento dentro do Unity. Isto é, quando o Unity recebe as mensagens filtradas pelo *python*, após a leitura da mensagem, o sistema associa o valor do parâmetro **classification** ao modelo correspondente, previamente definido num dicionário com pares de chave-valor. Desta forma, o modelo é definido inteiramente pelo valor do parâmetro mencionado e é classificado corretamente de acordo com a classificação proveniente do YOLO. Importa salientar que, nesta fase do desenvolvimento, as instâncias dos modelos ainda não integram a orientação dos objetos, que é definida pelo parâmetro **heading**. Este parâmetro representa a direção do movimento de um objeto. Por exemplo, no caso em que um carro se desloque na direção Sul (no sistema de eixos do ambiente virtual corresponde ao sentido negativo do eixo Z), era expectável que a sua instância estivesse orientada na mesma direção. Contudo, tal não se verifica, uma vez que os modelos importados para o Unity mantêm a orientação definida no seu sistema de coordenadas original aquando da sua criação no software de modelação 3D. No exemplo referido, mesmo que o veículo esteja a mover-se para Sul, a frente do modelo permanece orientada para Norte (sentido positivo do eixo Z) pois é esta direção que corresponde ao sentido do eixo das coordenadas de criação. Apesar do reconhecimento desta limitação, optou-se por privilegiar a funcionalidade do DT, comparativamente à correção visual da orientação dos modelos.

#### 4.4 ANÁLISE DA VISUALIZAÇÃO NO SISTEMA FÍSICO VS. DIGITAL

O Unity permite colocar câmaras no ambiente virtual nas exatas coordenadas e com a mesma orientação das câmaras físicas, proporcionando uma visualização estática semelhante à que se podem observar no ambiente real da câmara. Após a colocação e calibração das câmaras virtuais nos postes disponíveis até à data de desenvolvimento, nomeadamente P6-1, P4, P22, P25, P30, P35 e P50-1, é possível alterar entre as mesmas através da tecla “C” do teclado, apenas quando o sistema está em execução. Este *software* permite ainda uma navegação dinâmica no DT de forma independente das câmaras estáticas, ou seja, é possível visualizar o ambiente virtual a partir de perspetivas que não correspondem às câmaras físicas. A navegação dinâmica é realizada com recurso a dispositivos de entrada/saída (E/S, ou mais comum no inglês I/O), nomeadamente o rato e o teclado, permitindo operações como arraste, rotação, aproximação, diminuição do ponto de vista, entre outras. Assim, apesar da informação ser a mesma, através de uma visão diferente, mais flexível e abrangente, poderá potenciar novas formas de análise e, consequentemente, uma adoção de novas abordagens. A

Figura 4.3 ilustra a informação captada no sistema físico e a sua representação no DT, no P35. Mais concretamente, a Figura 4.3b foi retirada do NAP-VISION e é possível verificar a implementação do YOLO através da existência de caixas que limitam os objetos quando estes são identificados, na figura mencionada observa-se que os objetos identificados como carros apresentam um retângulo branco à sua volta. Na figura 4.3b é exibida a vista virtual da mesma câmara do P35, a visualização apresentada foi obtida através da navegação dinâmica para observar com mais detalhe o posicionamento dos objetos identificados.



(a) P35-1 câmara física.



(b) P35-1 representado no DT.

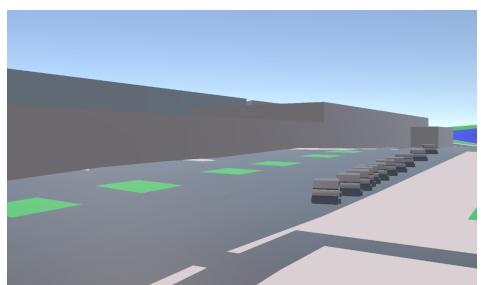
**Figura 4.3:** DT do P35-1.

Através da análise da Figura 4.3, verifica-se que as coordenadas dos objetos comparativamente às coordenadas das instâncias não correspondem de forma precisa, apesar de estarem representados no mesmo espaço apresentam uma margem de erro. Supõe-se que a origem deste erro poderá estar associada às coordenadas identificadas e enviadas pelo NAP-VISION, uma vez que, quando o *script* que contém o tradutor de coordenadas foi desenvolvido, o GPS-Mapper.cs, utilizaram-se pontos de referência para testar a sua precisão, os pontos referem-se a cantos de edifícios, centros de rotundas e interseção de estradas. As coordenadas GPS destes correspondiam com as coordenadas do sistema Unity. No entanto, apesar da margem de erro, é possível identificar um padrão na Figura 4.3a que se assemelha bastante com o que se observa na Figura 4.3b. Nota-se que a disposição dos objetos da esquerda para a direita em ambas a figuras é semelhante, e nota-se também que o sistema faz a distinção entre a proximidade dos objetos em relação à câmara.

Para comparação apresenta-se na Figura 4.4 a imagem capturada pela câmara física no P22 (Figura 4.4a) e a sua representação no DT (Figura 4.4b).



(a) P22 câmara física.



(b) P22 representado no DT.

**Figura 4.4:** DT do P22.

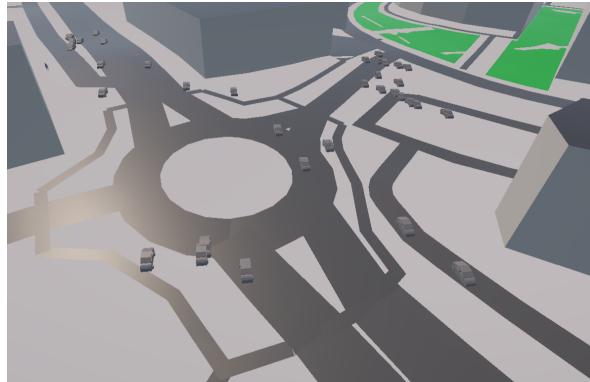
Na Figura 4.4 pode observar-se que as instâncias apresentam localizações mais próximas às coordenadas GPS reais, fortalecendo a hipótese da calibração das coordenadas do P35 não estar correta.

À data do desenvolvimento do projeto, apenas o poste P35, apresentava quatro câmaras disponíveis na mesma localização mas com orientações diferentes, esta configuração permite aos sistemas de VC uma percepção completa do local. Para o NAP-VISION e para o sistema desenvolvido as quatro câmaras distinguem-se por P35-1, P35-2, P35-3, P35-4. Sendo que cada uma destas apresenta perspetivas diferentes do ambiente captado, ilustrado na Figura 4.5.



**Figura 4.5:** Objetos detetados pelo NAP-VISION do P35. Quatro câmaras na mesma localização que apresentam orientações diferentes.

No contexto deste projeto, a integração deste tipo de câmaras é muito importante visto que permite obter mais informações e com mais detalhe do mesmo ambiente, mas alargado à sua periferia. Na Figura 4.6 é possível observar toda a informação que as quatro câmaras do P35 captam e a forma como complementam entre si a informação recolhida. Enquanto que no NAP-VISION a análise teria de ser dividida por quatro frames diferentes, o DT permite a condensação da informação toda num local e representa-a com total abstração dos objetos identificados. Desta forma, é possível obter uma visão contínua e total, replicando o sistema físico de forma mais fiel.

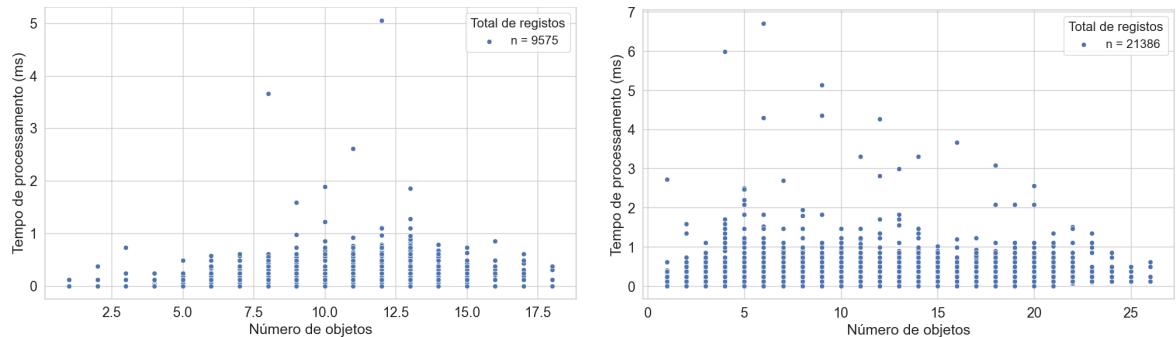


**Figura 4.6:** DT do P35 com integração das quatro câmaras.

#### 4.5 REGISTO DE TEMPOS

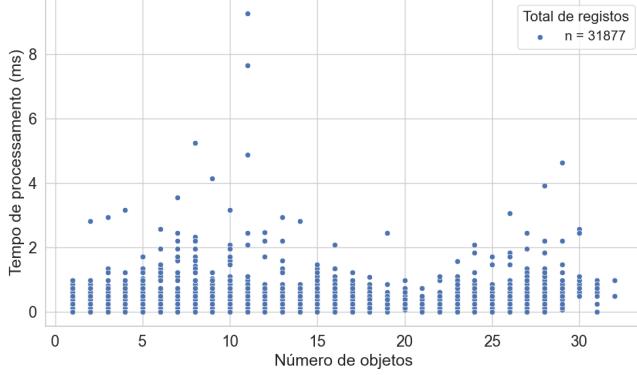
Para uma análise ao desempenho do sistema em tempo real, procedeu-se ao registo do tempo de processamento desde que as mensagens são passadas ao Unity até à instanciação dos objetos detetados. É fundamental sublinhar que o objetivo do trabalho desenvolvido, e a análise posterior, não incidem sobre a comunicação entre processos, ou seja, não se pretende avaliar métricas de latência entre dispositivos, ou relativas aos protocolos de comunicação, nem sobre tempos de envio/recepção associados às infraestruturas da rede. O foco centra-se na avaliação do comportamento e desempenho dos componentes que fazem parte da simulação, particularmente a forma como o Unity interpreta as mensagens recebidas e instancia os modelos correspondentes.

Assim, procedeu-se ao registo de três cenários diferentes no P35: apenas para uma câmara, para as quatro câmaras localizadas num poste, e para todas as câmaras implementadas no sistema. Para os três registos, o sistema esteve em execução  $\sim 30$  min.



- (a) Tempo de processamento no Unity em função do número de objetos para uma única câmara (P35-1). (b) Tempo de processamento no Unity em função do número de objetos para o P35 com cobertura visual total.

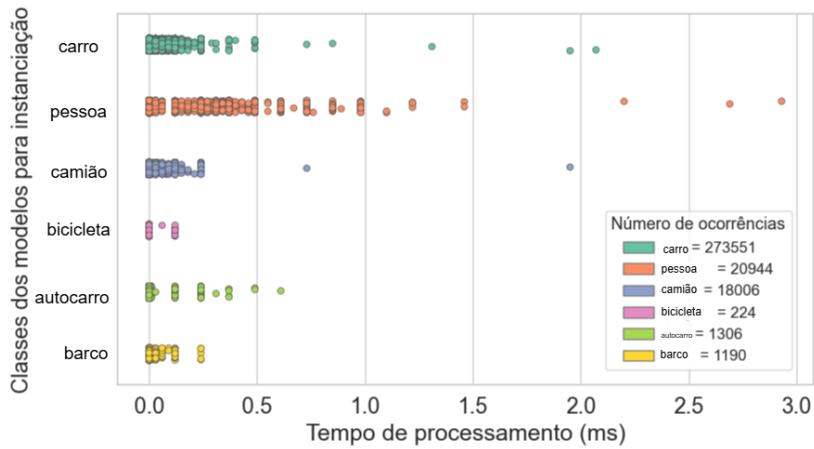
**Figura 4.7:** Comparação dos tempos de processamento no Unity para diferentes configurações de câmara no sistema P35.



**Figura 4.8:** Tempo de processamento no Unity em função do número de objetos para todas as câmaras implementadas no sistema.

A análise do gráfico da Figura 4.7a demonstra que o tempo de processamento é, de uma forma geral, inferior a 1 ms, registando alguns *outliers* com um máximo de ~5 ms. Observa-se também que o tempo médio mantém-se baixo, 0,123 ms, mesmo com o aumento do número de objetos. O gráfico da Figura 4.7b, correspondente à configuração das quatro câmaras no mesmo poste, revela um ligeiro aumento do tempo de processamento, ainda assim, mantém a média abaixo dos 2 ms, ~0,187 ms. Neste caso, registam-se alguns *outliers* até ~7 ms. O gráfico da Figura 4.8, que configura sistema com a implementação de todas as câmaras, regista um tempo médio de 0,180 ms, ligeiramente inferior ao gráfico anterior. No entanto, observa-se maior dispersão com alguns *outliers* a chegarem perto dos 9 ms. A diferença dos tempos médios dos gráficos das Figuras 4.7b e 4.8 poderá estar relacionada ao facto de, apesar do gráfico 4.8 apresentar um maior número de registos, é possível que uma parte desses correspondam a situações com poucos objetos por câmara individual, reduzindo o esforço computacional necessário e, consequentemente, o tempo de processamento.

O gráfico da Figura 4.9 representa o tempo de processamento para cada modelo, por instância individual. Esta análise permite identificar quais os modelos com maior complexidade e, por conseguinte, os que provocam maior esforço computacional.



**Figura 4.9:** Tempo de processamento no Unity para instanciar cada tipo de modelo.

Após análise da Figura 4.9 conclui-se que o modelo associado à classe “pessoa” apresenta

uma variação temporal muito maior comparativamente aos restantes modelos, permitindo afirmar que este modelo é o mais complexo de todos, prejudicando a eficiência do sistema em contexto de visualização em tempo real. Os tempos registados pelos outros modelos, de uma forma geral, apresentam todos tempos inferiores a 0,5 ms, refletindo uma resposta mais eficiente. Instanciar muitos objetos ao mesmo tempo supõe uma grande carga gráfica, o que naturalmente utiliza maior esforço computacional e supõe tempos de processamento superiores. No entanto, o sistema mostrou uma capacidade de processamento relativamente boa uma vez que, desde que a mensagem chega ao Unity até à sua representação o sistema apresenta, no pior dos casos, um tempo médio de 0,187 ms.

#### 4.6 CONSIDERAÇÕES FINAIS

Complementarmente à análise realizada, poderiam ter sido retiradas outras métricas que permitissem melhorar a avaliação do sistema. Mais exatamente, a monitorização da FPS durante a execução permitiria avaliar a fluidez do sistema, uma vez que este valor traduz a quantidade de imagens processadas por segundo. Paralelamente, o registo dos valores dos recursos computacionais consumidos, como a CPU, GPU e RAM, em diferentes cenários, como por exemplo com diferente número de instâncias, permitiria avaliar uma relação entre a complexidade do sistema com o esforço computacional necessário.

Em relação aos métodos desenvolvidos no Unity, o registo do tempo médio de execução do algoritmo de *ray-casting* poderia ter sido relevante, uma vez que este também afeta o desempenho do sistema em tempo real. Adicionalmente, o registo da taxa de atualização de *frames* por câmara constituiria uma análise relevante, dado que, diferentes câmaras podem gerar volumes distintos de informação. Assim, medir a quantidade de mensagens enviadas por segundo por cada câmara permitiria identificar quais as zonas com maior produção de dados, e portanto, quais a que provocam maior esforço computacional. Importa ainda referir que com vista à otimização e implementação de novas soluções, o sistema foi desenvolvido com uma arquitetura modular, facilitando a implementação de métricas adicionais ou a introdução de melhorias futuras.

## Conclusão

Este projeto teve como principal objetivo o desenvolvimento de um DT. Neste trabalho foram aprofundados conceitos de modelação em ambiente 3D, através do software Unity, e de transformação de dados de VC para o mesmo ambiente. O trabalho desenvolvido envolve a utilização de mensagens de sensores, tradução dessas mensagens para compatibilidade com o Unity e interpretação das mesmas pelo ambiente simulado, para representação visual dos objetos detetados.

Desta forma, e alinhado com o sistema proposto na secção 3.2 do Capítulo 3, o trabalho realizado integra protocolos de comunicação MQTT e TCP, permitindo que os dados captados no sistema físico sejam transmitidos para um ambiente de processamento de dados, optou-se pelo *python*, onde são filtrados de forma a tornarem-se comprehensíveis para o sistema de visualização, o Unity. Já no software Unity, foram desenvolvidos vários métodos com vista a assegurar uma representação fiável e coerente do ambiente físico, nomeadamente, realizou-se o mapeamento da cidade através do OSM, e obtiveram-se os modelos representativos dos objetos com recurso a ferramentas *open source* como o site “Free 3D”. Com uma perspetiva de otimizar a funcionalidade do DT desenvolveram-se *scripts* que permitem a tradução de coordenadas entre o sistema físico e o sistema virtual permitindo o correto posicionamento das instâncias, a classificação correta dos modelos associados às instâncias, o posicionamento das câmaras virtuais nas mesmas coordenadas do sistema físico. Por fim, o *script* que permite a injeção dos modelos em cena nas coordenadas recebidas do MQTT e ajusta o posicionamento vertical através do algoritmo *ray-casting*.

Os resultados obtidos evidenciam que o modelo virtual apresenta uma elevada semelhança com o sistema físico, embora exista uma margem de erro associadas à tradução das coordenadas GPS para coordenadas do sistema Unity. Verificou-se um correto funcionamento do DT para todas as câmaras virtuais implementadas, salientando o bom desempenho do sistema quando a representação virtual é ativada para mais do que uma câmara. Foram registados os tempos de processamento das instâncias, tendo-se observado um melhor desempenho no cenário com menos objetos instanciados, com um tempo de processamento médio de 0,123 ms para todos

os objetos identificáveis. Confirmou-se ainda que os modelos são instanciados corretamente consoante a sua classificação, tendo sido analisados também os tempos de processamento por classe. O modelo associado à classe “pessoa” apresentou a maior variação temporal, o que indica ser o modelo de maior complexidade e, portanto, com menor eficiência computacional.

Este trabalho constitui uma base fundamental para obter uma representação virtual da cidade em estudo, fornecendo os pilares para a partilha de mensagens que transportam a informação captada pelos sistemas físicos. Como perspetivas futuras, destaca-se a implementação do parâmetro ***heading*** nas instâncias dos objetos, uma vez que é um elemento crucial para a correta orientação espacial e para uma representação mais realista e informativa do sistema urbano. Considera-se também que deverá ser incluído um cálculo para a margem de erro associada à conversão de coordenadas, para que seja possível quantificar e corrigir desvios entre o sistema físico e virtual. Para melhorar o desempenho do sistema particularmente ao nível do processamento e renderização em tempo real, propõe-se a integração de *multithreading*.

# Referências

- [1] G. Caprari, G. Castelli, M. Montuori, M. Camardelli e R. Malvezzi, «Digital twin for urban planning in the green deal era: A state of the art and future perspectives,» *Sustainability*, vol. 14, n.º 10, p. 6263, 2022.
- [2] M. Alvi, H. Dutta, R. Minerva, N. Crespi, S. M. Raza e M. Herath, «Global perspectives on digital twin smart cities: Innovations, challenges, and pathways to a sustainable urban future,» *Sustainable Cities and Society*, vol. 126, p. 106356, 2025, ISSN: 2210-6707. DOI: <https://doi.org/10.1016/j.scs.2025.106356>. URL: <https://www.sciencedirect.com/science/article/pii/S221067072500232X>.
- [3] P. O. Hristov, D. Petrova-Antanova, S. Ilieva e R. Rizov, «ENABLING CITY DIGITAL TWINS THROUGH URBAN LIVING LABS,» *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLIII-B1-2022, pp. 151–156, 2022. DOI: 10.5194/isprs-archives-XLIII-B1-2022-151-2022. URL: <https://isprs-archives.copernicus.org/articles/XLIII-B1-2022/151/2022/>.
- [4] S. Moroni, «Insurmountable limitations of city-scale digital twins? On urban knowledge and planning,» *Computational Urban Science*, vol. 5, n.º 1, p. 17, 2025.
- [5] S. Mazzetto, «A Review of Urban Digital Twins Integration, Challenges, and Future Directions in Smart City Development,» *Sustainability*, vol. 16, n.º 19, 2024, ISSN: 2071-1050. DOI: 10.3390/su16198337. URL: <https://www.mdpi.com/2071-1050/16/19/8337>.
- [6] G. J. Rosa, J. M. Afonso, P. D. Gaspar, V. N. Soares e J. M. Caldeira, «Detecting wear and tear in pedestrian crossings using computer vision techniques: approaches, challenges, and opportunities,» *Information*, vol. 15, n.º 3, p. 169, 2024.
- [7] M. Mendes, «Computer vision for traffic and object detection,» Dissertação de Mestrado, University of Aveiro, Aveiro, Portugal, 2023. URL: <http://hdl.handle.net/10773/41041>.
- [8] M. Mendes, G. Perna, P. Rito, D. Raposo e S. Sargento, «Real-time Object and Event Detection Service through Computer Vision and Edge Computing,» *arXiv preprint arXiv:2504.11662*, 2025.
- [9] A. Das, M. Degeling, X. Wang, J. Wang, N. Sadeh e M. Satyanarayanan, «Assisting Users in a World Full of Cameras: A Privacy-Aware Infrastructure for Computer Vision Applications,» jul. de 2017. DOI: 10.1109/CVPRW.2017.181.
- [10] P. Rito, A. Almeida, A. Figueiredo et al., «Aveiro Tech City Living Lab: A Communication, Sensing, and Computing Platform for City Environments,» *IEEE Internet of Things Journal*, vol. 10, n.º 15, pp. 13 489–13 510, 2023. DOI: 10.1109/JIOT.2023.3262627.
- [11] N. Architectures e P. ( de Telecomunicações – University of Aveiro). «Aveiro Tech City Living Lab.» Acedido em: 24-06-2025. (2025), URL: <https://new.aveiro-living-lab.it.pt/infrastructure/>.
- [12] Z. Sun, R. Zhang e X. Zhu, «The progress and trend of digital twin research over the last 20 years: A bibliometrics-based visualization analysis,» *Journal of Manufacturing Systems*, vol. 74, pp. 1–15, 2024, ISSN: 0278-6125. DOI: <https://doi.org/10.1016/j.jmsy.2024.02.016>. URL: <https://www.sciencedirect.com/science/article/pii/S0278612524000384>.
- [13] M. Jafari, A. Kavousi-Fard, T. Chen e M. Karimi, «A Review on Digital Twin Technology in Smart Grid, Transportation System and Smart City: Challenges and Future,» *IEEE Access*, vol. PP, pp. 1–1, jan. de 2023. DOI: 10.1109/ACCESS.2023.3241588.

- [14] D. Yang, H. R. Karimi, O. Kaynak e S. Yin, «Developments of digital twin technologies in industrial, smart city and healthcare sectors: A survey,» *Complex Engineering Systems*, vol. 1, n.º 1, N-A, 2021.
- [15] N. Kshetri, «Chapter 3 - Amplifying the value of blockchain in supply chains: combining with other technologies,» em *Blockchain and Supply Chain Management*, N. Kshetri, ed., Elsevier, 2021, pp. 67–88, ISBN: 978-0-323-89934-5. DOI: <https://doi.org/10.1016/B978-0-323-89934-5.00003-9>. URL: <https://www.sciencedirect.com/science/article/pii/B9780323899345000039>.
- [16] C.-Y. Wang, A. Bochkovskiy e H.-Y. M. Liao, «YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors,» em *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 7464–7475. DOI: [10.1109/CVPR52729.2023.00721](https://doi.org/10.1109/CVPR52729.2023.00721).
- [17] L. He, Y. Zhou, L. Liu e J. Ma, «Research and Application of YOLOv11-Based Object Segmentation in Intelligent Recognition at Construction Sites,» *Buildings*, vol. 14, n.º 12, 2024, ISSN: 2075-5309. DOI: [10.3390/buildings14123777](https://doi.org/10.3390/buildings14123777). URL: <https://www.mdpi.com/2075-5309/14/12/3777>.
- [18] T.-Y. Lin, M. Maire, S. Belongie et al., «Microsoft coco: Common objects in context,» em *Computer vision-ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, Springer, 2014, pp. 740–755.
- [19] G. Jocher, J. Qiu e A. Chaurasia, *Ultralytics YOLO*, versão 8.0.0, jan. de 2024. URL: <https://github.com/ultralytics/ultralytics>.
- [20] M. Rostami, P. Pradhan, N. Karki et al., «A comprehensive review of extended reality and its application in aerospace engineering,» *Progress in Aerospace Sciences*, p. 101118, 2025, ISSN: 0376-0421. DOI: <https://doi.org/10.1016/j.paerosci.2025.101118>. URL: <https://www.sciencedirect.com/science/article/pii/S0376042125000442>.
- [21] M. A. Khan, M. A. Khan, S. U. Jan et al., «A Deep Learning-Based Intrusion Detection System for MQTT Enabled IoT,» *Sensors*, vol. 21, n.º 21, 2021, ISSN: 1424-8220. DOI: [10.3390/s21217016](https://doi.org/10.3390/s21217016). URL: <https://www.mdpi.com/1424-8220/21/21/7016>.
- [22] M. R. Kabir, B. B. Yedla Ravi e S. Ray, «A Virtual Prototyping Platform for Exploration of Vehicular Electronics,» *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, set. de 2023. DOI: [10.1109/JIOT.2023.3267339](https://doi.org/10.1109/JIOT.2023.3267339).
- [23] OpenStreetMap contributors, *Planet dump retrieved from https://planet.osm.org, https://www.openstreetmap.org*, 2017.