Published online 6 June 2017 *Nucleic Acids Research, 2017, Vol. 45, No. 14* ***e132***
doi: 10.1093/nar/gkx511

COSINE: non-seeding method for mapping long noisy sequences

Pegah Tootoonchi Afshar[1] and Wing Hung Wong[2,*]

[1]Department of Electrical Engineering, School of Engineering, Stanford University, Stanford, CA 94305, USA and
[2]Department of Statistics and Department of Biomedical Data Science, Stanford University, Stanford, CA 94305, USA

Received August 28, 2016; Revised May 16, 2017; Editorial Decision May 29, 2017; Accepted June 04, 2017

ABSTRACT

Third generation sequencing (TGS) are highly promising technologies but the long and noisy reads from TGS are difficult to align using existing algorithms. Here, we present COSINE, a conceptually new method designed specifically for aligning long reads contaminated by a high level of errors. COSINE computes the context similarity of two stretches of nucleobases given the similarity over distributions of their short *k*-mers ($k = 3$–4) along the sequences. The results on simulated and real data show that COSINE achieves high sensitivity and specificity under a wide range of read accuracies. When the error rate is high, COSINE can offer substantial advantages over existing alignment methods.

INTRODUCTION

The advent of next generation sequencing technologies (NGS) has revolutionized many areas of biology and medicine (1–5). NGS technologies can be classified into second generation sequencing (SGS) or third generation sequencing (TGS) technologies, as summarized in Table 1. SGS methods measure signals from a colony of identical copies of the template to reach high signal to noise ratio (6). Hence they achieve high accuracy (>98%) in base calling in the early cycles of a synchronized chain growing process, however, the synchronization deteriorates with each new cycle and this leads to loss of accuracy in the later cycles. As a result, SGS can only generate short reads of no more than a few hundred base pairs long. In contrast, TGS methods measure real time signal from a single DNA template and the read length is limited only by stochastic events such as dissociation of the DNA polymerase from the template or DNA molecule length. Currently, a TGS run typically produces a set of reads of variable lengths ranging from 500 bp to ~70 kb, with median length around 5–15 kb. Although its throughput (currently up to 10^6 templates per run) is much lower than that of SGS (10^9 reads per run), TGS technologies are highly scalable and can be expected to achieve very high throughput eventually. Furthermore, some semiconductor based TGS instruments are portable, require little input DNA, and have a simple workflow. These are highly desirable properties for many applications.

Since TGS is based on a single molecule measurement, it is intrinsically noisy. The base-calls from raw TGS signal has error rate in the range of 15–35% (7–10). To improve sequencing accuracy, the primary strategy is to identify a set of noisy reads (call a read cluster) covering the same target sequence (e.g. a particular region of the genome being sequenced) and infer a consensus sequence from such reads. This is illustrated in Figure 1 which has two parts: (a) read cluster identified by alignment to a reference and (b) read cluster identified by pairwise alignments. The situations depicted in Figures 1A and 1 B arise respectively in sequence alignment and sequence assembly applications. In Figure 1 A we need to map a noisy long read to a location within a large reference sequence. The word 'map' is used to indicate that the alignment does not have to be precise as long as the alignment location on the reference is roughly correct. Afterwards, detailed alignment can be performed by more time consuming algorithms based on dynamic programing. On the other hand, in Figure 1B we need to align two noisy long reads to each other to determine if they overlap. Note that in this case the aligner must be able to handle a very high level of error, as the level of mismatches between two reads is about two times that between a read and the reference. In this paper, we propose a new approach for the mapping task of Figure 1A. Although the proposed method should also be relevant for assembly task of Figure 1B, we will treat that topic in a separate paper.

There are many algorithms available to map sequence reads to a reference, and several of them have been designed to handle long reads (see paragraph below). These aligners are mostly based on the seed-and-extend concept, where exact matches of *k*-mers (seeds) between the read and the reference are first identified, and then more detailed alignment is performed around the seed locations in the reference. The high error rate in TGS reads causes serious difficulty for seed-based methods in alignment to large genomes. First, *k* has to be large enough so that the expected number of exact matches is small. However when the error rate

*To whom correspondence should be addressed. Tel: +1 650 725 2915; Fax: +1 650 725 8977; Email: whwong@stanford.edu

Table 1. Summary of characteristics of different next generation sequencing technologies

Name	Single molecule	Cycling	Semi-conductor	Read length	Generation
Illumina	No	No	No	Short	2nd
Ion torrent	No	No	Yes	Short	2nd
Pacific biosciences	Yes	Yes	No	Long	3rd
Oxford nanopore	Yes	Yes	Yes	Long	3rd

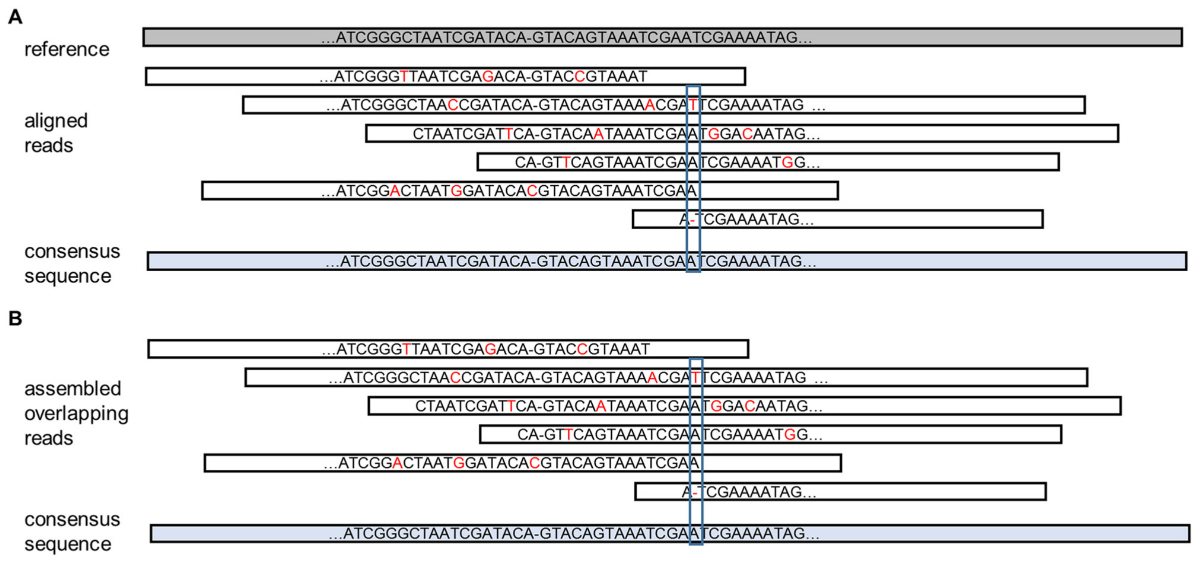

A
reference
...ATCGGGCTAATCGATACA-GTACAGTAAATCGAATCGAAAATAG...
aligned reads
...ATCGGGTTAATCGAGACA-GTACCGTAAAT
...ATCGGGCTAACCGATACA-GTACAGTAAAACGATTCGAAAATAG ...
CTAATCGATTCA-GTACAATAAATCGAATGGACAATAG...
CA-GTTCAGTAAATCGAATCGAAAATGG...
...ATCGGACTAATGGATACACGTACAGTAAATCGAA
A-TCGAAAATAG...
consensus sequence
...ATCGGGCTAATCGATACA-GTACAGTAAATCGAATCGAAAATAG...
B
assembled overlapping reads
...ATCGGGTTAATCGAGACA-GTACCGTAAAT
...ATCGGGCTAACCGATACA-GTACAGTAAAACGATTCGAAAATAG ...
CTAATCGATTCA-GTACAATAAATCGAATGGACAATAG...
CA-GTTCAGTAAATCGAATCGAAAATGG...
...ATCGGACTAATGGATACACGTACAGTAAATCGAA
A-TCGAAAATAG...
consensus sequence
...ATCGGGCTAATCGATACA-GTACAGTAAATCGAATCGAAAATAG...

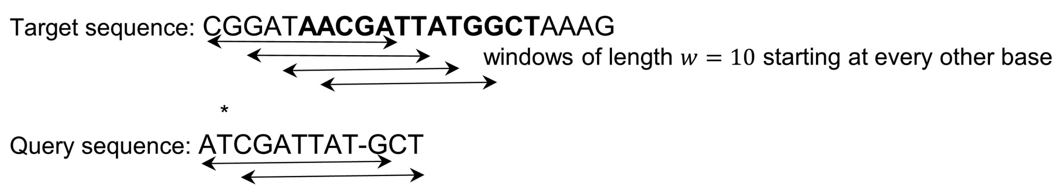
Figure 1. Highly accurate consensus sequence is generated from a set of noisy reads covering the same region of the target sequence. (**A**) Cluster of reads are generated from aligning reads to the reference. (**B**) Cluster of reads are generated from pairwise alignments.

is high, the chance for a long k-mer match to be correct is very small. Here 'correct' means that the k-mer in the read is a faithful, error-free copy of the reference sequence at the matched position. What is worse, the chance for an incorrect match increases not only with the error rate but also with the size of the reference. To illustrate, two popular seed-based aligners, BLASR and BWA-MEM were used in high-sensitivity mode to align 10 kb reads simulated from the human genome. It is seen that when the per-base error rate in the read increases from 15% to 45%, the percentage of missed and incorrectly aligned reads for BLASR rose from 0.1% to 13.9%, and that for BWA-MEM rose from 1.2% to 8.7% (Table 3). To realize the full potential of TGS technology, it is thus important to develop alternative alignment approaches with performance scaling more gracefully with increase in error rate and reference genome size. In this paper, we propose a 'non-seeding' approach that uses the distributions of short *k*-mers along the sequence as an identifiable characteristic between a reference region and noisy reads sequenced from the same region. In our approach, content similarity nearness (COSINE), we compute the number of occurrence of short *k*-mers (k = 3–4) in spaced windows of several hundred bases long (w = 100–500) for both target and query sequences. Our hypothesis is that similarity of short k-mer distribution is a more robust alignment criterion (i.e. less sensitive to base-call errors) than exact matches of long *k*-mers. Indeed, our numerical experiments show that the performance of COSINE remains acceptable for very noisy reads. For human genome reads with 45% errors, the proportion of reads that are missed or aligned to a wrong location by COSINE is 1.6%, which is much lower than that of seed-based methods (13.9% for BLASR and 8.7% for BWA-MEM, Table 3). Figure 2 illustrates an example of computing *k*-mer count vectors and detecting the sequencing position.

Currently common aligners for TGS reads are mainly based on different variations of seed-and-extend concept. LAST(11) builds spaced suffix array of the reference sequence to find spaced adaptive seeds. In adaptive seed technique, LAST finds variable length matches instead of fixed-length seeds depending on the number of occurrences of each match. It then extends initial hits with gapped alignments using X-drop algorithm. BLASR(12) uses suffix array or BWT-FM data structure to index the target sequence and then finds exact matches shorter than the longest common prefix (LCP) at each read position. BLASR selects top scoring chains of seeds based on the rareness of seeds to perform sparse dynamic programming (SDP). High scored SDP alignment paths are then used to restrict the search region for detailed banded dynamic programming. BWA-MEM (https://arxiv.org/abs/1303.3997) is designed for mapping long sequences to a reference, supporting both single-end (SE) and paired-end (PE) reads. It starts with finding the maximal exact matches at each read position and extending optimal chains of seeds using Z-dropoff banded dynamic programming.

There are also previous works to find sequence similarity between protein or DNA sequences using direct cross-correlation. For instance, MAFFT(13) transforms protein sequences to two vectors based on volume and polarity of

A


Target sequence: CGGATAACGATTATGGCTAAAG
windows of length $w = 10$ starting at every other base
*
Query sequence: ATCGATTAT-GCT

B

Unique k-mers ($k = 2$)	Number of k-mer occurrences in windows starting at offsets i						
	i = 0	i = 2	i = 4	i = 6	i = 8	i = 10	i = 12
AA	1	1	1	0	0	1	2
CA	0	0	0	0	0	0	0
GA	2	2	1	1	1	0	0
TA	1	1	2	1	1	2	1
AC	1	1	1	1	0	0	0
CC	0	0	0	0	0	0	0
GC	0	0	0	0	1	1	1
TC	0	0	0	0	0	0	0
AG	0	0	0	0	0	0	1
CG	2	1	1	1	0	0	0
GG	1	0	0	1	1	1	1
TG	0	0	0	1	1	1	1
AT	1	2	2	2	2	1	1
CT	0	0	0	0	1	1	1
GT	0	0	0	0	0	0	0
TT	0	1	1	1	1	1	0

C

Unique k-mers ($k = 2$)	Number of k-mer occurrences in windows starting at offsets j	
	j = 0	j = 2
AA	0	0
CA	0	0
GA	1	1
TA	1	1
AC	0	0
CC	0	0
GC	0	1
TC	1	0
AG	0	0
CG	1	1
GG	0	0
TG	1	1
AT	3	2
CT	0	1
GT	0	0
TT	1	1

D

Euclidean distance between pair of vectors in query and target sequence		
target\query offset	j = 0	j = 2
i = 0	3.46	3.16
i = 2	2.45	2.45
i = 4	2.45	2.45
i = 6	**2.00**	2.00
i = 8	2.45	**1.41**
i = 10	3.46	2.45
i = 12	4.00	3.16

E

Cosine distance between pair of vectors in query and target sequence		
target\query offset	j = 0	j = 2
i = 0	0.43	0.41
i = 2	0.21	0.25
i = 4	0.21	0.25
i = 6	**0.14**	0.18
i = 8	0.22	**0.09**
i = 10	0.46	0.27
i = 12	0.61	0.45

Figure 2. An example illustrating the computation of *k*-mer count vectors for target and query sequences and comparing their pairwise distance. (**A**) Example of a read sequenced from the target region marked in bold. Mutated and deleted bases are labeled with '*' and '-', respectively. Unique *k*-mers of length 2 are counted within windows of length $w = 10$, each starting with offset $d = 2$ from the previous one. (**B**) The target *k*-mer count vectors for each window starting at position *i*. (**C**) The query k-mer count vectors for each window starting at position *j*. (**D**, **E**) Euclidean and cosine distances between pairs of *k*-mer count vectors from target and query sequences. The associated vectors have the lowest distance within the expected alignment region of $i =$ 6, 8.

amino-acid residues or in (14,15) DNA sequences are converted to four indicator vectors for each nucleotide acid type A, T, C and G, or a vector of complex values assigning distinct 1, –1, i and –i values to each one. Cross-correlation between these numeric vectors has distinguishably higher values when homologous regions in two protein sequences or similar DNA regions are aligned. However these methods are limited to substitution variations and do not support frequent insertions and deletions as matched segments are expected to be aligned at the same offset.

MATERIALS AND METHODS

COSINE similarity score

COSINE finds the content similarity of two sequences at different aligned offsets, which is a different approach compared to the seeding concept that finds groups of perfect or near-perfect matches between the two sequences (see Figure 3). COSINE starts with computing the number of occurrences of consecutive short *k*-mers ($k = 3$ or 4) in windows of length *w* along the sequence. For a window starting at

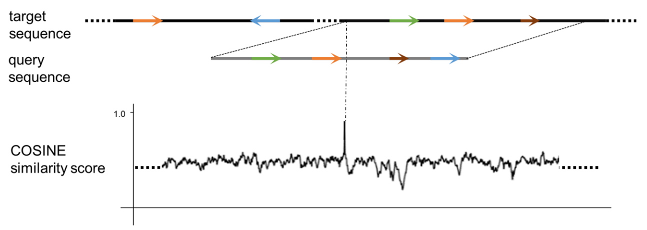


Figure 3. To determine the sequencing location, COSINE computes the overall similarity score of the query sequence across the target, instead of finding clusters of exact matches as in seed-based methods. Each colored arrow indicates a different matched sequence between the query (gray bar) and the target sequence (black bar), a notion used in seed-based methods to determine the sequencing location. Here, the direction of arrow indicates the strand of matched sequence.

position i in a sequence of length l

$$V_i = [n_0, n_1, ..., n_q], \quad 0 \le i < l - w + 1$$

where

$$q = |\Sigma|^k, \quad \Sigma = \{A, T, C, G\}$$
$$\mathcal{Q} = \text{list of k-mers } \{v_j\}, \quad 0 \le j < q$$
$$n_j = count(v_j) \text{ in a window of length } w$$

The local similarity of two sequences, distinguished by superscripts 1 and 2, at positions i^1 and i^2 is defined as the cosine similarity between their k-mer count vectors at positions i^1 and i^2, respectively .

$$S_{i^1 i^2} = \frac{V^1_{i^1} \cdot V^2_{i^2}}{\| V^1_{i^1} \|_2 \| V^2_{i^2} \|_2} \tag{1}$$

Here, sequence 1 may represent the query sequence while sequence 2 represents the reference. The overall similarity score between the two sequences of length l^1 and l^2, aligned at offset m is defined as the average over $S_{i^1 i^2}$ on sampled positions. Assuming positions are selected equally apart with distance d

$$S[m] = \frac{1}{|\mathcal{I}_m|} \sum_{(i^1, i^2) \in \mathcal{I}_m} S_{i^1 i^2} \tag{2}$$

where

$$\mathcal{I}_m = \{(0, m), \ (d, m + d), \ ..., \ (pd, m + pd)\}$$
$$\tilde{l}_{.} = l_{.} - w + 1$$
$$p = \min(\frac{\tilde{l}^1}{d}, \frac{\tilde{l}^2 - m}{d})$$

In presence of deletions and insertions, two aligned sequences at offset m have their error-free segments shifted at different offsets $\tilde{m}$. Therefore, instead of directly correlating sequence nucleotide bases at a fixed offset m, the distribution of short k-mers in a series of windows are compared as a measure of similarity. Here, the advantage of using cosine similarity score compared to other metrics, for instance euclidean distance, is its easy implementation and that computation of S can be done efficiently using fast Fourier transform (FFT) (Supplementary Note S4).

Short k-mer counts performance

COSINE uses distribution of short k-mers within fixed-length windows to detect similar regions and accordingly alignment positions. To demonstrate the effectiveness of this approach, 1000 locations from *Escherichia coli* genome are randomly selected and the cosine similarity is computed between k-mer count vectors of non-overlapping windows and between each window and its randomly muted sequences with average error rates of 15% and 35% (similar performance is achieved using euclidean distance. Results are not shown). Figure 4 presents histograms of cosine similarities for $w = 100$ and $k = 3, 4$ (See Supplementary Figure S1 for different window sizes of $w = 50$, 100, 500 and 1000). This shows that extracted sequences have higher similarity with their mutated versions than from unrelated locations in the genome, but the difference becomes less distinguishable for the higher error rate of 35%. Supplementary Figure S2, represents similar results for more challenging genomes, *S. coelicolor* genome with 72% GC content and human genome. Here, we also observe that the high GC content of *S. coelicolor* genome, increases the overall cosine similarity of unrelated locations, which adversely affects the separation between the two distributions, especially with higher error rate of 35%.

To enhance the discriminatory power, COSINE uses series of spaced windows along the two sequences and computes the similarity score as the average over cosine similarities of k-mer count vectors between associated windows from the two sequences. Figure 5 shows the distribution of COSINE similarity scores for 1000 sequences of 5 kb long selected randomly from *E. coli* genome. The score is computed between each sequence and ten other non-overlapping sequences (in red) and ten simulated sequences by applying random errors to the original read (in blue). The error pattern is an equal distribution over substitution, insertion and deletion error types. The accumulation over cosine similarities of corresponding k-mer count vectors improves the separation of two distributions especially in the case of high error rate.

COSINE accuracy mainly depends on the choice of k-mer size (k), window size (w) and windows shift (d). In general, $k = 4$ has better sensitivity compared to $k = 3$ in higher error rate cases. Smaller window shift (d) increases the accuracy but as it is shown in Results section, setting d up to half the size of window, would not noticeably affect the alignment performance. For three *E. coli*, *S. coelicolor* and human genomes, Supplementary Figures S3–S8 represent the above-mentioned results with different window sizes of $w = 50$, 100, 500 and 1000 and two different simulated error patterns (sub:ins:del proportions) of 34:33:33 with equal insertion and deletion rates and 10:62:28 with about two times insertion rate compared to deletion, which is typical for PacBio reads (12,16). Use of large window size (w = 250–500 bp) increases alignment sensitivity in high indel rates, especially with unequal proportions but the resulting alignment locations are of lower resolution. On the other hand, the choice of short windows (w = 50–100 bp) has higher alignment sensitivity in presence of higher substitution errors and equal insertions and deletions error rates. In general, with only substitution type of error, direct cross-

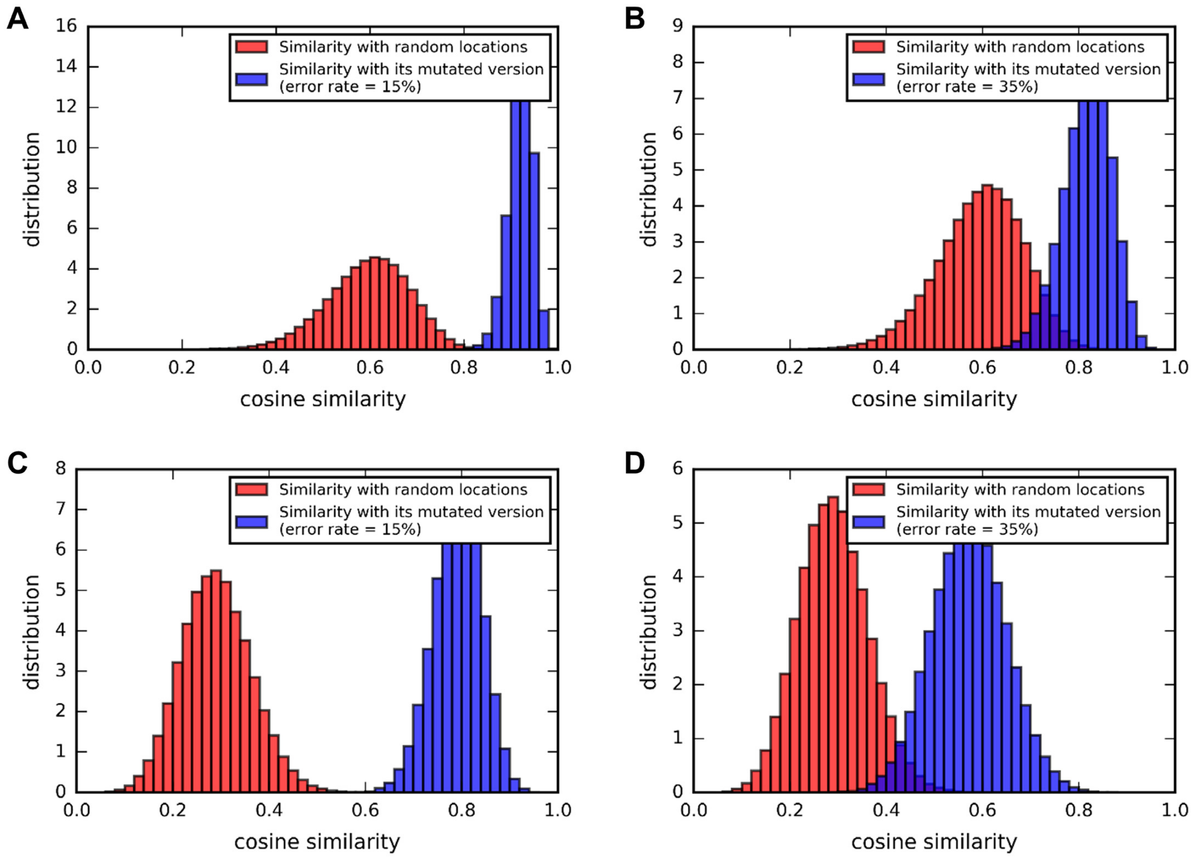


Figure 4. Simulated probability distribution of cosine similarity of *k*-mer count vectors between random positions in *E. coli* genome and with their mutated versions. 1000 locations are randomly selected from *E. coli* genome and the cosine similarity is computed between each position and other non-overlapping windows (in red) and between each position and its 100 randomly mutated sequences (in blue). Parameter settings are w =100 with error rate = 15%, 35% (equal sub, del and ins error ratio). (**A**, **B**) $k = 3$. (**C**, **D**) $k = 4$.

correlation between nucleotide-acid types, as introduced in (17) gives the strongest signal between similar sequences. However, indels alter the offset between exact matches and degrade its performance. For comparison, the same aforementioned data is used to generate the two similarity distributions between unrelated sequences and each sequence and its noisy versions (see Supplementary Figure S11). The score in direct cross-correlation approach is the sum of four cross-correlation values computed between nucleobase indicator vectors of the two sequences (similar to COSINE setting of $k = 1$, $w = 1$, $d = 1$) at offset zero. This is the same as finding the match rate of their gapless alignment. In Supplementary Figure S11A and B, noisy reads are simulated with only substitution errors and results in Supplementary Figures S11C–F include insertions and deletions, as well. As expected, in presence of indels, and especially out of proportion insertions and deletion rates, the original sequence and its noisy simulated reads become less distinguishable compared to sequences from other parts of the genome.

As third generation sequencing technologies have consistent increase on their average read length; to examine the effect of read length on the performance of COSINE approach, above-mentioned analyses are repeated for simulated reads of 5, 10, 15 and 20 kb long from *E. coli* genome (Supplementary Figures S9 and S10). Specifically, in the case of unequal insertion and deletion error rates, increase of read length has negative effect on the average similarity score computation (Equation 2), due to the drift in alignment offset between matched fragments along the two sequences. As discussed in the following subsection on alignment procedure, depending on the error profile, longer reads should be split to shorter fragments to limit the bias in alignment offsets.

Alignment procedure

As an example, Figure 6 shows the similarity score for a simulated noisy read across the whole *E. coli* genome with setting $k = 3$, $d = 10$, $w = 500$. The 5 kb read sequence is extracted from *E. coli* K12 region 2,720,230–2,725,230 and simulated with error rates of 15% and 35% where the highest similarity is detectable close to the sampled position. Offset position m with a high score $S[m]$ (Equation 2) gives the approximate start position for locating the read sequence. A detailed alignment is then derived using local banded dynamic programming (BDP) at the detected region.

In the alignment process, COSINE first selects local peaks as positions with maximum similarity score within consecutive segments along the target sequence (default: segment_size = read length). Local peaks with compara-

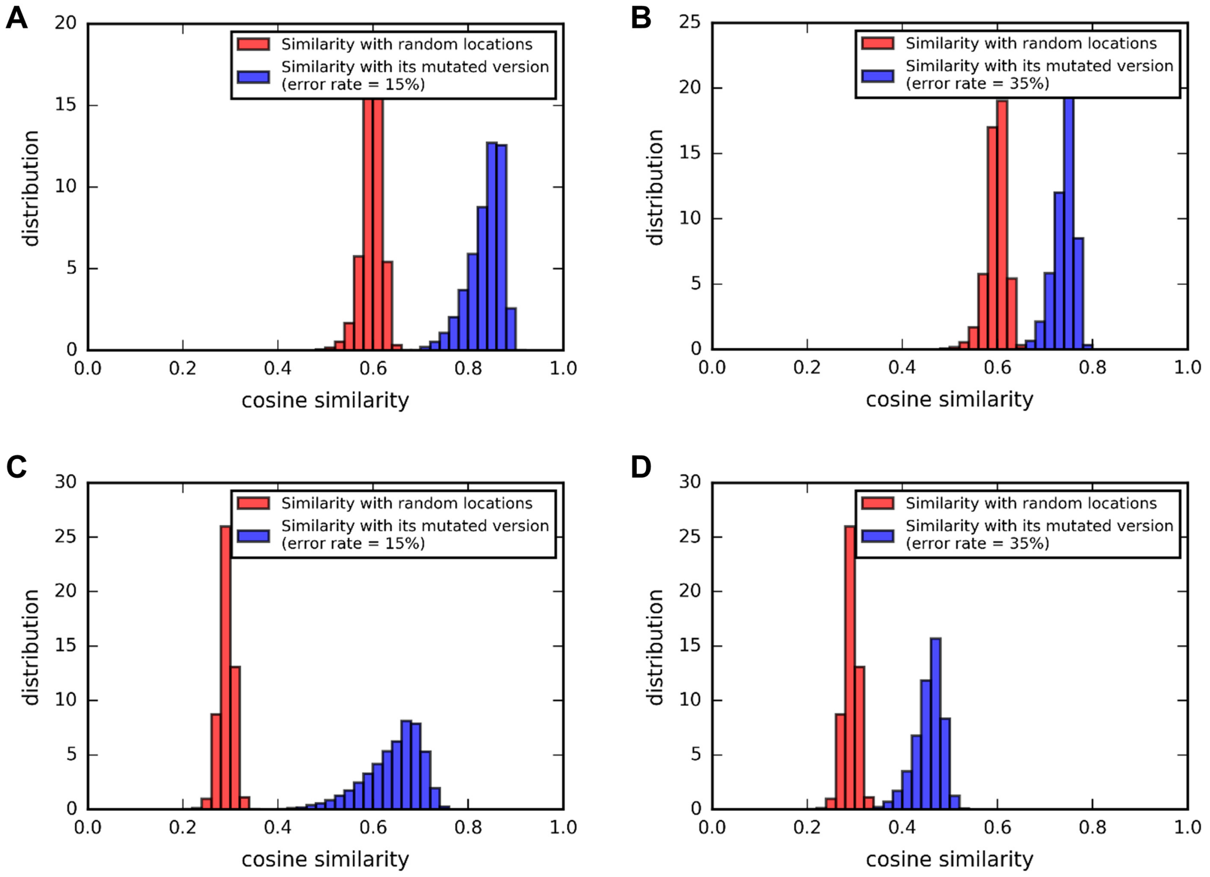


Figure 5. Simulated probability distribution of average cosine similarity of associated k-mer count vectors between 1000 sequences of 5 kb long, randomly selected from *E. coli* genome and with their own simulated noisy versions. The average similarity score is computed between each sequence and 10 other random non-overlapping sequences (in red) and between each sequence and its 10 randomly mutated sequences (in blue). Parameter settings are $d = 10$ and $w = 100$ with error rate = 15%, 35% (equal sub, del and ins error ratio). (**A**, **B**) $k = 3$. (**C**, **D**) $k = 4$.

ble higher similarity scores are marked as significant. The position p_{max} with $p_{max} = \arg\max_p S[p]$ and $S[p_{max}] \geq mean(S) + f \times std(S)$ (default: $f = 2.0$) is considered a significant peak compared to the noise level of mapping the read to unrelated locations (the mean and standard deviation are estimated from randomly sampled score values). Other local peaks with score not less than $g \times std(S)$ from the maximum score are also considered significant (i.e. $S[p] \geq S[p_{max}] - g \times std(S)$, default: $g = 1.0$), limited to a maximum number N_{max} (default: $N_{max} = 10$). Supplementary Figures S12 and S13 show the distribution of peak ranks for correctly mapped reads from aligning simulated reads to *E. coli* and human genome (simulated datasets are generated using PBSIM tool, explained in Results section). For dataset with average simulated error rate of 15%, the highest scored peak (rank = 1) indicates the correct alignment location for ≥99.9% and ≥98.5% of reads in *E. coli* and human genome, respectively. And with average error rate of 35%, ≥99.7% and ≥97.4% of the maximum scored peaks indicate correct mapping positions, respectively. It is natural that with higher error rate and larger genome size, similarity score for repeat and homologous regions become less distinguishable. Therefore, BDP is applied to top candidates to detect the regions with longer stretches of exact matches.

To better examine the effect of repetitive regions on detecting significant peaks, 5000 reads are simulated from a human genome segment with low repetitiveness and a segment with relatively high repetitiveness (Supplementary Note 3). Supplementary Figures S14 and S15 illustrate the distribution of peak ranks for 10 kb long and 5 kb long reads from human genome segments of chr5:0–10 Mb (a lower repetitive region) and chrX:70–80 Mb (a higher repetitive region). As expected, the repeat areas degrade the alignment performance. With average simulated error rate of 15%, the highest scored peaks correctly refer to sequencing location of ≥99.7% and ≥97.4% reads from chr5:0–10 Mb and chrX:70–80 Mb segments, respectively. And with 35% error rate, they correctly determine the alignment location for ≥98.9% and ≥94.6% of reads, respectively. However, with higher error rate of 45% (see Supplementary Figures S14E–S14F and S15E–S15F), there is a clear drop in performance in both scenarios, which indicates the dominant effect of high number of errors irrespective of the complexity of genome sequence.

For long noisy reads, the standard banded dynamic programming is not efficient as the band length is still $O(l)$ due to the high error rate (where l = read length). As a result, COSINE first finds chains of exact matches between

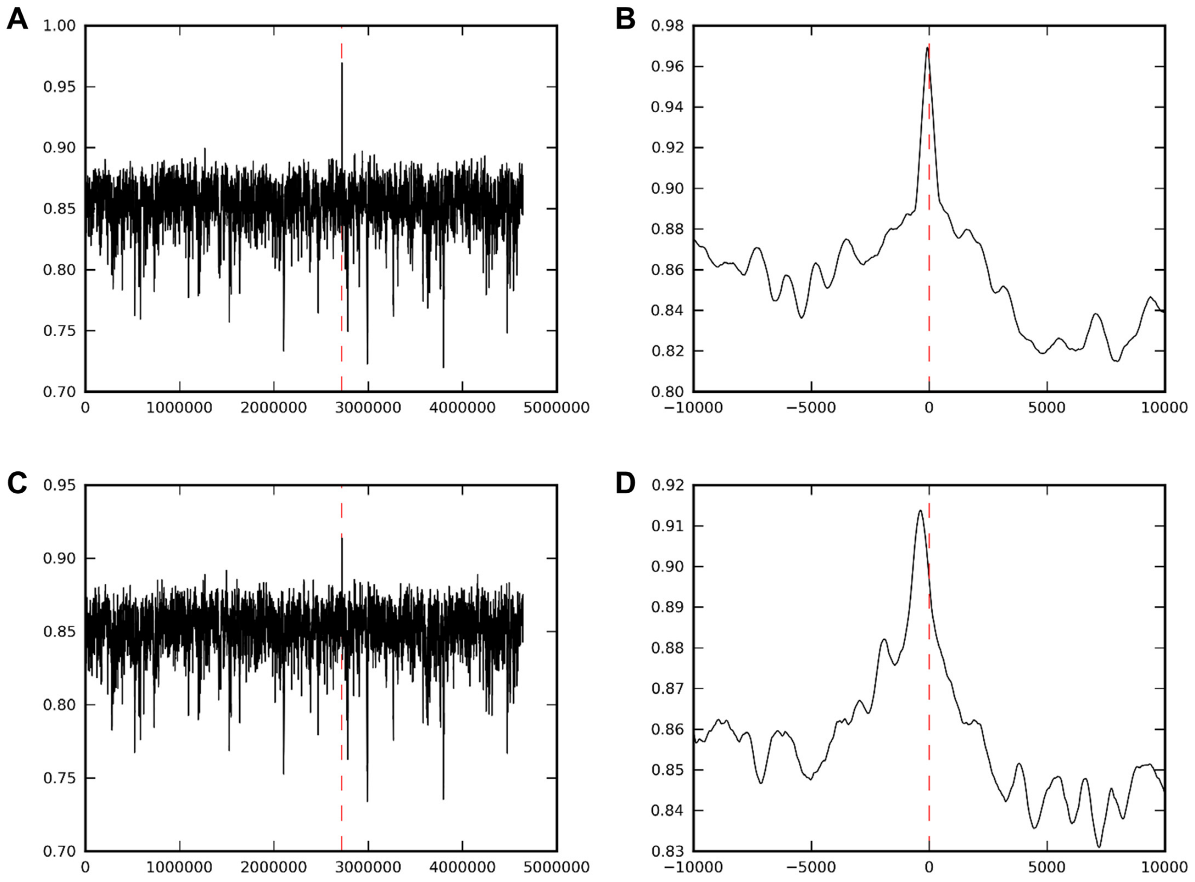


Figure 6. COSINE similarity score for a 5 kb read extracted from E. coli genome with simulated error rates of 15% and 35% and parameter settings $k = 3$, $d = 10$, $w = 500$. Error pattern is insertion = 62%, deletion = 28%, substitution = 10% (typical for PacBio reads (12,16)). Figures on the right are centered at the sequenced position. (**A**, **B**) error rate = 15%. (**C**, **D**) error rate= 35%.

the read sequence and segments of the target sequence detected as highly similar ($t[p - o, p + l + o]$ where t = target sequence, p = a significant peak position, l = read length and o = margin to compensate for the detected peak resolution and read length inaccuracy due to indels). It then computes the optimal path within a banded area along the longest chain of exact matches (default: band_length = 2 × 100). This is a similar approach to SDP step in BLASR. COSINE computes pairs of k-mers and their locations (k = 8–11 depending on the read length) in the read and selected fragments of the reference sequence. To generate the list of maximal exact matches (MEMs) between the read and reference, all consecutive exact matched k-mers between the read and a reference segment are merged; then the longest chain of MEMs is found using dynamic programming. The condition to chain two MEMs is if the difference between the distance of the two in the read and reference is within an acceptable range considering the indel rate. The reference segment with the longest chain of MEMs is selected for the BDP step. Lastly, COSINE reports the alignment if its BDP score is higher than the minimum score threshold (default: 100).

The case of very long reads requires special treatment, specifically when the rates of insertion and deletion are different (for instance, PacBio reads have higher rate of insertion than deletion error). In such cases, absolute value of #insertions – #deletions becomes comparable to the window size as the window moves toward the end of the read (see Supplementary Figure S15). In order to reduce the biases caused by the accumulated indel errors, reads longer than L_t (default: $L_t = 5000$) are split to fragments of length L_t and located separately; significant peak positions of different fragments within expected distance range are then merged back together.

RESULTS

COSINE performance in simulations

Simulation settings. Performance of COSINE is evaluated using *E. coli*, *S. coelicolor* and human simulated datasets with various accuracy levels of 95%, 90%, 85%, 75%, 65% and 55%. Reads are simulated using PBSIM (18) (Supplementary Note S1). COSINE (version 1.00) is run with k-mer sizes of k = 3 and 4 as two sensitivity modes. Reads are skipped by COSINE if the read length is shorter that the window size (w), or if no significant peak is detected or BDP score is less than the minimum threshold. For comparison, BLASR (version 1.3.1) is run with -bestn 1 (reports the best alignment) and also with -maxScore (maximum score threshold) set to a large positive value 10,000 to output alignments of lower accuracy. BWA-MEM (version 0.7.12) is run with (-x ont2d) and a more sensitive setting (-k12 -W12 -r10 -A1 -B1 -O1 -E1 -L0). A BWA-MEM alignment is accepted only if it has alignment quality score not <20 (i.e. MAPQ $\geq$ 20), otherwise the result is counted as a skipped read. This quality filter is imposed otherwise, the false alignment rate may become unacceptably high; for instance, >20% of 5 kb long human reads (with 45% error) are mapped to incorrect locations by the unfiltered BWA-MEM. LAST (version 719) is run with recommended settings for ONT reads (-r 1 -a 1 -b 1 -q1) (19). For all methods, read alignment is marked as correct if the start position of primary alignment or in the case of multiple primary alignments, the one with highest alignment score (AS tag), is within the read length of original simulated location (start alignment position is compensated for clipped sequences). Simulated reads overlapped with unassembled regions of human genome are excluded in the evaluation. Finally, each tool is run with multiple configurations to evaluate the range of sensitivity and computational efficiency.

Comparing aligners performance. Full reports of COSINE, BLASR, BWA-MEM and LAST in these simulations are presented in Supplementary Tables S1–S4. We summarize some of the results in Table 2 (for 5 kb reads) and Table 3 (for 10 kp reads) in order to illustrate the performance of COSINE relative to other methods. With relatively high quality long reads (accuracy levels of 95% and 90%), all tools can achive high alignment rates with recommended settings. As a result, Tables 2 and 3 represent results for each tool from three accuracy levels of 85%, 65% and 55% and from two sensitivity modes, with rows containing results from the high sensitivity mode listed immediately below those from the default or lower sensitivity mode. In the case of a small reference genome (*E. coli*), it is seen from Tables 2 and 3 that there is little difference between COSINE ($k = 4$) and two of its competitors: BWA-MEM (high sensitivity) and LAST. For both read sizes (5 and 10 kb) and all three noise levels (15%, 35%, 45%), each of them achieves >99% mapping accuracy (i.e. <1% skipped or incorrectly mapped reads relative to the total number of reads in the relevant simulation). While COSINE ($k = 3$) suffers a slight loss of sensitivity, it runs 2–4× faster than COSINE ($k = 4$). Compared to the other methods, BLASR performs relatively poorly (>5% skipped or incorrectly mapped), and perplexingly, its performance becomes worse as the read length increases.

In the case of more challenging *S. coelicolor* genome with high GC content (72%), all aligners have slightly lower alignment sensitivity compared to *E. coli* genome. For instance, with reads of 10 kb long from *E. coli* genome, COSINE, BWA-MEM and LAST can achieve 100% correct alignment rate, while in *S. coelicolor* genome, there is a performance drop with lower quality reads (accuracy level 55%). Regarding the high error rate of 45%, LAST (with setting -r 1 -a 1 -b 1 -q2) and COSINE (with setting $k = 4$, $w = 500$ and $d = 50$) achieve the best correctly alignment rates of 96.6% and 96.1%, respectively for 5 kb long reads and 99.5% and 99.4% for 10 kb reads. Although, for reads of higher accuracy, all aligners reach mapping rate of >99% (see Supplementary Tables S1–S4). Furthermore, comparing the mapping accuracy between reads of 5 and 10 kb long, longer reads achieve higher correctly alignment rate as they usually extend to unique sequences next to repetitive and complex regions.

In the case of a large reference genome (human), all methods achieve very high mapping accuracy when the error rate is low (15%). As the error rate increases, the mapping accuracy declines for each method but the decline for COSINE is by far the slowest. For 5 kb reads at 45% error rate, COSINE ($k = 4$) maintains a mapping accuracy of 93.8% (2.3% incorrect, 3.9% skipped) while the other methods suffers either very high rates for incorrectly mapped reads (>14% for both modes of BLASR) or very high rates for skipped reads (>23% for both modes of BWA-MEM and LAST). For 10 kb reads at 45%, the performance of all methods improves but the 98.4% mapping accuracy of COSINE ($k = 4$) (0.8% incorrect, 0.8% skipped) is still substantially better than that of the other methods: 13.9% incorrect for BLASR (high sensitivity), 0.5% incorrect and 8.2% skipped for BWA-MEM (high sensitivity), 3.8% incorrect and 6.4% skipped for LAST (high sensitivity). We note that the performance by COSINE ($k = 4$) in this case would likely be good enough to support downstream analysis such as inferring consensus sequence by multiple alignment.

In this section, we have simulated reads with PBSIM default error pattern of PacBio CLR reads (substitution = 10%, insertion = 60%, deletion = 30%). For comparison, *E. coli* reads are also simulated with equal error types proportion of substitution = 34%, insertion = 33% and deletion = 33% (see Supplementary Table S1). As discussed in subsection Short k-mer count performance, unequal insertions and deletions rates require a choice of larger window sizes ($w = 250$–500) to overcome the relative shifts in alignment offset of matched segments. In contrast, in this case of equal ratio, even with a smaller window size of $w = 100$, COSINE achieves >99.9% correctly alignment rate for reads of average 55% (and above) accuracy level.

Next, we discuss the computational feasibility of the methods. Since this does not seems to be an issue for smaller genomes such as *E. coli*, we will focus on the human simulations. When error rate is low (15%), COSINE has substantially higher computational cost compared to the other methods. For instance, for 5 kb reads, the clock time for COSINE ($k = 4$) is ~50 times of those needed for BWA-MEM (high sensitivity). However, as the error rate increases, the computational cost increases rapidly for the high sensitivity modes of the other methods, but not for COSINE. Thus for 5 kb reads at 45% error, the computational cost for COSINE ($k = 4$) is much more comparable to that of BWA-MEM (high sensitivity): its clock time is 3 times higher. Furthermore, the clock time for BWA-MEM (high sensitivity) increases by 60% when the read length increases from 5 to 10 kbp, while the clock time for COSINE ($k = 4$) remains unchanged. Further examination of the full results in Supplementary Tables S1–S4 reveals that while the computational costs of the other three methods vary wildly depending on the parameter settings, noise level and read length, there is very little variation for COSINE. The highly predictable computational requirement for COSINE makes it easy to plan for distributed computation.

Table 2. COSINE, BLASR, BWA-MEM and LAST alignment performance on simulated datasets from *E. coli* and human genome with different error rates and average read length of 5 kb

Read accuracy	Correctly mapped reads	Incorrectly mapped reads	Skipped reads	Wall clock time	CPU max resident size
20× simulated *E. coli* dataset, average read length of 5 kb					
COSINE ($k = 3$, $d = 50$, $w = 250$, $f = 1.5$)					
85%	18456 (1.000)	2 (0.000)	0 (0.000)	00:01:38	0.286G
65%	18619 (1.000)	3 (0.000)	2 (0.000)	00:01:41	0.289G
55%	18320 (0.987)	18 (0.001)	220 (0.012)	00:01:53	0.290G
COSINE ($k = 4$, $d = 50$, $w = 250$)					
85%	18454 (1.000)	4 (0.000)	0 (0.000)	00:02:18	0.304G
65%	18617 (1.000)	6 (0.000)	1 (0.000)	00:02:23	0.304G
55%	18536 (0.999)	17 (0.001)	5 (0.000)	00:02:29	0.304G
BLASR (-bestn 1 -maxScore 10000)					
85%	18458 (1.000)	0 (0.000)	0 (0.000)	00:01:12	0.528G
65%	18494 (0.993)	130 (0.007)	0 (0.000)	00:01:12	0.511G
55%	17457 (0.941)	1101 (0.059)	0 (0.000)	00:00:59	0.398G
BLASR (-bestn 1 -minMatch 12 -maxLCPLength 13 -maxScore 10000)					
85%	18458 (1.000)	0 (0.000)	0 (0.000)	00:01:25	0.527G
65%	18496 (0.993)	128 (0.007)	0 (0.000)	00:01:16	0.508G
55%	17512 (0.944)	1046 (0.056)	0 (0.000)	00:01:02	0.407G
BWA-MEM (-x ont2d) (MAPQ >= 20)					
85%	18432 (0.999)	0 (0.000)	26 (0.001)	00:01:25	0.564G
65%	18556 (0.996)	1 (0.000)	67 (0.004)	00:02:36	0.661G
55%	15884 (0.856)	8 (0.000)	2666 (0.144)	00:01:39	0.618G
BWA-MEM (-k12 -W12 -r10 -A1 -B1 -O1 -E1 -L0) (MAPQ >= 20)					
85%	18431 (0.999)	1 (0.000)	26 (0.001)	00:01:42	0.561G
65%	18576 (0.997)	0 (0.000)	48 (0.003)	00:03:37	0.779G
55%	18482 (0.996)	3 (0.000)	73 (0.004)	00:06:27	1.616G
LAST (-s 2 -T 0 -a 1 -q 1 -b 1 -r 1)					
85%	18458 (1.000)	0 (0.000)	0 (0.000)	00:01:51	0.459G
65%	18621 (1.000)	0 (0.000)	3 (0.000)	00:01:41	0.464G
55%	18418 (0.992)	2 (0.000)	138 (0.007)	00:01:38	0.472G
0.15× simulated human dataset, average read length of 5 kb					
COSINE ($k = 3$, $d = 200$, $w = 500$, $f = 1.5$)					
85%	85092 (0.995)	420 (0.005)	2 (0.000)	05:29:27	4.133G
65%	84131 (0.982)	1267 (0.015)	299 (0.003)	04:58:28	4.133G
55%	72155 (0.841)	3017 (0.035)	10601 (0.124)	05:12:53	4.133G
COSINE ($k = 4$, $d = 200$, $w = 500$)					
85%	85169 (0.996)	344 (0.004)	1 (0.000)	10:33:43	7.894G
65%	84912 (0.991)	761 (0.009)	24 (0.000)	10:01:56	7.894G
55%	80428 (0.938)	2005 (0.023)	3340 (0.039)	09:55:25	7.895G
BLASR (-bestn 1 -maxScore 10000)					
85%	85378 (0.998)	136 (0.002)	0 (0.000)	00:26:04	19.380G
65%	83999 (0.980)	1698 (0.020)	0 (0.000)	00:23:57	17.233G
55%	50871 (0.593)	34902 (0.407)	0 (0.000)	00:23:36	16.258G
BLASR (-bestn 1 -minMatch 12 -maxLCPLength 13 -maxScore 10000)					
85%	85393 (0.999)	121 (0.001)	0 (0.000)	02:09:00	20.111G
65%	85093 (0.993)	604 (0.007)	0 (0.000)	01:55:36	17.961G
55%	73386 (0.856)	12387 (0.144)	0 (0.000)	01:49:04	16.933G
BWA-MEM (-x ont2d) (MAPQ >= 20)					
85%	84111 (0.984)	37 (0.000)	1366 (0.016)	00:15:40	5.765G
65%	77988 (0.910)	474 (0.006)	7235 (0.084)	00:23:10	5.952G
55%	36106 (0.421)	1209 (0.014)	48458 (0.565)	00:13:25	6.074G
BWA-MEM (-k12 -W12 -r10 -A1 -B1 -O1 -E1 -L0) (MAPQ >= 20)					
85%	84111 (0.984)	39 (0.000)	1364 (0.016)	00:21:32	5.992G
65%	82073 (0.958)	240 (0.003)	3384 (0.039)	02:45:46	18.695G
55%	65726 (0.766)	275 (0.003)	19772 (0.231)	03:17:49	23.571G
LAST (-s 2 -T 0 -a 1 -q 1 -b 1 -r 1)					
85%	-	-	-	-	-
65%	72419 (0.845)	9108 (0.106)	4170 (0.049)	00:30:44	17.965G
55%	32503 (0.379)	29182 (0.340)	24088 (0.281)	00:17:43	16.235G
LAST (-s 2 -T 0 -a 1 -q 1 -b 1 -r 1 -e 120 -m 100)					
85%	-	-	-	-	-
65%	83211 (0.971)	957 (0.011)	1529 (0.018)	05:54:16	26.331G
55%	59333 (0.692)	5907 (0.069)	20533 (0.239)	16:53:24	18.609G

COSINE similarity score computation is done on GPU (Nvidia Titan X, 12GB DDR5) and the GPU memory usage is not accounted in 'CPU max resident size' column. BLASR results are shown with settings ([|-minMatch 12 -maxLCPLength 13] -bestn 1 -maxScore 10000 -sa index_file -nproc 10), BWA-MEM with ([-x ont2d |-k12 -W12 -r10 -A1 -B1 -O1 -E1 -L0] -t10) and LAST with (lastal [|-m 100 -e 120] -s 2 -T 0 -a 1 -q 1 -b 1 -r 1 -P 10 |last-split |maf-convert -n sam). COSINE is run with default parameter settings explained in subsection Alignment procedure (unless otherwise noted) and DP step is run using 10 CPU threads. Blank rows are unsuccessful runs due to exceeding memory limit of 40G.

Table 3. COSINE, BLASR, BWA-MEM and LAST alignment performance on simulated datasets from *E. coli* and human genome with different error rates and average read length of 10 kb

Read accuracy	Correctly mapped reads	Incorrectly mapped reads	Skipped reads	Wall clock time	CPU max resident size
20× simulated *E. coli* dataset, average read length of 10 kb					
COSINE ($k = 3$, $d = 50$, $w = 250$, $f = 1.5$)					
85%	9274 (1.000)	0 (0.000)	0 (0.000)	00:01:31	0.271G
65%	9280 (1.000)	0 (0.000)	0 (0.000)	00:01:33	0.279G
55%	9268 (0.998)	0 (0.000)	16 (0.002)	00:01:48	0.281G
COSINE ($k = 4$, $d = 50$, $w = 250$)					
85%	9274 (1.000)	0 (0.000)	0 (0.000)	00:02:06	0.304G
65%	9280 (1.000)	0 (0.000)	0 (0.000)	00:02:09	0.304G
55%	9284 (1.000)	0 (0.000)	0 (0.000)	00:02:14	0.304G
BLASR (-bestn 1 -maxScore 10000)					
85%	9274 (1.000)	0 (0.000)	0 (0.000)	00:01:51	0.466G
65%	8517 (0.918)	763 (0.082)	0 (0.000)	00:01:49	0.449G
55%	8473 (0.913)	810 (0.087)	1 (0.000)	00:01:28	0.350G
BLASR (-bestn 1 -minMatch 12 -maxLCPLength 13 -maxScore 10000)					
85%	9274 (1.000)	0 (0.000)	0 (0.000)	00:02:24	0.463G
65%	8539 (0.920)	741 (0.080)	0 (0.000)	00:02:01	0.443G
55%	8497 (0.915)	786 (0.085)	1 (0.000)	00:01:39	0.349G
BWA-MEM (-x ont2d) (MAPQ >= 20)					
85%	9274 (1.000)	0 (0.000)	0 (0.000)	00:02:20	0.586G
65%	9280 (1.000)	0 (0.000)	0 (0.000)	00:04:47	0.765G
55%	9084 (0.978)	3 (0.000)	197 (0.021)	00:02:49	0.768G
BWA-MEM (-k12 -W12 -r10 -A1 -B1 -O1 -E1 -L0) (MAPQ >= 20)					
85%	9274 (1.000)	0 (0.000)	0 (0.000)	00:02:44	0.589G
65%	9279 (1.000)	1 (0.000)	0 (0.000)	00:06:18	0.920G
55%	9283 (1.000)	1 (0.000)	0 (0.000)	00:11:40	1.625G
LAST (-s 2 -T 0 -a 1 -q 1 -b 1 -r 1)					
85%	9274 (1.000)	0 (0.000)	0 (0.000)	00:01:53	0.512G
65%	9280 (1.000)	0 (0.000)	0 (0.000)	00:01:45	0.476G
55%	9283 (1.000)	0 (0.000)	1 (0.000)	00:01:40	0.479G
0.15× simulated human dataset, average read length of 10 kbp					
COSINE ($k = 3$, $d = 200$, $w = 500$, $f = 1.5$)					
85%	42712 (0.998)	100 (0.002)	1 (0.000)	05:09:56	4.133G
65%	42712 (0.996)	155 (0.004)	11 (0.000)	04:46:24	4.133G
55%	40462 (0.945)	726 (0.017)	1608 (0.038)	04:41:56	4.133G
COSINE ($k = 4$, $d = 200$, $w = 500$)					
85%	42712 (0.998)	100 (0.002)	1 (0.000)	10:28:28	7.894G
65%	42741 (0.997)	135 (0.003)	2 (0.000)	09:50:19	7.894G
55%	42097 (0.984)	339 (0.008)	360 (0.008)	09:47:59	7.894G
BLASR (-bestn 1 -maxScore 10000)					
85%	42767 (0.999)	46 (0.001)	0 (0.000)	00:25:41	18.614G
65%	41982 (0.979)	896 (0.021)	0 (0.000)	00:24:06	16.773G
55%	32196 (0.752)	10600 (0.248)	0 (0.000)	00:24:18	15.930G
BLASR (-bestn 1 -minMatch 12 -maxLCPLength 13 -maxScore 10000)					
85%	42771 (0.999)	42 (0.001)	0 (0.000)	03:02:56	19.514G
65%	42005 (0.980)	873 (0.020)	0 (0.000)	02:27:51	17.669G
55%	36838 (0.861)	5955 (0.139)	3 (0.000)	02:17:37	16.783G
BWA-MEM (-x ont2d) (MAPQ >= 20)					
85%	42280 (0.988)	48 (0.001)	485 (0.011)	00:20:11	5.813G
65%	41438 (0.966)	227 (0.005)	1213 (0.028)	00:32:57	6.184G
55%	27702 (0.647)	624 (0.015)	14470 (0.338)	00:15:44	6.306G
BWA-MEM (-k12 -W12 -r10 -A1 -B1 -O1 -E1 -L0) (MAPQ >= 20)					
85%	42279 (0.988)	52 (0.001)	482 (0.011)	00:24:49	5.837G
65%	41773 (0.974)	185 (0.004)	920 (0.021)	03:59:17	18.054G
55%	39078 (0.913)	227 (0.005)	3491 (0.082)	05:11:33	25.116G
LAST (-s 2 -T 0 -a 1 -q 1 -b 1 -r 1)					
85%	Job did not finish successfully in 0:51:43, 50.409G				
65%	41554 (0.969)	1067 (0.025)	257 (0.006)	00:33:42	18.405G
55%	26155 (0.611)	12785 (0.299)	3856 (0.090)	00:19:21	16.586G
LAST (-s 2 -T 0 -a 1 -q 1 -b 1 -r 1 -e 120 -m 100)					
85%	-	-	-	-	-
65%	42693 (0.996)	33 (0.001)	152 (0.004)	23:43:59	29.085G
55%	38400 (0.897)	1642 (0.038)	2754 (0.064)	20:56:51	20.211G

COSINE similarity score computation is done on GPU (Nvidia Titan X, 12GB DDR5) and the GPU memory usage is not accounted in 'CPU max resident size' column. BLASR results are shown with settings ([|-minMatch 12 -maxLCPLength 13] -bestn 1 -maxScore 10000 -sa index_file -nproc 10), BWA-MEM with ([-x ont2d |-k12 -W12 -r10 -A1 -B1 -O1 -E1 -L0] -t10) and LAST with (lastal [|-m 100 -e 120] -s 2 -T 0 -a 1 -q 1 -b 1 -r 1 -P 10 |last-split |maf-convert -n sam). COSINE is run with default parameter settings explained in subsection Alignment procedure (unless otherwise noted) and DP step is run using 10 CPU threads. Blank rows are unsuccessful runs due to exceeding memory limit of 40G.

Table 4. COSINE, BLASR, BWA-MEM and LAST alignment performance on ONT real dataset downloaded from (http://bit.ly/loman006)

Dataset			Correctly mapped reads	Incorrectly mapped reads	Skipped reads	Correctly mapped bases (Mbp)	Incorrectly mapped bases (Mbp)	Wall clock time	CPU max resident size
MAP006–1	pass	COSINE	49289	33	605	451.97M	0.17M	00:26:08	0.687G
		BLASR	48048	1869	10	414.55M	0.50M	00:16:40	1.452G
		BWA-MEM	48733	9	1185	437.43M	0.09M	00:11:37	0.868G
		LAST	49442	18	467	424.03M	0.15M	00:09:21	0.802G
	fail	COSINE	9936	132	932	79.42M	0.76M	00:16:40	0.728G
		BLASR	8643	2341	16	42.01M	0.97M	00:11:15	0.862G
		BWA-MEM	8869	116	2015	67.36M	0.67M	00:05:20	0.996G
		LAST	9937	121	942	71.12M	0.65M	00:04:18	0.720G
MAP006–2	pass	COSINE	28220	10	228	256.38M	0.04M	00:14:40	0.685G
		BLASR	27136	1306	16	230.16M	0.30M	00:06:27	0.868G
		BWA-MEM	27979	2	477	249.69M	0.01M	00:06:31	0.791G
		LAST	28259	8	191	242.30M	0.05M	00:05:16	0.765G
	fail	COSINE	9443	97	772	78.11M	0.53M	00:12:04	0.730G
		BLASR	7965	2331	16	37.29M	0.79M	00:13:53	0.682G
		BWA-MEM	8497	76	1739	67.43M	0.45M	00:04:09	0.769G
		LAST	9422	81	809	70.35M	0.44M	00:03:12	0.723G
MAP006-PCR-1	pass	COSINE	39917	19	442	255.67M	0.08M	00:15:03	0.684G
		BLASR	39579	798	1	239.66M	0.19M	00:04:31	0.942G
		BWA-MEM	39515	3	860	250.02M	0.01M	00:04:48	0.640G
		LAST	40143	4	231	246.02M	0.01M	00:05:07	0.624G
	fail	COSINE	6817	50	726	40.63M	0.13M	00:06:33	0.702G
		BLASR	6328	1264	1	22.27M	0.28M	00:01:59	0.439G
		BWA-MEM	5998	34	1561	34.18M	0.10M	00:01:41	0.525G
		LAST	6846	35	712	37.00M	0.10M	00:01:46	0.490G
MAP006-PCR-2	pass	COSINE	79159	62	1094	505.02M	0.21M	00:30:33	0.685G
		BLASR	77274	3041	0	457.41M	0.60M	00:08:47	1.724G
		BWA-MEM	78213	14	2088	493.31M	0.05M	00:09:58	0.669G
		LAST	79269	18	1028	485.32M	0.07M	00:10:55	0.636G
	fail	COSINE	22842	90	2379	135.72M	0.23M	00:20:00	0.721G
		BLASR	19954	5353	4	68.28M	0.93M	00:11:14	1.038G
		BWA-MEM	19822	55	5434	116.04M	0.16M	00:04:57	0.589G
		LAST	22214	64	3033	124.67M	0.18M	00:05:00	0.563G

Alignment accuracies are computed based on '1D' reads from 'known' dataset. COSINE results are shown with setting (-k 4 –window_size 100 –window_shift 10 –min_sig_score 2. –min_dp_score 50.), BLASR with setting (-bestn 1 -minMatch 12 -maxLCPLength 13 -nCandidates 10 -maxScore 10000 -nproc 10 -sa index_file). BWA-MEM results are shown with setting (-k12 -W20 -r10 -A1 -B1 -O1 -E1 -L0 -t10) and filtering alignments with mapping quality less than 20 and LAST with setting (lastal -s 2 -T 0 -a 1 -q 1 -b 1 -r 1 -P 10 |last-split |maf-convert -n sam). Full reports on four aligners with multiple parameter settings are in Supplementary Table S5.

Evaluation on real data

In order to assess the performance of COSINE with real ONT data, *E. coli* Oxford Nanopore sequencing data (R7.3 chemistry, SQR-MAP006 protocol) is downloaded from (http://bit.ly/loman006). Metrichor software categorizes based-called data to two 'pass' and 'fail' categories based on their quality. We used poretools (version 0.5.1) (20) to generate '2D' and '1D' (template and complement reads) fasta files from basecalled fast5 data. As the sequencing location for real data is unknown, we estimated true sequencing locations for '1D' reads using the higher quality '2D' reads. '2D' reads are obtained from the consensus sequence of both template and complement strands, if available. We first aligned '2D' reads using all four aligners COSINE, BLASR, BWA-MEM and LAST to the reference. If all tools aligned the read to the same region, then the read and its alignment position (average of individual tool start positions), are added to the 'known' dataset. 'Known' dataset is the set of '2D' reads that we estimated their sequencing location as explained above. '1D' reads that have their respective '2D' read in the 'known' dataset are used in evaluating tools mapping accuracy (Supplementary Note S2). The performance evaluation is done on four datasets from different runs (MAP-006-1, MAP-006-2, MAP-006-PCR-1, MAP-006-PCR-2) with average read length of $\gtrsim$8 and $\gtrsim$6 kbps for runs without and with PCR step, respectively (Supplementary Table S8). Supplementary Table S5 report each tool alignment performance with different settings. Table 4 has the summary of alignment statistics from all four tools. In general, BLASR (Supplementary Table S5) had relatively lower mapping accuracy compared to other tools for noisy '1D' reads and COSINE and LAST have comparable alignment rate in both 'pass' and 'fail' categories. Overall, the comparisons based on real ONT reads is largely consistent with those from simulated *E. coli* data sets. Finally, we note that as COSINE is currently designed as a global aligner which evaluates the similarity of the whole read fragment against the target, it is expected to miss partially mappable and short sequences. In this regard, running COSINE with $L_t = 3000$ (instead of default 5000) increases its correct alignment ~2% for noisy '1D' reads in 'fail' categories. This is due to the computation of similarity scores over shorter fragments of the read which reduces the noise from unmappable segments (Supplementary Table S5).

Cluster setup

All runs with BLASR, BWA-MEM and LAST are done with 10 CPU threads and 40G allocated memory on Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20 GHz. COSINE similarity score computation is implemented using NVIDIA cuFFT library. All COSINE runs are on Nvidia Titan X 12GB DDR5 and Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40 GHz. COSINE DP step is run with 10 CPU threads. The runtime for COSINE similarity score computation, which mainly relies on GPU, is reported in parentheses in Supplementary Tables S1, S5 and S6 (see 'wallclock time' column).

Alignment runtimes are reported with precomputed reference index files for all aligners. Indexing step for COISNE (i.e. computation of FFT blocks of reference sequence) is ≤2 min and ≤5 min for *E. coli* and human genome, faster compared to other aligners.

DISCUSSION

Herein, we present a method (COSINE) based on a new approach of comparing short *k*-mer distributions to align long sequences with high variations or errors including insertions and deletions to the target. From analyses presented in Results section, COSINE is a robust method and has a predictable computational resource usage for mapping reads from *E. coli* genome, *S. coelicolor* genome with high GC conect or complex human genome. In the case of repetitive or high GC/AT content sequences, there is no systematic limitation found compared to other aligners.

The advantage of COSINE is to maintain a high alignment accuracy in a wide range of error rates and genome sizes with minimal tuning. Currently, TGS reads reach multi-kbp long but they have relativity lower accuracy. While other available aligners supporting TGS technologies have acceptable performance given current reads error profile, ONT reads in low quality category still contain considerable portion of sequencing data (10) which are mostly not considered in downstream analysis due to its high error rate. In these cases, COSINE could be used as a standalone aligner or in addition to other mappers to retrieve skipped reads. Furthermore, high alignment accuracy becomes more important in applications such as assembly. In assembly, mapping error rate is about twice the original raw reads and high sensitivity and precision in detecting overlaps among noisy reads is a crucial step in this process. We intend to further apply this technique in assembly applications.

AVAILABILITY

COSINE software and a reference manual is available from https://github.com/wonglab/cosine

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

ACKNOWLEDGEMENTS

The authors would like to thank Chen-Yu Tseng and Tung-Yu Wu for providing assistance with GPU cluster and NVIDIA CUDA Toolkit.

FUNDING

National Institutes of Health [R01HG007834]. Funding for open access charge: Foundation for the National Institutes of Health [R01HG007834].
Conflict of interest statement. None declared.

REFERENCES

1. Zhang,J., Chiodini,R., Badr,A. and Zhangd,G. (2011) The impact of next-generation sequencing on genomics. *J. Genet. Genomics*, **38**, 95–109.
2. Liu,L., Li,Y., Li,L., Hu,N., He,Y., Pong,R., Lin,D., Lu,L. and Law,M. (2012) Comparison of next-generation sequencing systems. *J. Biomed. Biotechnol.*, **2012**, doi:10.1155/2012/251364.
3. Koboldt,D.C., Steinberg,K.M., Larson,D.E., Wilson,R.K. and Mardis,E.R. (2013) The next-generation sequencing revolution and its impact on genomics. *Cell*, **155**, 27–38.
4. Buermans,H.P. and den Dunnen,J.T. (2014) Next generation sequencing technology: advances and applications. *Biochim. Biophys. Acta.*, **1842**, 1932–1941.
5. LeBlanc,V.G. and Marra,M.A. (2015) Next-generation sequencing approaches in cancer: where have they brought us and where will they take us? *Cancers (Basel)*, **7**, 1925–1958.
6. Shendure,J. and Ji,H. (2008) Next-generation DNA sequencing. *Nat. Biotechnol.*, **26**, 1135–1145.
7. Laehnemann,D., Borkhardt,A. and McHardy,A.C. (2016) Denoising DNA deep sequencing data-high-throughput sequencing errors and their correction. *Brief Bioinform.*, **17**, 154–179.
8. Goodwin,S., Gurtowski,J., Ethe-Sayers,S., Deshpande,P., Schatz,M.C. and McCombie,W.R. (2016) Oxford Nanopore Sequencing and de novo assembly of a eukaryotic genome. *Genome Res.*, **26**, 342–350.
9. Lavera,T., Harrisona,J., O'Neilla,P.A., Moorea,K., Farbosa,A., Paszkiewicza,K. and Studholmea,D.J. (2015) Assessing the performance of the Oxford Nanopore Technologies MinION. *Biomol. Detect. Quant.*, **3**, 1–8.
10. Ip,C.L.C., Loose,M., Tyson,J.R., de Cesare,M., Brown,B.L., Jain,M., Leggett,R.M., Eccles,D.A., Zalunin,V., Urban,J.M. *et al.* (2015) MinION analysis and reference consortium: phase 1 data release and analysis. *F1000Res.*, **4**, 1075.
11. Kiełbasa,S.M., Wan,R., Sato,K., Horton,P. and Frith,M.C. (2011) Adaptive seeds tame genomic sequence comparison. *Genome Res.*, **21**, 487–493.
12. Chaisson,M.J. and Tesler,G. (2012) Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. *BMC Bioinformatics*, **13**, 238.

13. Katoh,K., Misawa,K. and Miyata,T. (2002) MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res.*, **30**, 3059–3066.
14. Rajasekaran,S., Jin,X. and Spouge,J.L. (2002) The efficient computation of position-specific match scores with the Fast Fourier Transform. *J. Comput. Biol.*, **9**, 23–33.
15. Rockwood,A.L., Crockett,D.K., Oliphant,J.R. and Elenitoba-Johnson,K.S.J. (2005) Sequence alignment by cross-correlation. *J. Biomol. Tech.*, **16**, 453–458.
16. Magi,A., Giusti,B. and Tattini,L. (2016) Characterization of MinION nanopore data for resequencing analyses. *Brief Bioinform.*, doi:10.1093/bib/bbw077.
17. Felsenstein,J., Sawyer,S. and Kochin,R. (1982) An efficient method for matching nucleic acid sequences. *Nucleic Acids Res.*, **10**, 133–139.
18. Ono,Y., Asai,K. and Hamada,M. (2013) PBSIM: PacBio reads simulator—toward accurate genome assembly. *Bioinformatics*, **29**, 119–121.
19. Quick,J., Quinlan,A.R. and Loman,N.J. (2014) A reference bacterial genome dataset generated on the MinION™ portable single-molecule nanopore sequencer. *Gigascience*, **3**, 22.
20. Loman,N.J. and Quinlan,A.R. (2014) Poretools: a toolkit for analyzing nanopore sequence data. *Bioinformatics*, **30**, 3399–3401.