BMC Bioinformatics

RESEARCH

Open Access

FindCSV: a long-read based method for detecting complex structural variations

Yan Zheng[1*] and Xuequn Shang[1*]

*Correspondence:
yan.zheng@mail.nwpu.edu.cn;
shang@nwpu.edu.cn

[1] School of Computer Science, Northwestern Polytechnical University, West Youyi Road 127, Xi'an 710072, China

Abstract

Background: Structural variations play a significant role in genetic diseases and evolutionary mechanisms. Extensive research has been conducted over the past decade to detect simple structural variations, leading to the development of well-established detection methods. However, recent studies have highlighted the potentially greater impact of complex structural variations on individuals compared to simple structural variations. Despite this, the field still lacks precise detection methods specifically designed for complex structural variations. Therefore, the development of a highly efficient and accurate detection method is of utmost importance.

Result: In response to this need, we propose a novel method called FindCSV, which leverages deep learning techniques and consensus sequences to enhance the detection of SVs using long-read sequencing data. Compared to current methods, FindCSV performs better in detecting complex and simple structural variations.

Conclusions: FindCSV is a new method to detect complex and simple structural variations with reasonable accuracy in real and simulated data. The source code for the program is available at https://github.com/nwpuzhengyan/FindCSV.

Keywords: Complex structural variations, Long-read sequencing data, Deep learning, Consensus sequences

Background

Structural variations (SVs) are large-scale mutations in the genome involving over 50 nucleotides [1]. They encompass different types of mutations, such as deletions (DEL), insertions (INS), duplications (DUP), inversions (INV), and translocations (TRA), and are characterized by significant changes in the genome. Although SVs are relatively rare compared to single nucleotide polymorphisms (SNPs) and smaller insertions or deletions (Indels), recent studies have highlighted their substantial contribution to heritable genetic variations [2]. Furthermore, SVs have been implicated in various genetic disorders, including cancer, autism spectrum disorders, and Alzheimer's disease [3–6]. They also significantly affect species evolution, gene expression, and phenotypic diversity [7–11].

In addition to simple SVs such as INSs and DELs, recent research has revealed the existence of complex SVs (CSVs) [12, 13]. Detecting CSVs poses a greater challenge compared to simple SVs. Simple SVs involve one or two breakpoints in the genome and have relatively straightforward structures. In contrast, CSVs typically involve multiple breakpoints in the genome, resulting in more intricate structures. CSVs can be considered as nested SVs composed of combinations of simple SVs, leading to more complex alignment outcomes in high-throughput sequencing data. Although CSVs occur less frequently than simple SVs in the genome, their genomic structures are more complex, and therefore, CSVs can have a greater impact on individuals than simple SVs [14].

In recent years, there has been a significant increase in the number of detection methods targeted at simple SVs. Initially, studies focused on SV detection using short reads (100–150bp). Several methods were developed, including Delly [15], Lumpy [16], PindelcitePindel, Manta [17], Gustaf [18], and SurVIndel [19]. More recently, several SV callers based on long-read data have emerged, such as DeBreak [20], cuteSV [21], Sniffles [22], Sniffles2 [23], NanoSV [24], picky [25], SVIM [26], PBHoney [27], SVision [28] and SVcnn [29].

However, there is currently a lack of precise detection methods to detect CSVs. Experimental results have revealed two primary challenges in CSV detection: 1. The absence of a precise definition for CSVs; 2. The diverse nature of CSVs presents difficulties for detection methods in accurately identifying and reporting the correct breakpoints.

Regarding the first challenge, detecting CSVs involves identifying nested SVs that can be considered as combinations of simple SVs. However, determining the threshold distance at which two adjacent SVs can be merged into a CSV remains a challenge. Recent researches lack a clear and definitive definition for CSVs, which hinders their accurate detection. Addressing this question is crucial for improving the precision of CSV detection. Regarding the second challenge, CSVs can exhibit a theoretically infinite number of structural types as they are formed by combining and nesting simple SVs. Consequently, exploring the internal structure of CSVs becomes a complex task for detection methods.

Therefore, there is an urgent imperative to develop better detection methods specifically tailored to target CSVs and overcome these challenges. In response to these challenges, we developed a novel detection method called FindCSV, specifically designed to detect CSVs in third-generation sequencing data. To evaluate the performance of various detection methods, we have obtained three real datasets and conducted extensive experiments using these datasets. The download links for the datasets can be found in the Supplementary file Table S1. The experimental results unequivocally demonstrate that FindCSV has excellent performance compared to existing detection methods, not only in detecting CSVs but also in detecting simple SVs. This comprehensive evaluation showcases the effectiveness and superiority of FindCSV as a better tool for detection of both CSVs and simple SVs.

Results

Details of research data

Second-generation sequencing data and third-generation sequencing data are both considered high-throughput data. However, third-generation sequencing data offers several advantages over second-generation sequencing data. One notable advantage is the longer read lengths provided by third-generation sequencing, which inherently improves sequence alignment. Leveraging the alignment results from third-generation sequencing data allows for simplified detection of SVs and helps mitigate potential matching errors caused by excessive parameters [30]. As a result, SV detection performed on third-generation sequencing data often yields superior results compared to second-generation sequencing data.

Recent research shows that the number of SVs in human individuals should be between 22,000 and 25,000 [31]. According to the survey results, it was found that short-read-based detection methods can only detect approximately 40–60% of true positive SVs, whereas long-read-based detection methods can detect 80–90% of true positive SVs. Furthermore, long-read sequencing technologies enhance the ability to identify SVs, particularly in the detection of CSVs and those located within repetitive regions [32, 33]. Therefore, we selected third-generation sequencing data as the research data.

The two most popular long-read sequencing technologies are PacBio [34] and Oxford Nanopore (ONT) [35]. Compared to PacBio data, ONT data exhibits higher error rates in its reads. Consequently, detecting SVs in ONT data presents more significant challenges. For this reason, we have downloaded both PacBio and ONT sequencing data, with a focus on evaluating the detection performance specifically on ONT data. By prioritizing the evaluation on ONT data, we aim to assess the effectiveness and robustness of our detection method in the context of challenging long-read sequences with higher error rates.

Details of current detection methods

Due to the abundance of long-read-based detection methods, it was impractical to individually test each method. Therefore, we carefully selected five detection methods from a range of long-read SV callers: Sniffles [22] (version 1.0.11), cuteSV [21] (version 2.0.2), DeBreak [20] (version 1.3), SVcnn [29] (version 1), and SVision [28] (version 1.4). These five methods were chosen based on their recent release and demonstrated accuracy in SV detection. Sniffles is the current most popular SV detection method. DeBreak and cuteSV are efficient methods that have been shown to have very high recall and precision. SVcnn is the method we previously used to detect simple SVs. SVision is the method specifically used to detect CSVs on third-generation sequencing data. The reason why Sniffles2 was not selected is that it cannot detect CSVs.

Benchmark and selected samples

The HG002 SVs dataset provided by GIAB (Genome in a Bottle) is widely recognized and extensively used as a high-quality benchmark for evaluating SV detection methods. This benchmark is derived from the hg19 reference genome and is considered a reliable standard for SV detection evaluation [36]. The dataset comprises 12,742 SVs of the PASS type, including 5463 DELs and 7279 INSs. However, recent research suggests that the number of SVs in human individuals should range from 22,000 to 25,000 [31]. This means the benchmark dataset likely misses thousands of true positive SVs. When using this benchmark for evaluating SV detection methods, it is important to consider this limitation, as the dataset may not fully capture the complete spectrum of SVs present in the human population.

To address this issue, we adopted a novel approach to generate a new benchmark by leveraging the results obtained from different SV detection methods and genome assemblies. HiFi reads are generated using circular consensus sequencing mode on PacBio long-read systems, which exhibit an impressively low error rate of approximately 0.1% [37]. This high-quality data can significantly enhance the reliability and confidence of the benchmark. The steps involved in creating this new benchmark, including the utilization of HiFi data and genome assemblies, are thoroughly described in Sect. 2 of the supplementary file. These steps outline the process by which we use the outputs of multiple SV detection methods and genome assemblies to construct a more comprehensive and accurate benchmark for SV detection evaluation.

To generate new benchmarks, it is necessary to have genome assemblies for the samples. Hence, we carefully selected three samples: HG002 [38], CHM13 [39], and HG00733 [40], for which genome assemblies were available. HG002 and HG00733 are the most popular datasets with accurate and complete genome assemblies. Compared with HG002 and HG00733, CHM13 has near-complete homozygosity. Therefore, we can regard CHM13 as a homozygous human genome and ignore the influence of multi-allelic SVs. The download links for these genome assemblies can be found in the supplementary file Table S2. By utilizing these genome assemblies, we were able to derive SV datasets specifically for these three samples. These datasets serve as crucial resources for evaluating the performance of SV detection methods.

Detection evaluation for simple SVs

This section aims to evaluate the detection performance of different methods for simple SVs. We obtained the ONT long-read data for three individuals: HG002 [38], CHM13 [39], and HG00733 [40]. These long-read datasets were aligned to the hg38 reference genome, resulting in BAM files. The details of the long-read data and reference links can be found in Tables S1 and S2 of the supplementary file. The Linux commands used for the alignment process are described in Sect. 1 of the supplementary file.

We applied selected SV detection methods to the generated BAM files. Each method independently analyzed the long-read data and produced SV detection results. To evaluate the performance of the different SV detection methods, we generated new benchmarks for different samples, as described in detail in Sect. 2 of the supplementary file. The benchmark contains information on DELs and INSs. Due to the specific characteristics of DUPs and INVs, we focused on evaluating DELs and INSs in this manuscript. Finally, we compared the SV detection results from each method with the benchmark dataset. Section 5 of the supplementary file provides the method used to verify whether an SV output from a detection method is present in the benchmark dataset.

Figure 1 shows the recall, precision, and F1-score of these methods across the CHM13, HG002, and HG00733 datasets. The results indicate that the detection performance of FindCSV has not significantly improved compared to SVcnn. This is because some simple SVs are mistakenly identified as CSVs, leading to changes in the length of the SVs. Consequently, during the SV comparison process, certain SVs that have been misjudged as CSVs fail to match the simple SVs in the benchmark. Nevertheless, besides SVcnn, the detection performance of FindCSV surpasses that of other existing methods across all datasets. Therefore, it can be concluded that FindCSV exhibits the second-best performance in detecting simple SVs, trailing only the SVcnn.

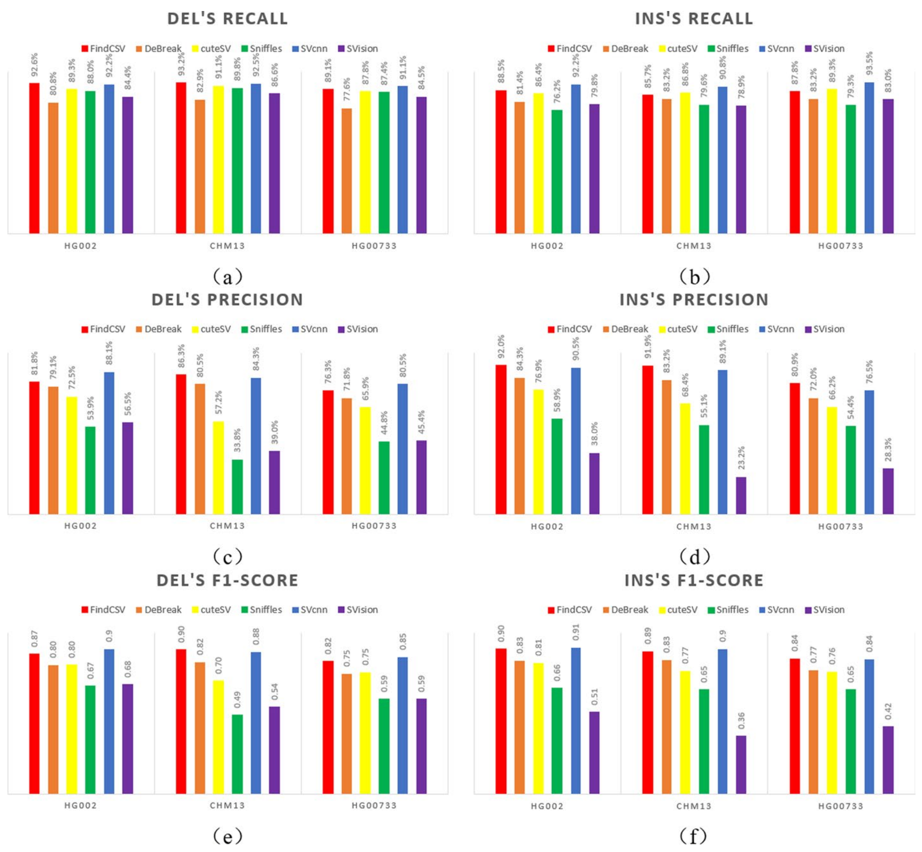

Fig. 1 The figure illustrates the recall, precision, and F1-score of FindCSV, DeBreak, cuteSV, Sniffles, SVcnn, and SVision across the CHM13, HG002, and HG00733 datasets

Detection evaluation for CSVs

This section aims to evaluate the detection performance of different methods for identifying CSVs. However, the current research lacks high-quality and reliable benchmark datasets specifically designed for evaluating CSV detection methods. To address this, we utilize assembled genomes to generate a more comprehensive and accurate benchmark for CSVs, as detailed in Sect. 2 of the supplementary file. Using this method, our experiments identified 366, 268, and 223 CSVs in HG002, CHM13, and HG00733, respectively.

Most existing SV detection methods are not designed to detect CSVs, and only Sniffles and SVision can output CSVs. Therefore, we applied FindCSV, Sniffles, and SVision to the ONT data and HiFi data of three samples, respectively. Using the CSV comparison method described in Sect. 5 of the supplementary file, we counted the number of CSVs in the benchmark that were detected by each method. Additionally, we calculated the recall, precision, and F1-score of each method to assess their performance in detecting CSVs. The obtained values are shown in Fig. 2.

From Fig. 2, it is evident that the performance of all current detection methods is not particularly superior in detecting CSVs. Among these methods, FindCSV demonstrates better performance in terms of recall, precision, and F1-score. FindCSV achieves high recall values, with the lowest recall for CSVs being over 50%. However,

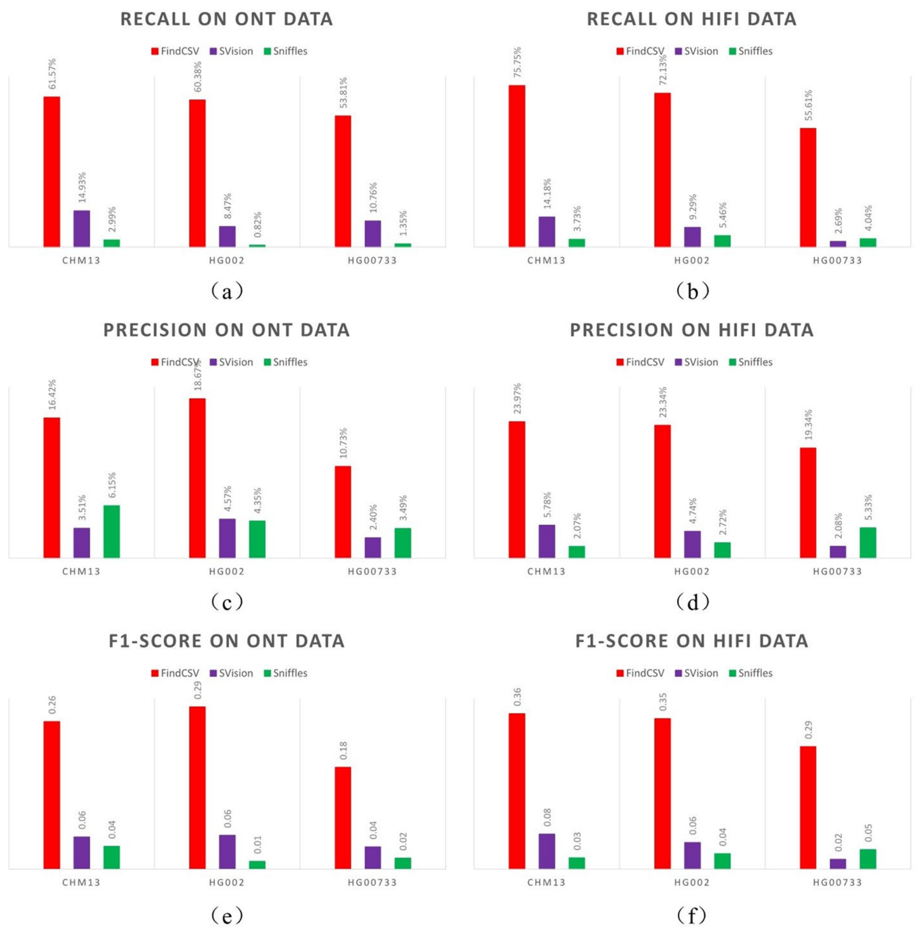

Fig. 2 The figure illustrates the F1-score of FindCSV, Sniffles, and SVision about CSVs on the ONT and HiFi datasets

its precision values are relatively low, with the highest precision being approximately 24%. Furthermore, FindCSV performs significantly better on HiFi data compared to ONT data. This disparity can be attributed to the higher noise levels present in ONT data, which may lead to the formation of false positive SVs. The presence of these false positive SVs can cause FindCSV to mistakenly detect simple SVs as CSVs. Consequently, the number of CSVs detected exceeds the actual number, leading to a decrease in detection accuracy.

In comparison, the other two detection methods, Sniffles and SVision, exhibit much lower performance in detecting CSVs. The highest recall achieved by these methods is around 15%, and the highest precision is less than 6%. This discrepancy may arise from differences in the definition of CSVs. FindCSV focuses on identifying SVs with breakpoint distances less than 1000 bp located on the same read as CSVs. Other detection methods may not consider SVs with distances less than 1000 bp as CSVs or incorrectly classify multi-allelic SVs from both parents as CSVs. These factors contribute to the poorer performance of Sniffles and SVision in detecting CSVs. For specific examples, please refer to Sect. 8 of the supplementary file.

Detection evaluation on simulated datasets

In real datasets, existing detection methods have far lower performance in detecting CSVs than FindCSV, possibly due to different methods having different definitions of CSVs. If there is no unified standard, the evaluation results obtained will be meaningless.

To obtain more meaningful evaluation results, this section generates simulated CSVs on the CHM13 dataset to evaluate different methods. The reason for choosing CHM13 is that it can be considered a haploid genome without considering the impact of multi-allelic SVs. The process of generating simulated SVs involves modifying the assembly genome of CHM13 and subsequently aligning the sequencing data of CHM13 to the modified genome, thereby creating a set of simulated SVs. Through this method, we generated 4600 CSVs of different types, composed of two different types of SVs with consistent breakpoint positions.

Next, the FindCSV, Sniffles, and SVision were applied to the simulated datasets, respectively. Finally, the recall, precision, and F1-score of each method were calculated to comprehensively assess their performance in detecting the simulated CSVs. The obtained values are shown in Fig. 3.

From Fig. 3, it can be observed that on simulated data, FindCSV can detect nearly 68% of CSVs, SVision can detect about 44% of CSVs, and Sniffles can only detect 50 CSVs, accounting for only 0.8% of the actual number, which can be almost negligible. Furthermore, FindCSV also demonstrates the highest precision of 92%. In contrast, SVision and Sniffles show lower precision of 89% and 74%, respectively. Despite the relatively small differences in precision among the methods, the significant disparities in their recall values ultimately lead to notable differences in their F1-scores. In conclusion, FindCSV's performance in detecting CSVs far surpasses the existing methods.

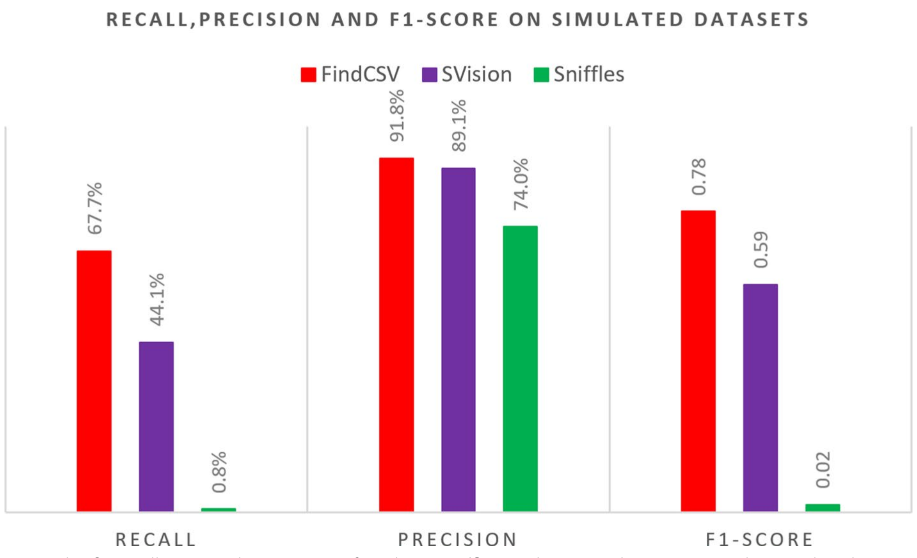

Fig. 3 The figure illustrates the F1-score of FindCSV, Sniffles, and SVision about CSVs on the simulated datasets

The running time and memory

In this section, the aforementioned methods were executed on the same Linux server, and their running time and maximum memory usage to process the same data were recorded. The detailed results are presented in Table 1.

Analyzing Table 1, it is evident that FindCSV exhibits relatively low memory usage. However, its detection efficiency is not superior compared to the other methods. In terms of the time required for SVs detection, FindCSV falls in the middle range among these detection methods. Notably, both FindCSV and Sniffles are Python-based programs, but FindCSV tends to run slower primarily due to the inclusion of multiple additional steps such as remapping and converting images, deep learning filtering, and constructing consensus sequences.

Conclusions

We have developed a novel CSVs detection method called FindCSV. Currently, the detection of CSVs faces two main challenges:

1. Lack of a precise definition for CSVs: To address this, we define SVs with breakpoint intervals less than 1000 bp located on the same parental genome as CSVs. This provides a clear and specific definition for identifying CSVs.

2. Diversity of CSVs: The diverse nature of CSVs poses difficulties for detection methods to accurately identify and report the correct breakpoints. To overcome this challenge, FindCSV employs a multi-step approach. It first distinguishes and clusters different reads originating from both parents. Then, it generates consensus sequences based on the clustering results and performs remapping. Finally, FindCSV determines CSVs by analyzing the new mapping results.

The experimental results demonstrate that while the FindCSV algorithm performs slightly worse than SVcnn in detecting simple SVs, it outperforms the other methods in the detection of CSVs. However, FindCSV still exhibits certain limitations in its ability to accurately detect CSVs. For instance, due to the presence of noise in the sequencing data, FindCSV may occasionally misclassify some simple SVs as CSVs.

Methods

The design of FindCSV

The detection of CSVs faces two significant challenges, as discussed in the previous section. The first challenge is the absence of a precise definition for CSVs. To address this issue, we propose the following definition: Multiple SVs from the same parent with breakpoint distances less than 1000bp are considered CSVs. As shown in Fig. 4, When there are no other SVs near an SV, it is a simple SV. when two SV breakpoints from either the father's or mother's genome are within a distance of less than 1000 bp, these two SVs are considered as CSVs. Otherwise, these two SVs are considered as two independent simple SVs. However, if the distance between SVs is less than 1000 bp but originates from both parents, they are considered as multi-allelic SVs. The second challenge lies in the diversity of CSVs, which makes it difficult for detection methods to identify and report the correct breakpoints accurately. Although

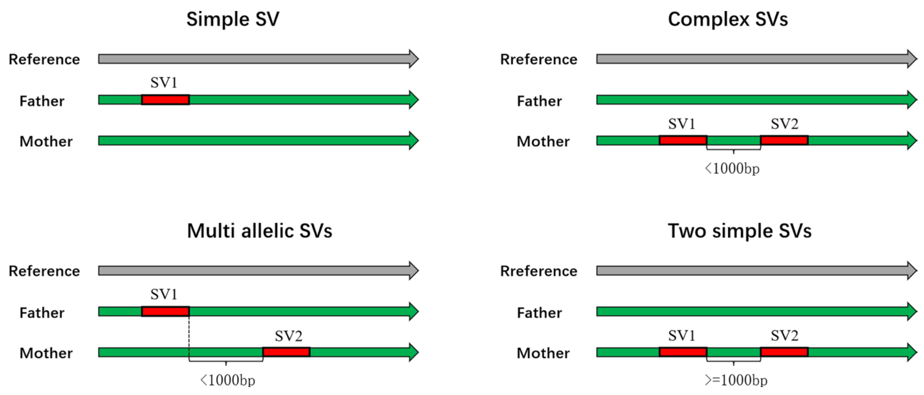

Fig. 4 The figure illustrates the differences between simple SV, CSVs, and multi-allelic SVs

there are theoretically infinite types of CSVs, the most common types in real data are DEL+INS and DEL+INV.

To overcome the second challenge, we employ the following approach: 1. Identification of candidate SV regions. 2. Determination of parental allele presence. 3. Clustering of parental reads. 4. Generation of consistent sequences. 5. Remapping of sequences. 6. Analysis of remapping results. Furthermore, to address false positive SVs, we introduce a deep learning model that screens and filters out such SVs. The detailed steps of this proposed method, including the identification of candidate SV regions, clustering of parental reads, generation of consistent sequences, remapping, and analysis of remapping results, are outlined in the next section.

Overview of FindCSV

The input for FindCSV comprises two components: (i) a sorted long-read BAM file and (ii) a FASTA file. FindCSV primarily consists of three main steps: (1) Identify candidate SV regions, (2) Cluster reads and construct consensus sequences, and (3) Filter and output SVs. Figure 5 illustrates the steps involved in FindCSV, while the detailed steps are explained in the following sections.

Identify candidate SV regions

The first step of FindCSV involves identifying candidate SV regions, where SVs may exist. In contrast to other methods, this method allows us to focus the subsequent analysis on these candidate regions. This streamlines the detection process and improves efficiency, compared to analyzing the entire genome.

Parameter estimation Before detecting candidate SV regions, FindCSV performs preprocessing on the BAM file by estimating the average coverage, denoted as $\bar{C}$. This estimation is achieved by randomly selecting 1000 positions from the reference genome and calculating the average coverage of these positions using the following formula:

$$\bar{C} = \frac{1}{1000}\sum_{i=1}^{1000} C_i$$

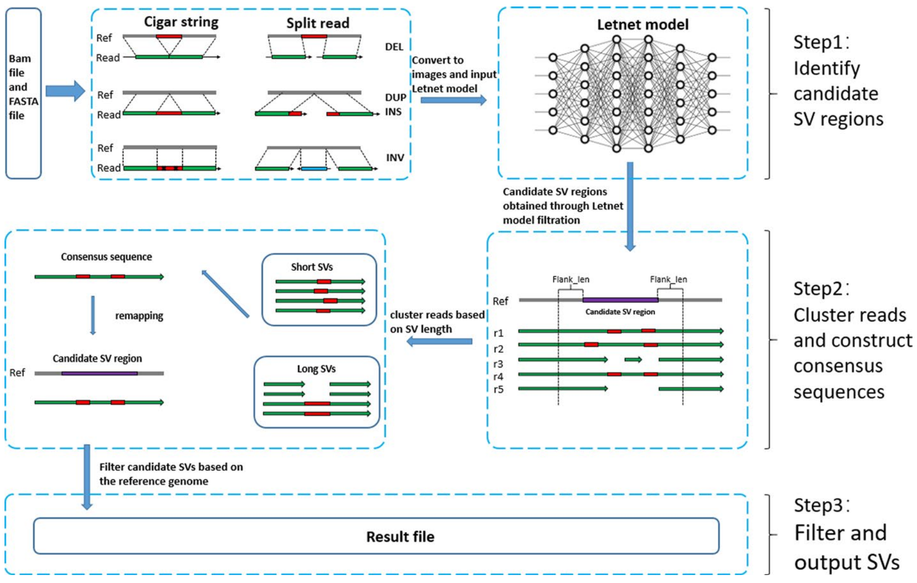
Bam file and FASTA file
Cigar string
Split read
Ref
Read
DEL
DUP
INS
INV
Convert to images and input Letnet model
Letnet model
Step1: Identify candidate SV regions
Candidate SV regions obtained through Letnet model filtration
Flank_len
Candidate SV region
r1
r2
r3
r4
r5
Step2: Cluster reads and construct consensus sequences
cluster reads based on SV length
Consensus sequence
remapping
Short SVs
Long SVs
Filter candidate SVs based on the reference genome
Result file
Step3: Filter and output SVs

Fig. 5 The overview of FindCSV. The input consists of a sorted BAM file and a FASTA file. FindCSV mainly consists of three main steps: (1) Identify candidate SV regions, (2) Cluster reads and construct consensus sequence, and (3) Filter and output SVs

In certain cases, certain regions in the genome exhibit exceptionally high coverage, sometimes exceeding dozens of times the average coverage. Consequently, it becomes challenging to accurately detect SVs in these regions. Fortunately, these high-coverage regions contain very few SVs. Leveraging this observation, FindCSV implements a filtering mechanism to discard these high-coverage regions. Specifically, if a region has coverage that surpasses ten times the average coverage, it is filtered out and excluded from further analysis. By filtering out these high-coverage regions, FindCSV aims to mitigate the complexity introduced by read similarity between different regions, focusing its efforts on regions with more reliable read assignments and a higher likelihood of containing SVs.

Detect candidate SV regions To identify candidate SV regions, FindCSV follows a two-step approach. In the first step, it searches for all potential SVs and generates a list of candidate SVs. The second step involves merging these candidate SVs to obtain candidate SV regions. FindCSV applies three criteria to filter out alignments that are deemed useless, retaining only those alignments that satisfy the following conditions:

1 The MAPQ of alignment is greater than 20.
2 The alignment is primary.
3 The alignment length is greater than 1000.

By applying these filtering criteria, FindCSV ensures that only alignments that meet specific quality and length requirements are considered as potential SVs, enhancing the reliability and accuracy of the subsequent steps in the SV detection process. To detect candidate SVs, FindCSV employs two complementary approaches: one based on CIGAR strings and the other one based on split reads. These approaches complement each other and enhance the accuracy of SV detection.

CIGAR strings FindCSV examines the CIGAR strings of aligned reads to identify potential SVs. The CIGAR string represents the alignment pattern between the read and the reference genome. By analyzing variations in the CIGAR strings, such as insertions, deletions, or mismatches, FindCSV can identify regions where SVs may be present.

Split reads FindCSV utilizes split reads to detect candidate SVs. Split reads are reads that span an SV breakpoint, with a portion of the read aligning to one genomic location and another portion aligning to a different location. By identifying split reads, FindCSV can infer the presence of SVs and determine their breakpoints. More detailed steps are outlined in Sect. 3 of the supplementary file.

By combining these two approaches, FindCSV maximizes its ability to detect candidate SVs. A length-6 tuple is used to record the characteristics of each candidate SV, which consists of the following elements: *.(Chr, S, E, L, T, R).* Here is the breakdown of each element in the tuple: Chr refers to the chromosome name, S and E represent the start and end positions of the SV respectively, L denotes the SV length, T indicates the type of SV (DEL, INS, INV), and R specifies the name of read where the SV is located.

After obtaining the tuple for each candidate SV, FindCSV proceeds to merge tuples with closer distances to generate candidate SV regions. This merging process is performed according to the following approach: (i) For each tuple, FindCSV creates a length-4 tuple (Chr, S, E, 1). This new tuple serves as the initial region, where the value '1' represents the initial read count in this region. (ii) Let region1 = (Chr1, S1, E1, 1) and region2 = (Chr2, S2, E2, 1) be two initial regions. If these two regions are located on the same chromosome and the distance between their breakpoints is less than 1000bp, FindCSV merges them into a single region. The merged region is recorded as (Chr1, min(S1, S2), max(E1, E2), 2), where the value '2' represents the combined read count of the two initial regions. (iii) This merging process continues iteratively until no further regions can be merged. (iv) To eliminate the effects of noise, FindCSV retains only the regions with final values greater than three. These filtered regions are considered as the candidate SV regions. By merging tuples with closer distances, FindCSV aggregates candidate SVs into candidate SV regions, providing a more comprehensive and accurate representation of the SVs present in the analyzed data.

Convert regions to images In this section, the regions obtained from the previous steps are converted into images. To gather more comprehensive information about SVs, FindCSV introduces a parameter called "flank_len", denoted as F. For each candidate SV region (Chr, S, E), FindCSV extracts all CIGAR strings within an extended region (Chr, S-F, E+F). The extended region ensures that an adequate genomic context around the SV breakpoints is included. The value of F is determined as the minimum of (200, 2*(E-S)), where 200 represents a fixed length, and 2*(E-S) ensures that the extended region covers twice the length of the SV region.

To represent the CIGAR strings of each candidate SV region, FindCSV utilizes a five-color image. In this image representation, each line corresponds to a read, and each column corresponds to a position on the reference genome. The conversion process involves converting each character of the CIGAR strings into a pixel, following specific rules:

1 The M (Match) of the CIGAR string with a plus strand (+) is represented as a yellow pixel.
2 The Match of the CIGAR string with a minus strand (-) is represented as a blue pixel.
3 The D (DEL) of the CIGAR string is represented as a black pixel.
4 The I (INS) of the CIGAR string is represented as a red pixel.
5 The X (Mismatch) of the CIGAR string is represented as a green pixel.

By converting each character of the CIGAR strings into pixels following these rules, FindCSV creates a five-color image that visually represents the variations and alignment patterns of the reads within the candidate SV region. However, there is a challenge when representing INSs in the image because they do not occupy a position on the reference genome. This means that if FindCSV uses columns to represent positions on the reference, the insertions will not be displayed in the image, and their information will be lost. To address this issue, FindCSV implemented the following method: In the BAM file, the majority of positions are occupied by matches (M). FindCSV leverages this fact to replace a portion of the matches with insertions (I) based on the insertion lengths. To avoid excessive information loss for matches (M), for every 10bp insertions (I), FindCSV replaces one match (M) on the reference with an insertion (I). For example, if the length of an insertion is 50 base pairs, FindCSV identifies five consecutive match characters (M) on the reference genome near the insertion and replaces them with five insertion characters (I). By incorporating these modifications, FindCSV ensures that the insertions are represented in the resulting image. After the replacement process, the region is converted into an image, which is referred to as an SV image. In the SV image, a vertical red line is used to indicate the presence of an insertion, providing a visual representation of the inserted sequence. Figure 6 illustrates the SV images representing three different types of SVs, showcasing the ability of FindCSV to visually capture and differentiate various types of SVs.

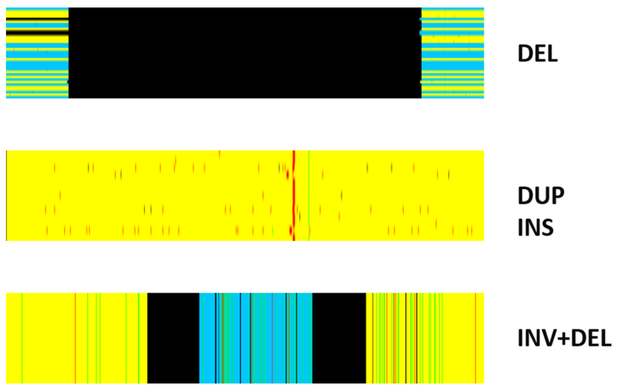

Fig. 6 The alignments in (Chr, S-F, E+F) are converted into an image. Each read occupies a row in the image. Yellow and blue pixels represent matches, black pixels represent deletions, red pixels represent Insertions and green pixels represent mismatchs

Train LetNet model In this section, FindCSV employs a Convolutional Neural Network model to filter the candidate SV regions. Specifically, FindCSV selects the LeNet model as the filtering model. The LeNet model consists of three convolutional layers, two subsampling layers, and two fully connected layers. These layers are designed to extract features and learn representations from the input data. The detailed parameters of the seven layers in the LeNet model are provided in Fig. 7.

To accommodate the fixed input image size requirement of the LeNet model, it is necessary to normalize the previously obtained SV images. This normalization process involves using the resize function from the Python library to resize the images to a uniform size of 224*224*3 pixels. The training dataset plays a crucial role in training the LeNet model. FindCSV utilizes the HG002_SVs_Tier1_v0.6 dataset, which is widely accepted as a benchmark dataset in SV research. However, this benchmark dataset only includes two types of SVs: DELs and INSs. To improve the performance of the model, FindCSV augments the training data by simulating additional types of SVs. Specifically, FindCSV generates 4000 INVs and 3258 regions with no SVs (Abbreviated as noSVs). The noSVs are randomly selected regions from the reference genome, ensuring that there are no SVs present in those regions. The generation process for simulating INVs is slightly more complex. First, a region is randomly selected from the reference genome, and it is confirmed that there are no SVs in that region. Then, the region is reversed to create an inverted mutation. Finally, the sequencing data is remapped onto the inverted reference genome to obtain the mapping results for the simulated INVs. This way, FindCSV obtains a training dataset comprising 20,000 regions, including four types: 5463 DELs, 7279 INSs, 4000 INVs, and 3258 noSVs.

Subsequently, FindCSV converts these regions into two types of images based on ONT data and HiFi data. This results in a total of 40,000 images in the training dataset. FindCSV inputs these normalized SV images into the LeNet model and trains it using a ten-fold cross-validation approach. The dataset of 40,000 images is divided into ten groups, each containing 4000 images. The training and validation data are alternated, with nine groups used for training and one group used for validation in each iteration. The training process is performed for 5000 iterations, and the accuracy of the model is recorded. Finally, the model with the highest accuracy is retained for further use.

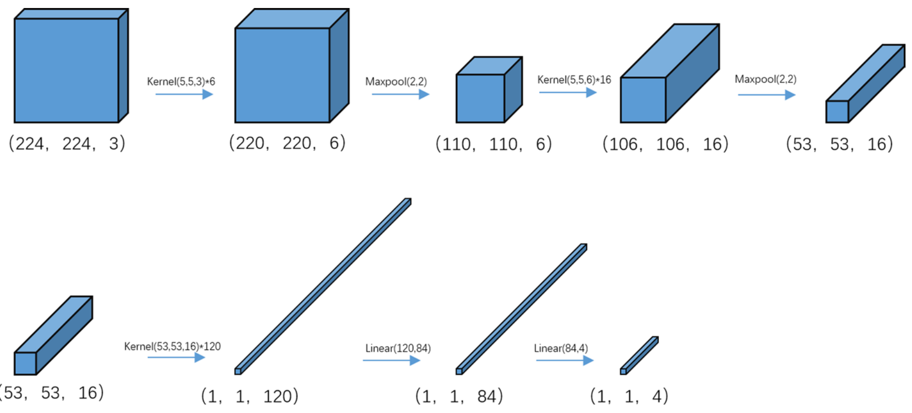

Fig. 7 The detailed parameters of the seven layers in the LeNet model

After training the model, FindCSV applies the trained LeNet model to the SV images of candidate SV regions obtained in the previous steps. The model is utilized to determine the probability of an SV occurring within each region. The label with the highest probability is selected as the judgment result. If the model determines that there is no SV present in a particular region, FindCSV discards the candidate SV region. However, if the model identifies a DEL, INS, or INV, FindCSV proceeds to calculate the precise length and breakpoints of the SV in the subsequent step.

Cluster reads and construct consensus sequences

After obtaining the filtered candidate SV regions in the previous steps, FindCSV proceeds to cluster the reads within these regions based on their alignments. The goal of this clustering step is to identify potential multi-allelic SVs originating from both parents. By clustering the reads from both parents, FindCSV aims to distinguish different alleles and their corresponding alignments within the SV region. This clustering allows for the identification of variations that may exist between the two parental genomes. Once the clustering is performed and the reads are grouped accordingly, FindCSV constructs a consensus sequence based on the reads within each cluster. The consensus sequence represents the most likely sequence representation of the alleles present within the SV region. Finally, FindCSV obtains the final SV by remapping the consensus sequence onto the reference genome. This remapping process aligns the consensus sequence to the reference genome, providing the precise location and sequence information of the identified SV.

Cluster reads in candidate SV region FindCSV extracts all reliably aligned reads for each candidate SV region and refines them to obtain a candidate SV. However, in candidate SV regions, particularly in repeat regions (Chr, S, E), some reads may have incorrect alignments. Figure 8 illustrates examples of read alignments in repeat regions, and it can be observed that the alignment of $Read4$, $Read6$, $Read7$ in Fig. 8 leads to two small false positive SVs, which can impact the overall detection results. To mitigate the impact of wrong alignments, FindCSV employs a filtering strategy and retains only two types of long reads. The first type includes reads that span the entire region (Chr, S-1000, E+1000), such as $Read1 - Read4$ in Fig. 8. The second type comprises reads that have two or more alignments, such as $Read5$ in the figure. By retaining these two types of long reads, FindCSV ensures that the majority of the retained reads are less susceptible to incorrect alignments caused by repeat regions. In Fig. 8, the reads ($Read1 - Read5$) within the green and blue boxes are considered reliable reads and are retained for further analysis. Conversely, the reads ($Read6 - Read7$) within the red box are deemed unreliable reads and are filtered out. This filtering process, which removes unreliable reads, enables FindCSV to achieve improved results, particularly in repeat regions where incorrect alignments can be more prevalent.

For reliable reads, FindCSV performs a merging process if a read contains more than one SV within the candidate SV region. This merging process aims to consolidate multiple SVs into a single, correctly identified SV. For instance, in the case of $Read4$ depicted in Fig. 8, which exhibits two small false positive DELs, FindCSV endeavors to merge them into a single, accurate DEL. Suppose a read contains two SVs, namely SV1 (Chr, S1, E1, L1) and SV2

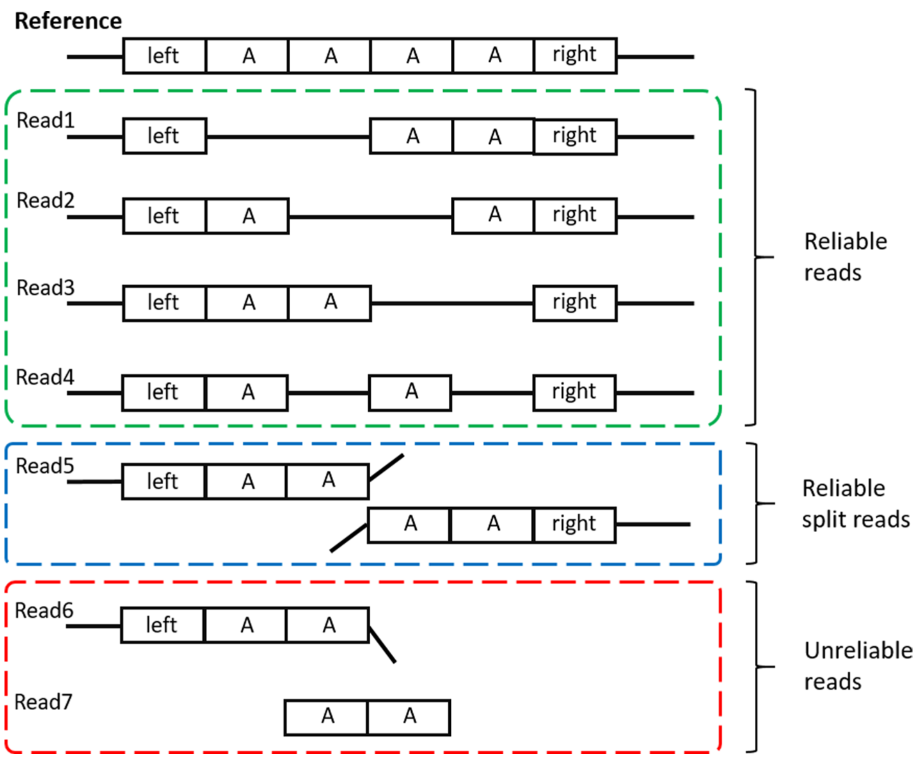

Fig. 8 In the provided figure, we present the alignments of seven distinct reads. The reads enclosed within the green and blue boxes will be classified as reliable reads and will be retained for further analysis. Conversely, the reads encompassed by the red box will be identified as unreliable reads and will be filtered out from subsequent processing

Table 1 The run time and memory of different methods

	FindCSV	DeBreak	cuteSV	Sniffles	SVision	SVcnn
Time	17 h 34 m	48 m	16 m	6 h 42 m	2 d 21 h 45 m	15 h 42 m
Memory	467 MB	2.05 GB	1.98 GB	585 MB	1.2 GB	324 MB

(Chr, S2, E2, L2). FindCSV merges these SVs to create a merged SV, denoted as SV_m: (Chr, min(S1, S2), max(E1, E2), L1 + L2). By combining the start positions (S1, S2), end positions (E1, E2), and lengths (L1, L2), FindCSV forms a merged SV with adjusted start and end positions and an updated length. In addition, even for reliable reads that do not contain any SVs within the candidate SV region, FindCSV still records them as (Chr, 0, 0, 0) to maintain a comprehensive record of all the reliable reads. Subsequently, FindCSV proceeds to cluster the reads based on the length of the SVs within each candidate SV region. For a candidate SV region with N reads, FindCSV calculates the average SV length, denoted as $\bar{L}$. The formula for calculating the average SV length is as follows:

$$\bar{L} = \frac{1}{N}\sum_{i=1}^{N} L_i$$

FindCSV classifies the candidate SV regions into two types, namely long SVs and short SVs, based on the average SV length ($\bar{L}$) exceeding or being less than 500bp. The clustering methods employed for these two types of regions are as follows:

Long SVs ($\bar{L}$ > 500 bp) Sort the candidate SV region's SVs based on SV length. Each SV is recorded as an individual cluster, with the average length of all SVs in the cluster set as the cluster_length. If the difference in cluster_length between two clusters is less than 20%, the two clusters are merged into a single cluster. The above merging process is repeated for the remaining clusters. The clusters are continuously compared, and if the difference in cluster_length is below the specified threshold, they are merged. This iterative merging continues until no further clusters can be merged based on the given criteria.

Short SVs ($\bar{L}$ ≤ 500 bp) For short SVs, the influence of sequencing noise on SV length is significant. Consequently, distinguishing between two short SVs of similar length can be challenging. To overcome this issue, FindCSV utilizes hierarchical clustering to separate these reads and identify distinct SVs within the cluster. Specifically, if a bimodal distribution is observed for all SV lengths within a given cluster, it indicates the presence of two heterozygous SVs. The detailed steps for this approach can be found in Sect. 6 of the supplementary file.

Construct consensus sequences Following the aforementioned steps, FindCSV proceeds to perform read clustering for the candidate SV regions. The objective of this step is to construct a consensus sequence based on the reads within each cluster. This process aims to reduce noise inherent in third-generation sequencing data and improve mapping results. The specific steps to get consensus sequences are shown in Sect. 9 of the supplementary file.

Once the consensus sequence is obtained, FindCSV performs remapping of the consensus sequence to the reference genome. This remapping process enables the detection of SVs in the new mapping results. If two or more SVs are identified in the new mapping result, and their breakpoint distance is less than 1000 bp, FindCSV merges these SVs into a single CSV. The breakpoint positions and types of each SV are recorded for further analysis.

Filter and output SVs

In the previous step, FindCSV generates a set of candidate SVs and proceeds to filter them based on their lengths. Only candidate SVs with lengths greater than 30 bp are retained for further analysis. However, due to limitations in sequencing technology, it has been observed that there are many false SVs in simple repeat regions. To address this issue, FindCSV implements a stricter criterion to remove false SVs specifically in simple repeat regions. The determination of whether a region is a simple repeat region is described in Sect. 7 of the supplementary file. When a candidate SV is located in a simple repeat region, FindCSV examines the SV lengths of all reads at that location. If it is found that more than half of the reads have SV lengths greater than 40 bp, the candidate SV is considered valid and retained for further analysis. By employing this stricter criterion, FindCSV aims to mitigate the impact of false SVs in simple repeat regions, ensuring that only reliable SVs are included in the final results.

Finally, for each candidate SV that is retained after the filtering process, FindCSV performs the following calculations: the reliable read count (R_c) and the support read count (R_s). These metrics are then used to calculate the support rate, denoted as Rate, for each candidate SV.

$$Rate = R_s/R_c * 100\%$$

If the support rate of a candidate SV is determined to be greater than 20% and the number of supporting reads exceeds 3 (default threshold), FindCSV considers these SVs as true positive SVs and includes them in the final output. The output SVs are then saved in a Variant Call Format file.

Performance measure

To assess the performance of different detection methods, this paper employs three evaluation metrics: Recall, Precision, and F1-score. These metrics provide quantitative measures of the method's ability to accurately detect and classify true positive and false positive results. All three measurements range between 0 and 1, and they are defined as follows:

$$\text{Recall} = \frac{TP}{TP + FN}$$
$$\text{Precision} = \frac{TP}{TP + FP}$$
$$\text{F1-score} = \frac{2 * recall * precision}{recall + precision}$$

Please note that in the context of evaluating the performance of a detection method, the following definitions are used: TP (True Positives) refers to the number of SVs that are correctly identified by the method and also appear in the benchmark dataset. TP+FN corresponds to the total number of SVs present in the benchmark dataset, regardless of whether they are detected by the method or not. TP+FP represents the total number of SVs predicted by the method, including both true positives and false positives.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-024-05937-w.

Additional file 1

Acknowledgements
We would like to thank the helpful discussion with Professor Wing-Kin Sung in the computing school of NUS.

Author Contributions
YZ designed the method and implemented the algorithm and complete the manuscript. XS is the major coordinator, who contributed a lot of time and effort to the discussion of this project. All authors read and approved the final manuscript.

Funding
Publication costs were funded by the National Natural Science Foundation of China (Grant No. 62072374). This Project has been funded by the National Natural Science Foundation of China (Grant Nos. 61702420, 61332014, 61702420 and 61772426) and the Top International University Visiting Program for Outstanding Young Scholars of Northwestern Polytechnical University.

Availability of data and materials
The FindCSV is available at https://github.com/nwpuzhengyan/FindCSV. Other datasets' download links are shown in the supplementary file.

Declarations

Ethics approval consent to participate
Not Applicable.

Consent for publication
Not Applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 24 March 2024 Accepted: 18 September 2024
Published online: 28 September 2024

References

1. Sudmant PH, Rausch T, Gardner EJ, Handsaker RE, Abyzov A, Huddleston J, Zhang Y, Ye K, Jun G, Hsi-Yang Fritz M, et al. An integrated map of structural variation in 2,504 human genomes. Nature. 2015;526(7571):75–81.
2. Alkan C, Coe BP, Eichler EE. Genome structural variation discovery and genotyping. Nat Rev Genet. 2011;12(5):363–76.
3. Macintyre G, Ylstra B, Brenton JD. Sequencing structural variants in cancer for precision therapeutics. Trends Genet. 2016;32(9):530–42.
4. Weischenfeldt J, Symmons O, Spitz F, Korbel JO. Phenotypic impact of genomic structural variation: insights from and for human disease. Nat Rev Genet. 2013;14(2):125–38.
5. Rovelet-Lecrux A, Hannequin D, Raux G, Meur NL, Laquerrière A, Vital A, Dumanchin C, Feuillette S, Brice A, Vercelletto M, et al. App locus duplication causes autosomal dominant early-onset Alzheimer disease with cerebral amyloid angiopathy. Nat Genet. 2006;38(1):24–6.
6. Hedges DJ, Hamilton-Nelson KL, Sacharow SJ, Nations L, Beecham GW, Kozhekbaeva ZM, Butler BL, Cukier HN, Whitehead PL, Ma D, et al. Evidence of novel fine-scale structural variation at autism spectrum disorder candidate loci. Mol Autism. 2012;3:1–11.
7. Dennenmoser S, Sedlazeck FJ, Iwaszkiewicz E, Li X-Y, Altmüller J, Nolte AW. Copy number increases of transposable elements and protein-coding genes in an invasive fish of hybrid origin. Mol Ecol. 2017;26(18):4712–24.
8. Lupski JR. Structural variation mutagenesis of the human genome: Impact on disease and evolution. Environ Mol Mutagen. 2015;56(5):419–36.
9. Chiang C, Scott AJ, Davis JR, Tsang EK, Li X, Kim Y, Hadzic T, Damani FN, Ganel L, et al. The impact of structural variation on human gene expression. Nat Genet. 2017;49(5):692–9.
10. Zichner T, Garfield DA, Rausch T, Stütz AM, Cannavó E, Braun M, Furlong EE, Korbel JO. Impact of genomic structural variation in drosophila melanogaster based on population-scale sequencing. Genome Res. 2013;23(3):568–79.
11. Jeffares DC, Jolly C, Hoti M, Speed D, Shaw L, Rallis C, Balloux F, Dessimoz C, Bähler J, Sedlazeck FJ. Transient structural variations have strong effects on quantitative traits and reproductive isolation in fission yeast. Nat Commun. 2017;8(1):14061.
12. Quinlan AR, Hall IM. Characterizing complex structural variation in germline and somatic genomes. Trends Genet. 2012;28(1):43–53.
13. Weckselblatt B, Rudd MK. Human structural variation: mechanisms of chromosome rearrangements. Trends Genet. 2015;31(10):587–99.
14. Hadi K, Yao X, Behr JM, Deshpande A, Xanthopoulakis C, Tian H, Kudman S, Rosiene J, Darmofal M, DeRose J, et al. Distinct classes of complex structural variation uncovered across thousands of cancer genome graphs. Cell. 2020;183(1):197–210.
15. Rausch T, Zichner T, Schlattl A, Stütz AM, Benes V, Korbel JO. Delly: structural variant discovery by integrated paired-end and split-read analysis. Bioinformatics. 2012;28(18):333–9.
16. Layer RM, Chiang C, Quinlan AR, Hall IM. Lumpy: a probabilistic framework for structural variant discovery. Genome Biol. 2014;15:1–19.
17. Chen X, Schulz-Trieglaff O, Shaw R, Barnes B, Schlesinger F, Källberg M, Cox AJ, Kruglyak S, Saunders CT. Manta: rapid detection of structural variants and indels for germline and cancer sequencing applications. Bioinformatics. 2016;32(8):1220–2.
18. Trappe K, Emde A-K, Ehrlich H-C, Reinert K. Gustaf: detecting and correctly classifying SVs in the NGS twilight zone. Bioinformatics. 2014;30(24):3484–90.
19. Rajaby R, Sung W-K. Survindel: improving CNV calling from high-throughput sequencing data through statistical testing. Bioinformatics. 2021;37(11):1497–505.
20. Chen Y, Wang A, Barkley C, Zhao X, Gao M, Edmonds M, Chong Z. Debreak: deciphering the exact breakpoints of structural variations using long sequencing reads (2022)
21. Jiang T, Liu Y, Jiang Y, Li J, Gao Y, Cui Z, Liu Y, Liu B, Wang Y. Long-read-based human genomic structural variation detection with cuteSV. Genome Biol. 2020;21:1–24.
22. Sedlazeck FJ, Rescheneder P, Smolka M, Fang H, Nattestad M, Von Haeseler A, Schatz MC. Accurate detection of complex structural variations using single-molecule sequencing. Nat Methods. 2018;15(6):461–8.
23. Smolka M, Paulin LF, Grochowski CM, Horner DW, Mahmoud M, Behera S, Kalef-Ezra E, Gandhi M, Hong K, Pehlivan D, et al. Comprehensive structural variant detection: from mosaic to population-level. BioRxiv. 2022-04 (2022)
24. Cretu Stancu M, Van Roosmalen MJ, Renkens I, Nieboer MM, Middelkamp S, De Ligt J, Pregno G, Giachino D, Mandrile G, Espejo Valle-Inclan J, et al. Mapping and phasing of structural variation in patient genomes using nanopore sequencing. Nat Commun. 2017;8(1):1326.

25. Gong L, Wong C-H, Cheng W-C, Tjong H, Menghi F, Ngan CY, Liu ET, Wei C-L. Picky comprehensively detects high-resolution structural variants in nanopore long reads. Nat Methods. 2018;15(6):455–60.
26. Heller D, Vingron M. SVIM: structural variant identification using mapped long reads. Bioinformatics. 2019;35(17):2907–15.
27. English AC, Salerno WJ, Reid JG. Pbhoney: identifying genomic variants via long-read discordance and interrupted mapping. BMC Bioinform. 2014;15:1–7.
28. Lin J, Wang S, Audano PA, Meng D, Flores JI, Kosters W, Yang X, Jia P, Marschall T, Beck CR, et al. SVision: a deep learning approach to resolve complex structural variants. Nat Methods. 2022;19(10):1230–3.
29. Zheng Y, Shang X. SVcnn: an accurate deep learning-based method for detecting structural variation based on long-read data. BMC Bioinform. 2023;24.
30. Ho SS, Urban AE, Mills RE. Structural variation in the sequencing era. Nat Rev Genet. 2020;21(3):171–89.
31. Mantere T, Kersten S, Hoischen A. Long-read sequencing emerging in medical genetics. Front Genet. 2019;10:432668.
32. Dashnow H, Lek M, Phipson B, Halman A, Sadedin S, Lonsdale A, Davis M, Lamont P, Clayton JS, Laing NG, et al. Stretch: detecting and discovering pathogenic short tandem repeat expansions. Genome Biol. 2018;19:1–13.
33. Chiu R, Rajan-Babu I-S, Friedman JM, Birol I. Straglr: discovering and genotyping tandem repeat expansions using whole genome long-read sequences. Genome Biol. 2021;22(1):224.
34. Roberts RJ, Carneiro MO, Schatz MC. The advantages of SMRT sequencing. Genome Biol. 2013;14:1–4.
35. Jain M, Olsen HE, Paten B, Akeson M. The oxford nanopore minion: delivery of nanopore sequencing to the genomics community. Genome Biol. 2016;17:1–11.
36. Zook JM, Hansen NF, Olson ND, Chapman L, Mullikin JC, Xiao C, Sherry S, Koren S, Phillippy AM, Boutros PC, et al. A robust benchmark for detection of germline large deletions and insertions. Nat Biotechnol. 2020;38(11):1347–55.
37. Hon T, Mars K, Young G, Tsai Y-C, Karalius JW, Landolin JM, Maurer N, Kudrna D, Hardigan MA, Steiner CC, et al. Highly accurate long-read HiFi sequencing data for five complex genomes. Sci Data. 2020;7(1):399.
38. Garg S, Fungtammasan A, Carroll A, Chou M, Schmitt A, Zhou X, Mac S, Peluso P, Hatas E, Ghurye J, et al. Chromosome-scale, haplotype-resolved assembly of human genomes. Nat Biotechnol. 2021;39(3):309–12.
39. Nurk S, Koren S, Rhie A, Rautiainen M, Bzikadze AV, Mikheenko A, Vollger MR, Altemose N, Uralsky L, Gershman A, et al. The complete sequence of a human genome. Science. 2022;376(6588):44–53.
40. 1000 Genomes Project Consortium. An integrated map of genetic variation from 1,092 human genomes. Nature 2012;491(7422):56.