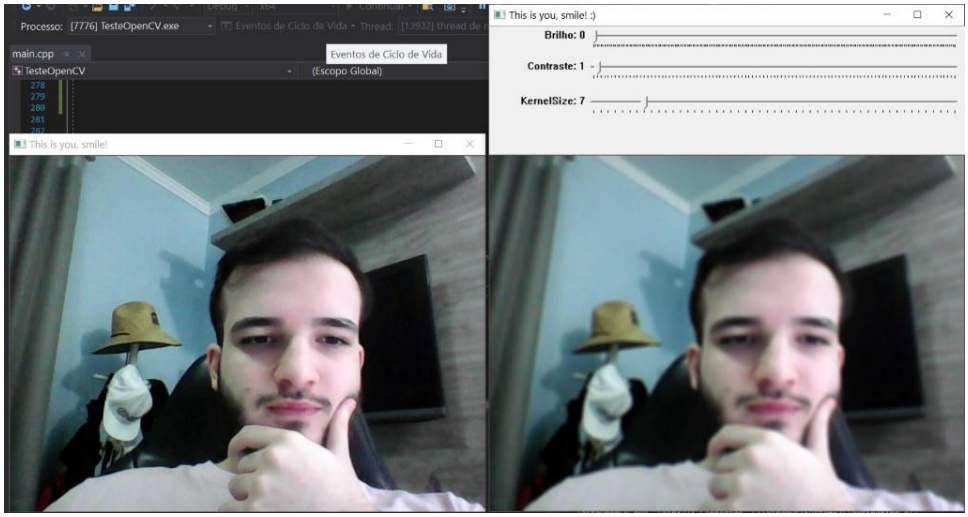
RELATÓRIO TRABALHO 3

2.GaussianBlur:

O comando GaussianBlur depende basicamente do tamanho do kernel. Uma trackbar foi implementada para controlar o tamanho do kernel.

Uma observação é que a escolha de um tamanho par para o kernel causa erro no programa, para corrigir isso, foi acrescentado um no tamanho quando ele for par. Também poderia ser decrementado, porém causaria erro no kernel de dimensão 1.



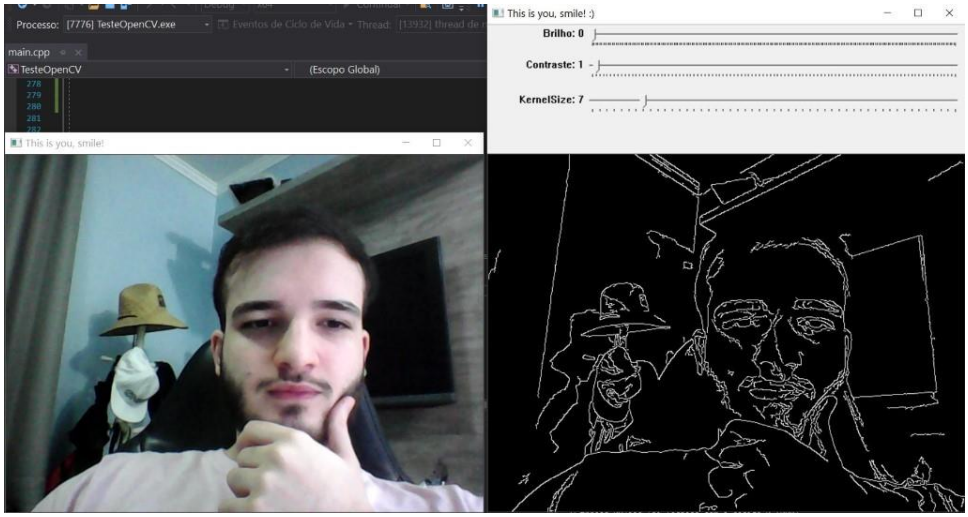
3.Canny:

O comando Canny depende de 2 “thesholds” para a implementação. Depois de uma procura na internet de melhores valores, acabei optando por 40 e 120, que resultaram em um resultado melhor.

Além disso para o comando funcionar é necessário converter a imagem para tons de cinza antes, e para obter um melhor resultado é importante que se aplique o GaussianBlur antes.

Ademais, depois do procedimento a imagem é convertida de volta para RGB para tratar algumas exceções caso uma sequência especifica de comandos seja aplicada.

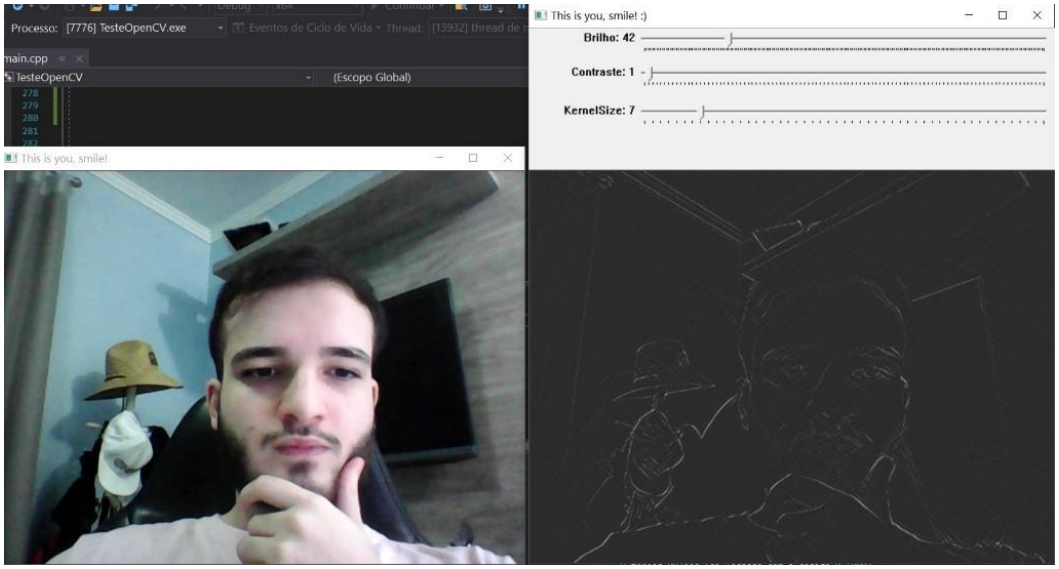
Canny sem aplicação de gblur:(Com gblur pode ser visto no link disponibilizado).



4.Sobel:

O comando Sobel depende do depth da imagem, para isso foi usado a função .depth(), além das ordens das derivadas dx, dy que no programa foi usado derivadas de ordem 1.

Ademais o tamanho do kernel deve ser igual a 1, 3, 5 ou 7 para não causar erro no programa.

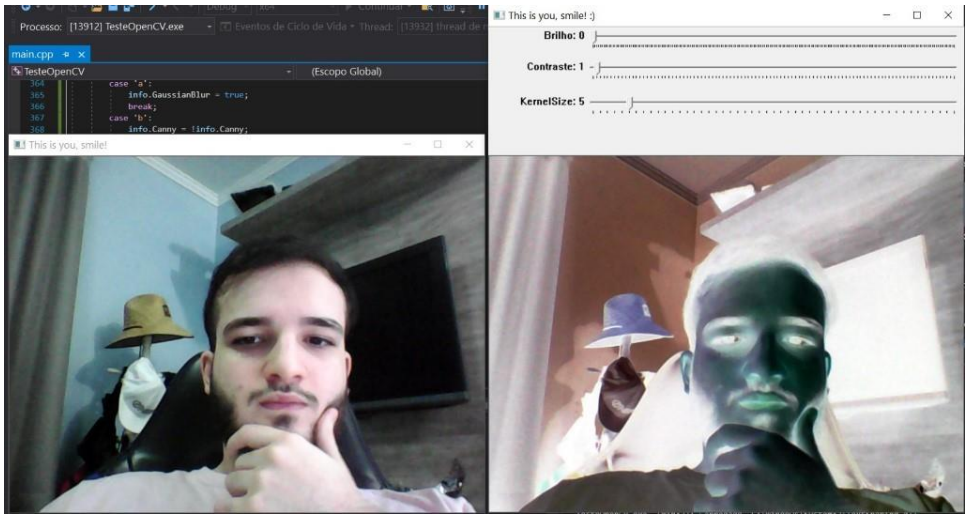


5:Brilho, Contraste, Negativo:

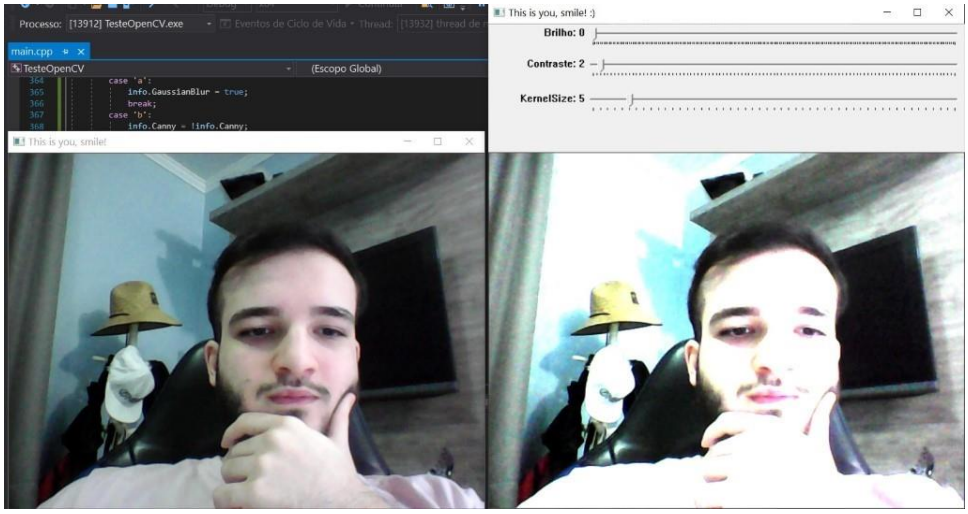
O comando ConvertTo depende de duas variáveis alpha e beta, que representam o contraste e o brilho, respectivamente.

- **Brilho:** Foi criada outra trackbar para controlar o brilho livremente no programa.
- **Contraste:** Foi criada outra trackbar para controlar o contraste livremente no programa.
- **Negativo:** Para obter o negativo é necessário que o alpha seja –1 e beta =255.

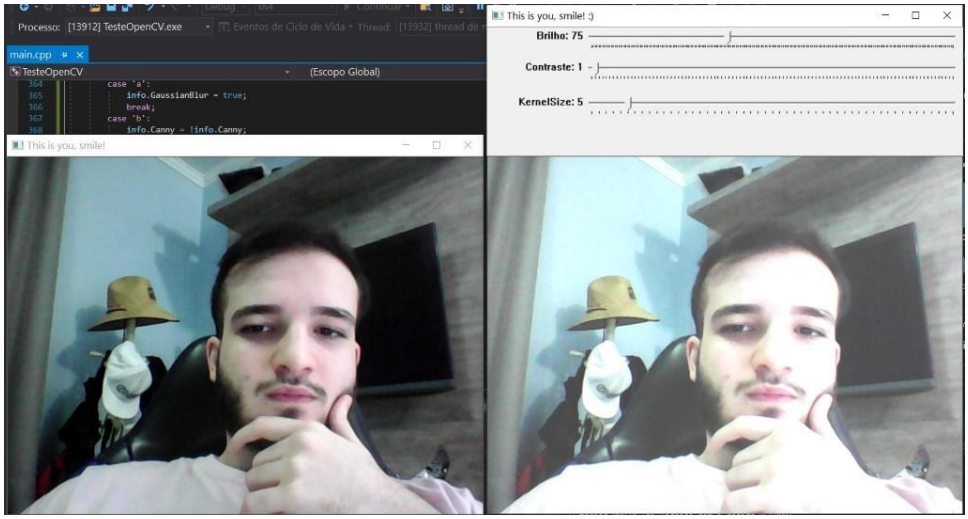
Negativo:



Contraste=2:

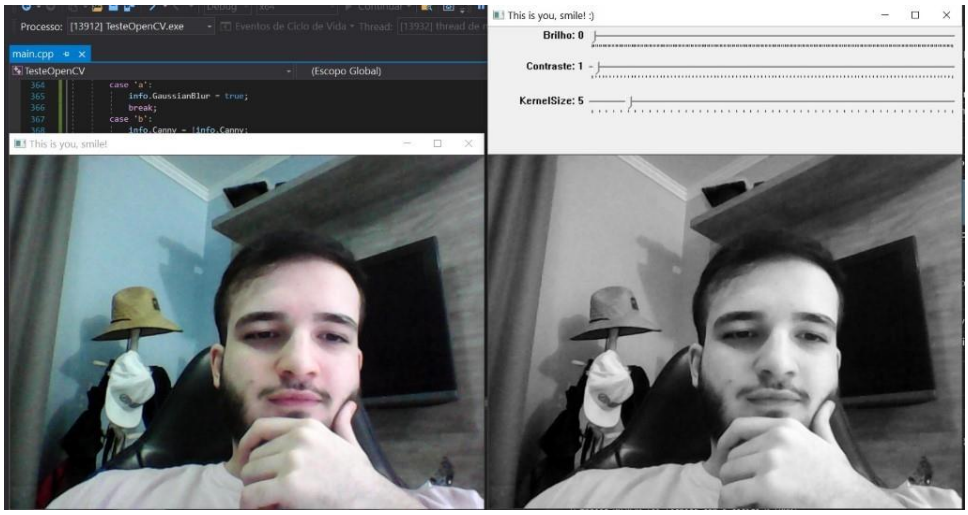


Brilho=75:



6:Tons de Cinza:

Converter uma imagem em tons de cinza é bem simples, basta apenas usar o comando cvtColor com os parâmetros para converter em tons de cinza que são cvtColor(src, dst, COLOR_BGR2GRAY).



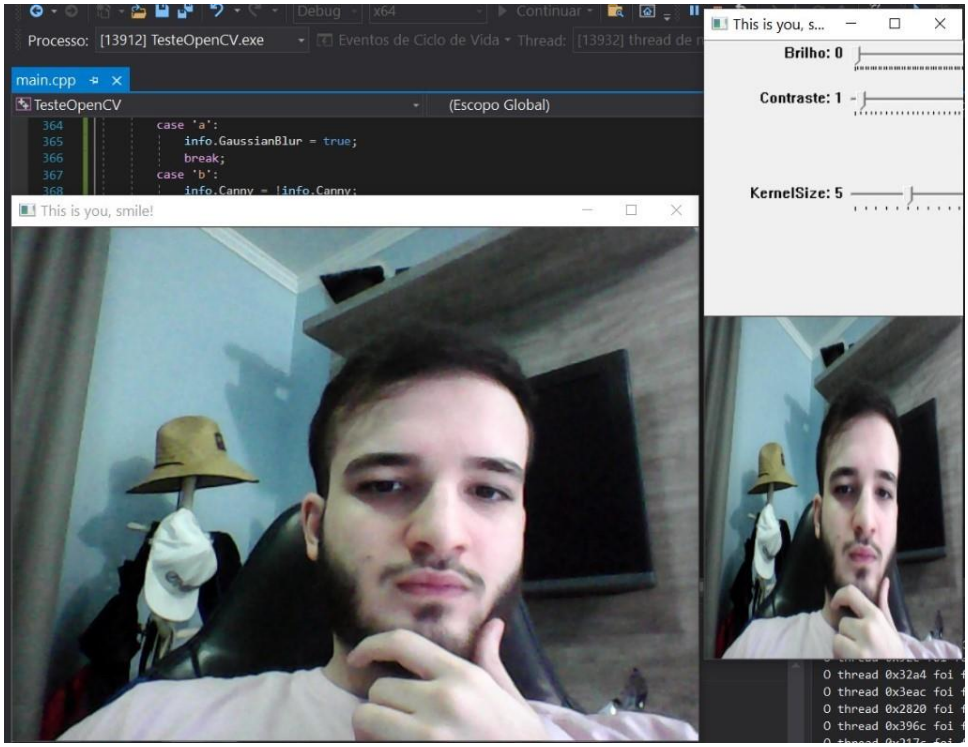
7: Redimensionamento:

Para o redimensionamento foi usado o comando cv::resize que recebe como argumentos a imagem fonte, destino e o tamanho da nova imagem, exemplo:

resize(src,dst,Size(NewRow,NewCol)) para fazer o redimensionamento.

Além disso foi usado potências para reduzir pela metade as dimensões, exemplo: quando redimensionamento for 1 se divide por 2, quando redimensionamento for 2 se divide por 4, e assim por diante.

Redimensionado 1 vez:



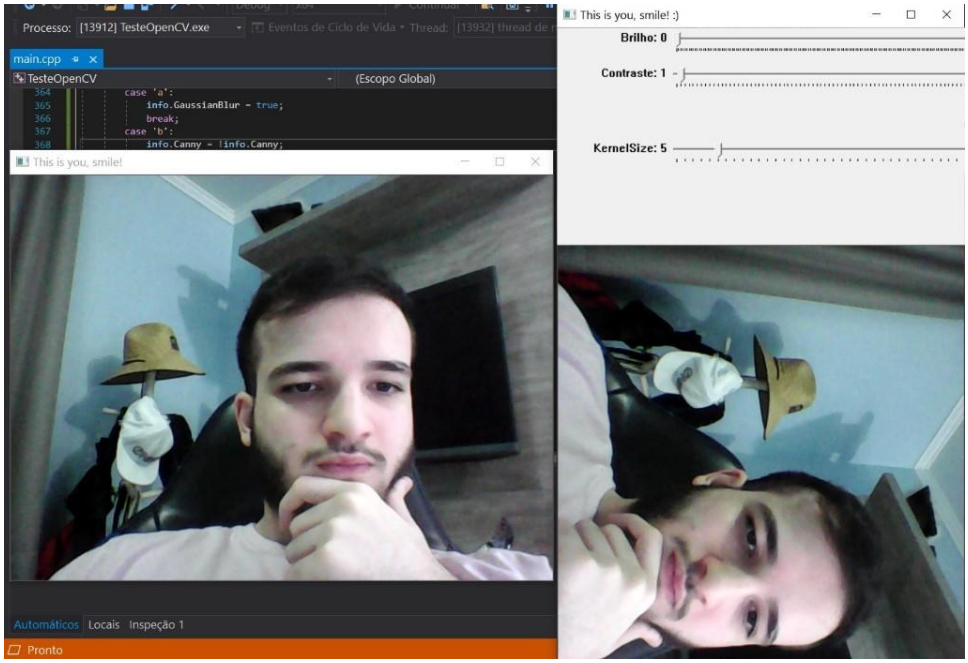
8:Rotação:

Para a rotação foi usado o comando cv::rotate que recebem como argumentos a imagem fonte, destino e o sentido em que se deseja rotacionar.

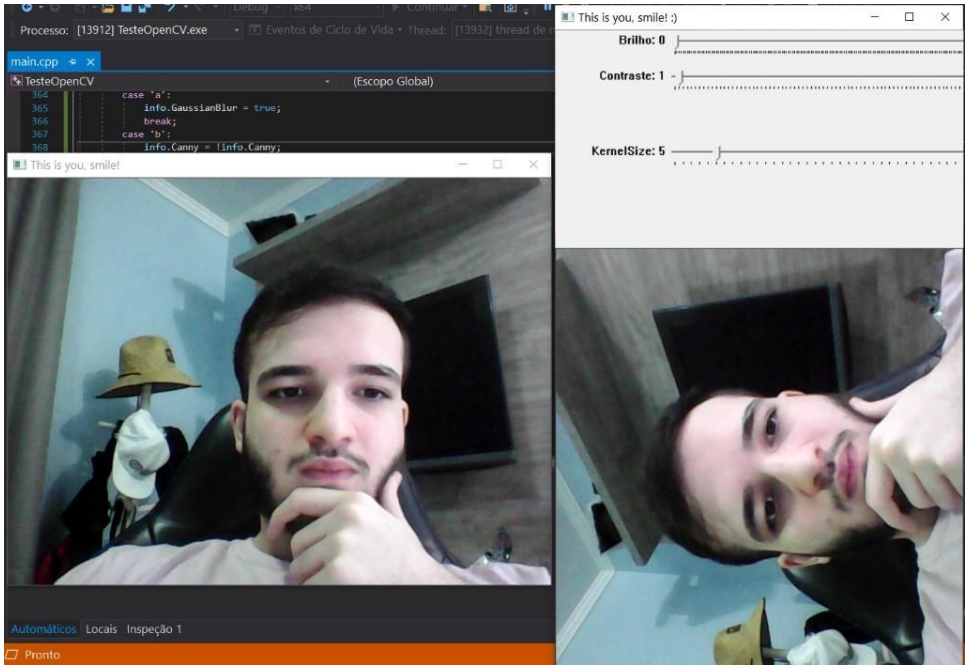
Exemplo 90 graus clockwise: cv::rotate(src, dst, ROTATE_90_CLOCKWISE)

Exemplo 180 graus: cv::rotate(src, dst, ROTATE_180)

Rotação 90 graus:



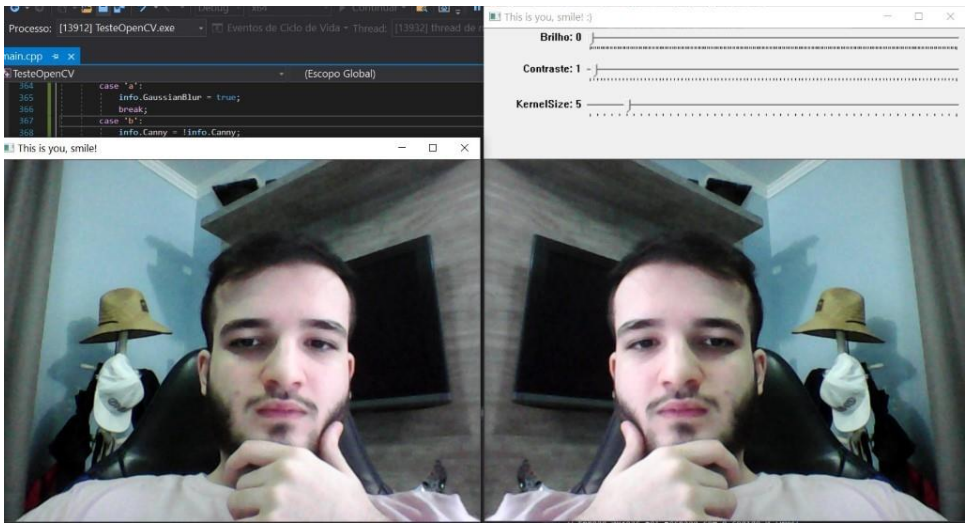
Rotação 270 graus:



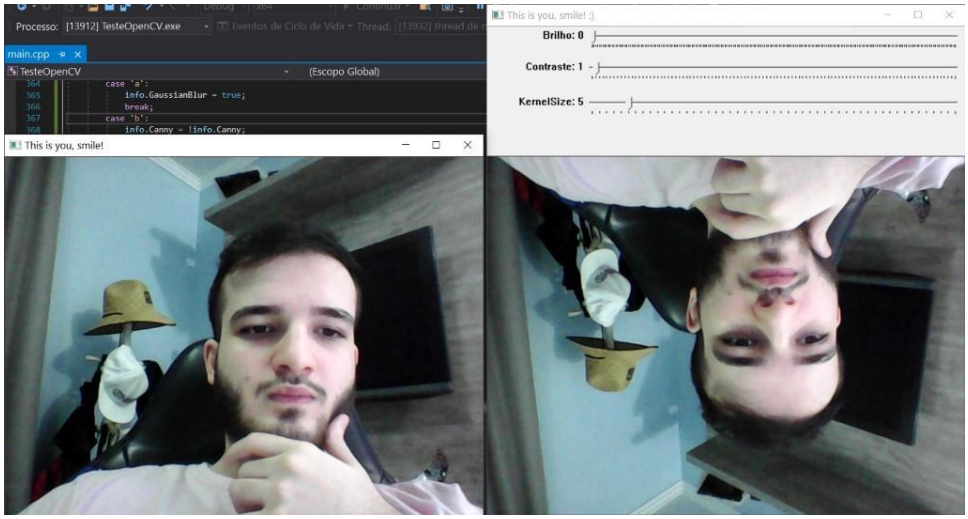
9: Espelhamento:

Para o espelhamento foi usado o comando flip que recebe como argumentos a imagem fonte, destino e um flipcode, que é 0 quando o espelhamento é no eixo x e é 1 quando o espelhamento é no eixo y.

Horizontal:



Vertical:



10:Gravação de Vídeo:

O resultado das transformações, exceto redimensionamento e rotação, está contido no vídeo gerado pelo programa no link do youtube.

Link: https://www.youtube.com/watch?v=CPj_zflt_U4

TABELA DE COMANDOS DO PROGRAMA:

```
switch (key)
{
case 'a':
    info.GaussianBlur = true;
    break;
case 'b':
    info.Canny = !info.Canny;
    break;
case 'c':
    info.Sobel = !info.Sobel;
    break;

case 'd':
    info.Negative = !info.Negative;
    break;

case 'e':
    info.GrayScale = true;
    break;

case 'f':
    info.espelhamentoHorizontal = !info.espelhamentoHorizontal;
    break;

case 'g':
    info.espelhamentoVertical = !info.espelhamentoVertical;
    break;

case 'h':
    if (info.Rotacao == ROTATE_90)
        info.Rotacao = ROTATED_180;
    else
        if (info.Rotacao == ROTATED_180)
        info.Rotacao = ROTATE_270;
    else
        if (info.Rotacao == ROTATE_270)
        info.Rotacao = ROTATE_360;
    else
        if(info.Rotacao=ROTATE_360)
        info.Rotacao = ROTATE_90;
    break;
case 'i':
    info.Redimensionamento += 1;
    break;
case 'j':
    gravar = true;
    break;
case 'k':
    salvargrav = true;
    break;
case '0':
    info.reinicializa();
    break;
default:
```

‘a’:GaussianBlur

‘b’:Canny

‘c’: Sobel

‘d’:Negativo

‘e’: Tons de Cinza

‘f’: Espelhamento Horizontal

‘g’: Espelhamento Vertical

‘h’: Rotação

‘i’: Redimensionamento

‘j’: Começar Gravação

‘k’: Salvar Gravação

‘0’: Reinicializar variáveis

OBS: O brilho e o contraste podem ser sempre ajustados na trackbar, não necessitando do acionamento de uma tecla específica.