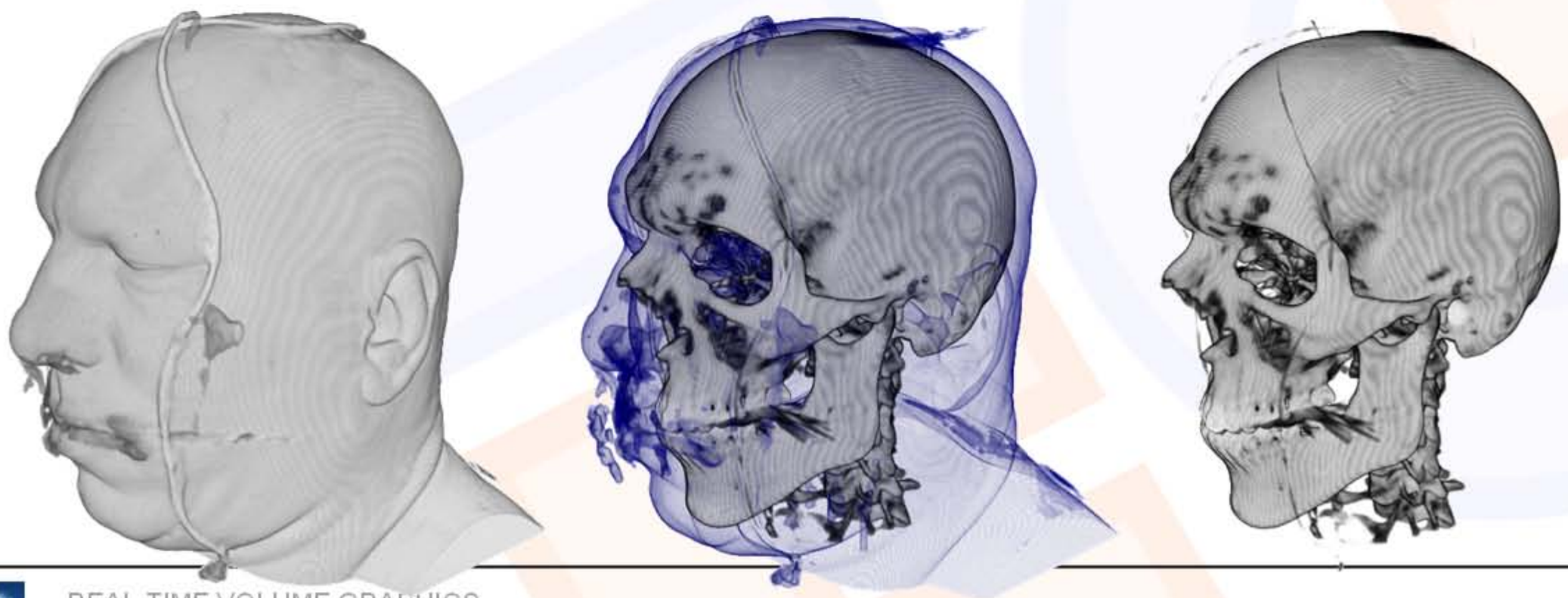
Real-Time Volume Graphics

[05] Transfer Functions

Classification

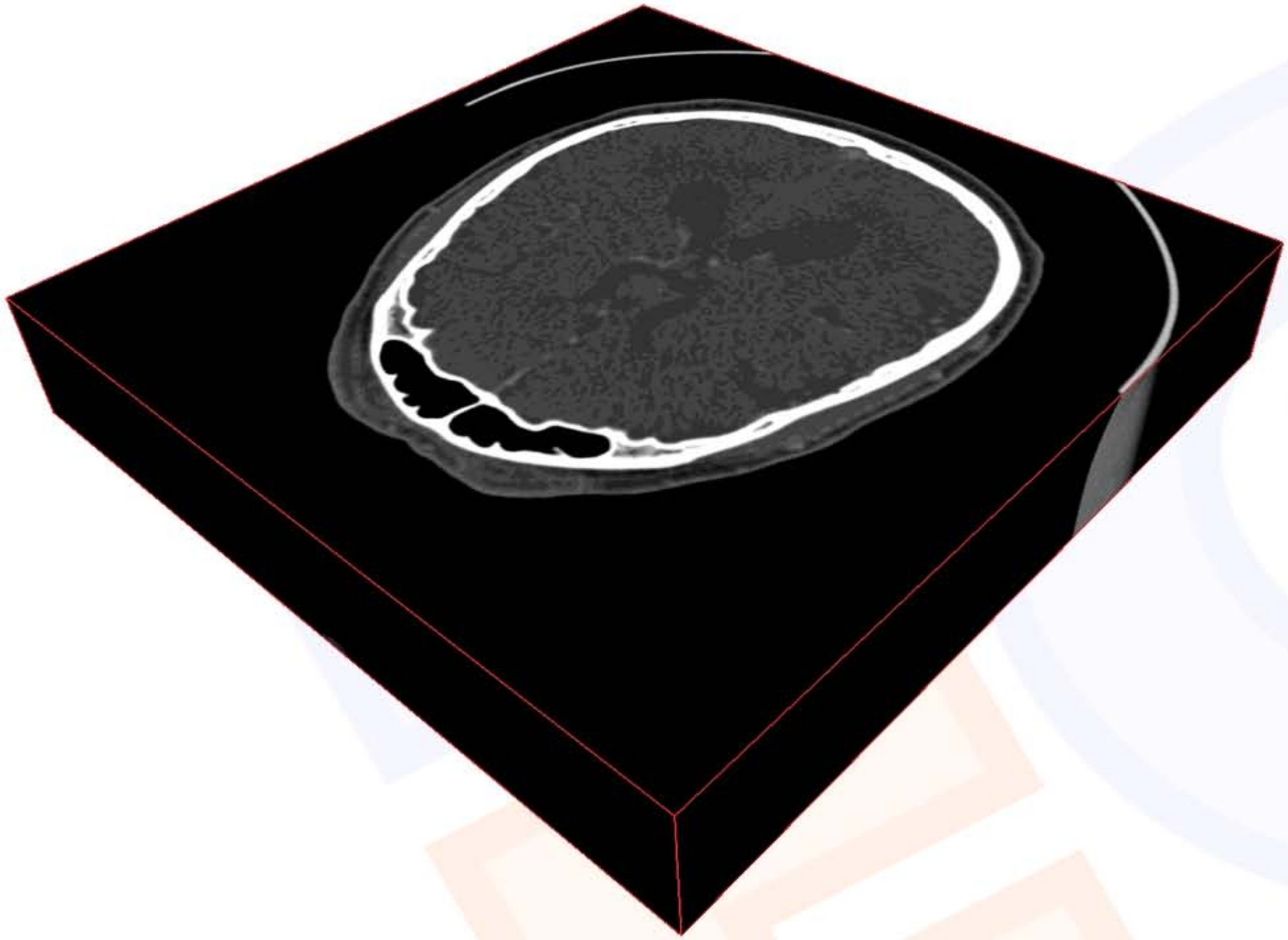
- During Classification the user defines the *„Look"* of the data.
  - Which parts are transparent?
  - Which parts have which color?



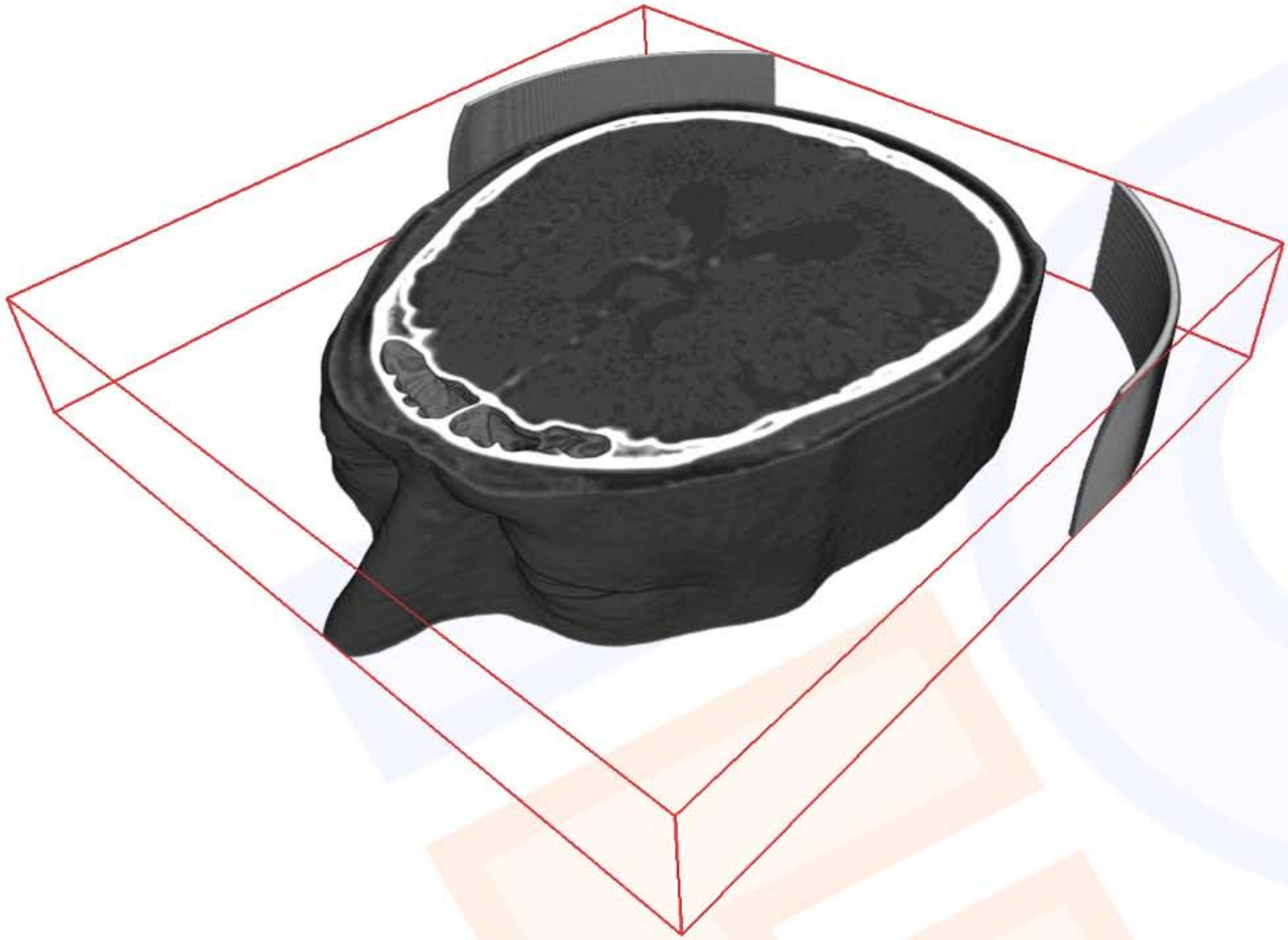
Classification

- During Classification the user defines the *„Look"* of the data.
  - Which parts are transparent?
  - Which parts have which color?
- The user defines a ***Transferfunction***.

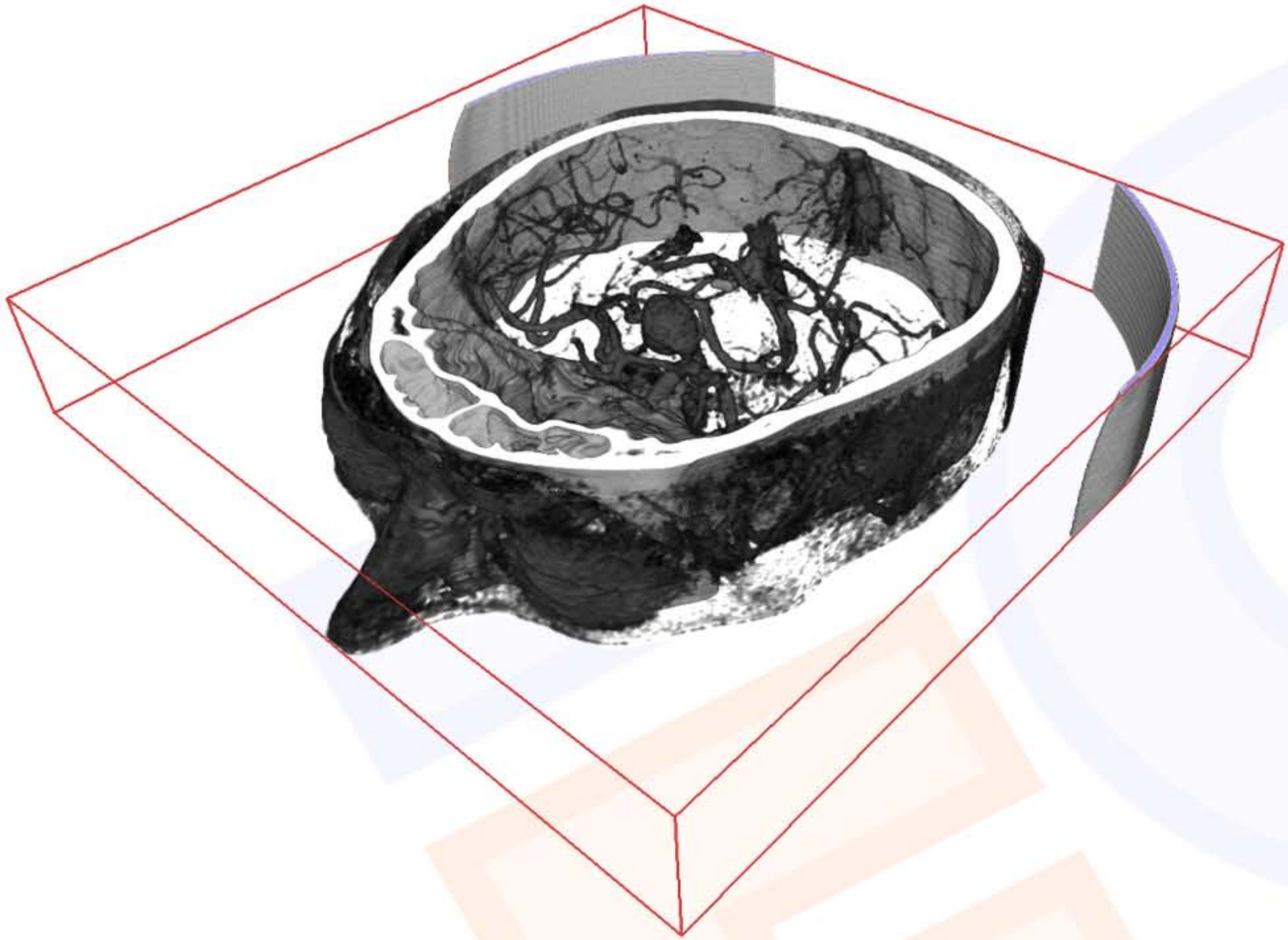
Classification



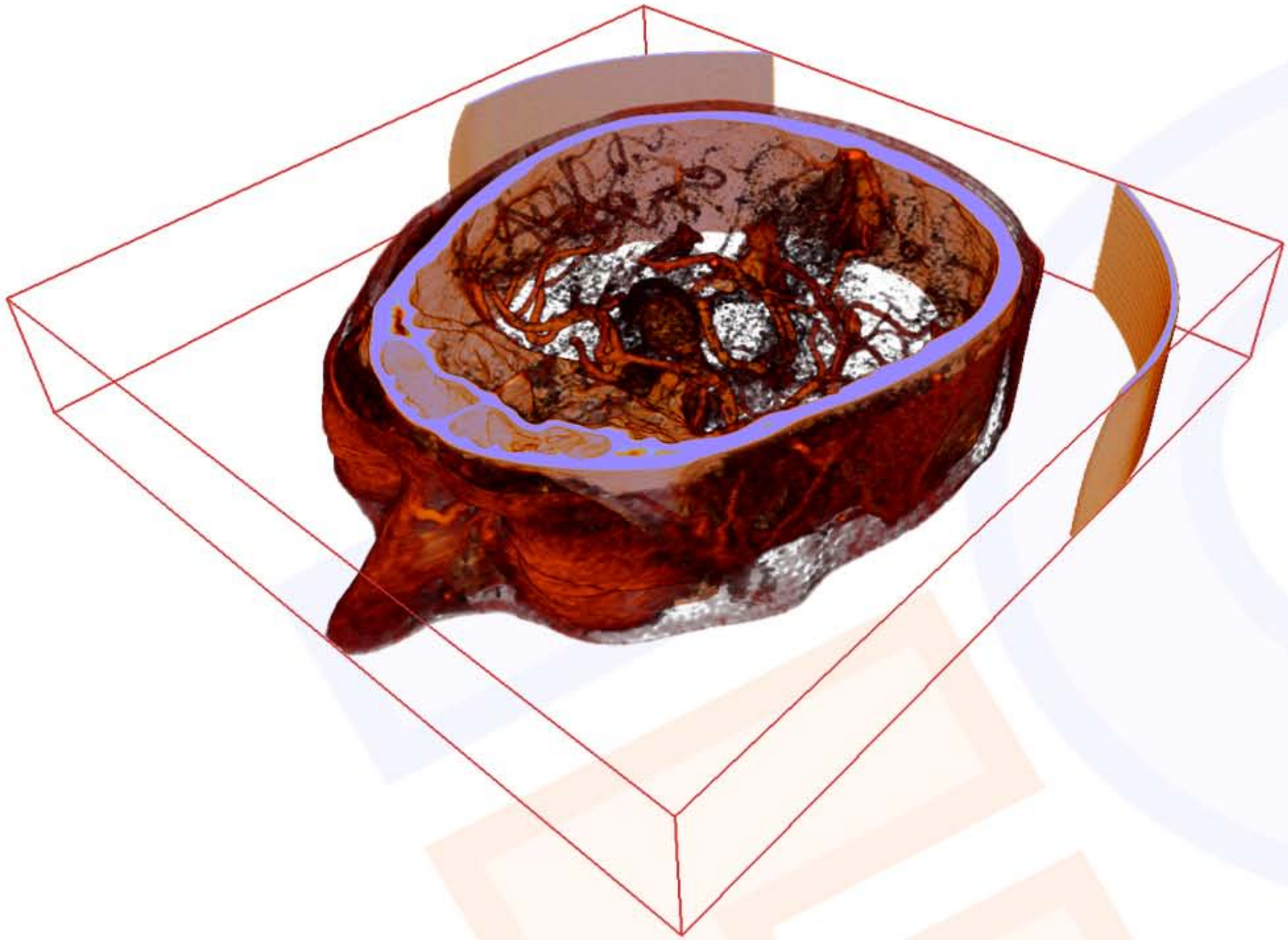
Classification



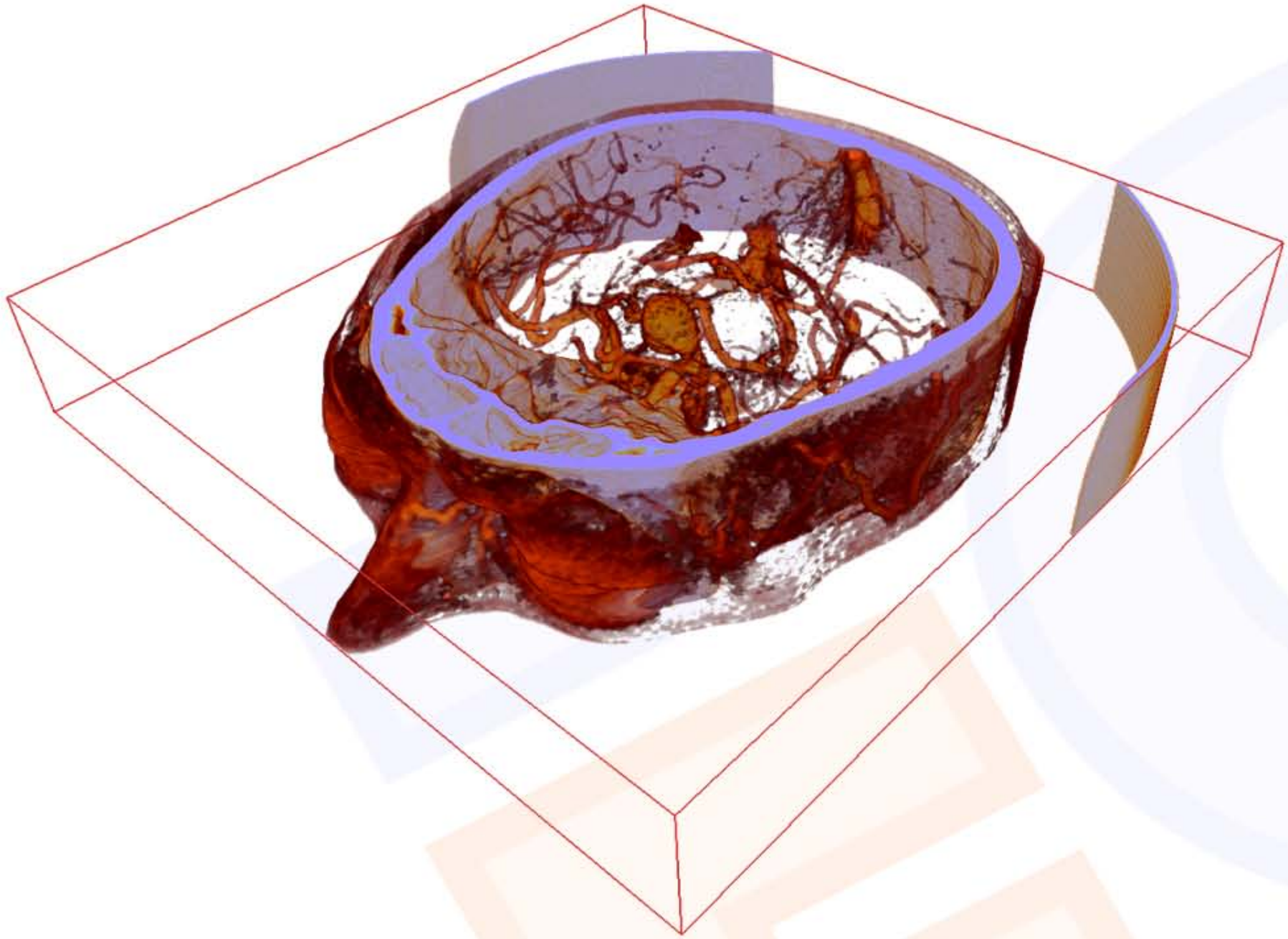
Classification



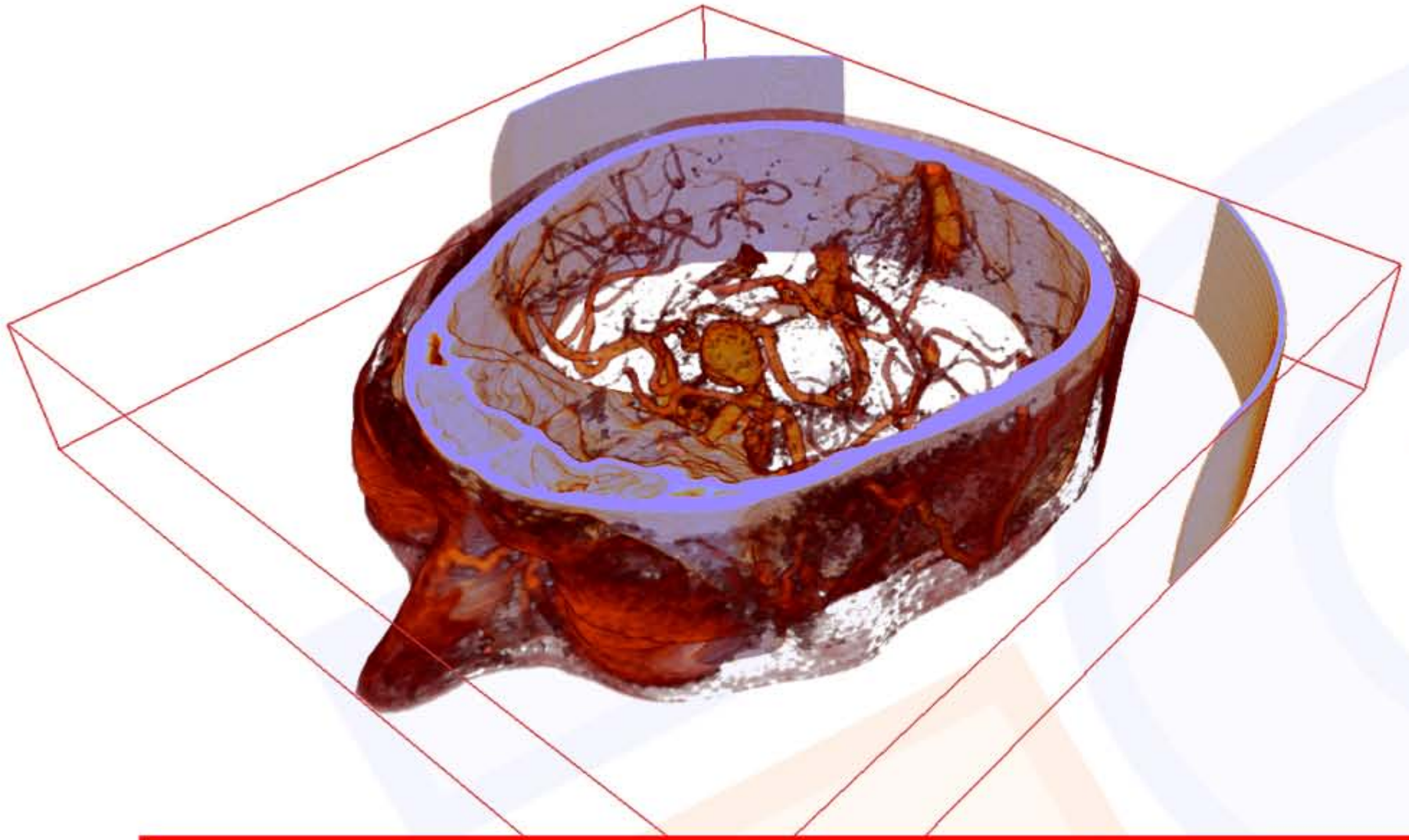
Classification



Classification

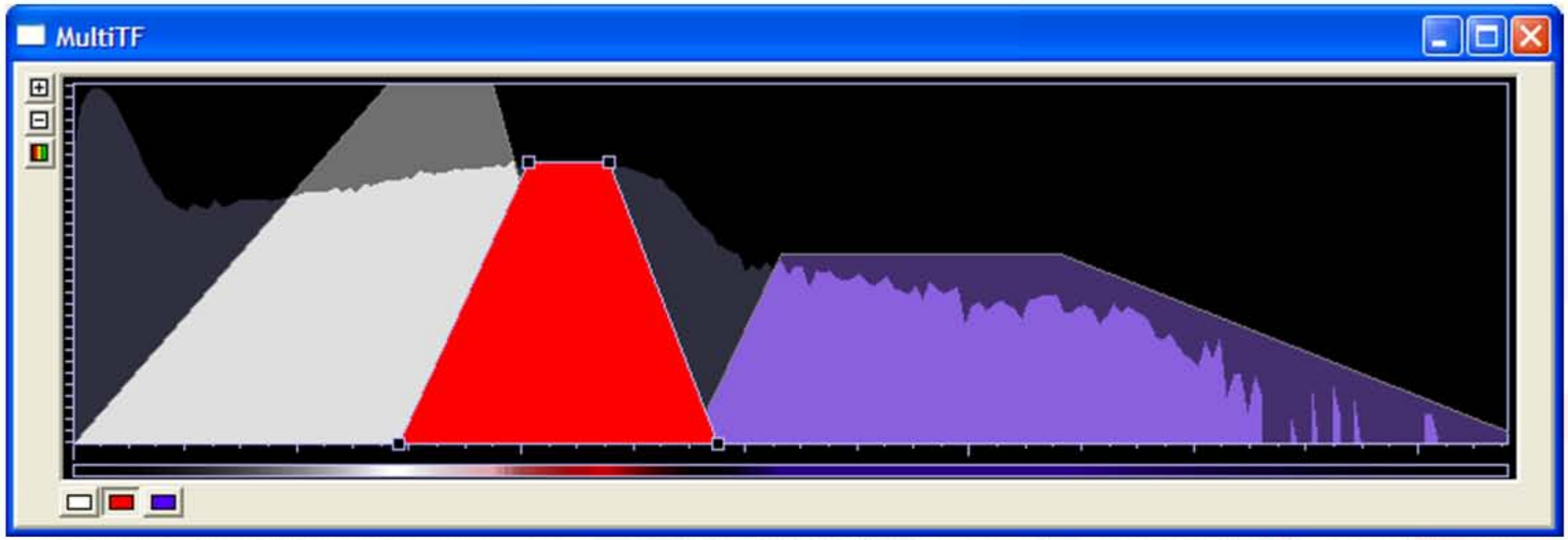


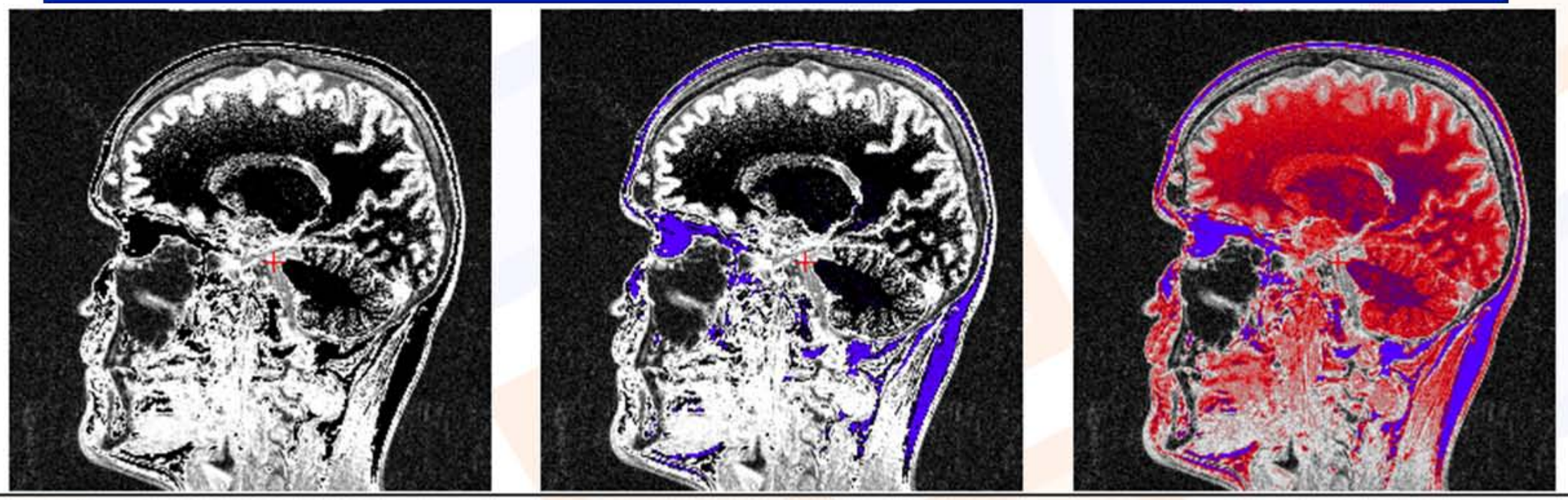
Classification



Real-Time update of the transfer function necessary!!!

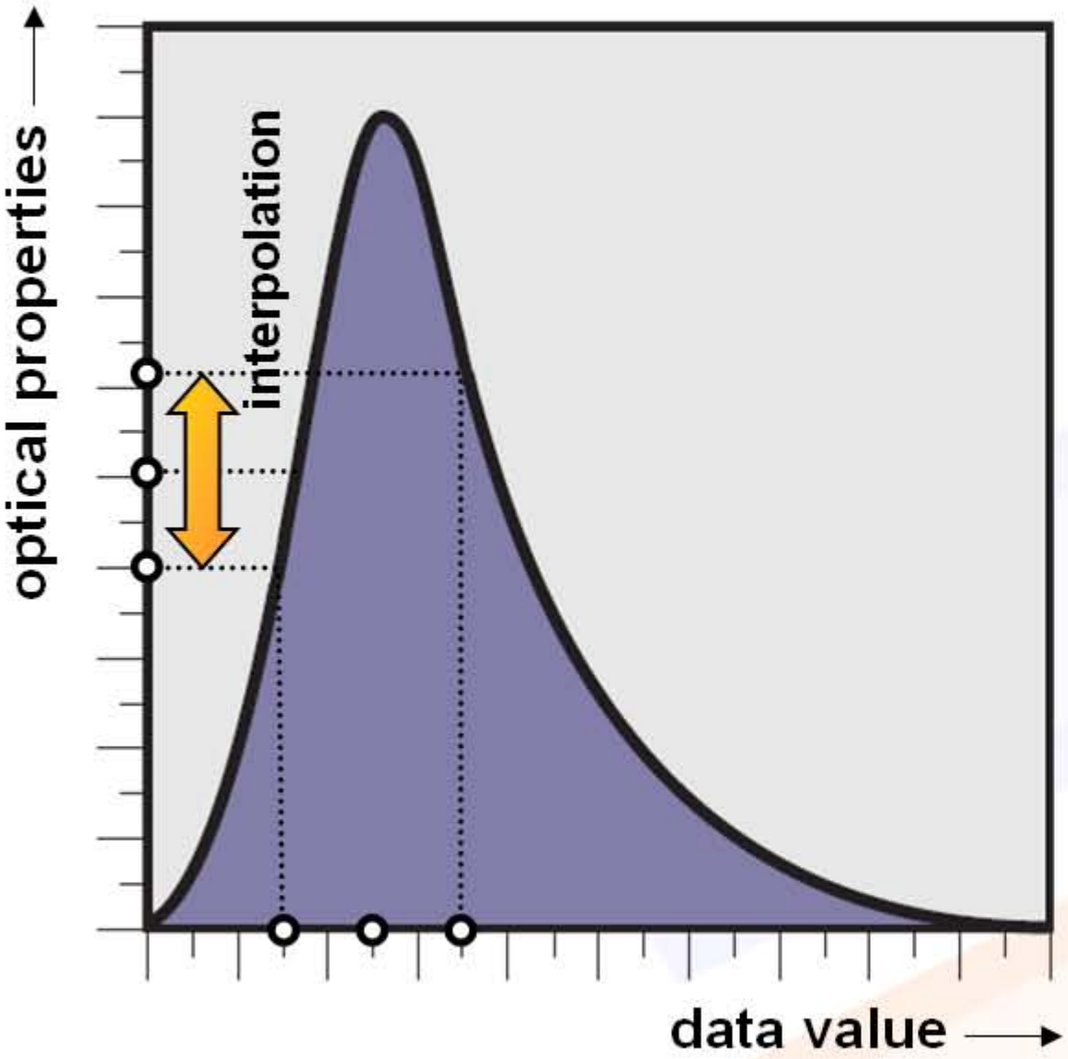
Classification



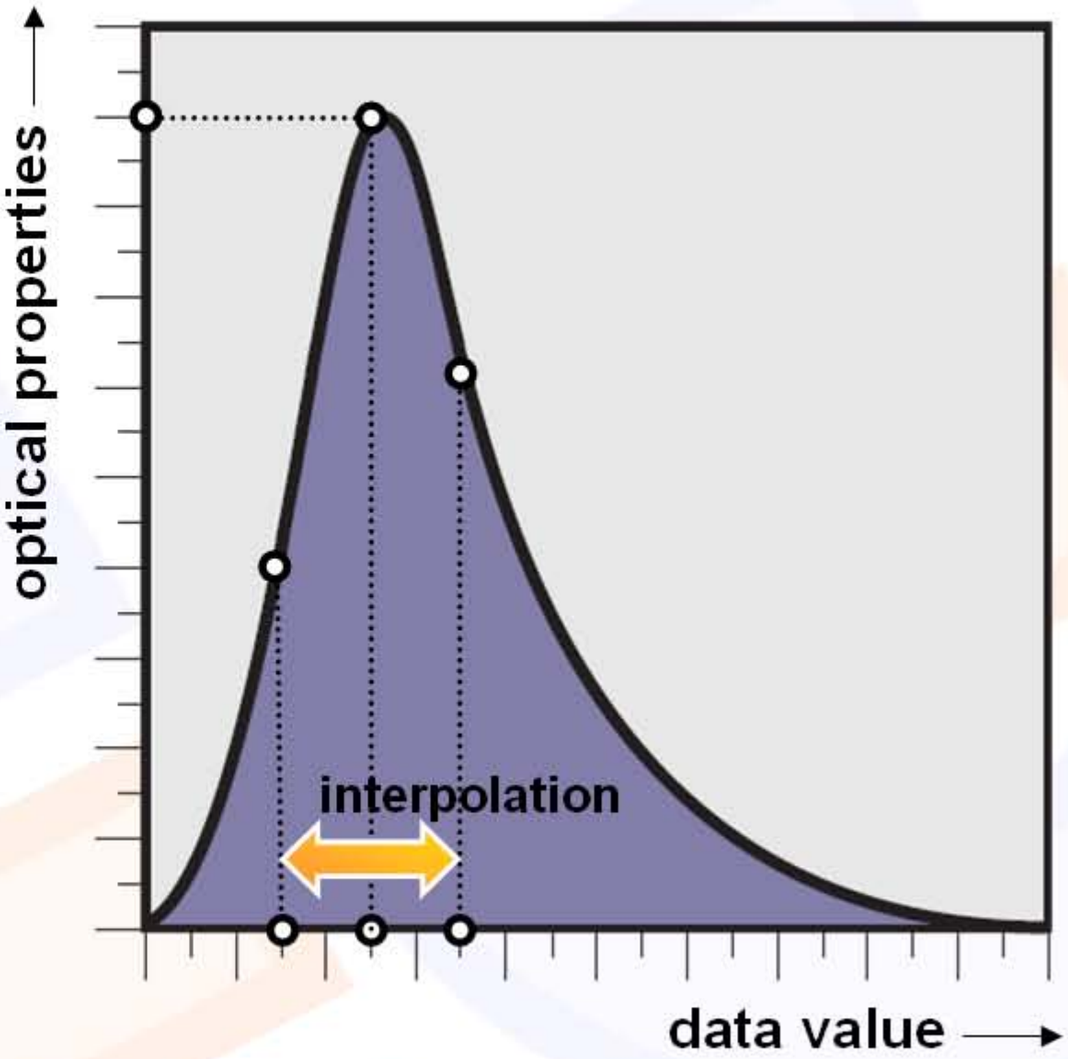


Pre- vs Post-Interpolative Classification

PRE-INTERPOLATIVE



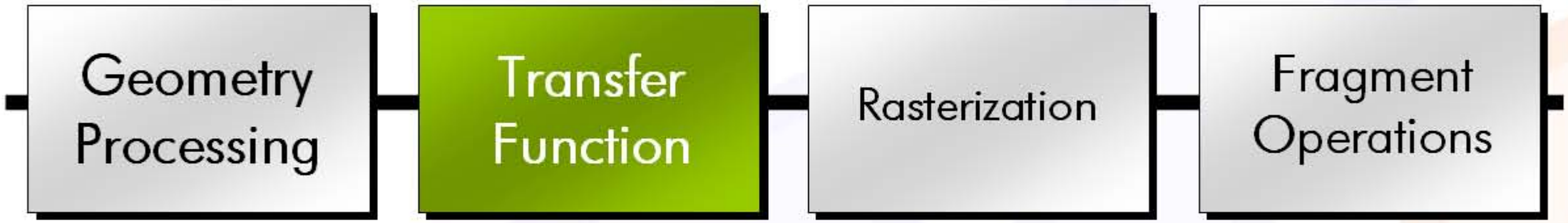
POST-INTERPOLATIVE



Pre-Classification

- **Pre-Classification:**

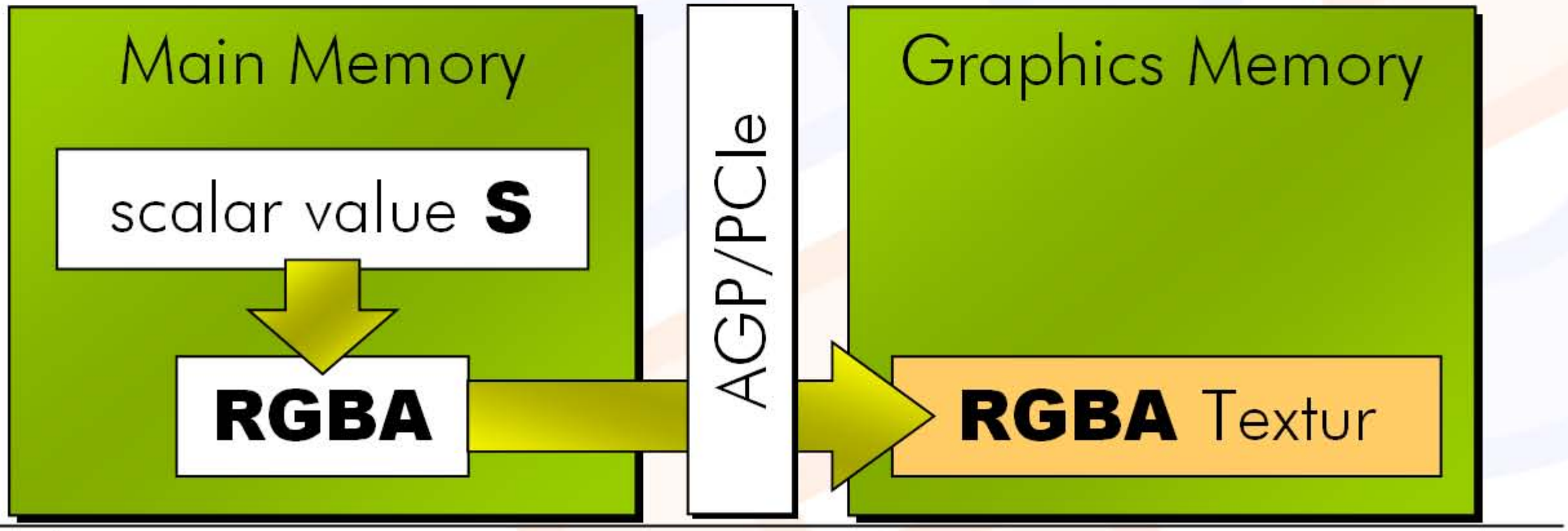
  Color table is applied before interpolation.
  (pre-interpolative Transferfunction)



- A color value is fetched from a table
  for each Voxel
- A RGBA Value is determined
  for each Voxel

Possible Implementations

- The *naive* Approach:
  Save Emission- and Absorptionterms directly in the Texture.

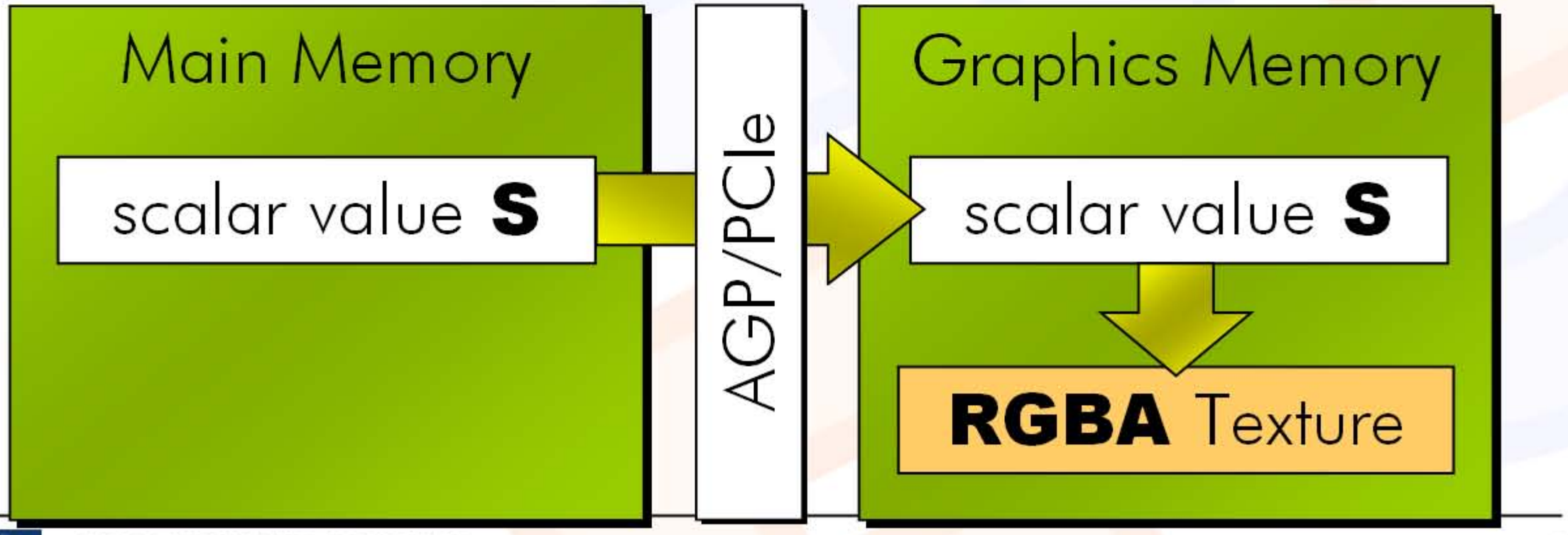


Possible Implementations

- The *naive* Approach:
  Save Emission- and Absorptionterms directly in the Texture.
- Very high memory consumption
  - Main Memory (RGBA und scalar volumes)
  - Graphics Memory (RGBA volume)
- High Load on memory bus
  RGBA Volume must be transferred.
- **Upload necessary on TF change**

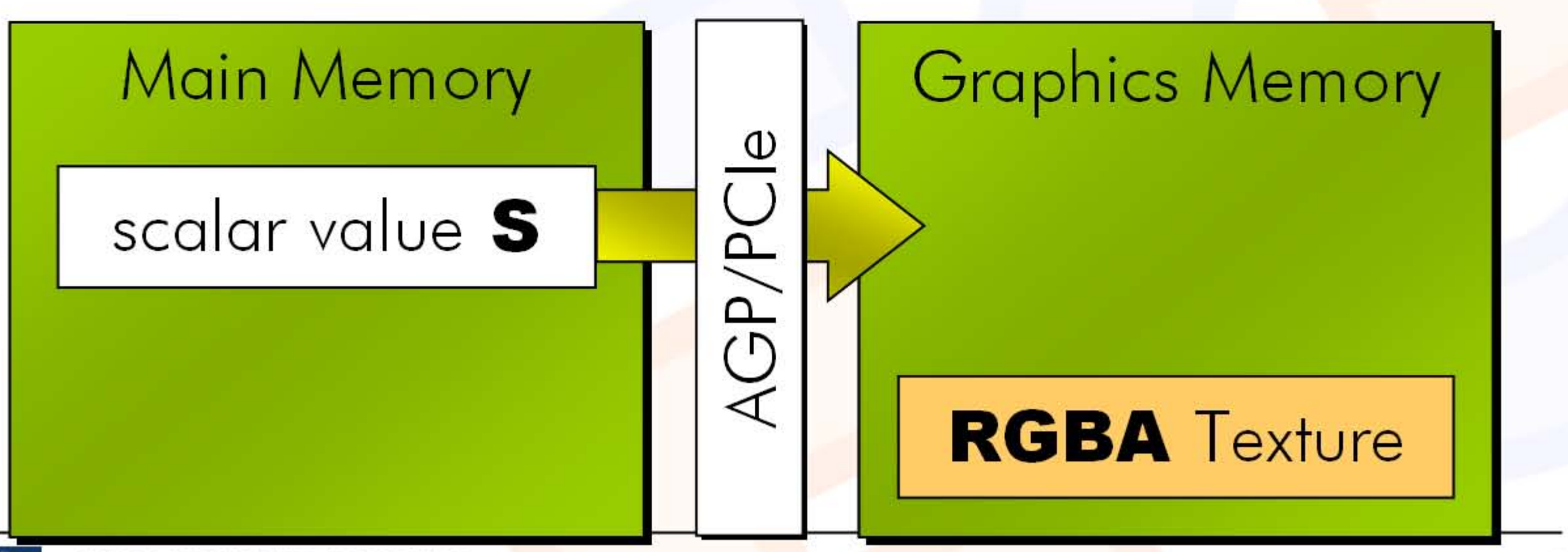
Possible Implementations

- A better Approach:
  Apply color table during
  texture transfers from main memory to
  graphics card (standard OpenGL feature)



Possible Implementations

- A better Approach:
  Apply color table during
  texture transfers from main memory to
  graphics card (standard OpenGL feature)

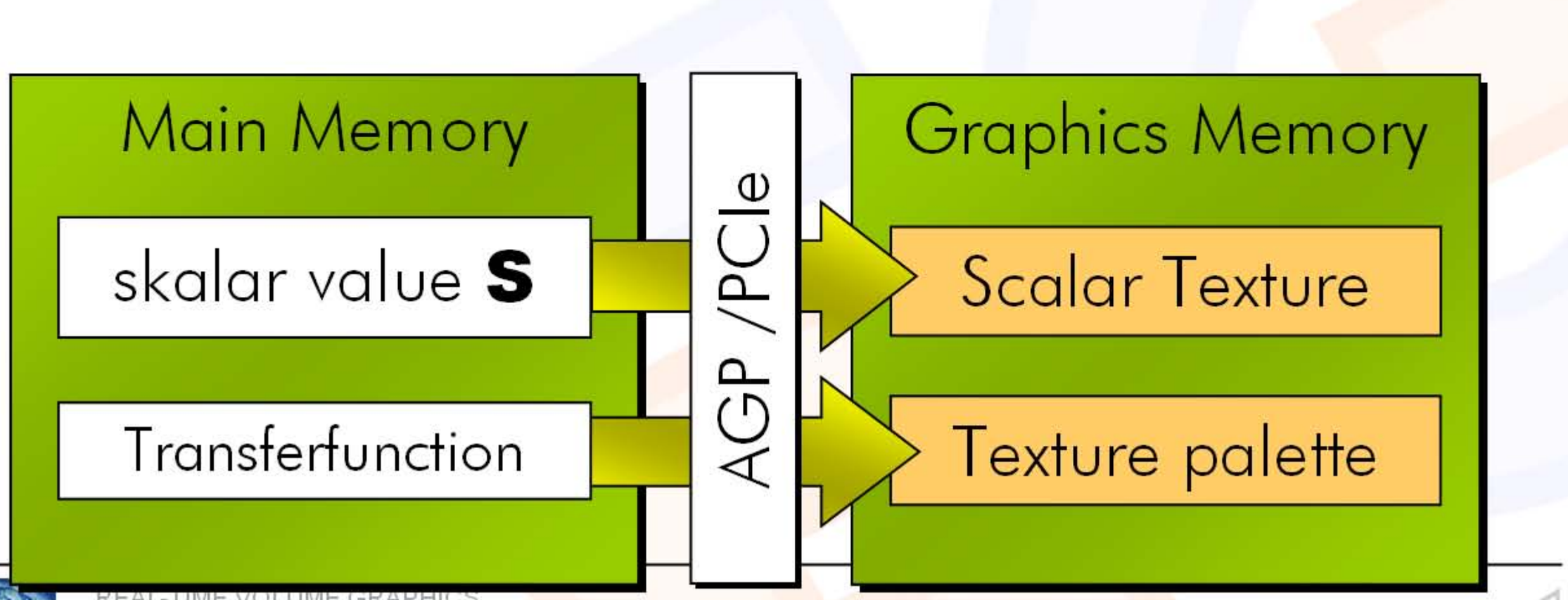


Possible Implementations

- A better Approach:
  Apply color table during
  texture transfers from main memory to
  graphics card (standard OpenGL feature)
- High memory consumption
  - Main Memory (only scalar volume)
  - Graphics Memory (RGBA volume)
- Reduced load on memory bus
  - Only the scalar volume is transferred.
- Upload necessary on TF change

Possible Implementations

- The best approach: Paletted Textures
  Store the scalar volume together with the color table directly in graphics memory.
- Hardware-Support necessary!



Possible Implementations

- The best approach: Paletted Textures
  Store the scalar volume together with the color table directly in graphics memory.
- Hardware-Support necessary!
- Low memory consumption
  - Main Memory (scalar volume can be deleted!)
  - Graphics Memory (scalar volume + TF)
- Low load on memory bus
  - Scalar volume must be transferred only once!
- Only the color table must be re-uploaded on TF change

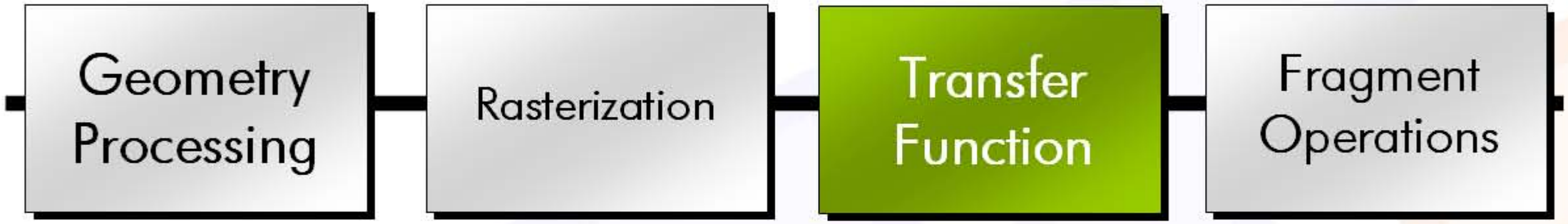
Pre-Classification Summary

- Summary Pre-Classification
  - Application of the Transferfunction before Rasterization
  - One RGBA Lookup for each Voxel
  - Different Implementations:
    - Texture Transfer
    - Texture Color Tables (paletted textures)
  - Simple and Efficient
  - Good for coloring segmented data

Post-Classification

- **Post-Classification:**

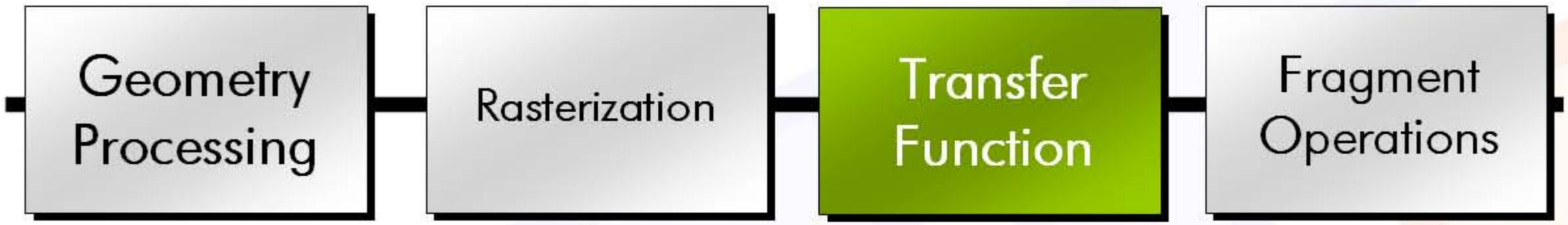
  The color table is applied after Interpolation (***post-interpolative*** *Transferfunction*).



Post-Classification

- **Post-Classification:**

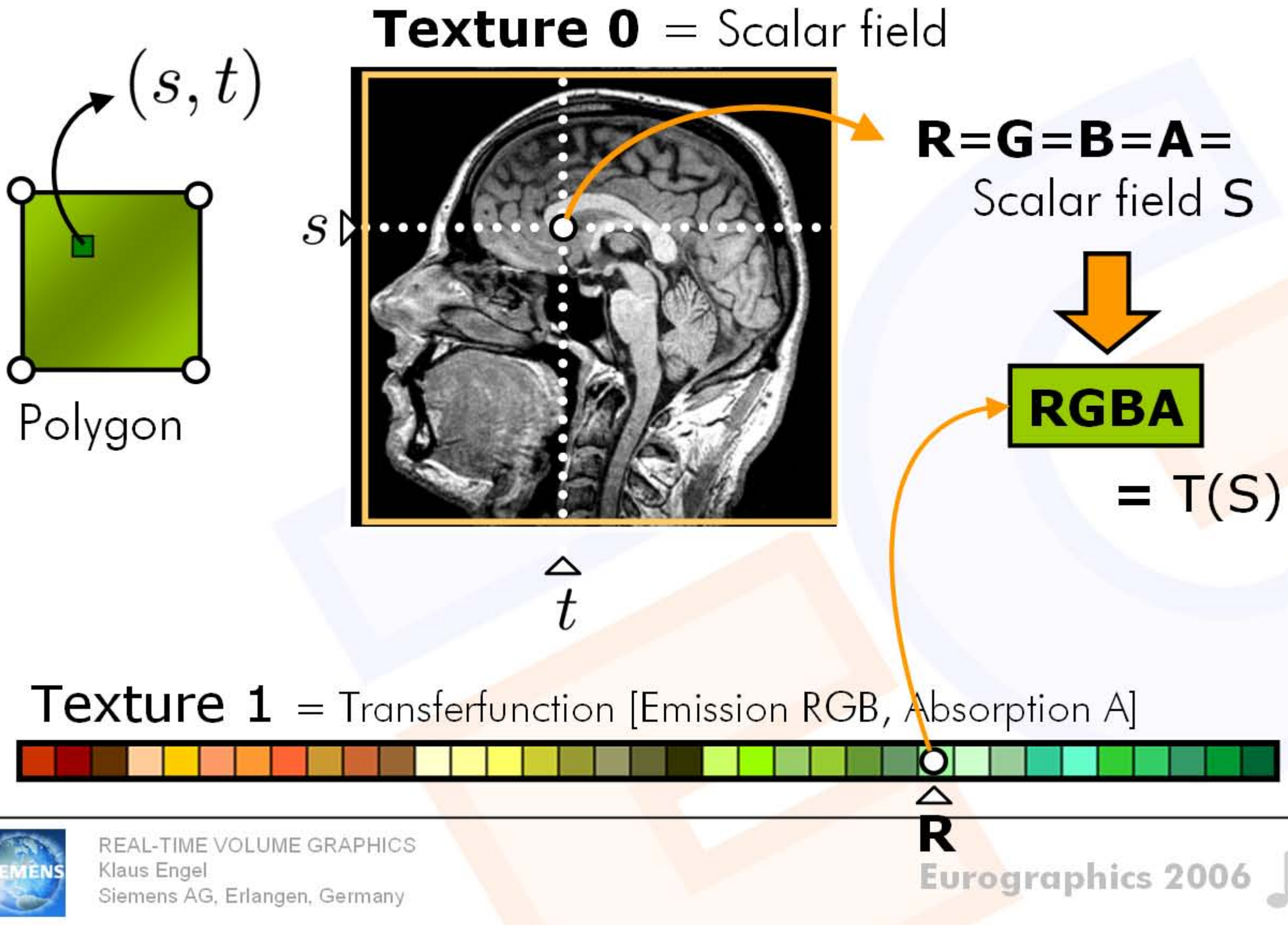
  The color table is applied after Interpolation (***post-interpolative*** *Transferfunction*).



- A color is fetched from the color table

  for each Fragment

Post-Classification

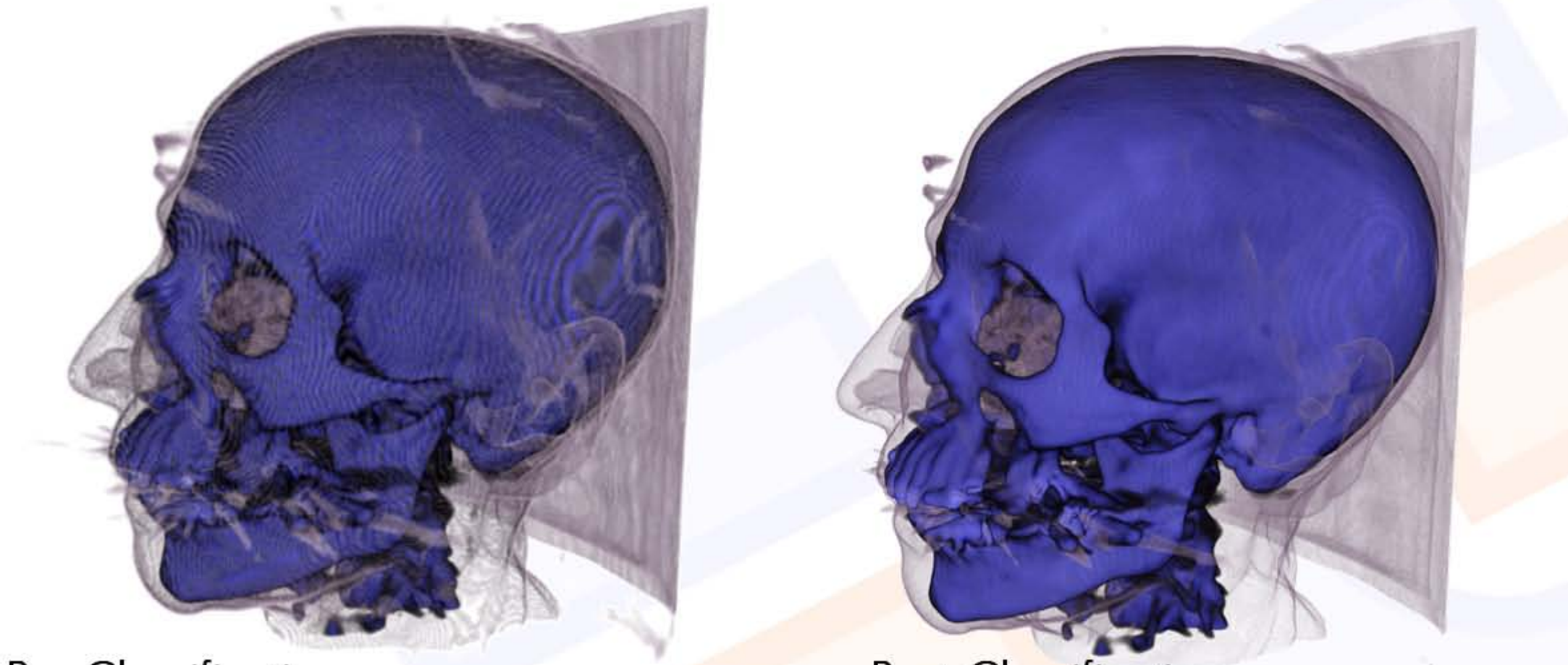


CG Implementation

```
//fragment program for post-classification
//using 3D textures
float4 main (float3 texUV : TEXCOORD0,
             uniform sampler3D volume_texture,
             uniform sampler1D transfer_function) :
   COLOR
{
   float index = tex3D(volume_texture, texUV);
   float4 result = tex1D(transfer_function, index);
   return result;
}
```

Quality: Pre- vs. Post-Classification

- Comparison of image quality

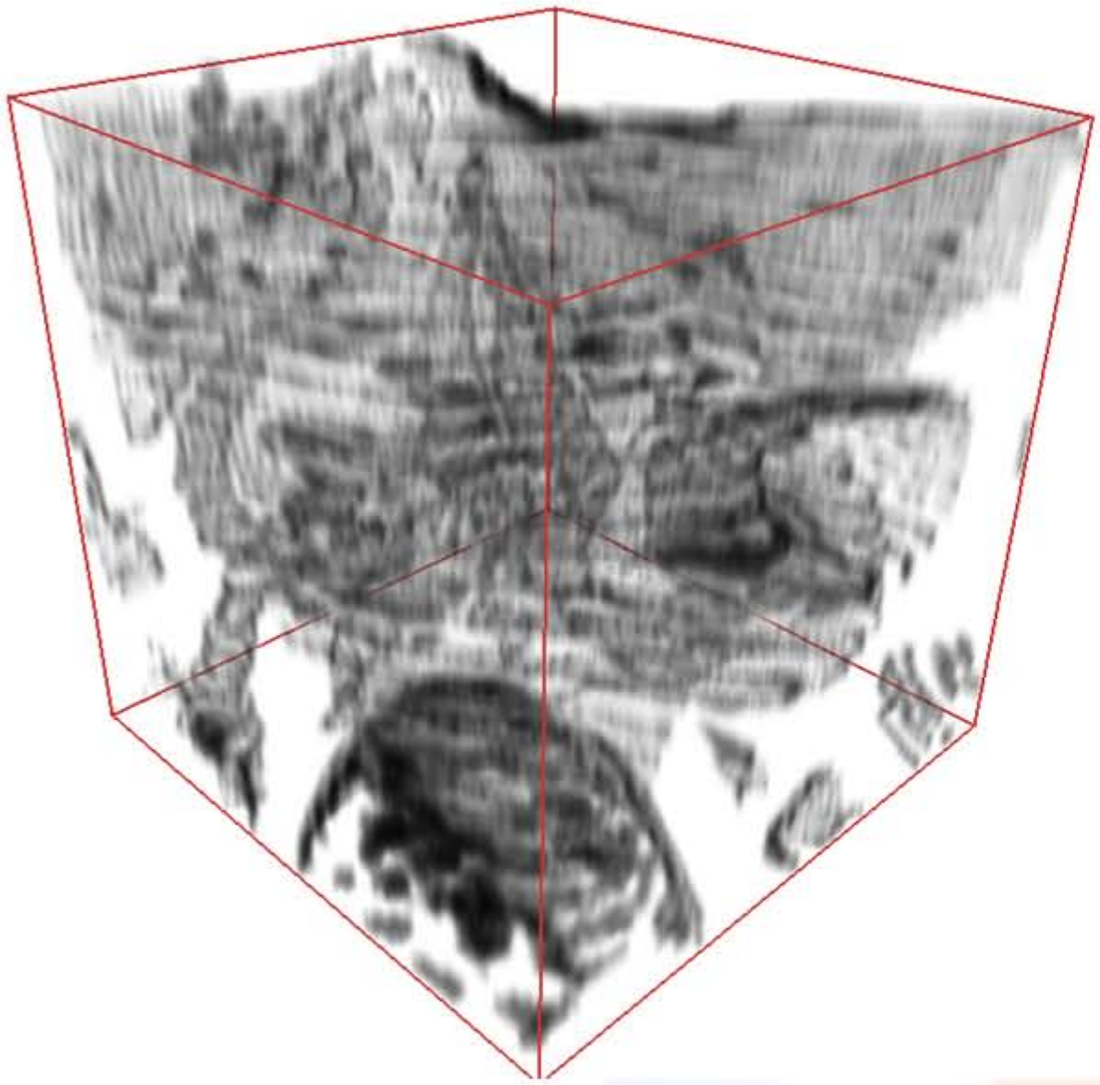


Pre-Classification

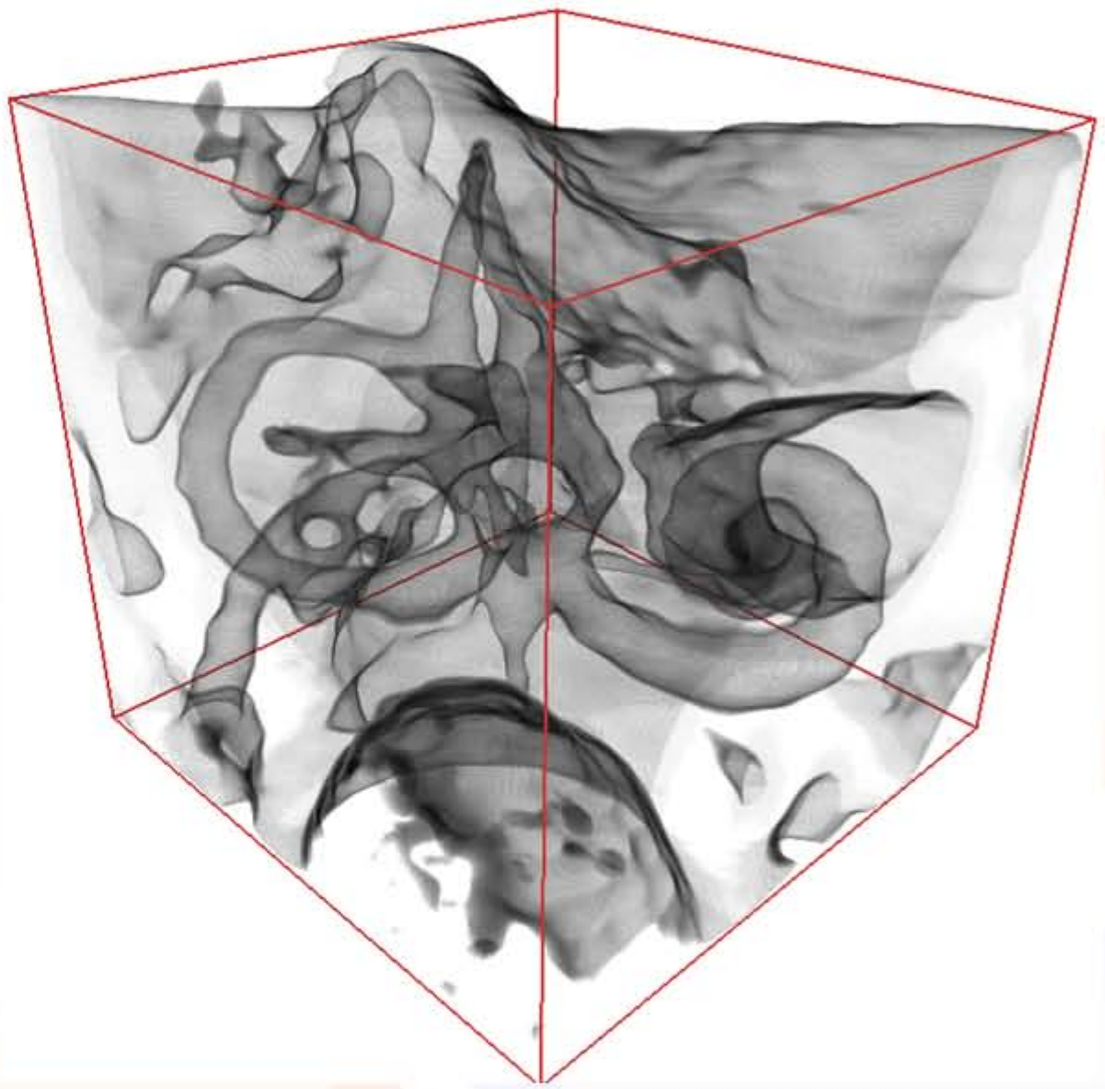
Post-Classification

Same TF, same Resolution, same Sampling Rate

Quality

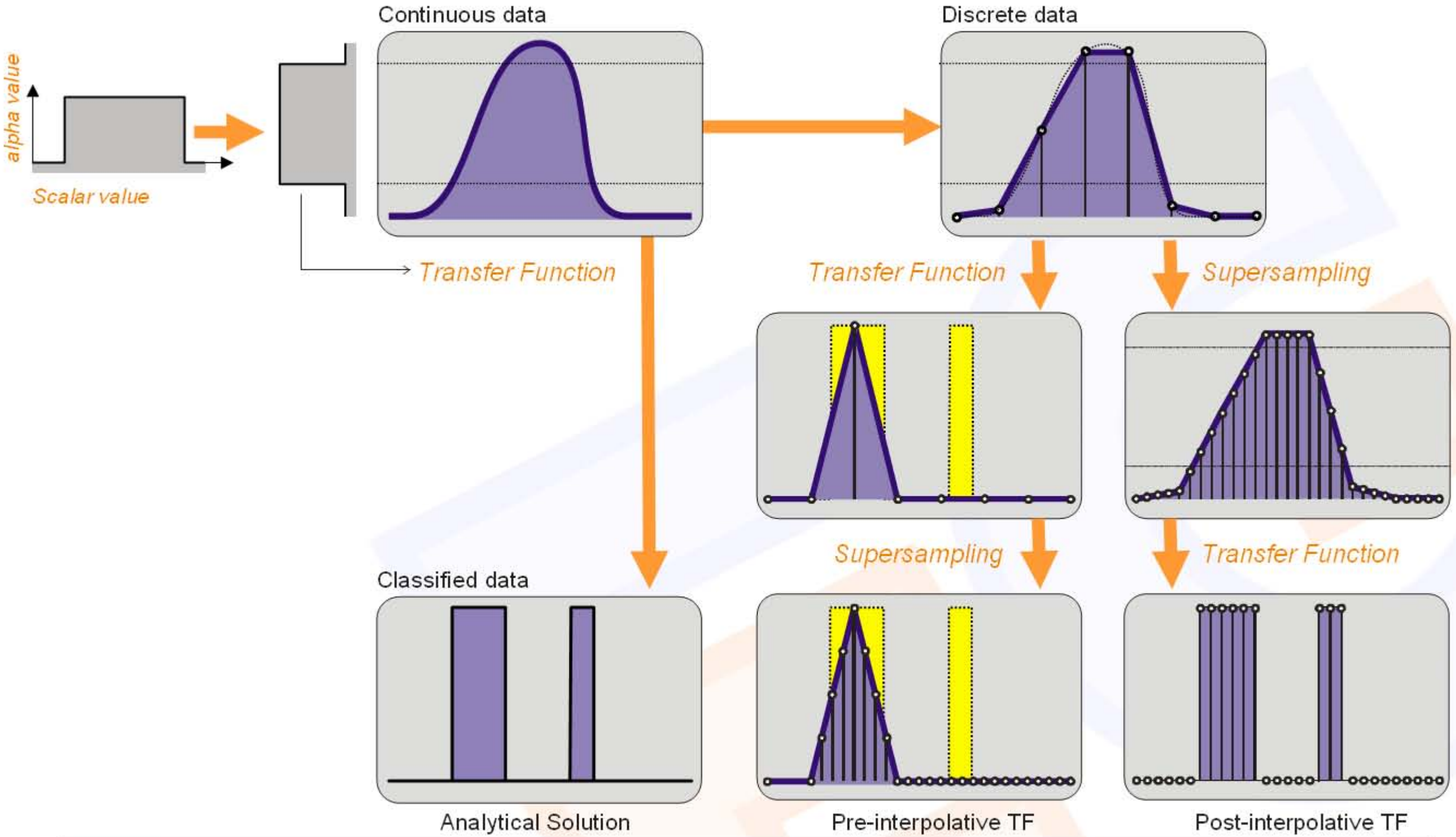


Pre-Classification

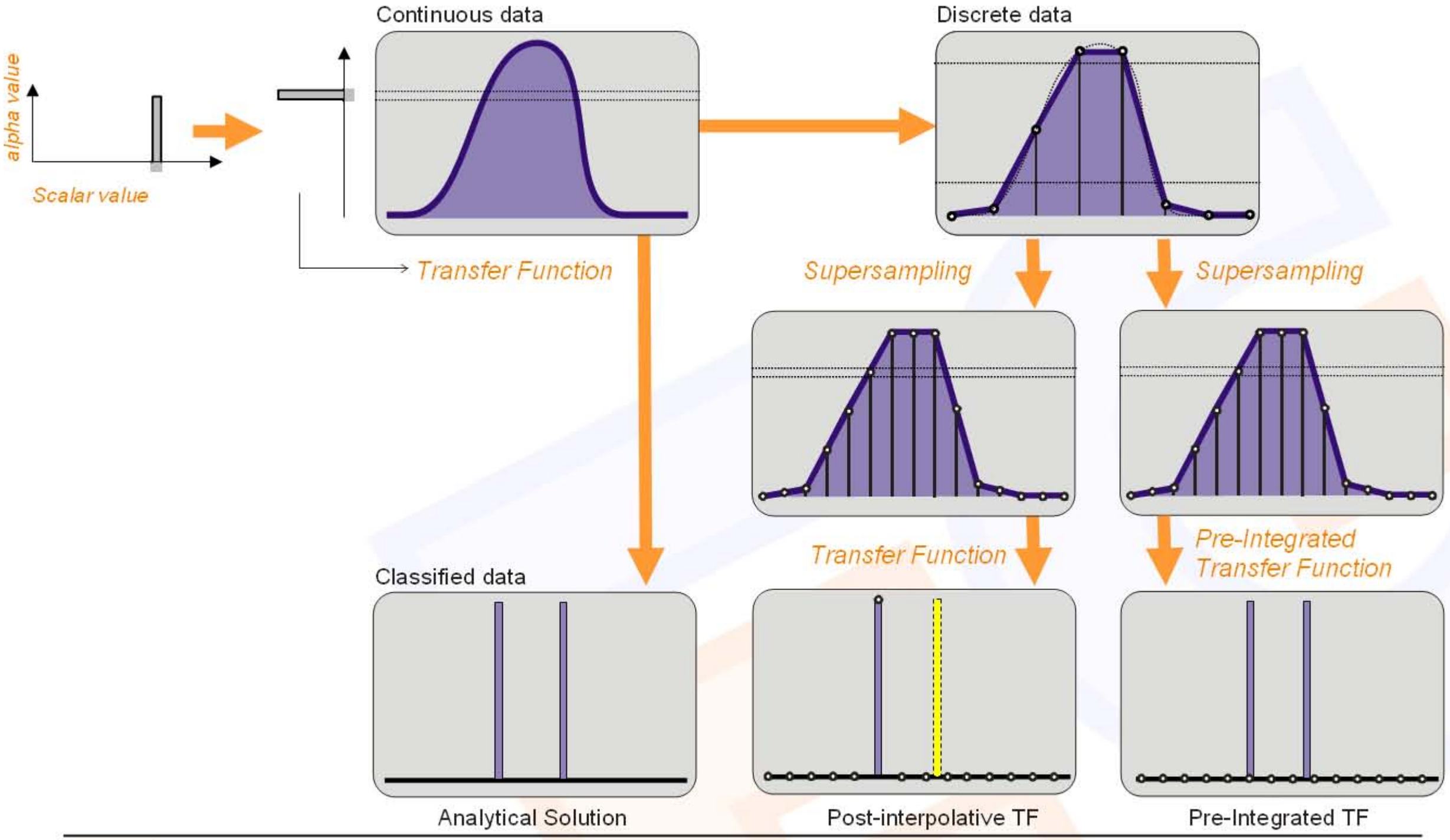


Post-Classification

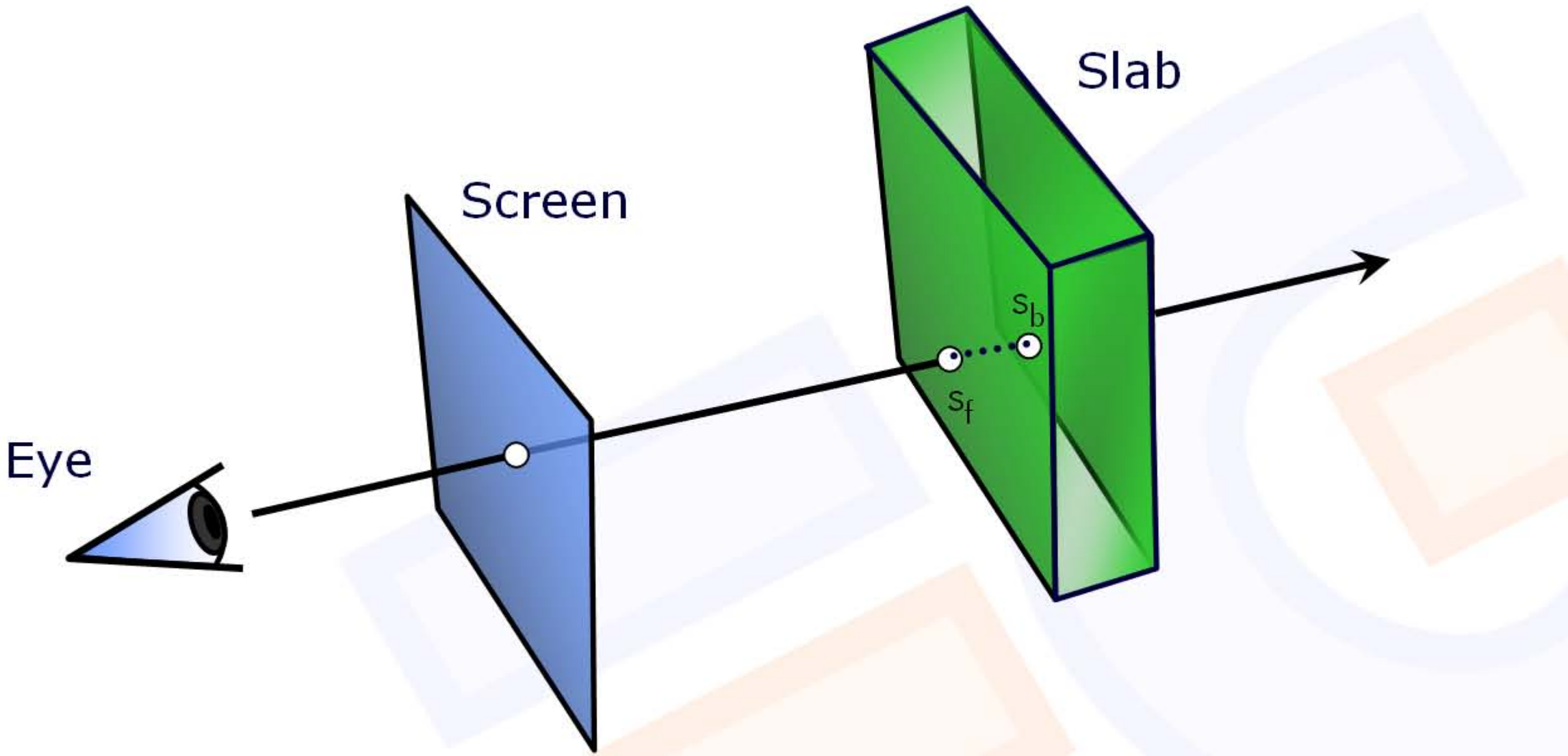
Pre- vs Post-Classification



Post- vs Pre-Integrated Classification



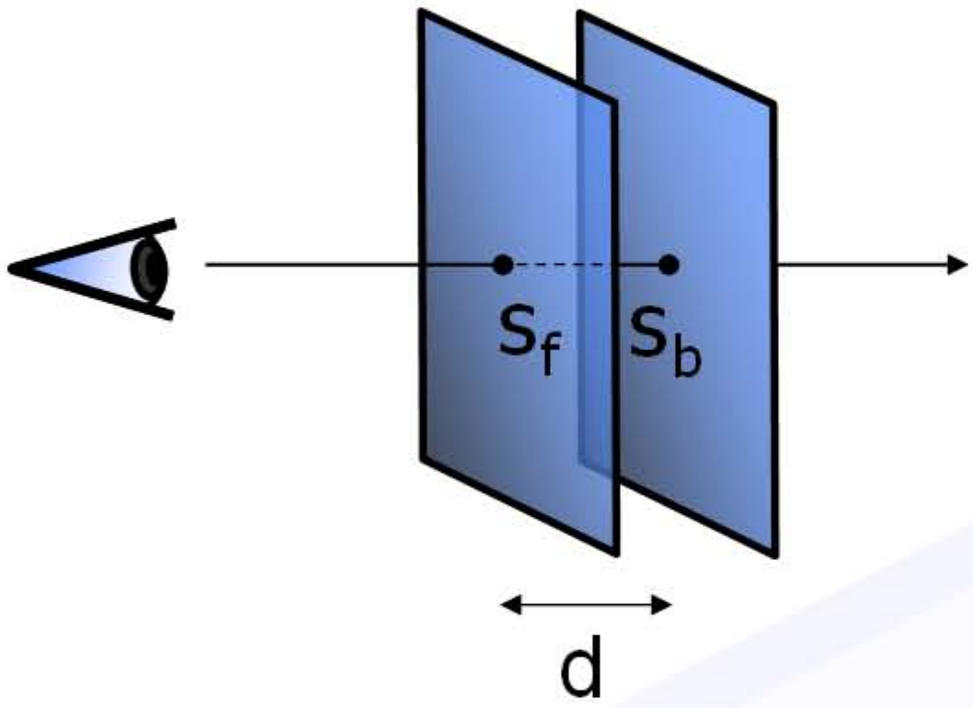
Classification Artifacts / Pre-integration



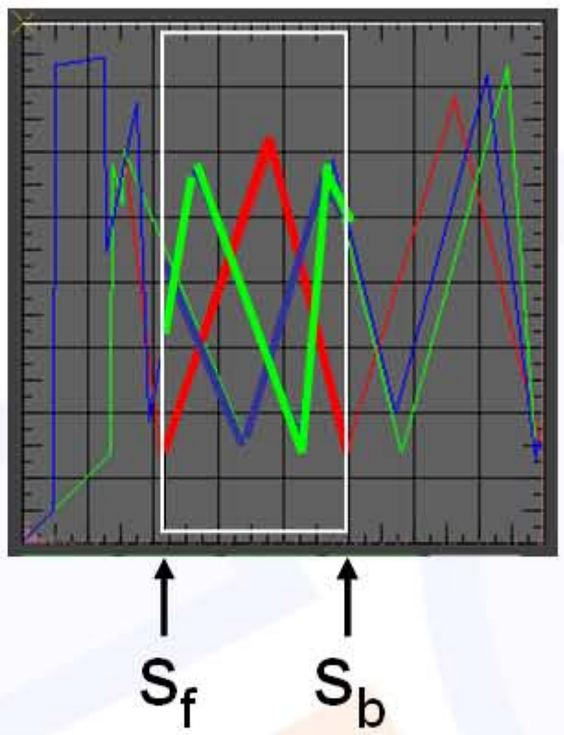
Pre-Integrated Classification

front slice

back slice



pre-integrate all possible combinations in the TF

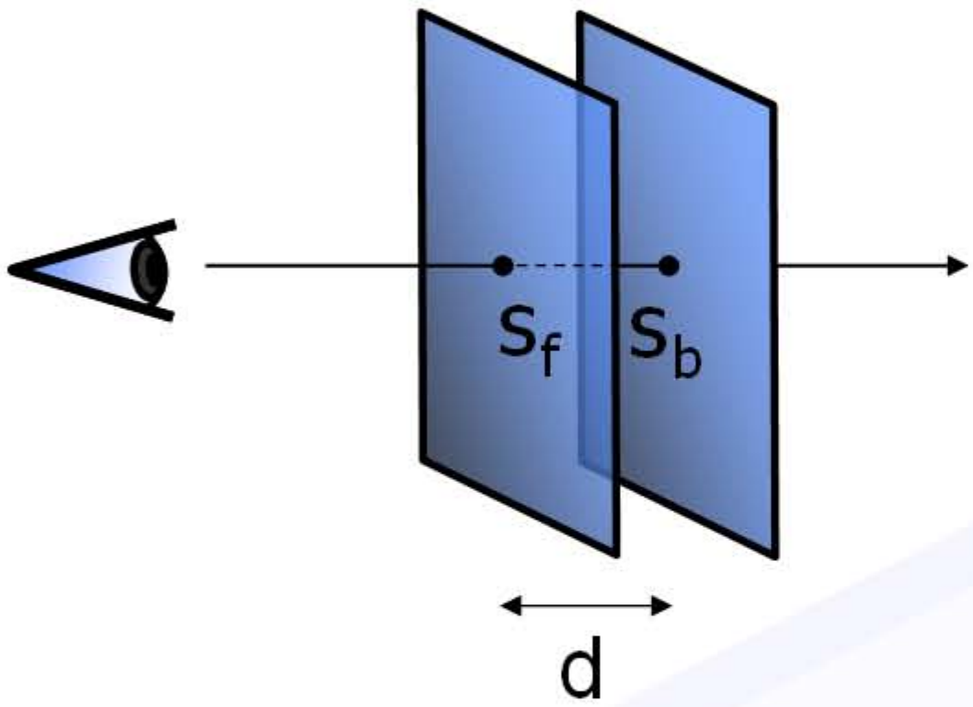


Assume constant sampling distance d

Pre-Integrated Classification

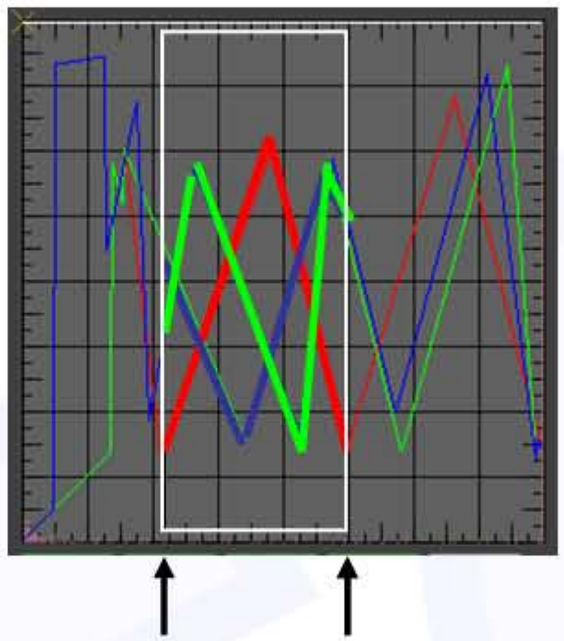
front slice

back slice



Assume constant sampling distance d

pre-integrate all possible combinations in the TF

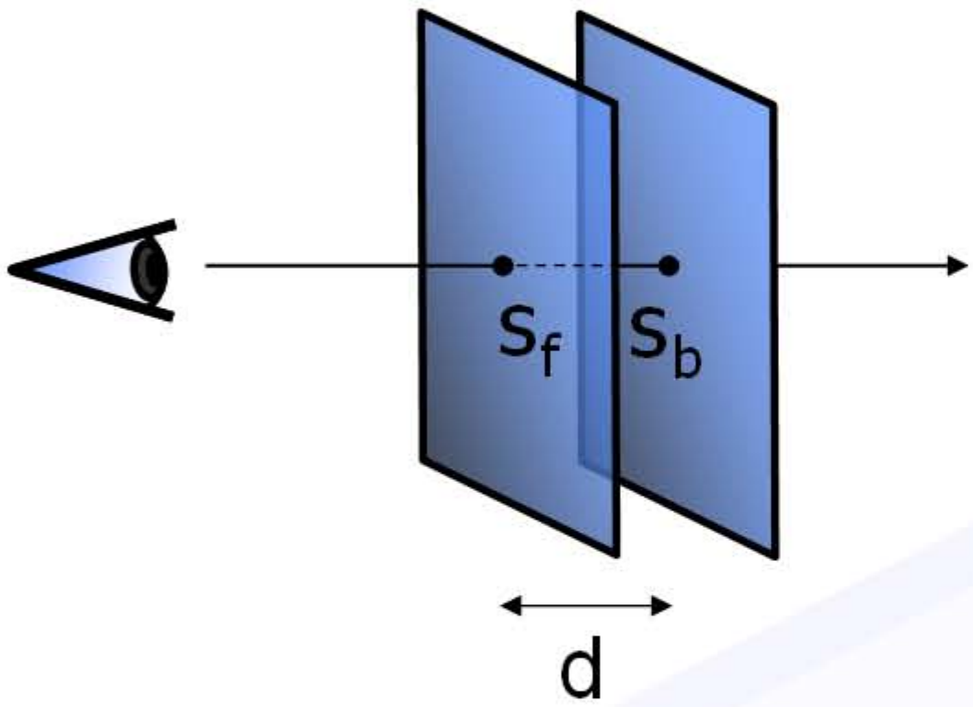


store integral into table

Pre-Integrated Classification

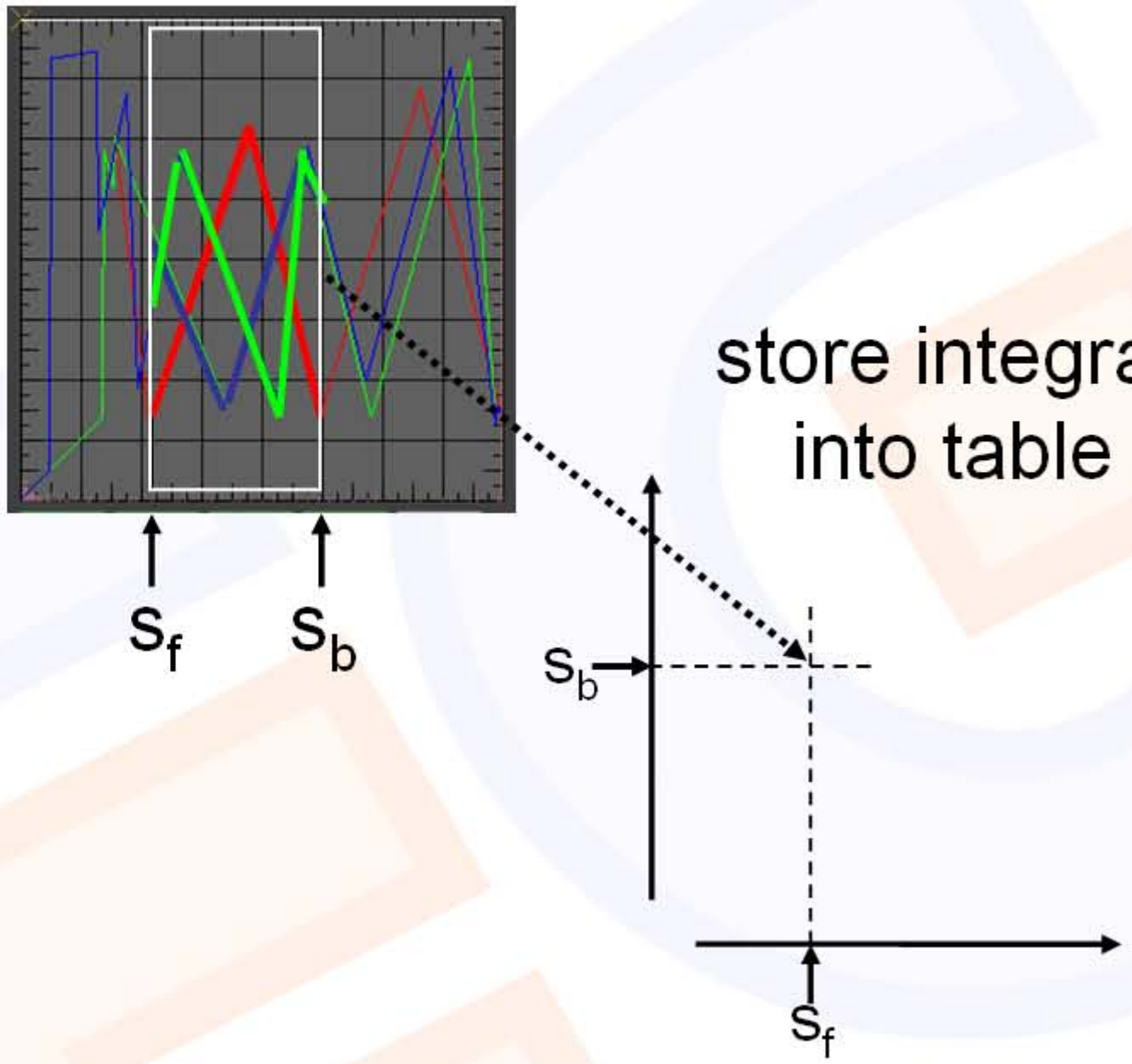
front slice

back slice



Assume constant sampling distance d

pre-integrate all possible combinations in the TF

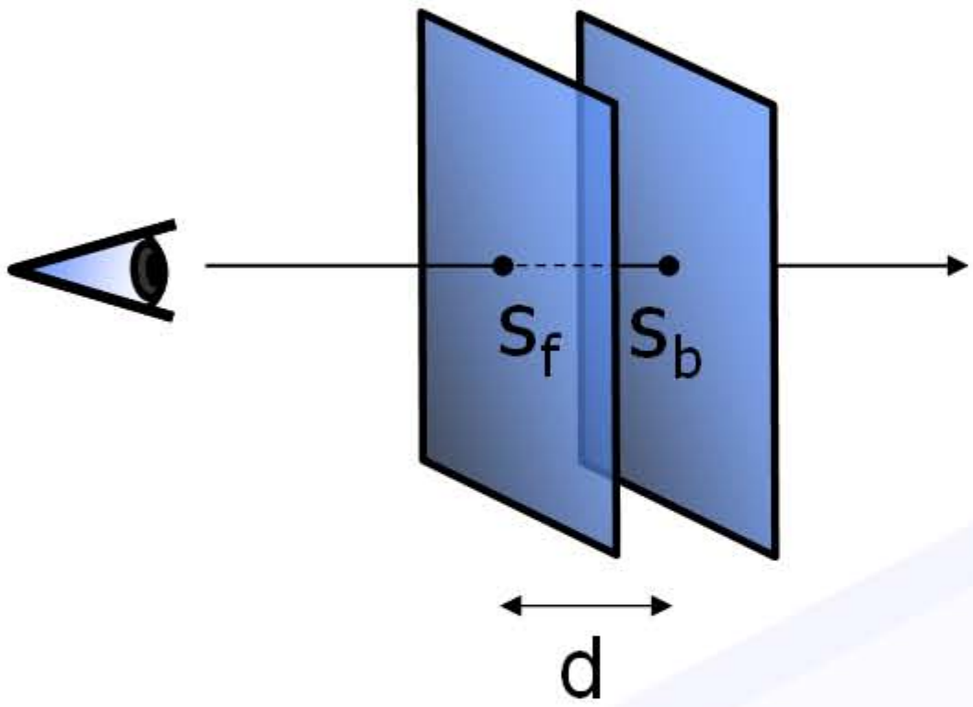


store integral into table

Pre-Integrated Classification

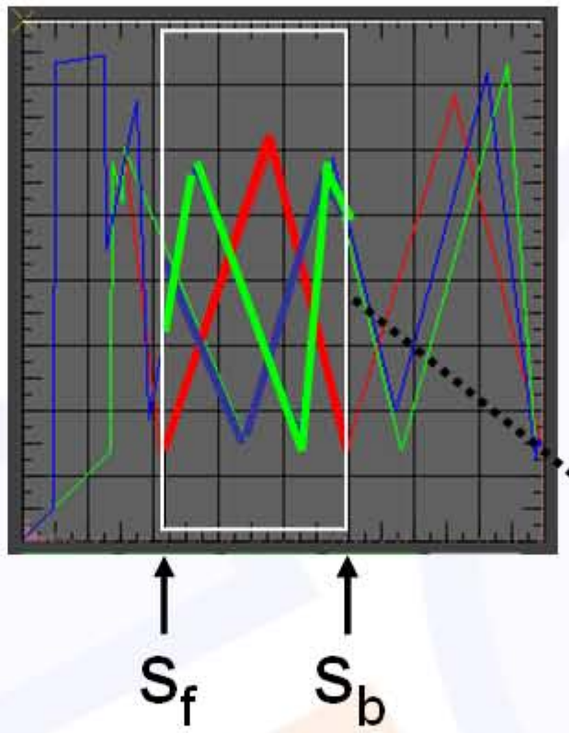
front slice

back slice

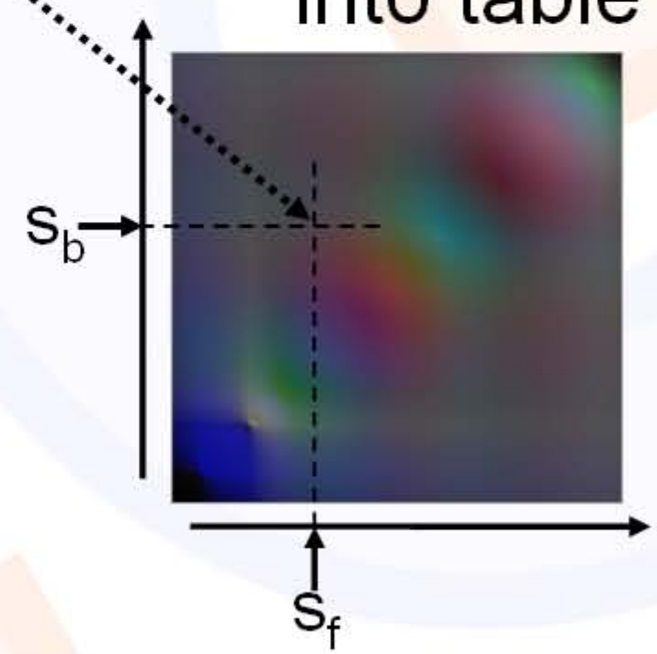


Assume constant sampling distance d

pre-integrate all possible combinations in the TF



store integral into table



Classification Artifacts / Pre-integration

Cg Fragment Program

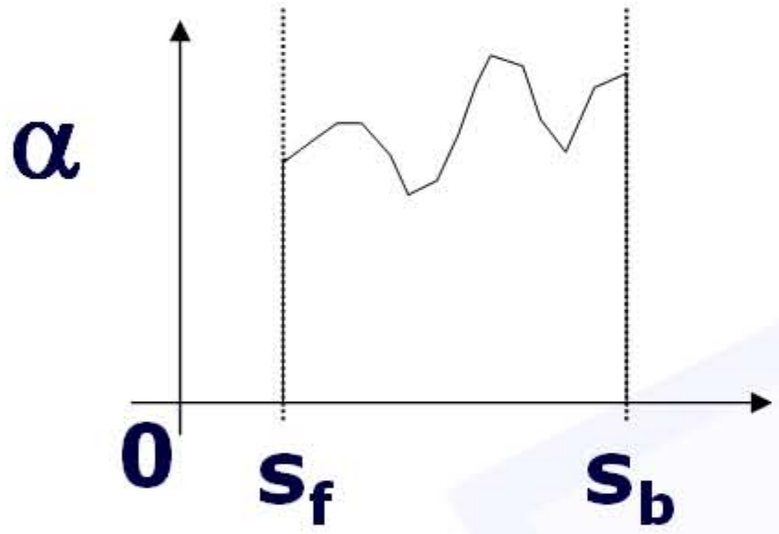
```
struct v2f_simple {
    float4 Hposition : POSITION;
    float3 TexCoord0 : TEXCOORD0;
    float3 TexCoord1 : TEXCOORD1;
float4 Color0 : COLOR0;
};

float4 main(v2f_simple IN,
            uniform sampler3D Volume,
            uniform sampler2D TransferFunction,
            uniform sampler2D PreIntegrationTable) : COLOR
{
    float4 lookup;
    //sample front scalar
    lookup.x = tex3D(Volume, IN.TexCoord0.xyz).x;
    //sample back scalar
    lookup.y = tex3D(Volume, IN.TexCoord1.xyz).x;

    //lookup and return pre-integrated value
    return tex2D(PreIntegrationTable, lookup.yx);
}
```

Pre-Integrated Classification

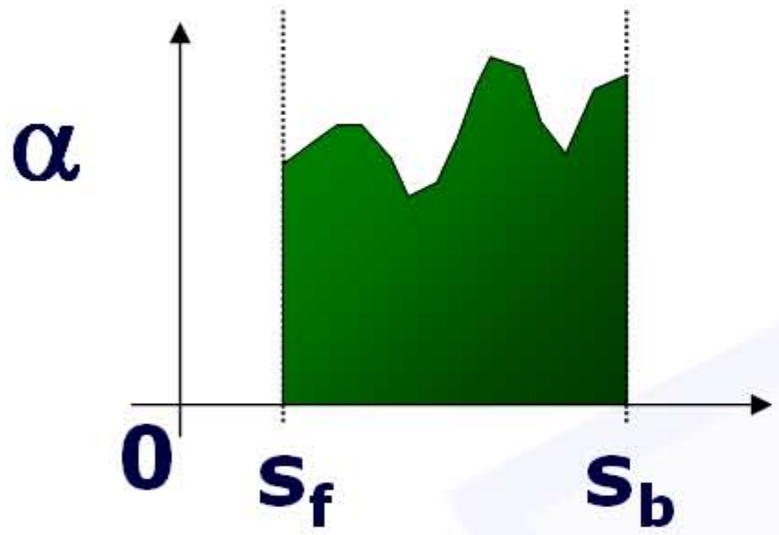
- Fast re-computation of the pre-integration table when transfer function changes
  - Use Integral functions



- Hardware-Accelerated Computation: Roettger, Ertl. A Two-Step Approach for Interactive Pre-Integrated Volume Rendering of Unstructured Grids. In *Proc. VolVis '02*

Pre-Integrated Classification

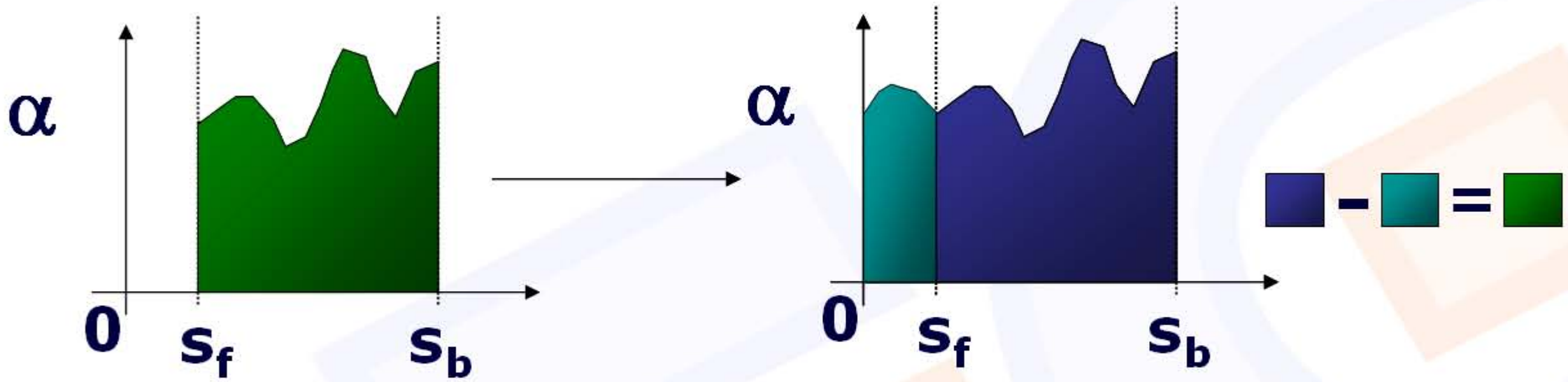
- Fast re-computation of the pre-integration table when transfer function changes
  - Use Integral functions



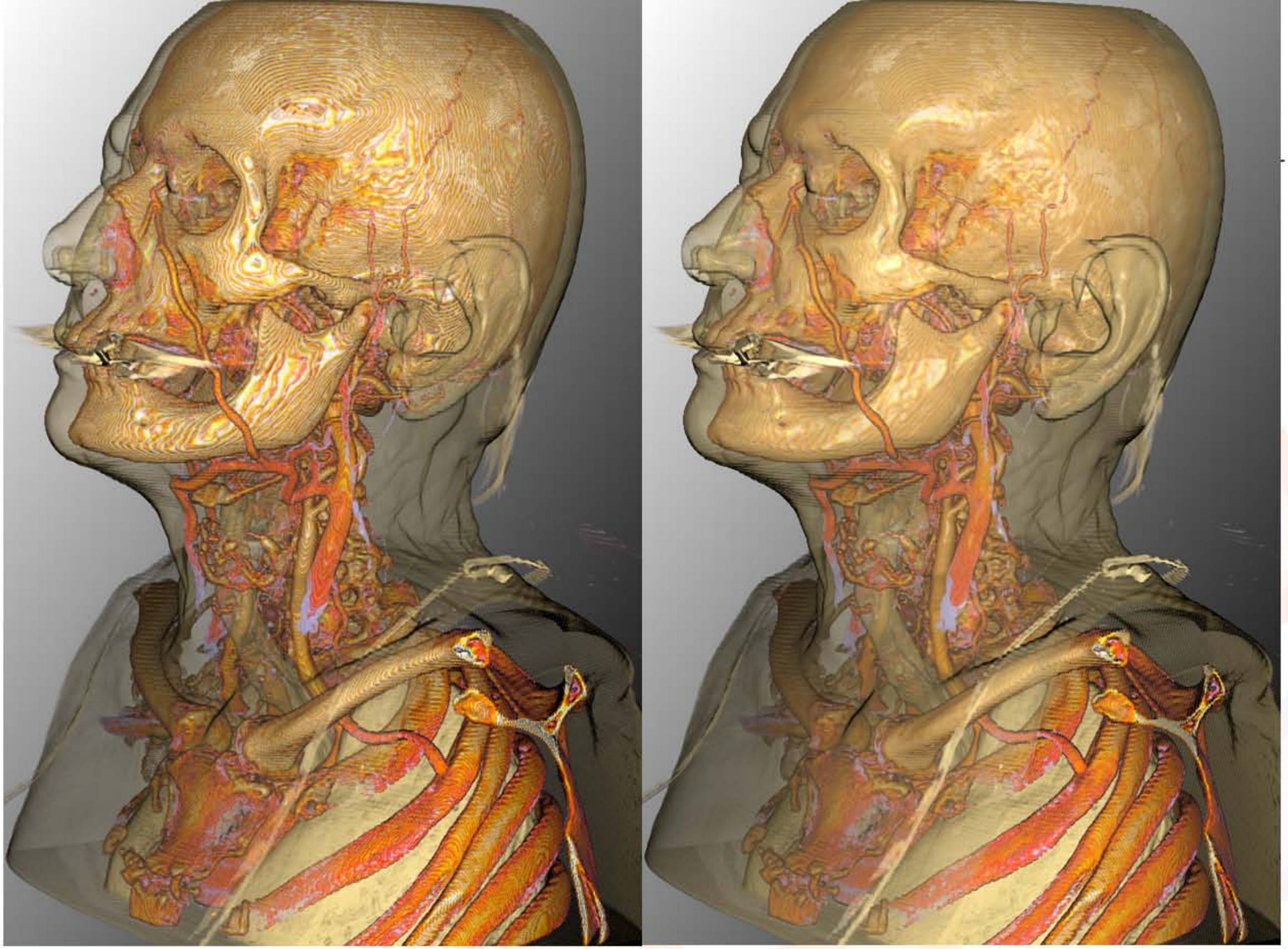
- Hardware-Accelerated Computation: Roettger, Ertl. A Two-Step Approach for Interactive Pre-Integrated Volume Rendering of Unstructured Grids. In *Proc. VolVis '02*

Pre-Integrated Classification

- Fast re-computation of the pre-integration table when transfer function changes
  - Use Integral functions



- Hardware-Accelerated Computation: Roettger, Ertl. A Two-Step Approach for Interactive Pre-Integrated Volume Rendering of Unstructured Grids. In *Proc. VolVis '02*



When to use which Classification

- Pre-Interpolative Classification
  - If the graphics hardware does not support fragment shaders
  - For simple segmented volume data visualization
- Post-Interpolative Classification
  - If the transfer function is "smooth"
  - For good quality and good performance (especially when slicing)
- Pre-Integrated Classification
  - If the transfer function contains high frequencies
  - For best quality