Error correcting optical mapping data

Mukherjee, Kingshuk

Oxford University Press
2018-05-25

Mukherjee, K, Washimkar, D, Muggli, M D, Salmela, L & Boucher, C 2018, 'Error correcting optical mapping data', GigaScience, vol. 7, no. 6, pp. 1-10. https://doi.org/10.1093/gigascience/giy061

http://hdl.handle.net/10138/247093
10.1093/gigascience/giy061

GigaScience, 7, 2018, 1–10

doi: 10.1093/gigascience/giy061
Advance Access Publication Date: 25 May 2018
Technical Note

TECHNICAL NOTE

Error correcting optical mapping data

Kingshuk Mukherjee [1,*], Darshan Washimkar[2], Martin D. Muggli[2], Leena Salmela[3] and Christina Boucher[1,*]

[1]Department of Computer and Information Science and Engineering, University of Florida, Gainesville, [2]Department of Computer Science, Colorado State University, Fort Collins and [3]Department of Computer Science, Helsinki Institute for Information Technology HIIT, University of Helsinki

*Correspondence address. Kingshuk Mukherjee.432 Newell Dr, E301, Gainesville, FL 32611. E-mail: kingdgp@ufl.edu http://orcid.org/0000-0002-1647-8741; Christina Boucher.432 Newell Dr, E301, Gainesville, FL 32611. E-mail: cboucher@cise.ufl.edu

Abstract

Optical mapping is a unique system that is capable of producing high-resolution, high-throughput genomic map data that gives information about the structure of a genome . Recently it has been used for scaffolding contigs and for assembly validation for large-scale sequencing projects, including the maize, goat, and *Amborella* genomes. However, a major impediment in the use of this data is the variety and quantity of errors in the raw optical mapping data, which are called Rmaps. The challenges associated with using Rmap data are analogous to dealing with insertions and deletions in the alignment of long reads. Moreover, they are arguably harder to tackle since the data are numerical and susceptible to inaccuracy. We develop cOMet to error correct Rmap data, which to the best of our knowledge is the only optical mapping error correction method. Our experimental results demonstrate that cOMet has high prevision and corrects 82.49% of insertion errors and 77.38% of deletion errors in Rmap data generated from the *Escherichia coli* K-12 reference genome. Out of the deletion errors corrected, 98.26% are true errors. Similarly, out of the insertion errors corrected, 82.19% are true errors. It also successfully scales to large genomes, improving the quality of 78% and 99% of the Rmaps in the plum and goat genomes, respectively. Last, we show the utility of error correction by demonstrating how it improves the assembly of Rmap data. Error corrected Rmap data results in an assembly that is more contiguous and covers a larger fraction of the genome.

Keywords: optical mapping; error correction

Introduction

In 1993 Schwartz et al. developed optical mapping, a system for creating an ordered, genome-wide, high-resolution restriction map of a given organism's genome [1]. Since this initial development, genome-wide optical maps have found numerous applications including discovering structural variations and rearrangements [5], scaffolding and validating contigs for several large sequencing projects [4,3, 6], and detecting misassembled regions in draft genomes [7]. Thus, optical mapping has assisted in the assembly of a variety of species including various prokaryote species [8, 9, 10], rice [11], maize [2], mouse [12], goat [3], parrot [6], and *Amborella trichopoda* [4]. The raw optical mapping data are generated by a biological experiment in which large DNA molecules cling to the surface of a microscope slide using electrostatic charge and are digested with one or more restriction enzymes. The restriction enzymes cut the DNA molecule at occurrences of the enzyme's recognition sequence, forming a number of DNA fragments. The fragments formed by digestion are painted with a fluorescent dye to allow visibility under laser light and a CCD (Charged Coupled device) camera. Computer vision algorithms then estimate fragment length from consolidated intensity of fluorescent dye and apparent distance between fragment ends.

The resulting data from an experiment are in the form of an ordered series of fragment lengths [13]. The data for each single molecule produced by the system is called an *Rmap*. Rmap data have a number of errors due to the experimental conditions

Received: 22 June 2017; **Revised:** 4 December 2017; **Accepted:** 16 May 2018

and system limitations. In an optical mapping experiment, it is unlikely to achieve perfectly uniform fluorescent staining. This leads to an erroneous estimation of fragment sizes. Also, restriction enzymes often fail to digest all occurrences of their recognition sequence across the DNA molecule. This manifests as missing restriction sites. Additionally, due to the fragile nature of DNA, additional breaks can incorrectly appear as restriction sites. Last, the limitations of the imaging component of the optical mapping system and the propensity for the DNA to ball up at the ends introduces more sizing error for smaller fragments. Interested readers will find more details about the causes of these errors in Valouev et al. [14] and Li et al. [15]. Because of all these experimental conditions, Rmap data generated through optical mapping experiment have insertion (added cut sites) and deletion (missed cut sites) errors along with fragment sizing errors.

In most applications of optical map data, the Rmaps need to be assembled into a genome-wide optical map. This is because the single molecule maps need redundant sampling to overcome the presence of the aforementioned errors and because single molecule maps only span on the order of 500 Kbp [14]. The first step of this assembly process involves finding pairwise alignments among the Rmaps. In order to accomplish this, the challenge of dealing with missing fragment sizes has to be overcome. This challenge is analogous to dealing with insertions and deletions in the alignment of long reads [16]. In fact, it is arguably harder since the data are numerical. At present, the only nonproprietary algorithmic method for pairwise alignment of Rmaps is the dynamic programming-based method of Valouev et al. [14], which runs in $O(\alpha \times \beta)$ time where α and β are the number of fragments in the two Rmaps being aligned. To align an optical map dataset containing n Rmaps, the complexity becomes $O(n^2 \times \ell^2)$ where ℓ is the average size of an Rmap.

This method is inherently computationally intensive. However, if the error rate of the data could be improved, then non-dynamic programming-based methods that are orders of magnitude faster such as Twin [17], OMBlast [18], and Maligner [19] could be used for alignment. This would greatly improve the time required to assemble Rmap data. Thus, we present cOMet in order to address this need. To the best of our knowledge, it is the first Rmap error correction method. Our experimental results demonstrate that cOMet has high precision and corrects 82.49% of insertion errors and 77.38% of deletion errors in Rmap data generated from the *Escherichia coli* K-12 reference genome. Out of the deletion errors corrected, 98.26% are true errors. Similarly, out of the insertion errors corrected, 82.19% are true errors. Furthermore, we show that the assembly of Rmaps is more contiguous and covers a larger fraction of the genome if the Rmaps are first error corrected. It also successfully scales to large genomes, improving the quality of 78% and 99% of the Rmaps in the plum and goat genome, respectively.

Background

From a computer science perspective, optical mapping can be seen as a process that takes in two strings, a nucleotide sequence $S_i[1, n]$ and a restriction sequence $B[1, b]$, and produces an array (string) of integers $R_i[1, m]$. The array R_i is an Rmap corresponding to S_i and contains the string-lengths between cuts produced by B on S_i. Formally, R_i is defined as follows: $R_i[j] = y - x$ where y represents the location (starting index) of j^{th} occurrence of B in S_i and x represents the location of $(j-1)^{th}$ occurrence of B in S_i and $R_i[1] = y - 1$ and $R_i[m] = n - x$. For example, say we have $B = act$ and $S_i = atacttactggactactaaact$. The locations of B in S_i are as follows: 3,7,12,15,20. Then, R_i will be represented as R_i= 2, 4, 5, 3, 5, 2. The size of an Rmap denotes the number of fragments in that Rmap. Therefore, the size of R_i is 6.

We note that millions of Rmaps are produced for a single genome since optical mapping is performed on many cells of the organism and each cell provides thousands of Rmaps. The Rmaps can be assembled to produce a genome-wide optical map. This is analogous to next-generation shotgun sequencing where Rmaps are analogous to reads and a genome-wide optical map is analogous to the assembled whole genome.

There are three types of errors that can occur in optical mapping: (1) missing cut sites, which are caused by an enzyme not cleaving at a specific site; (2) additional cut sites, which can occur due to random DNA breakage; and (3) inaccuracy in the fragment size due to the inability of the system to accurately estimate the fragment size. Continuing again with the example above, a more representative example Rmap would include these errors, such as $R_i' = 7, 6, 3, 4$.

The error rates of optical maps depend on the platform used for generating the maps. Li et al. [15] recently studied the error rates of optical maps produced by the Irys system from BioNano Genomics. According to their study, a missing cut site type of error, i.e., error type (1),happens when a restriction site is incompletely digested by the enzyme and causes two flanking fragments to merge into one large fragment. The probability of complete digestion of a restriction site can be modeled as a Bernoulli trial whose probability of success is a function of the size of the two flanking fragments. Additional cut sites, i.e., error type (2),results from random breaks of the DNA molecule. The number of false cuts per unit length of DNA follows a Poisson distribution. The inaccuracy of the fragment sizes, i.e., error type (3), is modeled using a Laplace distribution. If the observed and actual size of a fragment are o_k and r_k, respectively, then the sizing error is defined as $s_k = o_k/r_k$ and

$$s_k \sim Laplace(\mu, \beta)$$

where μ and β, the parameters of the laplace distribution, are functions of r_k. In practice, when aligning a pair of Rmaps, one should allow for twice the error rate of a single Rmap since each Rmap will deviate from the genomic map by the above parameters.

Valouev et al. [14] provides a dynamic programming algorithm for pairwise alignment, which generates a score for every possible alignment between two Rmaps and returns the alignment that achieves the highest score, which is referred to as the S-score. It is computed within a standard dynamic programming framework, similar to Smith-Waterman alignment [20]. The scoring function is based on a probabilistic model built on the following assumptions: the fragment sizes follow an exponential distribution, the restriction sites follow an independent Bernoulli process, the number of false cuts in a given genomic length is a Poisson process, and the sizing error follows a normal distribution with mean zero and variance following a linear function of the true size. Last, a different sizing error function is used for fragments less than 4 kbp in length since they do not converge to the defined normal distribution. The score of an alignment is calculated as the sum of two functions: one function that estimates and scores the sizing error and a second that predicts and scores the presence of additional and/or missing cut sites between the fragments. The S-score will be used later to evaluate the error correction process.

Methods

Given a set of n Rmaps $R = \{R_1, .., R_n\}$, our method aims to detect and correct all errors in R by considering each $R_i \in R$ and finding a set of Rmaps that originate from the same part of the genome as R_i. This step is performed heuristically in order to avoid aligning every pair of Rmaps in R.

Preprocessing

Our first step is to remove the first and last fragments from each Rmap in R. These fragments have one of their edges sheared by artifacts of the DNA prep process (preceding the optical mapping process) and not by restriction enzymes. Unless removed, they can misguide alignment between two Rmaps during the error correction process. In addition, short Rmaps, i.e., those that have fewer than 10 fragments, are removed at this stage since any Rmap that contains fewer than 10 fragments is typically deemed too small for analysis even in consensus maps [21]. Next, the data are quantized so that a given genomic fragment is represented by the same value across multiple Rmaps despite the noise. Our quantization method assigns a unique value to a range of fragment sizes by dividing each fragment size by a fixed integer, denoted as b, and rounding to the nearest integer. For example, if an Rmap $R_i = \{36, 13, 15, 20, 16, 5, 21, 17\}$ is quantized using b= 3, then the quantized Rmap will be $R_i^{quantized} = \{12, 4, 5, 7, 5, 2, 7, 6\}$. Say another Rmap, $R_j = \{17, 23, 34, 12, 14, 21, 14, 5\}$ has overlap with R_i; however, due to noise in the data, this relation is not apparent. By quantizing R_j using the same b= 3, we get $R_j^{quantized} = \{6, 8, 11, 4, 5, 7, 5, 1\}$. This allows us to uncover a region (in this case, $\{4, 5, 7, 5\}$) that is common to both the Rmaps. It should be noted that, in some cases, a fragment may have different values across two Rmaps even after quantization (e.g., the fragment values 36 from R_i and 34 from R_j are quantized to 12 and 11, respectively). The quantized data are used to find the set of *related* Rmaps, as explained in the next section.

The setting of parameter b depends on the amount of sizing error in the optical map data. With zero sizing error, b can be set at 1. As sizing error increases, the value of b is increased accordingly. If the value of b is too small, we are not be able to uncover relations between overlapping Rmaps. If the value is too large, then unrelated Rmaps have common regions in their quantized states, which makes them appear related. Considering the error rate of optical maps from BioNano genomics, the default value of $b = 4000$.

Finding related Rmaps

We refer to two Rmaps as *related* if their corresponding error-free Rmaps originate from overlapping regions of the genome. Next, we define a k-mer as a string of k consecutive fragments from a (quantized) Rmap. For example, if we have the Rmap $R = \{3, 3, 5, 2, 6, 5, 5, 1\}$ and $k = 4$, then the following k-mers can be extracted from R: (3,3,5,2), (3,5,2,6), (5,2,6,5), (2,6,5,5),and (6,5,5,1). In order to avoid aligning all pairs of Rmaps to find the related Rmaps, we use the number of common k-mers to discriminate between pairs of Rmaps that are related and those that are not. To accomplish this efficiently, we first extract all unique k-mers in each quantized Rmap and construct a hash table storing each unique k-mer as a key and the list of Rmaps containing an occurrence of that k-mer as the value. We call this the k-mer index. Next, we consider each R_i in R and use the k-mer index to identify the set of Rmaps that have m or more k-mers in common with R_i. Unfortunately, although this set contains all related Rmaps, it also likely contains Rmaps that are not related to R_i. Therefore, we filter this set of Rmaps using a simple heuristic that tries to match each Rmap in this set with R_i in order to ascertain if it is related to R_i. The heuristic traverses through two Rmaps (R_i and one Rmap from the set, say R_j) attempting to match subsets of the fragments from each until it either reaches the end of one Rmap or it fails to match the fragments. We start the traversal from the first matching k-mer between R_i and R_j. We denote the position of the next fragment to be matched in R_i and R_j as x and y, respectively, and assume that each fragment prior to these positions is matched. Next, we consider all combinations of matching the fragments at positions x, x+ 1, and x + 2 of R_i with fragments at positions y, y+ 1, and y + 2 of R_j. We evaluate the cost of each combination based on the difference in the total size of fragments from R_i and R_j. That is $\forall \alpha, \beta = [0, 2]$,

$$cost(x + \alpha, y + \beta) = \left| \sum_{g=x}^{x+\alpha} R_i[g] - \sum_{h=y}^{y+\beta} R_j[h] \right|$$

where $R_i[g]$ and $R_j[h]$ denote the g-th and h-th fragments of R_i and R_j, respectively. We select the combination with the least cost; if there exists a tie, we select the match that has the least number of added or missing cut sites (i.e., the combination with the least value of $\alpha + \beta$). If this selected match leads to a cost that is greater than a specified threshold (which was set to 25% of the larger-sized fragment in practice), then we conclude that there is not a match at these positions and return that R_i and R_j are unrelated. Otherwise, we increment x and y accordingly and move onto the next fragments. If this heuristic continues until the last fragment of either R_i or R_j is reached, then we return that R_i and R_j are related. Using this heuristic, we filter out the Rmaps that were deemed to be related based on the number of k-mers in common with R_i but are in fact unrelated to R_i.

The setting of parameters k and m are correlated. If the value of k is increased, that makes the k-mers more specific, hence, the value of m is lowered. On the other hand, if the value of k is reduced, then we increase the value of m. The value of k is increased when there are fewer insertion and deletion errors and decreased otherwise. The default values are $k = 4$ and $m = 1$.

Rmap alignment

Next, for each R_i in R, we use the alignment method of Valouev et al. [14] to find the S-score of all pairwise alignments between R_i and each Rmap in its set of related Rmaps. The Rmaps that have an alignment score, i.e., S-score less than a defined threshold (which we denote as S_t), are removed from the set of related Rmaps, and the alignments of the remaining Rmaps are stored in a multiple alignment grid, denoted as A_i. This grid is a two-dimensional array of integer pairs, where the number of rows is equal to the number of remaining Rmaps in the set of related Rmaps of R_i and the number of columns is equal to the number of fragments in R_i. An element of this array, $A_i[j, k]$, stores an integer pair in the form of (x, y) representing that x fragments of R_i (which includes the k-th fragment of R_i) matches to y fragments of R_j in the optimal alignment between R_i and R_j. Figure1 illustrates an example of A_i. The first fragment of R_i does not match with any fragment of R_j and therefore (0,0) is stored at this position. Fragments 2, 5, 6, 8, and 9 of R_i each matches with one fragment of R_j, e.g., 1, 3, 4, 7, and 8, respectively. To represent these matches, we store a (1,1) in the 2nd, 5th, 6th, 8th, and 9th columns of row j. Fragments 3 and 4 of R_i match with

	1st	2nd	3rd	4th	5th	6th	7th		8th	9th	10th	11th	12th	13th	
R_i	1.474	3.625	2.092	2.164	8.424	2.331	24.824		7.267	2.954	12.578	2.358	8.955	22.943	
R_j		3.331	4.464		8.287	2.481	10.314	13.391	7.711	3.143	8.448	5.921	13.795	4.143	6.119

A_i	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	11th	12th	13th
:	:	:	:	:	:	:	:	:	:	:	:	:	:
j	(0,0)	(1,1)	(2,1)	(2,1)	(1,1)	(1,1)	(1,2)	(1,1)	(1,1)	(2,2)	(2,2)	(2,3)	(2,3)
:	:	:	:	:	:	:	:	:	:	:	:	:	:

Figure 1: An alignment between R_i and R_j as given by Valouev et al. [14] and its corresponding entry in the multiple alignment grid A_i. Each column of A_i represents one fragment from R_i, and each row represents one Rmap from its set of related Rmaps. The fragment sizes are in Kbp.

one fragment of R_j, i.e., the 2nd fragment. To represent this, we store (2,1) in $A_i[j, 3]$ and $A_i[j, 4]$. Fragment 7 of R_i matches with two fragments of R_j, i.e., the 5th and 6th fragments. To represent this, we store (1,2) in $A_i[j, 7]$. Fragments 10 and 11 of R_i match with two fragments of R_j, i.e., the 9th and 10th fragments. To represent this, we store (2,2) in positions $A_i[j, 10]$ and $A_i[j, 11]$. Finally, fragments 12 and 13 match with three fragments of R_j, i.e., fragments 11, 12, and 13. In this case, we store (2,3) in positions $A_i[j, 12]$ and $A_i[j, 13]$.

The setting of parameter S_t controls the number of Rmaps that are included in the multiple-alignment grid of an Rmap. If we increase the value of S_t, fewer Rmaps will be added to the grid, but the ones included will be of higher quality (i.e., have greater overlap with the Rmap under consideration). The default value for the parameter $S_t = 8$. We show in the Experiment section how we select this value.

Error correcting using the consensus

The multiple alignment grid is used to find the consensus grid, denoted as C_i, for Rmap R_i. The grid C_i is a one-dimensional array of integer pairs with size equal to the number of fragments in R_i. The grid is constructed for each R_i in R by iterating through each column of A_i and finding the most frequent integer pair, breaking ties arbitrarily. The most frequent integer pair is stored at each position of C_i if the frequency is above a given threshold d; otherwise, (0,0) is stored. Figure2 illustrates the construction of a consensus grid from an alignment grid. The type of error in each fragment of R_i can be identified using $C_i[k] = (x, y)$ as follows: if x and y are equal, then a sizing error occurs at the k-th fragment of R_i, otherwise, if x is greater than y, then an additional cut site exists, and, last, if x is less than y, then a missing cut site exists. Next, we use C_i and A_i to correct these errors in R_i. For each fragment of R_i, we consider the consensus stored at the corresponding position of C_i, identify the positions in the corresponding column of A_i that are equal to it, and replace the fragment of R_i with the mean total fragment size computed using the values at those positions in A_i. If C_i is equal to (0,0) at any position, then the fragment at that position in R_i remains unchanged since it implies that there is no definitive result about the type of error in that position. In addition, if consecutive positions in C_i are discordant, then the fragments in those positions in R_i also remains unchanged. For example, if there is a (2,1) consensus at some position of C_i, then we expect the preceding or successive position to also have a (2,1) consensus. However, if this is not the case, then we do not error correct those fragments since the consensus is discordant at those positions. Figure2 shows this error correction. As it is illustrated, to error correct the second fragment of R_i, we compute the average of the matched fragments from related Rmaps 2, 3, 4, 5, and 6 and replace the second fragment of R_i with that value as shown in Fig. 2. Similarly, to correct the third fragment in this example, we identify that (2,1) is in the consensus, which implies that the majority of the related Rmaps are such that two fragments of R_i match with one fragment from the set of related Rmaps and therefore replace the third and fourth fragments with the average from the corresponding Rmaps and positions.

The threshold d determines the accuracy and precision of error correction. A high value of d improves precision but lowers accuracy as many fragments are left uncorrected. Similarly, a low value of d improves accuracy but lowers precision. The default setting is $d = 3$.

Complexity

We define ℓ to be the length of the longest Rmap in R. Quantization of the Rmaps takes $O(\ell \times n)$ time. Constructing the k-mer index also takes $O(\ell \times n)$ time. The k-mer index stores the occurences of each quantized k-mer across all Rmaps. Let u be maximum frequency of a k-mer. That is, a k-mer occurs in max u Rmaps (in practice $u << n$). Then, the complexity of finding related Rmaps from the k-mer index is $O(n \times \ell \times u)$. For each Rmap, the filtering heuristic runs in time linear to the size of the Rmap. Therefore, filtering the set of related Rmaps also takes linear $O(\ell \times n)$ time. The most expensive step is the pairwise alignment that uses the Valouev et al. aligner. As mentioned earlier, this aligner is based on DP and therefore has a $O(\ell^2)$ time complexity to perform one pairwise alignment. If the maximum cardinality of the set of related Rmaps for any Rmap is v, then the total complexity of this step is bounded by $O(n \times v \times \ell^2)$. The value of v depends on the coverage of the optical map data. The alignment generated using the Valouev et al. method is stored in the multiple alignment grid in constant time, and it takes $O(n \times v \times \ell)$ time to generate the consensus maps for n Rmaps and error-correct them. Thus, the runtime of cOMet is $O(n \times v \times \ell^2)$.

Datasets

We performed experiments on both simulated and real data. For the real data, we used the Rmap data from the plum [22] and domestic goat [3] sequencing projects. These datasets were built on the OpGen mapping platform and are more error prone. We also experimented on a human dataset [23] built on the new BioNano platform. This dataset is built using the latest optical mapping

Multiple alignment grid (A_i)

A_i	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	11th	12th	13th
1	(1,1)	(2,2)	$(2,2)_{\{2.168,3.511\}}$	$(1,1)_{\{2.107\}}$	$(2,1)_{\{10.221\}}$	(2,1)	(1,2)	(1,1)	(1,1)	(2,2)	(2,2)	(1,1)	(1,2)
2	(0,0)	(1,1)	$(2,1)_{\{4.344\}}$	$(2,1)_{\{4.344\}}$	$(1,1)_{\{8.488\}}$	(1,1)	(1,2)	(1,1)	(1,1)	(2,2)	(2,2)	(2,3)	(2,3)
3	(0,0)	(1,1)	$(2,1)_{\{4.129\}}$	$(2,1)_{\{4.129\}}$	$(1,1)_{\{8.132\}}$	(1,1)	(1,2)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)	(1,2)
4	(0,0)	(1,1)	$(2,1)_{\{4.311\}}$	$(2,1)_{\{4.311\}}$	$(1,1)_{\{8.964\}}$	(1,1)	(1,2)	(1,1)	(1,1)	(2,2)	(2,2)	(2,3)	(2,3)
5	(1,1)	(1,1)	$(2,1)_{\{4.611\}}$	$(2,1)_{\{4.611\}}$	$(2,1)_{\{10.692\}}$	(2,1)	(3,3)	(3,3)	(3,3)	(1,1)	(1,1)	(1,1)	(1,2)
6	(0,0)	(1,1)	$(2,1)_{\{4.710\}}$	$(2,1)_{\{4.710\}}$	$(1,1)_{\{9.432\ \}}$	(1,1)	(1,2)	(1,1)	(1,1)	(2,2)	(2,2)	(2,3)	(2,3)

Consensus grid (C_i)

C_i	(0,0)	(1,1)	(2,1)	(2,1)	(1,1)	(1,1)	(1,2)	(1,1)	(1,1)	(2,2)	(2,2)	(1,1)	(1,2)

Error correction

R_i	1.474	3.625	2.092	2.164	8.424	2.331	24.824		7.267	2.954	12.578	2.358	8.955	22.943	
C_i	(0,0)	(1,1)	(2,1)	(2,1)	(1,1)	(1,1)	(1,2)		(1,1)	(1,1)	(2,2)	(2,2)	(1,1)	(1,2)	
R'_i	1.474	3.273	4.421		8.754	2.608	9.988	13.891	7.184	3.472	8.532	6.032	10.038	5.633	15.869

Figure 2: Example of a multiple alignment grid and a consensus grid. The figure shows the multiple alignment grid A_i for an Rmap R_i and its consensus grid C_i. Each row of the multiple alignment grid represents the alignment of R_i with one of its related Rmaps, and the columns represent the fragments of R_i. The figure also demonstrates error correction using the consensus grid, with the error-corrected Rmap denoted as R_i'. The fragment sizes are in Kbp. To demonstrate the error correction process for the 3rd, 4th, and 5th fragments, we also include the fragments (in parentheses) to which they align. The error-corrected fragment is the mean of the fragments from the corresponding positions that have the same alignment as the consensus. For example, for the 5th fragment, the consensus is (1,1). Therefore, the mean of the aligned fragments with (1,1) alignment, i.e., 8.488, 8.132, 8.964, and 9.432, is the error-corrected value for the 5th fragment.

Table 1: Summary of the real and simulated data

Genome	Size	No. of Rmaps
E. coli	4.6 Mbp	2,504
Plum	284 Mbp	749,895
Goat	2.66 Gbp	3,447,997

Rmaps with fewer than 10 fragments were omitted from all the experiments. cOMET was run on the remaining 2,504, 548,779, and 3,049,439 Rmaps for the *E. coli*, plum, and goat genomes, respectively.

Table 2: Summary of the real and simulated BioNano data.

Genome	Size	No. of Rmaps
E. coli	4.6 Mbp	123,251–157,743
Human	3.2 Gbp	793,199

OMSim was used to simulate eight different BioNano datasets, each of which had varying error rates and, thus, had a different number of Rmaps.

Table 3: Results for the data simulated from *E. coli* K-12 MG 1655.

Total no. of insertion errors corrected	556
TPR of corrected insertions	82.49 % (457)
FPR of corrected insertions	0.21 % (99)
Total no. of deletion errors corrected	5,894
TPR of corrected deletions	77.38 % (5,792)
FPR of corrected deletions	0.25 % (102)

The data was simulated according the algorithm described in Datasets. This simulation resulted in 2,505 Rmaps containing 7,485 deletion and 554 insertion errors.

technology and has significantly better quality than the plum and goat genomes. The genome size and number of Rmaps for these species are shown in Tables 1 and 2.

In addition, we simulated Rmap data from *E. coli* K-12 substr. MG 1655 as follows. First, the reference genome was copied 200 times, and uniformly distributed random loci were selected for each of these copies. These loci form the ends of a single molecule that would undergo *in silico* digestion. Next, molecules smaller than 150 Kbp were discarded, and the cleavage sites for the RsrII enzyme were then identified within each of these simulated molecules. These error-free Rmap data are used for validating the output of our method. Last, deletion, insertion, and sizing errors were incorporated into the error-free Rmaps according to the error model discussed by Li et al. [15]. The error model was described earlier in the Background section. This simulation resulted in 2,505 Rmaps containing 7,485 deletion and 554 insertion errors.

Last, we simulated optical map data from a simulation software called OMSim [24] that generates synthetic optical maps that mimics real Bionano Genomics data. The software takes two parameters as input: the false positive rate (FPR) rate, which is the number of additional cut sites erroneously inserted per 100kbp, and the false negative rate (FNR), which is the percentage of times a cut site is missed. Using this method, we simulated eight datasets of Rmaps from *E. coli* K-12 substr. MG 1655 using the restriction enzyme BspQI. The default FPR and FNR for BspQI are 1% and 15%, respectively. We generated additional datasets with the following error rates (FPR,FNR) : (0.5%,15%), (1.0%,15%), (2.0%,15%),(5.0%,15%), (1.0%,5%), (1.0%,25%), (2.0%,5%),and (2.0%,25%).

Experiments and Discussion

We performed all experiments on Intel E5-2698v3 processors with 192 GB of random access memory (RAM) running 64-bit Linux. The input parameters to cOMET include b (quantization bucket size), k (k-mer value), m (the number of k-mers needed to be conserved between two Rmaps), and d (the minimum number of Rmaps required to form consensus at a position). The default parameters are $b = 4{,}000$, $k = 4$, $m = 1$, and $d = 3$and led to the best result across all datasets.

Determining the value of S_t

The setting of the parameter S_t depends on the sensitivity of the Valouev et al. aligner. If the alignment score between two Rmaps is less than S_t, then the aligned Rmaps are deemed to be unrelated. We say an Rmap, R_s is overlapping with an Rmap, R_t if at least 50% of R_s overlaps with R_t. That is, either the first half or the second half of R_s is entirely and exactly (exact fragment matches) contained in R_t.

We carried out the following experiment to determine the optimum setting for S_t. From the set of simulated error-free Rmaps, we computed the set of overlapping Rmaps for each Rmap. We denote this set as *related Rmaps*. Then, we used the Valouev et al. aligner to score all pairwise wise alignments between the simulated Rmaps (with errors added) and plot the scores in the form of a histogram, which is shown in Fig.3. The percentage of related Rmaps with an S-score less than 8 is 6.06%. Hence, we choose the setting of $S_t = 8$.

Experiments with our simulated data

The cOMet error correction was run on the simulated *E. coli* data. The corrected Rmaps were then aligned to the error-free Rmaps to determine the number of corrected insertions and deletions. The results of this experiment are shown in Table 3. To determine the quality of error correction, we computed the true positive rate (TPR), which is the ratio between the number of insertion (or deletion) errors that cOMet correctly identified and removed and the number of insertion (deletion) errors, and the FPR, which is the ratio between the number of insertion (or deletion) errors that cOMet incorrectly identified and removed and the total number of fragments not containing an insertion (deletion) error. The TPR is 82.49% and 77.38% with respect to the number of corrected insertions and deletion errors; whereas the FPR is 0.21% and 0.25% with respect to the number of corrected insertions and deletion errors. This demonstrates the high accuracy of the correction made by cOMet. Our method also has high precision. Of the deletion errors corrected, 98.26% are true errors. Similarly, of the insertion errors corrected, 82.19% are true errors.

Additionally, for each corrected Rmap, we computed the alignment S-score of both the original Rmap and the corrected Rmap with the error-free Rmap. We found that for 96.5% of the Rmaps, the S-scores improved after error correction. In other words, cOMet brought 96.5% Rmaps closer to their error-free state. The mean S-score before error correction was 44.91 and it improved by 14.03% to 51.30 after error correction. For 17.5% of the Rmaps (415 Rmaps), the S-score improved by more than 10. Last, we mention that the error correction was achieved in 241 central processing unit (CPU) seconds and using 79.54 MB of memory.

To demonstrate the importance of error correction, we assembled the Rmaps before and after error correction using the Valouev et al. assembler [25]. Table 4 summarizes the results of this experiment. We assembled the uncorrected data into five assembled optical maps and the error-corrected data into two assembled optical maps. The N50 statistic of the assembly increased from 1,242 Kbp for the uncorrected data to 3,348 Kbp for the corrected data. Next, we aligned each assembled map to the genome-wide (error-free) optical map using the Valouev et al. aligner in order to locate their positions on the genome and calculate the percentage of the genome that was covered by at least one of the assembled maps. The genome fraction covered by the five assembled maps from the uncorrected Rmaps was 80%, while the genome fraction covered by the two assembled maps from the corrected Rmaps was 82%. Moreover, the assembled maps from the uncorrected data had 47 insertion and deletion errors when aligned to the reference, while the error-corrected data had only 34 such errors. In order to further contextualize these results, we assemble the error-free Rmap dataset and summarize this assembly in Table 4.

Experiments with OMSim data

To present the robustness of our method and its applicability across datasets, we conducted experiments on synthetic data from an optical map–simulating software called OMSim [24]. As described in the Datasets section, we generated eight datasets of synthetic optical maps by varying the insertion and deletion error rates.

In the first experiment, we fixed the FNR at 15% and varied the FPR between 0.5, 1.0, 2.0, and 5.0, respectively. For each of the four datasets, we align each Rmap (using the Valouev et al. aligner) before and after error correction to the reference optical map obtained using the same restriction enzyme and report the percent of Rmaps whose alignment S-score increased after error correction and the mean increase in the S-score. We note that for each Rmap, the aligner returns the highest-scored alignment, and the score represents how closely the Rmap aligns to the reference genome-wide optical map. Table 5 summarizes the results from this experiment. We observe that the efficiency of error correction improves as the FPR is initially increased. When the FPR reaches a high value of 5, the efficiency of error correction drops. The mean S-score improves by more than 9 ($\sim$10%) when the FPR is reasonable.

In the second experiment, we first fix the FPR at 1.0 and vary the FNR between 5%, 15%,and 25%, respectively. We then fix the FPR at 2 and vary the FNR between 5%, 15%,and 25%, respectively. We report the same results as in the previous experiment. Table 6 shows the results. Similar to the previous experiment, we find that the efficiency of error correction improves as the FNR increases from 5% to 25%. The error correction improves the quality of a high percentage of Rmaps (>70%) for all values of parameters.

Experiments with real data

Table 7 summarizes the results of running cOMet on the plum and goat datasets. The plum and goat datasets do not contain error-free Rmaps. Therefore, we are restricted to reporting the number of corrections made and the improvement to the S-score. In order to compute the S-score before and after error correction, we generated an *in silico* digested genome-wide optical map from the reference genome and aligned both the uncorrected and corrected Rmap to the genome-wide optical map. If it aligned to multiple positions, then we considered the alignment position where the corrected Rmap aligned with greatest S-score and considered the difference in the S-score when the uncorrected and corrected Rmap aligned to that position. However, we note that this process is error prone because of the fragmented nature of the draft genomes and possible misassemblies present in the genomes. We observed that the S-score after error correction improved for 78% of the plum Rmaps and 99% of the goat Rmaps. Figures 4 and 5 show the histograms of the distribution of S-scores before and after error correction. For the plum genome, the mean S-score improved from 8.60 before error correction to 14.72 after error correction (a 71% improvement in the score), while for the goat genome, it improved from 9.38 before

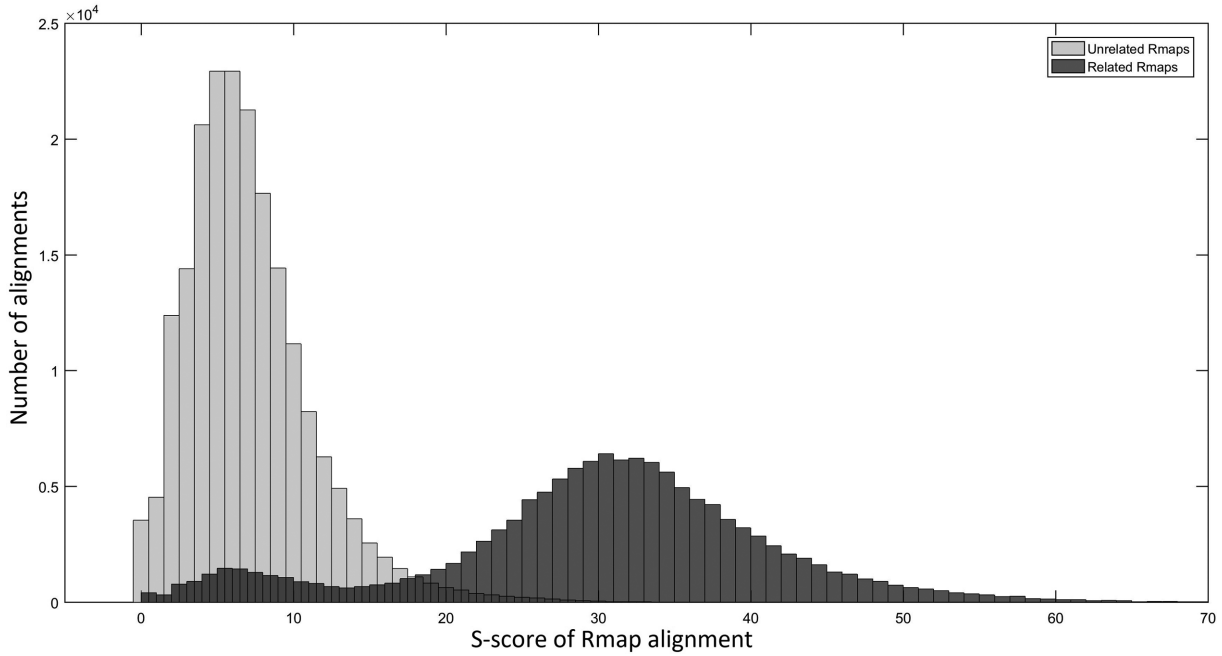

Figure 3: Distribution of S-scores of Rmap alignments between related Rmaps and unrelated Rmaps.

Table 4: Assembly results of uncorrected Rmaps, corrected Rmaps, and error-free Rmaps using the Valouev et al. assembler

Rmap status	Assembled Map_id	Number of fragments	Map length (Kbp)	Alignment location in reference (start-loci, end-loci)
Uncorrected Rmaps	Assembled Map_0	75	921.41	(246,321)
	Assembled Map_1	88	1,242.40	(11,95)
	Assembled Map_2	30	531.65	(225,255)
	Assembled Map_3	44	759.16	(181,228)
	Assembled Map_4	107	1,699.60	(87,194)
Corrected Rmaps	Assembled Map_0	102	1,397.60	(225,322)
	Assembled Map_1	237	3,348	(8,230)
Error-free Rmaps	Assembled Map_0	60	808.74	(185,239)
	Assembled Map_1	91	1,100.5	(241,324)
	Assembled Map_2	104	2,474.4	(19,185)

The Rmaps are simulated from the *E. coli* genome. Each assembled map is aligned to the reference genome-wide (error-free) optical map using the Valouev et al. aligner. The genome-wide optical map contains 383 fragments.

Table 5: Efficiency of error correction when the FPR is varied and the FNR is fixed at 15%

FPR	No. of Rmaps	Percent of Rmaps with improved S-score	Mean S-score before error correction	Mean S-score after error correction	Mean S-score improvement
0.5	129,820	93.42	78.66	91.12	12.46
1.0	126,133	94.01	76.25	89.44	13.19
2.0	140,623	92.99	71.93	85.29	13.36
5.0	130,019	81.35	64.65	71.36	6.71

correction to 16.97 after correction (an 80.92% improvement in the score).

We also measured the genome coverage, i.e., the fraction of the genome covered by at least one Rmap, for both the original Rmaps and the corrected Rmaps as follows. First, we aligned all Rmaps to the genome-wide optical map and then picked the best alignment for each original Rmap and each corrected Rmap. Based on these alignments, we then computed the fraction of the genome covered by at least one original Rmap and the fraction of the genome covered by at least one corrected Rmap. On the goat genome, the genome coverage was 73.08% before correction and it increased to 84.56% after correction. The increase in genome coverage shows that our method is able to correct Rmaps from across the genome. Furthermore, it shows that even

Table 6: Efficiency of error correction when the FNR is varied

FPR	FNR (%)	No. of Rmaps	Percent of Rmaps with improved S-score	Mean S-score before error correction	Mean S-score after error correction	Mean S-score improvement
1.0	5	142,684	87.36	81.32	87.62	6.3
	15	126,133	94.01	76.25	89.44	13.19
	25	123,252	96.09	70.24	87.44	17.20
2.0	5	148,912	89.23	76.54	84.47	7.93
	15	140,623	92.99	71.93	85.29	13.36
	25	130,763	93.02	66.95	81.65	14.7

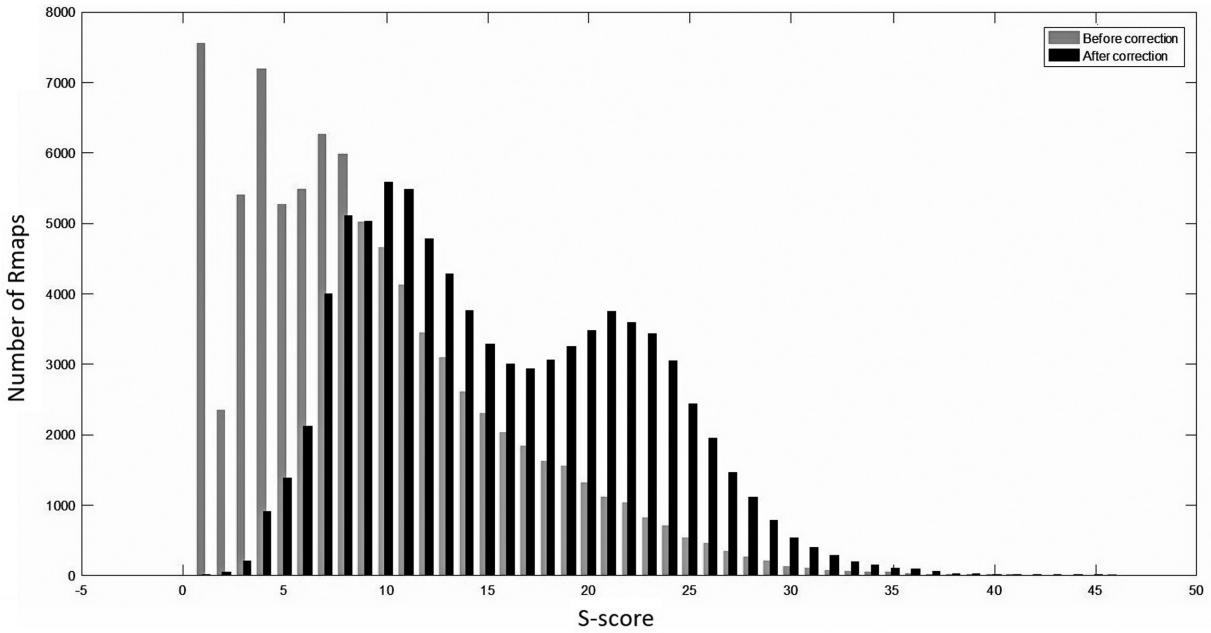

Figure 4: Alignment scores of Rmaps from the plum genome with the reference optical map. Before error correction, the S-score had a mean of 8.6 with a standard deviation of 6.49. After error correction, the mean S-score improved to 14.72 with standard deviation of 6.72.

Table 7: Results on the Rmap data of plum and goat genomes

Genome name	Plum	Goat
Running time	7.4 days	105.7 days
Memory	12.20 GB	113.56 GB
No. of insertion errors corrected	433,282	2,530,060
No. of deletion errors corrected	430,329	3,187,023

Peak memory was measured as the maximum resident set size as reported by the operating system with sufficient RAM to avoid paging. Running time is the user process time, also reported by the operating system.

if Rmaps could not originally be reliably aligned to some regions of the genome, our method is sensitive enough to recover similar Rmaps from these regions; thus, after correction, the fraction of the genome covered by aligned Rmaps is higher. For the plum, the genome coverage dropped negligibly from 99.01% before error correction to 98.85% after error correction (which is less than 1% of the genome size).

In addition, as shown in Table 7, the running time and peak memory usage were recorded for the plum and goat genomes. Although these experiments have significant running times (7.4 and 105.7 CPU days for plum and goat, respectively), these figures are not prohibitive given that this computation can easily be parallelized since the error correction process for each Rmap is independent. For example, we ran the goat genome on 20 machines, and it required 126.84 hours for all Rmaps to be corrected. In addition, we note that error correction of a dataset will likely only be done once for any dataset, so 5.2 human days for a large genome is not unreasonable. Last, the peak memory usage was 12.20 GB and 113.56 GB for plum and goat, respectively; thus, cOMet is able to run on any modern server.

Next, we ran experiments on the human dataset. Again, since we do not possess the error-free Rmaps corresponding to the raw Rmaps for this dataset, we follow a similar evaluation method as in the previous experiments. We performed our evaluation on an *in silico* digested human reference genome (GenBank assembly accession: GCA_000001405.15, Genome Reference Consortium Human Build 38) using BspQI, which was the restriction enzyme that was used for generating the Rmap data. cOMet improved the S-score of 74.78% of the Rmaps. The average S-score improved from 85.96 before error correction to 88.65 after error correction.

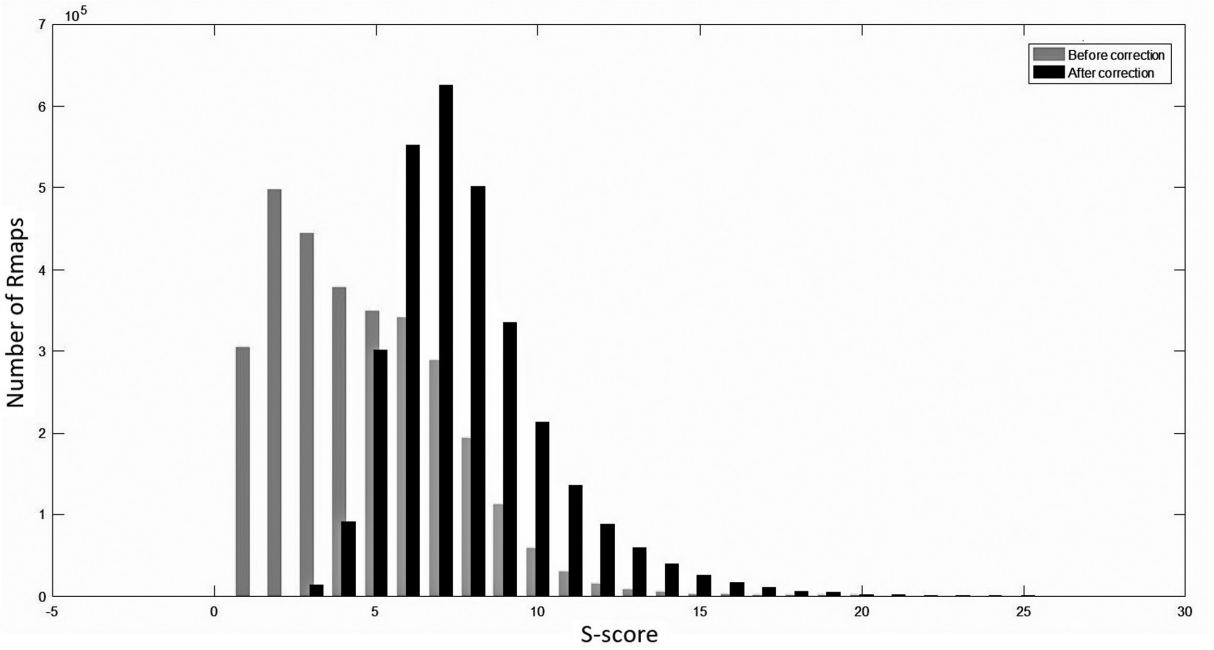

Figure 5: Alignment scores of Rmaps from the goat genome with the reference optical map. The mean and standard deviation of the S-scores before error correction were 9.38 and 6.54, respectively. After error correction, the mean S-score improved to 16.97 with a standard deviation of 6.21.

Conclusion

Error correction of high-throughput sequencing data has become an imperative preprocessing step in genome assembly since 2008 when Chaisson and Pevzner showed the dramatic improvement it can have on the quality of the assembly [26, 27, 28]. For example, after error correction, the contig N50 size of an assembly of *Rhodabacter sphaeroides* improved from 233 bp to 7,793 bp using the same assembler [28]. Due to this inarguable benefit on genome assembly, many methods have been developed for error correction of sequence reads, including BFC [29], Coral [30], EULER [26, 31], and Reptile [32]. Unfortunately, even though there has been a massive effort in error correction of sequence data, there currently does not exist a publicly released method for error correction of Rmap data—a method that would likely improve the quality of genome-wide optical map assemblies and allow such assemblies to be computed with greater efficiency.

Here, we presented cOMet, an error correction method for Rmap data and demonstrated that it corrects and improves the quality of a high percentage of Rmaps in both simulated and real datasets. As previously discussed, Rmap data are subject to high error rates. In addition to insertion and deletion errors, they contain sizing errors that necessitate the use of a dynamic programming algorithm for pairwise alignment and, subsequently, assembly. By correcting a significant number of errors in Rmap data, cOMet can make it possible to use faster alignment methods [18, 19, 17] and explore the development of more efficient Rmap assembly algorithms.

Availability of source code

Project name: cOMet
Project home page : https://github.com/kingufl/cOMet
Operating system(s): Linux
Programming language: C++
Other requirements: GCC version 5.2.0 or higher
License: GNU General Public License
Research resource identifier: SCR_016276
An archival copy of the code is available via the *GigaScience* repository GigaDB[33].

Availability of supporting data

The optical mapping data for plum and goat are publicly available and can be accessed from their respective manuscripts [3, 22] and via GigaDB [34] The human optical map data can also be accessed from its manuscript [23]. The simulated data for *E. .coli* are provided in the github repository along with the python scripts used to generate it, and snapshots of the data and code are also included in GigaDB[33].

Additional material

In Figure S.1 we show the distribution of lengths of Rmaps whose S-score increases after error correction. From the distribution, we can tell that our method is able to error correct Rmaps of all sizes. We also show the distribution of fragment sizes from Rmaps whose score increases after error correction in Figure S.2.

Figure S.1: Distribution of Rmap lengths whose S-score increased after error correction. The Rmaps are simulated from the *E. coli* K-12 substr. MG 1655 as explained in the text.

Figure S.2: Distribution of fragment sizes of Rmaps whose S-score increased after error correction. The Rmaps are simulated from the *E. coli* K-12 substr. MG 1655 as explained in the text.

Abbreviations

CPU: central processing unit; FNR: false negative rate; FPR: false positive rate; RAM: random access memory; TPR: true positive rate; CCD : charged coupled device.

Funding

K.M., D.W., M.M., and C.B. were funded by the National Science Foundation (1618814). L.S.was funded by the Academy of Finland (grants 284598 [CoECGR], 308030, and 314170).

References

1. Schwartz DC, Li X, Hernandez LI, et al. Ordered restriction maps of *Saccharomyces Cerevisiae* chromosomes constructed by optical Mmapping. Science 1993;**262**:110–114.
2. Zhou S, Wei F, Nguyen J, et al. A single molecule scaffold for the maize genome. PLoS Genetics 2009 **11**;5:e1000711.
3. Dong Y, Xie M, Jiang Y, et al. Sequencing and automated whole-genome optical mapping of the genome of a domestic goat (*Capra hircus*). Nature Biotechnol 2013. http://dx.doi.org/10.5524/100082.
4. Chamala S, Chanderbali AS, Der JP, et al. Assembly and validation of the genome of the nonmodel basal angiosperm *Amborella*. Science 2013;**342**(6165):1516–1517.
5. Teague B, Waterman MS, Goldstein S, et al. High-resolution human genome structure by single-molecule analysis. Proc Natl Acad Sci U S A 2010;**107**(24):10848–10853.
6. Ganapathy G, Howard JT, Ward JM, et al. *De novo* high-coverage sequencing and annotated assemblies of the budgerigar genome. GigaScience 2014;**3**, 1–9.
7. Muggli MD, Puglisi SJ, Ronen R, et al. Misassembly detection using paired-end sequence reads and optical mapping data. Bioinformatics 2015;**31**(12):i80–i88.
8. Reslewic S, Zhou S, Place M, et al. Whole-senome Shotgun optical mapping of *Rhodospirillum rubrum*. Appl Environ Microbiol 2005;**71**(9):5511–5522.
9. Zhou S, Deng W, Anantharaman TS, et al. A whole-genome Shotgun optical map of *Yersinia pestis* strain KIM. Appl Environ Microbiol 2002;**68**(12):6321–6331.
10. Zhou S, Kile A, Kvikstad E, et al. Shotgun optical mapping of the entire *Leishmania major* Friedlin genome. Mol Biochem Parasitol 2004;**138**(1):97–106.
11. Zhou S, Bechner MC, Place M, et al. Validation of rice genome sequence by optical mapping. BMC Genomics 2007;**8**(1):278.
12. Church DM, Goodstadt L, Hillier LW, et al. Lineage-specific biology revealed by a finished genome assembly of the mouse. PLoS Biology 2009;**7**(5):e1000112+.
13. Zhou S, Herschleb J, Schwartz DC. A single molecule system for whole genome analysis. Perspectives in Bioanalysis 2007;**2**, 265–300.
14. Valouev A, Li L, Liu YC, et al. Alignment of optical maps. J Comp Biol 2006;**13**(2):442–462.
15. Li M, Mak ACY, Lam ET, et al. Towards a more accurate error model for BioNano optical maps. In: ISBRA 2016;pp. 67–79.
16. Chaisson MJ, Tesler G. Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. BMC Bioinformatics 2012;p. 238.
17. Muggli MD, Puglisi SJ, Boucher C. Efficient indexed alignment of contigs to optical maps, Algorithms in Bioinfomatics. WABI. 2014;68–81.
18. Leung AKY, Kwok TP, Wan R, et al. OMBlast: alignment tool for optical mapping using a seed-and-extend approach. Bioinformatics 2016;p. btw620.
19. Mendelowitz LM, Schwartz DC, Pop M. Maligner: a fast ordered restriction map aligner. Bioinformatics 2016;**32**(7):1016–1022.
20. Smith TF, Waterman MS. Identification of common molecular subsequences. J Mol Biol 1981;**147**(1):195–197.
21. Bradnam KR, Fass JN, Alexandrov A, et al. Assemblathon 2: evaluating *de novo* methods of genome assembly in three vertebrate species. GigaScience 2013;**2**(1):1–31.
22. Cai M, Chen W, Du D, et al. Genomic data of the plum (*Prunus mume*). GigaScience Database 2014. http://dx.doi.org/10.5524/100084.
23. Shi L, Guo Y, Dong C, et al. Long-read sequencing and de novo assembly of a Chinese genome. Nature Communications 2016;http://dx.doi.org/10.1038/ncomms12065.
24. Miclotte G, Plaisance S, Rombauts S, et al. OMSim: a simulator for optical map data. Bioinformatics 2017;2740–2.
25. Valouev A, Schwartz DC, Zhou S, et al. An algorithm for assembly of ordered restriction maps from single DNA molecules. Proc Natl Acad Sci U S A 2006;**103**(43):15770–15775.
26. Chaisson MJ, Brinza D, Pevzner PA. De novo fragment assembly with short mate-paired reads: does the read length matter? Genome Res 2009;**19**(2):336–346.
27. Ekblom R, Wolf JBW. A field guide to whole-genome sequencing, assembly and annotation. Evolutionary Applications 2014;**7**(9):1026–1042.
28. Salzberg SL, Phillippy AM, Zimin A, et al. GAGE: a critical evaluation of genome assemblies and assembly algorithms. Genome Res 2012;**22**(3):557–567.
29. Li H. BFC: correcting Illumina sequencing errors. Bioinformatics 2015;**31**(17):2885.
30. Salmela L, Schröder J. Correcting errors in short reads by multiple alignments. Bioinformatics 2011;**27**(11):1455–1461.
31. Pevzner PA, Tang H, Waterman MS. An Eulerian path approach to DNA fragment assembly. Proc Natl Acad Sci 2001;**98**(17):9748–9753.
32. Yang X, Dorman KS, Aluru S. Reptile: representative tiling for short read error correction. Bioinformatics 2010;**26**(20):2526.
33. Mukherjee K, Washimkar D, Muggli MD, et al. Supporting data for “Error Correcting Optical Mapping Data.” GigaScience Database; 2018. http://dx.doi.org/10.5524/100434.
34 Bian C, Chen J, Chen W, Genomic data of the goat (Capra hircus). GigaScience Database 2014, http://dx.doi.org/10.5524/100082.