A New Representation of Air Traffic Data Adapted to Complexity Assessment

Georges Mykoniatis†, Florence Nicol‡, Stephane Puechmorel (*)

Ecole Nationale de l'Aviation Civile
Toulouse France
Email: †georges.mykoniatis@enac.fr, ‡florence.nicol@enac.fr, (*)stephane.puechmore@enac.fr

***Abstract*—Air traffic is generally characterized by simple indicators like the number of aircraft flying over a given area or the total distance flown during a time window. As an example, these values may be used for estimating a rough number of air traffic controllers needed in a given control center or for performing economic studies. However, this approach is not adapted to more complex situations such as those encountered in airspace comparison or air traffic controllers training. An innovative representation of the traffic data, relying on a sound theoretical framework, is introduced in this work. It will pave the way to a number of tools dedicated to traffic analysis. Based on an extraction of local covariance, a grid with values in the space of symmetric positive definite matrices is obtained. It can serve as a basis of comparison or be subject to filtering and selection to obtain a digest of a traffic situation suitable for efficient complexity assessment.**

***Keywords*—Air traffic complexity; spatial data; manifold valued images; covariance function estimation; non-parametric estimation.**

I. INTRODUCTION

Key performance indicators (KPI) are of common use in air transportation. However, they are designed mainly to address global aspects of the systems and cannot address problems, where it is mandatory to be able to distinguish between traffic situations based on the structure of the trajectories. As an example, the training of air traffic controllers relies on carefully selected traffic patterns that are presented to the trainees in an order of increasing perceived complexity. Creating such scenarios is quite a lengthy process, involving hundreds of hours of works by experienced controllers. On the other hand, it is easy to start from real situations, with known flight plans, and to use a traffic simulator to put the trainees in a nearly operational setting. The drawback of this approach is the need to evaluate the traffic patterns in order to assess a complexity value for each of them. It has to be done automatically, to avoid having to resort to human experts.

In a more operational context, nearly the same question arises when trying to find the right number of controllers needed at a give time to take care of the incoming flights in their assigned airspace. Too many controllers induces an extra cost and too few put a high pressure on the operators, with possible detrimental effects on flight safety. Assessing the right level of complexity of the expected traffic may greatly improve over the current state of the art that simply estimates the number of aircraft that will be present. Once again, it is mainly a matter of finding an adequate traffic complexity indicator [1] [2].

A lot of work was dedicated to the issue of air traffic complexity measurement. Unfortunately, no really satisfactory solution exists, as the problem itself is ill posed: depending on the point of view, the complexity may be a concept roughly equivalent to the cognitive workload or, on the contrary, be based on purely structural features, without any reference to the way it will be used. One of the most widely used complexity measures is the dynamic density [3], that combines several potential indicators, like number of maneuvering aircraft, number of level changes, convergence and so on. All these values are used as inputs of a multivariate linear model, or in recent implementations, of a neural network. The tuning of the free parameters of the predictors is made using examples coming from an expertized database of traffic situations. While being quite efficient for assessing complexity values in an operational context, the method has two important drawbacks:

- The tuning procedure requires a sufficient number of expertized samples. A costly experiment involving several air traffic controllers must be set up.
- The indicator is valid only in a specific area of the airspace. Adaptation to other countries or even control centers requires a re-tuning that is almost as complicated as the first one.

The last point is a severe flaw if one wants to use dynamic complexity in the context of air traffic databases, as a world covering has to be obtained first. Even for country sized databases, some geographical tuning has to be added.

Another way to deal with complexity is through purely geometrical indicators [4] [5]. Within this frame, there is no reference to a perceived complexity but only to structural features. An obvious benefit is that the same metric may be used everywhere, without needing a specific tuning. It is also the weak point of the method as the relation with the controllers workload is not direct.

The present article introduces the theoretical material underlying a new approach to complexity assessment and more generally to traffic characterization, based on a representation of traffic situations as images whose pixels are covariance matrices. The idea underlying it is that local disorder is an indicator of complexity that captures most of the elementary metrics entering the dynamic density. This is a work in

 ISBN: 978-1-61208-631-6

progress that will ultimately allow the use of deep learning on such pseudo-images in conjunction with an expertized database to produce a complexity metric with low tuning requirements. A by product is the ability to compute distances between traffic situations, allowing for efficient indexing in dedicated databases. The rest of the paper is structured as follows. In Section II, the traffic is modeled after a Gaussian random field, whose covariance function is estimated on two dimensional grid. In Section III, tools dedicated to the processing of such grids of symmetric positive definite matrices are introduced. Finally, in Section IV, a conclusion is drawn, introducing the next generation of algorithms able to exploit this novel representation.

II. Traffic representation

The first stage in the computation of the complexity index is the processing of traffic samples in order to summarize their local features. It is done by estimating a local covariance matrix at each point of an evenly spaced grid. While aircraft positions are actually points in $\mathbb{R}^3$, the altitude plays a special role and is not presented on controllers displays. The choice made in the present work is to use a planar representation, disregarding the altitude, which is in compliance with the operational setting. From here on, all the derivations will be made using aircraft positions in $\mathbb{R}^2$.

All samples are assumed to be dated positions (t, x) where t is the sampling time and $x \in \mathbb{R}^2$ is the sample position. Finally, a dataset is simply a sequence $(t_i, x_i)_{i=1\dots N}$ of samples collected in a given time interval and spatial area. Please note that samples will not be distinguished by the trajectory they belong to, so that different flight patterns may generate exactly the same dataset. This is a limitation of the present work that will be addressed in a future extension of the method.

A. A Gaussian field model

Collected samples without taking time into consideration may be viewed as realizations of an underlying spatial stochastic process X with values in $\mathbb{R}^2$. Such a process is called a Gaussian vector field when for any collection of points $(x_1, \dots, x_p)$, the joint distribution of the random variables $(X(x_1), \dots, X(x_p))$ is Gaussian. Such a process is characterized by its mean and covariance functions:

$$\mu(x) = E[X(x)] \tag{1}$$

$$C(x, y) = E\left[(X(x) - \mu(x))(X(y) - \mu(y))^t\right] \tag{2}$$

In practice, μ and C must be estimated from a dataset of couples $(x_i, v_i)_{i=1\dots N}$ where v_i is the observed vector value at position x_i. Available methods fall into two categories: parametric and non parametric. In the parametric approach, μ and C are approximated by members of a family of functions depending on a finite number of free parameters that are tuned to best match the observations. A usual choice is to use finite cubic splines expansions whose coefficients are identified by a least square fitting. The efficiency of parametric estimation is heavily dependent on the actual mean and covariance functions and may be computationally expensive for large datasets. In the non parametric approach, a different methodology is used: the samples themselves act as coefficients of an expansion involving so-called kernel functions. Apart from the obvious benefit of avoiding a costly least square procedure, most of the kernels used in practice are compactly supported, so that evaluating an approximate mean and covariance at a given location requires far less terms that the number of samples. Due to its simplicity and the previous property, a non parametric estimation was selected to process the traffic.

B. Mean and covariance matrix estimation

The key ingredient in non parametric estimation is the kernel function, that is a smooth enough mapping $K\colon \mathbb{R}^2 \to \mathbb{R}^+$ that integrates to 1.

It is usually more tractable to restrict kernels to a smaller class:

Definition 1. *A radial kernel is a mapping $K\colon \mathbb{R}^2 \to \mathbb{R}^+$ that is of the form $K(x) = K(\|x\|)$ and such that :*

$$\int_{\mathbb{R}^2} K(x)dx = 1$$

From here on, only radial kernels will be used.

Kernels are most of the time at least continuous. Furthermore, as mentioned above, compactly supported function will save a lot of computation in the evaluations and it is thus advisable to use kernels pertaining to this class. Classical examples are the multivariate Epanechnikov kernel [6] and its higher regularity versions which are widely used in the non-parametric estimation community:

$$K_e(x) = \frac{2}{\pi}\left(1 - \|x\|^2\right)_+ \tag{3}$$

$$K_2(x) = \frac{3}{\pi}\left(1 - \|x\|^2\right)_+^2 \tag{4}$$

$$K_3(x) = \frac{4}{\pi}\left(1 - \|x\|^2\right)_+^3 \tag{5}$$

Given a radial kernel K and a positive real number h, the scaled kernel K_h is defined to be:

$$K_h(x) = \frac{1}{h^2} K\left(\frac{x}{h}\right) \tag{6}$$

h is termed as the bandwidth of the kernel and controls the degree of smoothing introduced by the kernel and its support. It has to be tuned with respect to the dataset to obtain the best compromise between smoothing and accuracy. A kernel estimator of the covariance function C of a stationary stochastic process X can be found in [7]. Using a straightforward extension, a kernel estimator for the correlation function of a locally stationary random field is given in [8]. Finally, a weighed maximum likelihood approach is taken in [9], for computing at any location x the mean $\mu(x)$ and variance $C(x, x) = \Sigma(x)$ of a Gaussian random field. This last work can easily be

 ISBN: 978-1-61208-631-6

generalized to yield an estimate for the covariance function, under normality assumption for the random field X. Given a dataset $(x_i, v_i)_{i=1\dots N}$, where the sampling positions x_i are assumed to be distinct, the weighted joint log likelihood of the couples $(x_i, v_i), (x_j, v_j), j \neq i$ at locations x, y is given, for a fixed kernel bandwidth h, by:

$$L(x,y) = \\ -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} V_{ij}^t \Sigma^{-1}(x,y) V_{ij} K_h\left(\|x_i - x\|\right) K_h\left(\|x_j - y\|\right) \\ +\frac{1}{2}\log\left(\left|\Sigma^{-1}\right|\right)\left(\sum_{i=1}^{N}\sum_{j=1^N} K_h\left(\|x_i - x\|\right) K_h\left(\|x_j - y\|\right)\right) \\ + A$$

where:

$$V_{ij} = (v_i - m(x), v_j - m(y))$$

$m(x)$ is the mean of $X(x)$ and $\Sigma(x,y)$ is the variance matrix of the Gaussian vector $(X(x), X(y))$. The term A accounts for the $\log$ of the normalizing constant occurring in the expression of the multidimensional normal distribution and will play no role in the sequel. Please note that the kernel was selected to be a product of elementary kernels.

The differential of the log likelihood with respect to the mean value can be computed as:

$$\sum_{i,j} \operatorname{tr}\left(V_{ij}^t \Sigma^{-1}(x,y)\right) K_h\left(\|x_i - x\|\right) K_h\left(\|x_j - y\|\right) \quad (7)$$

The first order optimality condition yields for the non-parametric estimate for the mean function:

$$\hat{m}(x) = \frac{\sum_{i=1}^{N} v_i K_h\left(\|x - x_i\|\right)}{\sum_{i=1}^{N} K_h\left(\|x - x_i\|\right)} \quad (8)$$

A similar derivation can be made to obtain the differential with respect to the Σ matrix, using the two identities below:

$$d\Sigma^{-1} = -\Sigma^{-1} d\Sigma \Sigma^{-1} \quad (9)$$

$$d\log\left(\left|\Sigma^{-1}\right|\right) = -\operatorname{tr} d\Sigma \Sigma^{-1} \quad (10)$$

The non-parametric estimator for $\Sigma(x,y)$ is then:

$$\hat{\Sigma}(x,y) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} \hat{C}_{ij}(x,y) K_h\left(\|x_i - x\|\right) K_h\left(\|x_j - y\|\right)}{\sum_{i=1}^{N}\sum_{j=1}^{N} K_h\left(\|x_i - x\|\right) K_h\left(\|x_j - y\|\right)} \quad (11)$$

with:

$$\hat{C}_{ij}(x,y) = \begin{pmatrix} v_i - \hat{m}(x) \\ v_j - \hat{m}(y) \end{pmatrix} \begin{pmatrix} v_i - \hat{m}(x) \\ v_j - \hat{m}(y) \end{pmatrix}^t$$

Using the definition 8 of $\hat{m}$, it appears that $\hat{\Sigma}(x,y)$ is block diagonal:

$$\begin{pmatrix} \hat{\Sigma(x)} & 0 \\ 0 & \hat{\Sigma(y)} \end{pmatrix} \quad (12)$$

with:

$$\Sigma(x) = \frac{\sum_{i=1}^{N} \hat{C}_{ii}(x) K_h\left(\|x_i - x\|\right)}{\sum_{i=1}^{N} K_h\left(\|x_i - x\|\right)} \quad (13)$$

This estimator is similar to the one in [8], and is of Nadaraya-Watson [10] type. It enjoys asymptotic normality. The reason for the vanishing of the off diagonal blocks is a consequence of the special shape of the kernel that implicitly approximates the joint distribution of $X(x)$ and $X(y)$ by product laws.

C. Computation of the mean and covariance functions

In order to allow further treatments, mean and covariance functions will be evaluated only at points located on an evenly spaced two dimensional grid whose points are located at coordinates:

$$p_{nm} = (x_0 + n\delta_x, y_0 + m\delta_y)$$
$$n \in \{-L, \dots, L\},\ m \in \{-M, \dots, M\}$$

where δ_x, δ_y are respective step sizes along x and y axis. In the expressions 8,13 of the mean and covariance functions evaluated at grid point p_{nm}, the kernel appears as $K_h\left(\|x_i - p_{nm}\|\right)$. If the grid is fine enough, one can approximate it by $K_h\left(\|p_{kl} - p_{nm}\|\right)$ where p_{kl} is the grid point closest to x_i. Using coordinates, we have:

$$K_h\left(\|p_{kl} - p_{nm}\|\right) = K_h\left(\sqrt{(n-k)^2\delta_x^2 + (m-l)^2\delta_y^2}\right) \quad (14)$$

Values depends only on the difference between the grid points indices and are thus independent on the location p_{kl}. Furthermore, since K is assumed to have compact support, the same is true for K_h so that $K_h\left(\sqrt{(n-k)^2 + (m-l)^2}\right)$ will vanish when indices differences exceed a given threshold. Gathering things together, all non-zero values of the kernel can be tabulated on a grid of size $(2P+1) \times (2Q+1)$ if the support of the kernel K_h is contained in the square $[-P\delta_x, P\delta_x] \times [-Q\delta_y, Q\delta_y]$:

$$K_h(i,j) = K_h\left(\sqrt{n^2\delta_x^2 + (m)^2\delta_y^2}\right) \quad (15)$$

$$n \in \{-P, \dots, P\}, m \in \{-Q, \dots, Q\} \quad (16)$$

All the entries in the equation 15 can be scaled so that they sum to 1: this saves the division by the sum of kernel values. Simultaneous evaluation of the mean at all grid points can then be made in an efficient manner using Algorithm 1. Once the mean has been computed, the covariance is estimated the same way (see Algorithm 2).

When the grid is not fine enough to replace the true sample location by a grid point, a trick based on bilinear interpolation can be used. Using again the equation 15 and the closest grid point $p_{k_1 l_1}$ to x_i, the true sample position x_i will be located within a cell as indicated in Figure 1. The kernel value can be approximated as:

$$K_h(k_1, l_1) + \frac{dx}{\delta_x}a + \frac{dy}{\delta_y}b + \frac{dx}{\delta_x}\frac{dy}{\delta_y}c \quad (17)$$

 ISBN: 978-1-61208-631-6

Algorithm 1 Mean kernel estimate

for $i \leftarrow 0, 2L; j \leftarrow 0, 2 * M$ **do**
  $m(i,j) \leftarrow 0$
end for
for $k \leftarrow 0, N-1$ **do**
  $(k,l) \leftarrow$ ClosestGridPoint(x_i)
  for $i \leftarrow -P, P; j \leftarrow -Q, Q$ **do**
    if $k+i \geq 0 \wedge k+i \leq 2L$ **then**
      if $l+j \geq 0 \wedge l+j \leq 2M$ **then**
        $m(k+i, l+j) \leftarrow m(k+i, l+j) + K_h(i,j)v_i/N$
      end if
    end if
  end for
end for

Algorithm 2 Covariance kernel estimate

for $i \leftarrow 0, 2L; j \leftarrow 0, 2 * M$ **do**
  $C(i,j) \leftarrow 0$
end for
for $k \leftarrow 0, N-1$ **do**
  $(k,l) \leftarrow$ ClosestGridPoint(x_i)
  for $i \leftarrow -P, P; j \leftarrow -Q, Q$ **do**
    if $k+i \geq 0 \wedge k+i \leq 2L$ **then**
      if $l+j \geq 0 \wedge l+j \leq 2M$ **then**
        $A \leftarrow (v_i - m(k,l))(v_i - m(k,l))^t$
        $C(k+i, l+j) \leftarrow C(k+i, l+j) + K_h(i,j).A/N$
      end if
    end if
  end for
end for

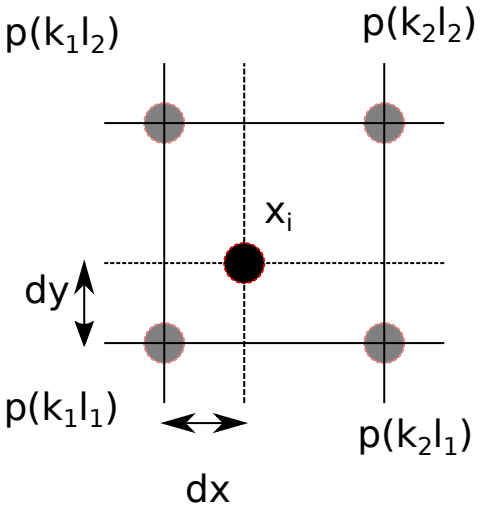

Fig. 1. Bilinear interpolation

with:

$$a = K_h(k_2, l_1) - K_h(k_1, l1)$$
$$b = K_h(k_1, l_2) - K_h(k_1, l1)$$
$$c = K_h(k_2, l_2) + K_h(k_1, l1) - K_h(k_2, l_1) - K_h(k_1, l_2)$$

Gathering terms by tabulated values yields a kernel value:

$$K_h(k_1, l_1)\,(1 - s_x - s_y + s_x s_y) \quad (18)$$
$$+ K_h(k_2, l_1)\,(s_x - s_x s_y) \quad (19)$$
$$+ K_h(k_1, l_2)\,(s_y - s_x s_y) \quad (20)$$
$$+ K_h(k_2, l_2) s_x s_y \quad (21)$$

where:

$$sx = \frac{dx}{\delta_x},\ sy = \frac{dy}{\delta_y}$$

It is thus possible to compute the mean and covariance functions on a coarser grid using Algorithms 1 and 2 by applying them on the four locations $(k_1, l_1), (k_2, l_1), (k_1, l_2), (k_2, l_2)$, with an observed value multiplied by their respective coefficients $(1 - s_x - s_y + s_x s_y)$, $(s_x - s_x s_y)$,$(s_y - s_x s_y)$,$K_h(k_2, l_2) s_x s_y$.

The overall complexity of the algorithm is linear in the number of grid points and in the number of samples. It is similar to filtering an image and can be implemented the same way on modern Graphics Processing Units (GPU). Please note also that for kernels with large supports, a fast Fourier transform may be used at the expense of a slight increase in the complexity order that will be balance by the constant term due to the support size.

III. Processing tools

The preceding phase allows the computation of a traffic pattern digest as a two dimensional grid of Symmetric Positive Definite (SPD) matrices. It may be used as is for building an index in a database, using the same procedure as for images. However, the geometry underlying the space of 2×2 positive definite matrices is not euclidean, but hyperbolic. The proposed index is an adaptation of images distances, using hyperbolic geometry.

A. The Riemannian structure of symmetric positive definite matrices

The purpose of this part is to introduce at a basic level the tools used to build the index. Results are given without proofs, the interested reader may refer to [11] for a more in-depth exposition.

Proposition 1. *The space of $n \times n$ SPD matrices, denoted by $SPD(n)$, may be endowed with a Riemannian manifold structure with metric at point A given by the differential:*

$$ds^2 = \mathrm{tr}\left(A^{-1} dA\right) \quad (22)$$

Proposition 2. *Let A, B be SPD matrices. It exists a unique minimizing geodesic joining them in $SPD(n)$. It is given in parametrized form by:*

$$\gamma : t \in [0,1] \mapsto A^{1/2}\left(\exp t \log\left(A^{-1/2}BA^{-1/2}\right)\right)A^{1/2} \tag{23}$$

Proposition 2 yields the geodesic distance between any two matrices A, B from $SPD(n)$ as $d(A,B) = \sqrt{\mathrm{tr}\log^2(A^{-1}B}$. It can be expressed as $d(A,B) = \sqrt{\sum_{i=1}^{n}\log^2\lambda_i}$ with $\lambda_i, i = 1\ldots n$ the eigenvalues of $A^{-1}B$.

The geodesic distance between matrices from $SPD(2)$ may be used to compute a distance between grids produced by the traffic processing phase in a very simple way, as indicated in Algorithm 3.

Algorithm 3 Distance between grids

1: A, B are $P \times Q$ grids of $SPD(2)$ matrices.
2: $dsq = 0$
3: **for** $i \leftarrow 0, P-1; j \leftarrow 0, Q-1$ **do**
4: $\quad dsq \leftarrow dsq + \mathrm{tr}\log^2(A(i,j)^{-1}B(i,j)$
5: **end for**
6: $d(A,B) = \sqrt{dsq}$

Please note that this distance is based on a point-wise comparison and is very similar to the L^2 distance used for images. It has a higher cost of evaluation due to the distance computation in $SPD(2)$ that involves an matrix inverse, product and logarithm. However, it is not as critical in $SPD(2)$ than it may be in a general $SPD(n)$, since eigenvalues and eigenvectors can be computed in closed form (this is true also in $SPD(3)$ and $SPD(4)$, with more complex expressions). Furthermore, grid distance computation is easy to parallelize on modern graphics hardware since it involves independent operations on small matrices. As an example, computing the distance between two grids of size 100×100 on a TitanX pascal card from Nvidia takes around 100μs.

B. Grid filtering

In the traffic processing phase, grids have sizes ranging from 100×100 to 300×300. Due to the processing cost incurred by the $SPD(2)$ setting, it is advisable in many cases, and especially if one wants to use the grids as index in a traffic database, to reduce the size of grids to more tractable dimensions, say 10×10 to 50×50. This has to be done without wiping out the salients features of the traffic captured by the original grid. In the spirit of what is done in the first layers of an image processing deep network, it is proposed to apply in sequence a filtering and a selection process on the original grid.

Definition 2. *Let $A_i, i = 1\ldots n$ be a sequence of elements of $SPD(n)$, $w_1, \ldots w_n$ be a sequence of real numbers and B be an element of $SPD(n)$. The log-euclidean weighted combination (LWC) at B of the $(A_i)_{i=1\ldots n}$ with weights $(w_i)_{i=1\ldots n}$ is the matrix:*

$$B^{1/2}\exp\left(\sum_{i=1}^{n} w_i \log\left(B^{-1/2}A_iB^{-1/2}\right)\right)B^{1/2} \tag{24}$$

The LWC may be used to compute a filtered version of a grid using the same procedure as for an image. The process is given in Algorithm 4 that yields the filtered grid as B.

Algorithm 4 Grid filtering

1: A is a $P \times Q$ grid of $SPD(2)$ matrices.
2: $w_i, i = 1\ldots9$ is a sequence of real numbers
3: **for** $i \leftarrow 0, P-1; j \leftarrow 0, Q-1$ **do**
4: $\quad (C_1, \ldots C_9)$ are the adjacent cells to $A(i,j)$ and itself.
5: $\quad B(i,j) \leftarrow LWC(C_1, \ldots, C_9)$ with weight $w_i, i = 1\ldots9$ at $A(i,j)$.
6: **end for**

The filtering process on $SPD(2)$ grids behaves roughly like in image processing: when the weights are real numbers in the interval $[0,1]$ that sum to 1, then a weighted mean is produced. It tends to smooth out the grid, making spatially close matrices more similar. On the opposite, when weights sum to 0, the equivalent of a high pass filter is produced, that emphases sharp variations. Please note that the size of the grids after filtering is unaltered.

The second processing phase is simplification to reduce grid size. The main idea is to replace a block of grid cells by a single one using a digest. An obvious approach is to replace a block by its mean, that can be obtained from LWC by using equal positive weights $1/n$ if n is the number of cells in the block. A major drawback is that the important information tends to be lost, with matrices going close to multiples of the identity in many cases. Another way of dealing with the problem is to introduce an order on $SPD(2)$ and to select the largest (resp. lowest) element in the block. This procedure has two benefits:

- The selected matrix is an element of the original grid.
- As in deep learning networks, it will select the most representative elements.

After some experiments on simulated matrix images, the order chosen is a lexicographic one, the first comparison being made on the determinant of the matrices and the second on the trace. After the selection phase, the size of the grid is reduced by the ratio of the number of elements considered in a block. In the current implementation, it is 3×3, thus shrinking the grid by a factor 3 in each dimension. The filtering/selection phases may be chained in order to get smaller grids. As for the distance computation, it is quite easy to implement the process on a GPU, all operations being independent.

IV. Conclusion and future work

The work presented here is still in early stage of development. Only theoretical concepts and computer implementation

of the traffic processing, filtering and selection phases are completed or nearly completed. Testing on real data is yet to be done, and a complete complexity computation metric has to be built. The next steps are:

- Constitution of a traffic database from surveillance (Radar) and ADS-B data. This work has been launched, and data collection is in progress.
- Complexity index coding. Although most of the software bricks needed are available or close to release, it remains to implement the overall process. A weight adjustment procedure is lacking for the filtering phase: it is the topic of a more theoretical work and involves some Lie group techniques.
- Evaluation against existing indicators and expert advices.

Based on the experience gathered on the topic, it is expected that the new approach presented here will outperform the current state-of-the-art metrics. Furthermore, thanks to the ability to compute the distance between two grids of $SPD(2)$ elements, it offers the unique opportunity to derive an index for a traffic situation database that will be an invaluable tool for practitioners in the field of air traffic management.

REFERENCES

[1] M. Prandini, L. Piroddi, S. Puechmorel, and S. L. Brazdilova, "Toward air traffic complexity assessment in new generation air traffic management systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 809–818, Sept 2011.

[2] A. Cook, H. A. Blom, F. Lillo, R. N. Mantegna, S. Miccich, D. Rivas, R. Vzquez, and M. Zanin, "Applying complexity science to air traffic management," *Journal of Air Transport Management*, vol. 42, pp. 149 – 158, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0969699714001331

[3] L. I. S. Shelden, R. Branstrom, and C. Brasil, "Dynamic density: An air traffic management metric," NASA, Tech. Rep. NASA/TM-1998-112226, 1998.

[4] K. Lee, Feron, and A. E.and Prichett, "Air traffic complexity : An input-output approach," in *Proceedings of the US Europe ATM Seminar*. Eurocontrol-FAA, 2007, pp. 2–9.

[5] D. Delahaye and S. Puechmorel., "Air traffic complexity based on dynamical systems." in *Proceedings of the 49th CDC conference*. IEEE, 2010.

[6] V. A. Epanechnikov, "Non-parametric estimation of a multivariate probability density," *Theory of Probability & Its Applications*, vol. 14, no. 1, pp. 153–158, 1969. [Online]. Available: https://doi.org/10.1137/1114019

[7] P. Hall, N. I. Fisher, and B. Hoffmann, "On the nonparametric estimation of covariance functions," *Ann. Statist.*, vol. 22, no. 4, pp. 2115–2134, 12 1994. [Online]. Available: https://doi.org/10.1214/aos/1176325774

[8] Y. Li, N. Wang, M. Hong, N. D. Turner, J. R. Lupton, and R. J. Carroll, "Nonparametric estimation of correlation functions in longitudinal and spatial data, with application to colon carcinogenesis experiments," *Ann. Statist.*, vol. 35, no. 4, pp. 1608–1643, 08 2007. [Online]. Available: https://doi.org/10.1214/009053607000000082

[9] J. Yin, Z. Geng, R. Li, and H. Wang, "Nonparametric covariance model," *Statistica Sinica*, vol. 20, no. 1, pp. 469–479, 2010. [Online]. Available: http://www.jstor.org/stable/24309002

[10] E. A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964. [Online]. Available: https://doi.org/10.1137/1109020

[11] F. Nielsen and R. Bhatia, *Matrix Information Geometry*. Springer Berlin Heidelberg, 2012. [Online]. Available: https://books.google.fr/books?id=MAhygTspBU8C

 ISBN: 978-1-61208-631-6