Sensor de Software Robusto para Identificar Pessoas em uma Cena de Vídeo

Arthur Hernandez Perez[1], Karran Cardoso de Araújo Lemos[1], Evandro Luiz Cardoso Macedo[1], Alexandre Sztajnberg[1,2]

[1]Departamento de Sistemas e Computação, Laboratório de Ciência da Computação, (LCC)
Instituto de Matemática e Estatística (IME)

[2]Pós-Graduação em Engenharia Eletrônica (PEL)
Pós-Graduação em Ciências Computacionais e Modelagem Matemática (CompMat)

Universidade do Estado do Rio de Janeiro (UERJ), Rio de Janeiro – RJ – Brasil

arthurhernandezp@gmail.com, karran.lemos@gmail.com

evandro.macedo@ime.uerj.br, alexszt@ime.uerj.br

Abstract. *The development of smart cities and the Internet of Things paradigm allows the creation of more efficient, sustainable and safe cities. Among the innovations, smart cameras stand out for identifying objects and people, enabling continuous monitoring and real-time data collection for different environments. This article presents a smart software sensor for counting people in video scenes, developed for Computer Sciences Lab. and integrated with the LCC-IoT application. It allows monitoring and generating alarms in specific situations, such as the presence of people outside of permitted hours, while preserving users' privacy. The sensor was implemented to maintain continuous and concurrent operation across multiple cameras, controlling data acquisition and consistent generation of telemetry through semaphores, in addition to recording errors, making it robust. The solution uses the OpenCV and YOLO for image processing, MQTT and HTTP application protocols for sending telemetry.*

Resumo. *O desenvolvimento de cidades inteligentes e o paradigma de Internet das Coisas permite a criação de cidades mais eficientes, sustentáveis e seguras. Entre as inovações, destacam-se câmeras inteligentes com a identificação de objetos e pessoas, viabilizando monitoramento contínuo e coleta de dados em tempo real para diversos ambientes. Este artigo apresenta um sensor inteligente de software para contagem de pessoas em cenas de vídeo, desenvolvido para o Lab. de Ciência da Computação e integrado à aplicação LCC-IoT . O sensor permite monitorar e gerar alarmes em situações específicas, como a presença de pessoas fora do horário permitido, enquanto preserva a privacidade dos usuários. O sensor foi implementado para manter operação contínua e concorrente em múltiplas câmeras, controlando a aquisição de dados e a geração consistente de telemetria por meio de semáforos, além de registrar erros, tornando o mesmo robusto. A solução utiliza o OpenCV e YOLO para processamento de imagens, os protocolos de aplicação MQTT e HTTP para o envio de telemetria.*

1. Introdução

O desenvolvimento de cidades inteligentes vem sendo alavancado ao longo dos anos especialmente por conta da influência do paradigma de Internet das Coisas, tornando as cidades mais eficientes, sustentáveis e seguras [Ahmed et al. 2016, Motta et al. 2024]. Em particular, sistemas de câmeras inteligentes instrumentalizados com capacidade de identificação de objetos e pessoas têm se difundido a fim de melhorar a segurança de ambientes de maneira geral. Tais sistemas habilitam a criação de ambientes inteligentes com monitoramento contínuo e em tempo real dos mais diversos cenários, sejam eles empresariais, fábricas, residências, salas de aula, entre outros. Isso permite, por exemplo, o acompanhamento de atividades residenciais (especialmente de pessoas idosas ou com doenças crônicas), o funcionamento de esteiras de produção, a captura de ocorrências, entre outros, além de permitir uma coleta massiva de dados sobre os ambientes.

Este trabalho tem como motivação a criação de um sensor inteligente de software capaz de contar pessoas em uma cena de vídeo. Este sensor foi desenvolvido para atender às necessidades de monitoramento contínuo do Lab. de Ciência da Computação através da integração do sensor com a aplicação LCC-IoT [LCC, UERJ 2025]. O LCC-IoT é uma aplicação para ambientes inteligentes que emprega dispositivos, protocolos de aplicação e uma plataforma integrada de aplicações para IoT. O ambiente do LCC contempla duas salas de aula e um laboratório de pesquisa. Uma das características do LCC-IoT é combinar a informação do número de pessoas presentes nas salas, aumentadas com outras informações de ambiente, habilitando a geração de alarmes e rotinas de atuação conforme necessário. Casos como, por exemplo, pessoas na sala após as 23:00, ou nenhuma pessoa na sala com iluminação ligada e temperatura abaixo de 25°C (no Rio de Janeiro), podem gerar alarmes, os quais dispariariam notificações para os usuários dos sistema para atuarem na situação apresentada.

Diversos refinamentos foram feitos nos componentes do sensor inteligente a fim de aumentar a acurácia e diminuir o número de casos de falsos positivos. O sensor foi estruturado na forma de um serviço Linux que automatiza a inicialização do sensor e controla a sua operação de maneira contínua. O serviço permite também a execução concorrente de várias instâncias do sensor para serem aplicadas em diferentes câmeras. Também foi desenvolvido um sistema de registro de erros que captura e armazena informações sobre falhas durante a detecção de pessoas e possíveis problemas no envio de dados. Além disso, foi necessário o emprego de semáforos para garantir exclusão mútua durante o processamento das imagens, evitando o envio de telemetrias com valores inconsistentes, dada a complexidade das bibliotecas utilizadas. Estes refinamentos tornam o sensor de software robusto como ferramenta que pode operar continuamente. O sensor desenvolvido está em operação, incorporado ao LCC-IoT e utiliza diversas tecnologias, tais como, o protocolo MQTT, HTTP e YOLO [Redmon and Farhadi 2018]. Ressalta-se que a solução preserva a privacidade das pessoas, tendo em vista que não emprega reconhecimento facial nem considera a identidade delas durante o processo de análise das imagens.

O restante do artigo está estruturado da seguinte forma. Na Seção 2 apresentamos trabalhos relacionados e produtos com funcionalidade semelhante. Em seguida, apresentamos brevemente as tecnologias utilizadas na Seção 3. A Seção 4 apresenta a estrutura geral do sensor e detalhes de implementação são tratados na Seção 5. A Seção 6 apresenta o caso de uso do sensor na aplicação LCC-IoT e em seguida a Seção 7 conclui o trabalho.

2. Trabalhos Relacionados

O módulo de detecção de pessoas das câmeras IP Eocortex [Eocortex 2022] é utilizado para detectar e contar pessoas em um estabelecimento, assim como detectar pessoas saindo e entrando em suas premissas. Outra aplicação com funções similares é a solução de contagem de pessoas da Camlytics [Camlytics 2022] que, assim como o Eocortex, busca auxiliar na detecção e contagem de pessoas. Ambos oferecem aplicações de configuração de detecção e a exibição de dados, mas apenas oferecem a possibilidade de exportar estes dados no formato CSV ou enviar por e-mail. Diferente dos trabalhos, nossa solução permite o acompanhamento contínuo do ambiente monitorado, com os dados de telemetria podendo ser consumidos por protocolos de aplicação.

O sistema proposto em [Chen et al. 2011] utiliza subtração de fundo para detectar pessoas utilizando os pontos de diferença entre os quadros. Ele também rastreia a posição dos indivíduos para lidar com casos em que diferentes pessoas em movimento cobrem as outras no campo de visão da câmera, assumindo que, se duas pessoas convergirem em um só objeto, esse objeto provavelmente representa as duas pessoas. O trabalho emprega algumas técnicas que também foram utilizadas no nosso sensor, como a subtração de fundo, porém, a proposta não considera aspectos de escalabilidade no envio de telemetria.

O sistema proposto em [Yang et al. 2003] detecta silhuetas utilizando subtração de fundo em imagens obtidas de diversas câmeras posicionadas em diferentes pontos de vista em uma sala e utiliza um algoritmo para projetar essas silhuetas em um plano bidimensional, que consegue ter uma boa precisão e um crescimento linear de gasto de processamento. Sua precisão depende diretamente do número de câmeras usadas para filmar um mesmo ambiente de diferentes ângulos. No sensor proposto optamos por não usar a técnicas de subtração de fundo para detectar pessoas, mas adotamos a técnica como auxiliar para detectar movimento entre duas imagens consecutivas.

3. Tecnologias Utilizadas

O sensor desenvolvido integra algumas bibliotecas e mecanismos para adquirir e tratar as imagens das câmeras, detectar o número de pessoas, compor a mensagem de telemetria e enviar esta como suporte de um protocolo de aplicação.

Utilizamos a biblioteca **OpenCV**[1] [Bradski 2000], que oferece várias ferramentas e soluções para tratamento de imagem. No OpenCV foram utilizados:

- **Módulo Deep Neural Network (DNN)**. O módulo DNN do OpenCV contém as funções necessárias para carregar o sistema de detecção de pessoas YOLO [Redmon and Farhadi 2018], escrito com o framework Darknet [Redmon 2016], usado para detectar pessoas em vídeos no projeto.
- **Algoritmo de subtração de fundo MOG2**. Esta técnica é usada na implementação do detector de movimento que, por utilizar menos processamento que o sistema de detecção de pessoas por redes neurais, auxilia na maior eficiência da aplicação ao ser usado para limitar o uso do detector mais computacionalmente complexo enquanto a cena não mudar, como no caso de uma sala vazia ou de uma sala onde todos os indivíduos estão sentados sem mudar de posição.

[1] OpenCV 4.4.0.42

- **Non-Maximum Suppression (NMS)**. É um algoritmo usado para combinar diversas caixas bem próximas em uma só. O detector encontra diferentes possíveis "caixas" que representam uma pessoa, que são então unidas em uma só que representa, com certa precisão, a pessoa real [Mueller 2021].
- **Utilitários**. A biblioteca OpenCV também é usada para acessar fluxos de vídeo (webcam, arquivo de vídeo e fluxos RTSP), para gerar a janela de visualização de detecção quando o módulo de depuração é habilitado (Figura 1).

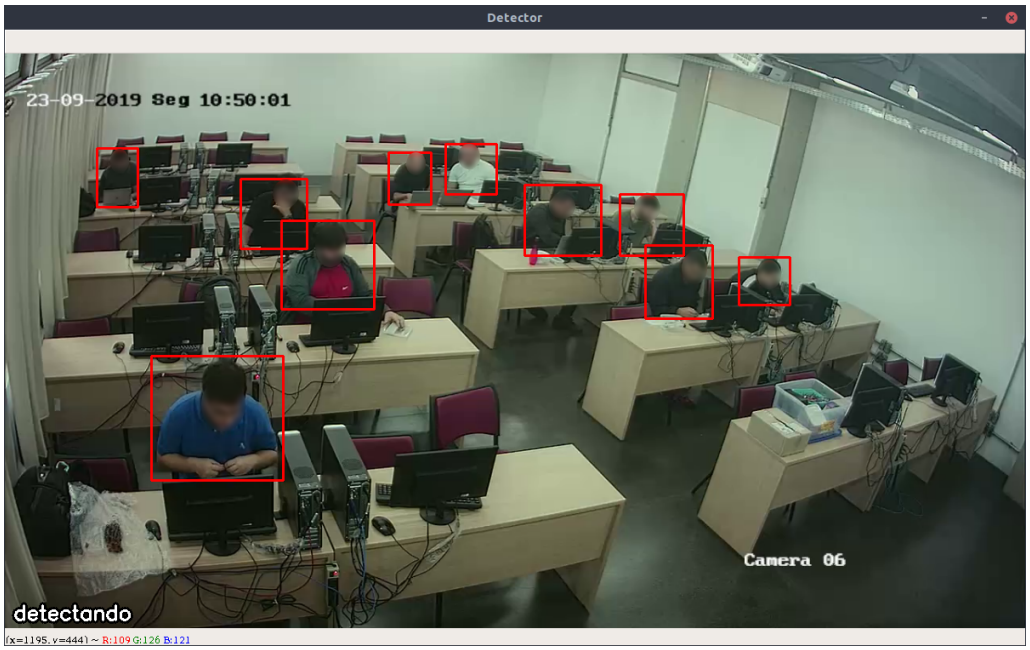

Figura 1. Modo depuração com as caixinhas em tempo real

Também usamos no nosso sensor os sistema de detecção **You Only Look Once YOLO**[2] [Redmon and Farhadi 2018]. O YOLO é um sistema de detecção de objetos em imagens implementado usando o *framework* de aprendizado profundo Darknet, uma arquitetura de rede neural convolucional. Seus criadores disponibilizam pesos pré-treinados com um bom grau de precisão para detecção de pessoas. Adicionalmente, o OpenCV possui um método de leitura de modelos criados com o Darknet, então o YOLO pode ser usado sem configurações adicionais.

Como protocolo de comunicação, o **Paho MQTT**[2] [Eclipse Paho 2018] é utilizado. A biblioteca Paho MQTT para Python é uma implementação de código aberto do protocolo MQTT, utilizado para envio de telemetrias. Além destes módulos, empregamos também bibliotecas SciPy[2] e Numpy[2] para alguns cálculos nas rotinas do sensor. Utilizamos também a biblioteca *logging* para implementar o registro de eventos e *requests* para enviar mensagens de telemetria, alternativamente, por HTTP.

4. Estrutura do Sensor

A Figura 2 apresenta a estrutura geral do serviço Linux onde os módulos do sensor estão encapsulados.

[2]YOLO 3; Paho MQTT 1.6.1; Scipy 1.4.1; Numpy 1.18.5; Python 3.5.10

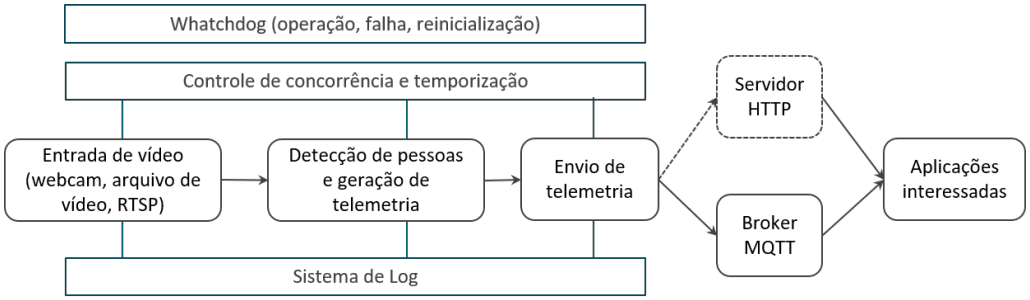

Figura 2. Estrutura do sensor

O sensor é construído em Python[2], integrando bibliotecas, serviços, sistema de *log* e aspectos de concorrência tratados com mecanismo de semáforos do Linux. A página **https://www.lcc.ime.uerj.br/sensor-pessoas** contém o link para o código do sensor, onde também está disponível a documentação. Um vídeo com uma demonstração do sensor em operação no caso de uso relatado na Seção 6 também está disponível nesta página.

O módulo **Entrada de Vídeo** é responsável por buscar e amostrar imagens a partir de fluxos de vídeo obtidos de alguma fonte, seja uma *webcam*, arquivo de vídeo ou acessando um fluxo com o protocolo RTSP em uma URL. A seleção da entrada é parametrizada. O módulo **Detecção de Pessoas e geração de telemetria**, núcleo do sensor, é composto por submódulos importantes:

- **Detector de movimento:** Este módulo executa em uma *thread* separada. Periodicamente lê um quadro de vídeo do módulo de entrada e calcula se houve movimento ou não no vídeo e guarda essa informação em uma propriedade da classe.O processamento gasto pela rotina de detecção de pessoas é significativo e justifica o uso do módulo caso GPUs ou o suporte à GPUs não estejam disponíveis. Assim, implementou-se um detector de movimentos, utilizando o MOG2 (Seção 3), para que o detector de pessoas só fosse acionado quando houvesse movimento no vídeo. De forma resumida, são geradas máscaras de primeiro plano do último quadro e verifica-se a quantidade de *pixels* detectados como parte do primeiro plano. Se a porcentagem passar de um certo valor, é considerado que houve movimento.
- **Detector de Pessoas (DP):** Caso tenha sido detectado movimento, ou tenha passado um período de tempo sem movimento, o módulo DP aciona o OpenCV com o YOLO para detectar indivíduos e gerar retângulos em volta deles, incluindo um parâmetro de precisão que é a probabilidade da detecção não ser um falso positivo.
- **Fusão das Caixas:** A lista de retângulos e parâmetros de precisão resultante do módulo DP é filtrado utilizando o NMS, que avalia os retângulos com interseção que representam a mesma pessoa e os funde, gerando a contagem final do número de pessoas. A Figura 3 ilustra este procedimento.

Os dados coletados e calculados são adicionados a um histórico, finalizando o período de coleta e tratamento. Tal histórico é consumido pelo modo de Geração de Telemetria, ilustrado no Código 1.

Figura 3. Fusão de caixas na imagem utilizando o NMS

```python
def finaliza_periodo(self):
    if self.tempo_decorrido != 0:
        media = self.soma_ponderada/self.tempo_decorrido
        fps = self.total_frames / self.tempo_decorrido
    else:
        media = 0.0
        fps = 0.0
    final_max = max(self.max_atual, 0)
    final_min = 0 if self.min_atual == float('inf') else self.min_atual
    novo_item_historico = PessoasHistorico.ItemHistorico(media, final_min, final_max,
    ↪  self.tempo_decorrido, self.tempo_inicial, datetime.now(), self.total_frames,
    ↪  fps)
    self.historico.append(novo_item_historico)
    self._inicia_novo_periodo()
    return novo_item_historico
```

Código 1: Construção do registro e inserção no histórico.

O módulo **Geração de telemetria** prepara uma estrutura JSON com as informações do registro no histórico, como no exemplo do Código 2. Este módulo permite adaptar o conteúdo da telemetria também para um padrão XML específico ou introduzir os elementos das informações obtidas na estrutura obtida no histórico em um texto para ser enviado por e-mail ou SMS. A telemetria gerada é também armazenada como um registro em um histórico, que é consumido pela módulo de Envio de Telemetria.

```json
{"MediaPessoas": "6.30", "MaximoPessoas": 10, "MinimoPessoas": 2, "TempoTotal": "8.98",
↪  "HorarioAnalise": "2025-02-13 18:00:11.730018"}
```

Código 2: JSON com telemetria.

O módulo **Envio de Telemetria** possui uma *thread* que periodicamente verifica se existem registros de telemetria pendentes para envio e procede o envio. O endereço de destino é parametrizado ao se inicializar o serviço. Primeiro uma tentativa de envio é realizada por MQTT, com QoS 0 e porta padrão MQTT. Se o envio não for bem sucedido alternativamente o protocolo HTTP é utilizado com método POST e cabeçalho "Content-Type": "application/json". Observa-se que as configurações dos protocolos MQTT e HTTP estão vinculadas à plataforma ThingsBoard que utilizamos no LCC-IoT , onde

estamos testando o sensor. Mas, também seria possível utilizar outras configurações e até outros protocolos de aplicação.

O módulo **Sistema de Log** registra as capturas de erros e falhas de transmissão, fornecendo um acompanhamento detalhado e rápido do sistema em operação. Esses registros são úteis para diagnosticar problemas, rastrear o histórico de atividades e analisar o desempenho do sistema.

5. Detalhes de Implementação

Por limitações de espaço, não abordamos todos os detalhes de implementação, mas alguns pontos valem destaque. A integração de diversos elementos, bibliotecas, chamadas a serviços distribuídos e a possibilidade de execução concorrente para várias câmeras podem levar à condição de falha, por mais elaborada que seja a estratégia de tratamento de exceções. Assim, implementamos um *whatchdog* simples para monitorar a operação e possíveis falhas, providenciando a reinicialização automática do serviço.

Na mesma linha, adicionamos um mecanismo de controle de concorrência e temporização. Um semáforo de *lock* em arquivo, assegurando a exclusão mútua de partes críticas de execução do código, principalmente a busca pelas imagens obtidas remotamente nas câmeras e as rotinas de detecção, permitindo o processamento com recurso de CPU livre. Com isso conseguimos obter escalabilidade para criar instâncias do sensor para várias câmeras. Para limitar o tempo de cada instância do sensor os módulos são protegidos por temporizadores que interrompem o processamento e limpam os históricos. Observa-se que cada instância do sensor também recebe o parâmetro do período de amostragem que deve ser utilizado.

6. Caso de Uso

A integração do sensor com a aplicação LCC-IoT é direta. A plataforma ThingsBoard recebe mensagens de telemetria em JSON através de protocolos como HTTP, MQTT ou SMNP. Atualmente utilizamos o MQTT e o HTTP como alternativa.

O *script* que inicializa os processos do sensor passa como parâmetro a URL do fluxo RTSP da câmera, a identificação do dispositivo na plataforma e o endereço do *broker* MQTT. Para cada câmera um processo é criado, juntamente com os *logs* e semáforo para evitar o envio de telemetrias inconsistentes (conforme Código 3).

```
1 LOG_PATH="detector-de-pessoas-por-video-2.0/detector/var/log"
2 LOCK_MUTEX_PATH="detector-de-pessoas-por-video-2.0/detector/src/lock_mutex.py"
3 SCRIPT_PATH="detector-de-pessoas-por-video-2.0/detector/run/lcc.py"
4 TB_URL="thingsboard.lcc.ime.uerj.br"

5 python3 $LOCK_MUTEX_PATH python3 $SCRIPT_PATH --camera 1 --mostrar-video
  ↪  --mostrar-caixas --from $RTPC_CAM_1 --to $TB_URL --token-mqtt $CAMERA_1_ID &
```

Código 3: Script de inicialização simplificado.

No LCC-IoT , configuramos as 8 instâncias do sensor com um período para coleta de imagens e envio de telemetria de 2 minutos. Este período é satisfatório para a aplicação dado que não se esperam movimentos repentinos de entrada e saída de pessoas nas salas.

As câmeras do modelo *TecVoz* estão ligadas a um DVR através de conexões específicas por cabos coaxiais e são acessadas pelo serviço do sensor através do protocolo RTSP. A Figura 4 ilustra um dos *dashboards* da aplicação que apresenta a contagem atual de pessoas em cada câmera.

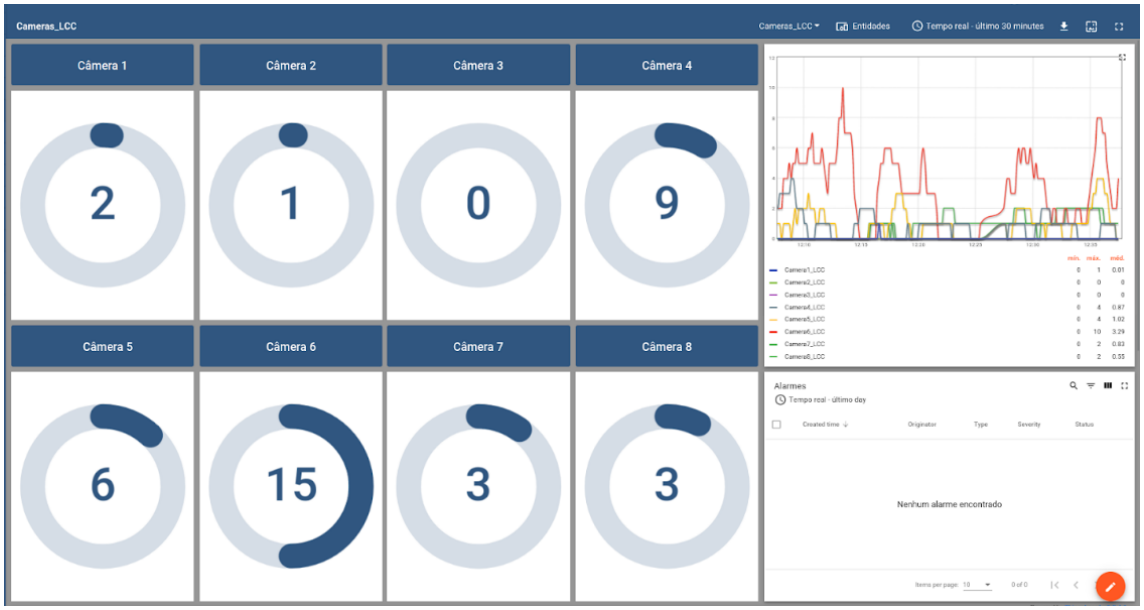

Figura 4. *Dashboard* do LCC-IoT com a contagem das 8 câmeras

Os processos das 8 instâncias do sensor executam em uma máquina virtual com sistema operacional Linux CentOS 7.9.2009 (Core), com 8 GB RAM e 12 CPUs virtuais, com 1 core por socket. A máquina hospedeira possui 71.9 GB RAM e 12 CPUs Intel Xeon ES-2603 v4 @ 1.70GHz. Quando o processo de cada câmera executa são consumidos, em média, 23% de CPU e 160 MB RAM da máquina virtual. O mecanismo adotado para controle de concorrência não permite que o consumo de recursos aumente muito além disso.

7. Conclusão

Este trabalho apresentou um sensor de software capaz de contar pessoas em uma cena de vídeo, para atender às demandas da aplicação LCC-IoT do Laboratório de Ciência da Computação (LCC) do IME/UERJ. O software desenvolvido foi registrado no INPI, com nome de *Sensor de Número de Pessoas na Cena* sob número BR512024004010-2 e utiliza o sistema YOLO que possibilita uma detecção rápida e precisa de pessoas nas imagens capturadas. A coleta e envio periódico de dados de telemetria para a aplicação permite a geração de alarmes e tomada de decisões no ambiente monitorado.

O sensor é adaptável a diferentes cenários e necessidades de monitoramento, considerando ainda a integração das diferentes partes da solução via protocolo MQTT e HTTP. Em relação à reutilização de recursos, a implementação do sensor segue as boas práticas de Engenharia de Software. Em particular, adotamos uma classe auxiliar para o envio dos dados coletados para o *broker* MQTT, permitindo a padronização e reutilização de recursos em outros projetos do laboratório que utilizam o mesmo protocolo de comunicação.

Como trabalhos futuros, pretende-se explorar soluções para melhorar a escalabilidade do sensor a fim de lidar com um número maior de câmeras e ambientes, atuando nas limitações de recursos e de infraestrutura. Outro ponto a ser explorado é em relação ao consumo elevado de processamento, dado que o sistema ainda exige uma quantidade significativa de recursos de CPU, tendo em vista a necessidade de execução de diversos processos por cada câmera, o que pode levar a problemas de desempenho e atrasos no processamento de imagens, especialmente em sistemas com recursos limitados.

Agradecimentos. Agradecemos aos programas ProDocência e ProCiência da UERJ. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Referências

Ahmed, E. et al. (2016). Internet-of-things-based smart environments: state of the art, taxonomy, and open research challenges. *IEEE Wireless Comm.*, 23(5):10–16.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Camlytics (2022). People counting solution. Web. `https://camlytics.com/solutions/people-counting` [Access: 07/2022].

Chen, C.-C., Lin, H.-H., and Chen, O. T.-C. (2011). Tracking and counting people in visual surveillance systems. In *2011 IEEE ICASSP*, pages 1425–1428.

Eclipse Paho (2018). Eclipse paho javascript client. Web page. `https://www.eclipse.org/paho/clients/js/`.

Eocortex (2022). People counting. Web. `https://eocortex.com/products/video-management-software-vms/people-counting`.

LCC, UERJ (2025). Sensor de pessoas em uma cena. Web. https://www.lcc.ime.uerj.br/sensor-pessoas/.

Motta, R. C., Batista, T. V., and Delicato, F. C. (2024). The Intersection of the Internet of Things and Smart Cities: A Tertiary Study. *JISA*, 15(1):325–341.

Mueller, V. (2021). Non-maximum suppression. `https://towardsdatascience.com/non-maxima-suppression-139f7e00f0b5`.

Redmon, J. (2013–2016). Darknet: Open Source Neural Networks in C. `http://pjreddie.com/darknet/`.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *CoRR, arXiv*, http://arxiv.org/abs/1804.02767.

Yang, Gonzalez-Banos, and Guibas (2003). Counting people in crowds with a real-time network of simple image sensors. In *9th IEEE ICCV*, pages 122–129 vol.1.