Fundamentos de Processamento de Imagens - Trabalho Prático III

Vinícius Daniel

November 27, 2023

Contents

1 Introdução

O presente trabalho possui como finalidade o manuseio de vídeos em tempo real mediante o emprego da biblioteca OpenCV na linguagem de programação C++. Para tanto, implementou-se uma interface que lê a câmera do usuário e permite a manipulação da imagem por meio de interações via teclado.
Espera-se ao fim deste projeto que se compreenda a semelhança entre as operações sobre imagem e sobre vídeo. Ademais, é esperado que se tenha uma noção de como a biblioteca OpenCV opera sobre dados de entrada e saída, bem como a forma de manipulação de imagens e vídeos.

2 Desenvolvimento

1. O programa foi construído em C++ utilizando a IDE *CLion Nova*. Para a administração de bibliotecas, utilizou-se o mecanismo *CMake*. A adição da biblioteca ao projeto foi executada sem muitos problemas ao seguir o passo a passo a seguir.
   Vislumbre o fluxo geral da aplicação aqui.
   Para os vídeos a seguir, recomenda-se definir a velocidade para $0.5\times$.

2. A utilização do comando em si é bastante trivial, o que tornou o processo bastante simples. A única complicação, no entanto, foi que as dimensões do *kernel* precisam ser ímpares, então foi necessário implementar um mecanismo que arredonda os valores pares da *TrackBar*.
   Veja o resultado aqui.

3. Após a resolução dos impasses causados pela etapa anterior, a implementação desta foi extremamente simples, tendo apenas que chamar a função predefinida com os valores de *threshold* padrão que a documentação sugere $(100, 200)$.
   Veja o resultado aqui.

4. A aplicação do Sobel não foi imediata, dado que é necessário separar nos valores de x e y. No entanto, após seguir a documentação a seguir, tornou-se extremamente óbvia a implementação.
   Veja o resultado aqui.

5. De longe, uma das etapas mais simples, dado que a própria função **convertTo** já oferece os parâmetros α e β, precisando somente chamá-la com os valores adequados.
   Veja o resultado aqui.

6. Tal operação é trivial, basta usar a função **cvtColor(source, dst, COLOR_RGB2GRAY)**, a qual serve para converter entre diversas representações de cores.
   Veja o resultado aqui.

7. Similarmente, a trivialidade de tal operação mostrou-se imediata, dado que existe a função **resize**, a qual pode ser usada da seguinte maneira para reduzir a imagem pela metade:

resize(source, dst, Size(), 1.0 / 2, 1.0 / 2).

8. Mais obviamente do que as etapas precedentes, as funções de rotação são
   rotate(source, dst, ROTATE_90_CLOCKWISE)
   e
   rotate(source, dst, ROTATE_90_COUNTERCLOCKWISE), o que torna a implementação imediata.

9. O espelhamento dá-se mediante o emprego da função **flip(source, dst, CODE)**, onde **CODE** pode ser **1** para espelhamento horizontal, **0** para espelhamento vertical e **-1** para ambos. Dessa forma, a implementação é trivial.
   Veja-o aqui.

10. Ao consultar a documentação, constatou-se que a função
    VideoWriter::open("output.avi", VideoWriter::fourcc('M', 'J', 'P', 'G'), 60, Size(processed.cols, processed.rows))
    pode ser usada para abrir um arquivo de saída, e a função **VideoWriter::write(frame)** pode ser usada para escrever um *frame* no arquivo. Ademais, vale-se ressaltar que o *frame* deve ser convertido para o formato de cores (3 canais) antes de ser escrito no arquivo, mediante **cvtColor(source, dst, COLOR_GRAY2RGB)**.

3 Conclusão

Infere-se, destarte, a simplicidade de manipular imagens e vídeos com a biblioteca OpenCV, assim como a geral semelhança entre as operações sobre imagem e sobre vídeo. Dessa forma, pode-se afirmar que o objetivo deste trabalho foi alcançado.