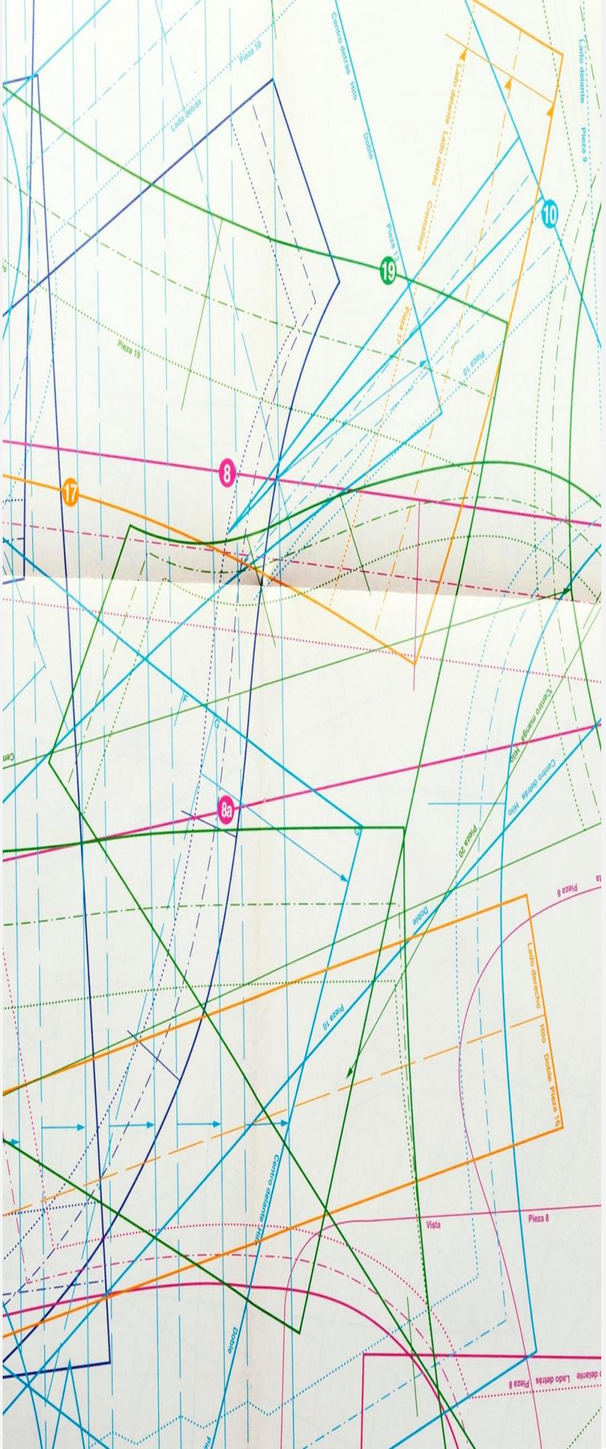
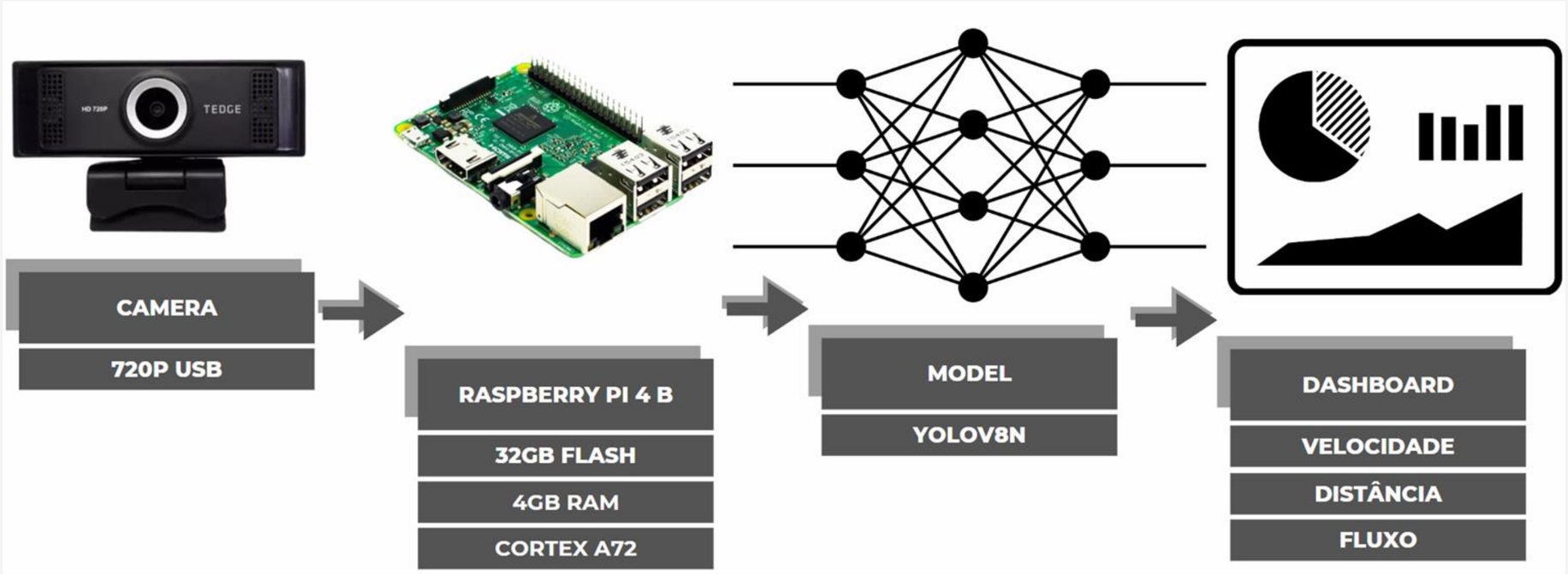
EdgeML aplicado ao Rastreamento de veículos e controle de tráfego

Hyago Vieira Leme Barbosa Silva
hyago.silva@mtel.inatel.br



Dispositivos e arquitetura da solução



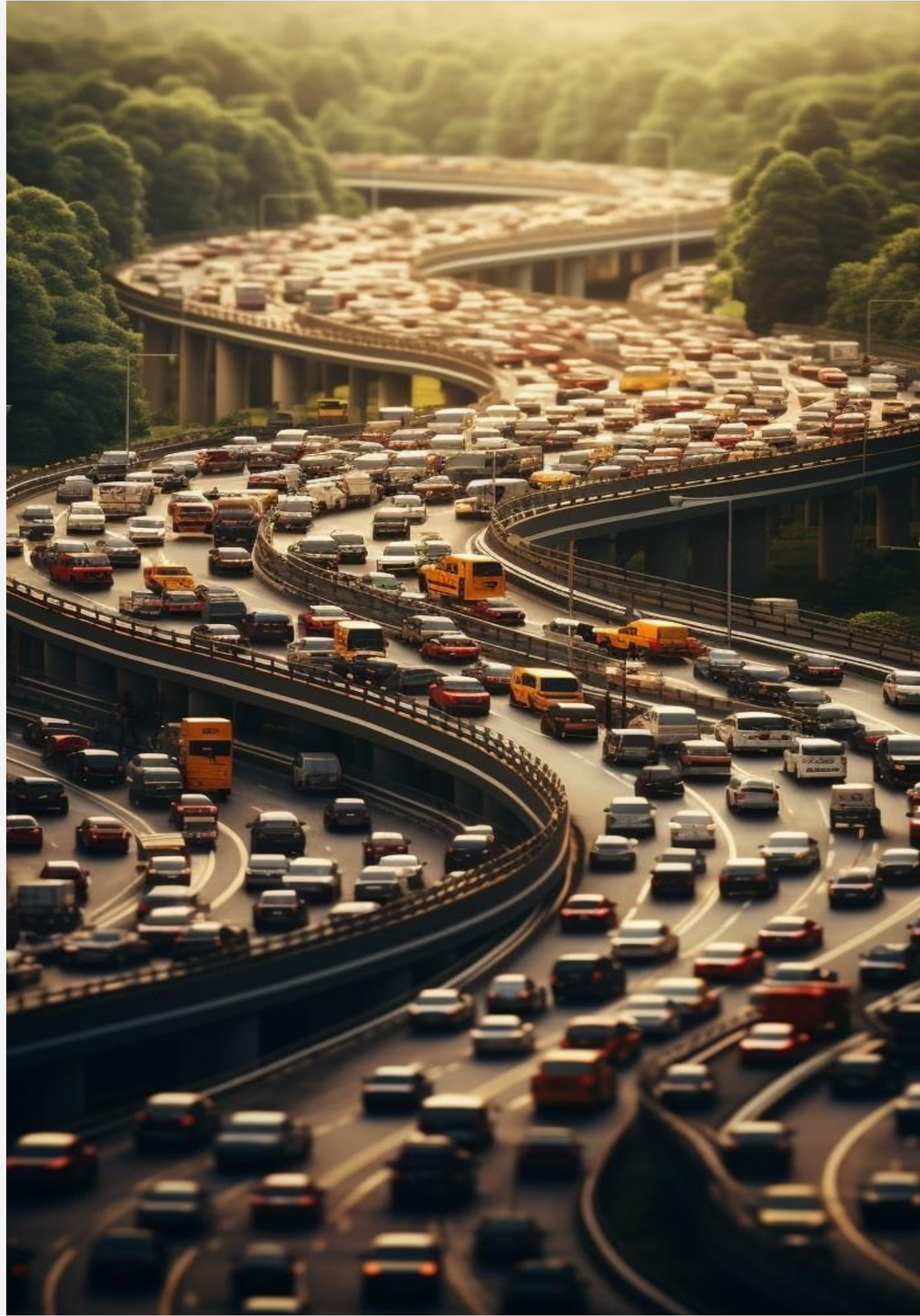
Objetivos do projeto

Contexto:

A necessidade de **sistemas com controle** e **análise** de dados eficientes de **rastreamento de veículos** para **logística**, **segurança** e **gestão** de forma automática, no tráfego e **mobilidade** brasileira.

Problema:

Latência, **largura** de **banda** e **custos** de transmissão de dados envolvendo altos custos. **Mobilidade**, no Brasil, é um grande **problema**, **colisões**, **velocidades**, **ultrapassagens** enfim. Demanda de espaço e vagas em eventos. Controle de tempo de carros. Análise de segurança sobre carros, sendo eles roubados, ou acidentes que possam acontecer. E principalmente o risco à vida.



Projeto e Solução

A proposta utiliza **rastreamento** de **veículos**, proporcionando análise de algumas **métricas**, através de um sistema com **dashboard**, uma solução mais eficiente e **econômica.**

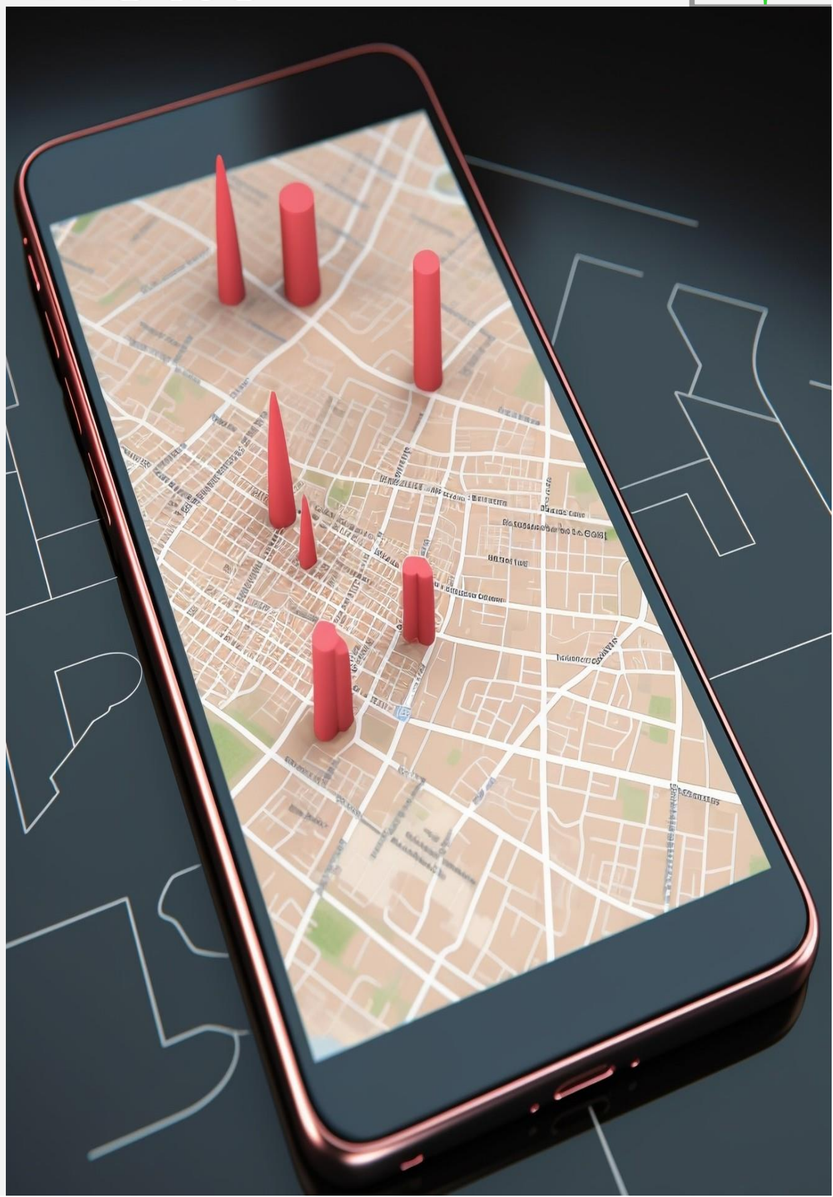
- Redução de Custos;
- Controle de Velocidade e Distância;
- Análise de Fluxo de Tráfego;
- Identificação e notificação de infração;
- Segurança.

Desafios atuais para EdgeML e visão computacional

- Sistema com maior **velocidade de processamento**, **recursos escassos**, **memória** e **energia**, frisando o bem ao meio ambiente, buscando ainda uma boa precisão.

- **Modelos complexos**, **redução de custos** e **desenvolvedores** de **soluções** são **escassos**.

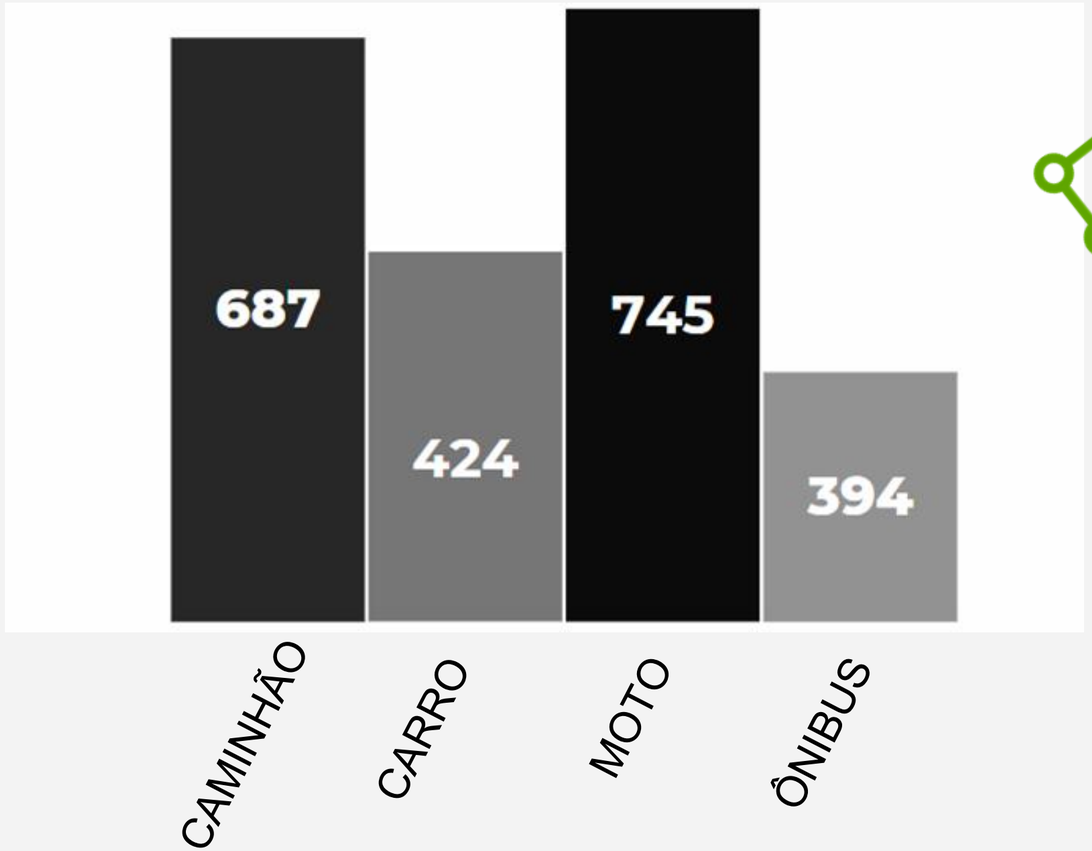
- **Segurança no trânsito**, com problemas devido à **falha humana**, tecnologias complexas, custosas e antigas



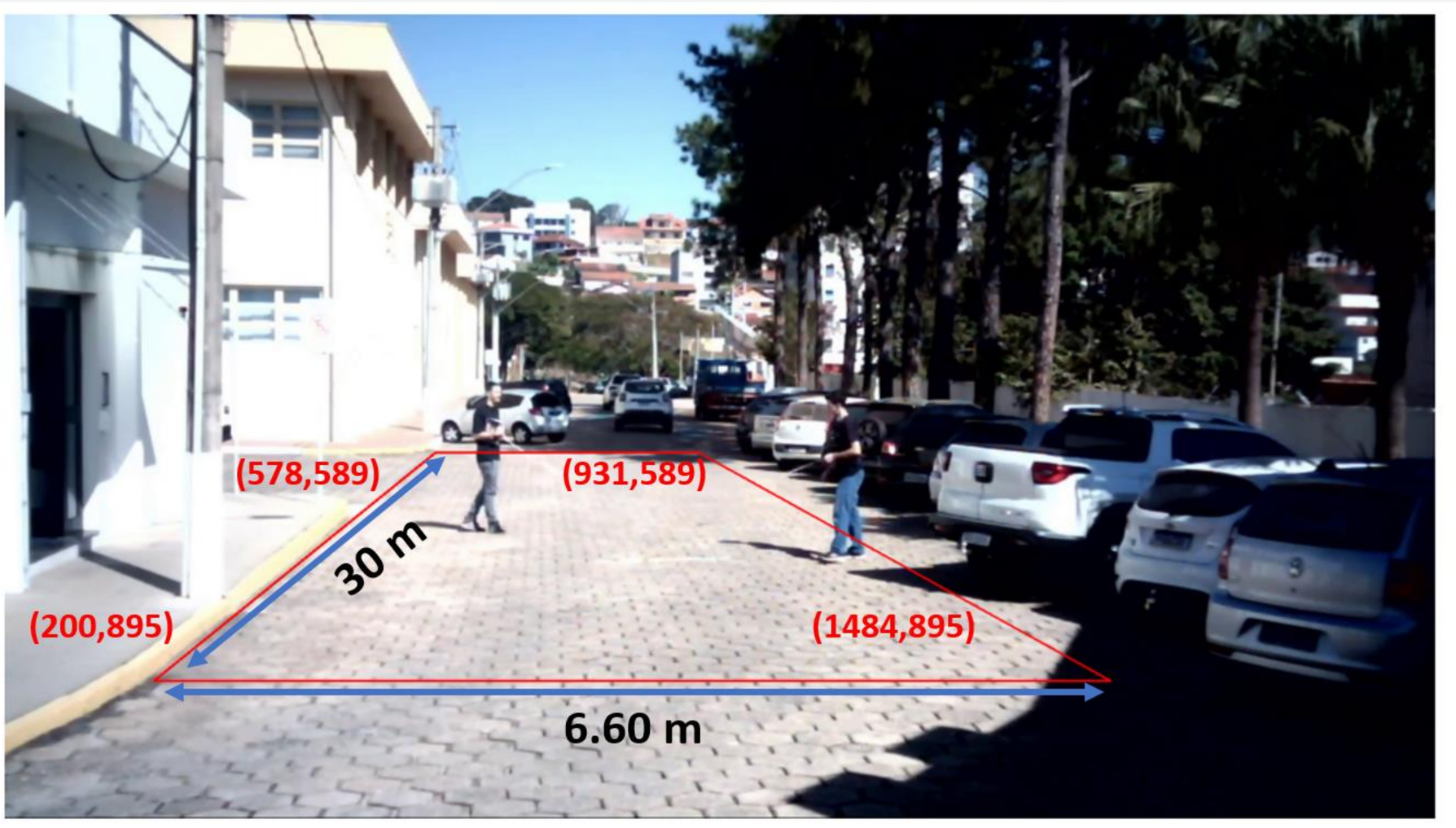
Base de dados utilizada

Roboflow [1]

- **Divisão 70% Treino, 20% validação e 10% teste**

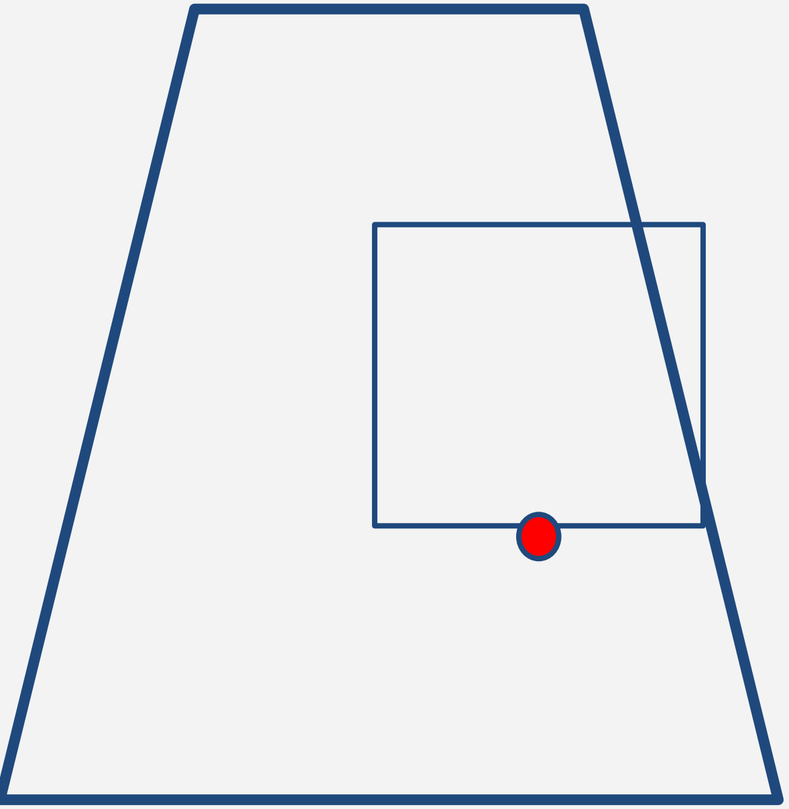


Mapeamento da área de interesse



1920x1080 PX

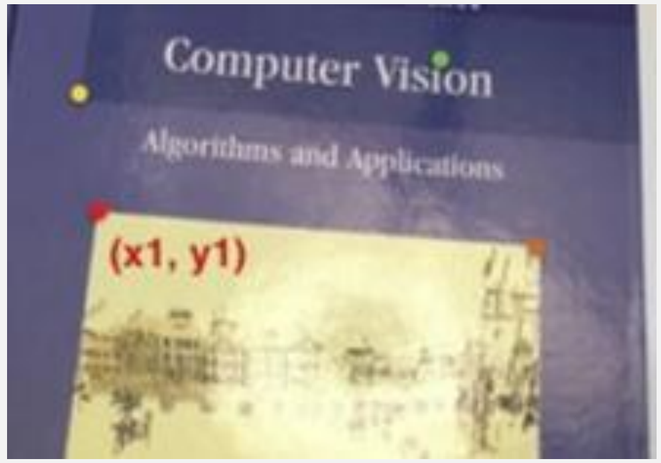
Mapeamento da área de interesse

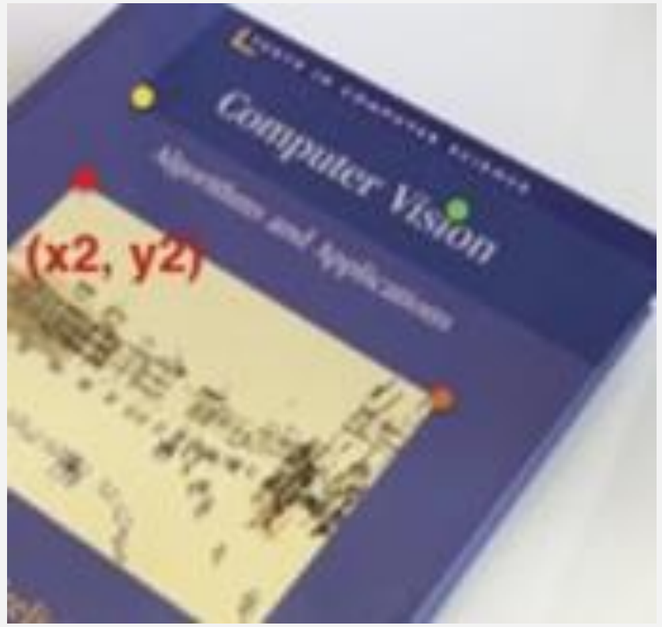


Algorithm 1 Transformação de Pixel para Metro

```
Class ViewTransformer
function VIEWTRANSFORMER(source, target)
    self.matH ← CV2.FINDHOMOGRAPHY(source, target)
end function
function TRANSFORMPOINTSPIXELTOMETER(pointsPixel)
    if pointsPixel == 0 ou self.matH == None then
        return pointsPixel
    end if
    pointsMeter ← CV2.PERSPECTIVETRANSFORM(pointsPixel,
self.matH)
    return pointsMeter
end function
```

Mapeamento da área de interesse





$$H = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix}$$

Considerando o primeiro conjunto de pontos (x1,y1) na primeira imagem e(x2,y2) na segunda imagem. Então, a Homografia **H** mapeia-os da seguinte maneira.

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Mapeamento da área de interesse

Exemplo

Cálculo da Velocidade

1. Ao obter a **matriz de transformação**, de **píxeis para metros**, a cada **frame**, ou **instante desejado**, é obtido a **coordenada** atual no veículo e a mesma é armazenada em uma lista que possui: coordenadas em um **espaço de tempo** e o respectivo **identificador** do objeto.

2. Calcula-se a **distância percorrida** pelo objeto como a **diferença**, em **módulo**, entre a coordenada **atual** e **anterior.**

3. Determina-se o **tempo decorrido** durante o **movimento** do objeto, utilizando o número de **quadros armazenados** na lista de coordenadas e a **taxa** de **quadros por segundo** do vídeo ou webcam.

4. Calcula-se a **velocidade** do objeto utilizando a **fórmula** da **velocidade média** e já convertido para quilômetros por hora, multiplicando por 3,6.

$$\text{Velocidade} = \frac{\text{distância}}{\text{tempo}}$$

Cálculo da Velocidade

4. Caso a **velocidade ultrapasse 30km/h**, é retirado uma **foto**, e enviada por email utilizando o **APP key do Gmail**, com protocolo **SMTP**. Informando no e-mail, a imagem com o ID, velocidade e as distâncias entre outros objetos, se existirem.

5. Aplica-se a **média móvel exponencial** (**MME**) para suavizar a série temporal das velocidades. A fórmula da MME é dada por:

$$\mathrm{MME}_t = \alpha \cdot \mathrm{Vm}_t + (1 - \alpha) \cdot \mathrm{MME}_{t-1}$$

Obs.: Onde **α** (**alfa**), é o fator de **suavização**. Quanto **maior** o valor de alfa, a **velocidade** é mais atrelada ao **presente**, quanto **menor** o valor de alfa, estaria mais corresponde ao **passado**.

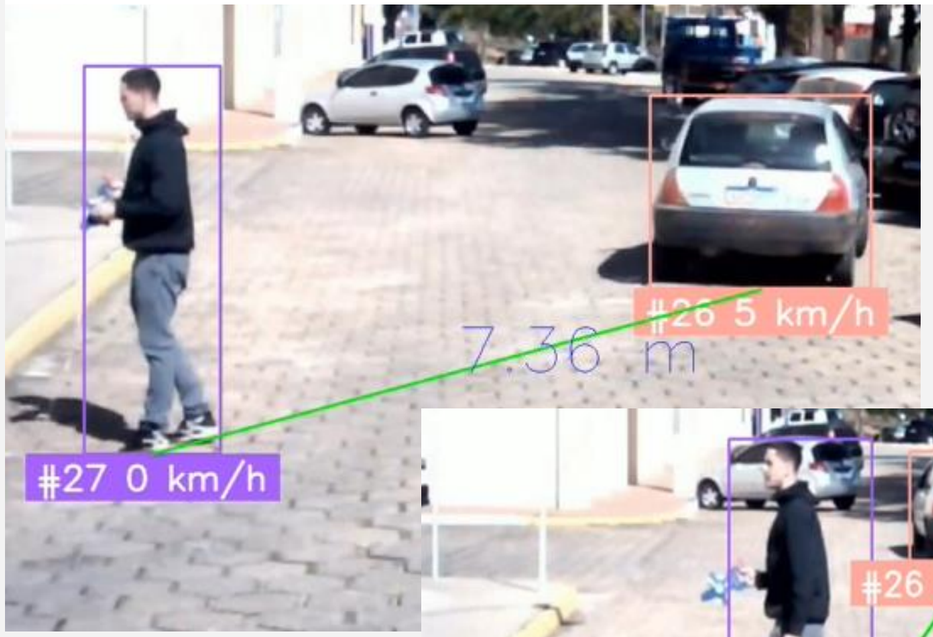
6. Por fim, é adicionado a velocidade suavizada ao *bounding boxes* do objeto.

Cálculo da Velocidade

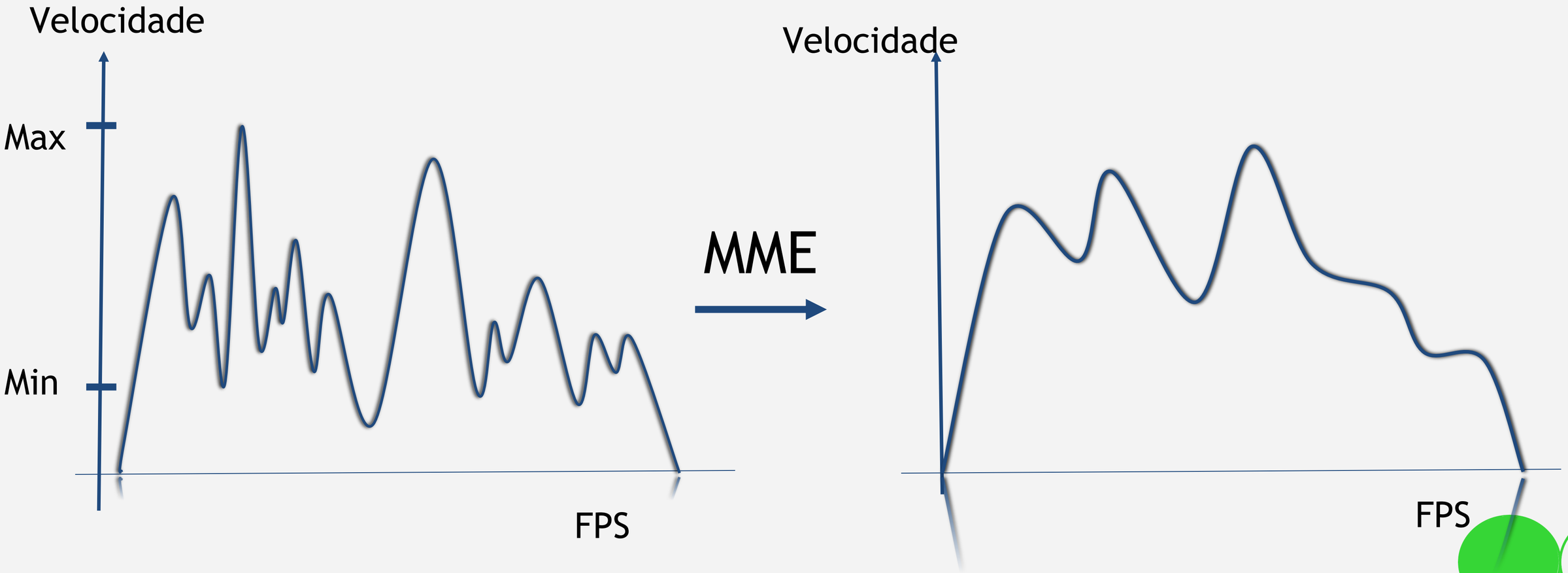
```
function CALCULATESPEED(labels, detections, coordinates, fps)
    for each id in detections.id do
        if id ∉ coordinates then
            labels.append(f"#{id}")
        else
            coordinate_anterior ← coordinates[id][-1]
            coordinate_atual ← coordinates[id][0]
            distance ← abs(coordinate_anterior - coordinate_atual)
            time ← len(coordinates[id]) / video_fps
            speed ← distance / time · 3.6
            if previous_ema is None then
                previous_ema ← speed
            else
                ema_speed ← CALCULATE_EMA(previous_ema, speed, ALPHA)
                if ema_speed > 30 then
                    SENDEMAIL(frame, ema_speed, mensagem)
                end if
            end if
            labels.append(f"#{id} {int(ema_speed) km/h")
        end if
    end for
end function
```

Algorithm 2 Calcular Velocidade

```
const ALPHA = 0.5
function CALCULATEEMA(previousEMA, currentSpeed)
    return ALPHA · currentSpeed + (1 - ALPHA) · previousEMA
end function
```



Cálculo da Velocidade



$$\mathrm{MME}_t = \alpha \cdot \mathrm{Vm}_t + (1 - \alpha) \cdot \mathrm{MME}_{t-1}$$

$0 < \alpha < 1$

Cálculo da Distância

Algorithm 3 Cálculo da distância entre veículos

function EUCLIDEAN_DIST(pointA, pointB)
  return $\sqrt{(pointA_x - pointB_x)^2 + (pointA_y - pointB_y)^2}$
end function
function DRAW_LINE(frame, pointA, pointB, distance)
  CV2.LINE(pointA, pointB)
  text_position $\leftarrow$ $((pointA_x + pointB_x)//2, (pointA_y + pointB_y)//2)$
  CV2.PUTEXT(frame, distance, text_position)
  return frame
end function

Código principal

Algorithm 4 Código principal

Require: algoritmos 1, 2 e 3

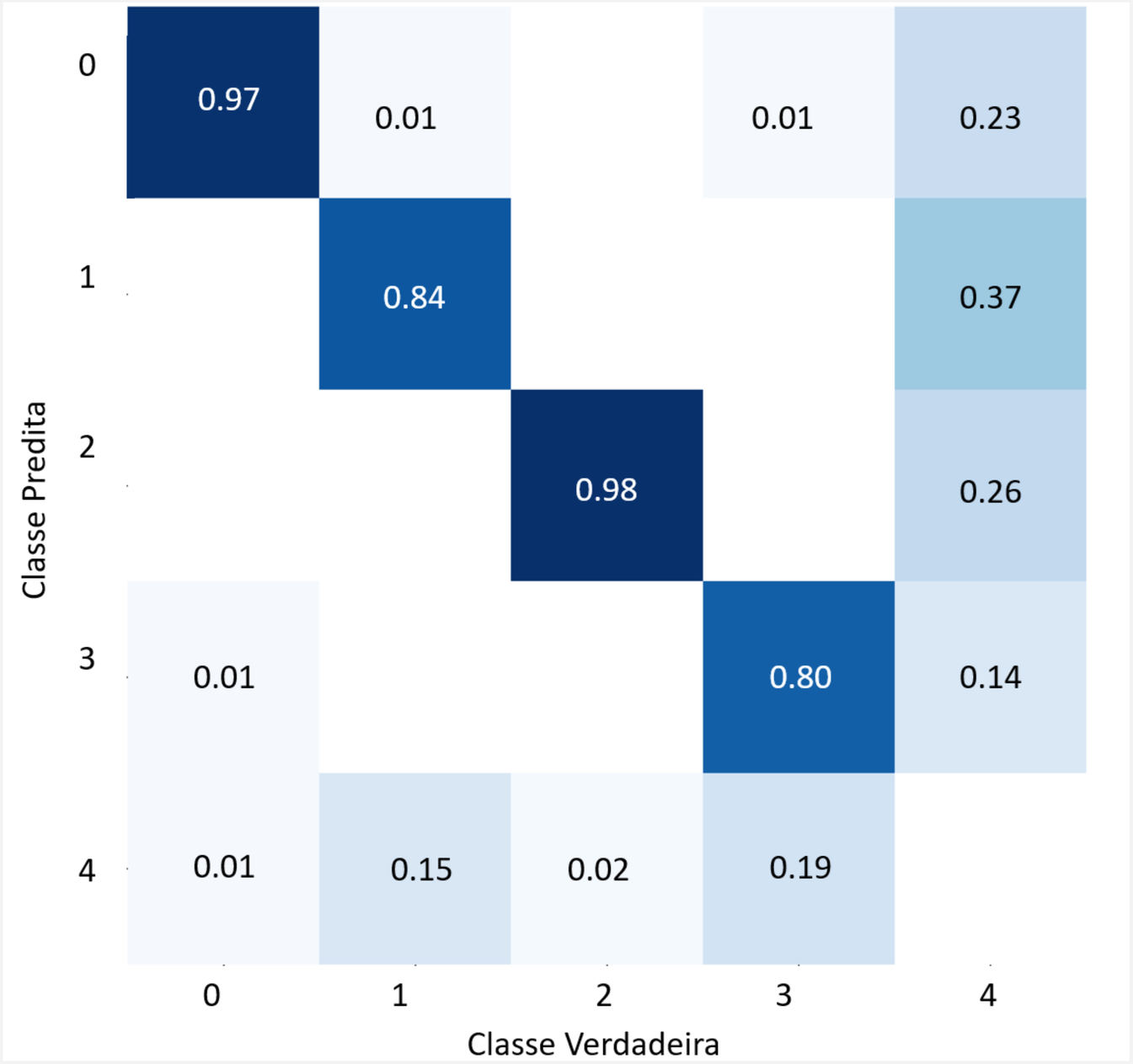
```
view ← ViewTransformer(SOURCE,TARGET)
model ← YOLO(MODEL_NAME)
previous_ema ← None
video_info ← SV.VIDEOINFO.FROM_VIDEO_PATH(VIDEO_PATH)
frame_gen ← SV.GET_VIDEO_FRAMES_GENERATOR(VIDEO_PATH)
tracker ← SV.BYTETRACK(frame_rate=video_info.fps,
track_activation_threshold=CONFIDENCE_THRESHOLD)
polygon_zone ← SV.POLYGONZONE(polygon=SOURCE)
coords ← DEFAULTDICT(lambda:deque(maxlen=video_info.fps))
```

```
for each frame in video_info.total_frames do
    result ← MODEL(frame)[0]
    detec ← SV.DETECTIONS.FROM_ULTRALYTICS(result)
    detec ← polygon_zone.trigger(detec)
    detec ← DETEC.WITH_NMS(IOU_THRESHOLD)
    detec ← BYTE_TRACK.UPDATE_WITH_DETECTIONS(detec)
    pts ← DETEC.GET_ANCHORS_COORDINATES(BC)
    pts_mt ← VIEW.TRANSFORMPXTOMT(pts)
    for each id, [_, y] in ZIP(detec.tracker_id, pts_mt) do
        coords[id].append(y)
    end for
    labels ← []
    CALCULATESPEED(labels, detec, coords, video_info.fps)
    for each i in range(len(pts_mt) - 1) do
        for each j in range(i + 1, len(pts_mt)) do
            ptA ← tuple(pts[i])
            ptB ← tuple(pts[j])
            distance ← EUCLIDEAN_DIST(pts_mt[i], pts_mt[j])
            annotated_frame ← DRAW_LINE(frame, ptA, ptB,
distance)
        end for
        Anota o frame no video alvo
    end for
```

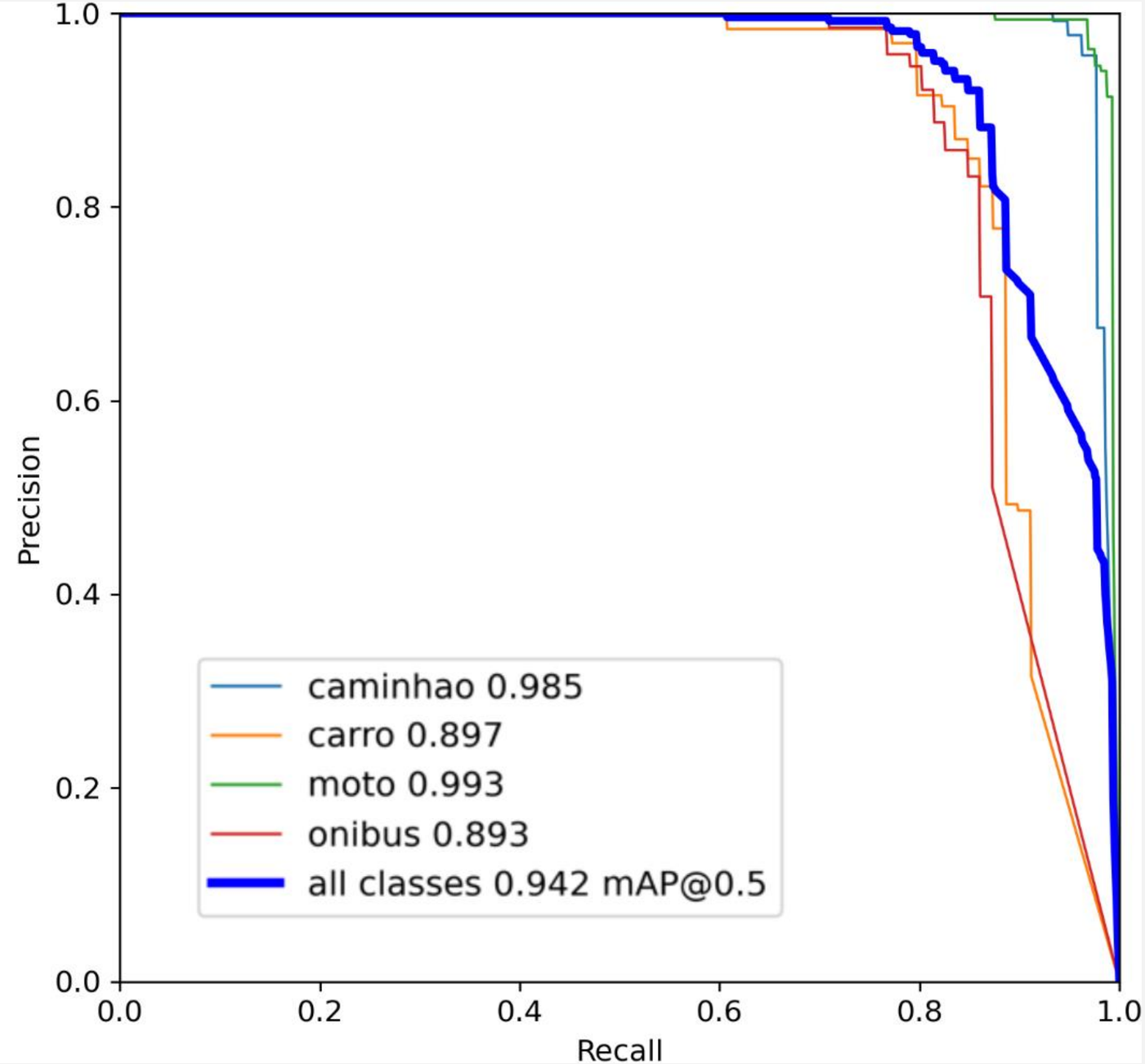
ultralytics
YOLOv5

Resultados

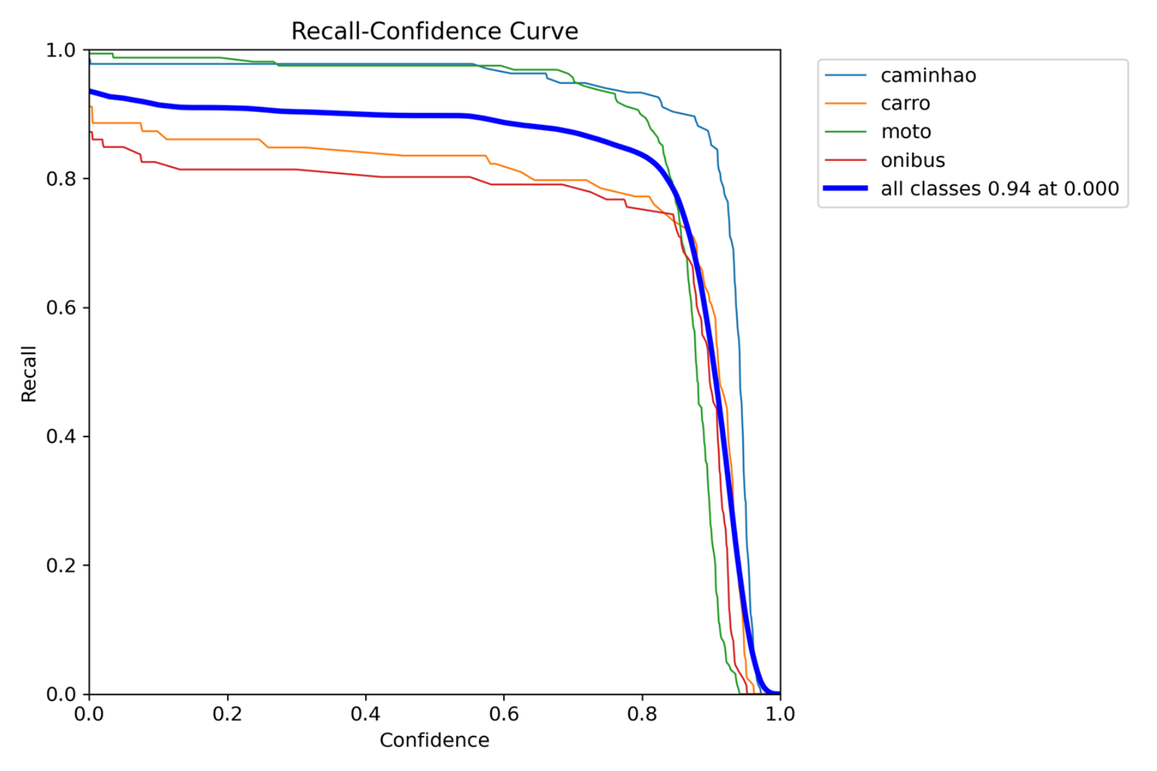
- 0 – Caminhão;
- 1 – Carro;
- 2 – Moto;
- 3 – Ônibus e;
- 4 –Background (Fundo).

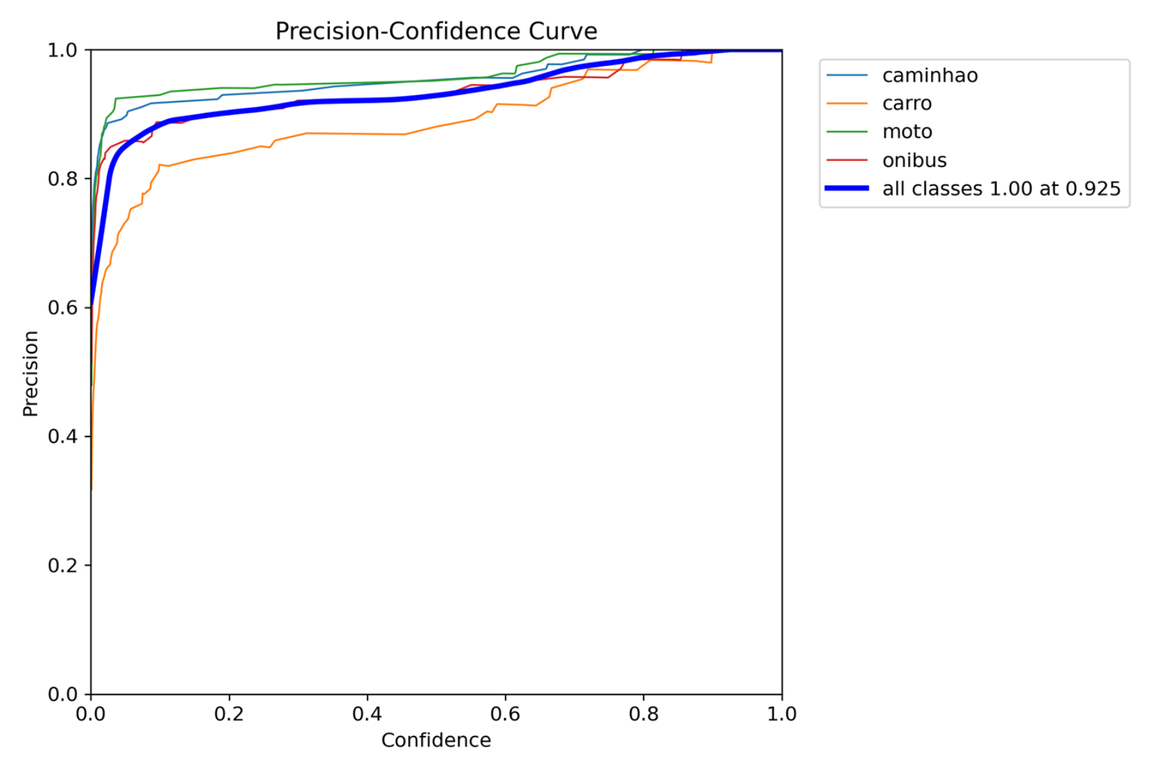


Resultados

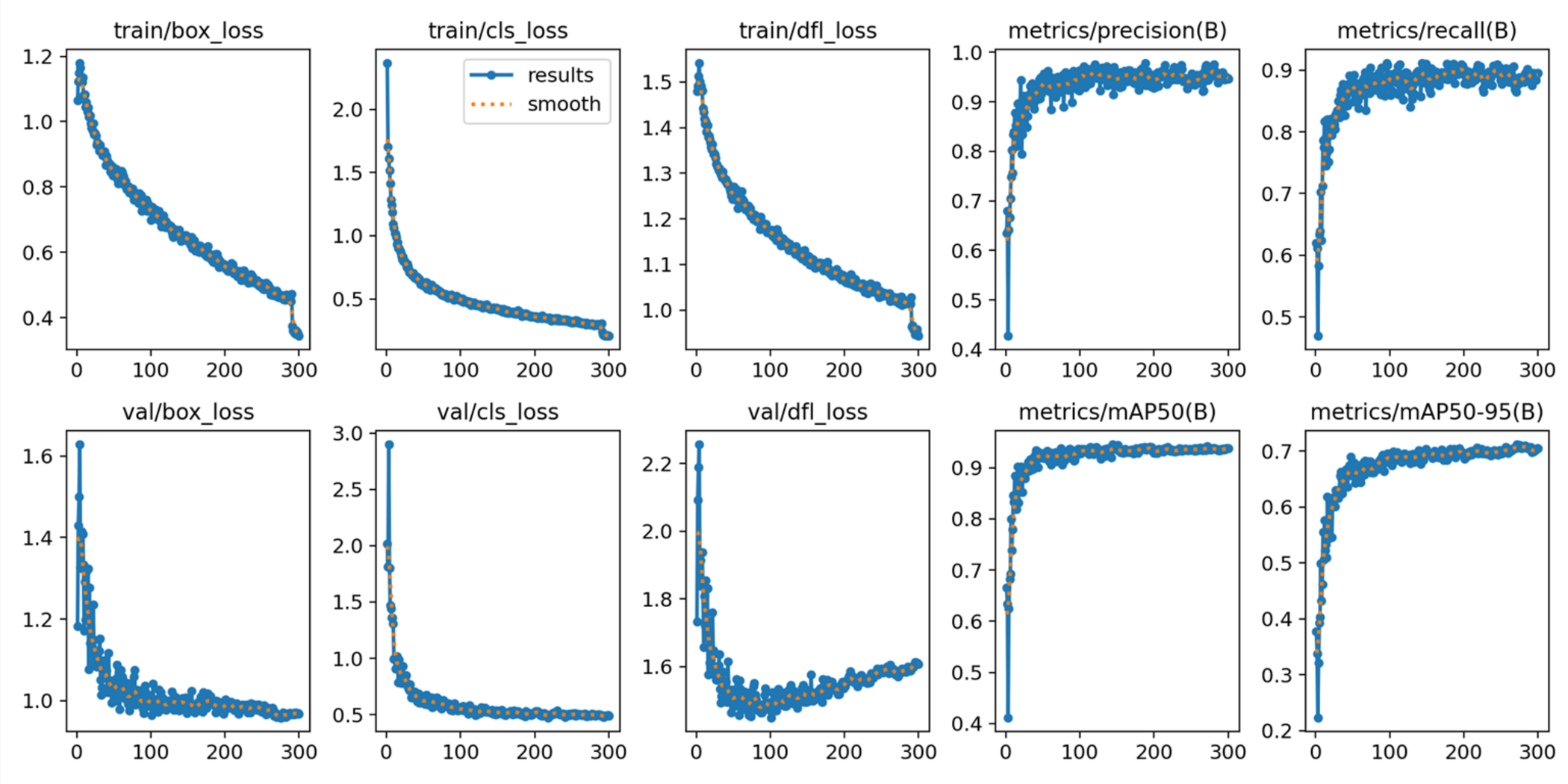

1.0
0.8
0.6
0.4
0.2
0.0
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall
caminhao 0.985
carro 0.897
moto 0.993
onibus 0.893
all classes 0.942 mAP@0.5

Resultados


Recall-Confidence Curve
Recall
Confidence
0.0
0.2
0.4
0.6
0.8
1.0
caminhao
carro
moto
onibus
all classes 0.94 at 0.000


Precision-Confidence Curve
Precision
Confidence
0.0
0.2
0.4
0.6
0.8
1.0
caminhao
carro
moto
onibus
all classes 1.00 at 0.925

Resultados

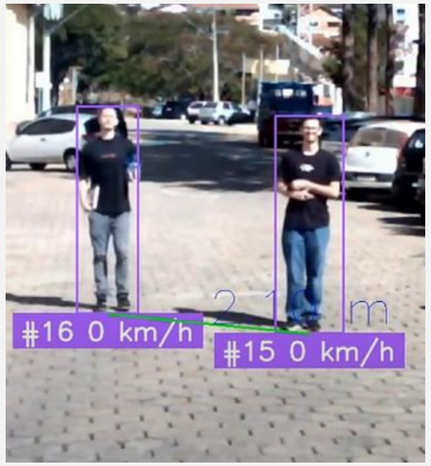

train/box_loss
train/cls_loss
results
smooth
train/dfl_loss
metrics/precision(B)
metrics/recall(B)
val/box_loss
val/cls_loss
val/dfl_loss
metrics/mAP50(B)
metrics/mAP50-95(B)

Resultados

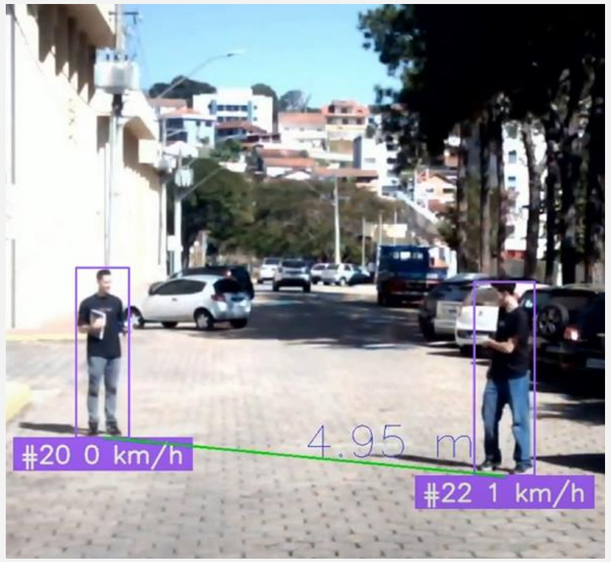
Float32 - best_float32.tflite							
FPS	**CPU**	**RAM**	**mAP50**	**Precisão**	**Recall**	**F1-Score**	**Tamanho**
1,5	25%	1,5 GB	94%	97%	88%	92%	12 MB
FLOAT16 - best_float16.tflite							
2,3	23%	1,2 GB	92%	93%	88%	90%	6 MB
INT8 - best_int8.tflite							
2,5	20%	1,1 GB	92%	94%	87%	90%	3 MB

Resultados

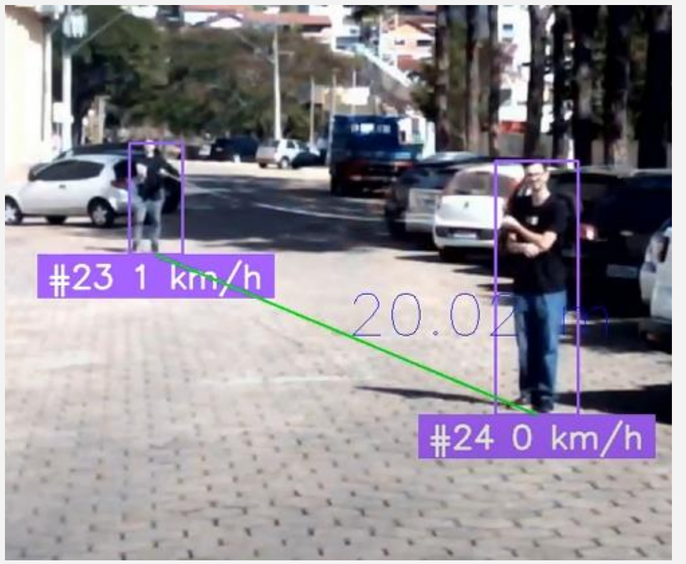
Medição:1 m
Resultado: 0.91 m



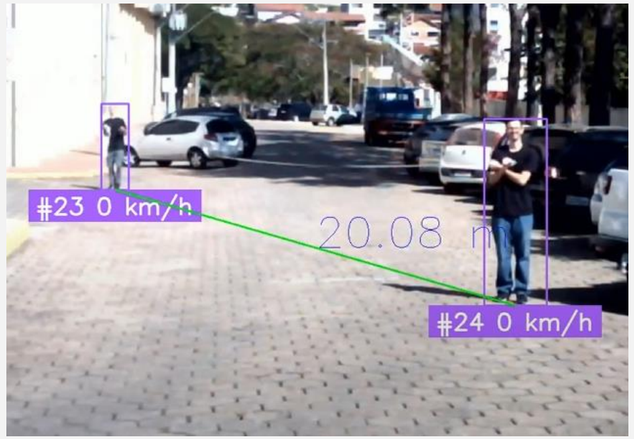
Medição:2 m
Resultado: 2.11 m



Medição:5 m
Resultado: 4.95 m



Medição:20 m
Resultado: 20.02 m



Medição:20 m
Resultado: 20.08m

Estes foram alguns **testes de distância** feitos, para validar se realmente estava conseguindo obter um valor mínimo confiável. Utilizamos uma trena de **30 metros** para mensurar e compara o valor real com o obtido em simulação. A margem de erro em módulo foi de **± 0.11m.**

Obs.: teste feito com 1 hora de gravação.

Resultados

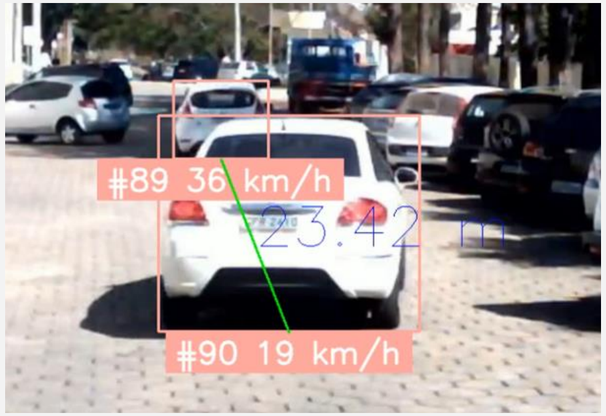
Medição: 20 km/h
Resultado: 20 km/h

Medição: 10 km/h
Resultado: 10 km/h

Medição: 30 km/h
Resultado: 28 km/h

Medição: 30 km/h
Resultado: 32 km/h

Medição: 10 km/h
Resultado: 10 km/h

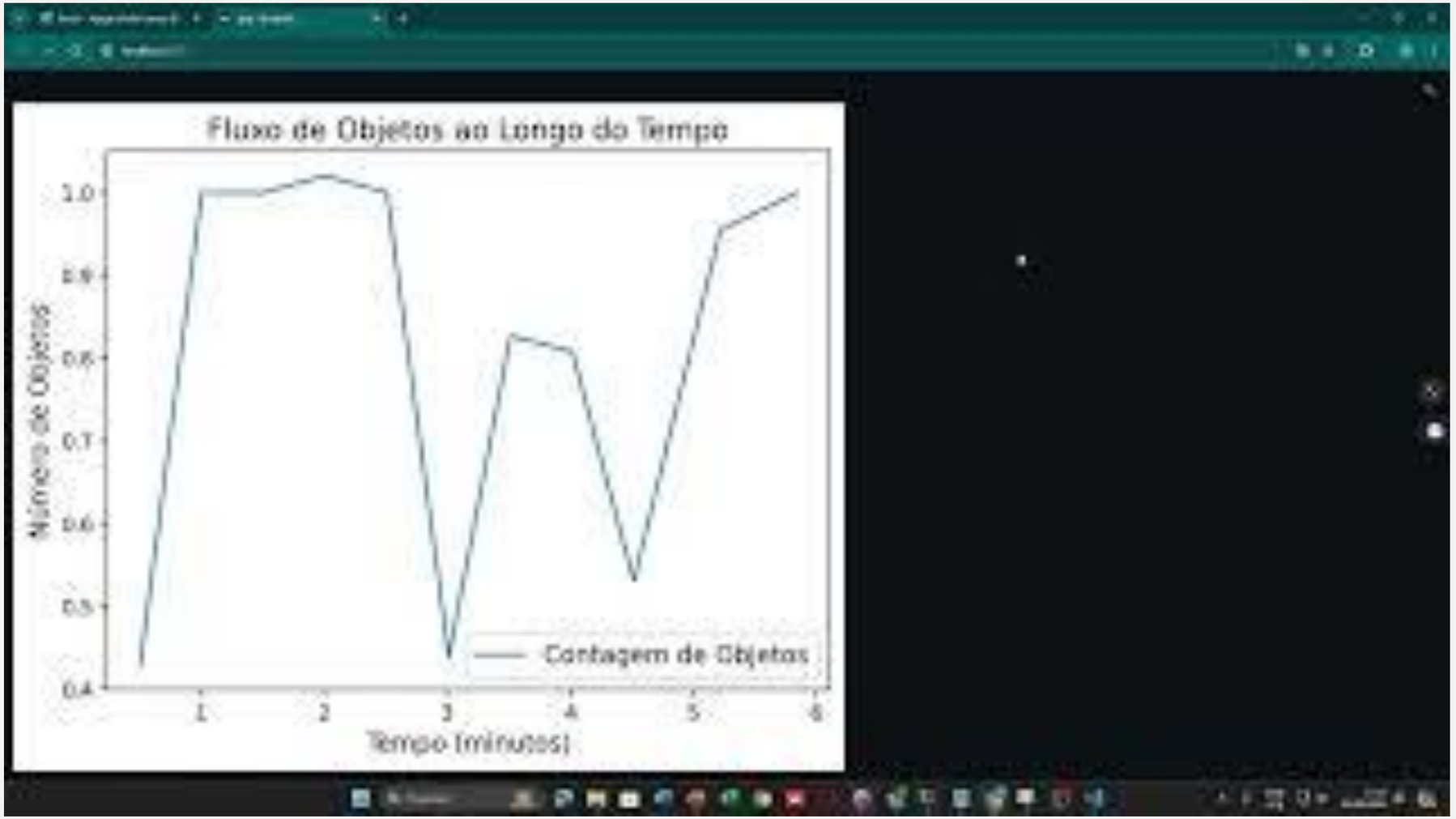


Medição: 30 km/h
Resultado: 19 km/h

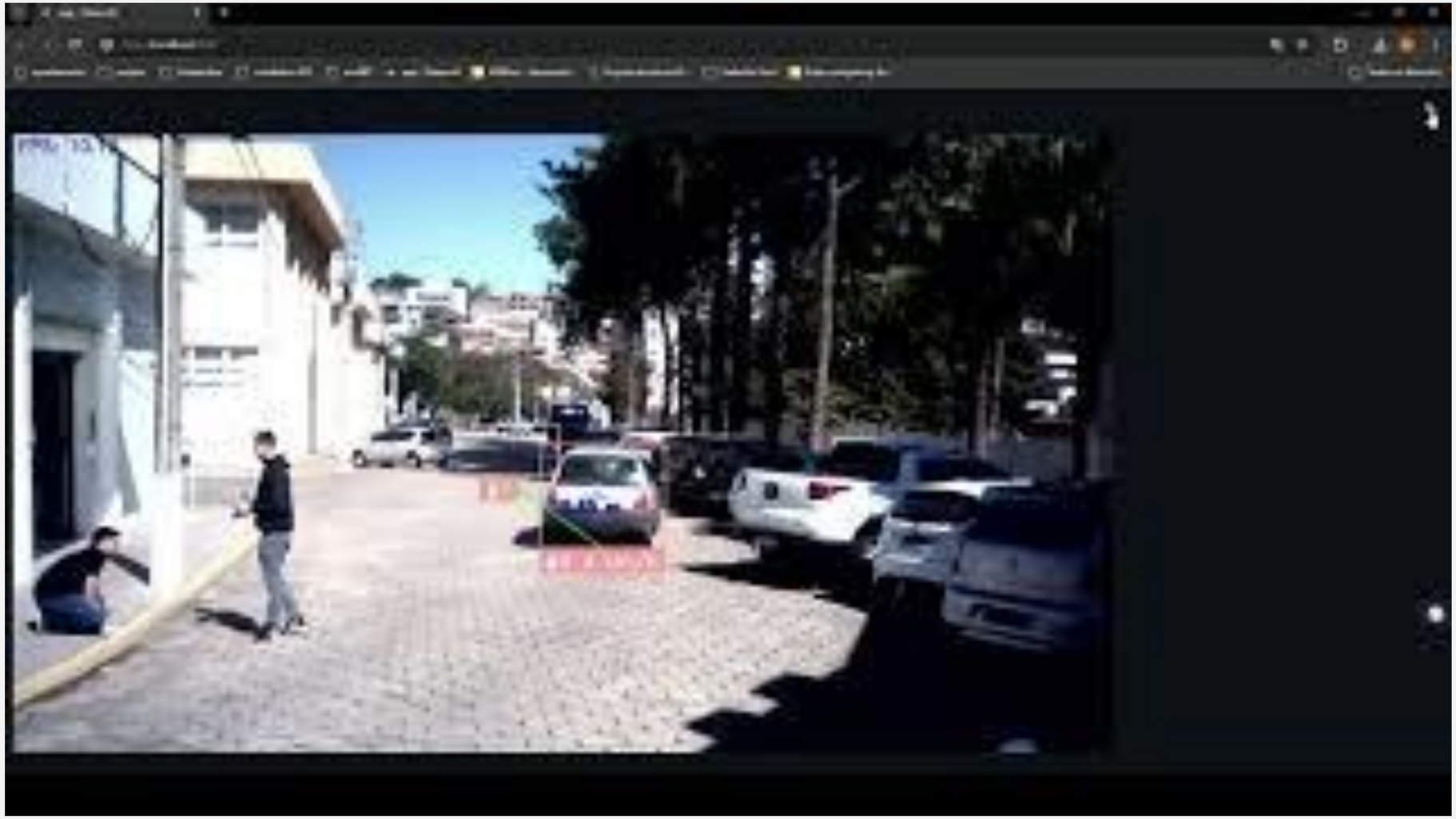
Estes agora foram os testes de velocidade, solicitando aos motoristas que fiquem uma **velocidade** X constante. A margem de erro para este teste foi de aproximadamente **± 4 km/h.**

obs.: Teste feito com 1 hora de gravação.

Resultados - Local - modelo padrão



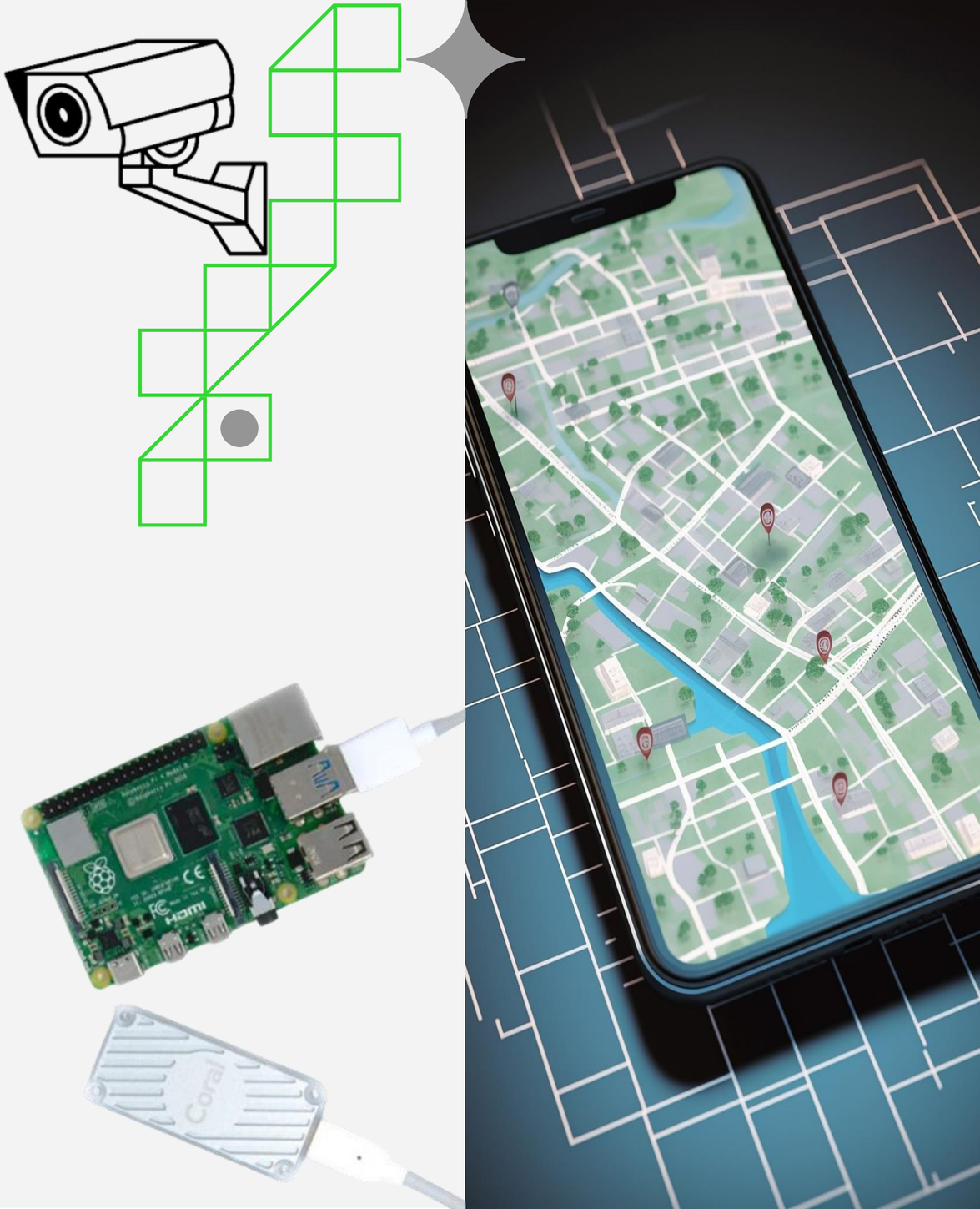
Resultados - Local - modelo quantizado



Resultados - RaspBerry PI 4 Model B

Melhorias

- Otimizar o código, para alcançar maior desempenho sem perda de frame;
- Utilizar acelerador de Redes Neurais, sendo por software ou hardware, como PyArmn, ou Google Edge TPU Coral.
- Utilizar a câmera na perspectiva lateral do carro evitando sobreposição do bounding boxes. E buscar uma área maior.
- Ao utilizar uma área maior, buscar mais pontos e averiguar outras métricas com a Matriz de Homografia.



Melhorias

- Aumentar a quantidade de dispositivos, sincronizar os dispositivos com algum servidor em cloud.
- Aumentar a resolução do modelo para torná-lo mais preciso, porem ainda mantendo o FPS mais alto.
- Utilizar mecanismos de criptografia em borda, e em transmissão, evitando entrada ou manipulação de dos dados e borramento dos rostos e placas dos carros.
- Utilizar algum algoritmo OCR que capture as placas dos carros e identifique-as, notificando além da velocidade, o nome da placa do veículo.

Conclusão

A solução baseada em algoritmos de detecção de objetos YOLO e dispositivos de borda como o Raspberry Pi mostrou-se eficaz para gerenciar o tráfego.

Ofereceu dados precisos sobre o fluxo de veículos, velocidade e distância, melhorando a segurança, demonstrando uma margem de erro pequena em questão de velocidade e distância, com uma inferência em um tempo razoavelmente bom, suficiente para se poder processar o que é necessário.

Referências

- https://www.uio.no/studier/emner/matnat/its/TEK5030/v23/lectures/07-feature-matching/estimating-homographies-from-feature-correspondences_2023.pdf
- https://auth.berkeley.edu/cas/login?service=https%3a%2f%2finst.eecs.berkeley.edu%2f%7eee290t%2ffa19%2flectures%2flecture8-homography.pdf
- https://learnopencv.com/homography-examples-using-opencv-python-c/
- Jocher, G., Chaurasia, A., & Qiu, J. (2023). Ultralytics YOLO (Version 8.0.0) [Computer software]. https://github.com/ultralytics/ultralytics
- https://universe.roboflow.com/dataset-ic/tipos-de-veiculos
- https://blog.roboflow.com/estimate-speed-computer-vision/
- https://auth.berkeley.edu/cas/login?service=https%3a%2f%2finst.eecs.berkeley.edu%2f%7eee290t%2ffa19%2flectures%2flecture8-homography.pdf
- https://blog.roboflow.com/video-stream-analysis/
- https://pessoal.dainf.ct.utfpr.edu.br/rminetto/projects/vehicle-speed/
- https://arxiv.org/pdf/2003.13137v2
- https://ieeexplore.ieee.org/abstract/document/7576700/

Obrigado!

Perguntas?

hyago.silva@mtel.inatel.br
+55 35 9 9725 3305

https://www.linkedin.com/in/hyagovieira/

https://github.com/HyAgOsK/