Universidade da Beira Interior
Departamento de Informática

Neural Radiance Fields Meet Object Recognition: Is it a Match?

Elaborado por:

Julieta Torres Neves Sequeira

Orientador:

Professor Doutor Hugo Proença

29 de junho de 2025

Agradecimentos

O desenvolvimento e a conclusão deste trabalho não seriam possíveis sem a orientação do Professor Doutor Hugo Proença, que me ajudou a avançar e a encontrar soluções em momentos mais desafiantes.

Agradeço também aos meus pais, família e amigos, pelo apoio constante ao longo desta jornada — sem eles, tudo teria sido consideravelmente mais difícil.

Por fim, deixo um agradecimento especial aos meus colegas de curso, que estiveram sempre prontos a ajudar e a partilhar este percurso académico comigo.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

UBI	Universidade da Beira Interior
NeRF	Neural Radiance Fields
SAM2	Segment Anything Model v2
MLP	Multilayer Perceptron
SfM	Structure-from-Motion
MVS	Multi-View Stereo
SAM	Segment Anything Model
WSL	Windows Subsystem for Linux
SSD	Soma das Diferenças Quadráticas
NSSD	Soma Normalizada das Diferenças Quadráticas
CMC	*Cumulative Match Characteristic*

Capítulo

1

Introdução

1.1 Enquadramento

A visão computacional tem registado avanços significativos nos últimos anos, permitindo o desenvolvimento de técnicas cada vez mais sofisticadas para a reconstrução e interpretação de cenas tridimensionais. Neste contexto, os Neural Radiance Fields (NeRF) surgem como uma abordagem inovadora para a geração de representações 3D a partir de imagens 2D.

Com recurso a modelos de aprendizagem profunda, os NeRF são capazes de modelar interações complexas da luz com superfícies, ao captar detalhes finos e variações de iluminação. Esta capacidade tem vindo a transformar áreas como a realidade aumentada, a robótica e a visualização tridimensional.

O presente trabalho insere-se no âmbito da unidade curricular de Projeto de Licenciatura em Engenheria Informática Universidade da Beira Interior (UBI).

1.2 Motivação

Apesar dos avanços alcançados com os modelos NeRF, levanta-se uma questão crítica: será possível distinguir imagens reais de imagens geradas por estes modelos? Esta problemática é particularmente relevante em domínios como a forense digital ou a deteção de conteúdos sintéticos, onde a autenticidade visual é fundamental.

Motivado por este desafio, o presente projeto propõe-se investigar a possibilidade de detetar, com recurso a métodos computacionais, se uma imagem real apresenta características visuais compatíveis com as de imagens sintetizadas por modelos NeRF.

1.3 Objetivos

Este trabalho tem como objetivo principal investigar a possibilidade de replicar imagens reais através de modelos NeRF, contribuindo para uma melhor compreensão das capacidades e limitações desta tecnologia.

Para tal, definiram-se os seguintes objetivos específicos:

- Construir uma base de dados de modelos gerados por um método NeRF;
- Implementar um sistema de comparação de imagens, com recurso a métricas de avaliação que permitam estimar a compatibilidade entre imagens reais e imagens produzidas por NeRF;
- Validar a abordagem proposta utilizando conjuntos de dados compostos por imagens reais dos mesmos objetos usados na geração dos modelos.

1.4 Organização do Documento

1. O **Capítulo 1 – Introdução** apresenta o enquadramento do projeto, a motivação que impulsionou o seu desenvolvimento, os objetivos, bem como a estrutura do próprio documento.
2. O **Capítulo 2 – Estado da Arte** analisa os principais conceitos e tecnologias que fundamentam o trabalho.
3. O **Capítulo 3 – Tecnologias e Ferramentas Utilizadas** descreve o ambiente de desenvolvimento adotado, incluindo hardware, software, linguagens de programação, bibliotecas, frameworks e ferramentas relevantes para a implementação do sistema.
4. O **Capítulo 4 – Implementação** detalha a metodologia aplicada na construção da base de dados, segmentação das imagens, conversão e preparação dos dados, bem como o processo de comparação entre imagens reais e sintetizadas.
5. O **Capítulo 5 – Experiências e Resultados** apresenta e analisa os resultados obtidos durante os testes, incluindo métricas de avaliação, matriz de confusão, e uma análise pormenorizada por objeto.
6. **Capítulo 6 – Conclusões e Trabalho Futuro** resume as principais conclusões retiradas ao longo do projeto e são propostas abordagens para trabalhos futuros que possam complementar e melhorar os resultados alcançados.

Capítulo

2

Estado da Arte

2.1 Introdução

Este capítulo tem como objetivo apresentar os principais conceitos e tecnologias que fundamentam o desenvolvimento do presente trabalho, com particular foco na deteção de compatibilidade entre imagens reais e imagens renderizadas por modelos NeRF.

A área da visão computacional tem registado avanços significativos nos últimos anos, especialmente no campo da criação de imagens tridimensionais. Neste contexto, os NeRF destacam-se por permitirem a representação e renderização de cenas 3D a partir de um conjunto limitado de imagens 2D, com elevado realismo e eficiência [6].

De forma a contextualizar o trabalho desenvolvido, este capítulo encontra-se organizado em três secções principais. Numa primeira fase, é apresentada uma descrição detalhada dos NeRF e das suas principais variações. De seguida, são abordados os métodos utilizados na deteção de imagens sintéticas, com especial destaque para a técnica de *template matching*. Por fim, são exploradas as técnicas de segmentação de imagem que se revelaram fundamentais para a metodologia adotada neste projeto.

2.2 Neural Radiance Fields

2.2.1 Conceito Fundamental

Os NeRF constituem uma abordagem recente e promissora para a representação de cenas tridimensionais, permitindo produzir novas perspetivas de uma cena a partir de um conjunto limitado de imagens 2D. Em vez de recorrer a

representações discretas, como malhas 3D, os NeRF modelam a cena como uma função contínua em cinco dimensões: três correspondentes à posição espacial (x, y, z) e duas à direção de observação (θ, ϕ).

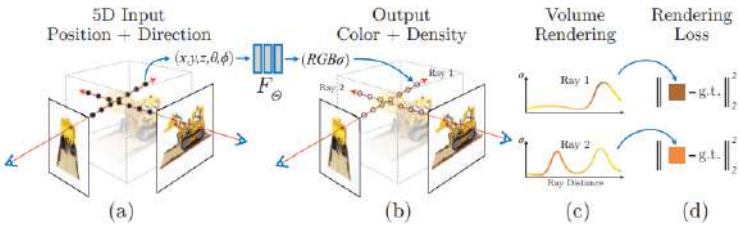

Figura 2.1: Visão geral do funcionamento dos NeRF. A imagem mostra o processo de renderização diferenciável: (a) amostragem de coordenadas 5D (posição e direção) ao longo de raios da câmara; (b) uso de uma rede MLP para prever cor e densidade volumétrica; (c) composição das amostras usando técnicas de *volume rendering*; (d) cálculo da perda entre a imagem renderizada e a observada para otimização do modelo. Trecho Retirado de Mildenhall et al. (2020) [6].

Esta função é aprendida através de uma rede neuronal do tipo MLP que, para cada ponto no espaço e direção de visualização, prevê a cor RGB e a densidade volumétrica. Com esta informação, é possível simular a forma como a luz interage com os objetos da cena, permitindo a produção de imagens altamente realistas sob diferentes pontos de vista.

A principal vantagem dos NeRF reside na capacidade de estes representarem detalhes visuais complexos e efeitos como oclusões ou reflexos, que são de difícil modelação por métodos tradicionais.

2.2.2 Funcionamento dos NeRF

O processo de renderização nos NeRF baseia-se em técnicas clássicas de *volume rendering*. Para cada píxel de uma imagem, é lançado um raio a partir da câmara, sendo amostrados múltiplos pontos ao longo da sua trajetória. A MLP calcula, para cada ponto, a cor e a densidade volumétrica. Estes valores são posteriormente integrados ao longo do raio, de forma diferenciável, combinando as contribuições de cada amostra para obter a cor final do píxel. Como ilustrado na Figura 2.1, este processo envolve a amostragem de coordenadas espaciais e direcionais, a previsão da cor e da densidade por parte da rede neuronal, e a composição final dos dados. O modelo é então otimizado com base na diferença entre a imagem renderizada e a imagem observada.

Durante a fase de treino, a rede é ajustada com o objetivo de minimizar o erro entre as imagens reais e as imagens geradas, recorrendo, geralmente, à métrica do erro quadrático médio (MSE).

2.2.3 Ferramentas e Métodos NeRF

No âmbito deste trabalho, foram exploradas diversas implementações e variantes de NeRF, com destaque para ***Nerfstudio*** [8] — uma framework modular, *open-source* e de utilização intuitiva, que permite experimentar diferentes arquiteturas de NeRF, com suporte para calibração automática, variados *datasets* e visualização em tempo real.

Adicionalmente, a *Nerfstudio* integra ferramentas que facilitam a criação de *datasets* personalizados, que permitem organizar os dados e realizar automaticamente a calibração necessária ao treino dos modelos. Esta funcionalidade revelou-se particularmente útil, possibilitando o desenvolvimento de conjuntos de dados adequados às necessidades específicas do projeto.

Apesar da elevada qualidade visual alcançada pelos modelos NeRF, a sua aplicabilidade em contextos de tempo real continua limitada, exceto quando utilizadas variantes otimizadas como o *Splatfacto*.

Splatfacto [4]

A variante escolhida neste projeto foi o *Splatfacto*, uma técnica recente baseada em *Gaussian Splatting*. Em vez de representar a cena com uma rede neuronal volumétrica, o *Splatfacto* utiliza pontos 3D com gaussianas elípticas (*soft 3D spheres*), cada uma definida por posição, cor, orientação e opacidade. O processo de renderização destes pontos é altamente eficiente e paralelizável, permitindo visualizações em tempo real e tempos de treino significativamente reduzidos — muitas vezes de apenas alguns minutos, em contraste com as horas ou dias requeridos por métodos NeRF tradicionais.

Adicionalmente, o *Splatfacto* conjuga os pontos fortes dos modelos NeRF (como a utilização de técnicas de otimização e *backpropagation*) com uma representação mais leve e rápida.

Outras Implementações Relevantes:

- **NeRF-PyTorch**: uma das implementações mais populares, escrita em PyTorch. Apresenta boas otimizações e com suporte para GPU [7];

- **Mip-NeRF**: inspirada em *mipmaps*, melhora a representação multiescala e reduz artefactos de *aliasing*. É cerca de 7% mais rápida e consome metade da memória do NeRF original [1];
- **NeRF (original, 2020)**: implementação de referência, em TensorFlow. Requer aproximadamente 200 000 iterações (cerca de 15 horas de treino), sendo uma das mais exigentes em termos computacionais [6];
- **PyNeRF**: biblioteca em Python com interfaces simplificadas e suporte para vários *datasets* comuns [11].

2.2.3.1 COLMAP: Reconstrução 3D e Calibração de Câmaras

Uma das ferramentas mais utilizadas no contexto da reconstrução tridimensional a partir de imagens é o *COLMAP* [10]. Trata-se de um processo de reconstrução baseada em Structure-from-Motion (SfM) (Structure-from-Motion) e Multi-View Stereo (MVS) (Multi-View Stereo), que permite estimar as poses das câmaras e gerar uma nuvem de pontos densa da cena. No contexto dos NeRF, o *COLMAP* é frequentemente utilizado para realizar a calibração das imagens, etapa essencial para garantir a precisão da reconstrução.

Neste trabalho, o *COLMAP* foi utilizado através da integração disponibilizada pela *framework Nerfstudio*, que recorre a esta ferramenta para efetuar automaticamente a calibração das câmaras e permitir a criação de *datasets* personalizados com reconstrução estrutural da cena.

2.3 Template Matching para Deteção de Imagens Sintéticas

O *template matching* [3] é uma técnica bastante utilizada na visão computacional para encontrar padrões específicos dentro de uma imagem. A ideia consiste em comparar um pequeno recorte da imagem original — o chamado *template* — com diferentes regiões da imagem maior, de forma a identificar onde esse padrão ocorre ou onde existe maior semelhança. Esta técnica é útil em vários contextos, como deteção de objetos ou reconhecimento de formas. No âmbito da deteção de imagens sintéticas, o *template matching* permite analisar se determinadas regiões de uma imagem real se encontram representadas, de forma semelhante, numa imagem renderizada. Isto ajuda a perceber até que ponto a imagem produzida digitalmente consegue reproduzir com precisão os detalhes estruturais e visuais do mundo real. Através desta comparação, é possível identificar diferenças subtis que indicam origem artificial, como artefactos, irregularidades ou ausência de continuidade em zo-

nas que deveriam ser visivelmente semelhantes. Ao aplicar esta abordagem, torna-se possível avaliar o grau de semelhança entre regiões específicas de imagens reais e as suas correspondentes em imagens geradas por modelos NeRF.

2.3.1 Abordagens de Template Matching

Esta técnica contém duas abordagens principais:

- ***Area-Based***: realizam comparações diretas entre valores de intensidade dos pixeis.
- ***Feature-Based***: recorrem a descritores como pontos de interesse, contornos ou estruturas geométricas para guiar o processo de correspondência.

2.3.2 Métricas de Correspondência

As métricas de correspondência são fundamentais nesta técnica, pois estas permitem quantificar a semelhança entre o *template* e as diferentes regiões da imagem. A utilizada para este trabalho foi a versão normalizada da :

- **Soma das Diferenças Quadráticas (SSD)**: esta métrica é usada para quantificar a dissimilaridade entre um template e uma região de uma imagem. A fórmula da SSD é dada por:

$$\mathrm{SSD}(u,v) = \sum_{x,y} [f(x,y) - t(x-u, y-v)]^2 \tag{2.1}$$

  Onde:

  - $f(x,y)$: representa a intensidade do pixel na imagem alvo na coordenada (x,y).
  - $t(x-u, y-v)$: representa a intensidade do pixel na imagem de modelo deslocada por (u,v).

A versão utilizada para a implementação deste trabalho foi a Soma Normalizada das Diferenças Quadráticas (NSSD). Embora a SSD seja uma métrica eficaz para medir a dissimilaridade de intensidade, a sua versão normalizada, a NSSD, oferece vantagens significativas, especialmente em cenários com variações de brilho e contraste. A NSSD é menos sensível a estas variações, uma vez que normaliza a medida de diferença, tornando-a mais robusta. A NSSD é calculada pela seguinte fórmula, conforme apresentado no

artigo [2]:

$$NSSD(x,y) = \frac{\sum_{u,v}(T(u,v) - I(x+u, y+v))^2}{\sum_{u,v} I(x+u, y+v)^2} \tag{2.2}$$

Nesta fórmula, $T(u,v)$ representa a intensidade do pixel no *template* na coordenada (u,v), e $I(x+u, y+v)$ representa a intensidade do pixel na imagem de origem na coordenada $(x+u, y+v)$.

2.4 Técnicas de Segmentação de imagens

A segmentação de imagens é uma área fundamental da visão computacional, cujo objetivo principal é dividir uma imagem em diferentes regiões com significado, geralmente correspondentes a objetos ou partes de objetos. Esta técnica é amplamente utilizada em aplicações como reconhecimento do ambiente, análise médica, condução autónoma, entre outras.

Nos últimos anos, surgiram modelos de segmentação avançados baseados em redes neuronais profundas, capazes de generalizar para uma grande variedade de imagens e domínios visuais. Um exemplo recente é o modelo Segment Anything Model v2 (SAM2), desenvolvido pela Meta AI [5]. Este modelo representa uma evolução do Segment Anything Model (SAM), que oferece melhorias substanciais em termos de precisão, cobertura de objetos e capacidade de generalização.

Enquanto o SAM original já permitia gerar máscaras de forma automática e interativa com base em *prompts* simples, o SAM2 introduz melhorias na arquitetura e nos mecanismos de atenção, proporcionando uma segmentação mais precisa, robusta e rápida, especialmente em contextos não supervisionados.

No contexto deste trabalho, a segmentação de imagens desempenhou um papel fundamental na fase de pré-processamento e análise. Especificamente, o SAM2 foi empregado para realizar a segmentação semântica e de instâncias em áreas específicas de interesse, tanto nas imagens reais de entrada quanto nas imagens sintéticas criadas pelo método NeRF.

A precisão e a consistência oferecidas pelo SAM2 foram cruciais para a seleção e isolamento de regiões-chave das imagens. Esta extração precisa garante que as áreas comparadas entre o conteúdo real e o conteúdo sintético fossem rigorosamente definidas, permitindo uma análise mais controlada.

2.5 Conclusões

Neste capítulo, foram apresentados os conceitos teóricos e tecnológicos fundamentais para o trabalho, destacando os NeRF e as suas variantes, como o *Splatfacto*, que melhoram a qualidade e o desempenho na síntese de imagens 3D. Também foram discutidos métodos de deteção de similaridade entre imagens, com ênfase no *template matching*. Por fim, abordou-se a segmentação de imagens, especialmente o modelo SAM2, importante para analisar regiões específicas e comparar imagens reais com as produzidas pelos NeRF. Este enquadramento serve de base para os capítulos seguintes, que detalham os métodos, testes e resultados da deteção de compatibilidade entre imagens reais e sintetizadas.

Capítulo

3

Tecnologias e Ferramentas Utilizadas

3.1 Introdução

Este capítulo descreve as ferramentas, linguagens e bibliotecas utilizadas ao longo do projeto. A Secção 3.2 descreve as especificações do sistema utilizado. A Secção 3.3 aborda o ambiente de desenvolvimento, incluindo editores e ferramentas de virtualização. A Secção 3.4 especifica as versões das linguagens de programação utilizadas. A Secção 3.5 apresenta as bibliotecas e pacotes mais relevantes. Finalmente, a Secção 3.6 descreve o processo específico de conversão de imagens.

3.2 Especificações de Hardware

O treino e testes relacionado com o trabalho foram realizados num computador com as seguintes características:

Componente	**Descrição**
Memória RAM	16 GB
Processador	AMD Ryzen 7 5800H
GPU	NVIDIA GeForce RTX 3060
Disco	SSD de 1 TB

Tabela 3.1: Especificações do hardware utilizado no desenvolvimento do projeto.

3.3 Ambiente de Desenvolvimento

O projeto foi desenvolvido no Windows Subsystem for Linux (WSL) com a distribuição **Ubuntu 22.04 LTS**, a qual oferece compatibilidade com bibliotecas necessárias e frameworks com suporte a GPU como o *Nerfstudio*.

O ambiente virtual necessário à execução do *Nerfstudio* foi configurado através do *Conda*, conforme recomendado pela própria documentação da framework. Este ambiente garantiu o correto isolamento das dependências, incluindo versões específicas de *Python*, *PyTorch* e *CUDA*.

Foram utilizados os seguintes editores e ferramentas:

- **Sublime Text**: Utilizado para edição rápida de *scripts* e visualização de código leve, sendo ideal para ajustes pontuais sem a necessidade de um ambiente de desenvolvimento completo.

- **PyCharm**: Preferido para desenvolvimento mais estruturado, especialmente na integração e teste do modelo SAM2. O PyCharm ofereceu um suporte robusto a ambientes virtuais, facilitando a gestão de dependências.

3.4 Linguagens de Programação

Foram utilizadas três versões de Python, conforme a necessidade de compatibilidade com as bibliotecas envolvidas:

Versão	**Utilização**
Python 3.8.20	Execução do *Nerfstudio* com *Splatfacto*, em ambiente Conda
Python 3.10.12	Scripts auxiliares e processamento de imagens
Python 3.13.2	Execução do modelo SAM2

Tabela 3.2: Versões de Python utilizadas.

3.5 Bibliotecas e Pacotes Relevantes

O desenvolvimento envolveu um conjunto de bibliotecas que suportaram desde o pré-processamento de imagens até à reconstrução e segmentação. As principais bibliotecas e pacotes utilizados no projeto encontram-se resumidos na Tabela 3.3.

Biblioteca	Descrição
`torch`, `torchvision`	Treino de modelos, tensores e redes neurais
`opencv-python (cv2)`	Manipulação de imagens
`pillow`, `pillow-heif`	Conversão de imagens HEIC para JPEG
`matplotlib`	Visualização de dados
`os`	Gestão de diretorias, ficheiros e formatos
`segment_anything`	Biblioteca utilizada para o *SAM2*
`nerfstudio`, `splatfacto`	Frameworks utilizadas na reconstrução 3D e visualização de cenas

Tabela 3.3: Bibliotecas e pacotes utilizados.

3.6 Conversão de Imagens HEIC

As imagens, recolhidas por dispositivos móveis, estavam no formato **HEIC**, que apresenta uma compatibilidade limitada com diversas bibliotecas de processamento de imagem. Para resolver este problema, foi utilizado o pacote `pillow-heif`, permitindo a conversão automática para o formato **JPG**.

3.7 Conclusão

Este capítulo apresentou o ambiente técnico utilizado no desenvolvimento do projeto, incluindo hardware, software, bibliotecas e ferramentas relevantes. No capítulo seguinte, serão detalhados os *datasets* e os modelos utilizados na reconstrução e segmentação das cenas.

Capítulo

4

Implementação

4.1 Introdução

Este capítulo descreve em detalhe, os passos técnicos e metodologias aplicadas na implementação do projeto. Serão abordadas a **construção da base de dados**, incluindo a aquisição e preparação dos dados dos objetos físicos. Em seguida, o foco recairá no **método de segmentação** utilizado, e no **processo de comparação de imagens via *template matching***. Finalmente, serão apresentados os **principais excertos de código** que ilustram as fases cruciais do sistema desenvolvido.

4.2 Aquisição dos Dados

Para a criação da base de dados utilizada neste projeto, foram selecionados 10 objetos físicos distintos, de modo a garantir uma diversidade significativa em termos de forma, cor, textura e transparência. A escolha dos objetos foi realizada de forma estratégica.

A Tabela 4.1 apresenta a lista dos objetos utilizados neste estudo.

ID	**Descrição do Objeto**	Nome do Ficheiro
1	Garrafa de água cheia (transparente)	agua
2	Almofada amarela	almofada_a
3	Almofada vermelha (textura brilhante)	almofada_v
4	Caixa de óculos preta	caixinha
5	Caneca preta	caneca_j
6	Caneca preta com desenho	caneca_r
7	Copo Coca-Cola amarelo	copo_amarelo
8	Copo Coca-Cola rosa	copo_rosa
9	Figura Funko Pop!	funko
10	Rato de computador	rato

Tabela 4.1: Objetos utilizados na criação da base de dados.

A seleção incluiu objetos com características visuais distintas:

- **Variação de cor:** Objetos com cores sólidas e distintas, mas estes tem o mesmo formato, objetos foram: copo amarelo e rosa, almofada amarela e rosa .
- **Diferença de forma e volume:** Desde objetos simples como canecas, até formas mais complexas como a Figura Pop!.
- **Texturas e materiais variados:** A almofada vermelha, por exemplo, apresenta um brilho superficial que altera a sua aparência consoante a luz, enquanto a almofada amarela tem uma textura mate mais uniforme.
- **Transparência:** A garrafa de água foi incluída como caso desafiante, por ser parcialmente translúcida, testando os limites dos métodos de reconstrução volumétrica baseados em NeRF.
- **Texturas e cores semelhantes**: O rato de computador e a caixa de óculos, foram incluídos por ter uma textura semelhantes e serem levemente distintos morfologicamente.

A Figura 4.1 apresenta os dez objetos utilizados.

Figura 4.1: Vista geral dos objetos físicos utilizados para treinar os modelos NeRF.

Desta forma, foi possível construir um conjunto de dados diversificado e representativo, adequado para o treino dos modelos NeRF. A segmentação posterior de cada objeto permitiu estruturar a base de dados de forma organizada, facilitando as etapas subsequentes de análise e deteção de compatibilidade entre imagens reais e sintetizadas.

4.2.1 Captação dos dados

Os vídeos foram gravados com recurso a um dispositivo móvel, com o objetivo de capturar múltiplas perspetivas de cada objeto em rotação (vista a 360º).

Cada vídeo teve, em média, cerca de 20 segundos de duração, capturando várias centenas de *frames*. A partir destes, foram extraídos aproximadamente **100 *frames* por objeto**, uniformemente distribuídos ao longo do tempo, de forma a tentar garantir uma boa cobertura angular do objeto.

Adicionalmente, para as imagens de teste, foram capturadas 10 fotografias estáticas por objeto utilizando o mesmo dispositivo móvel. Estas fotos foram tiradas em condições variadas de pose e iluminação, com o intuito de simular diferentes cenários. Como os ficheiros originais estavam no formato **HEIC**, foi necessário convertê-los para o formato **JPG** antes do processamento, recorrendo a um simples *script*.

4.3 Segmentação dos Dados

Como referido no Capitulo 2 para a segmentação dos dados foi utilizado o SAM2. Especificamente, foi empregado para remover o fundo dos objetos nas imagens originais (para a criação dos modelos). Nas imagens renderizadas a partir dos modelos (que compõem a base de dados) e nas imagens de teste, o fundo foi preenchido a preto e as imagens foram recortadas de forma a minimizar a presença de fundo.

Figura 4.2: Esta figura apresenta, da esquerda para a direita: a imagem de teste original e a mesma imagem de teste após a segmentação, com o fundo recortado e preto. De seguida a imagem original já segmentada, com fundo transparente (usada para o treino do modelo) e a imagem antes da segmentação, com o fundo original.

4.3.1 Processamento das Máscaras

Após a identificação do objeto em um *frame* inicial (com clique manual), o SAM2 propagou a segmentação ao longo dos restantes *frames* do vídeo. Os resultados foram máscaras binárias correspondentes ao objeto identificado.

Para os objetos utilizados no treino dos modelos NeRF, as imagens foram exportadas com fundo transparente (canal alfa). Este formato é especialmente útil para treinar modelos 3D sem interferência de ruído de fundo.

Para os objetos utilizados nos testes e para as imagens renderizadas a partir dos modelos aplicou-se um fundo preto nas regiões não pertencentes ao objeto, e recorte com base na máscara, de forma a eliminar o fundo restante e facilitar comparações.

4.3.1.1 Exerto de Código

A implementação base do SAM2 foi reproduzida a partir de um tutorial em vídeo disponível no canal *Aleksandar Haber PhD*, que utiliza diretamente a função `build_sam2_video_predictor` da biblioteca `sam2` para facilitar a segmentação de vídeo. Tendo sido posteriormente adaptada com código adicional desenvolvido para aplicar fundos pretos ou transparentes, conforme as necessidades ao longo do desenvolvimento do projeto [9].

O seguinte código permite construir imagens segmentadas com fundo neutro (preto), com foco no objeto recortado. Para os dados de treino, foi aplicada uma versão ligeiramente modificada, onde a imagem é exportada com canal alfa (RGBA) e fundo transparente.

```
for out_frame_idx in range(0, len(frame_names), vis_frame_stride):
    print("Saving image", out_frame_idx)
```

```
    # Carrega a imagem original
    img_path = os.path.join(videoPath, frame_names[out_frame_idx])
    img = np.array(Image.open(img_path).convert("RGB"))
    for out_obj_id, out_mask in video_segments[out_frame_idx].items():
        if out_mask.ndim == 3:
            out_mask = np.squeeze(out_mask, axis=0)
        mask = out_mask.astype(bool)
        if mask.shape != img.shape[:2]:
            raise ValueError(f"Mask shape {mask.shape} doesn't match
                image {img.shape[:2]}")
        coords = np.argwhere(mask)
        if coords.size == 0:
            print(f"Objeto {out_obj_id} frame {out_frame_idx} esta vazio
                , pulando...")
            continue
        # Encontra as coordenadas onde mask == True
        y_min = max(coords[:, 0].min() - margin, 0)
        x_min = max(coords[:, 1].min() - margin, 0)
        y_max = min(coords[:, 0].max() + margin, mask.shape[0] - 1)
        x_max = min(coords[:, 1].max() + margin, mask.shape[1] - 1)
        # Recorta a imagem original usando a bounding box
        cropped_img = img[y_min:y_max + 1, x_min:x_max + 1]
        # Cria uma mascara recortada para essa regiao
        cropped_mask = mask[y_min:y_max + 1, x_min:x_max + 1]
        # Cria imagem preta do tamanho da bounding box
        black_background_cropped = np.zeros_like(cropped_img)
        # Copia so os pixels do objeto
        black_background_cropped[cropped_mask] = cropped_img[
            cropped_mask]
        # Salva a imagem com fundo preto e bounding box
        black_save_path = os.path.join(output_dir, f"{out_frame_idx:04}
            _obj{out_obj_id}.png")
        Image.fromarray(black_background_cropped, mode='RGB').save(
            black_save_path)
```

Excerto de Código 4.1: Criação de imagens segmentadas com fundo preto

Assim, a segmentação automática permitiu uniformizar os dados de treino e de teste, removendo o fundo, o que garantiu que as imagens contivessem unicamente o objeto de interesse. Esta preparação revelou-se fundamental para os processos posteriores de reconstrução 3D e comparação visual, ao eliminar ruído contextual e aumentar a precisão das análises.

4.4 Redimensionamento das Imagens

Para garantir consistência e otimizar as etapas subsequentes do projeto, todas as imagens utilizadas — tanto as que compõem a base de dados quanto

as imagens de teste — passaram por um processo de redimensionamento. Um *script* dedicado foi desenvolvido para identificar o menor denominador comum em termos de dimensões entre todas as imagens, e então redimensionar cada uma para essa medida mínima. Este passo padroniza os *inputs* para o *template matching* contribuindo para a robustez e eficiência da análise.

A seguinte função exemplifica o processo de identificação da menor largura e altura entre todas as imagens das várias pastas, para depois ser possível redimensionar a imagem:

```
def encontrar_menor_tamanho(pastas):
    menor_largura, menor_altura = None, None
    for pasta in pastas:
        for nome_arquivo in os.listdir(pasta):
            if nome_arquivo.lower().endswith('.png'):
                caminho_imagem = os.path.join(pasta, nome_arquivo)
                with Image.open(caminho_imagem) as img:
                    largura, altura = img.size
                    if menor_largura is None or largura < menor_largura:
                        menor_largura = largura
                    if menor_altura is None or altura < menor_altura:
                        menor_altura = altura
    return menor_largura, menor_altura
```

Excerto de Código 4.2: Função para encontrar o menor tamanho entre imagens.

4.5 Comparação das Imagens com Template Matching

Para a avaliação da consistência visual entre os dados reais e os dados sintéticos produzidos, desenvolveu-se uma ferramenta de comparação baseada na técnica de *template matching*, com uso da biblioteca `OpenCV`.

Este método tem como objetivo estimar se partes recortadas de uma imagem de teste correspondem visualmente a regiões existentes nas imagens de referência (base). A parte de comparação foi implementada num *script* em `Python` e opera da seguinte forma:

Leitura e pré-processamento: Todas as imagens das pastas de teste e da base de dados são carregadas de forma recursiva, percorrendo todas as diretorias.

Produção de recortes aleatórios: Para cada imagem de teste, são produzidos automaticamente 10 recortes quadrados de 64×64 pixeis, distribuídos

por diferentes setores da imagem (em grelha 3×3). Apenas recortes com desvio padrão acima de um limiar x, são considerados válidos, de forma a evitar zonas com muito fundo, caso possível.

Visualização dos recortes: É criada uma cópia da imagem original com os quadrados desenhados sobre as regiões selecionadas, para validação visual posterior (ver Figura 4.3).

***Template Matching*:** Cada recorte é comparado com todas as imagens da base dados com a função `cv2.matchTemplate()` que usa o método `TM_SQDIFF_NORMED`, conforme ilustrado no Código 4.3. Com este método, valores menores indicam uma melhor correspondência visual, o valor mínimo (`min_loc`) e a respetiva posição (min_loc) são registados para analisar posteriormente.

```
def template_matching_quadrado_em_base(quadrado, imagem_base):
    resultado = cv2.matchTemplate(imagem_base, quadrado, cv2.
        TM_SQDIFF_NORMED)
    min_val, _, min_loc, _ = cv2.minMaxLoc(resultado)
    return min_val, min_loc
```

Excerto de Código 4.3: Função de Template Matching

Avaliação de correspondência: Para cada recorte, é registada a imagem da base de dados com o menor *score* de correspondência e a respetiva posição. Os resultados são depois agregados por objeto para determinar qual objeto teve mais correspondências de alta qualidade.

Exportação de resultados: Os resultados são guardados num ficheiro de texto contendo:

- Melhor *match* por recorte;
- o número de ocorrências por objeto;
- o *score* mínimo (melhor correspondência) por objeto.

A Figura 4.3 mostra um exemplo da visualização dos recortes gerados numa imagem de teste, enquanto no capitulo seguinte serão apresentados os resultados extraídos da execução desta ferramenta.

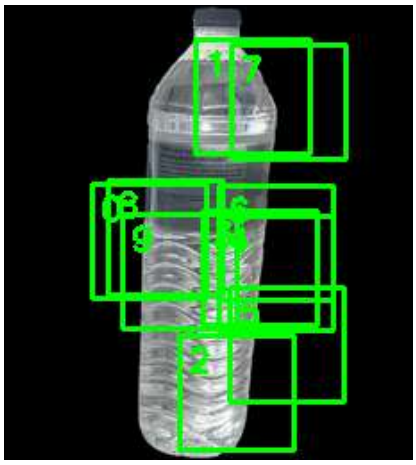

Figura 4.3: Exemplo de recortes aleatórios gerados sobre uma imagem de teste (fundo preto).

```
def generates_10_recortes_aleatorios(imagem, tamanho=64, limiar_std=10):
    h, w = imagem.shape[:2]
    recortes = []
    ...
    if np.std(recorte_gray) >= limiar_std:
        recortes.append({'imagem': recorte, 'posicao_original': (x, y)})
    ...
    return recortes
```

Excerto de Código 4.4: Geração de recortes aleatórios

Como se observa no Código 4.4, apenas são selecionados recortes com contraste suficiente (desvio padrão acima de 10), o que ajuda a evitar regiões visuais pouco informativas.

4.6 Conclusões

Este capítulo detalha as fases cruciais da implementação do projeto. Começamos com a criação da base de dados, que envolve a seleção e preparação de diversos objetos físicos. Em seguida, descreve a segmentação da imagem, com uso do modeloSAM2 e prepara-a tanto para o treino do modelo como para o teste. Por fim, apresenta-se a ferramenta de comparação baseada em *template matching*. Este método permite a criação de um recorte automático com critério de contraste e a sua comparação sistemática com a base dados, o que resulta na identificação de uma correspondência visual. O resultado obtido, é organizado por objeto e exportado para análise, estabelece uma base

quantitativa fundamental para a avaliação da semelhança entre os dados reais e a reconstrução sintética. No próximo capítulo, irá apresentar-se e discutir em detalhe este resultado.

Capítulo

5

Experiências e Resultados

5.1 Introdução

Este capítulo apresenta a análise dos resultados obtidos durante a avaliação do sistema desenvolvido para a deteção de compatibilidade entre imagens reais e imagens geradas por modelos NeRF.

Inicialmente, é fornecida uma visão geral dos resultados e destacam-se as principais observações dos testes realizados. Posteriormente, procede-se a uma análise mais detalhada, objeto a objeto, permitindo compreender o comportamento do método em diferentes condições, tipos de objetos e variações na aquisição das imagens.

Por fim, é realizada uma discussão crítica dos resultados, onde são identificados pontos fortes e limitações do método. Este capítulo constitui, assim, uma parte fundamental para a avaliação da eficácia da abordagem proposta.

5.2 Visão Geral dos Resultados

A avaliação do desempenho do sistema baseou-se em métricas quantitativas como precisão e o *score* médio por classe (objeto real), bem como na percentagem de acerto baseada na frequência de ocorrências (ver Figuras 5.1 e 5.2). Para além da análise global, foi realizada uma avaliação individual por objeto, com o objetivo de identificar padrões de desempenho e dificuldades específicas.

Os resultados revelam uma variação significativa na eficácia da deteção conforme o objeto. Objetos como `almofada_amarela` (*score* médio: 0.047) e `agua` (0.095) apresentaram os valores mais baixos, o que reflete uma correspondência mais precisa entre as imagens reais e os respetivos modelos NeRF.

Por outro lado, objetos como `caneca_r` (0.198) e `funko` (0.196) registaram *scores* mais elevados, indicando maior dificuldade na correspondência visual.

A percentagem de acerto baseada em ocorrências acompanhou, na maioria dos casos, o *score* médio. No entanto, observa-se que os objetos `funko` e `caneca_r`, que obtiveram *scores* médios mais elevados (o que indica uma menor correspondência visual), não apresentaram as menores taxas de acerto.

- A abordagem apresenta um desempenho robusto para objetos visualmente distintos, com contornos e características bem definidos.
- Entre as dificuldades mais frequentes estão a deteção de objetos com superfícies planas e as superfícies reflexivas de cores mais escuras.
- Um ponto forte é a capacidade de generalização da abordagem, mesmo em cenários com objetos de natureza variada.
- As principais limitações observadas relacionam-se com a sensibilidade a variações de iluminação, orientação e textura.

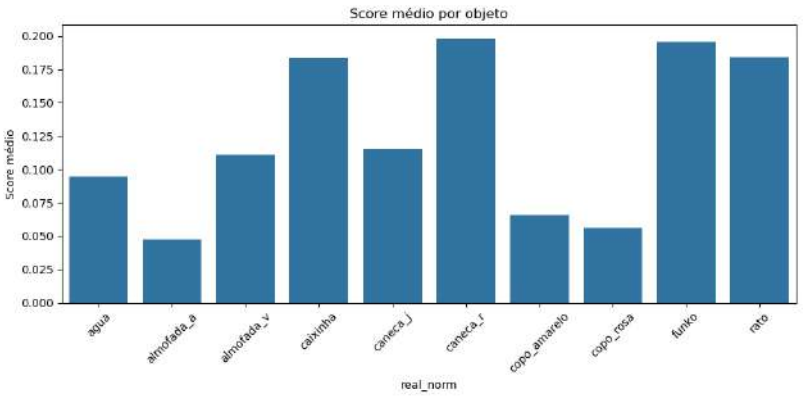

Figura 5.1: Score médio por objeto real.Valores mais baixos representam melhor correspondência.

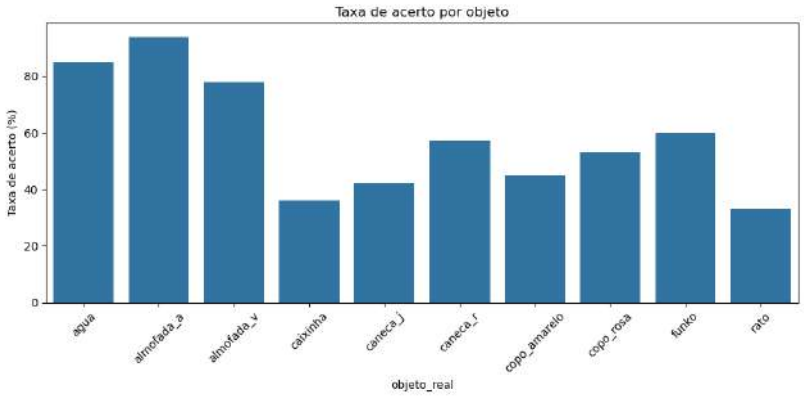

Figura 5.2: Percentagem de acerto por objeto com base na frequência de identificação do objeto correto.

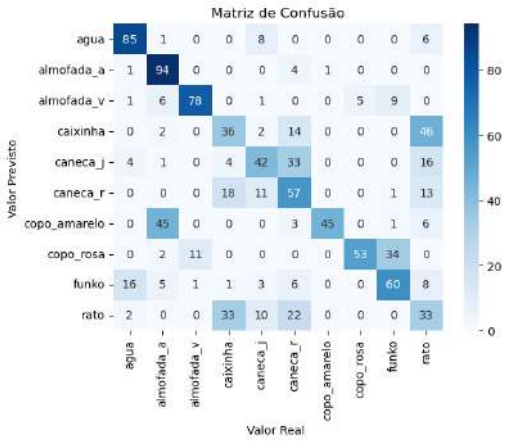

Figura 5.3: Matriz de confusão entre objetos reais e previstos.

A matriz de confusão (Figura 5.3) revela padrões específicos de confusão entre algumas classes. A classe `caixinha`, por exemplo, foi frequentemente confundida com `rato` (46 casos), e vice-versa. Em geral, considerando que mais de metade dos objetos apresentaram mais de 50 classificações corretas, pode concluir-se que o desempenho global do sistema foi satisfatório, tendo em conta que o *matching score* foi relativamente baixo, o que é positivo.

5.3 Análise Detalhada por Objeto

Conforme detalhado no Capítulo de Implementação, na Secção 4.5, cada imagem de teste é dividida em 10 quadrados aleatórios para a realização do *template matching*. Nesta secção, será realizada, também, uma análise individual dos objetos detetados e para uma avaliação mais aprofundada do desempenho do sistema, será também analisada a Curva Acumulada Rank-n. Esta curva permite entender a capacidade do sistema em posicionar corretamente a correspondência do objeto no *ranking* de resultados, indicando a proporção acumulada de vezes em que o objeto verdadeiro foi encontrado até à 10.ª posição, cada conjunto de teste., através da analise a cada quadrado.

5.3.1 Objeto 1: Garrafa de água (agua)

A `agua` é um recipiente cilíndrico de plástico predominantemente transparente com o rótulo e a tampa de cor azul. Durante a avaliação do objeto `agua`, o sistema atingiu uma taxa de acerto de 85% e verificaram-se 15 casos de confusão. Tendo em conta as imagens de teste, a imagem 7 apresentou o melhor resultado, alcançando uma taxa de sucesso de 100%. Em contrapartida, a imagem 8 demonstrou o pior desempenho, registando apenas 60% de acertos. Adicionalmente, mesmo nos casos em que a classificação para a imagem de teste 8 foi correta, o *score* de proximidade não foi o mais baixo, indicando que houve uma melhor correspondência para outras classes (incorretas) do que para a classe verdadeira do objeto nesta imagem de teste.

Ao analisar a Curva Acumulada Rank-n para o objeto `agua`(Figura 5.4), podemos observar o seguinte:

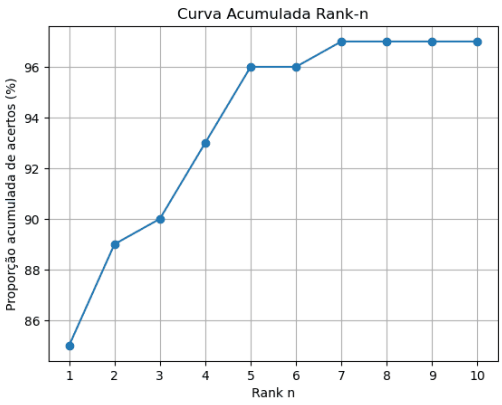

Figura 5.4: Curva Acumulada Rank-n para o reconhecimento da agua

- **Desempenho no Rank 1:** A proporção de acertos no Rank 1 é de aproximadamente **85%**. Isso significa que, para 85% das instâncias de `agua` no conjunto de testes, o sistema identificou corretamente a `agua` como a correspondência de maior confiança (com o *score* mais baixo). Este valor sugere que o algoritmo de *template matching* tem uma boa capacidade de posicionar a correspondência exata da `agua` na primeira posição.
- **Evolução da Precisão Acumulada:** A capacidade de reconhecimento do sistema melhora de forma gradual à medida que se consideram mais posições no ranking:
  - No **Rank 3**, a taxa de acertos atinge cerca de **90%**.
  - No **Rank 5**, este valor sobe para aproximadamente **96%**.
  - No **Rank 8**, o valor é de cerca de **97%** de acertos acumulados. (Nota: Observa-se um patamar de estagnação entre o Rank 7 e o Rank 10, onde a precisão acumulada permanece constante em aproximadamente 97
- **Desempenho no Top-10 (Rank 10):** Ao considerar as 10 melhores correspondências devolvidas pelo sistema, a taxa acumulada de acertos para o objeto `agua` é, no total, aproximadamente **97%**.

A Curva acumulada reforça a boa performance do sistema na priorização da correspondência correta para a `agua`, indicando uma alta capacidade de recuperação no Top-10. Este comportamento confirma que, embora o sistema possa não acertar sempre à primeira, é altamente capaz de incluir a correspondência correta entre as melhores opções.

A confusão mais frequente foi com a `caneca_j`. Esta confusão não se deve necessariamente à semelhança entre os objetos reais, mas sim a problemas na construção do modelo da `caneca_j`. Mais especificamente, durante a organização dos dados, o `COLMAP` teve dificuldades em calcular com precisão as poses (ângulos e posições) das imagens de entrada. Essa imprecisão levou à criação de um modelo renderizado com falhas estruturais: partes da geometria foram preenchidas com interpolações incorretas e triangulações artificiais, dando resultado a aparência visual menos coesa com a realidade. Consequentemente, o objeto reconstruido da `caneca_j` apresentava zonas com mistura de cores, semelhantes às características visuais da garrafa de água.

Este caso evidencia como pequenas imprecisões na estimação das poses durante a reconstrução podem comprometer significativamente a fidelidade visual do modelo renderizado, afetando de certa forma o desempenho do classificador.

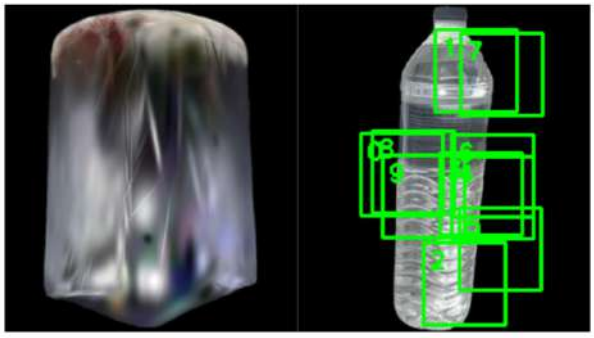

Figura 5.5: Renderização da `caneca_j` pelo método NeRF e uma das imagem teste obteve correspondência no quadrado 9.

Adicionalmente, observou-se uma confusão, embora menos frequente, com o objeto `rato`. Esta ocorrência está intrinsecamente ligada à incidência luminosa e aos efeitos de reflexão. O que faz com que, em condições de iluminação intensa e em ângulos específicos, a superfície do `rato` exiba uma reflexão acentuada, resultando numa aparência visual clara e quase branca (ver Figura 5.6). Esta característica tornava-o visualmente semelhante às imagens da garrafa de água capturadas sob forte exposição luminosa (casos dos testes 8 e 9, representados na Figura 5.7), induzindo o classificador ao erro.

Uma situação semelhante foi verificada com a `almofada amarela`, embora em menor escala. Em ângulos específicos, com incidência direta de luz intensa, a almofada adquiriu um tom muito claro, o que pode ter induzido o sistema a associá-la erroneamente à garrafa de água, cujo modelo renderizado também apresenta regiões extremamente iluminadas.

Figura 5.6: Objeto `rato` sob forte iluminação, criando um ponto de luz muito forte(praticamente branco), apresentando coloração branca semelhante à garrafa de água.

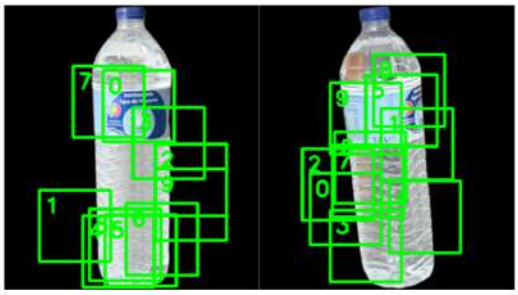

Figura 5.7: Imagem de teste da garrafa de água (testes 8 e 9 respetivamente), com exposição intensa à luz.

Estas confusões evidenciam que, para além das limitações inerentes ao modelo, as variações de iluminação e os reflexos no ambente de captura podem afetar significativamente o desempenho da classificação.

Apesar destas dificuldades, a análise dos valores de encaixe (ou *scores* de similaridade) revela que, em todos os casos com **exceção notável da imagem de teste 8**, o objeto correto obteve consistentemente o valor de encaixe mais baixo. Este padrão indica que, quando o modelo conseguiu identificar a classe correta, a precisão e a confiança nessa escolha foram elevadas.

5.3.2 Objeto 2: Almofada Amarela (almofada_a)

A classe `almofada_a` apresentou um desempenho global muito positivo, com 94 acertos em 100 tentativas, ou seja, uma taxa de acerto de 94% e 6 casos de confusão.

Para este objeto, a maioria das imagens de teste apresentaram uma taxa de sucesso de 100%. As exceções foram as imagens 7, 8 e 9. Entre estas, a imagem 7 destacou-se pelo pior desempenho, registando apenas 7 ocorrências corretas do objeto. Além disso, nesta imagem, o *score* de similaridade foi maior para objetos incorretamente identificados, sugerindo que o algoritmo atribuiu maior proximidade a uma classe incorreta.

A consistência das previsões corretas ao longo de múltiplos testes demonstra que o modelo foi capaz de identificar esta classe de forma confiável, mesmo em diferentes condições de teste. A simplicidade da almofada, caracterizada pela sua cor amarela e superfície bastante fosca, contribuiu para a sua pouca alteração de cor sob forte incidência de luz.

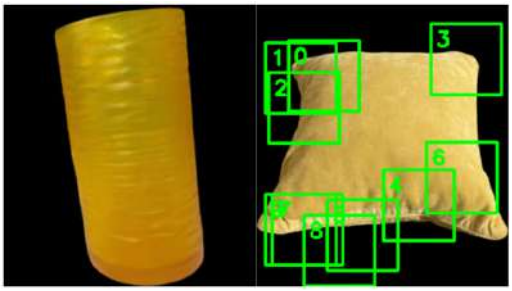

Figura 5.8: Comparação visual entre a almofada_a e o copo_amarelo. A semelhança de cor pode justificar a confusão registada.

Ao analisar a Curva Acumulada Rank-n para `almofada_amarela` (Figura 5.9), podemos observar o seguinte:

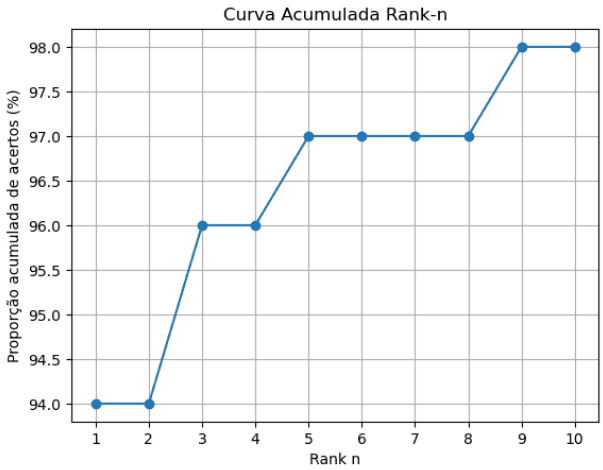

Figura 5.9: Curva Acumulada Rank-n para o reconhecimento da almofada amarela.

- **Desempenho no Rank 1:** A proporção de acertos no Rank 1 é de aproximadamente **94%**. Isto significa que, em 94% das instâncias do objeto `almofada_a` no conjunto de teste, o sistema conseguiu posicionar corretamente a correspondência na primeira posição.
- **Evolução da Precisão Acumulada:** A capacidade de reconhecimento do sistema melhora consistentemente à medida que mais opções de rank são consideradas:

  - No **Rank 3**, a proporção de acertos atinge cerca de **96%**.
  - No **Rank 5**, este valor sobe para aproximadamente **97%**.
  - No **Rank 8**, observa-se um desempenho de cerca de **97%** de acertos acumulados. (Nota: Há uma estagnação da precisão acumulada entre o Rank 5 e o Rank 8, onde ela se mantém em 97%.)

- **Desempenho no Top-10 (Rank 10):** Ao considerar as 10 melhores correspondências devolvidas pelo sistema, a taxa acumulada de acertos para o objeto `almofada_a` é de aproximadamente **98%**. Isso implica que o sistema conseguiu localizar uma imagem equivalente da `almofada_amarela` dentro da sua lista de top 10 previsões em quase todos os casos.

A análise da Curva Rank-n evidencia que o objeto `almofada_a` surge na primeira posição na maioria dos casos e tem uma frequência muito elevada dentro do Top-10. Este comportamento traduz-se num desempenho muito sólido do sistema.

A única confusão com o objeto `copo_amarelo` pode ser explicada pela semelhança cromática entre este e a `almofada_a`, dado que ambos apresentam uma tonalidade amarela predominante. Esta semelhança pode ter induzido o modelo a erro em situações específicas de iluminação ou ângulo de captura. Na Figura 5.8, apresenta-se uma imagem comparativa entre os dois objetos,onde se percebe que, dependendo da iluminação e considerando a análise de quadrados aleatórios de 64x64, algumas áreas podem apresentar-se muito semelhantes.

No caso das confusões com o objeto `caneca_r`, esta pode estar relacionada com regiões específicas da almofada que, sob determinadas condições de iluminação, refletem a luz de forma semelhante ao interior da caneca, apresentando áreas circulares claras. Essa aparência pode induzir o modelo a interpretar tais regiões como a parte interna da caneca. A Figura 5.10 ilustra este fenómeno. Relativamente à confusão entre a `almofada_a` e o objeto da classe `agua`, verificou-se que, em uma das instâncias, a garrafa de água foi identificada como sendo a almofada amarela. Esta situação pode estar associada à transparência da garrafa, que permite a interferência visual do fundo do ambiente na sua aparência. Em condições de iluminação intensa, a garrafa pode apresentar reflexos esbranquiçados e amarelados, adquirindo tonalidades semelhantes às da almofada. Este fenómeno é visível na Figura 5.11, onde a garrafa adquire tons que se assemelham aos da almofada devido à forma como a luz é refletida e à coloração do fundo que a atravessa.

Figura 5.10: Comparação visual entre a almofada_a e a caneca_r.

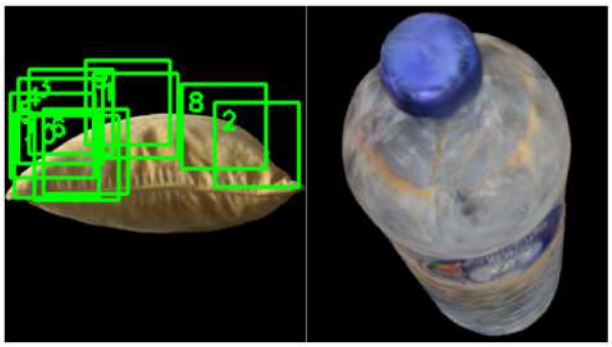

Figura 5.11: Reflexos na garrafa de água causam confusão com a almofada_a

5.3.3 Objeto 3: Almofada Vermelha (almofada_v)

A classe `almofada_v` registou 78 acertos em 100 tentativas, o que corresponde a uma taxa de acerto de 78%. Embora o modelo tenha demonstrado uma capacidade razoável de reconhecimento deste objeto, verificaram-se 22 casos de confusão, distribuídos entre cinco outras classes. Esta almofada é um objeto relativamente simples, mas a sua textura mais brilhante e a variação de cor em função da iluminação podem ter contribuído para esta taxa de acerto mais baixa.

As imagens de teste que obtiveram a melhor taxa de sucesso foram as imagens 2 e 9, ambas com 100% de acertos. Em contraste, a imagem 6 foi a que apresentou o pior desempenho, com apenas 40% de acertos, como se pode observar na Figura 5.12. Esta discrepância pode estar relacionada com fatores como a presença de objetos visualmente semelhantes, iluminação desfavorável, os quais podem comprometer a correta identificação por parte do sistema.

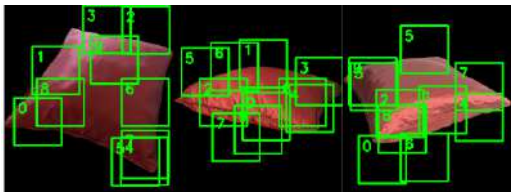

Figura 5.12: Imagens teste 2, 9 e 6 respetivamente.

Ao analizar a Curva Acumulada Rank-n para o objeto `almofada_v` (Figura 5.13), podemos observar o seguinte:

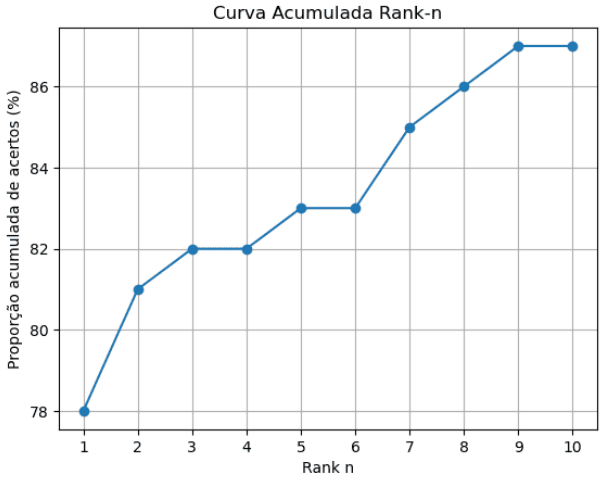

Figura 5.13: Curva Acumulada Rank-n para o reconhecimento da almofada vermelha.

- **Desempenho no Rank 1:** A proporção de acertos no Rank 1 é de aproximadamente **78%**.
- **Evolução da Precisão Acumulada:** O sistema demonstra um aumento constante na taxa de acertos acumulados à medida que mais ranks são considerados:
  - No **Rank 3**, a proporção de acertos atinge cerca de **82%**.
  - No **Rank 5**, este valor sobe para aproximadamente **83%**.
  - No **Rank 8**, observa-se um desempenho de cerca de **86%** de acertos acumulados.

- **Desempenho no Top-10 (Rank 10):** Ao considerar as 10 melhores correspondências retornadas pelo sistema, a taxa acumulada de acertos para a `almofada_v` é de aproximadamente **87%**. Isso implica que, em mais de 87% dos casos, o sistema conseguiu localizar uma imagem correta de `almofada_v` dentro da lista de top 10 previsões.

A Curva Rank-n mostra-nos que o objeto em análise é frequentemente posicionado entre as 10 melhores correspondências, o que evidencia um reconhecimento consistente e um bom desempenho por parte do sistema.

A classe mais frequentemente confundida com a `almofada_v` foi a `funko`, registando 9 ocorrências. Além disso, o `copo_rosa` foi confundido 5 vezes com a almofada_v. A principal razão para estas confusões reside na semelhança cromática. Em certas condições de iluminação, alguns modelos de `funko` e o próprio `copo_rosa` apresentam tonalidades de rosa que se assemelham à cor da `almofada_v`. Quando a luz incide diretamente sobre a almofada, a coloração vermelha pode parecer mais clara ou rosada, facilitando estas confusões. A Figura 5.14 ilustra visualmente estas semelhanças, o que contribui para a compreensão do desafio enfrentado pelo modelo.

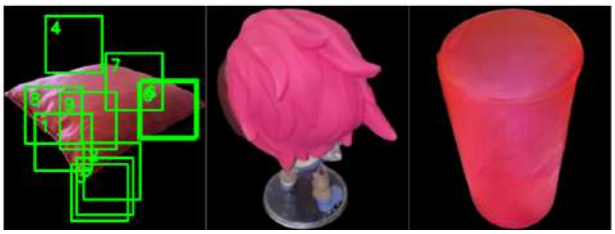

Figura 5.14: Imagens que criaram a confusão com a almofada_v

Além das confusões com os objetos `funko` e `copo_rosa`, a classe `almofada_v` foi erroneamente classificada como `almofada_a` em 6 ocasiões. Essas confusões ocorreram, em geral, em zonas de forte incidência luminosa localizadas nos cantos da almofada, onde a cor vermelha se apresenta significativamente esbatida. Tendo em consideração que o fundo das imagens foi removido, a distinção depende exclusivamente das características do próprio objeto. Nessas condições, o modelo parece ter interpretado os tons esbranquiçados como semelhantes ao amarelo claro da `almofada_a`, o que explica a dificuldade de separação entre as duas classes (ver Figura 5.15).

Uma situação semelhante ocorreu numa única instância de confusão com a classe `agua`. O objeto desta classe, sendo translúcido, refletiu a luz de modo que partes da sua superfície se tornaram visualmente semelhantes aos cantos esbatidos da `almofada_v`, mesmo sem a influência de um fundo ambiente.

Essa ambiguidade cromática é visível na Figura 5.15. Apesar da sua taxa de

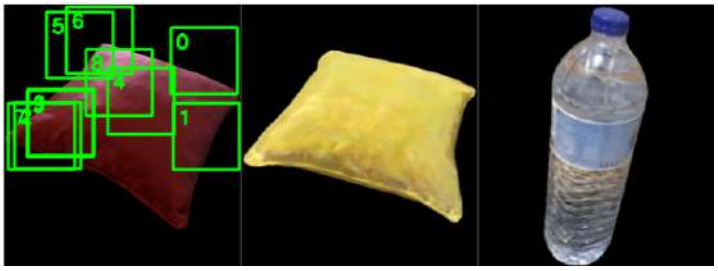

Figura 5.15: Zonas mais claras da almofada estão nos quadrados 5 e 6 o quais foram os confundidos pelos objetos adjacentes.

acerto não ter sido a mais elevada em termos de ocorrências, na maioria dos testes, o *score* de distância para a `almofada_v` foi mais baixo, com exceção de alguns casos em que o menor *score* foi atribuído ao `copo_rosa`. Esta situação pode ser justificada pela semelhança de cor entre os dois objetos, o que pode ter levado o sistema a confundi-los em determinadas circunstâncias.

5.3.4 Objeto 4: Caixa de Óculos (caixinha)

A classe `caixinha` apresentou um dos piores desempenhos entre os objetos analizados, com 36 acertos em 100 tentativas, correspondendo a uma taxa de acerto de 36% e 64 casos de confusão. Este objeto, apesar de simples e de cor preta, demonstrou ser um desafio significativo para o modelo.

As imagens de teste que obtiveram a melhor taxa de sucesso foram as 5 e 6, com 100% de acertos, enquanto as que apresentaram o pior desempenho foram as 2 e 8, com 0% de acertos, como se pode observar na Figura 5.16. Este resultado deve-se, em alguns casos, à luminosidade e ao contraste das imagens, que resultaram numa proximidade excessiva dos quadrados, o que dificulta a correta deteção dos objetos. Em outros casos, apesar de haver uma boa distribuição dos quadrados, a identificação continua a apresentar erros. Isto sugere que a proximidade visual dos conteúdos dos quadrados não está diretamente correlacionada com a correta identificação, mas influencia a probabilidade de confusão, uma vez que os quadrados apresentam características visuais muito semelhantes.

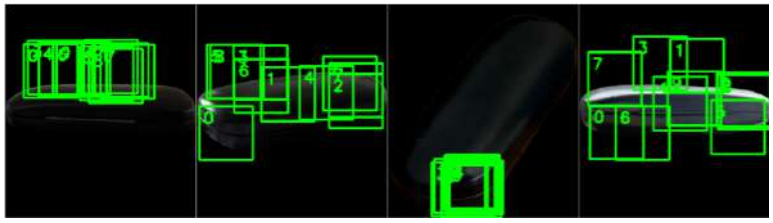

Figura 5.16: Imagens de teste 5, 6, 2 e 8 respetivamente

Ao analisar a Curva Acumulada Rank-n relativa à `caixinha` (Figura 5.17), é possível observar os seguintes aspetos:

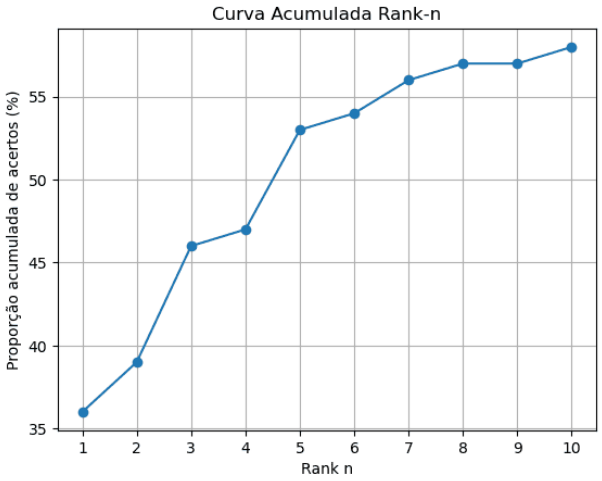

Figura 5.17: Curva Acumulada Rank-n para o reconhecimento do objeto caixa de óculos.

- **Desempenho no Rank 1:** A proporção de acertos no Rank 1 é de aproximadamente **36%**, o que indica que, em uma parte significativa dos casos, uma imagem correspondente foi corretamente identificada como a melhor correspondência pelo sistema.
- **Evolução da Precisão Acumulada:** É possível observar uma evolução gradual na taxa de acertos acumulados à medida que se consideram mais posições no ranking:
  - No **Rank 3**, a taxa de acerto atinge cerca de **46%**;
  - No **Rank 5**, este valor sobe para aproximadamente **53%**;

  - No **Rank 7**, observa-se uma taxa acumulada de acertos de cerca de **56%**.

- **Desempenho no Top-10 (Rank 10):** Ao considerar as 10 melhores correspondências retornadas pelo sistema, a percentagem acumulada de acertos para o objeto em análise é de aproximadamente **58%**. Isso implica que, para pouco mais da metade das instâncias, o sistema consegue localizar uma imagem correta dentro da lista de top 10 previsões.

A Curva CMC do objeto em análise obteve um desempenho moderado no Rank 1. No entanto, a percentagem de acerto aumenta gradualmente, mostrando que o objeto é frequentemente incluído entre as principais hipóteses. Esse padrão sugere que, embora o modelo possa ter algumas dificuldades em identificar a `caixinha` de forma direta, ele é capaz de incluí-la nas suas melhores previsões.

Grande parte das confusões da classe `caixinha` ocorreram com a classe `rato`, totalizando 46 ocorrências. Esta confusão pode ser atribuída ao fato de ambos os objetos partilharem características visuais muito semelhantes: tamanho compacto, textura homogénea e coloração preta. Apesar disso, a classe `caixinha` apresentou uma média de distância (*matching distance*) consideravelmente mais baixa nas deteções corretas em comparação com o `rato`, indicando que, quando o modelo acerta, a sua confiança é maior. Isso demonstra que o modelo consegue identificar características mais distintivas na `caixinha` quando elas estão bem visíveis, mesmo que o número de erros com o `rato` tenha sido elevado.

A Figura abaixo (5.18) ilustra um exemplo claro da semelhança entre os dois objetos, em especial nas zonas mais iluminadas, reforçando o motivo da sobreposição entre as classes.

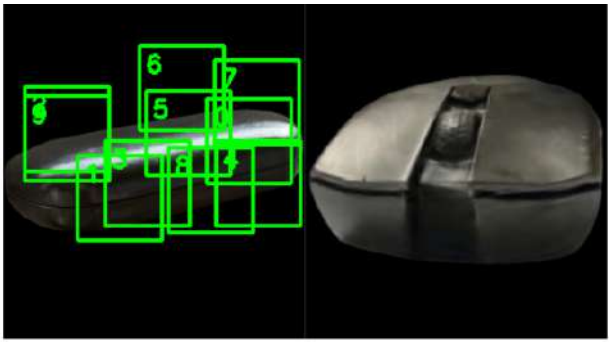

Figura 5.18: Semelhança visual entre a caixinha e o rato, destacando como a iluminação afeta a perceção das suas superfícies escuras.

Outro grupo de confusões recorrentes envolveu a classe `caneca_r`, que foi erroneamente atribuída em várias ocasiões. Este comportamento pode ser justificado por dois fatores principais: a semelhança na coloração (ambas possuem predominantemente tons escuros, especialmente pretos) e o comportamento da superfície sob iluminação intensa. A `caneca_r`, por ser feita de um material mais reflexivo, apresenta zonas de brilho que, dependendo do ângulo da luz e da posição da câmara, assemelham-se ao brilho observado na superfície da caixa de óculos. Essas reflexões criam padrões visuais parecidos, sobretudo em imagens com incidência de luz forte, tornando a distinção entre os dois objetos mais difícil para o modelo. Este fenómeno pode ser observado na figura 5.19

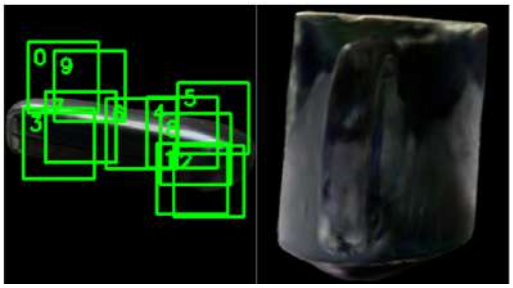

Figura 5.19: Comparação visual entre a caixa de óculos e a caneca_r, isto evidencia os padrões de brilho semelhantes sob iluminação intensa.

Os erros de classificação que envolveram a `caneca_j` podem, em grande parte, ser justificados por características visuais que dificultam a distinção entre classes. A `caneca_j` apresenta zonas com tonalidades mais esbranquiçadas quando exposta a luz direta e também devido ao facto de as imagens de base para o modelo terem incluído uma superfície branca, o que faz com que certas áreas da sua superfície se assemelhem visualmente a partes claras de outras classes — como, por exemplo, a `almofada_a`. A imagem fornecida (5.20) ilustra claramente como estes padrões de iluminação podem comprometer a capacidade do modelo para distinguir entre classes diferentes, e que, sob determinadas condições de luminosidade, tornam-se visualmente ambíguas.

Em síntese, embora a classe `caixinha` tenha, na maioria dos casos, apresentado uma distância média mais baixa — o que reflete uma boa correspondência visual quando a deteção é correta —, verificaram-se também limitações significativas. Houve situações em que o objeto real não foi sequer identificado como uma das opções pelo método de comparação, e noutras, obteve a maior distância de comparação. Este facto evidencia as fragilidades do modelo face a variações específicas nas condições das imagens, como

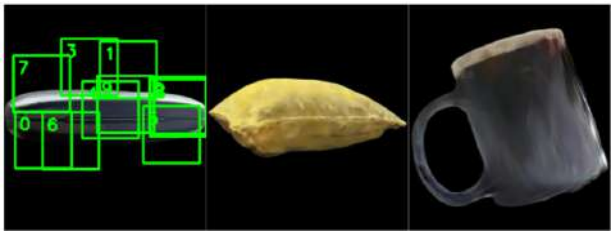

Figura 5.20: Impacto da iluminação na caneca_j e almofada_a: formação de zonas esbranquiçadas que dificultam a distinção de classes.

iluminação ou ângulo de captura, que comprometem a sua capacidade de reconhecimento

5.3.5 Objeto 5: Caneca Preta (caneca_j)

A classe `caneca_j` apresentou um desempenho misto ao longo dos testes. Embora tenha sido corretamente identificada em 42 das 100 tentativas, registaram-se 58 casos de confusão, correspondendo a uma taxa de acerto de 42%. Esta taxa relativamente baixa pode ser explicada por vários fatores visuais que contribuíram para a semelhança com outras classes. A imagem de teste com melhor desempenho foi a número 9, com um total de 7 acertos, enquanto a de pior desempenho foi a imagem 1, com apenas 2 acertos. Estas diferenças de desempenho estão intrinsecamente relacionadas com fatores como iluminação, ângulo de captura ou semelhanças visuais com outros objetos. As imagens correspondentes encontram-se representadas na Figura 5.21.

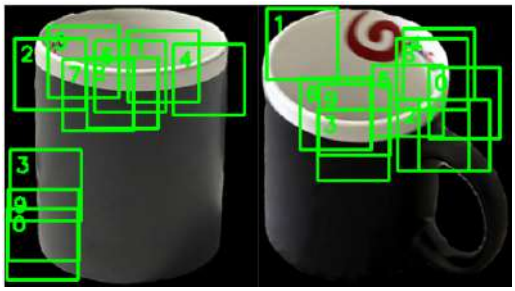

Figura 5.21: Imagens teste 9 e 1 respetivamente.

Ao analisar a Curva Acumulada Rank-n para a `caneca_j` (Figura 5.22), é possível observar o seguinte:

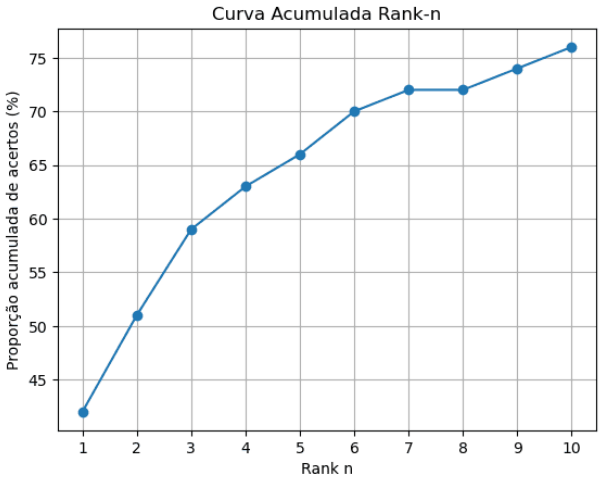

Figura 5.22: Curva Acumulada Rank-n para reconhecimento da caneca_j.

- **Desempenho no Rank 1:** A proporção de acertos no Rank 1 é de aproximadamente **42%**.
- **Evolução da Precisão Acumulada:** O sistema apresenta um aumento consistente na taxa de acertos à medida que se consideram mais ranks:
  - No **Rank 3**, a percentagem de acertos atinge cerca de **59%**.
  - No **Rank 5**, este valor sobe para aproximadamente **66%**.
  - No **Rank 8**, observa-se um desempenho de cerca de **72%** de acertos acumulados.
- **Desempenho no Top-10 (Rank 10):** Considerando as 10 melhores correspondências devolvidas pelo sistema, a proporção acumulada de acertos para o objeto em causa é de aproximadamente **76%**. Isso significa que, em mais de três quartos dos casos, o sistema consegue localizar uma imagem correta da `caneca_j` dentro das 10 principais previsões.

O comportamento da curva acumulada mostra que o modelo tem uma capacidade razoável de identificar diretamente a `caneca_j` e frequentemente a inclui entre as principais hipóteses. As confusões visuais podem decorrer de semelhanças estéticas ou outros fatores que dificultam o reconhecimento preciso em 100% dos casos.

A confusão mais frequente ocorreu com a classe `caneca_r`, justificada pela proximidade estética entre ambas, como se pode observar na Figura 5.23, são ambas canecas pretas, com superfícies que refletem a luz de forma semelhante. Estas reflexões criam padrões visuais próximos que desafiam os métodos baseados em aparência.

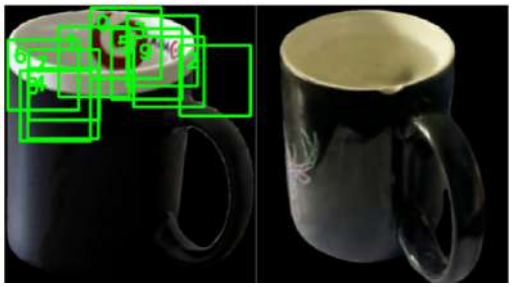

Figura 5.23: Caneca j e caneca r, pode-se observar que tem algumas semelhanças desde a forma às cores.

Para além disso, registaram-se várias confusões com o rato de computador e a caixa de óculos, dois objetos que, apesar de diferentes em forma, apresentam texturas visuais semelhantes à da `caneca_j`. O `rato`, em particular, foi identificado em diversas imagens como a correspondência principal.

Importa ainda referir que algumas imagens da `caneca_j` apresentam zonas mais esbranquiçadas, devido a reflexos ou à incidência direta de luz, o que poderá ter causado confusão com outras classes, como a `agua` e a `almofada_a`, cujas áreas mais claras se aproximam do branco, resultando em classificações erróneas.

Em síntese, os resultados demonstram que, embora a `caneca_j` tenha sido frequentemente confundida com outros objetos, em aproximadamente metade dos casos apresentou a distância média mais baixa, o que revela uma consistência notável no reconhecimento, ainda que não dominante.

5.3.6 Objeto 6: Caneca Preta com Desenho (caneca_r)

A classe `caneca_r` registou 57 acertos em 100 tentativas, o que resulta numa taxa de acerto de 57%. Embora o modelo tenha demonstrado uma capacidade razoável de reconhecimento para este objeto, verificaram-se 43 casos de confusão, distribuídos por 3 classes principais e uma ocorrência isolada noutra classe. A `caneca_r` é um objeto de formato relativamente simples, com uma textura preta lisa e brilhante, e possui um desenho idêntico em ambos os lados. A imagem de teste 2 apresentou o melhor resultado, com 100% dos quadrados corretamente identificados como o objeto real. Em contraste, as ima-

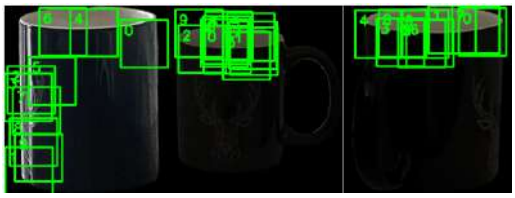

Figura 5.24: Imagens teste: 2 (melhor resultado), 4 e 5 (pior resultado), respetivamente, em linha.

gens 4 e 5 demonstraram o pior desempenho, registando zero acertos. Esta falha de reconhecimento é provavelmente atribuída à fraca luminosidade das imagens, que poderá ter comprometido a visibilidade das características do objeto. As Figuras 5.24 ilustram essas imagens de teste.

Com analise da Curva acumulada da `caneca_r` (Figura 5.25), podemos observar o seguinte:

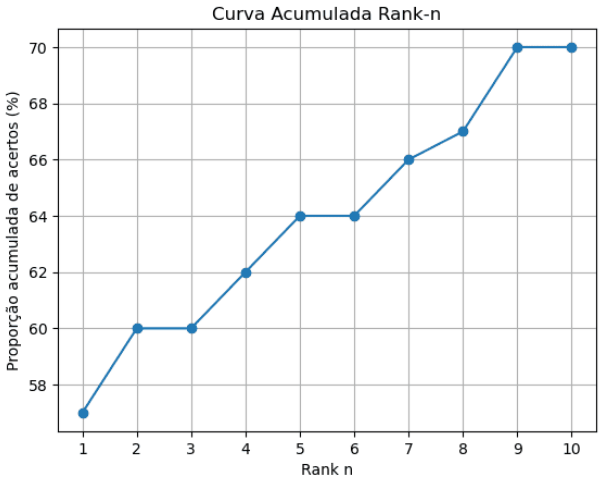

Figura 5.25: Curva Acumulada Rank-n para o reconhecimento da caneca_r.

- **Desempenho no Rank 1:** A proporção de acertos no Rank 1 é de aproximadamente **57%**. Este valor é superior ao de objetos anteriormente analisados como a `caixinha`, e encontra-se num patamar inferior ao da `almofada_a` e `agua`, isto indica que a primeira previsão do sistema para a `caneca_r` é correta em mais da metade dos casos.
- **Evolução da Precisão Acumulada:** O sistema apresenta um aumento consistente na taxa de acertos acumulados à medida que mais ranks são considerados:

  - No **Rank 3**, a proporção de acertos atinge cerca de **60%**.
  - No **Rank 5**, este valor sobe para aproximadamente **64%**.
  - No **Rank 8**, observa-se um desempenho de cerca de **67%** de acertos acumulados.

- **Desempenho no Top-10 (Rank 10):** Considerando as 10 melhores correspondências retornadas pelo sistema, a proporção acumulada de acertos para o objeto `caneca_r` é de aproximadamente **70%**. Isto significa que, em 70% dos casos, o sistema consegue localizar uma imagem correta da `caneca_r` dentro da sua lista de top 10 previsões.

Ao analisarmos a *Cumulative Match Characteristic* (CMC), observamos que o modelo reconhece consistentemente o objeto entre as principais hipóteses de correspondência. No entanto, persistem dificuldades em identificar corretamente o objeto como a previsão mais provável em todos os cenários, especialmente em casos de maior complexidade visual ou semelhança com outros objetos.

Contrariamente às expectativas iniciais de que a maioria das confusões ocorreria com a classe `caneca_j` — devido à sua semelhança estrutural e material com a `caneca_r` e também pelo que foi descrito na subsecção 5.3.5 — os erros mais recorrentes ocorreram com a `caixinha` (18 ocorrências) e o `rato` (13 ocorrências). Esta tendência sugere que o modelo teve dificuldades em distinguir objetos com formas simples e cores semelhantes, especialmente sob determinadas condições de iluminação. Nestas condições, superfícies planas e pouco texturizadas refletem a luz de maneira similar. A confusão com a `caneca_j`, embora esperada devido à semelhança de cor, textura e formato estrutural quase idêntico, ficou em terceiro lugar com 11 ocorrências. Esta dificuldade de distinção puramente visual é um fator relevante, conforme ilustrado na Figura 5.26, onde se observam as semelhanças que desafiam o modelo.

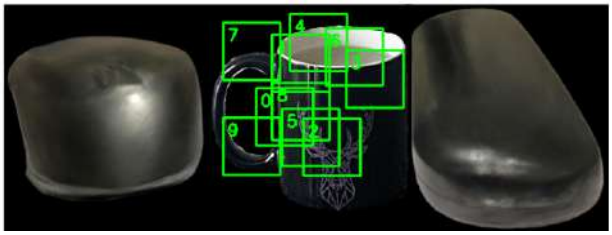

Figura 5.26: Exemplos visuais dos objetos

Apesar destas confusões, é importante notar que, em praticamente todos os casos de teste (com exceção de um), o valor da distância no *template matching* foi mais baixo para a `caneca_r`. Isso indica que o sistema, mesmo com diversas confusões com outros objetos, ainda considerou a `caneca_r` como o objeto visualmente mais próximo em termos de similaridade quando a deteção foi correta.

5.3.7 Objeto 7: Copo Coca-Cola Amarelo (copo_amarelo)

O copo amarelo da Coca-Cola não apresentou os melhores resultados, tendo obtido 45 em 100 de acertos, correspondendo a uma taxa de acerto de 45%, sendo que no conjunto de teste o melhor desempenho foi na imagem teste 7, onde obteve 100% de acertos, e os piores resultados foram a imagem teste 4 e 9, onde foram observados apenas dois acertos - as três imagens podem ser observadas na Figura 5.27. Este é um objeto simples, cilíndrico e de cor amarela com um relevo ondulado.

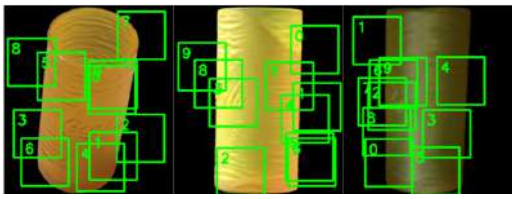

Figura 5.27: Imagens de teste, 7 a qual obteve melhor resultado, 4 e 9 as quais obtiveram o pior resultado respetivamente.

Através da analise da Curva Acumulada Rank-n para o `copo_amarelo` (Figura 5.28), podemos observar o seguinte:

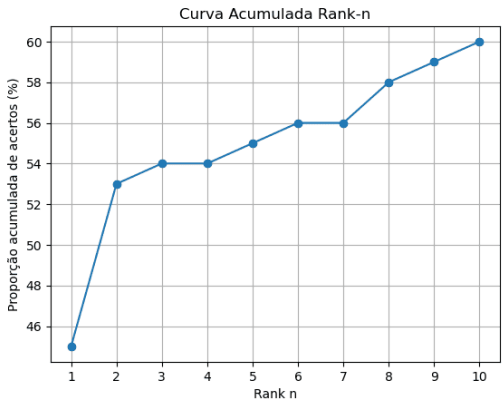

Figura 5.28: Curva Acumulada Rank-n para o reconhecimento do copo_amarelo.

- **Desempenho no Rank 1:** A proporção de acertos no Rank 1 é de aproximadamente **45%**. Este valor indica que o sistema consegue posicionar o `copo_amarelo` correto como a primeira correspondência de maior confiança em quase metade dos casos.
- **Evolução da Precisão Acumulada:** Conforme são consideradas mais posições no ranking, observa-se um crescimento progressivo na taxa de acertos acumulados:
  - No **Rank 3**, a percentagem de acertos atinge cerca de **54%**.
  - No **Rank 5**, este valor sobe para aproximadamente **55%**.
  - No **Rank 8**, observa-se um desempenho de cerca de **58%** de acertos acumulados.
- **Desempenho no Top-10 (Rank 10):** Ao considerar as 10 melhores correspondências devolvidas pelo sistema, a taxa acumulada de acertos para o `copo_amarelo` é de aproximadamente **60%**. Isto significa que, em mais da metade dos casos, o sistema consegue localizar uma imagem correspondente dentro da sua lista das top 10 previsões.

A Curva Acumulada Rank-n revela um desempenho razoável no Rank 1, com um aumento gradual da taxa de acerto em ranks superiores. Isso indica que o modelo tem alguma capacidade de posicionar o objeto em análise como a primeira hipótese de previsão e, mais frequentemente, incluí-lo entre as principais.

O erro mais frequente foi a classificação incorreta de `copo_amarelo` como `almofada_a`, representando 82% dos erros (45 de 55). Tal resultado deve-se ao

facto de estes dois objetos serem da mesma cor. Como se pode observar na Figura 5.29, dependendo da luz, as tonalidades de amarelo tornam-se muito semelhantes e em certas partes do copo a textura ondulada é menos percetível.

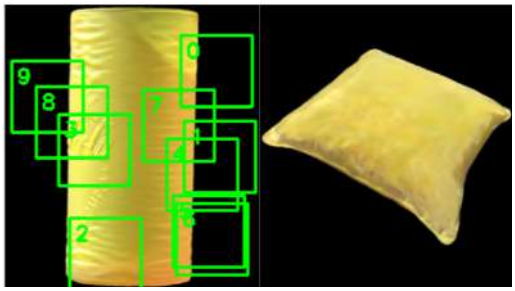

Figura 5.29: Imagem teste do copo e imagem da base de dados da almofada amarela, onde é possível observar a semelhança de cor.

Outros erros envolveram os objetos `rato` (6 vezes), `caneca_r` (3 vezes) e `funko` (1 vez). Quanto à confusão com o `rato` e a `caneca_r` estas ocorreram quando a imagem de teste estava em condições de luminosidade mais reduzidas, onde o copo se apresentava mais escuro e a tonalidade de amarelo menos intensa, o que pode ter induzido o modelo ao erro.

Além da elevada taxa de confusão entre `copo_amarelo` e `almofada_a`, observou-se que, nos casos em que a classificação foi incorreta, o *score* de similaridade (distância) calculado pelo *template matching* foi frequentemente mais baixo com a `almofada_a`, mesmo este não sendo o objeto correto.

Idealmente, o valor de similaridade mais baixo deveria ocorrer com o copo amarelo, dado que é o objeto real. No entanto, a almofada amarela apresentou valores mais baixos na maioria desses erros, o que sugere que o algoritmo está a encontrar maior semelhança visual com a almofada amarela. Este fenómeno pode estar relacionado com a semelhança de cor entre os dois objetos (ambos com tonalidade amarela), o que pode levar o algoritmo a privilegiar características de cor e não considerar tão acentuadamente as texturas.

5.3.8 Objeto 8: Copo Coca-Cola Rosa (copo_rosa)

Para o objeto, `copo_rosa`, o resultado foi razoável, sendo reconhecido 53 vezes em 100 tentativas, ou seja, obteve uma taxa de acerto de 53%. Os restantes 47 casos foram erros de classificação entre três classes diferentes. Este é um objeto cilíndrico com um leve relevo e coloração rosa. O melhor desempenho foi observado na imagem teste 2, que alcançou 90% de acertos. Nesses 9 casos

de reconhecimento correto, o *score* de distância mínima foi consistentemente mais baixo. Em contraste, o pior resultado foi na imagem teste 8, onde o objeto não foi reconhecido em nenhum dos quadrados. As Figuras 5.30 ilustram essas imagens de teste e as variações de cor resultantes de diferentes ambientes de captura.

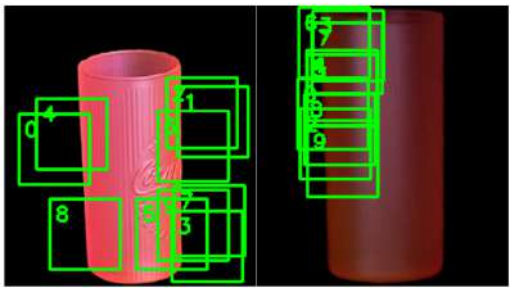

Figura 5.30: Imagens teste 2, com o melhor resultado, e 8 com o pior resultado, respetivamente

Ao analisar a Curva Acumulada Rank-n para o `copo_rosa` (Figura 5.31), é possível observar o seguinte:

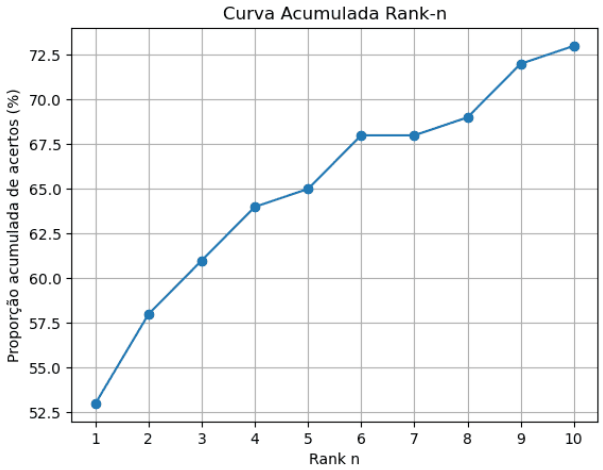

Figura 5.31: Curva acumulada Rank-n para reconhecimento do copo_rosa.

- **Desempenho no Rank 1:** A proporção de acertos no Rank 1 é de aproximadamente **53%**. Este valor indica que o sistema consegue posicionar o `copo_rosa` como a correspondência mais provável em mais da metade dos casos.

- **Evolução da Precisão Acumulada:** O sistema demonstra um aumento consistente na taxa de acertos acumulados à medida que mais ranks são considerados:
  - No **Rank 3**, a taxa de acertos atinge cerca de **61%**.
  - No **Rank 5**, este valor aumenta para aproximadamente **65%**.
  - No **Rank 8**, observa-se um desempenho de cerca de **69%** de acertos acumulados.
- **Desempenho no Top-10 (Rank 10):** Ao considerar as 10 melhores correspondências devolvidas pelo sistema, a taxa acumulada de acertos para o objeto em causa termina por ser aproximadamente **73%**. Tal resultado indica que, apesar de não ter 100% de acertos no Rank 1, o sistema consegue identificar corretamente uma instância do `copo_rosa` dentro do conjunto das 10 melhores previsões, na grande maioria dos casos.

De forma geral, a análise da curva acumulada do `copo_rosa` evidencia que o sistema tem um desempenho sólido, que vai melhorando progressivamente à medida que mais hipóteses são consideradas. O facto de alcançar uma taxa de acerto de 73% no Top-10 demonstra que o modelo é capaz de identificar corretamente o objeto na maioria dos casos, ainda que nem sempre na primeira posição. Este comportamento sugere que o sistema tem uma boa perceção visual do `copo_rosa`.

O elevado número de confusões com o `funko` deve-se, em grande parte, à presença de áreas extensas com tonalidades rosa no próprio `funko`, visíveis a partir de certos ângulos (conforme ilustrado na Figura 5.32). Essa semelhança cromática resultou numa forte sobreposição entre o *template* do `copo_rosa` e regiões da figura Pop (*funko*), induzindo o algoritmo a correspondências incorretas.

As confusões com a `almofada_v` parecem estar associadas a variações de luminosidade nas imagens. Partes da almofada apresentaram maior claridade devido à incidência de luz, o que resultou num tom aproximado à imagem teste do `copo_rosa`. Inversamente, o copo apresentou-se mais escuro em algumas imagens de teste, como na imagem 8, o que facilitou a confusão entre os objetos (Figura 5.33). Por sua vez, as confusões com a `almofada_amarela` ocorreram sobretudo em imagens onde o `copo_rosa` assumiu uma tonalidade mais alaranjada ou amarelada, possivelmente por reflexo ou desvio cromático nas condições de captura. Essa alteração visual tornou mais difícil distinguir entre os dois objetos. Este fenómeno é ilustrado na Figura 5.34, onde é visível como o copo adquire uma aparência mais amarelada, o que pode ter

Figura 5.32: A imagem da base de dados demonstra que esta área é predominantemente rosa, o que justifica confusão com objetos rosados.

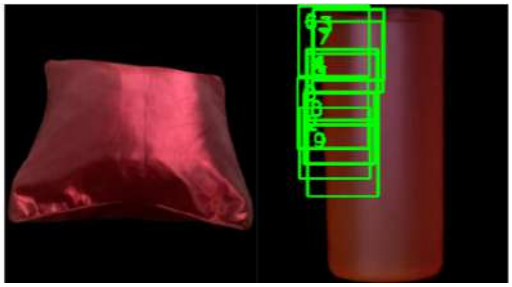

Figura 5.33: Imagem teste numero 8 e imagem da base de dados que deu mais vezes match

induzido o erro. Por fim, relativamente ao *score* de distância obtido através do *template matching*, verificou-se que, na maioria dos casos, o valor mais baixo foi atribuído ao `copo_rosa`. No entanto, existiram algumas exceções, nomeadamente em casos de identificação errónea do `funko` — nestes, o *score* mais baixo foi por vezes atribuído a este objeto, sugerindo que o algoritmo o considerou mais semelhante ao *template*. Além disso, um caso em que o objeto correto (`copo_rosa`) não foi sequer identificado entre as hipóteses, o que comprometeu o resultado, independentemente do *score* obtido. Com exceção destas situações, sempre que o `copo_rosa` foi corretamente reconhecido, foi também o objeto com o valor de distância mais baixo, o que demonstra consistência entre a métrica e a escolha feita pelo algoritmo.

5.3.9 Objeto 9: Figura Pop (funko)

O objeto Figura Pop (Funko) destacou-se como o mais complexo entre os itens analisados, devido à sua variação de formas e cores. Inicialmente, supôs-

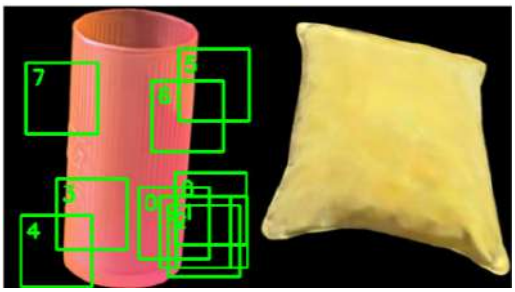

Figura 5.34: Copo rosa, com tons mais amarelados e almofada amarela para comparação

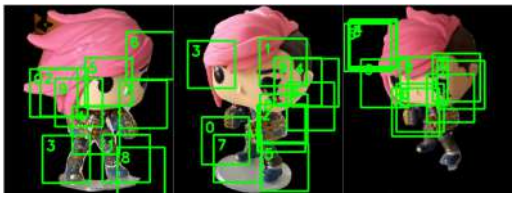

Figura 5.35: Os melhores resultados foram observados nas imagens de teste 1 e 4, e o pior na imagem 6.

se que essa complexidade visual poderia facilitar a identificação automática, ao oferecer mais pistas distintivas ao modelo. No entanto, os resultados demonstraram uma realidade mais ambígua: o objeto obteve 60 acertos em 100 amostras, correspondendo a uma taxa de erro de 40%. Esse desempenho sugere que, apesar da riqueza visual, a complexidade também introduziu desafios ao modelo, confirmando parcialmente a hipótese de que maior detalhe pode aumentar a dificuldade de reconhecimento em determinadas condições.

Adicionalmente, observou-se um elevado número de confusões, sendo este o objeto com o maior número de classes confundidas (sete classes distintas) em toda a análise.

O melhor desempenho foi registado nas imagens teste 1 e 4, ambas com 8 acertos. O pior resultado ocorreu na imagem teste 6, onde se registaram apenas 4 acertos. As Figuras 5.35 ilustram essas imagens de teste.

Com base na Curva Acumulada Rank-n para o `funko` (Figura 5.36), podemos observar o seguinte:

- **Desempenho no Rank 1:** A proporção de acertos no Rank 1 é de aproximadamente **60%**.
- **Evolução da Precisão Acumulada:** À medida que o número de ranks

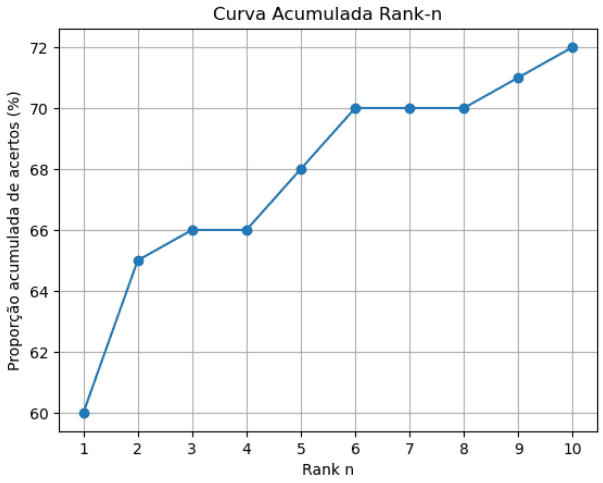

Figura 5.36: Curva Acumulada Rank-n para o reconhecimento da Figura Pop.

incluídos aumenta, o desempenho acumulado do sistema melhora gradualmente:

  - No **Rank 3**, a taxa de acertos atinge cerca de **66%**.
  - No **Rank 5**, este valor sobe para aproximadamente **68%**.
  - No **Rank 8**, observa-se um desempenho de cerca de **70%** de acertos acumulados.

- **Desempenho no Top-10 (Rank 10):** Tendo em conta as 10 melhores correspondências dadas pelo sistema, a proporção acumulada de acertos para o objeto `funko` é de aproximadamente **72%**. Tal resultado indica que, em mais de dois terços dos casos, o sistema consegue localizar uma imagem correspondente do `funko` dentro das 10 primeiras previsões.

De modo geral, a curva acumulada para o `funko` apresenta um bom desempenho inicial, seguido de uma melhoria progressiva à medida que mais posições no ranking são consideradas. Para este objeto, o sistema atinge 72% de acertos no Top-10, o que demonstra uma capacidade sólida de identificação. O modelo consegue reconhecê-lo com boa consistência dentro de um conjunto mais alargado de hipóteses.

A confusão mais frequente ocorreu com a classe `agua`. Um exemplo deste fenómeno é visível na imagem de teste 0, onde se observa confusão nas pernas da figura. Para esta mesma imagem, o mesmo acontece com a `caneca_r`. Isto sugere que estas confusões decorrem da semelhança de cores observada

nessas partes do objeto `funko` com as cores dos objetos `agua` e `caneca_r`, conforme ilustrado na Figura 5.38 para fins de comparação. No entanto, o *score* de distância é significativamente mais alto para estas classes incorretas do que quando o objeto é corretamente identificado como o `funko`. Situação análoga ocorreu com as classes `rato`, `caneca_j` e `caixinha`, onde partes do corpo do `funko` (como braços, tronco e pernas) foram confundidas com imagens desses objetos na base de dados.

As almofadas amarela e vermelha foram confundidas com a cabeça da Figura Pop. A confusão com a almofada vermelha, especificamente na região do cabelo do `funko`, é compreensível dada a coloração similar. Já a almofada amarela foi confundida no quadrado 7, na região da face do `funko`, como ilustrado na Figura 5.37. Isso sugere que a tonalidade amarelada da face do `funko` pode ter gerado confusão com a almofada amarela.

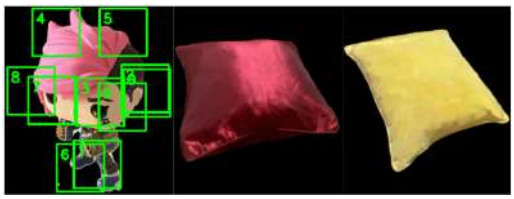

Figura 5.37: Ilustração dos pontos de confusão na imagem de teste, onde as almofadas amarela e vermelha foram incorretamente associadas à cabeça da Figura Pop. A almofada vermelha causou confusão na área do cabelo e a amarela na face (quadrado 7)

Figura 5.38: Imagem do teste 0: quadrados 3 e 4 mostram previsões incorretas para as classes caneca_r e agua.

Apesar do número significativo de confusões na identificação, o `funko` foi corretamente identificado como o objeto mais próximo na maioria dos casos. Isto é, o *score* de proximidade (distância mínima) foi consistentemente mais baixo para o objeto `funko`, mesmo em cenários de alta confusão com outras classes.

5.3.10 Objeto 10: Rato de Computador (rato)

A classe `rato` registou 33 acertos em 100 tentativas, correspondendo a uma taxa de acerto de 33%. Embora o modelo tenha sido capaz de reconhecer corretamente o objeto em algumas situações, o seu desempenho global revela fragilidades significativas, com 67 previsões incorretas, distribuídas predominantemente por três classes: `caixinha` (33 ocorrências), `caneca_r` (22 ocorrências) e `caneca_j` (10 ocorrências). Houve ainda 2 casos de confusão com a classe `agua`, o que indica que, apesar da predominância dos erros concentrados, o modelo apresenta uma certa dispersão nos padrões de confusão. O objeto `rato` é relativamente simples: preto, sem grande relevo, possui alguns botões e é pequeno.

Entre os diferentes testes realizados, a imagem que apresentou melhor desempenho foi a 0, com 9 acertos em 10 tentativas (90%). Este resultado demonstra que, em condições favoráveis, o modelo é capaz de efetuar uma classificação precisa do objeto. Em contraste, a imagem teste de 8 obteve 0 acertos, com todas as previsões incorretamente atribuídas a outras classes. A Figura 5.39 ilustra os exemplos de melhor e pior desempenho do modelo na tarefa de reconhecimento do objeto `rato`, evidenciando o impacto das condições visuais sobre a performance do sistema de classificação. Ao analisar a

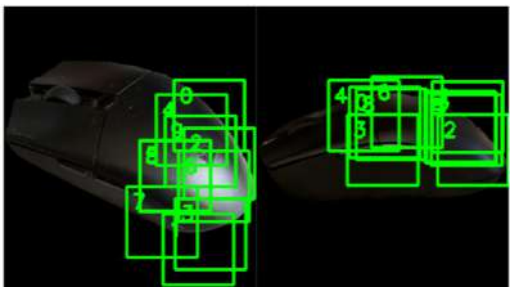

Figura 5.39: Imagens teste 0, melhor resultado, e 8 respetivamente

Curva Acumulada Rank-n para o objeto "Rato de Computador"(Figura 5.40), é possível observar o seguinte:

- **Desempenho no Rank 1:** A proporção de acertos no Rank 1 é de aproximadamente **35%**.
- **Evolução da Precisão Acumulada:** O sistema apresenta um aumento gradual na taxa de acertos acumulados, à medida que mais ranks são considerados:
  - No **Rank 3**, a taxa de acertos atinge cerca de **49%**.

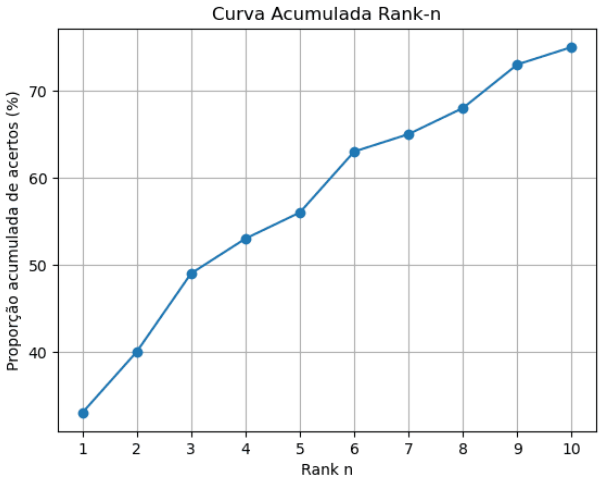

Figura 5.40: Curva Acumulada Rank-n para o Reconhecimento do Objeto "Rato de Computador".

  - No **Rank 5**, este valor sobe para aproximadamente **56%**.
  - No **Rank 8**, observa-se um desempenho de cerca de **68%** de acertos acumulados.

- **Desempenho no Top-10 (Rank 10):** Ao analisar as 10 melhores correspondências fornecidas pelo sistema, observa-se que a taxa acumulada de acertos para o objeto `rato` é cerca de **74%**. Isso indica que, em aproximadamente três quartos das situações, o sistema consegue identificar corretamente uma imagem do `rato` dentro de 10 previsões.

A curva acumulada para o `rato` revela que a taxa de acertos acumulados atinge cerca de 74% no Top-10, indicando que o sistema é capaz de reconhecer o objeto com boa consistência dentro de um conjunto ampliado de hipóteses, mesmo que não seja sempre a primeira previsão.

A classe que originou mais confusões foi a `caixinha`, o que é compreensível, uma vez que tanto este objeto como o `rato` são pequenos, apresentam texturas bastante semelhantes e partilham a mesma cor. Apesar disso, nos casos em que a confusão ocorreu apenas entre estas duas classes, a classe `rato` foi, de forma consistente, aquela que apresentou o valor de distância mais baixo, o que é um resultado positivo. Este comportamento era, aliás, expectável, dado que a `caixinha` também foi a classe que registou mais confusões com o `rato`, conforme observado na Subsecção 5.3.4.

Relativamente às classes `caneca_r` e `caneca_j`, estas também foram frequentemente confundidas entre si e, em menor grau, com o objeto `rato`. Esta

tendência poderá estar relacionada com o facto de todos apresentarem coloração preta e possuírem texturas simples, como se pode observar na Figura 5.41. A `caneca_r` revelou-se mais propensa a ser confundida, possivelmente devido à sua cor preta mais uniforme e sólida. Em contraste, a `caneca_j`, embora igualmente preta, apresenta algumas variações na tonalidade, com reflexos brancos subtis, o que poderá ter contribuído para uma taxa de confusão ligeiramente inferior.

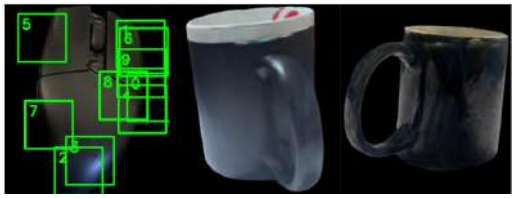

Figura 5.41: Objetos `caneca_r`, `caneca_j` e `rato`, pode se observar que estes partilham características visuais semelhantes, especialmente na cor

No que diz respeito à classe `agua`, as poucas confusões observadas ocorreram predominantemente em zonas do `rato` com elevada incidência de luz, o que justifica, até certo ponto, a semelhança visual com a agua. No entanto, este tipo de confusão não é desejável, uma vez que compromete a fiabilidade da classificação.

Por fim, no que se refere ao *score* de distância mínima, o seu desempenho revelou-se insatisfatório na tarefa de atribuir a menor distância ao objeto correto. Em aproximadamente 60% dos casos, o objeto `rato` obteve um valor de distância superior ao atribuído a outras classes, o que indica uma limitação significativa do método neste contexto

5.4 Discussão dos Resultados

Apesar das limitações e confusões, um ponto forte consistente observado em vários objetos (como a `agua`, a *almofada_a*, a *almofada_v*, a `caneca_r`, o `copo_rosa` e o `funko`) é que, na maioria dos casos em que a classificação foi correta, o objeto verdadeiro obteve consistentemente o menor *score* de encaixe (distância). Isto sugere que o algoritmo de *template matching* é capaz de identificar a semelhança visual mais forte com o objeto correto. As falhas de classificação, portanto, muitas vezes não decorrem da incapacidade de encontrar o "melhor encaixe", mas sim da existência de outros objetos com similaridades elevadas (falsos positivos) que acabam por "vencer"na votação ou da ambiguidade visual de certos quadrados. No caso da caixa de óculos, por exemplo, o modelo

conseguiu identificar características distintivas quando bem visíveis, apesar do elevado número de erros com o `rato`.

As dificuldades com a `caixinha` e o `rato` também realçam a limitação do método de dividir a imagem em 10 quadrados aleatórios de 64x64. Embora essa abordagem possa verificar alguma robustez, pode conduzir à perda de contexto global do objeto. Em casos de objetos pequenos ou com poucas características texturais, os quadrados podem apresentar ambiguidade visual, dificultando a deteção correta.

A análise detalhada das Curvas Acumuladas Rank-n revela um padrão de desempenho interessante em todo o conjunto de objetos. Embora a precisão no Rank 1 varie amplamente — sendo muito alta para objetos como a `almofada_amarela` (94%) e a `agua` (85%), mas mais baixa para a `caixinha` (36%) e o `rato` (35%) —, todos os objetos demonstram uma notável capacidade de recuperação em rankings superiores. A taxa de acertos acumulados no Top-10 é consistentemente alta, superando 50% para a maioria dos objetos e chegou a 98% para a `almofada_amarela`. Este comportamento sugere que o sistema é bastante eficaz em gerar um conjunto de hipóteses que, com alta probabilidade, inclui a correspondência correta, mesmo que nem sempre a posicione como a primeira escolha. Isso demonstra a soli10 do modelo em fornecer previsões relevantes.

Em síntese, o desempenho global do sistema, considerando que mais da metade dos objetos atingiu uma taxa de acerto superior a 50%, pode ser considerado satisfatório. No entanto, as análises revelam que as principais limitações residem na sensibilidade a variações de iluminação e reflexos, a semelhança cromática entre certos objetos e, crucialmente, a qualidade da reconstrução dos modelos NeRF a partir de dados de pose imperfeitos. Estas observações sublinham a importância da escolha dos objetos e das condições de captura das imagens reais, bem como a necessidade de aprimorar tanto os modelos 3D quanto os métodos de correspondência visual.

5.5 Conclusão

Este capítulo apresentou a análise e discussão crítica dos resultados obtidos com o sistema de comparação entre imagens reais e modelos NeRF. Desde a visão geral até à análise detalhada por objeto, foi possível compreender as nuances do desempenho, identificando tanto as classes onde a correspondência foi eficaz (evidenciada pelo *scores* mais baixos), quanto as que apresentaram maiores desafios.

A discussão aprofundada dos resultados permitiu constatar que, apesar das limitações intrínsecas ao método (como a sensibilidade a iluminação e

reflexos, a ambiguidade de objetos com semelhanças cromáticas e morfológicas, e as inerentes falhas na reconstrução NeRF), o algoritmo de *template matching* demonstrou capacidade para identificar o melhor encaixe visual na maioria dos casos, para o objeto correto. Esta observação reforça a validade do mecanismo de correspondência, ao mesmo tempo que realça a necessidade de abordagens complementares para mitigar os "falsos positivos"na fase de votação e os desafios colocados por quadrantes ambíguos.

Em suma, este capítulo forneceu a fundamentação empírica crucial para a avaliação da abordagem proposta, dando destaque aos pontos fortes e as principais limitações. Os *insights* aqui apresentados serão determinantes para a formulação das conclusões gerais do presente trabalho, a serem detalhadas no capítulo seguinte.

Capítulo

6

Conclusões e Trabalho Futuro

6.1 Conclusões Principais

O objetivo principal deste projeto foi investigar a viabilidade de desenvolver um sistema capaz de detetar a compatibilidade visual entre imagens reais de objetos e os seus modelos 3D gerados por um NeRF, onde para realização de identificação de objetos foi usada uma abordagem baseada em *template matching*.

Para tal, foi implementado um conjunto de procedimentos, onde se recorreu a uma *framework* (nerfstudio) para a criação dos modelos NeRF a partir de imagens de objetos reais. Estes modelos permitiram a renderização de imagens de referência, que constituíram a base de dados para a comparação. A etapa final envolveu a aplicação da técnica de *template matching* sobre quadrados aleatórios, comparando secções das imagens de teste com a base de dados dos modelos NeRF.

A abordagem desenvolvida demonstrou ser promissora e viável, embora com margem para melhorias. O sistema alcançou um desempenho global satisfatório, classificando corretamente mais de metade dos objetos testados.

A eficácia da deteção revelou-se, contudo, significativamente dependente das características do objeto e das condições de captura. Objetos visualmente distintos, como a garrafa de água e a almofada amarela, obtiveram os melhores resultados, caracterizados por *scores* de proximidade consistentemente baixos, o que indica uma forte correspondência visual e um bom "encaixe"com os respetivos modelos NeRF.

Contrariamente, objetos com superfícies reflexivas, cores escuras ou formas mais ambíguas, como a caixa de óculos e o rato, apresentaram maiores desafios. As principais fontes de erro identificadas foram a sensibilidade a va-

riações de iluminação e reflexos, a semelhança cromática e morfológica entre classes distintas e, notavelmente, as imprecisões decorrentes da fase de reconstrução dos modelos NeRF, em particular no que concerne à estimação das poses.

A análise aprofundada dos resultados confirmou que, apesar de o algoritmo ser frequentemente capaz de identificar o melhor encaixe para o objeto correto (evidenciado pelos *scores* mais baixos), este obteve muitas confusões ao nível dos quadrados individuais. Tal levou a classificações incorretas quando existiam classes com similaridades enganosas ou quando os quadrados apresentavam ambiguidade visual, sobrepondo-se ao verdadeiro melhor encaixe na fase de votação.

Em última análise, as conclusões deste trabalho sublinham a necessidade e o potencial de abordagens híbridas que combinem a fidelidade dos modelos NeRF com mecanismos de classificação mais robustos e menos suscetíveis a ruído e ambiguidades visuais.

6.2 Trabalho Futuro

Otimização da Reconstrução NeRF: Um aspeto que mereceria um maior aprofundamento seria a otimização da fase de reconstrução dos modelos NeRF, nomeadamente a qualidade da estimação das poses das câmaras (e.g., com o COLMAP). As falhas identificadas na caneca_j demonstraram como imprecisões nesta fase comprometem significativamente a fidelidade do modelo 3D e, consequentemente, a sua utilidade para o *template matching*. Apesar das várias tentativas para contornar estes problemas, o tempo disponível para a realização do projeto impediu uma resolução detalhada e exaustiva. **Expansão e Diversificação do Dataset:** Embora o *dataset* utilizado contasse com um número razoável de objetos variados, a sua escala poderia ser enriquecida através da inclusão de mais objetos com diferentes graus de semelhança visual. Tal permitiria uma análise mais abrangente dos dados. Contudo, a recolha manual e a preparação dos dados consomem bastante tempo, e o tempo total de realização do projeto era limitado, o que restringiu o âmbito do *dataset* no contexto deste trabalho. **Melhoria e Diversificação dos Métodos de Reconhecimento:** Com um maior tempo de desenvolvimento, seria interessante explorar e integrar outros métodos de reconhecimento de objetos, ou mesmo a junção de abordagens distintas. Por exemplo, um classificador, como uma rede neuronal convolucional, poderia ser treinado para analisar os resultados do *template matching* nos vários quadrados. Esta abordagem permitiria integrar um contexto global e tomar decisões mais robustas perante ambiguidades visuais, o que poderia mitigar as confusões observadas

em objetos como a caixa de óculos e o rato, e consequentemente melhoraria a análise dos dados.

Bibliografia

[1] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *arXiv preprint arXiv:2103.13415*, 2021.

[2] Wisarut Chantara, Ji-Hun Mun, Dong-Won Shin, and Yo-Sung Ho. Object Tracking using Adaptive Template Matching. *IEIE Transactions on Smart Processing and Computing*, pages 1–10, February 2015.

[3] Nazanin Sadat Hashemi, Roya Babaie Aghdam, Atieh Sadat Bayat Ghiasi, and Parastoo Fatemi. Template matching advances and applications in image analysis. *arXiv preprint arXiv:1610.07231*, 2016.

[4] Bernhard Kerbl, Georgios Kopanas, Thomas Leimk
"uhler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.

[5] Meta. SAM 2: Segment Anything in Images and Videos, 2023. [Online] `https://github.com/facebookresearch/segment-anything`. Acedido em 08 de maio de 2025.

[6] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *ACM Transactions on Graphics (TOG)*, 39(4):1–15, 2020.

[7] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

[8] Nerfstudio Team. Nerfstudio Documentation, 2022. [Online] Acedido em 10 de junho de 2025.

[9] Aleksandar Haber PhD. Meta segment anything 2.1 - sam2.1 - install locally and run in python and windows, 2024. Vídeo do YouTube.

[10] Johannes L. Schoenberger. Colmap — colmap 3.12.0.dev0 documentation. `https://colmap.github.io`, 2024. Acedido em [Data de Acesso, ex: 11 de Abril de 2025].

[11] Haithem Turki, Michael Zollhöfer, Christian Richardt, and Deva Ramanan. Pynerf: Pyramidal neural radiance fields. *haithemturki.com/pynerf/*, 2023.