

# TRABALHO FINAL DE GRADUAÇÃO – JULHO/2022

UNIVERSIDADE FEDERAL DE ITAJUBÁ  
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

## SISTEMA AUTOMÁTICO DE MONITORAMENTO DE ANIMAIS DOMÉSTICOS

**Patrícia Rosa**

Orientador: Prof. Mateus Gabriel Santos

Instituto de Engenharia de Sistemas e Tecnologia da Informação(IESTI)

**Resumo** - Este artigo apresenta o desenvolvimento prático de um protótipo de monitoramento de cães e gatos com a finalidade de disponibilizar vídeos desses animais domésticos para os tutores. Para tal protótipo, será utilizada uma câmera conectada a um *Raspberry Pi*, um programa que faz a detecção dos animais utilizando o módulo de deep neural networks da biblioteca *OpenCv* e faz a captura dos vídeos, um servidor em *Node Js* que recebe as filmagens e um aplicativo feito em *React Native* que disponibiliza os vídeos para o tutor através de dispositivos móveis. Ao final do projeto foi observado que a detecção dos animais teve alta precisão de acerto, e as postagens no servidor e *download* no aplicativo ocorreram como esperado. Algumas propostas de melhoria também serão discutidas no final do artigo.

**Palavras-Chave:** protótipo, monitoramento automático, reconhecimento de imagens, animais domésticos, *Raspberry Pi*, *OpenCv*, aplicativo, servidor

### I. INTRODUÇÃO

O Brasil é um dos países com maior número de animais domésticos do mundo, segundo a ABINPET, associação brasileira da indústria de produtos para animais de estimação. O Brasil tem a segunda maior população de cães, gatos e aves canoras e ornamentais em todo o mundo e é o terceiro maior país em população total de animais de estimação. O total é de 139,3 milhões de *pets*, o que demonstra a força potencial do setor na economia brasileira [1].

Vale ressaltar que, segundo uma pesquisa feita pelo COMAC, comissão de animais da companhia, o número de animais aumentou durante a pandemia, nesse período muitas pessoas passaram a trabalhar em casa e a adotar animais de estimação, grande porcentagem das vezes para ter mais companhia [2].

Com a flexibilização da pandemia, muitas pessoas retornaram a seus trabalhos presenciais e muitas vezes os

animais ficam sozinhos sem supervisão de seus tutores. Isso gera um problema para o animal que fica sozinho e pode precisar da ajuda de seu dono em situações de estresse do animal e necessidade de ajuda e para o tutor que fica preocupado com o bem estar de seu animal de estimação enquanto está fora de casa. Os trabalhos [3] e [4] abordam o assunto de animais domésticos que são deixados em casa sozinhos durante horas e que passam por complicações e estresse.

Assim, surge o interesse por ferramentas que permitam que o tutor consiga monitorar o seu animal de estimação à distância com o objetivo de garantir o bem estar dos animais, verificando se eles estão ativos no ambiente e o seu comportamento.

Com o objetivo de atender essa demanda, neste artigo será desenvolvido um protótipo de monitoramento de animais domésticos.

Para capturar imagens do ambiente será utilizada uma câmera conectada a um *Raspberry Pi*.

O *Raspberry Pi* é um computador portátil onde se pode instalar bibliotecas, rodar programas, adicionar periféricas como mouse, monitor, câmera e também periféricos para captura de dados como sensores. Sua função no projeto é receber imagens da câmera, executar o programa de detecção de cães e gatos. Dentre as diversas utilizações do *Raspberry Pi*, uma bastante interessante é o monitoramento de residências, como mostrado nos artigos [5], [6].

Pensando no desenvolvimento de um sistema otimizado, a gravação do ambiente apenas deverá ser feita quando o animal estiver presente. Para isso, deverão ser utilizadas técnicas de visão computacional que verificam se o animal está presente no ambiente e apenas nessa condição fazem a gravação do vídeo.

A visão computacional é uma área de estudo que tem como objetivo fazer com que as máquinas tenham a capacidade visual semelhante a do ser humano, ou seja,

conseguem reconhecer objetos, animais, e até mesmo diferenciar pessoas. A visão computacional comprehende os campos do processamento da imagem, modelos de reconhecimento, análises da cena, interpretação da cena, processamento de vídeo e a compreensão da imagem [7].

A visão computacional foi desenvolvida utilizando redes neurais profundas da biblioteca *OpenCv* com o objetivo de detectar cães e gatos em um ambiente. O *OpenCv* é "uma biblioteca de software de visão computacional e aprendizado de máquina de código aberto, construído para fornecer uma infraestrutura comum para aplicativos de visão computacional e acelerar o uso da percepção da máquina nos produtos comerciais". [8], [9]

O servidor foi feito com o *Node Js* que é um interpretador de JavaScript muito utilizado na criação de servidores. Sua função é receber os vídeos dos animais que foram gravados no *Raspberry Pi*, fazer o armazenamento desses vídeos, e disponibilizá-los para uma aplicação.

Para que o tutor possa acessar os vídeos dos animais foi escolhido o desenvolvimento de uma aplicação móvel devido a facilidade do acesso aos vídeos pelo tutor. O trabalho [10] argumenta sobre as vantagens em se utilizar aplicações em dispositivos móveis para os usuários. O aplicativo foi criado usando a biblioteca *React Native*, biblioteca *JavaScript* muito utilizada para desenvolver aplicativos para os sistemas *Android* e *iOS* de forma nativa.. O artigo[11], por exemplo, aborda um aplicativo de encomenda de medicamentos *online* utilizando a biblioteca *React Native*. Dessa forma temos uma solução completa e otimizada de monitoramento de animais domésticos e, o tutor poderá, de longe de sua residência, checar se seu animal de estimação está seguro.

Neste artigo é detalhado o passo a passo da construção do projeto, ele está organizado da seguinte maneira:

Na seção II temos a fundamentação teórica que apresenta os dispositivos e tecnologias utilizados no projeto. A seção III mostra o passo a passo do desenvolvimento do protótipo começando pela montagem do *hardware*, configurações iniciais feitas no *Raspberry Pi*, criação do programa em *Python* que faz a detecção dos animais e envia os vídeos para o servidor, a construção do servidor e por último a criação do aplicativo. Na seção IV temos os resultados e discussões do projeto, assim como propostas de melhorias em trabalhos futuros. Na seção V são apresentadas as conclusões sobre o projeto.

## II. FUNDAMENTAÇÃO TEÓRICA

Para criação do sistema de monitoramento foram utilizados diversos dispositivos e tecnologias, nessa seção serão listados e explicados rapidamente cada um deles.

### II.1 Raspberry Pi 3 B Plus

O Raspberry Pi modelo 3 B Plus, de dimensões 85mm x 56mm, lançado em 2018, é um computador portátil no qual é possível instalar um sistema operacional e diversos

programas necessários para uma aplicação, tem conexão com a *internet* e *bluetooth* e é possível conectar diversos dispositivos de hardware como a câmera, cabo de *internet*, fonte de energia, entre outros. Nesse projeto o *Raspberry Pi* será utilizado para receber as imagens de uma câmera, rodar um programa em *python* que fará a detecção dos animais no ambiente e também enviar os vídeos para um servidor.

Periféricos que deverão ser conectados ao *Raspberry Pi* para desenvolvimento do projeto:

#### II.1.1 MicroSD

O *Micro SD* é um cartão de memória portátil no qual deve ser instalado um sistema operacional, neste projeto foi instalado o *Raspberry Pi OS*, sistema operacional oficial da *Raspberry Pi*.

O *MicroSD* escolhido foi da marca *SanDisk Ultra* de 32GB de memória, pois é um modelo de bom custo benefício.

#### II.1.2 Câmera

A câmera utilizada foi a oficial da *Raspberry Pi* de 5MP. A conexão é feita através da porta CSI (*camera serial interface*). Ela possui lente de foco fixo, capaz de fornecer resolução de 2592 x 1944 pixels para imagens estáticas e 1080p30, 720p60 e 640x480p60/90 para vídeos. A câmera será utilizada para capturar imagens do ambiente que o tutor preferir.

#### II.2 Python

O *Python* é uma linguagem de programação de alto nível que permite trabalhar rapidamente e integrar os sistemas de forma mais eficaz.

O programa de detecção dos animais foi feito utilizando *Python* devido ao grande uso dessa linguagem em aplicações com técnicas de *Machine Learning* que são utilizadas no projeto.

#### II.3 OpenCv

O *OpenCv* é uma biblioteca de código aberto que contém diversos algoritmos de visão computacional e aprendizado de máquina. Essa biblioteca pode processar imagens e vídeos para detectar objetos em tempo real.

Métodos de *deep learning* são muito utilizados na detecção de objetos em tempo real e em movimento, essa área de estudo tem recebido bastante atenção e estão disponibilizadas em bibliotecas de forma pré-treinada, tornando a sua aplicação em projetos com relativa facilidade.

Nesse artigo, para detecção dos animais do *OpenCv* em movimento será utilizado o módulo de redes neurais do *OpenCv*. Os trabalhos [12] e [13] abordam o uso de redes neurais e do *OpenCv* na detecção de objetos.

O módulo de redes neurais do *OpenCv* possui diversas classes relacionadas à construção de redes neurais profundas, nesse artigo será utilizada a classe *DetectionModel*. Essa classe cria uma rede neural profunda e deve receber como parâmetros os pesos e configurações para processamento da imagem de entrada, define a entrada de pré-processamento, executa a passagem direta e retorna as detecções de resultados. Para esse modelo de detecção, as topologias *SSD*, *Faster R-CNN* e *YOLO* são suportadas [14].

#### II.4 Coco

*COCO* significa Objetos Comuns em Contexto, é um conjunto de dados de detecção, segmentação e rotulagem de objetos em grande escala.

Nesse projeto será utilizado a rede de detecção de objetos o *OpenCv* usando o conjunto de dados COCO para detectar animais nos vídeos.

#### II.5 Node JS

O *Node Js* é um *software* de código aberto, multiplataforma, que permite a execução de códigos *JavaScript*. É um *software* muito utilizado na construção de servidores. [15]

O *Node Js* será utilizado na construção do servidor que recebe os vídeos vindos do *Raspberry Pi* e envia os vídeos para a aplicação.

#### II.6 Express

O *Express* é um framework para aplicativo da web do *Node.js* mínimo e flexível que fornece um conjunto robusto de recursos para construção de servidores web. Possui métodos utilitários *http* e middleware que permitem criar aplicações robustas facilmente. [15]

#### II.7 React Native

*React Native* é uma biblioteca *Javascript* criada pelo *Facebook*. É usada para desenvolver aplicativos para os sistemas *Android* e *iOS* de forma nativa. Essa biblioteca será utilizada para construir o aplicativo pelo qual o tutor verá os vídeos.

Para a reprodução será utilizada a biblioteca *React Native Video*, ela é um pacote que deve ser instalado no projeto e fornece um componente de vídeo para o qual deve ser passada a fonte e a estilização do vídeo, como largura e altura.

#### II.8 Axios

O *Axios* é uma biblioteca de *client HTTP* que permite fazer solicitações para determinada rota. Ele será utilizado no aplicativo para fazer as requisições para o servidor.

### III. PROJETO

O projeto é um protótipo de um dispositivo que faz o monitoramento de animais domésticos em determinado local de uma residência. O dispositivo pode ser colocado em algum cômodo da casa que os animais fiquem com mais frequência, como uma sala, corredor ou quintal.

O dispositivo é formado por uma câmera que captura as imagens do ambiente e um *Raspberry Pi* responsável pelo processamento das imagens e detecção dos animais domésticos.

Quando o animal é detectado, o vídeo é enviado para um servidor local que o armazena. O tutor pode acessar o aplicativo a qualquer momento e acessar todos os vídeos do servidor. No projeto, o servidor foi feito em uma rede doméstica para prova de conceito, para uso fora da residência o servidor deverá ser enviado para nuvem ou um servidor físico com uma porta de comunicação aberta.

A seguir será detalhado cada etapa da construção do projeto.

#### III.1. Preparação do MicroSD

No microSD deve ser instalado um sistema operacional, o escolhido foi o *Raspberry Pi OS*, sistema operacional oficial da *Raspberry* que é recomendado para a maioria dos usuários.

#### III.2. Montagem do hardware

Para montagem do protótipo devem ser conectados todos os periféricos listados no item II.1 (fonte de alimentação, microventilador cooler, câmera e microSD), além de mouse, teclado e monitor para que seja possível desenvolver e testar o programa que será criado no *Raspberry Pi*.

#### III.3. Configuração do Raspberry Pi

Após a montagem do *hardware* é possível ligar o *Raspberry Pi* para que sejam feitas as configurações iniciais. As configurações são duas, habilitar a *LegacyCamera* e instalar o *OpenCv*, e são detalhadas a seguir:

##### III.3.1. Habilitar LegacyCamera

A *LegacyCamera* deve ser habilitada para que a câmera conectada no *Raspberry Pi* seja detectada, para isso deve-se ir nas configurações de conexões de periféricos do *Raspberry Pi* e habilitar a *LegacyCamera*. Para acessar as configurações, o seguinte comando é utilizado:

*sudo raspi-config*

O comando abrirá uma tela de configurações do *Raspberry*, apresentada na figura 3:

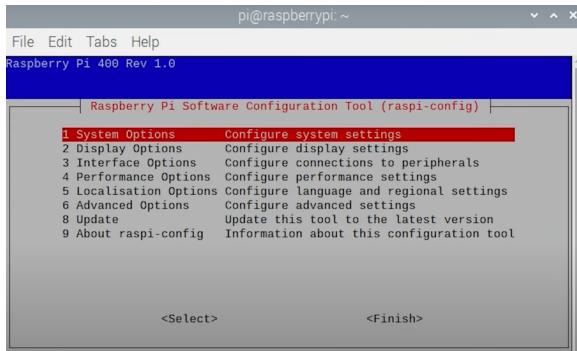


Fig. 3 - Configuração do Raspberry Pi

Deve ser selecionada a opção 3 "Interface Options" que diz respeito às configurações dos periféricos. Ao clicar nessa opção será disponibilizada a tela de configuração dos periféricos, indicada na figura 4. A opção um, "Legacy Camera" deverá ser habilitada.

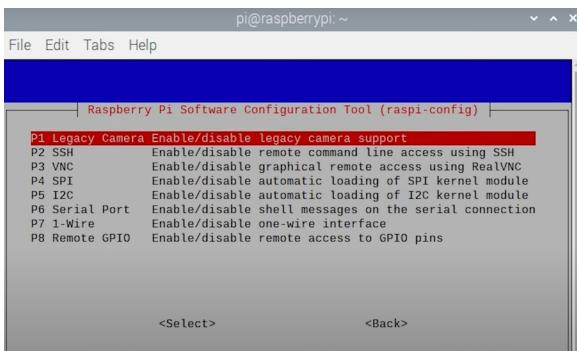


Fig. 4 - Configuração dos periféricos

### III.3.2. Instalar OpenCv

Para detecção de imagem será utilizada a biblioteca *OpenCv*. A versão a ser instalada do *OpenCv* é a 4.5.5 e pode ser feita seguindo a documentação oficial do *OpenCv*.

### III.4 Criação do programa em Python

Com a montagem do protótipo e configurações iniciais feitas é possível usar os vídeos vindo da câmera no *Raspberry Pi* para identificar os animais.

Para a implementação do programa foi utilizada a linguagem de programação *Python*, essa linguagem foi escolhida pelo seu grande uso no emprego de técnicas de *Machine Learning* e a simplicidade de sua aplicação. O *Python* por padrão vem instalado no *Raspberry Pi OS*. Deve ser criada uma pasta onde ficarão todos os arquivos do projeto de detecção dos animais. Nessa pasta deverá ser criado um arquivo chamado *object-identifier* com extensão *.py* que identifica um arquivo *Python*.

Para detecção dos animais no programa *object-identifier.py* serão utilizadas técnicas de detecção de objetos do *OpenCv*.

O carregamento da biblioteca *OpenCv* no programa *Python* é feito através do comando:

```
import cv2
```

Para detecção de imagem foi utilizada a classe *DetectionModel* do *OpenCv*. Para esta classe é passado o peso da rede neural, que é um arquivo binário que contém pesos pré treinados e a configuração, arquivo de texto que contém as configurações de arquitetura dessa rede. Os arquivos de peso e configuração foram copiados de um artigo que faz a detecção de animais utilizando módulo *DetectionModel* do *OpenCv* [16]. A chamada e configuração do módulo de detecção é feita conforme apresentada na figura 5:

```
net = cv2.dnn_DetectionModel(weights, config)
```

Fig. 5 - Configuração do modelo de detecção

Deve ser criada uma função que detecta os objetos na filmagem, ela deve receber como parâmetros a imagem da câmera, limite mínimo de confiança nas imagens, limite de objetos detectados simultaneamente, e os objetos que serão detectados. A função retornará uma variável chamada *objectCaptured* com valor *True* caso o gato ou cachorro tenha sido detectado e com valor *False* caso nenhum animal tenha sido detectado. Essa função é apresentada na figura 6:

```
def getObjects(
    img,
    thres,
    nms,
    draw=True,
    objects=[],
):
    objectCaptured = False
    (classNames, confs, bbox) = net.detect(img, confThreshold=thres,
                                         nmsThreshold=nms)

    if len(objects) == 0:
        objects = classNames
    objectInfo = []
    if len(classNames) != 0:
        for (classId, confidence, box) in zip(classNames.flatten(),
                                              confs.flatten(), bbox):
            className = classNames[classId - 1]
            if className in objects:
                objectInfo.append([box, className])

    if objectInfo == []:
        objectCaptured = False
    if objectInfo != []:
        objectCaptured = True

    return (img, objectCaptured)
```

Fig. 6 - Função que verifica se o animal foi capturado

Declaradas as funções e módulos necessários, será declarado o módulo principal do programa onde começa a captura de vídeo, detecção dos animais, gravação do vídeo e o envio para o servidor.

Para capturar a imagem da câmera deve ser usado a função *VideoCapture(index)* do *OpenCv*, sendo o index o

número que representa a câmera, para apenas uma câmera o *index* utilizado é 0.

Antes de começar a leitura dos frames também será necessário declarar uma variável de controle de gravação que foi nomeada "*recording*" que inicialmente recebe o valor *False*. Essa implementação é mostrada na figura 7 a seguir:

```
if __name__ == '__main__':
    cap = cv2.VideoCapture(0)
    recording = False
```

Fig. 7 - Chamada da captura de vídeo da câmera e criação de variável de controle "recording"

A seguir deverá ser criado um *loop* onde sempre ocorrerá a leitura das imagens usando o método *.read()* no vídeo que estará sendo capturado. Além disso, também deverá ocorrer a detecção dos animais nas imagens utilizando a função *getObjects* explicada e declarada anteriormente. Para essa função deve ser passada a imagem, um limite mínimo de confiança de 0.45, um limite de detecção de objetos de 0.2 e um objeto que contém os objetos a serem detectados, que será um *array* de cães e gatos. O *loop* com a captura das imagens e detecção dos objetos nessas imagens é mostrado na figura 8:

```
while True:
    (success, img) = cap.read()
    (result, objectCaptured) = getObjects(img, 0.45, 0.2,
                                          objects=['cat', 'dog'])
```

Fig. 8 - Chamada da função que captura os objetos nas imagens

Quando a variável "*objectCaptured*" retornada pela função *getObjects* tiver valor *True*, e "*recoding*" tiver valor *False* significa que o animal foi capturado mas a filmagem ainda não começou, nesse caso deve ser instanciada a gravação do vídeo utilizando a função *VideoWriter()* do *OpenCv*[17], passando como parâmetros o nome do arquivo de vídeo que será criado (no projeto foi chamado "*captured 8.mp4*"), o *fourcc* que é uma sequência de quatro bytes usada para identificar exclusivamente os formatos de dados, o número de *frames* por segundo (para teste foram utilizados 5 frames por segundo) e o tamanho da imagem que deve ser o mesmo tamanho do vídeo capturado. O *fourcc* utilizado foi o "*mp4v*" usado para criar arquivos de extensão *.mp4*, esse formato de vídeo foi escolhido por ser o formato suportado pelo componente de vídeo utilizado na aplicação *React Native*. O arquivo de vídeo "*captured8.mp4*" deverá ser criado no diretório do projeto. Na figura 9 é apresentada essa condição:

```
if objectCaptured and recording == False:
    frame_width = int(cap.get(3))
    frame_height = int(cap.get(4))
    size = (frame_width, frame_height)
    fourcc = cv2.VideoWriter_fourcc('m', 'p', '4', 'v')
    videoWriter = cv2.VideoWriter('captured8.mp4', fourcc, 5,
                                 size)
```

Fig. 9 - Condição de criação da instância de gravação vídeo

Quando o animal é capturado e a instância de vídeo é criada, a variável de controle "*recording*" deve receber o valor *True* e deve iniciar a montagem do vídeo, para isso foi utilizado o método *.write()* passando a imagem como parâmetro. Essa condição é ilustrada na figura 10:

```
if objectCaptured:
    recording = True
    videoWriter.write(img)
```

Fig. 10 - Condição que faz a montagem do vídeo

Quando o animal sumir de cena e a gravação estiver ocorrendo, deverá ser chamado o método *.release()* que encerra a instância do vídeo.

Nesse momento deve ser realizada uma requisição *http* de método *post* enviando o vídeo que acabou de ser gravado para o servidor. Para isso foi utilizada a biblioteca *requests*, essa é uma biblioteca *http* para linguagem *Python*.

A variável "recording" deverá receber o valor *False* dado que a filmagem finalizou. Essa condição é ilustrada na figura 11 a seguir:

```
if objectCaptured == False and recording == True:
    videoWriter.release()
    req = requests.post('http://192.168.18.2:8080/videos/upload',
                        files={'foto': open('captured8.mp4',
                                             'rb')})
    recording = False
```

Fig. 11 - Finalização da instância de gravação e envio do vídeo para o servidor

No conjunto de gráficos da figura 12 é mostrado um resumo de como a captura do animal, a gravação do vídeo e envio do mesmo se comportam através do tempo. Quando o objeto é capturado é iniciada a gravação do vídeo. Quando o objeto desejado não é mais capturado, a gravação é finalizada e ocorre o envio do vídeo.

A figura 13 apresenta o diagrama de fluxo do programa de detecção para melhor entendimento do programa.

### III.5 Servidor

O servidor foi criado com o *NodeJs*. Para criar um servidor *http* foi usado o *framework Express*.

O servidor terá três funções:

*1) Upload do vídeo:* O servidor deve disponibilizar uma forma do *Raspberry Pi* enviar o vídeo gravado para que o servidor armazene o vídeo, ou seja, o *upload* do arquivo. Para realizar o *upload* foi usada a biblioteca *Multer* [18], esta biblioteca é um intermediário responsável pela manipulação de dados *multipart/form-data*.

A biblioteca *Multer* deve ser carregada e seu método *multer* deve ser chamado passando dois parâmetros: a

pasta de destino do arquivo e o nome do arquivo, que deverá ser a data e horário em que o arquivo chegou no servidor. Com as configurações de *upload* de arquivos feitas, deve ser criada uma rota do tipo *post* para que o servidor receba o arquivo.

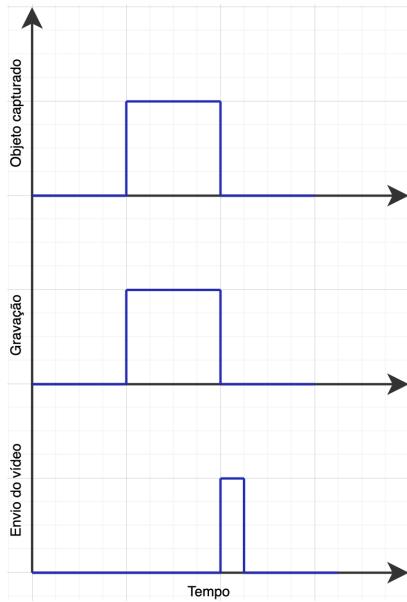


Fig. 12 - Gráfico de Objeto capturado, Gravação e Envio de vídeo x Tempo

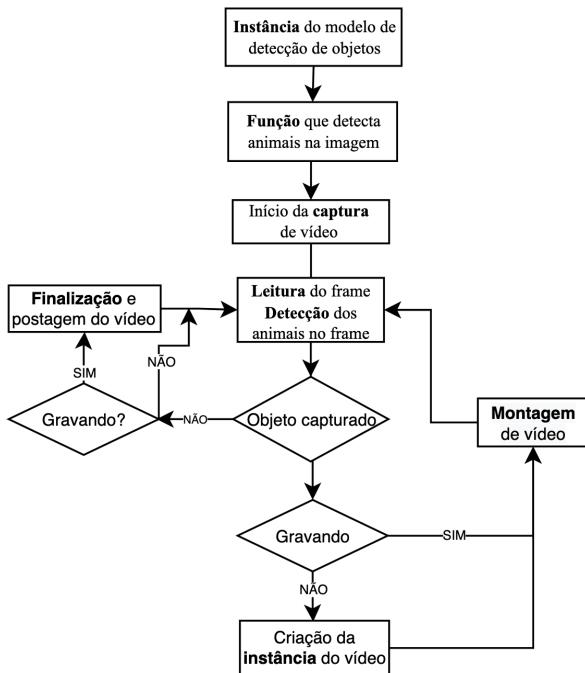


Fig. 13 - Diagrama da gravação de vídeos de animais domésticos

2) *Listar os vídeos disponíveis no servidor:* Deve ser criada uma rota do tipo *get* para que o servidor envie o nome dos vídeos disponíveis.

Para capturar o nome dos arquivos na pasta uploads foi utilizado o módulo *fs* do *Node.js*, esse módulo disponibiliza diversas funcionalidades úteis para acessar e interagir com os arquivos do sistema, a funcionalidade de interesse será a *readdir()* [19] para o qual iremos passar o diretório em que a pasta uploads está situada.

3) *Download do arquivo:* O servidor deve disponibilizar os vídeos que foram armazenados, para isso deve ser criada uma rota do tipo *get* que recebe um identificador que é o nome do vídeo que se deseja fazer o *download*.

### III.6 Aplicativo

A aplicação será criada utilizando a biblioteca React Native. Ela é usada para desenvolver aplicativos para os sistemas Android e iOS de forma nativa, porém será desenvolvido apenas para Android neste artigo.

É necessário a instalação do *Android Studio*, um ambiente de desenvolvimento integrado para desenvolver para a plataforma Android.

O aplicativo possui 3 telas que são apresentadas a seguir:

1) *Tela inicial:* A tela inicial apresentada na figura 14 contextualiza o aplicativo e possui um botão que faz a navegação para a tela de listagem de vídeos. A imagem utilizada na tela foi encontrada no site *flaticon*. [20]



Fig. 14 - Tela Inicial do aplicativo

2) *Tela de listagem de vídeos:* Ao entrar na tela de listagem de vídeo será feita a requisição que pega a lista dos nomes dos vídeos disponíveis no servidor. Para fazer essa requisição será utilizado o Axios, um cliente *HTTP* baseado em promessas para o *node.js* e para o navegador detalhado na documentação oficial do *axios* [21].

A requisição para pegar os vídeos será feita utilizando o método *get* do *Axios* passando como parâmetro apenas a

*url* que contém os vídeos. Nesse projeto será utilizado o *ip* local.

A tela de listagem de vídeos é apresentada na figura 15. Ao clicar em um dos vídeos ocorrerá a navegação para a tela que reproduz o vídeo:

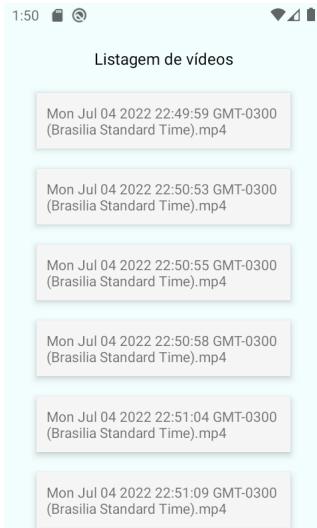


Fig. 15 - Tela de listagem de vídeos do aplicativo

3) *Tela de reprodução de vídeo:* Para reproduzir o vídeo no React Native será utilizado um componente de vídeo *react-native-video* [22].

A tela de reprodução do vídeo é apresentada na figura 16. A implementação desse aplicativo está disponível no *github* da autora do artigo. [23]

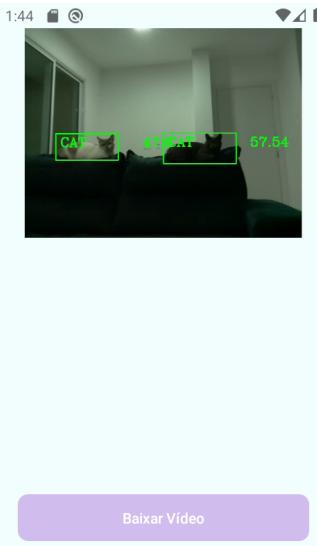


Fig. 16 - Tela de reprodução de vídeo do aplicativo

#### IV. RESULTADOS E DISCUSSÕES

Com o dispositivo funcionando em campo, foi possível obter resultados, discussões e melhorias para o projeto.

O dispositivo pode ser colocado em um ambiente em que os animais domésticos aparecem com mais frequência ou algum lugar de preferência do tutor. A seguir são apresentadas algumas imagens de teste.

Na figura 17 estão algumas imagens da detecção de um gato a uma mesma distância da câmera, com confiabilidade de 53.69% e 76.35 respectivamente:

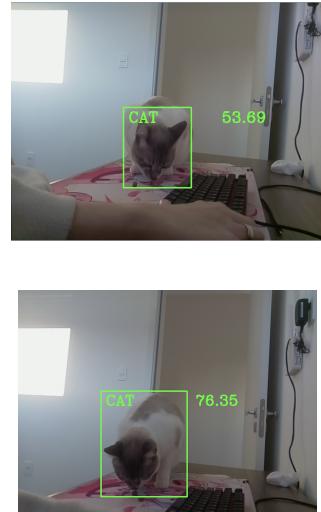


Fig. 17 - Detecção do gato a uma mesma distância

Com o afastamento do gato da câmara houveram erros na detecção dos gatos que às vezes são detectados como cachorros. Um desses casos é apresentado na figura 18, com uma confiabilidade de 57.64%. Essa condição é mostrada na figura 18:

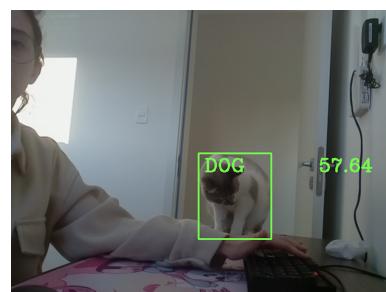


Fig. 18 - Gato sendo detectado como um cachorro

Por meio dos testes foi possível perceber que a proximidade do animal com a câmera melhora a diferenciação entre gato e cachorro.

Foram realizados mais testes com um segundo gato, podendo verificar que no mesmo *frame* ele foi detectado como um gato e cachorro, com confiabilidade de 61% e

64% respectivamente. Os resultados desse teste aparecem na figura 19.

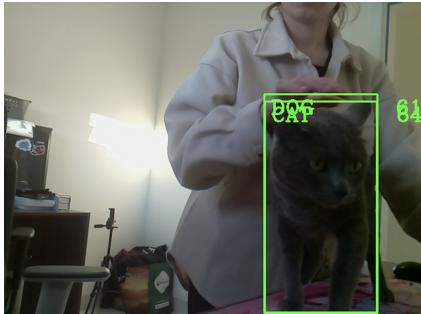


Fig. 19 - Gato sendo detectado como um cachorro

A figura 20 mostra o teste de detecção feito com um cachorro, em 9 frames o cachorro foi detectado como um cachorro e não houve confusão com gatos, a seguir são mostrados 4 frames, tendo graus de confiança de 76.99%, 84.56%, 47.0% e 47.89 :

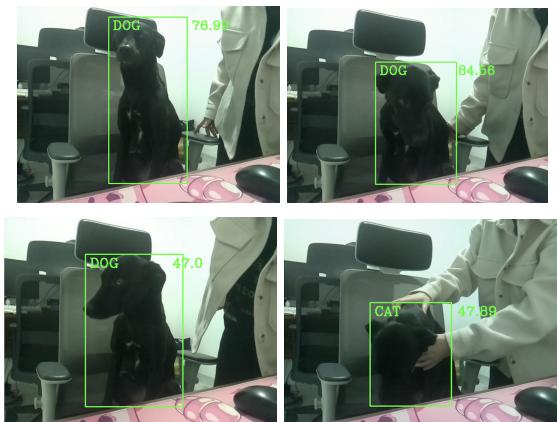


Fig. 20 - Detecção de um cachorro

Fazendo a análise de 50 vídeos, foi possível verificar os acertos e erros na detecção de cães e gatos. Nas figuras 21 e 22 temos os gráficos de erros e acertos na detecção de cachorro e gato respectivamente

Para esse protótipo não há grandes problemas na confusão entre cães e gatos, pois o objetivo é detectar um dos animais e gravar o vídeo sem a necessidade de diferenciação deles. Mas em trabalhos futuros e para escalabilidade do produto é indicado criar uma rede neural mais ajustada com um treinamento específico para cães e gatos. Durante os testes foi possível observar que os vídeos apresentaram 5 frames por segundo como configurado na criação do vídeo.

### Teste de detecção de cachorro

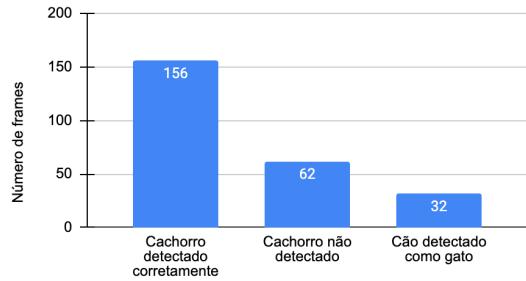


Fig. 21 - Gráfico de acertos e erros na detecção de cachorro

### Teste de detecção de gato

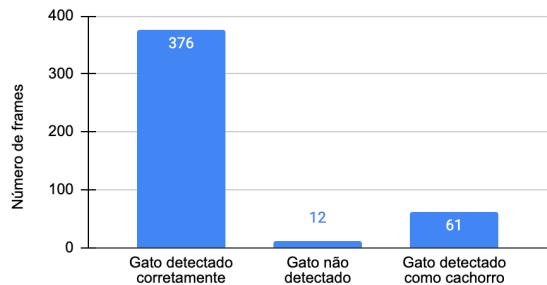


Fig. 22 - Gráfico de acertos e erros na detecção de gato

Os vídeos dos animais foram gravados e enviados para o servidor com sucesso em todas as vezes. Em três dias de teste foram gravados em média 100 vídeos por dia, todos contendo imagens dos animais.

Com o aplicativo foi possível buscar a lista de vídeos e reproduzir os vídeos com sucesso.

Nos testes foi possível detectar um cenário que ocorre diversas vezes e que não traz um resultado otimizado para o projeto. O cenário ocorre quando o animal está em frente à câmera sendo detectado em determinado número de frames contínuos, e pode acontecer de a detecção falhar em um único frame, e logo em seguida voltar a detecção. O resultado desse cenário são dois vídeos mais curtos devido a falha de detecção em apenas um único frame. Para vídeos mais longos e consistentes, seria interessante introduzir um contador de frames que, ao perceber que ocorreu a perda de detecção do animal em apenas um ou dois frames, continua a filmagem normalmente para verificar se ocorre o retorno da detecção. Caso não ocorra o retorno da detecção o vídeo deverá ser finalizado.

Dentro do projeto foi utilizado um servidor local para prova de conceito, para utilizar o projeto fora da rede doméstica é necessário utilizar um servidor físico com uma porta de comunicação aberta ou um servidor na AWS ou Google Cloud.

O protótipo desenvolvido é um mínimo produto viável para o monitoramento dos animais domésticos, para um sistema escalável é necessário pesquisar equipamentos mais baratos que garantam o funcionamento e qualidade do monitoramento.

## V. CONCLUSÃO

Esse trabalho apresenta importantes contribuições tecnológicas na integração de equipamentos e sistemas como *Raspberry Pi*, *OpenCv*, servidor e aplicativo para o desenvolvimento de um protótipo de monitoramento de animais domésticos.

Fazendo uso de técnicas de visão computacional e aprendizado de máquina é possível detectar quando os animais estão presentes no ambiente. Para isso foi utilizado uma técnica de detecção de objetos do OpenCv e uma base de dados que permite a detecção de cães e gatos.

As técnicas de visão computacional e aprendizado de máquina estão cada vez mais elaboradas e permitem a existência de projetos cujas máquinas tenham percepções semelhantes às dos seres humanos, reconhecendo padrões, identificando objetos, e também aprendendo a reconhecer novos padrões. Por fim, a técnica utilizada no protótipo atendeu bem a sua função no projeto de detectar cães e gatos, mesmo havendo uma confusão entre esses animais em alguns momentos.

Para que fosse possível rodar esse programa que faz a detecção dos animais em qualquer ambiente foi utilizado o *Raspberry Pi*, um mini computador portátil, que atendeu os requisitos e teve uma ótima performance no protótipo, executando o programa de detecção por vários dias sem apresentar problemas.

O servidor utilizado foi criado em *Node Js* e teve três funções principais: *upload* e *download* de vídeo e listagem dos vídeos disponíveis. No servidor foi possível utilizar bibliotecas e frameworks que atenderam os três pré-requisitos do servidor listados anteriormente e atenderam todas expectativas, recebendo os vídeos, armazenando-os e enviando-os assim que requisitado.

A segunda aplicação para que fosse possível ao tutor acessar os vídeos foi um aplicativo de celular. Com a popularização dos celulares e melhoria de sua performance em questão de espaço, processamento, entre outros, a adesão a aplicativos de celular cresceu muito e é um meio de comunicação com o usuário que atendeu os requisitos do projeto. O aplicativo foi desenvolvido em *React Native* e permite a criação do aplicativo para android e ios. O aplicativo tem duas funções principais: listar os vídeos disponíveis e reproduzir o vídeo selecionado pelo tutor. O aplicativo performou como esperado, fazendo requisições dos vídeos para o servidor e reproduzindo os vídeos normalmente.

Dessa forma conclui-se que a implementação do sistema de monitoramento de animais, com a finalidade de manter o tutor atualizado do bem estar de seus animais, possibilita um monitoramento de 24 horas do animal, otimizado por técnicas de visão computacional, visando principalmente o cuidado com o animal .

Em relação a trabalhos futuros é interessante estudar uma maneira de tornar o sistema escalável, com produtos mais baratos que mantenham a qualidade do monitoramento. Para isso podemos usar o Edge Impulse, plataforma que permite colocar aprendizado de máquina em produtos reais e escaláveis, tornando possível rodar algoritmos sofisticados de aprendizado de máquina nos mais pequenos microcontroladores [24].

Ainda em relação a trabalhos futuros é interessante pesquisar métodos de detecção de cães e gatos que tenham um grau de acerto e detecção melhores.

## VI. AGRADECIMENTOS

Gostaria de agradecer ao meu namorado Diogo que disponibilizou grande parte dos componentes de hardware utilizados no trabalho e por apoiar meu crescimento na área de tecnologia. Gostaria de agradecer também minha Mãe Maria Célia e meu pai Renato que desde cedo me ensinaram o valor e a importância dos estudos durante a escola e a graduação. Ao meu orientador Mateus Gabriel Santos gostaria de agradecer por todo suporte e ensinamentos passados durante o desenvolvimento deste trabalho.

Por fim gostaria de agradecer a todo o corpo docente da Universidade Federal de Itajubá pelo conhecimento transmitido e pela formação da minha base de conhecimento de tecnologia.

## VII. REFERÊNCIAS

- [1] Associação Brasileira da Indústria de Produtos para animais de estimação - Informações gerais do setor Pet. Disponível em: <[http://abinpet.org.br/infos\\_gerais/](http://abinpet.org.br/infos_gerais/)>. Acesso em: 20 de jun. 2022
- [2] Comissão animais de companhia (COMAC) - Mercado Pet da Pandemia. Disponível em: <<https://www.sindan.org.br/wp-content/uploads/2021/07/Apresentacao-Radar-2021-Coletiva-de-Imprensa-1.pdf>>. Acesso em: 20 de jun. 2022
- [3] Gerrit Stephan, Joachim Leidhold, Kurt Hammerschmidt, "Pet dogs home alone: A video-based study", Applied Animal Behaviour Science, Volume 244, 2021
- [4] E. Scaglia, S. Cannas, M. Minero, D. Frank, A. Bassi, C. Palestini, "Video analysis of adult dogs when left home alone", Journal of Veterinary Behavior, Volume 8, Issue 6, 2013

- [5] I Gusti Made Ngurah Desnanjay - "Home security monitoring system with IoT-based Raspberry Pi", In: Indonesian Journal of Electrical Engineering and Computer Science
- [6] V. Patchava, H. B. Kandala and P. R. Babu- "A Smart Home Automation technique with Raspberry Pi using IoT," 2015 International Conference on Smart Sensors and Systems (IC-SSS), 2015, pp. 1-4, doi: 10.1109/SMARTSENS.2015.7873584.
- [7] Nilton Pinto Ribeiro Filho - VISÃO COMPUTACIONAL: UM NOVO CAMPO DE PESQUISA EM COGNIÇÃO VISUAL. Acesso em: 01 de mai. 2022
- [8] OpenCv - OpenCv. Disponível em: <<https://opencv.org/about>>. Acesso em: 01 de mai. 2022
- [9] Chandan G, Ayush Jain, Harsh Jain, Mohana, "Real Time Object Detection and Tracking Using Deep Learning and OpenCV ", Proceedings of the International Conference on Inventive Research in Computing Applications (ICIRCA 2018) IEEE
- [10] Fatih Nayebi, Jean-Marc Desharnais, Alain Abran, "The State of the Art of Mobile Application Usability Evaluation", In: IEEE, Canadian Conference on Electrical and Computer Engineering (CCECE), 2012
- [11] Kamali Gupta, Gupta Deepali, "React Native Application Development", March 2019
- [12] Chandan G, Ayush Jain, Harsh Jain, Mohana, "Real Time Object Detection and Tracking Using Deep Learning and OpenCV", 2018 International Conference on Inventive Research in Computing Applications (ICIRCA)
- [13] J. Vieira Filho, P. A. Jaskowiak , "Comparação de Métodos de Deep Learning Pré-Treinados da Biblioteca OpenCV para Detecção de Pessoas em Ambientes Internos", v. 18 n. 4 (2020): Revista Eletrônica de Iniciação Científica em Computação
- [14] OpenCV team. Open Source Computer Vision. Disponível em: <[https://docs.opencv.org/4.6.0/d3/dfl/classev\\_1\\_1dnn\\_1\\_1DetectionModel.html](https://docs.opencv.org/4.6.0/d3/dfl/classev_1_1dnn_1_1DetectionModel.html)>. Acesso em: 29 de mai. 2022
- [15] StrongLoop, Inc., e outros contribuidores do expressjs.com - Fundação Node.js. Express 4.x. API Reference. Disponível em: <<https://expressjs.com/en/4x/api.html>>. Acesso em: 24 de jun. 2022
- [16] Core Electronics - Object and Animal Recognition With Raspberry Pi and OpenCV. Disponível em: <<https://core-electronics.com.au/guides/object-identify-raspberry-pi/>>. Acesso em: 24 de jun. 2022
- [17] OpenCv - Vídeo Writer. Disponível em: <[https://docs.opencv.org/3.4/dd/d9e/classcv\\_1\\_1VideoWriter.html#ad59c61d8881ba2b2da22cff5487465b5](https://docs.opencv.org/3.4/dd/d9e/classcv_1_1VideoWriter.html#ad59c61d8881ba2b2da22cff5487465b5)>. Acesso em: 24 de jun. 2022.
- [18] Upload de arquivos e imagens utilizando o Multer - express - Node.js. Disponível em: <<https://consolelog.com.br/upload-de-arquivos-imagens-utilizando-multer-express-nodejs/>>. Acesso em: 24 de jun. 2022.
- [19] VithalReddy - Get List of all files in a directory in Node.js. Disponível em: <<https://medium.com/stackfame/get-list-of-all-files-in-a-directory-in-node-js-befd31677ec5>> pag., Jan. 1993. Acesso em 24 de jun. 2022
- [20] noomtah - Pet care icons. Disponível em: <<https://www.flaticon.com/free-icons/pet-care>>. Acesso em: 24 jun. 2022.
- [21] 2014-present Matt Zabriskie & Collaborators - Axios. Disponível em: <<https://axios-http.com/ptbr/docs/intro>>. Acesso em: 24 de jun. 2022.
- [22] Brent Vatne - react-native-video. Disponível em: <<https://github.com/react-native-video/react-native-video>>. Acesso em: 24 de jun. 2022.
- [23] Patrícia Rosa - petsitterapp. Disponível em: <<https://github.com/pati-rosa/petsitterapp>>
- [24] Gitbook - EdgeImpulse. Disponível em: <<https://docs.edgeimpulse.com/docs/what-is-embedded-machine-learning-anyway>>

## BIOGRAFIA:

### **Patrícia Rosa**

Nasceu em São Bento do Sapucaí (SP) em 1995. Em 2014 ingressou na Universidade Federal de Itajubá - Campus Itabira, no curso de Engenharia Elétrica, mudando futuramente para o Campus Itajubá, no curso de Engenharia de Controle e Automação, onde participou do MakerSpace durante o ano de 2019 atuando na prototipagem de projetos de hardware, robótica, eletrônica e conectividade (IoT).