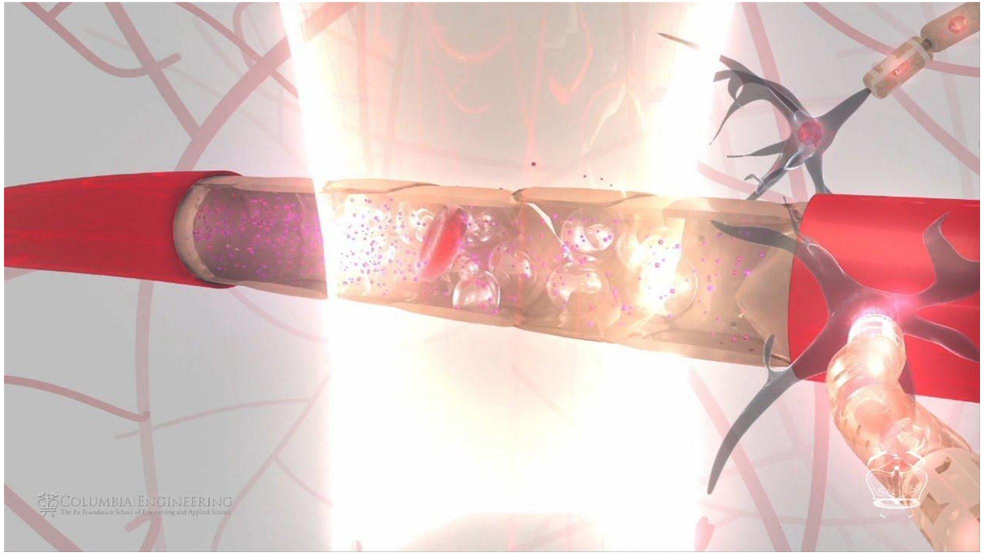
Spatial-temporal Frequency Filtering for Ultrasound image guidance using open-source tools

Andy Xia
Ziqi Zhao

VUIIS

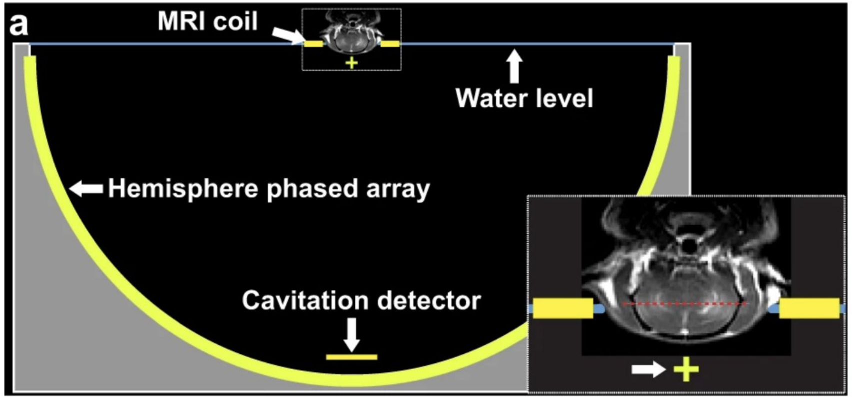
LATI | Laboratory of Acoustic Therapy & Imaging

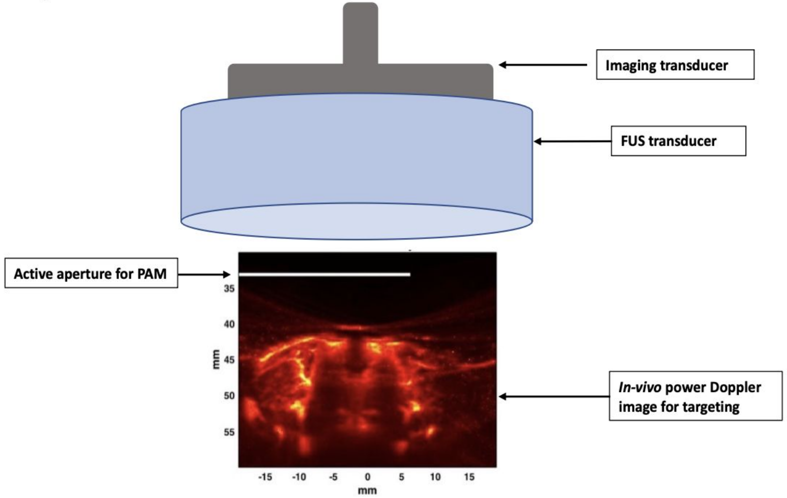
Ultrasound Imaging and Ultrasound Therapy

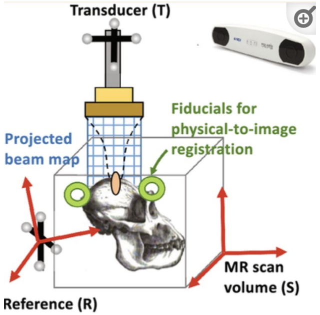


Reference: 1. Hamilton Radiology 2. Konofagou, E(2018)

Image guidance methods for BBBO

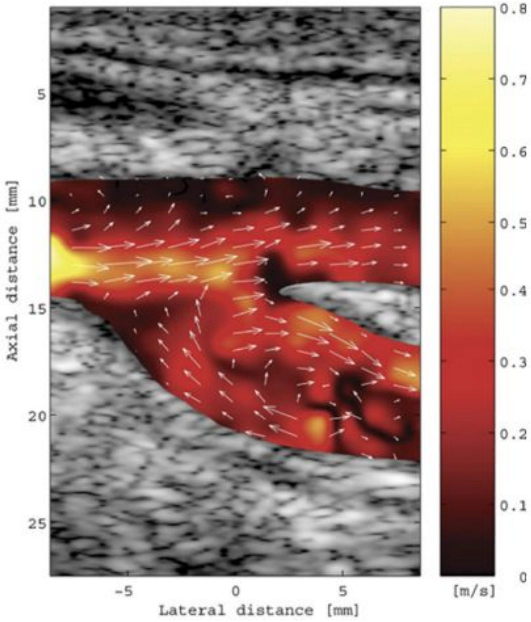






Reference: 1. McDannold, N (2020) 2. Chaplin, V(2019) 3. Singh, A(2022)

Some Doppler

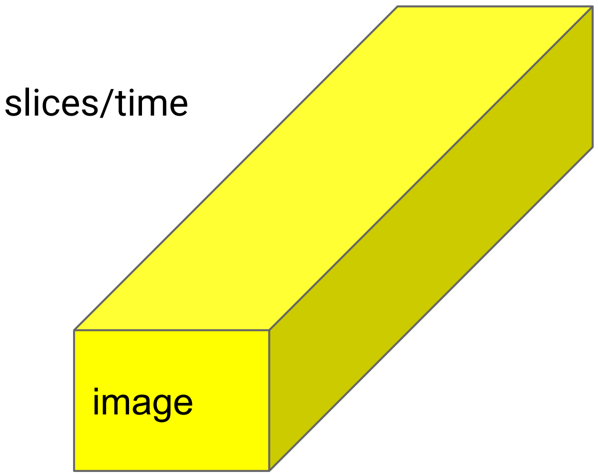


Reference: 1. Photo: Sense Jan van der Molen 2. Image from: Byram, B

Our Initial Data

- Data source: IEEE_IUS 2020 Super-resolution short course
- Rat brain data
- Data type: `complex double`, we use `single`
- File type: `.mat` Convert to `.nii` and `.txt`
- Demodulated/Beamformed Data
- Speckles are microbubbles, the ultrasound contrast agent
- Matrix info: dense matrix with `78*128*800` size

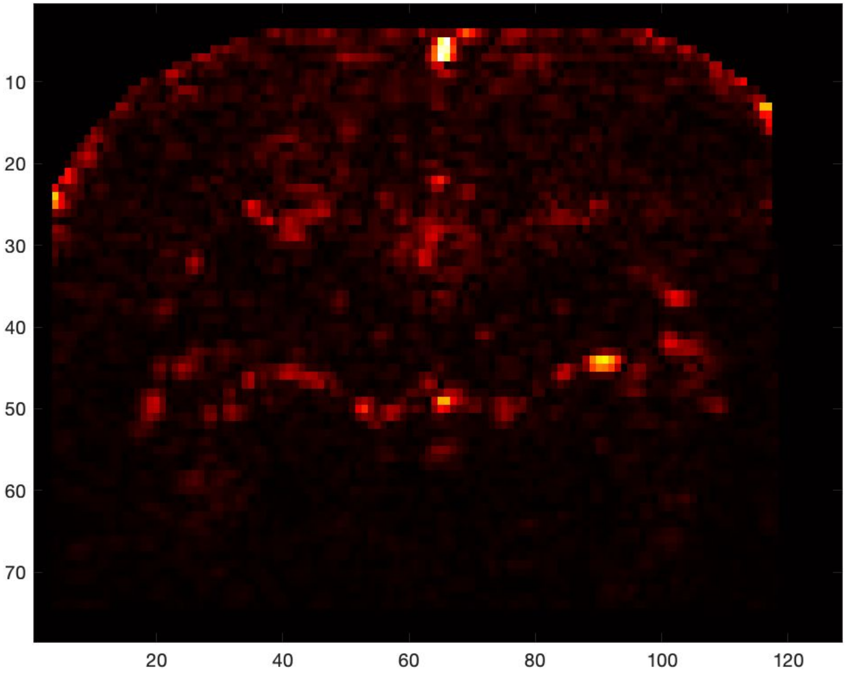
rows*cols*slices



Our Initial Data

- Data source: IEEE_IUS 2020 Super-resolution short course
- Rat brain data
- Data type: `complex double`, we use `single`
- File type: `.mat` Convert to `.nii` and `.txt`
- Demodulated/Beamformed Data
- Speckles are microbubbles, the ultrasound contrast agent
- Matrix info: dense matrix with `78*128*800` size

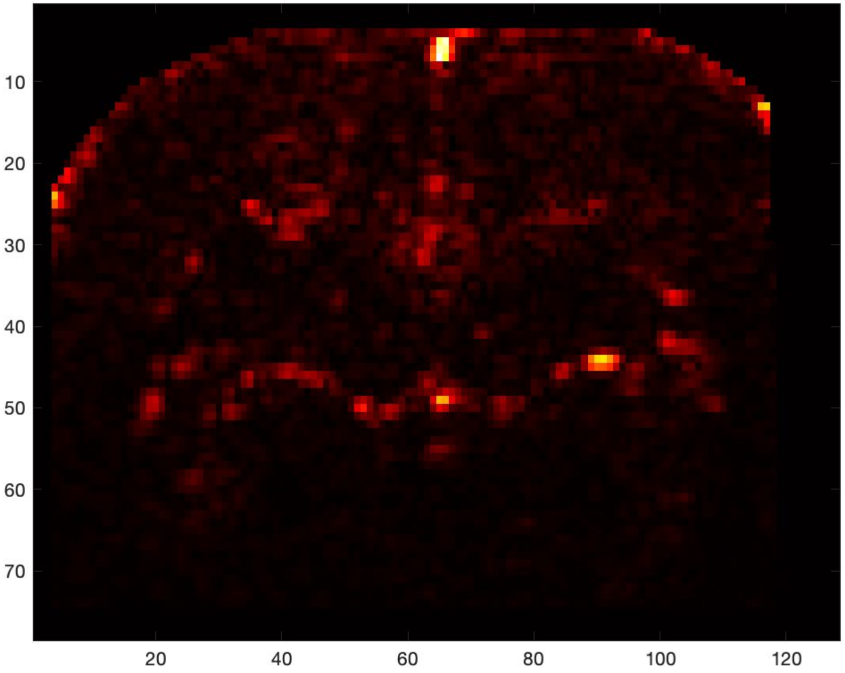
rows*cols*slices



Our Initial Data - Accumulated

- Data source: IEEE_IUS 2020 Super-resolution short course
- Rat brain data
- Data type: *`complex double`*, we use *`single`*
- File type: `.mat` Convert to `.nii` and `.txt`
- Demodulated/Beamformed Data
- Speckles are microbubbles, the ultrasound contrast agent
- Matrix info: dense matrix with `78*128*800` size

rows*cols*slices

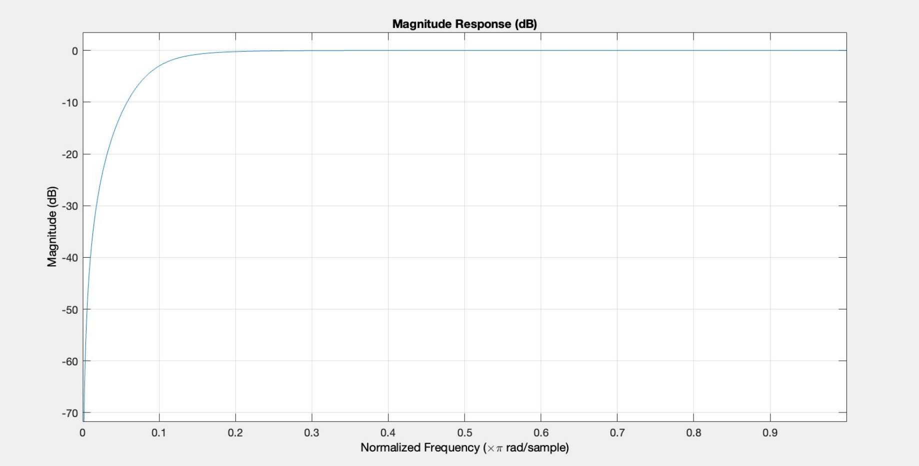


Why C++ and open-source tool

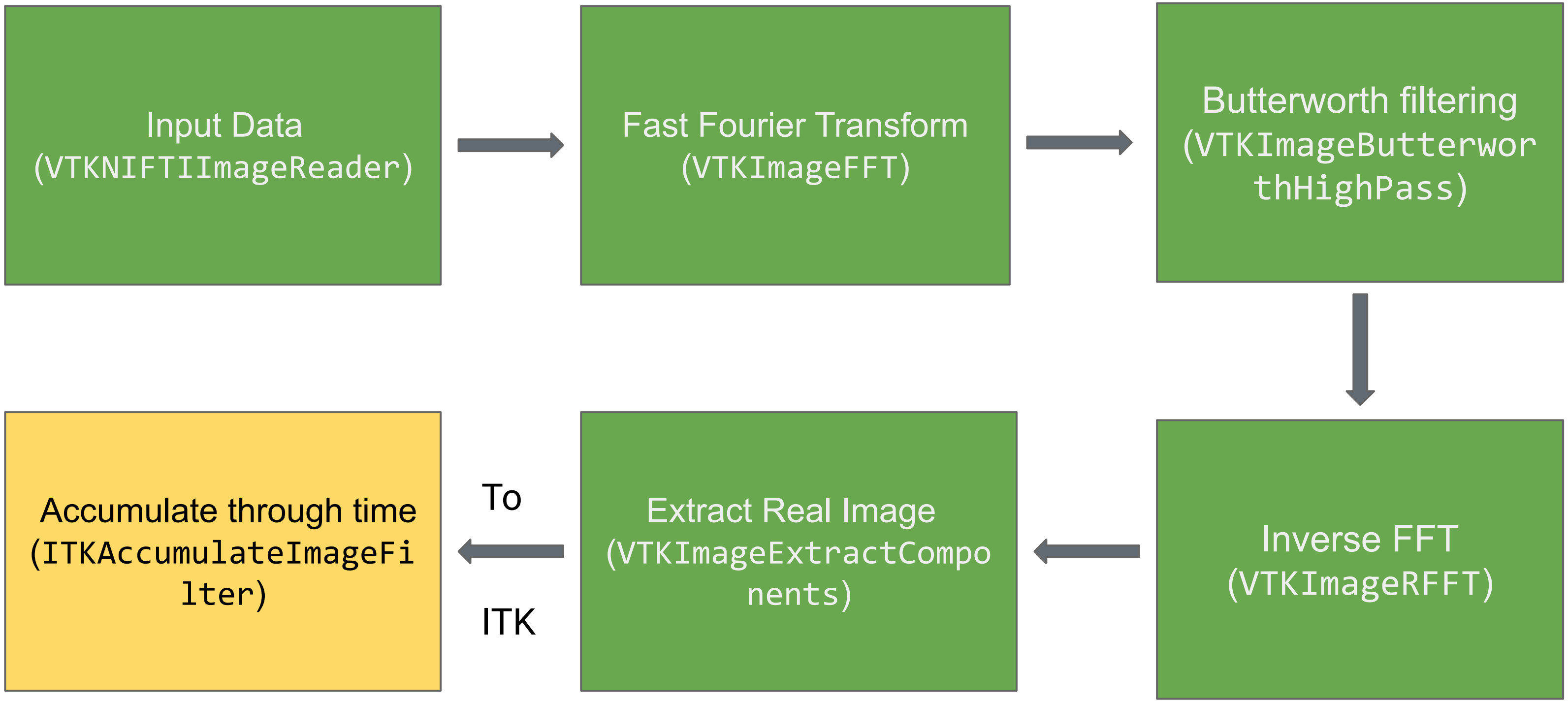
- We can build customize imaging/therapy system that doesn’t based on MATLAB.
- Practice what we learned in class.
- Explore possibilities to compute faster.
- Controllability and Transparency

IIR frequency filtering (Butterworth)

- 2nd order high pass Butterworth filter.
- Passband at 0.1* sampling frequency.
- Filter out tissue signal.
- Maintain major blood flow signals.
- Bandpass could also filter high frequency noise.



Implementation of butterworth filter



Input Data
(VTKNIFTIImageReader)

Fast Fourier Transform
(VTKImageFFT)

Butterworth filtering
(VTKImageButterworthHighPass)

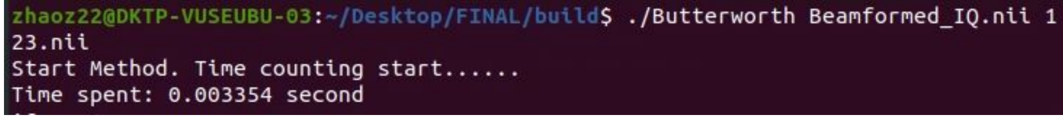
Inverse FFT
(VTKImageRFFT)

Extract Real Image
(VTKImageExtractComponents)

To ITK

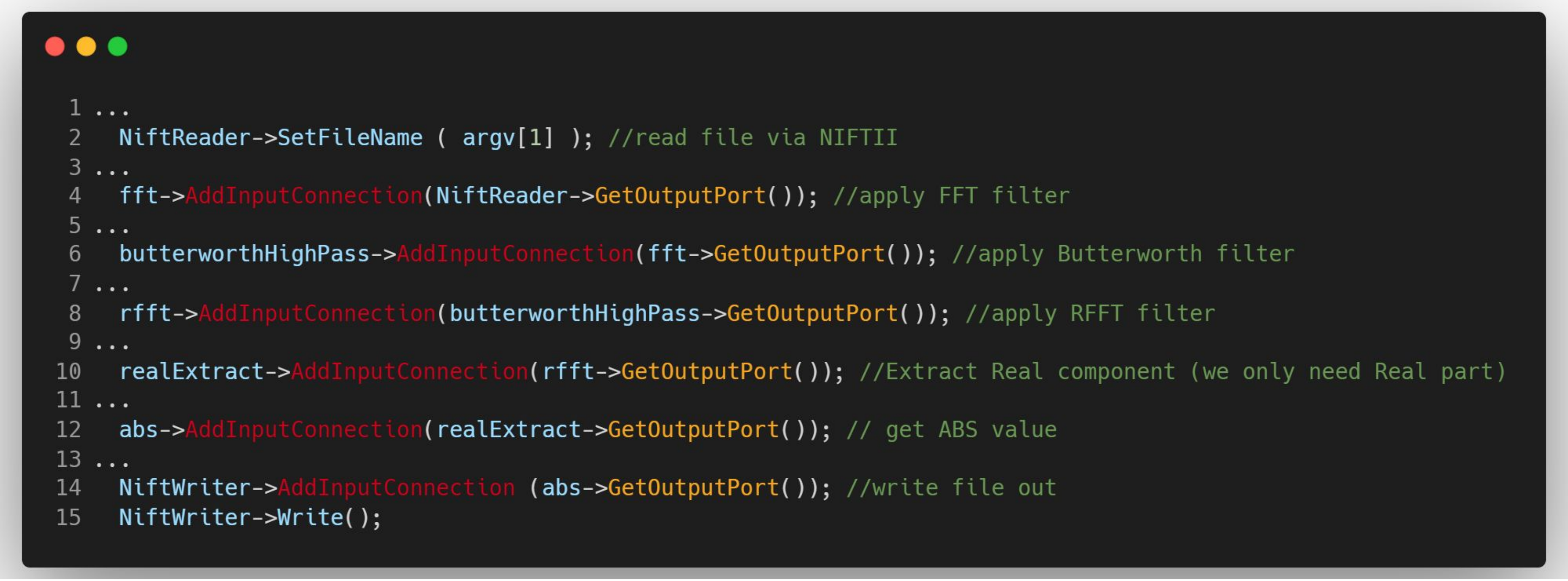
Accumulate through time
(ITKAccumulateImageFilter)

Use Pipeline in VTK



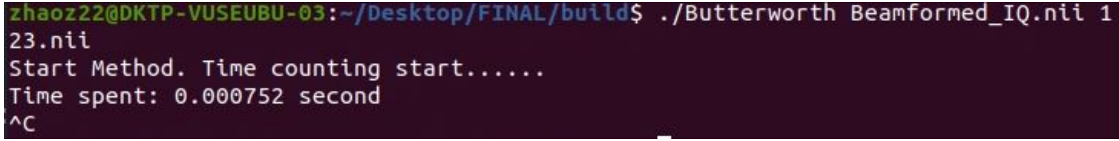
```
zhaoz22@DKTP-VUSEUBU-03:~/Desktop/FINAL/build$ ./Butterworth Beamformed_IQ.nii 1
23.nii
Start Method. Time counting start......
Time spent: 0.003354 second
```

Figure.1 SetInputData + GetOutPut



```
...
  NiftReader->SetFileName ( argv[1] ); //read file via NIFTII
...
  fft->AddInputConnection(NiftReader->GetOutputPort()); //apply FFT filter
...
  butterworthHighPass->AddInputConnection(fft->GetOutputPort()); //apply Butterworth filter
...
  rfft->AddInputConnection(butterworthHighPass->GetOutputPort()); //apply RFFT filter
...
  realExtract->AddInputConnection(rfft->GetOutputPort()); //Extract Real component (we only need Real part)
...
  abs->AddInputConnection(realExtract->GetOutputPort()); // get ABS value
...
  NiftWriter->AddInputConnection (abs->GetOutputPort()); //write file out
  NiftWriter->Write();
```

Figure.2 Code skeleton



```
zhaoz22@DKTP-VUSEUBU-03:~/Desktop/FINAL/build$ ./Butterworth Beamformed_IQ.nii 1
23.nii
Start Method. Time counting start......
Time spent: 0.000752 second
^C
```

Figure.3 AddInputConnection + GetOutputPort

Intermediate Result (Slice 400)

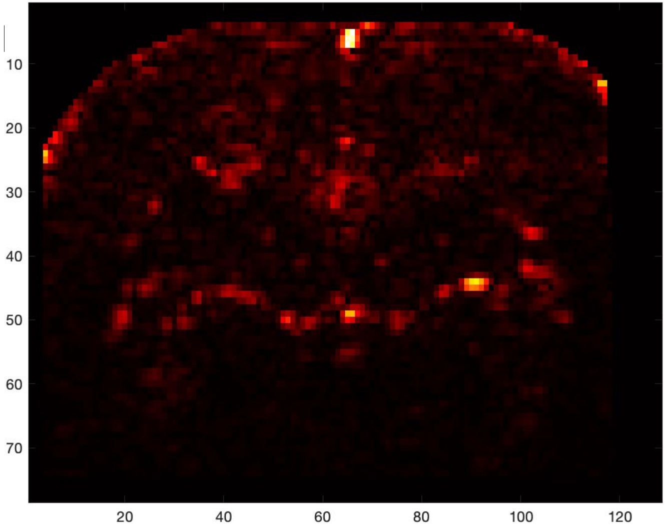


Figure.1 Unfiltered

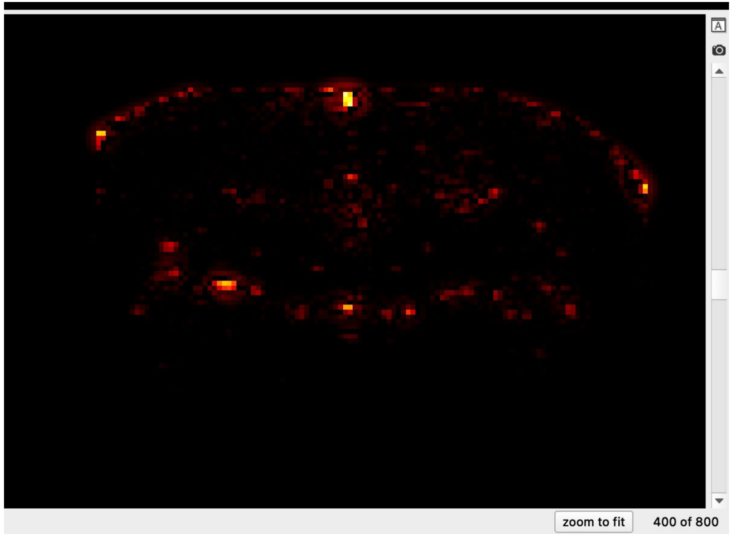
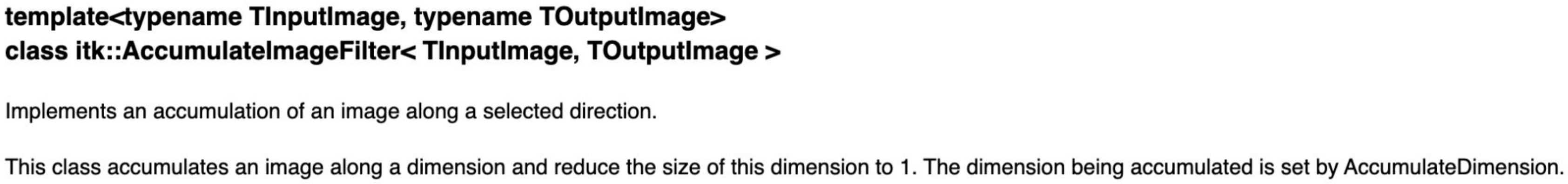


Figure.2 Butterworth filtered
(Single 400th slice)

Sum across the slices



template<typename TInputImage, typename TOutputImage>
class itk::AccumulateImageFilter< TInputImage, TOutputImage >

Implements an accumulation of an image along a selected direction.

This class accumulates an image along a dimension and reduce the size of this dimension to 1. The dimension being accumulated is set by AccumulateDimension.

- Barriers and Methods we try:
- `ITKVTKGlue`
    - Failed, need to re-compile whole VTK.
- Independent ITK program
- We suspect difference in Filter design

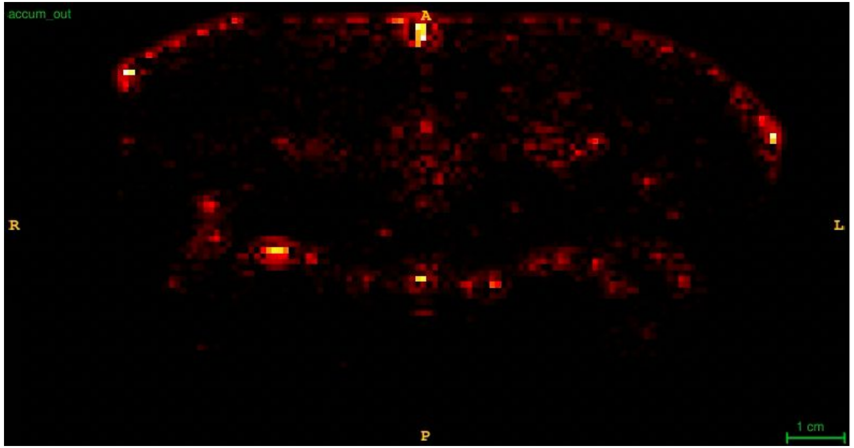
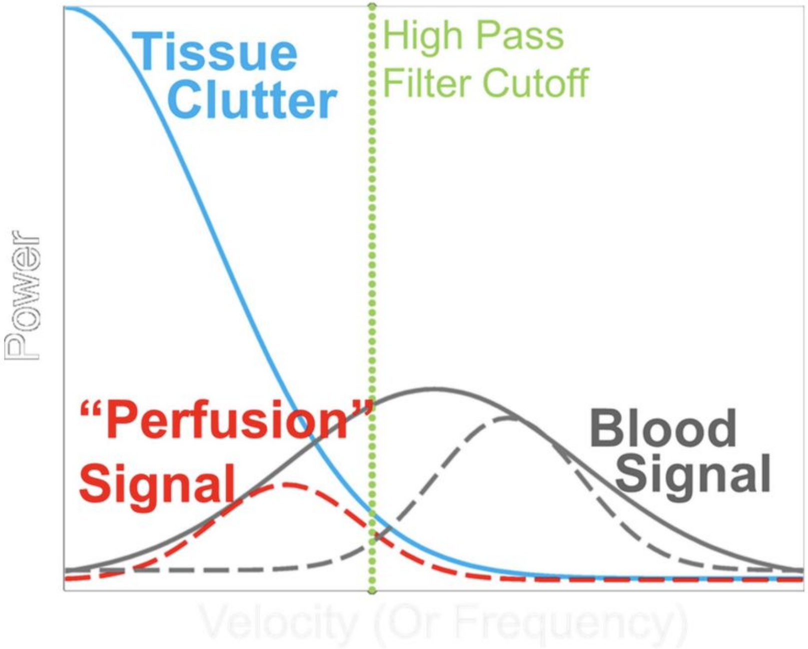


Figure.1 Accumulated Butterworth filtered image

Why IIR filtering is not good enough

- Microvasculature have comparatively slow flowing speed
- Picking a single frequency cutoff can be tricky
- Combining multiple filters might make it better, but significant lost of information is still expected
- Introducing Singular Value Decomposition(SVD) Method



Reference: Graph source: Byram, B

Covariance Estimation

Considering the spatial temporal characteristics of blood and tissue signals

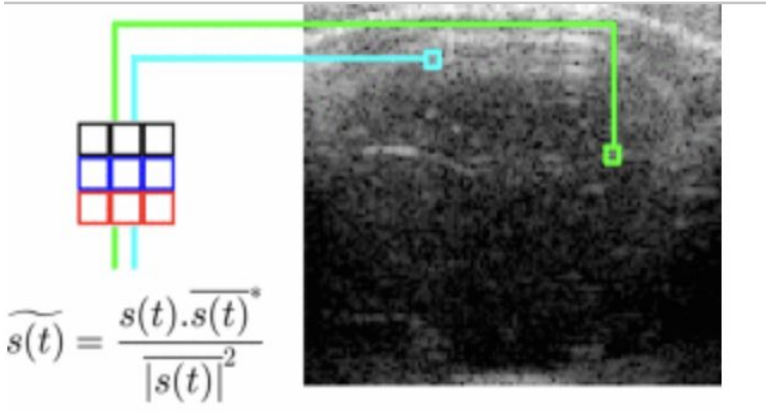
$$M = U * S * V'$$

M is Casaroti Matrix -> Reshape our image

U -> Temporal singular vector

S -> Diagonal singular value matrix

V -> Spatial singular vector



$$\widetilde{s(t)} = \frac{s(t).\overline{s(t)}^*}{\overline{|s(t)|}^2}$$

Reference: Demené, C et al. (2015)

SVD Filtering Method

High singular value -> strong coherence -> static low f signal -> tissue

Low singular value -> weak coherence -> unstable high f signal -> blood flow

SVD -> filtering: remove High value in S matrix with 0

S matrix is sorted from high to low values

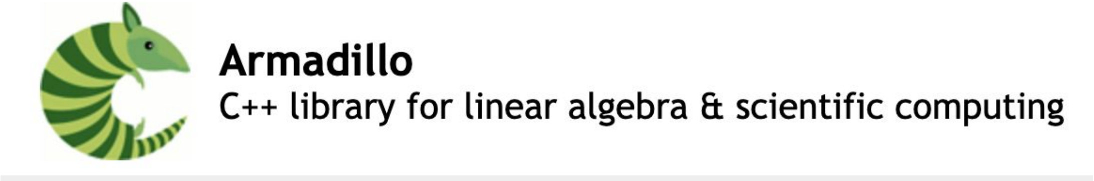
Reconstruct the image matrix using the same equation

Using ITK/VTK?

- VTK: Don’t have **`Matrix`** type.
  - **`VTKPCAStatistics`** seems to be able to compute eigenvectors and decomposition.
  - Request line by line **`VTKdoubleArray`** as input.
  - Not designed for processing large 2D/3D image matrix.
- ITK: Have **`Matrix`**, and even an **`ITKSymmetricEigenAnalysis`** class.
  - We tried read from image / text, don’t know the correct way to import the correct matrix externally.

Explore other open-source tool designed for linear algebra

Eigen

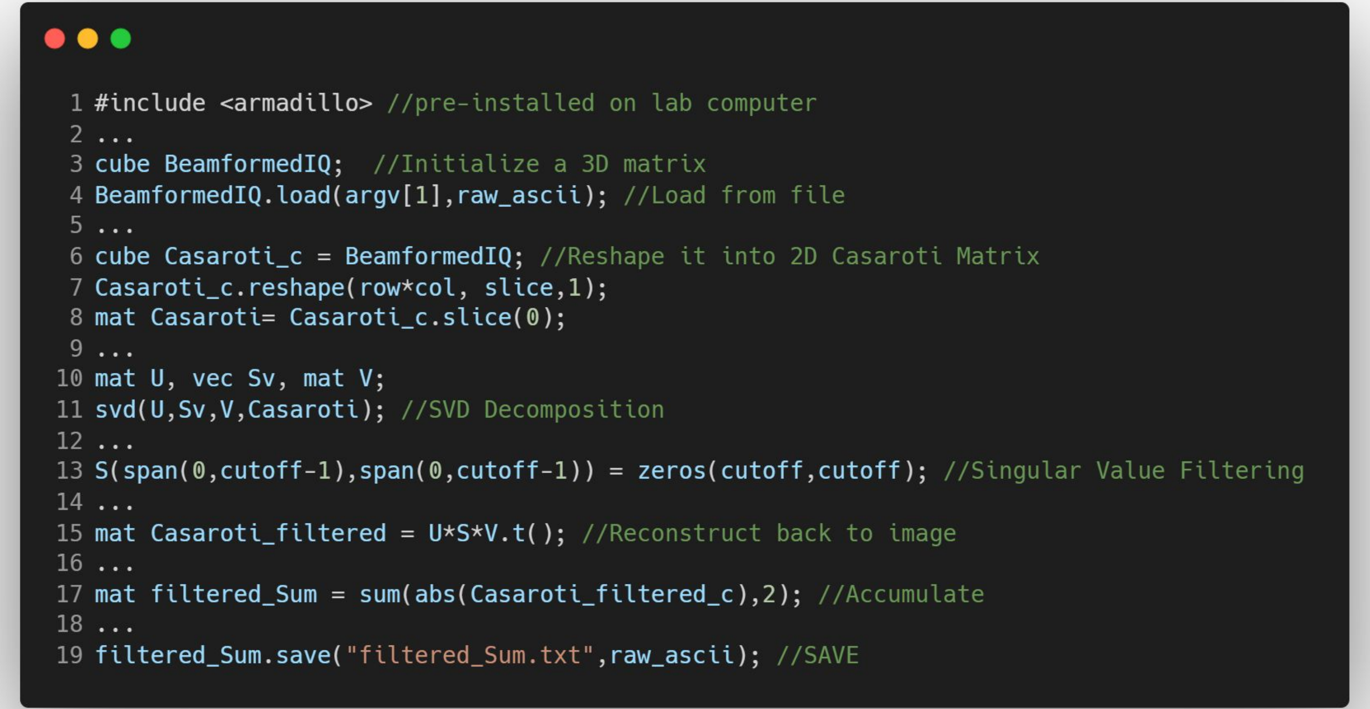


Armadillo

C++ library for linear algebra & scientific computing

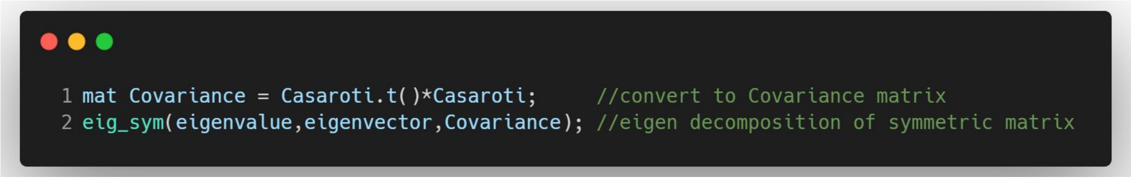
Implement Armadillo

- It is pre-installed on our Engineering Ubuntu Lab machine (Yah!)



```
#include <armadillo> //pre-installed on lab computer
...
cube BeamformedIQ;  //Initialize a 3D matrix
BeamformedIQ.load(argv[1],raw_ascii); //Load from file
...
cube Casaroti_c = BeamformedIQ; //Reshape it into 2D Casaroti Matrix
Casaroti_c.reshape(row*col, slice,1);
mat Casaroti= Casaroti_c.slice(0);
...
mat U, vec Sv, mat V;
svd(U,Sv,V,Casaroti); //SVD Decomposition
...
S(span(0,cutoff-1),span(0,cutoff-1)) = zeros(cutoff,cutoff); //Singular Value Filtering
...
mat Casaroti_filtered = U*S*V.t(); //Reconstruct back to image
...
mat filtered_Sum = sum(abs(Casaroti_filtered_c),2); //Accumulate
...
filtered_Sum.save("filtered_Sum.txt",raw_ascii); //SAVE
```

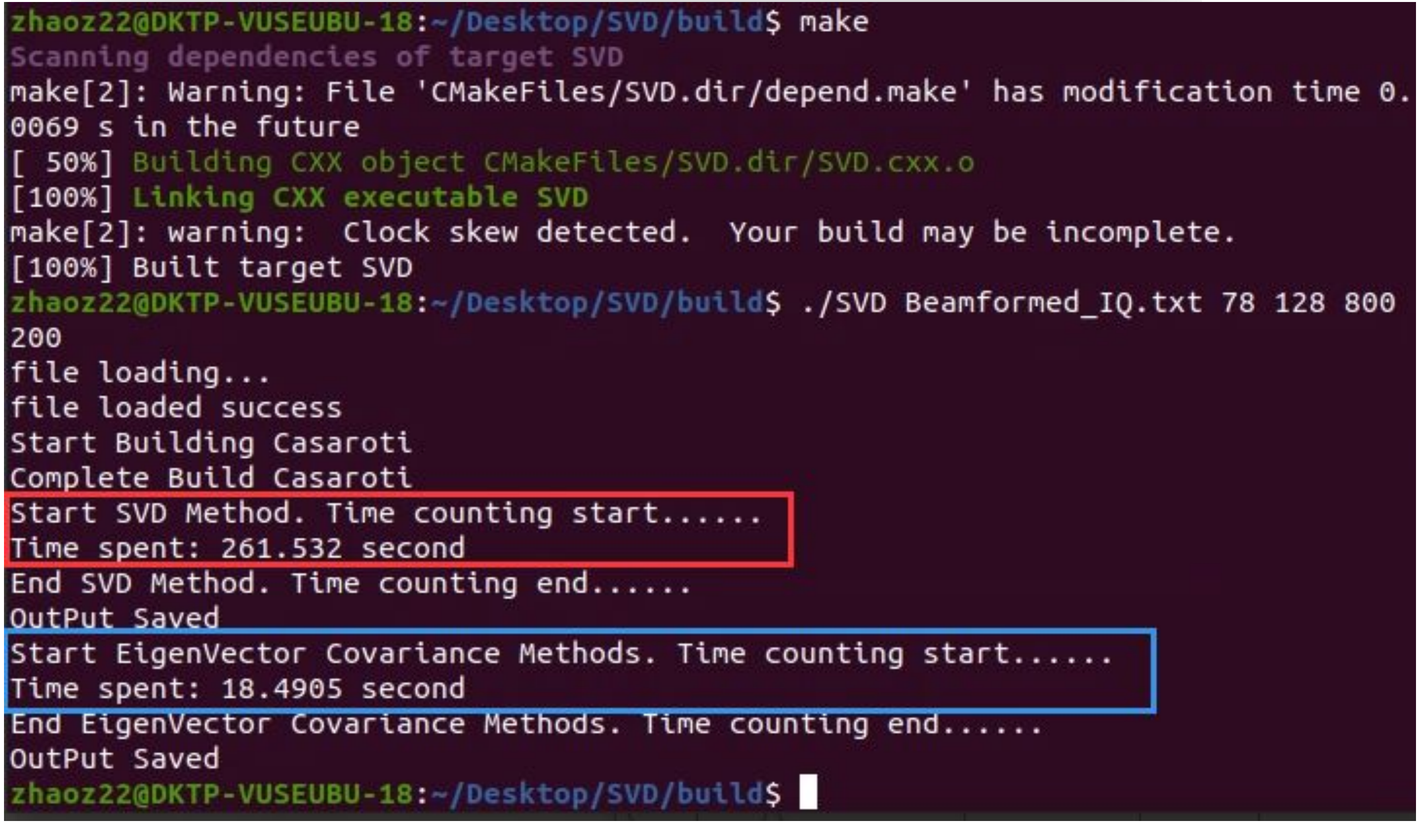
Speed comparison between two methods

- 

  ```
  mat Covariance = Casaroti.t()*Casaroti;     //convert to Covariance matrix
  eig_sym(eigenvalue,eigenvector,Covariance); //eigen decomposition of symmetric matrix
  ```

- Microbubbles usually last few minutes before it dissolve/excrete
- Raw Casaroti Matrix is `9984*800`
- The SVD takes significant amount of computing time on big matrix like this
- The Covariance Matrix (*Casaroti* * *Casaroti* (transpose)) is `800*800`
- `Symmetric Eigenvector Analysis <-> SVD` are **Mathematically equal**

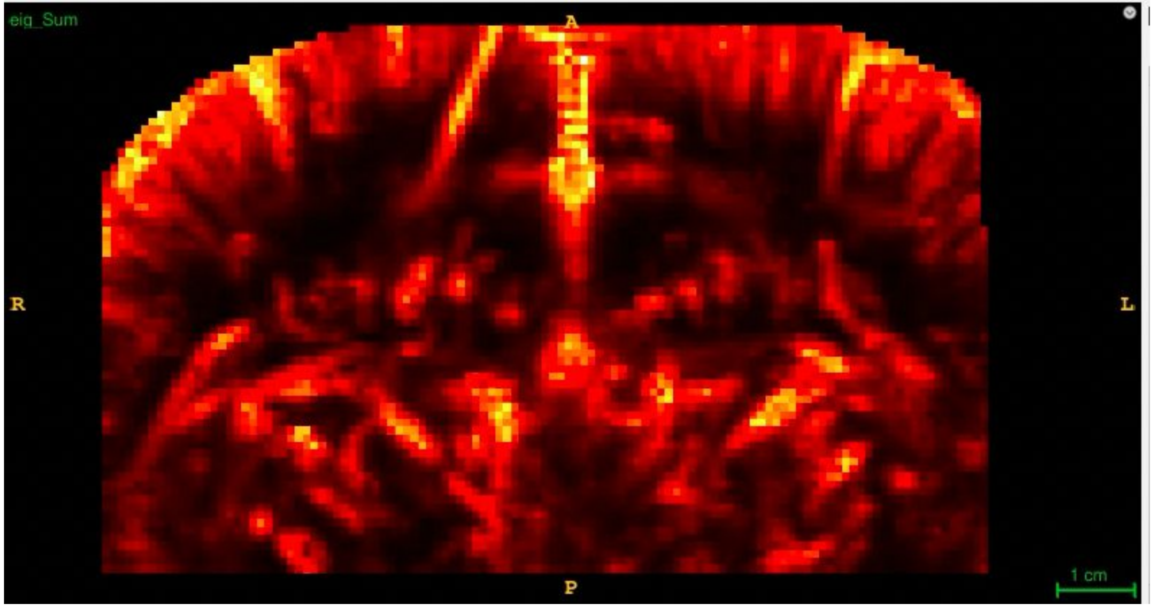
Speed comparison between two methods



```
zhaoz22@DKTP-VUSEUBU-18:~/Desktop/SVD/build$ make
Scanning dependencies of target SVD
make[2]: Warning: File 'CMakeFiles/SVD.dir/depend.make' has modification time 0.
0069 s in the future
[ 50%] Building CXX object CMakeFiles/SVD.dir/SVD.cxx.o
[100%] Linking CXX executable SVD
make[2]: warning:  Clock skew detected.  Your build may be incomplete.
[100%] Built target SVD
zhaoz22@DKTP-VUSEUBU-18:~/Desktop/SVD/build$ ./SVD Beamformed_IQ.txt 78 128 800
200
file loading...
file loaded success
Start Building Casaroti
Complete Build Casaroti
Start SVD Method. Time counting start......
Time spent: 261.532 second
End SVD Method. Time counting end......
OutPut Saved
Start EigenVector Covariance Methods. Time counting start......
Time spent: 18.4905 second
End EigenVector Covariance Methods. Time counting end......
OutPut Saved
zhaoz22@DKTP-VUSEUBU-18:~/Desktop/SVD/build$
```

Write output to image

- Output is now in txt/csv format
- We add a header to convert it to an nrrd file.
- Visualize using itk-snap



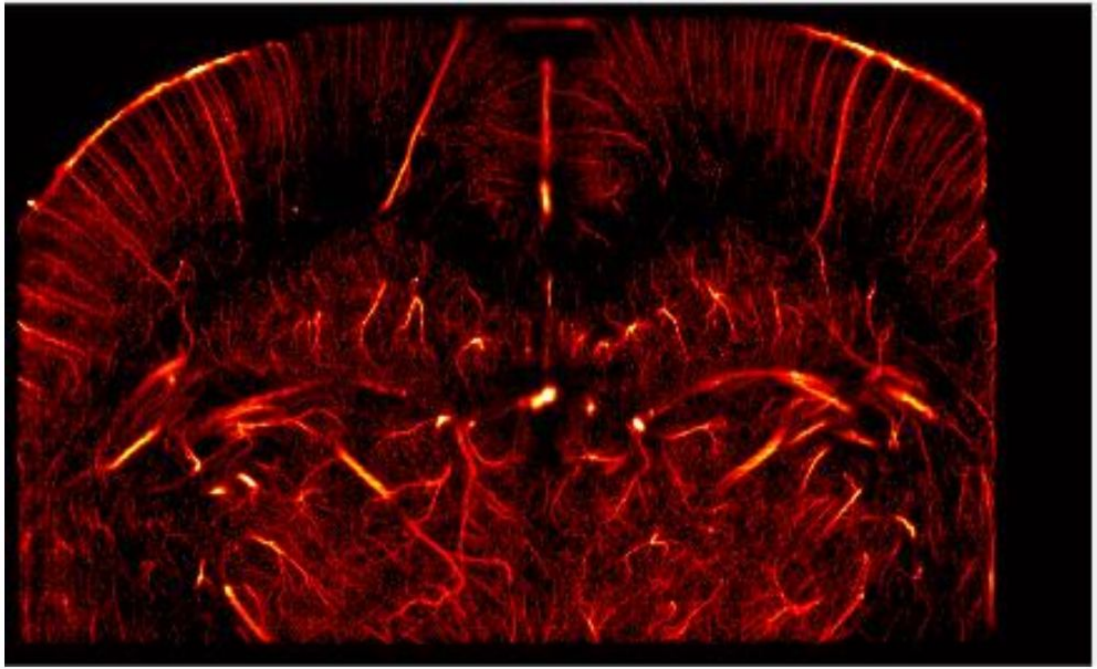
Room of improvement

- The initial data is in complex number, we could potentially process complex numbers in either methods

- Localizing and Tracking individual bubbles trajectory can make better vascular images at a cost of computational speed

Room of improvement

- The init ess comple
- Localizi scular images



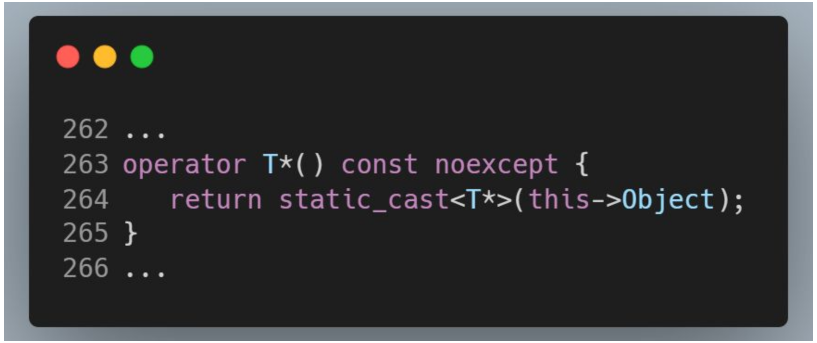
Room of improvement

- The initial data is in complex number, we could potentially process complex numbers in either methods

- Localizing and Tracking individual bubbles trajectory can make better vascular images at a cost of computational speed

- Smoother transition between different open-source tools without creating multiple programs and intermediate files.

A little Tips and Traps of VTK



```
...
operator T*() const noexcept {
    return static_cast<T*>(this->Object);
}
...
```

- Check the version of VTK before using it.
- `vtkSmartPointer & vtkNew`
  - Before ver 8.0, we can not mixed use `vtkSmartPointer & vtkNew.` Why?
  - Implicit type conversion will happen under 4 cases. One of it is when passing parameter into function.
  - C++ will try their best to let function called successfully. So It has a implicit type conversion sequence……
  - In the source code of `vtkNew`, before version 8.0, VTK intentionally disabled auto casting (didn’t provide user-defined conversion function).
  - After 8.0, they enabled that one.
  - Don’t recommend to use `vtkNew` before VTK 8.0, trying to use it after VTK 8.0

Any questions