Supplementary Material

S1 OMKar Report

For ease of use, OMKar optionally compiles a report that includes 1) the observed chromosomal segment lists with reconstructed karyotype, 2) a list of interpreted events using ISCN notation, 3) a visualization of each corresponding chromosome with cytoband and event labels, and 4) a table of important genes that are near the breakpoints of an event or have copy number (CN) alteration. An html version of the report is prepared for easy viewing.

By default, OMKar outputs a molecular karyotype in a text format to unambiguously describe the karyotype as follows: It first lists all defined segments across the reference genome. For each segment, the following information is provided: segment number, chromosome, start and end coordinates, and the graph nodes representing the segment. Each segment is represented by two nodes, connected by a segment edge. All segments are forward-oriented (i.e., the end coordinate is greater than or equal to the start coordinate) and are sorted by chromosome groups (from 1 to Y) and increasing coordinates. The segments are non-overlapping, with no gaps between them, although telomeric regions of the reference genome may be excluded.

Following the segment definitions, OMKar reports the reconstructed paths that represent a karyotype. Each path consists of a list of segments in the format "Path number = segment number followed by direction." The segments are traversed either in the forward ('+') or reverse ('-') direction, where '+' indicates traversal from the start to the end of the segment, and '-' indicates traversal from the end to the start of the segment. For example, "Path1 = 1+ 2+ 3-" means that the path traverses segment 1 in the forward direction, segment 2 in the forward direction, and segment 3 in the reverse direction. Additionally, the number of centromeres present in each path is reported, which should ideally be one, indicating a valid chromosome structure.

S2 Terminal Event Simulation and Validation

We simulated a total of 117 terminal Structural Variations. During analyses, it was found that only 50 of the 117 SVs (42.7%) were reconstructed under the 0.2 Mbp matching distance (same

distance used for non-terminal SVs; Supplementary Table S1). However, 78 of the 117 SVs (66.7%) were reconstructed under the 5 Mbp matching distance. Because one of the endpoint was in the terminal masking region, most terminal SV edge did not have an SV call. For unbalanced SVs, OMKar reconstructed these variations using the information from CNV calls, which had much greater error in the event boundaries. For inversion, as a balanced SV, missing the SV call meant OMKar had no other information on the SV, thus, resulting in a much lower recall rate at all distances. In addition, terminal events such as arm deletion may result in a false loss of centromere call, which result in a relatively higher false positive chromosomal-loss aneuploidy reconstruction from OMKar.

From the real data we received, we observed no terminal structural variation, hinting that terminal SVs were far less frequent than our simulation. For future reference, OMKar may incorporate additional information such as alignment contigs to improve the boundary accuracy and recall rates.

S3 OMKar details

S3.1 Filtering SV and CNV calls

OMKar utilizes the CNV and SV calls from the Bionano pipeline. We applied several filters to improve data quality. First, we filtered out low-confidence CNV calls, defined as those with a confidence level of 0.95 or below, as well as those located in masked regions that could interfere with the analysis. Next, we excluded CNVs smaller than 200 kbp. However, if an excluded CNV was supported by a corresponding SV, it was retrieved during later processing, ensuring no relevant variations were missed.

Following these steps, SV calls were filtered based on variation-specific confidence thresholds established by BioNano pipeline: translocations ($T_{trans} = 0.05$), inversions ($T_{inv} = 0.7$), and indels ($T_{indel} = 0$). Additionally, SV calls in masked regions were discarded. Breakpoints from SVs were processed by sorting them based on chromosomal and genomic coordinates and merging adjacent breakpoints within a 50 kbp window to simplify the breakpoint graph. To ensure accurate representation, CNV segments were split if breakpoints occurred within their boundaries, guaranteeing that all breakpoints exclusively connect the terminal coordinates of segments.

S3.2 Converting to linear constraints

To tackle the inherent non-linearity of the absolute value and sign functions within the optimization problem, we introduce new variables and constraints to linearize these functions.

To convert the sign function into linear constraints, we employed the following approach: The ILP formulation for $s'_e = \text{sgn}(s_e)$ when $s_e \geq 0$ is:

$$s_e \geq s'_e,$$
$$s_e \leq 1000 \cdot s'_e$$

where s'_e is a binary variable. This formulation ensures that $s'_e = 1$ when $s_e > 0$ and $s'_e = 0$ when $s_e = 0$, while maintaining linearity in the constraints. The constant 1000 is used as a sufficiently large value to approximate an upper bound on s_e.

The absolute value function can be linearized $y_v = |x_v|$ as follows:

$$y_v \geq x_v,$$
$$y_v \geq \quad x_v,$$
$$y_v \geq 0,$$

where y_v represents the absolute value of x_v. These constraints ensure that y_v is equal to $|x_v|$ by considering both the positive and negative cases for x_v. Since our objective is to minimize y_v, it naturally corresponds to $|x_v|$. Thus:

$$\text{Objective Function} = \gamma \sum_{v \in V} (c_v + x_v \quad r_v \quad \sum_{e \in SV \ v)} s_e) + \sum_{v \in V} \beta_v |x_v| \quad \sum_{s_e \in SV} \ _e \text{sgn}(s_e) + \lambda \sum_{v \in V} o_v$$

$$= \gamma \sum_{v \in V} (c_v + x_v \quad r_v \quad \sum_{e \in SV \ v)} s_e) + \sum_{v \in V} \beta_v y_v \quad \sum_{s_e \in SV} \ _e s'_e + \lambda \sum_{v \in V} o_v$$

S3.3 Parameter determination

In determining $_e$, we consider the characteristics of structural variation e. If e signifies a high-confidence call and connects two distinct chromosomes, we set $_e = 72$ otherwise, $_e = 9$. A higher value is given to intra-chromosomal translocations to prioritize their preservation in the final karyotype. For the calculation of β_v, we introduce a penalty parameter that becomes more pronounced as segments grow in length. To achieve this, we define $\beta_v = 4 \times \lceil \frac{\text{length } v)}{5 \ \ 10^6} \rceil$. This parameter β_v ensures that the penalty for modifying longer segments is appropriately weighted.

The parameter γ is empirically determined and set to a value of 5, while λ is fixed at 1 to ensure that if two candidate decompositions are nearly identical, our preference is to select the one with fewer odd vertices. This empirical choice addresses the unique requirements of our approach, providing a balanced framework for the optimization process (Supplementary Table S4).

S3.4 OMKar Eulerian Path algorithm

To reconstruct chromosomal structures from breakpoint graphs, OMKar computes an Eulerian path, ensuring each edge is traversed exactly once while maintaining biologically meaningful constraints. It begins with a breakpoint graph, where vertices represent segment boundaries, and edges denote segment continuity, reference adjacencies, or breakpoint rearrangements. To enforce chromosomal structure, the algorithm prioritizes reference edges (minimizing deviation from the reference genome), followed by breakpoint edges (capturing structural variations), and lastly, segment edges. Using a recursive depth-first traversal, it starts from a telomeric vertex—representing a natural chromosomal endpoint—and evaluates edges with a validity function that prevents breaking segment integrity or disconnecting essential graph components. The pseudo-code is given in Algorithm 1.

Algorithm 1: Find Eulerian Tour Starting from Vertex v in Graph G

```
Data: Graph G
Result: Eulerian Tour
Function ValidEdge(edge e(v,w), edge e(u,v)):
    if deg v)==1 then
        return True
    else if e(v,w) or e(u,v) is segment edge then
        if e(v,w) is not bridge edge then
            return True
    return False
Function FindEulerianTour(current vertex v, previous vertex u):
    Add v to the Eulerian tour list E;
    Initialize an empty list W;
    for each edge e(v,w) do
        if ValidEdge(e(v,w), e(u,v)) then
            Add w to the valid edges list W
    Sort W based on the edge types (E_r, E , E_s);
    Pop w from list W;
    remove e(v,w) from G
    FindEulerianTour(vertex w, vertex v)
Initialize an empty list E;
FindEulerianTour(Telomeric vertex v , -1);
```

S3.5 OMKar Report: Event Interpretation

After segregating the chromosomes, OMKar interprets the structural variations in each chromosomal cluster using ISCN notation(Hastings et al., 2024) as follows:

1. For each reconstructed chromosome in the chromosomal cluster, its chromosomal identity is assigned based on the highest represented centromere, and if it is acentric, by the overall highest-represented chromosome in the remaining segments.
2. In a pre-processing step, the Wild Type (WT) chromosome corresponding to the reconstructed chromosome is segmented prior to alignment, using the segments in the reconstruction.
3. Next, OMKar performs an alignment between each reconstructed chromosome and the corresponding Wild Type (WT) using an alignment that maximizes the longest common subse-

quence, with a linear penalty for indels.

4. After alignment, blocks of aligned segments are separated into three types: 1) concordant block represents matching between the reconstruction and the WT, 2) insertion block represents segments inserted in the reconstruction, and 3) deletion block represents segments deleted in the reconstruction. Adjacent blocks of the same type, and representing contiguous genomic coordinates are merged to minimize the total number of blocks.
5. The final interpretation assigns a representative SV name (using ISCN nomenclature) to each insertion or deletion block, reporting a potential cause of the deviation from the WT. Each SV has its unique signature in the combination of block types (Supplementary Table S5). For example, an inversion is always an insertion block of inverted segments next to a deletion block of non-inverted segments. In this example, it is more likely that a single inversion resulted in both the deletion and the insertion, instead of two separate events (inverted insertion and deletion). Similarly during the interpretation step, a compound SV is preferred over a sequence of simpler SVs (preference is given to reducing the total number of events).

To minimize the number of events, event types are resolved from the most complex to the least, using the following preference order: (1) inter-chromosomal balanced translocation and transposition; (2) intra-chromosomal balanced translocation and transposition; (3) other intra-chromosomal variations. Finally, if any deletions or insertions remain unaccounted for, they are marked as simple deletions or duplicated-insertions. During each variation type's resolution, each un-resolved block is iterated over, with the goal of being associated with the signature of that variation type. Balanced translocations attempt to associate anywhere in the cluster or chromosome, for inter- and intrachromosomal search, respectively. All other variations associate adjacent blocks.

For balanced translocations, the following step is performed after all insertion and deletion blocks are resolved. All balanced translocations are initially denoted as *transposition*, associating a deletion block and a non-adjacent insertion block of an overlapping set of segments, with allowance for small indels. When transpositions form a cycle, they are interpreted as *balanced translocations*. This is implemented recursively, by jumping between each associated deletion-insertion pair, and then looking for nearby transposition blocks of the opposite type. When exhausted, if a cycle is formed, an n-break balanced translocation is interpreted, and otherwise, all transpositions are

interpreted as individual transpositions.

When associating between di erent blocks, a procedure called "seed-matching" is applied. For each block that is currently being resolved, it searches for an associated block, such that the common subsequence between the blocks (in indivisible-unit of each segment) is sufficiently long (10 kbp by default). The sizes of the flanking segments not matched may be limited by an "indel allowance". For example, an insertion block of $(B \quad, A \quad)$ next to a concordant block of $(B + C+)$ will be interpreted as a left-duplication-inversion only if $A \quad$ is less than 50 kbp. On the other hand, the size of $C+$ is irrelevant for associating a duplication-inversion, as it is not between the two blocks. These allowances are determined empirically (Supplementary Section S4).

S3.6 OMKar Report: Visualization

The visualization is achieved by intersecting the coordinates in the observed chromosomal segment list with a given cytoband coordinate table. Each band pattern is then stacked and displayed using Matplotlib (v3.7.5). In an alternative plot, the bands are segregated by the segment boundaries in the OMKar output, instead of the cytoband. The event label is then applied to the corresponding segment location on the band stack.

By default, the table of important genes comes from genes that have CN alteration or within 20 kbp of an event's breakpoint. This list of genes are further filtered to only include those in the Developmental Disorders panel in the Gene2Phenotype database (DDG2P)(Thormann et al., 2019).

S4 Analyses of Event Distances

The resolution of OGM was determined by the matching distances between the Truth SV edge and the reconstructed SV edge, using KarCheck. Observations from this analyses were used to estimate the sensitivity of OMKar event interpretation (Supplementary Table S5).

When applying KarCheck between the simulated and the reconstructed karyotypes, history logs from KarSim's output was used to label SV edges on K_t. This step marks each SV edge with its causal rearrangement. During KarCheck matching, the distance between a pair of matched edges were recorded with the causal SV type. This distance is further separated as the two distances of

the endpoint.

Then, the matched edge from K_t was searched in OGM's SMAP output (contains all SV calls), with a proximity of 50 kbp, equal to the SV call merging distance of OMKar. From Fig. S7, we observed duplication inversions tend to have much larger distances than all remaining structural variations simulated. Most of the non-duplication-inversion were reconstructed with less than 5 kbp distance from the truth, much lower than the average gene size of 10-15 kbp. Thus, if an SV contains or interrupts a gene, OGM and OMKar are likely to reconstruct the correct boundary to perform genotype-to-phenotype inference. Additionally, for all distances greater than 50 kbp, the true SV edge was not captured with a high proximity in the SMAP, therefore, OMKar either had to reconstruct based on a distant SV call or had to infer the missing SV call based on CNV call.

S5 KarSim

S5.1 KarSim module for simulating karyotypes

The KarSim module outputs a molecular karyotype file, a corresponding FASTA file, and a history log with event segments and edges noted are outputted for downstream usage. The molecular karyotype and history log can be used for KarCheck comparison, while the FASTA file can be used as input for simulating the sequencing technology of choice, given that many of such simulators are already available. KarSim is publicly available at `https://github.com/MolecularKaryotype/KarSimulator`.

Three steps are taken in order, with step 2 being optional:

1. A template karyotype is created given the counts of autosomes and sex chromosomes.
2. (Optional) a series of SVs are applied to the karyotype
3. The FASTA formatted sequence of the rearranged chromosomes is generated.

All intermediate files in steps 1 and 2 are molecular karyotypes, which can be read-in for multiple parallel edits or outputting the FASTA file.

There are two methods to introduce additional SVs to a karyotype: 1) manual addition of SVs given the variation type and exact boundaries, and 2) using a parameter JSON file that contains the number of SVs, each SV type's likelihood, max/min size for each SV type, terminal occurrence

likelihood etc. Both processes can be applied multiple times and in an interleaved fashion. The types of SVs supported can be found in Supplementary Table S6. In addition, the user has the option to include a masking file such that SVs are not generated within proximity to any of these regions. Another parameter can be passed in to prevent events that result in the formation of segments smaller than a certain size, useful for sequencing technologies which have a resolution limit.

S5.2 Implementation

KarSim is implemented using Python (version 3.9). A flowchart of the algorithm is in Supplementary Fig. S8A. Each chromosome in a karyotype is represented as a sequence of oriented segment objects of the hg38 genome, denoted by the chromosome, start index, and end index. To simulate an SV, left and right breakpoints are first introduced to the chromosome, breaking up the chromosome into segments. This results in each SV breakpoint being on the exact border of a segment. Then, the corresponding rearrangement is applied to the segments to simulate the intended SV.

For the parameterized-random SV selection, SVs are selected one at a time, for the number of SVs indicated on the parameter file as follows:

1. The SV type is randomly selectedbased on its likelihood.
2. The event size is selected from a uniform distribution between the maximum and minimum size indicated for the SV type.
3. The left breakpoint of the event location is selected uniformly among all chromosomes.
4. The right breakpoint is calculated based on the size of the event selected earlier. For a balanced reciprocal translocation, it is selected similar to the left breakpoint.
5. The breakpoints are applied to partition the segments, and if a masked region file or a smallest segment allowance is applied, the resulting breakpoints and segments are checked for legality. If the result is illegal under the parameters, steps three through five are recomputed.

S5.3 Generation and Processing of Simulated OGM Data

The FASTA files generated by the KarSim module were processed by OMSim(Miclotte et al., 2017) to simulate a BNX file containing OGM molecules with added noise. Parameters for the enzyme,

OGM generation, and noise were sourced from publicly available repositories(RaeisiDehkordi, 2024). The Bionano Solve pipeline (v3.7)(BionanoGenomics, 2018) was then used to compute CNVs, SV calls, and contig alignments, and these outputs were used as input for OMKar to reconstruct the final virtual karyotype.

When simulating structural variations, we used the masking files provided in Bionano Solve v3.7 (also included in repository (RaeisiDehkordi, 2024)). The masking file is on reference hg38, with size of 423.1 Mbp (13.65%), including 128.1 Mbp (4.13%) in centromeres and telomeres, and 64.5 Mbp (2.15%) in acrocentric chromosomes' p-arm. None of the SVs was simulated with breakpoint boundary within 200 kbp of any masking region. Events were simulated with size 50 kbp to 2 Mbp.

S6 KarCheck

KarCheck is designed to be a symmetric comparator between two unphased karyotypes, denoted by K_r and K_t, by comparing their SV and CNV calls. In addition, to accommodate molecular techniques that do not have a nucleotide-level resolution, KarCheck has an adjustable tolerance for small breakpoint mis-matching for an SV that is present in both karyotypes. KarCheck is publicly available at `https://github.com/MolecularKaryotype/KarComparator`.

S6.1 KarCheck Preprocessing

Preprocessing is first applied to K_t and K_r to partition chromosome groups into *chromosome clusters*. Two chromosome groups are linked into the same cluster when there exists a breakpoint connecting them (signaling an inter-chromosomal SV). A maximal connected component of linked chromosome groups is denoted as a chromosome cluster.

Recall that we define each karyotype as an ordered list of segments. To make the two karyotypes comparable, we further partition their segments such that the two karyotypes share an identical set of segments (Supplementary Fig. S8B). To achieve this, all left/right endpoints of segments are collected, and if a segment has an endpoint internal to it, it is split into two, ensuring the left/right endpoint are on the boundaries (Supplementary Fig. S8B).

S6.2 SV similarity computation.

After pre-processing, the SVs between the two karyotypes are compared as a directed-multi-graph, which o ers ease of checking the orientation of each SV edge (Supplementary Fig. S8C). On this graph, nodes represent either the start or the end of a segment (denoted as s and t), with the addition of a source node (S) and a sink node (T) for transition at the start and end of each chromosome. Edges represent the presence of a transition between two segments. For example, a chromosome of [k+, l-], where the WT is [k+, l+], will be represented by edges (S, k_s),(k_s, l_t), and (l_s, T).

Since the two karyotypes share the same set of segments, they also share the same set of nodes. Therefore, the comparison between the two sets of edges on this graph is equivalent to the comparison of the two sets of SVs. To allow tolerance in breakpoint matching, we define a linear distance function. Denote two non-segment edges as (a, b, o) and (c, d, o'), where o (and o') describe orientation. Define distance $D((a, b, o), (c, d, o'))$ as:

$$D((a,b,o),(c,d,o')) = \begin{cases} \infty & \text{Chr}(a) \neq \text{Chr}(c) \text{ or } \text{Chr}(b) \neq \text{Chr}(d) \\ \infty & o \neq o' \\ |\text{pos}(a) \quad \text{pos}(c)| + |\text{pos}(b) \quad \text{pos}(d)| & \text{otherwise} \end{cases}$$

Prior to distance computation, identical edges between the two graphs are pruned. Second, if a transition is s-to-t or t-to-s and is between two segments from the same chromosome with a small distance ($\leq$ 5 kbp), it is pruned. This is justified by that these transitions represent small indels without change in orientation, which are not responsible if the technique has a minimum resolution threshold. Finally, minimum weight bipartite matching is performed for the remaining transition edges between the two karyotypes, with a maximum allowed matching distance 200 kbp. Matching pairs are pruned, because they are considered similar SV edges. The matching distances are recorded for downstream analyses upon resolution. Finally, all residual non-segment edges are the di erential SV edges between the two karyotypes. We use K_r to denote the reconstructed chromosome, and K_t as the WT or true chromosome. Therefore, residual edges in K_t represent false negatives, and residual edges in K_r represent false positives.

SV Similarity Metrics. The similarity of each individual SV edge is compared via the count of K_t edges after the two initial pruning procedures, and the residual edges in K_t and K_r after final matching. TP = |inital edge| $|K_t$ residual|, FN = $|K_t$ residual|, and FP = $|K_r$ residual|. A Jaccard Similarity is computed for each cluster of compared simulated data to penalize both false positive and false negative events. For comparison against real data, only the recall is computed for each cluster, as the cytogenetic methods employed for the reference calls do not necessarily have small enough resolution or ability to catch de-novo balanced events.

S6.3 Copy Number similarity comparison and metrics

CN comparison is done by binning the whole genome (excluding prefix/suffix masked region) into spanning, non-intersecting bins of 50 kbp +/- 100 bp (exact size chosen to maximize the size of the last bin on the chromosome). Each bin is used to store the average CN within that region, separately for the K_t and the K_r.

Then, for each cluster, the chromosome groups are determined by the union of all the chromosomal origins of the segments in the cluster. Each cluster's CN bins are the subset of the total CN bins, to only include the chromosome groups within the cluster. A WT expected count is determined to each chromosome group by rounding the average bin CN from K_t (diploid for autosomes and XX or XY for sex chromosomes).

The values of the CN bins' of K_t and K_r are computed with CNs from all corresponding segments. If a bin has a gain or loss of more than 0.05 CN from the WT expected count, it is a marked as "CN gain" or "CN loss". Otherwise, it is marked as "CN neutral". This forms a paired CNV array where the Jaccard Similarity can be computed. The denominator of the similarity is the count of bins marked as CN gain or loss in either K_t or K_r, and the numerator is the count of bins where K_t and K_r agree on CN gain or loss.

S6.4 Addtional Functionalities for Downstream Analyses

Edges on both K_t and K_r can be labeled with additional input. For example, Table 2 was computed where each edge in K_t from the simulation was labeled with the Structural Variation event type, so a summary statistics was generated from the residual edge count of each event type.

When matching transition edges, the distances between the matched edges can be collected for

analysis. This can be further categorized by having K_t's transition edge labeled with the causal event type. A detailed analysis on OMKar's reconstructions' distances against the simulation can be found in Supplementary Section S4.

S6.5 Usage in validating real data reconstructions with previous cytogenetic records

A function was implemented in KarCheck to allow efficient input of a karyotype for the purpose of validating a reconstruction against previous cytogenetic records on that karyotype. For each karyotype, its aneuploidy (if any), each event's induced SV-breakpoints, and CN changes are taken as input for the validation. This information is sufficient to populate a full molecular karyotype, while KarCheck assumes the rest of the genome is WT.

For real data with cytogenetic records, the cytogenetic records were used as the "truth" karyotype (K_t) and compared against the reconstruction (K_r). CN gains called with microarray do not specify the structural breakpoints of the amplification. For these, we first assumed each amplification was a tandem duplication, and if the matching failed, we manually verified if the reconstruction contained the amplification as a segmental duplication. Additionally, previous cytogenetic techniques using microarray and staining did not fully capture every event on the genome, so the K_t was treated as incomplete, with potentially missed TP events. Therefore, for our final statistics of the comparison, we only computed the recall to verify if OMKar reconstruction included all the previously identified TPs.