

**FACULDADE ANHANGUERA DE SÃO JOSÉ DOS CAMPOS**

**6º CIÊNCIA DA COMPUTAÇÃO**

**INTELIGÊNCIA ARTIFICIAL E COMPUTACIONAL**

**PROF. NEWTON S. JUAN**

Reconhecimento facial e de objetos com OpenCV

Breno Renan da Cunha **RA:231910411919**

Gustavo da Silva Gouvêa **RA:227486011919**

|  |  |
| --- | --- |
| **Proposta de projeto de IAC** | |
| **Proposta:** | Localizar e destacar faces em imagens ou vídeos. |

Criação de um programa para detecção de faces, Será utilizada a linguagem python juntamente com os classificadores **haarcascade** da biblioteca OpenCV para localizar a área onde a face se encontra e destacá-la através de uma marcação. Inicialmente encontrar faces genericamente, se possível implementarmos um banco de dados para guardar e identificar pessoas específicas.

Componentes a serem utilizados:

* **OpenCV:** Biblioteca voltada ao desenvolvimento de aplicativos na área de Visão computacional.
* **haarcascade:** Grupo de classificadores disponíveis no Opencv.
* **SQLite3:** Banco de dados portátil.
* **Numpy:** Trabalha arrays e matrizes multidimensionais.

Execução do código:

Nosso código começa em um menu, cujas opções são:

1. Identificar rostos em webcam/ vídeo.
   1. O programa primeiramente abre um feed de vídeo, então ele roda cada frame pelos classificadores e tem como output aquele que obteve melhor resultado, então é desenhado um quadrado ao redor da área detectada e mandada na janela do programa.
2. Identificar rostos em imagens contidas na pasta padrão.
   1. O programa lê as imagens em formato .jpg e as carrega em um vetor, então as roda em loop pelos classificadores que tem como saída o que encontrou mais rostos, então os quadrados são desenhados ao redor da área detectada e temos então o output.
3. Identificar objetos específicos, de acordo com características
   1. O programa foi treinado com base em imagens ‘positivas’, ou seja, que contém o objeto desejado, colocando-as contra imagens ‘negativas’ (que não contêm), assim criamos um classificador melhorado para detecção de uma objeto específico. O processo de detecção é o mesmo, porém utilizamos o novo classificador **haarcascades** gerado neste processo.

Funcionamento das haarcascades na detecção:

Uma das grandes habilidades dos seres humanos é a capacidade de rapidamente identificar padrões em imagens. Isso sem dúvida foi crucial para a sobrevivência da humanidade até os dias de hoje. Busca-se desenvolver a mesma habilidade para os computadores através da visão computacional e várias técnicas foram criadas nos últimos anos visando este objetivo. O que há de mais moderno (estado da arte) atualmente são as técnicas de “deep learning” ou em uma tradução livre “aprendizado profundo” que envolvem algoritmos de inteligência artificial e redes neurais para treinar identificadores. Outra técnica bastante utilizada e muito importante são os “**haar-like cascades features**” que traduzindo seria algo como “características em cascata do tipo haar” já que a palavra “haar” não possui tradução já que o nome deriva dos “wavelets Haar” que foram usados no primeiro detector de rosto em tempo real.

Historicamente os algoritmos sempre trabalharam apenas com a intensidades dos pixels da imagem. Contudo, uma publicação de Oren Papageorgio "A general framework for object detection" publicada em 1998 mostrou um recurso alternativo baseado em Haar wavelets *(A Transformada de Haar é um* [*transformada*](https://pt.wikipedia.org/w/index.php?title=Transformada&action=edit&redlink=1) *matemática* [*discreta*](https://pt.wikipedia.org/wiki/Discreta) *usada no* [*processamento*](https://pt.wikipedia.org/wiki/Processamento_de_sinais) *e* [*análise de sinais*](https://pt.wikipedia.org/w/index.php?title=An%C3%A1lise_de_sinais&action=edit&redlink=1)*, na* [*compressão de dados*](https://pt.wikipedia.org/wiki/Compress%C3%A3o_de_dados) *e em outras aplicações de engenharia e ciência da computação tendo como caracteristica um pulso quadrado*) em vez das intensidades de imagem, a principal vantagem da técnica é a baixa necessidade de processamento para realizar a identificação dos objetos, o que se traduz em alta velocidade de detecção. Seus criadores então adaptaram a idéia de usar ondas Haar e desenvolveram as chamadas Haar-like features ou características Haar.

Uma característica Haar-like considera as regiões retangulares adjacentes num local específico (janela de detecção) da imagem, então se processa a intensidades dos pixel em cada região e se calcula a diferença entre estas somas. Esta diferença é então usada para categorizar subseções de uma imagem. Neste trabalho por exemplo utilizamos faces humanas. É uma característica comum que entre todas as faces a região dos olhos é mais escura do que a região das bochechas. Portanto, uma característica Haar comum para a detecção de face é um conjunto de dois retângulos adjacentes que ficam na região dos olhos e acima da região das bochechas. A posição desses retângulos é definida em relação a uma janela de detecção que age como uma caixa delimitadora para o objeto alvo (a face, neste caso).

Na fase de detecção da estrutura de detecção de objetos, uma janela do tamanho do alvo é movida sobre a imagem de entrada, e para cada subseção da imagem é calculada a característica do tipo Haar. Essa diferença é então comparada a um limiar aprendido que separa não-objetos de objetos. Como essa característica Haar é apenas um classificador fraco (sua qualidade de detecção é ligeiramente melhor que a suposição aleatória), um grande número de características semelhantes a Haar são necessárias para descrever um objeto com suficiente precisão. Na estrutura de detecção de objetos Viola-Jones, as características de tipo Haar são, portanto, organizadas em algo chamado cascata de classificadores para formar classificador forte. A principal vantagem de um recurso semelhante ao Haar sobre a maioria dos outros recursos é a velocidade de cálculo. Devido ao uso de imagens integrais, um recurso semelhante a Haar de qualquer tamanho pode ser calculado em tempo constante (aproximadamente 60 instruções de microprocessador para um recurso de 2 retângulos).

Outra característica Haar-like pode ser definida como a diferença da soma de pixels de áreas dentro do retângulo, que pode ser em qualquer posição e escala dentro da imagem original. Esse conjunto de características modificadas é chamado de características de 2 retângulos. Viola e Jones também definiram características de 3 retângulos e características de 4 retângulos. Cada tipo de recurso pode indicar a existência (ou ausência) de certos padrões na imagem, como bordas ou alterações na textura. Por exemplo, um recurso de 2 retângulos pode indicar onde a borda está entre uma região escura e uma região clara. A OpenCV já possui o algoritmo pronto para detecção de Haar-like features, contudo, precisamos dos arquivo XML que é a fonte dos padrões para identificação dos objetos. A OpenCV já oferece arquivos prontos que identificam padrões como faces e olhos.

Processo de treinamento:

1- Após obter as imagens, precisamos de realizar um tratamento, para as imagens positivas, passando à escala de cinza, redimensionando (para maior agilidade no processo de detecção), e retornando as coordenadas de onde há o objeto em questão. Para as negativas, somente a última etapa não é feita.

2- Após, começamos o processo de detecção, transpondo as imagens negativas à positivas, para o treinamento, criando o que chamamos de **samples**, ou “exemplos”, em tradução ao português.

3- o terceiro passo é criar um arquivo de vetores com todas as coordenadas das imagens positivas.

4- O treinamento, efetivamente, comparando todos os dados coletados juntamente com a imagem que queremos analisar.

Referências:

CREPALDE – Neylson – Reconhecimento facial com Python e OpenCV - Machine Learning – 2017 – <http://neylsoncrepalde.github.io/2017-02-21-reconhecimento-facial/>– acesso em:12/11/2019.

SILVA - Regis - Gerenciando banco de dados SQLite3 com Python - Parte 1 - 2014 - <http://pythonclub.com.br/gerenciando-banco-dados-sqlite3-python-parte1.html> - acesso em: 12//11/2019.

CARLOS - Juan Ferreira - Reconhecimento\_facial - 2018 - [https://github.com/juanFCA/Reconhecimento-Facial](https://github.com/juanFCA/Reconhecimento-Facial/blob/testes/ReconhecimentoBasico/03_ReconhecimentoFacialCAM.py) - acesso em: 12/11/2019.

ROSEBROCK - Adrian - OpenCV Face Recognition - 2018 - <https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/> - acesso em: 12/11/2019.

ANTONELLO - Ricardo - Introdução a Visão Computacional com Python e OpenCV - 2017 - <http://professor.luzerna.ifc.edu.br/ricardo-antonello/wp-content/uploads/sites/8/2017/02/Livro-Introdu%C3%A7%C3%A3o-a-Vis%C3%A3o-Computacional-com-Python-e-OpenCV-3.pdf> - acesso em: 12/11/2019.

Python Programming - Creating your own Haar Cascade OpenCV Python Tutorial -

<https://pythonprogramming.net/haar-cascade-object-detection-python-opencv-tutorial/> - acesso em: 12/11/2019.