ROUTE PLANNING FOR OPTIMIZED ON TIME DELIVERY

VOLUME ESTIMATION

- An **easy-to-install** tool that can be integrated with a **conveyer belt**
- **Realsense** v/s **Kinect**
- **Image segmentation** using bitmask generated from variation in **depth map**
- **2.5D Point cloud flattening** to find the exact area of contour as seen in top view.
- Object classification: **ResNet50** and **PointNet**, into 4 classes
- Employ the respective volume estimation method
- Dead weight computation using **load cell** and **HX711** module
- Erroneous object segmentation based on deviation in dimensions from median values

ALGORITHM

The algorithm can be divided two clear parts:

- **Initial clustering/routing**
  - Path Cheapest Arc Insertion
  - Cost function: Distance
- **Objective function minimization for obtaining the routes**
  - Brute Forcing with bitmask dynamic programming if the number of nodes in all the initial clusters < 13
    - Ability to get exact solution
  - Otherwise, state-of-the-art algorithms:
    - Local Search for minimizing the objective function
    - Metaheuristics - Guided Local Search or Tabu Searching for escaping local minima

OBJECTIVE FUNCTION

$$f(x) = D(x) + \sum_{i \in N_a} T_{penalty} * T(i) + \sum_{i \in N_d} Drop_{penalty}$$

Here,
Na: Accepted points & Nd: Dropped Points
T(i): max(0, Exp. Delivery Time(i) - EDD(i))
D(x): Total distance traveled in solution x
T_penalty, Drop_penalty: Penalty Constraints

DYNAMIC PICKUPS

$$f(P) = argmin_{(X,Y)} \; extra\_time(X,P,Y) + T_{penalty} \cdot time\_window\_penalty(Y,P),$$

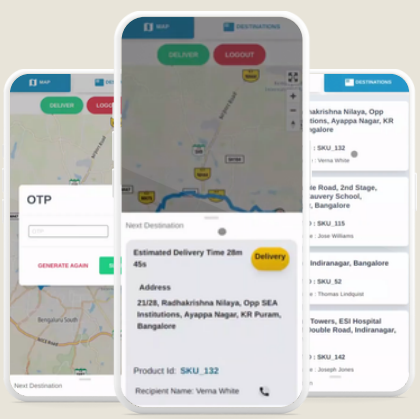
$$extra\_time(X,P,Y) = distance(X,P) + distance(P,Y) - distance(X,Y)$$

$$time\_window\_penalty(Y,P) = \sum_{i \in N_Y} t_i$$

- Here, a greedy approach was followed using the objective function given above
- The function was calculated for insertion of the pickup point P between every two consecutive points (X,Y)
- The final position of P was calculated w.r.t the minimum value thus obtained.

Driver Productivity Inclusion

- We assumed that the ratings of the drivers would be provided
- The ratings are then normalized (converted to a value between 0 and 1)
- Then all distances and times are multiplied by a factor of (2 - new_rating)



APPLICATION INTERFACE

- Orchestrated completely using docker swarm
- Mobile-first web application for easy updates
- Facility to add and delete points in between tours
- Ability to manually reroute tours
- Uncluttered and clear directions to delivery locations
- Routes considerate of real-time traffic & route quality
- Easily scalable to 10000 concurrent users
- Fraud prevention measures in the form of OTP
- Performance-based validation of drivers