BIBLIOTECA PARA ANÁLISE DE DADOS EM IMAGENS ESTEREOSCÓPICAS

Aluno: Ricardo I Salvador

Orientador: Marcel Hugo

FURB
UNIVERSIDADE DE BLUMENAU

Roteiro

- Introdução à biblioteca
- Objetivo da biblioteca
- Fundamentação Teórica
- Trabalhos Correlatos
- Requisitos
- Especificação
- Implementação
- Operacionalidade
- Resultados

Introdução

- Porque criar uma biblioteca?
- Motivação para a criação da biblioteca estéreo?
- Código aberto?

Objetivos

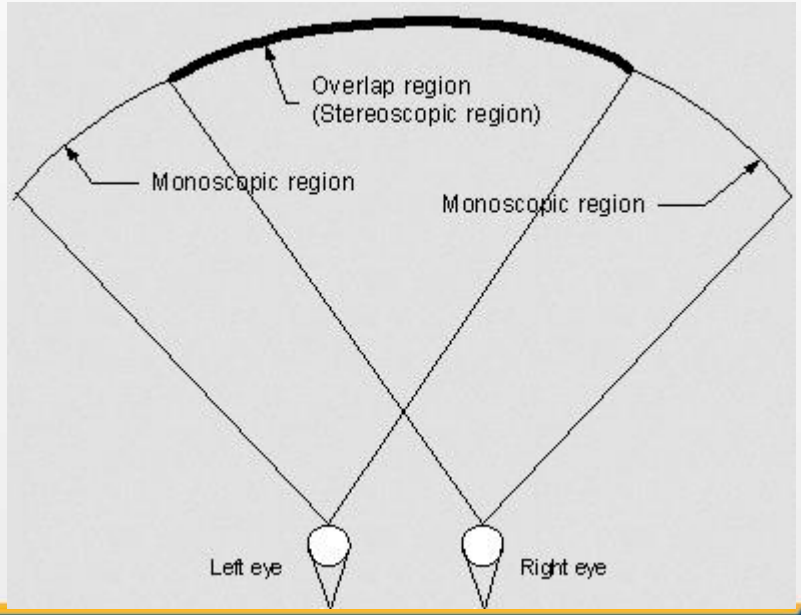
- Desenvolver uma biblioteca e que trabalha com algoritmos de visão computacional aplicados para imagens estereoscópicas;
- Desenvolver uma ferramenta que faz uso e demonstra o funcionamento da biblioteca desenvolvida;
- Disponibilizar a biblioteca e o seu código fonte para uso público;
- Disponibilizar a ferramenta e o seu código fonte para uso público com intuito de explicar o funcionamento da biblioteca.

Fundamentação Teórica

- O que é estereoscopia?

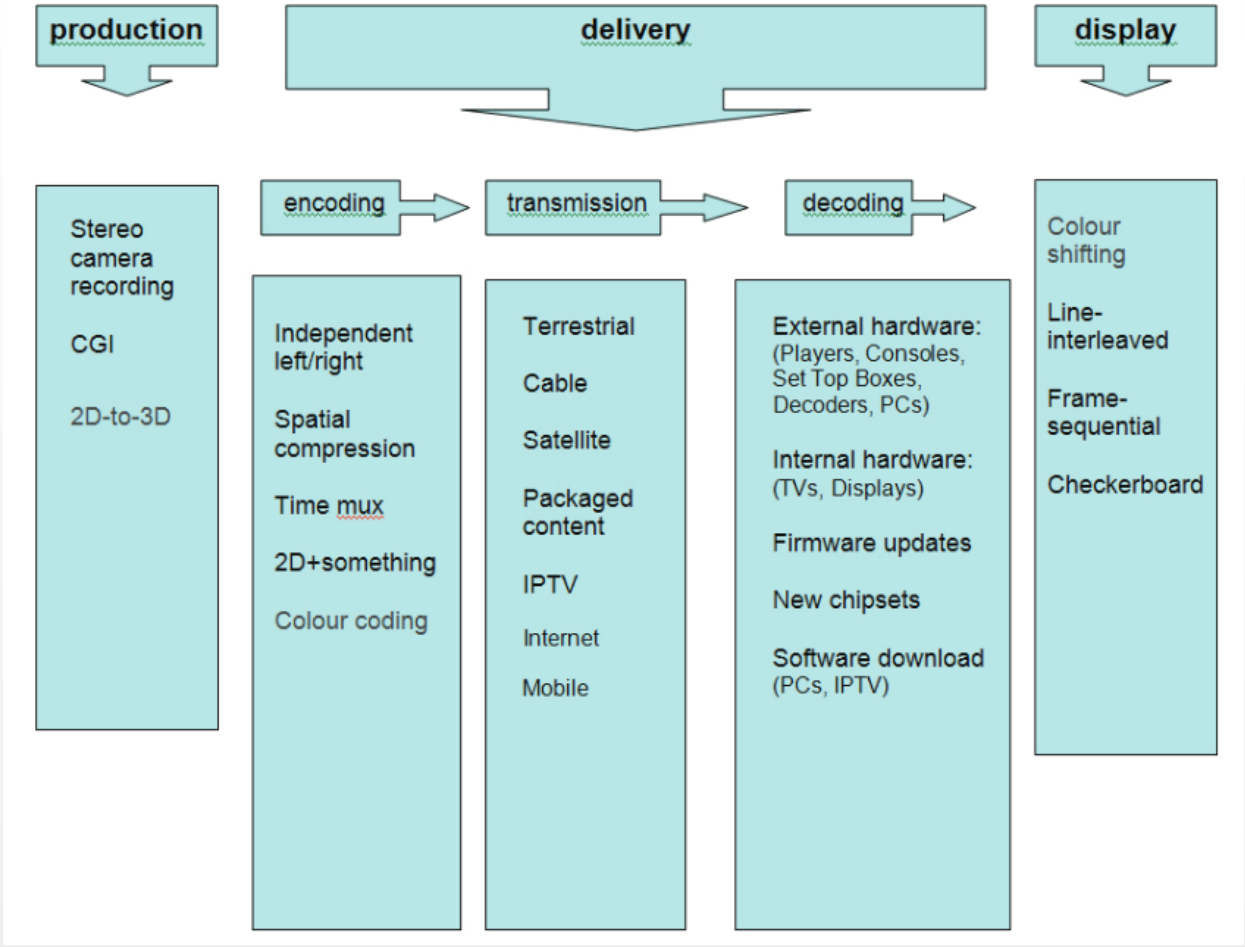
O termo estereoscopia deriva das palavras gregas ***stereos*** e ***skopein***, que significam respectivamente sólido/relevo e olhar/ver, que quer dizer, visão em relevo.

- Por que usar?



Fundamentação Teórica

- **CICLO DE VIDA DO CONTEÚDO 3D**



Fundamentação Teórica

- **PRODUÇÃO DO CONTEÚDO:**

a) Captura por câmeras;

b) Imagens geradas pelo computador (CGI);

c) Conversão de imagens 2D para 3D.

Fundamentação Teórica

- **DISTRIBUIÇÃO DO CONTEÚDO:**

a) Codificação do conteúdo;

- Compressão espacial;
- Interlação temporal;
- 2D + metadados;
- Deslocação de cores.

b) Transmissão do conteúdo;

c) Decodificação do conteúdo

Fundamentação Teórica

- **REPRODUÇÃO DO CONTEÚDO:**

Trabalhos Correlatos

- **CÁLCULO DE VOLUME EFETIVO DE OBJETOS EM MOVIMENTO USANDO ESTEREOSCOPIA (BARROS, 2009);**
- **PROTÓTIPO DE UM AMBIENTE DE VISUALIZAÇÃO COM TÉCNICAS DE ESTEREOSCOPIA (MOMM, 2001);**
- **OPENSTEREO: UMA BIBLIOTECA PARA SUPORTE AO DESENVOLVIMENTO DE APLICAÇÕES ESTEREOSCÓPICAS (LEITE, 2006).**

Requisitos

- RF001 – disponibilizar uma biblioteca que permite a aplicação de algoritmos de visão computacional em imagens estereoscópicas
- RF002 – inserir registro no banco de dados do dispositivo móvel;
- RF003 – biblioteca deve permitir o carregamento de vídeos estéreo;
- RF004 – biblioteca deve permitir a aplicação de filtros do OpenCV nos vídeos carregados;
- RF005 – biblioteca deve permitir salvar os vídeos alterados em arquivos compatíveis com dispositivos de exibição estéreo;
- RF006 – disponibilizar uma ferramenta que faz uso e demonstra o funcionamento da biblioteca que foi desenvolvida;
- RF007 – utilizar algoritmos da biblioteca *Open source computer vision library* (OpenCV) e aplicá-los em vídeos 3D;

- RNF001 – deve ser compatível com a plataforma Windows.
- RNF002 – especificação feita com a ferramenta Astah Community
- RNF003 – realizar a implementação no ambiente Visual Studio

Especificação – Caso de Uso

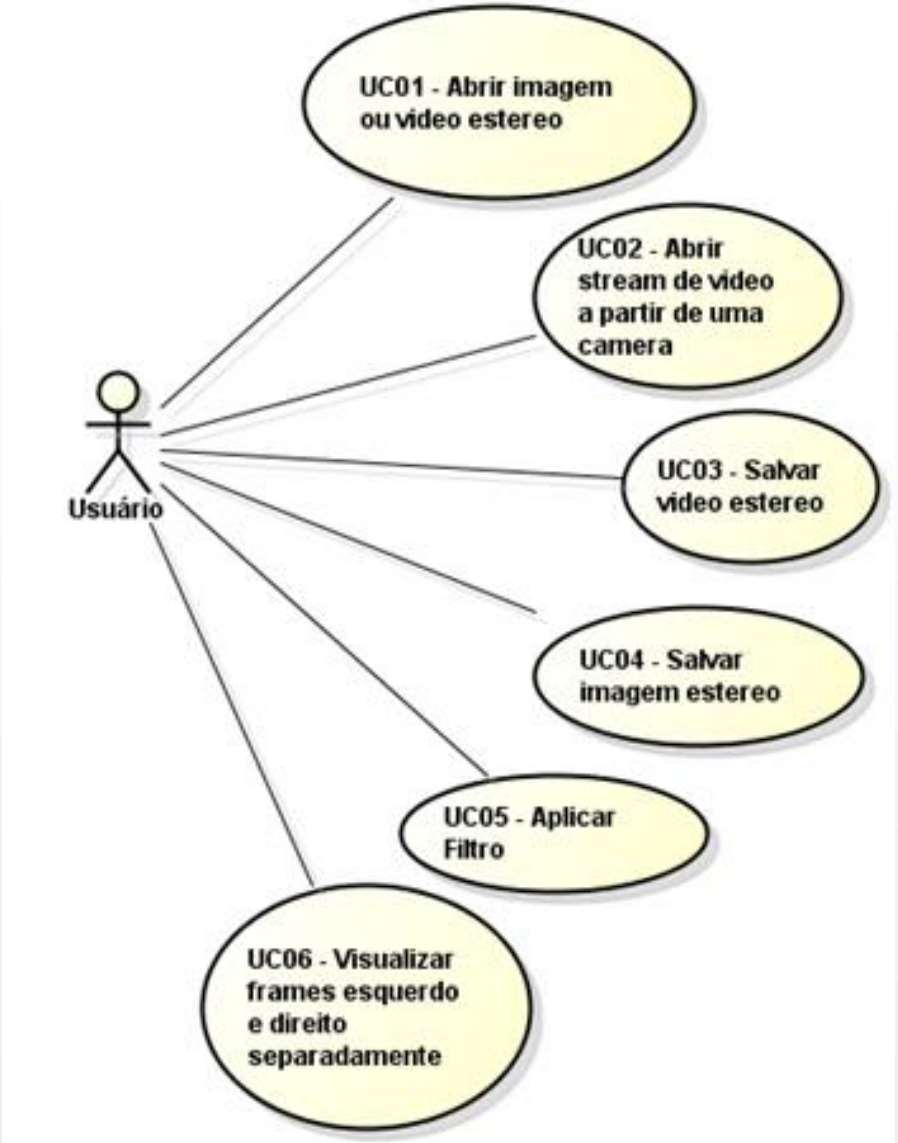
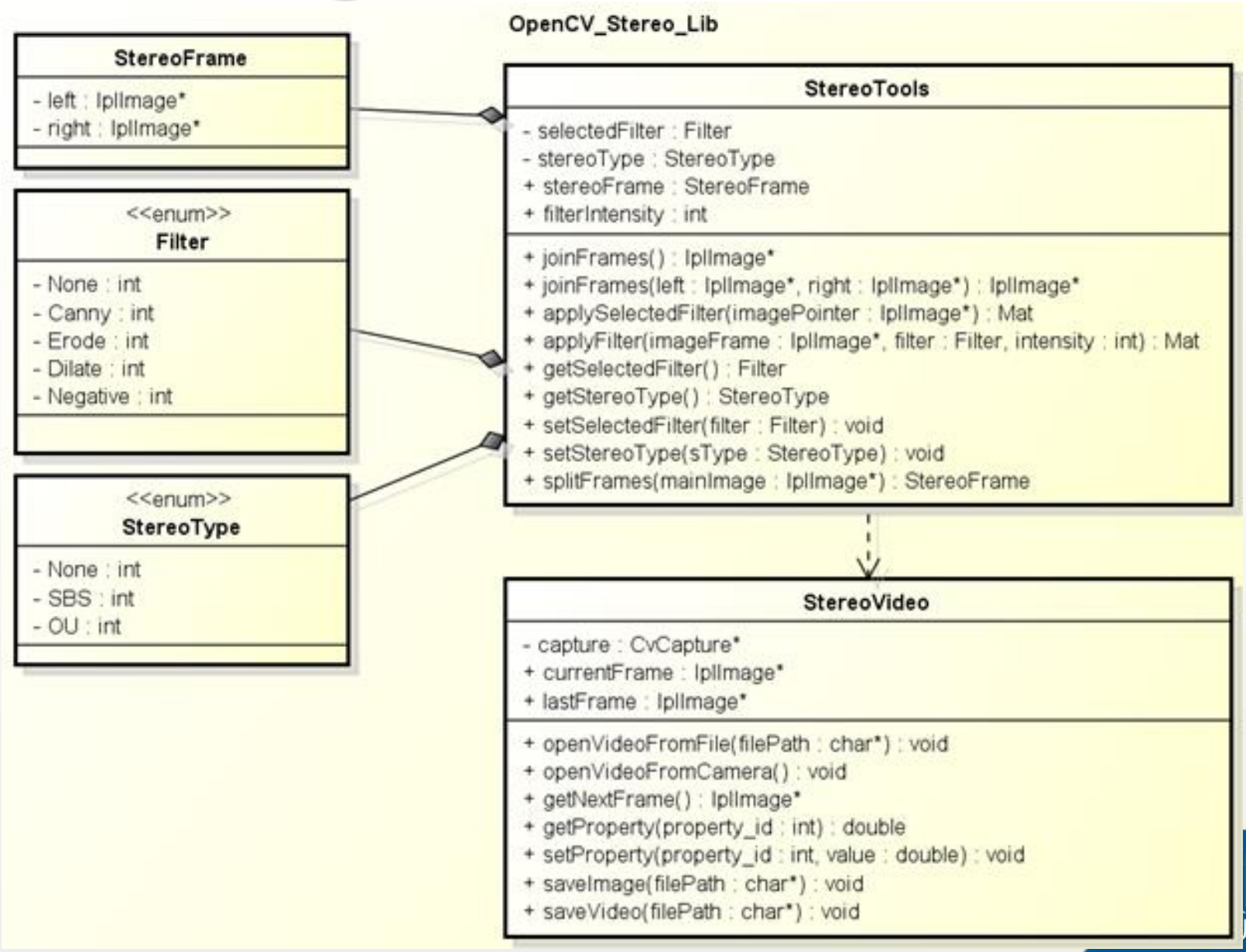
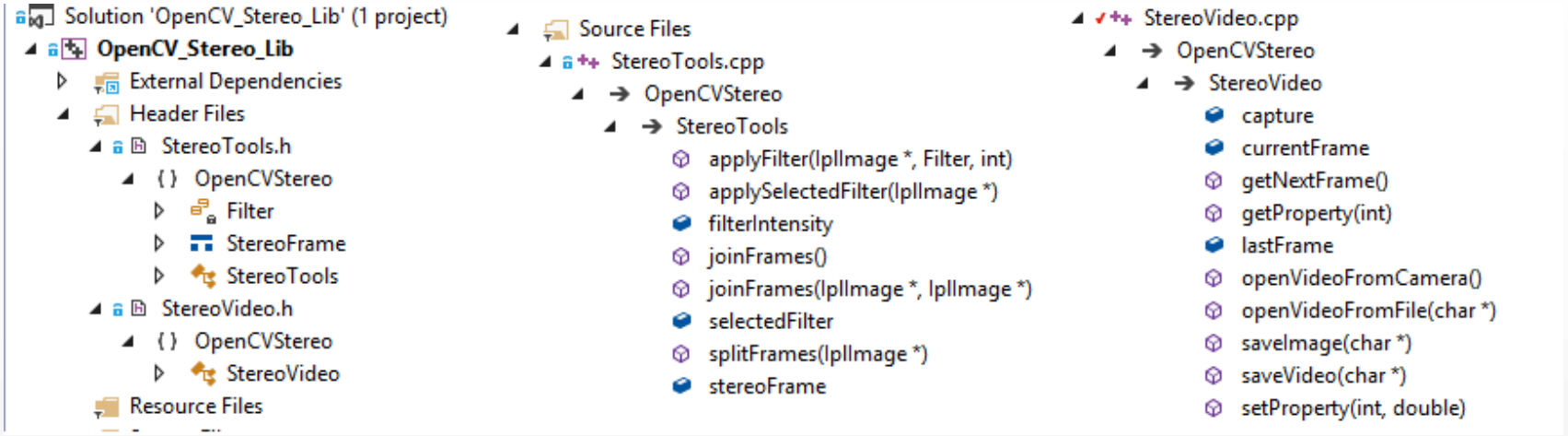


Diagrama de Classes

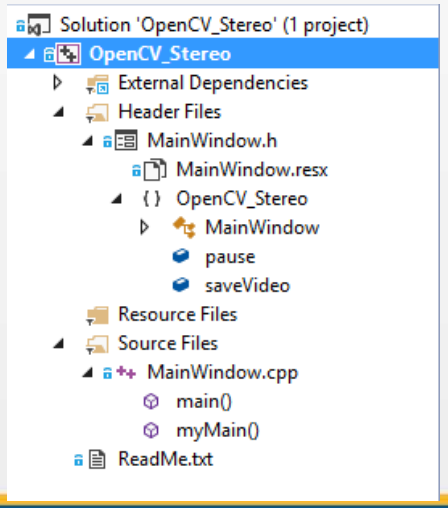


Implementação

- Biblioteca

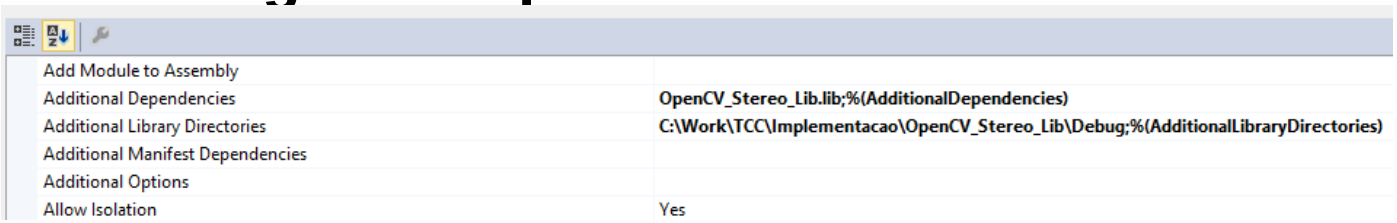


- Ferramenta



Biblioteca – Como usar?

1 – Configurar dependências.



2 - Adicionar includes

```
#include "StereoTools.h"
#include "StereoVideo.h"
```

3 – Chamar funções

```
StereoVideo::openVideoFromFile(fileName);
StereoVideo::getNextFrame();
StereoTools::splitFrames();
Mat left = StereoTools::applySelectedFilter(splittedFrame.left);
Mat right = StereoTools::applySelectedFilter(splittedFrame.right);
StereoVideo::saveImage(fileName);
StereoVideo::saveVideo(fileName);
```

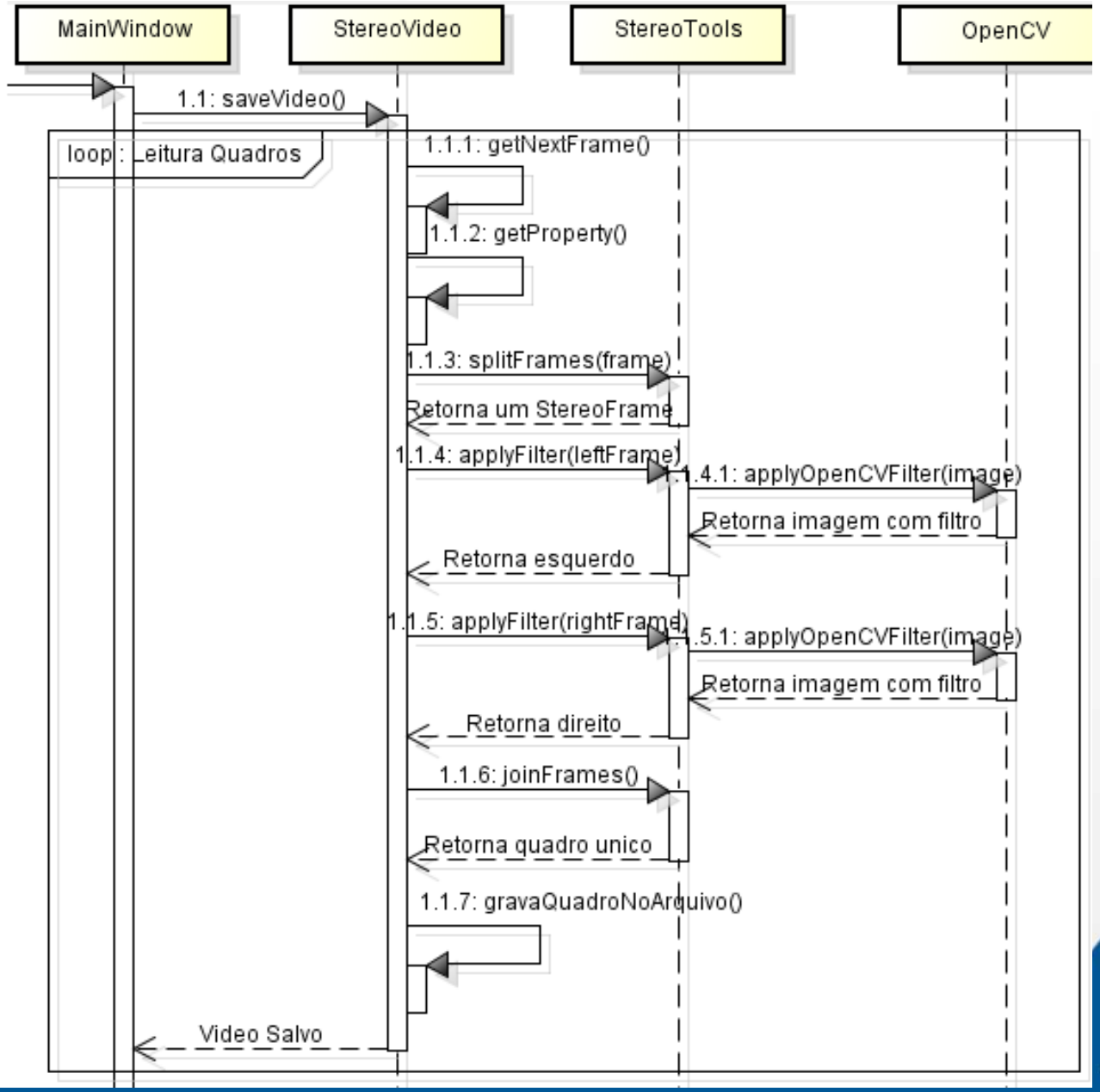
Biblioteca – StereoVideo

```
class StereoVideo
{
private:
	static CvCapture* capture;
public:
	static IplImage* currentFrame;
	static IplImage* lastFrame;
	static void openVideoFromFile(char* filePath);
	static void openVideoFromCamera();
	static IplImage* getNextFrame();
	static double getProperty(int property_id);
	static void setProperty(int proparty_id, double value);

	static void saveImage(char* filePath);
	static void saveVideo(char* filePath);
};
```

Biblioteca - StereoTools

```
class StereoTools
{
private:
	static Filter selectedFilter;
public:
	static StereoFrame stereoFrame;
	static int filterIntensity;

	static IplImage* joinFrames();
	static IplImage* joinFrames(IplImage* left, IplImage* right);

	static cv::Mat applySelectedFilter(IplImage* imagePointer);
	static cv::Mat applyFilter(IplImage* leftFrame, Filter filter =
Filter::None, int intensity = 0);

	static Filter getSelectedFilter(){ return selectedFilter; }

	static void setSelectedFilter(Filter filter){ selectedFilter =
filter; }

	static StereoFrame splitFrames(IplImage* mainImage);

};
```

DS – Split Frames



FURB
UNIVERSIDADE DE BLUMENAU

Biblioteca – SplitFrames()

```
StereoFrame OpenCVStereo::StereoTools::splitFrames(IplImage*
mainImage){
      if (!mainImage)
      {
            mainImage = StereoVideo::currentFrame;
      }
      cvReleaseImage(&stereoFrame.left);
      cvReleaseImage(&stereoFrame.right);
      stereoFrame.left = NULL;
      stereoFrame.right = NULL;
      //LEFT FRAME
      /* sets the Region of Interest  - rectangle area has to be
__INSIDE__ the image */
      cvSetImageROI(mainImage, cvRect(0, 0, mainImage->width / 2,
mainImage->height));
      /* create destination image  - cvGetSize will return the width and
the height of ROI */
      stereoFrame.left = cvCreateImage(cvGetSize(mainImage), mainImage-
>depth, mainImage->nChannels);
      /* copy subimage */
      cvCopy(mainImage, stereoFrame.left, NULL);
      ///* always reset the Region of Interest */
      cvResetImageROI(mainImage);
```

... Mesma operação para o quadro direito**...**

```
      return stereoFrame;

}
```

Biblioteca – JoinFrames()

```
IplImage*  OpenCVStereo::StereoTools::joinFrames(IplImage* left,
IplImage* right)
{
	IplImage* returnFrame;
	CvSize cvSize = CvSize();
	cvSize.height = cvGetSize(left).height;
	cvSize.width = cvGetSize(left).width + cvGetSize(right).width;


	returnFrame = cvCreateImage(cvSize, left->depth, left->nChannels);


	cvSetImageROI(returnFrame, cvRect(0, 0, left->width, left-
>height));
	cvCopy(left, returnFrame, NULL);

	cvSetImageROI(returnFrame, cvRect(left->width, 0, returnFrame-
>width, returnFrame->height));
	cvCopy(right, returnFrame, NULL);

	/* always reset the Region of Interest */
	cvResetImageROI(returnFrame);

	return returnFrame;

}
```

Biblioteca – OpenCV Filters

```
switch (filter)
{
case Filter::Canny:
    cvtColor(cv::Mat(frame), myFrameResult, CV_BGR2GRAY, 3);
    GaussianBlur(myFrameResult, myFrameResult, Size(7, 7), 1.5, 1.5);
    Canny(myFrameResult, myFrameResult, 0, 20 + intensity, 3);
    break;
case Filter::Erode:
    cvErode(frame, frame, 0, 2 + intensity);
    myFrameResult = frame;
    break;
case Filter::Dilate:
    cvDilate(frame, frame, 0, 2 + intensity);
    myFrameResult = frame;
    break;
case Filter::Negative:
    cvNot(frame, frame);
    myFrameResult = frame;
    break;
case Filter::None:
default:
    return frame;
    break;
}
```

Ferramenta – setFrameToPicture()

```
private: System::Void setFrameToPicture(Mat frame, PictureBox^ picture)
{
    //LeftImage
    System::Drawing::Graphics^ graphicsLeft = picture-
>CreateGraphics();
    System::IntPtr ptrLeft(frame.ptr());
    System::Drawing::Bitmap^ bL;
    switch (frame.type())
    {
    case CV_8UC3: // imagens com três canais são indexadas aqui.
     bL = gcnew System::Drawing::Bitmap(frame.cols, frame.rows,
frame.step,
         System::Drawing::Imaging::PixelFormat::Format24bppRgb,
ptrLeft);
     break;
    case CV_8UC1: // imagens com um canal são indexadas aqui
     bL = gcnew System::Drawing::Bitmap(frame.cols, frame.rows,
frame.step,
         System::Drawing::Imaging::PixelFormat::Format8bppIndexed,
ptrLeft);
     break;
    default:
     break;
    }
    System::Drawing::RectangleF rectL((float)0, (float)0,
(float)picture->Width, (float)picture->Height);
    graphicsLeft->DrawImage(bL, rectL);
}
```

Ferramenta – Carregando do arquivo

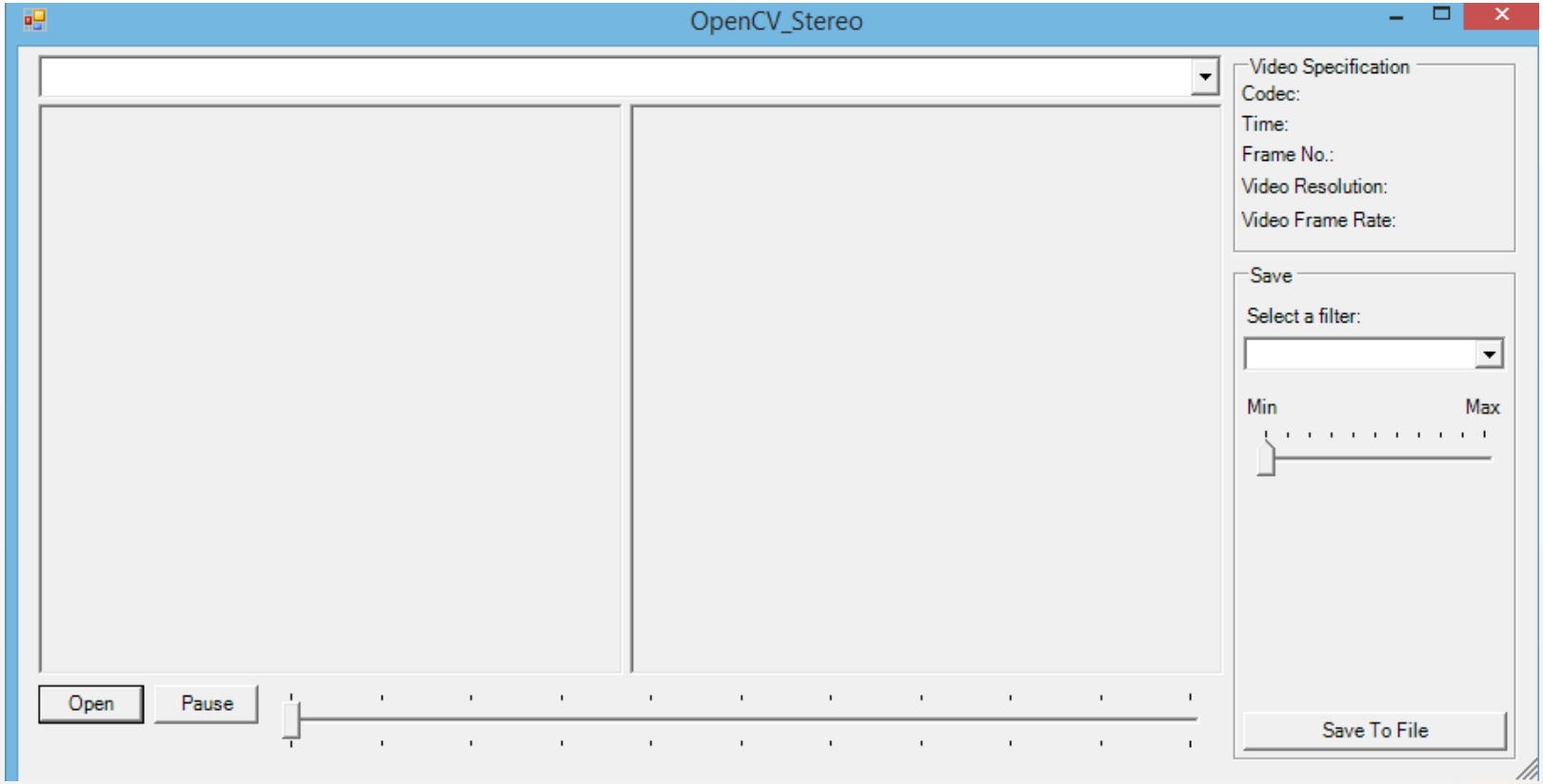
```
else if (comboBox1->Text == "Capture From File")
{
    btnPlay->Text = "Stop";
    openFileDialog1->Filter = "AVI files (*.avi)|*.txt|All files (*.*)|*.*";
    openFileDialog1->FilterIndex = 2;
    openFileDialog1->RestoreDirectory = true;
    openFileDialog1->FileName = "";
    bool fileSelected = openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK;
    if (fileSelected)
    {
        pause = false;
        char *fileName = (char*)System::Runtime::InteropServices::Marshal::StringToHGlobalAnsi(openFileDialog1->Fil
        OpenCVStereo::StereoVideo::openVideoFromFile(fileName);

        trackBar1->Minimum = 0;
        trackBar1->Maximum = (int)OpenCVStereo::StereoVideo::getProperty(CV_CAP_PROP_FRAME_COUNT);
        trackBar1->Visible = ((int)OpenCVStereo::StereoVideo::getProperty(CV_CAP_PROP_FRAME_COUNT) > 1);
        groupSpecification->Visible = ((int)OpenCVStereo::StereoVideo::getProperty(CV_CAP_PROP_FRAME_COUNT) > 1);
        btnPlay->Text = "Stop";
        timer1->Start();
    }
}
```

Ferramenta – Tocando Video

```
//Include da biblioteca OpenCVStereo
//Necessário para fazer uso das funções da biblioteca
#include "opencvStereo.h"

...
private: System::Void timer1_Tick(System::Object^  sender,
System::EventArgs^  e){
    ...
    //captura o proximo quadro
    OpenCVStereo::StereoVideo::getNextFrame();
    ...
    //Se algum quadro foi capturado
    if (OpenCVStereo::StereoVideo::currentFrame != NULL)
    {
        OpenCVStereo::StereoFrame splittedFrame =
OpenCVStereo::StereoTools::splitFrames(nullptr);

        Mat left =
OpenCVStereo::StereoTools::applySelectedFilter(splittedFrame.left);
        Mat right =
OpenCVStereo::StereoTools::applySelectedFilter(splittedFrame.right);
        //Joga os quadros capturados nas picture boxes
        setFrameToPicture(left, pbMainVideo);
        setFrameToPicture(right, pbRightVideo);
        ...
    }
}
```

Operacionalidade da Ferramenta

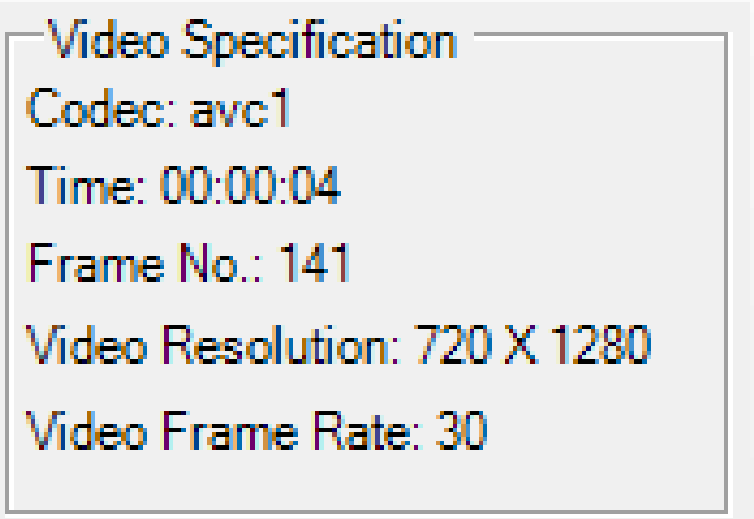
- Tela inicial da ferramenta



Operacionalidade – Propriedades Vídeo

Propriedades do vídeo podem ser extraídas diretamente da biblioteca.

```
double codec_double = OpenCVStereo::StereoVideo::getProperty(CV_CAP_PROP_FOURCC);
label1->Text = "Codec: " + System::Text::Encoding::UTF8->GetString(BitConverter::GetBytes((int)codec_double));
label2->Text = "Time: " + (TimeSpan::FromMilliseconds(OpenCVStereo::StereoVideo::getProperty(CV_CAP_PROP_POS_MSEC)).
label3->Text = "Frame No.: " + (int)OpenCVStereo::StereoVideo::getProperty(CV_CAP_PROP_POS_FRAMES);
label4->Text = "Video Resolution: " + (int)OpenCVStereo::StereoVideo::getProperty(CV_CAP_PROP_FRAME_HEIGHT) + " X "
    + (int)OpenCVStereo::StereoVideo::getProperty(CV_CAP_PROP_FRAME_WIDTH);
label5->Text = "Video Frame Rate: " + (int)Math::Round(OpenCVStereo::StereoVideo::getProperty(CV_CAP_PROP_FPS));
```



Exemplo Ferramentas

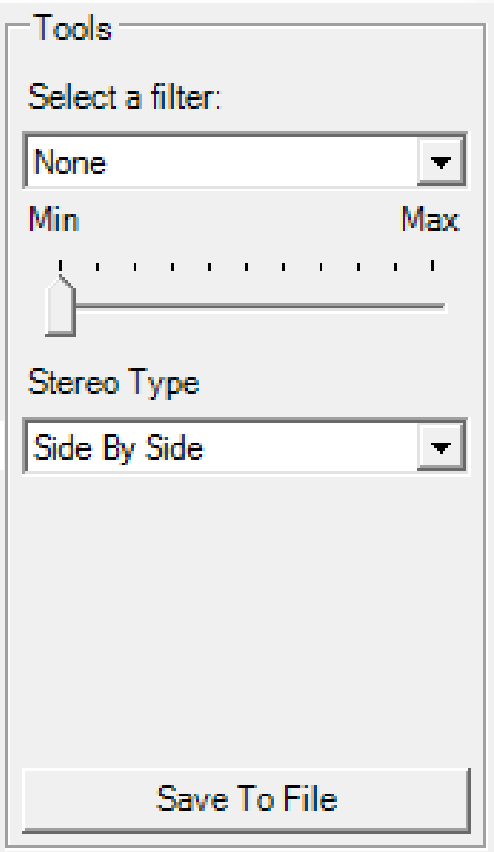
Propriedades do vídeo podem ser extraídas diretamente da biblioteca.

```
StereoTools::setSelectedFilter((OpenCVStereo::Filter) comboFilter->SelectedIndex);
```

```
StereoTools::filterIntensity = trackFilter->Value;
```

```
StereoTools::setStereoType((OpenCVStereo::StereoType) comboStereoType->SelectedIndex);
```

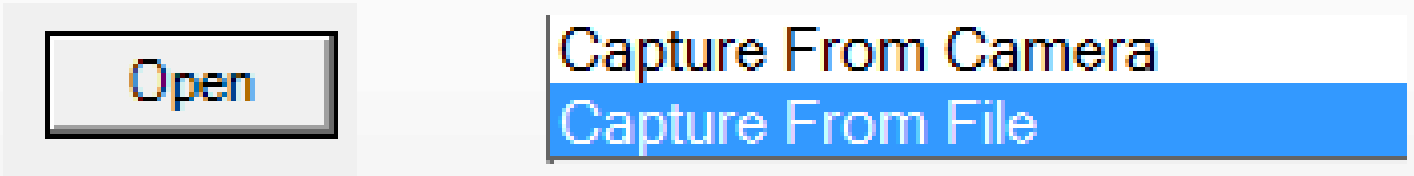
```
StereoVideo::saveVideo(fileName);
```



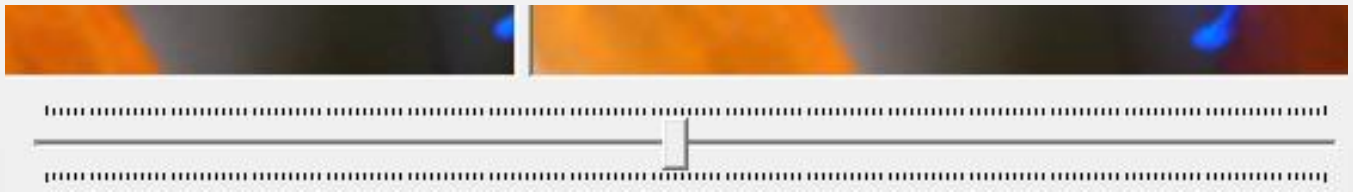
Exemplo - Carregando Texturas

```
OpenCVStereo::StereoVideo::openVideoFromCamera();
```

```
OpenCVStereo::StereoVideo::openVideoFromFile(fileName);
```



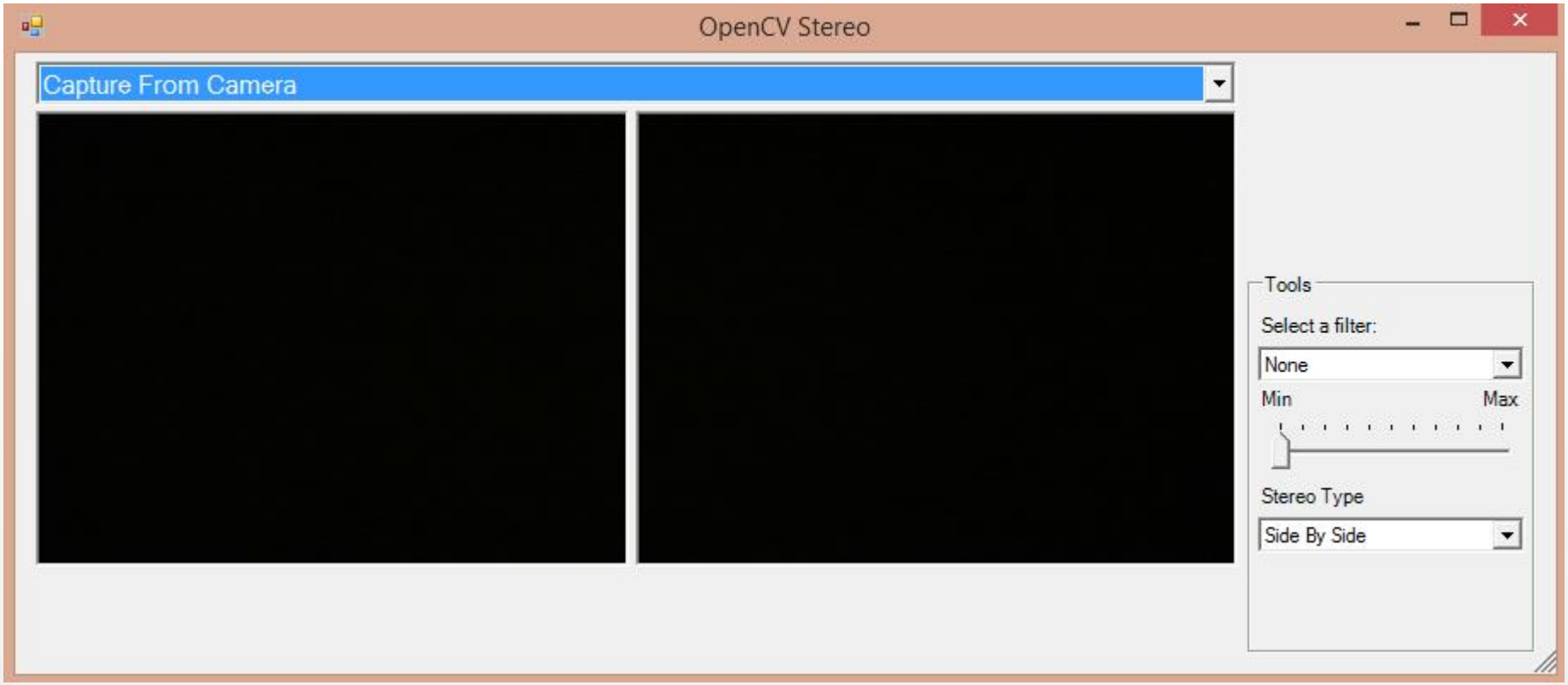

```
trackBar1->Value = (int)OpenCVStereo::StereoVideo::getProperty(CV_CAP_PROP_POS_FRAMES);
```



Operacionalidade da Ferramenta

- Reprodução de conteúdo do arquivo

Operacionalidade da Ferramenta



Resultados e Discussões

- Biblioteca funcional;
  - Abstrai processos de decodificação e codificação de conteúdo estereoscópico.
  - Disponibiliza funções para auxiliar no desenvolvimento de aplicações estéreo.

- Ferramenta:
  - Permite testar funcionalidades da biblioteca;
  - Disponibiliza exemplos de uso e melhores práticas da biblioteca.
  - Uso da memoria reproduzindo um vídeo em 1280x720 foi de 42.960K

Comparação com correlatos

Características / Nome dos trabalhos	Biblioteca Referida	OPENSTEREO	Cálculo de volume efetivo usando estereoscopia.	Protótipo de um ambiente de visualização
Suporte a câmeras estéreo	Sim	Não	Não (usa uma única câmera com auxilio de esteira)	Não
Suporte a abertura de arquivos estéreo	Sim	Não	Não	Sim
Suporte a captura de ambiente gráfico	Não	Sim	Não	Não
Saída Anaglyph	Não*	Sim	(Não gera imagens)	Sim
Saída Side By Side	Sim	Não	(Não gera imagens)	Não
Saída Over/Under	Sim	Não	(Não gera imagens)	Não
Realiza cálculos de volume	Não*	Não	Sim	Não

Conclusões

- Biblioteca implementada.
- Ferramenta fazendo uso da biblioteca.
- Passos para implementação foram explicados para facilitar a reprodução e extensibilidade desse trabalho.
- Código disponibilizado abertamente em um repositório no serviço *BitBucket*.

Sugestões

- Otimizar a performance dos algoritmos em imagens e vídeos estéreo;
- Explorar outros algoritmos do OpenCV que não foram implementados neste trabalho;
- Tornar a biblioteca mais independente da plataforma Windows, visando compilar para outras plataformas como Mac e Linux;
- Adicionar suporte a outros formatos de arquivo 3D que não foram explorados neste trabalho.