# Trajectory optimization considering contrails

Gaspard DANNET

January 17, 2023

**Abstract**

Under certain atmospheric conditions, the water vapour emitted by airplanes is the source of contrails (short for "condensation trails"). These "contrail cirrus", which can spread over hundreds of square kilometers, trap heat coming off the earth that would otherwise head for space. In this article a method is described to take into account areas favorable to contrails when calculating the trajectory of an aircraft using a FMT* algorithm in order to reduce the environmental impact of the aviation. As the computed trajectories take good consideration of obstacles, the results are encouraging.

## 1 Introduction

As it is crucial nowadays to halt global warming and knowing that aviation is one of the most important global economic activities in the modern world, there is an urgent need to decline aviation's contribution to global warming. To do so, all stakeholders are trying to improve the aerodynamics of aircraft, develop new engines or even use renewable energies such as hydrogen or bio-fuel. However all of these ideas tend to reduce the aviation $CO_2$ emissions but aviation is a sector having significant non-$CO_2$ radiative forcing. Indeed, non-$CO_2$ effects could have the same or even greater impact on the climate than $CO_2$ [1]. These non-$CO_2$ effects result form the emissions of nitrogen oxide ($NO_x$), water vapor, aerosols and contrail cirrus clouds. Contrails being the main contributor.

Contrails evaporate quickly if the ambient air is dry, but persist and evolve into more cirrus clouds if the ambient air is humid enough [2]. Proposals have been made to avoid contrail formation, for example by re-routing aircraft or optimizing flight times to avoid the more positive – warming – forcings (*e.g.* by avoiding night flights [3]).

Thus, in order to reduce the impact of aviation on climate change, we will in this article optimize the aircraft trajectory, based on favorable areas of contrails appearances. These areas will be considered as "soft" obstacles. Indeed, it is best to avoid them but without increasing too much flight distance which would increase $CO_2$ emissions.

This paper will fold into four parts. First of all, a summary of the previous related works is given. Then the modeling and the resolution of the problem are presented. Afterward, the results of the study are discussed.

## 2 Related Work

The optimization of aircraft trajectories has always been an active field of research. Indeed, even when climate change was not the main objective, saving fuel and therefore saving money was.

Chakravarty, in his article [4], poses the problem using the equation of aircraft dynamics and chooses to minimize fuel spent. To solve this problem he used Pontryagin's minimum principle which provides conditions that an optimal trajectory must satisfy. This study was done in many different scenarios, *e.g.* in cruise, in climb, in descent, in presence of wind, even with a constant Mach number or in constant flight path angle descent. Using the same resolution method, Sridhar *et al.* [5] add to the modeling the impact of contrails. Thus, given a cost associated with contrails areas and at fixed altitude, a wind-optimal trajectory is computed which avoids regions of airspace that facilitate persistent contrails formation.

However, other methods, instead of using the equations of the flight dynamics discretize the environment and work on the resulting graph, making the problem simpler and more similar to well-known shortest-path problems or tree search. In [6], in a deterministic way, a wind grid is computed and interpolated using Shepard's method [7]. Thus by using Bellman's algorithm on this grid, an optimal trajectory

is computed for one weather sample. But, in this article, Legrand *et al.* do not take into account the contrails in their resolution. And a major problem is the time complexity, indeed for a grid with $n$ nodes, Bellman's algorithm has a complexity of $\mathcal{O}(n^2)$. Moreover, the results depends on the grid. A differently configured grid may give a different outcome.

To solve these issues, we do not use a deterministic method in this article but rather a sampling based method. The key idea behind this method is to sample the environment more or less randomly and search the shortest path on this sampling. We will be using in this article the Fast Marching Tree algorithm (FMT*) [8]. It is proven to be asymptotically optimal and is shown to converge to an optimal solution faster than its state-of-the-art counterparts, chiefly PRM* [9] and RRT* [10].

# 3 Modeling

After presenting different methods to optimize aircraft trajectories considering wind and contrails the modeling of the problem will be studied.

In this article the goal is to minimize the environmental impact of aircraft based emissions during cruise. Thus, we will minimize the $CO_2$ and non-$CO_2$ effects, *i.e.* contrails formation. To do so, the flight time will be minimized and therefore the fuel consumption too. To model persistent contrails, we will be using penalty functions as areas to be avoided. The penalty functions enable a systematic way of generating aircraft trajectories that avoids the contrails formation areas by varying amounts.

Therefore, the problem is written as follows :

$$\underset{\gamma}{\text{minimize}} \quad \int_{t_0}^{t_f} C_t + C_r r(x,y)\,\mathrm{d}t \tag{1a}$$

$$\text{subject to} \quad \gamma \in \Gamma \tag{1b}$$

where $C_t$ is the cost coefficient of time, $C_r$ the cost associated with contrails, $r(x,y)$ the penalty function and $\Gamma$ the set of flyable trajectories. The penalty function used in this work will be a uniform penalty function defined by :

$$r(x,y) = \begin{cases} \text{constant} & \text{if aircraft in penalty area} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

# 4 Resolution

To solve the problem proposed at the end of the last section (see Equation 1), the trajectory of the aircraft is described by waypoints. Each waypoint defined by its coordinates – longitude and latitude. The altitude is not taken into account because we limit ourselves to the cruise in this study, it represents indeed the major part of the flight. A trajectory will then be a succession of waypoints linked together. To generate these waypoints, a probabilistic yet asymptotically optimal method is used : the Fast Marching Tree algorithm (FMT*).

A basic pseudo-code description of FMT* is given below :

**Algorithm 1** Fast Marching Tree Algorithm (FMT*)

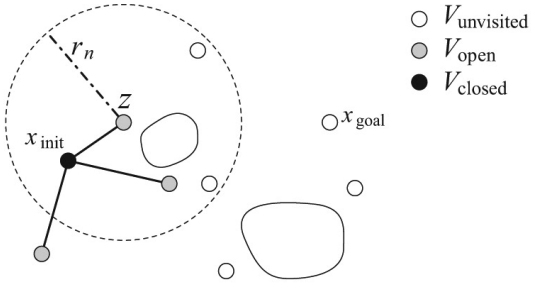
**Require:** Sample set V comprised of $x_{\text{init}}$ and $n$ samples in $\mathcal{H}_{\text{free}}$, at least one of which is also in $\mathcal{H}_{\text{goal}}$

1: Place $x_{\text{init}}$ in $\text{V}_{\text{open}}$ and all other samples in $\text{V}_{\text{unvisited}}$ ; initialise tree with root node $x_{init}$
2: Find lowest-code node $z$ in $\text{V}_{\text{open}}$
3: **for all** neighbors $x$ of $z$ in $\text{V}_{\text{unvisited}}$ **do**
4: Find neighbor nodes $y$ in $\text{V}_{\text{open}}$
5: Find locally-optimal one-step connection to x from among nodes $y$
6: If that connection is collision-free, add edge to tree of paths
7: Remove successfully connected node $x$ from $\text{V}_{\text{unvisited}}$ and add them to $\text{V}_{\text{open}}$
8: Remove $z$ from $\text{V}_{\text{open}}$ and add it to $\text{V}_{\text{closed}}$
9: Repeat until either :

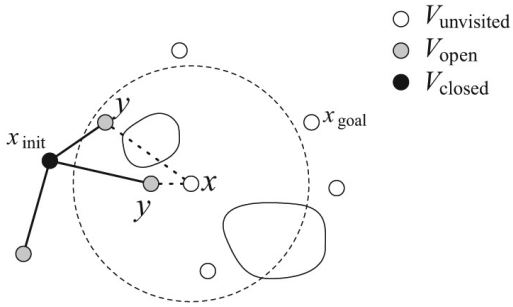
- $\text{V}_{\text{open}}$ is empty $\Rightarrow$ report failure
- Lowest-code node z in $\text{V}_{\text{open}}$ is in $\mathcal{H}_{\text{goal}}$ $\Rightarrow$ return unique path to $z$ and report success

The input of the algorithm is first of all the path planning problem definition, *i.e.*, $(\mathcal{H}_{\text{free}}, x_{\text{init}}, \mathcal{H}_{\text{goal}})$. Where $\mathcal{H}_{\text{free}}$ is the obstacle-free space, $x_{\text{init}}$ the initial condition and $\mathcal{H}_{\text{goal}}$ the goal region. The other input is a sample set V comprised of $x_{\text{init}}$ and n samples in $\mathcal{H}_{\text{free}}$.

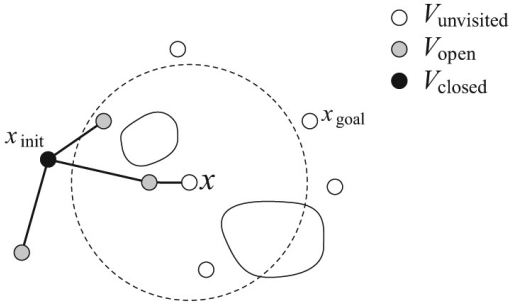
We refer to samples added to the tree of paths as nodes. Two samples $(u, v) \in V^2$ are considered *neighbors*, and hence connectable, if their Euclidean distance is below a given bound, referred to as the *connection radius*. The algorithm makes use of a partition of V into three subsets, namely $\text{V}_{\text{unvisited}}$, $\text{V}_{\text{open}}$ and $\text{V}_{\text{closed}}$. The set $\text{V}_{\text{unvisited}}$ consists of all of the samples that have not yet been considered for addition to the incrementally grown tree of paths. The set $\text{V}_{\text{open}}$ contains samples that are currently active in the sense that they have already been added to the tree (*i.e.*, a collision-free path from $x_{\text{init}}$ with a given cost-to-arrive has been found) and are candidates for further connections to samples in $\text{V}_{\text{unvisited}}$. The set $\text{V}_{\text{closed}}$ contains samples that have been added to the tree and are no longer considered for any new connections. Intuitively, these samples are not near enough to the edge of the expanding tree to actually have any new connections made with $\text{V}_{\text{unvisited}}$. Thus, this algorithm performs graph search and graph construction *concurrently*. An iteration of this algorithm is shown below in Figure 1.



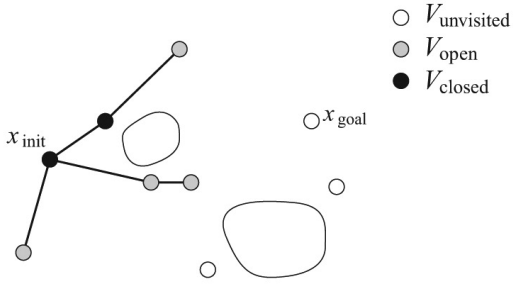

(a) Lines 2–3: FMT* selects the lowest-cost node $z$ from set $V_{\text{open}}$ and finds its neighbors within $V_{\text{unvisited}}$.




(b) Lines 4–5: given a neighboring node $x$, FMT* finds the neighbors of $x$ within $V_{\text{open}}$ and searches for a locally optimal one-step connection. Note that paths intersecting obstacles are also lazily considered.




(c) Line6:FMT* selects the locally optimal one-step connection to $x$ ignoring obstacles, and adds that connection to the tree if it is collision-free.

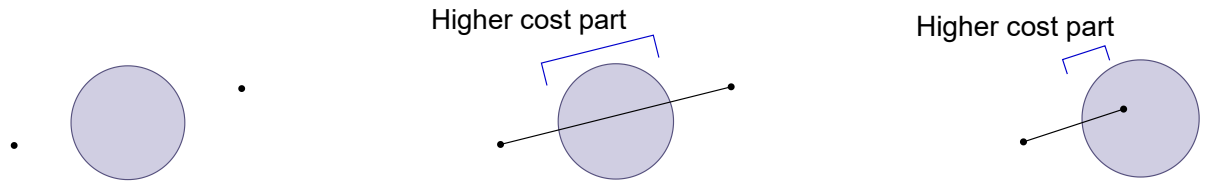



(d) Lines 7–8: After all neighbors of $z$ in $V_{\text{unvisited}}$ have been explored, FMT* adds successfully connected nodes to$V_{\text{open}}$, places $z$ in $V_{\text{closed}}$, and moves to the next iteration.

Figure 1: An iteration of the FMT* algorithm. FMT* *lazily* and *concurrently* performs graph construction and graph search. Line references are with respect to Fast Marching Tree Algorithm (FMT*). In panel (b), node $z$ is re-labeled as node y since it is one of the neighbors of node $x$. Pictures from [8].

However, this algorithm as implemented returns a path avoiding all the obstacles. Thus, if we consider the obstacles to be areas of contrails formation, the trajectory will never fly through these regions. This way all contrails will be avoided, drastically reducing the non-$CO_2$ radiative forcing made during the flight. But avoidance of contrail formation through re-routing may incur a fuel penalty and therefore additional $CO_2$ emissions. So instead of avoiding the obstacles, a cost associated with crossing them is added. Thus some changes modifications are necessary. In the Fast Marching Tree Algorithm (FMT*), at the line 6, the algorithm will therefore not verify if the connection is collision-free, the edge will automatically be added to the tree, and a cost related to the time of travel in the obstacle will be added. Moreover, the set V required at the start of the algorithm needs to contain samples in $\mathcal{H}_{\text{obstacle}}$ – the

obstacle region. Thus, the trajectory can travel trough the obstacles which will avoid making detours that would otherwise increase $CO_2$ emissions.

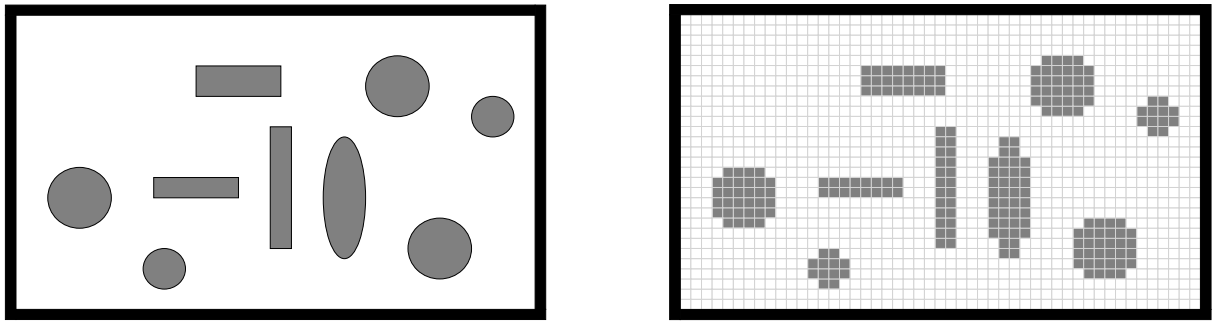



(a) (b) (c)

Figure 2: Examples showing the difference between passing or not in obstacles.

The Figure 2a illustrates the initial method where all obstacles are avoided, resulting in no connection between the two samples. In the other figures 2b and 2c, the crossing of an obstacle is allowed by an additional cost linked to the time spent in the obstacle. In 2b both nodes are outside the obstacle and in 2c, one node is inside an obstacle.

In this article, the obstacles will be discretized. Indeed, the map, describing the obstacles will be transformed into a grid. Each cell of that grid will be considered an obstacle if its center is in one (see Figure 3).



(a) Initial environment (b) Grid from the discretization of the obstacles

Figure 3: Example of a grid and the effect of discretization

To have a better result, the number of lines and columns of the grid must be high to reduce the obstacles approximation during discretization.

This discretization allows to work only on square shapes. Thus collisions with obstacles is easier. Indeed, it facilitates the search for the intersection point of a segment between two points and an obstacle.

Finally, the map is sampled. One can simply use an uniform sampling but it is interesting to sample more around obstacles which will lead to better circumvent them. And thus minimize both the $CO_2$ and non-$CO_2$ emissions.

First of all, two types of sample are defined : a sample inside an obstacle and a sample in the obstacle-free space. The proportion between these two types of samples is considered the same as the proportion of obstacles, which can be computed using the grid.

Then, to sample more around the boundary of an obstacle, it is necessary to know the shapes of the obstacles – these shapes will be limited to three in this paper : rectangle, circle and ellipse – as well as knowledge of their characteristics – length and width ; radius and center ; semi-major, semi-minor axes and center respectively. The grid does not contain these information. Outside knowledge is then needed to know how the obstacles have been defined in the map. Using these information, sampling is done from a mixture of distributions. Each distribution provides random points in an obstacle. And the probability of distributing points near the border of the obstacle if higher than near the centre. In addition, the probability associated with a distribution is related to the area of the represented obstacle.

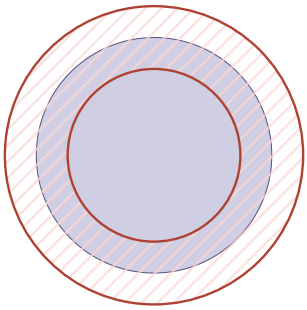


Figure 4: Example of distribution for a circle.

In Figure 4, an example of distribution for a circle is illustrated. In this distribution, the probability of sampling within the circle boundary (represented by the red hatched area) is higher than sampling near the center of the circle.

# 5 Results

This part aims to present the results with different parameters.

First of all, to use the algorithm, the environment is sampled. Using the same map used in Figure 3 the following sampling is obtained :

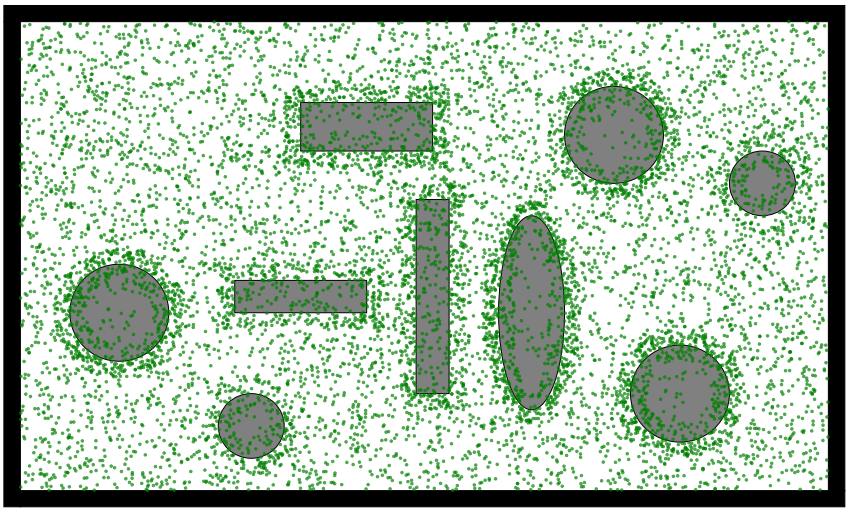
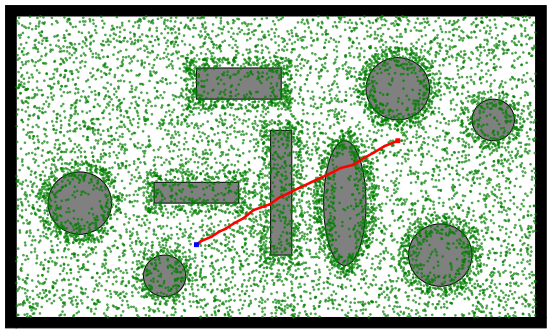
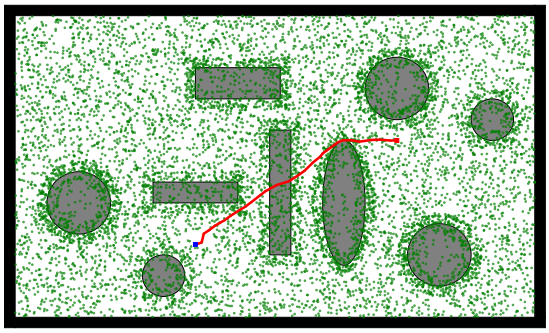


Figure 5: Sampling obtained with 10 000 samples. The samples are represented by green dots.

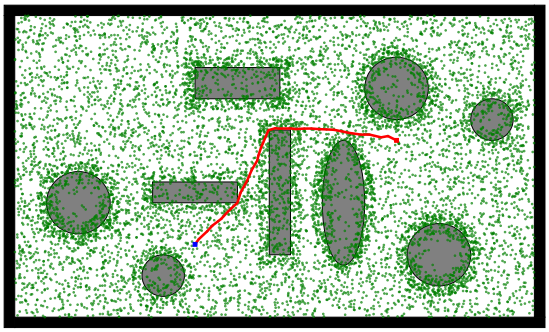
Then by applying the FMT* algorithm on an equivalent sampling, different paths can be found for different values of the cost related to the crossing of an obstacle, noted $C_r$ in Equation 1.



(a) $C_r = 0$



(b) $C_r = 1$



(c) $C_r = 2$

Figure 6: Multiple paths for various cost associated with obstacles. The origin point is a blue dot. The destination is a red dot. The path is the red line.

In Figure 6 the algorithm is applied on the same environment with the same number of samples. The sampling is different in all figures because it is done probabilistically. Each figure was computed using different cost associated with crossing an obstacle. In 6a, this cost – $C_r$ – was null, which means no penalty was added when travelling through an obstacle. So the returned path is a straight line. In 6b, $C_r$ is equal to 1. In that case, since in our resolution we set $C_t = 1$ and $r(x, y) = 1$ in an obstacle, then $C_r = 1$ will double the time spent in an obstacle. Likewise, when $C_r = 2$, the time spent in an obstacle is tripled.

Being able to change the cost linked to crossing an obstacle allows to obtain a solution path avoiding more or less obstacles. Thus the choice of this coefficient is a key-point. It is a balance between dodging all the obstacles or letting go through them. In our case, it is a balance between reducing the formation of contrails – which reduces non-$CO_2$ emissions – and minimizing the lengthening – related to fuel consumption, thus minimizing $CO_2$ emissions.

For the previous examples, the distance used to calculate the cost between two nodes in the graph – the cost being the travel time or simply the distance if wind is not considered – was the Euclidean distance. However the great-circle distance is more appropriate to compute an airplane trajectory. Thus, by implementing this distance, a trajectory respecting the great-circle distance can be computed, as shown in Figure 7.

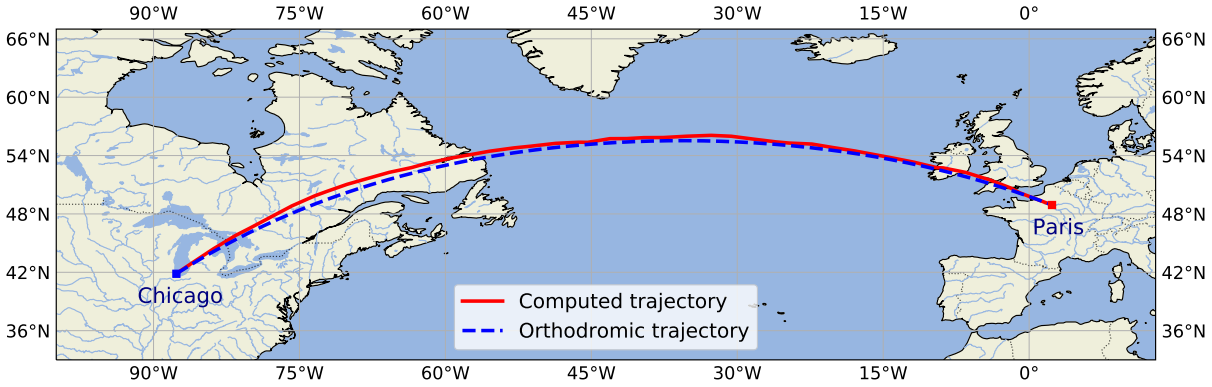
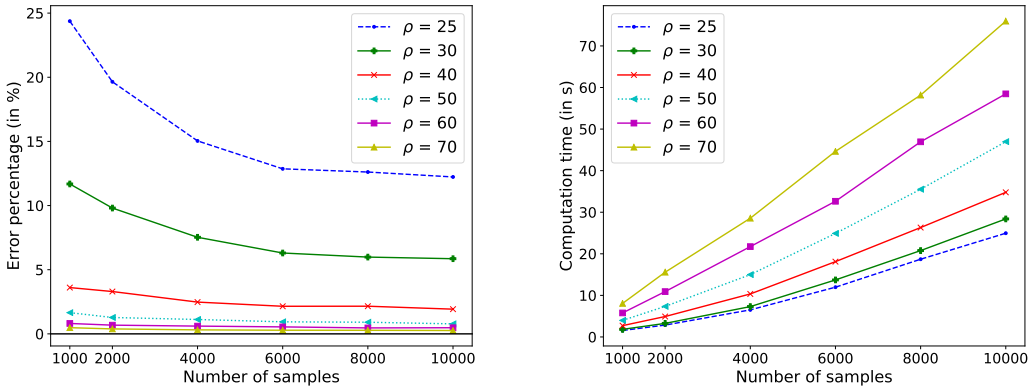



Figure 7: Comparison of the computed trajectory and the orthodromic trajectory with longitude on abscissa and latitude on ordinate.

Note that the great circle distance is used for the computation of edge cost. However, it is still the Euclidean distance which is used for identifying neighbors of a node in the FMT* algorithm.

Since the distance of the great circle path between two points can be calculated, we can then compare the computed trajectory with the real trajectory. Thus, different results were compared by varying two parameters such as the number of samples and the connection radius. Indeed, the higher the number of samples, the more accurately the space will be sampled, thus leading to a better approximation of the path. And more important the connection radius is, more neighbors a specific node will have, which will lead to more paths to optimize. But the quality of the calculated path comes with a higher complexity. More samples, lead to more calculations and higher connection radius means more neighbors resulting in more computation.

In the following we will use $n$ as the number of samples and $r_n$ as the connection radius. Where $r_n$ is defined by : $r_n = \rho\sqrt{\frac{\log(n)}{n}}$. In the following $\rho$ will be used to vary $r_n$, the connection radius.




(a) (b)

Figure 8: Two graphs comparing the number of samples and the connection radius on a given environment.

The data displayed are averages over several results to account for the fact that the sampling is probabilistic. In Figure 8, the first graph 8a, represents the error percentage of the returned path using different sample numbers and connection radii. We note that for each number of samples, the percentage of error is below 1% when $\rho$ is greater than 60. The other graph, 8b, shows the computation time according to the different parameters. We notice that when $n = 1000$ with $\rho = 60$, the computation time is about 6 seconds. And when $n = 10000$ with the same $\rho$, the computation time is up to 58 seconds. With $n = 1000$ and $\rho = 60$ the percentage of error is approximately 0.8%. With $n = 10000$, the percentage of error is about 0.5%. So between $n = 1000$ and $n = 10000$ the precision was multiplied by

1.6 but the computation time was multiplied by 9.7. Thus, in this example, it is more efficient to use a relative small number of samples.

However, in order to obtain quantitative results on known resolved instances, no obstacles are considered here. Having obstacles all over the map can lead to the need for more samples. These comparisons between the number of samples and the connection radius are not at all general and depend strongly on the environment.

# 6 Conclusion

To conclude, through this paper, after presenting the methods already used to generate aircraft trajectories considering obstacles, a new method has been presented using an algorithm based on probabilistic sampling. This way, by sampling the environment according to the obstacles and applying the FMT* algorithm on this sampling, 2D trajectories are generated. These trajectories can avoid all the obstacles or can allow to pass through them to avoid making too much a detour. The results obtained with different parameters such as the number of samples and the connection radius were compared. But the dependency of these parameters on the accuracy and computation time is highly dependent on the environment. However, in order to sample in accordance with the obstacles, outside knowledge is necessary. It could be interesting to be able to find all the characteristics of an obstacle using only a bitmap, as shown in Figure 3b. In addition, the trajectory could be smoothed to make the turns flyable. It would also be relevant to implement a method to exploit wind data, for example by using interpolation. Finally, using real data, such as real area of contrails appearances as obstacles and real wind data, would make the problem more real-accurate.

# References


[1] David S Lee, DW Fahey, Agnieszka Skowron, MR Allen, Ulrike Burkhardt, Q Chen, SJ Doherty, S Freeman, PM Forster, J Fuglestvedt, et al. The contribution of global aviation to anthropogenic climate forcing for 2000 to 2018. *Atmospheric Environment*, 244:117834, 2021.

[2] Ulrich Schumann. Formation, properties and climatic effects of contrails. *Comptes Rendus Physique*, 6(4-5):549–565, 2005.

[3] Nicola Stuber, Piers Forster, Gaby Rädel, and Keith Shine. The importance of the diurnal and annual cycle of air traffic for contrail radiative forcing. *Nature*, 441(7095):864–867, 2006.

[4] Abhijit Chakravarty. Four-dimensional fuel-optimal guidance in the presence of winds. *Journal of Guidance, Control, and Dynamics*, 8(1):16–22, 1985.

[5] Banavar Sridhar, Hok K Ng, and Neil Y Chen. Aircraft trajectory optimization and contrails avoidance in the presence of winds. *Journal of Guidance, Control, and Dynamics*, 34(5):1577–1584, 2011.

[6] Karim Legrand, Stephane Puechmorel, Daniel Delahaye, and Yao Zhu. Robust aircraft optimal trajectory in the presence of wind. *IEEE Aerospace and Electronic Systems Magazine*, 33(11):30–38, 2018.

[7] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524, 1968.

[8] Lucas Janson, Edward Schmerling, Ashley Clark, and Marco Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International journal of robotics research*, 34(7):883–921, 2015.

[9] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[10] Steven M LaValle et al. Rapidly-exploring random trees: A new tool for path planning. 1998.