Toonify:

[*]a project in image processing

1st Ariel Vieira Lima Serafim *UNIVERSITY OF BRASÍLIA*
DEPARTMENT OF COMPUTER SCIENCE
Brasília, Brazil
arielserafim@gmail.com
2nd Luigi Minardi Ferreira Maia *UNIVERSITY OF BRASÍLIA*
DEPARTMENT OF COMPUTER SCIENCE
Brasília, Brazil
luigimaia99@gmail.com

***Resumo*—This project is an attempt to reproduce and improve the algorithm proposed by Kevin Dade in his article [1] *Toonify: Cartoon Photo Effect Application*, that seeks to emulate the types of cel-sheding**

I. Introdução

O efeito toon é muito conhecido, trata-se de tornar objetos, seres e paisagens reais mais semelhantes à desenhos do estilo cartoon, para isso é preciso investir em alguma forma de simplificar as imagens.
A forma de simplificação utilizada aqui é uma adaptação do processo de Cel shading [2] para fotos.

II. Referências e trabalhos semelhantes

A base para o trabalho foi o artigo de Kevin Dade, Toonify: Cartoon Photo Effect Application [1].

III. Soluções propostas

Como objetivo do processo de cartoonificação das images é simplificar tanto as bordas quanto as cores da imagem, o processo é feito em três partes: processamento das bordas da imagem, processamento das cores da imagem e, no final, junta-se o resultado dos dois processos anteriores em apenas uma imagem.

Como o processo de adquirir as bordas da imagem e o de adquirir as cores são mutualmente independes, essas duas partes são feitas em paralelo a fim de economizar tempo de execução.

A. Adquirir uma imagem contendo as bordas

1) Filtro mediana: O primeiro processamento feito sobre a imagem é um filtro de mediana com uma matriz 7x7 como kernel, esse passo é importante para diminuir possíveis ruídos salt and pepper e também para diminuir o número de falsas bordas que poderiam ser detectadas caso a imagem estivesse detalhada demais.Vale lembrar que o tamanho do kernel garante a manutenção de detalhes relevantes.

2) Detector de bordas Canny: Esta é a etapa principal da parte de detecção de bordas. Outros algoritmos poderiam ter sido utilizados, como o algoritmo laplaciano, entretanto outras operações teriam de ser feitas sobre o resultado pois o laplaciano não garante que as bordas tenham largura 1 e isso gera efeitos indesejáveis no resultado final, além de dificultar o uso de operações morfológicas descritas a seguir.

3) Operações morfologicas: Obtidas as bordas da imagem, é necessário ainda que sejam feitos processamentos morfologicos para melhorar o efeito das bordas na imagem final. A operação escolhida aqui é a dilatação, esse processo realça as bordas e tende a juntar regiões, o que é importante para a eficiência do passo seguinte.

4) Filtrar bordas: Nesta etapa buscamos remover bordas muito pequenas, que geram um efeito muito semelhante ao ruído salt and pepper. Para remover esse "ruído"interpretamos as bordas como regiões conectadas e removemos as que possuirem uma área menor que um limite determinado.

B. Filtrar as cores da imagem

C. Adquirir o grau de "colorido"da imagem

Como cada imagem possui um aspecto único, filtrar diferentes imagens, sempre com os mesmos parâmetros pode significar priorizar o processamento de algumas imagens em relação a outras. O modo encontrado de diferenciar as imagens foi encontrando o seu grau de colorido por meio do processo descrito em *Measuring colourfulness in natural images* [3].

1) Filtrar bilateralmente: A partir da imagem de origem, utiliza-se um processo conhecido por borrar as cores da imagem, porém mantendo as bordas bem definidas. Como, descrito no artigo *A Gentle Introduction to Bilateral Filtering and its Applications* [4]. Como o filtro tem um grau de complexidade grande, opta-se por diminuir a imagem à metade e aplicar o filtro iterativamente com um kernel não tão grande de tamanho 9 por 9. O filtro possui duas entradas que são as variâncias σ_r^2 utilizadas pela gaussiana para cor e σ_s^2 que é a variância utilizada pelo filtro para distância entre os pixels. Além das entradas do filtro, é necessário escolher I que é quantas vezes o filtro será utilizado iterativamente. Com alguns testes, decidiu-se escolher as essas constantes da seguinte forma:

$$\sigma_r^2 = \frac{685}{C}$$

$$\sigma_s^2 = 5$$

$$I = 10 + \frac{C}{10}$$

em que C é o grau de colorido da imagem. No final, volta-se a imagem para o tamanho de origem.

2) Filtrar com a mediana: Neste estágio, deve-se utilizar o filtro da mediana para olcutar qualquer erro que tenha ocorrido durante o aumento da imagem para o tamanho original. Utilizou-se um kernel de tamaho 7 por 7.

3) Requantificar as cores: Para que o número de cores fosse reduzido, realiza-se sobre cada canal dos pixels uma operação de requantização que em dado pela seguinte fórmula:

$$p_{new} = \left\lfloor \frac{p}{a} \right\rfloor \cdot a$$

em que p é o valor do canal do pixel e a é um fator de pelo qual o número de cores possíveis será diminuido. Porém, a aplicação desse efeito tem como aspecto negativo escurecer as imagens, então, optou-se pelo uso da seguinte fómula:

$$p_{new} = \left\lfloor \frac{p}{a} \right\rfloor \cdot a + 255 - \left\lfloor \frac{255}{a} \right\rfloor \cdot a$$

em que 255 é o valor máximo das cores em cada canal de cores. Já o valor de a é escolhido a partir da seguinte fórmula:

$$a = 10 + \frac{C}{5}$$

em que novamente C é o grau de "colorido"da imagem.

IV. •

V. Resultados experimentais

VI. Conclusões

Referências

[1] Kevin Dade:
Toonify: Cartoon Photo Effect Application.

[2] Cel shading
https://en.wikipedia.org/wiki/Celshading

[3] David Hasler a and Sabine Süsstrunk
Measuring colourfulness in natural images

[4] Sylvain Paris[1], Pierre Kornprobst[2], Jack Tumblin[3] and Frédo Durand[4]
A Gentle Introduction to Bilateral Filtering and its Applications