UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL - UFRGS
CIÊNCIA DA COMPUTAÇÃO

# *TRABALHO 3: PROCESSAMENTO DE VÍDEO EM TEMPO REAL*

INF01046 – FUNDAMENTOS DE PROCESSAMENTO DE IMAGENS

Marcelo Basso
00579239

Porto Alegre - RS
2024/02

# SUMÁRIO

# INTRODUÇÃO

No presente trabalho, implementaram-se operações para tratamento de vídeo utilizando a biblioteca OpenCV em ambiente Linux, com a linguagem C++. Dentre essas operações estão:

1. Borramento Gaussiano,
2. Utilização do comando *Canny* para detectar arestas,
3. Utilização do filtro *Sobel* para detectar arestas,
4. Ajustes de brilho, contraste e cálculo do negativo,
5. Conversão para escala de cinza,
6. Redimensionamento da imagem,
7. Rotação em sentido horário,
8. Espelhamento (vertical e horizontal),
9. Gravação e salvamento de vídeo em disco.

A partir dessas implementações, pôde-se desenvolver uma maior compreensão sobre o processamento de vídeos e suas proximidades com o processamento de imagens.

Todos os efeitos são combináveis. Eles podem ser ligados - apertando sua respectiva opção - e desligados com um segundo clique. Além disse, ao clicar espaço, todos os efeitos são removidos e a imagem volta à sua forma original.

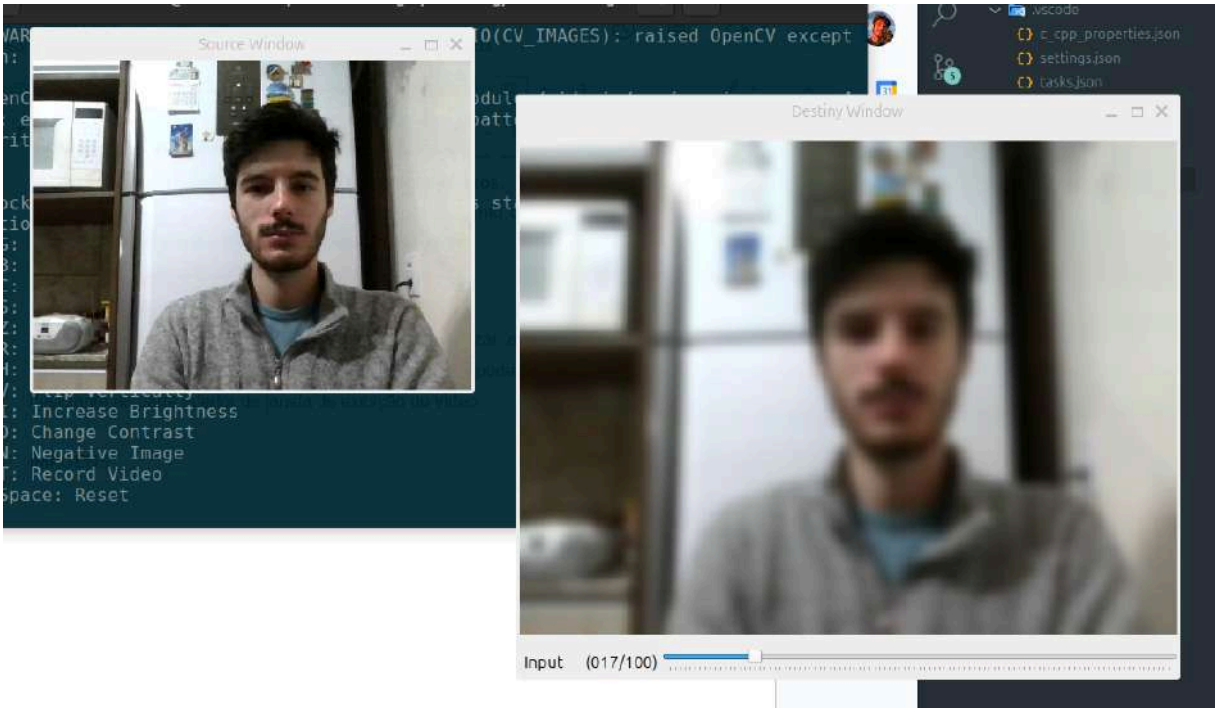
# OPERAÇÕES E ETAPAS

## Configuração do ambiente

Antes de iniciar as implementações, foi necessário baixar o código-fonte da biblioteca OpenCV, compilá-lo e configurar o ambiente de desenvolvimento no ambiente Linux escolhido. Após realizar esses passos, e utilizando algumas ferramentas para ajudar na compilação e desenvolvimento do código, iniciou-se a implementação das demais funções.

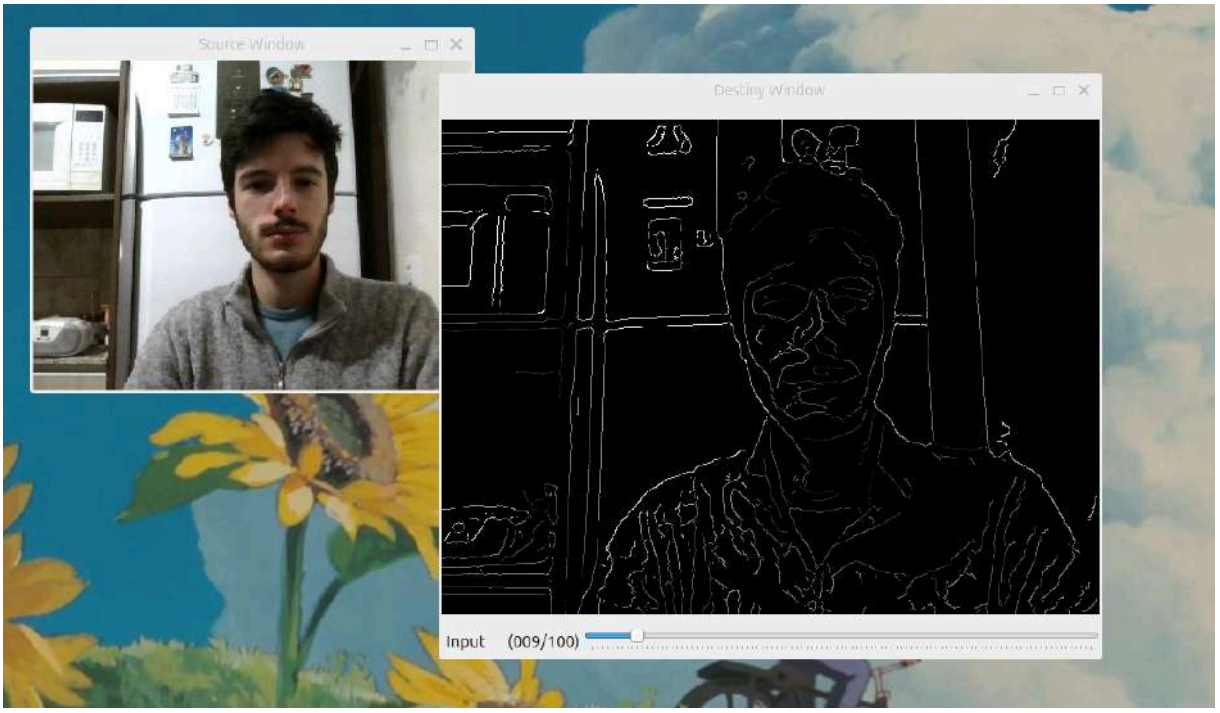
## Borramento Gaussiano

Utilizou-se a função `GaussianBlur()` para realizar a convolução de cada *frame* com o kernel Gaussiano. O tamanho do kernel pode ser definido no input deslizante na parte inferior da janela de exibição do vídeo.



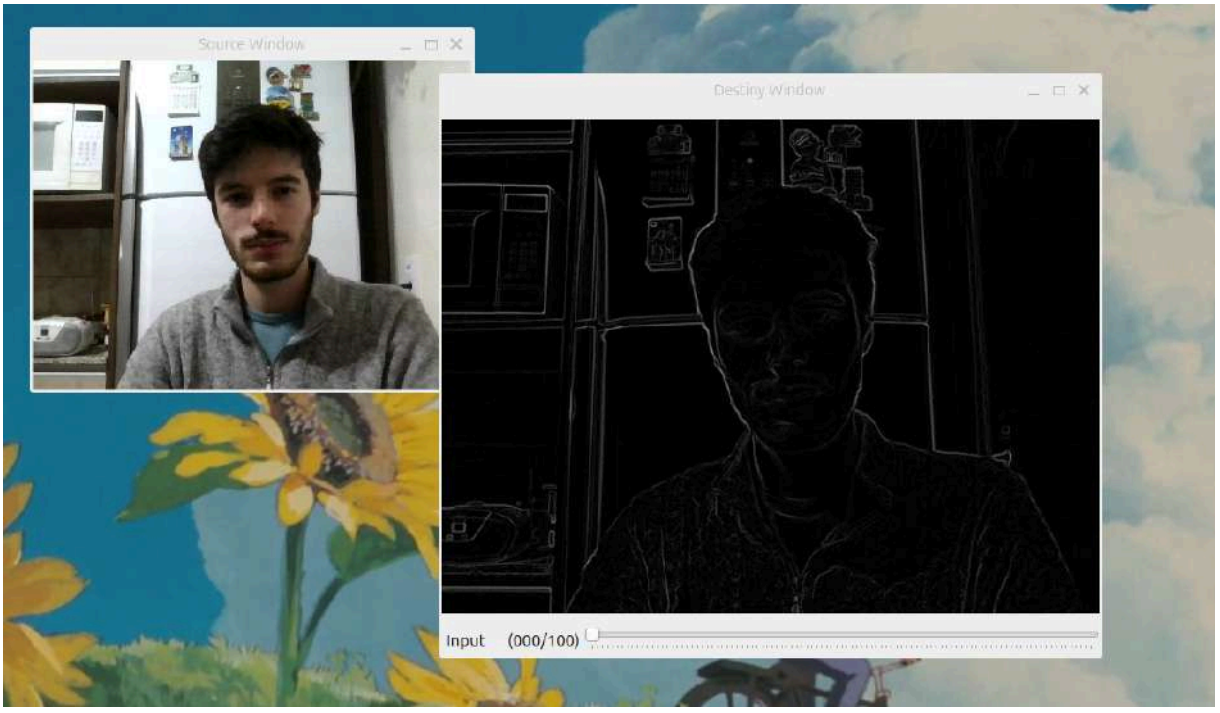

## Função *Canny*

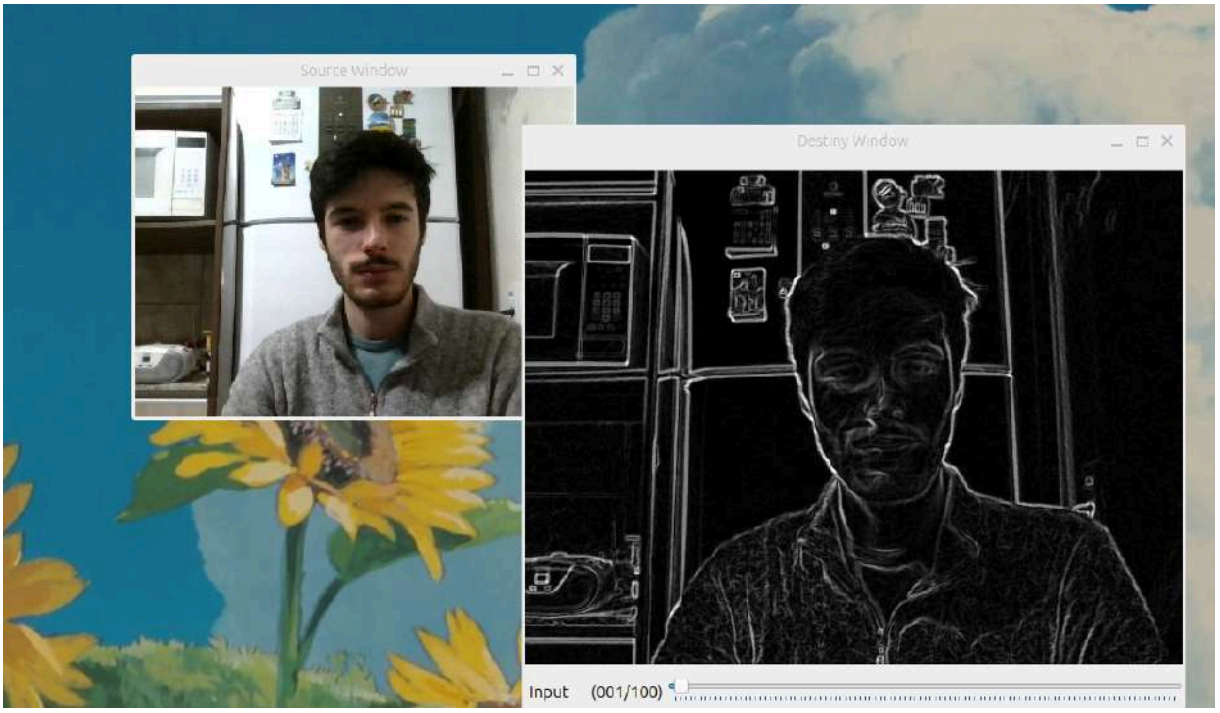
Utilizou-se a função `Canny()` para realizar a detecção de arestas. Ao selecionar essa opção, primeiramente a imagem é convertida para tons de cinza e após é convoluída com um kernel Gaussiano para minimizar o ruído. A intensidade da aplicação pode ser controlada pelo *input* deslizante.




## Função *Sobel*

A função `Sobel()` também possui o objetivo de detectar arestas, mas seus resultados são diferentes dos da função `Canny()` em diversos aspectos.

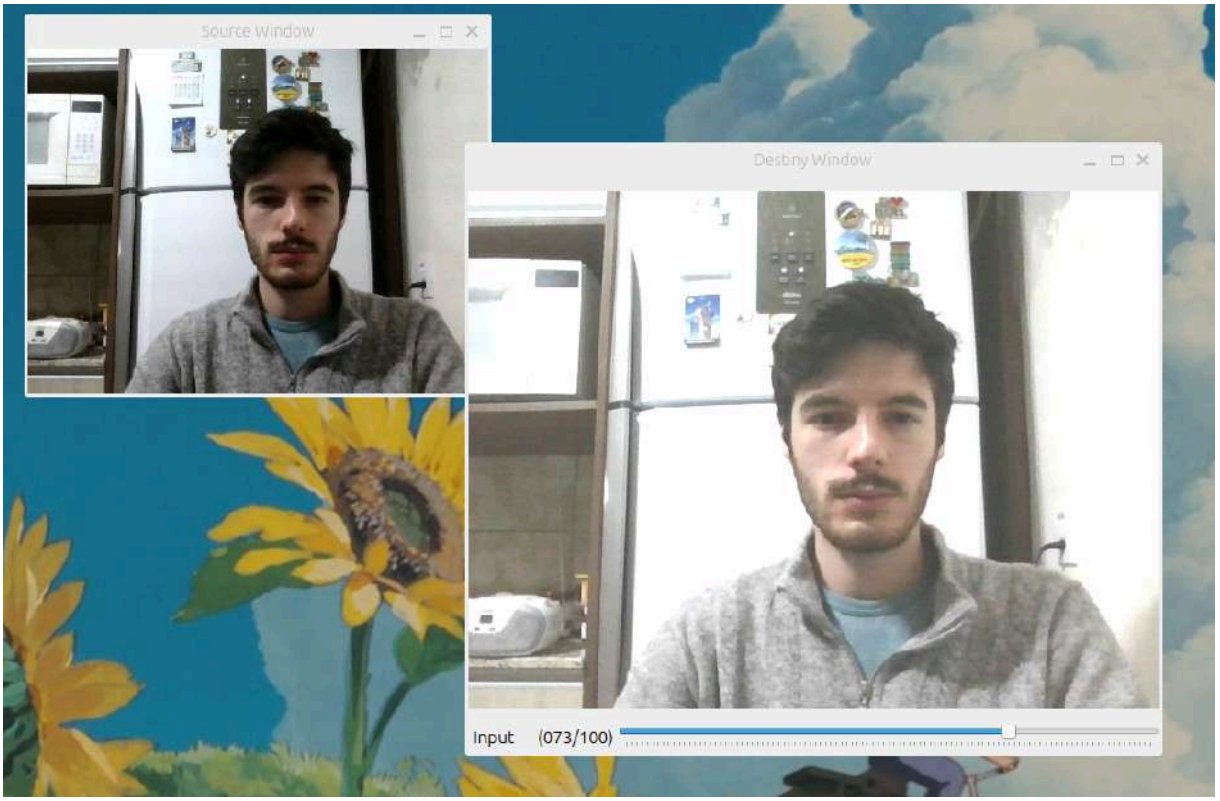


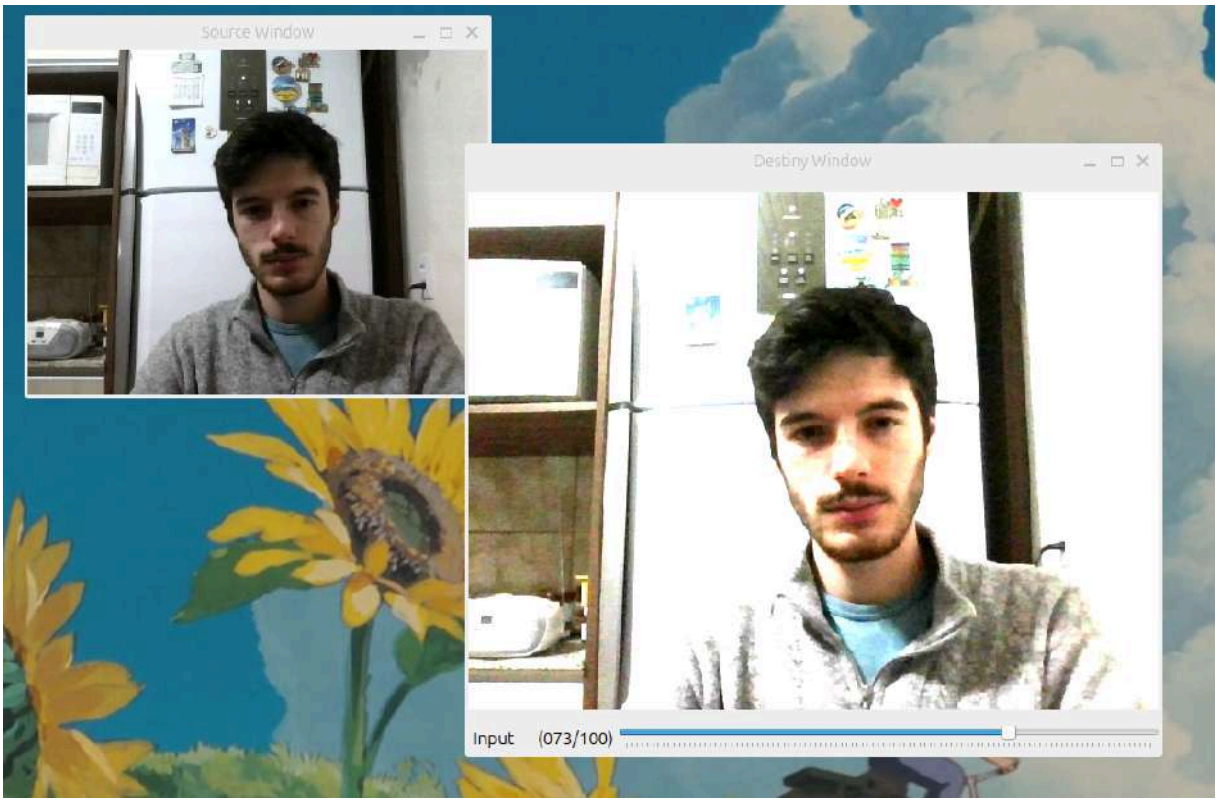



## Brilho, contrate e cálculo do negativo

O ajuste de brilho, contraste foram feitos utilizando a função `convertTo()`, via passagem de parâmetros *alpha* e *beta.* O negativo foi calculado através da subtração $255 - pixelValue$.

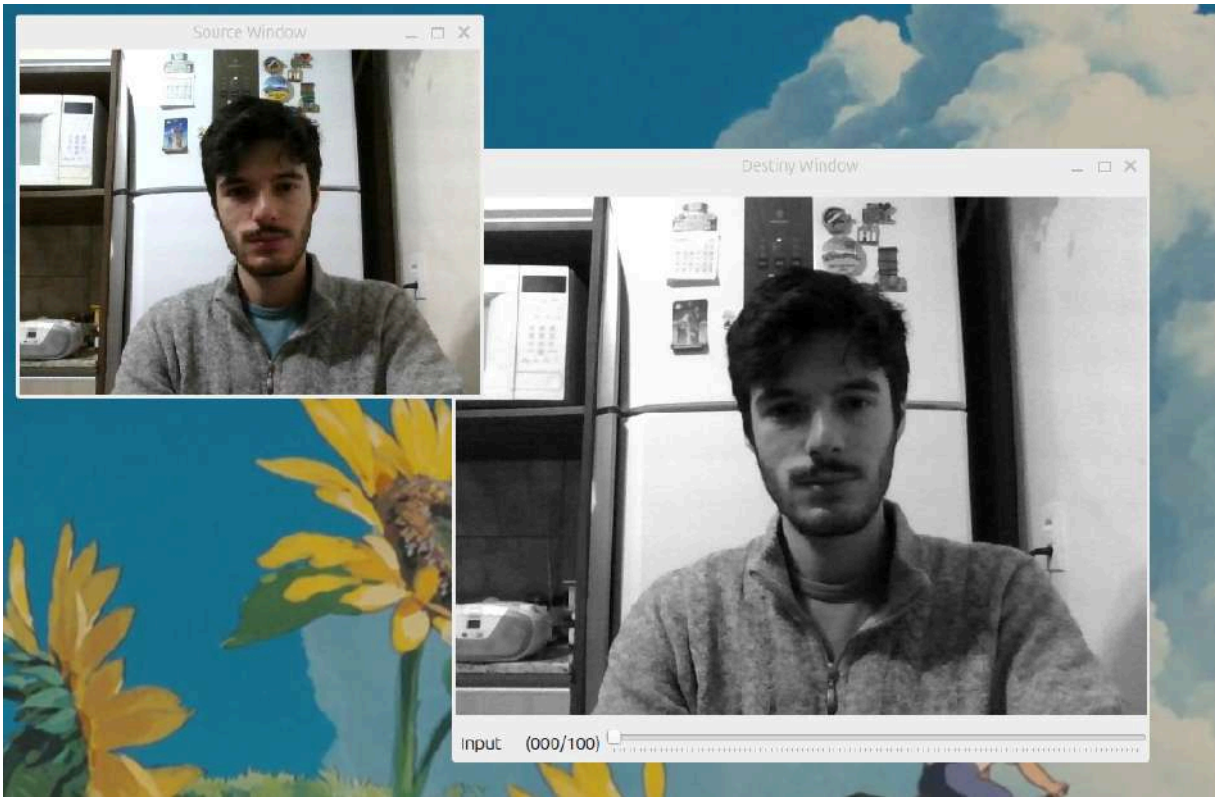
```
if (operations['I'].first) {
    aux = operations['I'].second;
    if (key_values.at(last_pressed) == 'I')
        aux = operations['I'].second = slider;
    dst.convertTo(dst, -1, 1, slider);
}

if (operations['O'].first) {
    aux = operations['O'].second;
    if (key_values.at(last_pressed) == 'O')
        aux = operations['O'].second = 1 + (slider / 50.0f);
    dst.convertTo(dst, -1, aux, 0);
}

if (operations['N'].first) {
    dst = Scalar::all(255) - dst;
}
```


Source Window
Desbny Window
Input (073/100)


Source Window
Desbny Window
Input (073/100)

## Cálculo da luminância (*grayscale*)

Fez-se o cálculo da luminância utilizando-se a função `cvtColor()`, que converte uma imagem de um para outro domínio de cor, no caso, COLOR_BGR2GRAY.




## Redimensionamento da imagem

Utilizou-se a função `resizeWindow()` para fazer o redimensionamento da imagem, dividindo, progressivamente, suas dimensões por 2, a cada novo clique na respectiva opção.

## Rotação

Implementou-se a rotação no sentido horário com a função `rotate()`, sucessivamente, conforme os cliques na sua respectiva opção. A imagem é rotacionada até voltar à sua orientação original.


## Espelhamento vertical e horizontal

O espelhamento foi implementado utilizando-se a função `flip()`, cujos parâmetros permitem escolher entre espelhamento vertical ou horizontal.

## Gravação de vídeo

Por fim, a gravação e salvamento de vídeo foram implementados utilizando o tipo *VideoWriter*, implementado pelo OpenCV para gravação de vídeos. Ademais, o formato de codificação utilizado foi o “MJPG”. O vídeo é gravado conforme a taxa de *fps* fornecida pela câmera de captura, sem um número pré-definido.

Vídeo demonstrativo: https://youtu.be/vJBhVck17OI.

Concluindo, pode-se afirmar que processamento de vídeo é uma atividade muito divertida e similar ao que foi executado nos trabalhos anteriores de processamento de imagens. A maior dificuldade encontrada, provavelmente, foi na etapa de configuração do ambiente e na familiarização com a documentação do OpenCV. No entanto, após esse primeiro choque, o desenvolvimento se deu de forma tranquila e engajante.