Accelerating Curvature Estimate in 3D Seismic Data Using GPGPU

Marcelo Arruda
marruda@tecgraf.puc-rio.br
Outline

• Introduction
• Volumetric Curvature Estimate
• Parallel Approach
• Results
• Conclusions
Introduction

• 3D Seismic Data

Stratigraphic layers in a seismic acquisition area [Petrobras]
Introduction

- 3D Seismic Data
Introduction

• 3D Seismic Data
Introduction

• 3D Seismic Data
Introduction

• Costs to drill an oil well
  
  – Pre-salt layer (depth 5000 to 7000 meters)

  First well drilled:
  
  – US$ 240 Million
  – 1 year

  Now, a similar well:
  
  – US$ 60 Million
  – 60 days

Oil exploration [Petrobras]
Introduction

• Seismic Interpretation

  – Structural and stratigraphic features

  – Indicates the presence or absence of reservoirs
Introduction

• Seismic Interpretation

  – **Structural** and **stratigraphic** features

  – Indicates the **presence or absence** of reservoirs
Introduction

- Seismic Interpretation
  - Faults
Introduction

• Seismic Interpretation
  – Faults

Fault interpretation process
Introduction

• Seismic Interpretation
  – Salt domes

Salt dome interpretation process
Introduction

- **Seismic Attributes Estimate**
  - Highlight important features
  - Provides visual aid on the task of manual interpretation
  - Less susceptible to incorrect interpretations
Introduction

• **Seismic Attributes Estimate**
  - **Highlight** important features
  - Provides Visual aid on the task of manual interpretation
    - Less susceptible to incorrect interpretations
Introduction

• **Seismic Attributes Estimate**
  – **Highlight** important features
  – Provides **Visual aid** on the task of manual interpretation
    • Less susceptible to incorrect interpretations
Introduction

• **Seismic Attributes Estimate**
  – **Highlight** important features
  – Provides **Visual aid** on the task of manual interpretation
    • Less susceptible to incorrect interpretations
Introduction

• Curvature Attributes
Introduction

But curvature attribute estimate is very slow.

The interpreter needs to fine tune some parameters.
Introduction

But curvature attribute estimate is very slow

The interpreter needs to fine tune some parameters
Objective

• Enable interactive real-time visualization of curvature attributes on user workstations
  – Allows fine tune parameters
  – Speeds up the interpretation process
  – Reduces the labor-intensive work
  – Decrease errors due the lack of experience
Objective

• Enable interactive real-time visualization of curvature attributes on user workstations
  – Allows fine tune parameters
  – Speeds up the interpretation process
  – Reduces the labor-intensive work
  – Decrease errors due the lack of experience
Objective

- Enable interactive real-time visualization of curvature attributes on user workstations
  - Allows fine tune parameters
  - Speeds up the interpretation process
  - Reduces the labor-intensive work
  - Decrease errors due the lack of experience
Objective

- Enable **interactive real-time visualization** of curvature attributes on user workstations
  - Allows fine tune parameters
  - Speeds up the interpretation process
  - Reduces the labor-intensive work
  - Decrease errors due to the lack of experience
Objective

- Enable **interactive real-time visualization of curvature attributes on user workstations**
  - Allows fine tune parameters
  - Speeds up the interpretation process
  - Reduces the labor-intensive work
  - Decrease errors due the lack of experience
Volumetric Curvature Estimate (VCE)

- Second-derivative-based
- Computationally intensive
- VCE takes several hours on user workstation
Volumetric Curvature Estimate (VCE)

- Second-derivative-based
- Computationally intensive
- VCE takes several hours on user workstation
Volumetric Curvature Estimate (VCE)

- Second-derivative-based
- Computationally intensive
- VCE takes several hours on user workstation

Volumetric Curvature Estimate (VCE)

• Second-derivative-based
• Computationally intensive
• VCE takes several hours on user workstation
• “A method to estimate volumetric curvature attributes in 3D seismic data”, proposed by Martins et al (2012)
Volumetric Curvature Estimate (VCE)

- Curvature Attributes

Maximum

Minimum
Volumetric Curvature Estimate

Amplitude volume
Volumetric Curvature Estimate

Amplitude volume

Horizon identifier volume

Curvature Attributes
Volumetric Curvature Estimate

Amplitude volume → Horizon identifier volume → Normal field
Volumetric Curvature Estimate

Amplitude volume → Horizon identifier volume → Normal field → Curvature Attributes

Curvature estimate method
Volumetric Curvature Estimate

• Three steps

1º) Computation of horizon identifier attribute
2º) Normal field estimate
3º) Curvature estimate
Volumetric Curvature Estimate

• Three steps

  1º) Computation of horizon identifier attribute
    • Vertical derivative
      • Improves lateral continuity of seismic surfaces
Volumetric Curvature Estimate

• Three steps
  
  1º) Computation of horizon identifier attribute
      • Vertical derivative
      • Improves lateral continuity of seismic surfaces
Volumetric Curvature Estimate

• Three steps

1º) Computation of horizon identifier attribute

2º) Normal field estimate

• Based on the gradient of horizon identifier attribute
• Input volume + 3 output normal volume

\[ \nabla F = \left( \frac{\partial F}{\partial x} \frac{\partial F}{\partial y} \frac{\partial F}{\partial z} \right) = (F_x \ F_y \ F_z) \]

\[ N(F) = \begin{cases} \nabla F, & \text{se} \quad F_z \geq 0 \\ -\nabla F, & \text{se} \quad F_z < 0 \end{cases} \]
Volumetric Curvature Estimate

- Three steps
  
  1º) Computation of horizon identifier attribute
  2º) Normal field estimate
    - Based on the gradient of horizon identifier attribute
    - Input volume + 3 output normal volume

\[
\nabla F = \left( \frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z} \right) = (F_x, F_y, F_z)
\]

\[
N(F) = \begin{cases} 
\nabla F, & \text{se } F_z \geq 0 \\
-\nabla F, & \text{se } F_z < 0 
\end{cases}
\]
Volumetric Curvature Estimate

• Three steps

1º) Computation of horizon identifier attribute

2º) Normal field estimate
  • Based on the gradient of horizon identifier attribute
  • Input volume + 3 output normal volumes

\[ \nabla F = \begin{pmatrix} \frac{\partial F}{\partial x} & \frac{\partial F}{\partial y} & \frac{\partial F}{\partial z} \end{pmatrix} = (F_x \ F_y \ F_z) \]

\[ N(F) = \begin{cases} \nabla F, \text{ se } F_z \geq 0 \\ -\nabla F, \text{ se } F_z < 0 \end{cases} \]
Volumetric Curvature Estimate

• Three steps

1º) Computation of horizon identifier attribute
2º) Normal field estimate
3º) Curvature estimate

• Normal field partial derivatives

\[
H(F) = \begin{pmatrix}
\frac{\partial^2 F}{\partial x^2} & \frac{\partial^2 F}{\partial y \partial x} & \frac{\partial^2 F}{\partial z \partial x} \\
\frac{\partial^2 F}{\partial y \partial x} & \frac{\partial^2 F}{\partial y^2} & \frac{\partial^2 F}{\partial y \partial z} \\
\frac{\partial^2 F}{\partial z \partial x} & \frac{\partial^2 F}{\partial y \partial z} & \frac{\partial^2 F}{\partial z^2}
\end{pmatrix} = \begin{pmatrix}
F_{xx} & F_{xy} & F_{xz} \\
F_{yx} & F_{yy} & F_{yz} \\
F_{zx} & F_{zy} & F_{zz}
\end{pmatrix} = \nabla (\nabla F)
\]

\[
k_G = \frac{H^*(F) N(F)^T}{|N(F)|^4}, \quad k_M = \frac{N(F)^* H^*(F)^* N(F)^T - |N(F)|^2 \text{Trace}(H^*)}{2 |N(F)|^3}
\]

\[
k = k_M \pm \sqrt{k_M^2 - k_G}
\]
Parallel approach

• **Convolution** of Gaussian derivative filters.

  • Derivative operator sizes
    – Short: enable details, more noise
    – Large: enable main features, less noise
  
  • Interpreters usually needs to vary the derivative operator size to highlight the features according to theirs needs.
Parallel approach

• **Convolution** of Gaussian derivative filters.
  
  • Derivative operator size
    - Small: enable details, more noise
    - Large: enable main features, less noise
  
  • Interpreters usually needs to vary the derivative operator size to highlight the features according to theirs needs.
Parallel approach

• **Convolution** of Gaussian derivative filters.
  
  • Derivative operator size
    – Small: enable details, more noise
    – Large: enable main features, less noise
  
  • Interpreters usually need to **vary the derivative operator size** to highlight the features according to their needs.
Parallel approach

- **Convolution** of Gaussian derivative filters
- **Stencil computation**
  - Bandwidth-to-compute
  - Data dependency

Data dependency on a 3 x 3 x 3 stencil computation operator
Parallel approach

- **Convolution** of Gaussian derivative filters
- **Stencil computation**
  - Bandwidth-to-compute
  - Data dependency on a $3 \times 3 \times 3$ stencil computation operator
Parallel approach

- **Convolution of Gaussian derivative filters**
- **Stencil computation**
  - Bandwidth-to-compute
  - Data dependency
Parallel approach

• **Convolution** of Gaussian derivative filters

• **Stencil computation**
  - Bandwidth-to-compute
  - Data dependency
Parallel approach

• Cost of each convolution
  – 3 x 3 x 3 operator: 27
  – 5 x 5 x 5 operator: 125
  – 13x13x13 operator: 2197

• Curvature estimate
  – 9 x 125 MADDs for a 5 x 5 x 5 operator
Parallel approach

• CPU implementation
  – OpenMP
  – Compiler: icc

• GPU implementation
  – CUDA
  – Compiler: nvcc
Parallel approach

**CPU:**

- **Three loops** to sweep through the volume
- **Three loops** to sweep through the derivative operator
- **Blocking** to maximize cache reuse
- Each thread process a **subset** of blocks
- Compiler **did not vectorize** the hot spot
- No manual vectorization with intrinsics
Parallel approach

- **CPU:**
  - Three loops to sweep through the volume
  - Three loops to sweep through the derivative operator
  - Blocking to maximize cache reuse
  - Each thread process a subset of blocks
  - Compiler did not vectorize the hot spot
  - No manual vectorization with intrinsics
Parallel approach

- **CPU:**
  - Three loops to sweep through the volume
  - Three loops to sweep through the derivative operator
  - **Blocking** to maximize cache reuse
    - Each thread process a subset of blocks
    - Compiler did not vectorize the hot spot
    - No manual vectorization with intrinsics
Parallel approach

• CPU:
  – Three loops to sweep through the volume
  – Three loops to sweep through the derivative operator
  – Blocking to maximize cache reuse
  – Each thread process a subset of blocks
  – Compiler did not vectorize the hot spot
  – No manual vectorization with intrinsics
Parallel approach

• CPU:
  – Three loops to sweep through the volume
  – Three loops to sweep through the derivative operator
  – Blocking to maximize cache reuse
  – Each thread process a subset of blocks
  – Compiler did not vectorize the hot spot
  – No manual vectorization with intrinsics
Parallel approach

- **CPU:**
  - Three loops to sweep through the volume
  - Three loops to sweep through the derivative operator
  - **Blocking** to maximize cache reuse
  - Each thread process a **subset** of blocks
  - Compiler **did not vectorize** the hot spot
  - No manual vectorization with intrinsics
Parallel approach

• GPU:
  – Each step in a **different CUDA kernel** (32x8 threads)
  – 60 registers per thread with no spill
  – Memory access optimization
    • Each thread process a column of samples
Parallel approach

• **GPU:**
  – Each step in a **different CUDA kernel** (32x8 threads)
  – **60 registers per thread** with no spill
  – Memory access optimization
    • Each thread process a column of samples
Parallel approach

• GPU:
  – Each step in a **different CUDA kernel** (32x8 threads)
  – **60 registers per thread** with no spill
  – Memory access optimization
    • Each thread process a **column of samples**
Parallel approach

• GPU:
  – Each step in a **different CUDA kernel** (32x8 threads)
  – **60 registers per thread** with no spill
  – Memory access optimization
    • Each thread process a **column of samples**
Parallel approach

• **GPU:**
  – Each step in a **different CUDA kernel** (32x8 threads)
  – **60 registers per thread** with no spill
  – Memory access optimization
    • Each thread process a **column of samples**
Parallel approach

- **GPU:**
  - Each step in a **different CUDA kernel** (32x8 threads)
  - **60 registers per thread** with no spill
  - Memory access optimization
    - Each thread process a **column of samples**
Parallel approach

- 3D shared memory **circular buffer**
  - Based on Paulius Micikevicius work with RTM (Reverse Time Migration)
  - A single round-robin pointer
Parallel approach

- **3D shared memory circular buffer**
  - Based on Paulius Micikevicius work with RTM (Reverse Time Migration)
  - A single round-robin pointer
Parallel approach

- 3D shared memory **circular buffer**
  - Based on Paulius Micikevicius work with RTM (Reverse Time Migration)
  - A single round-robin pointer
Parallel approach

- 3D shared memory **circular buffer**
  - Based on Paulius Micikevicius work with RTM (Reverse Time Migration)
  - A single round-robin pointer
Parallel approach

• Operators and Memory Usage
Results

• CPU: i7 3970x
  – 6 cores
• GPU: GeForce GTX 580
  – 512 cores
• 56% of peak in GFlops
• 1.67 instructions per clock out of maximum 2
Results

• Maximum Curvature Volume
Results

- Minimum Curvature Volume
Results

• Operator size

(a) 5 x 5 x 5  (b) 7 x 7 x 7  (c) 11 x 11 x 11  (d) 17 x 17 x 17
## Results

- **Input volume: 244MB**

<table>
<thead>
<tr>
<th>Operator size</th>
<th>CPU seq. time (s)</th>
<th>CPU with OpenMP time (s)</th>
<th>GPU time (s)</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 x 3 x 3</td>
<td>26.68</td>
<td>3.91</td>
<td>0.31</td>
<td>12.61</td>
</tr>
<tr>
<td>5 x 5 x 5</td>
<td>55.84</td>
<td>6.75</td>
<td>0.55</td>
<td>12.27</td>
</tr>
<tr>
<td>7 x 7 x 7</td>
<td>125.25</td>
<td>15.39</td>
<td>1.42</td>
<td>10.83</td>
</tr>
<tr>
<td>11 x 11 x 11</td>
<td>446.23</td>
<td>53.49</td>
<td>3.79</td>
<td>14.11</td>
</tr>
<tr>
<td>17 x 17 x 17</td>
<td>1586.15</td>
<td>119.84</td>
<td>13.88</td>
<td>8.6</td>
</tr>
</tbody>
</table>

Time spent processing the curvature method

Input volume: 244MB - CPU: i7 3970x - GPU: Geforce GTX 580
Results

• Processing of a single slice of a 1.1GB volume
Results

- **Input volume: 1.1GB**

<table>
<thead>
<tr>
<th>Operator size</th>
<th>CPU with OpenMP time (ms)</th>
<th>GPU time (ms)</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 x 3 x 3</td>
<td>570</td>
<td>10</td>
<td>57x</td>
</tr>
<tr>
<td>5 x 5 x 5</td>
<td>790</td>
<td>18</td>
<td>43.88x</td>
</tr>
<tr>
<td>7 x 7 x 7</td>
<td>920</td>
<td>31</td>
<td>29.67x</td>
</tr>
<tr>
<td>11 x 11 x 11</td>
<td>1220</td>
<td>89</td>
<td>13.70x</td>
</tr>
<tr>
<td>17 x 17 x 17</td>
<td>2130</td>
<td>570</td>
<td>3.73x</td>
</tr>
</tbody>
</table>

Time spent processing a single inline slice
Input volume: 1.1GB – Inline resolution: 951 x 462
CPU: i7 3970x - GPU: GeForce GTX 580
Conclusion

• The use of a massively parallel architecture to calculate the curvature attribute can lead to great improvement

• GPU architecture fits better to stencil computation compared to CPU

• The use of domain characteristics can avoid compute a lot of data that might never be used
Conclusion

• The use of a **massively parallel** architecture to calculate the curvature attribute can lead to **great improvement**

• GPU architecture fits better to stencil computation compared to CPU

• The use of domain characteristics, can avoid compute a lot of data that might never be used
Conclusion

• The use of a **massively parallel** architecture to calculate the curvature attribute can lead to great improvement

• GPU architecture fits better to stencil computation compared to CPU

• The use of **domain characteristics**, can avoid compute a lot of data that might never be used
Thank you

• Questions???