Universiteit Leiden
Opleiding Informatica

Analysis and Visualisation

of Data of an Outdoor

Sports Mobile Application

Name:	Mark Post
Date:	23/06/2015
1st supervisor:	Dr. Michael Emmerich
2nd supervisor:	Dr. Wojtek Kowalczyk

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Almost everyone has a smartphone these days. All these phones have a GPS and that can be used to record a track you walk, run or bicycle. There are many application that for all mobile platforms that do that, but the ones we reviewed did not do any trend analysis. In this thesis we used the data recorded by an external application, imported it into our application, analyse it and try doing trend analysis.

The project has three parts. We need to store the data somewhere first before using it. So the first part is designing a database. The resulting database is a compact database that can run on a smartphone. After storing the data, we can use it. In the second part we analyse the data. This is called time series analysis. Time series analysis is done by linear regression, Gaussian processes (kriging) and a combination of both. The final part of the thesis is about visualising the created regression model and other statistics generated from the imported data.

Contents

Chapter 1

Introduction

These days most people have a smartphone, a small computer with a lot of functionality. For example the GPS is a wonderful part. It can track all your movement. We can use the phone to track our movements through space. During this we store that location data on the phone. And then? What can we do with that data?

There are many sport and activity tracking mobile applications available. The application stores of all mobile operating systems are filled with them. For this project we will review six popular ones. Those are Endomondo [11], MapMyRun [8], Nike+ [9],RunKeeper [7], Runtastic [6] and Miles Calls [3]. All applications can track your moving activities, such as running or bicycling, and most of them can do some simple analysis with the collected data. One of the reviewed applications can only collect the data and do not do any detailed analysis. That is the application Miles Calls. This project is done in external collaboration with Dostware, the developer of Miles Calls. We will use the data that is collected by using the Miles Calls application and analyse it. None of the reviewed applications analyses trends in the gathered data. We will do that when analysing it.

1.1 Research Questions

The Miles Calls application collects valuable information and it does not analyse it. In this project we will use that collected data and do something useful with it: we will analyse it and try to find trends. Before we analyse it, we need to structure and store the data in a database. Then we can analyse it in some way. And at the end, we need a way to show the results. For this project we will develop a mobile application that imports the data from the Miles Calls application, stores the data, analyses and finally visualises it. So this project has three main parts. We can formulate the following main research questions:

- How can we organize the collected data in an SQLite database, taking into account restrictions of mobile applications?
- What meaningful information can we compute from location and time data from a walked, ran or bicycled tracks?
- How can we present this in an Android application to the user?

1.2 Thesis Overview

In this thesis we will start by introducing the six activity tracking applications including the Miles Calls application, the problem and research questions. Three following chapters contain the three parts of this research. We will start by introducing the used database design, then we will describe the time series analysis methods used to analyse the data and finish with the visualisation of the found results. This thesis ends with the conclusion and outlook.

Chapter 2

Activity Tracking Applications

In the world of tracking your activity there are many mobile applications available. In this project we will first review five popular ones: Endomondo, MapMyRun, Nike+, RunKeeper and Runtastic. After that we will review Miles Calls. We only took a look to the free version of the applications and looked into the functionality they provided without using wearables. Almost all can track more variables when used with wearables.

2.1 Analysis of Applications

All analysed applications have similar functionality with a slightly different user interfaces. They all record a track by using the phones GPS and store that data. Most of them also provide the user with a few live statistics, which means the application shows statistics that are updated in real time with the incoming data. After finishing the track all applications calculate a variety of statistics.

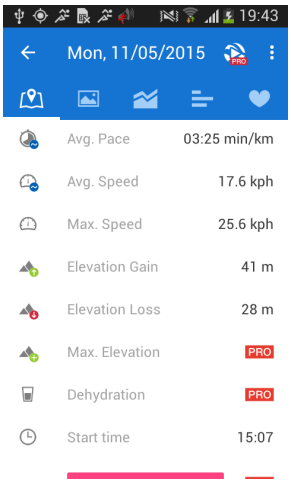

Figure 2.1: Statistics of a track in the Runtastic sport tracking application.

The statistics the other applications provide are similar to those in figure 2.1. It is mostly visualised in a table. The statistics they provide are the maximum and average speed and pace of a track, the maximum, minimum and average elevation, gained and lost elevation during the track and some provide the user with the burned calories during doing the activity.

We tested only the free versions of the applications. So we were limited by what the applications provided in the free version. All applications provided the user with more statistics in the paid version of the applications: for example they only allow the user to use the application with wearables when the paid version is bought. Also some of the statistics that are provided in the free version of application A are only available in the paid version of application B.

All five applications use graphs to visualise one or more statistics over the course of a track. The graphs look slightly different in each application depending on the used user interface. The applications usually use the speed or pace over the course of the track as data for the graph. Some of the applications also show the elevation over the course of a track in a graph.

Besides showing the statistics of a track in tables or graphs all applications also show it on a map. All except one of the analysed applications show the track in one colour on the map, as seen in figure 2.2, left handside. One of the applications, Runtastic, changes the colour of the track on the map based on the speed/pace of the user at that point of the track. This is shown in figure 2.2, right handside.

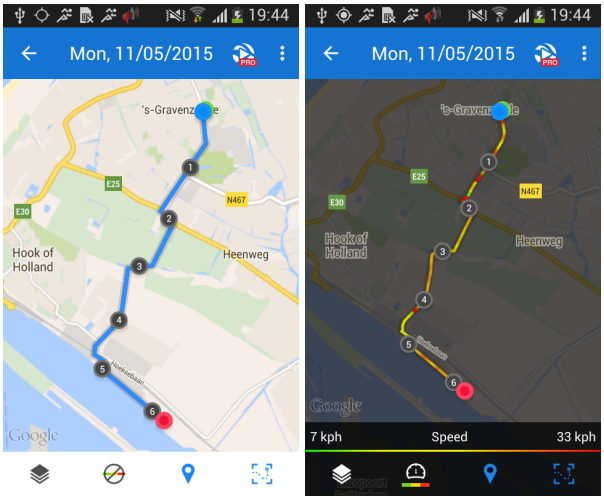

Figure 2.2: Screenshots of a track in Runtastic. The left one shows the track in one color, the right one shows the track in a color based on the speed at that point of the track.

2.2 Miles Calls

In this project we will work with the application Miles Calls. The application is developed by Dostware in Dortmund in Germany. The application has less functionality compared to the applications described in the section above. It can only collect the data from the track that is walked, ran or bicycled and it does not provide functionality to do post-tracking analysis. The application cleans the data as much as possible before exporting it. It can only show a map of the track. Figure 2.3 is a screenshot of the home screen of the application.

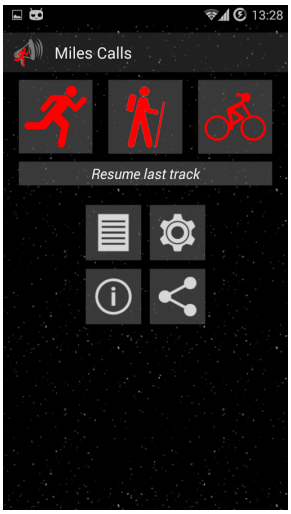

Figure 2.3: Screenshot of the Miles Calls Application.

2.3 Related Work

Computer science is more and more used in sport and acitivity tracking. These areas are called sport informatics. Sport informatics is an emerging area in computer science. The recent publication by Link and Lames [10] describes this field.

Other research looks into using computer science for training plans, sport gear and even the biological themes. Also a book about computer science in sport is written. It is called "Computer Science in Sport Research and Practice" and it is written by Arnold Baca [1].

Chapter 3

Database Design

3.1 Definitions

Before we start with the database design, some explanation of the used vocabulary is required. A track is in this project a series of points along a path. These points are gathered while the user is in motion. The set of points gathered is a record where you have been. This information can later be used to determine your path and speed. A track is recorded in one session.

A trip in this project is a set of tracks. It contains one or more tracks. These tracks can be recorded during a part of a day as well as a longer period. For example when you go on a biking holiday you may bicycle for multiple days. With a trip you can organize your tracks and keep the ones from that holiday together.

In running the word pace is used often. Pace is defined as the duration in minutes to run one kilometre or mile (in this project kilometre). So with an eight minute pace the runner runs one kilometre or mile in eight minutes. [5]

3.2 Requirements Database

The database has a wide variety of requirements. There are technical requirements such as what software it should run on and functional requirements such as what data is stored in the database. The most important question here is: ”What is the purpose of the database?”. This database will be used to store the data that is parsed from the imported files from the Miles Calls application. After importing a file, the application will calculate some statistics such as maxima, minima, medians and averages of speed, pace and elevation. It will calculate these for each trip and track. These statistics should also be stored in the database.

The application will be built for Android and for databases in Android is usually SQLite used. So the database design should be converted to SQLite. The database will run on a mobile device, so it should also be lightweight.

The input data of the database are files that are generated by the Miles Calls application. It uses the GPX file format. This is a format that is based on XML and it is a data format to store and interchange GPS data. Each data file contains one track and it contains a special tag that specifies the activity type of the track is. A track contains one or more segments. If the user uses the pause option in the Miles Calls application, then a new segment is created. A track segment contains a sequence of locations called track points. An example of a track point is shown in listing 3.1. Each track point contains location data as longitude and latitude values, a timestamp and the elevation at the measurement point. In the database we want to make the distinction between three main components: locations, tracks and trips. Trips are used to organize tracks.

```
1 <trkpt lat="52.1775215" lon="4.5797361">
2   <time>2014  0 1  0 2 T 1 7 :56:13+01:00</time>
3   <ele>68.83830758305048</ele>
4 </trkpt>
```

Listing 3.1: A track point in the GPX file format

A location is a measurement point or track point as described in the last paragraph. Each location is part of a track. So each location object contains information of which track it is part. Two or more locations make a track. A track contains information such as the data that it was walked, ran of bicycled. Each track is part of a trip. So each track contains information of which trip it is part. A trip contains one or more tracks. Each trip should have tracks from only one kind of activity type. It has also a start date (date earliest track) and an end date (date latest track).

When the application is finished with importing and storing the data of a file, it will calculate some statistics. It will calculate statistics similar to the ones that other activity tracking applications provide, for example maxima, minima, medians and averages of elevation, speed and pace. These statistics will be generated for the newly imported track as well as updating the global or overall statistics of the parent trip with the statistics of the new track.

A wide variety of queries should be able to be executed on this database. Firstly, it should store the new imported track with its locations and its statistics. Secondly, it should update the database entry of an existing trip or store a new entry when a it is created. Next, it should be able to delete a single or multiple locations, tracks and trips. It should also be possible to empty the whole database at once. So deleting all locations, tracks and trips at once. Last, it should be possible to retrieve data from the database. For this there are three cases. Case one is that the application needs to visualise some statistics from trip or track data. This is mainly just a single entry from the database table. The other case is when the application either needs a list of trips or tracks or when it needs data from all tracks to analyse that data. In that case the query will retrieve many entries. The last requirement for the database is that each location and each track should have the option to turn that location respectively track on or off. This is in case of measurement error or when the user want to exclude a track from analysing it.

3.3 ER Diagram in Databases

The ER diagram is in database design a model to describe the structure of a database. It uses entities, attributes and relationships to do this. An entity is an object from the real world like a car, person or area. A set of similar entities is called a entity set. Every entity is described by a set of attributes. Each entity of an entity set have the same attributes. For example the entity student has the attributes student number, name and year. So each entity in the entity set of student has these attributes. Each attribute has a domain of possible values. For example the student number is an unique five digit number, a number between 00000 and 99999. A key is a minimal set of attributes that can be used to uniquely identify an identity in a set. We set one key as primary key. In the student example is the student number a good primary key.

The association between two or more entities is called a relationship. The entity set of students could be related to the entity set of classes. Figure 3.1 illustrates this. Here each student attends classes. This is an example of a many-to-many relationship: each student attends many classes and each class has many students. The other two types are one-to-one, where each entity is related to one other entity, and one-to-many, where one enitiy is related to many entities.

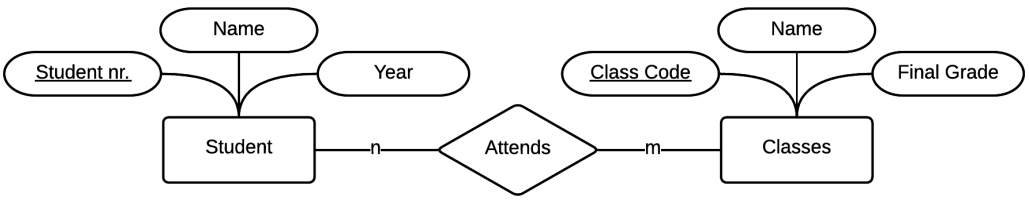

Figure 3.1: Example of a relationship between two entities. The primary key of Student is Student nr. and of Classes it is Class Code

We can also use a ISA (read: is a) hierarchy. Figure 3.2 shows an example of this. Each vehicle is either a car, train or bus. The entity Vehicle is a set of all vehicles and it has the attirbutes that all vehicles have. The entitites Car, Train and Bus have the attirbutes that only cars, trains and busses have [13].

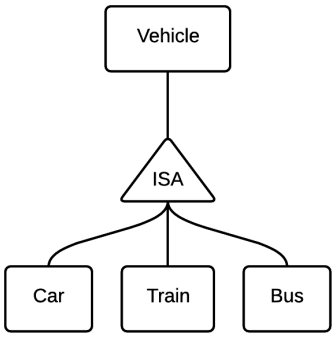

Figure 3.2: Example of a ISA class hierarchie

3.4 Final Design

Based on the requirements earlier in this chapter we have created the database design shown in figure 3.3. The ER diagram has three main entities: Trip, Track and Location. The Trip and Track entities have both an entity with their statistics information. In the diagram Max, Min, Ave, Med, Loc and Nr refer to respectivily maximum, minimum, average, median, Location and number.

The Trip entity has four attributes: Trip ID, Name, Start Date and End Date. The Trip ID is the primary key and will be used to identify a trip and used to relate a track to the trip. The Trip entity has a one-to-one relation with the Trip Information entity. This entity contains the statistics of a trip. Its attributes are Trip ID, Speed Max, Speed Ave, Speed Med, Elevation Max, Elevation Min, Elevation Ave, Elevation Med, Distance and Duration. In the database will only be the speed stored in meters per second and not the pace, because the pace can easily be calculated from speed.

The Track entity has three attributes: Track ID, Date, and File Name. The Track ID is the primary key for this entity. In File Name is the file name of the imported file stored. This is used to prevent importing a file twice. The Track entity has a one-to-one relationship with the Track Information entity. This entity has almost the same attributes as the Trip Information Entity. However, it has the attribute Track ID instead of Trip ID and it has the Turn On attribute. The Turn On attribute will be used to turn on and off tracks to include or exclude then from the application analysing them.

Between the Trip and Track entity is an ISA activity has relationship. The relationship is a one-to-many; one trip has (one or) many tracks. The ISA structure is used to make sure that each Trip entity has only Track entities from one of the three activity types.

The Location entity has seven attributes: Location ID, Longitude, Latitude, Segment Nr, Elevation, Time-stamp and Ignore Loc. The Location ID is the primary key for this entity. The Longitude, Latitude and Elevation attributes are used to store the geographical location of this location. The Time-stamp is used to store the time of the track point. The Segment Nr stores the segment of which segment of the track this location of part. Lastly, Ignore Loc gives the possibility to turn on and off a location in case of a measurement error. The Location entity has a one-to-many; one track has many locations relationship. Duplicates of a location are allowed, but they are very unlikely to occur. Duplicate locations are in this case two or more different location objects with exactly the same longtitude, latitude and elevation.

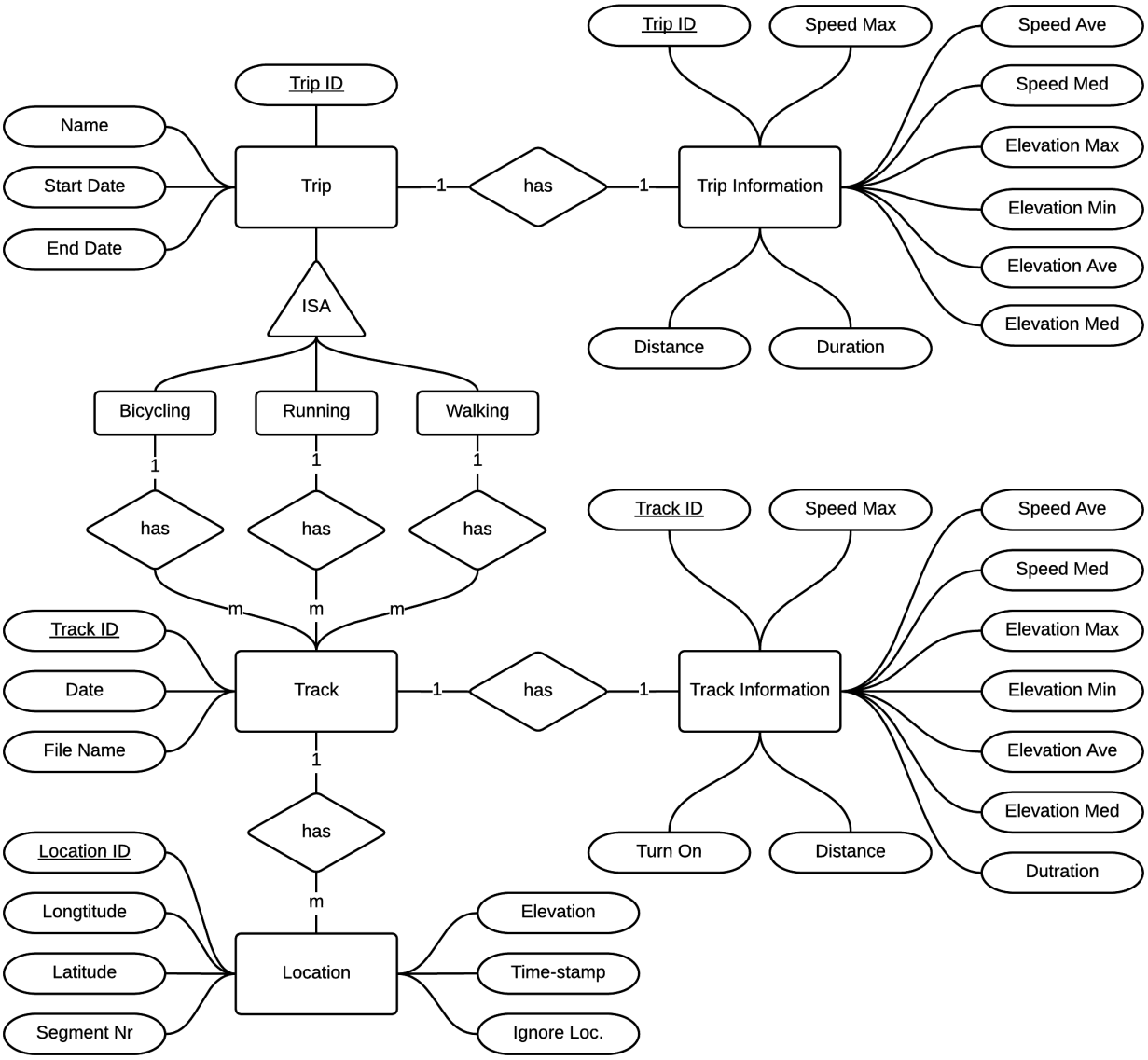

Figure 3.3: ER diagram of final design of the database

3.5 Discussion

When we converted this final database design from the ER diagram to tables in SQLite, we combined the Trip and Trip Information and Track and Track Information entity to each one table. We did this to make the actual database more compact. However, we kept the ER diagram like this to keep it more organized.

In the actual database we converted the ISA structure and relationship between the Trip and Track entity to a field activity type in both the Trip and Track table. The requirement that each trip contains only tracks from one specific activity type is checked in the software of the application. The application will check which trips are allowed to store a track.

Chapter 4

Function Approximation

We can analyse the data that is stored in the database. When we use the average or median speed of each track from a specific activity type from the database, then we get a sequence of measurement points over time. This is called a time series. In this chapter we want to get useful information from this data. So we will analyse this data. This is called time series analysis. [20] Function approximation is one type of time series analysis. There are many methods to do function approximation. We started by using linear regression. This was mainly to get into regression and get things working in the mobile application.

4.1 Linear Regression

Assume we have dataset A containing the average speeds of all or just by the user selected tracks. This dataset contains m pairs of data points (x_i, y_i), where x is time and y is speed. Each datapoint is the time (days since earliest track) and average speed of a track. So $A = \{(x_i, y_i) : i = 1, \ldots, m\}$. The goal of linear regression is to find a line 4.1 that fits the dataset best. Linear regression uses least squares fitting. This means that the overall solution has the smallest sum of the square of errors. It minimizes the overall sum of errors. Errors meaning here the distance between the line and the actual points of the dataset.

$$y = a + bx \tag{4.1}$$

The regression coefficients b [19] and a are calculated by the respectivily the equations 4.2 and 4.3.

$$b = \frac{ss_{xy}}{ss_{xx}} \tag{4.2}$$

$$a = \bar{y} - b\bar{x} \tag{4.3}$$

where

$$ss_{xy} = \sum_{i=1}^{m}(x_i - \bar{x})(y_i - \bar{y}) \tag{4.4}$$

$$ss_{xx} = \sum_{i=1}^{m}(x_i - \bar{x})^2 \tag{4.5}$$

In some of the formulas above are $\bar{x}$ and $\bar{y}$ used. They are the average of all x's respectively all y's. So to obtain $\bar{x}$ them we need to add up all x-values and divide that by the number of values. That is done in the following equation 4.6. [15]

$$\bar{x} \equiv \frac{1}{n}\sum_{i=1}^{m} x_i \tag{4.6}$$

$$\hat{y}_i = a + bx_i \tag{4.7}$$

By using these equations the best fitting line 4.1 can be found. The $\hat{y}_i$ is the vertical coordinate for each x-coordinate x_i with 4.7. [17] This is easy to implement and use in the application. However, a best fitting line gave not the best result for the used data. A better analysing approach was needed.

4.2 Ordinary Kriging

There are many methods that can be used to interpolate time series data. Ordinary kriging is one of them. Machine learning methods, like neural networks, could possibly also be used. However, for this data we want an uncertainty predictionary estimation. Ordinary kriging offers this.

So, the second approach we used is ordinary kriging. Kriging is "optimal interpolation based on regression against observed z values of surrounding data points, weighted according to spatial covariance values". [2] Interpolation is estimating unknown values by using a set of known values. The estimated values for a kriging model with exponential kernel are within the range of the known values. [16] However, we also want to show a trend for the future. So we also used the model to calculate values greater than the greatest known value.

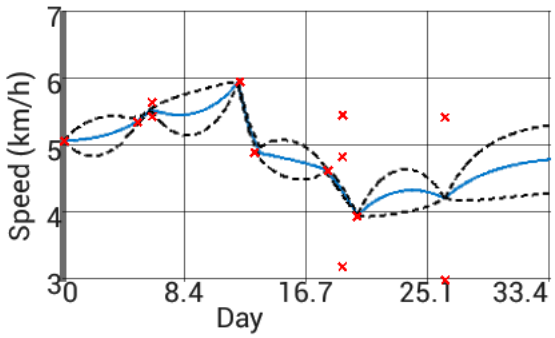

Figure 4.1: This is an example of one-dimensional kriging with an exponential kernel. On the y-axis is the speed in kilometers per hour and on the x-axis is the time in days. The red crosses the known values and the blue line are the estimated values obtained by interpolation. The top and bottom dashed lines is the likelihood of the estimated values.

Figure 4.1 illustrates an example of kriging in one dimension. The estimated values of the blue line are determined by interpolation of the known values, red crosses. The area between the blue line and the top and bottom dashed line is the confidence or likelihood of the prediction. When a point is further away from a known point, then the confidence decreases; the confidence range increases. On the other hand when a point is closer to a known value, then the confidence increases and the confidence range decreases.

There are a few types of kriging depending on the trend type. We choose ordinary kriging for this project. In ordinary kriging is a constant trend assumed. The base formula is shown in equation 4.8. In this formula is β the global constant trend and is r_x the local deviation.

$$\mu_x = \mathbb{E}(\mathcal{F}_x|A) = \beta + r_x \tag{4.8}$$

where

$$r_{\mathrm{x}} = \mathbb{E}(\mathcal{R}_{\mathrm{x}}|\mathrm{A}, \beta) = \sum_{i=1}^{m} \lambda^{(i)} \cdot c_\theta(\mathrm{x}, \mathrm{x}_i) \tag{4.9}$$

with

$$c_\theta(v, w) = exp(-\theta \cdot |v - w|) \tag{4.10}$$

and

$$[\lambda^{(1)}, ..., \lambda^{(m)}] = (\mathbf{y} - \mathbf{1}\hat{\beta}) \cdot \mathbf{C}^{-1} \tag{4.11}$$

The local deviation is a summation for all m data points. In the summation $\lambda^{(i)}$ is multiplied by exponential factor $c_\theta(\mathrm{x}, \mathrm{x}_i)$. This is shown in equation 4.9. Here is $c_\theta(\mathrm{x}, \mathrm{x}_i)$ the correlation between $\mathcal{F}_{\mathrm{x}}$ and $\mathcal{F}_{\mathrm{x,i}}$. The type of correlation is given as an assumption. A frequent choice is the exponential kernel $exp(-\theta \cdot |v - w|)$. $[\lambda^{(1)}, ..., \lambda^{(m)}]$ from 4.11 are the weights for each data point i. The matrix $\mathbf{C}$ contains the isotropic Gaussian correlation of each combination of pairs of x-values. This is shown in 4.12.

$$\mathbf{C} = \begin{bmatrix} c_\theta(\mathrm{x}_1, \mathrm{x}_1) & \dots & c_\theta(\mathrm{x}_1, \mathrm{x}_m) \\ \vdots & \ddots & \vdots \\ c_\theta(\mathrm{x}_m, \mathrm{x}_1) & \dots & c_\theta(\mathrm{x}_m, \mathrm{x}_m) \end{bmatrix} \tag{4.12}$$

$$\mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \tag{4.13}$$

In equation 4.8 β is used. Instead of using β we will use $\hat{\beta}$ instead. $\hat{\beta}$ is an estimation of β and can be determined by the following equation:

$$\hat{\beta} = \frac{\mathbf{1}^T \cdot \mathbf{C}^{-1} \cdot \mathbf{y}}{\mathbf{1}^T \cdot \mathbf{C}^{-1} \cdot \mathbf{1}} \tag{4.14}$$

The equations above are all the ones that are required to interpolate the values in between known values. However, the parameter θ is not explained here. That parameter will be explained in the next section. With the equations the blue line from figure 4.1 can be determined. To obtain the maximum and minimum likelihoofd values of the interpolated values are a few more equations needed.

$$t_{\mathrm{x}} = \mu_{\mathrm{x}} + \hat{s}(\mathrm{x}) \tag{4.15a}$$
$$b_{\mathrm{x}} = \mu_{\mathrm{x}} - \hat{s}(\mathrm{x}) \tag{4.15b}$$

Equations 4.15a and 4.15b are the base equations to obtain the y-values of the dashed lines in the figure 4.1. t_{x} is the top and b_{x} the bottom line. The equation 4.16 is used to calculate the likelihood at point x.

$$\hat{s}(x) = \hat{s} \cdot \left[1 - \mathbf{c}(x)^T \cdot \mathbf{C}^{-1} \cdot \mathbf{c}(x) + \frac{(1 - \mathbf{1}^T \cdot \mathbf{C}^{-1} \cdot \mathbf{c}(x))^2}{\mathbf{1}^1 \cdot \mathbf{C} \cdot \mathbf{1}}\right] \tag{4.16}$$

with

$$\mathbf{c}(x) = \left[c_\theta(x, x_1), ..., c_\theta(x, x_m)\right] \tag{4.17}$$

and

$$\hat{s} = \frac{(\mathbf{y} - \mathbf{1} \cdot \hat{\beta})^T \cdot \mathbf{C}^{-1} \cdot (\mathbf{y} - \mathbf{1} \cdot \hat{\beta})}{m} \tag{4.18}$$

Prediction of values with ordinary kriging is done in three phases. The first phase is the calibration phase. The parameters θ, β and $\hat{s}$ are in this phase calibrated and estimated. Their values are invariant with respect to $\mathcal{F}$. The calibration phase is the most time consuming phase of the whole process. The second phase is the training phase. In this phase are the weights for the linear predictor 4.8 with summation 4.9. The last phase is the prediction phase. In this phase are μ_x and $\hat{s}(x)$ calculated. [4, 14]

In matrix $\mathbf{C}$ are the values on the diagonal usually zero. However by adding a small value to those zero values on the diagonal, we can make the model work better. This is shown in equation 4.19. Here v is a very small positive value and δ (Kronecker symbol) is 1 if and only if $i = j$. Otherwise $\delta_{i,j}$ is 0. [12]

$$c_\theta(v, w) = exp(-\theta \cdot |v - w|) + v\delta_{i,j} \tag{4.19}$$

4.3 Estimation of θ

In the process of implementing ordinary kriging turned estimation the value of parameter θ out to be hardest and most time consuming part of the project. We ended up trying three different approaches. We started by setting a rough θ by hand. Then we tried to estimate the value by using maximum likelihood estimation. And we ended up by using cross-validation to estimate the value of θ.

4.3.1 Estimating θ by Hand

The first approach was to set a hardcoded θ by hand. The idea behind this approach was to experiment what the influence of the parameter on the model is and get a feeling what roughly an optimal value for θ is.

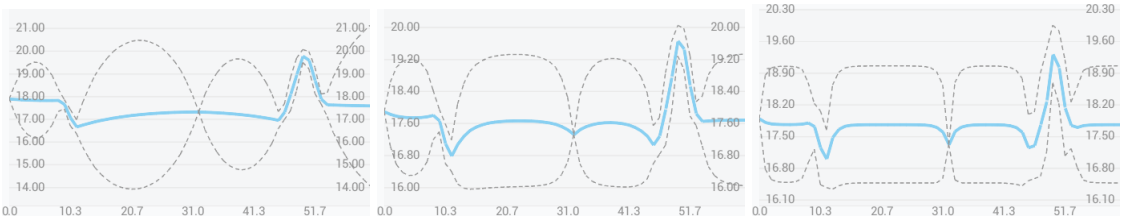

Figure 4.2: This illustrates the influence of parameter θ on the prediction. On the y-axis is the speed in kilometers per hour and on the x-axis is the time in days. Left to right: $\theta = 0.1$, $\theta = 0.4$ and $\theta = 1.0$. A smaller θ results in a smoother graph. With a bigger θ the predicted line goes faster to the global average.

Figure 4.2 shows three graphs. The parameter θ has a big impact how the graph looks. It controls how fast the interpolated line return to the global average $\hat{\beta}$. When the θ is small, then the model returns slower to the global average. This results in a smoother graph. When the θ is bigger, then the line returns quicker to the global average. The predicted line is in that case like a straight line with peaks where the known data points are.

However, estimating θ by hand and setting it hardcoded in the application is not an attractive option. Firstly, this is because finding the optimal theta by hand would cost a lot of time. Additionally, what is a optimal value for parameter θ. And lastly, the optimal value of θ varies per dataset.

4.3.2 Estimating θ by Maximum Likelihood Estimation

The second approach of estimating an optimal value for θ and the first approach to automate it was by using maximum likelihood estimation (MLE). It is a procedure of finding the value of a parameter by finding the maximum of a known likelihood distribution. [18]

$$l_\theta = m \log \hat{s}(\theta) + \log \det \mathbf{C}(\theta) \tag{4.20}$$

To estimate the value of θ with the maximum likelihood, we need to minimize the expression 4.20. To find the minimum we could use linear search, but that can take a long time. The search time depends on the step size of increasing the potential θ and the size of the search domain, the biggest θ that needs to be checked.

Another search algorithm is binary search. However, the normal binary search algorithm is not usable for us. Binary search requires the searched domain to be sorted. That is not here the case. But we can make binary search work for our minimization by modifying it. Listing 4.1 shows the modified binary search algorithm.

```
left = SMALLEST_POSITIV_VALUE;
right = 10;     //Bound search range
depth = 0;
maxDepth = 100; //Maximum depth recursion

while (depth < maxDepth) {
  //Calculate center and position closely left and right of center
  center = (left + right) / 2;
  smallStep = (right - left) / 100;
  leftOfCenter = center - smallStep;
  rightOfCenter = center + smallStep;

  //Calculate value for theta
  valueForThetaCenter = calculateValueForTheta(center);
  valueForThetaLeftOfCenter = calculateValueForTheta(leftOfCenter);
  valueForThetaRightOfCenter = calculateValueForTheta(rightOfCenter);

  if (valueForThetaLeftOfCenter < valueForThetaCenter) {
    //Minimum is left of current center
    right = leftOfCenter;
    depth++;
  } else if (valueForThetaRightOfCenter < valueForThetaCenter) {
    //Minuimum is right of current center
    left = rightOfCenter;
    depth++;
  } else {
    //Minimum is found. Further search is not needed
    return center;
  }
}
return center;
```

Listing 4.1: Pseudo code of modified binary search algorithm to find minimum

In the algorithm returns the function `calculateValueForTheta(mTheta)` the result of equation 4.20 with `mTheta` as θ. We want to find the minimum of the search domain with the algorithm. That is in this case `SMALLEST_POSITIV_VALUE` to 10. The domain is not sorted. When deciding on a center position, we do not know whether the minimum is left or right of it. To determine that, we need to do two extra calculations. We do a small step to the left and right of the middle and calculate the value for those θ's. When the value left of the center smaller is, then the minimum must be left of the center and we continue searching there. If the value right is smaller, then we continue there. In every iteration the binary search halves the range of the parameter. It assumes that the likelihood function 4.20 has only a single optimum.

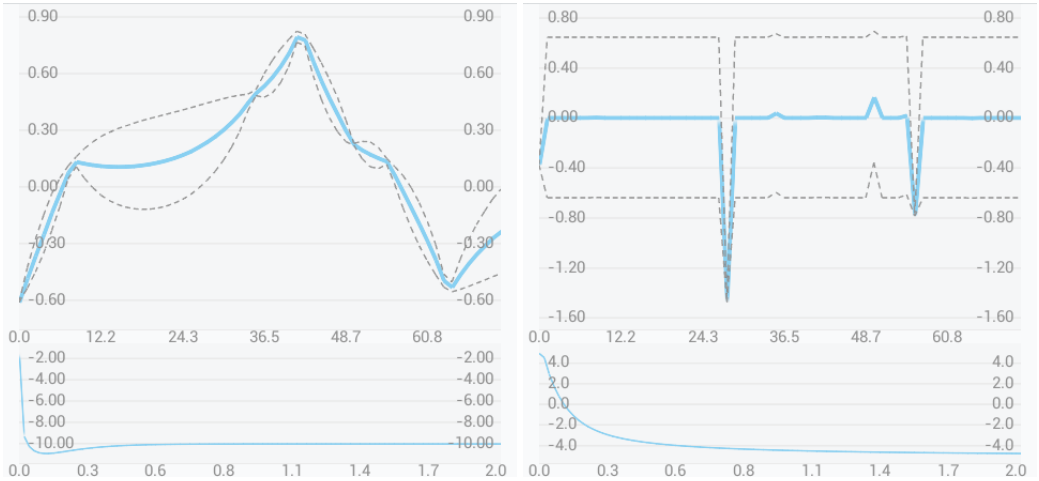

Figure 4.3: This is the interpolation of two different datasets by using maximum likelihood estimation to determine the optimal θ. The top graph is the model generated with ordinary kriging. On the y-axis is speed in kilometer per hour and x-axis the time in days. The bottom graph is the smallest median error rate plotted. On the y-axis is smallest median error rate and on the x-axis the used θ. Left, the minimization has a minimum on the search domain. Right, the minimization does not have a minimum.

The search algorithm and the estimation by maximum likelihood estimation work well as long as the minimization has only a single minimum. However, in about half of the cases when we tested it, the dataset did not have a minimum in the searched domain. An example of this is shown on the right of figure 4.3. The dataset of the left figure generates a nice minimum, but the dataset on the right not. So for this type of data, is maximum likelihood estimation not usable to estimate a reasonable θ.

4.3.3 Estimating θ by Cross-Validation

The third and last approach to estimate the optimal θ for a dataset, was by using leave-one-out cross-validation. The basic principle in this method is that we leave one data point out of your data set and then predict the data point with the model that is generated with the remaining data points. With a dataset of size m we need to repeat this m (number of datapoints) times so that every data point is left out exactly once. After we try to estimate the left out data point by interpolation, we can calculate the error rate. That is the difference or distance between the predicted value and the actual value of the data point. At the end we can determine the median error rate of the m individual error rates.

In this project we use this method to estimate the optimal θ for a given dataset. We repeat the describe procedure for each θ we want to check. In the application we checked all values from 0.01 to 2.00 with a 0.01 step size. So we did the procedure a 200 times. At the end we choose the θ with the smallest median error rate, because it is the most rebust method. Choosing average or mean are not attracftive, because uutliers could influence them.

During the implementation we did some experimenting with calculating the error rate. We tried to combine the distance between the predicted point and the actual point and maximum and minimum likelihood at that point. Listing 4.2 shows this.

```
y = calculateFx(actualXValue);
max_likelihood = Math.abs(calculateLikelihood(actualXValue, max));
min_likelihood = Math.abs(calculateLikelihood(actualXValue, min));
distance_y = calculateDistance(trainingData.get(i).yDataValue, y);
errorRate = WEIGHT_DISTANCE * distance_y
          + WEIGHT_LIKELIHOOD * (max_likelihood + min_likelihood);
```

Listing 4.2: Pseudocode of calculation likelihood

We did a small experiment to find out what the behavior of different weights in calculating the error rate is. We tested weights going from 0 to 1 with steps of 0.1. The combined weights were always 1. We did the experiment with two different datasets. Dataset 1 contains only hiking tracks and dataset 2 only bicycling datasets. (Datasets can be found in the appendix A) We tested θ's from 0.01 to 2.00. The results are shown in table 4.1. The results show that the

Weight dist.	Weight likel.	Sm. med. err. (1)	Found θ (1)	Sm. med. err. (2)	Found θ (2)
1	0	1.30192	2.00	0.50034	0.65
0.9	0.1	1.28306	2.00	0.75947	2.00
0.8	0.2	1.26419	2.00	0.99793	2.00
0.7	0.3	1.24533	2.00	1.23640	2.00
0.6	0.4	1.24792	2.00	1.47487	2.00
0.5	0.5	1.31288	0.78	1.62074	2.00
0.4	0.6	1.37377	0.13	1.85200	2.00
0.3	0.7	1.39998	0.11	2.13549	2.00
0.2	0.8	1.40673	0.09	2.36378	1.08
0.1	0.9	1.41668	0.08	2.57683	0.78
0	1	1.41697	0.07	2.77951	0.65

Table 4.1: Table of results experiment illustrating varying weights in calculation error rate.

behavior of different weights is different on each dataset. If the found θ is 2.00, then there is no smallest value on the chosen domain. A θ of 2.00 generates a graph as seen on the right figure in figure 4.3. That means that also with the cross validation the optimal θ overfits the dataset.

For most datasets cross-validation gives a 'good' optimal θ. Good meaning a θ that does not overfit the input data. We need here a balance between a statistically optimal and a solution that appears to be plausible to the user. The experimenting during the different approaches has shown that a user-friendly optimal θ should not be bigger than 0.6 or 0.7. With a bigger value the model starts to overfit. We need a solution in case there is not a smallest median error rate for the given error rate. We first limit the search domain to 0.01 to 1.00. Then we wrote an algorithm that chooses the θ with a median error rate approximately 0.1 higher than the one at $\theta = 1.00$. This algorithm is only used in case there is not a smallest value on the search domain found.

4.4 Combining Ordinary Kriging and Linear Regression

To better show the user the trend in his data we decided to create our own regression model by combining ordinary kriging with linear regression. With this model the user can better see and understand what his progress is. We note that this method is reminiscent of the so-called universal kriging with the difference that the linear trend model in our method in an given as a priori.

In this regression model we first calculate linear regression as explained in section 4.1. Then from all the y-values of the dataset is the corresponding y-value of linear regression subtracted. So the modified dataset is $B = \{(x_i, y_i') : i = 1, \ldots, m \wedge y_i' = y_i - \hat{y}_i\}$ where y_i is the y-value of the dataset and $\hat{y}_i$ is the value calculated by equation 4.7. Then the ordinary kriging model from section 4.2 and the θ estimation from section 4.3.3 are applied. At the end instead of equation 4.8, equation 4.21 is used. Here we take $\beta(\mathrm{x})$ as prior from linear regression, 4.22.

$$\mathcal{F}_{\mathrm{x}_i} = \beta(\mathrm{x}) + \mathcal{R}_{\mathrm{x}_i} \tag{4.21}$$

$$\beta(x) = b + ax \tag{4.22}$$

Figure 4.4 shows the result of combining ordinary kriging with linear regression. The light blue dashed line in the graph is the line generated by linear regression. The lines of the ordinary kriging are generated as described above.

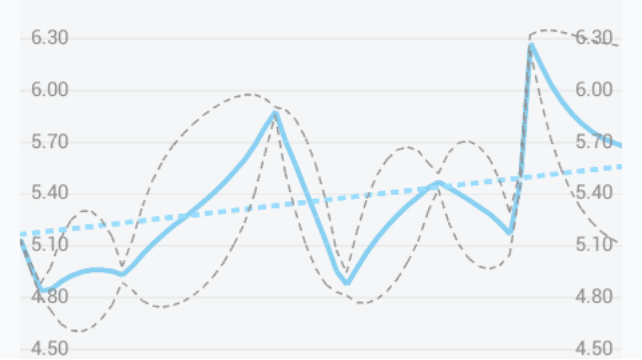

Figure 4.4: Graph showing result of combining ordinary kriging with linear regression.

4.5 Discussion

Finding a good working approach to estimate the parameter θ in ordinary kriging took the most effort. The datasets seemed too small and not suitable for the tried standard approaches. Maximum likelihood estimation worked for about half of the tested datasets. It relies too much on model assumption that the data is from a Gaussion process. Cross-validation later worked a lot better, but still could not find a good optimal θ sometimes. Although it worked better, cross validation is requires more computation power when datasets get bigger. The method can be slow on smartphones with a slower CPU. The number of times that the ordinary kriging model is generated is the number of θ's to be checked, 100 to 200 in our case multiplied by the size of the dataset, m. So with a dataset of size $m = 30$ and 100 θ's to be checked, the ordinary kriging model is generated for 3000 times.

Chapter 5

Visualising in a Mobile Application

The third part of the project is to visualise the imported data and the model generated. In this chapter we show the built application and explain our design choices. The application is build for the Android mobile platform. We choose Android, because it is one of the big three (Google Android, Apple iOS and Microsoft Windows Phone), it is easy to build an application for and the Miles Calls application is only available for Android.

5.1 Overview of Application

When the application is started, it shows the screen with the last imported track. A screenshot of this can be seen in figure 5.1. It shows a map and some statistics such as distance and duration of the track. Below the statistics is a button to view more details of the track. This brings the user to the Track Overview screen, which is later in this chapter described. The Last Track screen also shows some overall statistics of all the imported tracks. We choose the shows the last track to the user first, because it most likely the he wants statistics of and analyse that track when starting the application.

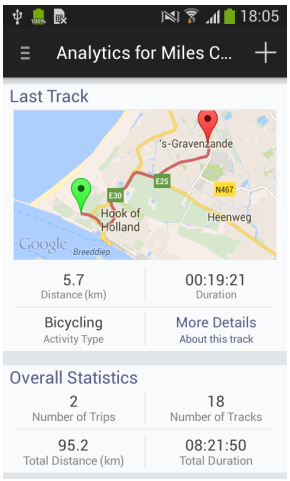

Figure 5.1: Screenshot of start screen of the application showing a map and statistics of the last imported track and overall statistics.

From every screen the user can navigate to the import track screen. Here we can navigate with a file browser to the GPX file to import it. We do not know in advance where the user stores his GPX files, so we give him the option to browse through the file system. A screenshot of this is shown in figure 5.2 left. The application first checks whether the file is a GPX file, then it parses it. If the file does not contain the activity type, then it asks the user to select the activity type as seen in figure 5.2 right. This to support GPX files from older Miles Calls versions. The next step is to select an existing trip to add the track to or create a new trip. So the user can organize his tracks. Then the track is stored in the database and the statistics described in the database design chapter are generated.

The All Trips screen is rather simple. It contains some statistics of all tracks of all trips and a list of all trips. The user can navigate here to a specific trip. The purpose of this screen is to let the user browse to the wanted trip. This screen is shown in figure 5.3 left image.

The Trip Overview has a similar design as the Trip Overview screen. It also has some statistics from all tracks of this trip, and a list of tracks of this trip. Each list entry shows its date, distance and duration. The purpose of this screen is to let the user browse to the wanted track. This screen is shown in figure 5.3 right image.

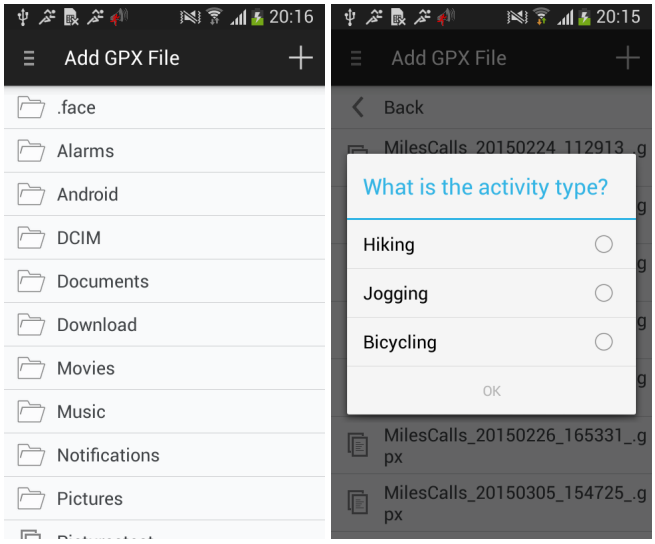

Figure 5.2: Screenshots of importing screen. The figure on the left shows the file browser and the one right shows selecting an acitivity type when the acitivity type is missing.

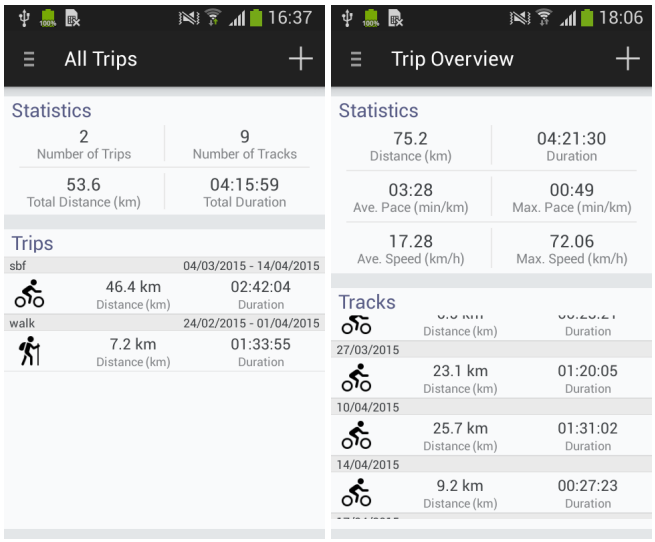

Figure 5.3: The left screenshot is from the All Trips screen and the right is a screenshot of the Trip Overview screen.

5.2 Overall Trends

The overall trends screen is where the results of the kriging are visualised. The main feature of here is the graph. In this graph is the model that is generated as described in the last chapter 4 plotted. This is shown in figure 5.4. We can see that equation 4.8 is plotted as the blue line. This is the interpolation and prediction. The likelihood of this prediction is plotted as a grey dashed line. The linear regression is the light blue dashed line. By default the y-axis is speed in kilometres per hour. We can switch to pace, minutes per kilometre, by pushing the button below the graph. Pushing the other button shows a screen to change the used dataset for the model. Also some statistics are shown below the graph.The graph is the most important part here. So the graph uses a big part of the screen. The buttons to switch between acitivities are at the bottom so the user can reach them easily.

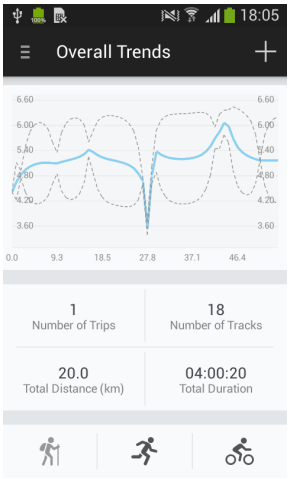

Figure 5.4: Screenshot of the Overall Trends screen.

5.3 Track Overview

The Track Overview is one of the most important parts of the application. It shows all information available of a track. The main parts here are the map and the graph. The map shows the track in either just a red line or a line colour that depends on the speed at that point. Yellow is the colour used for the average speed. A redder colour means that the speed at that point was slower than average and a greener colour means that the speed was faster than average. However the colours used can make the track has to see on the map, because roads can have similar colours. So we decided to add the option to turn the track to dark red. This makes viewing the track easier.

The graph has four options. It can show the speed and pace over the course of the track and it shows either a simplified, smoother graph or a graph with all measurement points that gives the graph more peaks. The graph also plots the average speed of the track as a grey dashed line. The option to have both pace and speed is to gve user the option to choose. The more determined users or more professional may prefer using pace. The ordinary user maybe prefer speed. We give them both options. The option to choose for a smooth and rough graph is to have a less cluttered and less spikey graph.

By using the ”Show Graph” button we can switch between graph view and table view. The table shows statistics such as distance, duration and average and maximum speed and pace. This is to give the users the option to view statistics in a table instead of a graph.

5.4 Discussion

This part of the project took a few itterations. With each iteration the design got optimized and new elements were added. It was sometimes hard to fit all content in. Our goal was to make the design clean and clear and still show a lot of information. At some moments the design got to cluttered and we had to redesign the screen. These were the moments where buttons to switch between graphs and tables were implemented.

Also in the progress of development the elevation data appeared to be unusable. The data contained too much unrealistic data. Elevation differences of hunderds of metres is not realistic. It is well known that the accuracy of the GPS data varies with the phone model. So we decided not to use this data although it is stored in the database.

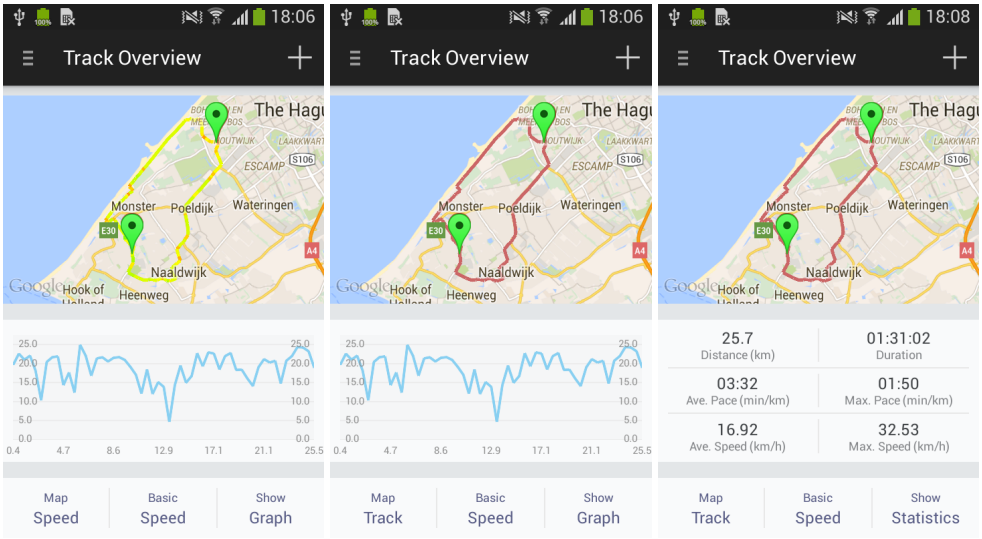

Figure 5.5: Three screenshots of the Track Overview screen. First one shows the track colored based on the speed at the part, second one shows the same track colored in one color and the last one shows statistics instead of the graph below the map.

Chapter 6

Conclusion

In this project we use the data recorded by the Miles Calls application and we wanted to get usefull information from it. First we started by designing a database to store the imported data. Then we developed a kriging model to interpolate. At the end we built a mobile application to visualise the data and results from the kriging. These parts of the project are related to the following research questions:

- How can we organize the data in an SQLite database, taking into account restrictions of mobile applications?
- What meaningful information can we compute from location and time data from a walked, ran or bicycled tracks?
- How can we present this in an Android application to the user?

We started by designing the database to store the data that is imported. The database is needed to analyse it and visualise it in a mobile application. Before designing the database we composed a list of requirements what the database must comply to. We designed the database shown in figure 3.3. This database design meets the set requirements. The design is compact and also future proof for the application. It can be used to store the tracks, organize them and store the corresponding statistics. So we can store and organize the imported data in a database designed and built in SQLite running on a mobile application.

With the database we can get data from all stored tracks and analyse it. We first used this data in a linear regression model. The linear regression was fast and easy to implement. However, linear regression on its own is not suitable for this kind of data. It gives a not precise enough trend. Next we used ordinary kriging. It is a method that uses interpolation to estimate values based on known data points. Ordinary kriging was mostly easy to implement. However, estimating the parameter θ was the hardest and most time consuming step. We compared three approaches to do this. First we set θ by hand. This worked somewhat, but it is a not usable approach. The value θ differs for with each data set. With the second and third approach we tried to automate estimating the value. The second approach was setting the θ by using maximum likelihood estimation. However, this worked for only half of the data sets. So we did a third approach. We estimated θ by using cross-validation. Cross-validation worked a lot better, but in some cases it still could not estimate a optimal value. We designed a small heuristic algorithm to make sure that there is always a best possible estimated value for θ. To improve the model we created our own regression model by combining ordinary kriging with linear regression. So we can generate a trend model based on past performance with this regression model.

In the mobile application developed in this project we visualise the generated model and some futher statistics. The application has a clean and clear interface which is easy to use. It uses maps, tables and graphs to visualise all information. We give the user the option to watch a track coloured based on speed and just in one colour. Graphs can switch between showing speed and pace over the track. So the application visualise all the information and data clean and clear.

Chapter 7

Outlook

The application and the kriging model are now usable, but much can be improved. The regression model is currently too slow when the dataset get bigger. In the future the cross validation could be improved. Doing so the cross validation only with a subset of the dataset. Estimating the optimal θ should be quicker. Also the method used to find the estimated θ on a given domain could be improved. Currently exhaustive search is used. Possibly a variant of the modified binary search that was used in maximum likelihood estimation could be used.

There are also many opportunities for the android mobile application. Currently a lot of basic management options are missing. We can add the options to turn off and delete locations and delete tracks. It would also be nice to name a track and to have the option to rename tracks and trips. Also tracks should be able to be moved to another trip.

The data that is recorded is as good as possible cleaned in Miles Calls. However, occasionally the data has incorrect measurements. They cause speeds to go to impossible values. Some kind of outlier detection of the imported data would be a nice feature to have. This feature would require some additionally research. A (very) fast part of the track could be a measurement error, but also just a fast runner or bicyclist.

Chapter A

Datasets Used in Experiment

These are the datasets used in the experiment described in chapter 4.3.3.

Time (days)	Speed (km/h)
0	3.4819
14	5.3617
28	4.3504
35	5.4348

Table A.1: Table of dataset 1. Activity type is hiking

Time (days)	Speed (km/h)
0	17.8892
9	17.8341
11	16.6291
32	17.3176
46	16.9222
50	20.1004
53	17.6403

Table A.2: Table of dataset 2. Activity type is bicycling

Bibliography

[1] Arnold Baca. *Computer Science in Sport: Research and Practice*. Routledge, 2014.

[2] Geoff Bohling. Kriging. Lecture Notes page 2. Kansas Geological Survey, University of Kansas, October 2005.

[3] Dostware. Miles calls by dostware. `https://play.google.com/store/apps/details?id=com.dostware.milescalls`, February 2015.

[4] Michael T. M. Emmerich. *Single- and Multi-objective Evolutionary Design Optimization Assisted by Gaussian Random Field Metamodels*. PhD thesis, Universität Dortmund, Germany, 2005.

[5] Emily Faherty. The ultimate guide to running lingo. `http://greatist.com/fitness/ultimate-guide-running-lingo`, April 2014.

[6] Runtastic GmbH. Runtastic. `https://www.runtastic.com`, February 2015.

[7] FitnessKeeper Inc. Runkeeper. `http://runkeeper.com`, February 2015.

[8] MapMyFitness Inc. Map my run. `http://www.mapmyrun.com`, February 2015.

[9] Nike Inc. Nike+. `https://secure-nikeplus.nike.com/plus/`, February 2015.

[10] Daniel Link and Martin Lames. Sport informatics-historical roots, interdisciplinarity and future developments. *International Journal of Computer Science in Sport*, 8(2):68–87, 2009.

[11] Endomondo LLC. Endomondo. `https://www.endomondo.com`, February 2015.

[12] Andrey Pepelyshev. The role of the nugget term in the gaussian process method. In *mODa 9–Advances in Model-Oriented Design and Analysis*, pages 149–156. Springer, 2010.

[13] Raghu Ramakrishnan and Johannes Gehrke. *Database management systems*. Osborne/McGraw-Hill, 2000.

[14] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[15] Eric W. Weisstein. "bar." from mathworld–a wolfram web resource. `http://mathworld.wolfram.com/Bar.html`, May 2015.

[16] Eric W. Weisstein. "interpolation." from mathworld–a wolfram web resource. `http://mathworld.wolfram.com/Interpolation.html`, May 2015.

[17] Eric W. Weisstein. "least squares fitting." from mathworld–a wolfram web resource. `http://mathworld.wolfram.com/LeastSquaresFitting.html`, May 2015.

[18] Eric W. Weisstein. "maximum likelihood." from mathworld–a wolfram web resource. `http://mathworld.wolfram.com/MaximumLikelihood.html`, May 2015.

[19] Eric W. Weisstein. "regression coefficient." from mathworld–a wolfram web resource. `http://mathworld.wolfram.com/RegressionCoefficient.html`, May 2015.

[20] Eric W. Weisstein. "time series analysis." from mathworld–a wolfram web resource. http://mathworld.wolfram.com/TimeSeriesAnalysis.html, May 2015.