Common Reflection Surface using Particle Swarm Optimization on Graphical Processing Unit Devices

Gabriel Araujo*, Felipe Amorim, Luiz Popoff, Pedro Caceres, Wander Amorim

Central de Imageamento Geofisico (CImaGeo), Natal, Rio Grande do Norte, Brazil

This paper was prepared for presentation at the 16^{th} International Congress of the Brazilian Geophysical Society, held in Rio de Janeiro, Brazil, 19-22 August, 2019.

Abstract

In this work we described an implementation of 2D Common Reflection Surface Stack (CRS2D) using the Particle Swarm Optimization (PSO). This implementation was done taking advantage of the parallelism properties of the GPU devices. We implemented the PSO on GPU devices using CUDA API for C language. we achieved higher computation speed in CRS parameters estimation and stack than the conventional CPU processing.

Introduction

One of the most important stages of seismic processing is the Common Depth Point (CDP) stacking, because in this stage, the information coming from the same reflector in subsurface is summed, enhancing the coherent signal and mitigating the noise present on the acquisition. The efficiency of CDP stacking depends on the quantity of traces belonging to a CDP gather, known as fold. The fold depends on the geometrical parameters of the acquisition as the number of channels per shot and the distances between shots and receivers (Cordsen et al., 2000).

Some seismic datasets have low fold and high noise content. For that reason, the stacked section obtained from CDP stack has low signal-to-noise ratio. To improve the fold, (Jäger et al., 2001) proposed the Common Reflection Surface (CRS) Stack, where to generate a stacked trace it is used, besides the traces of the CDP to be stacked, traces from the neighboring CDP gathers. The CRS use an alternative traveltime equation, different from the CDP stack traveltime equation.

For 2D, this equation requires three parameters that are related with dip of reflectors and the curvature of NIP and N waves (Jäger et al., 2001; Hubral, 1983). These three parameters must be obtained from the seismic data, for which is necessary to carry out a parameter search which is performed finding the parameters that maximizes the semblance (Neidell and Taner, 1971).

Because parameter search is performed for all samples of the final stacked section, this search is carried out thousand of times and the result obtained for one sample does not depend on the result of the neighboring samples. That means that it must be used a faster algorithm to find the maximum of a function and a parallelism computing scheme to do this procedure several times at the same time.

For this reason we implemented the CRS parameter search using the particle swarm optimization (PSO) that is a fast heuristic function optimizer and this was implemented in a GPU device, using the CUDA API developed by NVIDIA® for C language. Using this implementation we obtained a good performance of the CRS parameter search and stack.

Methodology

Common Reflection Surface

The conventional CDP stack uses an equation for traveltime t define by equation

$$t^2 = t_0^2 + \frac{x^2}{v^2} \tag{1}$$

where x is the source-receiver offset of the trace, t_0 is the traveltime for zero offset. This equation uses only one parameter to stack, that is the stacking velocity v. On the other hand, the CRS-2D uses the following traveltime equation (Jäger et al., 2001; Faccipieri et al., 2016)

$$t^2 = (t_0 + a(m - m_0))^2 + B(m - m_0)^2 + \frac{x^2}{v^2} \tag{2}$$

This equation uses three parameters to stack, a, B and v. The parameter a is related with the local apparent slope at zero offset, B is related with the local curvature at zero offset and v is the same stacking velocity as used in the equation 1. The variable m is the midpoint coordinate and m_0 is the central CDP coordinate where the trace is stacked. The distance $m - m_0$ is calculated for all traces within a CRS aperture.

This aperture is chosen as an interval of some hundred of meters around the central CDP. In our implementation, the quantity $m - m_0$ in equation 2 is calculated using the CDP binning numeration. If the distance between two adjacent CDP gathers is ΔCDP, then

$$m - m_0 = (CDPN - CDPN_0) * DeltaCDP \tag{3}$$

Where $CDPN$ and $CDPN_0$ are the CDP bin numbers associated to the coordinates m and m_0 respectively. This method allows to calculate easily the apertures and to choose the CDP gathers inside the apertures.

CRS parameters estimation

The values for the three CRS parameters must be estimated for each time sample of the zero offset section. This estimation is done by finding the maximum value of semblance S, that according to Neidell and Taner (1971) is defined as

$$S = \frac{\sum_{k=k_0-w/2}^{k_0+w/2} \left(\sum_{i=1}^{M} u_{i,t(i)} \right)^2}{M \sum_{k=k_0-w/2}^{k_0+w/2} \sum_{i=1}^{M} u_{i,t(i)}^2} \quad (4)$$

where $u_{i,t(i)}$ is the sample amplitude at the traveltime $t(i)$ calculated for ith trace on the gather and M is the number of traces. w is the length of sample window used to calculated the semblance. This semblance is a measure o coherence between the sample amplitudes and traveltime calculated, therefore, the maximum value of the semblance means that the traveltime calculated fits the best the data.

For parameter estimation, a search space must be built in order to restrict the parameters to physically feasible values. To restrict the velocity values we use a velocity guide which is nothing more than velocity function interpreted by the data processing analyst in the processing sequence. The search space is a percent variation of this velocity guide, usually around 10% or 20%. For a parameter we measure the steepest slope in the CDP section and we let the values vary between the positive and negative value of this slope. Due to the fact that the value of curvature can not be obtained easily in the seismic section, we tested some possible values for parameter B empirically. We used B values of the order of $10^{-7} s^2/m^2$.

Particle Swarm Optimization

The particle swarm optimization (PSO) (Kennedy and Eberhart, 1995) arose from a proposal to model the social behavior of some animals. Imitating the behavior of a group of animals looking for food, a group of particles finds a extreme point of a function. Differently from the gradient based algorithms that are local optimization strategies, the PSO is a global optimization strategy because it avoids to fall to local extremes of the function. As pointed out by Conn et al. (2009), this kind of algorithm are good enough for functions that depends on few variables (less than 100). Therefore, the PSO is ideal to use in the CRS parameter estimation by maximizing the semblance.

In PSO, a particle is a candidate solution for the optimization problem. Initially, a number N of particles are located in random positions in the search space. If at the iteration $i-1$, the particle j has a position $x_{i-1,j}$, the new position for iteration is calculated according the following evolution rule

$$x_{i,j} = x_{i-1,j} + vel_{i,j} \quad (5)$$

where the quantity $vel_{i,j}$, known as particle velocity, is calculated as

$$\begin{aligned} vel_{i,j} &= w * vel_{i-1,j} + c_1 * r_1 * (xbest_{i,j} - x_{i-1,j}) \\ &\quad + c_2 * r_2 * (xglobal_i - x_{i-1,j}) \end{aligned} \quad (6)$$

Where r_1 and r_2 are random number between 0 and 1, c_1 and c_2 are known as cognitive and social parameters respectively, w is called the inertial factor and is a number to attenuate the effect of the velocity of the last iteration. This attenuation is performed choosing an initial value w_{ini} and multiplying this value by a damping factor $w_{damp} < 1$.

The quantity $xbest_j$ is the best solution found by the particle j and $xglobal$ is the best solution found by all the swarm. After some iterations the solution is provided by the value of $xglobal$. Although there are other heuristic algorithms used in CRS parameters estimation like the differential evolution (DE) algorithm (See Barros et al., 2015; Faccipieri et al., 2016), we decided to use PSO because its implementation is straightforward in GPU programming.

GPU implementation using CUDA

Because the seismic acquisitions are increasing in size, it is necessary to develop strategies to process huge quantity of data in a parallel way. For this reason the use of GPU programming is gaining more importance in the processing of seimic data, for example, in reverse time migration (Wang et al., 2018) or Full Waveform Inversion (Wang et al., 2011).

In order to implement the PSO search for CRS parameters in GPU devices, we used the CUDA API, developed for C language by NVIDIA®. In our paralellism strategy we put in the GPU memory all the traces in the CRS aperture for one CDP. We distribute the GPU threads in such a way that one thread corresponds to each particle that looks for parameters in each sample and we calculate semblance values for all particle and samples simultaneously.

Results

To asses the implementation of our methodology, we used a real dataset from the Tacutu Basin, at northern part of Brazil. The data are from seismic 2D line 50-RL-90 acquired as part of a exploration project conducted by Petrobras at early 80's. The shots were recorded with 96 channels with a 50m receiver interval using a split-spread configuration. The record length was 4s and sample interval was 0.004s. The minimum and maximum absolute offsets of 150 m and 2500 m, respectively. Shot interval was 200m. The nominal fold obtained from these acquisition parameters was 12.

This line was processed with a standard processing sequence, applying statics, noise attenuation, deconvolution and doing a user velocity picking. A conventional CDP stack was done with the help of the user velocity and the equation (1) obtaining the stacked section in the Figure 1(a). Due to the low fold, short offset and the distance between receivers and sources exists some parts of the stack with low signal-to-noise ratio, mainly the steepest reflectors at CDP 700 and CDP 1300.

The CRS stack was applied with the implementation described in the methodology section. The aperture used for parameter search and stack was 500m. The CRS search space for parameter estimation was built as follows: the velocity was searched varying by 20% the values of the user velocity picking. For parameters a and B were searched at the ranges $[-3.5 \times 10^{-4}, 3.5 \times 10^{-4}] s/m$ and $[-5.0 \times 10^{-7}, 5.0 \times 10^{-7}] s^2/m^2$, respectively.

Search Alg.	Comp. Device	Approx Runtime (min)
Brute Force	CPU	8345
PSO	CPU	587
PSO	GPU	1.47

Table 1: Runtimes for CRS parameters estimation.

The resulting CRS stack is shown in Figure 1(b). The signal-to-noise was improved throughout section and some reflectors that barely appeared at CDP stack, in the CRS section showed a good continuity. A detail showing a zoomed window on the seismic section for the CDP stack and CRS stack is shown in the Figure 1(c). The slanted reflectors that was hardly noticed in CDP stack, appeared and were enhanced in the CRS stack.

The estimated values for CRS parameters a, B and v are shown in Figure 2. The values obtained for a and B agreed with the geometrical features of reflectors observed on the seismic section. For example, the a value showed a negative value at left flank of the anticline and a positive value at the right flank. The CRS added up a high frequency component to the stacking velocity and this helped to improve the strength of the stack of some reflectors, for example, the flat layers at the right side of the seismic section.

To show the improvement in computational speed we implemented three versions of the CRS parameters estimation. In the first one we implemented a brute force in the CPU search as follows. We discretized the search space, taking 42 values between minimum and maximum of velocity, and 24 values between extreme values of a and B. We calculated the semblance for the 24192 possible combination of parameters. The resulting parameters are the combination that maximized semblance value. In the second version we implemented in the CPU the PSO search described on the last section. We used the following PSO parameters: 30 particles, 50 iterations, $c1 = 1$, $c2 = 1$, $w_{ini} = 1$ and $w_{damp} = 0.99$. For these CPU implementations we use one CPU of a Intel i7-3770 CPU @ 3.40 GHz computer.

In the third one we implemented the same PSO search in GPU using CUDA programming and we used the same parameters of the second case. The CPU versions was implemented using C language. An approximate for run times are summarized in Table 1. For GPU implementation we use a NVIDIA GTX 1080Ti with 3584 CUDA cores. In our numerical test, the GPU programming increased the computation speed, turning the CRS 400 times faster than the version programmed in CPU and 5600 times faster than brute force search programmed in CPU. It must be noticed that the CPU runtimes can be improved in a cluster with several CPUs or even with the OpenMP API but these improvement depends on the quantity of the CPUs available.

Conclusions

The CRS parameter estimation using a implementation on a GPU device of the PSO showed a great efficiency in stack quality improvement and computational speed. We obtained the expected quality of the CRS stacking method but we increased computational speed using the parallelism properties of the GPU devices. The computational efficiency obtained lead to the possibility to perform more tests on the choice of CRS parameters without spent much processing time.

References

Barros, T., R. Ferrari, R. Krummenauer, and R. Lopes, 2015, Differential evolution-based optimization procedure for automatic estimation of the common-reflection surface traveltime parameters: GEOPHYSICS, **80**, WD189–WD200.

Conn, A. R., K. Scheinberg, and L. N. Vicente, 2009, Introduction to derivative-free optimization: Siam, **8**.

Cordsen, A., M. Galbraith, and J. Peirce, 2000, Planning land 3-d seismic surveys: Society of Exploration Geophysicists.

Faccipieri, J. H., T. A. Coimbra, L.-J. Gelius, and M. Tygel, 2016, Stacking apertures and estimation strategies for reflection and diffraction enhancement: GEOPHYSICS, **81**, V271–V282.

Hubral, P., 1983, Computing true amplitude reflections in a laterally inhomogeneous earth: GEOPHYSICS, **48**, 1051–1062.

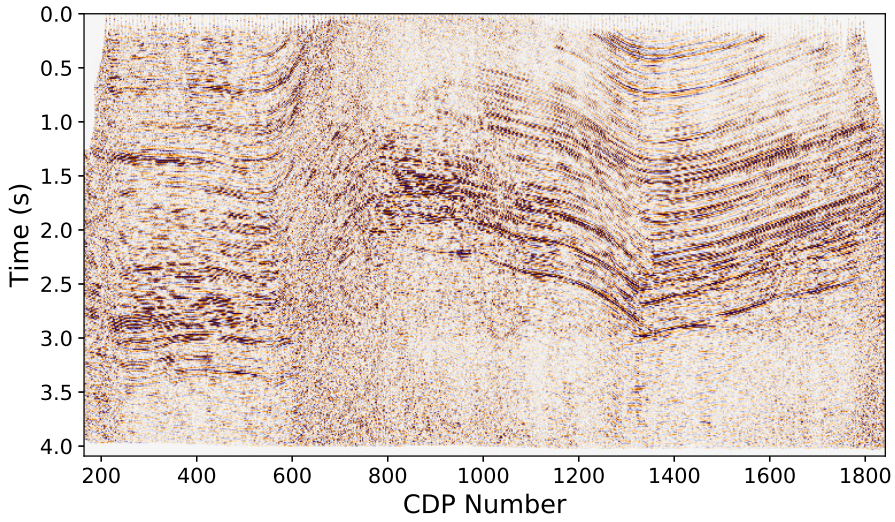
Jäger, R., J. Mann, G. Höcht, and P. Hubral, 2001, Common-reflection-surface stack: Image and attributes: GEOPHYSICS, **66**, 97–109.

Kennedy, J., and R. Eberhart, 1995, Particle swarm optimization: Presented at the Proceedings of ICNN95 - International Conference on Neural Networks, IEEE.

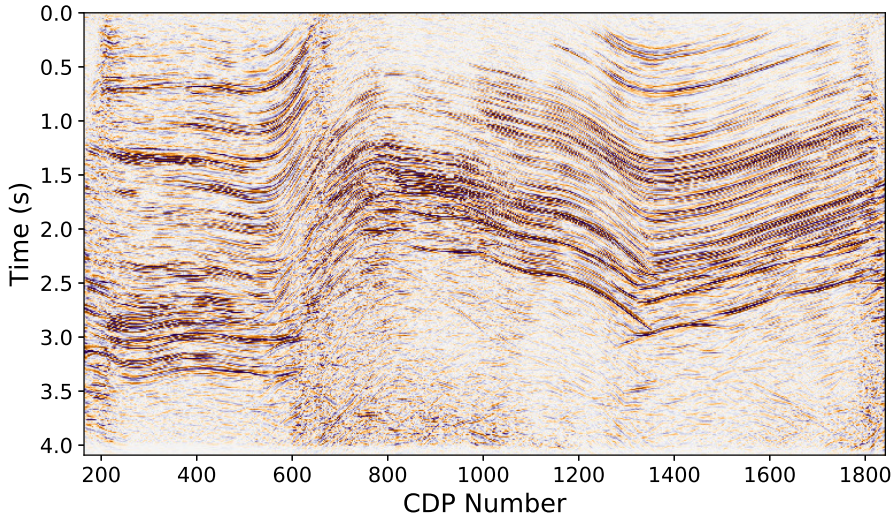
Neidell, N. S., and M. T. Taner, 1971, SEMBLANCE AND OTHER COHERENCY MEASURES FOR MULTICHANNEL DATA: GEOPHYSICS, **36**, 482–497.

Wang, B., J. Gao, H. Zhang, and W. Zhao, 2011, Cuda-based acceleration of full waveform inversion on gpu, *in* SEG Technical Program Expanded Abstracts 2011: Society of Exploration Geophysicists, 2528–2533.

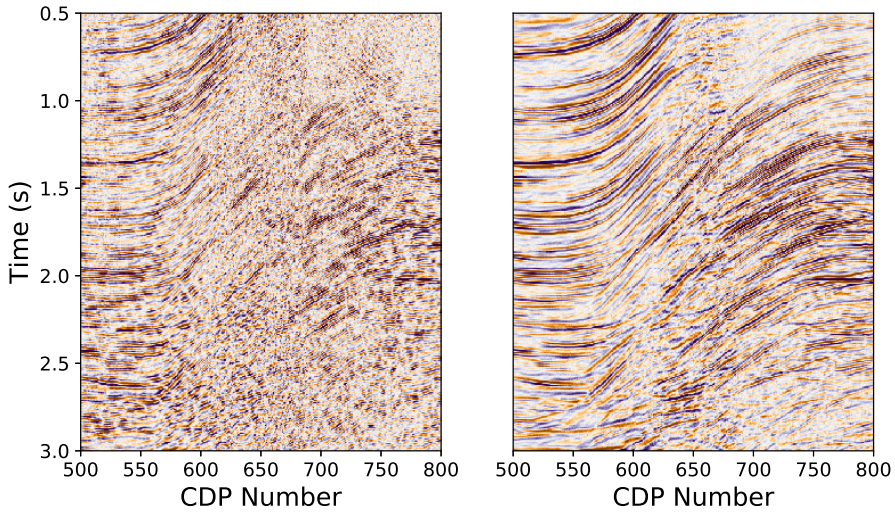
Wang, Y., H. Zhou, X. Zhao, Q. Zhang, P. Zhao, X. Yu, and Y. Chen, 2018, Cu q-rtm: A cuda-based code package for stable and efficient q-compensated rtm: Geophysics, **84**, 1–69.



(a)

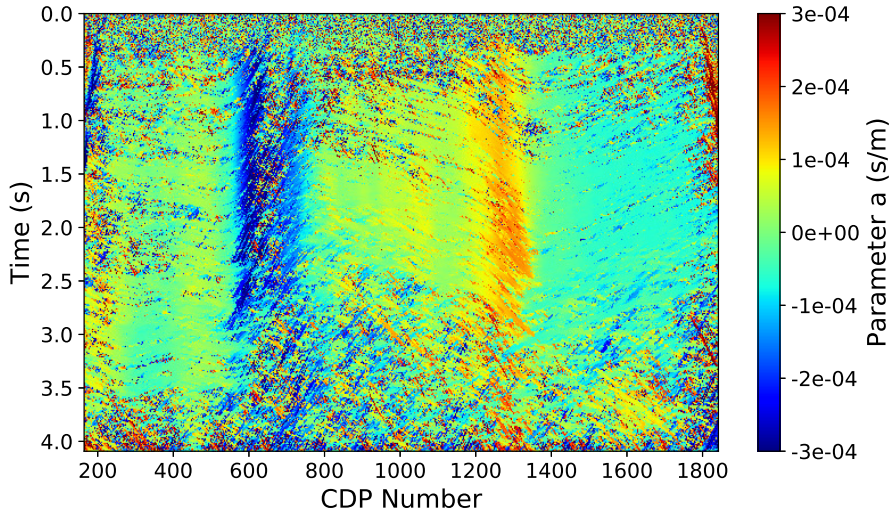


(b)

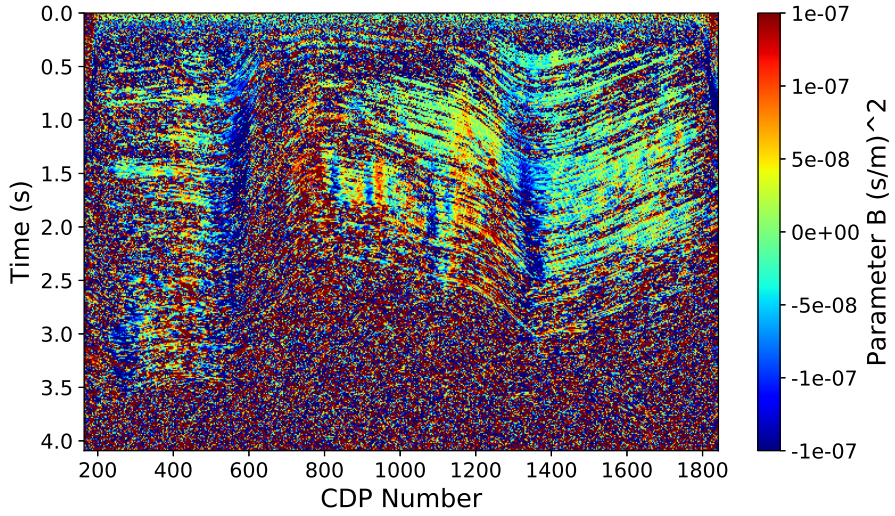


(c)

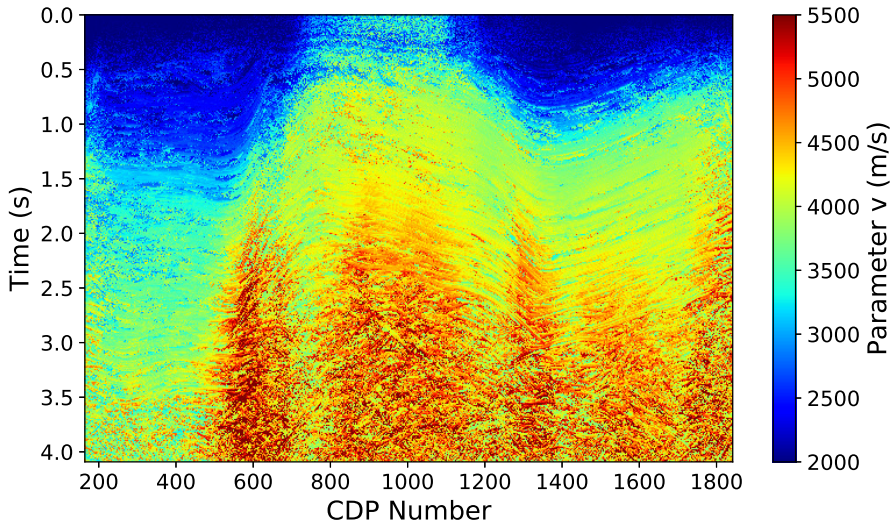
Figure 1: (a) Conventional CDP stack for Tacutu line (b) CRS Stack (c) Comparison of a zoomed detail. The CRS enhanced the slanted reflectors.



(a)



(b)



(c)

Figure 2: CRS stack parameters (equation (2)) estimated using PSO search for Tacutu Line (a) Parameter a (b) parameter B (c) stacking velocity v