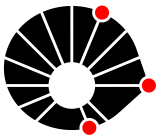

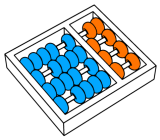

Monitoramento de Colmeias com Visão Computacional

E. S. Martins *G. S. P. Sobral* *L. F. Bittencourt*

Relatório Técnico - IC-PFG-25-36
Projeto Final de Graduação
2025 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Monitoramento de Colmeias com Visão Computacional

Elias Santos Martins[1], Gabriel Sanders Pereira Sobral[1], Luiz Fernando Bittencourt[1]

[1] Instituto de Computação Universidade Estadual de Campinas (UNICAMP), Caixa Postal 6176
13083-970 Campinas-SP, Brasil

Resumo. Este é o relatório do projeto realizado em parceria com o Prof. Roberto Greco do Instituto de Geociências, para monitoramento de colmeias de abelhas nativas brasileiras, orientado pelo Prof. Luiz Fernando Bittencourt . Tem por objetivo implementar um sistema embarcado, para ser utilizado em escolas, que realiza a contagem de abelhas de forma automatizada, utilizando uma câmera e as ferramentas mais recentes de visão computacional. É criado um dataset de vídeos de abelhas Jataí, com anotações dos momentos de entrada e saída e da localização e trajetória das abelhas. É implementado uma metodologia completa de contagem, que tem por entrada um vídeo gravado, e após o processamento, retorna em quais momentos do vídeo ocorreram entradas e saídas de abelhas. Na metodologia, são usados vários conceitos de aprendizado de máquina e visão computacional: aprendizado autossupervisionado, detecção de objetos com YOLO, rastreamento de trajetórias, e implementada uma heurística de contagem. Também é estudado como criar um sistema embarcado com câmera e conectá-la a um computador utilizando a rede Internet, possibilitando o envio do vídeo para processamento. Por fim, cada etapa da metodologia é avaliada com métricas adequadas as tarefas (detecção, rastreamento e contagem), e é proposta a forma de continuidade do projeto.

Palavras-Chave: Monitoramento de abelhas, aprendizado de máquina, visão computacional, detecção de objetos, internet das coisas, câmeras

1. Introdução

O projeto de monitoramento de colmeias por meio de tecnologias digitais surgiu como uma iniciativa interdisciplinar, envolvendo computação, geociências e engenharia elétrica, com o propósito de aliar ciência, educação e preservação ambiental. Desde sua concepção, buscou-se desenvolver um sistema acessível e de baixo custo para a observação do comportamento das abelhas, a coleta de informações ambientais e a disponibilização dos dados de forma clara e intuitiva a pesquisadores e a estudantes de escolas.

A primeira versão do sistema consistia em uma placa embarcada equipada com sensores básicos de temperatura, umidade e som, que transmitia os dados para um aplicativo móvel. Em versões posteriores, novas tecnologias foram incorporadas ao projeto, incluindo sensores adicionais de pressão atmosférica e de proximidade, além do emprego de placas com maior capacidade de processamento e conectividade Wi-Fi. Essas mudanças possibilitaram maior estabilidade na comunicação e no armazenamento dos dados, bem como a ampliação das métricas monitoradas.

Com a evolução do sistema, uma plataforma web foi introduzida para centralizar o acesso às informações e ampliar sua escalabilidade. Essa atualização também substituiu soluções iniciais de armazenamento por bancos de dados em nuvem mais flexíveis, permitindo o monitoramento simultâneo de múltiplas colmeias e facilitando a integração de diferentes fontes de dados.

Na etapa seguinte, técnicas de visão computacional foram incorporadas ao projeto, permitindo a contagem automática do fluxo de abelhas na entrada das colmeias. Essa inovação representou um marco, pois possibilitou a análise conjunta de parâmetros ambientais e de comportamento, enriquecendo os resultados e aproximando o sistema de aplicações tanto científicas quanto educacionais.

O presente trabalho dá continuidade a esse histórico de aprimoramentos, com foco em consolidar o uso da inteligência artificial na contagem de abelhas. O objetivo é superar as limitações identificadas em versões anteriores, aperfeiçoar os mecanismos de análise do fluxo de abelhas e oferecer uma solução mais robusta e versátil, alinhada às demandas de ensino e pesquisa.

A redação final deste relatório contou com o auxílio de ferramentas de Inteligência Artificial para revisão gramatical, aprimoramento da clareza e verificação ortográfica. O conteúdo intelectual, as análises técnicas e as conclusões são de autoria e responsabilidade integral dos autores humanos.

2. Evolução do Projeto e Objetivos

Durante a continuidade do projeto, realizou-se uma reunião com o professor Roberto Greco, do Instituto de Geociências, a fim de alinhar as necessidades e expectativas quanto à etapa atual de desenvolvimento. Na ocasião, o professor avaliou as versões anteriores e

destacou que, embora o hardware e a aplicação web já apresentem desempenho satisfatório, a funcionalidade de contagem de entradas e saídas das abelhas ainda apresenta resultados insatisfatórios. Além disso, o professor ressaltou a importância de oferecer um recurso de monitoramento ao vivo, a fim de enriquecer o uso do sistema tanto em contextos de pesquisa quanto em ambientes educacionais.

Diante desse levantamento de requisitos, definiu-se que o foco principal desta versão do projeto seria o aprimoramento da contagem de abelhas na entrada da colmeia, aliando maior precisão nos resultados a uma melhor experiência de monitoramento.

2.1. Objetivo Geral

O objetivo deste trabalho é desenvolver uma solução capaz de realizar, de forma confiável, a contagem do fluxo de abelhas, superando as limitações das versões anteriores, e disponibilizar recursos de monitoramento em tempo real, atendendo às demandas estabelecidas pelo professor Roberto Greco.

2.2. Objetivos Específicos

Para alcançar o objetivo geral, foram estabelecidas metas técnicas que orientaram o desenvolvimento do projeto:

- Investigar e implementar métodos de aprendizado não supervisionado para treinar modelos de visão computacional capazes de realizar a contagem automática de abelhas;

- Explorar diferentes arquiteturas de redes neurais, como YOLO e ResNet, avaliando seu desempenho e adequação ao problema proposto;

- Integrar os resultados obtidos ao sistema de monitoramento existente, de forma a permitir a análise conjunta das métricas ambientais e comportamentais;

- Desenvolver uma interface de monitoramento em tempo real, ampliando a usabilidade do sistema em aplicações educacionais e de pesquisa.

3. Revisão da literatura

Para atingir os objetivos propostos, buscamos na literatura o problema em questão, o que há de mais recente (2023 em diante) de pesquisa na área de contagem de abelhas.

3.1. Contagem de abelhas

O trabalho [1] trata do problema diretamente, relacionando abordagens que utilizam visão computacional clássica e deep learning para o problema de contagem de abelhas. Porém, consideramos os dados utilizados, que são de abelhas passando em um "túnel", como muito limitados e sendo um problema diferente do que buscamos resolver.

Fig. 1: Exemplo de dado que [1] trata

Em [2], é apresentada uma metodologia completa de contagem de abelhas com deep learning, dividida em três etapas: detecção das abelhas, tracking das abelhas no vídeo e heurística de contagem de abelhas. As abelhas são filmadas por uma câmera posicionada diretamente acima da saída da colmeia, conectada a um sistema IoT de monitoramento. Diferentes tamanhos da YOLO v8 são utilizados como detector, são comparados vários algoritmos de tracking de objetos (ByteTRACK, OC-SORT, Deep OC-SORT) e também são comparadas diferentes heurísticas de contagem de uma saída ou de uma entrada de abelha. São propostas várias métricas para avaliar cada uma das três etapas do pipeline. O resultado final é a comparação de como as diferentes técnicas de detecção, tracking e contagem influenciam as métricas propostas, bem como uma série temporal de quantas abelhas entraram e saíram ao longo de um dia, o que também é um objetivo do nosso trabalho. O paper é bem detalhado, e seu código está disponível online, por isso, decidimos por o tomar como referência para o nosso trabalho.

Fig. 2: Exemplo de dado que [2] trata

Gordon et al. [3] utilizam uma abordagem diferente ao buscar realizar a segmentação, e não a detecção, das abelhas no problema de contagem. O trabalho compara duas metodologias diferentes para realizar essa segmentação utilizando imagens, uma de visão clássica e outra utilizando a YOLO v8 [4] de segmentação. É notado um problema comum: as duas

metodologias têm alta precisão quando há uma ou poucas abelhas, mas perdem precisão quando há muitas abelhas juntas. Para vídeos, são testados modelos de Video Object Segmentation, como Pixellib [5] e Xmem [6] . O trabalho conclui que as metodologias possuem sucesso considerável na segmentação e tracking de abelhas, mas que a qualidade cai consideravelmente quando há muitas abelhas na cena. Os modelos se aplicam a condições controladas: filmagem perto da saída da colmeia, em ângulos específicos e em boas condições de iluminação, com restrições semelhantes às que usamos em nosso trabalho.

Fig. 3: Exemplo de dado tratado em [3]

Thompson et al. [7] comparam duas metodologias de detecção, uma utilizando YOLO, outra utilizando Optical Flow, que, em vez de detectar abelhas, busca analisar quais pixels do vídeo estão se movimentando mais, para determinar quais são as abelhas. São propostas formas diferentes de se determinar a heurística de contagem, utilizando não somente um retângulo, como é feito em [2], mas também um triângulo e um arco. São realizadas buscas de hiperparâmetros utilizando Grid Search para otimizar as duas metodologias. O trabalho conclui que os dois métodos têm acurácia semelhante na tarefa, mas o método que usa Optical Flow é bem menos custoso computacionalmente. Porém, consideramos o dataset utilizado como diferente do nosso trabalho, pois a filmagem é feita por cima e o fundo é uma placa de colmeia branca, o que favorece o modelo de optical flow, pois é um fundo de pouca variação, o que não é o caso das nossas filmagens.

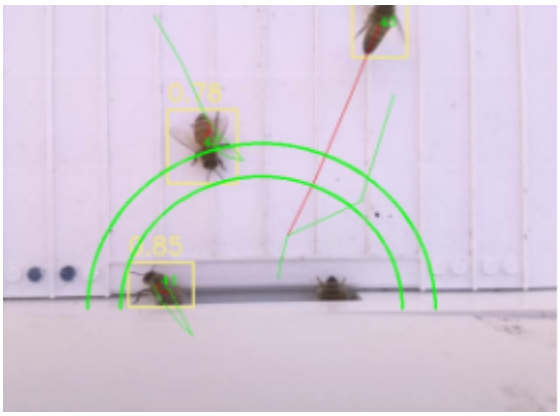

Fig. 4: Exemplo de dado tratado em [4]

3.2. Aprendizado autossupervisionado para detecção de objetos

Tendo em vista a grande dificuldade e o tempo necessário para anotar os vídeos coletados, buscamos técnicas que pudessem ser utilizadas sem necessidade de anotação. Aprendizado autossupervisionado é uma técnica em alta atualmente, na qual os dados não precisam ser anotados para o treino inicial do modelo. Os modelos autossupervisionados aprendem a realizar a representação dos dados, utilizando uma tarefa pretexto,como, por exemplo, aprender a embaralhar e reconstruir a imagem. Após esse treino inicial não supervisionado, podem ser treinados mais facilmente com uma pequena quantidade de dados anotados para a tarefa específica (downstream task), de maneira supervisionada.

É possível aplicar aprendizado autossupervisionado para a tarefa de detecção de objetos. Alina et al. [8] trazem um resumo das principais técnicas de aprendizado autossupervisionado aplicáveis a modelos de detecção de objetos. As técnicas são divididas em dois grandes grupos:

1. Técnicas de discriminação de instância adaptadas à detecção de objetos: buscam minimizar a distância no espaço latente entre features extraídas de instâncias que compartilham significado semântico semelhante. Obtemos essas instâncias semelhantes por meio da aumentação de dados. Para esta técnica ser aplicável à detecção de objetos, as aumentações são aplicadas no nível local, e não na imagem inteira
2. Técnicas de ocultação: outro grupo de técnicas busca ocultar uma parcela considerável da imagem e treinar o modelo para reconstruí-la. A ideia é trazer resistência à oclusão e localidade ao detector, que deve entender a "semântica" da imagem para reconstruí-la. As técnicas vão se diferenciar quanto à proporção de oclusão, ao formato das máscaras de oclusão e ao local de colocação das máscaras na imagem.

As técnicas também são diferenciadas quanto ao tipo de modelo a que se aplicam. Algumas são aplicáveis a CNNs , enquanto outras são aplicáveis apenas a Visual Transformers (ViTs).

O trabalho de [8] compara algumas técnicas selecionadas em uma tarefa geral de detecção de objetos, bem como em uma tarefa específica: detectar carros em imagens de satélite, o que é semelhante à que temos aqui, de detectar objetos pequenos. Das técnicas testadas, as que são aplicáveis ao nosso problema são a ReSim [9] e a SparK [10], pois se aplicam a modelos CNN.

Jun et al. [9] apresentam uma aplicação de aprendizado autossupervisionado para a detecção de falhas em objetos industriais, usando uma adaptação da YOLO v11.

4. Coleta de dados

4.1. Colmeias de Jataí

A primeira coleta foi realizada na casa do professor Roberto Greco, do Instituto de Geociências, que possui várias colmeias de abelhas nativas (Jataí) em sua residência.

Realizamos a coleta utilizando um smartphone Samsung Galaxy S23 Ultra, devido à sua qualidade de câmera. Variamos:

- ângulos de filmagem: perpendicular à saída, ou até 30 graus
- Zoom aplicado: sem zoom, 2x e 4x
- Frames por segundo (FPS): 30, 60 e 120 fps
- Colmeia filmada: a cada possuía várias colmeias em posições e tamanhos distintos

Posicionamos a câmera a uma distância de 30 cm da colmeia. Buscamos não ficar na frente da saída para não interferir na rota de voo das abelhas. Ao todo, obtivemos 16 vídeos, com duração de aproximadamente 1 minuto cada.

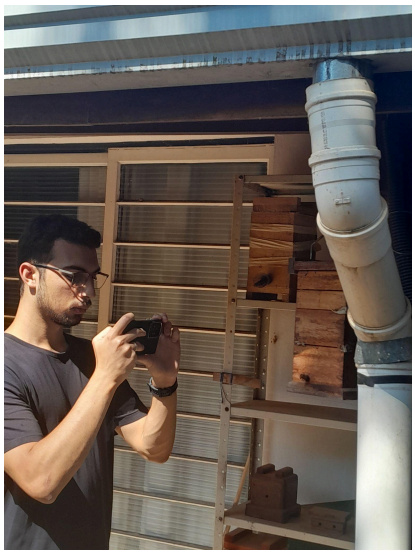

Fig. 5: Método de filmagem das abelhas

Fig 6: Frame do vídeo resultante

4.2. Dados externos

Após o primeiro teste do modelo de detecção, notamos que ele não estava generalizando bem para outros vídeos. Por isso, fomos em busca de mais dados. Utilizamos os dados anotados no trabalho anterior [11] , bem como datasets disponíveis online [12, 13, 14] em trabalhos da literatura ou na plataforma Kaggle. A tabela a seguir contém informações sobre os dados:

Origem	Local	Número de imagens
[12]	San Jose, Cupertino e Gilroy Califórnia, Estados Unidos	8080
[13]	Projeto BeeLivingSensor	4993
[14]	Reino Unido	3044
Dataset do grupo anterior	Brasil	1035

Tabela 1: Dados externos

Criamos um pré-processamento para padronizar estes dados, deixando as anotações no formato YOLO (arquivos .txt com coordenadas normalizadas) e as imagens já escaladas para o tamanho 640x640 (padrão da YOLO).

Ignoramos as imagens sem anotações. Ao final, obtivemos 13485 imagens de dados externos.

Fig. 7: Exemplos dos dados coletados externamente. Podemos ver que obtivemos uma grande diversidade quanto à quantidade, fundo em que estão, formato, posição e tamanho das abelhas

5. Anotação dos dados

Para a implementação do nosso pipeline, dois tipos de anotações são necessários para treinar e validar a técnica proposta.

5.1. Entrada e saída

O primeiro tipo de anotação exigido foi anotar os momentos de entradas e saídas de abelhas em cada vídeo, para testar e validar a contagem final realizada pela nossa técnica. Para isso, utilizou-se a ferramenta Behavioral Observation Research Interactive Software (BORIS) [15]. Essa ferramenta nos permite registrar de modo simples e preciso a ocorrência de eventos em um vídeo, bastando apenas definir os eventos, associar teclas para cada evento e então apertá-las no momento em que o evento acontece enquanto os frames de um vídeo passam.

Para os propósitos desse projeto, criamos apenas dois eventos: entrada e saída de uma abelha da colmeia. Então nós importamos os vídeos que gravamos no BORIS e fizemos o processo de anotação.

Como resultado desse processo, obtivemos uma tabela com apenas duas colunas: a primeira indica o número do frame em que o evento ocorreu e a segunda indica o tipo de evento (entrada ou saída). Essa anotação é simples e relativamente rápida, uma vez que não foram necessários mais de 15 minutos por vídeo para concluir essas anotações.

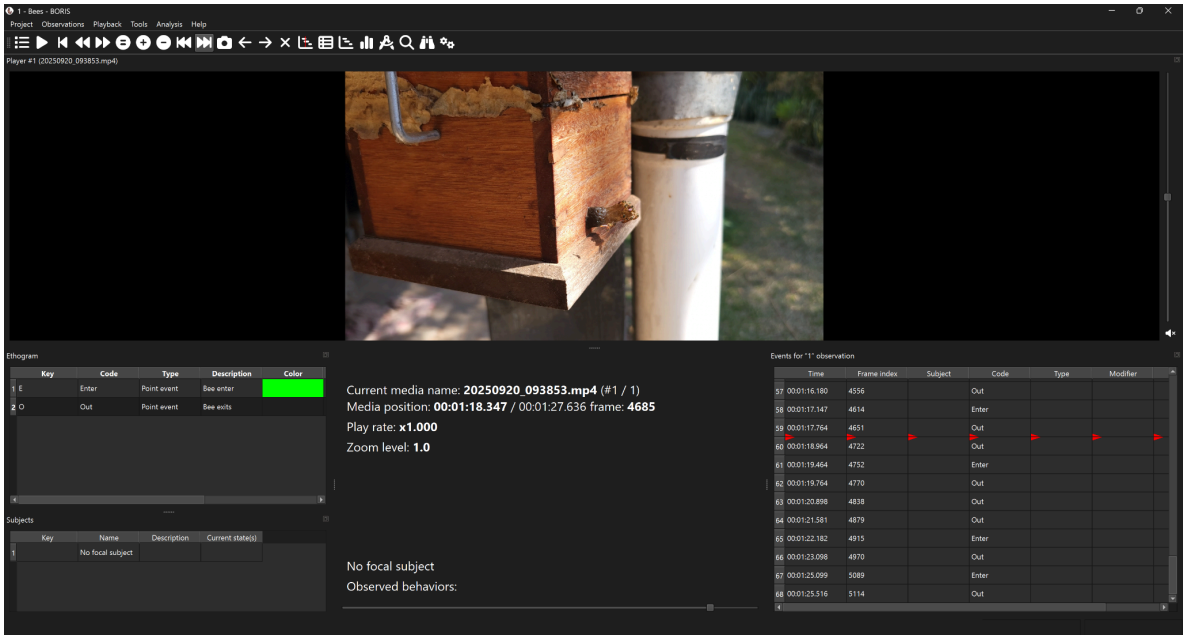

Fig. 8: Interface do software BORIS, exibindo o processo de codificação de comportamento com linha do tempo, lista de sujeitos e tabela de eventos comportamentais.

5.2. Tracking das abelhas

O segundo tipo de anotação necessária é o de tracking das abelhas, que é necessário tanto para treino do modelo de detecção quanto para a validação da técnica de tracking. Para tal, utilizamos a ferramenta Computer Vision Annotation Tool (CVAT) [16], que nos proporcionou uma maneira fácil de realizar o tracking das abelhas. Foi utilizada a versão self-hosted do software, de modo que todos os dados de anotação permaneceram armazenados localmente.

Essa aplicação permite a criação de bounding boxes em volta de abelhas e a atribuição de um ID a cada uma delas, de modo que cada abelha fica associada a uma bounding box, que deve ser reposicionada conforme o movimento das abelhas. Esse processo foi realizado para cada uma das abelhas presentes nos vídeos anotados, desde o momento em que apareciam no vídeo, até o momento em que saíam dele.

A definição de a partir de qual momento uma abelha que está aos poucos entrando na colmeia deve ter sua bounding box retirada e que momento uma abelha que está saindo da colmeia lentamente ou aparece apenas parcialmente deve ter a sua bounding box anotada foi uma tarefa não trivial. Levando em consideração que o nosso objetivo é detectar a entrada e saída de abelhas, e que para cada um desses eventos a abelha necessariamente tem que estar fora da colmeia em algum instante próximo ao evento, nós optamos por anotar abelhas somente quando elas conseguem ser observadas em sua totalidade, visando evitar ruído nas anotações.

Diferentemente do primeiro tipo de anotação, esse processo foi exaustivo e tomou uma grande quantidade de tempo, uma vez que o movimento das abelhas é muito rápido e há grande variação de movimento em poucos frames, além de existir sempre mais de uma abelha em cada frame. Anotamos vídeos de 1 minuto e de 1 minuto e 30 segundos, gravados a 60 fps, resultando em um total de cerca de 9000 frames anotados. Dividimos a tarefa para cada um dos membros do grupo de modo a tornar a anotação mais eficiente.

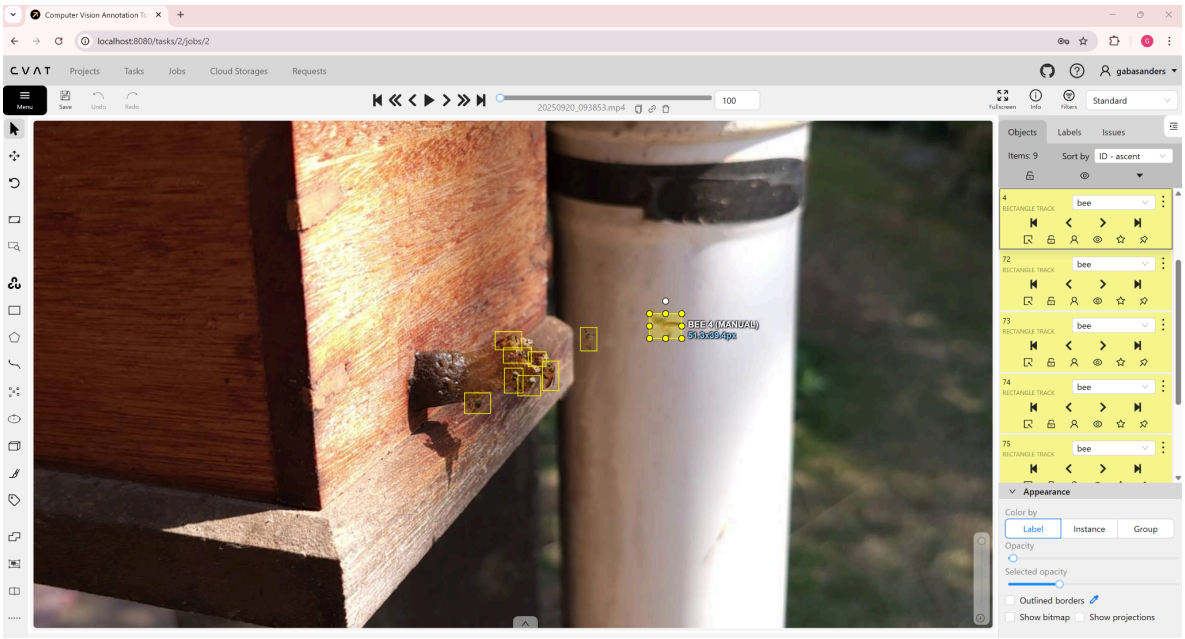

Fig. 9: Tela de anotação de dados do CVAT, com a criação de *bounding boxes* para marcar objetos e o processo de *tracking* desses objetos.

6. Hardware utilizado

6.1. Placa

Nesta versão do trabalho, utilizamos a placa Raspberry Pi 3 Model B+ [17] . A escolha desta placa se deve ao fato de ela possuir uma interface para conectar uma câmera de IoT, diferentemente das placas utilizadas nos trabalhos anteriores, que não possuem essa entrada. Também é muito útil ela possuir um módulo Wi-Fi, que permite o upload do vídeo da câmera para um servidor. A seguir estão as especificações da placa:

- Processador Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64 bits, 1,4 GHz
- 1 GB de memória LPDDR2 SDRAM
- Conectividade sem fio: Wi-Fi 2,4 GHz e 5 GHz (IEEE 802.11 b/g/n/ac), Bluetooth 4.2 e BLE
- Ethernet Gigabit via USB 2.0 (taxa máxima de ~300 Mbps)
- Header GPIO estendido de 40 pinos
- Porta HDMI em tamanho padrão
- 4 portas USB 2.0

- Porta CSI para conexão de câmera Raspberry Pi
- Porta DSI para conexão de display touchscreen Raspberry Pi
- Saída estéreo de 4 polos e porta de vídeo composto
- Slot microSD para sistema operacional e armazenamento
- Alimentação: entrada DC 5 V/2,5 A
- Suporte a Power-over-Ethernet (PoE) mediante uso de PoE HAT opcional

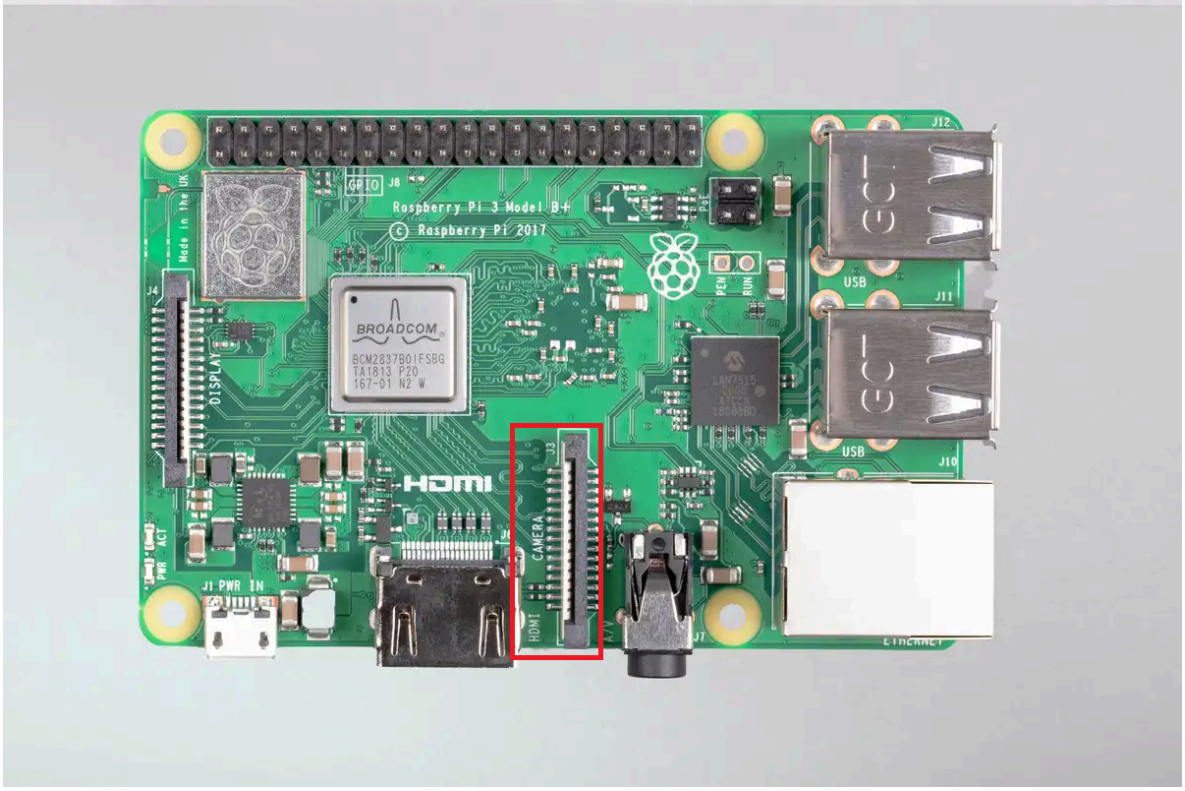

Fig. 10: Raspberry Pi 3 B+. Em vermelho está destacado a *Camera Serial Interface* (CSI), que motivou a escolha dessa placa e possibilita recuperar conteúdo da câmera

6.1.1. Configuração da placa

Realizamos a configuração básica da placa, instalando o sistema operacional Raspberry OS [18] em um pendrive. O passo a passo detalhado de como realizar a configuração da placa está disponível em nosso repositório. Após isso, conectamos a placa à rede de internet local e ativamos a conexão SSH da placa, para ser possível enviar arquivos do computador para a placa.

Em nossa opinião, a parte de IoT do projeto foi a mais trabalhosa e demorada a ser feita, sendo um grande desafio técnico.

6.2. Câmera

Como em todo problema de visão, a escolha da câmera é crucial para se obter resultados. Dado nosso problema, temos os seguintes desafios:

- Abelhas, especialmente jataís, são pequenas (4 a 5 mm)
- As abelhas se movem muito rápido, com velocidade que pode chegar a 7 metros por segundo, milhares de vezes seu tamanho [14]
- Normalmente abelhas estão conglomeradas perto da entrada da colmeia
- A filmagem será em um ambiente aberto
- A filmagem deve ser feita de modo contínuo e ininterrupto durante todo o período do dia

Levando em consideração todas essas restrições, a câmera escolhida foi a Innomaker IMX296 . A escolha se deu por ser a câmera de menor custo que atendia aos requisitos do problema. A seguir listamos as especificações da câmera:

- Sensor do tipo CMOS
- Distância focal 39mm
- resolução 1456 x 1088 pixels
- Captura até 60 frames por segundo
- Ângulo FOV de 90 graus
- Modo de captura global (global shutter)

Fig. 11: Câmera Innomaker IMX296

6.2.1. Global shutter x rolling shutter

Uma diferença fundamental a se entender é a de global shutter (GS) e rolling shutter (RS)

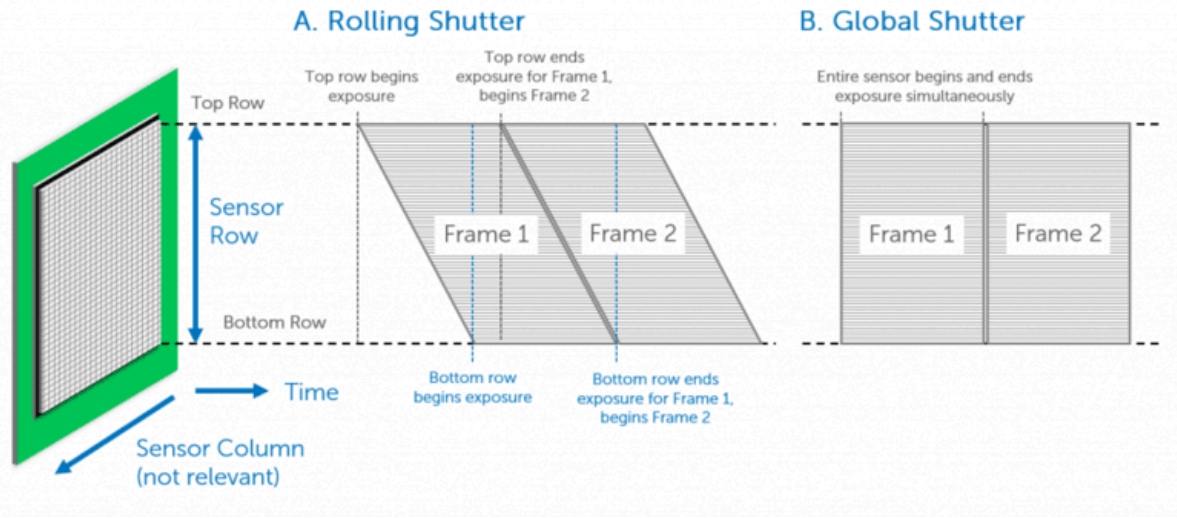

Fig. 12: Diferença entre global e rolling shutter

No Rolling shutter, a captura dos pixels é feita linha por linha dos sensores. Com isso, sensores de pixels de linhas diferentes da imagem entram em exposição em momentos ligeiramente diferentes. Isso causa alguns efeitos indesejados, principalmente com objetos que se movem muito rápido, como é o caso das abelhas

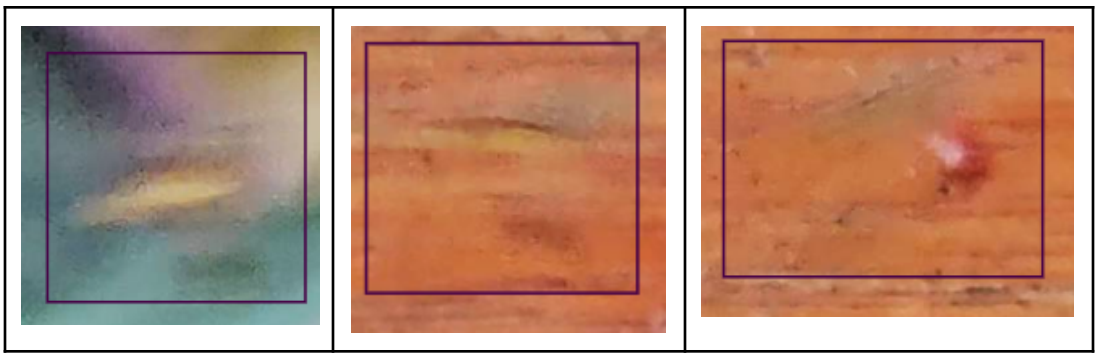

Fig. 13: Exemplos de artefatos gerados pela alta velocidade de voo das abelhas ao filmar com o celular (sensor RS)

Já no global shutter, a captura de pixels é feita em todos os sensores ao mesmo tempo, efetivamente capturando todos os pixels ao mesmo tempo, e evitando a criação de artefatos pelo movimento rápido.

Câmeras de celular e de IoT mais simples são feitas com rolling shutter, devido ao baixo custo. Porém, dadas as características do nosso problema, entendemos que uma câmera com global shutter é fundamental, dado que o RS faz com que abelhas que se movem rapidamente sejam capturadas como borrões, como pode ser visto na figura 13.

6.2.2. Configuração da câmera

A câmera foi conectada à Raspberry utilizando o cabo de conexão de 20 pinos e a interface CSI. Com a conexão feita, foi necessário configurar os drivers da câmera para uso na Raspberry, procedimento detalhado em nosso repositório.

6.3. Conexão

A Raspberry Pi, por ser um dispositivo embarcado, não teria poder computacional para executar o pipeline de processamento. Portanto, precisamos enviar de alguma forma a leitura da câmera para um computador de maior poder computacional.

Para isso, utilizamos a rede internet e sockets. Criamos um código Python para ser executado na placa. O código abre um socket TCP na placa, o que a torna um servidor. Criamos um código Python a ser executado no computador onde se deseja receber o feed da câmera, que se conecta ao socket da Raspberry utilizando o IP.

Após a conexão, o código da Raspberry recebe o feed da câmera no formato padrão, YUV420. Então, ele é processado na placa para o formato JPEG e tamanho compatível com a YOLO (640x480). Isso é feito para reduzir a quantidade de dados a serem transmitidos na rede, evitando o afogamento e os atrasos por falta de banda.

Após o processamento, os dados são escritos no socket e enviados. O computador recebe estes dados e os exibe na tela.

Futuramente, estes dados serão enviados ao pipeline de processamento, que contará as abelhas.

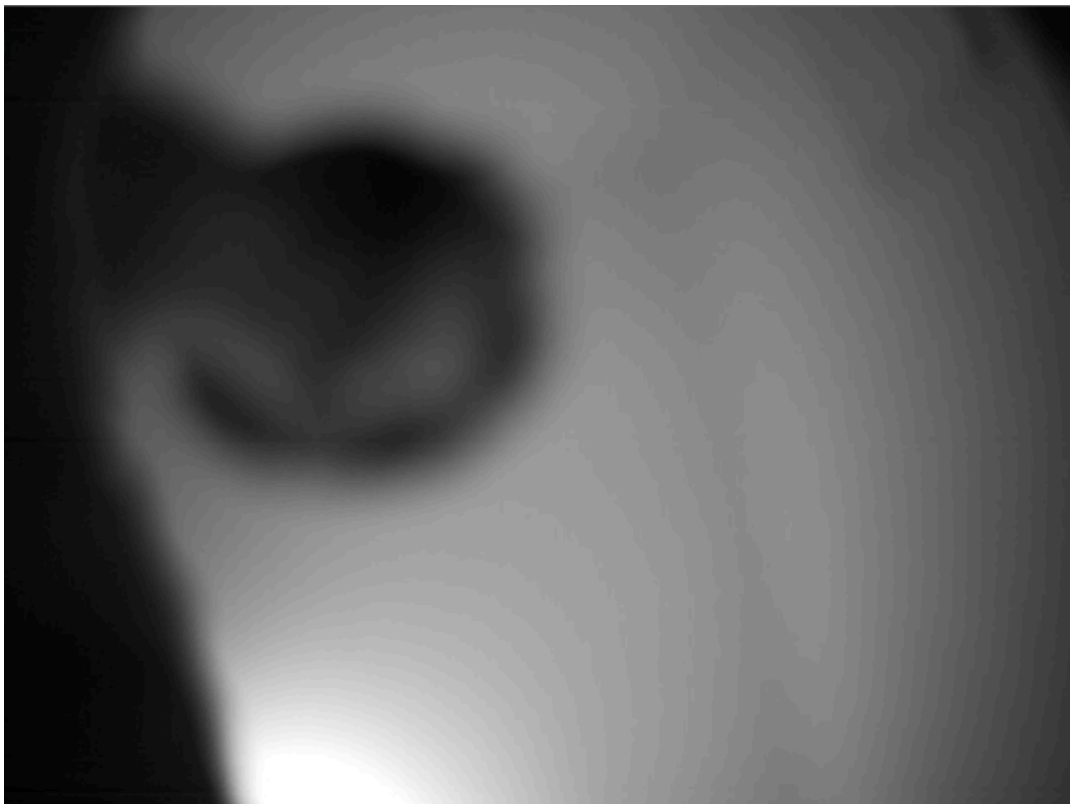

Fig. 14: Saída da câmera no computador. Podemos ver que ainda é necessária uma lente para que a câmera fique focada corretamente.

7. Detecção

7.1. Aprendizado auto supervisionado

Para começar o treinamento de um bom local, tentamos aplicar aprendizado autossupervisionado, para utilizar o grande volume de dados não anotados que possuímos. Em especial, tentamos reproduzir a técnica Region similarity representation learning [7] , utilizando o código disponibilizado pelos autores em um repositório do GitHub.

Porém, após várias tentativas, não conseguimos realizar a execução do código, devido a problemas de bibliotecas linkadas dinamicamente (DLLs) existentes no projeto, as quais não possuímos o conhecimento necessário para solucionar. Com isso, não utilizamos o aprendizado autossupervisionado e aplicamos apenas a parte supervisionada.

7.2. Aprendizado supervisionado

O aprendizado supervisionado foi a abordagem que conseguimos aplicar com sucesso para a tarefa de detecção de abelhas, dada a dificuldade de implementar as técnicas de aprendizado autossupervisionado.

Para essa abordagem, utilizamos a arquitetura YOLO (You Only Look Once) na décima primeira versão, uma rede neural de última geração conhecida por sua alta velocidade e precisão na detecção de objetos em tempo real. A YOLOv11 é a versão mais recente e também a mais adequada para o nosso projeto. Nós optamos por treinar tanto a versão nano quanto a versão large para sermos capazes de comparar diferentes tamanhos da YOLO e seus respectivos resultados. O modelo mais leve possível deveria ser capaz de ser executado diretamente em dispositivos embarcados (como uma Raspberry Pi com aceleradores de IA) ou até mesmo para garantir uma inferência rápida e de baixo custo no servidor, enquanto o modelo mais largo deveria ter uma alta acurácia, mas com uma inferência mais demorada.

Para o treino do modelo, utilizamos o conjunto de dados anotado pelo grupo, composto por vídeos próprios (seção 4.1) e por dados externos padronizados (seção 4.2), totalizando mais de 20.000 imagens para o treinamento.

O treinamento do modelo YOLOv11 foi realizado no computador de um dos integrantes, com uma GPU NVIDIA RTX 3060, que permitiu um treinamento rápido e eficiente.

Parâmetro	**Valor**
Arquitetura Base	YOLOv11n (nano) e YOLOv11l (large)
Tamanho da Imagem	640x640 pixels
Épocas	50
Batch Size	16

Tabela 2: Parâmetros do treino da YOLO

7.3. Métricas utilizadas

Após o treinamento, o modelo foi avaliado utilizando métricas padrão para detecção de objetos:

Métrica	Definição
Precision	Proporção de detecções corretas (*True Positives*) em relação ao total de detecções.
Recall	Proporção de objetos reais detectados (*True Positives*) em relação ao total de objetos reais.
mAP@50	Precisão média (Average Precision) calculada com um *Intersection over Union* (IoU) de 50%. É a métrica padrão para avaliar detectores.

Tabela 3: Métricas para avaliar a detecção

8. Tracking

Após realizar a detecção das abelhas em cada frame, temos um outro desafio a resolver: realizar a associação das abelhas detectadas em um frame com o frame anterior, qual a identidade de cada abelha. Isso consiste em um problema conhecido na visão computacional: o tracking de objetos em vídeo. Em especial, como temos muitas abelhas em cada frame, trata-se de um Multi-Object Tracking (MOT).

Lei et al. [2] compararam várias técnicas e algoritmos de tracking para o problema de rastrear as abelhas e concluíram pelos resultados que a melhor estratégia é a Observation Centric SORT (OC-SORT) [19] .

A técnica OC-SORT faz parte de um grupo de técnicas que se baseia nos movimentos dos objetos entre um frame e outro e não considera as características dos objetos em si. Isso o torna eficaz no tracking de abelhas, já que, como todas as abelhas são parecidas entre si, as características não ajudariam muito no tracking.

Por isso, implementamos a técnica OC-SORT em nosso pipeline. Qualitativamente, consideramos os resultados bons, mas ainda há uma certa perda do algoritmo na região perto da saída, onde muitas abelhas se acumulam.

8.1. Métricas utilizadas

Avaliar a qualidade de um MOT de maneira quantitativa não é algo direto. Na literatura, muitas métricas são propostas. Optamos por avaliar a qualidade do ByteTrack em nossos vídeos utilizando as mesmas métricas de [2] , sendo elas multiple object tracking accuracy (MOTA) [20] e Identity F1 Score (IDF1).

8.1.1. MOTA

O MOTA é uma métrica que avalia a qualidade global do tracking, considerando também a qualidade de detecção. A fórmula é dada por

$$MOTA = 1 - \frac{FN + FP + IDSW}{GT}$$

- **False Negatives (FN)**: objetos reais que o tracker não detectou.
- **False Positives (FP)**: detecções que não correspondem a nenhum objeto real.
- **Identity Switches (IDSW)**: momentos em que o tracker troca o ID de um objeto durante o rastreamento.
- **Ground truths (GT):** Total de objetos reais ao longo de todos os frames

O valor da MOTA varia de $-\infty$ a 1. 1 indica rastreamento perfeito (nenhum erro). Valores próximos de 0 indicam muitos erros combinados. Valores negativos ocorrem quando os erros (FN + FP + IDS) superam o número de objetos reais, o que indica desempenho muito ruim. O MOTA é fortemente influenciado pela qualidade da detecção; mesmo um bom associador pode obter um MOTA baixo se perder muitos objetos ou gerar falsos positivos.

8.1.2. IDF1

O IDF1 mede o quão bem o tracker mantém a consistência de identidade dos objetos ao longo do tempo. Ele calcula o F1-score entre as identidades previstas e as identidades reais, segundo a fórmula a seguir:

$$IDF1 = \frac{2 \cdot IDTP}{2 \cdot IDTP + IDFP + IDFN}$$

- IDTP (ID True Positive): O número de trajetórias corretamente identificadas
- IDFP (ID False Positive): predições do tracker que não correspondem a nenhuma trajetória do ground truth
- IDFN (ID False Negative): Trajetórias do ground truth corretamente identificadas pelo tracker

O IDF1 varia de 0 a 1. 1 indica que todas as identidades foram preservadas corretamente ao longo de todo o vídeo. Valores baixos indicam que o tracker frequentemente troca IDs, perde identidades ou associações estão inconsistentes. O IDF1 foca na estabilidade da identidade, sendo menos sensível a falhas de detecção isoladas.

Para cálculo das métricas, utilizamos a biblioteca py-motmetrics, disponível em https://github.com/cheind/py-motmetrics .

9. Contagem

Para realizar a contagem a partir das trajetórias detectadas na etapa de tracking, desenvolvemos uma abordagem própria, inspirada em [2] .

A primeira etapa é localizar a saída da colmeia. Para isso, nos baseamos na seguinte observação heurística: em todo momento, a grande maioria das abelhas Jataí presentes na filmagem está caminhando sobre o cano de saída. Portanto, para descobrir as coordenadas x e y da colmeia, caso tenhamos mais de 5 abelhas detectadas:

$$x_{colmeia} = median(X_{abelha})$$
$$y_{colmeia} = median(Y_{abelha})$$

Onde X e Y são o conjunto das coordenadas dos pontos centrais das bounding boxes de abelhas da etapa de detecção. Optamos por utilizar a mediana das coordenadas por ser mais estável que a média, e não ser afetada por poucas abelhas que saem voando. Caso haja menos que 5 abelhas, aproximamos o centro da colmeia como sendo centro da imagem.

Com isso, tendo a coordenada da entrada da colmeia, definimos um retângulo, centrado nesse ponto, com altura igual a ⅓ da altura do vídeo, e largura igual a ⅓ da largura do vídeo. Definimos o evento de entrada quando uma abelha passa do lado de fora para o lado de dentro desse retângulo, e o evento de saída quando uma abelha passa do lado de dentro para o lado de fora.

Para um mesmo id de abelha, consideramos apenas o primeiro evento dentro de uma janela de 5 segundos, para evitar uma sobrecontagem por movimentação próxima à borda do retângulo.

Fig. 15: exemplo do retângulo que vai definir entrada em saída (em amarelo), abelhas detectadas estão em laranja

9.1. Métricas utilizadas

Assim como na etapa de tracking, definir métricas quantitativas para a contagem não é algo direto, pois devemos associar as predições aos eventos reais, que podem ocorrer em momentos diferentes.

Definimos uma tolerância de 1 segundo para um evento ser detectado, isto é, se o nosso método de contagem detectou um evento até 1 segundo antes ou depois de um evento de entrada ou saída real, consideramos que o evento foi detectado corretamente.

Definimos:

- **True positive (TP):** eventos reais que foram corretamente detectados
- **False positive (FP):** eventos detectados, mas que não correspondem a nenhum evento real
- **False negative (FN):** eventos reais, mas que não tiveram nenhuma detecção

Note que não é possível definir o que seria um true negative, neste cenário. Mas com TP, FP e FN podemos calcular as métricas utilizadas por [2]

$$Precisão = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\ score = \frac{2 \cdot Precisão \cdot Recall}{Precisão + Recall}$$

10. Avaliação e métricas

10.1. Detecção

Os resultados obtidos no conjunto de validação foram:

Métrica	**Valor**
Precision	0.895
Recall	0.852
mAP@50	0.901

Tabela 4: Métricas da detecção no conjunto de validação

Estes resultados indicam que o modelo apresenta alta acurácia na detecção de abelhas, o que é um passo fundamental para o sucesso das etapas subsequentes de tracking e contagem. A alta performance, mesmo com a versão "nano", valida a abordagem de utilizar um grande e diverso conjunto de dados externos para melhorar a generalização do modelo.

Apesar do bom desempenho, notamos que o modelo ainda apresenta dificuldade em diferenciar abelhas individuais em aglomerados muito densos perto da entrada da colmeia, um problema comum destacado na literatura [3]. No entanto, as taxas de falsos negativos (FN) e falsos positivos (FP) foram consideradas aceitáveis para prosseguir com a implementação do pipeline de tracking.

Este detector treinado é o "melhor detector" mencionado na seção a seguir, que será utilizado em conjunto com o ByteTrack para a avaliação do tracking em um cenário real.

10.2. Tracking

Para testar a qualidade do ByteTrack na tarefa de tracking de abelhas, primeiramente, o testamos utilizando as boxes anotadas manualmente, isto é, supondo um detector perfeito. Os resultados estão na tabela a seguir:

Vídeo	**MOTA**	**IDF1**
20250920_093853	0.978	0.973
20250920_095125	0.976	0.981

Tabela 5: Resultados do ByteTrack na tarefa de tracking, considerando detecções anotadas à mão

Podemos verificar que, considerando uma detecção perfeita, o ByteTrack é capaz de manter bem as trajetórias das abelhas, pois ambas as métricas resultaram em valores próximos de 1.

Já considerando o cenário real, utilizamos nosso melhor detector em conjunto com o ByteTrack, e obtemos os resultados na tabela 6.

Vídeo	**MOTA**	**IDF1**
20250920_093853	0.800	0.367
20250920_095125	0.922	0.748

Tabela 6: Resultados do ByteTrack na tarefa de tracking, considerando detecções da YOLO

Como esperado, as métricas foram piores ao utilizar as detecções da YOLO, que são imperfeitas. É interessante notar que a métrica IDF1 caiu bem mais que a MOTA, o que indica que a qualidade global da detecção se manteve, mas os pequenos erros do detector

influenciaram negativamente a capacidade do tracker de manter a estabilidade de identidade.

10.3. Contagem

Por fim, executamos a contagem, e pudemos obter o resultado final de nosso pipeline. Obtivemos as métricas para os 5 vídeos dos quais anotamos entrada e saída. Como podemos observar nas tabelas 7 e 8, os resultados foram medianos, com contagem próxima dos valores reais, e uma precisão que varia por vídeo, de 20 a 60% . Chama atenção o vídeo 20250920_094449, onde nenhuma saída ou entrada foi detectada. Inferimos que isto foi causado pela angulação diferente que este vídeo apresenta em relação aos outros, o que faz com que o modelo detector não detecte as abelhas corretamente, impossibilitando o tracking e a subsequente contagem.

Vídeo	**Duração**	**Entradas reais**	**Saídas reais**	**Entradas detectadas**	**Saídas detectadas**
20250920_093853	87.6 s	29	39	63	57
20250920_094130	60.8 s	38	49	40	36
20250920_094449	60.6 s	17	41	0	0
20250920_094910	81.1 s	56	56	48	50
20250920_101010	30.4 s	16	26	17	9

Tabela 7: Resultados do pipeline completo para 5 vídeos, entradas e saídas

Vídeo	**Duração**	**Precisão**	**Recall**	**F1 Score**
20250920_093853	87.6 s	0.242	0.426	0.309
20250920_094130	60.8 s	0.487	0.425	0.454
20250920_094449	60.6 s	0	0	0
20250920_094910	81.1 s	0.276	0.241	0.257
20250920_101010	30.4 s	0.654	0.405	0.500

Tabela 8: Resultados do pipeline completo para 5 vídeos, métricas

10.4. Eficiência

Testamos a qual framerate conseguimos rodar diferentes tamanhos de detectores, utilizando o vídeo 20250920_093853, e vendo como os diferentes tamanhos influenciam a qualidade

dos resultados. Os testes foram realizados em notebook pessoal dos integrantes, com uma RTX 3050i .

Na tabela 9 podemos observar a comparação. A diferença de fps e notória, com o modelo large caindo quase pela metade a quantidade de frames que consegue processar por segundo. Porém, isso vem acompanhado de um ganho nas métricas. Para processar um vídeo em tempo real, precisamos que o FPS de processamento seja maior que o FPS em que o vídeo foi filmado, o que não foi o caso, pois os vídeos estão a 60 fps . Um hardware mais potente pode conseguir executar em tempo real até mesmo a YOLO large, evitando a perda de precisão por usar modelos menores.

Modelo detector	**Framerate**	**MOTA**	**IDF1**	**Precisão**	**Recall**	**F1 Score**
YOLOv11n	58.43 fps	0.632	0.214	0.440	0.162	0.237
YOLOv11m	41.11 fps	0.713	0.373	0.162	0.647	0.259
YOLOv11l	33.10 fps	0.799	0.367	0.241	0.426	0.309

Tabela 9: Comparação com diferentes tamanhos de YOLO

11. Conclusão

Obtivemos sucesso em implementar um método de contagem de abelhas utilizando visão computacional. Percebemos que a característica de maior importância neste processo é a qualidade do detector de abelhas, já que todas as outras etapas dependem das abelhas serem detectadas corretamente. Com mais desenvolvimento, seria possível aplicar esta metodologia em um sistema real, conectando um sistema embarcado que grava a colmeia com um servidor que realiza o processamento.

Foi possível aplicar os mais diversos conhecimentos obtidos ao longo da graduação, como desenvolvimento de software, machine learning, visão computacional, redes de computadores, eletrônica, entre outros.

Consideramos a precisão final do método satisfatória, dada a limitação de tempo e recursos, bem como os desafios encontrados, principalmente na parte de autossupervisão e sistema embarcado.

12. Futuros passos

Acreditamos que o planejamento dos próximos passos do projeto concentra-se em duas frentes principais: o aprimoramento da inteligência artificial e a conclusão da integração do sistema físico com o pipeline de processamento e a validação quantitativa dos resultados.

12.1. Aprimoramento da Detecção e Generalidade do Modelo

Esta etapa visa resolver as limitações atuais do modelo e garantir maior robustez sob diferentes condições. Primeiramente, é crucial solucionar os problemas de dependência que impediram a implementação do Aprendizado Autossupervisionado, pois isso é fundamental para aproveitar o grande volume de dados de vídeo não anotados (Seção 4.2), permitindo que a YOLO aprenda representações mais ricas. Isso irá aumentar sua generalidade e mitigar o problema de diferenciar abelhas em aglomerados densos, conforme observado na Seção 7.2. Em paralelo, é necessário dar continuidade ao processo exaustivo de anotação (tracking via CVAT e eventos de entrada/saída via BORIS, Seções 5.1 e 5.2) para novos vídeos. Essa expansão do conjunto de dados anotado é vital para refinar a precisão tanto do detector quanto do rastreador.

12.2. Integração e Validação do Sistema Completo

Esta frente foca na funcionalidade do sistema de monitoramento em tempo real. A principal prioridade é a finalização da integração do Global Shutter (GS). Isso envolve a gravação e anotação dos primeiros vídeos obtidos com a câmera Innomaker IMX296 (Seção 6.2). A investigação da câmera Global Shutter deve provar sua eficácia na redução de ruídos de movimento (borrões, Fig. 13), fornecendo dados de maior qualidade para treinamento e avaliação. Além disso, é necessário concluir a integração do sistema embarcado com todo o pipeline de software (Detecção, Tracking e Heurística de Contagem). O objetivo é colocar o sistema em operação contínua e em tempo real, atendendo ao requisito de monitoramento ao vivo estabelecido no Objetivo Geral, Seção 2.1.

13. Disponibilidade do código

A implementação do método descito neste relatório está disponível em https://github.com/eliasmarts/FYP-bee-monitoring-model-training . Os resultados podem ser reproduzidos seguindo o que está escrito no README, utilizando o arquivo main_pipeline.py.
O dataset dos vídeos gravados e anotações realizadas está disponível em www.kaggle.com/datasets/eliassantosmartins/bee-tracking-dataset .

Referências

[1] Bilik, Simon, et al. "Computer vision approaches for automated bee counting application." IFAC-PapersOnLine 58.9 (2024): 43-48.

[2] Lei, Chaokai, et al. "A Honey Bee In-and-Out Counting Method Based on Multiple Object Tracking Algorithm." Insects 15.12 (2024): 974.

[3] Hunter, Gordon, et al. "Bee Counted–an Intelligent System for Counting Honeybees in Images and Videos." 2024 International Conference on Intelligent Environments (IE). IEEE, 2024.

[4] J. Redmon, S. Divvala, R. Girshick, A. Farhadi A, (2016) “You Only Look Once: Unified, Real-Time Object Detection”, IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016), Las Vegas, NV, USA, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[5] A. Olafanwa (2020) “Pixellib : A Library for Pixel-level Image and Video Segmentation”. https://pixellib.readthedocs.io/en/latest/

[6] H-K Cheng, A. G. Schwing (2022) “XMem : Long-Term Video Object Segmentation with an Atkinson-Shiffrin Memory Model”, in Proceedings of the European Conference on Computer Vision (ECCV 2022)

[7] Dickson, R. Thompson, et al. "Bee traffic estimation with YOLO and optical flow." SoutheastCon 2024. IEEE, 2024.

[8] Ciocarlan, Alina, et al. "Self-supervised learning for real-world object detection: a survey." arXiv preprint arXiv:2410.07442 (2024).

[9] T. Xiao, C. J. Reed, X. Wang, K. Keutzer, and T. Darrell, “Region similarity representation learning,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 10 539–10 548.

[10] K. Tian, Y. Jiang, C. Lin, L. Wang, Z. Yuan et al., “Designing bert for convolutional networks: Sparse and hierarchical masked modeling,” in The Eleventh International Conference on Learning Representations, 2022.

[11] Renan's Workspace. Jataí Bees Detection. Roboflow Universe, 2024. Disponível em: https://universe.roboflow.com/renans-workspace/jatai-bees-detection/dataset/16. Acesso em: 2 Dez. 2025.

[12] This dataset was originally created by [Jordan Bird, Leah Bird, Carrie Ijichi, Aurelie Jolivald, Salisu Wada, Kay Owa, Chloe Barnes] (https://universe.roboflow.com/jjb-object-detection-projects/bee-detection-pry0w) of Nottingham Trent University (United Kingdom).

[13] Noninvasive bee tracking in videos: deep learning algorithms and cloud platform design specifications. Dataset, 2021.

[14] Liang, A. (2024). Developing a multimodal system for bee object detection and health assessment. IEEE Access, 12, 158703 - 15871. https://doi.org/10.1109/ACCESS.2024.3464559.

[15] Friard, O., & Gamba, M. (2016). BORIS: a free, open-source event logging software for video/audio coding and live observations. Methods in Ecology and Evolution, 7(11), 1325-1331. Disponível em: https://www.boris.unito.it/

[16] CVAT.AI. CVAT: Computer Vision Annotation Tool. Versão 2.11.0. [S. l.], 2024. Disponível em: https://github.com/cvat-ai/cvat. Acesso em: 2 Dez. 2025.

[17] Raspberry Pi Foundation. Raspberry Pi 3 Model B+. [S. l.], 2018. Disponível em: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/. Acesso em: 2 Dez. 2025.

[18] Raspberry Pi Foundation. Raspberry Pi OS. [S. l.], 2024. Disponível em: https://www.raspberrypi.com/software/operating-systems/. Acesso em: 2 Dez. 2025.

[19] Cao, Jinkun, et al. "Observation-centric sort: Rethinking sort for robust multi-object tracking." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2023.

[20] K. Kasturi, et al., The CLEAR 2007 evaluation: Multi-object tracking. Em Proc. Int. Workshop on Video and Image Processing in Security (VIPS 2008), 2008.