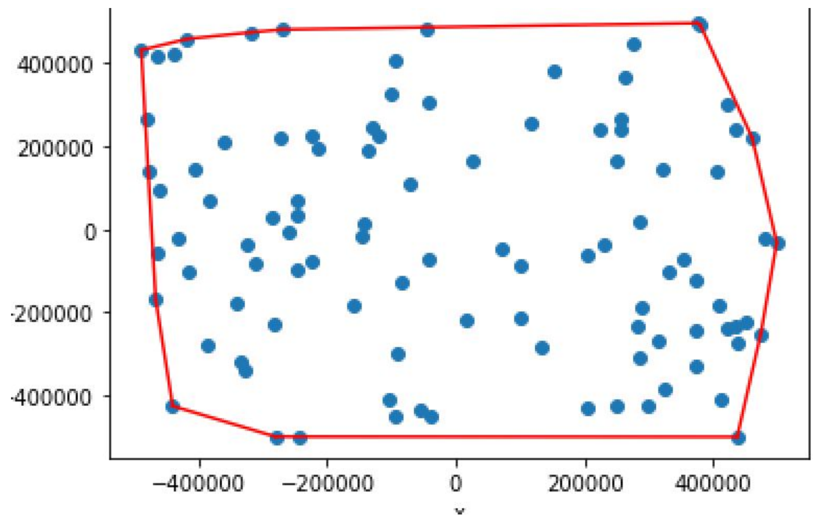
Convex Hulls, Skylines and Onion Convex Hull

CSE 5311-006: PROGRAMMING PROJECT REPORT

Team Members:
Harish Verlekar
Abhishek Modi
Bhavishek Malik
Spardha Gupta

Convex Hull: A polygon where, for any two points p and q inside the polygon, the entire line segment pq lies inside the polygon.
Example: We can define a convex hull by taking a thought example. Try imagining points as sticks coming out of a plane, take a rubber band, stretch it around the sticks and release. The rubber band snap around the sticks, forming a boundary-like for all the sticks. The area enclosed by the rubber band is called the convex hull.
Therefore, the convex hull of the polygon is the minimal convex set wrapping the polygon.
Properties of convex hull:
1. Coordinates monotonically increase or decrease.
2. The edges slopes monotonically decrease.



Algorithms/Ways to implement convex hull:
1. Graham Scan
2. Jarvis March
3. Divide and Conquer

Graham Scan Algorithm:
1. Graham Scan algorithm begins with selecting left-bottom most point (min x and min y coordinate).
2. Then it sorts all points in increasing polar angles.
3. Initialize hull with - P (anchor point), first element in sorted list and iterating each point in sorted list.
4. If traversing to it from two prior points, check for left turn (counterclockwise).
5. If it takes right turn (clockwise) delete points until adding a new point results in left turn.
6. Keep adding points to convex hull until it reaches back to the anchor point P.

Data Structure: stack / list of points
Time Complexity: **O(nlogn)**

a) Finding left-bottom most point = O(n)
b) Sorting of points = O(nlogn)
c) Pushing and popping from stack = O(n)
Total: T(n)= O(n)+O(n)+O(nlogn) = O(nlogn)
Benefit: It calculates polar angles for all points once and sort them.
Tradeoff: The points it initially pushes in the stack does not guarantees to be on the final convex hull.

Jarvis March Algorithm (Gift wrapping algorithm):

1. Jarvis March algorithm begins with selecting leftmost point (minimum x coordinate).
2. Calculate polar angles of all other points from that point.
3. Choose and connect with the smallest angle such that it makes move in counterclockwise direction.
4. Repeat steps 2 and 3 while we don't come back to the initial point.

Time Complexity: **O(nh)**
Data Structure: List
Where h is the number of points on convex hull.
Note: Jarvis March algorithm works faster than Graham Scan when h < logn.
Benefit: The points added in the list are guaranteed to be on the final convex hull.
Tradeoff: It calculates polar angles for all remaining points each time from a selected point.

Divide and Conquer Algorithm (Tangent Approach):

1. It begins with selecting minimum y coordinate (min y) point and maximum y coordinate (max x) point.
2. Join the two points to form a line segment 'L' which divides the whole set into two parts. Approach is applied on both the sets recursively.
3. For each part select a point P which is at maximum distance from L. This forms triangles with min y and max y and points inside the formed triangles can never be part of the convex hull).
4. Step 3 would divide the problem into further two subproblems, and the points residing outside the triangle - formed by line segments P and min y, and P and max y- are remaining problem set of points.
5. Repeat step 3 while there are no points left outside the triangular or line regions.
6. Join the upper and lower tangents of the formed convex hull to merge and form final convex hull. The final tangents should be common tangents for all the points on convex hull.

Time Complexity: **O(nlogn)**
Data Structure: List
Benefit: This algorithm runs efficiently for large data sets.

Skylines:
Given a set of points p1, p2,....pN, the skyline is a subset of points P such that each point in P cannot be dominated by any other point from the dataset.
For dominance reference: a point p1 dominates p2 if and only if both x and y dimensions/coordinates of p1 are better than p2. Such points are called dominant points and rest of points are known as maximal or skyline points.
Example: Point <5,10> dominates <3, 8> but not <10,8>.

Skyline can be implemented using:
1. Naive approach
2. Convex hull approach

Naive Approach for Skyline:

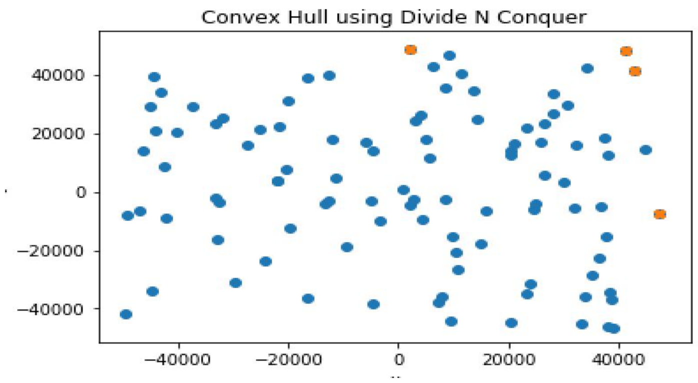
1. Naive approach uses brute force algorithm to generate skyline.
2. Select any point and compare it with every other point.
3. If the point p is dominated by any one other point r, remove p from the dataset and add p to rejection list.
4. Repeat steps 2 and 3 for all remaining points.
5. Subtract rejection list from the original data set and, remaining points will be skyline points.

Time Complexity: **O(** n^2 **)**

Convex Hull Approach for Skyline:

1. Build convex hull of the given dataset.
2. Find a point with maximum y (topmost point) and a point with maximum x (rightmost point).
3. Find all points between these two points and for each point check the condition of dominance.
4. The resulted dominant points create skyline.

Time Complexity: **O(nlogn)**

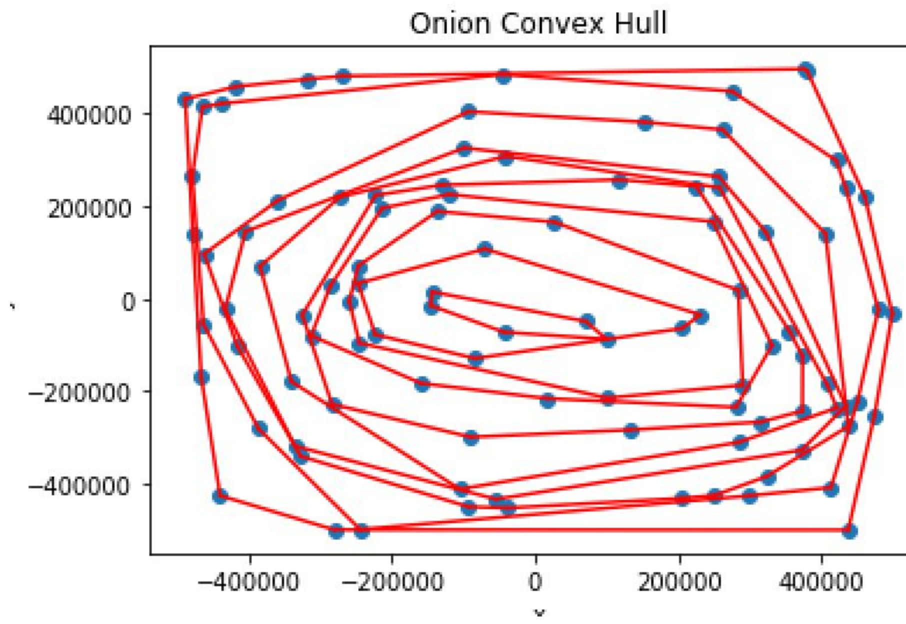


Onion Convex Hull:

Onion Convex Hull algorithm computes the concentric convex hulls recursively.

1. It starts by computing convex hull for the given dataset, using any efficient convex hull algorithm.
2. The points which are on the generated convex hull are removed from the dataset.
3. Again compute next convex hull for the remaining dataset.
4. Repeat the above steps while n (size of dataset) < 3, in which those two points would be considered as a hull itself.

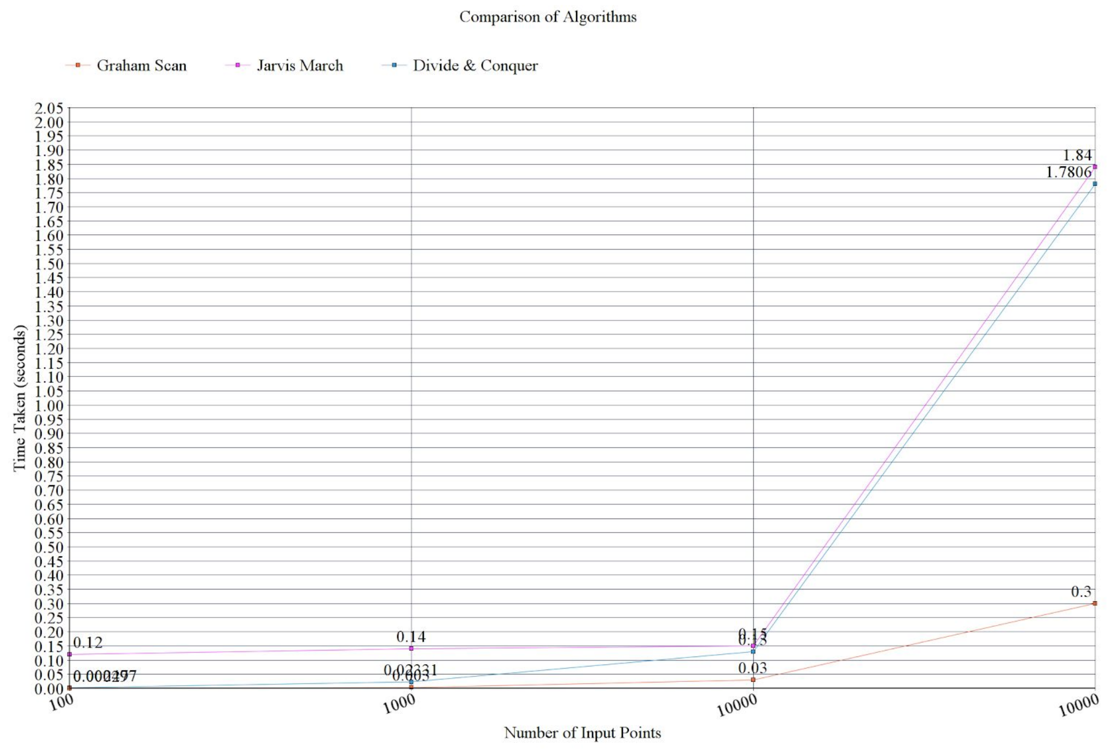
<u>Time Complexity:</u> **O(** n^2 **)**



Experiments:

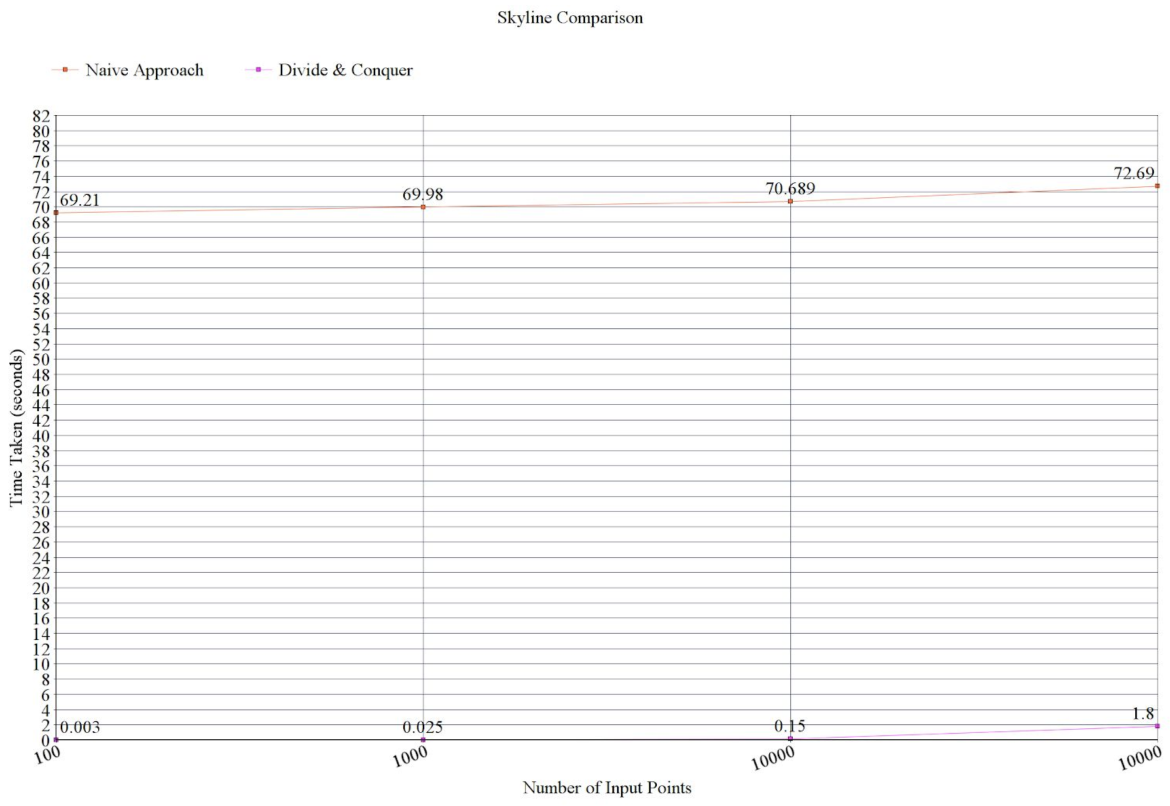
1. Comparing performance of different algorithms for different inputs

Input Points	Graham Scan	Jarvis March	Divide & Conquer
100	0.00029	0.12	0.002477
1000	0.003	0.14	0.02331
10000	0.03	0.15	0.13
100000	0.3	1.84	1.78061



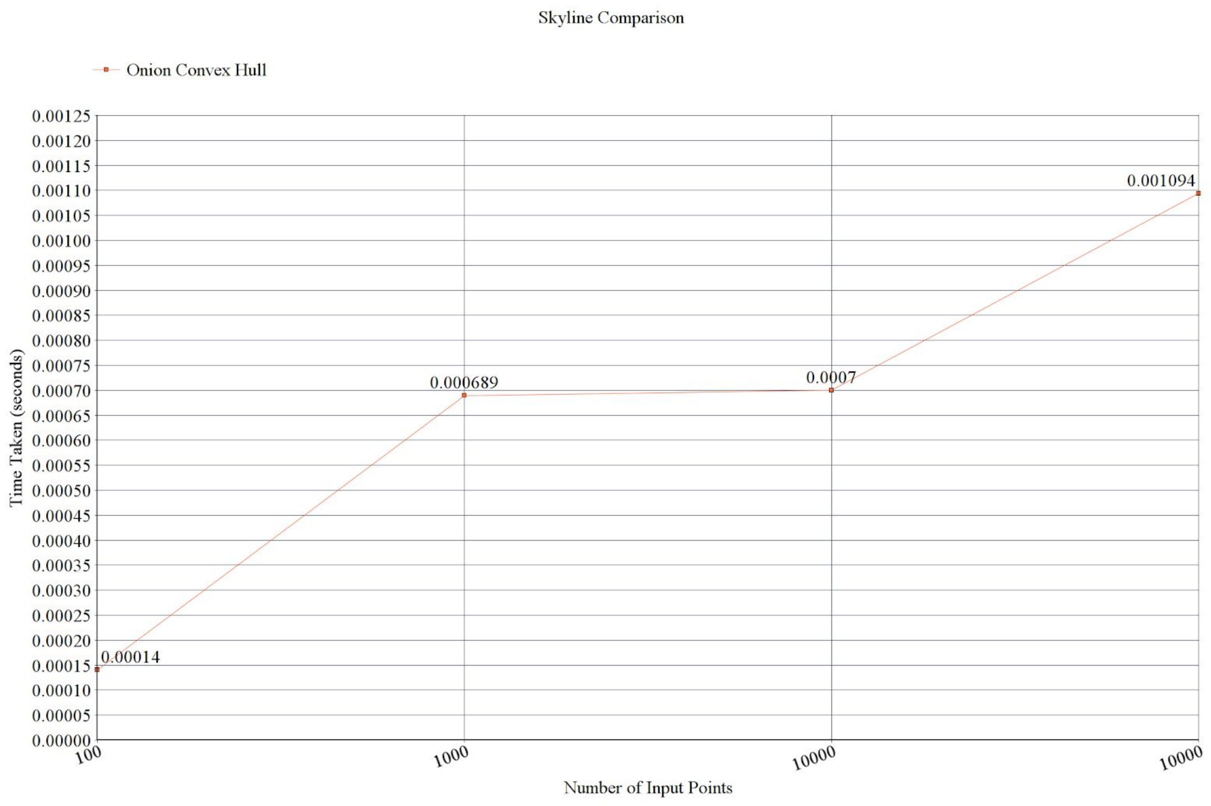
2. Comparison of different approaches for skylines.

Input Points	Naive Approach	Convex Hull
100	69.21	0.003
1000	69.98	0.025
10000	70.689	0.15
100000	72.69	1.8



3. Comparison of different inputs for onion convex hull

Input Points	Onion Convex Hull
100	0.00014
1000	0.000689
10000	0.0007
100000	0.001094

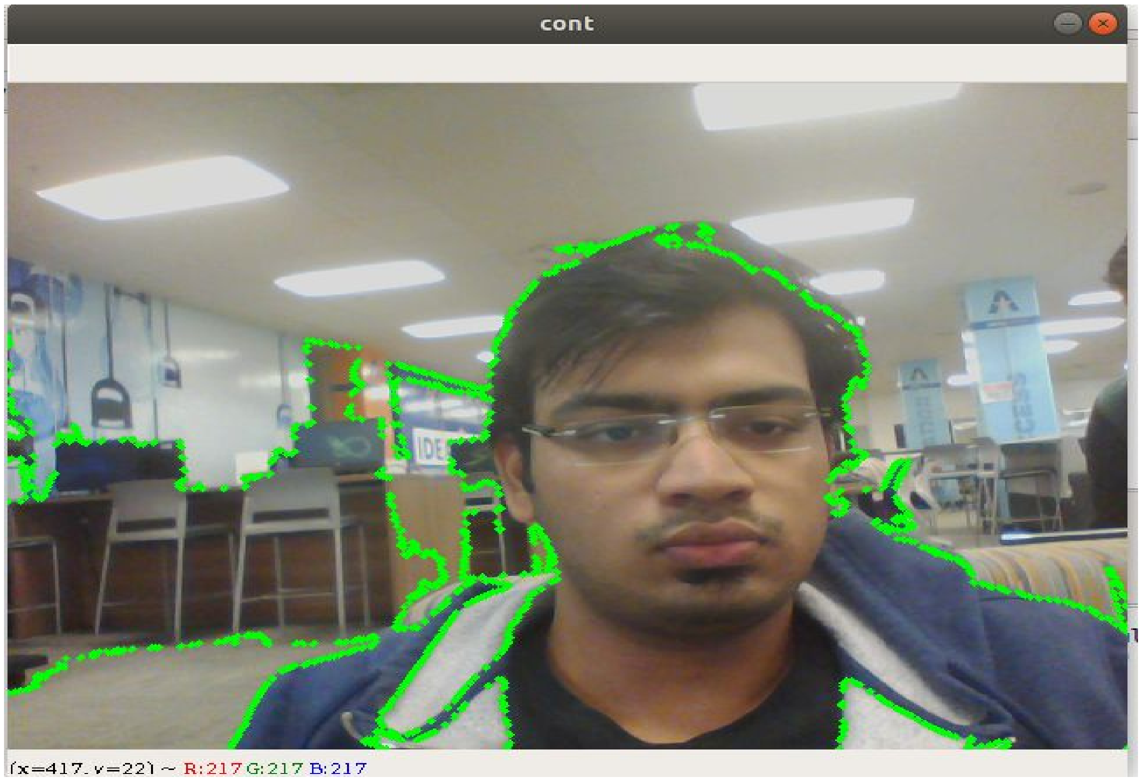


INNOVATION:
Computing Convex Hull from Image Contours using Graham Scan

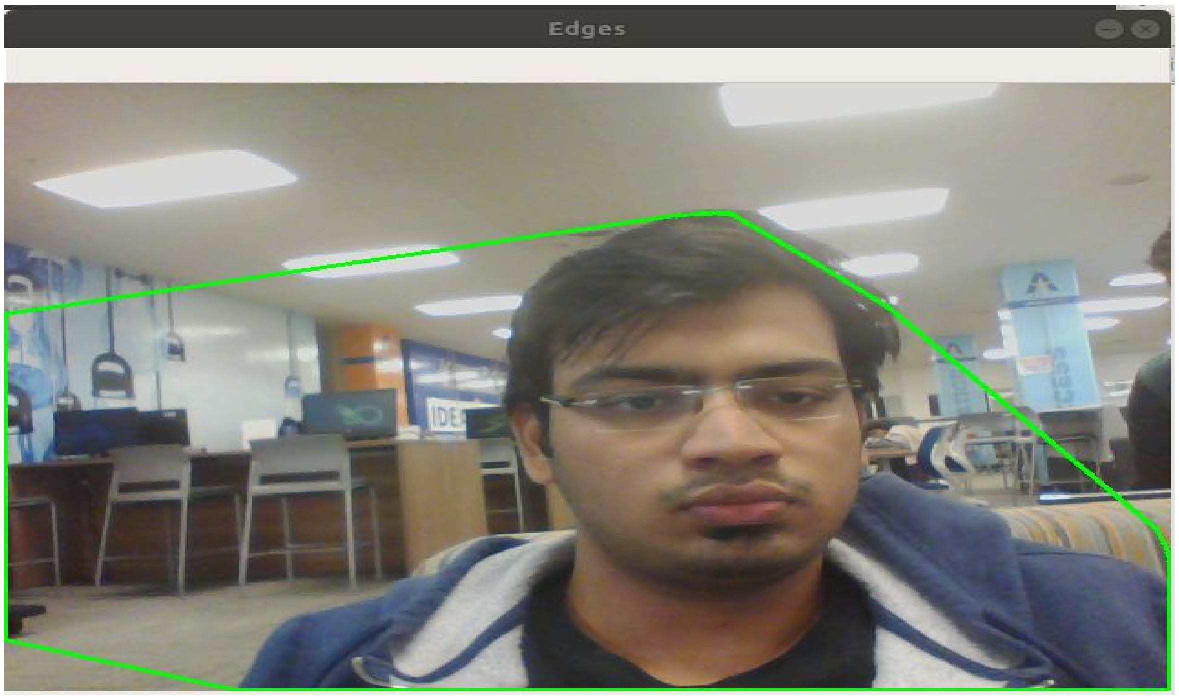
We have computed the convex hull using Graham Scan algorithm on a set of contours obtained by thresholding on a image. The following steps were followed.

1. Obtain video frames.
2. Convert the video frame to a gray scale image.
3. Apply Otsu's thresholding and convert the image to binary.
4. Find contours from the binary image. These contours act as coordinate points.
5. Feed those co-ordinates to the Graham Scan algorithm which was implemented before.
6. The output convex hull is then plotted on the image.
7. Repeat steps 1 to 6 for all video frames.

Results



Contour Points



Computed Convex Hull

References:

1. https://brilliant.org/wiki/convex-hull/
2. www.geeksforgeeks.com