UNIVERSIDADE FEDERAL DE SÃO CARLOS
CCET - Centro de Ciências Exatas e Tecnologias

HaarCascade vs. Mediapipe
Estudando a eficiência de algoritmos de reconhecimento facial

Sebastião Venâncio Guimarães Neto

São Carlos, 18 de fevereiro de 2023

Introdução:

Neste experimento, foram utilizados dois algoritmos para detecção facial, sendo um utilizando a biblioteca opencv juntamente com um algoritmo Haar Cascade para a detecção, e outro utilizando opencv com Mediapipe, que é biblioteca para reconhecimento de imagens criada pela Google, com o objetivo de comparar o tempo de execução real em diferentes condições. Além disso, uma versão adaptada do Haar Cascade também foi utilizada para fins de comparação.

Procedimento Experimental:

Para realizar este experimento, foi utilizado um notebook com processador Intel Core i5-8625U com 8GB de memória RAM, um SSD de 128GB e uma placa de vídeo MX110 2GB conectado à energia, em modo alto desempenho do Windows 10. Sendo assim, antes de iniciar as medidas, o computador foi reiniciado e, em seguida, apenas o VSCODE foi aberto.

As medidas foram realizadas utilizando a função time (time python nome.py) e o valor considerado foi o tempo real de execução, uma vez que os algoritmos não aguardam nenhuma resposta do usuário, fato que o torna o parâmetro mais adequado para a análise.

Assim, tais medições foram feitas com vídeos que foram gravados através do código gravador.py, a 20 fps e um tamanho de 640x480 para 3 casos diferentes, são eles: vídeo totalmente preto, vídeo com apenas um rosto e vídeo com 2 rostos, os quais são representados respectivamente por Caso1, Caso2 e Caso3, usados para testar a eficiência de cada algoritmo.

Por fim, com os dados obtidos foi possível calcular a média para chegar a um valor mais preciso, além de analisar o uso da CPU para o terceiro caso e concluir o experimento.

Apresentação dos resultados:

Medidas feitas com o vídeo totalmente preto:

Caso1	Tempo Real 1 [s]	Tempo Real 2 [s]	Tempo Real 3 [s]	Média [s]
CV2 + Haar	7,860	6,984	6,829	7,224
CV2 + MP	6,071	5,830	5,804	5,902

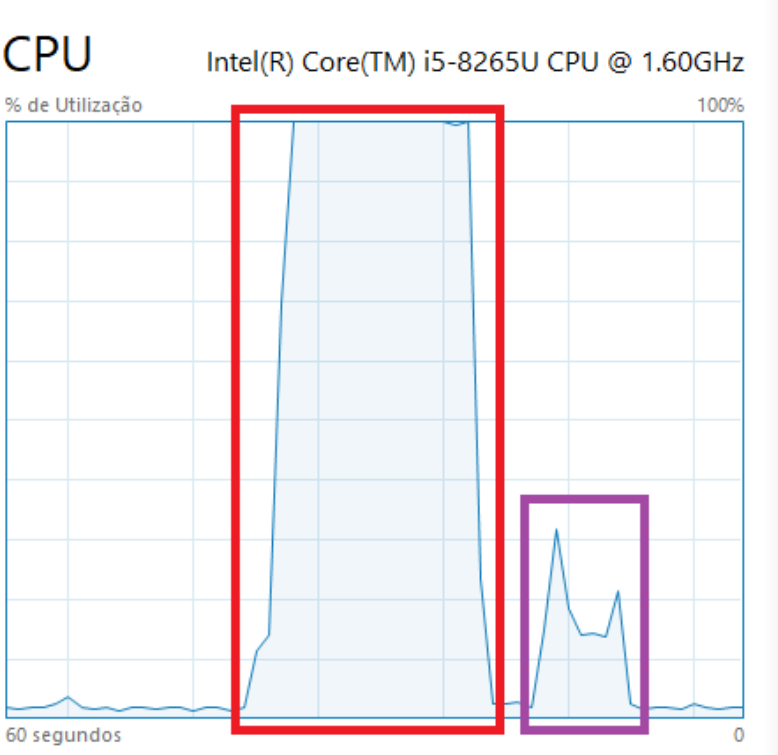
Medidas feitas com apenas um rosto:

Caso2	Tempo Real 1 [s]	Tempo Real 2 [s]	Tempo Real 3 [s]	Média [s]
CV2 + Haar	16,061	15,951	15,972	15,995
CV2 + MP	5,887	5,958	6,024	5,956

Medidas feitas com dois rostos:

Caso3	Tempo Real 1 [s]	Tempo Real 2 [s]	Tempo Real 3 [s]	Média [s]
CV2 + Haar	18,156	20,238	18,888	19,094
CV2 + MP	5,963	5,971	5,926	5,953

Uso da CPU durante a execução do algoritmo Haars e da biblioteca Mediapipe para o Caso 3:



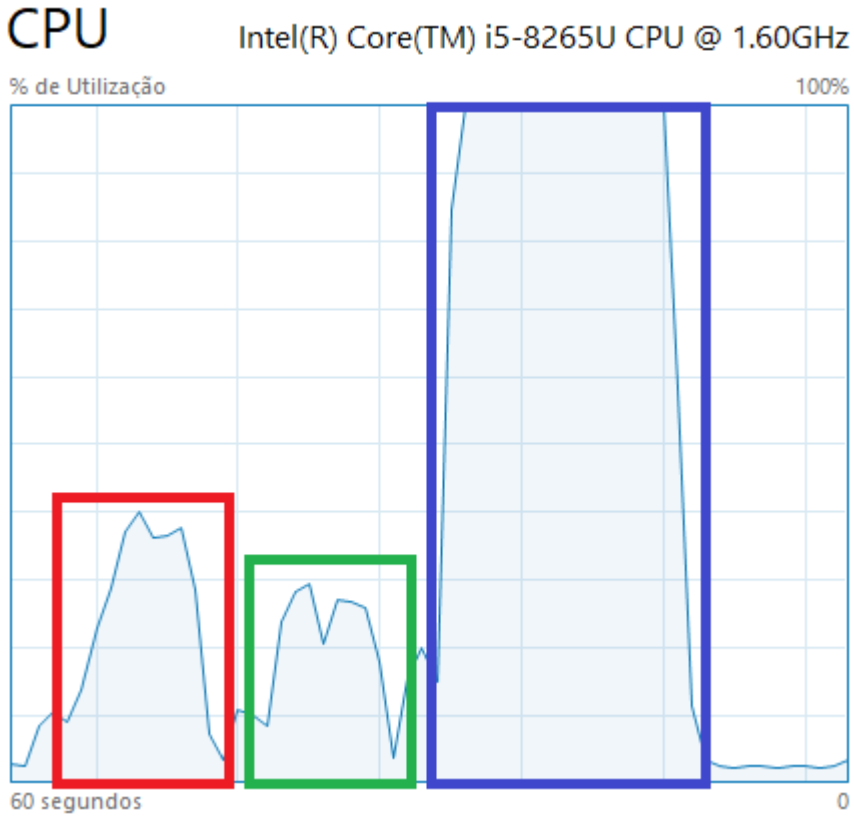
O retângulo vermelho representa o uso pelo Haar e o roxo representa o uso da CPU pelo Mediapipe.

O algoritmo HaarCascade faz comparações a cada frame, o que usa 100% da CPU, então, na tentativa de otimizar a execução do código, foi utilizado uma condição que limita o reconhecimento após uma quantidade fixa de frames, sem perder a precisão.

Resultados do tempo de execução do ModoEconomico do HaarCascade para o Caso3 (tempo vs reconhecimento a cada x frames):

5 FPS	8 FPS	10 FPS	15 FPS	20 FPS
9,057	7,958	7,589	7,248	7,025

Uso da CPU comparando duas execuções do algoritmo ModoEconomico vs uma execução do Haar:



O retângulo vermelho representa o algoritmo ModoEconomico configurado para reconhecer a cada 8 frames, o verde configurado para 10 frames e o azul é uma execução do Haar normal.

Conclusão:

Dados os resultados acima, pode-se analisar uma semelhança no tempo de execução para o caso do vídeo totalmente preto, enquanto que nos demais o algoritmo com Mediapipe se sobressaiu. Além disso, durante a execução dos algoritmos, foi possível notar que o Mediapipe mostrou uma melhor fluidez no vídeo apresentado, além de um melhor desempenho ao reconhecer a face, indicando também olhos, nariz, orelhas e boca, enquanto que, para o Haar Cascade, seria necessário adicionar mais arquivos .xml para adicionar essas funcionalidades de reconhecimento de partes da face, fato que diminuiria a eficiência desse, o qual não já não mostra tanta fluidez na execução do vídeo.

Pode-se indicar essa falta de fluidez do Haar no tratamento frame a frame, o qual aparenta atrasar a execução, visto que é feito explicitamente a conversão de imagem para cinza e a consulta ao arquivo .xml para fazer a detecção da face, além de utilizar um loop para desenhar o retângulo em cada face, assim, o Mediapipe, no Caso3 utilizou apenas 31,18% do tempo do Haarcascade.

Tomando o tratamento frame a frame como o principal motivo do atraso, uma versão para economizar processamento e acelerar a execução do HaarCascade foi desenvolvida visando realizar o reconhecimento a cada quantidade fixa de frames, fato que gerou bons resultados, comparando com a primeira versão do algoritmo. Assim, tendo como taxa fixa de reconhecimento 10 frames, tivemos uma redução de 60,25%, e, para 8 frames, 58,32%.

Por fim, nenhum dos algoritmos desenvolvidos usando HaarCascade não conseguiram superar a eficiência e qualidade do Mediapipe, o qual supera no tempo de execução, no uso da CPU e na qualidade do reconhecimento em vista dos outros.

Bibliografia:

1. https://www.hashtagtreinamentos.com/controlar-webcam-com-python#:~:text=Inicialmente%20n%C3%B3s%20vamos%20importar%20a,e%20o%20c%C3%B3digo%20vai%20continuar.
2. https://towardsdatascience.com/face-detection-in-2-minutes-using-opencv-python-90f89d7c0f81
3. https://www.hashtagtreinamentos.com/reconhecimento-facial-no-python