# Additional online material

# Appendix 1: Description of the two tested algorithms

# Appendix 2: Artificial GPS track generation

# Appendix 1: Description of the two tested algorithms.

## Details of Hariharan activity place extraction algorithm

Step-by-step description of the fixed threshold algorithm, as presented in Hariharan et al. [[1](#_ENREF_1)], noted *Aft*:

1. Define *Diameter* function to compute the greatest distance between all pairs of points in a set
2. Define *Medoid* function to compute the centroid of a point dataset
3. For each point Pi in the GPS point dataset, do:
	1. Find the next point Pj so that Pj – Pi ≥ ∆duration
	2. Check if *Diameter*(Pi, Pj) > ∆roam, in which case continue to point Pi+1
	3. Otherwise, do:
		1. Select all following points Pj’ so that *Diameter*(Pi, Pj’) ≤ ∆roam
		2. Extract stop S = *Medoid*(Pi, Pj’), with start and stop times ti and tj’
		3. Continue to point Pj+1’

## Details of the SPHERELab GPS activity place extraction algorithm

Step-by-step description of the kernel-density algorithm, noted *A*kd:

1. KD = kernelDensity(Ψ, β): A kernel density surface was calculated from the GPS point dataset Ψ using a quadratic kernel function with a fixed bandwidth *β* using ArcGIS kernel density tool. The resulting raster layer had a 10m square cell size and all cells with null-value were excluded from further analyses 🡪 **KD**.
2. HS = localMaxima(KD): Kernel density raster cells with a value lower than the global mean of the kernel density raster were excluded 🡪 **KD*f***. Watershed analyses (using ArcGIS Spatial Analyst) were run on the inversed surface to determine flow directions and sinks. Sinks corresponded to local density maxima. Raster sinks were then converted to vector points, which correspond to the detected activity places 🡪 **HS**.
3. ZHS = findZoneHS(KD*f*): From the filtered kernel density raster, we delineated the zone around each visited place within which GPS points contributed to the local density maximum, using the ArcGIS Basin Tool 🡪 **ZHS**.
4. Ψ1 = findAllocation\_1stPass(Ψ, KD*f*, ZHS, γ1): Zonal mean *µz* and standard deviation *σz* of kernel densities were calculated within each zone separately. These values were used to normalize the kernel density of each cell so that *nK* = (*K* – *µz*) / *σz* 🡪 **nKD*z***. Normalized kernel density, as well as zone identifier from **ZHS** raster, was then extracted to GPS point dataset Ψ through a spatial overlay. A convolution filter γ1 was then applied to the temporal sequence of GPS points, for smoothing of the normalized kernel density of each GPS point so that the normalized kernel value of past and following GPS points are taken in account, akin to a moving average of kernel density values. Filter γ1 was defined with the following coefficients: [0.1; 0.25; 0.3; 0.25; 0.1]. At this stage, each GPS point carried a tuple consisting of three values (*nK*, *nKc*, *ZHS*) corresponding respectively to the normalized kernel density, the convolved normalized kernel density and the zone identifier. Allocation of a GPS point *ψ* to a visited place was then based on the following rule:

if *nK* ≥ 1 OR *nKc* ≥ 1 then allocate *ψ* to corresponding visited place [R1].

Allocated points were marked with flag *falloc* = 1, while unallocated points were set to a null value.

1. Ψ2 = findAllocation\_2ndPass(Ψ1, γ2): A second pass to further refine allocation of GPS points Ψ1 was done to catch unallocated points with allocated neighbours, which may typically happen for GPS spike errors. Convolution filter γ2, with coefficients [0.5; 1; 1.5; 1; 0.5] was applied to the allocation flag *falloc*. Allocation of a GPS point *ψ* to a visited place ws then decided based on the second rule:

if *falloc* > 0 then allocate *ψ* to corresponding visited place [R2].

1. THS = findDuration(Ψ2): Once the set of allocated GPS points Ψ2 was established, the time spent at each visited place was computed by iterating over Ψ2 to find the starting and ending points of all contiguously allocated subsets of GPS points. Stop durations were derived from the timestamps for the corresponding visited place 🡪 **THS**.

NB: The kernel density algorithm presented here is offered online as an ArcToolbox for ArcGIS 10 on the authors’ website. Please go to [www.spherelab.org](http://www.spherelab.org), tools section.

# Appendix 2: artificial GPS track generation

The procedure to build the sets was adapted from Kami et al. [[2](#_ENREF_2)]:

1. Random selection of 750 pairs of coordinates within the Montreal Metropolitan Area; 2. Calculation of shortest road network for each pair of coordinates using Network Analyst extension in ArcGIS 10;

3. Transformation of resulting polylines into points, with a regular spacing of 10 m – corresponding to a travel speed of 36 km/h using a tracking device set at a one second epoch;

4. Random shift of points along X and Y axes to mimic imprecision of GPS tracks along followed routes; the shift following a bi-dimensional normal distribution centered on 0 with a standard deviation of 1.25 m;

5. Random selection of three data points per track to serve as stop locations;

6. Determination of stop durations λd using a Poisson distribution, with a seed value for the three stops randomly chosen from one of the three following normal distributions (mean ± stdev): (i) 10min ± 3min, (ii) 1h ± 15min and (iii) 5h ± 1h – these three classes of duration respectively representing typical short- (e.g. dropping kids at the kindergarten), medium- (e.g. going to the swimming pool) and long-term (e.g. work hours) activities.

7. Generation of additional stop points (1 point per second) with simulated noise along X and Y axes - noise added by shifting coordinates from original stop point using a normal distribution with mean 0 and standard deviation σn, σn being randomly drawn from a uniform distribution ranging from 10 to 200 m. Performance indicators are presented by groups of noise range – [0; 50[, [50, 100[, [100; 150[ and [150, 200] – bounds in meters.

Varying the two parameters λd and σn allowed to test the performance of algorithms in relation to stop duration and levels of GPS noise during stop periods.

NB: The algorithm for artificial track generation is available online as a Python code on the authors’ website. Please go to [www.spherelab.org](http://www.spherelab.org), tools section.

1. Hariharan R, Toyama K: **Project Lachesis: parsing and modeling location histories**. *Geographic Information Science* 2004:106-124.

2. Kami N, Enomoto N, Baba T, Yoshikawa T: **Algorithm for detecting significant locations from raw GPS data**. In: *Discovery Science: 2010*: Springer; 2010: 221-235.