Published in Image Processing On Line on 2022–07–06.
Submitted on 2021–10–12, accepted on 2022–04–22.
ISSN 2105–1232 © 2022 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials, software, datasets and online demo at
https://doi.org/10.5201/ipol.2022.384

Single Date Wind Turbine Detection on Sentinel-2 Optical Images

N. Mandroux, T. Dagobert, S. Drouyer, R. Grompone von Gioi

Centre Borelli, ENS Paris-Saclay, France

Communicated by Gregory Randall *Demo edited by* Nicolas Mandroux

Abstract

ESA's Sentinel-2 satellites have been in orbit for six years, acquiring massive amounts of data all over the world. They are a formidable tool for mass detection as they are freely available and cover the entire world. Given their future role in the energetic transition and their spread over countries or continents, wind turbines are natural candidates for such studies. We detail the implementation of a single-date automatic wind turbine detector for low resolution optical satellite images based on an a contrario approach, exploiting the geometry of wind turbines' shadows and hubs.

Source Code

The reviewed reference source code for this article is available from the web page of this article. The algorithm is implemented in Python3. Compilation and usage instructions are included in the README.txt file of the archive.

Keywords: remote sensing; wind turbine detection; a contrario method; NFA

1 Introduction

A wind turbine is a device that converts the wind's kinetic energy into electricity. In its 2019 report, the American Wind Energy Association [3] identifies wind as America's top renewable, no-emissions energy source. As a result, wind turbines may have a key role to play in the energetic transition coming within the next decades. Being able to detect automatically their locations and thus their number and installed capacity could prove useful in managing the electrical network and planning wind power plant projects. It could also allow for blade movement detection: once we know precisely the wind turbine's position, it may be possible to assess whether it is rotating or not. Despite the usefulness of knowing their locations for meteorological [5], environmental [4], radar analysis [11], or electricity management reasons [7, 1, 2], there are very few works addressing the problem in the literature, the majority of which use neural networks and require high resolution images (at least 2m/px).

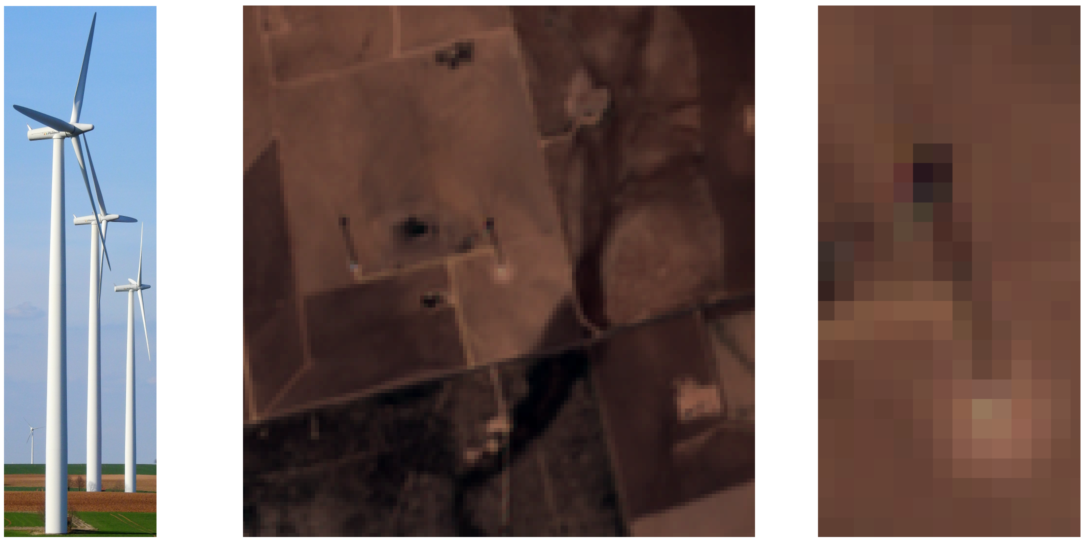

Figure 1: Pictures of wind turbines. (a) Seen from the ground. If we ignore the blades, we can see its T-shape: we call tower the vertical bar and hub the top horizontal bar. (b) Scene of a wind farm acquired with Sentinel-2. (c) Detail of picture (b) on the central wind turbine. We can spot it thanks to its dark shadow and its bright hub. This example is one of the most visible cases we can get with Sentinel-2 images given its low resolution.

For the objective of monitoring the electricity production of whole countries, huge areas must be acquired and scanned. Therefore, using aerial, drone or most high resolution commercial satellite images is too expensive. Images used must have an adequate resolution for detecting wind turbines – their hubs' length are around 10 meters – but still be inexpensive enough so that the cost of monitoring doesn't overwhelm the value of the extracted pieces of information. The Sentinel-2 constellation, launched by the European Spatial Agency (ESA) in 2015 and 2017, provides free optical images of the whole world with a revisit time of 2 to 3 days and a 10 meters resolution. Images produced by Sentinel-2 seem therefore appropriate for this task. As a consequence, the algorithm described here, based on the a contrario approach [6], was especially designed to work on low resolution images (10m/px). This work expands and provides further details on a wind turbine detection method initially proposed in [9].

The rest of this paper is organized as follows. In Section 2, we explain the theoretical framework. We detail the algorithm's functioning in Section 3; then in Section 4 we discuss the results and the parameters. Finally, Section 5 concludes the paper.

2 Proposed Method

The structure of a wind turbine is T-shaped, with a vertical tower at the top of which stands the hub, where the rotating blades connect (see Figure 1a). The height of a very large majority of towers is around 80 meters [8] (see Figure 2). For heat reasons they are all painted in white (see Figure 1a). As we can see in Figures 1b and 1c, a wind turbine's footprint is composed of two parts: the dark shadow induced by the tower, and the bright hub. Since this kind of footprint is shared by every wind turbine in the world, we can use this knowledge coupled with the known positions of the satellite and the sun (and a hypothesis on the height of the tower) to build a detector. This detector is the fusion of a shadow detector and a hub detector. To use most of that knowledge and quantify the degree of certainty of our detections, we choose the a contrario framework [6]. Only the B02 spectral band is used: the blue one, to avoid blurring effects due to rotating blades.

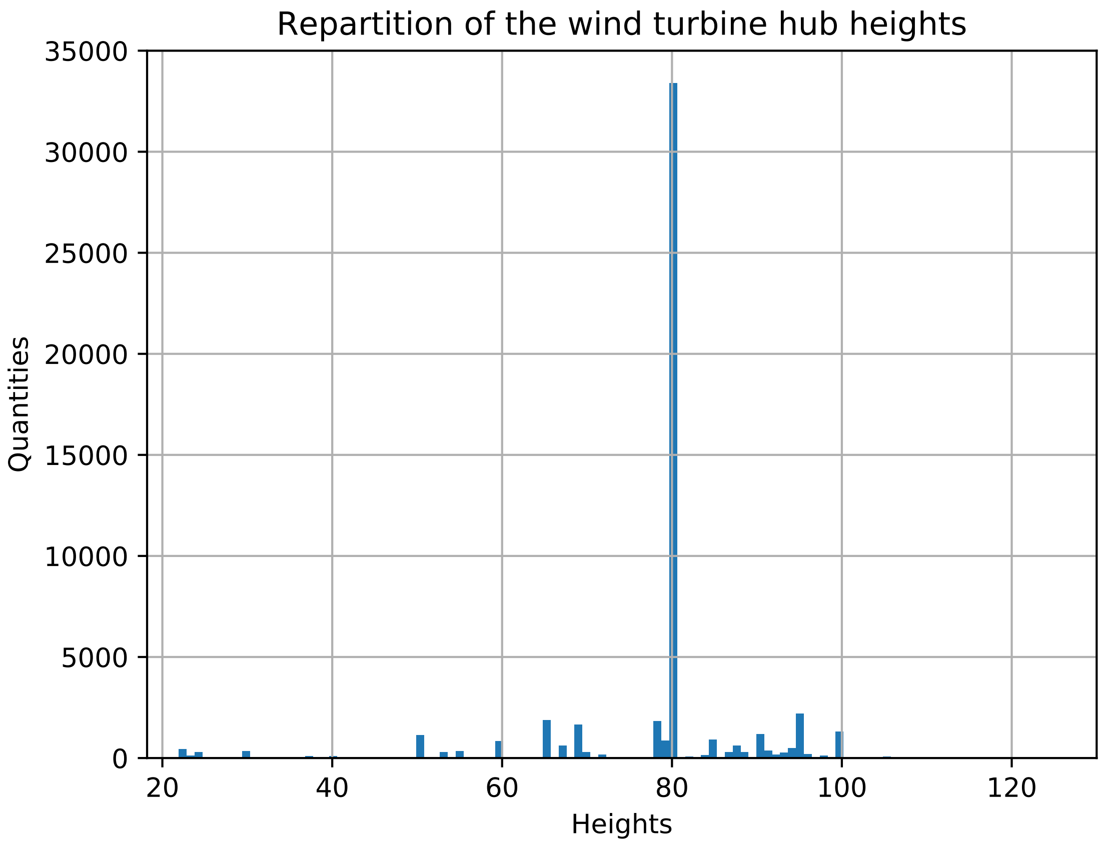

Figure 2: Height histogram of US wind turbines in meters [8].

2.1 A Contrario Framework

The *a contrario* approach is inspired by visual perception [6]. The human eye is attracted by patterns which differ from their background. If the background can be probabilistically modeled, one can question this model by observing the structure of some of the pixels present in the image, and then quantify how far it diverges from the model. If it diverges sufficiently, it means there is a structure to be detected. Informally, an observed structure is meaningful only when the relation between its parts is too regular to be the result of an accidental arrangements of independent parts. This leads to a statistical framework used to set detection thresholds automatically in order to control the number of false detections.

Following the a contrario methodology [6], we define the *Number of False Alarms* (NFA) of an event e with an observed value $s(e)$ as

$$\mathrm{NFA}(e) = N_{tests} \cdot \mathbf{P}\Big[S_{H_0}(e) \geq s(e)\Big], \tag{1}$$

where the right hand term is the probability of obtaining, in the background model H_0, a value $S_{H_0}(e)$ larger or equal to the observed one $s(e)$; N_{tests} is the total number of tests performed. The smaller the NFA, the more unlikely the event e is to be observed by chance in the background model H_0; thus, the more meaningful. The a contrario approach prescribes accepting as valid detections the candidates with NFA $< \varepsilon$ for a predefined threshold value ε. It can be shown [6] that under H_0, the expected number of tests with NFA $< \varepsilon$ is bounded by ε. As a result, ε gives an *a priori* estimate of the mean number of false detections under H_0.

In what follows, we consider a slightly different but equivalent formulation. Let $t^{NFA} = -\log_{10}(\varepsilon)$.

Now, detections will be declared when $-\log_{10}(\mathrm{NFA}) > t^{NFA}$, which is equivalent to NFA $< \varepsilon$. This change of variables serves two purposes. Firstly, it improves the visualization: rather than having white NFA maps with black pixels indicating significant detections, we have the opposite. White pixels are easier to spot on a black background than the contrary. Secondly, it converts very small ε thresholds into not-too-high t^{NFA} ones. For example, if $\varepsilon = 10^{-6}$, then $t^{NFA} = 6$.

2.2 Pixel Values Comparisons

Let I be an image. Consider each of its pixels u_i to be a continuous random variable, and suppose these variables are independent and identically distributed (i.i.d.). This is the background random model H_0 we will use for our a contrario setting.

The independence assumption here is, as always, questionable. Indeed, the necessary band limited spectral response of the optical system, and the image processing pipeline that produces the image, both result in some kind of correlation of the values of neighbor pixels. As a consequence, the estimate of NFA may not be fully accurate and some manually tuning of its threshold may be needed. There are three ways of dealing with this difficulty. A first approach is to adapt the method in order to satisfy the independence hypothesis; this can be done, for example, by increasing the sampling rate δ. The fatal flaw of this idea is that we already lack samples when working with low resolution images; we cannot afford losing more information. A second approach is to modify the background model to take into account the dependency; this can be done, for instance, using first and second order statistics [10]. While this is a valid and interesting possibility, it results in a more complex method and is beyond the scope of this article; future works may focus on this alternative. The third and last approach is to assume that neighbor pixels are only slightly correlated. If the model is good enough, useful results may be obtained in spite of the inaccurate hypothesis (cf. "all models are wrong, but some are useful"); this is the approach taken here.

Under these assumptions,

$$\mathbf{P}(u_{i_1} \leq u_{i_2}) = \mathbf{P}(u_{i_1} \geq u_{i_2}) = \frac{1}{2},$$

for $i_1 \neq i_2$; and more generally, for any $l \in \mathbb{N}$

$$\mathbf{P}\left(\bigcap_{k=1}^{l}(u_{i_0} \leq u_{i_k})\right) = \mathbf{P}\left(\bigcap_{k=1}^{l}(u_{i_0} \geq u_{i_k})\right) = \frac{1}{2^l}. \tag{2}$$

We should see l as the number of neighboring pixels to which the considered pixel value is compared.

Let $X_{i,j} = \mathbb{1}_{u_i \leq u_j}$ be a random variable, where $\mathbb{1}$ is the indicator function; thus $X_{i,j} = 1$ if $u_i \leq u_j$ and $X_{i,j} = 0$ otherwise. Symmetrically, let $Y_{i,j} = \mathbb{1}_{u_i \geq u_j}$. Denoting B the Bernoulli distribution, we know from Equation (2) that $X_{i,j} \sim \mathrm{B}(\frac{1}{2})$ and $Y_{i,j} \sim \mathrm{B}(\frac{1}{2})$.

Let $J = \{j_1, j_2, \ldots, j_l\}$ with j_k all different. We define the random variable

$$X_{i,J} = \prod_{k=1}^{l} X_{i,j_k}. \tag{3}$$

We should see J as the locations of the neighboring pixels to which the considered pixel value is compared.

By independence, $X_{i,J} \sim \mathrm{B}(\frac{1}{2^l})$. Symmetrically, we can define $Y_{i,J}$; $Y_{i,J} \sim \mathrm{B}(\frac{1}{2^l})$.

Finally, let $X = \{X_{i_1,J_1}, \ldots, X_{i_n,J_n}\}$, with $i_1 \neq \ldots \neq i_n$, $\#J_1 = \cdots = \#J_n = l$, and $\forall 1 \leq i,j \leq n$, $J_i \cap J_j = \emptyset$. We define $S_X = \sum_{s=1}^{n} X_{i_s,J_s}$. By independence and identical distribution,

$$S_X \sim \mathrm{Bin}\left(n, \frac{1}{2^l}\right), \tag{4}$$

where Bin is the binomial distribution. We should see n as the number of sampled pixels.

As a result of what we have stated, we can easily compute the probability for a group of specific pixels to have larger values than its neighbors. Intuitively, we will be surprised to find a group X of n random variables such that S_X is large (i.e., large relative to what would be expected under H_0).

2.3 Shadow Detection

For each pixel $u_i \in I$, we want to compute if it is a plausible candidate for the beginning of the wind turbine's shadow. Given the sun's altitude θ_{sun} and azimuth ϕ_{sun}, and the tower's height h, the coordinates (x, y) at the end of the shadow are

$$(x, y) = \left(-\frac{h \sin(\phi_{sun})}{|\tan(\theta_{sun})|}, -\frac{h \cos(\phi_{sun})}{|\tan(\theta_{sun})|}\right). \tag{5}$$

Angles θ_{sun} and ϕ_{sun} are usually provided on the satellite's metadata, allowing to compute the footprint of the shadow, and sample it. The number of obtained samples is noted n^{sh}. The sampling is detailed in Section 3.1.

For each of these samples u_{i_k}, $1 \leq k \leq n^{sh}$, we consider the two neighbors perpendicular to the shadow's direction: $J_{i_k} = \{u_{i_{k,1}}, u_{i_{k,2}}\}$, see Figure 3, middle. We consider

$$s_i^{sh} = \sum_{k=1}^{n^{sh}} X_{i_k, i_{k,1}} \times X_{i_k, i_{k,2}}, \tag{6}$$

where s_i^{sh} takes a value between 1 and n^{sh}, the larger the value the better the agreement with a shadow being observed at pixel u_i. Under the random assumptions H_0, and using Equation (4), s_i^{sh} becomes the random variable $S_i^{sh} \sim \text{Bin}(n^{sh}, \frac{1}{4})$.

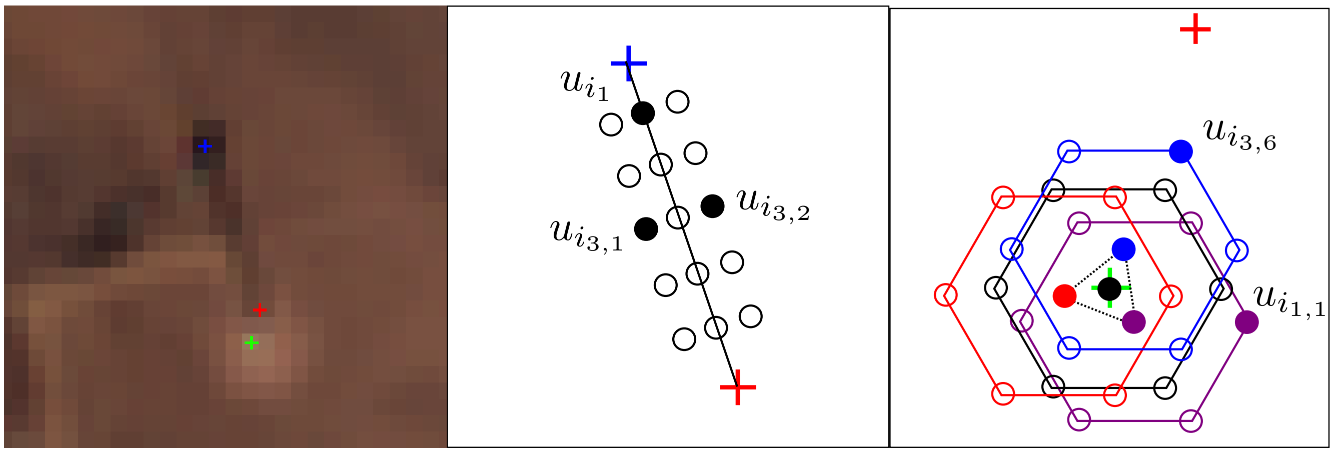

Figure 3: Left, a wind turbine view from a Sentinel-2 image, with its hub shadow (blue cross), pylon foot (red cross) and hub (green cross). Middle, zoomed in, the sampling along the pylon shadow where $n^{sh} = 5$. Right, zoomed in, the sampling around the hub where $n^{hub} = 4$.

2.4 Hub Detection

For each pixel $u_i \in I$, we want to compute if it is a plausible candidate for the bottom of the turbine's tower. Similarly as before, given the satellite's altitude θ_{sat} and azimuth ϕ_{sat}, and the tower's height h, the coordinates (x, y) at the end of the hub are

$$(x, y) = \left(-\frac{h \sin(\phi_{sat})}{|\tan(\theta_{sat})|}, -\frac{h \cos(\phi_{sat})}{|\tan(\theta_{sat})|}\right). \tag{7}$$

Given the satellite's metadata, one can compute the position of the hub (Figure 3 green cross) and sample it. The sampling is detailed in Section 3.2. Notice that the apparent hub is a bit shifted relative to the pylon foot because the satellite is not necessary at the wind turbine zenith; the more the wind turbine is far from the nadir point of view, the more it appears slanted. For each of these samples u_{i_k}, $1 \leq k \leq n^{hub}$, we consider the six neighbors which are the vertices of a hexagon centered in u_{i_k}: $J_{i_k} = \{u_{i_{k,1}}, \dots, u_{i_{k,6}}\}$, see Figure 3, right. We consider

$$s_i^{hub} = \sum_{k=1}^{n^{hub}} \prod_{l=1}^{6} Y_{i_k, i_{k,l}}, \tag{8}$$

where s_i^{hub} takes a value between 1 and n^{hub}, the larger the value the better the agreement with a hub being observed at pixel u_i. Under the random assumptions H_0, and using Equation (4), s_i^{hub} becomes the random variable $S_i^{hub} \sim \text{Bin}(n^{hub}, \frac{1}{2^6})$.

2.5 Shadow and Hub Aggregation

Once s_i^{hub} and s_i^{sh} are computed, we need to aggregate those two pieces of information to get an indicator of detection for the whole wind turbine. Let $s_i = s_i^{hub} + s_i^{sh}$. Since we know the distributions of S_i^{hub} and S_i^{sh}, we know the distribution of their sum $S_i \sim \text{Bin}(n^{sh}, \frac{1}{4}) + \text{Bin}(n^{hub}, \frac{1}{2^6})$. To facilitate the calculation, we approximate the sum of binomials in a new one: $\text{Bin}(n^{sh}, \frac{1}{2}) + \text{Bin}(n^{hub}, \frac{1}{2^6}) \approx \text{Bin}(n, p_w)$, with $n = n^{sh} + n^{hub}$ and p_w is the weighted average of the success probabilities of the two binomials: $p_w = \dfrac{n^{sh} \times \frac{1}{4} + n^{hub} \times \frac{1}{2^6}}{n^{sh} + n^{hub}}$. As a result we can compute the probability of false alarm $\mathbf{P}(S_i \geq s_i)$. Finally, the number of false alarms is $\text{NFA}_i = N_{tests} \times \mathbf{P}(S_i \geq s_i)$, which corresponds to the expected number of false alarms under H_0 when N_{tests} trials are made (here, the number of pixels in the image).

2.6 Tightening Tests and Empirical Probabilities

We have presented the theoretical framework of the method. Yet, we could dramatically improve the performances by tweaking the process a bit. The two ideas are to tighten the tests and then to replace the former theoretical probabilities by empirical ones.

Rather than wanting for a shadow-sampled pixel to be just darker than its two neighbors, we ask it to be darker than its two neighbors minus a threshold t^{shadow}. As a result, the probability for a randomly chosen pixel to pass the new test is not the theoretical $\frac{1}{2^2}$ anymore, but some unknown value which we estimate empirically. We note it $p_{emp}^{t^{shadow}}$. Similarly, for a hub-sampled pixel, we want it to be brighter than its six neighbors plus a threshold. We define $p_{emp}^{t^{hub}}$ the probability for a random pixel to be brighter than six neighbors plus threshold t^{hub}.

We replace $X_{i,j}$ by $\widehat{X}_{i,j}^t = \mathbb{1}(u_i \leq u_j - t)$ and $Y_{i,j}$ by $\widehat{Y}_{i,j}^t = \mathbb{1}(u_i \geq u_j + t)$. With these new notations, we can define the new hub and shadow scores

$$\widehat{s}_i^{hub} = \sum_{k=1}^{n^{hub}} \prod_{l=1}^{6} \widehat{Y}_{i_k, i_{k,l}}^{t^{hub}}, \qquad \widehat{s}_i^{shadow} = \sum_{k=1}^{n^{sh}} \widehat{X}_{i_k, i_{k,1}}^{t^{shadow}} \times \widehat{X}_{i_k, i_{k,2}}^{t^{shadow}}. \tag{9}$$

Finally, under the assumption H_0, $\widehat{s}_i^{sh}$ becomes the random variable $\widehat{S}_i^{sh} \sim \text{Bin}(n^{sh}, p_{emp}^{t^{shadow}})$, and $\widehat{s}_i^{hub}$ becomes the random variable $\widehat{S}_i^{hub} \sim \text{Bin}(n^{hub}, p_{emp}^{t^{hub}})$. The new scores are less likely to be high, and can lead to better significance.

The empirical shadow (resp. hub) probability is computed directly on the image: we count the number of pixels which satisfy the shadow (resp. hub) test, and divide by the total number of pixels.

3 Algorithms

The programming of our NFA computation is divided into three algorithms. Algorithm 1 gives the subpixel values of shadow samples and their neighbors, Algorithm 2 does the equivalent for the hub, and Algorithm 3 is the main algorithm which makes all the computations using the previous outcomes.

3.1 Shadow Sampling

We will now describe the procedure for sampling the shadow, as detailed in Algorithm 1. If there is a h-meter-tall wind turbine on a pixel i of coordinates $coords_i$, the induced shadow is a segment beginning at i and ending at a subpixel location. We name the latter coordinates $coords_i^{top}$. To determine these coordinates, we need to know the direction of the shadow and its length in pixels. The required information is usually available in the metadata. Here, the crucial piece of information is the position of the sun in the sky, that is to say the sun's azimuth ϕ_{sun} and the sun's altitude θ_{sun}. These are two angles: the first one gives the direction, the second one induces the stretch of the shadow. Given this, we can compute the metric shift (x, y) between the bottom and the top of the shadow using Equation (5). We then need to convert this metric shift into a subpixel shift; this is done simply by dividing it by the satellite's resolution, as can see in line 2 of the pseudo-code. Notice that only the shift needs to be divided by the resolution, as the coordinates are already in image units.

Now we need to sample the segment $[coords_i, coords_i^{top}]$. To do that, we just have to choose a sampling rate δ. We sample every δ meters, or equivalently $pix_\delta = \delta/resolution$ pixels. The whole list of samples is named $COORDS_i^{shadow}$. Finally, we can determine the two neighbors' positions for each sample. They are sampled in the orthogonal direction of the shadow segment, and distant of it by e_o meters (equivalently $e_o/resolution$ pixels, see Figure 3, middle). Table 1 gives the default values of the parameters of this algorithm.

Now we have managed to get all interesting pixels locations and, since most of them are in fact subpixel locations, we get their values by bilinear interpolation. In the code, the process is a little different: here, to write Algorithm 3 as light as possible, the computation of the pixels' and neighbors' values is made in the auxiliary algorithms.

3.2 Hub Sampling

We will now describe the procedure for sampling the hub, as detailed in Algorithm 2. Due to the tilt, the footprint of the hub is not necessarily at the same location of the base of the wind turbine. As we did before for the shadow, we have to calculate the shift between basis and top of the tower in the image, and then sample around the top. The shift is computed as detailed in Equation (7). We convert this metric shift into a subpixel shift by dividing it by the satellite's resolution, line 3 of the pseudo-code. Shifting away from the basis of the tower to the hub's footprint, we sample around it. The sampling shape is a regular $(n^{hub} - 1)$-sided polygon of side δ meters ($pix_\delta = \delta/resolution$ pixels), plus its center (the four red dots in Figure 3, right). The list of sample positions is named $COORDS_i^{hub}$ and contains thus n^{hub} coordinates. We then want to determine the positions of the samples' neighbors. Given a sampled pixel, we get 6 neighbors around it at e_h meters, regularly spaced. This forms a regular hexagon, see Figure 3, right. Table 1 gives the default values of the parameters of this algorithm.

Now we have managed to get all the interesting coordinates and, since most of them are subpixel locations, we get their values by bilinear interpolation. Here again, in the code, the process is a little

Algorithm 1: sample_shadow

```
   input     : $I$                                              Satellite image
   input     : $coords_i$                                       Tested pixel position
   input     : $\phi_{sun}, \theta_{sun}$                       Sun's zenith and azimuth
   parameter : $h$                                              Hypothetical height of wind turbine
   parameter : $r$                                              Resolution in meters
   parameter : $\delta$                                         Sampling rate in meters
   parameter : $e_o$                                            Neighbor distance in meters
   output    : $u_{i_j}, u_{i_{j,1}}, u_{i_{j,2}}$              Sampled shadow values and associated neighbors
 1 $shift = (x, y) \leftarrow (-\frac{h\sin(\phi_{sun})}{|\tan(\theta_{sun})|}, -\frac{h\cos(\phi_{sun})}{|\tan(\theta_{sun})|})$   Shift between bottom and top of shadow, Equation (5)
 2 $coords_i^{top} \leftarrow coords_i + shift/r$               Convert the shift from meters to pixels
 3 $COORDS_i \leftarrow$ Samples from $coords_i$ to $coords_i^{top}$ with a sampling rate of $\delta/r$.
 4 foreach $coords_{i_j}$ in $COORDS_i$ do
 5     $pixshift \leftarrow \frac{(-y,x)}{||shift||}\frac{e_o}{r}$     Shift between samples and neighbors, in pixels
 6     $coords_{i_{j,1}} \leftarrow coords_{i_j} + pixshift$
 7     $coords_{i_{j,2}} \leftarrow coords_{i_j} - pixshift$    Neighbors' coordinates computation
 8     $u_{i_j}, u_{i_{j,1}}, u_{i_{j,2}} \leftarrow$ bilinear interpolation of $I$ in $coords_{i_j}, coords_{i_{j,1}}, coords_{i_{j,2}}$ respectively.
 9 return $\{u_{i_j}, u_{i_{j,1}}, u_{i_{j,2}}\}$
```

different: to write Algorithm 3 as light as possible, the computation of the pixels' and neighbors' values is made in the auxiliary algorithms.

Algorithm 2: sample_hub

```
   input     : $I$                                              Satellite image
   input     : $coords_i$                                       Tested pixel position
   input     : $\phi_{sat}, \theta_{sat}$                       Satellite's zenith and azimuth
   parameter : $h$                                              Hypothetical height of wind turbine
   parameter : $r$                                              Resolution in meters
   parameter : $\delta$                                         Sampling rate in meters
   parameter : $e_h$                                            Neighbor distance in meters
   parameter : $n^{hub}$                Number of samples; number of sides of the sampling polygon minus 1
   output    : $u_{i_k}, u_{i_{k,l}}$                            Sampled hub values and associated neighbors
 1 $pix_\delta \leftarrow \delta/r$                              Sampling rate conversion: meters $\to$ pixels
 2 $shift \leftarrow (-\frac{h\sin(\phi_{sat})}{|\tan(\theta_{sat})|}, -\frac{h\cos(\phi_{sat})}{|\tan(\theta_{sat})|})$   Shift between bottom and top of tower, Equation (7)
 3 $coords_i^{top} \leftarrow coords_i + shift/r$               Shift converted from meters to pixels
 4 $COORDS_i \leftarrow$ Samples at $coords_i^{top}$ and at the vertices of a regular $n^{hub} - 1$-sided polygon
   with side $pix_\delta$ and center $coords_i^{top}$.
 5 foreach $coords_{i_k}$ in $COORDS_i$ do
 6     foreach $l$ in $\{1, \ldots, 6\}$ do
 7         $coords_{i_{k,l}} \leftarrow coords_{i_k} + \frac{e_h}{r}(\cos(\frac{2i\pi l}{6}), \sin(\frac{2i\pi l}{6}))$   Neighbors' coordinates computation
 8         $u_{i_{k,l}} \leftarrow$ bilinear interpolation of $I$ in $coords_{i_{k,l}}$
 9     $u_{i_k} \leftarrow$ bilinear interpolation of $I$ in $coords_{i_k}$
10 return $\{u_{i_k}, u_{i_{k,l}}\}$
```

3.3 Main Procedure

The main procedure is detailed in Algorithm 3. For each pixel in the image I, we want to calculate its wind turbine detection score, then convert it into a probability in order to finally get an NFA map. Given pixel i, we use Algorithms 1 and 2 to obtain the needed samples on the shadow and the hub's predicted footprint. We can then increment the score by comparing the sampled values with the sampled neighbors.

Given a shadow sample, the natural idea is to increment the score if it is darker than both neighbors. But the designed test is in fact tighter: a threshold t^{shadow} is added. We want a sampled shadow pixel to be darker than its neighbors minus the threshold, see line 11. The equivalent is done for the hub: a threshold t^{hub} is fixed. We want a sampled hub pixel to be brighter than each of its neighbors plus the threshold, see line 13. Since these thresholds are set, the theoretical probabilities are no longer valid; we need to compute empirical probabilities p^{shadow} and p^{hub}. It simply is the fraction of pixels in the image which satisfy the tightened shadow, respectively hub, test.

Once the scores and empirical probabilities are computed, we just have to wrap up: add the scores to get the likelihood score, compute the weighted average of the empirical probabilities and then the tail of the binomial distribution to convert likelihood into probability, and multiply by the total number of pixels examined to convert probability into NFA. Finally, threshold the $-\log_{10}(\text{NFA})$ by t^{NFA} to have the detection map.

Parameter	**Default value**
Height (h)	80m
Resolution (r)	10m
Sampling rate (δ)	10m
Distance neighbor-shadow (e_o)	15m
Distance neighbor-hub (e_h)	30m
Number of hub samples (n^{hub})	7
Number of hub neighbors per hub sample	6

Table 1: Default parameter values.

4 Results

Among all parameters set for the algorithm, the thresholds $t^{shadow}, t^{hub}, t^{NFA}$ and h are the most decisive. Since h is supposed fixed at $80m$, we analyze the three others. Figure 4 shows eight Sentinel-2 images and their detection map, with $(t^{shadow}, t^{hub}, t^{NFA}) = (25, 50, 1)$. Figure 5 shows a wider area of 1000 by 1000 pixels and its detection map.

On the eight small images, with this set of parameters, the algorithm manages to detect a decent amount of wind turbines, with some false detections. On the four bottom images, we can see a detection pattern in the form of a segment, centered on the hub. This segment has the same direction as the shadow; it comes from the overlapping of the computed shadows around the central hub. It should reduce or disappear if t^{NFA} was increased.

This detection pattern can be confusing. It does not come from noise, but from the detection of other parts of the shadow. We should aim at suppressing it, to get only one pixel detected for each wind turbine. A natural idea is to group these detection blobs, for instance by non-maximum suppression. We tried to do it and chose not to include it in the final algorithm: it led in numerous cases to a loss of true detections, because they got sucked by false or true ones when they were too close to each other.

```
Algorithm 3: Detection map computation
    input      : $I$                                   Satellite image
    input      : $\phi_{sun}, \theta_{sun}$            Sun's zenith and azimuth
    input      : $\phi_{sat}, \theta_{sat}$            Satellite's zenith and azimuth
    parameter  : height                                Hypothetical height
    parameter  : $t^{shadow}, t^{hub}, t^{NFA}$        Thresholds
    parameter  : $sh\_sampl, hub\_sampl$               Many parameters, detailed in appropriate algorithms
    output     : Detection map
 1  $p^{sh} \leftarrow$ compute $p_{emp}^{t^{shadow}}$   Empirical probability for a random pixel to be shadow
 2  $p^{hub} \leftarrow$ compute $p_{emp}^{t^{hub}}$     Empirical probability for a random pixel to be hub
 3  foreach pixel position $i$ do
 4      $tab_i^{sh} \leftarrow$ sample_shadow($I$,coords$_i$,$\phi_{sun}, \theta_{sun}$,height,$sh\_sampl$)   Algorithm 1
 5      $n^{sh} \leftarrow length(tab_i^{sh})$
 6      $tab_i^{hub} \leftarrow$ sample_hub($I$,coords$_i$,$\phi_{sat}, \theta_{sat}$,height,$hub\_sampl$)   Algorithm 2
 7      $n^{hub} \leftarrow length(tab_i^{hub})$
 8      $\widehat{s}_i^{sh} \leftarrow 0$
 9      $\widehat{s}_i^{hub} \leftarrow 0$
10      foreach $u_{i_j}^{sh} \in tab_i^{sh}$ do
11          $\widehat{s}_i^{sh} \longleftarrow \widehat{s}_i^{sh} + \mathbb{1}(u_{i_j}^{sh} < u_{i_{j,1}}^{sh} - t^{shadow}) \times \mathbb{1}(u_{i_j}^{sh} < u_{i_{j,2}}^{sh} - t^{shadow})$   Shadow score update
12      foreach $u_{i_k}^{hub} \in tab_i^{hub}$ do
13          $\widehat{s}_i^{hub} \longleftarrow \widehat{s}_i^{hub} + \prod_{l=1}^{6} \mathbb{1}(u_{i_k}^{hub} > u_{i_{k,l}}^{hub} + t^{hub})$   Hub score update
14      $s_i \leftarrow \widehat{s}_i^{sh} + \widehat{s}_i^{hub}$   Total score
15      $n \leftarrow n^{sh} + n^{hub}$
16      $p_w \leftarrow \frac{1}{n}(n^{sh}p^{sh} + n^{hub}p^{hub})$
17      $\mathbf{P}_i \leftarrow \mathbb{P}(\mathcal{B}(n, p_w) \geq s_i)$   Compute tail of Binomial distribution
18      NFA$_i \leftarrow \mathbf{P}_i \times$ #columns in $I$ $\times$ #rows in $I$   We get an NFA map, same size as I
19  Detection map $\leftarrow -\log_{10}$(NFA) $> t^{NFA}$
20  return Detection map
```

Some false positives coincide with spots on bright roads mistaken for a hub as we can see in row 3 columns 3–4, or along natural identifiable dark structures in the image, row 4 columns 1–2. This illustrates the weak part of this algorithm: we have designed a detector of bright dot and dark segment at the same time. A very visible shadow can be sufficient for a detection regardless of the presence of a bright dot, and vice-versa. Moreover, the shadow part of the detector can be easily misled by ground structures like paths, roads or crop fields; the hub part can be duped by bright structures such as buildings, roads or snow.

Some other false positives come from less visible yet existent structure, row 2 columns 1–2 for example. To get rid of the latter, we can try to increase t^{shadow} or t^{hub}. Some wind turbines are not detected, mostly in the first and last images. In these images, they are less visible than in the other ones; either the shadows do not emerge well from the background or the hubs are not very bright. On the borders of the images, a safe zone is not explored by the algorithm. That is because on the pixels too close to the borders it may not be possible to fully sample shadow and hub.

On the big image, the same observations follow. We can also see that bright buildings can be mistaken for wind turbines.

To quantify the influence of the examined parameters of the method, we designed the following

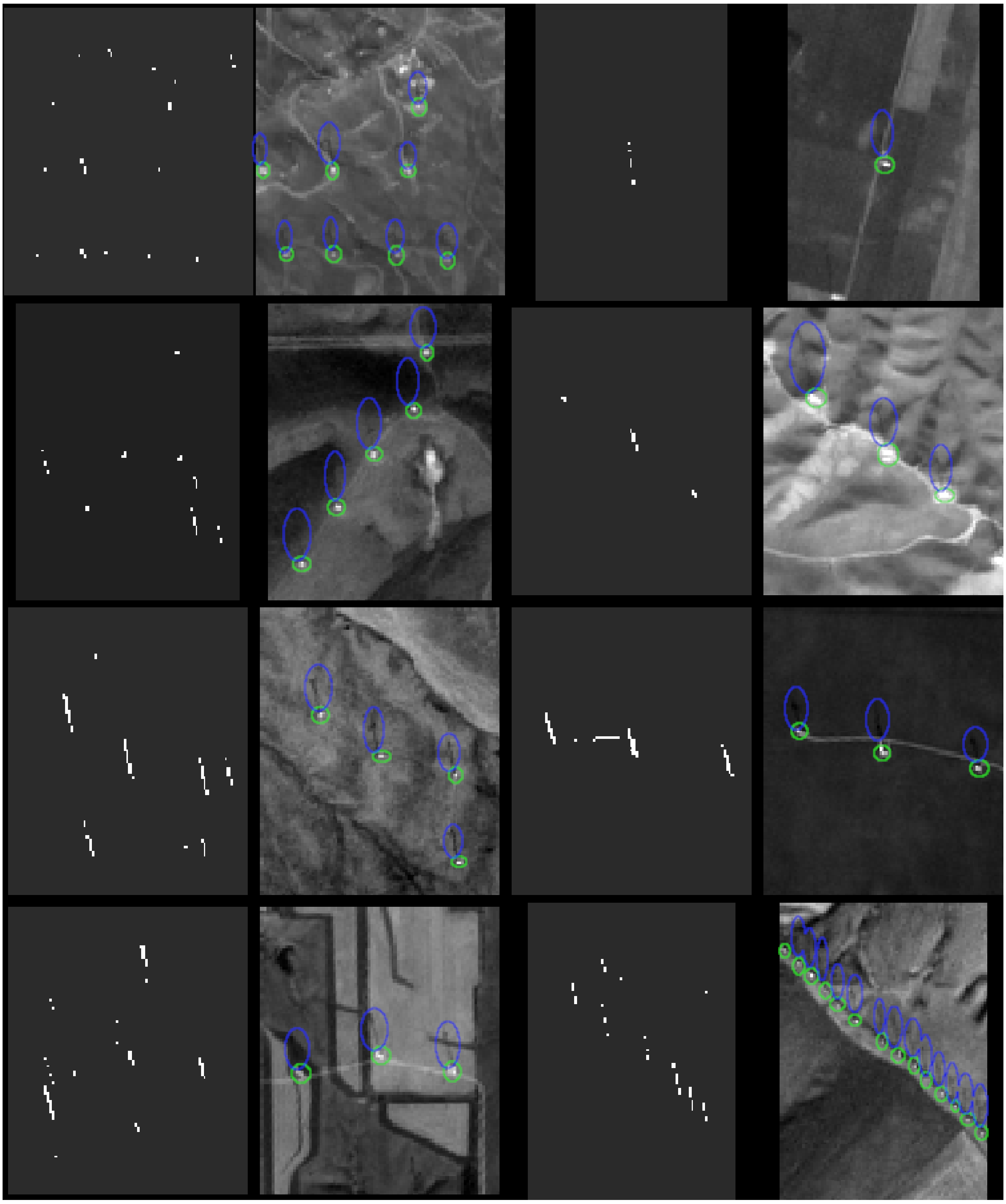
Figure 4: Wind turbine detection computed with the proposed algorithm on eight Sentinel-2 images. First and third columns are detection maps, where gray pixels correspond to undetected pixels while white corresponds to detected ones; second and fourth columns are corresponding Sentinel-2 images. The gray background on the detection maps helps visualizing the borders of the images, contrasting with the black background. The parameters $(t^{shadow}, t^{hub}, t^{NFA})$ are set to $(25, 50, 1)$. On the Sentinel-2 images, hubs are circled in green and shadows in dark blue.

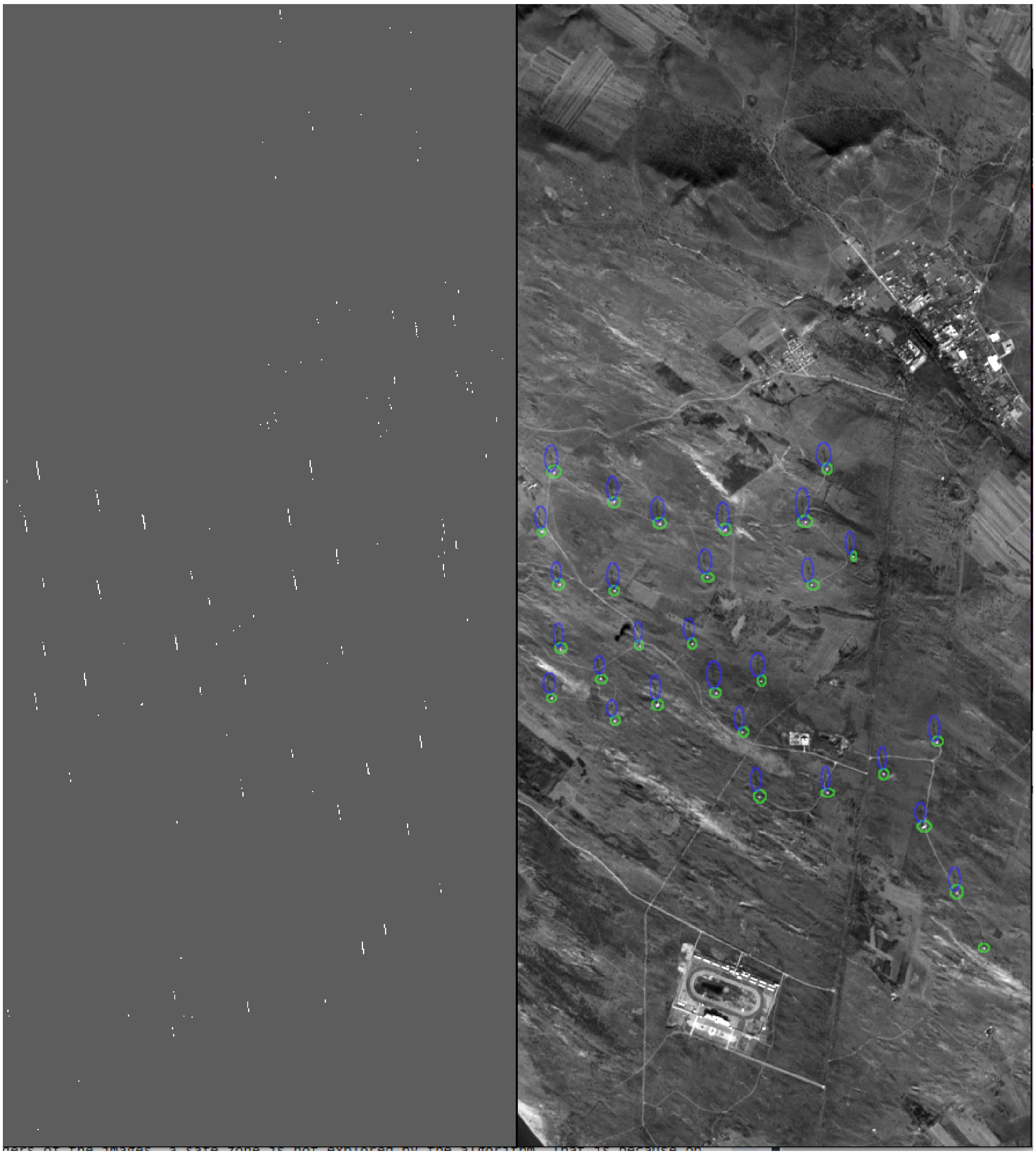

Figure 5: Wind turbine detection computed with the proposed algorithm on a wide Sentinel-2 image. On the left is the detection map, where gray pixels correspond to undetected pixels while white corresponds to detected ones; on the right is the corresponding Sentinel-2 image. The gray background the on detection maps helps visualizing the borders of the images. The parameters $(t^{shadow}, t^{hub}, t^{NFA})$ are set to $(25, 50, 1)$. On the Sentinel-2 image, hubs are circled in green and shadows in dark blue.

experiment: we applied the algorithm on two sets of around 300 Sentinel2 images: one containing a wind turbine in its center, and the other not. We only look at a central square of the images to assess if a detection is made or not. We can then compute the number of true and false positives, true and false negatives, and finally a F1-score. Each set of $(t^{shadow}, t^{hub}, t^{NFA})$ gives a F1-score; by making one of these parameters vary while fixing the other two we can check the robustness of the method. See Figure 6.

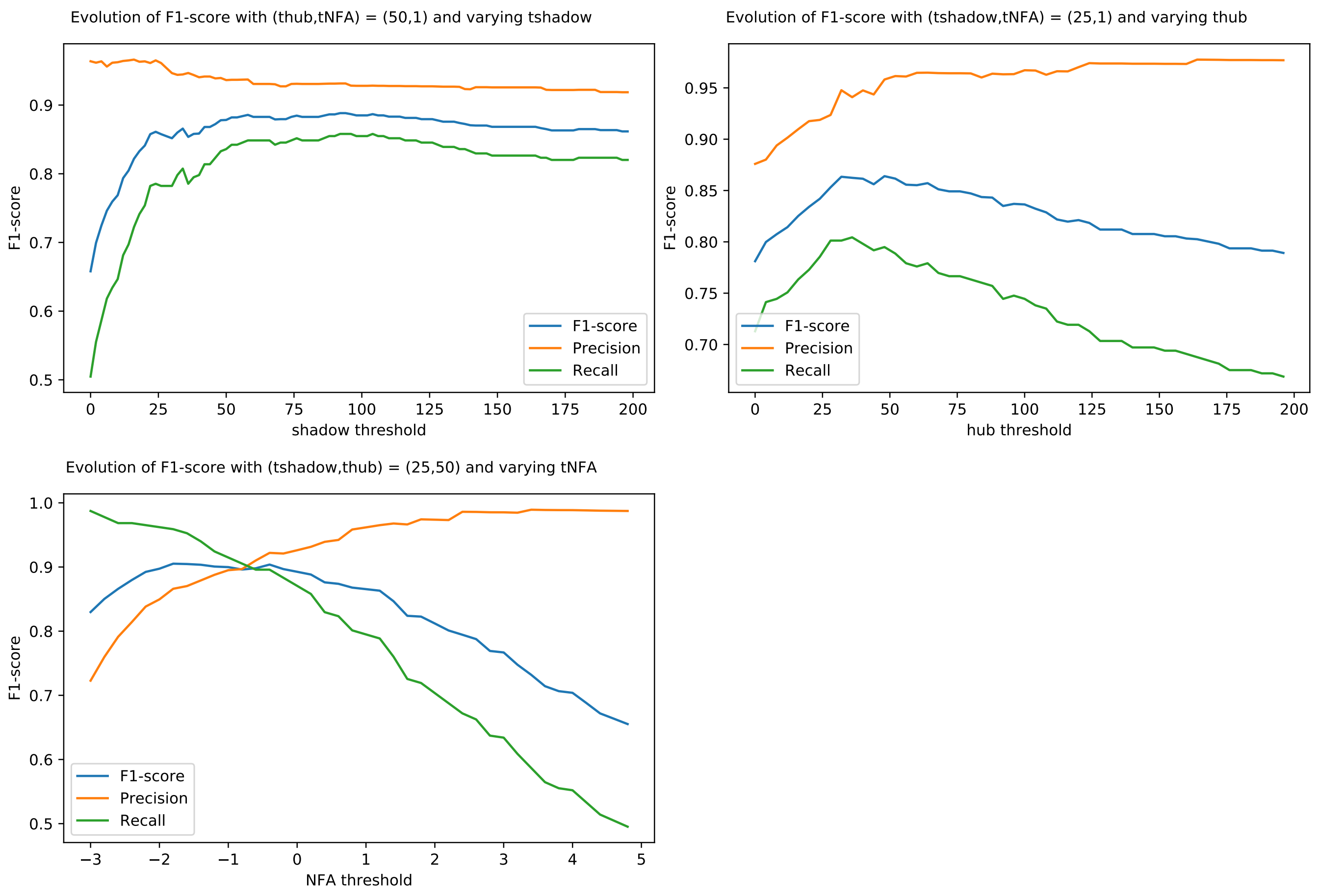

Figure 6: Performance curves. Upper left: evolution of F1-score (blue) when t^{shadow} varies from 0 to 200. Here, $(t^{NFA}, t^{hub}) = (1, 50)$, fixed. Precision (orange) and Recall (green) are plotted. Upper right: evolution of F1-score (blue) when t^{hub} varies from 0 to 200. Here, $(t^{shadow}, t^{NFA}) = (25, 1)$, fixed. Precision (orange) and Recall (green) are plotted. Lower left: evolution of F1-score (blue) when t^{NFA} varies from -3 to 5. Here, $(t^{shadow}, t^{hub}) = (25, 50)$, fixed. Precision (orange) and Recall (green) are plotted.

4.1 Impact of t^{shadow}

Figure 7 shows the same eight Sentinel-2 images of Figure 4 and their detection maps with the set of parameters: $(t^{shadow}, t^{hub}, t^{NFA}) = (100, 50, 1)$. We try to toughen the shadow test, to get only very visible shadows. In this figure, the segment-shaped detections mostly disappear. The detection maps are globally cleaner, with yet again false detections. The ones due to a misleading bright dot remain; those misidentified earlier in the image, in row 2 columns 1–2, disappear. Unfortunately the bottom-left image is way worse than before. Here, the ground dark structures are fully spotted as shadows, since their directions coincide well. In a way, this is just a worsening of the two-in-one detector problem. If we toughen the test for a shadow pixel, then its test becomes more significant and allows more easily to overlook the hub part. If we were to keep raising t^{shadow}, we would ultimately get a brightness detector. Indeed, if the threshold is too high, no pixel will pass the shadow test and only the hub detector will be discriminant.

Figure 6, upper left, shows the influence of the variation of t^{shadow} on the performances. There is a good improvement from 0 to 25, then it stagnates.

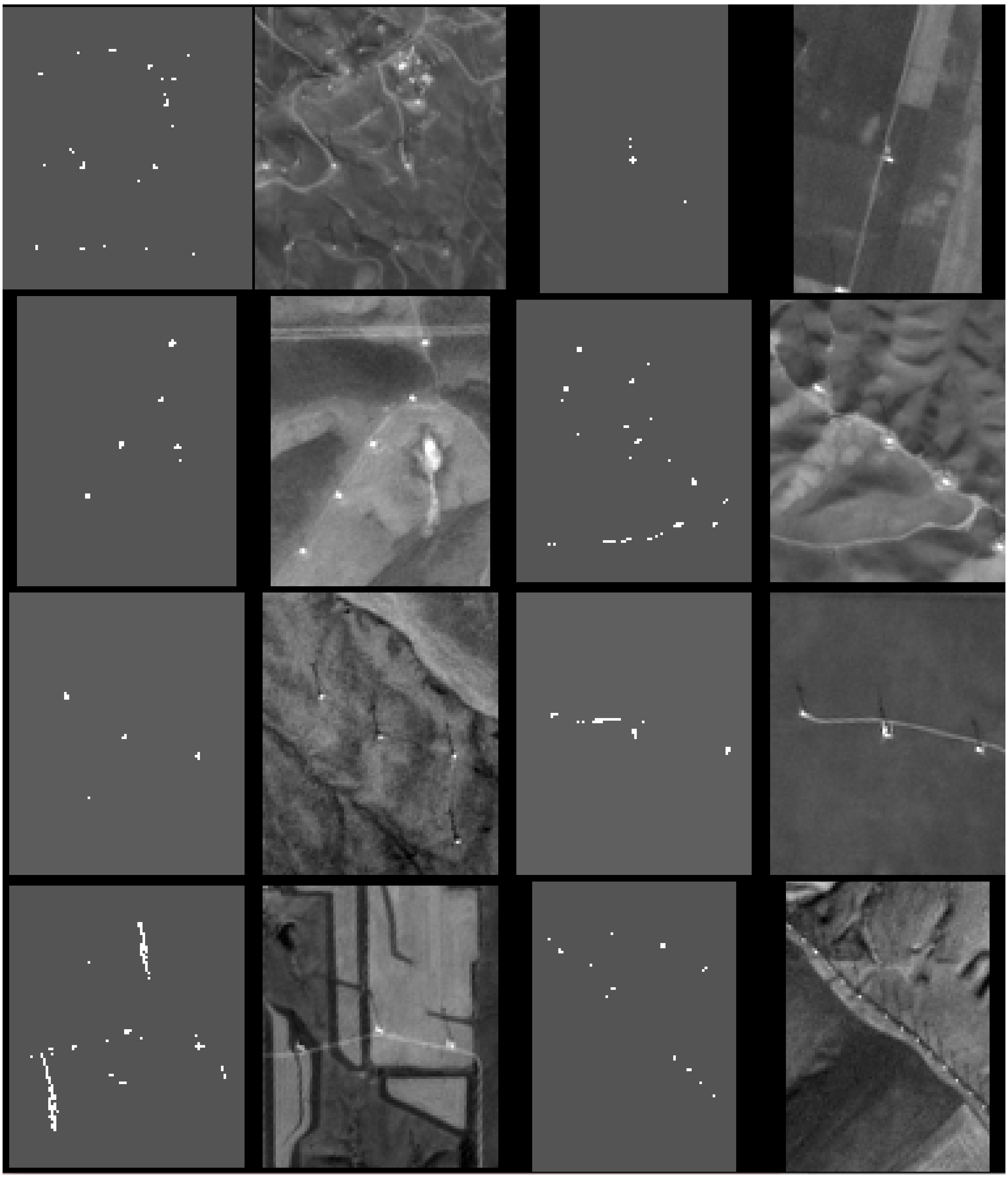

Figure 7: Wind turbine detection computed with the proposed algorithm on eight Sentinel-2 images. First and third columns are detection maps, where gray pixels correspond to undetected pixels while white corresponds to detected ones; second and fourth columns are corresponding Sentinel-2 images. The gray background on the detection maps helps visualizing the borders of the images, contrasting with the black background. The parameters $(t^{shadow}, t^{hub}, t^{NFA})$ are set to $(100, 50, 1)$.

4.2 Impact of t^{hub}

Figure 8 shows the same eight Sentinel-2 images and their detection maps with the set of parameters: $(t^{shadow}, t^{hub}, t^{NFA}) = (25, 200, 1)$. Here the detection maps are very close to what we had with the initial set of parameters $(25, 50, 1)$, see Figure 4. Some correct detections have disappeared (see rows 1 and 2 columns 1–2) and some false road detections have disappeared (row 3 columns 3–4). It can be explained as follows: the significance of unchanged detections was mainly driven by the shadow part of the detector; the differences with the first set of parameters appear on detections which were driven by the hub part of the detector. This shows how predominant in the detector the shadow part is compared to the hub part. If we were to keep increasing t^{hub}, we would ultimately get a shadow detector, for the reciprocal reason described in the previous section for t^{shadow}.

Figure 6, upper right, shows the influence of the variation of t^{hub} on the performances. The variations are quite small. A peak is around 50, then it decreases slowly.

4.3 Impact of t^{NFA}

Figure 9 shows the same eight Sentinel-2 images and their detection maps with the set of parameters: $(t^{shadow}, t^{hub}, t^{NFA}) = (25, 50, -3)$; Figure 10 shows the same eight Sentinel-2 images and their detection maps with the set of parameters: $(t^{shadow}, t^{hub}, t^{NFA}) = (25, 50, 5)$.

With $t^{NFA} = -3$, overdetection occurs. Indeed since we relax our final threshold, a lot of non-significant pixels are detected. We can see a great aggravation of the segment-shaped detections, at its utmost in the last image. We can also guess the limits where we set our safety gap away from the borders, in the first image for example: this is the rectangular frame without detection, which stands out thanks to a gestaltic completion.

On the contrary, with $t^{NFA} = 5$, we just have a few detections. It is expected since we ask for greater significance. The detected pixels are almost all correct, yet misleading dark components still provoke false detections, as we can see in rows 2, 3 and 4, columns 1–2. The segment-shaped detections have greatly decreased.

Figure 6, lower left, shows the influence of the variation of t^{NFA} on the performances. Between -2 and 1 it remains close to constant, outside this interval it worsens. While setting the NFA threshold too low means too many false detections, setting it too high means too few correct detections.

5 Conclusion

An algorithm for wind turbine detection in Sentinel-2 images was described based on the a contrario statistical framework. The method produces correct results when the parameters are well-chosen, but fails when misleading structures cover the ground. The most degrading ones are false shadows, created by paths, roads or crop fields. Adapting the algorithm to use time series with various shadow shapes should fix the problem.

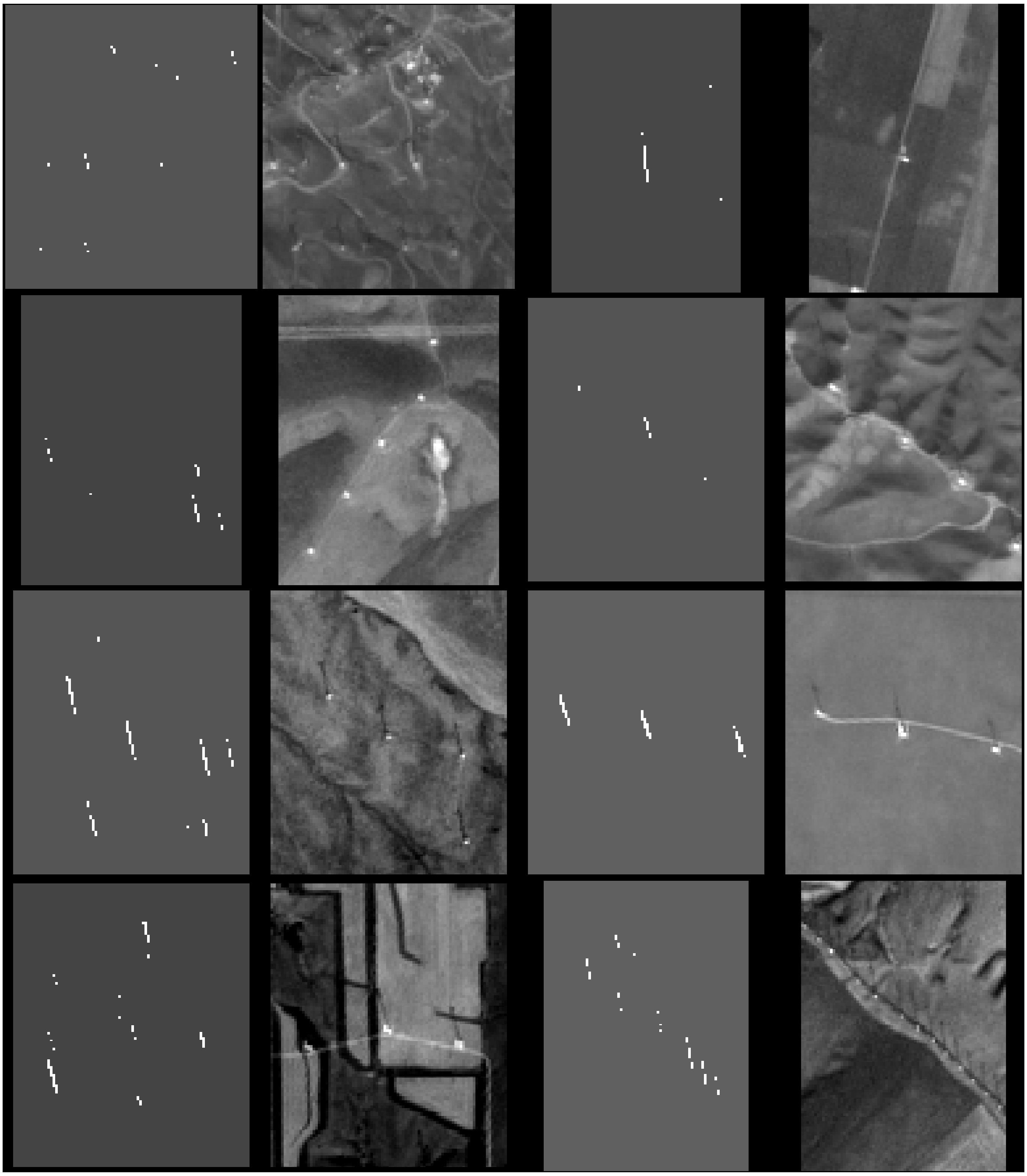

Figure 8: Wind turbine detection computed with the proposed algorithm on eight Sentinel-2 images. First and third columns are detection maps, where gray pixels correspond to undetected pixels while white corresponds to detected ones; second and fourth columns are corresponding Sentinel-2 images. The gray background on the detection maps helps visualizing the borders of the images, contrasting with the black background. The parameters $(t^{shadow}, t^{hub}, t^{NFA})$ are set to $(25, 200, 1)$.

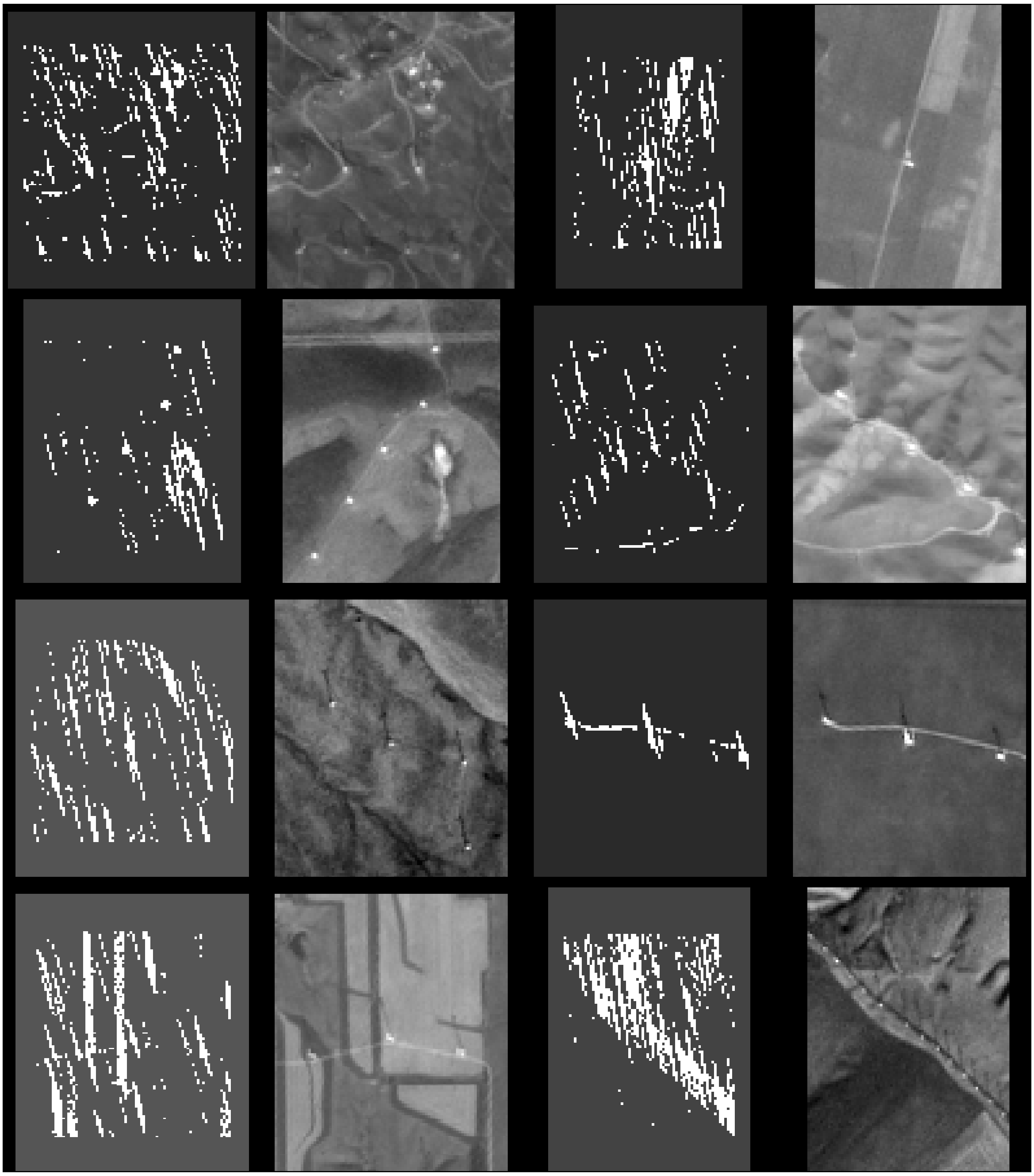

Figure 9: Wind turbine detection computed with the proposed algorithm on eight Sentinel-2 images. First and third columns are detection maps, where gray pixels correspond to undetected pixels while white corresponds to detected ones; second and fourth columns are corresponding Sentinel-2 images. The gray background on the detection maps helps visualizing the borders of the images, contrasting with the black background. The parameters $(t^{shadow}, t^{hub}, t^{NFA})$ are set to $(25, 50, -3)$.

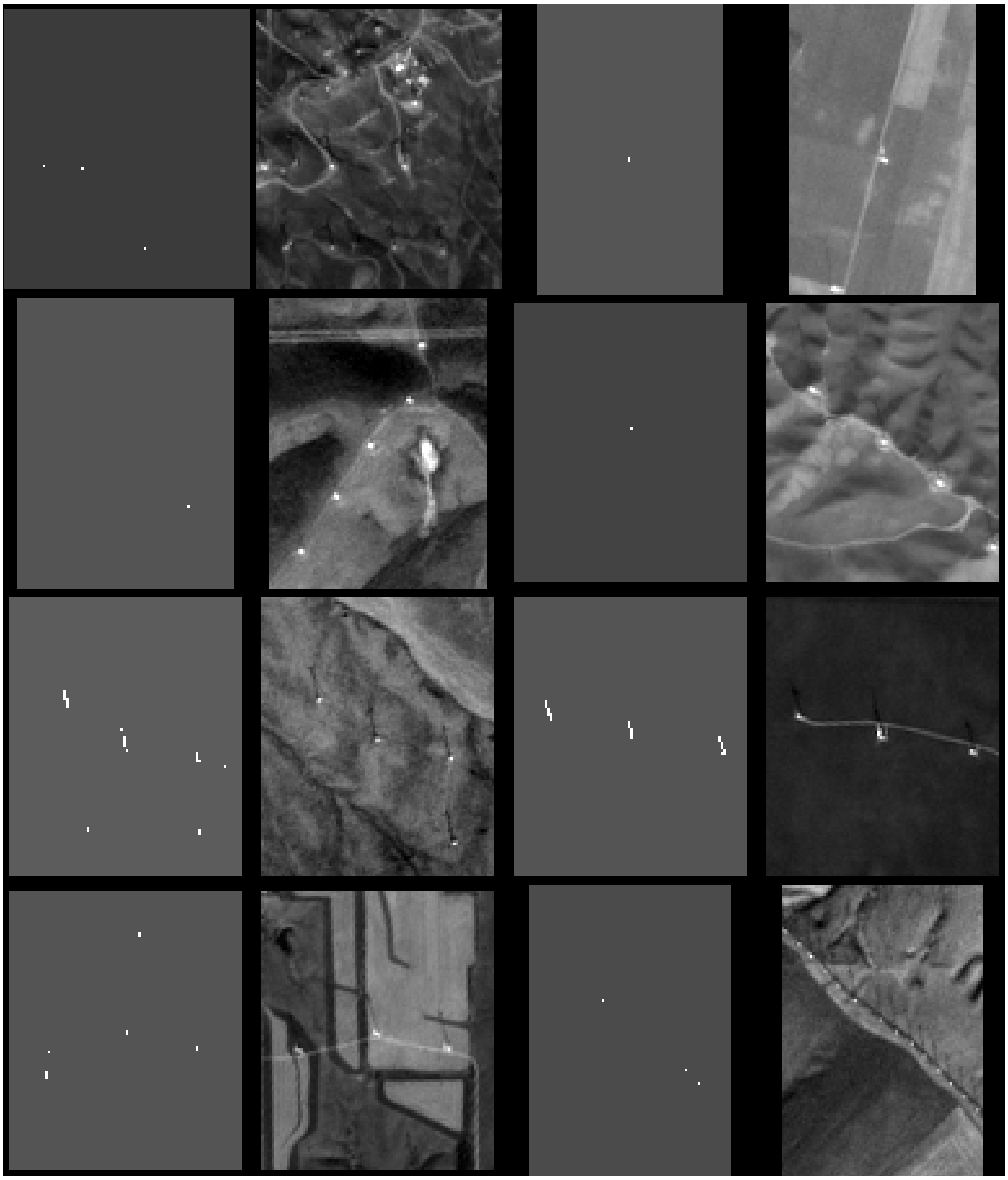

Figure 10: Wind turbine detection computed with the proposed algorithm on eight Sentinel-2 images. First and third columns are detection maps, where gray pixels correspond to undetected pixels while white corresponds to detected ones; second and fourth columns are corresponding Sentinel-2 images. The gray background on the detection maps helps visualizing the borders of the images, contrasting with the black background. The parameters $(t^{shadow}, t^{hub}, t^{NFA})$ are set to $(25, 50, 5)$.

Acknowledgments

This work is partly financed by Office of Naval research grant N00014-17-1-2552 and N00014-20-S-B001, DGA Astrid project "Filmer la Terre" nº ANR-17-ASTR-0013-01 and Fondation Mathématique Jacques Hadamard. This study is supported by the French Ministry for Higher Studies, Research and Innovation.

Image Credits

Pexels, CC0.
All other images are provided by the authors.

References

[1] *Deep learning utilizing satellite imagery feature detection to create a wind turbine register*, (2019). https://supperundsupper.com/wp-content/uploads/2019/10/Deep-Learning-Utilizing-Satellite-Imagery.pdf.

[2] F. Abedini, M. Bahaghighat, and M. S'hoyan, *Wind turbine tower detection using feature descriptors and deep learning*, Facta Universitatis, Series: Electronics and Energetics, 33 (2019), pp. 133–153. http://dx.doi.org/10.2298/FUEE2001133A.

[3] Awea Association, *Wind powers America amended annual report*, tech. report, AWEA, 2019.

[4] R. Chang, R. Zhu, and P. Guo, *A Case Study of Land-Surface-Temperature Impact from Large-Scale Deployment of Wind Farms in China from Guazhou*, Remote Sensing, 8 (2016), p. 790. https://doi.org/10.3390/rs8100790.

[5] M. Christiansen and C. Hasager, *Wake effects of large offshore wind farms identified from satellite SAR*, Remote Sensing of Environment, 98 (2005), pp. 251–268. http://dx.doi.org/10.1016/j.rse.2005.07.009.

[6] A. Desolneux, L. Moisan, and J-M. Morel, *From Gestalt Theory to Image Analysis: A Probabilistic Approach*, Springer Publishing Company, 1st ed., 2007. ISBN 0387726357.

[7] M. Han, H. Wang, G. Wang, and Y. Liu, *Targets mask U-Net for Wind Turbines Detection in Remote Sensing Images*, ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLII-3 (2018), pp. 475–480. http://dx.doi.org/10.5194/isprs-archives-XLII-3-475-2018.

[8] B.D. Hoen, J.E. Diffendorfe, J.T. Rand, L.A. Kramer, C.P. Garrity, and H.E. Hunt, *United states wind turbine database*, 2018 (ver. 3.3, January 14, 2021). https://doi.org/10.5066/F7TX3DN0.

[9] N. Mandroux, T. Dagobert, S. Drouyer, and R. Grompone von Gioi, *Wind turbine detection on Sentinel-2 images*, IEEE International Geoscience and Remote Sensing Symposium, (2021). https://doi.org/10.1109/IGARSS47720.2021.9554578.

[10] A. Myaskouvskey, Y. Gousseau, and M. Lindenbaum, *Beyond independence: An extension of the a contrario decision procedure.*, International Journal of Computer Vision, 101 (2013), pp. 22–44. https://doi.org/10.1007/s11263-012-0543-6.

[11] R. Nepal, J. Cai, and Z. Yan, *Micro-Doppler radar signature identification within wind turbine clutter based on short-CPI airborne radar observations*, IET Radar, Sonar & Navigation, 9 (2015), pp. 1268–1275. http://dx.doi.org/10.1049/iet-rsn.2015.0111.