TrackIt (Better!)

Miguel Pernas
IST- Instituto Superior Técnico
miguel.pernas@ist.utl.pt

Abstract. During the last years, the popularity of GPS tracking devices in sports and outdoor activities has been on the rise. This has caused an increase in the number of applications that are able to handle the data gathered by those devices.

In this document we introduce the TrackIt prototype, an application capable of analyzing and planning activities. The work described in this document was targeted at improving this prototype with new features like undo and redo support. We also added support for multiple sports, splitting of a GPS track into several segments for comparison purposes and the ability to identify climbs and descents by color code.

1. Introduction

In recent years, the popularity of GPS tracking devices for sports and outdoor activities has been rising among professional and amateur athletes alike [1] [2]. This rise in popularity is due not only to the improvements of these devices and supporting applications but also to their affordability. The rise of smartphones which include GPS capabilities was also a big factor in the increased popularity of GPS tracking applications since they’re now available to almost anyone. The amount of GPS tracking applications is vast and occupies a large portion of app stores in health and fitness related sections.

GPS tracking devices are capable of recording vast amounts of raw data. This requires software to extract meaning from that data and analyze it. The type of data gathered by these devices covers not only GPS information, like location, time and elevation, but also computed data, such as speed, distance and elevation gain. Some specialized devices can also record data such as heart rate and body temperature. Users can gather even more data, allowing improved analysis of sports and competitions [3].

There have also been improvements to digital maps, which are now widely available to everyone, from a variety of map providers such as Google Maps1 or OpenStreetMaps2. These maps are used to integrate user data and have become essential in GPS tracking devices and applications.

Due to the popularity increase of GPS tracking smartphone applications coupled with the improvements of digital maps GPS applications have evolved from platform specific to web based applications, usually with an accompanying mobile application as well. These applications have many features, which include, but are not limited to:

- Activity display, search and planning
- Performance monitoring and analysis
- Activity sharing among users
- Social networking
- Virtual community competitions

In an effort to incorporate the most useful features in a single application the TrackIt prototype is an application capable of analyzing and planning activities.

1 Google Maps (https://www.google.pt/maps/, last seen on 04/05/2016)
2 OpenStreetMap (https://www.openstreetmap.org/, last seen on 04/05/2016)
1.1 The TrackIt Prototype

TrackIt is a prototype application, developed in two previous iterations [4] [5]. It can read several GPS file formats, e.g. GPX3, KML4 and TCX5, and use that data to display GPS tracks based on several digital map providers (Figure 1). It also includes support for the Portuguese Military Maps.

![Figure 1 – GPS track displayed on TrackIt](image)

With TrackIt, users can analyze data from GPS tracks with charts for a general overview, and can also inspect each track element individually, from the whole tracks to a single track points. There are also several tools available that the user can utilize to create and edit GPS tracks and some other analysis tools. TrackIt also supports multimedia items which can be attached to a track.

The application utilizes a database which makes track metadata persistent through different sessions.

1.2 Objectives

Besides some improvements to the existing prototype, this work added the following features:

- Undo and redo operations
- Support for multiple sports
- Segment creation and comparison
- Color coded climbs and descents identification

Why have we chosen these features? TrackIt had plenty of operations to edit GPS tracks. These will ultimately change the original tracks. Unfortunately, there was no means to undo these operations and restoring the original tracks other than reopening them, creating the need for undo and redo operations. TrackIt only supported one sport type, road cycling. Some operations would use this sport's default values, like the default speed of 10 km/h when there was speed information missing in the tracks. We decided to improve upon this, adding support for more sport types, each with their own set of default values. The sports we decided to provide support for were:

- Road cycling
- Mountain biking
- Skiing
- Snowboarding
- Hiking
- Running
- Sailing

One issue was that it was difficult to analyze overlapping track sections. This made the creation of an operation that could divide a track into smaller segments a necessity [6]. This way, we could divide

---

3 GPX (www.topografix.com/gpx.asp/, last seen on 05/05/2016)
4 KML (www.opengeospatial.org/standards/kml/, last seen on 05/05/2016)
5 TCX (www8.garmin.com/xmlschemas/TrainingCenterDatabasev2.xsd/, last seen on 05/05/2016)
overlapping sections into their own segments. A better way to compare these segments, both visually and by analyzing their properties, was also created.

Finally, there was no way to visually identify climbs and descents on the map view. Inspired by elevation and relief maps with color coded elevation, we decided to create an operation that would do something similar to a GPS track, by coloring climbs and descents, according to their grade, with a color code on map view.

2. State of the Art

In order to better understand how to add the new proposed features, we analyzed some popular GPS tracking applications to assess if they shared the same features and how. The applications were:

- Google Earth
- Sports Tracker
- Endomondo
- Strava
- MyTourBook
- Wikiloc

After careful analysis we concluded that most applications did not offer undo/redo support and when they did – Strava and Wikiloc – it was merely for adding and removing track points. As for the support for multiple sports, the ones that do support them do so at a very basic level, as a form of track categorization, just to tell the users what sport type is a track referring to. Google Earth does not have any sport support while MyTourBook only supports cycling.

Segmentation is usually performed automatically, using the application’s criteria – only different distance lengths – which can sometimes be changed by the user. MyTourBook has the most segmentation criteria including, but not limited to, distance length, altitude and duration. In terms of segment comparison, MyTourBook is also the better one since it allows comparison through a table, with many different parameters, like duration, speed, grade, altitude gain, among others.

Finally, only Google Earth and MyTourBook have any support for the identification of different parameters by color coding, and Google Earth’s method is manual, requiring user input to decide the colors and to create the segments on KML files. MyTourBook, on the other hand, allows color coding of different parameters, with a different code for (Figure 2) altitude, grade and speed.

![Figure 2 - MyTourBook’s track color coding in altitude mode.](image)

3. New features

In the current work, we added the features discussed previously, which we’ll present in more detail, and we also corrected some of the prototype’s operations and added new ones. In particular, we’ve added several new criteria for the join operation. The join operation joins two or more GPS tracks into a

---

6 Google Earth (http://www.google.com/earth/, last seen on 05/05/2016)
7 Sports Tracker (http://www.sports-tracker.com/, last seen on 05/05/2016)
8 Endomondo (http://www.endomondo.com/, last seen on 05/05/2016)
9 Strava (http://www.strava.com/, last seen on 05/05/2016)
10 MyTourBook (http://mytourbook.sourceforge.net/mytourbook/, last seen on 05/05/2016)
11 Wikiloc (http://www.wikiloc.com/, last seen on 05/05/2016)
single track. The new criteria are used when the GPS tracks to join exceed a predefined distance between them, so we can calculate how to travel the distance between tracks. These criteria include different speed values to use and also the total duration to travel that distance.

We’ve also added the ability to change a track’s start and finish times individually and a new addition to the set pace operation. A new set pace method allows the user to select new start and finish times and track’s values are recalculated to reflect those changes, altering speed values and duration at each track point.

For the set pace operation we also added a new way to perform the smart pace method for hiking tracks, using Waldo Tobler’s formula [7], using a constant of 5 km/h:

\[ v = v_0 6 e^{-3.5 \frac{\text{dh}}{\text{dx}} + 0.05} + 0.175 \]

The operations to add/change and remove a pause inside a track were also added.

3.1. Undo and redo

One of the features that was missing in TrackIt was the ability to restore the application to a previous state. There are many editing operations in TrackIt that change the original track and it would be useful to return to previous state without having to reload the whole track. Due to type of data TrackIt handles, which is very complex, it would be wasteful to save the entire track’s state so it could be restored. Therefore, we decided to utilize TrackIt’s own operations to perform undo and redo since most are symmetrical. This way we only need to save the minimum amount of data to be able to restore the desired application state. But not all operations have undo/redo support. We decided to focus on the editing operations that include:

- Add/remove track points
- Change the start/finish time of a track
- Reverse a track
- Join/split tracks
- Add/remove pauses
- Remove all pauses
- Set pace
- Create a track segment
- Divide a track into segments

Some operations in the list above are symmetrical, like add/remove track points, join/split tracks and add/remove a pause. These were used to perform the undo/redo operations in those cases. But there are more symmetrical operations. To restore the start time of a track we can just change it back. The same for the finish time. Restoring a reversed course (i.e., a return course) is as simple as reversing it back. These operations are their own symmetrical. The set pace operation has several different methods to set the pace of a track but generally they all function in the same manner. A weight value is calculated which then multiplies values at each track point, changing speed and time values along with the track. This can be undone by simply multiplying each track point again, this time by the inverse of the weight value. So in a way, the set pace operation is also its own symmetrical. However, the smart pace method calculates each point’s values according to a formula or a table of values making it impossible to restore the original track with the same method. For this method alone, we store the entire track values that need be restored.

The actual undo/redo implementation was done based on a stack. We have two stacks, one for undo and another for redo. When a supported operation is executed it is added to the undo stack. Whenever an undo operation is executed, the last member of the stack is moved to the redo stack and vice versa. The redo stack is cleared when the redo stack is not empty and another supported operation is executed.

When a new operation is added to the undo stack, we store the values needed for the undo and redo operations to restore the desired state. After undo/redo is executed, we utilize a setup method to extract the values required and perform the operation needed to restore the application’s state.
3.2. New sports

Prior to this work, TrackIt had only support for a single sport, road cycling. Some values used by TrackIt, like the default speed of 10 km/h to be used in points with no speed information, made it so that tracks that resulted from a different sport used these values which were not tailored to them, producing wrong results. It was necessary to add support for different sports, each with their own default values, so we could use different values according to the track's sport type.

TrackIt already had a basic framework that allowed multiple sport types but it was incomplete and unused. This framework used the FIT\textsuperscript{12} protocol definition of sport and sub-sport, with the sport being the main sport type and the sub-sport a subset of that sport, like cycling/road for example. In 1.2 we presented some sport types we decided to provide sport for.

Table 1 presents these sport types again, with the definition of sport and sub-sport.

<table>
<thead>
<tr>
<th>Sport</th>
<th>Sub-sports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic</td>
<td>Generic</td>
</tr>
<tr>
<td>Running</td>
<td>Road, Trail, Track</td>
</tr>
<tr>
<td>Cycling</td>
<td>Road, Mountain</td>
</tr>
<tr>
<td>Walking</td>
<td>Road, Trail</td>
</tr>
<tr>
<td>Cross-country skiing</td>
<td>Track, Mountain</td>
</tr>
<tr>
<td>Alpine Skiing</td>
<td>Track, Slalom, Giant Slalom, Super-G, Downhill, Leisure</td>
</tr>
<tr>
<td>Snowboarding</td>
<td>Downhill, Mountain</td>
</tr>
<tr>
<td>Hiking</td>
<td>Trail</td>
</tr>
<tr>
<td>Sailing</td>
<td>Open Water</td>
</tr>
</tbody>
</table>

Table 1 – Combination of sports and sub-sports used by TrackIt.

Each sport is unique and can have multiple sub-sports with each combination having its own default values. We also added a generic (default) sport and sub-sport type to be used when the track's sport type is not supported by TrackIt. This work also allows future extension of the sports and sub-sports types.

Since these new sports and sub-sports were added so we could utilize different default values for each one, we needed to identify what type of default values would be useful to have. Based on the different parameters TrackIt used we came up with the following list:

- Default average speed: speed value to use when no speed information is available.
- Maximum allowed speed: the maximum speed allowed for a sport type, so we can't have cycling tracks with speeds over 100 km/h, for example (for future work, not currently in use).
- Pause threshold speed: track points below this speed will be considered a pause. Each sport needs an adequate threshold due to their different average speeds.
- Default pause duration: the default duration suggested to the user while inserting a pause.
- Follow roads: if the sport should follow the road. When joining two tracks, for example, the user can decide if they should be joined by a straight line or by creating an intermediate track that follows roads.
- Join maximum warning distance: maximum distance between two tracks during a join operation. The user is then warned that tracks are too far apart when this distance is exceeded.
- Join merge distance tolerance: the minimum distance between two tracks, during a join operation, where the connecting points are considered to be the same point.
- Join merge time tolerance: the minimum time difference between two tracks, during a join operation, where the connecting points are considered to be the same point (for future work, not currently in use).

\textsuperscript{12} FIT SDK (https://www.thisisant.com/resources/fit/, visto em 09/05/2016)
• Grade limit: maximum grade expected in a sport, used by the identification of climbs and descents by color code.

After identifying these parameters it was necessary to decide the values for each one and for each sport and sub-sport combination. This is difficult due to various different factors that can affect some of these parameters, like age, sex and weight. Therefore, we decided to analyze some tracks for each sport and sub-sport combination and identify adequate values for each parameter. In future work these values can be refined further by performing a more comprehensive study but, for now, they're useful to test the application.

We've also added a way for the user to change these values, in the application's preferences menu, so that users may customize them. There's also the possibility to reset them to their original values because these new parameters were added, alongside the sports and sub-sports, to TrackIt's database. This enables the persistence of each parameter's original value.

Another thing to note is that the user is the one that must decide each track’s sport and sub-sport since GPS track files usually do not carry this information.

3.3. Segment creation and comparison

With this new feature, our goals were to allow the creation of track segments and visually compare them and analyzing their data. Segment creation was obtained by defining a segment as a smaller track. By doing this we could still use TrackIt's track editing operations for these segments, particularly the ability to add and remove points if the segments required some adjustments.

We introduced 3 different methods to split a track into segments:

• Number of segments: split a track into \( n \) “equal” segments. The segments created have their lengths as close as possible so they may be perceived visually as of equal length (Figure 3).

• Duration of segments: split a track into \( n \) segments with a given duration, \( t \). The last segment will always have its duration equal to or less than \( t \).

• Length of segments: split a track into \( n \) segments of length \( d \). The last segment will always have its length equal to or less than \( d \).

![Figure 3 – A track split into 10 segments.](image)

These methods do not produce segments with perfectly equal lengths and durations because we split them at track point level. Even though we could define each segment’s length as 1000 meters, for example, there may not be a track point at that exact distance so we chose to create each segment as close as possible to its expected result.

We can also create individual segments by selecting the section of the track we want in TrackIt's chart view and converting it into a segment.

The segments created can then be compared visually and their data can be analyzed from a table. The data we chose to compare was each segment’s length, total length from the start of track, moving time and moving time from the start of the original track.
Figure 3 shows an example that allows visual segment comparison. But what if the segments do overlap? In this case, all overlapping segments need to be shifted on the map view to allow easier visual analysis.

Figure 4 do not completely remove overlapping. This can be improved by detecting the shifting location that produces minimal overlapping.

![Figure 3](image1.png)  ![Figure 4](image2.png)

**Figure 4 – 3 segments before (left) and after (right) shifting. After shifting we can see they’re easier to distinguish.**

### 3.4. Identifying climbs and descents by color code

Our original plan to identify climbs and descents by color coding was to use each sports own color code. However this is not that simple. Let’s take skiing’s color code for example. This code is different depending on the region where the activity is performed\(^\text{13}\). Data such as track width is also used to decide the color of each track, something TrackIt cannot measure. Due to these problems we decided to create a generic color code to be used by all sport types.

Our color code (Figure 5) is based on track grade. We create several grade intervals, with the middle one representing flat terrain (yellow). Climbs will be represented by the intervals higher (red) than the middle interval and descents by the lower (green) intervals.

![Figure 5](image3.png)

**Figure 5 – Color code used in TrackIt.**

We then split the track into sections, with each section belonging to a grade interval, according the each track point’s grade. Each section’s color is determined by the grade interval it belongs to.

Sports that are only performed in descents, like skiing, use a different color code (Figure 6). We use the same colors but red is used for the steeper descents and green for almost level ones. Black is used for climbs.

---

\(^{13}\) Europe uses a code that classifies ski runs as green, blue, red and black while North America’s code splits black runs into Diamond and double Diamond runs.
We then color each section in TrackIt’s map view (Figure 7) and also in the chart view (Figure 8). To make it easier to understand we added a legend so users can see what grade intervals are used for each color.

![Track with colored climbs and descents in TrackIt's map view.](image)

**Figure 7** – Track with colored climbs and descents in TrackIt’s map view.

![Track with colored climbs and descents in TrackIt's chart view.](image)

**Figure 8** - Track with colored climbs and descents in TrackIt's chart view.

4. Testing

To test the proposed solution we performed algorithm tests and usability tests with four users with different knowledge of these types of applications.

4.1. Usability tests

In the usability tests users completed all tasks they were asked to perform. However, some users struggled to complete some of the tasks due to their inexperience with the application and some of the concepts. The users began to familiarize themselves with the application the more they used it, resulting in a much better performance.

The tasks they had more difficulty with were tasks that required the selection of a track point or a segment in chart view since some operations are only active when the respective element is selected. They also had trouble identifying climbs and descents by color coding, not because they didn’t understand what to do but because they had to manually select the segments so that afterwards they could be shown on map view. This lead to the automatic segment coloring on the chart view (Figure 8). Some users also thought TrackIt’s operations were too scattered, with operations dispersed among the edit,
tools and context menus. Future work should regroup some of these operations together for better ease of use.

It’s also useful to note that no user had any difficulty with the undo and redo operations. We’ve even added tasks for them so the users knew those operations were available. This way they would know how to recover if they made a mistake.

4.2. Algorithm tests

To test each algorithm’s results we performed several tests, especially with the segmenting algorithm and the algorithm to identify climbs and descents by color coding.

The segmenting algorithm showed that the results are not far from the expected ones. Even though each segment is not perfect, since we divide the segments by track points, the standard deviation and its respective coefficient of variation were quite small, indicating a low dispersion relative to the expected values. We also determined that the algorithm was producing correct results and performing as intended. It was also discovered that segmenting by duration has issues when pauses are considered, especially pauses larger than the segment duration. This affects the duration of not only the segment that contains the pause but also some of its neighboring segments. The result is one segment with a very large duration containing the pause that exceeds the expected duration, followed by a couple of segments with much shorter durations than expected. Some work is necessary to handle these cases and allow users to decide when to include pauses in this process.

The tests of the algorithm to identify climbs and descents by color coding concluded that it performs as expected, with all track points correctly assigned to their respective grade intervals, with only some minor issues. One issue is due to the way the algorithm handles segments with only two track points. This happens when a segment contains one track point and the next one belongs to a different grade interval. In this case we use the average grade to determine the grade interval that segment belongs to. However, that may cause a point to belong to the same interval as the previous or following segment, resulting in continuous segments that belong to the same interval. This can be fixed by merging these segments. There is also the need to verify the track for possible grade errors. There were some tracks tested where a few track points’ grades were over 100%, a clear impossibility.

For sports that are only performed in descents it was noted that there were some short descents in-between some climbs which could not be attributed to the sport. These short descents were colored as a descent but in reality they should be considered part of the climb. There can also be some short climbs in the middle of descents.

5. Conclusions

The new features added to TrackIt proved to be adequate, with the new algorithms producing the expected results but there is still some work to be done. The algorithm to identify climbs and descents by color code needs to be further refined as well as the segmenting algorithm, so they can handle some specific cases identified in our testing. The segmenting algorithm also needs to improve its visual shifting method so each track can be shifted to produce minimal visual overlapping.

There are also some useful features which could be added in the future, such as:

- Using point interpolation in the segmenting algorithm to produce segments of the exact length, leaving it to the user to decide to use interpolation or not.
- Improve the default values for each sport and sub-sport combination.
- Add user profiles to allow individual preferences and tracks.
- User interface and general QOL improvements.
6. References


