UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO
CURSO DE ENGENHARIA MECATRÔNICA

RECAM: MÓDULO DE GRAVAÇÃO INTELIGENTE PARA CÂMERAS DE SEGURANÇA, BASEADO NA DETECÇÃO DE FACES EM CENA.

JOELSON DE CARVALHO ROCHA JÚNIOR: 20160153946

Natal-RN
2017

JOELSON DE CARVALHO ROCHA JÚNIOR: 20160153946

RECAM: MÓDULO DE GRAVAÇÃO INTELIGENTE PARA CÂMERAS DE SEGURANÇA, BASEADO NA DETECÇÃO DE FACES EM CENA.

Relatório apresentado à disciplina de Processamento Digital de Imagens - DCA0445, correspondente à avaliação da 3ª unidade do semestre 2017.2 do curso de Engenharia de Computação e Automação da Universidade Federal do Rio Grande do Norte, sob orientação do **Prof. Dr. Agostinho de Medeiros Brito Júnior**

Professor:

Agostinho de Medeiros Brito Júnior

Natal-RN
2017

RESUMO

O presente relatório busca demonstrar o desenvolvimento, em software, de um módulo para câmeras de segurança com o objetivo de realizar a gravação de vídeo baseando-se no algorítmo de detecção de faces de Viola-Jones, permitindo assim o uso de dispositivos de alta resolução em sistemas de vigilância. Ao final de seu desenvolvimento, foram obtidos resultados satisfatórios tais como: redução no tamanho do arquivo de gravação gerado, devido a redução do tempo do vídeo final.

Palavras-chave: Gravação de vídeo. Câmeras de segurança. Detecção de faces. Viola-Jones. Resolução.

Lista de Figuras

Sumário

1 INTRODUÇÃO

Sistemas de monitoramento remoto se tornaram cada vez mais comuns em estabelecimentos de toda natureza (comerciais, públicos, residenciais etc.). Existem vários motivos para o uso dessa tecnologia, desde a inibição de tentativas de assalto até ao estudo comportamentamental de indivíduos (animais ou humanos).

Infelizmente, em noticiários de todo o mundo, há vários registros de cenas onde a presença de câmeras não tem surtido efeito na ação criminosa de bandidos. Em muitos casos não há como identificar quais pessoas participaram do ocorrido, devido a limitação encontrada na transmissão de dados da rede utilizada. Por exemplo: para que sejam reproduzidos, simultaneamente e com alta qualidade, vídeos de 6 câmeras de vigilância em uma farmácia, se faz necessário investir em um computador com alta capacidade de processamento e em uma rede com alta taxa de transferência de dados. A solução encontrada por muitas pessoas é diminuir a resolução das câmeras, produzindo imagens como a da Figura 1.

Figura 1: Câmera de circuito interno em baixa resolução.

Nota-se que essa abordagem permite ao usuário ter um registro do que ocorreu a cena, porém, na grande maioria dos casos a identificação do(s) indivíduo(s) em cena se torna impossível.

1.1 Proposta do projeto

Este projeto propõe a elaboração de um módulo para câmeras de segurança que otimize a reprodução, simultânea, de vídeos em alta resolução, permitindo com que o usuário possa identificar, de forma mais rápida, pessoas em cena.

2 REFERENCIAL TEÓRICO

2.1 Reconhecimento de objetos com Viola-Jones

Em 2001, Paul Viola e Michael Jones [1] propuseram uma técnica de detecção de objetos chamada "Rapid Object using a Boosted Cascade of Simple Features", que apresenta uma sugestão voltada a detecção de faces frontais utilizando técnicas que prometem processamento de imagens extremamente rápidos e o alcance de altas taxas de detecção. A técnica foi dividida em 3 partes:

- **Imagem Integral**: conceito que permite que o detector compute caraterística de forma mais rápida;
- **Aprendizado baseado em AdaBoost**: seleciona características mais importantes de um grande conjunto de classificadores;
- **Classificadores em cascata**: aplicam classificadores mais complexos à medida que a imagem analisada passa por estágios classificatórios.

2.1.1 Imagem Integral

Esse procedimento classifica imagens baseando-se em 3 características simples denominadas de "características retangulares", sendo elas de dois, três e quatro retângulos. Elas nada são filtros aplicados à regiões da imagem, conforme ilustradas na Figura 2.

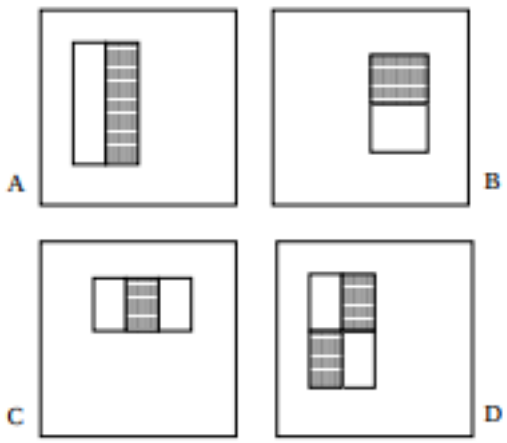

Figura 2: Característica de dois retângulos é mostrada em [A] e [B], de três em [C] e de quatro em [D].

O procedimento para cálculo da imagem integral é feito da seguinte forma: O valor de cada pixel da matriz da imagem integral é equivalente a soma dos valores de todos os pixels da matriz que estão imediatamente acima e à esquerda desse pixel, incluindo seu próprio valor, conforme a Equação 1.

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x',y') \tag{1}$$

De forma mais eficiente, é calculado o valor da imagem integral utilizando os valores já adquiridos da imagem. Dessa forma o método não realiza várias adições. A expressão 2 explicita tal raciocínio, onde $i(x,y)$ representa o pixel da imagem e $ii(x,y)$, o da imagem integral.

$$ii(x,y) = i(x,y) + ii(x-1,y) + ii(x,y-1) - ii(x-1,y-1) \tag{2}$$

2.1.2 Aprendizado baseado em AdaBoost

É utilizado para selecionar pequenos conjuntos de características e treinar os classificadores utilizados para detecção de objetos, aumentando assim a performance de classificação de algorítmos simples de aprendizagem.

De acordo com Rojas(2009)[x], esse algorítmo foi proposto por Yoav Freund e Robert Shapire, em 1995, como um método que é capaz de gerar um forte classificador à partir de um conjunto fraco. O método testa, classifica e atribui pesos à cada classificador, que por sua vez, são combinados, juntamente com seus pesos, e compõem um classificador mais robusto e eficiente.

Os pesos são ajustados a cada iteração, bem como os classificadores, retreinados utilizando uma determinda característica. Feito isso, o erro é calculado para cada classificador, escolhe-se o que possuir menor magnitude de erro e então os pesos são reajustados para se dar início a uma nova iteração. Por fim, o classificador mais forte é escolhido à partir de classificadores mais fracos, e é dados pela combinação linear entre eles e seus respectivos pesos como ilustrado na Equação 3.

$$C_{(m)} = \alpha_1 k_1(x_i) + \alpha_2 k_2(x_i) + ... + \alpha_m k_m(x_i) \tag{3}$$

Onde: $C_{(m)}$ é o conjunto de classificadores, m o número de iterações,α o peso relacionado a cada classificador, k o classificador e x o padrão.

Após utilizar o método, selecionando uma característica dentre as 180.000 possíveis, Viola e Jones relatam em seu trabalho que as duas características retangulares mais fortes acontecem geralmente em duas regiões próximas aos olhos:

- região dos olhos em constraste com as do nariz e bochechas;
- região dos olhos em constraste com a do osso do nariz.

É possível observar o resultado na Figura 3.

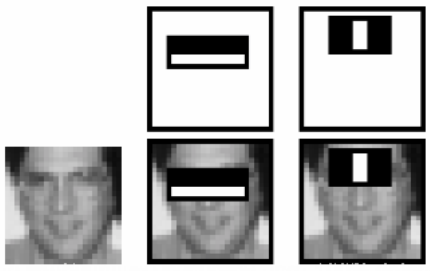

Figura 3: Melhores características selecionadas pelo algorítmo de AdaBoost.

2.1.3 Classificadores em cascata

São utilizados para reduzir a imagem em regiões que contenham o objeto de interesse, aumentando assim a performance do método.

Classificadores mais simples são aplicados à regiões da imagem.Caso a região passe na avaliação, ela é submetida a classificadores mais complexos, caso contrário, ela é descartada. Essa estratégia reduz o número de subjanelas a serem analizadas, e consequentemente o número de ocorrências de falsos positivos.

O treinamento do classificador se dá na forma de troca equivalente, ou seja, os classificadores que possuem mais características possuem maior taxa de detecção.

3 METODOLOGIA

O projeto foi desenvolvido na linguagem Python. Utilizou-se a biblioteca de processamento de imagens OpenCV3.0 [2] , WebCam USB bem como o sistema para controle de versões GIT. Também foi utilizado o treinador para detecção de faces frontais, disponibilizado pelo openCV. Foram feitas consultas no livro do Gonzalez [3] durante a disciplina que auxiliam no entendimento do algorítmo.

O código desenvolvido entitulado de *main.py* segue abaixo na Figura 4.

```
import numpy as np
import cv2
import sys

cascPath = "haarcascade_frontalface_default.xml" #treinamento
faceCascade = cv2.CascadeClassifier(cascPath)

cap = cv2.VideoCapture(1) #configura camera (0) para web cam e/ou (1) para camera usb

fourcc = cv2.VideoWriter_fourcc(*'MJPG') # Define o codec e cria o VideoWriter
out = cv2.VideoWriter('Rec.avi',fourcc, 10.0, (640,480)) #arquivo saida do VideoWriter

while(cap.isOpened()):

    ret, frame = cap.read() #captura frame
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) #converte para tons de cinza

    if ret==True:
        faces = faceCascade.detectMultiScale(
            gray,
            scaleFactor=1.1,
            minNeighbors=5,
            minSize=(30, 30),
        )
        # Desenha retangulo nas faces encontradas
        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            out.write(frame)

        cv2.imshow('video',frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
           # cv2.imwrite('frame.png',frame)
            break
    else:
        break

cap.release()
out.release()
cv2.destroyAllWindows()
```

Figura 4: Código comentado para uso no módulo ReCam.

4 RESULTADOS

O resultado obtido após meia hora de funcionamento do módulo foi um vídeo de 21 segundos onde só aparecem as pessoas em cena, conforme a figura 5.

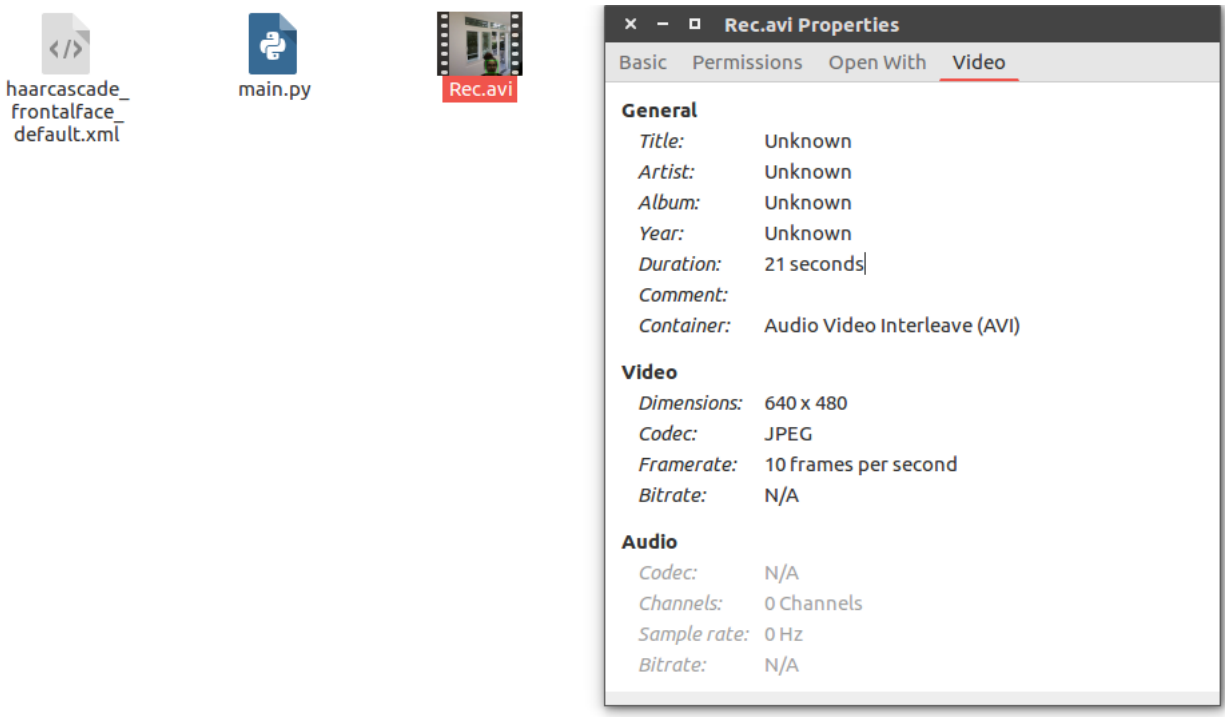

Figura 5: Imagem resultante.

Em suma, foi possível observar que realmente este tipo de abordagem é promissora. Provavéis melhorias serão implementadas para aprimorá-lo, tais como: utilizar o ReCam em um Raspberry Pi e conseguir utilizá-lo com o auxílio do sistema supervisório ZoneMinder.

Referências

[1] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” *Computer Vision and Pattern Recognition*, pp. 1–9, Feb. 2001.

[2] “Opencv documentation,” página na internet, Nov. 2017.

[3] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. 3 ed., 2009.