Trabalho 5 - Reconhecimento de Objetos e Padrões

Rauan Pires - 14103318

24 de outubro de 2018

1 Introdução

Este trabalho teve como objetivo desenvolver algortimos, utilizando a linguagem de programação C++ e a biblioteca OpenCV, para reconhecimento de objetos e padrões preestabelecidos. O trabalho foi dividio em duas etapas: A primeira foi detectar, em um vídeo ou webcam, três círculos e desenhar o triângulo formado pelos seus centros de massa, além de calcular e apresentar o tamanho de cada aresta e a area total do triângulo. Na segunda etapa, o objetivo foi identificar, em um vídeo ou webcam, algumas figuras preestabelecidas, desenhar um retângulo para indicar onde aparecem na tela e calcular uma função de correlação para estimar o quanto a imagem identificada se parece com a imagem que estava sendo buscada.

2 Primeira etapa - detecção de círculos

Levando em consideração o alto ruído de imagens de webcam, especialmente em ambiente escuros, o primeiro passo foi realizar algumas operações de abertura e fechamento da imagem, tornando-a assim mais homogênea e diminuido serrilhados. Sabendo que as imagens propostas para serem identificadas possuiam alto contraste em relação ao seu fundo(círculos pretos em uma folha branca), o passo seguinte foi estimar um threshold e binarizar o frame sendo processado. A imagem abaixo mostra um exemplo de frame suavizado e, posteriormente binarizado.

Figura 1: Amostra de frame binarizado

A partir da imagem binarizada obtida, a função "cv::HoughCircles"disponibilizada pela biblioteca OpenCV foi utilizada para identificar possíveis círculos na imagem sendo processada. A função HoughCircles retorna 3 valores para cada círculo encontrado: coordenada do centro x, coordenada do centro y e o raio. Baseado nesses três valores o círculo é desenhado no frame através da função "cv::circle".

Como o algoritmo já tem os valores dos centros dos círculos, basta ligá-los utilizando a função "cv::line", que desenha uma linha reta entre dois pontos. Para calcu

a area do triângulo formado pelas três arestas, a fórmula de Heron foi utilizada.

$$Area = \sqrt{(p * ((p-a) * (p-b) * (p-c)))};$$

Onde P = perímetro do triângulo, e 'a', 'b', e 'c' representam os lados. A imagem resultante obtida pelo algoritmo pode ser visualizada abaixo:

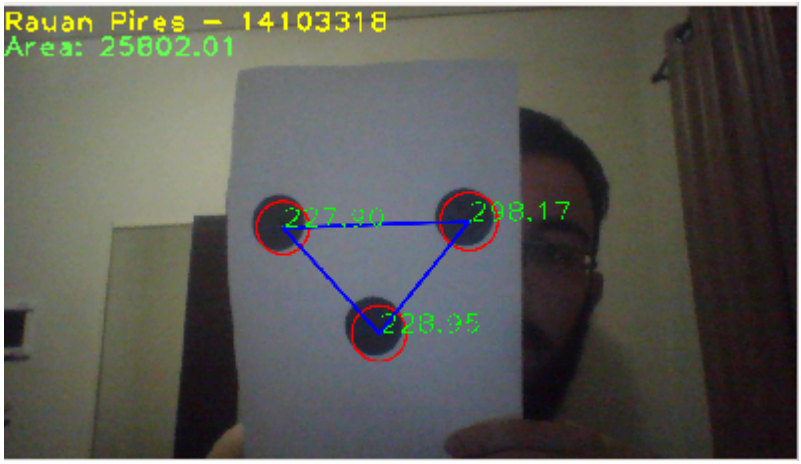

Figura 2: Amostra de frame processado

3 Segunda etapa - detecção de figuras

Para esta segunda etapa, o objetivo é detectar e localizar em vídeo/webcam algumas imagens previamente definidas. Para testar o algoritmo as seguintes imagens foram escolhidas:

Figura 3: Letra A

Figura 4: Letra B

Figura 5: Circulo

Figura 6: Quadrado

Figura 7: Triângulo

Como o triângulo difere bastante quando rotacionado, outras imagens de apoio foram utilizadas para essa forma.

Figura 8: Triângulo

Figura 9: Triângulo

Figura 10: Triângulo

A técnica para extração das das figuras de cada frame foi a seguinte:

1. Converter o frame para tons de cinza
2. Aplicar "medianBlur"para suavizar a imagem
3. Operações morfológicas para retirar serrilhados
4. Binarizar o frame
5. Encontrar contornos na figura
6. Para cada contorno de área maior que um determinado threshold:

(a) Utilizando a função "minAreaRect"extrair o mínimo retângulo que contenha o contorno. Útil para essa aplicação pois as letras em geral tem um tamanho maior na vertical, possibilitando ao programa recortar as imagens na direção correta, mesmo quando rotacionadas.

(b) Rotacionar o retângulo mínimo de cada figura para que fique orientado verticalmente

(c) Recortar cada retângulo mínimo

(d) Redimensionar a imagem extraída para comparação com as imagens bases que estão sendo buscadas

(e) Comparar cada imagem extraída com as imagens bases através de subtração simples

(f) Estimar um valor de fitness de 0 a 1, onde 0 significa que a imagem é totalmente oposta à comparada e 1 significa que a imagem é a mesma à comparada. A função pode ser calculada pela função da reta, onde o número de pixels errados máximo é dado por $100 * 100 * 255 = 2550000$ onde 100 é o tamanho do lado de cada imagem extraída após o redimensionamento e 255 é a intensidade máxima de cada píxel.

(g) Para a imagem correspondente de maior fitness, desenhar um retângulo que envolvendo o contorno no frame, e escrever seu nome.

A imagem de threshold da qual são extraídos os contornos e a imagem final do frame podem ser visualizadas abaixo:

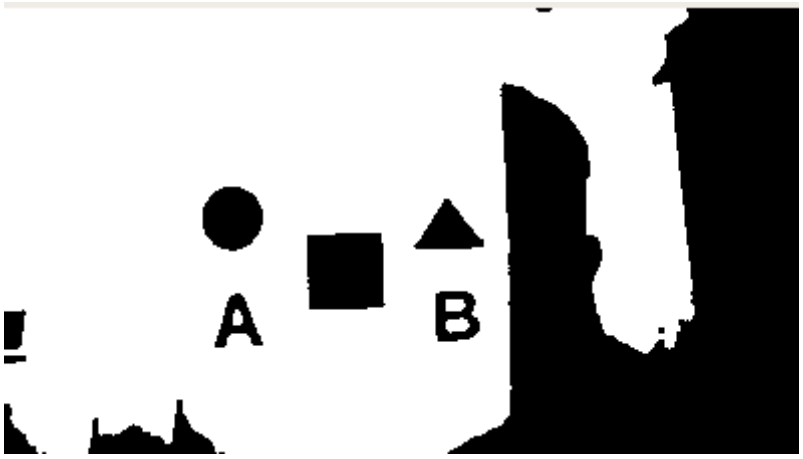

Figura 11: Amostra de frame binarizado

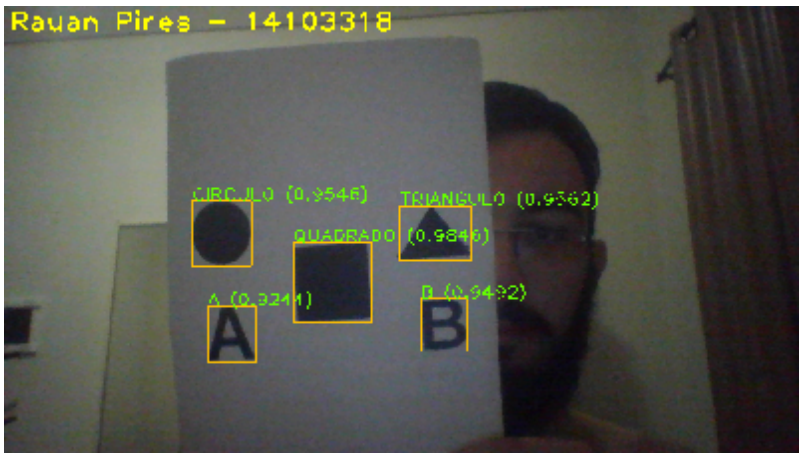

Figura 12: Amostra de frame processado

4 Conclusão

Foi possível confirmar, após a conclusão do trabalho, que mesmo utilizando funções relativamente simples da biblioteca OpenCV, foi possível detectar e identificar, com precisão suficientemente boa, as diferentes imagens propostas, sem a necessidade de implementação de técnicas mais complexas, como redes neurais, por exemplo.

Entre as dificuldades encontradas pode ser citada a baixa qualidade de imagem da webcam utilizada, o que fez necessário o uso de diversas operações morfológicas para suavizar o frame, diminuido assim a performance do algoritmo.

* Vídeos dos testes podem ser vistos em:
https://youtu.be/fkhkoOKclJc
https://youtu.be/FIPUPbpuzZY