Minkowski Sum of Convex Polygons

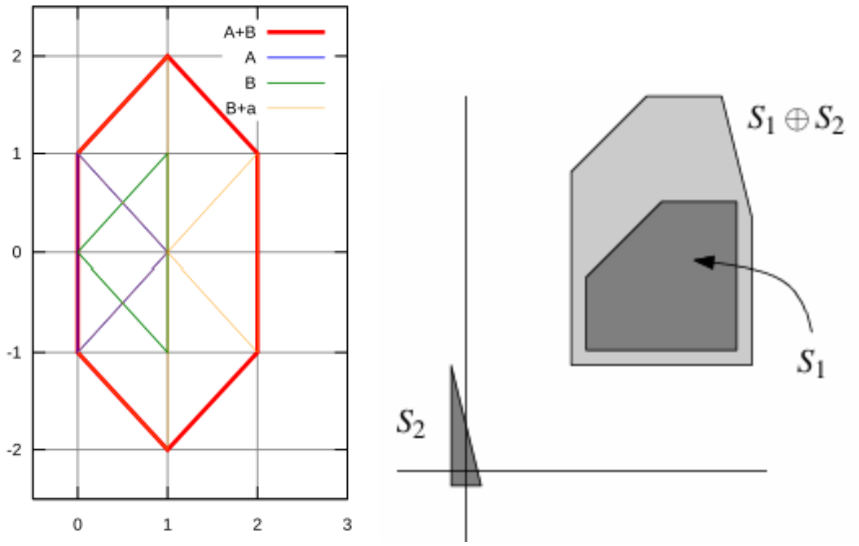
Executive Summary

Introduction / Problem Statement:

The Minkowski sum of convex polygons is defined as follows: given 2 sets A and B of points, the Minkowski sum is $\{a + b \mid a \in A, b \in B\}$, where, specifically in this case, the sets consist of polygons and their interiors. This operation is notated by $\oplus$. This operation is often used in computer-aided design and manufacturing, collision detection in physics engines, and robotics. Specifically for robotics, a key operation in planning the motion and path of a robot, aptly named robot motion planning, is being able to detect possible collisions with objects. Manually computing possible collisions at every possible point in order to compute a valid path is too slow and grows in complexity as the space increases. Thus, there is a need for an algorithm that can be used to solve this problem quickly. The Minkowski sum of convex polygons can be cleverly used to do this, given that it is implemented efficiently. Naive implementation of this idea would be $O(n^2)$ through manual computation of the addition of each combination of the points. However, since the sum is linear to the size of the initial polygons, the target runtime of the Minkowski sum calculation is linear, or $O(n)$.

Overall Intuitive Approach / Solution:

The key to solving this problem quickly is understanding certain properties given the constraints of convex polygons. Visualization of what the Minkowski sum looks like can be seen in the following pictures:



Intuitively, it can be seen that if a + b (as defined by the problem definition in the introduction) is a boundary point of A ⊕ B, a is a vertex and b is on an edge, or the other way around. Thus, it can be concluded that the Minkowski sum can be computed by considering only the convex hull vertices of the two convex polygons. This allows for the observation that we just need to figure out the order in which we add the edges.

Consider when the edges of the two polygons are ordered clockwise. It can be seen then that the edges are also ordered by polar angle. Thus, the two sets of edges can be merged while maintaining polar angle order in O(n) time. The edges of the convex polygon defined by P + Q can then be constructed by summing consecutive edges. In other words, the 2nd vertex of the edge attaches to the 1st vertex of the next edge.

Implementation:

This approach can be implemented without frequent summation of vectors, but instead through the iteration of the vertices of the sets. This implementation assumes that the polygons are given in counterclockwise order, as this is the common output of a convex hull algorithm. If not, this can easily be done first with a sorting algorithm with a custom comparator (using atan or cross-products).

The vertices of each polygon first need to be reordered with the lowest y-coordinate as the first vertex (picking the smaller x-coordinate if multiple have the same y-coordinate). This sorts the vertices and sides of the polygons by polar angle. We then create two pointers (set to 0), i and j, pointing to the vertices of polygon P and Q, respectively. Repeat the following steps while the pointers do not exceed the size of their respective polygons:

Add the sum of the current vertices of the polygons to your answer, P + Q (P_i + Q_j)

Increment the pointer of the current edges of both polygons ($P_iP_{(i+1)}$ and $Q_jQ_{(j+1)}$) with the smaller polar angle. If they are equal, increment both.

Summary of Programming Challenge:

The programming challenge asks the student to figure out if a robot can move in a certain direction without colliding with any objects. The environment contains various obstacles that can potentially block the robot, and the robot can be placed in any location in the given environment. The obstacles can be convex or non-convex polygons, but non-convex ones are split into convex polygons in the inputs.

The main objective of this programming challenge is to allow students to not only implement the Minkowski sum algorithm but also to be exposed to how it is used in common applications. In particular, we wanted the students to figure out how to use the algorithm in order to compute if two polygons intersect, as this is a very important concept used in collision detection for robotics and motion planning. In addition, introduces the concept of non-convex polygons being able to be split into convex polygons as well, which is a common way of using Minkowski sum in a more general scenario. There is an additional extension apart from just a simple collision between two objects as well, which we added because we wanted the programming challenge to incorporate some logical thinking and creative usage of the concept, without making the problem too challenging. It also should be noted that the difficulty of the challenge is based on a student who has taken advanced algorithms, as the solution uses computational geometry ideas from concepts we covered in class. The challenge accomplishes these goals by forcing the student to use the minkowski sum algorithm to find collisions quickly, but because the algorithm can't just naively check against each obstacle at each point of movement, the student has to come up with a clever solution to check against the obstacles for all of the movement in that direction quickly.

Key Ideas for Solving Programming Challenge:

For the first part of the problem, computing if two polygons P and Q intersect, the key idea is that we can find the Minkowski sum of polygon P and the reflection of Q through (0,0), -Q (just negate both x and y of every point of Q). We can then simply check if (0,0) is inside or on the boundary of the polygon. If it is, the polygons collide; otherwise, they don't. Intuitively, this works because we can only be at point (0,0) from a summation of 2 points if both x and y from the points are the same, and if the points are the same in the polygons, that means they collide. Figuring out this idea is one of the main challenges of the problem. While simple to implement, coming up with the concept may be somewhat difficult, but ultimately doable.

The second key idea of the problem is figuring out how to detect collision across the whole distance in fewer checks. This can be done by understanding that since we are moving in just a single direction, we can just translate the robot to the end position. From there, we can create a new convex polygon by using the start and end positions of the robot, which contains all the positions of the robot within it. From there, we can simply use the algorithm from before to check for collisions with any obstacles with this polygon.

We think that these changes are interesting because they really get the student to understand the algorithm and utilize it in creative ways. We think that the changes are not too difficult because the ideas are not fundamentally hard to implement - it really more are really simple extensions to the algorithm, but it may just need some thinking to figure out. We believe this twist is on par with the homeworks in the class throughout the semester, as many required students to think outside the box with the given algorithm or data structure. It can also be noted that there are other possible ways of solving this problem, such as extending a line from each point of the robot in the direction it is facing and doing a line intersection with each obstacle edge. However, we believe our solution to be faster, and so these other solutions may be restricted with a runtime limit on testcases.

Given N, the total number of points in all of the obstacles, and M, the number of points in the robot, the runtime is O(2N + MlogM), as we iterate through the obstacle edges twice - once in the minkowski sum and once when checking if (0,0) is on or contained in the sum. M is bounded by MlogM because we run a convex hull on the start and end of the robot, which requires sorting (it's technically (2M)log(2M) since it's the start and end).

Conclusion:

The Minkowski sum of convex polygons is a useful algorithm for computer-aided design and manufacturing and collision detection. While the naive implementation takes $O(n^2)$, the calculations can be sped up to linear time with key observations regarding the edges of the resulting sum and its relation to the original polygons. This results in an O(N) algorithm that is a relatively simple 2-pointer solution given 2 polygons ordered counter-clockwise.

The programming challenge associated with this topic allows students to explore the real-world applications of the algorithm by using the Minkowski sum in order to detect collisions between convex polygons. This, on top of a creative extension, allows students to explore creative usages of the sum.

While a seemingly random problem at first, the Minkowski sum is a widely used operation across fields from math to robotics. This specific application of it to convex polygons is particularly valuable to the field of robotics and will likely continue to be used in the near future.