TrackIt - SpotOn

Joana Maria Salvado Vicente
Instituto Superior Técnico, Universidade de Lisboa

***Abstract*—The popularity of sport oriented applications born from the ubiquity of GPS recording devices and the increase of these devices accuracy and portability brought forth the creation of features aimed at assisting dedicated athletes and enthusiastic amateurs in providing the analysis of past activities and facilitate future planning of their preferred sports.**

While TrackIt offers multiple tools that provide ample editing and analysis features, it's not without limitations. This document details the improvements made aiming to provide solutions to them and focused on three major issues present in TrackIt. A three-dimensional version of simplification algorithms was implemented to deal with the lack of considering elevation as a parameter; the assisted media placement was redesigned to provide more realistic and unbiased results, and introduced new evaluation metrics. Finally, an autonomous algorithm for the identification and elimination of GPS errors based on the recognition of erroneous patterns was introduced.

I. Introduction

It is imperative to maintain track accuracy and overall location data as close to the real trajectory as possible for correct analysis. Unfortunately, GPS recordings create data redundancies that generate problems for research, storage and data manipulation. Compression algorithms are a useful tool when these problems become unmanageable.

GPS errors can create various issues ti processing and analysing recorded data. These errors often create localized error points, that dont accurately represent a track. Two common displays of inaccurate GPS are tangles and spikes.
Spikes are characterized by sudden location jumps, further than it would be to travel in the recorded time, with equally sudden return. These are often caused by GPS signal loss.
Tangles are a far greater problem than spikes, and more problematic to resolve. They often occur in urban areas due to multiple reflective surfaces that cause GPS signals to degrade and echo. Yet, due to the multiple reflected signal bouncing, weak or intermittently signals are still being received, resulting in jumpy track recordings.

II. Objectives

TrackIt's editing features are its strongest advantage over other similar applications. Yet, they still suffer from issues that put at stake a track accuracy when these features are applied. While TrackIt supports track simplification it still ignores a key metric: elevation. Ignoring elevation data when compressing entails discarding this third axis and the information inherent to it completely. A set of points with great altitude change that resembles a straight line on a two-dimensional representation will suffer exaggerated point removal and track data errors, resulting in loss of information and inaccurate elevation profiles.

Another issue to take into account is the existence of GPS errors when simplifying a track. Not only are these damaging to the global accuracy of the track by disfiguring track shape and corrupting all data calculations, but due to their unexpected variances, they are considered as points of relevance, which are likely to be maintained. Users highly need an automated alternative, at the very least to help identify error sections as opposed to forcing users to manually verify each track for errors.

Another popular feature of TrackIt is the possibility of adding media to tracks as it allows better visual representation and contextualization. But misplacements create unacceptable errors due to enforcing incorrect results on the all media.

Based on these problems, and after a survey on the other available alternatives ascertained that no solution was being provided, we aimed at solving the aforementioned problems through improvement of previously existing features or the newly implementation of others. The proposed objectives were:

- Extend the current simplification algorithm to handle three-dimensional data, making elevation a parameter;
- Develop an algorithm capable of autonomously identifying possible erroneous points and offering the user the possibility of elimination of such errors;
- Adjust the media placement evaluator to account for biased placements, lessening the effect of wrong placings on rating;
- Modify the media placement algorithm to deal with non-identified non-recorded pauses and strict media capture assumptions.

III. State of the Art

We divided our research into the state of the art in two subcategories, a survey of the existing applications and a exploration of simplification algorithms.

For algorithm analysis, our research was mostly focused on simplification algorithms, due to two reasons: our primary objective was to improve the current simplification algorithm as include parameters that account for track three-dimensionality and to study simplification algorithms that considered track trajectory data to find unique points and suggest hypotheses for track errors identification.

A popular algorithm is the Ramer-Douglas-Peucker algorithm [1] which uses the perpendicular distance to the end points of a track segment to find redundant point, RDP is the base algorithm for our simplification algorithm. Though there are several alternatives for polyline simplification, the

Douglas-Peucker algorithm remains the algorithm with minimum values for shape dissimilarity and distortion although not the best choice for large data sets [2]. Despite the various developed algorithms one issue appears not to have been handled, the simplification of three-dimensional tracks.

But the analysis of cases where points are qualified as important or redundant can also serve as a technique to differentiate between errors and correctly recorded data. The use of heading in combination with speed was used by [3] to find a way to predict where the next point of a track likely be, and the decision to keep or discard a point will depends on its location relative to that prediction. Even though removal of erroneous points cannot be solved by simplification algorithms, their analysis showed that they're capable assisting in recognizing points that deviate from predicted behaviour.

For our first section we chose five applications (excluding TrackIt) based on three parameters: popularity, availability and presence of editing tools.

We assessed if any data compression option was offered, what sort of user input was required and the impact on track shape. We checked for the existence of autonomous detection of erroneous points and their subsequent elimination, but also chose to reduce the problem to manual segmentation and individual point removal as this is what TrackIt currently provides. Features relating to media placement were checked for their existence, how media upload to the application is made and method of placement, manual or according to average correspondence between timestamps of photos and points (interpolation techniques). We focused on applications that might provide any of the following features:

- Track simplification.
- Track segmentation and segment elimination.
- Media insertion and geotagging.

The applications examined were: Strava[1], GPSies[2], SportsTracks[3], GarminBasecamp[4], MyTourBook[5].

Through examination of their features it became evident that there are only a few and basic tools that attempt to solve our proposed problems, and that most are unable to provide precise solutions. None of the researched applications presented tools for GPS corrections, though MyTourBook came close with its use of automatic segmentation. But in most cases, only simple editing tools for manual point selection and elimination are available. Few offered simplification and, if so, it was done automatically and seemingly discarded elevation profiles completely. Regarding media placement, support for this feature was more present, but still lacked methodologies to fix incorrect timestamps nor do they provide accurate interpolation techniques. The closest solution, once again offered by MyTourBook, used time shifts supplied by the user to accommodate time zone differences and incompatibilities between cameras and GPS devices.

[1]Strava, {strava.com}
[2]GPSies, {gpsies.com}
[3]SportTracks, {sporttracks.mobi}
[4]Garmin Basecamp, {garmin.com/en-US/shop/downloads/basecamp}
[5]MyTourBook, {mytourbook.sourceforge.net}

A. *TrackIt*

TrackIt is a multi-platform modular desktop application, developed in Java[6] with a Swing[7] interface, with a front-end, a back-end, an import/export file module and a database using the SQLite[8] relational database management system. It offers extensive planning, visualization, analysis and editing tools. A wide range of editing options was developed with the aim of providing a full and customizable planning experience. TrackIt has previously undergone four iterations to improve its features. Various views are also available to provide assistance

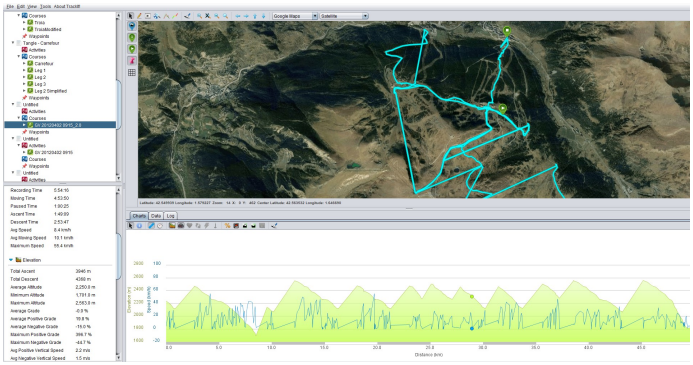


Fig. 1. TrackIt interface and its available views during track analysis.

to the users' viewing needs: Document View, Map View, Profile View, and Summary View (Figure 1). These allow users to make comparisons with information collected in different contexts and correlate them, allowing users to infer conclusions about past activities.

IV. Previous State of Features

A. *Simplification*

The originally implemented algorithm iterates through a track and creates a set of three contiguous points and draws a line between the outermost points, similarly to DP [1]. The trios middle point is used to calculate the perpendicular distance to the line connecting the end points. This distance is associated with the corresponding trio, stored in a list and sorted in ascending order. Until the desired limit of points (defined by the user) is reached the first element (smallest distance) is marked for removal. To help attain some of the goals a weight parameter is also considered to make sure that:

- the start and end points have the least priority;
- points from climb segments have lower priority than those of flat or descending ones.

It's worth mentioning that these weights should assume different priority values depending on activity type. This weighted priority configuration was implemented with cycling activities in mind, as for these climb sections are more important than descents or flat terrain. While the current implementation took great care to maintain track shape, disregarding elevation data

[6]Oracle, {java.com}
[7]Oracle, {oracle.com/technetwork/java/architecture-142923.html}
[8]SQLite, {sqlite.org}

is a common issue with simplification algorithms, including the one of TrackIt. Fully ignoring elevation means ignoring height discrepancies which ultimately lead to biased grade calculations and misrepresented elevation profile features.
Tracks that possess distinguishable latitude and longitude differences among points may not suffer this effect as these differences might also serve to preserve elevation information. But when elevation is the only discernible track shape variation, relatively straight line segments may be reduced to only start and end points.

B. Error Elimination

To deal with GPS errors TrackIt only provides the possibility of manual correction to remove incorrectly recorded points. Users needed to divide course/activities tracks into separate tracks, delete the unwanted track portions, and join the remaining segments. Thought the result might be acceptable, it still meant that the user would have to expend time and effort manually editing each track, a problem that grows exponentially for tracks with a large number of points. Another problem derived from this solution is the possible loss of consistency by segment removal and the difficulty of identifying the start and end of erroneous sections.

C. Media Placement

One of the previous TrackIt iterations focused on adding media to tracks. In addition to providing the option of importing media, these also are the problems that derive from attempting to correctly place it. When dealing with media placements we must pay special attention to the correctness of media geotagging information, as an improper placing leads to incorrect analysis and an unpleasant user experience. For geotagged photos we must consider issues with unsynchronized clocks between recording and media capture devices.

To aid in situations with time zone shifts and clock desynchronizations an assisted geotagging algorithm was implemented. The core idea is that to capture media the user will have to stop moving. The algorithm places the first picture in the middle of the first detected pause and proceeds to place all other pictures away from that first pause, using the time difference between photos and the first pauses's timestamp. It then places the first photo in the middle of the second pause and repeats this process, all the while assessing accuracy. The placement algorithm uses as a accuracy parameter the number of photos it can place within pauses. The algorithm finds in which pause the first photo must be placed to achieve the highest accuracy value, i.e. the highest number of photos fitting in pauses.

The problems for photo placement is obvious for activities with long pauses. Using only a single parameter to measure the correctness of a placement often creates biased results. When long pauses are present, the probability that a photo will be placed within a long pause is quite high, swaying the algorithm into believing that that placement is optimal. Another problem was the assumption that the first recorded media was captured in the middle of a pause, but it could have been recorded at any time of the pause. The problems with media placement can then be simplified to two issues: strict assumptions that photos are captured mid-pause and lack of parameters to deal with specific situations that create biased results.

V. Implementation

A. Simplification

The original implementation of the simplification algorithm was well developed and produced good results. To prioritize certain points for removal it adds weight parameters to give priority to sections with steep climbs or descents. Segment priority increased also if its associated triad of points has standard speed deviations smaller than a threshold (to remove more points in slow moving sections). Our work was to add elevation to the previously implemented calculations. To introduce tridimensionality to the operation we applied a simple logic between the Haversine formula and the Pythagorean theorem. The Haversine formula calculates, given two point's coordinates, its the great-circle distance. (Which is the shortest distance over a sphere's surface.) The Haversine formula essentially provides as-the-crow-flies distance between points, but ignores altitude variation.

A simple Pythagorean formula uses the perpendicular distance calculated by the Haversine formula and the altitude difference between the start and middle point of a triad of points and calculates its hypotenuse using it as a 3D perpendicular distance. It proceeds to associate this 3D distance (3DcrossTrackDistance) with its corresponding point and store it in a list of all distances (3DcrossTrackList). The simplification algorithm receives a list composed of all the points of a track (trackpoints) and the numerical final number of points desired by the user (finalNumberOfPoints) and it can be summarized as follows:

function SIMPLIFICATION($trackpoints$, $finalNumberOfPoints$)
  $3DcrossTrackList$ ▷ initialise 3DcrossTrackList
  for all $trackpoints$ **do**
    $compute\ 3DcrossTrackDistance$ ▷ calculate 3D cross tracks distances
    $compute\ crossTrackWeights$ ▷ calculates cross track weights
  end for
  $sort\ 3DcrossTrackList$ ▷ sort cross track list in descending priority order
  while $trackpoints > finalNumberOfPoints$ **do**
    remove first($3DcrossTrackList$) from $trackpoints$
    consolidate $3DcrossTrackList$ ▷ consolidate information for trackpoints in list after point removal
    $sort\ 3DcrossTrackList$
  end while
end function

B. Error Removal

The idea behind the solution for track recording errors is that if a sequence of contiguous points has characteristics that are outside expected parameters when we consider the previous points, we could infer it to be a sequence of points with suspicious behaviour and examine it further. We can divide this process into four phases. *An individual evaluation*, that assesses each point considering the average moving speed and the bearing of the previous point and flagging as needed. *A flagged sequence detection*, if consecutive points have individual flags then a new list is created that contains this sequence. *Sequence evaluation*, in which assembled sequence are evaluated for size, and false positive cases. *Elimination*, discarding the points that were clearly classified as erroneous, and whose sequence creates a tangle.

Two parameters were considered to find and classify these sequences: speed and trajectory. A new class, TangledPoint, keeps boolean flags pertaining to each evaluated parameter and is associated with each point of a previously created linked list with all points of a track. The individual evaluation is made for all points according to two parameters, bearing and speed changes.

Angular evaluation assesses the bearing variation between the previous and the current point. Points with bearing changes from the previous point higher than 55 degrees (towards each 'side' of the current bearing value) are flagged as suspicious, and successive points with this characteristic are highly indicative of errors. To deal with real sharp corner turning, another flag was created which keeps track of bearing differences between 75 to 105 degrees from previous points. This case was designed mainly for tracks in urban areas when turning around city blocks, or architecturally imposed obstacles that leave sharp angles which may lead to false positives.

The next individual point evaluation assesses speed and its change relative to the track's average moving speed. The flag associated with speed is enabled when for two consecutive points, if one of them is outside a previously defined variation for the average moving speed. when both are outside this may indicate an acceleration or deceleration.

After individual evaluation we find consecutive points that are flagged with any of the previous parameters flags, and if so add them to a list for the next step. Lists that with a single point, if the point possesses a flag identifying a sharp corner turn and no speed flags, are considered false positive cases and not flagged for deletion. For all other cases the entire sequence (except the final point) is marked for elimination. The last point of the sequence evaluation list is excluded from elimination as we cannot safely infer that is also an error as its evaluation is dependent on the previously erroneous points. The remaining number of identified tangles is used as a stopping point, if the total number of tangle points is less than 10% of the tracks total point size. This was implemented to ensure that the solution would not be infinitely looking to erase all tangles since some are correctly recorded (e.g. acceleration/deceleration sections, sharp corners). The algorithm and its auxiliary functions for flag identification is presented below. For a list containing all points from a track (trackpoints):

function ERRORREMOVAL($trackpoints$)
  $tangleCount \leftarrow 0$
  create $TangledPoints$ ▷ creates linked list for all trackpoints
  create $SuspiciousPoints$ ▷ stores points to be removed
  while $tangleCount >$ threshold **do**
    for $TangledPoints[N]$ **do** ▷ for each trackpoint in a track comprised of N points
      calculate bearing($TangledPoints[N]$) ▷ associate bearing with respective point
      INDIVIDUALEVALUATION($TangledPoints[N-1]$, $TangledPoints[N]$)
    end for
    for $i \leftarrow 3; i < size(TangledPoints[N]); i++$ **do**
      if $TangledPoints[N]$ is flagged **then**
        $tangleCount \leftarrow +1$
        $SuspiciousPoints \leftarrow TangledPoints[N]$ ▷ add point to list
      else
        SEQUENCEEVALUATION()$SuspiciousPoints$ ▷ proceed to sequence evaluation
        clear($SuspiciousPoints$) ▷ clear the list
      end if
    end for
    remove flagged error points from $trackpoints$
    consolidate remaining points in $trackpoints$
  end while
end function

---

function INDIVIDUALEVALUATION($TangledPoints[N-1]$, $TangledPoints[N]$)
  if speed variation between$TangledPoints[N-1]$, $TangledPoints[N]$ exceeds the expected **then**
    flag ($TangledPoints[N]$) for speed
  end if
  if bearing variation between$TangledPoints[N-1]$, $TangledPoints[N]$ exceeds the expected **then**
    flag ($TangledPoints[N]$) for bearing
  end if
  if $TangledPoints[N]$ is flagged by speed or bearing **then**
    flag($TangledPoints[N]$) ▷ flags TangledPoints[N] as a erroneous point
  end if
end function

---

function SEQUENCEEVALUATION($SuspiciousPoints$)
  if size($SuspiciousPoints$) == 1 and $SuspiciousPoints[T]$ identified as corner **then**

```
        tangleCount ← −1
        un-flag (SuspiciousPoints[T]) as possible tangle
    else
        for SuspiciousPoints[T] do
            flag SuspiciousPoints[T] as error
        end for
    end if
end function
```

C. Photo Placing

Though the initial idea behind the assisted geotagging algorithm holds true for some cases, evaluating a placement by the number of photos within pauses creates a misleading impression when we take into consideration the existence of pauses with extended duration. This issue is largely due to using only a single parameter as an evaluator. Using other parameters to evaluate the correctness of each placement shift allows for better assessments and consequently better adjustments.

To solve these issues we introduced a rating for each computed shift when iterating through the pauses. Pauses with duration longer than 2.5% of a track's total duration are qualified as "big pauses" and if more than 5% of all imported photos are placed inside a big pause, we consider that configuration to be biased and a rating penalty for each photo above the 5%. This new rating formula now accounts for photos placed as follows: Photos placed in pauses (picInPauses), outside pauses (picOutPauses) photos fully outside the recorded time (picOutTrack) and photos inside lengthy pauses (picInBigPauses) with a set of P photos, the rating will be calculated as follows:

$$((picInPauses \times 1.0) + (picOutPauses \times 0.5) + (picOutTrack \times -2.0) + (picInBigPauses \times -1.0))/(P \times 1.0)$$

This implementation decision was made to remove the (wrong) idea that a high number of photos inside a pause are a good indicator of a successful and correct placement.

Through some initial testing it became apparent that the core idea behind placement was too strict to allow better placement configuration. By assuming that all photos were taken inside pauses, we would always be dependent on the first photos placement. Even when an infinitesimal small pause would fail to be detected, the first pause would then be pushed further away from its actual capture point and, subsequently, all others. To manage this issue, we choose to iterate all photos through all pauses to find the best configuration. Although this may seem to contradict the initial idea that all photos must necessarily be captured inside pauses, we are merely dealing with the possibilities that some pauses were not properly recorded or detected. We place the first photo inside the first pause and calculate its new rating. We repeat this process with the second photo in the first pause and so on, ultimately moving into the second pause and once again comparing the results.

When the best rating is found, the chosen photo is shifted within the chosen pause to handle the issue of only considering middle-pause capture. The photo for the best rating is alternated between three timestamps of the pause: 0.25%, 0.50% and 0.75% of that pause's duration. Rating calculations are once again calculated for these three possibilities and for the best one, the time shift between the photo's original timestamp and the its new location (one of three possibilities inside the pause) is applied to all photos. The placement algorithm receives a list of all points of a track (trackpoints), a list of all the imported photos for that track (pictureList) and a list of all detected track pauses (pauseList) which contain their duration, start and end timestamps. It can be summarized as follows:

```
function AssistedPlacement(trackpoints, pictureList, pauseList)
    tempRating ← null
    finalRating ← null
    bestPause ← null
    targetPause ← null
    bestPicture ← null
    targetPicture ← null
    Intervals ← CalculatePicturesIntervals ▷ calculates and stores intervals between all imported photos
    for pictureList[N] do ▷ for each photo in the pictureList comprised of N photos
        for pauseList[T] do ▷ for each pause in the pauseList comprised of T pauses
            M ← middle(pauseList[T]) ▷ finds pause middle time
            timestamp(pictureList[N]) ← M ▷ sets picture timestamp to pause middle
            ApplyPlacementToAll(trackpoints, M, pictureList[N])
            R ← calculatedRating
            if R > tempRating then
                targetPause ← T
                targetPicture ← N
            end if
        end for
        if tempRating > finalRating then
            bestPause ← targetPause
            bestPause ← targetPicture
        end if
    end for
    FindShiftWithinPause(trackpoints, bestPause, bestPause)
    ApplyPlacementToAll(trackpoints, bestPause, bestPause)
end function
```

FindShiftWithinPause tries to shift the picture found for the best pause through the three distinct points mentioned above and after finding the best shift will call the ApplyPlacement-

ToAll function. This function takes the best configuration and applies that time shift to all photos of the pictureList.

VI. Testing

We next present the results obtained with the implemented solutions and discuss them to clarify the reasons that ultimately lead to their implementation choices and improvement.

Although the results of the evaluation can be generalized and seen as a response to the global problems, the solutions were tested on specific test situations in which our problems were clearly present. This allowed us to obtain a notion of how the solution behaves in specific situations of the same general problem, which in turn enabled us to understand their limitations and possible refinement. Results were obtained with generalized parametrization values but that still take advantage of the categories created by TrackIt, whose sport/subsport characteristics may help improve effectiveness and accuracy.

A. Simplification

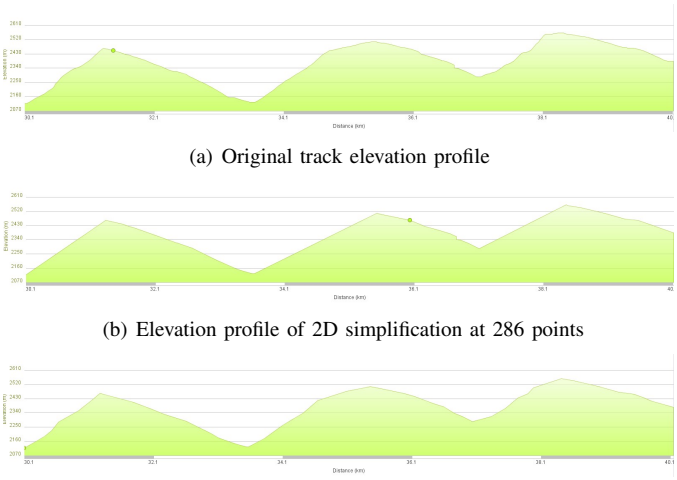
Evaluation of the three-dimensional simplification algorithm was done by comparing the same point reduction to the original 2D algorithm. The parameters chosen to evaluate solution success were the number of remaining points, overall visual shape and elevation profile. To evaluate our 3D simplification solution we chose to focus on the specific case of a ski track on mountainous terrain with high elevation changes of elevation. The presence of mechanical transport methods in the track (ski lifts) means that there are sections with drastic altitude variations in an almost linear path. This situation is precisely what we intended to evaluate as we believe that these situations are not being correctly preserved.

We tested a 1525 point track with three levels of simplification, aiming at three different number of remaining points: 750, 500 and 250. Overall, visual track shape has little variation from the original, even when the number of total remaining points is as low as 250. However, the most promising result to support our motivations for 3D simplification algorithm is the visual comparison between the elevation profiles. Reduction to 500 and 250 points create quite remarkable differences. When observing straight line track sections the difference between the 2D and 3D algorithmic version are clearly visible. Figure 2 displays comparison for the same section for three tracks: original, 2D simplification and 3D simplification.

The results for this test were highly satisfactory. Our solution was able to attain a higher level of shape accuracy, came closer to the desired remaining number of points and was still able to present good point density. The problem with ignoring elevation is most visible when the number of removed points is high and the most affected sections were found on straight line section which 2D algorithms eliminate more points in, as they appear to provide little variance and thus appear not worthy of keeping.

VII. Error Elimination

Error elimination was tested using tracks that possessed tangle problems. A manual corrected track was created for



(a) Original track elevation profile

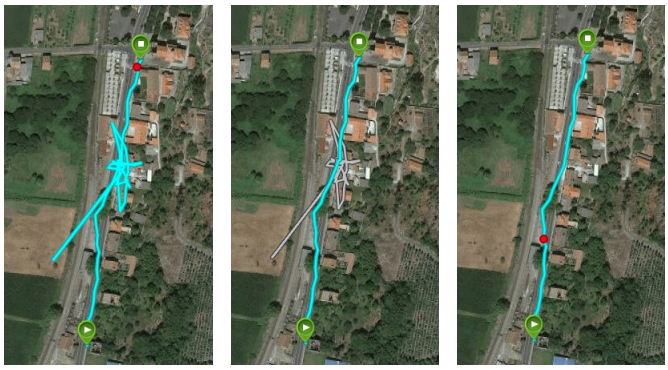
(b) Elevation profile of 2D simplification at 286 points

(c) Elevation profile of 3D simplification at 252 points

Fig. 2. Visual shape comparison between original, 2D and 3D solution for simplification at 250 points.

comparison purposes. We opted to evaluate the comparison with the manual corrected track with three metrics: 1) how many points remain on the resulting track; 2) visual track shape after elimination of erroneous points; 3) if any points were incorrectly detected (false positives).

The test track was recorded in an urban environment with flat terrain and has clear evidence of a high degree of successive drastic bearing variations combined with individual points with speed values over 37 times that of the track's average speed. Our solution required 6 iterations of the track and detected a total of 46 wrong points.



(a) Original (b) Manual Correction (c) Solution Result

Fig. 3. Visual shape comparison between original, manual correction and solution correction of GPS errors for test case "Escravitude".

The results obtained with our test cases were highly satisfactory as they succeeded to remove most manually identified GPS errors from the tracks and assist the user by providing a more understandable track for future planning or analysis.

A. Photo Placing

As the placement algorithm highly depends on pause identification, the algorithm had to carefully identify them

in accordance to the sport and subsport categories. TrackIt defines different speed thresholds for each category, and a misclassified track means that pauses are incorrectly detected, ultimately resulting in incorrect placements. Our main metric for evaluating our solution's success was the rating value developed to accompany this algorithm, and average distance shift applied to photos.

Our test case consists of a track recorded on a river bank environment in close proximity to a large body of water, and 27 geotagged photos were associated with it. The record time was of 1 hour, 32 minutes and 50 seconds and according to its category (Walking with Casual Walking as a subcategory) TrackIt establishes as a pause speed threshold of 1.5km/h, and 10 pauses were detected. The average pause length was 1 minute and 30 seconds, while the longest detected pause had a length of 3 minutes and 59 seconds. Three pauses were identified as "big pauses", meaning that their durations are at least 2.5% of the total duration of the track.

The optimal placement found by our algorithm was to place the 3rd photo of the set on the 2nd detected pause, which equaled to a time shift of -4 seconds relatively to that photo's timestamp. This time shift difference was applied to all photos, resulting in 20 photos placed inside pauses and 7 outside pauses. The previous algorithms calculated for the same track and speed threshold a time shift of +17 seconds, placing 14 photos in pauses and 13 outside. The rating comparison and distance shift between the original location and the location calculated by the our algorithm and the previous implementation are presented in Table I.

Distance Shift (m)	Our Algorithm	Original Algorithm
Average	4.9	12.3
Stan. Dev.	2.9	5.5
Maximum	12.5	19.8
Minimum	0.2	2.1
Metric	Our Algorithm	Original Algorithm
Rating	0.76	0.76
Time Shift (hh:mm:ss)	-00:00:04	+00:00:17

TABLE I
ASSISTED MEDIA PLACEMENT TEST CASE - COMPARISON BETWEEN OUR SOLUTION AND THE ORIGINAL ALGORITHM.

Although the overall rating remains the same, this is due to a trade off between more photos being placed within pauses and the penalty for placements in big pauses present in our rating algorithm.

There's a significant difference in both time shift and distance shift differences between the original and our solution. In the previous implementation several photos suffered deviations far larger than 10 meters, which warranted suspicion on the algorithm's accuracy.

The results allow us to conclude that both improved versions of the rating and the placement algorithm appear to provide a more realist approximation.

VIII. CONCLUSION

TrackIt has repetitively made great strides to provide features that efficiently manipulate recorded GPS data. However, these features have their own limitations or are unable to meet specific objectives.

During our survey of existing applications, we were able to conclude that neither TrackIt nor any of the currently available alternatives provided solutions for the key problems that motivated our work. We therefore developed and introduced into TrackIt three solutions:

- Three-dimensional track simplification
- Autonomous removal of erroneous points caused by GPS errors.
- Assisted media placement algorithm
- Improvement to the media placer evaluator

Track simplification, one of the most prevalent tools when dealing with GPS track focused applications, was upgraded to handle three-dimensional data and produced better simplification results due to the added information for point elimination decisions. Assisted media was also improved by discarding its original strict metrics that promoted biased results. Lastly, a tool aimed at correcting erroneous sections due to GPS errors was introduced, to provide an autonomous data correction solution based on distinct patterns.

When evaluating the results achieved by our solutions we were able to attain remarkably positive results. The 3D simplification algorithm produced the results we initially expected. When linear track segments were simplified the lack of information for elevation meant they were liable for elimination. Although start and end points for those sections are usually preserved the intermediate data would be lost, especially when dealing with lossy algorithms. Though we expected our solution to remove less points than its 2D version as more points appear to have relevant discrepancies, the reverse occurred. The added parameters lead to clearer distinctions about worthy points.

Regarding error removal results surpassed our expectations, GPS error recognition based on pattern identification lead to accurate results but still has room for refinement. One key element to explore in this solution is its parameterization. Small parameter changes for the identification phases lead to improved results in specific case tests. Our solution was able to eliminate an overwhelming majority of errors with little false positives cases.

The improvements made to the assisted media placement algorithm provided results superior to those of its predecessor. Both time and distance errors were remarkably reduced for the test cases, and when the correct time shift was known, results came extremely close. The new rating provided a more accurate and unbiased evaluation results. In the tests, more photos were placed within pauses, and by accounting for placements in "big pauses" the new rating was able to mitigate the "black hole" effect. One concern with this solution was the higher execution time when dealing with a high number of pauses and associated photos. Ultimately as this is not performed in real-time this is not a unmanageable problem, but is definitely worth mentioning.

Overall, our three solutions were more than able to meet our desired goals and thus improve the editing tools provided

by TrackIt.

IX. Future Work

Although the improvements implemented are functional and provide progress to TrackIt's tools with accurate results, there is still room for improvement especially regarding the role of user choices of some features and attributes for a more personalized experience, such:

- Apply different priority weights depending on the track type when simplifying. Different sports mean climb, flat and descend sections have different importance for accuracy;
- Explore "normal" angle variation for each sport through statistical evaluation, improving the accuracy of the recently added algorithm for GPS errors by providing a less generalized approach to pattern recognition;
- Add special cases to GPS error removal (identify special tangle cases and their characteristics) to reduce false negative cases;
- Apply assisted media placement to selected photos, as opposed to the full set;
- Divide photos captured with different devices and apply time shifts respectively, allowing for device based placement;

References

[1] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[2] W. Shi and C. Cheung, "Performance evaluation of line simplification algorithms for vector generalization," *The Cartographic Journal*, vol. 43, no. 1, pp. 27–44, 2006.

[3] M. Potamias, K. Patroumpas, and T. Sellis, "Sampling trajectory streams with spatiotemporal criteria," in *Scientific and Statistical Database Management, 2006. 18th International Conference on*. IEEE, 2006, pp. 275–284.

[4] K. Reumann, "Optimizing curve segmantation in computer graphics," *International Comp. Sympo. 1973. Amsterdam*, pp. 467–472, 1974.

[5] H. Opheim, "Smoothing a digitized curve by data reduction methods," in *Proceedings of Eurographics*, vol. 81, 1981, pp. 127–135.

[6] T. Lang, "Rules for robot draughtsmen," *Geographical Magazine, 42*, p. 5051, 1969.

[7] N. Meratnia and A. Rolf, "Spatiotemporal compression techniques for moving point objects," in *International Conference on Extending Database Technology*. Springer, 2004, pp. 765–782.

[8] J. Muckell, J.-H. Hwang, C. T. Lawson, and S. Ravi, "Algorithms for compressing gps trajectory data: an empirical evaluation," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2010, pp. 402–405.

[9] C. Long, R. C.-W. Wong, and H. Jagadish, "Direction-preserving trajectory simplification," *Proceedings of the VLDB Endowment*, vol. 6, no. 10, pp. 949–960, 2013.

[10] P. Gomes, "Track it (again)," Master's thesis, Instituto Superior Tcnico, Univ. of Lisbon, Lisbon, 2015.

[11] R. B. McMaster, "The integration of simplification and smoothing algorithms in line generalization," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 26, no. 1, pp. 101–121, 1989.

[12] Y. S. Saeedrashed, "An experimental comparison of line generalization algorithms in gis," *International Journal of Advanced Remote Sensing and GIS*, vol. 3, no. 1, pp. 446–466, 2014.

[13] H. W. Gellersen, A. Schmidt, and M. Beigl, "Multi-sensor context-awareness in mobile devices and smart artifacts," *Mobile Networks and Applications*, vol. 7, no. 5, pp. 341–351, 2002.

[14] I. Tjostheim and D. R. Fesenmaier, "Mobile devices as substitute or supplement to traditional information sources: city tourists, mobile guides and gps navigation," *Information and communication technologies in tourism 2008*, pp. 324–335, 2008.

[15] W. Viana, J. Bringel Filho, J. Gensel, M. V. Oliver, and H. Martin, "Photomap–automatic spatiotemporal annotation for mobile photos," in *International Symposium on Web and Wireless Geographical Information Systems*. Springer, 2007, pp. 187–201.

[16] J. Luo, D. Joshi, J. Yu, and A. Gallagher, "Geotagging in multimedia and computer visiona survey," *Multimedia Tools and Applications*, vol. 51, no. 1, pp. 187–211, 2011.

[17] R. Bajaj, S. L. Ranaweera, and D. P. Agrawal, "Gps: location-tracking technology," *Computer*, vol. 35, no. 4, pp. 92–94, 2002.

[18] H. Malheiro, "Trackit! enhancing gps-enabled applications for outdoor sports and activities," Master's thesis, Instituto Superior Tcnico, Univ. of Lisbon, Lisbon, 2014.