

INSTITUTO FEDERAL DO ESPÍRITO SANTO

ENGENHARIA DE CONTROLE E AUTOMAÇÃO

CONRADO COSTA

**ENSAIOS DE DETECÇÃO FACIAL EM DISPOSITIVOS DE BORDA,
PARA MONITORAMENTO E CONTROLE DE ACESSO EM ESPAÇOS
ABERTOS E INTELIGENTES**

SERRA – ES

2021

CONRADO COSTA

**ENSAIOS DE DETECÇÃO FACIAL EM DISPOSITIVOS DE BORDA,
PARA MONITORAMENTO E CONTROLE DE ACESSO EM ESPAÇOS
ABERTOS E INTELIGENTES**

Trabalho de conclusão de curso, apresentada como parte das atividades para obtenção do título de bacharel em Engenharia de Controle e Automação, do curso de Engenharia de Controle e Automação do Instituto Federal do Espírito Santo.

Orientador: Prof. Rafael Emerick Z de Oliveira

SERRA – ES

2021

Aqui entra a Ficha catalográfica.

Será gerada pela biblioteca!

CONRADO COSTA

**ENSAIOS DE DETECÇÃO FACIAL EM DISPOSITIVOS DE BORDA,
PARA MONITORAMENTO E CONTROLE DE ACESSO EM ESPAÇOS
ABERTOS E INTELIGENTES**

Texto submetido ao Curso de Graduação em Engenharia de Controle e Automação do Instituto Federal do Espírito Santo como requisito parcial para obtenção do título de Bacharel em Engenharia de Controle e Automação.

Aprovada em XX de XXXXX de XXXX.

COMISSÃO EXAMINADORA

Prof. Rafael Emerick Z de Oliveira
Instituto Federal do Espírito Santo - *campus Serra*

Prof. XXXXX
Instituto Federal do Espírito Santo - *campus Serra*

Prof. XXXXX
Instituto Federal do Espírito Santo - *campus Serra*

SERRA – ES

2021

DECLARAÇÃO DO AUTOR

Declaro, para fins de pesquisa acadêmica, didática e técnico-científica, que a presente Dissertação de Mestrado pode ser parcial ou totalmente utilizada desde que se faça referência à fonte e aos autores.

Conrado Costa

Serra, XX de XXXXX de XXXX.

Resumo

<a preencher>

Palavras-chave: Processamento Digital de Imagens; Inteligência Artificial; Rede Neural; Máquina de Vetor de Suporte;

Abstract

<a preencher>

Keywords: Digital Image Processing; Artificial Intelligence; Neural Network; Support Vector Machine.

Listas de ilustrações

Figura 1 – Fluxo de processo de reconhecimento de face.	20
Figura 2 – Características geométricas (destaque em branco), usada em experimentos de reconhecimento de faces.	21
Figura 3 – Arquitetura de computação distribuída	25
Figura 4 – Assinatura da função detectMultiScale do OpenCV.	27
Figura 5 – Interface e seus parâmetros.	29
Figura 6 – Exemplo de resultado retornado.	30
Figura 7 – Exemplo de resultado com poucas faces detectadas.	33
Figura 8 – Exemplo de resultado com várias faces detectadas e alguns falsos positivos.	34
Figura 9 – Exemplo de resultado possivelmente satisfatório.	35
Figura 10 – Exemplo de matriz de resultado com limites ajustados.	36
Figura 11 – Exemplo de resultado com as métricas.	38
Figura 12 – Imagem selecionada para testes da cena 1.	40
Figura 13 – Exemplo de variação de cena com redução de 40 faces.	41
Figura 14 – Dispositivos testados. À esqueda o Raspberry Pi 4 e à direita o Raspberry Pi Zero W.	47
Figura 15 – Otimização Cena 1 - resolução 1440p.	50
Figura 16 – Otimização Cena 1 - resolução 1080p.	52
Figura 17 – Otimização Cena 1 - resolução 720p.	54
Figura 18 – Tabela de Dados - resolução 720p.	55
Figura 19 – Faces detectadas e Tempos de execução por Variação de faces detectáveis.	57
Figura 20 – Faces detectadas e Tempos de execução por Variação de resolução.	59
Figura 21 – Comparativo de faces recortadas de diferentes resoluções, em tamanho real.	60
Figura 22 – Comparativo de tamanho médio de imagem encodada por face detectada em bytes.	60

Figura 23 – Comparativo de utilização de banda por quantidade de faces detectadas.	62
Figura 24 – Posicionamento para captura da face na primeira posição.	64
Figura 25 – Captura da imagem na primeira posição.	65
Figura 26 – Posicionamento para captura da face na segunda posição.	65
Figura 27 – Captura da imagem na segunda posição.	66
Figura 28 – Otimização Cena 2 - resolução 600p - matrizes. À esquerda posição 1 e à direita, posição 2	68
Figura 29 – Otimização Cena 2 - resolução 600p - faces detectadas. À esquerda posição 1 e à direita, posição 2	69
Figura 30 – Tabela de Dados - resolução 600p.	69
Figura 31 – Otimização Cena 2 - resolução 480p - matrizes. À esquerda posição 1 e à direita, posição 2	70
Figura 32 – Otimização Cena 2 - resolução 480p - faces detectadas. À esquerda posição 1 e à direita, posição 2	70
Figura 33 – Tabela de Dados - resolução 480p.	71
Figura 34 – Otimização Cena 2 - resolução 240p - matrizes. À esquerda posição 1 e à direita, posição 2	71
Figura 35 – Otimização Cena 2 - resolução 240p - faces detectadas. À esquerda posição 1 e à direita, posição 2	72
Figura 36 – Tabela de Dados - resolução 240p.	72
Figura 37 – Tempos de execução por Variação de resolução.	73
Figura 38 – Parcela de cada etapa no tempo de execução no Raspberry Pi 4, por resolução.	75
Figura 39 – Comparativo de faces recortadas de diferentes resoluções, em tamanho real.	76
Figura 40 – Comparativo de faces recortadas de diferentes resoluções, em tamanhos ajustados.	76
Figura 41 – Comparativo de utilização de banda.	78
Figura 42 – Utilização de banda em Mbps por quantidade de dispositivos e resolução.	79

Sumário

Lista de ilustrações	6
Sumário	8
 1 INTRODUÇÃO	10
 1.1 Objetivo Geral	12
1.1.1 Objetivos Específico	12
 1.2 Estrutura do Texto	13
 2 REVISÃO TEÓRICA	14
 2.1 O problema da segurança em espaços públicos e privados	14
 2.2 Smart cities e videomonitoramento inteligente	15
 2.3 Redes Neurais e Deep Learning	17
 2.4 Processamento de imagens, detecção e reconhecimento de faces	18
 2.5 A Internet das Coisas, Edge e Fog Computing	23
 3 DESENVOLVIMENTO	26
 3.1 O algoritmo de detecção de objetos	26
 3.2 Ferramenta de parametrização, otimização e obtenção de resultados	28
 3.3 Cenários de testes	39
3.3.1 Cena 1	39
3.3.1.1 Variação de quantidade de faces	40
3.3.1.2 Variação de resolução da imagem	42
3.3.1.3 Tempo médio de captura	43
3.3.2 Cena 2	43
3.3.2.1 Captura das imagens para teste	44
3.3.2.2 Variação de resolução da imagem	45
 3.4 Dispositivos testados	46
 4 EXPERIMENTOS REALIZADOS	48

4.1	Cena 1	48
4.1.1	Otimização de parâmetros	48
4.1.1.1	Resolução 1440p	49
4.1.1.2	Resolução 1080p	51
4.1.1.3	Resolução 720p	53
4.1.2	Análise dos resultados	55
4.1.2.1	Efeitos da variação por quantidade de faces	55
4.1.2.2	Efeitos da variação por resolução de imagem	57
4.1.2.3	Considerações sobre os dispositivos testados	60
4.1.2.4	Considerações sobre utilização de banda	61
4.2	Cena 2	63
4.2.1	Captura de imagens para teste	63
4.2.2	Otimização de parâmetros	66
4.2.2.1	Resolução 600p	67
4.2.2.2	Resolução 480p	69
4.2.2.3	Resolução 240p	71
4.2.3	Análise dos resultados	72
4.2.3.1	Efeitos da variação por resolução de imagem e considerações sobre os dispositivos testados	73
4.2.3.2	Considerações sobre utilização de banda	77
	5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS	80
	REFERÊNCIAS	81

1 INTRODUÇÃO

O problema de segurança pública sempre foi um problema sério no Brasil. Constantes crimes de furtos e roubos geram grande danos, principalmente financeiros, tanto para o indivíduo, no caso das vítimas, quanto para a sociedade (CERQUEIRA et al., 2007; G1, 2013). Um dos reflexos gerados por essa insegurança está no fato de que vários locais públicos devem ter seu acesso controlado e vigiado para garantir a segurança patrimonial.

As *smart cities* (cidades inteligentes) estão emergindo como uma prioridade para pesquisa e desenvolvimento em todo o mundo. Elas abrem oportunidades significativas em várias áreas, como crescimento econômico, saúde, bem-estar, eficiência energética e transporte, para promover o desenvolvimento sustentável das cidades (SONG et al., 2017). O conceito e oportunidades das *smart cities* são escaláveis para outros conceitos 'smart' como o *smart room*, *smart home*, *smart building*, etc (PACHECO et al., 2018). A proliferação de tecnologias de informação e comunicação possibilita o desenvolvimento de diversos serviços inteligentes. E um dos serviços comunitários mais essenciais é justamente a vigilância inteligente (CHEN et al., 2016; NIKOUEI et al., 2018).

Nos últimos anos, aplicações de reconhecimento facial a partir de imagens geradas por câmeras de videomonitoramento têm ganhado relevância, sendo largamente utilizadas para a verificação ou identificação de indivíduos em locais públicos. No âmbito da segurança, o reconhecimento facial em vídeo permite agilidade nas situações em que muitos indivíduos devem ser identificados rapidamente (QUIRITA, 2014).

Apesar de estarmos longe de conseguir com que a IA (inteligência artificial) se aproxime da performance humana, em algumas áreas, como reconhecimento de imagem, carros autônomos e jogos eletrônicos, ela se mostra equivalente, ou até mesmo superior (AGGARWAL, 2018). Em tarefas de visão computacional é possível rastrear o movimento de uma pessoa em um plano de fundo complexo. E com moderado sucesso, é possível tentar localizar e nomear todas as pessoas em uma fotografia,

através da detecção e reconhecimento de faces, roupas e cabelos (Szeliski, 2011).

O *deep learning* (aprendizagem profunda), ramo do *machine learning* (aprendizagem de máquina), tornou-se imensamente popular no reconhecimento de imagens, bem como em outras tarefas de reconhecimento e correspondência de padrões (Verhelst; Moons, 2017).

As redes neurais artificiais simulam o sistema nervoso humano com base em aprendizado de máquina, tratando as unidades computacionais em um modelo de aprendizado de maneira semelhante aos neurônios humanos. Não é uma tarefa fácil pois o poder computacional do computador mais rápido atualmente equivale a uma pequena fração do poder computacional de um cérebro humano (Aggarwal, 2018).

As *deep neural networks* (redes neurais profundas) envolvem uma complexidade computacional significativa, fazendo com que, até recentemente, seu processamento fosse viável apenas em plataformas de potentes servidores disponíveis na ‘nuvem’ (Verhelst; Moons, 2017). Quando é necessário armazenamento e computação de dados em larga escala, a computação em nuvem tem sido a solução. Porém, com o grande crescimento de dispositivos móveis e inteligentes, juntamente com as tecnologia de IoT (Internet das Coisas), o foco mudou para se obter respostas em tempo real. (Dolui; Datta, 2017).

Nos últimos anos, vê-se uma tendência de se incorporar o processamento de aprendizado profundo em dispositivos de borda, como celulares, dispositivos móveis e nos nós da IoT. Isso torna possível a análise de dados localmente, em tempo real, além de mitigar problemas de privacidade dos dados (Verhelst; Moons, 2017). Outro benefício da computação na borda (*Edge Computing*) é o descongestionamento da rede de dados, pois permite que o processamento seja feito próximo das fontes dos dados (Merenda; Porcaro; Iero, 2020). Assim, evita-se a comunicação desnecessária, que sobrecarrega não só a rede principal como também o datacenter na nuvem (Azam; Hu, 2014).

1.1 Objetivo Geral

Com este trabalho objetiva-se testar o desempenho de dispositivos de borda no processo de detecção de faces, avaliando sua capacidade de detecção e tempo de resposta para diferentes cenários, com possíveis aplicações de monitoramento inteligente e controle de acesso. Com os resultados obtidos, espera-se determinar, para cada cenário definido, se o dispositivo é capaz de processar de forma satisfatória a etapa de detecção de faces, e as vantagens de se realizar esse processo na borda, em uma arquitetura de processamento distribuído.

1.1.1 Objetivos Específico

Em um sentido mais estrito, pretende-se

- Definir diferentes cenários (com aplicabilidade para monitoramento inteligente e controle de acesso) e os requisitos a serem cumpridos, como tempos de respostas e capacidade de reconhecimento. Serão utilizadas imagens estáticas que representem cada cenário para os testes.
- Desenvolver uma ferramenta cliente-servidor para auxiliar na parametrização do algoritmo de detecção, buscando melhor otimização para cada cena e dispositivo, e na obtenção das métricas de desempenho.
- Analisar os resultados obtidos e determinar para quais cenários os dispositivos podem executar a detecção de face de forma satisfatória e quais os ganhos em se executar tal processamento na borda, principalmente no que se refere à utilização de banda na rede de um sistema com arquitetura de processamento distribuído, sua escalabilidade e tempos de resposta.

1.2 Estrutura do Texto

<a preencher>

2 REVISÃO TEÓRICA

2.1 O problema da segurança em espaços públicos e privados

O problema de segurança pública no Brasil é algo que está sempre em evidência. O Programa das Nações Unidas para o Desenvolvimento (Pnud), em seu relatório divulgado em 12/11/2013, constata que o Brasil apresentou a maior taxa de roubo da América Latina, segundo dados de 2011 repassados pelos países. Os dados apontam que para cada 100 mil habitantes no Brasil, há 572,7 ocorrências de roubo. E sabe-se que, na realidade, esse número tende a ser maior, tendo em vista que nem todos os roubos são reportados às autoridades (G1, 2013).

Estima-se um total de 15 milhões de ocorrências de roubos e furtos no Brasil no ano de 2003, incluído os casos que não foram notificados. E como parte da consequência, estima-se uma perda material de R\$ 8,4 bilhões (CERQUEIRA et al., 2007). Se corrigido para o ano de 2020 com base no IPCA (Índice de Preços ao Consumidor Amplo), esse valor seria de aproximadamente R\$ 20,2 bilhões. Esse cálculo foi feito com base em uma calculadora online disponível no site do Banco Central do Brasil (BCB). Esse é um problema que gera não só perdas para as vítimas dos roubos e furtos, mas também indiretamente para outros indivíduos da sociedade, uma vez que, essa transferência de valor pode ser considerada como recurso de oportunidade a serem aplicados no setor de crimes. Este, por sua vez, demanda recurso público para o seu combate, tornando o problema um causador não só de dano ao indivíduo, como também de custo social (ANDERSON, 1999; CERQUEIRA et al., 2007).

Os fatores que motivam ou favorecem esse tipo de crime são vários. A teoria das janelas quebradas, testada em um experimento por Philip Zimbardo, psicólogo de Stanford, propõe-se a explicar um deles. O experimento consistiu em deixar dois carros similares abandonados nas ruas de dois bairros diferentes de Nova Iorque, um nobre e outro na periferia. O que se observou foi quem o carro deixado na periferia foi atacado por vândalos nos primeiros 10 minutos, enquanto o segundo carro, deixado

no bairro nobre, ficou intocado por mais de uma semana. Então Zimbardo com uma marreta danificou parte do carro e o que se observou em seguida foi que várias pessoas que estavam transitando se juntaram ao carro, que, em poucas horas, estava completamente destruído. O que o experimento transmite é que a desordem e o crime estão de certa forma ligados. A desordem passa uma impressão de descuido, de forma que um indivíduo mal intencionado se sentirá muito mais à vontade em cometer algum delito, por ter a sensação de que ninguém irá notar, ou se importar. E por isso, há também uma sensação de impunidade. Uma propriedade mal cuidada se torna ideal para os que saem na intenção de vandalizar ou saquear, e até mesmo para aqueles quem nem pensariam em tais atitudes, mas as cometem ao enxergar o delito como uma oportunidade (WILSON; KELLING, 1982).

Sistemas de câmeras de vigilância têm sido cada vez mais implantados em muitos lugares, como prédios, ruas, instalações industriais e comerciais, escolas, shoppings, aeroportos e residências, provendo, segurança pública, monitoramento de ambientes internos, monitoramento de tráfego e proteção de infraestrutura (PUVVADI et al., 2015).

Em uma das formas de monitoramento, as imagens da câmera são monitoradas em tempo real por seguranças. Como outra forma, é possível registrar a saída de cada câmera no um gravador (VCR), para futura análise. Porém, na primeira forma de monitoramento, uma ocorrência ou incidente de segurança pode acabar não sendo verificado, devido, por exemplo, a uma falha humana. E no segundo caso, o momento da verificação do ocorrido pode não acontecer em um tempo satisfatório (OLSON, 2006; Ramos Lima; Marques Ciarelli, 2019). Mas hoje, com o avanço da tecnologia, temos opções mais inteligentes de monitoramento, como é tratado no item 1.2.

2.2 *Smart cities* e videomonitoramento inteligente

“Smart City” é um poderoso paradigma que aplica as mais avançadas tecnologias de comunicação aos ambientes urbanos, com o objetivo de melhorar a qualidade de vida nas cidades e fornecer um amplo conjunto de serviços de valor tanto para os cidadãos quanto à administração (CENEDESE et al., 2014). O recente conceito de Smart Cities, impulsionado pelo rápido crescimento da IoT (Internet das Coisas), atraiu

a atenção de planejadores urbanos e pesquisadores para aumentar a segurança e o bem-estar dos residentes. A proliferação de tecnologias de informação e comunicação conecta sistemas ciber-físicos e entidades sociais, bem como possibilita muitos sistemas inteligentes. Um dos serviços comunitários inteligentes mais essenciais é a vigilância inteligente (CHEN et al., 2016; NIKOUEI et al., 2018).

Nos últimos tempos, os sistemas de câmeras de vigilância evoluíram de simples aquisição de vídeo e sistemas de exibição para sistemas semiautônomos inteligentes, capazes de realizar procedimentos complexos. Hoje em dia, um sistema de vigilância por vídeo pode integrar alguns dos algoritmos de análise de imagem e vídeo mais sofisticados como de classificação (por exemplo, redes neurais), reconhecimento de padrões, tomada de decisão, aprimoramento de imagem e vários outros (TSAKANIKAS; DABIUKLAS, 2018). Isso permite uma grande possibilidade de aplicações como controle de acesso em áreas de interesse, reconhecimento de faces humanas, detecção de padrões e objetos, reconhecimento de comportamento, estatísticas de fluxo de multidões, análise de congestionamento, etc (HU et al., 2004).

Um sistema de vigilância moderno compreende não só dispositivos de aquisição de imagem e vídeo para exibição, mas também dispositivos para processamento de dados e unidades de armazenamento, componentes cruciais para a execução da tarefa (TSAKANIKAS; DABIUKLAS, 2018). Além disso, têm estado cada vez mais disponíveis dispositivos de vigilância com conectividade de rede que suportam o protocolo IP. Isso abre uma ainda maior gama de possibilidades já que os dados podem ser enviados praticamente para qualquer de equipamento onde quer que esteja localizado. Porém, isso traz junto a preocupação com a privacidade, pois os dados de imagem trafegados em rede estão sujeitos a interceptações. Uma abordagem utilizada é a criptografia dos dados em tráfego, porém isso traz uma carga maior de processamento que pode acabar prejudicando a performance do monitoramento em tempo real (PUVVADI et al., 2015).

Muitas das aplicações de videomonitoramento inteligente requerem recursos computacionais e de armazenamento significativos, para ser capaz de lidar com a grande quantidade de dados gerada pelos sensores de vídeo. De acordo com o estudo recente, os dados de vídeo dominam o tráfego em tempo real e criam uma carga de trabalho

pesada nas redes de comunicação. Por exemplo, vídeo online responde por 74% de todo o tráfego online em 2017 e 78% do tráfego móvel será de dados de vídeo em 2021. O volume de dados são cada vez maiores, à medida que se têm maiores taxas de quadro e maiores resoluções (PORTER; FRASER; HUSH, 2010).

O paradigma da computação em nuvem, ou Cloud Computing, oferece excelente flexibilidade para lidar com essa grande quantidade de transferência de dados, além de também de ser escalável, correspondendo ao número crescente de câmeras de vigilância (NIKOUEI et al., 2018). Para tarefas de vigilância urbana que requerem a combinação de dados complexos, a computação em nuvem têm sido amplamente aceita como a solução (CHEN et al., 2016). No entanto, existem obstáculos significativos para a arquitetura de vigilância inteligente baseada em nuvem remota (NIKOUEI et al., 2018). Os delay adicional devido à comunicação em rede pode não ser tolerável em aplicações que sejam sensíveis a latências mais altas, como as aplicações de tempo real (CHEN et al., 2016). Uma grande distância entre o sensor de vídeo e os servidores da nuvem, além dos possíveis congestionamentos na rede, torna ainda mais inviável essa abordagem.

2.3 Redes Neurais e Deep Learning

Deep Learning, ou aprendizado profundo, é um campo do aprendizado de máquina baseado nas em redes neurais artificiais (BROWNLEE, 2019).

“Redes neurais artificiais são técnicas populares de aprendizado de máquina que simulam o mecanismo de aprendizado em organismos biológicos. O sistema nervoso humano contém células, conhecidas como neurônios. Os neurônios são conectados uns aos outros com o uso de axônios e dendritos, e as regiões de conexão entre os axônios e dendritos são chamadas de sinapses” (AGGARWAL, 2018).

Na programação convencional, o programador, através de várias linhas precisas de código, determina quais tarefas que o computador deve executar, à risca. Grandes problemas são quebrados em problemas menores nos quais os computadores conseguem performar. Já no paradigma de redes neurais, o programador não precisa

dizer exatamente o que a máquina deve fazer. Ao invés disso, a partir de algoritmos de aprendizado e dados observacionais, a máquina consegue aprender por si só como resolver determinado problema (NEAPOLITAN, 2018).

Existem várias diferentes arquiteturas de redes neurais que são comumente usadas em diferentes aplicações, como as *Restricted Boltzmann Machines* (RBM), as *Recurrent Neural Networks* (RNN) e as *Convolutional Neural Networks* (CNN). As mais utilizadas atualmente são as CNN e as RNN. As RNN, ou redes neurais recorrentes, são projetadas para trabalhar com dados sequenciais, como frases de texto, séries temporais e outras sequências discretas, como sequências biológicas. As CNN, ou redes neurais convolucionais, são redes inspiradas biologicamente e são usadas em visão computacional para classificação de imagens e detecção de objetos. (AGGARWAL, 2018).

As redes neurais convolucionais têm se mostrado a mais bem-sucedidas de todos os tipos de redes neurais. São amplamente usadas para o reconhecimento de imagem, detecção de objetos, rastreamento e até mesmo processamento de texto. O desempenho dessas redes chegou a superar o desempenho dos humanos no problema de classificação de imagens (HE et al., 2016).

Existem hoje disponíveis gratuitamente vários frameworks e estruturas de aprendizado de máquina incluindo redes neurais, tornando o trabalho de treinar uma rede neural mais simples. Entre as mais conhecidas estão Keras, PyTorch, TensorFlow, Scikit-learn. O treinamento de redes neurais exige muito poder de processamento. Os cálculos de deep learning tendem a ser mais rápidos quando feitos por GPUs (*Graphical Processing Units*) (HELLER, 2019).

2.4 Processamento de imagens, detecção e reconhecimento de faces

Pesquisadores têm desenvolvido na área de visão computacional técnicas matemáticas para recuperar a forma tridimensional e a aparência de objetos em imagens. Hoje, há técnicas confiáveis para calcular com precisão um modelo 3D parcial de um ambiente a partir de milhares de fotografias sobrepostas. A visão computacional

é usada hoje em uma ampla variedade de aplicativos do mundo real, que incluem reconhecimento óptico de caractere (OCR), segurança automotiva, videomonitoramento, reconhecimento de digitais, detecção e reconhecimento de faces, entre outros (Szeliski, 2011).

A detecção de faces é um problema já bem resolvido na área de visão computacional. Isso deve-se ao fato de que a detecção de faces é um dos processos mais utilizados em sistemas de videomonitoramentos e é exigido em vários tipos de aplicações como reconhecimento de faces, rastreamento, análise comportamental, etc (Zafeiriou; Zhang; Zhang, 2015).

“O objetivo da detecção de face é, em primeiro lugar, determinar se algum rosto está representado em uma cena e, em segundo lugar, calcular e retornar as coordenadas dos rostos detectados. Esta tarefa envolve muitas condições não triviais, como variações de escala, localização, orientação e pose, bem como condições de iluminação, expressões faciais e oclusões” (TSAKANIKAS; DAGIUKLAS, 2018).

Entre várias técnicas detecção de face, o trabalho inovador de Viola e Jones (Viola and Jones, 2001) baseado em melhorar o processo de detecção de faces, foi o primeiro algoritmo que tornou a detecção de rosto praticamente viável em aplicações do mundo real. Até hoje é amplamente aplicada em câmeras digitais e softwares de organização de fotos (Zafeiriou; Zhang; Zhang, 2015). A abordagem proposta por Viola e Jones para detecção de objetos minimiza o tempo de processamento ao mesmo tempo em que consegue grande acurácia na detecção. E quando aplicado na detecção de faces, se mostrou 15 vezes mais rápido que qualquer abordagem precedente (VIOLA; JONES, 2001). A biblioteca OpenCV, multiplataforma e de uso totalmente livre (OpenCV), disponibiliza uma função de detecção de objeto baseada no método proposto por Viola e Jones.

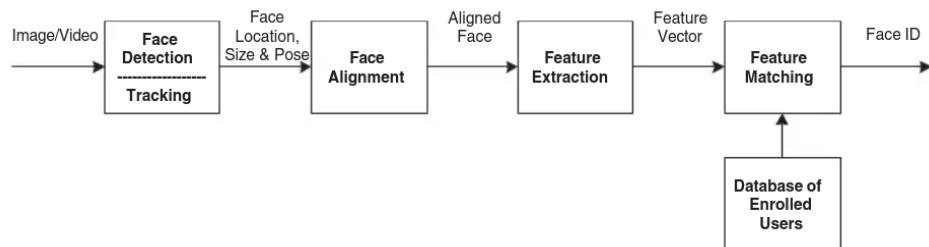
O reconhecimento de faces constitui o problema de identificar a face de uma pessoa mediante a comparação com uma base de dados de inúmeras outras faces previamente identificadas, a partir da qual obtém-se o grau de similaridade entre cada comparação (Quirita, 2014). O problema de reconhecimento de face requer que um rosto já tenha sido detectado em uma imagem, portanto, a detecção de face se torna um

pré-requisito para o processo de reconhecimento. Nos últimos tempos, os algoritmos de reconhecimento de faces evoluíram de tal forma que hoje podem ser usados em aplicativos do mundo real e em ambientes não controlados. (ZAFEIRIOU; ZHANG; ZHANG, 2015).

Uma das primeiras abordagens para reconhecimento de face baseia-se em localizar as características distintas da imagem, como olhos, nariz, boca, e medir a distância entre as posições de cada uma (FISCHLER; ELSCHLAGER, 1973; KANADE, 1977). Abordagens mais recentes baseiam-se na comparação de imagens em escala de cinza projetadas em subespaços dimensionais inferiores chamados de eigenfaces (Szeliski, 2011).

Um sistema de reconhecimento de faces, baseado em características, geralmente consiste de quatro partes: detecção, alinhamento, extração de características e combinação/verificação, conforme é esquematizado na figura 2.1 (Stan Z. Li, 2011).

Figura 1 – Fluxo de processo de reconhecimento de face



Fonte: (Stan Z. Li, 2011)

Na etapa de detecção “Face Detection”, temos um vídeo ou uma imagem como entrada. Essa etapa tem como função localizar e extrair uma ou mais faces que estejam presentes na imagem. No caso em que a entrada é um vídeo, se necessário, essa etapa também é responsável pelo rastreamento, “Tracking”, das faces, quadro a quadro. As faces extraídas passam então pela etapa de alinhamento, “Face Alignment”, que é responsável por normalizar a imagem em rotação e em escala, de forma que na imagem resultante a face esteja alinhada ao eixo horizontal do plano. (Stan Z. Li, 2011; QUIRITA, 2014).

Depois de já alinhada, a face passa pela etapa de extração de características, “Feature Extraction”. Essa etapa tem a função de identificar e extrair da face pontos

faciais distintos, como olhos, nariz, boca e outras marcas de referência que sejam suficientes para caracterizar a face, de forma que a mesma possa ser distinguida dentre as faces obtidas de outras pessoas. (Stan Z. Li, 2011). Em seguida são computadas as relações geométricas entre esses pontos faciais, reduzindo-se assim a imagem facial de entrada em um vetor de características geométricas (JAFRI; ARABNIA, 2009). A figura 2.2 destaca alguns pontos faciais característicos e as relações geométricas entre eles.

Os primeiros trabalhos realizados em reconhecimento de faces baseavam-se principalmente nessa técnica. Em uma das primeiras tentativas empregou-se um método simples de processamento de imagem para extrair um vetor de 16 parâmetros faciais, que eram relações de distâncias, áreas e ângulos, de forma a compensar o tamanho variável das imagens. Então usou-se uma medida de distância euclidiana simples para correspondência, chegando a um desempenho máximo de 75% em um banco de dados de 20 pessoas diferentes, usando 2 imagens por pessoa. (KANADE, 1977)

Figura 2 – Características geométricas (destaque em branco), usada em experimentos de reconhecimento de faces



Fonte: (Stan Z. Li, 2011)

Tendo as características geométricas da face calculadas e codificadas em um vetor, a próxima etapa para o reconhecimento da face é fazer a correspondência, ou “Feature Matching” (quarta e última etapa do processo de reconhecimento proposto na figura 1.2). Essa etapa consiste em comparar o vetor obtido da face que se deseja reconhecer com um banco de dados de vetores de outras faces que já são conhecidas e tiveram suas características extraídas pelo mesmo processo descrito até aqui. O reconhecimento é feito com base no grau de similaridades entre a face sendo verificada e as faces conhecidas (QUIRITA, 2014).

A principal vantagem oferecida pelas técnicas de reconhecimento baseadas em características é que, uma vez que a extração dos pontos característicos é anterior à análise feita para comparar a face com a de um indivíduo conhecido, esses métodos são relativamente robustos para variações de posição na imagem de entrada (JEBARA, 1996). A princípio, esquemas baseados em características podem ser invariáveis ao tamanho, orientação ou iluminação (COX; GHOSN; YANILOS, 1996). Outras vantagens de se utilizar técnicas baseadas em características faciais incluem a compactação de representação das imagens de face e a alta velocidade na tarefa de verificação de correspondência (identificação) (BRUNELLI; POGGIO, 1992).

A principal desvantagem dessa técnica é a dificuldade de detecção automática de características e o fato de que o desenvolvedor da aplicação deve tomar decisões arbitrárias sobre quais características são importantes para o processo de reconhecimento (CENDRILLON; LOVELL, 2000).

A biblioteca Dlib C++ disponibiliza algumas ferramentas para reconhecimento facial. Em seu site, está disponível gratuitamente um modelo pré-treinado para extração de características faciais, que alcançou uma marca de precisão de 99,38% no benchmark de reconhecimento facial da *Labeled Faces in the Wild* (LFW), que é comparável a outros métodos de última geração para reconhecimento facial em fevereiro de 2017. (DLIB...,). A LFW é uma referência pública para verificação de desempenho de reconhecimento facial (LABELLED...,).

Esse modelo mapeia a imagem de uma face humana em um vetor de 128 dimensões espaciais. Quando vetores de duas imagens diferentes são muito próximos, significa que a face tende a pertencer à mesma pessoa. Assim, o reconhecimento facial pode ser obtido mapeando-se várias faces em vetores de 128 dimensões e compará-las entre si calculando a distância Euclidiana (DLIB...,).

Esse modelo foi treinado a partir de um banco de cerca de 3 milhões de imagens. A precisão alcançada de 99,38% significa que, ao ser apresentada um par de imagens faciais, a ferramenta identificará corretamente se o par pertence à mesma pessoa ou é de pessoas diferentes em 99,38% das vezes (DLIB...,), o que o torna uma boa ferramenta para reconhecimento de faces de uma forma geral.

O grande avanço nas técnicas e ferramentas de aprendizado de máquina abriu um grande leque de possibilidades de aplicações de inteligência artificial nos últimos anos. Devido a isso, os recentes modelos de detecção e reconhecimento, não só de faces, mas de objetos em geral, estão cada vez mais precisos e rápidos. Isso, aliado ao expressivo avanço da Internet das Coisas (IoT), trazendo à tona o conceito de Edge Computing, torna cada vez mais viável as implementações de videomonitoramento automático.

2.5 A Internet das Coisas, Edge e Fog Computing

A Internet das Coisas, ou Internet of Things (IoT), é um paradigma de comunicação recente que prevê em um futuro próximo que os objetos da vida cotidiana serão equipados com microcontroladores, transceptores para comunicação digital e pilhas de protocolo adequadas que os tornarão capazes de se comunicarem entre si e com os usuários finais, tornando-se parte integrante da Internet (ATZORI; IERA; MORABITO, 2010).

O conceito de IoT, portanto, visa tornar a Internet ainda mais imersiva e abrangente. Além disso, ao permitir fácil acesso e interação com uma ampla variedade de dispositivos, como, por exemplo, eletrodomésticos, câmeras de vigilância, sensores de monitoramento, atuadores, monitores, veículos e assim por diante, a IoT promoverá o desenvolvimento de uma série de aplicações que fazem uso da quantidade e enorme variedade de dados gerados por tais objetos para fornecer novos serviços aos cidadãos, indústrias e administrações públicas. Esse paradigma encontra aplicação em muitos domínios diferentes, como automação residencial, automação industrial, assistência médica, saúde móvel, assistência a idosos, gerenciamento de energia inteligente e redes inteligentes, automotivo, gerenciamento de tráfego e muitos outros (Bellavista et al., 2013).

Com o advento da Internet das coisas nós estamos em uma era onde haverá uma grande quantidade de dados gerados por coisas que estão imersas em nosso dia a dia. Conforme estimado pelo Cisco Global Cloud Index, em 2019, os dados produzidos por pessoas, máquinas e “coisas” chegaria a 500 zetabytes (??). Para processar esses

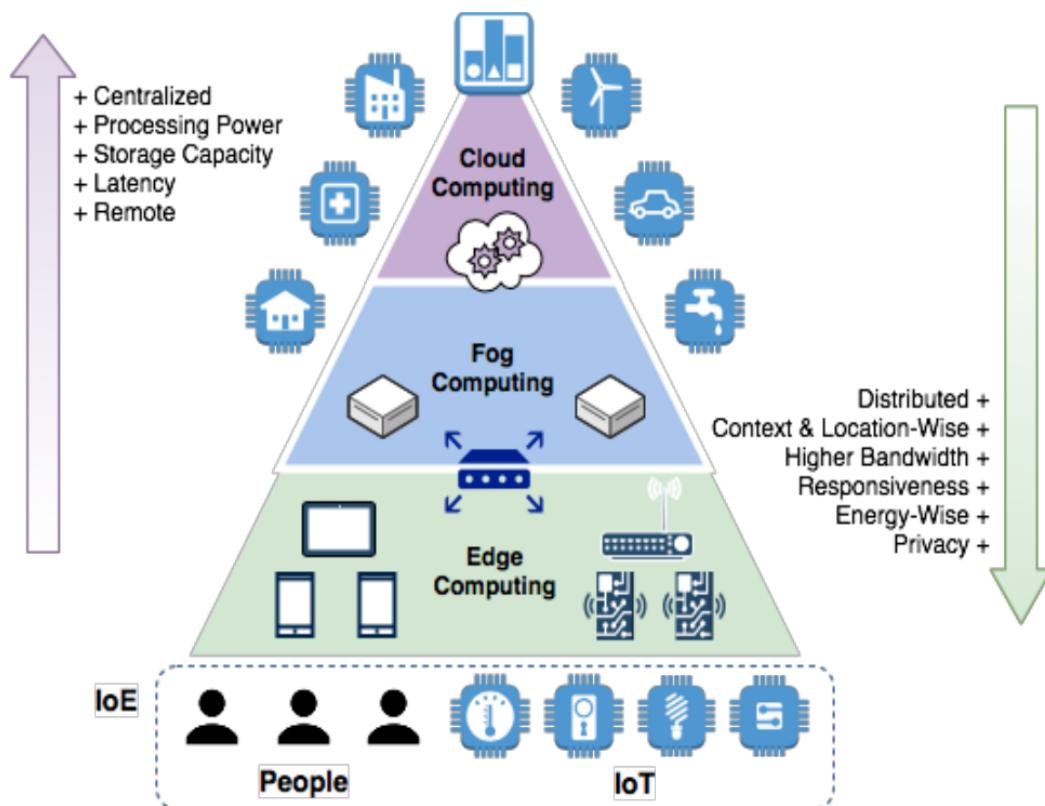
todos esses dados, é necessário muito poder computacional.

Hoje em dia, a computação em nuvem é uma plataforma econômica e prevalecente, oferecendo enorme poder de processamento e capacidade de armazenamento para treinamento de modelos de machine learning, reconhecimento facial, reconhecimento de fala, visão computacional, processamento automatizado de linguagem, classificação de texto e diversas aplicações de IoT e smart cities. No entanto, também tem algumas desvantagens importantes, como latência de resposta da rede e a segurança do sistema no que diz respeito a questões de privacidade, já que para chegar na nuvem os dados, possivelmente sensíveis, precisam trafegar por um longo caminho na rede mundial (PACHECO et al., 2018).

A maioria das ações de controle de IoT deve ser realizada em tempo real, portanto, o tempo de espera de processamento em nuvem, principalmente devido à latência da rede, não funciona bem para problemas de IoT (SINGH, 2017). Na tentativa de contornar algumas dessas limitações, aparecem alguns paradigmas recentes e complementares. São o Fog e o Edge Computing, que promete a capacidade de realizar tarefas de uma forma mais distribuída e responsiva, uma vez que os nós de IoT estão mais próximos das fontes de dados dos sensores. Além disso, também reduz o tráfego de rede e evita a exposição de dados privados do usuário (PACHECO et al., 2018).

A figura 2.3 demonstra muito bem a ideia de uma arquitetura de processamento distribuído, indicando as características que cada parte da rede dá à aplicação ao ser responsável por parte do processamento.

Figura 3 – Arquitetura de computação distribuída



Fonte: (PACHECO et al., 2018)

3 DESENVOLVIMENTO

3.1 O algoritmo de detecção de objetos

O algoritmo de detecção de objetos utilizado nesse estudo foi o *Haar cascade object detection*, proposto por Viola e Jones em sua pesquisa *Rapid Object Detection using a Boosted Cascade of Simple Features* (VIOLA; JONES, 2001), inicialmente proposto para detecção de faces, mas sendo possível ser utilizado na detecção de qualquer objeto. A biblioteca OpenCV fornece uma implementação desse algoritmo proposto para detecção de objetos (OpenCV-CascadeClassifier, 2022), que pode ser usado tanto para o treinamento de um novo modelo quanto para realizar a detecção em si, utilizando-se do modelo treinado ou de algum dos modelos pretreinados disponibilizados pelos próprios mantenedores do OpenCV em (OpenCV-PreTrainedModels, 2022). Neste trabalho foi utilizado o modelo pretreinado 'haarcascade_frontalface_alt.xml' para a detecção das faces.

Não é o algoritmo com melhor acurácia atualmente, se comparado com técnicas mais modernas que aplicam *deep learning*, porém é um algoritmo extremamente rápido e preciso, portanto ainda é muito útil para detecção de objetos em dispositivos com recursos limitados, como é o caso de dispositivos de borda, em geral.

Uma desvantagem nesse método é a tendência à detecção de falsos positivos. Existe também a necessidade de definição de alguns parâmetros na implementação fornecida no OpenCV, descritos resumidamente a seguir.

O método fornecido pelo objeto do tipo *CascadeClassifier* (OpenCV-CascadeClassifier, 2022) para performar a detecção em uma imagem é o *detectMultiScale*, que tem assinatura (em C++) conforme a figura 4. O modelo classificador pretreinado é carregado previamente utilizando-se do método *load* do mesmo objeto.

- *image* - matriz da imagem de entrada, na qual os objetos (no nosso caso, faces)

¹ Disponível em: <https://docs.opencv.org/3.2.0/d1/de5/classcv_1_1CascadeClassifier.html>

Figura 4 – Assinatura da função detectMultiScale do OpenCV.

```
void cv::CascadeClassifier::detectMultiScale ( InputArray image,
                                              std::vector< Rect > & objects,
                                              double scaleFactor = 1.1 ,
                                              int minNeighbors = 3 ,
                                              int flags = 0 ,
                                              Size minSize = Size() ,
                                              Size maxSize = Size()
)
```

Fonte: Documentação do OpenCV.¹

serão detectados;

- *objects* - referência ao vetor de retângulos que será populado com as áreas na imagem de entrada onde foram detectadas as faces;
- *scaleFactor* - esse parâmetro determina o quanto a imagem será reescalonada em cada passo durante a detecção. O modelo foi treinado com imagens de tamanhos fixos, então, para possibilitar a detecção de variados tamanhos de faces, a imagem tem sua resolução reduzida em vários passos, de forma que em algum determinado passo faces maiores tenham proporções próximas da utilizada no treinamento do modelo e, assim sejam detectadas. Por exemplo, um valor de 1.1 significa que a imagem será reduzida 10% a cada passo, um valor de 1.05, 5%. Quanto menor for o valor desse parâmetro maiores serão as chances de uma face se encaixar ao tamanho detectável em algum passo, porém mais lento será a resposta do algoritmo já que será exigido mais processamento conforme aumenta a quantidade de passos.
- *minNeighbors* - especifica quantos vizinhos cada retângulo candidato a uma face deve ter para ser considerado como uma detecção positiva. Em resumo, esse parâmetro afeta na qualidade da resolução. Valores baixos podem detectar maiores quantidades de faces porém com a presença de muitos falsos positivos, enquanto valores maiores tendem a aumentar a confiabilidade do que é detectado porém podendo aumentar a quantidade de falsos negativos (quando há uma face mas não é detectada pois não tem retângulos vizinhos suficientes);
- *flags* - não utilizado;
- *minSize* - indica o tamanho mínimo de face a ser detectada na imagem. Tamanhos mínimos maiores tendem a reduzir o tempo de resposta por aumentar o tamanho da

janela deslizante na varredura da imagem (o que demanda mais processamento), mas com risco de não detectar faces relativamente pequenas. Tamanhos mínimos menores aumentam as chances de detecção de faces relativamente pequenas, mas com aumento também na demanda de processamento devido à diminuição da janela deslizante.

-

O ajuste de parâmetros não é muito simples e depende do cenário da imagem, dos possíveis tamanhos de faces que se deseja detectar, etc. A escolha dos parâmetros influencia diretamente no resultado da detecção, na capacidade de detecção de determinados tamanhos de faces, na propensão em detectar falsos-positivos e no tempo de execução.

Devido a isso, viu-se necessário o desenvolvimento de uma ferramenta para se testar de uma só vez um range variável de valores de determinados parâmetros e verificar facilmente a qualidade e tempo de detecção para cada combinação de valores testados.

3.2 Ferramenta de parametrização, otimização e obtenção de resultados

O *design* da ferramenta foi pensado de forma a facilitar a análise do resultado de diversas combinações de parâmetros ao mesmo tempo, de diferentes imagens e em dispositivos diferentes, através de uma interface web.

Contém de duas partes: a parte cliente, uma interface web que pode ser executada em qualquer dispositivo através de um navegador, e a parte servidor, que deve rodar nos dispositivos que serão testados.

No cliente é feita a seleção da imagem a ser utilizada no teste, é definido o endereço do dispositivo a ser testado, rodando o servidor, e são definidos os parâmetros de testes. O cliente envia os dados para o servidor que executa o algoritmo de detecção conforme os parâmetros passados e retorna para o cliente o resultado de cada combinação de valores dos parâmetros solicitados. O resultado é composto por uma matriz contendo o

tempo médio de execução, a quantidade de faces detectadas e a posição de cada face detectada para cada par de valores dos parâmetros na matriz. As faces detectadas são visualizadas na imagem ao se selecionar um dos resultados, permitindo, assim, a verificação da qualidade da detecção e a presença de falsos-positivos.

Na figura a seguir temos um exemplo da interface com imagem e parâmetros selecionados, ainda sem exibição do resultado da análise.

Figura 5 – Interface e seus parâmetros.

Fonte: compilação do autor.²

À esquerda da interface há um elemento de entrada que permite a seleção a imagem a ser utilizada no ensaio. Na imagem, exibida logo abaixo, serão delimitadas as faces detectadas.

À direita há dois campos, "Host" e "Port", que permitem a seleção do dispositivo a ser testado, a partir do endereço IP e porta em que a aplicação estará "escutando".

Logo abaixo estão os parâmetros a serem definidos e enviados ao servidor para a execução do ensaio. Três deles, "Scale factor", "Min neighbors" e "Min size", são parâmetros passados na própria função *detectMultiScale* do OpenCV, e o significado de cada um foi descrito no início desta seção. O parâmetro "Number of Samples" determina quantas vezes o servidor executa cada combinação de parâmetros para obter um resultado médio.

² Imagem usada na compilação disponível em: <https://commons.wikimedia.org/wiki/File:WikiConference_North_America_Friday_Group_Photo.jpg> Arquivo de imagem sob a licença CC BY-SA 4.0: <<https://creativecommons.org/licenses/by-sa/4.0/deed.en>>

Para o parâmetro "Min Neighbors", é possível definir apenas um valor para cada execução, que será utilizado em todas as combinações de parâmetros. Já para os parâmetros "Scale Factor" e "Min Size" é possível definir valores mínimos, máximos e as quantidade de valores ("Steps") a serem distribuídos linearmente nos intervalos definidos para cada um dos dois parâmetros. O servidor executará todas as combinações possíveis a partir dos conjuntos de valores delimitados, e retornará uma matriz com um resultado para cada combinação.

Por fim, ao se clicar no botão "Send", a imagem e os parâmetros são enviados para o servidor. O servidor executa o algoritmo de detecção com todas as combinações possíveis e retorna uma matriz de resultados contendo número de faces detectadas e tempo médio de execução.

Figura 6 – Exemplo de resultado retornado.

Resultado

		Scale Factor				
		1.050	1.087	1.125	1.163	1.200
		All	Faces	Mean Time		
Min. Size (%)	0.5	F: 91 T: 3.157	F: 87 T: 1.675	F: 83 T: 1.254	F: 80 T: 0.963	F: 81 T: 0.831
		F: 90 T: 2.191	F: 87 T: 1.235	F: 83 T: 0.946	F: 79 T: 0.667	F: 81 T: 0.613
	1.5	F: 70 T: 1.249	F: 58 T: 0.656	F: 52 T: 0.457	F: 55 T: 0.41	F: 54 T: 0.342
		F: 23 T: 0.573	F: 15 T: 0.328	F: 14 T: 0.215	F: 20 T: 0.209	F: 13 T: 0.154
	2.5	F: 3 T: 0.347	F: 2 T: 0.179	F: 2 T: 0.132	F: 2 T: 0.111	F: 1 T: 0.11

Fonte: autor.

Como pode ser visto na figura 6, o resultado é exibido em forma de uma matriz. No eixo vertical têm-se a distribuição dos valores definidos para o parâmetro "Min. Size" e no eixo horizontal têm-se a distribuição dos valores definidos para o parâmetro "Scale Factor", de conforme os limites e quantidades de passos definidos para cada um.

Cada célula da matriz apresenta o resultado da detecção, utilizando a combinação de parâmetros corresponde, com base em duas métricas, a quantidade de faces detectadas, na parte superior da célula, e o tempo médio em segundos, na parte inferior da célula. Importante ressaltar que o tempo médio refere-se ao tempo que o algoritmo levou para a detecção de todas as faces detectadas, e não ao tempo médio de execução de cada face. A média se dá de acordo com a quantidade de vezes que o algoritmo de detecção foi executado para cada combinação de parâmetros, definido em "Number of Samples".

Em uma primeira análise, a partir resultado do exemplo dado na figura 6, é possível observar a tendência de se ter maior quantidade de faces detectadas, bem como maior tempo de execução, quanto mais ao topo e à esquerda está a célula na matriz. Essa observação é um tanto óvia tendo em vista que quanto menor o tamanho mínimo de faces a ser considerado pelo algoritmo (parâmetro "Min. Size"), mais faces candidatas tendem a ser encontradas e também mais iterações serão realizadas. O mesmo acontece para o parâmetro "Scale Factor", quanto menor o valor do parâmetro, menor o passo entre as escalas, maior a quantidade de imagens escalonadas avaliadas e, portanto, maior a chance de uma determinada face ser detectada, bem como maior quantidade de iterações realizadas.

A matriz de resultados por si só não é suficiente para se determinar que uma combinação de parâmetros é a mais adequada a se adotar. Isso se deve ao fato de que não se pode afirmar que o resultado com o maior número de faces detectadas é o melhor pois, dentro do conjunto de faces detectadas, alguma poucas ou até várias delas podem ser falsos positivos, de tal forma que a qualidade do resultado da detecção deva ser considerado ruim.

Outra questão a se considerar é que, dependendo do ambiente e do objetivo da aplicação, a detecção da maior quantidade de faces possível pode não ser a prioridade. Em determinado tipo de aplicação, pode ser mais interessante que o algoritmo detecte apenas faces que estejam mais próximas da câmera (e por conta disso relativamente maiores que as demais), com um tempo de resposta menor, do que detectar o máximo de faces possível, inclusive as mais distantes, que seriam descartadas, com um tempo de resposta maior.

Para tanto, é importante que haja uma análise qualitativa, ao se checar, para cada resultado analisado, quais foram as faces detectadas, resultantes da correspondente combinação de parâmetros. E, a partir da avaliação dos resultados disponíveis, comparando a presença de falsos positivos, a quantidade e tamanho das faces detectadas e o tempo médio de resposta, determinar uma combinação de parâmetros mais adequada a se adotar em uma aplicação de detecção no dispositivo testado ou concluir que o dispositivo não seria capaz de rodas a aplicação da forma desejada.

Para facilitar esse tipo de análise qualitativa, a matriz de resultados responde de forma interativa, de forma que, ao se clicar em uma célula da matriz, as faces detectadas a partir dos parâmetros correspondentes daquela célula são destacadas na imagem original através de retângulos verdes, facilitando assim a verificação de falsos positivos e quais faces presentes na imagem o algoritmo conseguiu detectar com tais parâmetros.

A seguir são apresentados alguns exemplos de resultados, a partir da matriz de resultado apresentada na figura 6. Para facilitar a visualização, serão apresentados apenas os cortes da matriz com o resultado selecionado e a imagem com as faces destacadas.

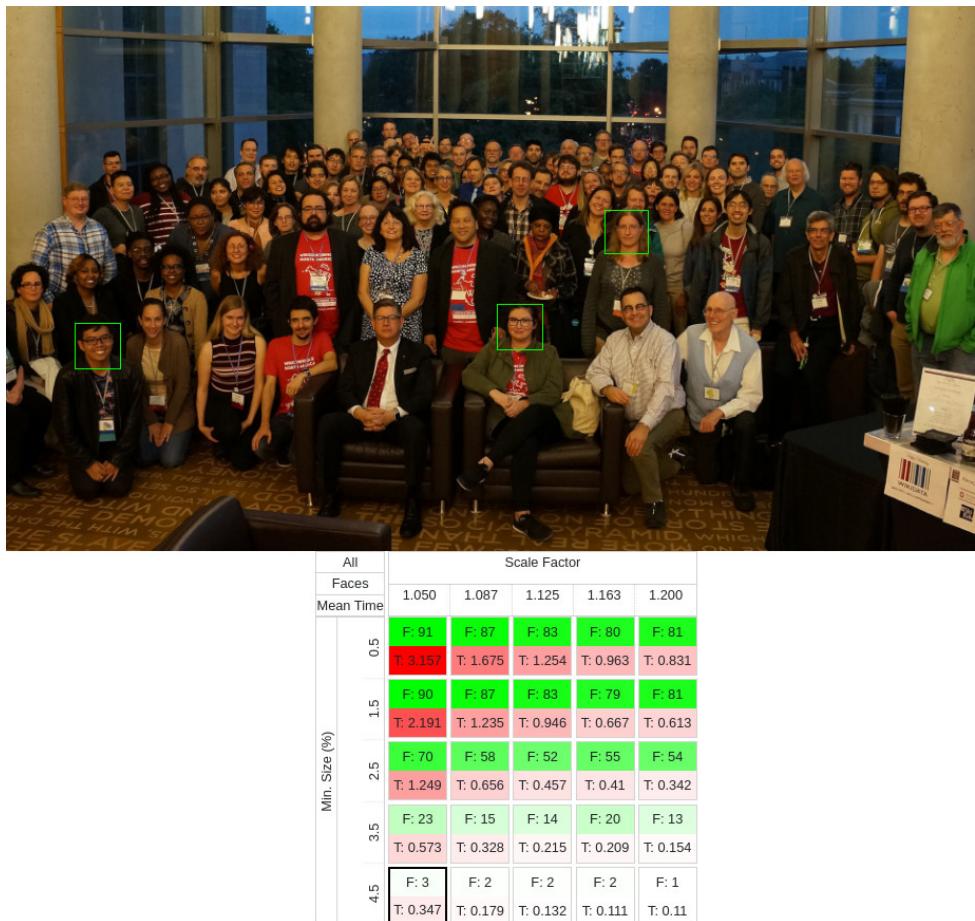
No exemplo da figura 7, têm-se um resultado com muito poucas faces detectadas devido ao valor relativamente alto do parâmetro "Min. Size".

Já a figura 8 apresenta um resultado com várias faces detectadas, inclusive pequenas faces ao fundo. Porém, devido aos valores relativamente baixos dos parâmetros "Min. Size" e "Scale Factor", vê-se claramente a presença de três falsos positivos, além de um tempo médio de detecção consideravelmente alto, acima de 3 segundos.

Por fim, a figura 9 apresenta um resultado que possivelmente pode ser considerado satisfatório para determinados tipos de aplicação. Nesse caso, a maioria das faces presentes na imagem foi detectada e com um tempo de resposta razoavelmente baixo, pelo menos se comparado ao resultado apresentado na figura 8.

Outra facilidade que a ferramenta traz é quanto à flexibilidade na escolha dos limites e quantidades de valores a serem testados para cada parâmetros, permitindo, assim,

Figura 7 – Exemplo de resultado com poucas faces detectadas.

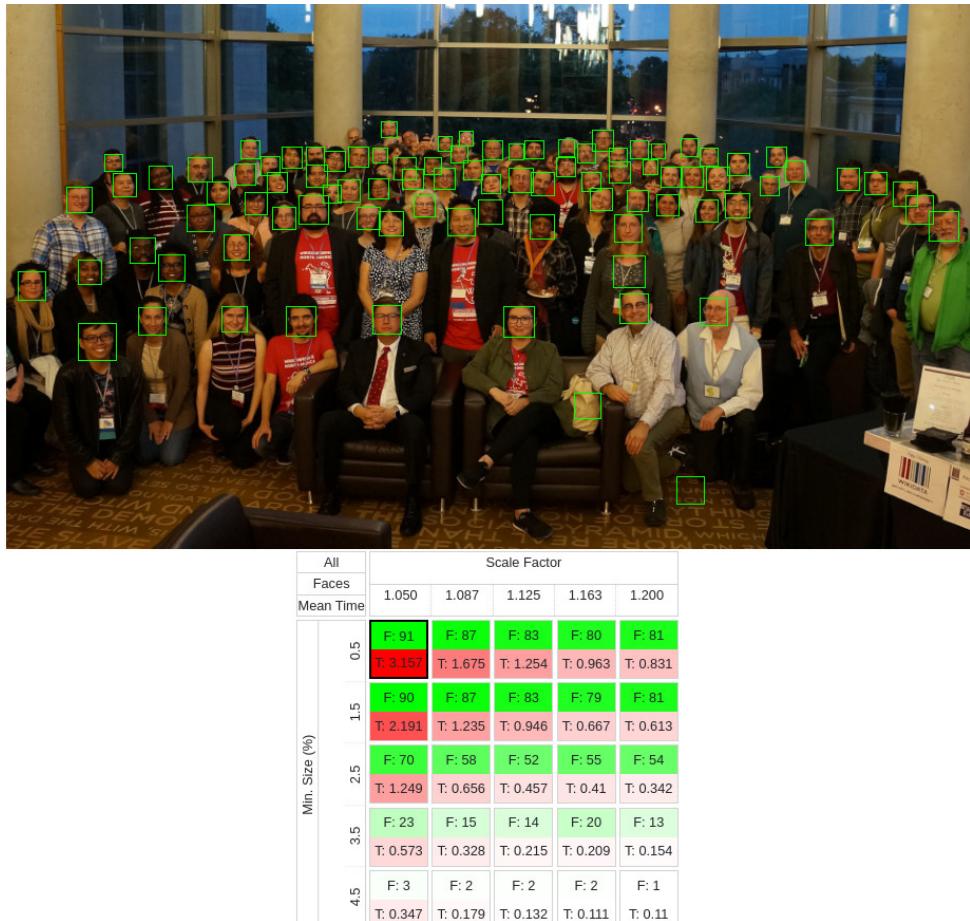


Fonte: compilação do autor.²

ajustar os mesmos iterativamente, de forma a se ter cada vez melhores resultados e, consequentemente, melhores opções de otimização.

Ainda partindo da matriz de resultados da figura 6, alguns limites podem ser ajustados para uma próxima rodada de análise. Por exemplo, observa-se que nos resultados em que o valor de "Min. Size" é maior que 2.5, a quantidade de faces detectadas é muito baixa. Caso seja considerado insatisfatório, o novo valor máximo na distribuição desse parâmetro pode ser definido como 2.5 ou 3.0. Analisando o parâmetro "Scale Factor", pode-se observar que valores abaixo de 1.125 resultam em um tempo de resposta muito alto e apresenta muitos falsos positivos. Além disso, alguns resultados com o valor máximo apresentado de "Scale Factor", 1.200, apresentam uma quantidade alta de faces detectadas, sendo possível que um valor maior possa apresentar um resultado igualmente bom mas com um tempo de resposta menor. Os limites de "Scale Factor" poderiam ser ajustados, por exemplo, para 1.080 e 1.250. Assim, uma próxima

Figura 8 – Exemplo de resultado com várias faces detectadas e alguns falsos positivos.



Fonte: compilação do autor.²

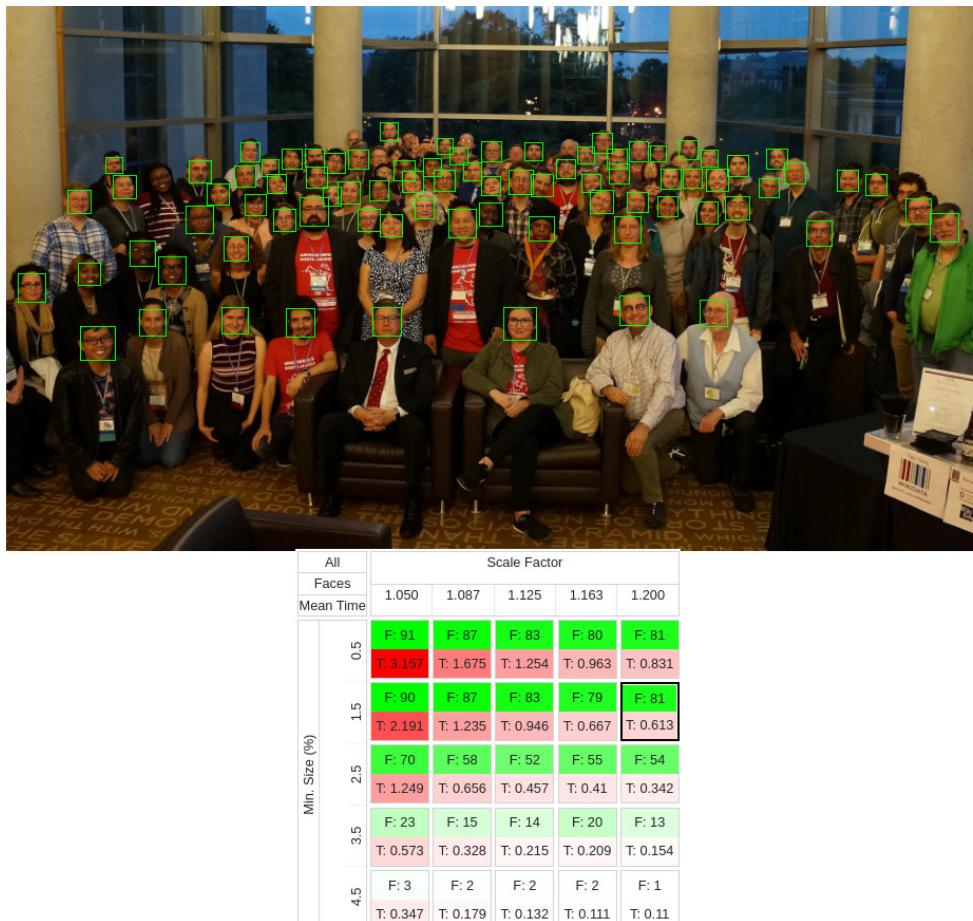
rodada de análise com os limites ajustados irão retornar uma maior e melhor gama de resultados.

A título de exemplo, a figura 10 apresenta a matriz de resultado com os limites ajustados. Observa-se uma melhor distribuição, com números de faces detectadas e tempo de resposta mais próximos do que podem ser considerados como satisfatórios, possibilitando uma análise mais precisa para determinar os melhores parâmetros.

Há de se observar que, pelo fato de a análise estar sendo feita a partir de apenas uma imagem, não é de se esperar que o resultado quanto à presença, ou não, de falsos positivos, seja o mesmo para todos os frames que serão analisados em uma aplicação real, mesmo a câmera estando fixa, capturando sempre o mesmo cenário.

Em uma aplicação real da ferramenta, é interessante que os valores escolhidos dos parâmetros sejam testados utilizando-se outras imagens da mesma cena com suas

Figura 9 – Exemplo de resultado possivelmente satisfatório.



Fonte: compilação do autor.²

possíveis variações, como por exemplo a diferente quantidade de pessoas e objetos, iluminações diferentes, etc.

Tendo definido-se uma melhor combinação de valores dos parâmetros, pode-se então obter uma análise mais aprofundada, com métricas adicionais, que auxiliará na análise e comparação dos resultados entre os diferentes cenários e dispositivos testados. Os resultados serão analisados considerando não só qualidade de detecção e tempo de resposta, como também a demanda de banda em rede e quantidade de dados trafegados para se progragar o resultado a uma próxima etapa de processamento dentro de um sistema distribuído.

Ao se clicar no botão "Get Metrics", no lado esquerdo da tela (vide figura 5), os parâmetros definidos de acordo com a célula selecionada na matriz de resultados são enviados ao servidor no dispositivo que está sendo testado. Este, por sua vez,

Figura 10 – Exemplo de matriz de resultado com limites ajustados.



Fonte: autor.

executa novamente o algoritmo de detecção, logando também o tempo de execução de outras etapas preparatórias, bem como outras informações referentes aos tamanhos das imagens.

Na figura 11, pode-se observar como os dados retornados são exibidos na interface do usuário. A seguir, uma breve explicação de cada item:

- 1.0 - Params: os parâmetros utilizados no algoritmos de detecção, conforme célula selecionada na matriz de resultados na etapa anterior;
- 1.1 - Full Image Resolution: a resolução da imagem original, em pixels;
- 1.2 - Full Image Size (bytes): o tamanho da imagem original, em bytes;
- 1.3 - Full Image Encoded Size (bytes): o tamanho da imagem original codificada em bitmap e base64, em bytes;
- 1.4 - Number of detected faces: o número de faces detectadas;

- 1.5 - Cropped Faces Images Total Size (bytes) / (% from full image size): o tamanho total das imagens de faces detectadas, recortadas da imagem original, em bytes, e o seu percentual com relação ao tamanho da imagem original;
- 1.6 - Encoded Faces Images Total Size (bytes) / (% from full image size): o tamanho total das imagens de faces detectadas, codificadas em bitmap e base64, em bytes, e o seu percentual com relação ao tamanho da imagem original codificada;
- 2.1 - Loading Image (s): o tempo de carregamento da imagem original (leitura em disco), em segundos;
- 2.2 - Convert Image to Gray (s): o tempo de conversão da imagem original para escala de cinza, em segundos. Etapa anterior necessária para execução do algoritmo de detecção;
- 2.3 - Detection (s): o tempo de execução do algoritmo de detecção em si, em segundos;
- 2.4 - Build Encoded Faces Images (s): o tempo de execução da etapa de encodamento das imagens em base64, em segundos;
- 2.5 - Total execution time (s): o tempo total de execução de todas as etapas, desde o carregamento da imagem até o encodamento em base64, em segundos.
- 3.1 - Faces images: as imagens das faces detectadas, recortadas da imagem original em seu tamanho real. Permite-se ter uma ideia da qualidade de resolução individual das faces, bem como facilita a identificar mais facilmente a presença de falsos positivos que possam não terem sido identificados na etapa de análise anterior.

É importante ressaltar que, como os testes são feitos a partir de imagens pré-selecionadas, a métrica 2.1 considera o tempo de carregamento em memória da imagem salva em disco. O problema é que esse tempo é limitado apenas pela velocidade de leitura no sistema de arquivos, enquanto que em uma aplicação real a aquisição das imagens se darão a partir de uma fonte, como um sensor conectado à placa, uma câmera USB ou via streaming, por exemplo.

Figura 11 – Exemplo de resultado com as métricas.

Get Metrics	
Metric	Value
1.0 - Params	Min. Size Face: 1.5 / Scale Factor: 1.208 / Min. Neighbors: 3
1.1 - Full Image Resolution	1080 x 1920
1.2 - Full Image Size (bytes)	6.220.800
1.3 - Full Image Encoded Size (bytes)	8.294.472
1.4 - Number of detected faces	82
1.5 - Cropped Faces Images Total Size (bytes) / (% from full image size)	543.111 / 8.7%
1.6 - Encoded Faces Images Total Size (bytes) / (% from full image encoded size)	738.056 / 8.9%
2.1 - Loading Image (s)	0.274
2.2 - Convert Image to Gray (s)	0.004
2.3 - Detection (s)	0.654
2.4 - Build Encoded Faces Images (s)	0.008
2.5 - Total execution time (s)	0.94
3.1 - Faces images	

Fonte: autor.

Para se ter no estudo um tempo de aquisição de imagem mais realista, foram feitos testes à parte para se obter o tempo médio de captura a partir de um módulo de câmera conectado diretamente ao dispositivo. Foi utilizado um módulo com sensor OmniVision OV5647, com capacidade de capturar imagens com resolução de até 2592x1944, e interface CSI, próprio para ser utilizado com um Raspberry Pi. Para captura de imagens, foi utilizado o pacote *picamera*, que provê uma interface simplificada em Python para o módulo da câmera.

Segundo a própria documentação (Picamera, 2022), o pacote *Picamera* oferece diferentes formas de se fazer a captura de imagens, chamados de *ports*. De acordo com o *port* utilizado, a câmera se comporta de maneira diferente durante a captura, o que irá influenciar na qualidade da imagem e tempo de aquisição dos frames. Como cada cena possui uma proposta diferente de aplicação, os testes de tempo médio de captura foram feitos de forma diferentes e mais adequada a cada cena e serão melhores detalhados nas próximas subseções da seção a seguir.

3.3 Cenários de testes

Nesta seção são apresentados os dois cenários definidos para os testes. Cada cenário representa uma possível aplicação diferente e possui diferentes requisitos que servirão de balizas para a calibração e comparação dos resultados.

Para um estudo mais completo, serão testadas algumas variações de cada cenário, seja quanto à quantidade de faces presentes e/ou quanto à resolução da imagem testada, além, claro dos diferentes dispositivos que serão testados.

3.3.1 Cena 1

Nessa primeira cena deseja-se representar o monitoramento de um espaço aberto e amplo, onde é esperado um fluxo alto de pessoas e que estas possam estar a qualquer distância da câmera, sendo desejável que o dispositivo consiga detectar faces muito pequenas a ponto de maximizar quantidade de faces detectadas.

- **Variações possíveis** - variação de quantidade de faces e variação de resolução da imagem.
- **Requisitos mínimos** (para balizar a parametrização) - máximo 5 segundos de resposta.
- **Imagen para testes** - para representar esta cena, foi selecionada uma imagem (figura 12) com várias faces olhando na direção da câmera e em diferentes distâncias. Ter todas as faces olhando para a mesma direção, distancia-se de um cenário real, mas torna-se ideal para o estudo pois serve como um pior caso para a cena e facilita a obtenção de variações por quantidade de faces.

Figura 12 – Imagem selecionada para testes da cena 1.



Fonte: Wikimedia Hackathon Barcelona 2018, por Ckoerner, 2018³.

3.3.1.1 Variação de quantidade de faces

Nessa cena, como a aplicação tem por objetivo detectar um elevado número de faces simultaneamente, pretende-se verificar como a variação da quantidade de faces presente na imagem afeta tanto no tempo de resposta do processo de detecção quanto no tamanho do conjunto das imagens das faces detectadas, recortadas da imagem completa, representando economia na utilização de banda para transmissão do resultado da detecção.

Para a realização desses testes, a escolha de uma imagem com um grande número de faces foi importante para que se pudesse obter 5 variações da mesma, reduzindo iterativamente a quantidade de faces em cada uma, da forma mais linear possível.

As variações foram preparadas utilizando-se o software de edição de imagem GIMP (GNU Image Manipulation Program), aberto e gratuito. Partindo-se da imagem original, uma certa quantidade de faces foram borradadas de forma a se tornarem indetectáveis, como se aquela faces não estivessem presentes na imagem. A imagem resultante se tornou a primeira variação por quantidade de faces. A partir da imagem dessa primeira

³ Disponível em: <https://commons.wikimedia.org/wiki/File:Wikimedia_Hackathon_Barcelona_2018_group_photo.jpg> Arquivo de imagem sob a licença CC BY-SA 4.0: <<https://creativecommons.org/licenses/by-sa/4.0/deed.en>>

variação, foi feito o mesmo procedimento, borrando a mesma quantidade de faces que ainda não haviam sido borradas. E assim foi feito sucessivamente até obter-se um total de 5 variações. Importante ressaltar que, durante a preparação das imagens, preocupou-se em borrar as faces de uma forma bem distribuída.

Figura 13 – Exemplo de variação de cena com redução de 40 faces.



Fonte: compilação do autor.³

Na figura 13, um exemplo de variação com a redução de 40 faces. Ao se comparar o resultado das variações, é de se esperar que a diferença de faces detectadas seja igual à diferente da quantidade de faces borradas, porém, não necessariamente será igual, pois pode ser que alguma face não seja detectada pelo algoritmo de qualquer forma.

Para obtenção dos dados nos testes de variação de quantidade de faces deve-se, a partir da imagem original, com todas as faces detectáveis, utilizar a ferramenta para encontrar uma combinação otimizada de parâmetros de detecção. Com os melhores parâmetros definidos, obtém-se as métricas da imagem original e também de todas as variações de quantidade de faces geradas a partir dessa mesma imagem para futura comparação.

3.3.1.2 Variação de resolução da imagem

Outra variação que é possível obter dessa mesma imagem e que fará parte do estudo é no que tange à resolução da imagem. Quanto maior a resolução da imagem, maior tende a ser a capacidade e qualidade de detecção de faces menores, mas também maior é a demanda de processamento e maior tende a ser o tempo de espera de uma detecção, bem como a utilização de banda para passar adiante as imagens das faces detectadas.

Dependendo do cenário e dos possíveis tipos de aplicação, considerar trabalhar a detecção em imagens com resoluções menores que a capacidade do sensor pode ser suficiente para viabilizar o uso do dispositivo de borda para a tarefa de detecção, desde que cumprindo os requisitos de tempo de resposta e capacidade de detecção.

Para a realização dos testes com variação de resolução, foram obtidas das imagens originais (a original e suas respectivas variações por quantidade de faces) imagens com resoluções menores, através de interpolação, utilizando-se também o software GIMP para o processamento. A imagem original desta cada cena foi selecionada já com resolução em quad-hd (2560x1440), e foram obtidas as variações nas resoluções full-hd (1920x1080) e hd (1280x720). Para cada redução de resolução, são obtidas 5 novas imagens, uma a partir da imagem original e outras quatro a partir da variações de quantidade de faces.

Para a obtenção dos dados nos testes de variação de resolução, utilizou-se novamente a ferramenta para encontrar a melhor combinação de parâmetros de detecção para as imagens na nova resolução e, a partir desses novos parâmetros, obteve-se as métricas de todas as imagens com a referida resolução.

3.3.1.3 Tempo médio de captura

Como já adiantado no final da seção anterior, o tempo de aquisição da imagem que será considerado no estudo será o tempo médio de captura de frames a partir de um sensor OV5647, utilizando-se o pacote *Picamera*. Para a cena em questão, onde objetiva-se maximizar a quantidade de faces detectadas, podendo estas estarem bem distantes da câmera, a qualidade da imagem é um fator importante.

Um dos possíveis *ports* usados pelo Picamera é o *still port*, que força a captura da imagem utilizando-se toda a área do sensor, que no caso é de 2592x1944 pixels, mesmo que a resolução de saída da imagem seja menor. Além disso, usa um forte algoritmo para redução de ruído, proporcionando maior qualidade de imagem, sendo assim a forma mais adequada de considerar na captura de imagem no contexto dessa cena.

Para se obter o tempo médio de captura foi escrito um algoritmo simples, que faz a captura consecutiva de 10 frames utilizando-se o *still port* e retorna o tempo médio de captura. No algoritmo é configurado a resolução de saída dos frames para corresponder às resoluções testadas.

3.3.2 Cena 2

Nessa segunda cena, deseja-se representar o controle de acesso a um ambiente, onde o dispositivo com a câmera está posicionado estrategicamente próximo à abertura de acesso (porta, portão, etc) de forma a capturar de perto a face de quem estiver adentrando ao local. É importante uma resposta rápida na detecção pois a pessoa entrando no ambiente estará em movimento, permanecendo no campo de visão da câmera por pouco tempo.

- **Variações possíveis** - variação de resolução da imagem.
- **Requisitos mínimos** (para balizar a parametrização) - máximo 0.3 segundo de resposta com detecção a partir de 1s de distância.

- **Imagens para testes** - para representar esta cena, foram obtidos imagens a partir do sensor OV5647 conectado ao dispositivo, com o próprio autor simulando a entrada no ambiente.

3.3.2.1 Captura das imagens para teste

Para a captura dos frames para os testes, o dispositivo com a câmera foi posicionado estrategicamente ao lado de uma porta, a uma altura aproximada de 1,70, e levemente inclinado em direção à entrada, de forma que a câmera pudesse captar faces mais próximas e também a distâncias de até 2m.

Foram feitos testes para obter a posição de uma pessoa mais próxima da entrada em que a câmera conseguisse capturar uma boa imagem do rosto. A partir dessa posição, mediu-se 1,7 m de distância, e capturou-se novos frames, com a pessoa posicionada a essa distância. A partir de uma distância de aproximadamente 1,7m, uma pessoa caminhando rapidamente a no máximo 6 km/h, leva pelo menos 1s para chegar à posição inicial obtida ($6 \text{ km/h} * 1000 \text{ m/km} * 1\text{h}/3600\text{s} = 1,67\text{m/s}$). A uma presença com duração de pelos menos 1s diante da câmera, com a face capturável, e a um tempo máximo de 0.3s de resposta definido na detecção, têm-se pelo menos 3 frames capturados e analizados enquanto uma mesma pessoa caminha até a entrada do ambiente. Esse será considerado nosso pior caso e ambos os frames obtidos (tanto o com a pessoa na posição mais próxima da câmera quanto o com a pessoa posicionada a 1.7m da posição inicial) serão utilizados nos testes para definição dos parâmetros e obtenção das métricas para comparação.

Dessa forma, a ferramenta de análise será utilizada nessa cena com uma abordagem diferente para se obter os melhores parâmetros para comparação. Ao invés de se buscar a maior quantidade de faces detectadas no menor tempo, como feito na primeira cena, dessa vez, como o objetivo é a detecção de apenas uma face próxima por vez, busca-se encontrar os parâmetros com os quais é possível detectar a face de uma pessoa se aproximando da entrada a partir de 1,7m de distância, no menor tempo. Como têm-se duas imagens diferentes sendo analisadas, com a face da mesma pessoa, mas em posição e tamanhos relativos diferentes, cada uma apresentará resultados dife-

rentes entre si. Portanto, serão considerados a combinação de parâmetros que atenda bem às duas imagens. Quanto aos dados obtidos das métricas para comparação de resultados, serão considerados os tempos de maior duração e os maiores tamanhos de face.

3.3.2.2 Variação de resolução da imagem

Como nessa cena considera-se a detecção de apenas uma face, variações por quantidade de faces não faz sentido. Portanto, será experimentado apenas variação por resolução de imagem.

Para a realização dos testes com variação de resolução, foram obtidas das duas imagens capturadas imagens com resoluções menores, através de interpolação, utilizando-se também o software GIMP para o processamento. As imagens originais originais foram capturadas já com a maior resolução que será considerada no estudo, SVGA (800x600), e foram obtidas as variações nas resoluções menores, VGA/480p (640x480) e QVGA (320x240). As imagens nas resoluções menores poderiam também terem sido capturadas diretamente do módulo da câmera, porém dificilmente se conseguiria obtê-las com as faces nas mesmas posições.

subsubsectionModo de captura e tempo médio

Na cena anterior foi utilizado o modo *still port* do Picamera para se obter o tempo médio de captura de cada frame, pois, como já explicado, obtém-se imagens com melhor qualidade devido à captura utilizando-se a área total do sensor e devido aos algoritmos de redução de ruído. Porém, obviamente, isso traz um custo no tempo de disponibilização do frame pelo módulo da câmera.

Nessa segunda cena, como o objetivo é a detecção de uma única face mais próxima da câmera e permanecendo por muito pouco tempo no campo de visão, torna-se então menos importante a qualidade da imagem e mais importante a velocidade com que os frames são disponibilizados.

Outro *port* de captura disponível é o *video port*, que não força a captura utilizando-se

toda a área do sensor e nem utiliza um forte algoritmo de redução de ruído, como feito no *still port*. Assim, consegue-se capturas de frames mais rápidas, porém com menor qualidade, tendo as imagens uma aparência mais granulada.

Para obtenção das imagens, bem como os tempos médios de captura, foi escrito um script simples que faz a captura 30 frames em sequência, utilizando-se o *video port*, salva os mesmos em arquivos de formato JPEG e calcula os tempos médios de captura que serão utilizados na base de dados para comparação. Esse tempo médio é obtido também para as resoluções menores definidas. O mesmo script foi utilizado para obter as duas imagens na resolução 800x600 a serem utilizadas na ferramenta de parametrização.

3.4 Dispositivos testados

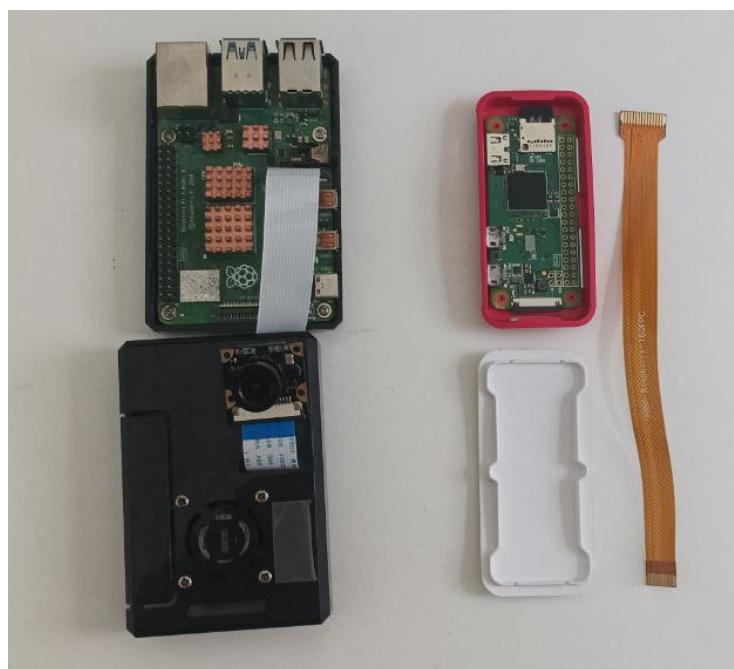
Foram testados dois dispositivos SBC (*Single Board Computer*) diferentes, ambos da família Raspberry Pi, cujos resultados serão comparados.

- SBC 1 - Raspberry Pi 4B: quad-core Cortex-A72 com clock de 1,5 GHz e 4 GB de memória RAM.
- SBC 2 - Raspberry Pi Zero W: single-core ARM1176JZF-S com clock de 1,0 GHz e 512 MB de memória RAM.

Na figura 14, os dois dispositivos utilizados. O Raspberry Pi 4B já com o módulo da câmera conectado na interface CSI. Esse mesmo módulo é conectado no Raspberry Pi Zero utilizando-se cabo flat adequado, para obter-se o tempo médio de captura de frames.

Apesar de configurações de hardware bem diferentes, os dois dispositivos foram configurados para rodar a mesma versão do sistema operacional Raspberry Pi OS (com kernel 5.10), bem como as mesmas versões do Python (3.7), Picamera (1.13) e OpenCV (3.2). Assim, elimina-se a possibilidade de diferentes versões de software influenciar no desempenho.

Figura 14 – Dispositivos testados. À esqueda o Raspberry Pi 4 e à direita o Raspberry Pi Zero W.



Fonte: autor.

4 EXPERIMENTOS REALIZADOS

A realização dos experimentos baseou-se na utilização da ferramenta de parametrização e obtenção de métricas, aplicado nas imagens tratadas de cada cena, conforme variações definidas no Capítulo anterior. Os dados obtidos foram organizados de forma a facilitar a análise e comparação entre as variações em cada cena e também entre os diferentes dispositivos sendo testados.

Para cada variação de resolução, que depende de uma nova seleção de parâmetros, será exibido a matriz de resultados final juntamente com as faces detectadas, após já feita a análise para se chegar a um melhor resultado, indicando quais foram os parâmetros definidos.

4.1 Cena 1

4.1.1 Otimização de parâmetros

Primeiramente, foi realizada a otimização dos parâmetros para cada variação de resolução, a partir das imagens com todas as faces disponíveis de cada resolução. No caso desta cena, que exige relativamente muito mais processamento, utilizou-se o Raspberry Pi 4B para a definição dos parâmetros, repetindo-os nos testes com o Raspberry Pi Zero W para base de comparação.

A definição dos parâmetros 'ótimos' não é objetiva. Para esta cena, buscou-se um melhor resultado em que houvesse a maior quantidade de faces detectadas com o mínimo ou nenhuma presença de falsos positivos e no menor tempo. Durante a análise, teve-se a razoabilidade de considerar na comparação entre os diferentes resultados da matriz que, um grande aumento no tempo de detecção não justifica um pequeno ganho relativo na quantidade de faces detectadas.

As próximas três subsubseções 4.1.1.1, 4.1.1.2 e 4.1.1.3 mostram através de

imagens, para cada resolução testada, a última iteração da matriz de resultados com os dados entrada, a célula destacada com os parâmetros escolhidos e a imagem com a faces detectadas, nos testes feitos com o Raspberry Pi 4B.

Os parâmetros definidos de cada resolução foram usados para obter as métricas de cada variação de quantidade de faces detectáveis, e em ambos os dispositivos testados. Não será apresentado nesse documento cada resultado das métricas obtidos em tela. Todos os dados obtidos foram tabelados e utilizados para as comparações feitas nas subseções seguintes. Uma tabela resumida com esses dados é exibida no final de cada uma das próximas três subseções.

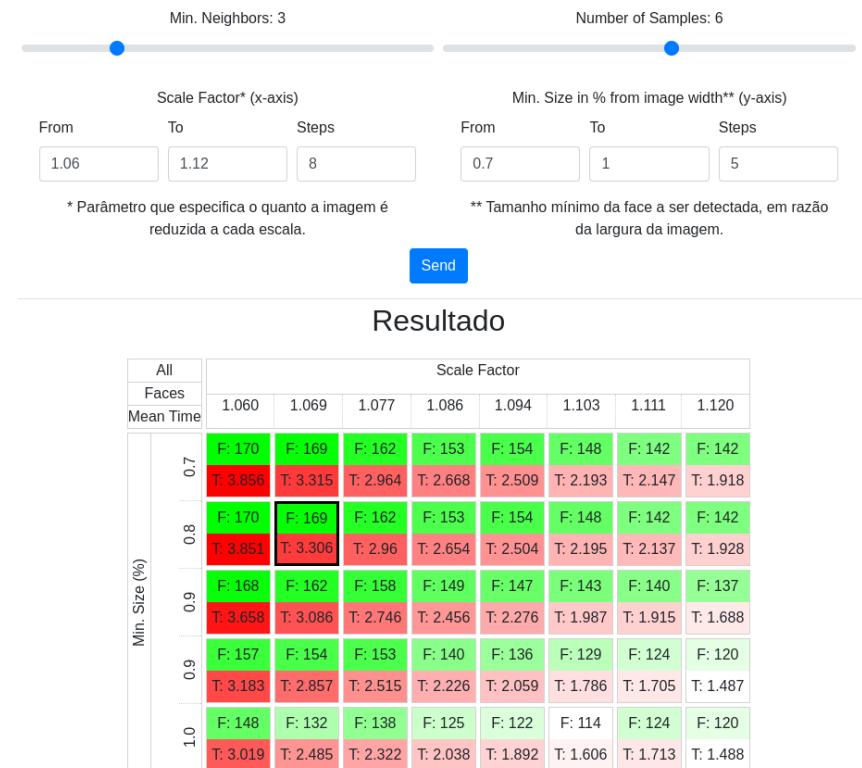
4.1.1.1 Resolução 1440p

Para essa resolução de 1440 x 2560 conseguiu-se chegar a uma quantidade muito boa de faces detectadas, mesmo as faces menores, de pessoas mais distantes, quase totalizando a quantidade de faces presentes na imagem. Como é normal, algumas faces não foram detectadas devido a algum adereço como chapéu, posição da face e outros possíveis fatores. Nesse caso, nenhum falso positivo foi detectado.

Parâmetros definidos:

- Min. Size Face: 0.8%
- Scale Factor: 1.069
- Min. Neighbors: 3

Figura 15 – Otimização Cena1 resolução 1440p.



Fonte: compilação do autor.¹

¹ Imagem usada na compilação disponível em: <https://commons.wikimedia.org/wiki/File:Wikimedia_Hackathon_Barc_group_photo.jpg> Arquivo de imagem sob a licença CC BY-SA 4.0: <<https://creativecommons.org/licenses/by-sa/4.0/deed.en>>

Tabela 1 – Dados obtidos - resolução 1440p.

Faces Variation	Device	Resolution	Full Image Encoded (bytes)	Detected Faces	Encoded Faces Size	Capturing Image (s)	Convert Image to Gray (s)	Detection (s)	Build Encoded Faces Images (s)	Total execution time with capture (s)
Full	Raspberry Pi 4	1440 x 2560	14,745,672	169	742,472	0.568	0.007	3.293	0.011	3.879
Minus 40	Raspberry Pi 4	1440 x 2560	14,745,672	130	562,352	0.568	0.007	3.286	0.009	3.870
Minus 80	Raspberry Pi 4	1440 x 2560	14,745,672	95	399,528	0.568	0.007	3.272	0.006	3.853
Minus 120	Raspberry Pi 4	1440 x 2560	14,745,672	58	241,748	0.568	0.007	3.218	0.003	3.796
Minus 160	Raspberry Pi 4	1440 x 2560	14,745,672	25	115,868	0.568	0.007	3.178	0.002	3.755
Full	Raspberry Pi Zero W	1440 x 2560	14,745,672	169	742,472	0.575	0.13	93.5	0.092	94.297
Minus 40	Raspberry Pi Zero W	1440 x 2560	14,745,672	130	562,352	0.575	0.208	92.359	0.081	93.223
Minus 80	Raspberry Pi Zero W	1440 x 2560	14,745,672	95	399,528	0.575	0.125	91.214	0.049	91.963
Minus 120	Raspberry Pi Zero W	1440 x 2560	14,745,672	58	241,748	0.575	0.127	90.538	0.041	91.281
Minus 160	Raspberry Pi Zero W	1440 x 2560	14,745,672	25	115,868	0.575	0.156	89.549	0.014	90.294

Fonte: autor.

4.1.1.2 Resolução 1080p

Nessa resolução de 1080 x 1920, algumas faces menores começam a não serem mais detectadas, mas a quantidade de faces detectadas é relativamente grande se comparado à quantidade de faces presentes na imagem.

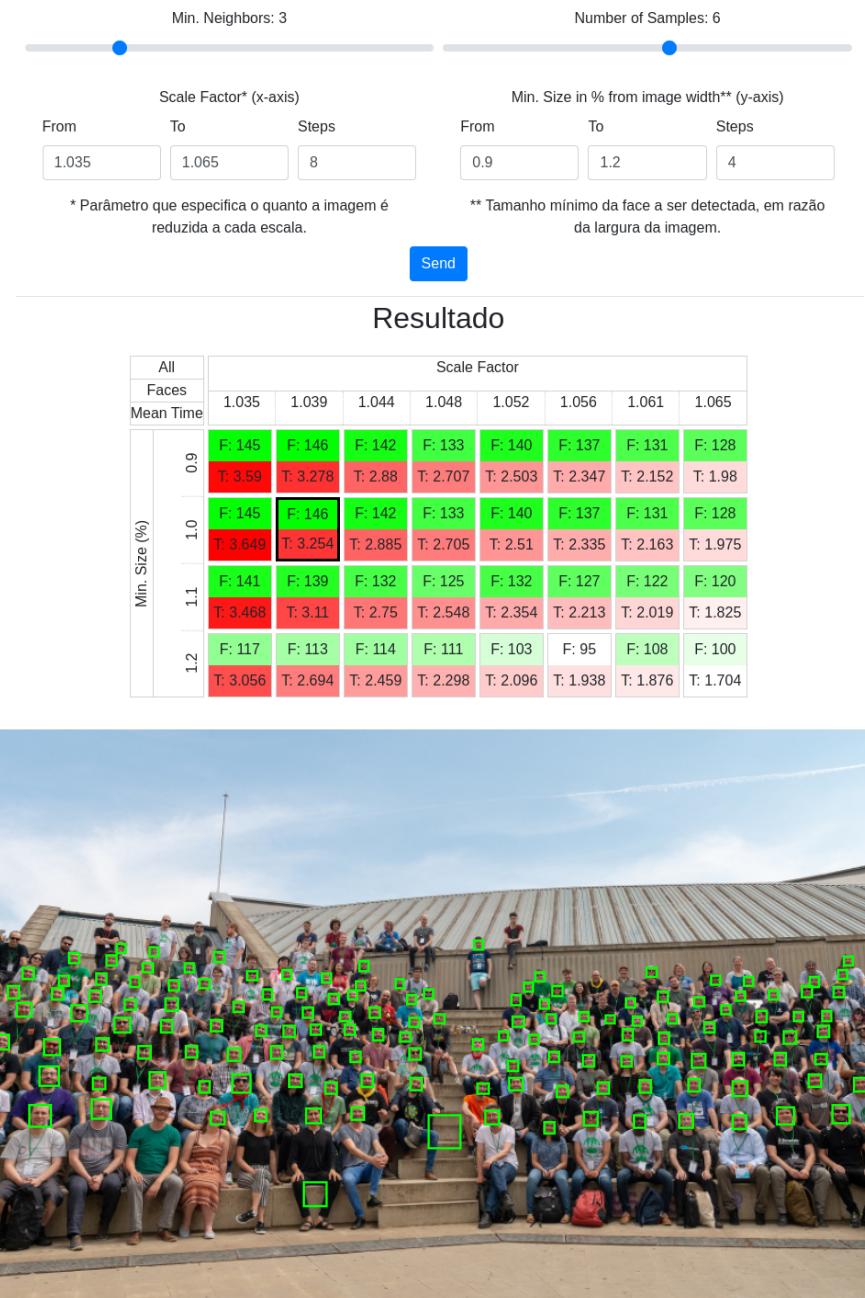
Observa-se uma diminuição no valor do parâmetro scale factor (que significa mais passos no escalonamento da imagem) para se obter quantidades maiores de faces detectadas e um aumento no do parâmetro min. size, já que as faces menores começam a ficar indetectáveis.

Dessa vez percebe-se a presença de alguns falsos positivos. Testou-se a variação do parâmetro min. neighbors, porém não houve melhoria.

Parâmetros definidos:

- Min. Size Face: 1.0%
- Scale Factor: 1.039
- Min. Neighbors: 3

Figura 16 – Otimização Cena1 resolução 1080p.



Fonte: compilação do autor.¹

Tabela 2 – Dados obtidos - resolução 1080p.

Faces Variation	Device	Resolution	Full Image Encoded (bytes)	Detected Faces	Encoded Faces Size	Capturing Image (s)	Convert Image to Gray (s)	Detection (s)	Build Encoded Faces Images (s)	Total execution time with capture (s)
Full	Raspberry Pi 4	1080 x 1920	8,294,472	146	424,700	0.520	0.004	3.27	0.007	3.801
Minus 40	Raspberry Pi 4	1080 x 1920	8,294,472	113	332,764	0.520	0.004	3.202	0.006	3.732
Minus 80	Raspberry Pi 4	1080 x 1920	8,294,472	82	259,464	0.520	0.004	3.212	0.004	3.740
Minus 120	Raspberry Pi 4	1080 x 1920	8,294,472	52	175,964	0.520	0.004	3.183	0.003	3.710
Minus 160	Raspberry Pi 4	1080 x 1920	8,294,472	27	114,540	0.520	0.004	3.171	0.002	3.697
Full	Raspberry Pi Zero W	1080 x 1920	8,294,472	146	424,700	0.518	0.071	87.892	0.092	88.573
Minus 40	Raspberry Pi Zero W	1080 x 1920	8,294,472	113	332,764	0.518	0.071	86.648	0.054	87.291
Minus 80	Raspberry Pi Zero W	1080 x 1920	8,294,472	82	259,464	0.518	0.07	85.841	0.039	86.468
Minus 120	Raspberry Pi Zero W	1080 x 1920	8,294,472	52	175,964	0.518	0.095	85.053	0.026	85.692
Minus 160	Raspberry Pi Zero W	1080 x 1920	8,294,472	27	114,540	0.518	0.071	84.656	0.014	85.259

Fonte: autor.

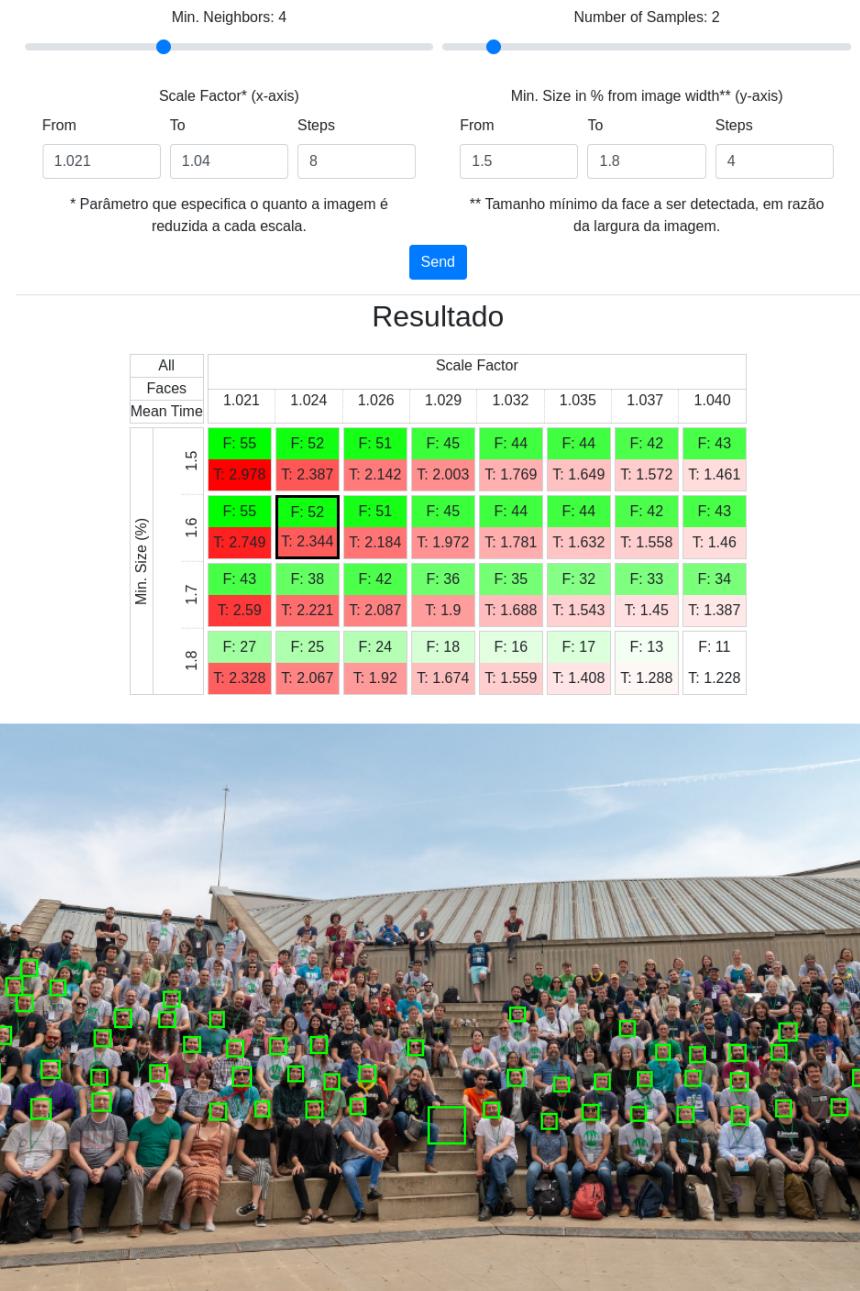
4.1.1.3 Resolução 720p

Nessa resolução de 720 x 1280, a queda na quantidade de faces detectadas já se torna bem considerável se comparando às resoluções maiores, sendo menos de um terço da quantidade de faces presente na imagem. Novamente, observa-se a diminuição do parâmetro de scale factor e aumento do parâmetro min. size, devido às faces menores não detectáveis. Também percebe-se a presença de falso positivo. Testou-se também a variação do parâmetro min. neighbors. Aumentando-se de 3 para 4, observou-se a redução de falsos positivos.

Parâmetros definidos:

- Min. Size Face: 1.6%
- Scale Factor: 1.024
- Min. Neighbors: 4

Figura 17 – Otimização Cena 1 - resolução 720p.



Fonte: compilação do autor.¹

Figura 18 – Tabela de Dados - resolução 720p.

Faces Variation	Device	Resolution	Full Image Encoded (bytes)	Detected Faces	Encoded Faces Size	Capturing Image (s)	Convert Image to Gray (s)	Detection (s)	Build Encoded Faces Images (s)	Total execution time with capture (s)
Full	Raspberry Pi 4	720 x 1280	3,686,472	52	117,928	0.498	0.002	2.374	0.003	2.877
Minus 40	Raspberry Pi 4	720 x 1280	3,686,472	37	86,424	0.498	0.002	2.362	0.003	2.865
Minus 80	Raspberry Pi 4	720 x 1280	3,686,472	26	61,884	0.498	0.002	2.358	0.001	2.859
Minus 120	Raspberry Pi 4	720 x 1280	3,686,472	17	43,460	0.498	0.002	2.362	0.001	2.863
Minus 160	Raspberry Pi 4	720 x 1280	3,686,472	9	26,156	0.498	0.002	2.348	0.001	2.849
Full	Raspberry Pi Zero W	720 x 1280	3,686,472	52	117,928	0.498	0.031	61.483	0.028	62.040
Minus 40	Raspberry Pi Zero W	720 x 1280	3,686,472	37	86,424	0.498	0.042	61.992	0.017	62.549
Minus 80	Raspberry Pi Zero W	720 x 1280	3,686,472	26	61,884	0.498	0.035	61.807	0.018	62.358
Minus 120	Raspberry Pi Zero W	720 x 1280	3,686,472	17	43,460	0.498	0.033	61.821	0.011	62.363
Minus 160	Raspberry Pi Zero W	720 x 1280	3,686,472	9	26,156	0.498	0.045	61.749	0.004	62.296

Fonte: autor.

4.1.2 Análise dos resultados

Nesta cena, são três fatores influenciando no resultado das detecções, seja quanto à capacidade de detecção e/ou quanto ao tempo de resposta. Esses fatores são a quantidade de faces detectáveis presentes na imagem, a resolução da imagem e o dispositivo rodando o algoritmo de detecção, sobre os quais serão feitas as comparações e análises a seguir.

4.1.2.1 Efeitos da variação por quantidade de faces

Os gráficos apresentados na figura 19 foram montados de forma a facilitar o entendimento de como a quantidade de faces influencia no tempo de detecção. Para maior clareza, os dados foram apresentados em três gráficos diferentes, um para cada resolução de imagem. No eixo x, têm-se as variações de quantidade de faces detectáveis disponíveis. "Full" refere-se à imagem original, sem alteração, com todas as faces detectáveis, "Minus 40" refere-se à imagem trabalhada para borrar 40 faces de forma distribuída, tornando-as indetectáveis, e assim por diante.

Há três séries de dados cujas métricas são apresentadas para cada variação de faces. A quantidade de faces detectadas pelo algoritmo representada pelas barras verdes, que seguem a escala do eixo y à esquerda; o tempo total de execução nos testes com o Raspberry Pi 4B, em azul escuro, e o tempo total de execução nos teste com o Raspberry Pi Zero, em azul claro, ambos seguindo a escala do eixo y à direita, com unidade em segundos. Para melhor visualização, o eixo y à direita está em escala

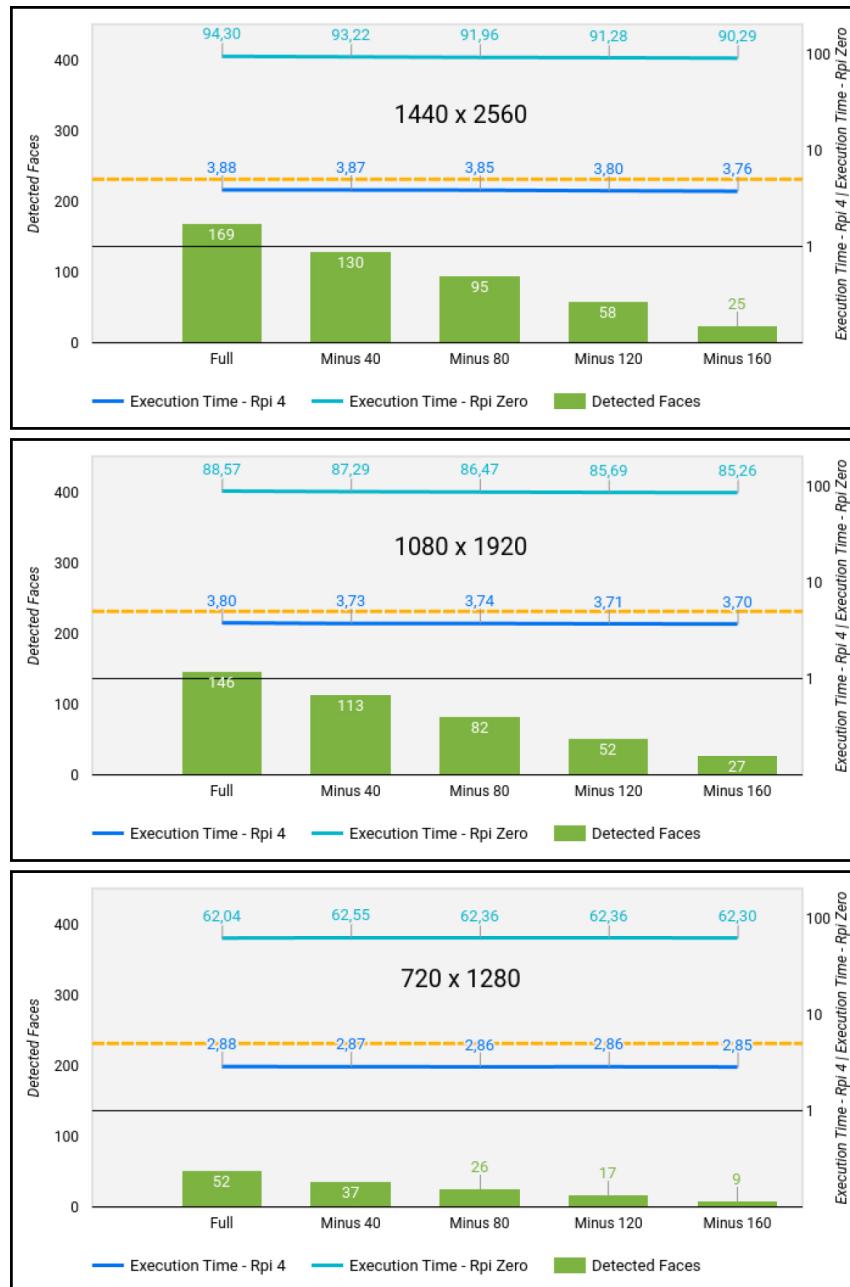
logarítmica. O tempo de execução total é a soma do tempo de aquisição de imagem (nesse caso já considerando o tempo de captura do módulo câmera e não de leitura da imagem em arquivo), o tempo de conversão da imagem em escala de cinza, o tempo de detecção em si e o tempo de encodamento da imagem para transporte. Há também a representação no gráfico, em linha amarela tracejada, do tempo de 5 segundos definido como limite para atender ao requisito da cena.

A ferramenta utilizada para a geração dos gráficos a partir da tabela de dados foi o *Data Studio* (DataStudio, 2022), da Google.

Como se pode notar pela análise dos gráficos, a variação da quantidade de faces presentes, e por consequência detectadas, influencia muito pouco no tempo de detecção, chegando a 4% a maior diferença. Observa-se esse comportamento em ambos os dispositivos testados e em todas as resoluções de imagem.

Com base nisso, pode-se dizer que, para a cena em questão, a quantidade de pessoas presentes no ambiente pouco no tempo de resposta do algoritmo em uso, o *Haar cascade object detection* (VIOLA; JONES, 2001), podendo então este fator ser desconsiderado para quesitos de tempo (mas não de utilização de banda, obviamente).

Figura 19 – Faces detectadas e Tempos de execução por Variação de faces detectáveis.



Fonte: autor.

4.1.2.2 Efeitos da variação por resolução de imagem

Como foi constatado que a variação da quantidade de faces tem efeito insignificante no tempo de resposta, considerou-se apenas os resultados dos testes nas imagens com todas as faces detectáveis para montar o gráfico da figura 20, que têm a mesma estrutura dos gráficos anteriores, com a exceção de que no eixo x têm agora a variação

por resolução de imagem.

Ao se comparar as duas maiores resoluções, 1440p e 1080p, observa-se que tanto o tempo de execução quanto a quantidade de faces detectadas são menores na resolução menor, como é de se esperar. A redução no tempo é muito pouco expressiva, enquanto a redução na quantidade de faces detectadas não é tão expressiva mas pode-se considerar razoável. Comparando-se as faces detectadas nessa duas resoluções, vide figuras 15 e 16, percebe-se que as faces não detectadas são de pessoas que estão mais distantes na cena. Essa diferença de 25 faces (desconsiderando-se 2 falsos positivos na resolução de 1080p) pode ou não fazer grande diferença para a tomada de decisão, vai depender da realidade da cena real, se seria comum pessoas posicionadas tão distantes sem se aproximar da câmera, etc.

É de se pensar que tendo menor resolução, a redução no tempo de resposta deveria ser razoavelmente menor devido à quantidade menor de pixels e quadros sendo processados. Porém, deve-se considerar também que, ao se buscar a maximização da quantidade de faces detectadas, reduziu-se o valor do parâmetro *scale factor* durante a otimização, o que aumenta a quantidade de iterações no escalonamento. Comparando-se as matrizes de resultado nas figuras 15 e 16, à medida que se caminha para a direita na matriz refente à resolução de 1080p, com valores maiores de *scale factor* se aproximando ao selecionado para 1440p, observa-se uma redução considerável no tempo de detecção, com esperado, porém com também considerável redução na quantidade de faces detectadas.

Já comparando-se os resultados obtidos com a imagem de menor resolução, 720p, e as demais, a perda com relação à quantidade de faces detectadas é bastante expressiva, chegando a cair para menos de um terço da quantidade de faces detectadas na imagem de 1440p. Como pode ser visto na imagem 17, apenas as faces maiores (de pessoas mais próximas da câmera) são detectadas. A tempo de resposta tem uma queda até que razoável, porém não compensa a perda na quantidade de faces detectadas. Dessa forma, o uso de imagens com resolução de 720p se torna inadequado para o cenário em questão.

Pode-se afirmar então que, a princípio, considerando-se apenas o *trade-off* entre

Figura 20 – Faces detectadas e Tempos de execução por Variação de resolução.



Fonte: autor.

quantidade de faces detectadas e tempo de resposta, a resolução de 1440p, a maior entre as testadas, se mostra a mais adequada a se utilizar nesse tipo de cenário, onde objetiva-se detectar o máximo de faces possível e de tamanhos relativos muito pequenos.

Outra questão em que a utilização de resolução maior torna-se vantajoso é quanto à qualidade das imagens recortadas das faces. Caso as imagens das faces sejam utilizadas em uma próxima etapa da aplicação em outro dispositivo na rede ou na nuvem, como por exemplo para reconhecimento facial ou simplesmente manter em registro, a qualidade e detalhes da face recortada pode fazer diferença no sucesso dessa próxima etapa.

Para se ter uma ideia mais visual da diferença, a figura 21 faz um comparativo entre duas faces detectadas (a de uma pessoa mais próxima e a de uma pessoa mais distante) e as diferentes resoluções.

Um fator que talvez possa pesar a favor de se utilizar resolução menor é a quantidade de dados sendo trafegados ou armazenados. A figura 22 traz um comparativo entre o tamanho médio das imagens das faces recortadas, coloridas e encodadas em base64. Nesse caso, observa-se que a utilização da resolução em 1080p reduziria o tamanho de dados trafegados e/ou armazenados em aproximadamente 34%. Claro que existe a possibilidade de realizar a detecção na resolução maior e redimensionar as imagens antes de serem transmitidas, mas deve-se levar em consideração que, para tal, haverá

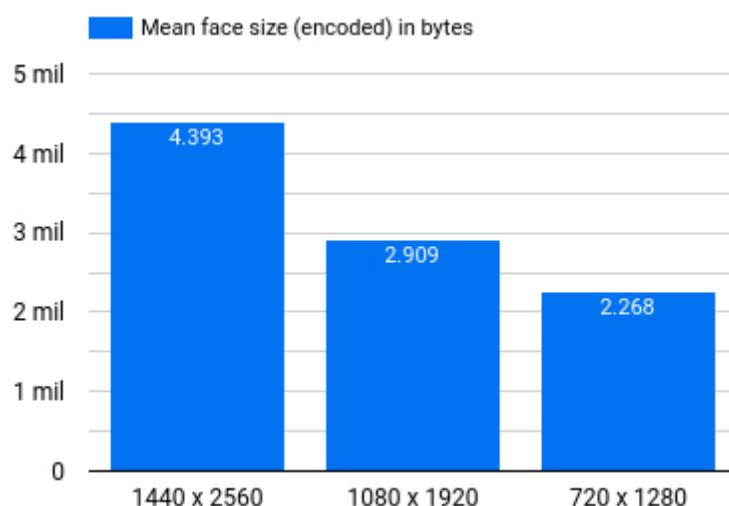
Figura 21 – Comparativo de faces recortadas de diferentes resoluções, em tamanho real.

1440 x 2560	1080 x 1920	720 x 1280
		Não detectado
		

Fonte: autor.

mais carga de processamento no dispositivo de borda.

Figura 22 – Comparativo de tamanho médio de imagem encodada por face detectada em bytes.



Fonte: autor.

4.1.2.3 Considerações sobre os dispositivos testados

Nos gráficos apresentados nas figuras 19 e 20 já fica bem evidente que o Raspberry Pi Zero W deixa a desejar, com tempos de resposta acima de 1 minuto, muito além do aceitável para a cena. Pode-se concluir então que a utilização deste não é adequada para processamento de detecção de faces em imagens com resoluções mais altas, como as testadas, independente da quantidade de faces presentes na imagem.

Já o Raspberry Pi 4B apresentou bom desempenho, conseguindo detectar 169 faces na maior resolução com tempo médio de resposta de 3,88, desde a captura do frame até a preparação das faces cortadas em dados prontos para transmissão. Um tempo razoavelmente menor que o máximo desejado no cenário, com uma margem 22% e que ainda pode ser aumentada tirando proveito de processamento assíncrono, principalmente na captura dos frames, o que não foi feito nesses testes.

4.1.2.4 Considerações sobre utilização de banda

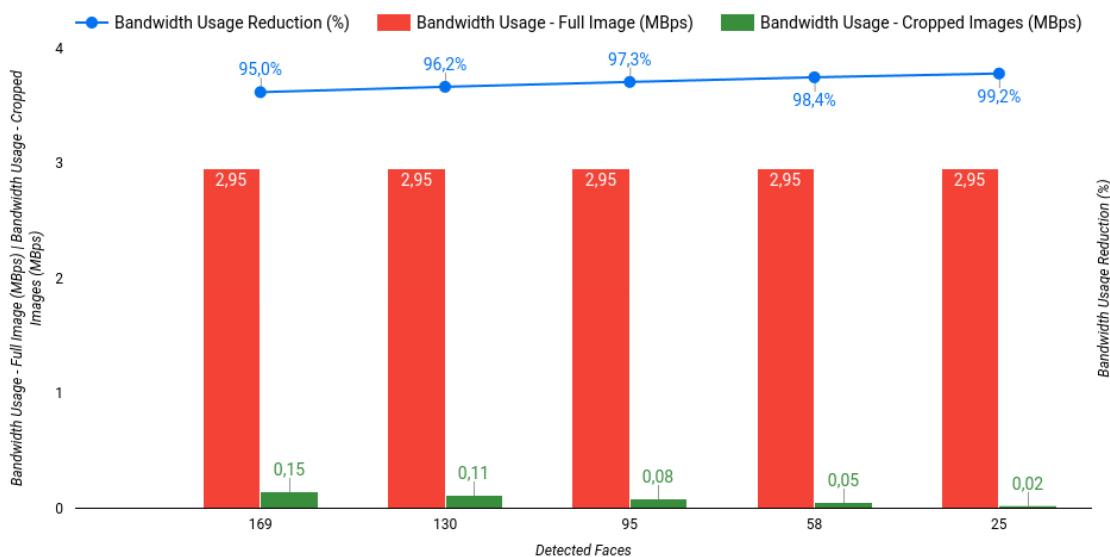
Quando em uma aplicação com arquitetura distribuída, uma das principais vantagens de se alojar parte do processamento nos dispositivos de borda é a economia que se pode ter na utilização de banda ao trafegar os dados gerados por esses dispositivos para outros positivos que estejam na mesma rede ou na nuvem, que desempenham outras funções dentro da aplicação. Quando se tem têm imagens sendo trafegadas, como é o caso desse estudo, o volume de dados é razoavelmente alto, podendo essa economia ser muito relevante para o projeto.

Trazendo para a cena em questão, caso o dispositivo de borda tivesse a função de apenas capturar a imagem, encodar e transmiti-la para um outro dispositivo realizar a detecção das faces, as imagens teriam que ser transmitidas em sua totalidade para se obter o mesmo resultado. Uma grande quantidade de dados trafegados, refentes às partes das imagens não úteis após a detecção, seriam descartados, desperdiçando-se assim a utilização de banda da rede. Inclusive, mesmo não havendo faces presentes, as imagens precisariam continuar sendo enviadas periodicamente, evidenciando ainda mais o desperdício. Isso pode gerar vários problemas como gargalos na rede e sobrecarga computacional.

Tendo-se a detecção das faces realizada pelo próprio dispositivo na borda, apenas as partes úteis da imagens, que são as faces recortadas, precisam ser transmitidas, proporcionando assim uma utilização mais eficiente na utilização da rede que interliga os diferentes dispositivos.

Durante os testes, junto com as métricas de tempo de processamento são obtidas

Figura 23 – Comparativo de utilização de banda por quantidade de faces detectadas.



Fonte: autor.

também as informações do tamanho total da imagem testada e também a soma das imagens das faces recortadas, que serão usadas para comparações na sequência. Estão sendo considerados os tamanhos das imagens após encodamento em base64.

A figura 23 mostra um comparativo de como seria a utilização de banda nos dois casos, transmitindo-se a imagem completa e transmitindo-se apenas as imagens das faces recortadas, com resolução de 1440 x 2560. Para o cálculo de utilização de banda, considerou-se intervalo entre as detecções de 5 segundos, conforme tempo máximo definido como requisito para esta cena.

No eixo x têm-se a quantidade de faces que foram detectadas, nas barras de cor vermelha a largura de banda que seria utilizada trafegando-se a imagem completa e nas barras de cor verde a largura de banda considerando-se o tráfego apenas do recorte das faces detectadas, ambas seguindo a escala à esquerda do eixo y e expressas em MBps. Na série em formato de linha azul têm-se a diferença percentual em relação ao tamanho da imagem completa, seguindo a escala à direita do eixo y.

Como se pode observar, a redução na utilização de banda para tráfego das imagens ao se utilizar o processamento na borda para detecção de face é bastante expressivo, partindo-se de 95% para o caso em que há grande quantidade de faces detectadas (196) e chegando a 99,2% no caso com muito poucas faces detectadas (25). De fato,

a redução pode chegar até a 100% em eventuais momentos em que não haja face alguma a ser detectada.

Pode-se pensar que talvez a utilização de quase 3 MBps para se transmitir a imagem completa não seria algo tão absurso assim, dependendo da infraestrutura de rede disponível e seu estado de utilização. Mas é importante atentar-se aqui que a unidade sendo usada está em Megabytes/s, então 3 MBps corresponderia a 24 Mbps (Megabits/s), nada mais sendo do que 24% da capacidade de banda de uma rede operando a 100 Mbps.

Além disso, está se considerando aqui a utilização de banda de apenas um dispositivo. Em uma aplicação real, é razoável considerar que podem haver dezenas de dispositivos realizando monitoramento e tendo de enviar imagens e dados a algum outro dispositivo centralizador, seja em uma rede local ou algum servidor na nuvem. Imagine um caso com apenas 5 dispositivos de borda distribuídos em ambientes parecidos com a cena em questão, capturando imagens em 1440 x 2560 e a cada 5s para monitoramento e detecção de faces. Caso não haja detecção de face na borda, a utilização de banda necessária para transmissão das imagens completas seria de pelo menos 120 Mbps, o que já seria inviável em uma rede local de 100 Mbps, quanto mais na nuvem. Agora, considere que todos esses 5 dispositivos tenham capacidade de realizar a detecção transmitir apenas as imagens das faces recortadas. Para o pior caso que se testou, 169 faces detectadas, a banda demandada pelos cinco dispositivos simultaneamente seria de aproximadamente 0,75 MBps, ou 6 Mbps, algo bastante razoável para se trafegar em uma rede local mas talvez ainda nem tanto para se transmitir continuamente pela nuvem.

4.2 Cena 2

4.2.1 Captura de imagens para teste

Conforme definido no capítulo anterior, as imagens para teste foram capturadas a partir de um módulo de câmera conectado no Raspberry Pi 4B.

A cena foi montada conforme pode ser visualizado na figura 24. O Raspberry Pi 4B, montado em sua case e com o módulo da câmera conectado foi fixado no batente lateral esquerdo da porta da entrada, de forma que a câmera ficasse posicionada a 1,7m do piso e levemente inclinada para baixo e para a dentro, na direção da passagem. A pessoa que aparece na imagem é o próprio autor.

Utilizando-se o script de captura de frames, foram feitos alguns testes e definiu-se essa como a posição mais próxima para se obter a primeira imagem a ser utilizada nos testes de desempenho, pois consegue pegar uma boa imagem do rosto e ainda com um pouco de folga vertical, considerando-se pessoas mais altas ou baixas.

Figura 24 – Posicionamento para captura da face na primeira posição.



Fonte: autor.

Na figura 25 têm-se a imagem capturada pela câmera na posição 1, na maior resolução definida para esta cena, 800x600.

Figura 25 – Captura da imagem na primeira posição.



Fonte: autor.

A partir dessa primeira posição, mediu-se 1,7m distanciando-se da entrada. Essa é a posição mais distante, a partir da qual deseja-se que a face seja detectável. A figura 26 demonstra esse posicionamento e na figura 27 tem-se a imagem capturada pela câmera que também será utilizada nos testes.

Figura 26 – Posicionamento para captura da face na segunda posição.



Fonte: autor.

Figura 27 – Captura da imagem na segunda posição.



Fonte: autor.

4.2.2 Otimização de parâmetros

Foi realizada a otimização dos parâmetros para cada variação de resolução, utilizando-se as imagens capturadas nas duas posições. Utilizou-se o Raspberry Pi 4B para a definição dos parâmetros, repetindo-os nos testes com o Raspberry Pi Zero W para base de comparação.

As imagens em ambas as posições foram analisadas em conjunto mas com o objetivo de se chegar a apenas um conjunto de parâmetros 'ótimos' para obtenção das métricas. Foram considerados 'ótimos' a mesma combinação de parâmetros que atendia às duas imagens, buscando-se os limiares com o menor tempo em que as faces são detectadas e sem falsos positivos. Para facilitar a visualização desses limiares, foram definidos passos maiores de 'Scale Factor' e 'Min. Size (%)' para se obter quantidades maiores de amostras.

Pode-se pensar, a princípio, que bastava analisar a imagem na posição 2, que é a mais distante, pois essa já seria o pior caso, já que a face teria menor resolução que a da posição 1. Porém, percebe-se claramente comparando as figuras 25 e 27 que quando a face está mais próxima da câmera e estando a pessoa olhando naturalmente

em direção à entrada, o angulo não é muito bom, podendo criar alguma dificuldade adicional na detecção da face. Por isso a importância da análise conjunta.

Para se considerar um pior caso, foram utilizados os dados das métricas que apresentaram maiores tempos de resposta e tamanho de imagem, isso para cada resolução.

As próximas três subsubseções 4.2.2.1, 4.2.2.2 e 4.2.2.3 mostram para cada resolução testada, a última iteração da matriz de resultados para a detecção em ambas as posições, com os dados entrada, a célula destacada com os parâmetros escolhidos e as imagens com a faces detectadas. Nas figuras das matrizes foram adicionadas linhas azuis (para a posição 1) e vermelhas (para posição 2), indicando limiares que separam os grupos contínuos de resultados onde há detecção de face. Foram então selecionados os melhores resultados dentro da intersecção desses grupos .

Após os parâmetros definidos no Raspberry Pi 4, os mesmos foram utilizados para obter a métricas no Raspberry Pi Zero W, sendo todos tabelados para comparação. Uma tabela resumida com esses dados é exibida no final de cada uma das próximas três subseções.

Devido ao tamanho maior das matrizes de resultados e a necessidade de visualização lado a lado, a parte da interface com os controles de parametrização não foram exibidos. Os prâmetros utilizados foram: "Min. Neighbors: 3", "Number of Samples: 6". Início e fim de escala dos demais parâmetros podem ser vistos nas matrizes.

4.2.2.1 Resolução 600p

A figura 28 apresenta lado a lado as matrizes de resultados dos testes realizados com as imagens em 600 x 800 das duas posições no Raspberry Pi 4B. Para os ranges e passos escolhidos, percebe-se que para a posição 1, com a face maior, o principal fator que delimita a região onde a face é detectada é o 'Scale Factor', com valores iguais ou menores a 1,4. Enquanto que para a posição 2, o principal delimitador para detecção da face é o 'Min. Size (%)', o que se é esperado, já que é a imagem que possui a face em tamanho menor.

As duas linhas delimitadoras na cor vermelha, na matriz à direita, foram posicionadas de forma um tanto livre, mas buscando isolar regiões onde todos os resultados detectam a face. Poderia também ter sido considerada apenas uma linha horizontal entre 'Min. Size (%)' 7.0 e 8.0, por exemplo. Não mudaria muito o resultado e não teria impacto significativo na estudo.

Considerando-se o menor tempo na interseção das regiões, os parâmetros definidos foram:

- Min. Size Face: 8.0%
- Scale Factor: 1.400
- Min. Neighbors: 3

Os mesmos parâmetros foram utilizados para obter as métricas detalhadas nos dois dispositivos testados. Os dados obtidos são apresentados na tabela da figura 30 que serão usados nas comparações de resultados.

Figura 28 – Otimização Cena 2 - resolução 600p - matrizes. À esquerda, posição 1, e à direita, posição 2.

All Faces Mean Time		Scale Factor										All Faces Mean Time		Scale Factor									
		1.100	1.150	1.200	1.250	1.300	1.350	1.400	1.450	1.500	1.550			1.100	1.150	1.200	1.250	1.300	1.350	1.400	1.450	1.500	1.550
Min. Size (%)	3.0	F:1	F:1	F:1	F:1	F:1	F:1	F:1	F:0	F:0	F:0	T:0.046	T:0.145	T:0.118	T:0.082	T:0.075	T:0.061	T:0.05	T:0.057	T:0.051	T:0.042		
	4.0	F:1	F:1	F:1	F:1	F:1	F:1	F:1	F:0	F:0	F:0	T:0.162	T:0.118	T:0.096	T:0.062	T:0.066	T:0.051	T:0.04	T:0.047	T:0.04	T:0.033		
	5.0	F:1	F:1	F:1	F:1	F:1	F:1	F:1	F:0	F:0	F:0	T:0.141	T:0.114	T:0.083	T:0.05	T:0.054	T:0.043	T:0.031	T:0.046	T:0.04	T:0.033		
	6.0	F:1	F:1	F:1	F:1	F:1	F:1	F:1	F:0	F:0	F:0	T:0.093	T:0.067	T:0.062	T:0.051	T:0.033	T:0.041	T:0.03	T:0.026	T:0.02	T:0.032		
	7.0	F:1	F:1	F:1	F:1	F:1	F:1	F:1	F:0	F:0	F:0	T:0.085	T:0.054	T:0.047	T:0.037	T:0.033	T:0.027	T:0.022	T:0.026	T:0.021	T:0.017		
	8.0	F:1	F:1	F:1	F:1	F:1	F:1	F:1	F:0	F:0	F:0	T:0.061	T:0.043	T:0.034	T:0.024	T:0.022	T:0.027	T:0.019	F:0	F:0	F:0		
	9.0	F:1	F:1	F:1	F:1	F:1	F:1	F:1	F:0	F:0	F:0	T:0.052	T:0.031	T:0.034	T:0.028	T:0.027	T:0.016	T:0.018	T:0.012	T:0.01	T:0.017		
	10.0	F:1	F:1	F:1	F:1	F:1	F:1	F:1	F:0	F:0	F:0	T:0.044	T:0.035	T:0.027	T:0.016	T:0.014	T:0.016	T:0.009	T:0.012	T:0.006			
	11.0	F:1	F:1	F:1	F:1	F:1	F:1	F:1	F:0	F:0	F:0	T:0.037	T:0.026	T:0.016	T:0.018	T:0.014	T:0.016	T:0.009	T:0.012	T:0.006			
	12.0	F:1	F:1	F:1	F:1	F:1	F:1	F:1	F:0	F:0	F:0	T:0.03	T:0.019	T:0.016	T:0.01	T:0.014	T:0.008	T:0.009	T:0.006	T:0.005	T:0.006		

Fonte: autor.

Figura 29 – Otimização Cena 2 - resolução 600p - faces detectadas. À esquerda, posição 1, e à direita, posição 2.



Fonte: autor.

Figura 30 – Tabela de Dados - resolução 600p.

Device	Resolution	Full Image Encoded (bytes)	Detected Faces	Encoded Faces Size	Capturing Image (s)	Convert Image to Gray (s)	Detection (s)	Build Encoded Faces Images (s)	Total execution time with capture (s)
Raspberry Pi 4	600 x 800	1,920,072	1	179,000	0.094	0.001	0.017	0.002	0.114
Raspberry Pi Zero W	600 x 800	1,920,072	1	179,000	0.099	0.015	0.453	0.006	0.573

Fonte: autor.

4.2.2.2 Resolução 480p

A análise para definição dos parâmetros seguiu-se da mesma forma como já foi explicado na subsubseção anterior. Novamente a região delimitada na matriz de resultados à direita, posição 2, foi a mais restritiva.

Considerando-se o menor tempo na interseção das regiões, os parâmetros definidos foram:

- Min. Size Face: 8.0%
- Scale Factor: 1.500
- Min. Neighbors: 3

A apresentação das imagens com a face detectada na figura 31 pode parecer repetitiva. A intenção é possibilitar uma comparação quanto à qualidade das imagens

quando comparadas às da figura 29, de maior resolução. Como não estão em seus tamanhos reais, pode ser difícil ter a percepção lendo o documento sem ampliação. Na próxima subseção as faces recortadas de cada imagem são postas lado a lado em seus tamanhos reais para possibilitar a percepção da diferença na qualidade.

Figura 31 – Otimização Cena 2 - resolução 480p - matrizes. À esquerda, posição 1, e à direita, posição 2.

All Faces Mean Time	Scale Factor										All Faces Mean Time	Scale Factor									
	1.150	1.200	1.250	1.300	1.350	1.400	1.450	1.500	1.550	1.600		1.150	1.200	1.250	1.300	1.350	1.400	1.450	1.500	1.550	1.600
Min Size (%)	F:1 T: 0.122	F:1 T: 0.088	F:1 T: 0.069	F:1 T: 0.063	F:1 T: 0.054	F:1 T: 0.046	F:1 T: 0.052	F:1 T: 0.045	F:0 T: 0.042	F:0 T: 0.038	F:1 T: 0.115	F:1 T: 0.084	F:1 T: 0.066	F:1 T: 0.059	F:1 T: 0.051	F:1 T: 0.044	F:1 T: 0.047	F:2 T: 0.042	F:1 T: 0.038		
3.0	F:1 T: 0.098	F:1 T: 0.074	F:1 T: 0.054	F:1 T: 0.052	F:1 T: 0.041	F:1 T: 0.035	F:1 T: 0.04	F:1 T: 0.034	F:0 T: 0.03	F:0 T: 0.026	F:1 T: 0.087	F:1 T: 0.067	F:1 T: 0.051	F:1 T: 0.053	F:1 T: 0.039	F:1 T: 0.032	F:1 T: 0.038	F:2 T: 0.032	F:1 T: 0.026	F:1 T: 0.024	
4.0	F:1 T: 0.08	F:1 T: 0.062	F:1 T: 0.048	F:1 T: 0.045	F:1 T: 0.035	F:1 T: 0.024	F:1 T: 0.03	F:1 T: 0.025	F:0 T: 0.023	F:0 T: 0.026	F:1 T: 0.076	F:1 T: 0.056	F:1 T: 0.038	F:1 T: 0.038	F:1 T: 0.03	F:1 T: 0.025	F:1 T: 0.03	F:1 T: 0.029	F:1 T: 0.019	F:1 T: 0.024	
5.0	F:1 T: 0.074	F:1 T: 0.053	F:1 T: 0.038	F:1 T: 0.038	F:1 T: 0.032	F:1 T: 0.026	F:1 T: 0.03	F:1 T: 0.024	F:0 T: 0.023	F:0 T: 0.02	F:1 T: 0.069	F:1 T: 0.052	F:1 T: 0.038	F:1 T: 0.036	F:1 T: 0.027	F:1 T: 0.024	F:1 T: 0.031	F:1 T: 0.026	F:1 T: 0.019	F:1 T: 0.016	
6.0	F:1 T: 0.074	F:1 T: 0.053	F:1 T: 0.038	F:1 T: 0.038	F:1 T: 0.032	F:1 T: 0.026	F:1 T: 0.03	F:1 T: 0.024	F:0 T: 0.023	F:0 T: 0.02	F:1 T: 0.056	F:1 T: 0.04	F:1 T: 0.031	F:1 T: 0.037	F:1 T: 0.025	F:1 T: 0.02	F:1 T: 0.016	F:1 T: 0.026	F:1 T: 0.019	F:1 T: 0.016	
7.0	F:1 T: 0.052	F:1 T: 0.04	F:1 T: 0.034	F:1 T: 0.036	F:1 T: 0.028	F:1 T: 0.023	F:1 T: 0.017	F:1 T: 0.026	F:0 T: 0.025	F:0 T: 0.021	F:1 T: 0.049	F:1 T: 0.032	F:1 T: 0.026	F:1 T: 0.025	F:1 T: 0.013	F:1 T: 0.011	F:1 T: 0.016	F:1 T: 0.012	F:1 T: 0.011	F:1 T: 0.016	
8.0	F:1 T: 0.049	F:1 T: 0.032	F:1 T: 0.026	F:1 T: 0.025	F:1 T: 0.017	F:1 T: 0.022	F:1 T: 0.017	F:1 T: 0.013	F:0 T: 0.011	F:0 T: 0.02	F:1 T: 0.036	F:1 T: 0.024	F:1 T: 0.026	F:1 T: 0.018	F:1 T: 0.013	F:1 T: 0.011	F:1 T: 0.008	F:1 T: 0.026	F:1 T: 0.02	F:1 T: 0.008	
9.0	F:1 T: 0.036	F:1 T: 0.03	F:1 T: 0.024	F:1 T: 0.026	F:1 T: 0.016	F:1 T: 0.012	F:1 T: 0.018	F:1 T: 0.013	F:0 T: 0.011	F:0 T: 0.008	F:1 T: 0.038	F:1 T: 0.026	F:1 T: 0.023	F:1 T: 0.022	F:1 T: 0.015	F:1 T: 0.012	F:1 T: 0.016	F:1 T: 0.012	F:1 T: 0.011	F:1 T: 0.016	
10.0	F:1 T: 0.026	F:1 T: 0.022	F:1 T: 0.016	F:1 T: 0.016	F:1 T: 0.016	F:1 T: 0.011	F:1 T: 0.009	F:1 T: 0.013	F:0 T: 0.011	F:0 T: 0.009	F:1 T: 0.028	F:1 T: 0.022	F:1 T: 0.016	F:1 T: 0.014	F:1 T: 0.008	F:1 T: 0.012	F:1 T: 0.011	F:1 T: 0.008	F:1 T: 0.011	F:1 T: 0.008	
11.0	F:1 T: 0.028	F:1 T: 0.022	F:1 T: 0.016	F:1 T: 0.014	F:1 T: 0.014	F:1 T: 0.012	F:1 T: 0.011	F:1 T: 0.005	F:0 T: 0.011	F:0 T: 0.008	F:1 T: 0.021	F:1 T: 0.013	F:1 T: 0.015	F:1 T: 0.015	F:1 T: 0.012	F:1 T: 0.012	F:1 T: 0.011	F:1 T: 0.011	F:1 T: 0.008		
12.0	F:1 T: 0.021	F:1 T: 0.014	F:1 T: 0.016	F:1 T: 0.008	F:1 T: 0.008	F:1 T: 0.012	F:1 T: 0.007	F:1 T: 0.005	F:0 T: 0.004	F:0 T: 0.008	F:1 T: 0.019	F:1 T: 0.013	F:1 T: 0.015	F:1 T: 0.006	F:1 T: 0.006	F:1 T: 0.012	F:1 T: 0.006	F:1 T: 0.004	F:1 T: 0.003	F:1 T: 0.008	

Fonte: autor.

Figura 32 – Otimização Cena 2 - resolução 480p - faces detectadas. À esquerda, posição 1, e à direita, posição 2.



Fonte: autor.

Figura 33 – Tabela de Dados - resolução 480p.

Device	Resolution	Full Image Encoded (bytes)	Detected Faces	Encoded Faces Size	Capturing	Convert Image to Gray (s)	Detection (s)	Build Encoded Faces Images (s)	Total execution time with capture (s)
Raspberry Pi 4	480 x 640	1,228,872	1	92,488	0.083	0.001	0.013	0.001	0.098
Raspberry Pi Zero W	480 x 640	1,228,872	1	92,488	0.094	0.012	0.257	0.004	0.367

Fonte: autor.

4.2.2.3 Resolução 240p

A análise para definição dos parâmetros seguiu-se da mesma forma como nas subsubseções anteriores. Novamente a região delimitada na matriz de resultados à direita, posição 2, foi a mais restritiva.

Considerando-se o menor tempo na interseção das regiões, os parâmetros definidos foram:

- Min. Size Face: 7.0%
- Scale Factor: 1.250
- Min. Neighbors: 3

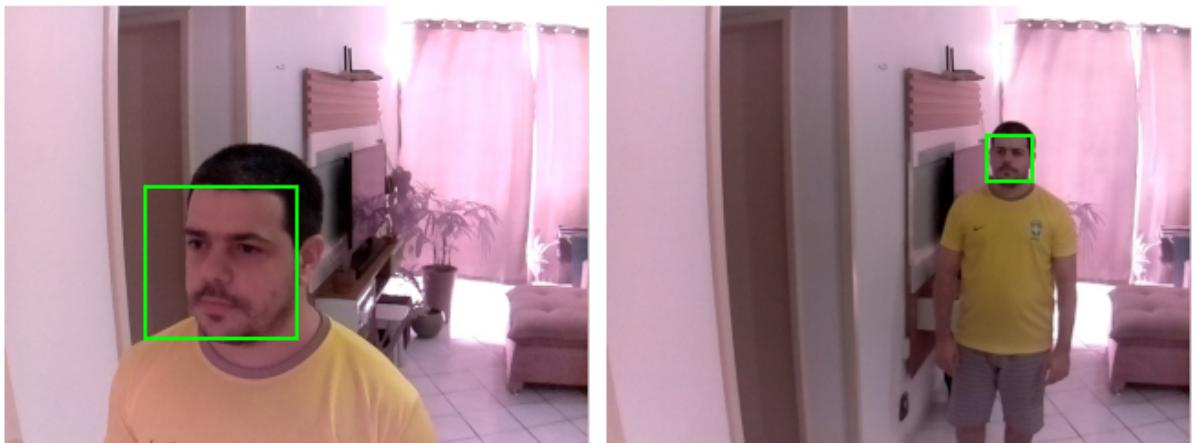
Figura 34 – Otimização Cena 2 - resolução 240p - matrizes. À esquerda, posição 1, e à direita, posição 2.



Fonte: autor.

Nessa resolução, já se consegue perceber mais claramente a diferença da qualidade das imagens apresentadas na figura 35 se comparadas às imagens das figuras 32 e 29, com resoluções maiores.

Figura 35 – Otimização Cena 2 - resolução 240p - faces detectadas. À esquerda, posição 1, e à direita, posição 2.



Fonte: autor.

Figura 36 – Tabela de Dados - resolução 240p.

Device	Resolution	Full Image Encoded (bytes)	Detected Faces	Encoded Faces Size	Capturing Image (s)	Convert Image to Gray (s)	Detection (s)	Build Encoded Faces Images (s)	Total execution time with capture (s)
Raspberry Pi 4	240 x 320	307,272	1	29,088	0.062	0.000	0.017	0.000	0.079
Raspberry Pi Zero W	240 x 320	307,272	1	29,088	0.072	0.003	0.298	0.001	0.374

Fonte: autor.

4.2.3 Análise dos resultados

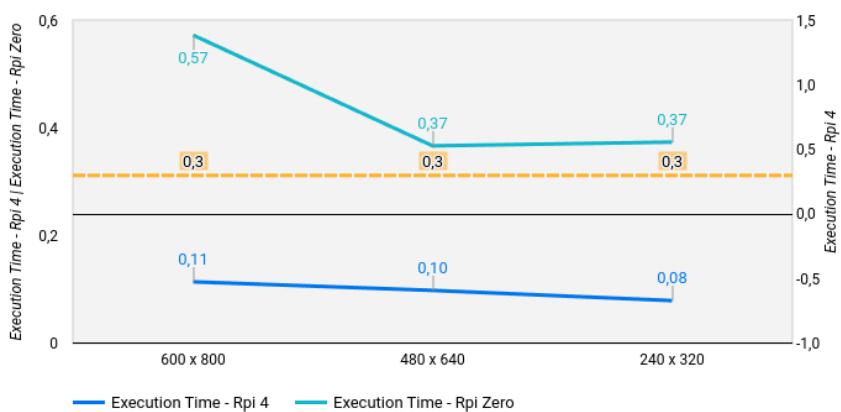
Nesta cena, são dois os fatores que influenciam no resultado das detecções, que são a resolução da imagem e o dispositivo rodando o algoritmo de detecção, sobre os quais serão feitas as comparações e análises a seguir.

Os dados das métricas em todas as resoluções foram obtidos a partir da imagem na posição 1, pois apresentou maiores tempos de resposta e maiores tamanhos de imagem da face recortada em relação à posição 2, sendo assim o pior caso.

4.2.3.1 Efeitos da variação por resolução de imagem e considerações sobre os dispositivos testados

O gráfico da figura 4.2.3.1 compara o tempo de resposta entre os dois dispositivos testados, linhas azuis claro e escuro, e nas diferentes resoluções, distribuídas no eixo x. Também foi representado no gráfico o tempo máximo de resposta estabelecido na cena, 0,3 segundo, linha tracejada amarela.

Figura 37 – Tempos de execução por Variação de resolução.



Fonte: autor.

Como pode ser observado, os tempos de resposta do Raspberry Pi 4B ficaram bem abaixo do tempo máximo definido para esta cena, chegando a no máximo 0,11 segundo (63% de margem), na resolução de 600 x 800, e reduzindo não muito expressivamente à medida que se reduz a resolução da imagem. Com isso, pode-se concluir que o Raspberry Pi 4B é capaz de realizar a detecção de faces únicas em tempo suficientemente pequeno para atender aos requisitos da cena nas três resoluções testadas. Isso significa que, considerando-se o cenário estabelecido e o ciclo de detecção rodando a cada 0,3s, o dispositivo é capaz de obter a imagem e realizar a detecção da face (se a posição for favorável) de uma mesma pessoa pelo menos três vezes, considerando-se essa pessoa caminhando rapidamente em direção à entrada do ambiente (um pior caso), conforme demonstrado na subsubseção 3.3.2.1.

Quanto à margem de pelo menos 63% no tempo de resposta do Raspberry Pi 4B, pode-se pensar em aproveitá-la de pelo menos duas formas. Uma delas seria considerar um período menor no ciclo de detecção. Se o ciclo de detecção rodar

conforme o tempo de resposta do dispositivo e não a cada 0,3s, a quantidade de frames capturados e analisados de uma mesma pessoa pode chegar a aproximadamente 10. Outra possibilidade seria a sobra de tempo de processamento para realizar outros tratamentos ou análises na imagem, como por exemplo o reconhecimento da face detectada. Realizar o reconhecimento de face na borda poderia gerar consideráveis ganhos em aplicações distribuídas, assunto que poderia ser objeto de um estudo futuro.

Uma questão interessante a se observar é quanto à parcela de cada etapa de processamento no tempo total de resposta. A figura 38 mostra o gráfico em barras empilhadas mostrando a contribuição no tempo de execução de cada etapa, no Raspberry Pi 4B, por cada resolução. Como pode ser observado, as etapas de converter a imagem para escala de cinza (barra amarela) e o encodamento das faces recortadas (barra laranja) têm tempos de execução tão pequenos, que foram aproximados a 0 na precisão de 2 casas decimais, podendo ser considerados então insignificantes nesse caso.

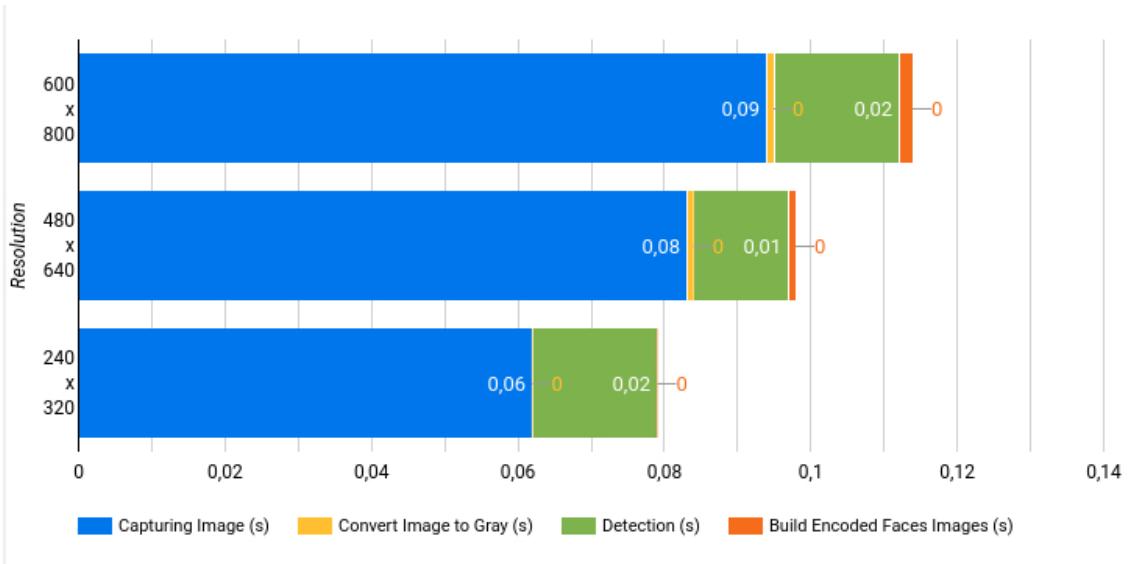
Quando se os tempos mais significantes observa-se que o tempo de execução de captura da imagem (em azul) é consideravelmente maior que o tempo de execução da própria detecção de faces em si, isso para todas as resoluções. Verifica-se também que a variação no tempo total de execução, dá-se por influência majoritária do tempo de captura dos frames que, logicamente, são menores conforme se reduz a resolução da imagem.

Importante ressaltar também que existe ainda o potencial de se reduzir ainda mais o tempo de resposta total implementando-se estratégia de processamento assíncrono entre a etapa de captura de frames e as demais, o que não foi feito para este estudo.

Com relação a qual das resoluções adotar, ainda analisando-se os resultados obtidos com o Raspberry Pi 4B, poderia-se dizer a menor resolução (240 x 320) seria a mais adequada em se considerando apenas tempo de resposta. A utilização da menor resolução também irá proporciona a vantagem de utilizar menor largura de banda para trafegar os dados das imagens na rede e também ocuparia menor espaço de armazenamento (se for o caso).

Porém, se a qualidade das imagens das faces for um fator importante para pos-

Figura 38 – Parcela de cada etapa no tempo de execução no Raspberry Pi 4, por resolução.



Fonte: autor.

teriores tratamentos das imagens dentro da aplicação, a escolha de se utilizar as resoluções maiores pode fazer grande diferença. Para demonstrar isso, as figuras 39 e 40, fazem comparativos entre as faces recortadas nas duas posições e nas diferentes resoluções. Na figura 39, as imagens estão apresentadas nas proporções reais entre si, considerando-se seu tamanho real, em pixels. Na figura 40, as faces de resoluções menores foram alargadas de forma que todas as faces ficassem aproximadamente do mesmo tamanho. Assim, é possível perceber com clareza a diferença na qualidade da imagens. Na face menor recortada da resolução 240 x 320 é evidente a perda de definição nos detalhes da face, algo que poderia fazer toda a diferença na qualidade do resultado de um reconhecimento de face, por exemplo.

Quanto aos resultados do Raspberry Pi Zero W, observa-se na figura que na maior resolução, o tempo de resposta foi relativamente alto, 0,57s, quase o dobro do limite máximo definido. Ao se utilizar resoluções menores, 480 x 640 e 240 x 320, o tempo de resposta reduziu consideravelmente, chegando a 0,37s em ambos os casos, apenas 0,07s acima do limite estabelecido para a cena. Por conta disso poderia-se considerar que o uso desse dispositivo não seria muito adequado para esse tipo de cena, porém, pode-se considerar algumas possibilidades de melhoria. Por exemplo, a otimização de todo o processo utilizando-se técnicas de processamento assíncrono ou o teste

Figura 39 – Comparativo de faces recortadas de diferentes resoluções, em tamanho real.



Fonte: autor.

Figura 40 – Comparativo de faces recortadas de diferentes resoluções, em tamanhos ajustados.



Fonte: autor.

de resoluções ainda menores têm o potencial de apresentar reduções consideráveis no tempo de resposta. Ou então a aplicação em cenários com critérios um pouco diferentes podem tornar esses tempos de respostas mais aceitáveis, como por exemplo em casos em que seja considerado um tempo maior de exposição das faces em frente à câmera.

Em termos apenas de performance, a utilização do Raspberry Pi 4B é mais recomendada, considerando-se que o tempo de resposta chega a ser em torno de 78% menor do que o observado no Raspberry Pi Zero W. Porém, há outros fatores que podem ser decisivos na escolha do dispositivo a ser utilizado em um projeto, como o valor e o tamanho. O Raspberry Pi Zero W custa em torno de 73% menos que o Raspberry Pi 4B² e possui área torno de 59% menor. O desempenho do Raspberry Pi Zero W não atendeu totalmente aos requisitos definidos para este cenário específico, porém se mostrou muito promissor. Caso fatores de custo e/ou dimensões sejam relevantes para o projeto, vale a pena considerá-lo como opção.

4.2.3.2 Considerações sobre utilização de banda

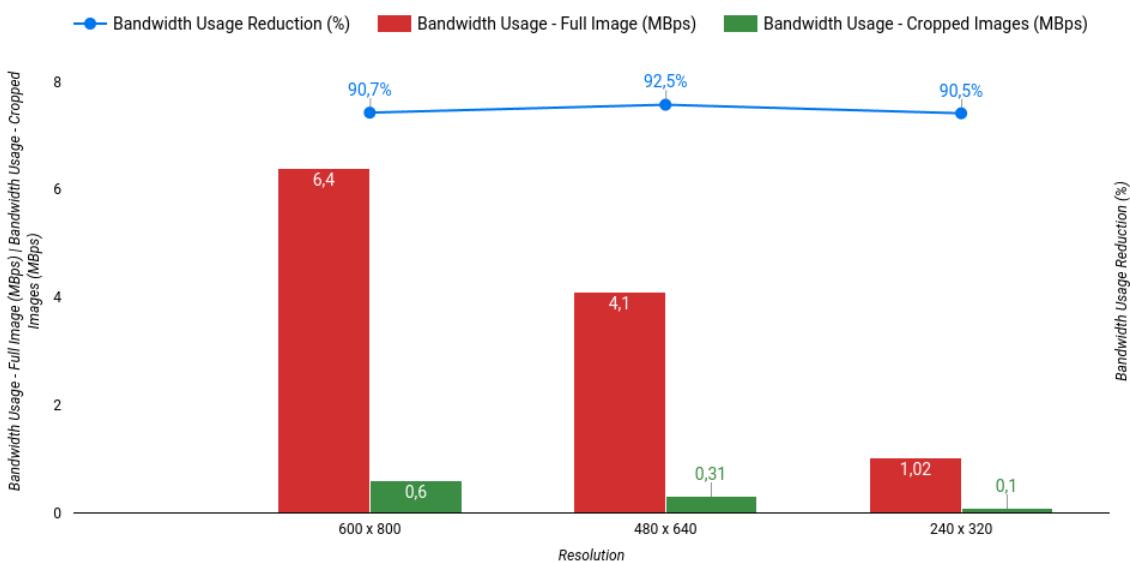
Considerando-se uma possível aplicação com arquitetura distribuída, a figura 41 mostra um comparativo de como seria a utilização de banda em dois possíveis casos, transmissão da imagem completa capturada pelo dispositivo da borda para outro dispositivo realizar a detecção, e transmissão apenas as imagens das faces já detectadas e recortadas pelo dispositivo na borda. Para o cálculo de utilização de banda, considerou-se intervalo entre as detecções de 0,3 segundo, conforme tempo máximo definido como requisito para esta cena.

São apresentados os resultados por resolução de imagem, eixo x. Nas barras de cor vermelha têm-se a largura de banda que seria utilizada trafegando-se a imagem completa e nas barras de cor verde a largura de banda considerando-se o tráfego apenas do recorte das faces detectadas, ambas seguindo a escala à esquerda do eixo y e expressas em MBps. Na série em formato de linha azul têm-se a diferença percentual em relação ao tamanho da imagem completa, seguindo a escala à direita do eixo y.

Como pode-se observar, a redução na utilização de banda é bastante expressivo quando se processa a detecção de faces na borda, variando em torno de 90% e 93% entre as diferentes resoluções. Importante ressaltar que esses dados foram obtidos a

² Conforme preços encontrados em visita à loja online com endereço www.robocore.net, no dia 30/11/2022. Como não havia disponível para venda o modelo exato do Raspberry Pi Zero W, foi considerado o valor do Raspberry Pi Zer 2 W, que possui configurações de hardware semelhantes.

Figura 41 – Comparativo de utilização de banda.



Fonte: autor.

partir da imagem da posição 1, quando o tamanho da face é maior. Ou seja, na média, esse ganho na redução tende a ser um pouco maior já que haverá frames capturados com faces menores, chegando a até 100% quando não houver face presente.

Reforçando-se que as unidades no gráfico estão em MBps e não em Mbps e considerando-se as imagens trafegando em uma rede de 100 MBps, transmitir a imagem completa na resolução 600 x 800 utilizaria em média 51% de toda a banda, para apenas esse dispositivo, enquanto que transmitir apenas as imagens das faces recortadas utilizaria em torno de 5% da banda da rede, em média.

Novamente, em uma aplicação real, é razoável considerar que podem haver dezenas de dispositivos realizando monitoramento e tendo de enviar imagens e dados a algum outro dispositivo centralizador. A tabela apresentada na figura 42 apresenta um comparativo de quanto seria a utilização média de banda para diferentes quantidades de dispositivos transmitindo as imagens das faces detectadas simultaneamente em uma mesma rede. São apresentados os valores para cada resolução testada.

Primeiramente, os dados da tabela mostram que, com a detecção de faces sendo processada pelo dispositivo de borda, a implementação de vários desses dispositivos funcionando simultaneamente em uma mesma rede se torna viável, graças à transmissão apenas das partes úteis das imagens. Outra questão que se observa é o quão

Figura 42 – Utilização de banda em Mbps por quantidade de dispositivos e resolução.

Utilização de banda (Mbps)	Resolução		
	600x 800	480 x 640	240 x 320
Quantidade Dispositivos	1	5	2
	5	24	12
	10	48	25
	20	96	49
	30	143	74
Quantidade Dispositivos	1	5	2
	5	24	12
	10	48	25
	20	96	49
	30	143	74

Fonte: autor.

importante se torna a escolha da resolução da imagem nesses casos. Dependendo da quantidade de dispositivos, a utilização da resolução 600 x 800 se tornaria inviável em uma rede de 100 Mbps, como é o caso de 20 dispositivos por exemplo, que ocuparia até 96% da banda disponível. Porém, para essa mesma quantidade na resolução de 240 x 320, a utilização de banda já cairia drasticamente para um máximo de 16%, claro que a custo da qualidade da imagem.

5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

lembretes para discussão:

cena 2 - no rpi4 todos atendem e quanto menos resolução mais rápido, porém menos qualidade. Talvez faça sentido usar resolução mais alta para ter mais qualidade na frente como em algum reconhecimento de face - no RpiZ não atende. Talvez possa ser usado para reconhecimento mais próximo como identificação de face para liberação de acesso com a face parada por mais tempo na frente, poderia-se até utilizar resolução menor, etc Coisa que pode entrar em trabalhos futuros tabelados

- Tempo de captura relativamente grande. Possibilidade de melhorar tempo total com threads separadas e outras formas de captura, etc. (tb possível em trabalhos futuros)

- AAZAM, M.; HUH, E. N. Fog computing and smart gateway based communication for cloud of things. In: *Proceedings - 2014 International Conference on Future Internet of Things and Cloud, FiCloud 2014*. [S.l.: s.n.], 2014. ISBN 9781479943586. 11
- AGGARWAL, C. C. *Neural Networks and Deep Learning*. [S.l.: s.n.], 2018. 10, 11, 17, 18
- ANDERSON, D. A. The aggregate burden of crime. *Journal of Law and Economics*, 1999. ISSN 00222186. 14
- ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Computer networks*, Elsevier, v. 54, n. 15, p. 2787–2805, 2010. 23
- Bellavista, P. et al. Convergence of manet and wsn in iot urban scenarios. *IEEE Sensors Journal*, v. 13, n. 10, p. 3558–3567, 2013. 23
- BROWNLEE, J. *What is Deep Learning?* 2019. Disponível em: <<https://machinelearningmastery.com/what-is-deep-learning/>>. 17
- BRUNELLI, R.; POGGIO, T. Face recognition through geometrical features. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [S.l.: s.n.], 1992. ISBN 9783540554264. ISSN 16113349. 22
- CENDRILLON, R.; LOVELL, B. Real-time face recognition using eigenfaces. In: *Visual Communications and Image Processing 2000*. [S.l.: s.n.], 2000. ISSN 0277786X. 22
- CENEDESE, A. et al. Padova smart City: An urban Internet of Things experimentation. In: *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, WoWMoM 2014*. [S.l.: s.n.], 2014. ISBN 9781479947867. 15
- CERQUEIRA, D. R. C. et al. ANÁLISE DOS CUSTOS e Consequencias da violência no Brasil. *Texto Para Discussão*, 2007. 10, 14
- CHEN, N. et al. Dynamic urban surveillance video stream processing using fog computing. In: *Proceedings - 2016 IEEE 2nd International Conference on Multimedia Big Data, BigMM 2016*. [S.l.: s.n.], 2016. ISBN 9781509021789. 10, 16, 17
- COX, I. J.; GHOSN, J.; YANILOS, P. N. Feature-based face recognition using mixture-distance. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 1996. ISSN 10636919. 22
- DataStudio. 2022. [Online; acessado em 20-11-2022]. Disponível em: <<https://datastudio.google.com/>>. 56
- DLIB C++ Library. Disponível em: <<http://dlib.net/>>. 22

DOLUI, K.; DATTA, S. K. Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In: *GloTS 2017 - Global Internet of Things Summit, Proceedings*. [S.I.: s.n.], 2017. ISBN 9781509058730. 11

FISCHLER, M. A.; ELSCHLAGER, R. A. The Representation and Matching of Pictorial Structures Representation. *IEEE Transactions on Computers*, 1973. ISSN 00189340. 20

G1. *Brasil tem a terceira maior taxa de roubos da América Latina, diz Pnud*. 2013. 10, 14

HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.I.: s.n.], 2016. ISBN 9781467388504. ISSN 10636919. 18

HELLER, M. *Melhores bibliotecas de Machine e Deep Learning*. 2019. Disponível em: <<https://cio.com.br/melhores-bibliotecas-de-machine-e-deep-learning/>>. 18

HU, W. et al. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 2004. ISSN 10946977. 16

JAFRI, R.; ARABNIA, H. R. A Survey of Face Recognition Techniques. *Journal of Information Processing Systems*, 2009. ISSN 1976-913X. 21

JEBARA, T. S. 3D POSE ESTIMATION AND NORMALIZATION FOR FACE RECOGNITION. *Department of Electrical Engineering*, 1996. 22

KANADE, T. *Computer recognition of human faces*. [S.I.: s.n.], 1977. 20, 21

LABLED Faces in the Wild. Disponível em: <<http://vis-www.cs.umass.edu/lfw/>>. 22

MERENDA, M.; PORCARO, C.; IERO, D. Edge machine learning for ai-enabled iot devices: A review. *Sensors (Switzerland)*, 2020. ISSN 14248220. 11

NEAPOLITAN, R. E. Neural Networks and Deep Learning. *Artificial Intelligence*, p. 389–411, 2018. 18

NIKOUEI, S. Y. et al. Smart surveillance as an edge network service: From harr-cascade, SVM to a Lightweight CNN. In: *Proceedings - 4th IEEE International Conference on Collaboration and Internet Computing, CIC 2018*. [S.I.: s.n.], 2018. ISBN 9781538695029. 10, 16, 17

OLSON, T. J. *AUTOMATIC VIDEO MONITORING SYSTEM WHICH SELECTIVELY SAVES INFORMATION*. 2006. 16 p. Disponível em: <<https://patentimages.storage.googleapis.com/f0/7b/a2/58558376b25dca/US7023469.pdf>>. 15

OpenCV-CascadeClassifier. 2022. [Online; acessado em 15-09-2022]. Disponível em: <https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html>. 26

OpenCV-PreTrainedModels. 2022. [Online; acessado em 15-09-2022]. Disponível em: <<https://github.com/opencv/opencv/tree/master/data/haarcascades>>. 26

- PACHECO, A. et al. A Smart Classroom Based on Deep Learning and Osmotic IoT Computing. In: *2018 Congreso Internacional de Innovacion y Tendencias en Ingenieria, CONIITI 2018 - Proceedings*. [S.I.: s.n.], 2018. ISBN 9781538681312. 10, 24, 25
- Picamera. 2022. [Online; acessado em 23-10-2022]. Disponível em: <<https://picamera.readthedocs.io/>>. 38
- PORTER, R.; FRASER, A. M.; HUSH, D. Wide-area motion imagery. *IEEE Signal Processing Magazine*, 2010. ISSN 10535888. 17
- PUVVADI, U. L. et al. Cost-effective security support in real-time video surveillance. *IEEE Transactions on Industrial Informatics*, 2015. ISSN 15513203. 15, 16
- QUIRITA, V. H. A. ESTUDO DE MÉTODOS AUTOMÁTICOS DE RECONHECIMENTO FACIAL PARA VÍDEO MONITORAMENTO. 2014. 10, 19, 20, 21
- Ramos Lima, G.; Marques Ciarelli, P. Sistema de Videomonitoramento com Identificação de Suspeitos Utilizando Biometria Facial. In: . [S.I.: s.n.], 2019. 15
- SINGH, S. Optimize cloud computations using edge computing. In: IEEE. *2017 International Conference on Big Data, IoT and Data Science (BID)*. [S.I.], 2017. p. 49–53. 24
- SONG, H. et al. *Smart Cities: Foundations, Principles, and Applications*. [S.I.: s.n.], 2017. ISBN 9781119226444. 10
- Stan Z. Li, A. K. J. *Handbook of Face Recognition*. Second edi. New York: Springer, 2011. 20, 21
- SZELISKI, R. Computer vision: algorithms and applications. *Choice Reviews Online*, 2011. ISSN 0009-4978. 11, 19, 20
- TSAKANIKAS, V.; DAGIUCLAS, T. Video surveillance systems-current status and future trends. *Computers and Electrical Engineering*, 2018. ISSN 00457906. 16, 19
- VERHELST, M.; MOONS, B. Embedded Deep Neural Network Processing: Algorithmic and Processor Techniques Bring Deep Learning to IoT and Edge Devices. *IEEE Solid-State Circuits Magazine*, 2017. ISSN 19430582. 11
- VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.I.: s.n.], 2001. ISSN 10636919. 19, 26, 56
- WILSON, J.; KELLING, G. Broken Windows: the police and neighborhood safety. *The Atlantic Monthly*, 1982. ISSN 0094-6575. 15
- ZAFEIRIOU, S.; ZHANG, C.; ZHANG, Z. A survey on face detection in the wild: Past, present and future. *Computer Vision and Image Understanding*, 2015. ISSN 1090235X. 19, 20