RAINMAN

Peshwas

2nd September, 2023

1 Algo for 2-D

We've used multiple initialization-based gradient descent optimization for optimizing the number of rain particles for a given **rotation angle** , **speed**.
For a given speed and rotation angle, we:

- Firstly rotated all the polygon points by the rotation angle. Now, along the path on a given line segment, we first determined the angle and velocity of the rain relative to the polygon. Then we calculated the projection of the polygon with respect to that angle.
- Now, number of rain particles is equal to :

  $$\frac{intensity \times projection \times Srel \times Vrel}{speed}$$

  where Srel is the distance between two points forming the line segment and Vrel is the relative rain velocity
- We now add up this quantity for all the line segments and minimize the quantity obtained.

2 Algo for 3-D

We used multiple initialization based gradient descent optimization for optimizing the number of rain particles for a given **rotation angle** about x-axis and y-axis , and **speed**.
For a given speed and rotation angles along the two axes:

- We first rotated all the polygon points by that angle. Now, along the path on a given line segment, we first determined the angle and velocity of the rain relative to the polygon. Then we calculated the projection of the 3-d structure along the direction of rain on the plane. Then, we calculated the convex hull of all the projected points on the plane and determined its area. Let this quantity be **p**.

- Now, number of rain particles is equal to :

$$\frac{intensity \times p \times Srel \times Vrel}{speed}$$

  where Srel is the distance between two points forming the line segment and Vrel is the relative rain velocity

- We now add this up for all the line segments and minimize the quantity obtained.

3 Optimization

For optimization, firstly we ran an iteration of **Nelder-Mead** (non gradient-descent based optimization) optimization technique. Then, for multiple rotation angles as initial parameters, we ran **L-BFGS-B** (gradient-descent based optimization) optimization.

4 Implementation

We have hosted the app on a web server. The frontend and backend are hosted separately. The frontend makes an API call to the the backend with the object vertex data, rain parameters, and path points, and displays the resulting answer from the backend visually. To visualize the result, the **Canvas API** is used. We have also used react in making the frontend for ease of development. In the backend, we used **Flask**. The algorithm has also been written in **Python**.