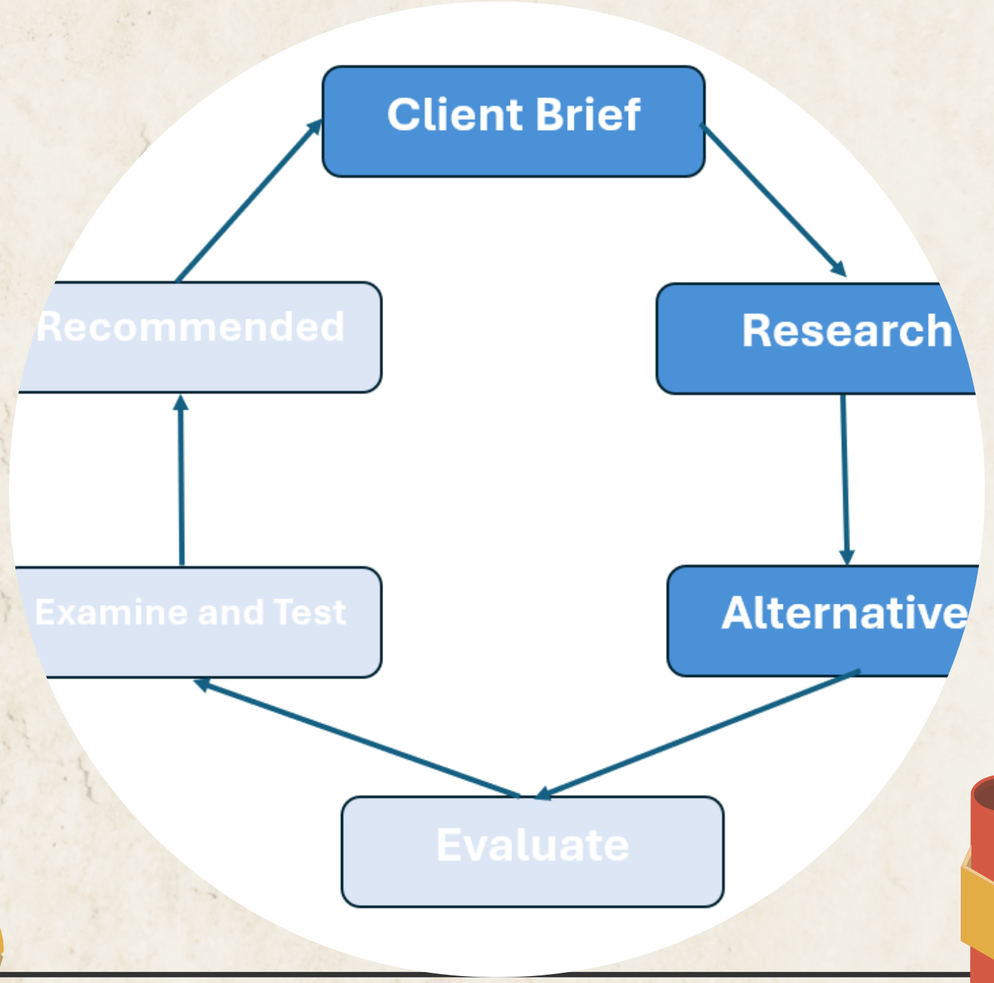
HACKATHON

Team 11: Junction Master
Yuxuan Zhang
Son Dang
Allwin Vincent
Nam Tran

01 Brainstorm process

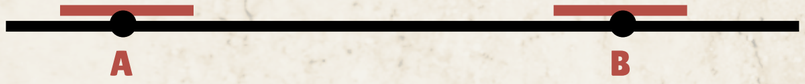


Identify the problem

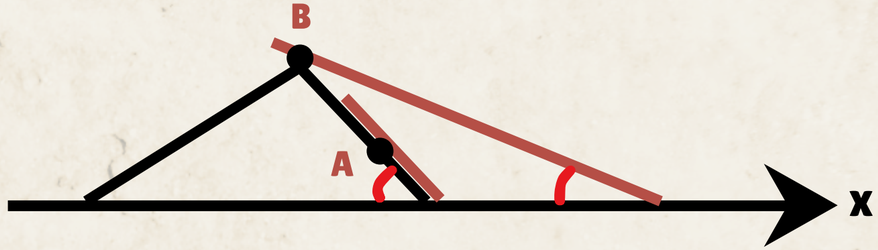
Finding the criteria for measuring efficiency of a curves

Perfect score achieved	Straight line
Worst score achieved	Sharp corner

Theta corner(slope of the tangent line): θ



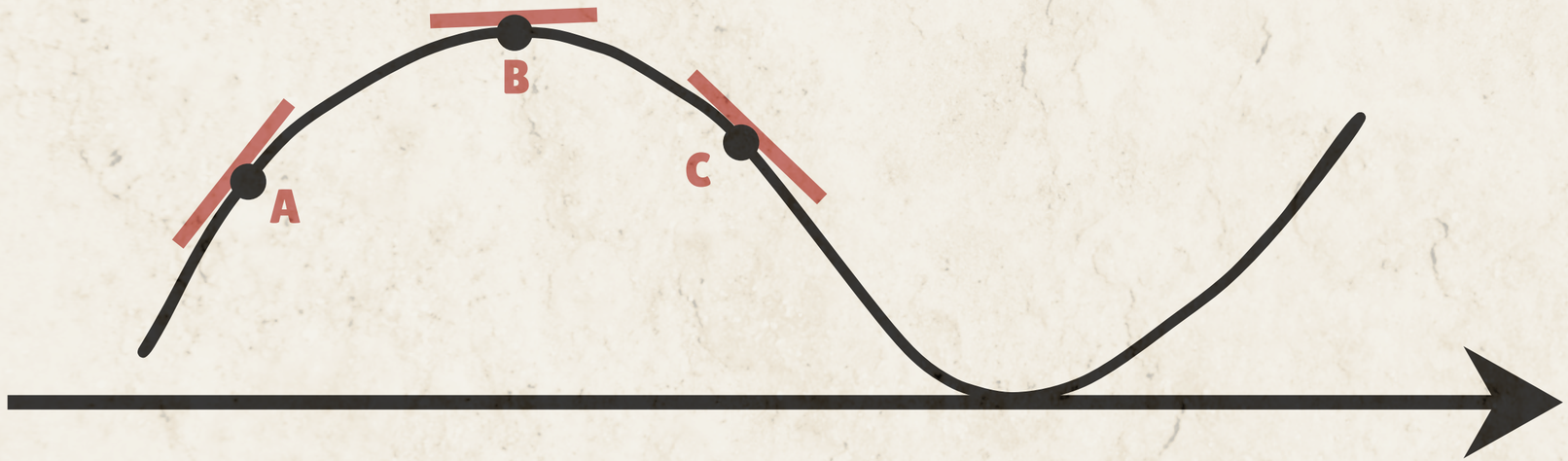
θ(A) = θ(B) => Δ θ(AB) = 0 = Δ θ at any points



θ(A) ≠ θ(B) => Δ θ(AB) >> 0

Prediction

Finding the criteria for measuring efficiency of a curves



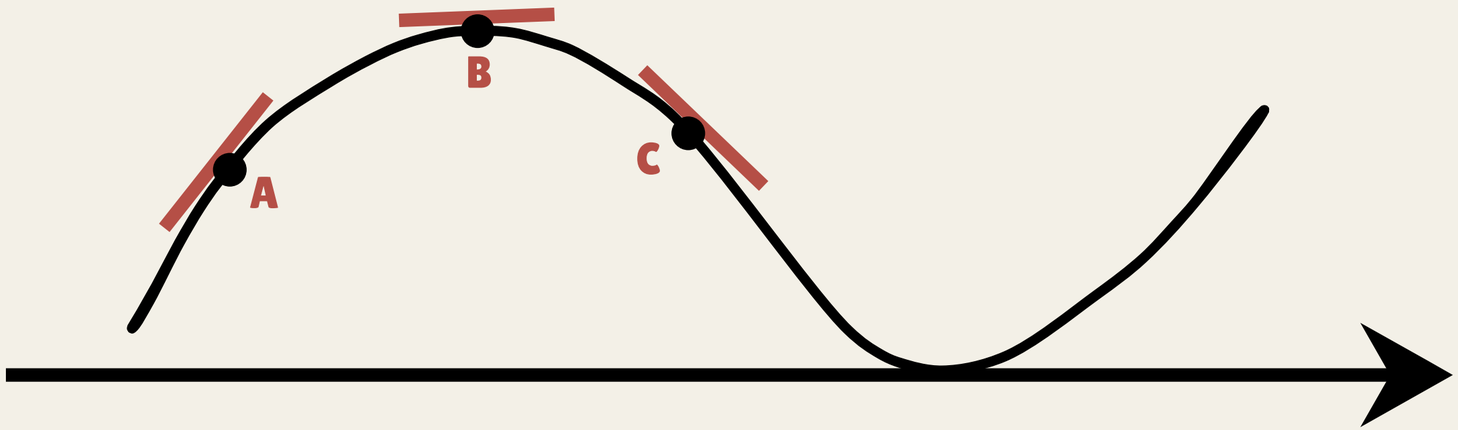
Good curve happens when: $\theta(A) - \theta(B) \approx \theta(B) - \theta(C)$

Bad curve happens when $\theta(A) - \theta(B) \neq \theta(B) - \theta(C)$ and $|\Delta\theta| >> 0$

Note that A, B, and C are very close points on the curve
=> Our prediction is that if the rate of change of delta is small, the curve will score high (good) on Rio Tinto's simulation

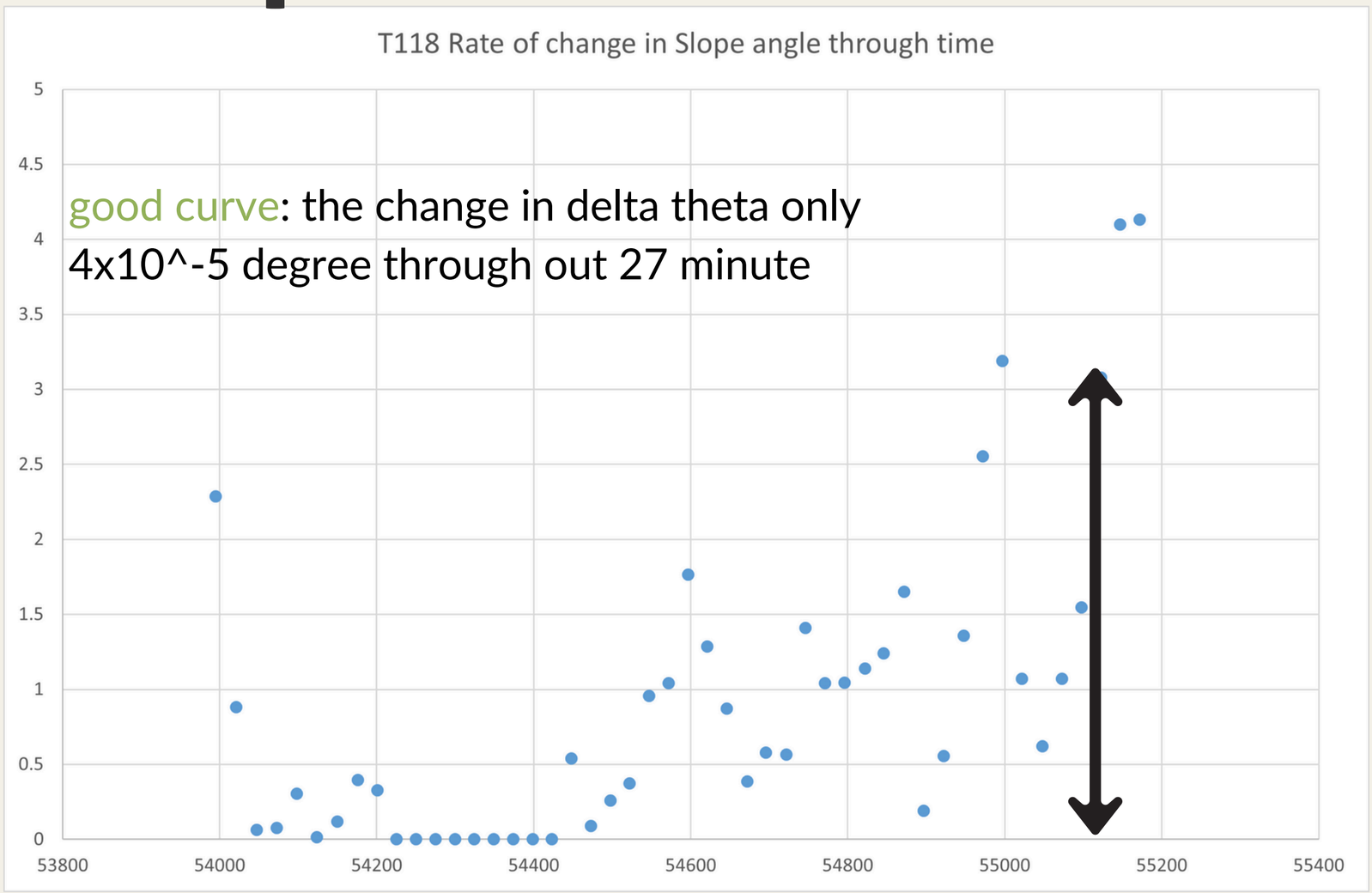
Prediction

Finding the criteria for measuring efficiency of a curves



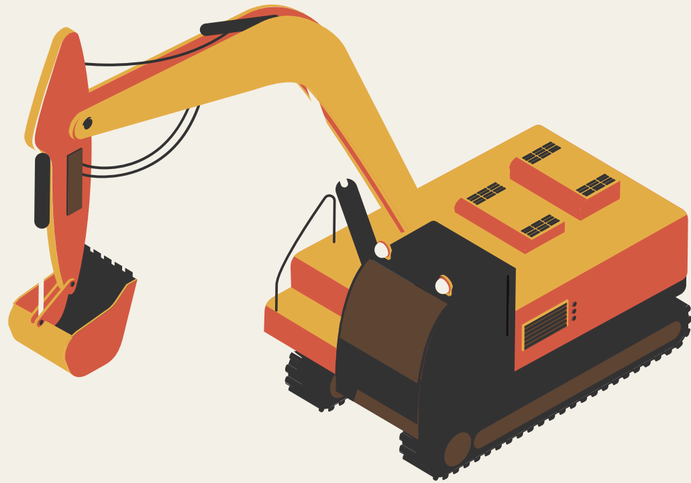
- To measure θ, we use Python to calculate arctan(y/x) with x and y provided in the data sets.
- To calculate the rate of change of θ, we use Python to compute θ at every point on the curve and store each value of $\Delta\theta = \theta(i+1) - \theta(i)$ in a list.
- We then plot these values on a Cartesian coordinate system where the y-axis represents the magnitude of Δθ and the x-axis represents the time in seconds. A good curve should have a plot close to a horizontal line, indicating that the change in angle is very consistent with no subtle variations.

Example

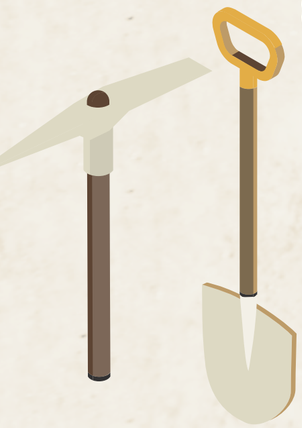


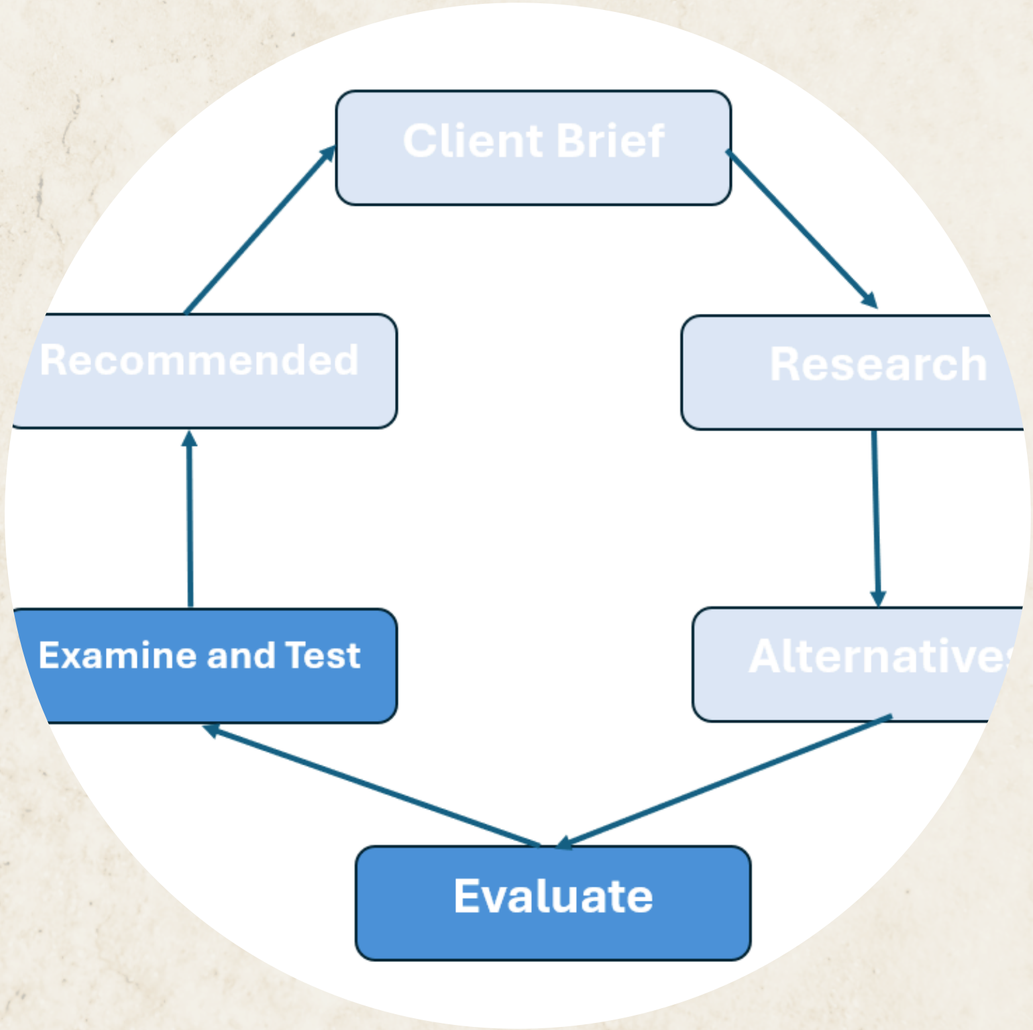
x-axis : time in second

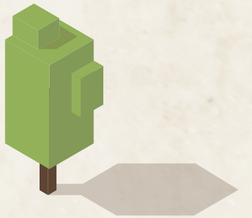
y-axis: delta theta x10^5



02 Technical explain







1. Curvature & Curvature rate

- Computes the curvature through the change in theta mentioned in Section 1.
- Further idea to Curvature rate
- Acted as a feature in machine learning model
- Expects uniformity in changes in curvature
- **df['CurvatureRate'] = df['Curvature'] / df['TimeDiff']**
- Rate calculated by rate of change of curvature

2. Label Generation

- Classify each curve as '**good**' or '**bad**' based on criteria.
- Label == **1** means good; Label == **0** means bad
- 4 criteria in total: curvature in combination with 3 constraints in the dataset

3. Machine Learning model

- **RandomForest** model used for generating optimal curves.
- Trains data using features: X, Y coordinates, curvature data and curvature rate
- Optimize curves to satisfy all classification criteria and maximize quality score
- For details, refer to the project codes submitted!

4. Optimal Curve Generation

- Score based on distance to edges, smooth entry/exit, curvature, and curvature rate
- Explicit formula for score is defined in code
- Might exist built-in score function but I wrote customised one (refer code repository for details)

- Use **RandomForest** model to generate curves with features mentioned before
- Evaluate generated curves based on quality score.
- Chooses the curve with **highest score** as **optimal curve**

5. Sending Data from Front End to Back End

- JavaScript Function to Send Data
- Create a function to send extracted data points to the Flask back-end
- Use fetch with a POST request to send data in JSON format.

6. Button for Curve Classification.

- Add a button in the HTML to trigger curve classification
- When clicked, extract curve data points and send them to the back end for classification

7. Button for Generating the Optimal Curve

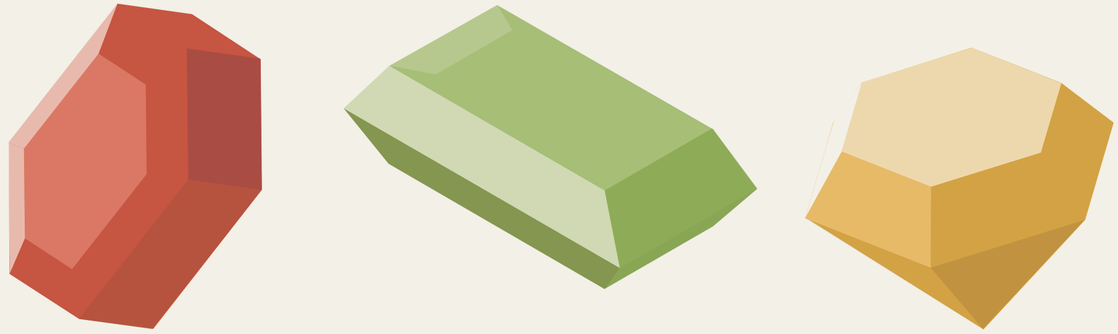
- Add a button in the HTML to trigger the generation of an optimal curve
- When clicked, send a request to the back end to generate the optimal curve
- Use the response data to visualize the generated curve on the SVG canvas
- Again, refer to code repository for details!

8. Limitation

- Simplified assumptions in curvature calculation
- Data inconsistencies with road coordinates
- RandomForest model used for classification may **not** capture all nuances
- Limited by computational resources and data available during the hackathon
- Deals with a simplified model -- didn’t consider one way intesection scenario

9. Further Research

- Experiment with different machine learning algorithms to improve classification and optimization.
- Develop a more sophisticated scoring mechanism for optimal curves
- Include additional criteria like uniform curvature change and real-time constraints (current demo **fails** to achieve this)
- Implement logic to handle **one-way intersections**



Thank you for listening