Projeto n° 3
Cálculo da curvatura ao longo do contorno paramétrico

Felipe Tavoni
Gabriel Rodrigues Malaquias
Lucas Cruz do Reis
Renan Bobadilla Morelli

Universidade Federal de São Carlos

21 Junho, 2021

1 Introdução

Com a extração da curva paramétrica de um objeto em uma imagem, podemos definir o quão curvo são os pontos ao longo do contorno. Segundo [1], a curvatura pode ser definida como a taxa de mudança da inclinação. Uma maneira de calcular tal variação pode ser feita através da aplicação da fórmula geral da curvatura, descrita a seguir:

$$k(t) = \frac{x'(t) * y''(t) - x''(t) * y'(y)}{(x'(t)^2 + y'(t)^2)^{3/2}}$$

2 Motivação do uso do método

A partir do cálculo dessa curvatura, mencionado em [1], podemos determinar o quanto o ponto se curva diante de uma reta. Ainda segundo [1], podemos ter valores positivos ou negativos à medida que se percorre a fronteira no sentido horário, no qual valores positivos indicam que o ponto faz parte de um segmento *convexo*, enquanto o oposto indica *concavidade*. Isso permite uma análise do contorno, podendo estimar o grau do mesmo, categorizando-o, por exemplo, como um ponto de canto o um ponto pertencente a uma reta.

3 Implementação do método

O cálculo da curvatura ao longo da reta foi cálculado usando a linguagem de programação Python, com o auxílio da biblioteca numpy. Segundo a função descrita em Listing 1, o cálculo se inicia com a operação de derivada sobre os pontos x e y. Esse cálculo é facilitado pelo uso da função *gradient* provida pelo numpy, bem similar ao uso da função *diff*. Assim, tem-se em x_t e y_t a primeira derivada nos pontos x e y. A seguir, repete-se o processo para então se obter a segunda derivada nos eixos, armazenando-as em xx_t e yy_t. Por fim, implementa-se a fórmula que calcula então a curvatura da reta.

Listing 1: Cálculo da curvatura

```
def curvature(cont):
    '''
    Calculo da curvatura ao longo do contorno de objetos.
    '''
    # Suavizando o contorno segundo uma funcao gaussiana
    x = gaussian_suavization_1d(cont[:, 0], 6)
    y = gaussian_suavization_1d(cont[:, 1], 6)

    # Calculo da primeira derivada no eixo x e y.
    dx = np.gradient(x)
    dy = np.gradient(y)

    # Calculo da segunda derivada no eixo x e y.
    ddx = np.gradient(dx)
    ddy = np.gradient(dy)

    # Por fim, o calculo da curvatura
    curvature = ((dx * ddy) - (ddx * dy)) / ((dx**2 + dy**2)**1.5)

    t = range(0, len(curvature))
    plt.plot(t, curvature_val)
    plt.show()

    return curvature_val
```

Dado a alta taxa de variação que o contorno pode apresentar, somado ao cálculo da curvatura que envolve derivadas, as variações são enaltecidas, tornando a interpretação dos valores resultantes difícil de ser compreendida. Dessa forma, é realizada uma suavização gaussiana sobre o sinal do contorno, para então realizar a operação desejada.

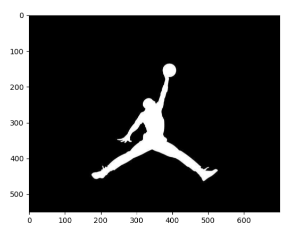
Listing 2: Suavização gaussiana

```
def gaussian_suavization_1d(signal, filter_size):
    sigma = filter_size/6.
    x = np.linspace(-3*sigma, 3*sigma, filter_size)
    y = np.exp(-x**2/(2*sigma**2))

    # Filtros de suavizacao precisam ter soma igual a 1
```

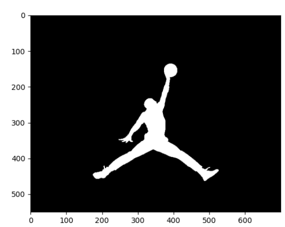
```
y = y/np.sum(y)

smoothed_signal = np.convolve(signal, y, 'same')

return smoothed_signal
```

4 Resultados

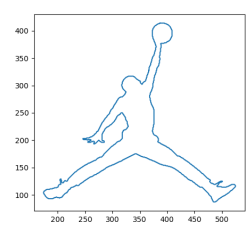
Por fim, um teste foi executado pela aplicação discutida. Uma imagem com um único objeto foi inicialmente limiarizada, mantendo-a com valores unitários, 0 e 1. A seguir, após a extração do contorno paramétrico, realizamos o cálculo da curvatura em cada ponto e então obtemos o resultado disposto em 1. Ademais, acredita-se que os altos valores no final estão diretamente associados ao fato de a segunda derivada retirar 2 valores do sinal original.



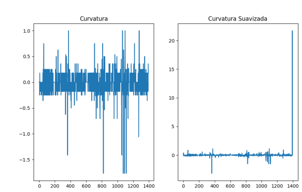
(a) Imagem original



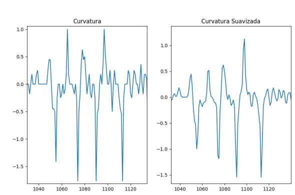
(b) Imagem limiarizada com Otsu



(c) Contorno paramétrico



(d) Cáclulo da curvatura



(e) Ampliação da curvatura

Figure 1: Resultados

Bibliografia

[1] Richard E. Woods Rafael C. Gonzales. *Processamento Digital de Imagens.* Pearson Universidades; 3ª edição, 2009. ISBN: 8576054019.