Trabalho Final de Introdução ao Processamento de Imagens: Toonify

Guilherme Coelho[1] e Vítor Ribas Bandeira[2]

***Abstract*— Este trabalho tem como objetivo alcançar resultados satisfatórios ao aplicar efeito *cartoon* em imagens comuns.**

I. INTRODUÇÃO

Esse projeto tem como objetivo criar um algoritmo para simular um efeito de cartoon em imagens realistas. É usado como base do nosso projeto um tipo de efeito chamado *cel-shading*, que consiste na renderização de imagens 3D para que fiquem parecidas com imagens 2D. Essa técnica pode ser observada em muitos filmes e jogos. Porém nosso projeto consiste em processar fotos 2D realistas para que tenham efeito cartoon. O programa tem como basem algumas ideias do paper de Kevin Dade, citado na sessão de *Referências*, sobre o efeito Toonify para plataformas mobile.

II. BACKGROUND AND RELATED WORK

Para conseguir o efeito cartoon, foi observado que as cores e as bordas são duas frentes que precisam ser bem trabalhadas. No paper do Kevin, foi observado a análise dessas duas frentes e nós decidimos seguir o mesmo caminho. Porém é possível que mais frentes possam ser analisadas para que melhore mais ainda o efeito desejado. Em uma imagem cartoon é importantíssimo o ajuste de cores. Imagens cartoon têm a caracterísica marcante de simplicidade de cores e algumas sobras, dando aspecto de profundidade. Tal ajuste pode ser feito usando técnicas de *Mean Shift Filter* ou de Filtragem Bilateral. Kevin propõem a utilização da Filtragem Bilateral, então tal método foi adotado. Além do ajuste de cores, o processamento de bordas é fundamental para contribuir na aparência cartoon na imagem, destacando o novo aspecto das cores resultantes do tratamento de cor

III. SOLUÇÃO PROPOSTA

Uma forma de implementar o efeito cartoon desejado é trabalhar em duas vertentes: tratamento de bordas e tratamento de cores.

A. Tratamento de bordas

A etapa de tratamento de bordas consiste nos seguintes procedimentos:

1) *Converter para grayscale*: Consiste em converter as imagens originais que estão em RGB para níveis de cinza. Ao converter, a imagem possui 3 canais (YCbCr), mas é apenas utilizado a componente Y (Luminance) para o tratamento das bordas. Essa etapa é importante porque o processo de detecção de bordas depende da variação da intensidade luminosa da imagem. Portanto, as componentes de cores são dispensáveis. Além disso, como cada pixel da imagem possui apenas um componente, facilita a sua manipulação.
2) *Median Filter*: Após a conversão, é aplicado um filtro passa-baixas para remover ruídos *Salt and Pepper* da imagem. Nessa etapa foi utilizada uma matriz 7x7 para fazer a filtragem da imagem. Foi percebido que quanto maior a matriz, mais borrada fica a imagem e mais difícil é para detectar as bordas. Portanto, chegamos na conclusão que a matriz 7x7 realizaria o trabalho de tirar os ruídos sem prejudicar as bordas.
3) *Função Canny*: Depois de filtrada a imagem, é aplicado um filtro passa-altas para detectar bordas na imagem. Dentre os filtros disponíveis (*Laplace, Canny, Difference of Gaussian*) foi escolhido o *Canny* pois as bordas resultantes possuem largura de apenas um pixel, facilitando os processos morfológicos que virão em seguida.
4) *Operações morfológicas*: Com a imagem contendo apenas bordas, é aplicado uma estrutura morfológica para dilatar as bordas com o objetivo de salientar e suavizar mais seus contornos. Para fazer a dilatação foi utilizado um quadrado de tamanho 2x2.
5) *Filtro de bordas*: Essa etapa tem como objetivo filtrar as bordas contidas na imagem que são indesejadas para o resultado final. Para fazer isso foi utilizado uma função que detecta componentes na imagem (conjunto de pixels conectados). Depois que são detectados, a função possui um filtro que estabelece um limite para o tamanho mínimo de cada componente na imagem. Todos os componentes que possuem um tamanho menor que o limite são desprezados na imagem.

B. Tratamento de cores

A etapa de tratamento de cores consiste basicamente em utilizar uma filtragem bilateral para suavizar zonas de uma imagem e quantizar os níveis de cores da imagem.

1) *Filtro Bilateral*: O filtro Bilateral tem a finalidade de suavizar areas da imagem preservando bordas ou transições bruscas de níveis de cor.
2) *Filtro Mediana*: Também é utilizado nessa etapa do tratamento de cores com duas finalidades distintas. Algumas imagens possuem ruído do tipo *Salt And Pepper*, então utilizamos o filtro para remover esse ruído presente. O outro propósito é homogeneizar alguns pixels vizinhos heterogêneos após a quantização das cores. O tamanho do *kernel* utilizado foi de 7x7.

[1]Guilherme Coelho, graduando de Engenharia de Computação em Universidade de Brasília, Distrito Federal, Brasil. Matrícula 160123046.

[2]Vítor Ribas Bandeira, graduando de Engenharia de Computação em Universidade de Brasília, Distrito Federal, Brasil. Matrícula 160148421.

3) *Quantização de Cores*: É uma etapa de grande peso para o efeito cartoon. Aqui, os 256 níveis de cores são reduzidos, o que cria faixas de cor para determinados níveis originais. O fator de quantização escolhido foi de 16 níveis de cor, pois se mostrou como sendo um fator com resultado satisfatório.

C. Recombinar

Com a etapa de processamento de bordas e cores terminada, resta apenas juntar os dois resultados para ter o resultado final. Para isso a gente optou pelo uso de uma função que multiplica a matriz da imagem de bordas com a matriz da imagem com as cores tratadas. Como a função canny retorna bordas de cor branca, foi preciso encontrar uma maneira de deixar as bordas pretas. Para isso a gente usou o complemento da imagem de borda, assim, as bordas ficam pretas e o fundo fica branco.

IV. RESULTADOS EXPERIMENTAIS

Aqui apresentamos de fato os resultados obtidos comparados às figuras originais:

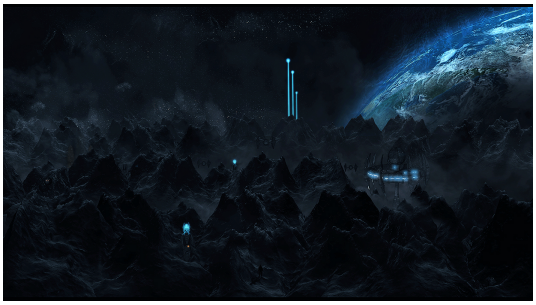

Fig. 1. Imagem original 1

Fig. 2. Imagem cartoon 1

V. CONCLUSÃO

O algoritmo proposto possui baixa complexidade, porém conseguiu cartoonizar imagens com grau interessante de qualidade. Como visto nas imagens resultados, o algoritmo falha bastante na detecção de bordas em expressões faciais devido a enorme variação de intensidade luminosa em um rosto. Além disso foi observado que nosso algoritmo produz cartoons melhores utilizando imagens de alta resolução ao invés das de baixa resolução. Comparando esse projeto com o do Kevin, é importante destacar que foi preferível usar um

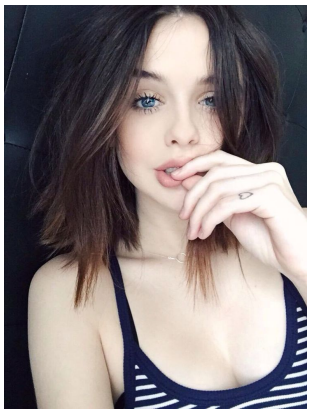

Fig. 3. Imagem original 2

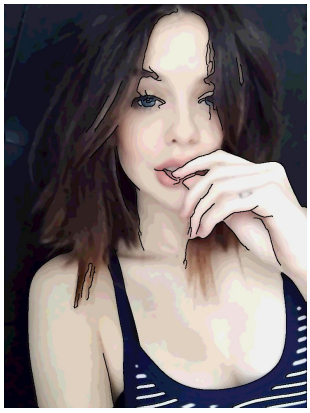

Fig. 4. Imagem cartoon 2

Fig. 5. Imagem original 3

Fig. 6. Imagem cartoon 3

algoritmo que despreze mais as pequenas bordas, na tentativa de melhorar o efeito cartoon. Porém, ao analisar os resultados das imagens de pessoas, acredita-se que foi conseguido um resultado bem positivo considerando que Kevin também teve muitos problemas nesse quesito.

REFERENCES

[1] Dade, Kevin, Toonify: cartoon photo effect aplicattion. Stanford, EUA: Department of Eletrical Engineering, Stanford University.

[2] M.Nagao and T. Matsuyama. Edge preserving smoothing. CGIP, 9:294-407, 1979.

[3] Calonder, Michael, et al. Brief: Binary robust independent elementary features. Computer Vision-ECCV 2010. Springer Berlin Heildelberg, 2010.