

**Processamento de Vídeo em Tempo Real**

Total de Pontos do Trabalho: 200

**OBJETIVO**

O objetivo deste trabalho é **familiarizar os estudantes com processamento de vídeo em tempo real** utilizando a biblioteca OpenCV.

Ao completá-lo, o(a) estudante terá compreendido como **realizar em tempo real**:

- **Captura e exibição** de vídeo a partir de câmeras conectadas a um PC ou laptop;
- **Processamento de vídeo** obtendo os seguintes efeitos que já foram aplicados a imagens (Trabalhos Práticos 1 e 2): **filtragem Gaussiana, detecção de arestas, estimativa de gradiente, conversão para tons de cinza, ajuste de brilho e de contraste, e cálculo de negativo**;
- **Redimensionamento de vídeo**;
- **Rotação e espelhamento de vídeo**;
- **Gravação do vídeo processado em um arquivo**.

**Descrição do Trabalho**

Baixe e instale a biblioteca OpenCV, que pode ser obtida a partir do endereço <https://sourceforge.net/projects/opencvlibrary/>. OpenCV está disponível em versões para Windows, Linux e Mac, à sua escolha. Este endereço também contém links para documentação e instruções de instalação. Em <http://opencv.org/> você encontrará farta documentação e tutoriais.

A descrição de algumas das tarefas abaixo referencia comandos em C++. Caso você prefira utilizar outra linguagem, como C ou Python, também suportadas por OpenCV, deverá identificar o comando corresponde na respectiva linguagem.

- 1) **(20 pontos) Configure o seu ambiente de programação** (e.g., Visual Studio, etc.) **compile e execute o programa básico disponibilizado no Apêndice A** deste documento. Este programa simples lhe permitirá capturar e exibir vídeos em tempo real, provendo a estrutura sobre a qual você implementará as tarefas solicitadas.

**Com cada quadro do vídeo capturado pela câmera, realize as operações abaixo e exiba o frame resultante em uma janela ao lado da original.** Para utilização dos comandos mencionados para completar a tarefa, pode ser necessário algum tipo de pre-processamento aplicado ao quadro em questão. Neste caso, é parte da tarefa a identificação e aplicação de tal pre-processamento.

- 2) **(20 pontos)** Utilize o comando **GaussianBlur** para aplicar borrimento ao vídeo. Utilize um **Trackbar** para definir o tamanho do kernel Gaussiano.
- 3) **(20 pontos)** Utilize o comando **Canny** para detectar as arestas no vídeo.
- 4) **(20 pontos)** Utilize o comando **Sobel** para obter uma estimativa do gradiente do vídeo.

- 5) **(20 pontos)** Utilize o comando **convertTo** para realizar ajuste de brilho, ajuste de contraste, e obter o negativo do vídeo.

A esta altura, você já deve encontrar-se minimamente familiarizado com a documentação de OpenCV. Para os itens abaixo, identifique os comandos que serão necessários para a realização das operações solicitadas e aplique-os ao vídeo, como feito nos itens (2) a (5).

- 6) **(10 pontos)** Conversão de cores (RGB) para **tons de cinza** (grayscale).
- 7) **(20 pontos)** **Redimensionamento do vídeo** para a metade do número de pixels em cada uma de suas dimensões.
- 8) **(20 pontos)** **Rotação do vídeo de 90 graus**.
- 9) **(20 pontos)** **Espelhamento do vídeo (horizontal e vertical)**.
- 10) **(30 pontos)** **Gravação de vídeo**, levando em conta todos os efeitos acima, **exceto Rotação e Redimensionamento**, visto que estas operações alteram as dimensões originais do frame, o que tenderia a ocasionar um erro durante a tentativa de gravação.

## Apêndice A: Programa básico para captura e exibição de vídeo em tempo real

```
#include <opencv2/opencv.hpp>
using namespace cv;

int main(int argc, char** argv)
{
    int camera = 0;
    VideoCapture cap;
    // open the default camera, use something different from 0 otherwise;
    // Check VideoCapture documentation.
    if(!cap.open(camera))
        return 0;
    for(;;)
    {
        Mat frame;
        cap >> frame;
        if( frame.empty() ) break; // end of video stream
        imshow("This is you, smile! :)", frame);
        if( waitKey(1) == 27 ) break; // stop capturing by pressing ESC
    }
    cap.release(); // release the VideoCapture object
    return 0;
}
```