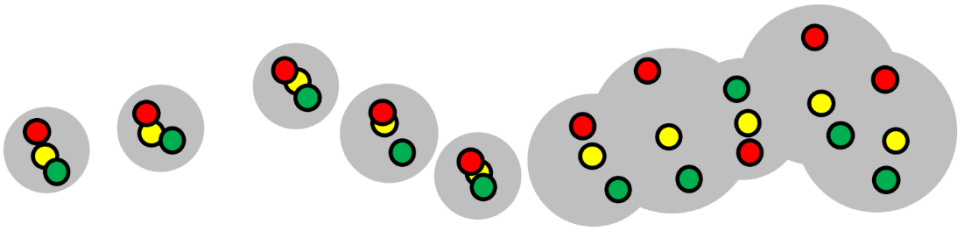

GPS AUTOMATIC MAPPING

Projeto de Sistemas de Navegação

Abstract

Mapeamento automático de um trajeto usando dados NMEA de várias passagens, com recurso ao algoritmo Dynamic Time Warping

João Louro e Samuel Cardoso

784590, 78639

Introdução

A grande disponibilidade atual de unidades GPS de tamanho reduzido levou à existência de uma enorme quantidade de dados de caminhos, tirados da atividade normal da população. São capturados dados GPS enquanto as pessoas andam, conduzem, ou andam de transportes públicos. Esta recolha de dados em grande quantidade tem uma grande utilidade, sendo até muitas vezes usada para objetivos menos convencionais. Um exemplo foi a avaliação feita recentemente dos distúrbios na parte superior da atmosfera provocados por um sismo no Nepal, usando dados de GPS em massa.

Podemos aperceber-nos, no nosso dia-a-dia, que repetimos certas secções de caminhos muito frequentemente. Ora, uma caraterística que se nota quando se avaliam grandes quantidades de dados GPS, é que há frequentemente diferentes trilhos GPS da mesma secção de estrada, ou para o mesmo percurso. Em geral, medir algo repetidamente, aproveitando todas as medições para chegar a um resultado final leva a uma maior precisão. O mesmo se aplica no caso de um caminho GPS. A questão é então: Será possível "tirar a média" de vários percursos GPS, e, se sim, será que isto nos leva a uma melhor estimativa da localização ao longo do percurso?

Caminho Escolhido

No caso deste projeto, quantas mais medições do trajeto percorrido se tiver, melhor e mais preciso se espera que seja o percurso medio final. Por essa razão, um dos caminhos escolhidos foi o percurso entre o Instituto Superior Técnico e a residência de um dos membros do grupo, com cerca de 1.3 km. Deste modo, foram conseguidas 10 medições deste percurso.

O grupo achou também que seria interessante, dado o nome dado ao projeto pelo professor (GPS Automatic Mapping), que se procedesse de facto ao primeiro mapeamento de um percurso previamente não mapeado. Para este efeito, foram tiradas cerca de 4 medições, durante várias caminhadas e voltas de bicicleta, a um troço de estradão, com cerca de 5 km, na freguesia da Cumeada, concelho da Sertã, no distrito de Castelo-Branco. O caminho do Instituto Superior Técnico será chamado ao longo deste enunciado de "caminho de cidade", e o caminho junto da comeada será chamado "caminho de estradão".

Algoritmo Usado

Inicialmente, o grupo decidiu usar para o projeto um algoritmo que identifica um caminho médio a partir de uma "nuvem" de pontos de GPS. Neste algoritmo iterativo, chamado Principal Curve Analisys, proposto por Hastie e Stuetzle em 1989, não se tinha assim em atenção de que "passagem" era proveniente cada uma das coordenadas de GPS, traçando-se uma curva pelo centro da tal nuvem de pontos. Este algoritmo, no entanto, é relativamente pesado do ponto de vista computacional, e bastante difícil de implementar. Depois de bastantes tentativas, e para não correr o risco de não conseguir entregar um projeto funcional, o grupo decidiu desistir deste algoritmo, e recomeçar o trabalho com outro. Isto levou a alguns problemas, dado que o tempo disponível para o projeto não foi o ideal, atrasando a data de entrega e

impossibilitando a realização de algumas ideias extra que o grupo gostaria de ter implementado no projeto.

O segundo algoritmo tentado, e que acabou por ser o implementado no projeto, foi o Dynamic Time Warping. Este é um algoritmo usado para comparar e alinhar duas séries temporais, encontrando o alinhamento não-linear ótimo entre duas sequências de valores numéricos. Dessa maneira, é possível encontrar padrões entre medições de eventos com diferentes ritmos. O Dynamic Time Warping pode usar-se para alinhar qualquer tipo de dados. Entre as diversas aplicações deste algoritmo, tem-se, por exemplo o reconhecimento de fala e de assinaturas.

Como este algoritmo consegue encontrar padrões entre medições com diferentes ritmos temporais, é extremamente apropriado para a utilização pretendida, dado que, em diferentes passagens pelo mesmo percurso, as velocidades vão ser diferentes, o que nos impede de simplesmente comparar coordenadas pela sua ordem de medição no percurso.

No geral, este é um método que calcula um pareamento ótimo entre duas sequências. As sequências são "torcidas" não-linearmente na dimensão temporal para determinar uma medida da sua similaridade, independentemente de variações não-lineares no tempo.

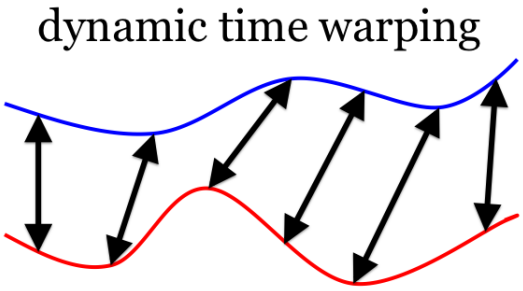

Figura 1 - Ilustração Simples de uma parte do funcionamento do algoritmo

Para além de apresentar uma medida da similaridade entre duas sequências, é também produzido um caminho médio. Isto consegue-se unindo todos os pontos correspondentes entre si nas duas medições, e tirando a média entre eles. No fim, esses pontos médios são unidos, de modo a termos um percurso médio das duas passagens.

Este algoritmo tem, no entanto, algumas limitações. A maior é que, ao contrário do Principal Curve Analysis, este algoritmo apenas nos dará uma média de dois caminhos de cada vez, o que impede a análise de todos os trajetos em simultâneo. Este é um problema contornável, agrupando os trajetos em grupos de 2, e atribuindo "pesos" a trajetos menos importantes.

Para aplicar o Dynamic Time Warping, uma informação muito importante é a distância entre os pontos. Para começar, começa-se então por calcular uma matriz com todas as distâncias, entre todos os pontos. Como cada medição do caminho de cidade tem aproximadamente 8oo pontos, estas matrizes têm normalmente, no nosso projeto, mais de 600000 pontos. Esta matriz é frequentemente chamada "Matriz de Custo", em aplicações mais matemáticas deste algoritmo, dado que se diz que duas medições têm baixo custo, se forem semelhantes entre si, e alto custo, se não o forem. Neste projeto, o custo corresponde à distância entre as coordenadas.

Nota: Como nem todos os pontos desta matriz são necessários, no código de matlab apenas serão calculados os pontos necessários da matriz, para poupar tempo de computação.

O objetivo é agora encontrar um alinhamento entre os dois vetores de coordenadas, correspondente à menor distância (menor custo). Este alinhamento será um percurso que percorre a matriz, "apanhando os pontos com menor distância.

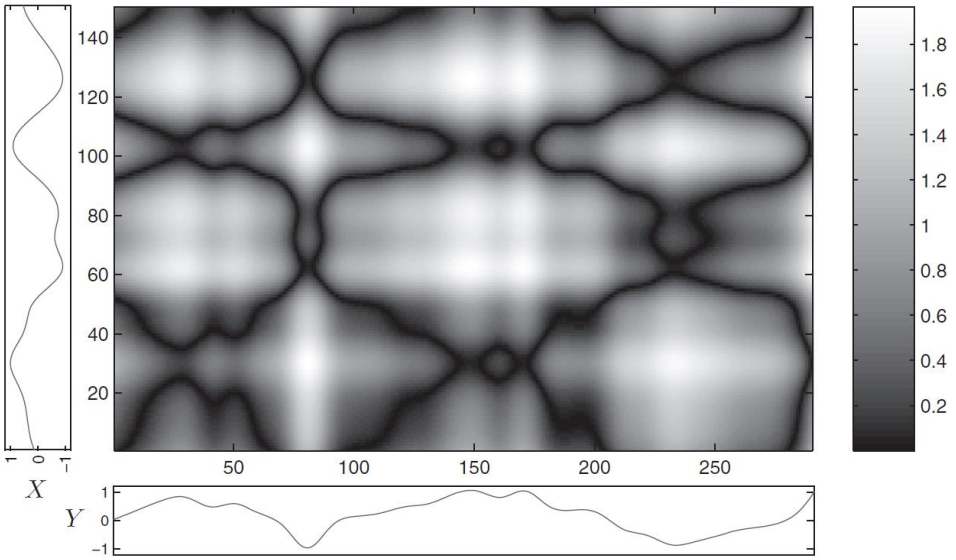

Figura 2 - Exemplo de uma matriz de distâncias

Neste exemplo, temos a matriz de distância para duas medições de um caminho com 140 e 250 medidas. Regiões de baixa distância são representadas por cores escuras, enquanto regiões com alta distância correspondem a cores próximas do branco.O alinhamento pretendido percorreria então os chamados "vales" da matriz de distância, ou seja, as secções a preto.

Este caminho corresponderia a um vetor, em que estariam presentes as correspondências entre os pontos dos dois caminhos, ou seja, pontos "equivalentes". Unindo estes pontos, e tirando metade da distância entre os dois pontos correspondentes, temos um ponto médio, entre dois pontos correspondentes nos dois caminhos. Tirando todos estes pontos e unindo, chega-se a um caminho médio.

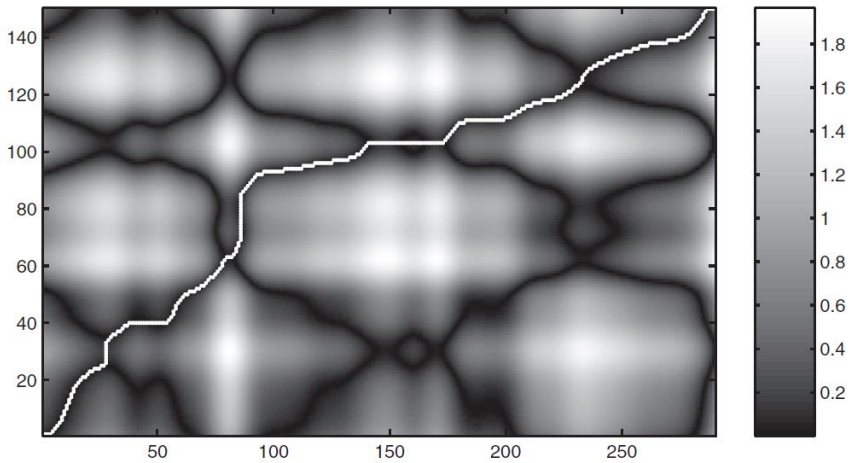

Figura 3 - Trajeto de distância mínima através da matriz de distância

Recetores

De seguida estão referenciados os recetores utilizados para obtenção de resultados.

QSTARZ BT-Q1300

BQ Aquaris E 4.5

Samsung Galaxy S l

Huawei Y6 II

Para a medição do trilho no estradão foram utilizados o BQ Aquaris E 4.5, o Huawei Y6 II e o Samsung Galaxy S Duo. Como os trilhos que estão a ser comparados foram obtidos de recetores diferentes, é de referir que alguma fonte de erro poderá vir deste facto. No trilho da cidade, foram utilizados o Huawei Y6 II e o QSTARZ BT-Q1300.

Durante as medições tivemos alguns problemas, principalmente pelo facto de a aplicação que foi utilizada, a GPS NMEA, não ter funcionado corretamente em todos os dispositivos que utilizámos. Sedo assim, no trilho do estradão, inicialmente tínhamos previsões de ter 10 ou 12 passagens, mas no entanto só conseguimos obter 5 completas. No trilho da cidade, como só foram utilizados dois recetores distintos, os erros (que são maiores que no outro percurso) devem-se principalmente ao multi caminho e à obstrução do sinal gps por árvores e passagens debaixo de prédios.

Estes recetores sãa todos de ralativo baixo custo, o que ilustra o facto de este ser um método rentável e simples caso alguém queira mapear zonas que não estão referenciadas.

Explicação do Código

Proceder-se-á agora a uma explicação do nosso código, incluido o script principal e as funções.

Note-se que todos os trajetos se encontram em ficheiros NMEA separados.

Script Principal

Análise dos Ficheiros NMEA

38.7381	-9.1389	142.5038
38.7381	-9.1389	142.3784
38.7381	-9.1389	142.3504
38.7381	-9.1389	142.4533
38.7382	-9.1389	141.8281
38.7381	-9.1389	141.5924
38.7382	-9.1389	141.3983
38.7382	-9.1388	139.8971
38.7382	-9.1388	139.4990
38.7382	-9.1388	139.2560
38.7382	-9.1388	139.3565
38.7382	-9.1388	139.4407
38.7382	-9.1388	139.5512
38.7382	-9.1388	139.9749
38.7382	-9.1388	139.8016

Figura 4 - Formato dos dados tratados

A primeira ação do código é analisar cada ficheiro NMEA com a função analisa_nmea_file. O que esta função faz é passar as coordenadas presentes nos ficheiros NMEA para vetores. Note-se que as coordenadas são retiradas das entradas GPGGA.

Os dados para cada caminho ficam assim distribuídos em matrizes n x 3, sendo n o nómero de entradas GPGGA dos ficheiros NMEA de cada uma das passagens pelos caminhos. No trilho de cidade, este valor de n encontra-se em valores normalmente inferiores 800, enquanto que no trilho de estradão não mapeado, os valores de n são significativalmente maiores, passando largamente das 1000 entradas.

Ordenamento das Matrizes de Coordenadas

Seguidamente, temos a função ordem.Esta função é muito curta, e sumpre uma função muito simples, que é a de assegurar que a ordem das entradas é a mesma em cada uma das matrizes de coordenadas. Passa-se a explicar: na recolha de dados, a direção em que o caminho foi percorrido não foi sempre a mesma. De facto, como o caminho de cidade é o caminho entre o técnico e a residência de um dos membros do grupo, e os dados foram recolhidos durante a comuta diária, seria de esperar que aproximadamente metade dos trajetos tenham sido gravados na direção oposta de outra metade. Na verdade, isto não ocorreu, devido a problemas ocasionais com o recetor. No entanto, é um facto que nem todos os caminhos foram gravados na mesma direção. O algoritmo Dynamic Time Warping, por si só, não seria capaz de tratar dados com a sequência temporal invertida. Foi então feita a função ordem, que faz com que todas as matrizes de caminhos fiquem na mesma ordem. Isto é conseguido de uma forma muito simples: começa-se por comparar os valores de latitude pa primeira e última entradas de cada trajeto. Se a longitude do primeiro ponto recolhido for maior que a do último ponto recolhido, a matriz de coordenadas é totalmente invertida verticalmente. Desta forma, temos todos os trajetos com a mesma direção.

Como neste projeto o ponto onde se termina e acaba o trajeto não é importante, podemos começar a analisar os pontos por qualquer um dos lados, desde que todos os caminhos tenham a mesma orientação, o que é conseguido por esta função.

Tratamento de Dados

Nos trajetos que normalmente percorremos, e, especificamente, nos dois trajetos que foram precorridos para a realização deste projeto, não é normal que existam gandes mudanças bruscas de direção. Assim, é de estranhar se uma coordenada GPS tiver um ângulo muito pequeno entre a anterior e a seguinte, dado que isto mostraria uma mudança de direção muito rápida e ineperada, algo que não é normal na comuta normal da população. Assim, as matrizes de dados de cada um dos caminhos foram seguidamente sujeitas a uma função que elimina os “pontos angulosos” dos caminhos.

Notou-se que, eliminando coordenadas que fazem ângulos menores que 90º com as anteriores e seguintes, se conseguia um tratamento de dados bastante satisfatório, dado que os trajetos eram um pouco "alisados", e picos resultantes de má recolha de dados eram eliminados. Esta eliminação de pontos angulosos foi corrida quatro vezes para cada caminho, uma vez que, certas vezes, alguns pontos não eram suficientemente bem corrigidos com apenas uma correção.

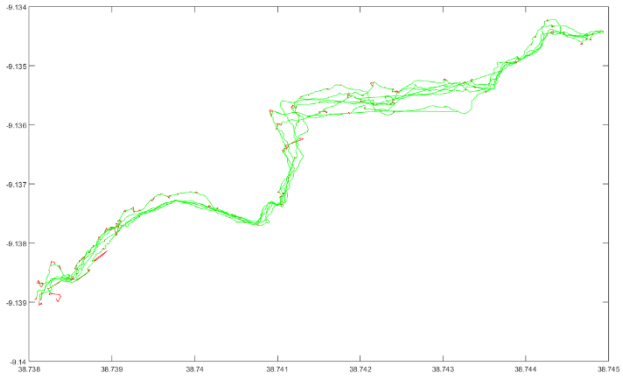

Figura 5 - Resultado do algoritmo de tratamento de dados

Na figura seguinte podem ver-se, a verde, os trajetos depois do tratamento de dados, e a vermelho, as partes que foram eliminadas de cada trajeto.

Como se pode ver, os caminhos foram levemente homogenizados, e alguns picos, resultantes de más medições, foram eliminados. Isto é de extrema importância, pois estes pontos, por vezes, davam origens a caminhos médios bastante errados, o que derrotava o propósito do projeto.

Na figura seguinte, mostra-se um detalhe da correção que foi realizada. Podemos assim mais facilmente ver o interesse em proceder a uma correção deste tipo, pois os dados tornam-se bastante mais fidedignos. Um ponto negativo desta abordagem é, no entanto, o facto de serem eliminados pontos que não são repostos, o que leva a uma redução do número total de amostras de cada caminho. Uma potencial forma de combater isto seria criar um novo ponto, no local do eliminado, com coordenadas médias entre o ponto anterior e o seguinte. Isto seria, no entanto, introduzir informação não fornecida pelo recetor GPS: Dado que a quantidade de pontos retirada, apesar de ser pequena, não é egligenciável, optou-se por não recorrer a esta solução.

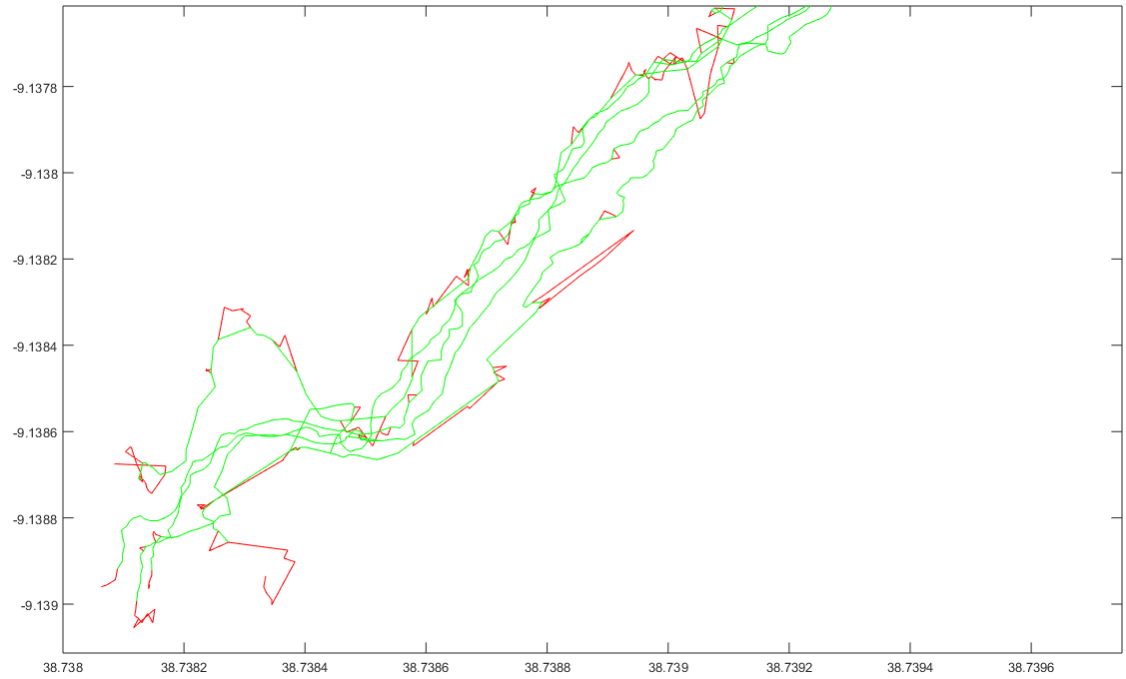

Figura 6 - Detalhe dos resultados do algoritmo de tratamento de dados nos trajetos de cidade

Aplicação do Dynamic Time Warping

Tendo já os caminhos bem ordenados e corrigidos, tem agora de se encontrar os caminhos médios. Para isso, usa-se a função caminhomedio.Simplificando, esta função recebe como inputs as matrizes de coordenadas de dois caminhos, e devolve como output uma matriz de pontos do caminho médio entre os dois trajetos fornecidos. Para além disto, são também recebidos como inputs e servidos como outputs vetores que nos indicam a distância entre os caminhos (importante para o indicador de confiança no caminho. Isto será melhor explicado à frente.

Apresentação de Resultados

Será agora feira uma breve apresentação dos resultados obtidos neste projeto. Os dados serão apresentados em formato gráfico, com o eixo horizontal a corresponder à latitude, e o vertical à longitude.

Inicialmente, irá apresentar-se o resultado da aplicação do algoritmo de tratamento de dados, que elimina pontos "angulosos".

Nota-se que houve uma boa correção dos dados, como foi já descrito anteriormente, na explicação do algoritmo.Parao caminho de estradão, temos o seguinte resultado:

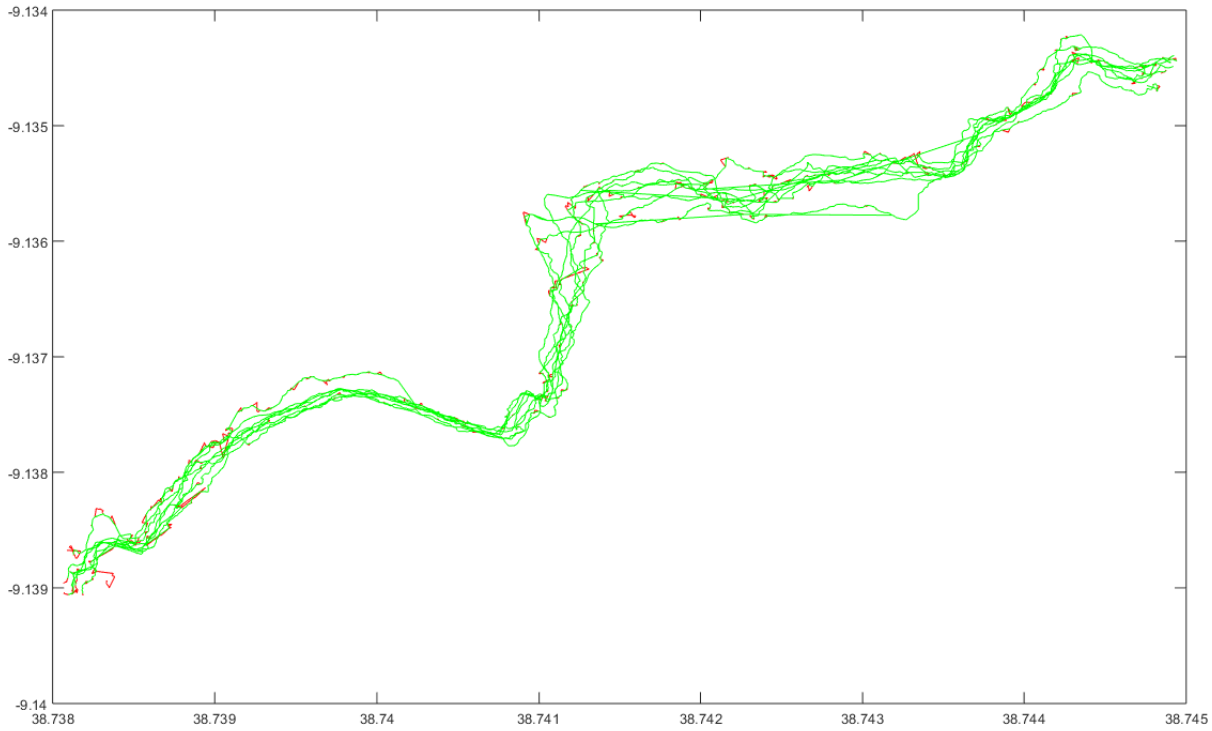

Figura 7 - Resultados do algoritmo de tratamento de dados, como mostrado anteriormente

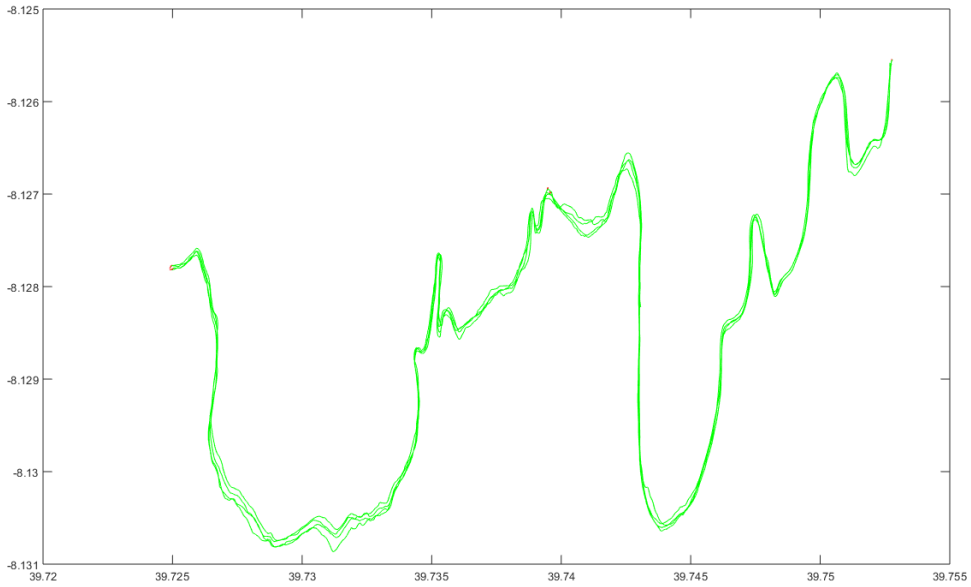

Figura 8 - Resultados do algoritmo de tratamento de dados nos trajetos de estradão

Neste caso, as correções foram muito menos significativas, havendo apenas eliminação de alguns pontos no início do trajeto, e perto do meio do caminho. Isto deve-se ao facto de os dados terem sido recolhidos num ambiente com muito menos obstruções, havendo uma receção de dados bastante melhor. Por esta razão, tem-se um seguimento da posição bastante melhor. Como no caminho realmente percorrido não há viragens rápidas em ângulos pequenos, não temos pontos "angulosos" para eliminar.

Como mencionado anteriormente, o algoritmo Dynamic Time Warping consegue apenas calcular o caminho médio de dois caminhos de cada vez. No gráfico seguinte apresentam-se, para o caminho de cidade e o estradão, os caminhos médios de caminhos iniciais (agrupando-se os caminhos dois a dois, calcularam-se os caminhos médios obtidos seguidamente).

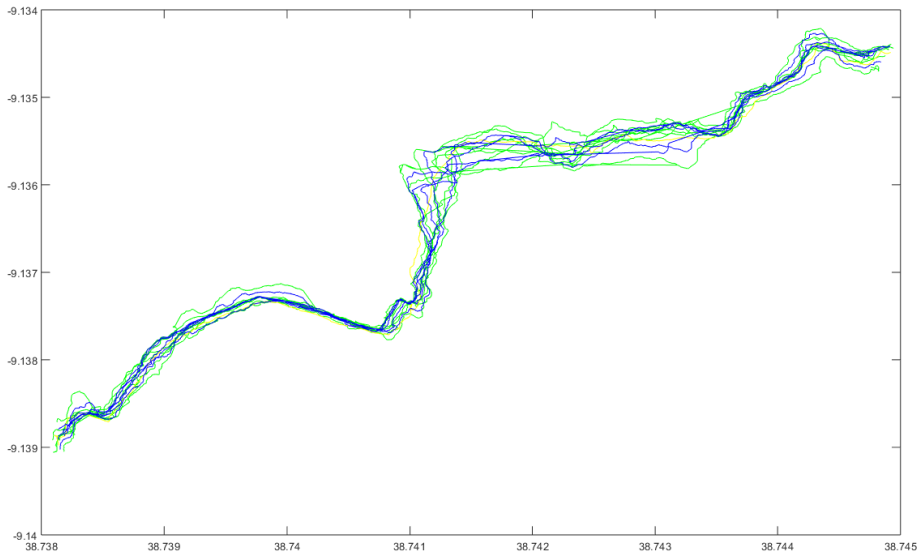

Figura 9 - Primeiras Médias do caminho de cidade

Como há 11 trajetos, e os caminhos médios são tirados de grupos de 2 caminhos, há um que vai ficar de fora. Este será mais tarde junto aos outros, numa média ponderada

Pode ver-se que os caminhos médios começam já a convergir para um caminho médio, mas ainda há uma grande distância entre os trajetos. Isto deve-se ao facto de este trajeto percorrer zonas com uma grande densidade de prédios, o que bloqueia muito o sinal.

Na figura seguinte, temos os primeiros caminhos médios do trajeto de estradão.

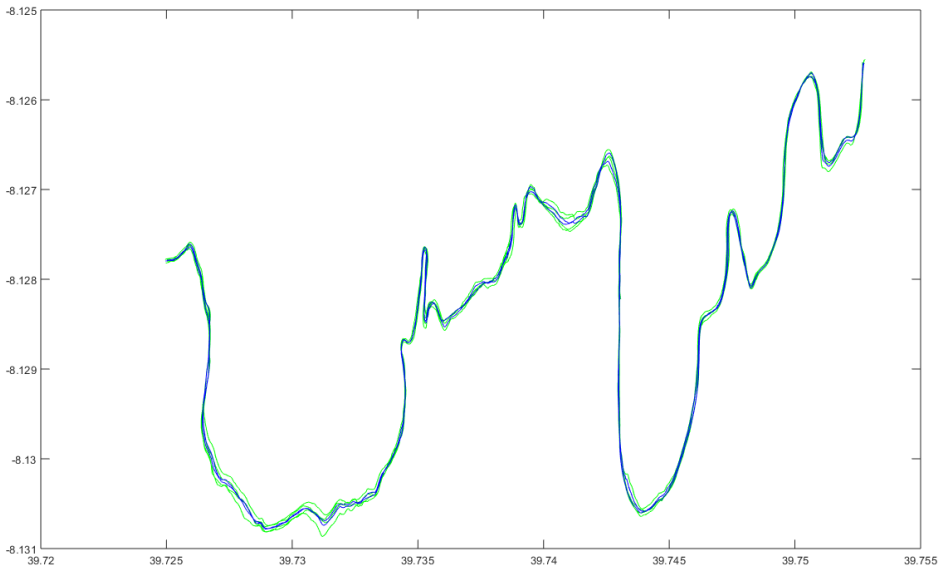

Figura 10 - Primeiras médias do caminho de estradão

Neste trajeto houve apenas quatro passagens com o recetor de GPS. No entanto, isso aparenta ter sido suficiente, dado que basta apenas fazer médias de pares de caminhos para se começar já a ver um claro padrão de convergência. Como houve 4 passagens, fazendo média dois a dois ficamos com dois caminhos médios a azul, que estão já bastante aproximados. No caminho de cidade faz-se depois a média dos caminhos a azul (sempre em grupos de dois, inserindo também o caminho 11, mostrado amarelo no gráfico presente da página anterior. Conseguimos então o caminho médio mostrado a vermelho:

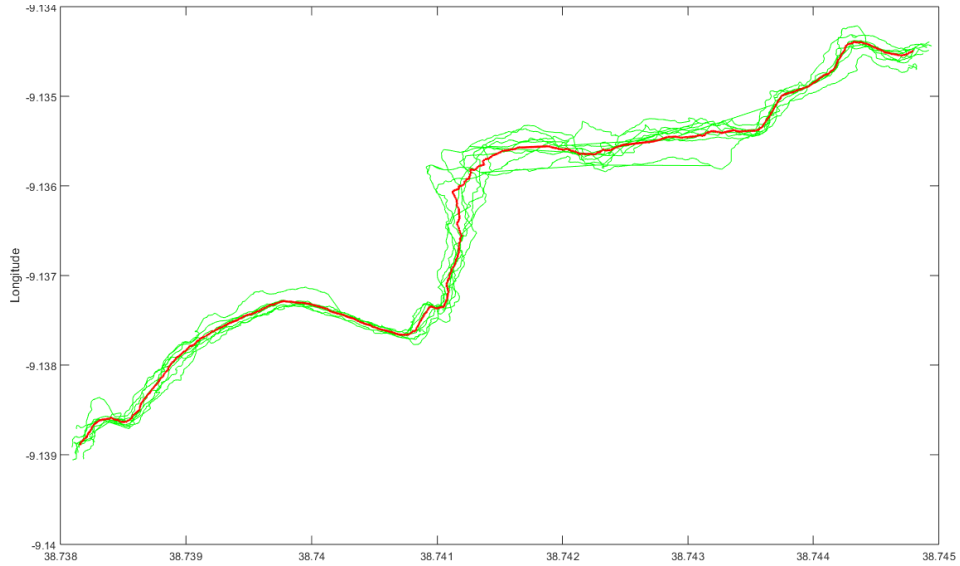

Figura 11 - Caminho médio final de cidade

Os caminhos a verde correspondem aos dados de GPS retirados dos sensores. Como podemos ver, o algoritmo Dynamic Time Warping conseguiu fornecer uma aproximação bastante boa, dado que o trajeto médio encontra-se sempre dentro dos "limites" do conjunto de trajetos, e aparentemente ao centro. A distância calculada deste caminho médio de cidade é de 1.1 km.

O mesmo se pode dizer para o trajeto de estradão:

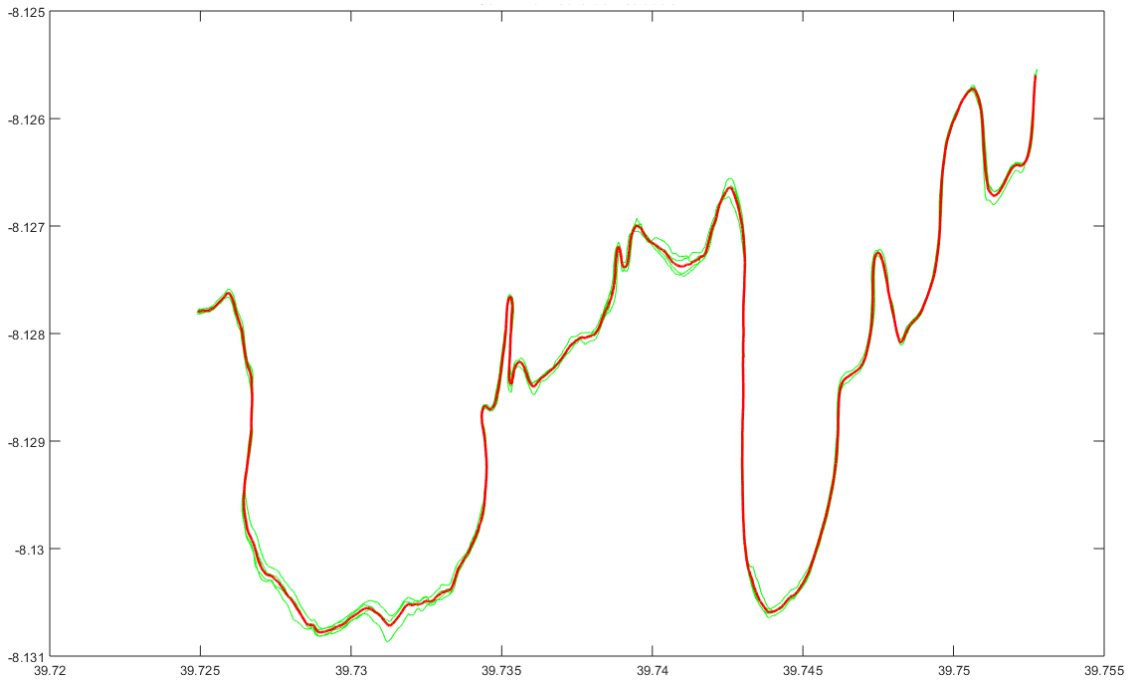

Figura 12 - Caminho médio final de estradão

Nota-se que a aproximação do caminho médio foi também a esperada. No entanto, espera-se que este caminho médio seja mais próximo do real que o caminho de cidade dado que, apesar de terem sido registados menos dados, estes são muito mais próximos entre si, ao contrário dos dados do caminho de cidade, que estão bastantes dispersos.

Foi também calculado uma estimativa da confiança que podemos ter para cada ponto. Nos gráficos seguintes esta confiança será apresentada de forma relativa usando linhas vericais em cada ponto do caminho médio. Assim, iremos ter uma noção, através dos gráficos, acerca das zonas em que o caminho é mais confiável, ou menos confiável. Espera-se que, onde há maior separação dos caminhos, o valor correspondente à dispersão seja maior, e onde hà menor dispersão dos caminhos, espera-se que esse valor seja menor. Nestes gráficos serão também mostrados os caminhos GPS, de forma comparar os valores de dispersão com os trilhos indicados pelos sensores GPS.

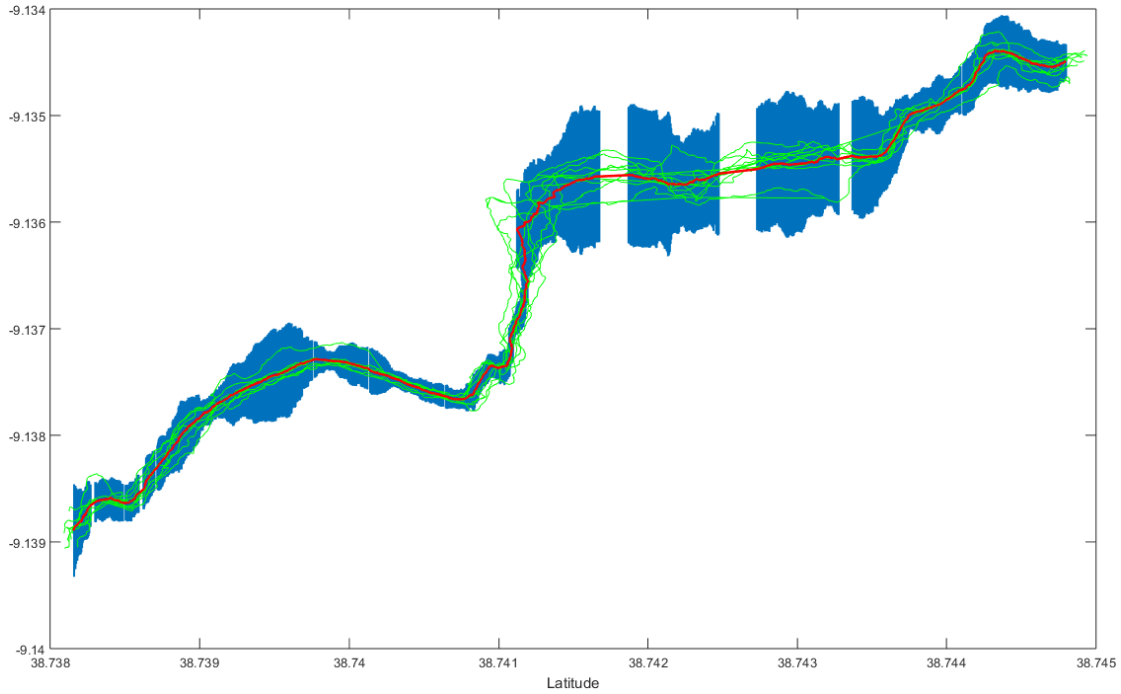

Figura 14 - Visualização da dispersão ao longo do caminho de cidade

Podemos ver então que a função nos dá uma boa indicação da confiança que podemos ter no caminho, dado que zonas com maior diferença entre os caminhos têm maiores valores de dispersão. Nota-se, mais perto da parte final do trajeto, que parece haver uma interrupção dos valores de tolerância. Isto deve-se apenas ao facto de, onde não temos valores de tolerância,

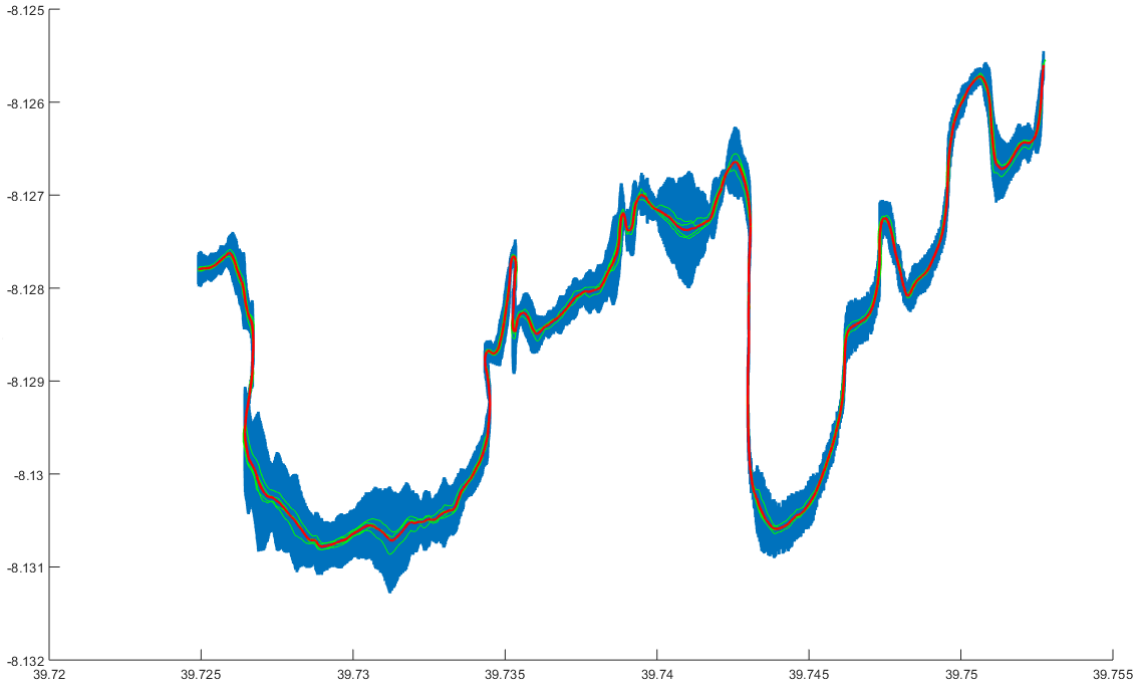

Figura 13 - Visualização da dispersão ao longo do caminho de estradão

o caminho médio não ter pontos, ou seja, o caminho médio é uma linha reta nessas secções.

Podemos ver que, neste trajeto de estradão, os valores de dispersão são também maiores quando temos caminhos mais espalhados entre si. Note-se, também, que para

este caminho os valores de dispersão são muito menores que para o caminho de cidade. Apenas parecem semelhantes pois, para estes gráficos, pretendia-se apenas uma visualização relativa, e por isso em ambos os trajetos as tolerâncias foram multiplicadas por constantes. No entanto, acedendo ao código, estão disponíveis os vetores confMFE e confMFC, que contêm para cada ponto dos trajetos de estradão e cidade, respetivamente, os valores de tolerância em metros.

Iremos agora então vizualizar os trajetos de GPS superimpostos aos mapas de satélite. Para isso, iremos usar ficheiros KML obtidos através da função Matlab e dos nossos dados, bem como o Google Eart, para ler e visualizar os ficheiros.

Figura 15 - Visualização dos caminhos de cidade superimpostos aos dados de satélite

Podemos ver então o conjunto dos 11 trajetos de cidade. No canto inferior esquerdo vê-se o Instituto Superior Técnico, que foi o ponto de partida ou chegada de todos os trajetos, nomeadamente o portão junto ao pavilhão de Eletrotecnia. Nesta secção, temos já uma receção relativamente má, nomeadamente até chegar à praça de Londres. Isto deve-se à existência de

prédios na zona, de um lado da rua, que bloqueiam o sinal. Ao chegar até à praça de Londres, e junto à igreja, tem-se uma área relativamente aberta, o que nos permite ter bastante boa receção de dados GPS, e, assim, uma posição bastante precisa.

Note-se que se tentou, na medição dos trajetos, fazer um caminho o mais constante possível, seguindo sempre as mesmas passadeirras, e caminhando sempre do mesmo lado da estrada. Depois de sair da praça de Londres tem-se a zona de pior receção. Esta é uma zona de prédios altos de ambos os lados da rua, bem como de árvores altas. Isto enfraquece o sinal que chega aos recetores, fazendo com que os trilhos guardados sejam muito espalhados. No ponto em que se vê uma curva mais apertada (cerca de 90º) há inclusive uma passagem por baixo de prédios, o que enfrquece o sinal ainda mais. Há passagens destas nos 3 cruzamentos seguintes, o que ajuda a explicar a grande dispersão dos sinais GPS na reta depois da curva. Até ao destino final, na Av. Padre Manoel da Nobrega 14C, os sinais convergem um pouco, porque há alguma

Figura 16 - Caminho médio de cidade obtido superimposto a imagens de satélite

redução no número de prédios, mantendo-se no entanto ainda bastante espalhados.

Temos na figura anterior o caminho final obtido. Nota-se logo que a aproximação está bastante boa, seguindo grande parte do caminho percorrido com bastante qualidade. A parte inicial do caminho tem um seguimento bastante satisfatório do caminho percorrido, encontrando-se bastante próxima do passeio, mesmo com a desvantagem de ser uma zona de prédios. Ao chegar à zona aberta, próximo da igreja, temos um seguimento bastante bom, dado que o caminho percorrido foi quase exatamente o mostrado. Atravessando a praça de Londres, mesmo com o bloqueio das árvores, a receção de sinal manteve-se bastante boa, como provado pela precisão do caminho seguido.

Ao entrar para a zona de prédios, no entanto, há uma degradação do sinal. Note-se que o caminho foi percorrido sempre nos passeios da esquerda. No entanto, nesta secção, por vezes o trajeto chega até ao lado direito da estrada, ou entra dentro de prédios, o que faria este caminho impossível de seguir. Note-se, no entanto, que as tolerâncias nesta zona são as maiores do trajeto. Um eventual viajante, a seguir este trajeto, estaria avisado da necessidade de ter em atenção o seu percurso. O seguimento nesta zona reta, é, no entanto, relativamente bom, tendo em conta as medições de onde foi obtido. Ao chegar ao final do trajeto, o caminho torma-se outra vez mais regular, seguindo com relativa exatidão o trajeto percorrido.

De seguida, serão analisados os resultados obtidos para o estradão ainda não mapeado.

Figura 17 - Resultado do caminho final obtido para o trajeto de estradão

À primeira vista, os resultados parecem bastante satisfatórios, dado que o estradão não está visível por baixo da linha vermelha, o que é um bastante bom indicador de que há um bom seguimento da estrada. De notar também que esta é uma zona relativamente isolada, como se nota pela ausência de povoamentos.

Como se nota na figura ao lado, há um seguimento bastante bom entre o estradão e os dados obtidos. Isto deve-se pricipalmente à ausência de prédios, dado que os recetores usados foram essencialmente os mesmos, e a quantidade de dados foi menos de metade quando comparativamente aos dados obtidos em ambiente de cidade. Como se pode comprovar pelo ficheiro KML, a linha média obtida sobrepõe-se quase inteiramente ao estradão, havendo apenas um ou dois ocasionais deslizes da linha para o exterior da estrada de terra batida.

Figura 18 - Secção do caminho de estradão

Para finalizar a análise de dados, temos então a comparação final entre os trilhos iniciais e a aproximação feita pelo nosso programa.

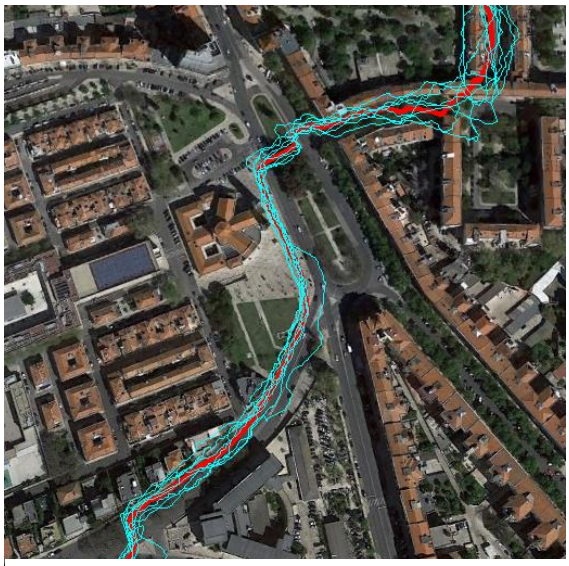

Figura 19 - Caminho médio e caminhos de GPS superimpostos a imagens de satélite

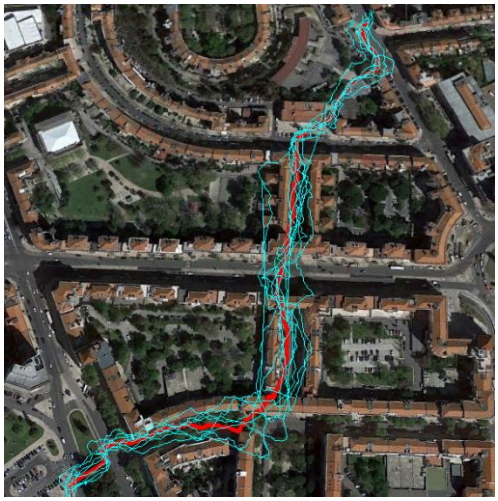

Figura 20 - Segunda parte do trajeto

Verifica-se que o caminho médio passa pelo centro de todos os pontos a cada instante, tal como foi proposto inicialmente. Mesmo quando há uma grande dispersão, a média final passa no trajeto previsto.

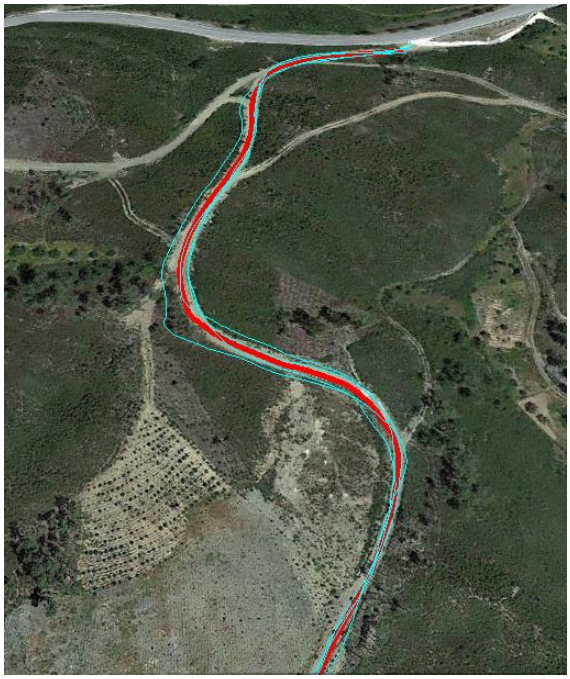

Figura 22 - Secção do trilho médio de estradão, superimposto aos caminhos de GPS e imagens de satélite

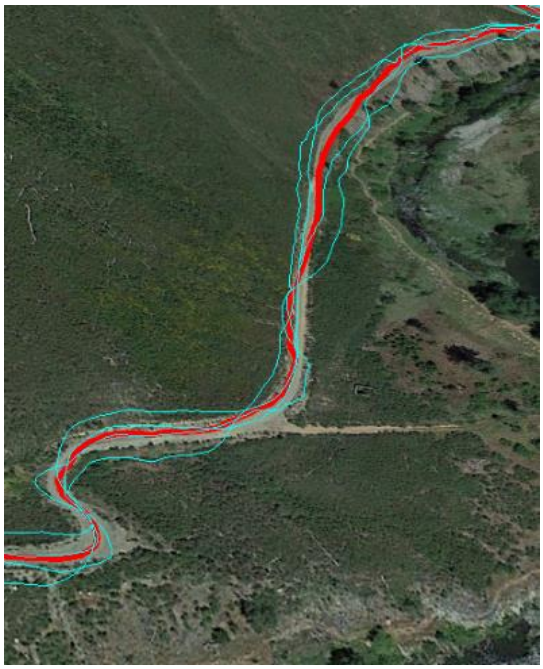

Figura 21 - Outra secção

No trilho do estradão, fazendo a sobreposição da média final e dos pontos iniciais, também se verifica o resultado esperado. Com uma maior proximidade, verifica-se que a média final faz um seguimento bastante satisfatório da estrada.

Conclusão

Conclui-se então que se consegue ter uma boa aproximação do trilho real utilizando o algoritmo implementado apesar de alguma discrepância em dados recolhidos, devido a erros de sinal ou de multicaminho. A média obtida com o programa que foi feito, fica de acordo com o esperado, tendo uma grande precisão, mesmo em regiões mais críticas (debaixo de prédios ou árvores).

De modo a ter uma perceção da validade do algoritmo, foi criado um índice de precisão, tendo um máximo (pior valor) de 16,43 no caso do trilho do estradão e de 21.97 no caso do trilho na cidade, o que é de esperar, pois o erro na cidade é maior.

Com este indice, conseguimos fornecer ao utilizador a segurança que este tem ao percorrer o trilho. Intuitivamente, se o utilizador verificar que a sua localização é diferente da do trilho a determinado instante, poderá verificar através deste indice a certeza de estar no caminho correcto ou não.

Verificou-se entre os recetores utilizados que o que se aproximava mais do trilho, ou seja, que tinha menos erros na medição, foi o do Huawei Y6-II.

Ao nível da implementação do algoritmo, apesar de ser um método fácil e rápido de obter uma média viável entre dois caminhos, no conjunto final de pontos, este mostra ser um pouco lento, pelo facto de estar a ser implementado muitas vezes de maneira a fazer médias e "médias de médias". Para contornar este problema, optámos por guardar o valor das variáveis em ficheiros .mat e posteriormente usamos o load cada vez que queremos correr o programa. Caso seja inserido outro precurso para juntar às medias já obtidas, tem de se repetir todo o procedimento inicial, o que leva a algum tempo de processamento.

Utilização do código

De modo a correr o código, deverá abrir o ficheiro ScriptPrincipal.m e caso queira fazer a execução total, ou seja, correr o algoritmo, deverá tirar os comentários das funções até à linha em que se faz save das variáveis. Neste caso, deverá ainda comentar as linhas em que se faz load das variáveis antigas.

Em ambos os casos, serão gerados ficheiros kml. Estes ficheiros estão organizados por estradao1, estradao2..., etc, cidade1, cidade2...,etc, e as médias finais são MFEkml (média final estradao) e MFCkml (média final cidade).

Código

Script Principal

```
clc; clear all; close all;
%% Pôr os pontos em matrizes, com a mesma orientação
%{
%Fornecer os ficheiros à função analisa_nmea_file, que vai retirar as
%coordenadas dos ficheiros NMEA, e pô-las numa matriz de coordenadas do
%tipo [latitude longitude altitude]
[cidade1]=analisa_nmea_file('cidade1.nmea');
[cidade2]=analisa_nmea_file('cidade2.nmea');
[cidade3]=analisa_nmea_file('cidade3.nmea');
[cidade4]=analisa_nmea_file('cidade4.nmea');
[cidade5]=analisa_nmea_file('cidade5.nmea');
[cidade6]=analisa_nmea_file('cidade6.nmea');
[cidade7]=analisa_nmea_file('cidade7.nmea');
[cidade8]=analisa_nmea_file('cidade8.nmea');
[cidade9]=analisa_nmea_file('cidade9.txt');
[cidade10]=analisa_nmea_file('cidade10.txt');
[cidade11]=analisa_nmea_file('cidade11.txt');

[estradao1]=analisa_nmea_file('estradao1.txt');
[estradao2]=analisa_nmea_file('estradao2.txt');
[estradao3]=analisa_nmea_file('estradao3.txt');
[estradao4]=analisa_nmea_file('estradao4.txt');

%Função ordem, que põe todos os vetores a seguir a mesma orientação do
%caminho (antes deste tratamento, os primeiros pontos de alguns dos caminhos
%correspondiam aos últimos de outros
cidade1=ordem(cidade1);
cidade2=ordem(cidade2);
cidade3=ordem(cidade3);
cidade4=ordem(cidade4);
cidade5=ordem(cidade5);
cidade6=ordem(cidade6);
cidade7=ordem(cidade7);
cidade8=ordem(cidade8);
cidade9=ordem(cidade9);
cidade10=ordem(cidade10);
cidade11=ordem(cidade11);


estradao1=ordem(estradao1);
estradao2=ordem(estradao2);
estradao3=ordem(estradao3);
estradao4=ordem(estradao4);


%%Plotagem dos precursos na cidade, antes da correção dos angulos e com os
precursos já ordenados

%{
plot(cidade1(:,1), cidade1(:,2),'r'); hold on;
plot(cidade2(:,1), cidade2(:,2),'r'); hold on;
plot(cidade3(:,1), cidade3(:,2),'r'); hold on;
plot(cidade4(:,1), cidade4(:,2),'r'); hold on;
plot(cidade5(:,1), cidade5(:,2),'r'); hold on;
plot(cidade6(:,1), cidade6(:,2),'r'); hold on;
plot(cidade7(:,1), cidade7(:,2),'r'); hold on;
plot(cidade8(:,1), cidade8(:,2),'r'); hold on;
plot(cidade9(:,1), cidade9(:,2),'r'); hold on;
plot(cidade10(:,1), cidade10(:,2),'r'); hold on;
plot(cidade11(:,1), cidade11(:,2),'r'); hold on;


%}
%% Tratamento de Dados
%Neste ciclo, aplica-se a função que remove os "pontos angulosos". Esta
%função é corrida 4 vezes para cada trajeto, pois notou-se que isto
%levava a bons resultados
```

```
for i=1:4
cidade1=trataangulos(cidade1, 90);
cidade2=trataangulos(cidade2, 90);
cidade3=trataangulos(cidade3, 90);
cidade4=trataangulos(cidade4, 90);
cidade5=trataangulos(cidade5, 90);
cidade6=trataangulos(cidade6, 90);
cidade7=trataangulos(cidade7, 90);
cidade8=trataangulos(cidade8, 90);
cidade9=trataangulos(cidade9, 90);
cidade10=trataangulos(cidade10, 90);
cidade11=trataangulos(cidade11, 90);

end

%%%%Plotagem dos precursos na cidade, depois da correção dos angulos e com os
precursos já ordenados

plot(cidade1(:,1), cidade1(:,2),'g'); hold on;
plot(cidade2(:,1), cidade2(:,2),'g'); hold on;
plot(cidade3(:,1), cidade3(:,2),'g'); hold on;
plot(cidade4(:,1), cidade4(:,2),'g'); hold on;
plot(cidade5(:,1), cidade5(:,2),'g'); hold on;
plot(cidade6(:,1), cidade6(:,2),'g'); hold on;
plot(cidade7(:,1), cidade7(:,2),'g'); hold on;
plot(cidade8(:,1), cidade8(:,2),'g'); hold on;
plot(cidade9(:,1), cidade9(:,2),'g'); hold on;
plot(cidade10(:,1), cidade10(:,2),'g'); hold on;
plot(cidade11(:,1), cidade11(:,2),'y'); hold on;



%% Caminhos Médios

%A função caminhomedio dá-nos como output o caminho médio dos dois caminhos
%fornecidos em input. Dá também uma indicação de "confiança" que podemos
%ter em cada ponto, ou seja, a distância entre os pontos dos caminhos de
%onde tirámos o caminho médio.
[MC1,confMC1,dMC1]=caminhomedio(cidade1,0,cidade2,0,1);
[MC2,confMC2,dMC2]=caminhomedio(cidade3,0,cidade4,0,1);
[MC3,confMC3,dMC3]=caminhomedio(cidade5,0,cidade6,0,1);
[MC4,confMC4,dMC4]=caminhomedio(cidade7,0,cidade8,0,1);
[MC5,confMC5,dMC5]=caminhomedio(cidade9,0,cidade10,0,1);


%%Plotagem das primeiras médias da cidade, antes da correção dos angulos e com os
precursos já ordenados
%{
plot(MC1(:,1), MC1(:,2),'r'); hold on;
plot(MC2(:,1), MC2(:,2),'r'); hold on;
plot(MC3(:,1), MC3(:,2),'r'); hold on;
plot(MC4(:,1), MC4(:,2),'r'); hold on;
plot(MC5(:,1), MC5(:,2),'r'); hold on;
%}


%% Tratamento de dados de caminhos médios
%Este ciclo é igual ao usado anteriormente para tratar os caminhos, mas
%agora são tratadas as primeiras médias dos caminhos
for i=1:4
MC1=trataangulos(MC1, 90);
MC2=trataangulos(MC2, 90);
MC3=trataangulos(MC3, 90);
MC4=trataangulos(MC4, 90);
MC5=trataangulos(MC5, 90);
end

%%Plotagem das primeiras médias da cidade, depois da correção dos angulos e com os
precursos já ordenados
%{

plot(MC1(:,1), MC1(:,2),'b'); hold on;
plot(MC2(:,1), MC2(:,2),'b'); hold on;
plot(MC3(:,1), MC3(:,2),'b'); hold on;
plot(MC4(:,1), MC4(:,2),'b'); hold on;
plot(MC5(:,1), MC5(:,2),'b'); hold on;
```

```
%}
[MMC1,confMMC1,dMMC1]=caminhomedio(MC1,confMC1,MC2,confMC2,1);
[MMC2,confMMC2,dMMC2]=caminhomedio(MC3, confMC3,MC4,confMC4,1);
[MMC3,confMMC3,dMMC3]=caminhomedio(MC5, confMC5,cidade11,confMC5,1/3);

[MMMC1,confMMMC1,dMMC4]=caminhomedio(MMC1,confMMC1,MMC2,confMMC2,1);

[MFC,confMF,dMFC]=caminhomedio(MMMC1,confMMMC1,MMC3, confMMC3,1/3);
disp( sprintf( 'A distância do caminho médio de cidade é de %s kilómetros', dMFC ) );

for i=1:length(MFC)
    line([MFC(i,1) MFC(i,1)], [MFC(i,2)-((confMF(i)/2))*7
MFC(i,2)+((confMF(i)/2))*7],'Linewidth',1.5); hold on;
end

title ('Caminho Médio de Cidade');
xlabel('Latitude');
ylabel('Longitude');

%%plotagem total com as médias intermédias e média final
plot(cidade1(:,1), cidade1(:,2),'g'); hold on;
plot(cidade2(:,1), cidade2(:,2),'g'); hold on;
plot(cidade3(:,1), cidade3(:,2),'g'); hold on;
plot(cidade4(:,1), cidade4(:,2),'g'); hold on;
plot(cidade5(:,1), cidade5(:,2),'g'); hold on;
plot(cidade6(:,1), cidade6(:,2),'g'); hold on;
plot(cidade7(:,1), cidade7(:,2),'g'); hold on;
plot(cidade8(:,1), cidade8(:,2),'g'); hold on;
plot(cidade9(:,1), cidade9(:,2),'g'); hold on;
plot(cidade10(:,1), cidade10(:,2),'g'); hold on;
plot(cidade11(:,1), cidade11(:,2),'g'); hold on;

%plot(MMC1(:,1), MMC1(:,2),'b'); hold on;
%plot(MMC2(:,1), MMC2(:,2),'b'); hold on;
%plot(MMC3(:,1), MMC3(:,2),'b'); hold on;
plot(MFC(:,1), MFC(:,2),'r', 'Linewidth', 2); hold on;


%}
%{
figure
%%Plotagem dos precursos do estradao, antes da correção dos angulos e com os
precursos já ordenados
%{

plot(estradao1(:,1), estradao1(:,2),'r'); hold on;
plot(estradao2(:,1), estradao2(:,2),'r'); hold on;
plot(estradao3(:,1), estradao3(:,2),'r'); hold on;
plot(estradao4(:,1), estradao4(:,2),'r'); hold on;
%}
for i=1:4
estradao1=trataangulos(estradao1, 90);
estradao2=trataangulos(estradao2, 90);
estradao3=trataangulos(estradao3, 90);
estradao4=trataangulos(estradao4, 90);

end
%%Plotagem dos precursos do estradao, depois da correção dos angulos e com os
precursos já ordenados

plot(estradao1(:,1), estradao1(:,2),'g'); hold on;
plot(estradao2(:,1), estradao2(:,2),'g'); hold on;
plot(estradao3(:,1), estradao3(:,2),'g'); hold on;
plot(estradao4(:,1), estradao4(:,2),'g'); hold on;



%}
%{
[ME1,confME1]=caminhomedio(estradao1,0,estradao2,0,1);
[ME2,confME2]=caminhomedio(estradao3,0,estradao4,0,1);

[MFE,confMFE,dMFE]=caminhomedio(ME1,confME1,ME2,confME2,1);


plot(ME1(:,1), ME1(:,2),'b'); hold on;
plot(ME2(:,1), ME2(:,2),'b'); hold on;
```

```
for i=1:length(MFE)
    line([MFE(i,1) MFE(i,1)], [MFE(i,2)-((confMFE(i)/2))*10
MFE(i,2)+((confMFE(i)/2))*10],'Linewidth',2); hold on;
end

%plotagem dos precursos iniciais e da média final no trilho do estradao
plot(estradao1(:,1), estradao1(:,2),'g'); hold on;
plot(estradao2(:,1), estradao2(:,2),'g'); hold on;
plot(estradao3(:,1), estradao3(:,2),'g'); hold on;
plot(estradao4(:,1), estradao4(:,2),'g','displayname','Trajetos de GPS'); hold on;


plot(MFE(:,1), MFE(:,2),'r','LineWidth',2); hold on;
title ('Caminho Médio de Estradão');
xlabel('Latitude');
ylabel('Longitude');
disp( sprintf( 'A distância do caminho médio de cidade é de %s kilómetros', dMFE ) );



for i=1:length(MFE)
    confMFE(i)=1000*deg2km(confMFE(i),'earth');
    confMF(i)=1000*deg2km(confMF(i),'earth');
end

save inicialvariables1.mat cidade1 cidade2 cidade3 cidade4 cidade5 cidade6 cidade7
cidade8 cidade9 cidade10 cidade11
save inicialvariables2.mat estradao1 estradao2 estradao3 estradao4
save variables1.mat MC1 confMC1 dMC1 MC2 confMC2 dMC2 MC3 confMC3 dMC3 MC4 confMC4 dMC4
MC5 confMC5 dMC5
save variables2.mat MMC1 confMMC1 dMMC1  MMC2 confMMC2 dMMC2 MMC3 confMMC3 dMMC3
save variables3.mat MMMC1 confMMMC1 dMMC4 MFC confMF dMFC
save variables4.mat ME1 confME1 ME2 confME2 MFE confMFE dMFE

%}

%load das variáveis necessárias para o programa correr

load inicialvariables1.mat
load inicialvariables2.mat
load variables1.mat
load variables2.mat
load variables3.mat
load variables4.mat

%%Escrita dos ficheiros kml

%médias finais
MFEkml='MFEkml';
kmlwritepoint(MFEkml, MFE(:,1), MFE(:,2),
'Color','red','AltitudeMode','clampToGround','Description',num2str(confMFE(1,:)));
MFCkml='MFCkml';
kmlwriteline(MFCkml, MFC(:,1), MFC(:,2),
'Color','red','AltitudeMode','clampToGround','Width',5);

%trilhos originais
filename1='estradao1';
filename2='estradao2';
filename3='estradao3';
filename4='estradao4';
filename5='cidade1';
filename6='cidade2';
filename7='cidade3';
filename8='cidade4';
filename9='cidade5';
filename10='cidade6';
filename11='cidade7';
filename12='cidade8';
filename13='cidade9';
filename14='cidade10';
filename15='cidade11';


kmlwriteline(filename1, estradao1(:,1), estradao1(:,2),
'Color','cyan','AltitudeMode','clampToGround','Width',1);
```

```
kmlwriteline(filename2, estradao2(:,1), estradao2(:,2),
'Color','cyan','AltitudeMode','clampToGround','Width',1);
kmlwriteline(filename3, estradao3(:,1), estradao3(:,2),
'Color','cyan','AltitudeMode','clampToGround','Width',1);
kmlwriteline(filename4, estradao4(:,1), estradao4(:,2),
'Color','cyan','AltitudeMode','clampToGround','Width',1);
kmlwriteline(filename5, cidade1(:,1),   cidade1(:,2),
'Color','cyan','AltitudeMode','clampToGround','Width',1);
kmlwriteline(filename6, cidade2(:,1), cidade2(:,2),
'Color','cyan','AltitudeMode','clampToGround','Width',1);
kmlwriteline(filename7, cidade3(:,1), cidade3(:,2),
'Color','cyan','AltitudeMode','clampToGround','Width',1);
kmlwriteline(filename8, cidade4(:,1), cidade4(:,2),
'Color','cyan','AltitudeMode','clampToGround','Width',1);
kmlwriteline(filename9, cidade5(:,1),   cidade5(:,2),
'Color','cyan','AltitudeMode','clampToGround','Width',1);
kmlwriteline(filename10, cidade6(:,1), cidade6(:,2),
'Color','cyan','AltitudeMode','clampToGround','Width',1);
kmlwriteline(filename11, cidade7(:,1), cidade7(:,2),
'Color','cyan','AltitudeMode','clampToGround','Width',1);
kmlwriteline(filename12, cidade8(:,1), cidade8(:,2),
'Color','cyan','AltitudeMode','clampToGround','Width',1);
kmlwriteline(filename13, cidade9(:,1),   cidade9(:,2),
'Color','cyan','AltitudeMode','clampToGround','Width',1);
kmlwriteline(filename14, cidade10(:,1), cidade10(:,2),
'Color','cyan','AltitudeMode','clampToGround','Width',1);
kmlwriteline(filename15, cidade11(:,1), cidade11(:,2),
'Color','cyan','AltitudeMode','clampToGround','Width',1);
```

Função analisa_nmea_file

```
%%ANALISA_NMEA_FILE
%Esta funcao percorre o ficheiro NMEA, procura as linhas GPGGA, e daí
%retiramos os valores de latitude, longitude e altitude para análise dos
%percursos que fizemos.
%Input: nome do ficheiro
%Output: matriz com os dados do recetor [lat long alt]

function [dados_recetor] =analisa_nmea_file(nome)
velocidade=[];
fileID = fopen(nome, 'r');
if fileID <0
    error('error opening file \n');
end

aux =0;
flag_GPGGA=0;
linha=[];
dados_recetor=[];   %lat long alt
linha = fgetl(fileID);
i=1;

while i~=length(linha)
        if linha(i)==',' && linha(i+1)==','
            for j = length(linha):-1:i+1
                linha(j+1)=linha(j);
            end
            linha(i+1)=' ';
        end
        i=i+1;
end

while true

    linha = fgetl(fileID);
    tamanho = size(linha);

    if linha==-1
        break;
    end

    [address_field, linha]=strtok(linha, '$,');
```

```
    if strcmp(address_field, 'GPGGA')
         aux=aux+1;
        flag_GPGGA=flag_GPGGA+1;
        %UTC
        [str_utc, linha] = strtok(linha,',');

        %latitude
        [str_lat, linha] = strtok(linha, ',');
        [lat_sign, linha]= strtok(linha, ',');
        lat_aux=str2double(str_lat)/100;
        lat_deg=fix(lat_aux);
        lat_min_aux=(lat_aux-lat_deg)*100;
        lat=[lat_deg, lat_min_aux];
        lat=dm2degrees(lat);

        if strcmp(lat_sign, 'S')
            lat=-lat;
        end
if(lat==0)
    lat=[];
end
        %longitude
        [str_long, linha] = strtok(linha, ',');
        [long_sign, linha]= strtok(linha, ',');
        long_aux=str2double(str_long)/100;
        long_deg=fix(long_aux);
        long_min_aux=(long_aux-long_deg)*100;
        long=[long_deg, long_min_aux];
        long=dm2degrees(long);

        if strcmp(long_sign, 'W')
            long=-long;
        end
        if(long==0||lat==0)
       long=[];
       end

        %quality indicator
        [QA, linha] = strtok(linha, ',');

        %num sat usados
        [num_sat, linha] = strtok(linha, ',');

        %HDOP
        [HDOP, linha] = strtok(linha, ',');

        %altitude
        if strcmp (QA, '1') || strcmp (QA, '2') || strcmp (QA, '3') || strcmp (QA, '4')
|| strcmp (QA, '5')
        [str_alt, linha] = strtok(linha, ',');
        alt=str2double(str_alt);

        if(long==0||lat==0||alt==0)
        alt=[];
        end
        dados_recetor(aux,1)=lat;
        dados_recetor(aux,2)=long;
        dados_recetor(aux,3)=alt;

        end


    end

end
end
```

Função caminhomedio

```
function [media,confianca, dtotal]=caminhomedio(cam1,conf1,cam2,conf2, ponderacao)
%CAMINHOMEDIO Calcula caminhos médios e confiança nos caminhos
```

```
%   Esta função aplica o algoritmo Dynamic Time Warping para calcular o
%   caminho médio entre 2 caminhos. A matriz de distâncias não é de facto
%   calculada, mas apenas as partes necessárias para encontrar o "caminho"
%   de menor distância. É também calculada uma estimativa da confiança que
%   podemos ter em cada ponto do caminho médio. Esta confiança é calculada,
%   quando já estamos a fazer "médias de médias", das confianças de
%   caminhos anteriores

%Vamos trabalhar apenas com latitudes e longitudes, dado que isto permite
%uma aceleração dos cálculos, e os dados de altitude não são necessários,
%dado que podemos, ao visualizar os dados no Google Garth, "fixar" os dados
%à altitude do chão
cam1(:,3)=[]; cam2(:,3)=[];
%% Encontrar o caminho pela matriz

%O vetor caminho é o vetor onde vão ficar as posições por onde o "caminho
%de menor distância" passa na matriz de distâncias
caminho(1,:)=[1 1];
i=1; j=1; a=2; %contadores

while i<(length(cam1)-1) && j<(length(cam2)-1) %enquanto não chegar ao fim da matriz
    %vai-se calcular a distância dos pontos à direita, abaixo e na
    %diagonal, partindo do canto superior esquerdo. As distâncias
    %calculadas não são euclideanas, mas sim entre coordenadas, que têm uma
    %fórmula mais complexa

    direita=distance(cam1(i,:),cam2(j+1,:));
    diagonal=distance(cam1(i+1,:),cam2(j+1,:));
    baixo=distance(cam1(i+1,:),cam2(j,:));


    %se a distância menor das 3 for à direita, ir para a direita
if direita<diagonal && direita<baixo
    caminho(a,:)=[i,j+1];
    j=j+1;
    a=a+1;
    continue
end
    %se a distância menor das 3 for na diagonal, ir para a diagonal
if diagonal<direita && diagonal<baixo
    caminho(a,:)=[i+1,j+1];
    i=i+1;
    j=j+1;
    a=a+1;
    continue
end
    %se a distância menor das 3 for abaixo, ir para baixo
if baixo<diagonal && baixo<direita
    caminho(a,:)=[i+1,j];
    i=i+1;
    a=a+1;
    continue
end
%Por vezes, por razão ainda desconhecida, o ciclo ficava preso. Para
%resolver isto temos a condição de que, não sendo nenhuma escolhida, iremos
%avançar na diagonal
 caminho(a,:)=[i+1,j+1];
 i=i+1;
 j=j+1;
 a=a+1;
end

%% Calcular o caminho médio e a confiança no caminho

media=zeros(length(caminho),2);
dtotal=0;
for i=1:length(caminho)
    %Os pontos do caminho médio estão a meio da linha que une pontos
    %"correspondentes" dos dois caminhos. Tem-se em conta que alguns
    %caminhos são menos "importantes", por isso faz-se uma ponderação
    media(i,1)=(cam1(caminho(i,1),1)+ponderacao*cam2(caminho(i,2),1))/(1+ponderacao);
    media(i,2)=(cam1(caminho(i,1),2)+ponderacao*cam2(caminho(i,2),2))/(1+ponderacao);

    if i>1
    dist=distance(media(i,1), media(i,2), media(i-1,1), media(i-1,2),'degrees' );
    dtotal=dtotal+deg2km(dist, 'earth');
    end
```

```
    %Isto é o calculador de confiança do trajeto. Se estivermos a tirar a
    %média de dois trajetos GPS (não são fornecidos valores de confiança
    %dos caminhos, logo conf1==0 (e conf2==0), o valor de confiança de um ponto é
    %inversamente proprorcional à distância dos dois pontos de onde a média
    %foi tirada. No caso de estarmos a fazer uma "média de caminhos
    %médios", são também fornecidos à função um indicador das distancias
    %entre os pontos de onde as médias foram tiradas. Nesse caso, faz-se,
    %para cada ponto, uma média dos valores de distância de cada ponto
    if conf1==0
        confianca(i)=distance(cam1(caminho(i,1),1), cam1(caminho(i,1),2),
cam2(caminho(i,2),1), cam2(caminho(i,2),2) );
    else
        confianca(i)=(conf1(caminho(i,1))+conf2(caminho(i,2)))/2;
    end
end
confianca=confianca';
media(:,3)=[0]; media(:,3)=[0];
```

Função ordem

```
function [cam ] = ordem( cam )
if cam(1,1)>cam(length(cam),1)
    cam= flipud (cam);
end
```

Função Trataangulos (tratamento de dados)

```
function [ cam ] = trataangulos( cam , angulomin)
%TRATAANGULOS elimina os pontos que têm um angulo maior que o angulo minimo
%entre eles.
%Esta função elimina os pontos angulosos do caminho, que normalmente são
%associados a erros de medição do gps quando estamos parados ou por causa
%de obstáculos no sinal.
for i=2:(length(cam)-1)
%cálculo de dois vetores entre tres pontos consecutivos para calcular o angulo entre
eles
vettras=[(cam(i-1,1)-cam(i,1)) (cam(i-1,2)-cam(i,2)) ];
vetfrente=[(cam(i+1,1)-cam(i,1)) (cam(i+1,2)-cam(i,2)) ];
%calculo do angulo entre os vetores
angulos(i)=( acos(min(1,max(-1, vettras(:).' * vetfrente(:) / norm(vettras) /
norm(vetfrente) ))))*180/pi;
end


a=1;
for i=1:length(angulos)
    %comparação e remoção dos pontos com angulos inferiores ao angulo
    %minimo
    if angulos(i)<angulomin
    remover(a)=i;
    a=a+1;
    end
end


cam(remover,:)=[];
%{
b=cam;
a=1;
for(i=1:length(b))
    for(j=1:length(b(1,:)))
        if(b(i,j)~=0)
            cam(a,1)=b(i,1);
            cam(a,2)=b(i,2);
            cam(a,3)=b(i,3);
```

```
            a=a+1;
        end
    end
end
  %}
```

Função Tratazerointervalos

```
function [latdeg,longdeg] = tratazerointervalos( dados_recetor )
%Funcao que tira os pontos que são erros de medicao do gps

%%eliminar os pontos que estão na origem
a=1;
for(i=1:length(dados_recetor))
    for(j=1:length(dados_recetor(1,:)))
        if(dados_recetor(i,j)~=0)
            dados_recetor1(a,1)=dados_recetor(i,1);
            dados_recetor1(a,2)=dados_recetor(i,2);
            dados_recetor1(a,3)=dados_recetor(i,3);
            a=a+1;
        end
    end
end

latdeg=dados_recetor1(:,1);
longdeg=dados_recetor1(:,2);
alt=dados_recetor1(:,3);
end
%{
%% converter as coordenadas de graus para XYZ
for(i=1:length(dados_recetor1(:,1)))
latrad(i,:) = deg2rad(latdeg(i,:));
longrad(i,:) = deg2rad(longdeg(i,:));
[X(i,:),Y(i,:),Z(i,:)]=RadparaXYZWGS84( latrad(i,:),longrad(i,:),alt(i,:));
end

%% retirar pontos que são erros de localização
X1=X;
Y1=Y;
Z1=Z;
for(i=1:size(X)-1)
if(abs(X(i+1,:)-X(i,:))>5||abs(Y(i+1,:)-Y(i,:))>5||abs(Z(i+1,:)-Z(i,:))>5)

    X1(i,:)=[];
    Y1(i,:)=[];
    Z1(i,:)=[];

end
end
   %}
```