# CMSC 754: Lecture (Supplemental)
# Final Review

**Final Exam Date:** Tuesday, May 12, 10:30am–12:30pm. The exam will be closed-book, closed-notes, but you are allowed **two sheets** of notes (front and back).

**General Recap:** We have introduced a number of the most fundamental aspects of the field of computational geometry. This includes:

**Geometric Structures:** such as convex hulls, triangulations and trapezoidal maps, Voronoi diagrams, and Delaunay triangulations.

**Duality and Arrangements:** We have discussed point-line duality and how to use this concept in conjunction with line arrangements to solve many computational problems involving points and lines.

**Geometric algorithmic methods:** We discussed a number of basic computational approaches for solving geometric problems, including plane-sweep, geometric divide-and-conquer, randomized incremental algorithms (and the associated concept of backwards analysis). We also discussed the use of linear programming to solve geometric optimization problems.

**Geometric approximation:** We discussed well-separated pair decompositions, coresets, $\varepsilon$-nets and samples, and applications.

**Motion Planning:** We discussed the notion of configuration spaces and configuration obstacles, and techniques like Minkowski sums and pseudo-disk systems for constructing and analyzing them.

**Principal Topics:**

**Convex hulls:** Convexity and convex hulls algorithms including Graham's scan ($O(n \log n)$), divide-and-conquer ($O(n \log n)$ time), Jarvis march ($O(hn)$ time), and Chan's algorithm ($O(n \log h)$ time). The latter two algorithms are examples of *output sensitive* algorithms, whose running time is a function of output size ($h$ being the number of vertices on the final hull).

**Line Segment Intersection:** We presented an $O((n + m) \log n)$ time algorithm, where $m$ is the number of intersection points. This introduced the important algorithm-design concept called *plane sweep*.

**Intersection of Halfplanes:** We presented an $O(n \log n)$ time divide-and-conquer algorithm. The key is a merging step that intersects two convex polygons using plane-sweep.

**Point-line duality:** We demonstrated a *dual transformation* that maps any point $(a, b) \in \mathbb{R}^2$ to the (nonvertical) line $y = ax - b$ and vice versa. We discussed how numerous affine (point-line) properties are preserved by this transformation. We observed that many problems involving lines/points can be converted to an equivalent problem involving points/lines through this transformation. As an example, we showed that computing lower and upper envelopes of a set lines can be reduced to the problem of computing the upper and lower convex hulls of the dual points.

**Polygon Triangulation:** We presented a simple $O(n \log n)$ time for polygon triangulation. This was based on two phases: (1) an $O(n)$-time algorithm for triangulating an $x$-monotone polygon and (2) an $O(n \log n)$ time algorithm to decompose a simple polygon into $x$-monotone pieces. Both algorithms are based on plane-sweep. (Note that there exists an $O(n)$-time algorithm for simple polygon triangulation, but it is quite complicated.)

**Linear Programming:** We presented a *randomized incremental algorithm* for linear programming in spaces of constant dimension $d$. The algorithm runs in $O(d!n)$ expected time, which is $O(n)$ when $d$ is a constant. The randomized incremental method involves inserting halfplanes at random and then updating the optimal point with each new insertion. We introduced the technique of *backwards analysis*, which analyzes the expected time of each step under the assumption that the *last* insertion is random. We explored a number of applications of (low-dimensional) linear programming.

**Trapezoidal Maps:** We introduced the useful planar decomposition of a collection of line segments called the *trapezoidal map*. We presented a randomized incremental $O(n \log n)$ expected-time algorithm for computing the trapezoidal map of $n$ line segments.

**Planar Point Location:** By building a *history DAG* for the trapezoidal-map construction, it is possible to answer point location queries in $O(\log n)$ time and $O(n)$ space. This data structure records the history of the trapezoids deleted/added by the construction algorithm. (Although the algorithm and space requirements from the construction are randomized and hold in expectation, we can run the algorithm multiple times until the worst-case space and worst-case query time are as desired.)

**Voronoi Diagrams:** We presented definitions and properties of Voronoi diagrams. We also presented Fortune's algorithm, and $O(n \log n)$ time plane-sweep algorithm for Voronoi diagrams. Fortune's algorithm uses plane-sweep, together with the clever idea of the beach line to achieve efficiency. We observed that combining Voronoi diagrams with point location yields an efficient data structure for nearest-neighbor queries.

**Delaunay Triangulations:** We presented definitions and properties of Delauay triangulations. This is the dual-graph of the Voronoi diagram. We showed that the Delaunay triangulation is a subgraph of the Euclidean minimum spanning trees, it maximizes the sorted angle sequence, and it is a spanner, meaning that shortest paths in the triangulation are within a constant factor of the Euclidean distance between points.

**Line Arrangements:** We presented a simple incremental construction algorithm for line arrangements in the plane, and proved its $O(n^2)$ running time with help of the *Zone Theorem*. We showed that line arrangements together with duality could be used for answering many problems having to do with lines and points in the plane. These problems were solved either by explicitly constructing the line arrangement or (more space efficiently) by applying plane sweep (or topological plane sweep), which run in $O(n^2)$ time, but use only $O(n)$ space.

**WSPDs:** We introduced the notion of an $s$-well separated pair and showed that all the pairs of any $n$-element point set in $\mathbb{R}^d$ can be encoded as a set of $O(s^d n)$ well separated pairs. We provided a number of applications of WSPDs for geometric approximation problems, such as computing the farthest pair, the MST, spanner graphs, and the closest

pair (exactly). Recall that a *t-spanner* is a graph with the property that the for any pair of points there exists a path whose length is within a factor $t$ of the Euclidean distance between the points.

**Coresets:** We presented the notion of coreset. Given a problem that involves some sort of numeric output (e.g., a geometric optimization problem or evaluting some statistic of a point set), a coreset is a subset of the point set such that the objective function is roughly the same on the subset as the original set. We presented algorithms for computing coresets for the directional width problem.

**Geometric Sampling:** We introduced the concepts of range spaces, $\varepsilon$-samples, and $\varepsilon$-nets. We showed how the concept of VC-dimension could be applied to bound the size and complexity of typical geometric range spaces.

**Motion Planning:** We discussed the notions of work space (where the robot resides) and configuration space (where the robot is modeled as a single point). We showed how obstacles in the work space are transformed into obstacles in the configuration space. This led to investigation of the Minkowski sum and systems of pseudo-disks.