Petroleum Science 21 (2024) 903–917

Contents lists available at ScienceDirect

Petroleum Science

journal homepage: www.keaipublishing.com/en/journals/petroleum-science

Original Paper

Automatic velocity picking based on optimal key points tracking algorithm

Yong-Hao Wang [a, b, c, d], Wen-Kai Lu [a, b, c, d, *], Song-Bai Jin [a, b, c, d], Yang Li [a, b, c, d], Yu-Xuan Li [a, b, c, d], Xiao-Feng Gu [a, b, c, d]

[a] *The Institute for Artificial Intelligence (THUAI), Tsinghua University, Beijing, 100084, China*
[b] *State Key Laboratory of Intelligent Technology and Systems, Tsinghua University, Beijing, 100084, China*
[c] *Beijing National Research Center for Information Science and Technology (BNRist), Beijing, 100084, China*
[d] *The Department of Automation, Tsinghua University, Beijing, 100084, China*

A R T I C L E I N F O

Article history:
Received 5 January 2023
Received in revised form
28 August 2023
Accepted 16 November 2023
Available online 27 December 2023

Edited by Jie Hao and Meng-Jiao Zhou

Keywords:
Velocity picking
Multi-object tracking
Density clustering
Kalman filter

A B S T R A C T

Picking velocities from semblances manually is laborious and necessitates experience. Although various methods for automatic velocity picking have been developed, there remains a challenge in efficiently incorporating information from nearby gathers to ensure picked velocity aligns with seismic horizons while also improving picking accuracy. The conventional method of velocity picking from a semblance volume is computationally demanding, highlighting a need for a more efficient strategy. In this study, we introduce a novel method for automatic velocity picking based on multi-object tracking. This dynamic tracking process across different semblance panels can integrate information from nearby gathers effectively while maintaining computational efficiency. First, we employ accelerated density clustering on the velocity spectrum to discern cluster centers without the requirement for prior knowledge regarding the number of clusters. These cluster centers embody the maximum likelihood velocities of the main subsurface structures. Second, our proposed method tracks key points within the semblance volume. Kalman filter is adopted to adjust the tracking process, followed by interpolation on these tracked points to construct the final velocity model. Our synthetic data example demonstrates that our proposed algorithm can effectively rectify the picking errors of the clustering algorithm. We further compare the performances of the clustering method (CM), the proposed tracking method (TM), and the variational method (VM) on a field dataset from the Gulf of Mexico. The results attest that our method offers superior accuracy than CM, achieves comparable accuracy with VM, and benefits from a reduced computational cost.

1. Introduction

Velocity analysis serves as a crucial step in seismic data processing. Taner and Koehler (1969) first proposed picking normal moveout (NMO) velocity from the dominant semblance trend in velocity spectral. In the case of Taner and Koehler (1969), semblances serve to measure NMO gather flatness. Generally speaking, picking velocity from the velocity spectrum is a time-consuming task. It requires an expert to visually examine a large number of semblance panels. With the increasing quantity of seismic data, researchers have gradually developed automatic velocity analysis algorithms.

Toldi (1989) proposed one of the first automatic velocity analysis methods. It aimed to maximize the stacking power along the moveout curves to obtain the best velocity model, which was represented by possible interval velocities. Based on this method, a series of methods adopted a similar idea to optimize the objective function defined on semblance for automatic velocity analysis (Symes and Carazzone, 1991; Mulder and ten Kroode, 2002, Symes, 1998). Moreover, Fomel (2009) further developed this method. It could be solved using a variational approach by taking the problem of automatic velocity picking on semblance spectra as a ray-tracing problem. Almarzoug and Ahmed (2012) studied the performance of this approach with real seismic data. A further extension of the

* Corresponding author.
E-mail address: lwkmf@mail.tsinghua.edu.cn (W.-K. Lu).

https://doi.org/10.1016/j.petsci.2023.11.019

method to the case of non-hyperbolic moveout was presented by Zhang et al. (2014). Machine learning (ML) based methods were also proposed for velocity picking. Waheed et al. (2019) performed automatic velocity picking based on the DBSCAN (density-based spatial clustering of applications with noise) algorithm. Wang et al. (2022) proposed an unsupervised clustering intelligent velocity picking method based on the Gaussian mixture model (GMM) and provided uncertainty analysis as quality control.

With the development of neural networks (LeCun et al., 2015), deep neural networks (DNN) had already been successfully applied in geophysics, such as seismic data processing (Zhu et al., 2019) and seismic interpretation (Wrona et al., 2021). Araya-Polo et al. (2018) used a calculated semblance cube as the input feature for deep learning to perform velocity picking. Biswas et al. (2018) adopted a recurrent neural network (RNN) to calculate stacking velocity directly from seismic data. Zhang et al. (2019) combined the you only look once (YOLO) (Redmon et al., 2016) and long short-term memory (LSTM) (Graves et al., 2013) to construct an algorithm for automatic velocity picking. Wang et al. (2021) designed and compared two different neural networks: classification and regression networks, for the extraction of root mean square velocity from semblance data. The limitations of deep learning (DL)-based methods were that these methods required a large amount of high-quality labeled data for training. Additionally, when applied to unseen data, additional transfer learning was needed, and the training process was also time-consuming.

Another way to deal with the problem of velocity estimation is to use local event slopes. Local event slopes measured on prestack seismic reflection data could be directly mapped to seismic velocities and all other moveout parameters (Fomel, 2007). Zhang et al. (2015) proposed a local event slope-mapped velocity spectrum with the histogram analysis algorithm. A further extension of the method was proposed by Zhang and Lu (2016), who developed an accelerated clustering algorithm for automatic velocity estimation. However, how to incorporate information from nearby gathers for velocity picking remains an open problem. Decker and Fomel (2022) proposed a variational approach for picking velocity fields from semblance volumes. This approach used spatially adjacent information for determining optimal surfaces from semblance-like volumes, but it was computationally expensive.

In this paper, we propose a novel method for automatic velocity picking based on multi-object tracking. The dynamic tracking process through different semblance panels efficiently incorporates information from nearby gathers without manual intervention. First, the accelerated density clustering algorithm (Zhang and Lu, 2016) is performed on each semblance panel to obtain the cluster centers, which correspond to the maximum likelihood velocities of the main subsurface structures. Second, we take the cluster centers on the initial semblance panel as the key points to start tracking. ZNCC (zero mean normalized cross correlation) and the Hungarian algorithm (Kuhn, 1955) are applied to match the key points on adjacent semblance panels. Furthermore, the Kalman filter (Kalman, 1960) is adopted to obtain the optimal estimate of the location of the key points on the next semblance panel. The tracking process is repeated until the key points are tracked to the last semblance panel. Consequently, the interpolation of these tracked points obtains a relatively high-accuracy stacking velocity model. With synthetic and field data examples, we demonstrate the performance of the proposed method. The remaining sections of this article are structured as follows: Section 2 details the theory underpinning our method; in Section 3, we present our experimental results; Section 4 contains a discussion on the method's robustness against noise and the impact of outliers on our tracking process; finally, Section 5 concludes the paper.

2. Methods

In this section, we illustrate the process of multi-object tracking for velocity picking, as shown in Fig. 1. We select an initial velocity spectrum and use the cluster points obtained by the accelerated density clustering algorithm proposed by Zhang and Lu (2016) as the initial points to start tracking along the profile. The Hungarian algorithm (Kuhn, 1955) and zero mean normalized cross correlation (ZNCC) (Stefano et al., 2005) are used together to search the position of the tracking points on the next velocity spectrum. At the same time, we also set some criteria to stop tracking certain points and add new points to be tracked. Because the tracking process is a dynamic process, we use the Kalman filter to obtain the optimal estimation of the tracking point position. Our tracking algorithm can accurately pick the velocity along the seismic horizon. Table 1 summarizes the important employed notation. First, in Section 2.1, we give a brief review of the accelerated density clustering algorithm (Zhang and Lu, 2016) used in our work. Then we explain in detail the process of the proposed multi-object tracking method using ZNCC and Hungarian algorithm in Section 2.2, and Kalman filtering process is introduced in Section 2.3.

2.1. Accelerated density clustering

In this section, we give a brief review of the accelerated density clustering algorithm (Zhang and Lu, 2016) used in our work. Density clustering is proposed to deal with general data distribution models by Rodriguez and Laio (2014). The advantage of density clustering is that we do not need to know the prior information of the number of clusters. The cluster centers can be determined by modeling the local density ρ_i and the minimum distance δ_i from points of higher density for each point i. The simple cutoff local density ρ_i and the minimum distance δ_i of data point i are defined as follows:

$$\rho_i = \sum_j \lambda(d_{ij} - d_c) \tag{1}$$

$$\delta_i = \min_{j:\rho_j > \rho_i} (d_{ij}) \tag{2}$$

where d_{ij} is the distance between point i and point j, d_c is the cutoff distance, and $\lambda(\bullet)$ is a function that satisfies

$$\lambda(x) = \begin{cases} 0 & x \geq 0 \\ 1 & x \leq 0 \end{cases} \tag{3}$$

Cluster centers are picked as points with relatively large ρ_i and δ_i. We can identify centers by observing rapid change in β, which is the product of ρ and δ. However, it is completely infeasible to directly apply density clustering algorithm to seismic data (Zhang and Lu, 2016), so Zhang and Lu (2016) proposed an accelerated density clustering algorithm. An estimate of the local density is obtained by applying the 2D histogram analysis method to the mapped zero-offset time and stacking velocity:

$$\rho = H(l_1, l_2) = m(l_1, l_2) \tag{4}$$

where l_1 and l_2 represent the discretized zero-offset time and the discretized stacking velocity, respectively. $H(l_1, l_2)$ denotes the 2D histogram function and $m(l_1, l_2)$ denotes the number of local event slopes mapped zero-offset time and stacking velocity that together fall into the bin represented by l_1, l_2. More details about the accelerated density clustering algorithm could be found in Zhang and Lu (2016).

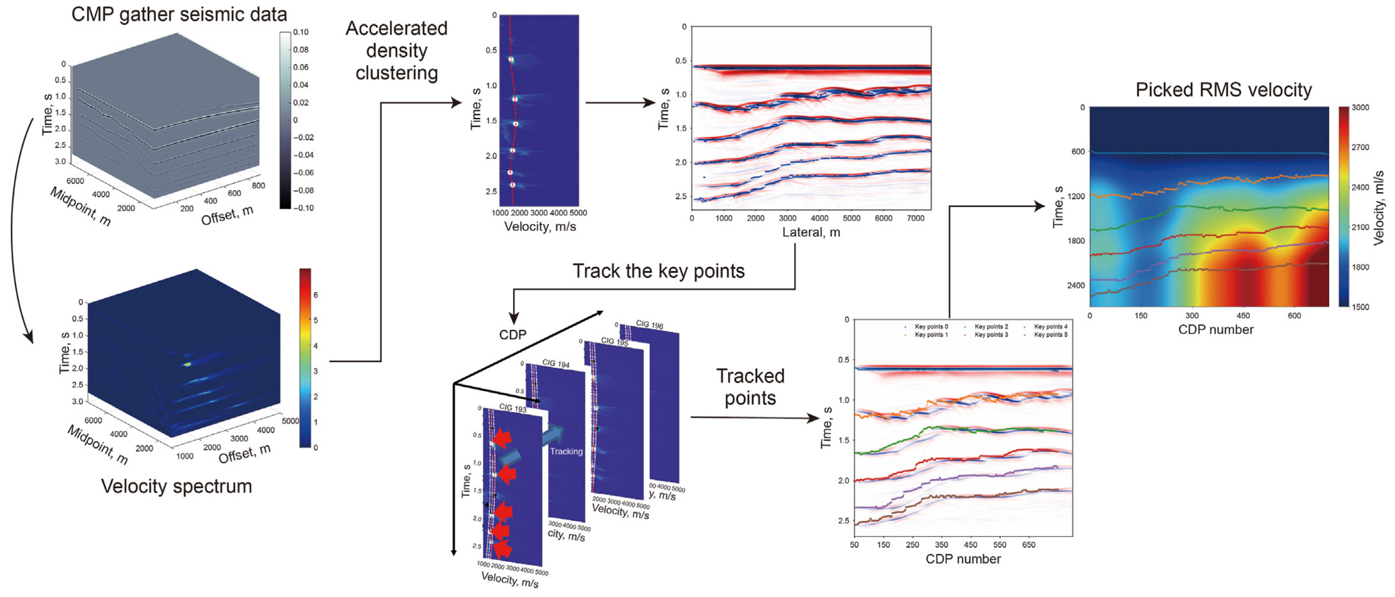

Fig. 1. The flowchart of the proposed method.

2.2. Key points tracking using ZNCC and Hungarian algorithm

By modeling the estimations of velocity and the location of each sample in a seismic gather into a Gaussian mixture model and using the accelerated density clustering algorithm, we can obtain the cluster center point representing the maximum likelihood velocity of the main subsurface structures. However, these cluster points are not continuous in the seismic line direction, and there may be outlier cluster center points. Therefore, we propose a multi-target key point tracking algorithm based on ZNCC and the Hungarian algorithm.

ZNCC is a common cross correlation calculation method in template matching. In image processing applications, ZNCC is more robust than NCC (normalized cross correlation) because it subtracts the mean value in the window and can resist changes in brightness. For a template $\mathscr{T}(x,y)$ with a subimage $\mathscr{F}(x,y)$, ZNCC is defined as follows:

$$ZNCC(\mathscr{T},\mathscr{F})=\frac{1}{n}\sum_{x,y}\frac{(\mathscr{T}(x,y)-\mu_{\mathscr{T}})(\mathscr{F}(x,y)-\mu_{\mathscr{F}})}{\sigma_{\mathscr{T}}\sigma_{\mathscr{F}}} \tag{5}$$

where n is the total number of elements in $\mathscr{T}$, x, y denote the row and column index, $\mu_{\mathscr{T}}, \mu_{\mathscr{F}}$ are the mean value of $\mathscr{T}$, $\mathscr{F}$, and $\sigma_{\mathscr{T}}, \sigma_{\mathscr{F}}$ are standard deviation of $\mathscr{T}$, $\mathscr{F}$. In our case, the template $\mathscr{T}(x,y)$ is a patch of the current velocity spectrum $\mathbf{S}_i$ with the size $w\times\ w$ centered on the tracking point $p_{i,j}$, and the subimage $\mathscr{F}(x,y)$ is the next velocity spectrum $\mathbf{S}_{i+1}$.

Fig. 2 displays how to use ZNCC to calculate the position of the tracking points on the velocity spectrum corresponding to the next CMP (common midpoint). Denote the current velocity spectrum as $\mathbf{S}_i$, and for each tracking point $p_{i,j}$ on $\mathbf{S}_i$, we extract a patch with the size $w\times w$ centered on the tracking point $p_{i,j}$ and calculate the ZNCC values of the patch with respect to the next velocity spectrum $\mathbf{S}_{i+1}$. We set an appropriate threshold β and a search radius r. The point with the largest ZNCC value s among all the points whose distance to $p_{i,j}$ is less than r is selected as the tracking point $p_{i+1,j}$ on $\mathbf{S}_{i+1}$ if $s>\beta$.

However, using only ZNCC to track points poses two problems. (1) During the tracking process, there may be new key points to be tracked, but using ZNCC cannot find them. (2) The tracking process is susceptible to the choice of the threshold β it is easy to track the wrong points if β is too small or to lose the tracked points prematurely if β too large. Therefore, we need to use the cluster points obtained from accelerated density clustering on each velocity spectrum to aid key point tracking by matching clustering points on adjacent velocity spectra. Specifically, we calculate the distance from the current tracking point to the cluster points on the next velocity spectrum, and use the Hungarian algorithm to determine which cluster points on the next velocity spectrum match the current tracking point.

The Hungarian algorithm is used to solve the assignment problem, a fundamental combinatorial optimization problem. Its formal definition is given as follows (Ramshaw and Tarjan 2012): Given two sets $\mathscr{A}$ and $\mathscr{B}$ together with a weight function $W:\mathscr{A}\times\mathscr{B}\rightarrow\mathbb{R}$, find a mapping $f:\mathscr{A}\rightarrow\mathscr{B}$ such that the cost function

$$\begin{aligned}&\sum_{i=1}^{N}W(a_i,b_i)\ \text{ for }\ a_i\in\mathscr{A},b_i\in\mathscr{B}\\&\text{s.t.}N=\min(|\mathscr{A}|,|\mathscr{B}|),f(a_i)=b_i\end{aligned} \tag{6}$$

is minimized. In our case, the set $\mathscr{A}$ is the set of tracking points on the current velocity spectrum, and the set $\mathscr{B}$ is the set of cluster points on the next velocity spectrum. The weight function W is the Euclidean distance matrix between the tracking point and the cluster point. The Hungarian algorithm can find the optimal assignment of the tracking points to the cluster points, and the cluster points that are not assigned to the tracking points are considered as new key points when they are far enough from the existing key points.

The Hungarian algorithm solves the assignment problem in polynomial time. It is based on the theorem: If a row or a column of the cost matrix adds or subtracts a number at the same time, the optimal assignment of the new cost matrix is still the optimal assignment of the original cost matrix. It is a greedy algorithm, and the optimal assignment of the cost matrix is obtained by iteratively finding the minimum element in each row and subtracting it from the row, and then finding the minimum element in each column and subtracting it from the column. The process is repeated until all the elements in the cost matrix are non-negative. The assignment of the cost matrix is obtained by finding the zero elements in the cost matrix. Python library *scipy. optimize* provides a Hungarian algorithm implementation (Virtanen et al., 2020).

In brief, using Hungarian algorithm and ZNCC for tracking requires the following three steps:

1. The Hungarian algorithm is used to match each current tracking point $p_{i,j}$ to the cluster center $c_{i+1,j}$ on the next velocity spectrum. If $c_{i+1,j}$ is the cluster center assigned to $p_{i,j}$ and the distance $\left\|p_{i,j} - c_{i+1,j}\right\|$ between two points is less than the specified threshold r, the matching point $c_{i+1,j}$ will be set as the tracked point $p_{i+1,j}$.
2. For tracking point $p_{i,j}$ without an assigned clustered point, we extract a patch with the size $w \times w$ centered on the tracking point $p_{i,j}$ and calculate the ZNCC of the patch with respect to the next velocity spectrum $\mathbf{S}_{i+1}$, which denoted as *Score*. The new tracking point $p_{i+1,j}$ should satisfy: $p_{i+1,j} = \underset{\|p-p_{i,j}\|<r}{\operatorname{argmax}} (Score[p])$, s.t. $Score[p_{i+1,j}] > \beta$, where β is a predefined threshold.
3. For unassigned cluster points, if the minimum distance to all tracking points is greater than the specified threshold value d_ϵ, they will be set as new key points to be tracked.

The above algorithm could be summarized as Algorithm 1.

Algorithm 1. Key points tracking

Algorithm 1 Key points tracking

```
Input   Points to be tracked $\mathbf{p}_i$, clustered points $C_{i+1}$ on $\mathbf{S}_{i+1}$,velocity spectrum $\mathbf{S}_i$,$\mathbf{S}_{i+1}$, patch size $w$,
        threshold $\beta$,$r$ and $d_\epsilon$
Output  Tracked points $\mathbf{p}_{i+1}$
1:      $\mathbf{p}_{i+1} = \{\}$
2:      D = CalculateDistanceMatrix($\mathbf{p}_i$,$C_{i+1}$)
3:      $\mathbf{p}_a$ = HungarianAlgorithm(D) // $\mathbf{p}_a$ is the assignment of $\mathbf{p}_i$ to $C_{i+1}$ using Hungarian algorithm.
4:          for each $p_{i,j}$ in $\mathbf{p}_i$ do
5:              If $||\mathbf{p}_a(j) - p_{i,j}|| < r$ then // $\mathbf{p}_a(j)$ denotes the assigned cluster point to $p_{i,j}$.
6:                  $\mathbf{p}_{i+1}$.append($\mathbf{p}_a(j)$)
                Else
7:                  $Score = ZNCC(Patch(\mathbf{S}_i, p_{i,j}, w), \mathbf{S}_{i+1})$ // $Patch(\mathbf{S}_i, p_{i,j}, w)$ is the patch with the size of
        $w \times w$ and centered on $p_{i,j}$ from $\mathbf{S_i}$.
8:                  $p_s = \underset{||p - p_{i,j}|| < r}{\operatorname{argmax}} (Score[p])$
9:                  If $Score[p_s] > \beta$ then
10:                     $\mathbf{p}_{i+1}$.append($p_s$)
11:                 else
12:                     Stop tracking the point $p_{i,j}$
13:                 End if
14:             End if
15:         End for
16      If length($\mathbf{p}_l$)$> 0$
17          $D'$ = CalculateDistanceMatrix($\mathbf{p}_l$,$P_{i+1}$)
18          For each $p_l$ in $\mathbf{p}_l$ do
19              If $\min(D'(p_l)) > d_\epsilon$ then
20                  $\mathbf{p}_{i+1}$.append($p_l$) // Add new points to track
21              End if
22          End for
23      Return $\mathbf{p}_{i+1}$
```

Table 1
Summary of the important employed notation.

Symbol	Description
$\mathbf{S}_i$	The i-th velocity spectrum
$\mathbf{p}_i$	Tracked key points on the i-th velocity spectrum
C_i	The set of cluster centers of $\boldsymbol{S}_i$
w	Patch size
$\beta, r.d_\epsilon$	Threshold parameters
M	The number of velocity spectrum (also the number of CDPs)
$\mathbf{x}$	Mean variable of the Kalman filter
$\mathbf{V}$	Covariance matrix of the Kalman filter
$\mathbf{A}$	State transition matrix of the Kalman filter
$\mathbf{H}$	Observation matrix of the Kalman filter
$p_{i,j}$	The j-th tracked point on the i-th velocity spectrum, $p_{i,j} \in \boldsymbol{p}_i$
$c_{i,j}$	The j-th cluster center on the i-th velocity spectrum, $c_{i,j} \in C_i$

2.3. The optimal estimation of the tracked key points

Kalman filter is widely used in many fields such as communication (Battistelli et al., 2018), navigation (Cooper and Durrant-Whyte, 1994), guidance (Penizzotto et al., 2015) and control (Lee and Ricker, 1994). It uses the observed data of the system at each moment to provide a more accurate estimate of the state of the system. When the noise contained in the observed data of the system is assumed to obey the Gaussian distribution, the Kalman filter can obtain the optimal estimation of the system state (Kalman, 1960). According to Humpherys et al. (2012), if the process and measurement covariance are known, the Kalman filter is the best linear estimate in the sense of minimum mean square error, regardless of the Gaussian nature of the noise.

In our case, our key points tracking process requires the estimation of two state variables: (1) Mean variable **x**. It defines on the four-dimensional state space $(t, v, \dot{t}, \dot{v})$ that contains the location of the key point (t, v) on the velocity spectrum and their respective velocities $(\dot{t}, \dot{v})$ in image coordinates. (2) Covariance matrix **V**. It defines the uncertainty of the state variable **x** and is a 4× 4 diagonal matrix, with larger numbers in the matrix indicating greater uncertainty and can be initialized with any value. We use a standard Kalman filter with constant velocity motion and linear observation model, where we take the location of the key point (t, v) as direct observations of the object state.

The Kalman filter works by two-phase process. For the prediction phase, the Kalman filter produces estimates of the current state variables **x** and **V**. Once the tracked points (including noise) is obtained using Algorithm 1, these estimates are updated based on a weighted average, with greater weight given to estimates with greater certainty. The filter performs recursively and in real time, using only the present input measurements and the state calculated previously and its uncertainty matrix.

In the prediction phase, the Kalman filter will predict the state variables **x** and **V** at step s based on the previous state variables at step $s-1$:

$$\mathbf{x}_s = \mathbf{A}\mathbf{x}_{s-1} \tag{7}$$

$$\mathbf{V}_s = \mathbf{A}\mathbf{V}_{s-1}\mathbf{A}^{\mathrm{T}} + \mathbf{Q} \tag{8}$$

where $\mathbf{x}_{s-1} = [t_{s-1}, v_{s-1}, \dot{t}_{s-1}, \dot{v}_{s-1}]^{\mathrm{T}}$; **A** is the state transition matrix; **Q** is the noise matrix of the system, representing the reliability of the whole system, which is usually initialized to a very small value. In our case, the state transition matrix **A** is a 4 × 4 matrix with the following form:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \Delta s & 0 \\ 0 & 1 & 0 & \Delta s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

where Δs is the time interval during the dynamic tracking process between two consecutive velocity spectrum and is set to 1 in our implementation.

In the update phase, the Kalman filter will update the state variables **x** and **V** at step s based on the observation data. The update phase is described as follows:

$$\mathbf{y}_s = p_s - \mathbf{H}\mathbf{x}_s \tag{10}$$

$$\mathbf{Q}_s = \mathbf{H}\mathbf{V}_s\mathbf{H}^{\mathrm{T}} + \mathbf{R} \tag{11}$$

$$\mathbf{K} = \mathbf{V}_s\mathbf{H}^{\mathrm{T}}\mathbf{Q}_s^{-1} \tag{12}$$

$$\mathbf{x}_s = \mathbf{x}_s + \mathbf{K}\mathbf{y}_s \tag{13}$$

$$\mathbf{V}_s = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{V}_s \tag{14}$$

where p_s is the tracked points at step s with only observed state variables $p_s = (t, v)$; **R** is the noise matrix of the observation and can be initialized randomly; **H** is the observation matrix. In our case, the observation matrix **H** is a 2 × 4 matrix, and $\mathbf{x}_s$ is a vector. They have the following form:

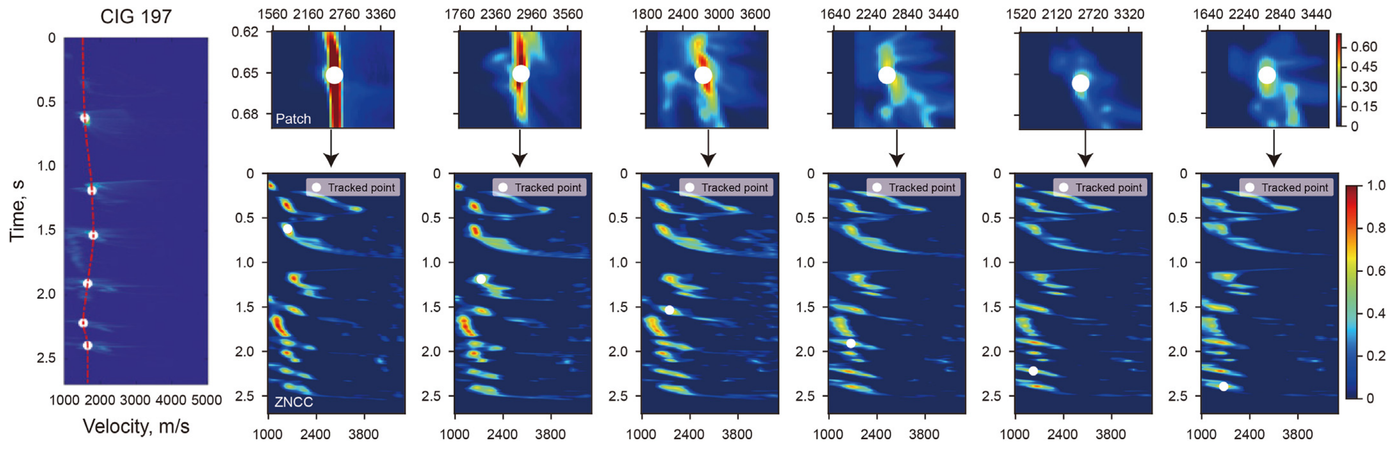

Fig. 2. Extract a patch with the size $w \times w$ centered on each tracking point and calculate the ZNCC value of each patch with the next velocity spectrum.

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \mathbf{x}_s = [t_s, v_s, \dot{t}_s, \dot{v}_s]^{\mathrm{T}} \tag{15}$$

Eq. (12) calculates the Kalman gain $\mathbf{K}$ for estimating the importance of the error $\mathbf{y}_s$. Eqs. (13) and (14) update the state variables $\mathbf{x}$ and $\mathbf{V}$. The Kalman filtering algorithm is summarized in Algorithm 2.

In summary, our method uses the velocity spectrum as input, and accelerated density clustering is performed to obtain the cluster centers on each velocity spectrum. Then, we select the cluster center point on the initial velocity spectrum $\mathbf{S}_0$ as the key point and start to use the tracking algorithm (Algorithm 1) to obtain the position of the key point on the next velocity spectrum. Because the noise in the velocity spectrum may cause deviation to the tracked key point position, we use the Kalman filtering algorithm to update the obtained key point position, retain all the tracked key points, and continue to track until the end. The overall tracking algorithm is summarized in Algorithm 3. When all key points are tracked, interpolation is performed to build the effective velocity model. Piecewise cubic Hermite interpolating polynomial (PCHIP) (Fritsch and Butland, 1984) is adopted as the interpolation function.

Algorithm 2. Kalman filtering

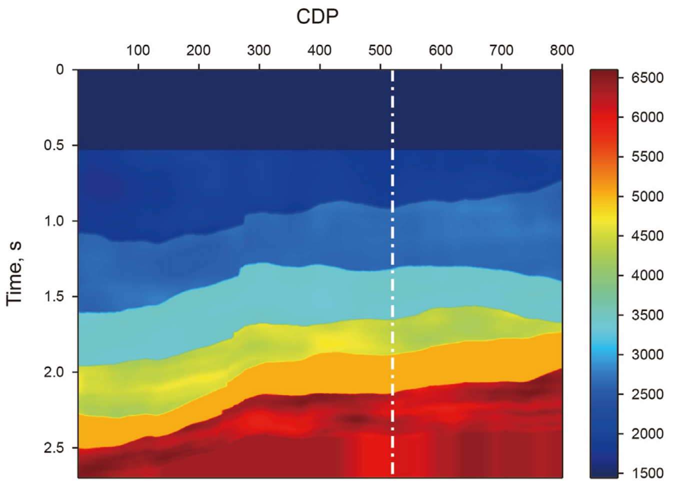

Fig. 3. Synthetic velocity model. White dashed line indicates the location of CDP 520.

automatically estimated in both cases, and NMO correction is performed with the estimated velocity. We illustrate our tracking process through a synthetic data example in section 3.1 and compare the effects of the clustering method (CM) (Zhang and Lu, 2016), the proposed tracking method (TM), and variational method (VM) (Decker and Fomel, 2022) on the field data set in section 3.2. Our work is implemented on a workstation with 13th Gen Intel(R) Core(TM) i9-13900K CPU and 128 GB RAM.

```
Algorithm 2 Kalman filtering
Input   Tracked points $p_s$ at step $s$, state variables $\mathbf{x}_{s-1}$ and $\mathbf{V}_{s-1}$ at step $s-1$.
Output  Updated state variables $\mathbf{x}_s$ and $\mathbf{V}_s$ at step $s$
1:    Function KALMANPREDICT($\mathbf{x}_{s-1}$, $\mathbf{V}_{s-1}$)
2:        $\mathbf{x}_s = \mathbf{A}\mathbf{x}_{s-1}$
3:        $\mathbf{V}_s = \mathbf{A}\mathbf{V}_{s-1}\mathbf{A}^{\mathrm{T}} + \mathbf{Q}$
4:        Return $\mathbf{x}_s, \mathbf{V}_s$
5:    End function
6:    Function KALMANUPDATE($\mathbf{x}_s$, $\mathbf{V}_s$, $p_s$).
7:        $\mathbf{y}_s = p_s - \mathbf{H}\mathbf{x}_s$
8:        $\mathbf{Q}_s = \mathbf{H}\mathbf{V}_s\mathbf{H}^{\mathrm{T}} + \mathbf{R}$
9:        $\mathbf{K} = \mathbf{V}_s\mathbf{H}^{\mathrm{T}}\mathbf{Q}_s^{-1}$
10        $\mathbf{x}_s = \mathbf{x}_s + \mathbf{K}y_s$
11        $\mathbf{V}_s = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{V}_s$
12        Return $\mathbf{x}_s, \mathbf{V}_s$
10:   End function
```

3. Experiments

To demonstrate the effectiveness of the proposed method, we apply it on synthetic and field data sets. Stacking velocity model is

3.1. Synthetic example

Our synthetic velocity model is shown in Fig. 3. The size of the synthetic model is 800 × 800, where the lateral direction *x* and the depth direction *z* were spaced 10 m apart (d*x* = 10 m, d*z* = 10 m). 2D const density acoustic wave equation is adopted to do the forward modeling, and finite difference is used during calculation.

Algorithm 3. Density clustering aided optimal key points tracking

```
Algorithm 3 Density clustering aided optimal key points tracking
Input   Local event slope mapped velocity spectrum $\{\mathbf{S}_i\}_{i=0}^{M-1}$, patch size $w$, threshold $\beta$, $r$ and $d_\epsilon$ // $M$ is the number of velocity spectrum
Output  The set of tracked key points $\mathbf{P}$
1:  $\mathbf{P}=\{\}$
2:  For i = 0 to $M-2$ do
3:      $C_{i+1}$ = AcceleratedDensityClustering($\mathbf{S}_{i+1}$) // $C_{i+1}$ is the set of cluster centers of $\mathbf{S}_{i+1}$, $C_{i+1}=\{c_{i+1,j}|j=1,\cdots,n_{i+1}\}$, $n_{i+1}$ is the number of cluster centers
4:      If i==0 then // Initialization
5:          $C_0$ = AcceleratedDensityClustering($\mathbf{S}_0$)
6:          $\mathbf{p}_0=C_0$
7:          $T=\{(\mathbf{x}_j,\mathbf{V}_j)|\mathbf{x}_j=[p_{0,j},0],\mathbf{V}_j=\mathbf{0}\}$ // Initialize the Kalman filter for each of the key points in $\mathbf{p}_0$
8:          $\mathbf{P}$.add($\mathbf{p}_0$)
9:      End if
10:     $\mathbf{p}_{i+1}$ = KeyPointsTracking($\mathbf{p}_i$, $C_{i+1}$, $\mathbf{S_i}$, $\mathbf{S_{i+1}}$, $w$, $\beta$, $r$, $d_\epsilon$) // Algorithm 1.
11:     For each $(\mathbf{x}_{i,j},\mathbf{V}_{i,j})$ in $T$ do
12:         $\mathbf{x}_{i+1,j},\mathbf{V}_{i+1,j}$ = KalmanPredict($\mathbf{x}_{i,j},\mathbf{V}_{i,j}$)
13:         $\mathbf{x}_{i+1,j},\mathbf{V}_{i+1,j}$ = KalmanUpdate($\mathbf{x}_{i+1,j},\mathbf{V}_{i+1,j},p_{i+1,j}$) // $p_{i+1,j}\in\mathbf{p}_{i+1}$
14      End for
15      If New key points in $\mathbf{p}_{i+1}$ then
16          Add new track to $T$ based on new points
17      End if
18      $\mathbf{P}$.add($\{\mathbf{H}\mathbf{x}_{i+1,j}|\mathbf{x}_{i+1,j}\in T\}$) // Add the tracked points on $\mathbf{S}_{i+1}$ to $\mathbf{P}$, $\mathbf{H}$ is the observation matrix, which project the state variable $\mathbf{x}$ to observations.
19  End for
20  Return $\mathbf{P}$
```

A 20 Hz Ricker wavelet is deployed in our modeling with 350 shots in total. The source position corresponding to the first shot is 10 m, the depth is 10 m, and the shot spacing is 20 m. The depth of the receiver is 10 m, and it is located on the right side of the shot point. The minimum offset is 10 m, the receiver spacing is 10 m, and the number of receivers per shot is 99. The time sampling interval of single shot recording is 1 ms, and the recording length is 2.7 s.

Fig. 4(a) shows a single synthetic CMP gather, with the cluster center points obtained by the CM depicted in Fig. 4(b). Detailed insight into the dynamic tracking process is offered in Fig. 5, where the key points on CDP 705 are tracked using our algorithm, thus deriving the tracking points on CDP 706. By leveraging information from adjacent gathers, our tracking algorithm successfully identified the key points missed by the clustering algorithm on CDP 706, thereby demonstrating its effectiveness.

The tracking results are illustrated in Fig. 6, where the seismic data is overlaid with clustering points for clarity. The clustering points' positions denote the location of the automatic velocity picking, which ideally should align with the reflection surface's

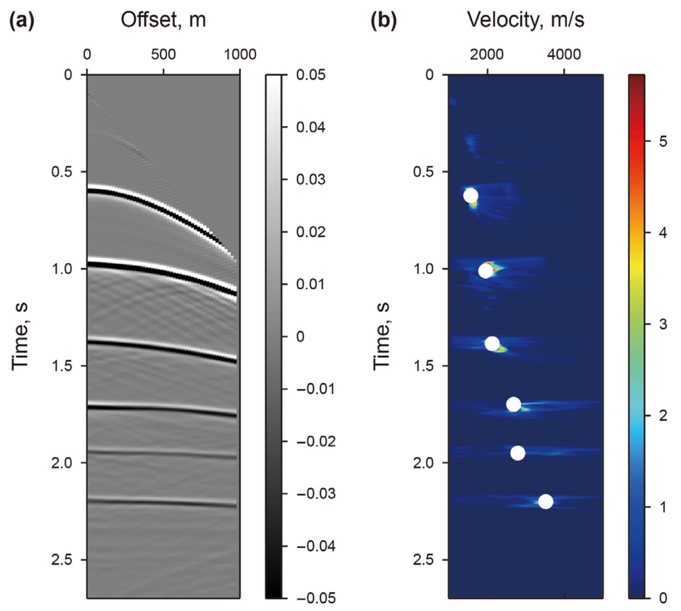

Fig. 4. (**a**) The stacked image using NMO-corrected CMP gathers by our TM. (**b**) Interpolated velocity model by TM. (**c**) The stacked image using NMO-corrected CMP gathers by CM. (**d**) Interpolated velocity model by CM.

position. Hence, this display method serves as an effective means of evaluating the accuracy of the obtained clustering points.

As can be seen from Fig. 6(a), the cluster centers obtained by the CM are not continuous along the reflector, and there are still some outliers, as shown by the arrow in the figure. In contrast, the key points tracked by the proposed method can effectively improve the problems of discontinuity and outliers, as shown in Fig. 6(b). Fig. 7 displays the comparison of NMO correction results by different methods. It can be seen that the NMO-corrected CMP gather using the velocity picked by TM is flatter than that using the CM, especially at the position indicated by the red dashed line, which shows that our method can pick up stacking velocity more accurately. The reason why the CM fails is that it does not find the shallow cluster centers, which leads to the inaccurate interpolation velocity at the shallow part. However, the tracking-based method can track the key points that should exist at the shallow layer by incorporating the information of previous gathers to obtain more accurate velocity. Fig. 8 shows the stacked image using NMO-corrected CMP gathers and the corresponding interpolated velocity model. From the position indicated by the white arrow, it can be seen that due to the inaccurate velocity obtained by CM, the corresponding stacking results have low resolution in the shallow layer and obvious edge effects. The tracking method can capture shallow velocity well, and even at the edges, it has good stacking results.

3.2. Field example

To validate the performance of the proposed method on real seismic data, we select the Gulf of Mexico data set (Claerbout and Black, 2001) available in Madagascar package (Alkhalifah et al., 2006). Fig. 9 displays the tracked points and cluster center points on the velocity spectrum of different CDPs. It can be seen that the tracked points are better distributed along the high value of the velocity spectrum than the clustered points, and it follows the dominant trends in the semblance and is more closely and continuously distributed.

As shown in Fig. 10, the tracked points can be better distributed along the geological horizon compared with the cluster points. As illustrated in Fig. 11(a), CM fails to find the cluster centers exceeding 2.4 s, which results the error in the picking velocity, especially at the deep area. It can be seen from Fig. 11 that the velocity picked by TM and VM is more consistent with the dominant trend of the velocity spectrum, and the NMO corrected gathers are flatter (as indicated by the red dotted lines). Fig. 12 shows the enlarged content of the green rectangular box in Fig. 11. It can be seen that the events of TM and VM at the positions indicated by the red arrows are flatter than those of CM, showing the comparable performance between TM and VM. Fig. 13 shows the velocity models

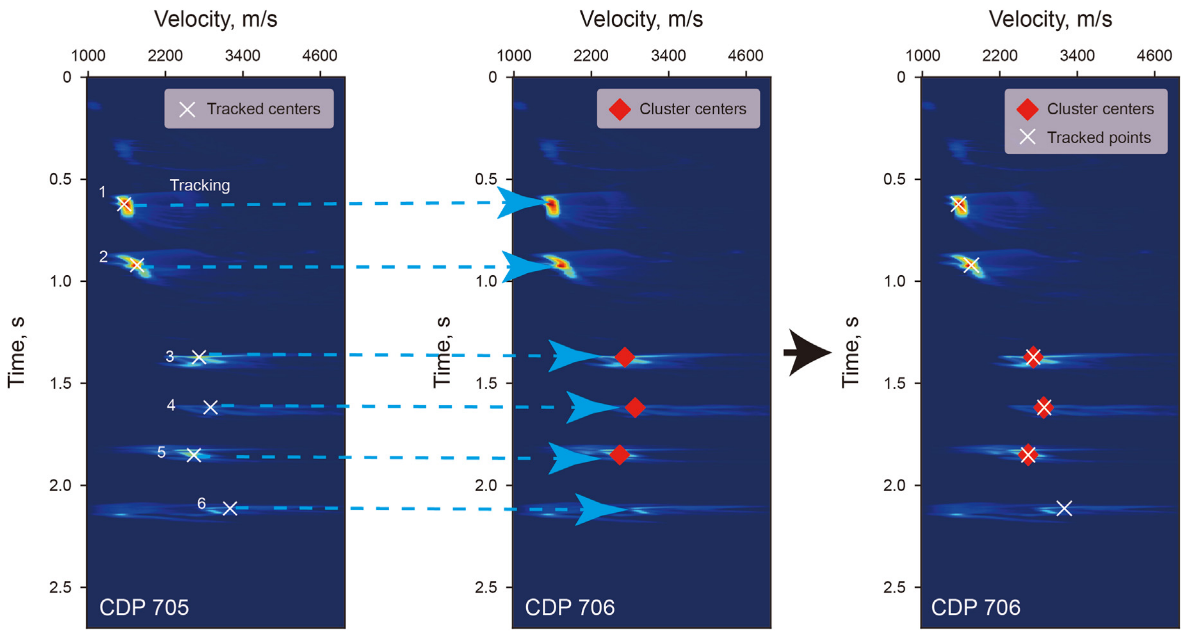

Fig. 5. The dynamic tracking process. The key points on CDP 705 were tracked using a tracking algorithm to obtain the tracking points on CDP 706. Points 1, 2, and 6 are tracked by ZNCC matching, and points 3, 4, 5 are tracked by Hungarian algorithm.

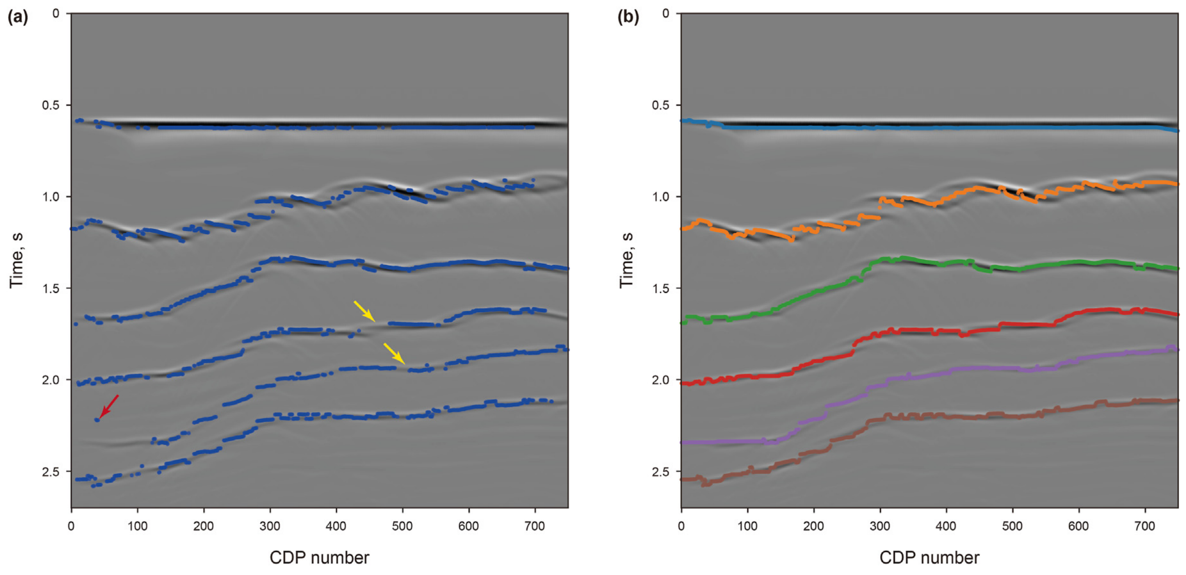

Fig. 6. (**a**) The cluster centers obtained by CM. (**b**) The key points tracked by the proposed TM.

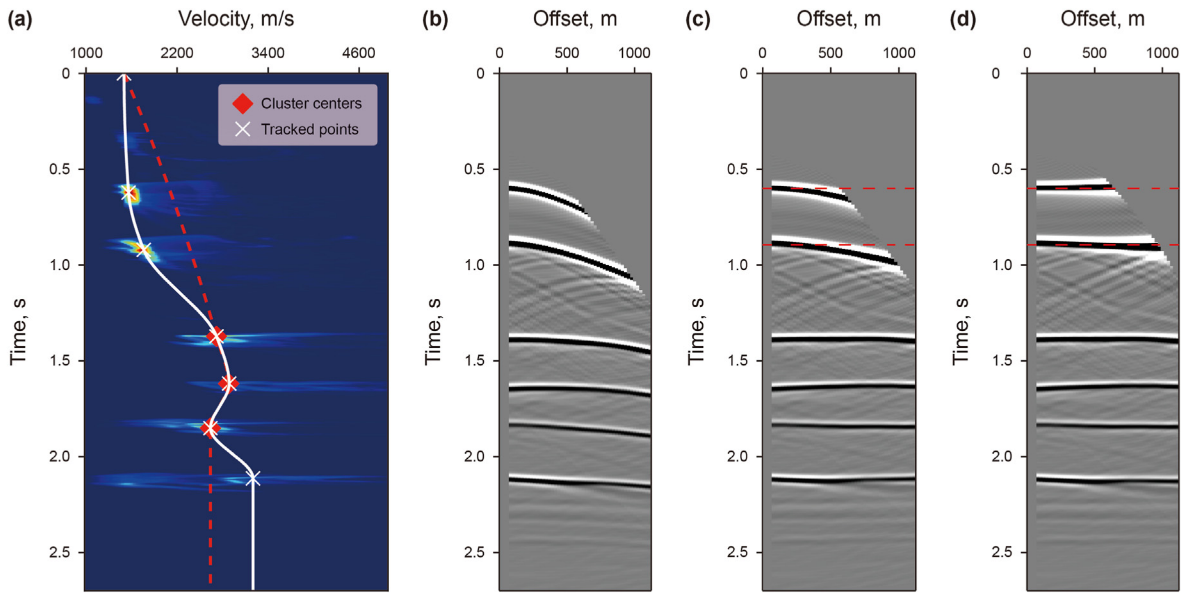

Fig. 7. Comparison of NMO correction results by different methods. (**a**) The red dashed line represents the velocity picked by CM. The solid white line represents the velocity picked up by our TM. (**b**) Single CMP gather from the synthetic seismic data. (**c**) NMO-corrected CMP gather using the velocity picked by CM. (**d**) NMO-corrected CMP gather using the velocity picked by the TM.

obtained by the three methods and the corresponding NMO stack results. The vertical red dotted line on the stack result corresponds to the CMP gather in Fig. 11. We zoom in the area in the cyan box to display in the upper right corner of figure. It can be seen that the seismic events of TM and VM methods are more focused and continuous. At the same time, it can be seen from Table 2 that TM takes less time compared with VM. This shows that our method can achieve excellent performance with higher computational efficiency.

The results of manually picked velocities have also been incorporated and can be seen in Fig. 14. A total of 250 CMP gathers are present in the Gulf of Mexico dataset. Starting from the initial gather, we manually pick every 50th CMP gather, allowing us to produce the final manually picked velocity model through interpolation. The white dashed line shown in Fig. 14(c) marks the location of the manual pick. As evidenced by the regions pointed out by the white arrows in Fig. 14(b) and (d), the stacking results derived using the TM are more focused, clearer, and offer a discernible stratigraphic characterization, surpassing the quality of stacking results based on manually picked velocities.

4. Discussion

4.1. Noisy synthetic example

In this section, to demonstrate the robustness of our method to noise, we conducted experiments on synthesized data. For the forward generated data of section 3.1, we added different levels of Gaussian noise, and compared the velocity picking results and stacking results of TM method and CM method, as shown in Figs. 15

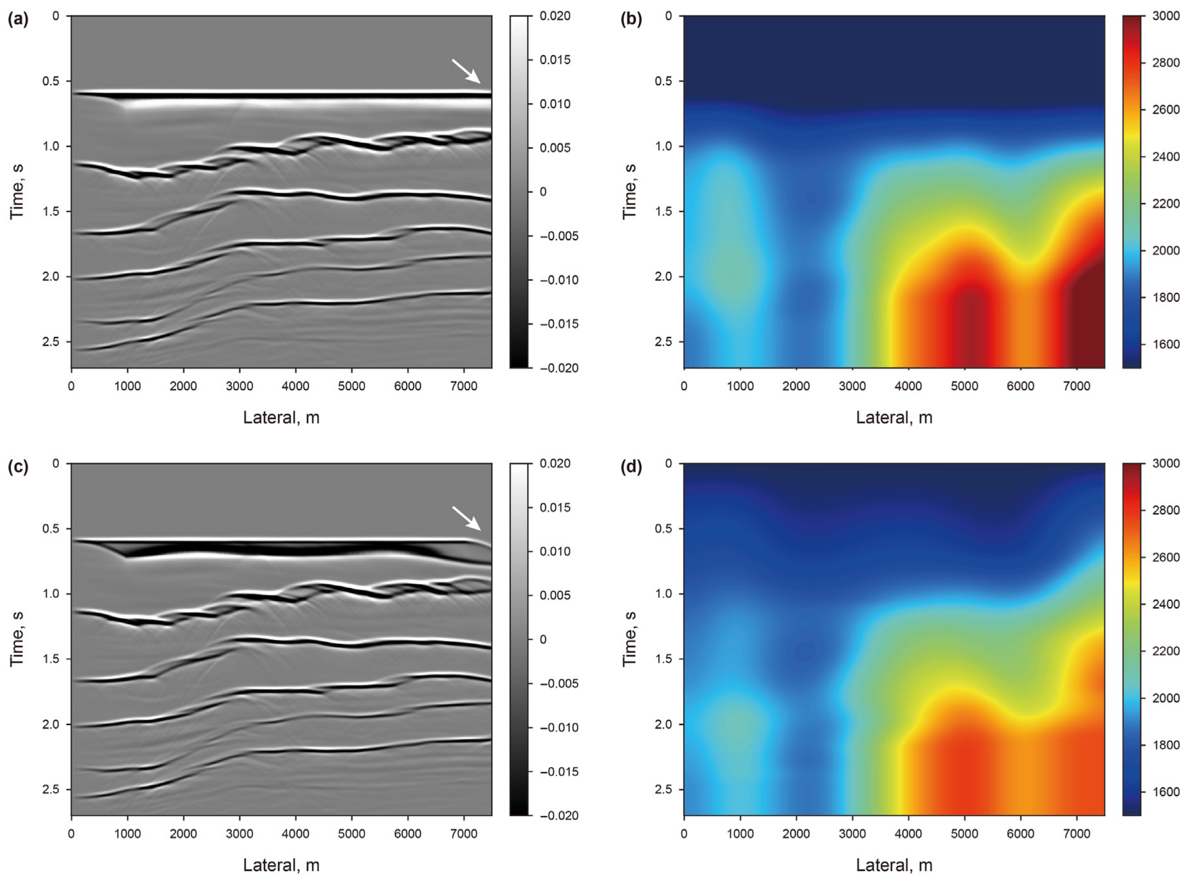

Fig. 8. **(a)** The stacked image using NMO-corrected CMP gathers by our TM. **(b)** Interpolated velocity model by TM. **(c)** The stacked image using NMO-corrected CMP gathers by CM. **(d)** Interpolated velocity model by CM.

and 16. Fig. 15(a)–(c) show the picked velocity by CM with SNRs (signal-to-noise ratio) of 1, 10, and 50 dB, Fig. 15 (d)–(f) show the picked velocity by TM with SNRs of 1, 10, and 50 dB.

With the decrease of SNR, the picked velocity value of CM and TM method in deep areas tends to decrease. However, overall, TM has better robustness to noise. The picked results are more stable as the noise level increases.

Fig. 16(a)–(c) show the cluster center points and corresponding stacking results obtained by the CM with SNR of 1, 10, and 50 dB. Fig. 16(d)–(f) show the tracked points and corresponding stacking results obtained by the TM with SNR of 1, 10, and 50 dB. As the SNR decreases, more outliers appear in the cluster center points obtained by the CM, indicating that the position of its picking velocity is incorrect, and the tracking points obtained by the TM can still be well distributed along the reflection layer in the presence of high noise, indicating the rationality of the picking location. Although the TM still shows some unreasonable picked points when the SNR is 1 dB, such as the position indicated by the yellow arrow, the picked results of TM are still more reasonable than those of CM. As indicated by the white arrows in Fig. 16, the stacking results obtained by TM are more focused and continuous, and the stratigraphic characteristics are more obvious.

From the above experiments, we can see that in the case of low SNR, CM will generate more outliers, resulting in inaccurate picking velocity positions. However, in the case of low SNR, TM can incorporate the information of the near gathers to keep the key points of tracking all the time, so the tracking points can distribute well along the reflection layer, indicating the better robustness of TM.

4.2. How to start tracking

The proposed TM has the ability to select any CMP gather as the starting gather, and initiate tracking in both left and right directions. An inherent risk is the potential for outliers within the initial CMP gather, which can influence the tracking results. However, the ability of the tracking algorithm to synthesize information from nearby traces serves as a counterbalance. Even if the process begins with outliers, the algorithm will not select tracking points that correspond to these outliers. This is primarily because the surrounding gathers do not exhibit velocity spectrum features similar to that of outliers. By implementing a post-processing operation to eliminate the tracking set with too few tracked points, the impact of outliers on the tracking results can be effectively minimized.

As evident in Fig. 17(a), tracking commences from CDP 40, denoted by the red dashed line. Fig. 17(b) illustrates the resultant tracking, emphasizing the comparatively smaller number of elements in the tracking set associated with outliers. Consequently, by discarding the set of tracking points with elements below a specific threshold, an enhanced tracking result is achieved, as shown in Fig. 17(c). It should be noted, however, that despite these improvements, the results may still be sub-optimal compared to initiating the tracking from a CDP free of outliers.

Another technique is to observe the clustering results after obtaining the results of CM and manually select CMP gathers without outliers to start tracking. It is a relatively simple technique, so we recommend using it in practical applications.

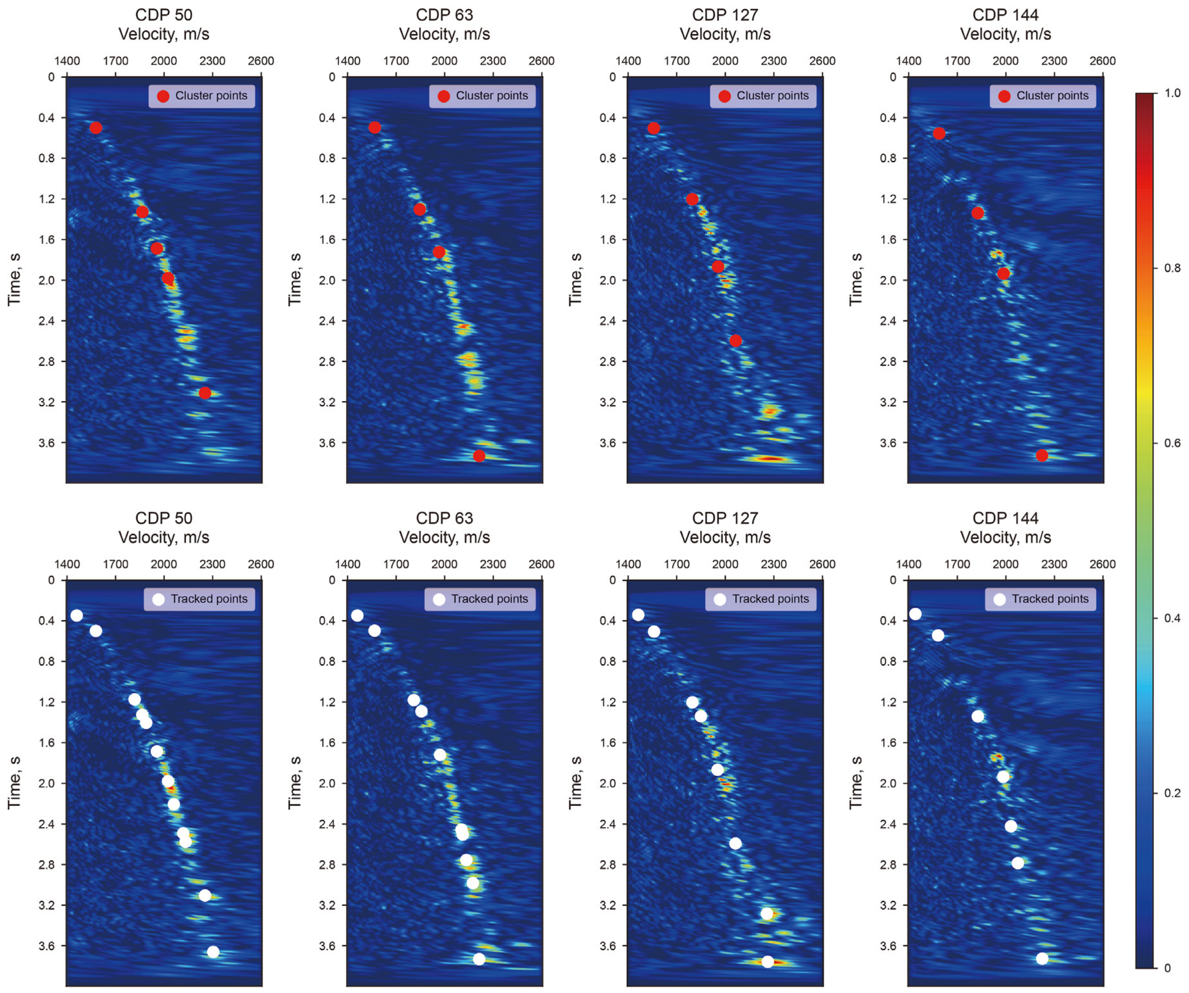

Fig. 9. Tracked points and cluster center points on the velocity spectrum of different CDPs. The first row illustrates the cluster points obtained by CM, and the second row illustrates the tracked points obtained by TM.

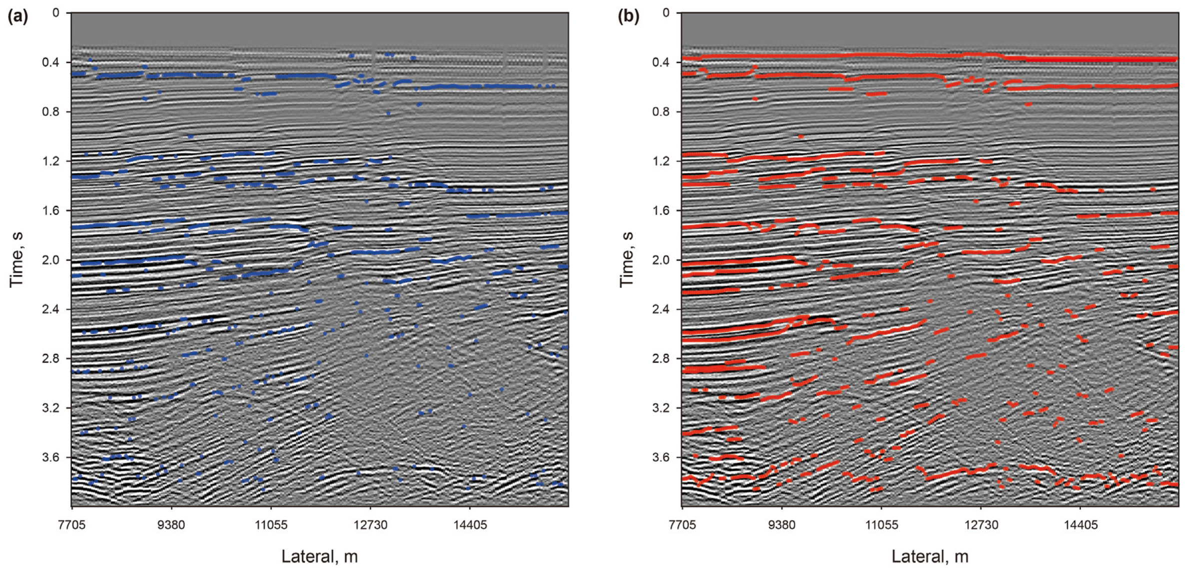

Fig. 10. **(a)** The cluster centers obtained by CM (marked by blue points). **(b)** The key points tracked by TM (marked by red points).

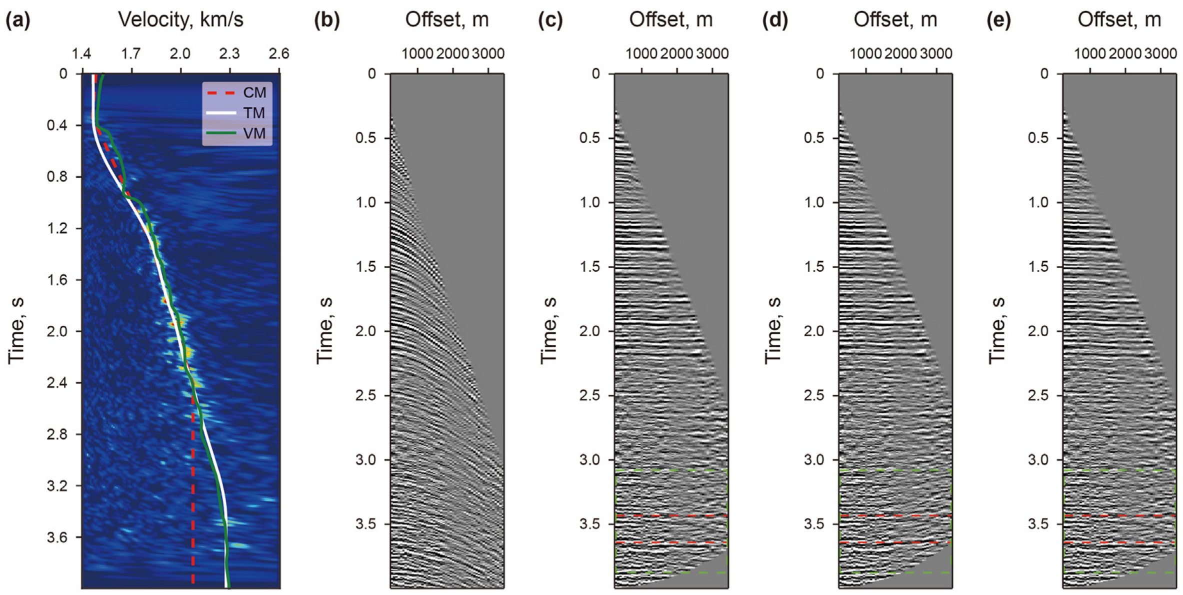

Fig. 11. NMO-corrected CMP gathers using different velocity. **(a)** Velocity picked by different methods. **(b)** The single CMP gather located at 12.9 km. **(c)** NMO-corrected CMP gather using the velocity picked by TM. **(d)** NMO-corrected CMP gather using the velocity picked by CM. **(e)** NMO-corrected CMP gather using the velocity picked by VM.

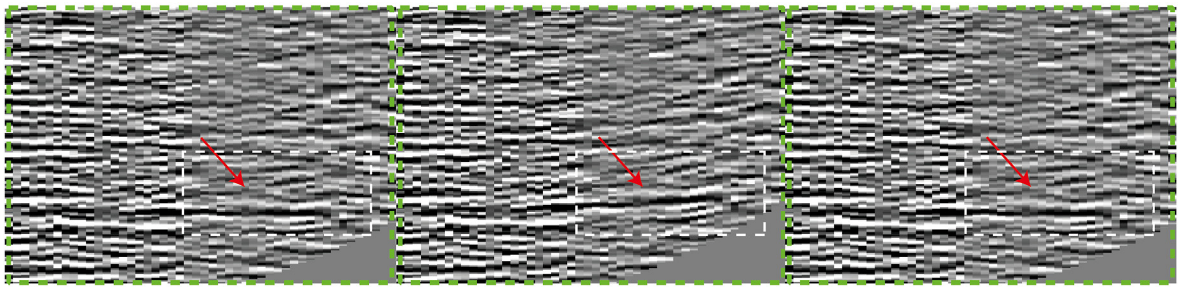

Fig. 12. The enlarged content of the green rectangular box in Fig. 11. From left to right, the results of TM, CM, and VM are sequentially displayed. The events of TM and VM at the positions indicated by the red arrows are flatter than those of CM. The areas in the white dashed box are the parts with significant differences.

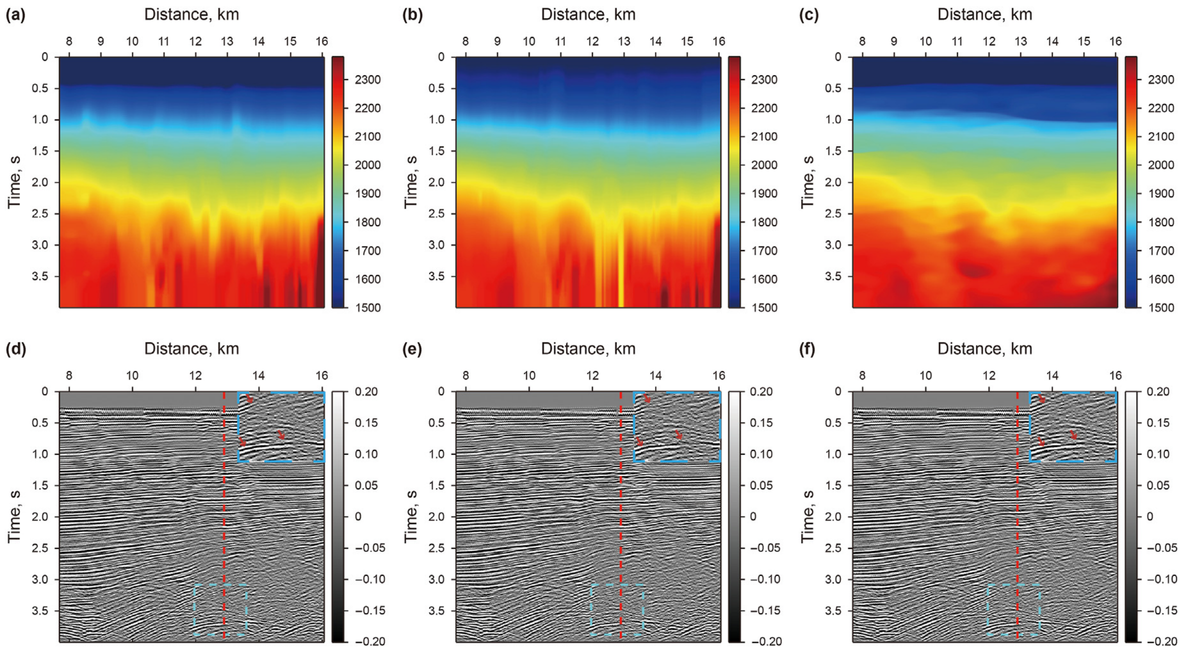

Fig. 13. **(a)** Velocity model determined by TM. **(b)** Velocity model determined by CM. **(c)** Velocity model determined by VM. **(d)** NMO stack of the data using the velocity model from **(a)**. **(e)** NMO stack of the data using the velocity model from **(b)**. **(f)** NMO stack of the data using the velocity model from **(c)**.

Table 2
Time cost of different methods on Gulf of Mexico data.

Method	Time cost
CM	0.94 s
TM	0.94 s + 4.6 s
VM	10.4 min

5. Conclusions

In this paper, we present an innovative density clustering-assisted optimal key points tracking method, specifically designed for automatic velocity picking. The novelty lies in extending the concept of object tracking, predominantly applied in computer vision, to the realm of geophysics. To address the issue of object tracking on the velocity spectrum, we propose a unique combination of zero normalized cross-correlation (ZNCC) and the Hungarian algorithms. Further enhancing this method, Kalman filtering technology, a technique rooted in control theory, is introduced to optimize the tracking process. Hence, our proposed methodology represents a cutting-edge intersection of research across computer vision, automatic control, and geophysics.

The center points determined by the CM represent the highest likelihood velocity of the primary subsurface structure. Nevertheless, this technique does not consider the correlation between adjacent gathers and may yield inaccurate results, particularly when SNR is low. While existing velocity picking methods VM from semblance-like volumes (Decker and Fomel, 2022) incorporate spatially adjacent information, they are computation-intensive. In contrast, our tracking-based approach seamlessly integrates information from adjacent gathers, offering both high computational efficiency and robustness to noise.

The CM method obtains inconsistent cluster center points from adjacent velocity spectra, which may be induced by noise or algorithm parameter settings. At times, some cluster center points may not be detected, leading to errors in the picking results. Our TM, however, integrates information from nearby gathers to consistently maintain the key tracking points. This results in robust performance even in areas with a low SNR. Experimental evidence, both from synthetic and real data, substantiates the efficacy of our proposed method. However, the proposed method is not without its limitations. For instance, the tracking process may be influenced by outliers in the initial CMP gather. As we have suggested, this can be mitigated by observing the clustering results and manually selecting CMP gather without outliers to initiate tracking. However, a possible more robust approach is to select multiple CMP gathers to start the tracking algorithm separately and then merge the

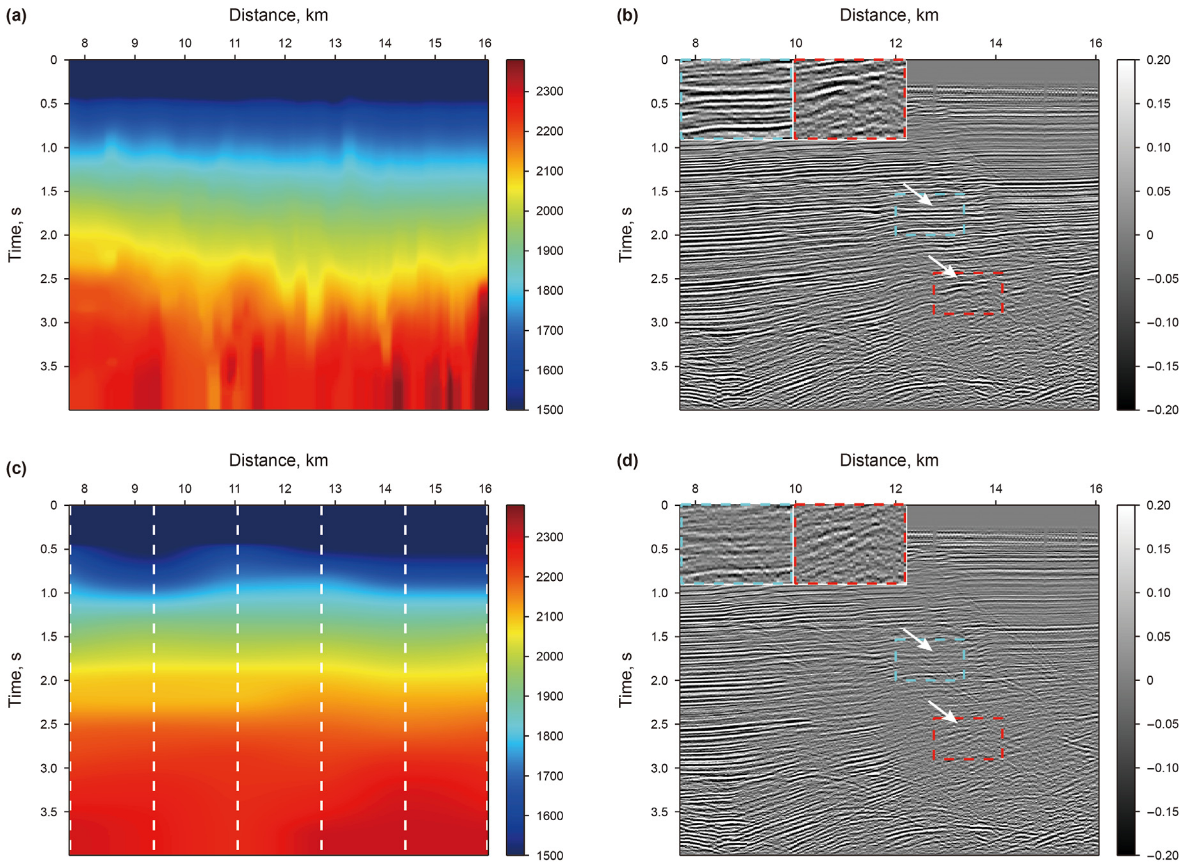

Fig. 14. Comparison of the velocity and its stacking results obtained by TM with the manually picked velocity and its stacking results. The areas in the cyan and blue boxes are magnified in the upper left corner of stacked profile image. **(a)** The velocity obtained by TM; **(b)** NMO stack of the data using velocity determined by TM; **(c)** the velocity obtained by manual picking, with six white dashed lines indicating the position of manual picking; **(d)** NMO stack of the data using the manually picked velocity.

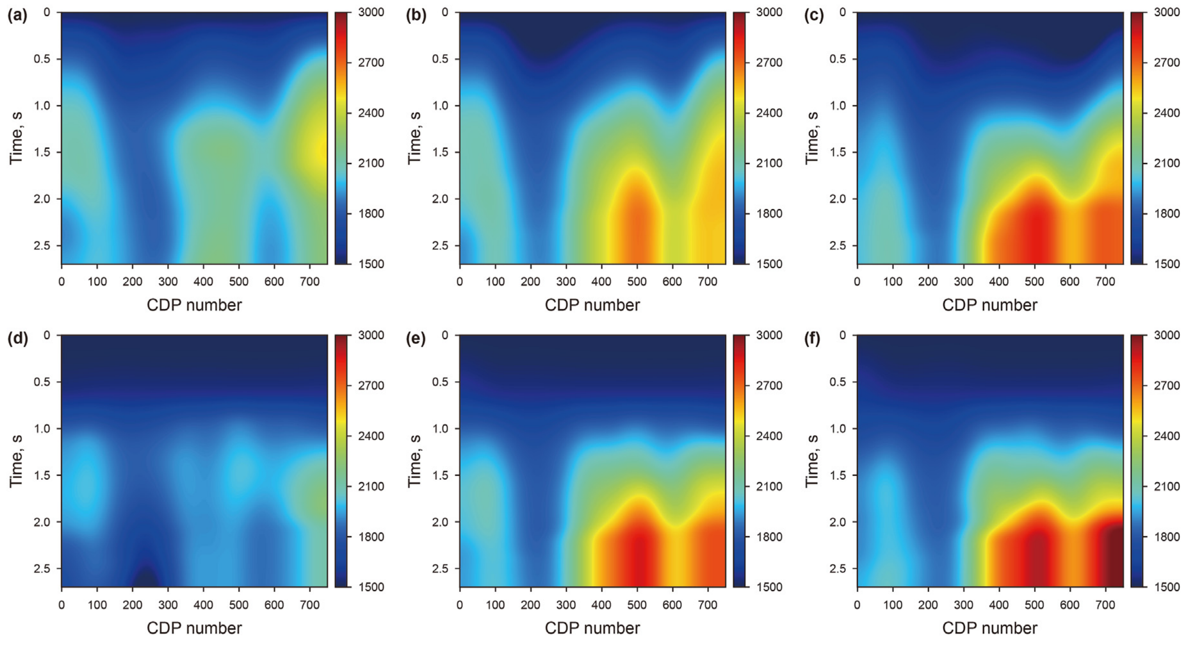

Fig. 15. (**a**) Velocity model obtained by CM at SNR 1 dB. (**b**) Velocity model obtained by CM at SNR 10 dB. (**c**) Velocity model obtained by CM at SNR 50 dB. (**d**) Velocity model obtained by TM at SNR 1 dB. (**e**) Velocity model obtained by TM at SNR 10 dB. (**f**) Velocity model obtained by TM at SNR 50 dB.

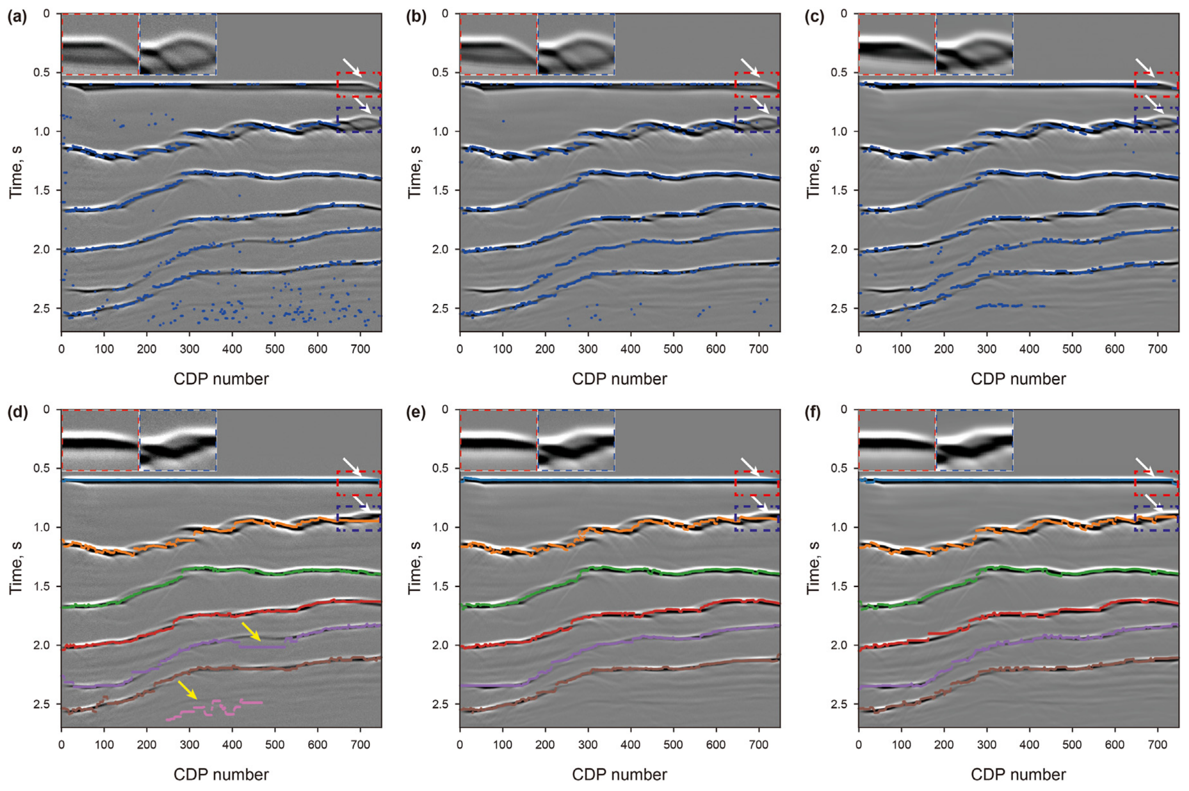

Fig. 16. Comparison of the stacked profiles obtained by CM and TM methods under different SNRs. The areas in the red and blue boxes are magnified in the upper left corner of each image. (**a**) The cluster center points and corresponding stacking results obtained by CM at SNR 1 dB. (**b**) The cluster center points and corresponding stacking results obtained by CM at SNR 10 dB. (**c**) The cluster center points and corresponding stacking results obtained by CM at SNR 50 dB. (**d**) The tracked points and corresponding stacking results obtained by TM at SNR 1 dB. (**e**) The tracked points and corresponding stacking results obtained by TM at SNR 10 dB. (**f**) The tracked points and corresponding stacking results obtained by TM at SNR 50 dB.

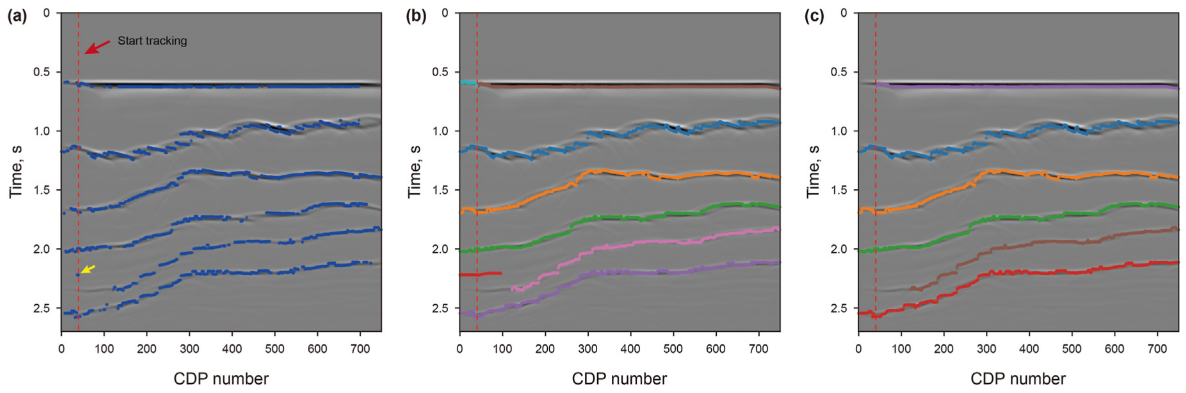

Fig. 17. (**a**) The cluster centers obtained by accelerated density clustering. The yellow arrow denotes the location of the outlier point, while the red dashed line indicates CDP 40. (**b**) Tracked key points. (**c**) Post-processed tracked key points.

tracking results for post-processing. This can effectively reduce the impact of outliers and make our method more reliable. How to design such post-processing algorithms is our future research direction.

Acknowledgements

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFA0702501, and in part by NSFC under Grant 41974126, 41674116 and 42004101.

References

Alkhalifah, T., Joe Dellinger, B.P., Michael, B.H.P., et al., 2006. https://reproducibility.org/wiki/Main_Page.

Almarzoug, A.M., Ahmed, F.Y., 2012. Automatic Seismic Velocity Picking. SEG Technical Program Expanded Abstracts. https://doi.org/10.1190/segam2012-0294.1.

Araya-Polo, M., Jennings, J., Adler, A., et al., 2018. Deep-learning tomography. Lead. Edge 37 (1), 58–66.

Battistelli, G., Chisci, L., Selvi, D., 2018. A distributed Kalman filter with event-triggered communication and guaranteed stability. Automatica 93, 75–82. https://doi.org/10.1016/j.automatica.2018.03.005.

Biswas, R., Vassiliou, A., Stromberg, R., et al., 2018. Stacking velocity estimation using recurrent neural network. In: SEG International Exposition and Annual Meeting, SEG-2018-2997208. https://doi.org/10.1190/segam2018-2997208.1.

Claerbout, J.F., Black, J.L., 2001. Basic Earth Imaging, version 2.4.

Cooper, S., Durrant-Whyte, H., 1994. A Kalman filter model for GPS navigation of land vehicles. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94) 1, 157–163. https://doi.org/10.1109/IROS.1994.407396.

Decker, L., Fomel, S., 2022. A variational approach for picking optimal surfaces from semblance-like panels. Geophysics 87 (3), U93–U108. https://doi.org/10.1190/geo2021-0336.1.

Fomel, S., 2007. Velocity-independent time-domain seismic imaging using local event slopes. Geophysics 72 (3), S139–S147. https://doi.org/10.1190/1.2714047.

Fomel, S., 2009. Velocity analysis using AB semblance. Geophys. Prospect. 57 (3), 311–321. https://doi.org/10.1111/j.1365-2478.2008.00741.x.

Fritsch, F.N., Butland, J., 1984. A method for constructing local monotone piecewise cubic interpolants. SIAM J. Sci. Stat. Comput. 5 (2), 300–304. https://doi.org/10.1137/0905021.

Graves, A., Mohamed, A.R., Hinton, G., 2013. Speech recognition with deep recurrent neural networks. In: IEEE International Conference on Acoustics, Speech and Signal Processing, 6645–6649. https://doi.org/10.1109/ICASSP.2013.6638947.

Humpherys, J., Redd, P., West, J., 2012. A fresh look at the Kalman filter. SIAM Rev. 54 (4), 801–823. https://doi.org/10.1137/100799666.

Kalman, R.E., 1960. A new approach to linear filtering and prediction problems. J. Basic Eng. 82 (1), 35–45. https://doi.org/10.1115/1.3662552.

Kuhn, H.W., 1955. The Hungarian method for the assignment problem. Nav. Res. Logist. Q. 2 (1–2), 83–97. https://doi.org/10.1002/nav.3800020109.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444. https://doi.org/10.1038/nature14539.

Lee, J.H., Ricker, N.L., 1994. Extended Kalman filter based nonlinear model predictive control. Ind. Eng. Chem. Res. 33 (6), 1530–1541. https://doi.org/10.1021/ie00030a013.

Mulder, W.A., Ten Kroode, A.P.E., 2002. Automatic velocity analysis by differential semblance optimization. Geophysics 67 (4), 1184–1191. https://doi.org/10.1190/1.1500380.

Penizzotto, F., Slawinski, E., Mut, V., 2015. Laser radar based autonomous mobile robot guidance system for olive groves navigation. IEEE Latin America Transactions 13 (5), 1303–1312. https://doi.org/10.1109/TLA.2015.7111983.

Ramshaw, L., Tarjan, R.E., 2012. On Minimum-Cost Assignments in Unbalanced Bipartite Graphs. HP Labs, Palo Alto, CA, USA. Tech. Rep. HPL-2012-40R1, 20.

Redmon, J., Divvala, S., Girshick, R., et al., 2016. You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779–788. https://doi.org/10.1109/CVPR.2016.91.

Rodriguez, A., Laio, A., 2014. Clustering by fast search and find of density peaks. Science 344 (6191), 1492–1496. https://doi.org/10.1126/science.1242072.

Stefano, L.D., Mattoccia, S., Tombari, F., 2005. ZNCC-based template matching using bounded partial correlation. Pattern Recogn. Lett. 26 (14), 2129–2134. https://doi.org/10.1016/j.patrec.2005.03.022.

Symes, W.W., 1998. High frequency asymptotics, differential semblance, and velocity estimation. In: SEG Technical Program Expanded Abstracts, pp. 1616–1619. https://doi.org/10.1190/1.1820229.

Symes, W.W., Carazzone, J.J., 1991. Velocity inversion by differential semblance optimization. Geophysics 56 (5), 654–663. https://doi.org/10.1190/1.1443082.

Taner, M.T., Koehler, F., 1969. Velocity spectra—digital computer derivation applications of velocity functions. Geophysics 34 (6), 859–881. https://doi.org/10.1190/1.1440058.

Toldi, J.L., 1989. Velocity analysis without picking. Geophysics 54 (2), 191–199. https://doi.org/10.1190/1.1442643.

Virtanen, P., Gommers, R., Oliphant, T.E., et al., 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods 17 (3), 261–272. https://doi.org/10.1038/s41592-019-0686-2.

Wang, W., McMechan, G.A., Ma, J., et al., 2021. Automatic velocity picking from semblances with a new deep-learning regression strategy: comparison with a classification approach. Geophysics 86 (2), U1–U13. https://doi.org/10.1190/geo2020-0423.1.

Waheed, U., Al-Zahrani, S., Hanafy, S.M., 2019. Machine learning algorithms for automatic velocity picking: K-means vs. DBSCAN. SEG International Exposition and Annual Meeting, SEG-2019-3215809. https://doi.org/10.1190/segam2019-3215809.1.

Wang, X., Gao, Y., Chen, C., et al., 2022. Intelligent velocity picking and uncertainty analysis based on the Gaussian mixture model. Acta Geophys. 70 (6), 2659–2673. https://doi.org/10.1007/s11600-022-00859-8.

Wrona, T., Pan, I., Bell, R.E., et al., 2021. 3D seismic interpretation with deep learning: a brief introduction. Lead. Edge 40 (7), 524–532. https://doi.org/10.1190/tle40070524.1.

Zhang, B., Zhao, T., Qi, J., et al., 2014. Horizon-based semiautomated nonhyperbolic velocity analysis. Geophysics 79 (6), U15–U23, 0.1190/geo2014-0112.1.

Zhang, H., Zhu, P., Gu, Y., et al., 2019. Automatic Velocity Picking Based on Deep Learning. In: SEG Technical Program Expanded Abstracts, pp. 2604–2608.. https://doi.org/10.1190/segam2019-3215633.1.

Zhang, P., Lu, W., 2016. Automatic time-domain velocity estimation based on an accelerated clustering method. Geophysics 81 (4), U13–U23. https://doi.org/10.1190/geo2015-0313.1.

Zhang, P., Lu, W., Zhang, Y., 2015. Velocity analysis with local event slopes related probability density function. J. Appl. Geophys. 123, 177–187. https://doi.org/10.1016/j.jappgeo.2015.10.010.

Zhu, W., Mousavi, S.M., Beroza, G.C., 2019. Seismic signal denoising and decomposition using deep neural networks. IEEE Trans. Geosci. Rem. Sens. 57 (11), 9476–9488. https://doi.org/10.1190/segam2019-3215809.1.