Event Matching Algorithm

Situering

Dit bestand bevat de uitleg over hoe we testen of een punt op een route ligt die gedefinieerd is door waypoints. Het hele algoritme kunt u vinden de map "backend/src/main/java/be/ugent/backend/businesslayer/businnesworkflow/akka/algorithms".

Om te weten of een event zich heeft voorgedaan op een route van een user, moeten we dus dat event matchen met één van zijn routes. Dit bleek echter geen sinecure. Dit bestand geeft uitleg over het algoritme dat wij ontwikkeld hebben om dit probleem op te lossen.

Algoritme

Basis

Eerst geven we wat high level uitleg over het hele matching proces.

Gegeven een lijst van users, gaan we per event na of het relevant is voor die gebruiker. Vooraleer we kijken of het event op een route ligt, kijken we eerst of het wel voldoet aan de andere eisen van de gebruiker. Met de eisen bedoelen we de filters die de gebruiker heeft ingesteld: is de user geïnteresseerd in dit type van event, voldoet het wel aan de timeconstraints van de gebruiker (de juiste dag, binnen het interval) etc.

Pas als het event aan al deze eisen voldoet, wordt het eigenlijke algoritme aangeroepen.

Omzetting coördinaten

De waypoints van een route worden in de database opgeslagen als een koppel lengte- en breedtegraden. Dit zijn 2 kommagetallen nauwkeurig tot op ongeveer 6 cijfers na de komma. Hierbij is het wel belangrijk om op te merken dat de minst beduidende cijfers in de echte wereld nog heel beduidend zijn. Kleine variaties in de laatste 2 cijfers, scheelt al gauw 100 meter.

Omdat computers niet goed zijn in het rekenen met kommagetallen en de nauwkeurigheid bewaren, kunnen we dus geen gebruik maken van dit coördinatensysteem.

Aan het begin van ons algoritme worden dan ook alle coördinaten omgezet van lengte- en breedtegraden naar ons eigen XY-coördinatensysteem. In dit systeem worden coördinaten niet in graden maar in meter uitgedrukt. Meer info over dit coördinatensysteem en de omzetting ernaar vindt u in deze link. Deze formule houdt met veel meer rekening dan de meeste formules. Als u meer wil weten over deze formule, is het ten sterkste aangeraden om het artikel eens te lezen.

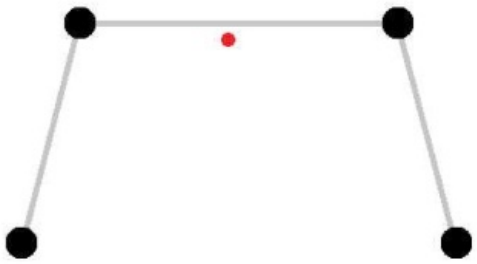
Om een omzetting te maken naar het XY-assenstelsel moeten we een centraal gebleven referentiepunt kiezen. Wij hebben voor de Kouter gekozen. De centrale ligging van dit punt is belangrijk want de omzettingen worden onnauwkeurig als de coördinaten te ver van dit punt liggen.

Doordat nu de coördinaten in meter uitgedrukt worden, hebben afrondingsfouten een veel kleiner effect dan wanneer ze in graden uitgedrukt worden. Deze omzetting is dan ook noodzakelijk en vergemakkelijkt het rekenen aangezien alles in meter is.

Onnauwkeurigheden die toch nog optreden zijn allemaal te wijten aan afrondingsfouten en het feit dat er meer factoren meespelen in de echte wereld (vb. hellingen)

Het eigenlijke algoritme

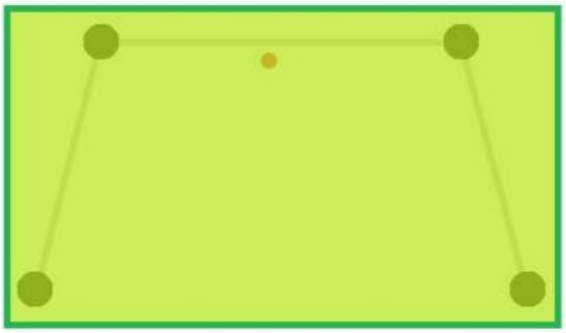
Om het algoritme gemakkelijker te kunnen uitleggen, stellen we alles versimpeld voor zoals op de figuur hieronder. De zwarte punten stellen de waypoints van de route voor en het rode punt is het event dat we willen matchen. De grijze lijnen zijn er enkel om de route duidelijker zichtbaar te maken.



Stap 1

In deze stap wordt er erg grof gefilterd. Omdat deze stap redelijk goedkoop is, loont het de moeite om events die zeer ver van de route liggen, al vroeg te kunnen uitsluiten.

Deze eerste filtering gebeurt door een rechthoek rond deze route te tekenen die de route volledig bevat. Voor de zekerheid wordt er een marge van 10m toegevoegd. Deze rechthoek wordt bepaald door alle coördinaten te overlopen en de minimum- en maximumwaarden bij te houden. Alle punten die niet in deze rechthoek liggen, vallen al af. Op onderstaande figuur wordt dit geillustreerd. Het rode punt ligt in de rechthoek, dus het algoritme wordt niet vroegtijdig gestopt

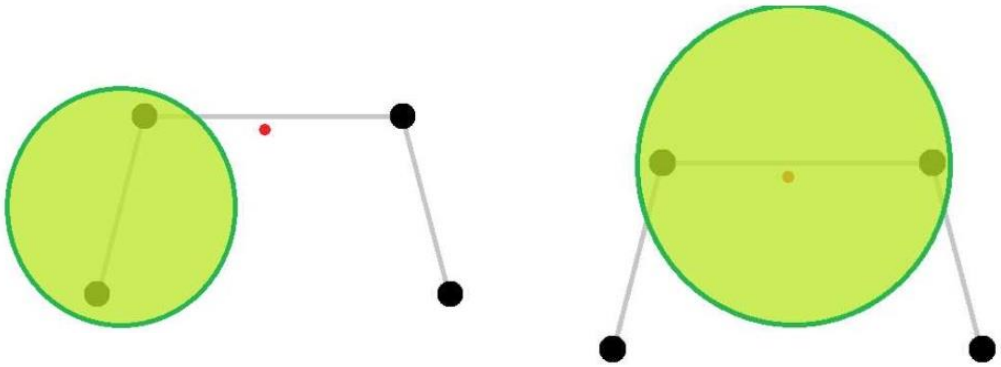


Stap 2

In de volgende stap worden alle koppels van 2 opeenvolgende waypoints van de route beschouwd. Van elk koppel wordt het middelpunt bepaald om vervolgens een cirkel te kunnen tekenen waarin beide waypoints liggen. De cirkel heeft als middelpunt het midden van de 2 waypoints en als straal de afstand van dat midden tot een van de waypoints. Aan de straal wordt een marge van 10m toegevoegd. De formules die we gebruiken zijn exact (op onvermijdelijke afrondingsfouten na) en steunen op de wiskunde.

De bedoeling van deze stap is kijken of een event in de buurt ligt van de 2 waypoints. Als dit het geval is, gaat het algoritme verder naar stap 3 van het algoritme, indien niet, dan wordt het volgende koppel genomen.

De volgende 2 afbeeldingen tonen dit proces, het event ligt niet in de buurt van de eerste 2 waypoints dus worden de volgende 2 in achting genomen. Hier zien we dat het event wel in de cirkel ligt, dus zal er naar de volgende en laatste stap overgegaan worden.

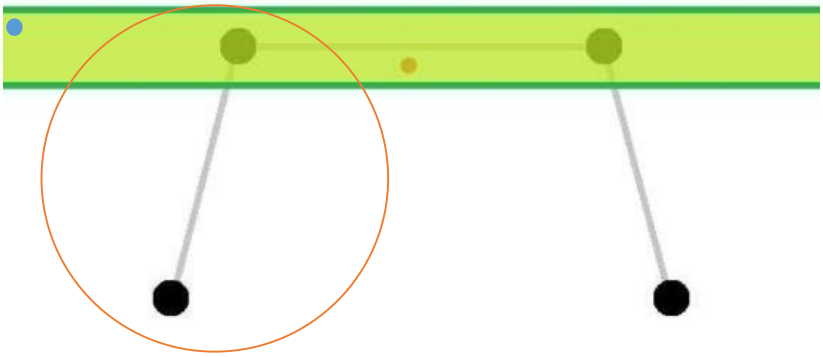


Stap 3

In deze laatste stap, wordt er gekeken of het event op een bepaalde afstand ligt van de rechte gedefinieerd door de 2 waypoints. De bepaalde afstand is wederom 10m. Als dit zo is, kunnen we ervan uitgaan dat het punt op de route ligt.

Hier wordt ook de belangrijkheid van stap 2 nog eens duidelijk. Kijk bijvoorbeeld naar het blauwe punt: Dit ligt wel nog binnen de marge van de recht, maar ligt toch niet in de buurt van de route. Als stap 2 overgeslagen wordt, zou het blauwe punt dus een false positive zijn. Dit omdat er een oneindig lange rechte getekend wordt. Maar doordat het dus buiten de oranje cirkel valt en al uitgesloten werd is stap 2, treden er dus geen false positives op.

Op de afbeelding kunnen we dit mooi zien. Het rode punt ligt nog duidelijk in het groene gebied en wordt dus gematched.



Waarom steeds die marge?

Als men enkel de lijnstukken tussen 2 punten zou beschouwen als de route, dan zijn de routes oneindig smal. De straten die de routes voorstellen zijn dat echter niet. Een straat minstens een paar meter breed en een event kan zich dus voordoen over die gehele breedte.

We hebben er dus voor gekozen om overal een marge van 10 meter in te voeren. Een event wordt dus nog gematched als het 10 meter van een oneindig smalle route ligt. Dit is breed genoeg om alle onnauwkeurigheden in de positie van de events op te vangen en toch nog klein genoeg om geen interferentie te hebben met parallelle straten (vb. de 2 richtingen van een autostrade).

Complexiteit

Als *n* het aantal waypoints van een route is, dan zal het testen van 1 event een complexiteit hebben van $\Theta(n)$. In zowel het beste als slechtste geval is deze hetzelfde, er is wel een constante waar rekening mee gehouden moet worden. In het beste geval geraakt het algoritme niet in de 2^e stap en zal de constante zeer laag zijn (punten omzetten gebeurt in constante snelheid + overlopen van alle punten zijn slechts enkele vergelijkingen). De 2^e en 3^e stap zijn wel wat zwaarder maar ook hier gebeuren er *n* berekeningen van constante tijd, maar hier zijn er wel vierkantswortels/machtsverheffingen en die zijn een pak intensiever dan slechts enkele vergelijkingen.