

# Rights Protection of Trajectory Datasets

Claudio Lucchese <sup>\*</sup>, Michail Vlachos <sup>†</sup>, Deepak Rajan <sup>†</sup>, Philip S. Yu <sup>‡</sup>

<sup>\*</sup>University of Venice

<sup>†</sup>IBM T.J. Watson Research Center

<sup>‡</sup>University of Illinois at Chicago

**Abstract**—This work presents a technique of convincingly claiming ownership rights over a trajectory dataset. The presented methodology distorts imperceptibly a collection of sequences, effectively embedding a secret key, while retaining as well as possible the neighborhood of each object, which is vital for operations such as similarity search, classification or clustering.

## I. INTRODUCTION

Data sharing is an important aspect of scientific or business collaboration. However, data owners are also concerned with the protection of their rights on the datasets, which in many cases have been obtained after expensive and laborious procedures. The ease of data exchange through the Internet has compounded the need to assemble technological mechanisms for effectively protecting one’s intellectual or pragmatic property.

In this work we deal with the rights protection of trajectory datasets. Trajectories abound in applications such as GPS tracking experiments, video and motion capture data, and even image shapes can be considered as 2-dimensional trajectories. We provide ownership assurances on such datasets using watermarking principles. While there is a rich literature on watermarking for multimedia datasets, previous work is primarily concerned with watermarking a single object and not a collection of objects. Here, we consider the watermarking problem from a new perspective, by focusing on the additional maintenance of the inter-relationship between objects. Our technique embeds a secret key in each of the dataset objects, distorting them imperceptibly, while taking special consideration in retaining the original neighboring object. We call this operation *Neighbor Preserving* (NP) watermarking. Guaranteeing preservation of the nearest objects is very important for an array of search and mining operations, such as similarity search or Nearest-Neighbor(NN)-classification. Contrary to privacy-preserving approaches for data-mining that first add noise and then reconstruct the original data distributions based on the known noise model, our approach learns/calculates the *largest* amount of noise that can be added, so that nearest neighbors are not distorted. While the naive algorithms for determining the said watermarking power are costly, we show efficient ways of speeding up the process making it more than 2 orders of magnitude faster, thus allowing the technique to be applicable to large datasets.

### A. Watermark Embedding

Each trajectory is a vector of complex numbers  $x = \{x_1, \dots, x_n\}$ , with  $x_k = a_k + b_k i$ , where the real and

imaginary parts describe the 2D coordinates of the  $k$ -th point in  $x$ . In each object we embed a *watermark*, which is a *secret information* that will be hidden inside each sequence. The watermark is encoded as a vector  $W \in \{-1, 1, 0\}^n$ . In order to provide better resilience under attacks or data distortion, we will not embed the watermark in the original *space domain* but in the *frequency domain* instead. An additive Fourier embedding will generate a watermarked trajectory  $\hat{x}$  by replacing the magnitudes of each Fourier descriptor of  $x$  with watermarked magnitudes  $\hat{\rho}_j$ :

$$\hat{\rho}_j = \langle \rho_j + pW_j \rangle \stackrel{\text{def}}{=} \max(0, \rho_j + pW_j)$$

where *power*  $p > 0$  specifies the intensity of the watermark. Notice that the original phases will remain unchanged. An overview of this methodology is provided in Figure 1.

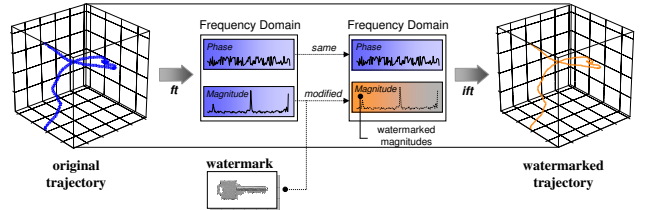


Fig. 1. Overview of watermarking technique.

The watermark will be embedded in the coefficients with the highest energy, because they capture the overall shape of the trajectory, making the watermark removal more difficult. Additional randomization can also be inserted in this portion, in order to provide better cryptographic properties.

**Error introduced by the watermark:** Altering a trajectory during the watermark embedding, adds some noise in the dataset. The relative error  $\epsilon$  introduced in a sequence  $x$  is:

$$\begin{aligned} \|x - \hat{x}\| &= \|X - \hat{X}\| = \dots = \\ &= \sqrt{\|\rho - \hat{\rho}\|^2 + 2 \sum_j \rho_j \hat{\rho}_j [1 - \cos(\phi_j - \hat{\phi}_j)]} \\ &= \|\rho - \hat{\rho}\| \quad (\text{since } \phi_j = \hat{\phi}_j) \\ &= \|\rho - \langle \rho + pW \rangle\|^2 \end{aligned}$$

Given a power  $p$ , we can calculate a tight upper bound of the error that may be introduced by assuming that  $\langle \rho + pW \rangle = (\rho + pW)$ . We denote this error with  $\epsilon_p$ , and we can write that:

$$\begin{aligned} \epsilon_p(x, \hat{x}) &= \frac{1}{\|x\|} \|x - \hat{x}\| = \frac{1}{\|x\|} \|\rho - (\rho + pW)\| \\ &= \frac{1}{\|x\|} \|-pW\| = \frac{1}{\|x\|} p \sqrt{l} \end{aligned}$$

Given this direct relationship between  $p$  and  $\epsilon$ , we will use  $p_\epsilon$  to indicate the power  $p$  equivalent to a given error  $\epsilon$  and similarly for  $\epsilon_p$ . Notice, that if  $\rho_j + pW_j < 0$  for some  $j$ , it means that  $W_j < 0$  and that the value of  $p$  is too large since it will produce a negative magnitude. We refer to this implicit power reduction phenomenon as the *power loss*.

In Figure 2 we can see the result of the additive Fourier embedding for a watermark of length  $l = 64$  at different powers. In this dataset, an error of  $\epsilon = 1\%$  is already visible, and therefore the user can set this as a potential upper bound on the watermark embedding power.

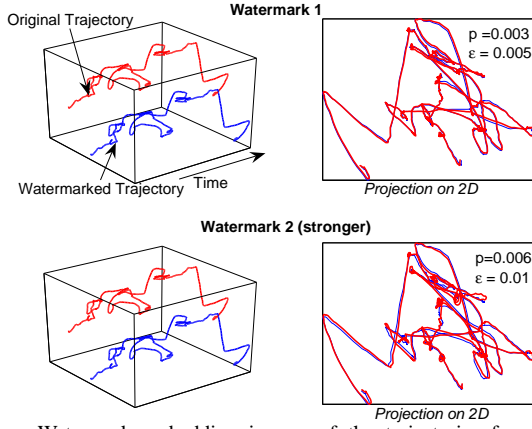


Fig. 2. Watermark embedding in one of the trajectories from a video tracking project. Here we embed the strongest possible watermark that we allow (relative error  $\epsilon = 1\%$ ) for showing the maximum possible distortion which still preserves perfectly the shape of the trajectory.

**Watermark Detection:** The detection step measures the correlation  $\chi$  between between  $W$  and the average magnitudes  $\mu(\hat{\mathcal{D}})$ , where  $\mu_j(\hat{\mathcal{D}})$  is the average magnitude  $\hat{\rho}_j$  for every trajectory  $\hat{x}$  in the modified dataset  $\hat{\mathcal{D}}$ . Every set of magnitudes  $\hat{\rho}$  is dominated by the original level of average magnitudes  $\mu(\mathcal{D})$ , which, in a sense, behave like a background noise, masking the embedded watermark  $pW$  we want to discover. We record  $\mu(\mathcal{D})$  during the embedding process and remove this sort of bias before the detection takes place.

$$\chi |_{\mu(\mathcal{D})} (W, \hat{\mathcal{D}}) = (\mu(\hat{\mathcal{D}}) - \mu(\mathcal{D})) \otimes W$$

where  $\otimes$  is the inner product of two vectors. So, we need to record  $2l$  values for the new watermark. We say that  $W$  has been embedded in  $\hat{\mathcal{D}}$  if  $\chi(W, \hat{\mathcal{D}}) \geq \omega$ , for a given threshold  $\omega$ . The closer  $\omega$  to the perfect correlation value  $pl$ , the smaller the probability of having false positives. We choose  $\omega$  being the threshold that provides the same probability of having a false positive or a false negative.

## II. NEIGHBOR-PRESERVING WATERMARKING

Now, we provide a solution for the **Neighbor-Preserving (NP) Watermarking Problem:** *Given dataset  $\mathcal{D}$ , minimum threshold ( $p_{min}$ ) and maximum threshold ( $p_{max}$ ), find the largest  $p$ ,  $p_{min} \leq p \leq p_{max}$ , such that after the watermark embedding, at most  $\tau \cdot |\mathcal{D}|$  watermarked objects  $\hat{x}$  have changed their original nearest neighbor.*

Equivalently, we want to find the largest power  $p$ ,  $p_{min} \leq p \leq p_{max}$ , such that the number of trajectories  $x \in \mathcal{D}$  that satisfy the inequality  $\hat{D}_p(x, NN(x)) > \hat{D}_p(x, y)$  is smaller than  $\tau \cdot |\mathcal{D}|$ . For our setting  $\tau = 0$ .

Solving this optimization problem for every pair of trajectories  $x, y$  can be very expensive, since it would require solving  $O(|\mathcal{D}|^2)$  inequalities, for counting how many trajectories lose their neighbor at any given power  $p$ . In the upcoming section we show how to mitigate the computational expense of explicitly solving all the pairwise distance inequalities.

**Shape of distance function:** One can show that the squared (Euclidean) distance between two watermarked trajectories  $\hat{D}_p^2(x, y)$  is a *piecewise* quadratic function of the embedding power  $p$  (proof is omitted for brevity). In Figure 3 we illustrate the squared distance between two trajectories with respect to the embedding power  $p$ . One can observe the piecewise form of the quadratic distance function (continuous dark colored line).

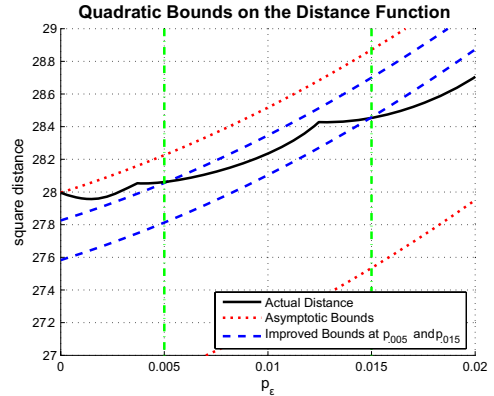


Fig. 3. Quadratic distance with respect to watermark embedding power (dark line). Asymptotic upper and lower bounds (dotted red line). Tighter bounds within power range (dashed blue line).

Recalling that by construction  $W$  is a vector in  $\{1, 0, -1\}^n$ , we can identify three types of contributions to the function  $\hat{D}_p^2(x, y)$ . Those frequencies corresponding to values  $W_j = 0$  are unaltered by the watermarking. The contribution given by those frequencies where  $W_j > 0$  can be easily described as a quadratic function of the power  $p$ . But, each frequency where  $W_j < 0$  induces a power loss phenomenon when  $p$  is sufficiently large. By definition, if  $W_j < 0$  the watermarked magnitude  $\hat{\rho}_j$  stops to decrease linearly when  $p > \rho_j$ .

However, for each interval of powers such that the number of frequencies affected by power loss does not change, the distance function is a quadratic. Since  $x$  and  $y$  may induce  $l/2$  points of discontinuity each, at most  $l + 1$  parabolas are sufficient to represent  $\hat{D}_p^2(x, y)$ .

Now, the problem of solving the inequality  $\hat{D}_p(x, NN(x)) > \hat{D}_p(x, y)$  reduces to the problem of solving as many quadratic inequalities as the number of pieces composing the two distance functions  $\hat{D}_p(x, NN(x))$  and  $\hat{D}_p(x, y)$ . In this case, the number of distinct end-points is at most  $3/2 \cdot l$ . By solving the inequality for every  $x$ , we gain complete knowledge about how its neighborhood is affected with respect to the embedding power  $p$ . This makes it possible

to discover the largest power that provides a misclassification rate of at most  $\tau$ . We call this algorithm *Exhaustive Search*, since it explores completely the behavior of every pair of objects at every power interval. An additional cost in the whole process is the cost of finding the nearest neighbor  $NN(x)$  for each trajectory. The total cost of the Exhaustive Search algorithm is  $O(nl|\mathcal{D}|^2 + n|\mathcal{D}|^2) = O(nl|\mathcal{D}|^2)$ .

### A. Fast Search Algorithms

One can speedup the above process by providing simple (i.e. *non-piecewise*) bounds for  $\widehat{D}_p$ . We define  $\widehat{DU}_p^2(x, y)$  to be the distance function  $\widehat{D}_p^2$ , where we use  $p = 0$  for those frequencies with  $W_j < 0$ . Similarly, we define  $\widehat{DL}_p^2(x, y)$  to be the distance function  $\widehat{D}_p^2$ , where we ignore the (non negative) contributions coming from those frequencies with  $W_j < 0$ , that is, we use  $p \rightarrow \infty$ . It is easy to see that  $\widehat{DL}_p^2 \leq \widehat{D}_p^2 \leq \widehat{DU}_p^2$ , and additionally it is possible to prove that these are asymptotically tight lower and upper bounds (Fig. 3 red dotted lines).

Unlike  $\widehat{D}_p^2$ , the two asymptotic bounds are simple quadratic functions, and therefore they can be used to approximate  $\widehat{D}_p^2$  improving our previous algorithm. In fact, if for some  $x$  and  $y$  the inequality  $\widehat{DL}_p(x, NN(x)) > \widehat{DU}_p(x, y)$  never holds according to the upper and lower bounds, then we can avoid to check every single discontinuity and skip to the next pair of trajectories. We call this new algorithm *Fast Asymptotic Search*.

These bounds can be improved if the user provides a specific search range  $[p_{min}, p_{max}]$ . For our experiments,  $p_{min} = p_0$  and  $p_{max} = p_{10^{-2}}$ . Now, we can define two improved bounds  $\widehat{DU}_p(x, y)$  and  $\widehat{DL}_p(x, y)$ , by using respectively  $p = p_{min}$  and  $p = p_{max}$  only for those frequencies where  $W_j < 0$ . It clearly holds that  $\widehat{DL}_p^2 \leq \widehat{D}_p^2 \leq \widehat{DU}_p^2$ . We use these improved bounds in a new algorithm called *Fast Improved Search* (Fig. 3 blue dashed lines).

Note that the fast search algorithms will return *exactly the same* result as the exhaustive search, since the bounds are only used to avoid (whenever possible) the examination of every single piece of discontinuity in the distance functions.

## III. EXPERIMENTS

We quantify the effectiveness of the Neighbor-Preserving algorithms for determining the watermark embedding power. Additionally, for the said power we examine the resilience of the rights protection scheme under a number of potential attacks. We utilize various datasets to verify our findings which cover areas such as video tracking, handwritten data and image contours.

**Comparison of NP algorithms:** We compare empirically the complexity of the Exhaustive Search and the Fast Search algorithms in terms of the number of solved inequalities. As reported in Table I, the improvement given by the use of improved bounds  $\widehat{DU}_p$  and  $\widehat{DL}_p$  is dramatic.

We also report the speed-up in running time achieved by the improved bounds. The speed of execution is strongly dependent on the dataset size; the larger the cardinality of objects in the dataset, the larger the improvement gained by the Fast Search algorithm. In the case of the Leaves dataset, the Exhaustive search took more than two days, compared with the 9 minutes taken by the improved search, leading to a 244-fold speedup.

Dataset	Exhaustive Search	Fast Search		Speedup
		asymptotic	improved	
VT1	6861	1854	271	<b>38x</b>
VT2	27273	2712	810	<b>48x</b>
Tablet	491664	51519	8381	<b>144x</b>
Skulls	224	1	1	<b>88x</b>
Fish	60515	72	10	<b>147x</b>
Leaves	1263375	2817	152	<b>244x</b>

TABLE I

NUMBER OF INEQUALITIES SOLVED WITH DIFFERENT DATASETS ( $p_{min} = p_0, p_{max} = p_{10^{-2}}, \tau = 0$ ). AND SPEEDUP BASED ON THE RUNNING TIMES OF THE FAST IMPROVED SEARCH VS. THE EXHAUSTIVE SEARCH.

**Resilience to attacks:** After having determined the best watermark embedding power, we test the watermark detectability under various adversarial attacks. These attacks attempt to transform the data with ultimate objective that of destroying the watermark, while at the same time not hindering significantly the dataset usability (i.e. the general shape of the trajectory cannot be completely distorted). In each test the embedding power  $p$  utilized, is the maximum power that preserves the original neighbor of all the trajectories. Geometric attacks, such as translation, scaling and rotation of trajectories (and combination of them), do not affect at all the data watermark, due to the embedding scheme and robust detection process. In fact, the false positive/negative rates under geometric attacks are smaller than  $10^{-10}$ , which means that we can detect the presence of the watermark with almost perfect accuracy. We also examine the effectiveness of the rights protection scheme under various types of attacks; under *noise addition* in the space and frequency domain, and under *resampling* attacks (see Fig. 4). These experiments show that the detectability of the embedded watermark is not hindered by such attacks, and a malicious adversary would have to destroy the usability of the dataset (distort the trajectories significantly) in an effort to erase the hidden watermark.

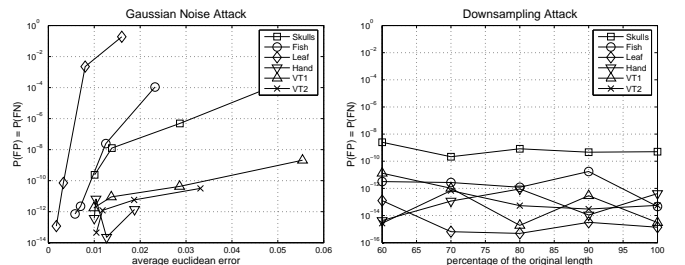


Fig. 4. Resiliency to attacks; (left) Gaussian Noise in Time, (Right) Downsampling attack.