Universidade da Beira Interior
Departamento de Informática

Background Subtraction on SOTA Object Detectors: Does It Worth?

Elaborado por:

Joel Sebastian Leiva Tapia

Orientador:

Hugo Proença

30 de junho de 2025

Agradecimentos

A conclusão deste trabalho, bem como da grande maior parte da minha vida académica não seria possível sem a ajuda e apoio da minha família e dos meus amigos. Gostaria de fazer um agradecimento especial a João Mendoça, Manuel Garcia e Herique Ramos pelo seu apoio significativo neste projeto, bem como ao professor Hugo Proença pela orientação e ajuda que me deu a mim e aos meus colegas. Finalmente, quero deixar um último e especial agradecimento ao meu gato que tornou possível toda esta carreira em Engenharia Informática.

Conteúdo

Conteúdo **iii**

Lista de Figuras **v**

Lista de Tabelas **vii**

1 Introdução **1**
1.1 Enquadramento 1
1.2 Motivação UBI 1
1.3 Objetivos 2
1.4 Organização do Documento 2

2 Estado da Arte **3**
2.1 Introdução 3
2.2 Visão por Computador 3
2.3 Background Substraction 4
2.3.1 K-NN 4
2.3.2 GMM e MOG 2 5
2.4 Object Detection Models 6
2.4.1 Faster R-CNN 6
2.4.2 YOLO 7
2.5 Conclusões 8

3 Tecnologias e Ferramentas Utilizadas **9**
3.1 Introdução 9
3.2 OpenCV 10
3.3 Frameworks dos Modelos de Deteção 10
3.3.1 Ultralytics e YOLO 10
3.3.2 Detectron2 e Faster R-CNN 10
3.4 Amazon Web Services 11
3.5 Conclusões 11

4 Experiências e resultados **13**
4.1 Introdução 13

4.2 Processamento de Vídeos 13
4.2.1 Aplicação de Máscaras e Operações Morfológicas 14
4.2.2 KNN 14
4.2.3 MOG2 16
4.3 Integração da Pipeline 16
4.3.1 Aplicação dos Modelos de Deteção 17
4.3.1.1 Variante YOLO 17
4.3.1.2 Variante Faster R-CNN 19
4.3.2 Fluxo: Background Subtraction e Modelos de Deteção . 20
4.3.3 Filtragem das Previsões com Base nas Máscaras 21
4.4 Execução na Nuvem 22
4.5 Métricas e Validação 24
4.6 Resultados e Discussão 26
4.6.1 Visão Geral 27
4.6.2 Casos Particulares 28
4.6.2.1 Tomas Próximas 29
4.6.2.2 Tomas Intermédias 29
4.6.2.3 Tomas Distantes 30
4.7 Análise dos Resultados 30
4.7.1 Melhores Desempenhos 30
4.7.1.1 Geral 30
4.7.1.2 Modelo & Background Subtraction 31
4.7.1.3 Deteções Filtradas 32
4.7.2 Influência dos planos de câmara 32
4.7.3 Interpretação dos Resultados 34
4.8 Conclusões 36

5 Conclusões e Trabalho Futuro **37**
5.1 Conclusões Principais 37
5.2 Trabalho Futuro 38

Bibliografia **41**

Lista de Figuras

4.1 Deteção com YOLO sem subtração de fundo 35
4.2 Deteção com YOLO com subtração de fundo 35

Lista de Tabelas

4.1 Resultados gerais do YOLO 27
4.2 Resultados gerais do Faster R-CNN 27
4.3 Resultados dos modelos YOLO e Faster R-CNN com variantes baseadas em background subtraction (Tomas Distantes) 29
4.4 Resultados dos modelos YOLO e Faster R-CNN com variantes baseadas em background subtraction (Tomas Distantes) 29
4.5 Resultados dos modelos YOLO e Faster R-CNN com variantes baseadas em background subtraction (Tomas Distantes) 30
4.7 Comparativa entre YOLO e Faster R-CNN, com as tecnicas de Background Substraction 31
4.6 Comparativa entre YOLO e Faster R-CNN 31
4.8 Comparativa entre as deteções filtradas pelas tecnicas de Background Substraction 32
4.9 Comparação tendo em conta o ângulo da câmara 33

Acrónimos

YOLO	You Only Look Once
R-CNN	Region Based Convolutional Neural Networks
BGS	Background Subtraction
MOG2	Mixture of Gaussians
K-NN	K-nearest neighbors
KDE	Kernel Density Estimation
GMM	Gaussian Mixture Model
NMS	Non-Max Suppression
IOU	Intersection over Union
RPN	Region Proposal Network
SVM	Support Vetor Machine
CUDA	Compute Unified Device Architecture
AWS	Amazon Web Services
EC2	Elastic Compute Cloud
S3	Simple Storage Service
IoU	Intersection over Union
ROI	Região de interesse
SOTA	State-Of-The-Art

Capítulo

1

Introdução

1.1 Enquadramento

Nos últimos anos, o avanço na área da Visão por Computador tem sido marcado pela criação de modelos que utilizam a Aprendizagem Profunda (Deep Learning), especialmente em tarefas como a deteção de objectos e, neste caso, a deteção de pessoas.

Modelos como o You Only Look Once (YOLO) e o Region Based Convolutional Neural Networks (R-CNN) trouxeram grandes melhorias ao resultado global da deteção de objectos, tornando-os o foco da investigação nesta área. No entanto, antes do aparecimento destes métodos baseados em redes neuronais profundas, eram utilizadas técnicas tradicionais como o Background Subtraction (BGS) separar objectos em movimento em vídeos, especialmente em casos de vigilância ou monitorização. Com a aceitação generalizada de novos paradigmas de deteção, a importância destas técnicas clássicas tem vindo a diminuir gradualmente. Mesmo assim, resta saber se será possível melhorar o desempenho combinando-as com os novos modelos, nomeadamente através da filtragem dos dados de entrada.

1.2 Motivação UBI

A principal motivação deste trabalho é entender de forma crítica e prática o funcionamento dos modelos de deteção computacional mais avançados, bem como entender as técnicas tradicionais que envolviam o BGS. Para além de aprofundar os temas mais actuais em torno da inteligência artificial, e especialmente numa área que mistura processamento de imagem e, portanto, envolve um aspeto visual que também gera muitos desafios para resolver.

1.3 Objetivos

Os objetivos deste projeto serão divididos em três partes, tendo em conta que queremos medir empiricamente cada modelo e técnica em geral.

1. Analisar o desempenho dos modelos atuais (YOLO-R-CNN) em vídeos capturados por drones, e assim ter uma média estimada da sua precisão em casos especiais em que a altura, o ângulo e as sombras podem jogar contra o modelo.
2. Comparar resultados e responder à pergunta: Vale a pena usar técnicas de Background Subtraction para melhorar o desempenho destes modelos?
3. Analisar cada caso específico não só para compreender os resultados, mas também para compreender mais profundamente o funcionamento dos algoritmos implementados, bem como obter tópicos para trabalho futuro.

1.4 Organização do Documento

De modo a refletir o trabalho que foi desenvolvido, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o projeto, a motivação para a sua escolha, o enquadramento do tema, os objetivos definidos e a respetiva organização do documento.
2. O segundo capítulo – **Estado da Arte** – apresenta uma revisão dos principais trabalhos relacionados com a subtração de fundo e os modelos modernos de deteção de objetos, destacando os seus princípios, aplicações e limitações.
3. O terceiro capítulo – **Tecnologias e Ferramentas Utilizadas** – descreve os conceitos mais relevantes no âmbito deste projeto, bem como as tecnologias e ferramentas empregues durante o seu desenvolvimento.
4. O quarto capítulo – **Experiências e Resultados** – detalha a metodologia adotada, a arquitetura da solução implementada, os testes realizados e a análise crítica dos resultados obtidos.
5. O quinto capítulo – **Conclusões e Trabalho Futuro** – apresenta as conclusões retiradas a partir dos resultados alcançados e propõe possíveis linhas de investigação futura.

Capítulo

2

Estado da Arte

2.1 Introdução

Esta secção apresenta a principal literatura relacionada com as técnicas de subtração de fundo e os modelos de deteção de objectos baseados em redes neuronais, geralmente designados por SOTA (State-of-the-Art)

Além disso, faremos uma breve revisão da sua história antes de explicarmos os seus princípios no capítulo seguinte.

2.2 Visão por Computador

A visão computacional é um subcampo da inteligência artificial que procura emular as capacidades da visão humana, que inclui não só a capacidade de ver, mas também de perceber e extrair informação significativa do que é observado.

Dentro deste âmbito, vários tópicos surgiram para tentar abarcar a complexidade deste desafio, alguns dos principais sobre os quais nos iremos debruçar neste trabalho foram:

- Segmentação de Imagens: A divisão de uma imagem em várias regiões com o objetivo de encontrar regiões com significado.
- Deteção e Reconhecimento de Objectos: Foca-se na localização e identificação de objectos de interesse tanto em imagens como em sequências de vídeo.
- Estimativa e seguimento de movimentos: Identificar e seguir objectos ao longo do tempo.

2.3 Background Substraction

O Background Subtraction é uma técnica utilizada para detetar objetos em movimento a partir de uma sequência de imagens ou vídeo, normalmente de uma câmara estática. Estas técnicas permitem a segmentação de imagens ou fotogramas para separar os objetos em movimento (Foreground) e o fundo estático (Background), uma vez que, em alguns casos, é aconselhável trabalhar apenas em regiões de interesse.

Coloca-se então a questão: como construir um algoritmo capaz de gerar inicialmente um modelo de fundo preciso, como podemos detetar e apontar de forma robusta os objetos em movimento e de que forma atualizaremos eficazmente o modelo em função das variações do ambiente (luz, oclusões, etc.).

Neste relatório, analisaremos o Mixture of Gaussians (MOG2) e o K-nearest neighbors (K-NN), dois algoritmos clássicos que enfrentaram este problema de formas diferentes e que nos ajudarão a ter uma boa perspetiva destas técnicas.

2.3.1 K-NN

K-Nearest Neighbors K-NN é uma técnica tradicional de classificação baseada na ideia de que a classe de um determinado dado pode ser inferida a partir das classes dos seus vizinhos mais próximos. Para isso, é necessário definir uma métrica de distância, frequentemente a distância euclidiana, e um valor K, que determina quantos vizinhos serão considerados para a decisão.

No nosso caso, utilizamos o conceito de K-NN no contexto da estatística não paramétrica, onde não assumimos uma distribuição teórica prévia dos dados. Em vez disso, a distribuição é inferida diretamente a partir dos dados observados. Isso significa que não é necessário ajustar os dados a um modelo estatístico teórico, mas sim construir um modelo com base na própria distribuição empírica.

No contexto da Background Subtraction, aplicamos esta ideia considerando que os dados de entrada são os valores de cor (R,G,B) de cada pixel ao longo do tempo, o que define um espaço tridimensional. A distância entre o valor atual do pixel e os valores anteriores armazenados é calculada usando a distância Euclidiana normalizada.

Esta distância, no entanto, não é usada diretamente para classificar o pixel como Background ou Foreground. Em vez disso, é usada como base para realizar uma Kernel Density Estimation (KDE). Esta técnica permite estimar a densidade de probabilidade de ocorrência de um determinado valor no espaço de cor.

Como proposto por Zivkovic e van der Heijden [1]

> *"Density estimation using a uniform kernel starts by counting the number of samples k from the data set XT that lie within the volume V of the kernel. The volume V is a hypersphere with diameter D. The density estimate is given by"*

$$p(\vec{x}) = \frac{1}{TV} \sum_{i=t-T}^{t} K\left(\frac{\|\vec{x} - \vec{x}_i\|}{D}\right) \tag{2.1}$$

Intuitivamente, este processo pode ser entendido como desenhar uma esfera (ou hiperesfera) em torno do ponto atual $\vec{x}$ e contar quantas amostras anteriores $\vec{x}_i$ estão dentro dela. A densidade é então estimada como a razão entre o número de vizinhos, determinado pela função núcleo $K(\cdot)$ que calcula adistância euclidiana entre o pixel atual $\vec{x}$ e as amostras passadas, e o volume V do kernel, que é proporcional à dimensão D dos dados. O parâmetro T define o intervalo temporal considerado, de forma que o cálculo se faz somando todas as amostras desde o instante $t - T$ até o tempo atual t. Se esta densidade estimada for suficientemente alta, o pixel é classificado como fundo; caso contrário, como primeiro plano.

No algoritmo utilizado, o número de amostras T é limitado (por exemplo, 50 ou 100) e o número de vizinhos K é definido empiricamente como uma fração destas, tipicamente

$$K \approx 0.1T$$

O que reduz a complexidade e torna a estimativa mais robusta a variações locais.

2.3.2 GMM e MOG 2

O Gaussian Mixture Model (GMM) é um modelo probabilístico que utiliza uma combinação de distribuições Gaussianas para representar a distribuição dos valores observados numa determinada região. Para tal, assume que os dados de cada pixel podem pertencer simultaneamente a várias distribuições (componentes) com diferentes probabilidades. O modelo constrói estas distribuições Gaussianas (ou “componentes”) e atribui a cada novo valor de pixel uma probabilidade de pertencer a cada uma delas. Ao longo do tempo, o número de componentes é ajustado dinamicamente para encontrar a quantidade ótima que descreve adequadamente a variabilidade do fundo. Uma vez que o fundo pode mudar ao longo do tempo, os parâmetros de cada Gaussiana, média (valor esperado da cor), variância (dispersão) e peso (frequência relativa), são continuamente atualizados através de equações de atualização.

Graças a este processo adaptativo, o modelo é capaz de aprender e ajustar a sua representação de fundo de forma eficiente. O algoritmo MOG2 implementa completamente esta lógica, mas introduz uma melhoria essencial, em vez de usar um número fixo de distribuições gaussianas, ajusta dinamicamente a quantidade de distribuições de acordo com a complexidade da cena. Isso permite representar com maior precisão as cores e adaptar-se melhor a ambientes em constante mudança

2.4 Object Detection Models

2.4.1 Faster R-CNN

O Region Based Convolutional Neural Networks (R-CNN) foi um dos primeiros modelos a aplicar as redes neuronais convulsionais à tarefa de deteção e classificação de objetos. O seu funcionamento baseia-se numa abordagem em duas etapas. Em primeiro lugar, ao receber uma imagem de entrada, esta é segmentada em várias regiões através de um algoritmo conhecido como Pesquisa Seletiva. Este algoritmo identifica áreas da imagem que potencialmente podem conter objetos, reduzindo assim o número de regiões que devem ser processadas pela rede neuronal e evitando gastar recursos em áreas irrelevantes do fundo. Uma vez identificadas estas regiões de interesse, cada uma é redimensionada e passada individualmente por uma rede neuronal convulsionai para extrair as suas características. Em seguida, um classificador externo, que normalmente é um Support Vetor Machine (SVM) treinado separadamente, atribui uma classe a cada região detetada. E, em paralelo, é utilizado um regressor para ajustar as coordenadas das caixas delimitadoras (caixas delimitadoras).

Apesar do seu impacto, a R-CNN tem várias limitações importantes:

- O algoritmo de pesquisa seletiva é lento e não aprende, o que impõe um custo computacional elevado e constante.
- O treino é efetuado em várias etapas separadas (CNN e SVM), o que complica a otimização conjunta.
- O tempo de inferência é muito elevado, tornando-o inviável para aplicações em tempo real.

Estas desvantagens levaram ao desenvolvimento de modelos sucessores mais eficientes.

O Fast R-CNN surge como uma melhoria direta do R-CNN, com o objetivo de reduzir o seu elevado custo computacional. Em vez de processar cada região proposta separadamente, este modelo começa por extrair um mapa de características de toda a imagem utilizando uma rede convolucional. Em seguida, sobre esse mapa, aplica as propostas de regiões (ainda geradas pelo algoritmo externo Selective Search), que são redimensionadas com uma técnica chamada ROI Pooling para obter regiões de tamanho fixo. Isto permite a ordenação e a regressão das caixas num único passo de treino, aumentando significativamente a eficiência. No entanto, a dependência da Pesquisa Seletiva continua a ser uma limitação.

Faster R-CNN resolve esta última deficiência substituindo a pesquisa seletiva por uma Region Proposal Network (RPN) [2].

> *A Region Proposal Network (RPN) takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness score. We model this process with a fully convolutional network.*

Esta rede capaz de aprender a gerar propostas de regiões relevantes torna o modelo completamente treinável de ponta a ponta, eliminando a necessidade de algoritmos externos e alcançando uma melhoria considerável tanto na precisão quanto na velocidade.

2.4.2 YOLO

O You Only Look Onces (YOLOs) é um modelo de deteção e classificação de objetos baseado numa única rede neural convulsionai, o que lhe permite atingir velocidades significativamente mais elevadas do que outros detetores mais complexos.

O funcionamento do YOLO baseia-se numa série de passos encadeados que permitem a deteção e classificação de objetos de forma eficiente. Primeiro, a imagem de entrada é redimensionada para um tamanho fixo e dividida numa grelha de SxS células. Cada uma destas células é responsável por determinar se o centro de um objeto se encontra dentro dos seus limites. Em caso afirmativo, a célula fornece um número fixo de caixas delimitadoras (um valor de confiança associado a cada caixa, que reflete tanto a certeza da presença de um objeto como a precisão espacial da previsão) e as probabilidades de pertencer a cada classe possível.

Para avaliar a precisão de uma caixa prevista em relação ao objeto real, utilizamos o índice Intersection over Union (IOU), que compara a sobreposição

entre a caixa prevista e a caixa real. Posteriormente, de forma a eliminar previsões redundantes e sobrepostas, é aplicada a técnica conhecida como Non-Max Suppression (NMS), que preserva apenas as detecções que apresentam maior confiança. Em comparação com modelos como o R-CNN, o YOLO demonstrou ser mais rápido e cometer menos erros quando confunde regiões de fundo com objectos. De acordo com o artigo original [3]:

> *"Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance. Fast R-CNN, a top detection method [14], mistakes background patches in an image for objects because it can't see the larger context. YOLO makes less than half the number of background errors compared to Fast R-CNN."*

No entanto, uma limitação importante do YOLO, especialmente nas suas primeiras versões, é a sua reduzida capacidade de generalização em cenários pouco comuns ou com objetos invulgares, um aspeto que será explorado mais adiante neste relatório.

2.5 Conclusões

Os métodos de subtração de fundo examinados, como o KNN e o MOG2, revelam-se soluções eficazes para segmentar objectos em movimento em ambientes controlados, apresentando bom desempenho em vídeos estáticos, com baixo custo computacional.

Por outro lado, os modelos de object detection como YOLO e Faster R-CNN recorrem a redes neuronais profundas para detetar e classificar objetos de forma direta. YOLO destaca-se pela sua velocidade e simplicidade, enquanto Faster R-CNN proporciona maior precisão através da utilização de propostas regionais aprendidas. Estas abordagens são mais robustas a variações complexas na cena e apresentam melhores resultados em contextos com múltiplas classes e condições visuais diversas.

Com todo este enquadramento técnico estabelecido, avançamos agora para a apresentação das ferramentas e frameworks utilizadas na implementação destas técnicas e modelos.

Capítulo

3

Tecnologias e Ferramentas Utilizadas

3.1 Introdução

A implementação prática foi complicada em alguns pontos, sendo o primeiro a integração de bibliotecas capazes de executar processamento de imagens eficazmente. Para tal, foi recorrido à biblioteca OpenCV, a qual oferece técnicas de Background Subtraction altamente otimizadas, e funcionalidades para processamento de vídeo, geração e aplicação de máscaras aos frames, e filtragem complexa com base em informação pixel a pixel.

Para a implementação dos modelos de deteção, foram usadas duas frameworks de código aberto bem conhecidas: Ultralytics, que oferece várias versões do modelo YOLO, incluindo versões pré-treinadas com o conjunto de dados COCO; e Detectron2, construída pela Meta AI, que suporta a implementação de modelos como o Faster R-CNN, apoiando-se nas tecnologias Compute Unified Device Architecture (CUDA) da NVIDIA através da biblioteca PyTorch.

Dado o volume significativo de dados necessários para a realização dos testes experimentais, tornou-se impraticável utilizar computadores pessoais sem capacidades gráficas avançadas. Assim, optou-se pela utilização dos serviços em nuvem disponibilizados pela Amazon Web Services (AWS), os quais oferecem instâncias otimizadas para tarefas intensivas em GPU, permitindo obter predições de forma mais eficiente e aproximando o projeto de uma perspetiva mais escalável.

As subsequente subsecções descrevem, em termos técnicos, o papel e o uso de cada uma dessas ferramentas no âmbito do presente trabalho.

3.2 OpenCV

A biblioteca OpenCV é a base essencial sobre a qual todos os scripts deste projeto foram construídos. A sua utilização mostrou-se inestimável no pré-processamento dos vídeos analisados, uma vez que estes não são utilizados diretamente como entradas nos modelos de deteção. Em vez disso, cada vídeo é transformado em uma série de quadros, que são então processados com base na intenção - usando métodos Background Subtraction ou diretamente com os modelos de deteção.

Dados o tamanho e a resolução dos quadros gerados, a decisão não foi armazena-los em forma permanente no disco. Neste sentido, os quadros são processados sequencialmente na memória RAM, o que permite uma maior utilização dos recursos disponíveis e evita a necessidade de armazenamento intermediário.

Em relação à implementação das metodologias de Background Subtraction, as duas técnicas explicadas foram utilizadas: K-NN e MOG2, ambas parte da atual versão do OpenCV.

3.3 Frameworks dos Modelos de Deteção

3.3.1 Ultralytics e YOLO

Para a implementação do modelo YOLO (You Only Look Once), foi utilizado o framework Ultralytics YOLO, uma implementação moderna e popular da família YOLO, desenvolvida na plataforma PyTorch. A implementação fornece várias versões atualizadas do modelo (YOLOv5 e YOLOv8, entre outros), que se destacam pela sua eficiência e por serem precisos na deteção de objetos em tempo real.

Esse framework também permite o uso otimizado de GPUs com CUDA, ou que fornece uma melhoria marcante no desempenho durante uma inferência.

3.3.2 Detectron2 e Faster R-CNN

Detectron2 é uma plataforma de detecção de objetos muito sofisticada, criada por Meta AI e construída sobre PyTorch. Neste projeto, foi aproveitado para a implementação do modelo Faster R-CNN, que é bem reconhecido por sua precisão superior na detecção de objetos, especialmente em ambientes desafiadores.

O uso do Detectron2 possibilitou alavancar a aceleração fornecida pela GPU (através da CUDA) para alcançar um ótimo desempenho e resultados estáveis na detecção.

3.4 Amazon Web Services

A plataforma Amazon Web Services (AWS) foi utilizada para viabilizar a implementação de modelos de deteção e processamento de vídeo em larga escala. Particularmente, a opção Elastic Compute Cloud (EC2) g4dn foi escolhida porque forneceu GPUs NVIDIA T4, garantindo uma capacidade de processamento gráfico dedicada e acelerada adequada para tarefas de aprendizagem profunda e inferência de modelos usando YOLO e Faster R-CNN.

Esses exemplos permitem a execução adequada dos frameworks PyTorch e Detectron2 com suporte ao CUDA, o que reduz significativamente o tempo de processamento e torna o aplicativo mais escalável.

A Amazon Simple Storage Service (S3), que oferece uma opção escalável, segura e de longo prazo para armazenar grandes quantidades de vídeos e resultados processados, foi usado para organização e gerenciamento de dados. A utilização do S3 permitiu uma rápida transferência de dados entre o EC2 e as instituições de armazenamento, o que otimizou o fluxo de processamento de dados e proporcionou a durabilidade dos resultados para análises futuras.

A integração desses serviços da AWS foi essencial para a implementação prática do projeto, fornecendo o poder de computação necessário sem as limitações das restrições de hardware local e a capacidade de dimensionar o sistema no futuro.

3.5 Conclusões

Como foi possível observar, para a aplicação desta investigação foi necessário o uso e combinação de várias bibliotecas e frameworks robustos, que permitiram a obtenção de resultados significativos. Podemos concluir que, graças ao esforço colaborativo da comunidade de código aberto, este trabalho foi possível de realizar, destacando-se também a importância dos serviços da Amazon Web Services (AWS). Estes serviços em nuvem permitem ultrapassar as limitações de hardware local, oferecendo a capacidade computacional necessária para projetos exigentes como este.

Finalmente, após apresentar as tecnologias utilizadas, procederemos à sua implementação detalhada e à análise dos resultados obtidos.

Capítulo

4

Experiências e resultados

4.1 Introdução

Aqui, será explicada a aplicação real das técnicas e metodologias empregadas neste projeto, juntamente com os experimentos realizados para o cálculo de seu desempenho. A criação dos principais scripts que constituem o pipeline experimental será explicada, incluindo a utilização das técnicas de subtração de fundo e a implementação dos modelos de detecção de objetos de última geração, ou seja, YOLO e Faster R-CNN.

Além disso, será seguida a descrição do pipeline, começando com o pré-processamento de vídeos e prosseguindo com a extração e tratamento de quadros, a aplicação de modelos de detecção e, finalmente, a recuperação e análise dos resultados. Esta metodologia permite ter uma visão da sequência lógica e operacional que sustenta o estudo, bem como os critérios adotados na comparação das técnicas em estudo.

4.2 Processamento de Vídeos

O processamento de vídeos constitui o ponto de partida de toda a pipeline desenvolvida neste trabalho. Por isso usaremos os métodos dados pela biblioteca OpenCV para a leitura de vídeos e a sua conversão em frames individuais. Estes frames são posteriormente utilizados conforme os requisitos de cada abordagem.

Importa destacar que, nesta implementação, os frames não são guardados em disco, uma vez que o seu armazenamento representa um custo significativo em termos de espaço e, no nosso caso, não traz benefícios relevantes. Em vez disso, cada frame é processado de forma sequencial e imediata na me-

mória (RAM), o que reduz a complexidade do fluxo e melhora o desempenho sem comprometer a integridade dos dados.

Após esta etapa de conversão, cada frame é encaminhado para a seguinte etapa do pipeline,neste caso detalharemos o seu processamento nas técnicas de background subtraction.

4.2.1 Aplicação de Máscaras e Operações Morfológicas

As operações morfológicas são técnicas fundamentais no processamento de imagens binárias, sendo amplamente utilizadas na refinação de máscaras geradas por algoritmos de segmentação, como os métodos de background subtraction utilizados neste trabalho. Estas técnicas baseiam-se na morfologia matemática e atuam sobre os píxeis da imagem tendo em conta a vizinhança de cada um.

No contexto desta investigação, destacam-se duas operações principais:

- Erosão: reduz as regiões brancas (foreground) da imagem, eliminando pequenos ruídos e desconectando áreas mal definidas.
- Dilatação: expande as regiões brancas, preenchendo pequenos espaços e ligando objetos próximos.

Ambas as operações utilizam um componente essencial denominado kernel (structuring element), que define a forma e o tamanho da vizinhança considerada durante o processamento. No caso de Kernels menores produzem alterações discretas e são ideais para ajustes finos, e ao contrário Kernels maiores provocam mudanças mais significativas, adequadas para limpezas ou ligações de zonas maiores.

No âmbito deste trabalho, estas operações foram fundamentais para limpar as máscaras geradas por MOG2 e KNN, assegurando que apenas as áreas de interesse relevantes fossem preservadas e enviadas aos modelos de deteção. Assim, contribuem significativamente para a robustez do pipeline e a fiabilidade dos resultados obtidos.

4.2.2 KNN

Para o método KNN, foram utilizados os parâmetros predefinidos, com exceção da ativação da opção detectShadows, a qual se mostrou relevante para melhorar a segmentação. Após a aplicação desta técnica, o resultado é submetido a uma série de operações morfológicas, erosão e dilatação, com o objetivo de limpar e refinar as áreas de interesse no foreground.

Estas operações requerem um kernel, que foi adotado da seguinte maneira:

- Erosão com kernel 2×2
- Dois iterações de dilatação com kernel 5×5
- Erosão final com kernel 2×2

Esta configuração foi escolhida com base em testes empíricos, demonstrando proporcionar os melhores resultados para os vídeos em questão, resultando em zonas de interesse mais limpas e bem definidas para serem utilizadas como entrada nos modelos de deteção.

```
import cv2
def process_video(video_path):
    cap = cv2.VideoCapture(video_path)
    kernel = np.ones((2, 2), dtype=np.uint8)
    kernel2 = np.ones((5, 5), dtype=np.uint8)

    knn = cv2.createBackgroundSubtractorKNN(
       detectShadows=True)

    if not cap.isOpened():
        print("Error: Cant open")
        exit()

    while True:
        ret, frame = cap.read()

        fg_mask = knn.apply(frame)
        mask = cv2.erode(fg_mask, kernel, iterations=1)
        mask = cv2.dilate(mask, kernel2, iterations=2)
        mask = cv2.erode(mask, kernel, iterations=1)

        foreground = cv2.bitwise_and(frame, frame, mask
           =mask)
        if not ret:
            break
    cap.release()
```

Excerto de Código 4.1: Aplicaçao das Operaçoes Morfologicas em KNN

4.2.3 MOG2

Para a técnica MOG2, além da ativação do parâmetro detectShadows, foram ajustados dois parâmetros adicionais: O varThreshold, definido com valor 16 para melhor sensibilidade à diferença entre o fundo e o primeiro plano, e o shadowThreshold, com o intuito de refinar a detecção de sombras.

Após a segmentação, aplica-se igualmente um conjunto de operações morfológicas com configuração distinta em relação ao método KNN. Neste caso, optou-se pelos seguintes kernels e iterações:

- Erosão com kernel 2×2
- Duas iterações de dilatação com kernel 4×4
- Erosão final com kernel 2×2

A escolha de um kernel de maior dimensão (4×4) na dilatação visa compensar pequenas descontinuidades e reforçar as zonas segmentadas, enquanto o kernel reduzido (2×2) nas erosões garante precisão na delimitação dos contornos. Esta configuração demonstrou-se eficaz para os vídeos analisados, equilibrando limpeza e continuidade nas áreas de interesse geradas pelas máscaras.

4.3 Integração da Pipeline

A pipeline desenvolvida nesta investigação foi concebida para avaliar os modelos de deteção em três formatos distintos, permitindo assim obter uma visão mais abrangente sobre o desempenho das tecnologias utilizadas, tanto de forma individual como em combinação.

A primeira abordagem consiste na utilização direta dos modelos de deteção YOLOv8 e Faster R-CNN, ambos pré-treinados com o dataset COCO. Estes modelos serão utilizados exclusivamente para a detecção da classe “pessoa”, descartando-se todas as outras classes retornadas. Qualquer deteção fora desta classe será considerada como não relevante ou incorrecta, caso tenha detetado corretamente a pessoa mas falhado na sua classificação.

A segunda abordagem incorpora o pré-processamento por técnicas de Background Subtraction (BGS), conforme descrito anteriormente. Nesta etapa, cada modelo de deteção será combinado com uma técnica de subtração de fundo, MOG2 ou KNN, originando um total de quatro variantes distintas. É importante salientar que os parâmetros das técnicas de BGS foram ajustados de forma independente, tendo como objetivo maximizar a qualidade da máscara gerada, sem considerar o modelo de deteção a ser utilizado. Desta forma,

evitamos enviesamentos e asseguramos uma comparação justa entre os métodos.

Por fim, a terceira variante propõe um uso alternativo das máscaras obtidas por BGS: em vez de restringirem a área de inferência, serão utilizadas para filtrar os resultados gerados pelos modelos na primeira abordagem. Isto significa que as deteções realizadas sem qualquer pré-processamento serão validadas ou descartadas com base na intersecção com a máscara de foreground, resultando em uma potencial “limpeza” dos resultados", hipótese que será devidamente analisada na secção de resultados e validação.

Com esta estrutura bem definida, passaremos a apresentar os principais trechos de código em cada uma das abordagens.

4.3.1 Aplicação dos Modelos de Deteção

4.3.1.1 Variante YOLO

A biblioteca Ultralytics conta até à data com onze versões de YOLO. Para esta investigação, optou-se pela utilização da versão 8 (YOLOv8), por se tratar de uma alternativa estável, amplamente compatível com outras bibliotecas, e com documentação consolidada.

Os modelos da série YOLOv8 estão disponíveis em diferentes tamanhos (por exemplo: n, s, m, l, x), que variam em complexidade, velocidade e precisão. Nesta implementação, foi escolhida a variante x (YOLOv8x), considerada a mais robusta em termos de desempenho, ainda que com maior custo computacional,algo viável graças ao uso de instâncias com GPU.

O processo de implementação decorrera da seguinte forma:

- O modelo YOLOv8x é carregado e instanciado como uma constante.
- Iteramos sobre os vídeos presentes numa diretoria específica.
- Cada vídeo é processado frame a frame, realizando-se a inferência em tempo real para cada imagem.
- As detecções da classe “pessoa” são filtradas e convertidas para coordenadas absolutas (pixel a pixel), já que o formato de saída do modelo é o tradicional formato YOLO (coordenadas normalizadas).
- Os resultados são guardados em ficheiros .txt, num diretório externo, com o mesmo nome do vídeo de origem.

Esta estrutura de implementação permite uma organização clara dos resultados e facilita etapas posteriores de validação e comparação.

```
def Yolo_predict(model, video_path, output_folder,
   SAVE_IMAGES=False):
    cap = cv2.VideoCapture(str(video_path))

    if not cap.isOpened():
        print(f"Error: Cant open")
        return

    frame_count = 0
    video_stem = Path(video_path).stem

    output_video_dir = output_folder / video_stem
    output_video_dir.mkdir(parents=True, exist_ok=True)

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        results = model(frame,conf=0.2)
        boxes = results[0].boxes
        person_boxes = boxes[boxes.cls == 0]
        person_boxes = person_boxes.xyxy.cpu().numpy()

        txt_path = output_video_dir / f"frame_{
           frame_count:06d}.txt"
        with open(txt_path, "w") as f:
            for box in person_boxes:
                x1, y1, x2, y2 = box[:4]
                f.write(f"{x1:.2f} {y1:.2f} {x2:.2f} {
                   y2:.2f}\n")

        if SAVE_IMAGES:
            img_path = output_video_dir / f"frame_{
               frame_count:06d}.jpg"
            img_with_boxes = results[0].plot()
            cv2.imwrite(str(img_path), img_with_boxes)

        frame_count += 1

    cap.release()
```

Excerto de Código 4.2: Funçao de inferencia de YOLO

4.3.1.2 Variante Faster R-CNN

A biblioteca Detectron2, criada pelo Meta AI, é uma das opções mais sólidas e completas disponíveis para realizar tarefas de detecção de objetos baseadas em arquiteturas da família R-CNN. No âmbito desta pesquisa, foi utilizada a arquitetura Faster R-CNN R-50 FPN, ou seja, uma de suas variantes mais modernas e eficientes, que junto com um alto desempenho apresenta baixo custo computacional.

Para o projeto em questão, foram estabelecidos vários parâmetros de configuração significativos.

Primeiramente, a aplicação de pesos previamente treinados utilizando o conjunto de dados COCO assegura uma fundação sólida para a detecção de indivíduos; em segundo lugar, a determinação de um limiar de confiança `score_threshold` de 0,2 possibilita a filtragem de previsões com baixa probabilidade; por último, o estabelecimento de um limiar para a supressão de não-máximos `nms_threshold` de 0,3 visa reduzir a sobreposição das bounding boxes.

Com estes parâmetros ajustados, a inferência segue uma lógica semelhante à aplicada com o modelo YOLO:

- Cada vídeo é analisado quadro a quadro.
- Cada imagem é passada através do modelo R-CNN mais rápido para obter as previsões.
- As detecções são traduzidas para coordenadas absolutas e filtradas para deixar apenas a classe "pessoa".
- Os resultados finais são armazenados em arquivos . txt, que são nomeados após o vídeo original.

```
def load_model():
    cfg = get_cfg()
    cfg.merge_from_file(model_zoo.get_config_file("COCO
       -Detection/faster_rcnn_R_50_FPN_1x.yaml"))
    cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.2
    cfg.MODEL.ROI_HEADS.NMS_THRESH_TEST = 0.3
    cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("
       COCO-Detection/faster_rcnn_R_50_FPN_1x.yaml")
    cfg.MODEL.DEVICE = "cuda" if torch.cuda.
       is_available() else "cpu"
    return DefaultPredictor(cfg)
```

Excerto de Código 4.3: Ajuste dos parametros em Faster R-CNN

4.3.2 Fluxo: Background Subtraction e Modelos de Deteção

Como foi descrito anteriormente, a aplicação das técnicas de Background Subtraction será padronizada para ambos os modelos de deteção, YOLO e Faster R-CNN. Isso significa que a estrutura da implementação será essencialmente a mesma para ambas as abordagens, variando apenas nas configurações individuais dos modelos, conforme já abordado nos subcapítulos anteriores. Assim, descreveremos aqui o padrão geral que será utilizado nas quatro versões geradas por estas combinações.

O primeiro passo, uma vez definida a função do modelo de inferência a ser utilizado, será aplicar as técnicas de pré-processamento descritas anteriormente. Isso inclui a leitura do vídeo, a conversão para frames individuais e, neste caso, a aplicação direta das técnicas de Background Subtraction, como MOG2 ou KNN, logo após a extração dos frames. A máscara binária gerada por estas técnicas será então refinada por meio de operações morfológicas, de forma a realçar as regiões de interesse.

Com a imagem devidamente pré-processada, o próximo passo será aplicar a máscara sobre o frame e enviar o resultado ao modelo de deteção para realizar a inferência. Os dados resultantes serão então armazenados em uma pasta definida, mantendo um formato uniforme para posterior avaliação.

```
def process_video(video_path, model, output_base_path,
    save_images=False):
    cap = cv2.VideoCapture(str(video_path))
    if not cap.isOpened():
        print(f"Cant open: {video_path}")
        return

    video_name = video_path.stem
    output_txt_dir = output_base_path / video_name / "
        txt"
    output_img_dir = output_base_path / video_name / "
        frames" if save_images else None

    kernel = np.ones((1, 1), dtype=np.uint8)
    kernel2 = np.ones((4, 4), dtype=np.uint8)

    mog2 = cv2.createBackgroundSubtractorMOG2(
        varThreshold=16, detectShadows=True)
    mog2.setShadowThreshold(0.5)

    while True:
        success, frame = cap.read()
```

```
        if not success:
            break

        fg_mask = mog2.apply(frame)
        mask = cv2.erode(fg_mask, kernel, iterations=1)
        mask = cv2.dilate(mask, kernel2, iterations=2)
        mask = cv2.erode(mask, kernel2, iterations=1)

        foreground = cv2.bitwise_and(frame, frame, mask
            =mask)

        yolo_predict(model, foreground, frame_id,
            output_txt_dir, save_images, output_img_dir,
            video_name)


    cap.release()
```

Excerto de Código 4.4: Exemplo de Implementaçao do Background Subtraction + Modelos de Deteçao

4.3.3 Filtragem das Previsões com Base nas Máscaras

Nesta terceira abordagem, ao contrário das primeiras, a máscara gerada pelas técnicas de Background Subtraction não será utilizada como entrada direta para o modelo de deteção; em vez disso, ela será usada como filtro que ocorre após as inferências. Isso é feito para remover caixas delimitadoras que não correspondem às regiões de interesse detetadas na máscara para que, posteriormente, as deteções sejam mais precisas.

A normalização prévia dos formatos de saída dos modelos YOLO e Faster R-CNN, bem como a conversão de todas as caixas delimitadoras para coordenadas absolutas, permite que o processo de filtragem seja executado independentemente do modelo de origem.

Portanto, a implementação é uma função que foi projetada para ler cada arquivo de texto produzido por o modelo, analisar as coordenadas das caixas delimitadoras e aplicá-las sobre a máscara binária gerada para cada frame. Para cada caixa, uma região de interesse (ROI)Região de interesse (ROI) é estabelecida na máscara, esta região será analisada com base em dois fatores:

- A dimensão do ROI deve ser maior que zero
- A proporção de pixels positivos (valor 255) para o total da região deve ser igual ou maior que um valor limiar, que neste caso é 10

Uma vez que as caixas que satisfazem ambos os critérios são mantidas em suas coordenadas originais. As outras são removidas. Após a conclusão do processo, a função retorna uma lista de caixas filtradas, que serão salvas como um novo arquivo de texto, garantindo a conformidade com o formato usado nas abordagens anteriores.

Este processo permite aplicar as vantagens das técnicas de segmentação por fundo como mecanismo de validação adicional, promovendo maior robustez na seleção final de detecções relevantes.

```
def filter_boxes_with_mask(boxes, mask, threshold=0.1):
    filtered = []
    for box in boxes:
        x1, y1, x2, y2 = map(int, box)
        x1, y1 = max(0, x1), max(0, y1)
        x2, y2 = min(mask.shape[1] - 1, x2), min(mask.
           shape[0] - 1, y2)
        roi = mask[y1:y2, x1:x2]
        if roi.size == 0:
            continue
        non_zero = cv2.countNonZero(roi)
        if roi.shape[0] * roi.shape[1] > 0 and non_zero
            / (roi.shape[0] * roi.shape[1]) >=
           threshold:
            filtered.append((x1, y1, x2, y2))
    return filtered
```

Excerto de Código 4.5: Funçao que Filtra as Bounding Box

4.4 Execução na Nuvem

Com os scripts principais definidos e testados localmente, tornou-se evidente a necessidade de um ambiente computacional mais robusto, especialmente devido à elevada exigência de processamento dos modelos de deteção, como YOLO e Faster R-CNN. Assim, optou-se pela utilização de serviços em nuvem da AWS, nomeadamente EC2 e S3.

O serviço Elastic Compute Cloud (EC2) permitiu o acesso a instâncias equipadas com GPUs NVIDIA T4, otimizadas para tarefas de aprendizagem profunda. Para maximizar a compatibilidade e o desempenho, foi utilizada a imagem oficial *Deep Learning AMI (Ubuntu)*, já preparada com os principais pacotes e bibliotecas necessários. O acesso à instância foi realizado via SSH, possibilitando a configuração do ambiente, instalação de dependências, cri-

ação de ambientes virtuais e transferência dos scripts de processamento para execução direta na nuvem.

Entretanto, surgiu uma nova preocupação: a transferência direta de grandes volumes de dados (como vídeos de alta resolução) entre a máquina local e a instância Elastic Compute Cloud (EC2) pode implicar custos adicionais e atrasos. Para contornar essa limitação, recorreu-se ao serviço Simple Storage Service (S3), que funciona como um repositório escalável e seguro para armazenamento de dados.

A integração entre EC2 e S3 garantiu transferências internas rápidas e eficientes, permitindo que os vídeos fossem carregados previamente no S3 e posteriormente acedidos diretamente pelas instâncias EC2 durante o processamento. Este fluxo reduziu significativamente o custo e tempo de transferência, além de manter uma estrutura organizada e centralizada dos dados utilizados nos testes.

```
import boto3
import os

BUCKET_NAME = 'darvideosbucket'
S3_PREFIX = 'raw_videos/'
LOCAL_DOWNLOAD_PATH = './videos/'

os.makedirs(LOCAL_DOWNLOAD_PATH, exist_ok=True)

s3 = boto3.client('s3')

def download_first_40_videos():
    paginator = s3.get_paginator('list_objects_v2')
    page_iterator = paginator.paginate(Bucket=
       BUCKET_NAME, Prefix=S3_PREFIX)

    count = 0
    for page in page_iterator:
        if 'Contents' not in page:
            print("No files.")
            return

        for obj in page['Contents']:
            key = obj['Key']
            base_name = os.path.splitext(os.path.
               basename(key))[0]

            if key.lower().endswith(('.MP4')):
```

```
                local_filename = os.path.join(
                    LOCAL_DOWNLOAD_PATH, os.path.
                    basename(key))
                print(f"Download {key} ...")
                s3.download_file(BUCKET_NAME, key,
                    local_filename)
                count += 1
                if count >= 80:
                    print("Download 80 videos.")
                    return
```

Excerto de Código 4.6: Exemplo do uso de S3 nas instancias EC2

4.5 Métricas e Validação

Uma vez que os dados foram processados adequadamente, foi possível reunir um total de 10 diretórios com os arquivos de arquivos de texto que correspondiam à deteção de cada vídeo, classificados de acordo com as variantes do pipeline. Em seguida, foi realizada a comparação dessas deteções com os dados de referência (Ground Truth) que haviam sido anotados anteriormente. Para manter a consistência entre os formatos, todas as anotações foram ajustadas às coordenadas absolutas, como tinha sido feito com as saídas dos modelos.

Dada a natureza dos dados, os bounding box, bem como o procedimento de avaliação do modelo, seguiu-se uma abordagem tradicional baseada em métricas específicas do domínio no campo da visão computacional.

A etapa inicial de avaliação envolveu o uso da métrica Intersection over Union (IoU), permitindo quantificar a extensão da sobreposição entre duas caixas delimitadoras. O IoU é dado pela razão da área de intersecção para a área de união de duas caixas. Esta medida é considerada muito eficaz para quantificar a concordância entre as previsões do modelo e as anotações reais.

Tendo uma função implementada de IoU, pode-se construir uma matriz de correspondência entre as caixas detetadas e os correspondentes ao Ground Truth. Tal matriz serviu então para aplicar um algoritmo optimal attribution, baseando-se no algoritmo húngaro (implementado em *scipy.optimize.linear-sum-assignment*), atribuindo cada caixa detetada à caixa anotada com maior probabilidade, com um limiar mínimo de IoU.

A função utilizada para esta correspondência foi a seguinte:

```
    def IoU(box1, box2):
    x1_min, y1_min, x1_max, y1_max = box1
    x2_min, y2_min, x2_max, y2_max = box2
```

```
    inter_x_min = max(x1_min, x2_min)
    inter_y_min = max(y1_min, y2_min)
    inter_x_max = min(x1_max, x2_max)
    inter_y_max = min(y1_max, y2_max)

    inter_width = max(0, inter_x_max - inter_x_min)
    inter_height = max(0, inter_y_max - inter_y_min)
    inter_area = inter_width * inter_height

    area1 = (x1_max - x1_min) * (y1_max - y1_min)
    area2 = (x2_max - x2_min) * (y2_max - y2_min)

    union_area = area1 + area2 - inter_area

    if union_area == 0:
        return 0

    iou = inter_area / union_area
    return iou

def compute_iou_matrix(boxes_A, boxes_B):
    iou_matrix = np.zeros((len(boxes_A), len(boxes_B)))

    for i, box_A in enumerate(boxes_A):
        for j, box_B in enumerate(boxes_B):
            iou_matrix[i, j] = IoU(box_A, box_B)

    return iou_matrix
```

Excerto de Código 4.7: Implementaçao do IoU

```
def match_boxes_optimal(iou_matrix, iou_threshold=0.5):
    if iou_matrix.size == 0:
        return [], list(range(iou_matrix.shape[0])),
           list(range(iou_matrix.shape[1]))

    cost_matrix = 1 - iou_matrix
    row_ind, col_ind = linear_sum_assignment(
       cost_matrix)

    matched_indices = []
    unmatched_A = list(range(iou_matrix.shape[0]))
```

```
    unmatched_B = list(range(iou_matrix.shape[1]))

    for i, j in zip(row_ind, col_ind):
        if iou_matrix[i, j] >= iou_threshold:
            matched_indices.append((i, j))
            unmatched_A.remove(i)
            unmatched_B.remove(j)

    return matched_indices, unmatched_A, unmatched_B
```

Excerto de Código 4.8: Implementaçao do Max Optimal Attribution

A partir dos resultados desta função, foi possível calcular as métricas clássicas de desempenho:

- True Positive (TP): Os resultados que são corretamente previstos como positivos
- False Positive (FP): Os resultados incorretamente previstos como positivos
- False Negative (FN): Os resultados incorretamente previstos como negativos
- Precision: Razão entre as detecções corretas e o total de detecções feitas.
- Recall: Razão entre as detecções corretas e o total de objetos reais.
- F1-score: Média harmónica entre precisão e recall, fornecendo um valor único de desempenho equilibrado.

Estas métricas permitiram quantificar, de forma padronizada, o desempenho relativo de cada variante da pipeline e extrair conclusões significativas sobre a eficácia da combinação entre modelos de deteção e técnicas de subtração de fundo.

4.6 Resultados e Discussão

Com a pipeline completamente implementada e executada, procedeu-se à consolidação de todos os dados num conjunto de ficheiros CSV, de forma a facilitar a sua análise e visualização. No total, o volume de dados considerado nesta fase é composto por 82 vídeos, recolhidos por drones em diferentes contextos operacionais.

Estes vídeos foram agrupados em três categorias distintas, consoante a distância da câmara em relação aos objetos filmados:

- 31 vídeos com capturas próximas.
- 24 vídeos com capturas intermédias.
- 27 vídeos com capturas distantes.

Esta divisão visa permitir duas abordagens complementares à análise: uma visão global dos resultados combinados e uma visão segmentada, centrada nas especificidades de cada grupo. Esta estratégia proporciona uma base sólida para a comparação dos diferentes modelos e técnicas, permitindo uma compreensão mais aprofundada do seu comportamento em contextos variados.

4.6.1 Visão Geral

Começando com Yolo, a Tabela 4.1 apresenta os resultados gerais obtidos nas suas diferentes variantes, considerando todas as distâncias.

Modelo	**TP**	**FP**	**FN**	**Precisão**	**Recall**	**F1-Score**
YOLO	**6955**	20349	**2802**	**0.59**	**0.45**	**0.31**
YOLO + KNN	4846	22433	1507	0.20	0.42	0.23
YOLO + MOG2	4517	**22788**	1430	0.19	0.39	0.21
Filtrado por KNN	6672	20198	2735	0.26	0.44	0.29
Filtrado por MOG2	6672	20198	2734	0.26	0.44	0.28

Tabela 4.1: Resultados gerais do YOLO

Na tabela, foram destacados os maiores valores obtidos em cada métrica, com o objetivo de facilitar a visualização e interpretação dos resultados.

De seguida, serão apresentados os dados obtidos com o modelo Faster R-CNN, apresentados na Tabela 4.2.

Modelo	**TP**	**FP**	**FN**	**Precisão**	**Recall**	**F1-Score**
Faster R-CNN	**7371**	19748	4151	**0.28**	**0.43**	**0.31**
Faster R-CNN + KNN	3066	24214	1243	0.13	0.36	0.15
Faster R-CNN + MOG2	2965	**24315**	1236	0.12	0.36	0.16
Filtrado por KNN	7345	19774	**4177**	**0.28**	**0.43**	**0.31**
Filtrado por MOG2	7344	19775	4162	**0.28**	**0.43**	**0.31**

Tabela 4.2: Resultados gerais do Faster R-CNN

Com estes dados, é possível obter uma primeira impressão do desempenho geral dos modelos. No entanto, para ampliar a nossa compreensão dos resultados, procederemos à apresentação dos dados de forma separada, conforme referido anteriormente.

4.6.2 Casos Particulares

Aqui será feito um desdobramento mais específico dos resultados, guiando-nos pelas diferentes categorias de captação dos vídeos por drone. A métrica utilizada para classificar uma tomada como próxima, intermédia ou distante baseou-se exclusivamente na altura do drone, a qual pode ser identificada no nome dos vídeos, organizados no seguinte formato: "006_006_04_04_2024_11_15_SCS-SRTMUNIN_15_10_60".

Neste exemplo, os três últimos números representam. A altura do drone em metros em relação ao solo (15); A distância horizontal desde o ponto onde se encontram as pessoas (10); E o ângulo de inclinação da câmara em graus (60).

Com base neste critério, estabeleceu-se a seguinte classificação:

- Tomas próximas: altura inferior a 25 metros.
- Tomas intermédias: altura entre 25 e 40 metros.
- Tomas distantes: altura superior a 40 metros.

Com esta explicação estabelecida, passamos agora à apresentação e análise dos resultados para cada uma destas categorias.

4.6.2.1 Tomas Próximas

Modelo	Variante	TP	FP	FN	Precisão	Recall	F1
YOLO	YOLO	13745	9462	4777	0.58	0.67	0.60
	Modelo + MOG2	7648	17513	2472	0.33	0.40	0.35
	Modelo + KNN	11092	12115	3172	0.47	0.67	0.51
	Filtrado por MOG2	12793	9330	4490	0.52	0.62	0.55
	Filtrado por KNN	12794	9330	4492	0.52	0.62	0.55
Faster R-CNN	Faster R-CNN	12321	10886	5658	0.51	0.59	0.53
	Modelo + MOG2	7128	16079	2407	0.30	0.61	0.37
	Modelo + KNN	5829	19333	2182	0.25	0.38	0.28
	Filtrado por MOG2	12311	10896	5658	0.51	0.59	0.53
	Filtrado por KNN	12311	10896	5668	0.51	0.59	0.53

Tabela 4.3: Resultados dos modelos YOLO e Faster R-CNN com variantes baseadas em background subtraction (Tomas Distantes)

4.6.2.2 Tomas Intermédias

Modelo	Variante	TP	FP	FN	Precisão	Recall	F1
YOLO	YOLO	5799	23546	2345	0.19	0.54	0.26
	Modelo + MOG2	5837	20246	1857	0.24	0.57	0.29
	Modelo + KNN	2095	27250	702	0.07	0.44	0.11
	Filtrado por MOG2	5778	23567	2364	0.19	0.53	0.26
	Filtrado por KNN	5778	23567	2366	0.20	0.54	0.26
Faster R-CNN	Faster R-CNN	6768	22027	3759	0.23	0.44	0.29
	Modelo + MOG2	897	28448	750	0.03	0.36	0.05
	Modelo + KNN	2988	23095	1110	0.11	0.51	0.16
	Filtrado por MOG2	6736	22059	3782	0.23	0.44	0.29
	Filtrado por KNN	6737	22058	3791	0.23	0.44	0.29

Tabela 4.4: Resultados dos modelos YOLO e Faster R-CNN com variantes baseadas em background subtraction (Tomas Distantes)

4.6.2.3 Tomas Distantes

Modelo	Variante	TP	FP	FN	Precisão	Recall	F1
YOLO	YOLO	453	29315	1074	0.014	0.155	0.023
	Modelo + MOG2	269	29772	213	0.009	0.264	0.017
	Modelo + KNN	123	29645	311	0.004	0.111	0.007
	Filtrado por MOG2	447	29321	1077	0.013	0.155	0.023
	Filtrado por KNN	447	29321	1080	0.013	0.155	0.024
Faster R-CNN	Faster R-CNN	2231	27537	2920	0.064	0.244	0.097
	Modelo + MOG2	26	29742	323	0.001	0.095	0.001
	Modelo + KNN	217	29824	463	0.007	0.204	0.013
	Filtrado por MOG2	2192	27577	2936	0.063	0.241	0.096
	Filtrado por KNN	2192	27576	2959	0.063	0.241	0.096

Tabela 4.5: Resultados dos modelos YOLO e Faster R-CNN com variantes baseadas em background subtraction (Tomas Distantes)

4.7 Análise dos Resultados

Com todos os dados recolhidos dos diferentes modelos e as suas variantes, é agora possível proceder a uma análise detalhada dos seus desempenhos. Nesta secção, realizaremos uma comparação quantitativa entre os resultados obtidos, identificando os melhores desempenhos por métrica, e, posteriormente, apresentaremos uma interpretação qualitativa que procura justificar o comportamento observado em cada abordagem. O objetivo é chegar a conclusões claras e fundamentadas sobre a eficácia de cada método utilizado.

4.7.1 Melhores Desempenhos

4.7.1.1 Geral

Começando pelos modelos na sua forma base, sem qualquer tipo de modificação ou integração com background subtraction, é possível observar que o YOLOv8 apresenta uma média geral de desempenho superior ao Faster R-CNN. Em particular, YOLO regista valores mais elevados nas métricas de precision, recall e um valor idêntico no F1-Score 4.6 .

Modelo	TP	FP	FN	Precisão	Recall	F1-Score
YOLOv8 + KNN	**4846**	22433	1507	**0.20**	**0.42**	**0.23**
YOLOv8x + MOG2	4517	**22788**	1430	0.19	0.39	0.21
Faster R-CNN + KNN	3066	24214	1243	0.13	0.36	0.15
Faster R-CNN + MOG2	2965	24315	1236	0.12	0.36	0.16

Tabela 4.7: Comparativa entre YOLO e Faster R-CNN, com as tecnicas de Background Substraction

Modelo	TP	FP	FN	Precisão	Recall	F1-Score
YOLO	6955	20349	2802	0.59	0.45	0.31
Faster R-CNN	7371	19748	4151	0.28	0.43	0.31

Tabela 4.6: Comparativa entre YOLO e Faster R-CNN

No entanto, importa destacar que YOLO tende a gerar mais falsos positivos, ou seja, deteta mais pessoas do que realmente existem nas imagens. Por outro lado, o Faster R-CNN mostra um melhor desempenho quando se considera apenas os verdadeiros positivos, portanto, deteta com maior exatidão as pessoas que consegue identificar corretamente.

Isto sugere que o ponto fraco do Faster R-CNN, neste cenário, reside na sua menor capacidade de cobertura do panorama completo, uma vez que tende a omitir algumas pessoas presentes na imagem. Em contrapartida, quando efetivamente deteta um objeto, a sua classificação tende a ser mais precisa.

4.7.1.2 Modelo & Background Subtraction

Prosseguindo com a análise, os modelos combinados com técnicas de background subtraction têm um claro decréscimo em praticamente todas as métricas avaliadas. Isto demonstra que os modelos não conseguem adaptar-se de forma eficaz aos frames modificados, apresentando um aumento significativo no número de falsos positivos 4.7.

Este comportamento indica que os modelos tendem a confundir mais facilmente elementos no cenário, resultando em deteções ou classificações incorretas que penalizam o seu desempenho global. Consequentemente, também se verifica uma redução no número de verdadeiros positivos.

Ainda assim, é interessante notar que ocorre uma ligeira redução nos falsos negativos, o que significa que, embora os modelos confundam mais, também deixam escapar menos alvos potenciais.

4.7.1.3 Deteções Filtradas

Por último, a filtragem das deteções dos modelos através das técnicas de background subtraction demonstrou diminuir a precisão dos mesmos, assim como reduzir ligeiramente os restantes valores. Este resultado evidencia uma situação particular: ao trabalhar com as próprias deteções do modelo, seria de esperar uma melhoria no desempenho, sobretudo na redução de falsos positivos e falsos negativos. No entanto, estes dados sugerem limitações das técnicas de background subtraction quando aplicadas ao nosso dataset, questão que será aprofundada na subsecção de interpretação.

De qualquer forma, neste cenário observamos, pela primeira vez, que o Faster R-CNN supera o YOLO em métricas como True Positive e False Positive. Embora se verifique um ligeiro aumento nos falsos negativos, o que reduz o recall, em termos gerais, o Faster R-CNN obtém uma maior precisão e um F1-Score superior 4.8.

Modelo	**TP**	**FP**	**FN**	**Precisão**	**Recall**	**F1-Score**
YOLOv8 & KNN	6672	20198	2735	0.26	**0.44**	0.29
YOLOv8x & MOG2	6672	**20198**	2734	0.26	**0.44**	0.28
Faster R-CNN & KNN	**7345**	19774	**4177**	**0.28**	0.43	**0.31**
Faster R-CNN & MOG2	7344	19775	4162	**0.28**	0.43	**0.31**

Tabela 4.8: Comparativa entre as deteções filtradas pelas tecnicas de Background Substraction

4.7.2 Influência dos planos de câmara

Agora que temos uma noção mais clara de qual modelo apresenta melhor desempenho em termos gerais, podemos concentrar-nos nos casos específicos que influenciam estes resultados.

O primeiro foco recai sobre a distância das tomadas e a sua influência no desempenho dos modelos. Observando os dados, verifica-se que o desempenho de ambos os modelos em planos próximos é, de forma geral, satisfatório. Importa referir que, nestes casos, as detecções filtradas pelas técnicas de background subtraction não apresentam um decréscimo tão acentuado como se verificou anteriormente, embora se mantenham abaixo do nível esperado. Por outro lado, os modelos que combinam a entrada com estas técnicas registam melhorias significativas em relação aos resultados médios, destacando-se uma clara redução dos falsos negativos, o que se traduz em valores de recall que igualam, ou até superam (no caso do Faster R-CNN), os dos modelos originais 4.6.2.1.

Avançando para as detecções intermédias, observa-se que, em alguns casos, como o YOLO combinado com MOG2, os resultados superam o modelo base em todas as métricas. Contudo, esta tendência não se verifica para o Faster R-CNN, que mantém um desempenho inferior em comparação com a sua versão sem combinações 4.4.

Por fim, nas cenas distantes, confirma-se a dificuldade acrescida para ambos os modelos, apresentando valores mais baixos, na média já observada para este cenário 4.5.

Como segundo foco, devemos referir-nos a um ponto essencial. Até ao momento, focámo-nos sobretudo na distância, mas existe outro fator que influencia significativamente os resultados, o ângulo da câmara. A análise mostra que ângulos mais acentuados, como 90 graus, afetam de forma notória o desempenho de ambos os modelos 4.2, reduzindo a precisão global. Apesar de estes casos serem minoritários no conjunto de dados, é importante reconhecê-los como uma limitação adicional.

Modelo	**Angulo**	**TP**	**FP**	**FN**	**Precisão**	**Recall**	**F1-Score**
YOLOv8	30	13144	14696	3802	0.50	0.65	0.54
YOLOv8	60	7768	23066	2920	0.28	0.57	0.33
YOLOv8	90	2281	21469	2132	0.12	0.25	0.14
Faster RCNN	30	14730	13111	5761	0.55	0.66	0.59
Faster RCNN	60	8814	21513	4171	0.31	0.58	0.37
Faster RCNN	90	1085	22664	2846	0.07	0.16	0.08

Tabela 4.9: Comparação tendo em conta o ângulo da câmara

Além disso, é possível verificar que o Faster RCNN apresenta um desempenho superior ao YOLO quando analisamos isoladamente o fator ângulo da câmara. Esta constatação permite uma nova leitura dos resultados anteriormente apresentados, dado que o YOLO revela melhor desempenho em ângulos de 90 graus, enquanto o Faster RCNN se destaca nos ângulos de 30 e 60 graus. Assim, torna-se evidente que a distribuição dos vídeos segundo os ângulos de captação tem influência direta nos resultados globais. Apesar disso, uma vez que o conjunto de dados procura refletir a complexidade e variabilidade de cenários reais, não se podem desconsiderar nem os resultados globais, nem esta leitura segmentada, sendo ambos complementares para uma interpretação mais completa do comportamento dos modelos.

4.7.3 Interpretação dos Resultados

Com base em todos os resultados apresentados, cabe agora tentar interpretar as razões que os justificam. Tendo em conta que a componente quantitativa já foi devidamente abordada, propomos aqui uma interpretação própria para explicar o comportamento observado.

O primeiro ponto a salientar é o desempenho inferior da combinação de Background Subtraction com os modelos de deteção. À partida, poderia supor-se que, ao limitar as regiões de interesse, o modelo teria menos informação irrelevante para processar, aumentando assim a probabilidade de uma deteção correta. No entanto, os nossos resultados não confirmam esta hipótese, o que nos leva a duas possíveis interpretações.

A primeira prende-se com a natureza do treino dos modelos utilizados. Tanto o YOLO, da Ultralytics, como o Faster RCNN da Meta AI, foram pré-treinados com o dataset COCO, um conjunto de dados composto por imagens em condições limpas e ricas em informação de contexto, precisamente o tipo de informação que no nosso caso, é removido da imagem através da subtração de fundo. Esta redução drástica de contexto parece ter impacto direto na eficácia dos modelos. Podemos ilustrar este fenómeno com as figuras 4.24.1: numa dada frame, o YOLO é capaz de detetar praticamente todas as pessoas de forma correta; contudo, ao aplicar a Background Subtraction, surgem artefactos (pequenas regiões negras) sobre partes do corpo das pessoas, o que, mesmo sendo mínimas, acaba por ser suficiente para o modelo descartar ou falhar uma deteção.

Para além deste fator, existe ainda uma segunda questão relacionada com os parâmetros de configuração das técnicas de subtração de fundo. Sendo os vídeos analisados captados a diferentes distâncias e ângulos, o ajuste fino dos parâmetros a cada caso seria crucial para maximizar a qualidade das máscaras geradas. No entanto, este ajuste individual não foi realizado no âmbito deste projeto. Optou-se por definir um conjunto de parâmetros médios, com o objetivo de englobar todos os vídeos e cenários de forma genérica. Naturalmente, esta abordagem faz com que, em certos casos, os parâmetros sejam demasiado permissivos, enquanto noutros poderão ser demasiado restritivos. Este compromisso entre generalização e precisão influencia diretamente os resultados obtidos.

Figura 4.1: Deteção com YOLO sem subtração de fundo

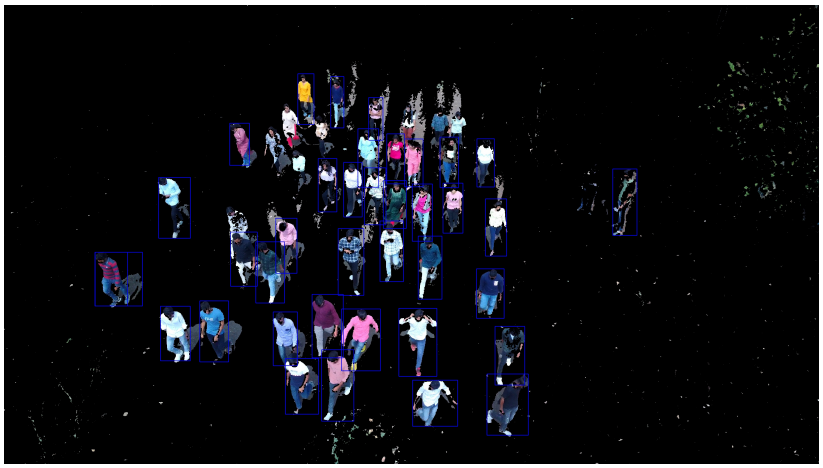

Figura 4.2: Deteção com YOLO com subtração de fundo

Por último, e considerando todos os pontos referidos, é possível concluir que estas limitações acabaram por resultar numa filtragem ineficaz das deteções dos modelos através da Background Subtraction. Ao filtrar as predições, é importante ter em conta que os parâmetros do método de BGS podem não detetar corretamente pessoas que permanecem imóveis durante um certo número de frames, ou que, por limitações da técnica, acabam por não ser segmentadas de forma adequada. Isto significa que, sempre que uma pessoa não é corretamente segmentada, a respetiva deteção do modelo é eliminada, conduzindo a uma redução do número de objetos detetados.

Ainda que estas perdas possam parecer mínimas envolvendo uma ou duas pessoas por sequência, o seu efeito acumulado ao longo do tempo torna-

se relevante, refletindo-se diretamente nos resultados obtidos. Esta é também uma das razões pelas quais os resultados finais não diferem drasticamente dos obtidos apenas com o modelo base: o erro introduzido pelo Background Subtraction afeta apenas uma pequena percentagem das deteções, mantendo-se o restante desempenho praticamente inalterado.

4.8 Conclusões

Com a implantação deste pipeline robusto, foi possível validar de forma consistente diferentes variantes dos modelos de detecção, isoladas ou combinadas com técnicas de Background Subtraction, viabilizando todo o processamento intensivo através da infraestrutura da AWS. A padronização dos resultados em coordenadas absolutas permitiu uma análise comparativa sólida, explorando o desempenho de cada abordagem em cenários diversos.

Os resultados obtidos evidenciam que tanto o YOLO quanto o Faster R-CNN são extremamente sensíveis a alterações no contexto, variações no ângulo de captação, na distância dos alvos ou mesmo a aplicação de máscaras influenciam de forma significativa a performance. Isso demonstra a importância de uma configuração precisa dos modelos e da seleção criteriosa das técnicas de subtração de fundo, pois ajustes aparentemente mínimos, como a escolha do kernel ou dos parâmetros, podem impactar de forma substancial a precisão de deteção, para melhor ou para pior.

Assim, as experiências conduzidas e as análises apresentadas consolidam uma base de evidências clara sobre os limites e potencialidades de cada abordagem, servindo de suporte para as conclusões finais.

Capítulo

5

Conclusões e Trabalho Futuro

5.1 Conclusões Principais

O objetivo deste projeto foi dividido em três partes principais, que foram analisadas ao longo deste relatório, mas que, reunidas agora, nos permitem responder a cada uma de forma mais direta.

Em primeiro lugar, verificou-se que os modelos atuais, YOLO e Faster R-CNN, apresentam um desempenho aceitável na deteção de imagens captadas por drones. Observou-se, no entanto, que alterações em variáveis como a altura ou o ângulo da câmara podem reduzir significativamente a precisão. Isto significa que estes modelos falham sempre nestes cenários? A resposta é não. É importante recordar, conforme analisado anteriormente, que estes modelos foram treinados com datasets específicos (como o COCO), que não contemplam a totalidade dos contextos que testámos. Assim, não é possível generalizar totalmente o seu desempenho para casos fora do domínio de treino. Se, em contrapartida, separássemos os nossos dados e re-treinássemos cada modelo com exemplos ajustados às nossas condições, seria plausível esperar melhorias relevantes. Mesmo assim, verificou-se que, de forma geral, os modelos mantêm uma capacidade de generalização robusta, o que justifica o seu destaque atual na investigação e aplicação prática.

Em segundo lugar, respondendo à questão central deste trabalho. Vale a pena utilizar técnicas de background subtraction com modelos SOTA? Podemos concluir que, em cenários gerais e dinâmicos, não compensa. Existem várias razões para isto: em primeiro lugar, os modelos de deteção modernos foram concebidos para rapidez e aplicação em tempo real. A introdução de um pré-processamento adicional, como o background subtraction, implica não só um custo computacional extra como também uma necessidade cons-

tante de ajuste a variáveis do ambiente (clima, luz, sombras). Este fator compromete a vantagem principal de técnicas como o YOLO, que é precisamente a rapidez e eficiência em contextos variáveis.

No entanto, em ambientes mais controlados e estáticos, poderá fazer sentido recorrer a estas técnicas. Desde que o tempo de processamento não seja um fator crítico e que a potência de cálculo disponível seja suficiente, é plausível alcançar melhorias. Isto ficou patente em parte dos testes realizados: ao configurar os parâmetros de background subtraction para distâncias e planos de câmara mais restritos, observou-se uma tendência de melhoria nos resultados. Ainda assim, esta hipótese carece de validação mais aprofundada, uma vez que se baseia numa fração limitada dos dados e em pequenas variações.

Assim, com este trabalho confirma-se que a combinação de background subtraction com modelos SOTA exige uma afinação cuidada e uma análise custo-benefício rigorosa, para que o potencial ganho justifique o esforço adicional de configuração e processamento.

5.2 Trabalho Futuro

Uma vez exposto tudo o que foi analisado, surge uma ideia bastante promissora a partir deste estudo, que poderá ser explorada em trabalhos futuros. Ao aplicar técnicas de Background Subtraction, o objetivo principal foi facilitar os modelos State-Of-The-Art (SOTA) a concentrarem-se apenas nos objetos em movimento, aumentando assim a precisão na deteção exclusiva de pessoas.

Embora neste caso específico essa abordagem não tenha apresentado os resultados esperados, o conceito pode ser expandido. Em vez de aplicar a máscara de BGS de forma destrutiva removendo informação da imagem poder-se-ia utilizar essa máscara como guia para indicar ao modelo onde deve concentrar a sua atenção.

Esta ideia não é nova no domínio da Visão por Computador: tecnologias recentes como o GPT-4V (na sua componente de Visão, V*) demonstram como modelos podem ser orientados a "ver" certas regiões da imagem para responder a questões do utilizador, imitando a forma como o cérebro humano racionaliza onde procurar antes de analisar todos os detalhes.

Com esta inspiração, seria possível conceber uma nova implementação de modelos de deteção que, no pré-processamento, não eliminem partes da imagem, mas antes gerem pistas sobre as zonas de interesse, otimizando assim o uso dos recursos computacionais e potenciando a melhoria da precisão. Esta abordagem combinaria o melhor de ambos os mundos: a eficiência dos

modelos de deteção SOTA e a filtragem inteligente proporcionada pelas técnicas de BGS.

Bibliografia

[1] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773–780, 2006.

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, volume 28, pages 91–99, 2015.

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[4] F. Porikli, O. Tuzel, and P. Meer. A comparison between background modelling methods for vehicle segmentation in highway traffic videos. In *Proceedings of the IEEE International Conference on Pattern Recognition (ICPR)*, pages 777–780. IEEE, 2004.

[5] Chris Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, pages 246–252. IEEE, 1999.

[6] Qingyao Chen, Andrew Rabinovich, Xingchao Liu, Yao Zhao, Wei Ping, Ajay Jain, Rafael Valle, Mohammad Shoeybi, William Chan, and Bryan Catanzaro. V: Guided visual search as a core mechanism in multimodal llms. *arXiv preprint arXiv:2311.05697*, 2023.