E-Infrastructures

H2020-EINFRA-2015-1

EINFRA-5-2015: Centres of Excellence for computing applications

EoCoE

Energy oriented Center of Excellence for computing applications

Grant Agreement Number: EINFRA-676629

D2.10 - M36

GHI forecasts at local scale

Project and Deliverable Information Sheet

EoCoE	Project Ref:	EINFRA-676629
	Project Title:	Energy oriented Centre of Excellence
	Project Web Site:	http://www.eocoe.eu
	Deliverable ID:	D2.10 M36
	Lead Beneficiary:	CEA
	Contact:	Isabelle Herlin
	Contact's e-mail:	Isabelle.Herlin@inria.fr
	Deliverable Nature:	Report
	Dissemination Level:	PU*
	Contractual Date of Delivery:	M36 30/09/2018
	Actual Date of Delivery:	M36 28/09/2018
	EC Project Officer:	Carlos Morais-Pires

* - The dissemination level are indicated as follows: PU – Public, CO – Confidential, only for members of the consortium (including the Commission Services) CL – Classified, as referred to in Commission Decision 2991/844/EC.

Document Control Sheet

Document	Title :	GHI forecasts at local scale
	ID :	D2.10 M36
	Available at:	http://www.eocoe.eu
	Software tool:	LaTeX
Authorship	Written by:	Dominique Béréziat, Isabelle Herlin
	Contributors:	Etienne Huot
	Reviewed by:	PEC members

Contents

List of Figures

List of Tables

1. Introduction

This deliverable discusses the forecast results on two criteria:

- Execution time. Operational conditions require issuing one forecast every ten seconds, for operational and real time use in the system monitoring the energy network.
- Quality of results. Evaluation is proposed on data sets provided by EdF R&D with comparison to the persistence method, as described in the EoCoE proposal.

Section 2 first summarizes the method under evaluation, which is extensively described in Deliverable D2.9 [1]. The issue of execution time is discussed in Section 3, quantifying results obtained from the research work achieved thanks to EoCoE. Then, comparison with the persistence method is given in Section 4.2, on three data sets.

2. Synthesis of the *Solar Nowcast* method

As described in Deliverable D2.9 [1], the forecasting method, called *Solar Nowcast* in this document, is based on two steps: Motion Estimation and Forecasting.

- Forecasting is obtained by integrating in time, up to the chosen horizon, the following equations:

$$\frac{\partial F}{\partial t} + \nabla F.\mathbf{w} = 0 \tag{1}$$

$$\frac{\partial \mathbf{w}}{\partial t} + \nabla \mathbf{w}.\mathbf{w} = 0 \tag{2}$$

  from initial conditions on F as the last available acquisition and on $\mathbf{w}$ as the velocity map computed during the Motion Estimation step.

- Motion Estimation is described as an optimization problem with minimization of the following energy function:

$$J(\mathbf{X}(t_0)) = \int_\Omega \int_{t_0}^{t_3} \frac{\epsilon_I(\mathbf{x},t)^2}{R_I(\mathbf{x},t)} d\mathbf{x}dt + \int_\Omega \int_{t_0}^{t_3} \frac{\epsilon_\phi(\mathbf{x},t)^2}{R_\phi(\mathbf{x},t)} d\mathbf{x}dt + \int_\Omega \frac{\epsilon_b(\mathbf{x})^2}{B(\mathbf{x})} d\mathbf{x} \tag{3}$$

  with state vector $\mathbf{X} = \begin{pmatrix} I_s & \phi & \mathbf{w} \end{pmatrix}^T$ describing respectively the synthetic image, the signed distance map to image structures (clouds) and the velocities map.
  Let us define the model $\mathbb{M}$ that describes the dynamics of the state vector:

$$\frac{\partial \mathbf{X}}{\partial t} + \mathbb{M}(\mathbf{X}) = 0 \tag{4}$$

  It summarizes the following equations:

$$\frac{\partial I_s}{\partial t}(\mathbf{x},t) + \nabla I_s(\mathbf{x},t).\mathbf{w}(\mathbf{x},t) = 0 \tag{5}$$

$$\frac{\partial \phi}{\partial t}(\mathbf{x},t) + \nabla \phi(\mathbf{x},t).\mathbf{w}(\mathbf{x},t) = 0 \tag{6}$$

$$\frac{\partial \mathbf{w}}{\partial t}(\mathbf{x},t) + \nabla \mathbf{w}(\mathbf{x},t).\mathbf{w}(\mathbf{x},t) = 0 \tag{7}$$

  Let $\mathbb{H}$ be the observation operator defined by $\mathbb{H} = \begin{pmatrix} I_s - I \\ |\phi| - D_c \end{pmatrix}$.

Minimization of J is transformed as the computation of the solution of an optimality system:

$$\frac{\partial \mathbf{X}}{\partial t}(\mathbf{x},t) + \mathbb{M}(\mathbf{X})(\mathbf{x},t) = 0,\ \mathbf{x}\in\Omega,\ t\in]t_0,t_3] \quad (8)$$

$$\mathbf{X}(\mathbf{x},t_0) - \mathbf{X}_b(\mathbf{x}) = \epsilon_b(\mathbf{x}) \quad (9)$$

$$I_s(\mathbf{x},t) - I(\mathbf{x},t) = \epsilon_I(\mathbf{x},t),\ \text{if } t=t_i, \quad (10)$$

$$|\phi(\mathbf{x},t)| - D_c(\mathbf{x},t) = \epsilon_\phi(\mathbf{x},t),\ \text{if } t=t_i \quad (11)$$

and implemented with:

1. the forward integration in time of the following equations:

$$\mathbf{X}(\mathbf{x},0) = \mathbf{X}_b(\mathbf{x}) \quad (12)$$

$$\frac{\partial \mathbf{X}}{\partial t}(\mathbf{x},t) + \mathbb{M}(\mathbf{X})(\mathbf{x},t) = 0 \quad (13)$$

2. the backward integration of:

$$\lambda(\mathbf{x},t_3) = 0 \quad (14)$$

$$-\frac{\partial \lambda}{\partial t}(\mathbf{x},t) + \left(\frac{\partial \mathbb{M}}{\partial \mathbf{X}}\right)^* \lambda(\mathbf{x},t) = \left(\frac{\partial \mathbb{H}}{\partial \mathbf{X}}\right)^* R^{-1}\mathbb{H}(\mathbf{X},I,D_c)(\mathbf{x},t) \quad (15)$$

3. an optimization process, which computes new initial conditions on the state vector, $\mathbf{X}(t_0)$, with a steepest method descent. The gradient of J is given by:

$$\nabla J = B^{-1}(\mathbf{X}(0) - \mathbf{X}_b) + \lambda(t_0). \quad (16)$$

The steepest descent is performed by the quasi-Newton L-BFGS solver.

The Motion Estimation step is summarized with the algorithm:

read data ;
$\mathbf{X}(t_0) \leftarrow \mathbf{X}_b$;
while *! convergence* **do**
  Forward pass: compute $\mathbf{X}(t)$ for $t=t_0$ up to t_3, compute J ;
  Backward pass: compute $\lambda(t)$ for $t=t_3$ down to $t=t_0$, compute ∇J ;
  Optimization pass: compute the new value of $\mathbf{X}(t_0)$ from L-BGFS, J and ∇J ;
end
write results ;

Algorithm 1: 4D-Var for Motion Estimation.

In both Forecasting and Motion Estimation, the computing cost comes from the numerical integration of the equations and efforts must be made to optimize these parts.

3. Optimization

3.1 Forecasting

The costly part of Forecasting comes from the temporal integration of Equations (1,2). Reader is referred to Deliverable D2.9 for details about the numerical schemes that are applied for implementing these equations.

Serial run. On a test sequence of 3601 time steps (corresponding to a 10 minutes horizon forecast), the initial code (written in C and C++, and named C_1) ran in 169

seconds with one thread of the JURECA HPC system[1]. Optimization of the code is a major task of the work achieved in this EoCoE project[2], with the following objectives:

- exploit vectorization capabilities of modern CPUs (Intel Xeon E5-2680),
- reduce the number of conditional or redundant statements,
- remove useless memory copy statements.

On the same experimental setting, the optimized code (named C_2) performs in 76.9 seconds, with a speedup of 2.22.

Parallel run. The initial code C_1 contains OpenMP statements in order to distribute tasks on a partition of the image domain. To measure the run time performance of the proposed method, we use the scaling efficiency metrics. It is a common statistics used in the HPC community to measure the efficiency of a parallel code when the number of threads or processors increases. It is defined as $\frac{T_1}{T_n \times n}$ with T_1 the run time of the serial code and T_n the run time of the parallel code running on n threads. The initial code C_1 that is running in 169 seconds on one thread, reduces to 34 seconds on 8 threads and to 26 seconds on 24 threads. Consequently scaling efficiency is of 62% on 8 threads and 27% on 24 threads. This remains quite poor compared to the operational requirements. To improve scalability, OpenMP statements are refined to get a better granularity: bigger independent chunks allow to reduce the number of threads (a thread initialization is expensive) and benefit from vectorization. With the serial optimized code C_2, running in 76.9 seconds on 1 thread, this improvement allows the following performance: 11.5 seconds on 8 threads with a scaling efficiently of 84% and 4.8 seconds on 24 threads, with a scaling efficiently of 67%. Table 1 summarizes the performances of codes.

Table 1: Benchmarks on original and optimized Forecasting step of Solar Nowcast.

Threads	1	8	24
C_1 (time in seconds)	169	34	26
C_1 (scaling efficiently)		62%	27%
C_2 (time in seconds)	76.9	11.5	4.8
C_2 (scaling efficiently)		84%	67%

3.2 Motion Estimation

The original code of Motion Estimation (again named C_1) ran in 68 seconds on one thread, when motion is initialized with a null value. The objective is to reach a run time of about 5 seconds for an operational use. As it can be seen in Algorithm 1, Motion Estimation is an iterative process, each iteration having three costly components: the forward integration, the backward integration and the call to the solver. We detail, for each one, the issues to be solved and the retained solution.

Forward integration. Equation (8) is integrated in time in the same way than for the Forecasting step. Consequently the optimization is the same than the one described in Subsection 3.1.

The cost function J must be additionally computed. This is speed up using OpenMP

[1] `http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JURECA/Configuration/Configuration_node.html`

[2] Energy oriented Centre of Excellence for computer applications, H2020-EINFRA-2015-1

statements (parallelization thanks to a domain decomposition, then reduction of thread computations on a unique thread).

Backward integration. Equation (15) is first integrated backward in time. Then the gradient of J, ∇J, is computed according to Equation (16).

The computation of ∇J is speed up using OpenMP parallelization statements.

It is important to note that the computation of the adjoint variable requires the determination of the adjoint model $(\frac{\partial \mathbb{M}}{\partial \mathbf{X}})^*$. As the model $\mathbb{M}$ corresponds to a piece of code, the adjoint model also corresponds to a piece of code that is obtained with an automatic differentiation tool. We use the Tapenade software [2] for this purpose. The code submitted to Tapenade for differentiation is the optimized code obtained for the Forward integration. The adjoint code, resulting from Tapenade, is however further optimized and parallelized as explained below.

It should be noted that Tapenade has limitations for identifying independent iterations of a loop, which are occurring in the main loop applied on the image domain. Tapenade has to store each image variable in a stack for computing the adjoint, with the resulting three drawbacks of breaking vectorization, increasing the memory size requirement and forbidding OpenMP parallel directives. However, the adjoint code can be significantly improved if few directives are given to Tapenade for identifying these independent statements inside the loops. This consequently removes all Tapenade stack operations. The resulting adjoint code benefits from vectorization and can then easily be parallelized with OpenMP directives. The new run with this optimized adjoint code gives a result in 49.3 seconds on one thread.

Having introducing OpenMP statements in the adjoint code, we study its scaling efficiency: on 8 threads, the time run is 21.9 seconds, on 24 threads, the time run is 19.4 seconds.

Solver call. This process corresponds to the call of the L-BFGS solver, which is a sequential code written in Fortran 77. In the Fortran code of the solver, we identify four costly loops with independent statements. We then introduce OpenMP statements in order to parallelize these loops. L-BFGS using a number of BLAS calls, the code is also linked with the OpenBLAS library for replacing the BLAS module provided by L-BFGS software distribution.

Final optimization results. We obtain the following results for Motion Estimation on the JURECA HPC system: 55.4 seconds on one thread, 20.1 seconds on 8 threads and 12.5 seconds on 24 threads. Table 2 gives a more accurate view of the scaling efficiency of the Motion Estimation code. It can be seen that the scalability is good for the Forward and Backward processes and moderated for the solver. As the requirement is to perform a forecast in less than 10 seconds, the additional issues are currently investigated:

- First, the solver is not fully parallelized. The literature must be investigated to find a scalable alternative to L-BFGS.
- Second, the physical model included in Motion Estimation can be simplified. A stationary model of velocity is tested with good results. The mathematical equation, $\frac{\partial \mathbf{w}}{\partial t} = 0$, is simple and its use significantly decreases the computational cost of Forward and Backward processes. Additionally, the component ϕ of the state

Table 2: Scalability of Motion Estimation on JURECA node after optimization of L-BFGS.

Threads	Forward	Backward	Solver	Total time	Scaling efficiency
1	14.9	25.2	13.5	55.4	
2	12.9	25.0	11.9	51.7	54%
4	6.8	13.0	8.5	29.6	47%
8	4.3	7.3	7.2	20.1	34%
12	3.2	4.9	6.4	15.4	30%
16	2.8	3.8	6.2	13.8	25%
24	2.5	2.8	6.3	12.5	18%

vector may be suppressed without strongly decreasing the quality of results.

- Third, in the context of GHI nowcasting with the sliding-window technique, the velocity background value, $\mathbf{w}_b$, is initialized with the value computed on the previous window. With this improved initialization, much closer to the solution than a null initialization, the algorithm requires less iterations before convergence. As an example, we display in Table 3 the total time of Motion Estimation when taking as background velocity the value on the previous window. In that case, it can

Table 3: Run of Motion Estimation with initial velocity obtained from the previous window.

Threads	Total time	Scaling Efficiently
1	12.19	
2	6.934	89%
4	3.451	89%
8	1.777	86%
12	2.126	47%
24	2.269	22%

be seen that the run time of the Motion Estimation is less than 3 seconds. The objective to deliver a forecast at an horizon of 10 minutes each 10 seconds is then fully achieved.

The scalability of Motion Estimation remains essentially limited by the call to L-BFGS that has only been partially parallelized. The scalability is only good up to 8 threads, as shown in Tables 2 and 3. To expect decreasing significantly the run time with an higher number of threads, an alternative scalable solver has to be used in the future.

4. Experiments

4.1 Evaluation setting

To evaluate our approach, we propose to compare the results with a state-of-the-art method based on the persistence property. Persistence is frequently used in the Solar irradiance forecasting community [3, 4]. It assumes that the Solar irradiance persists in time series. In the *simple persistence* version, the evaluation of the forecasted GHI at horizon $t+h$ is given by its last known measure of GHI at time t:

$$\widehat{GHI}(t+h) = GHI(t) \tag{17}$$

In the *smart persistence* version, the forecast evaluation of GHI takes into account both the GHI at time t and a correcting factor K:

$$\widehat{GHI}(t+h) = GHI(t) \times K(t,h) \tag{18}$$

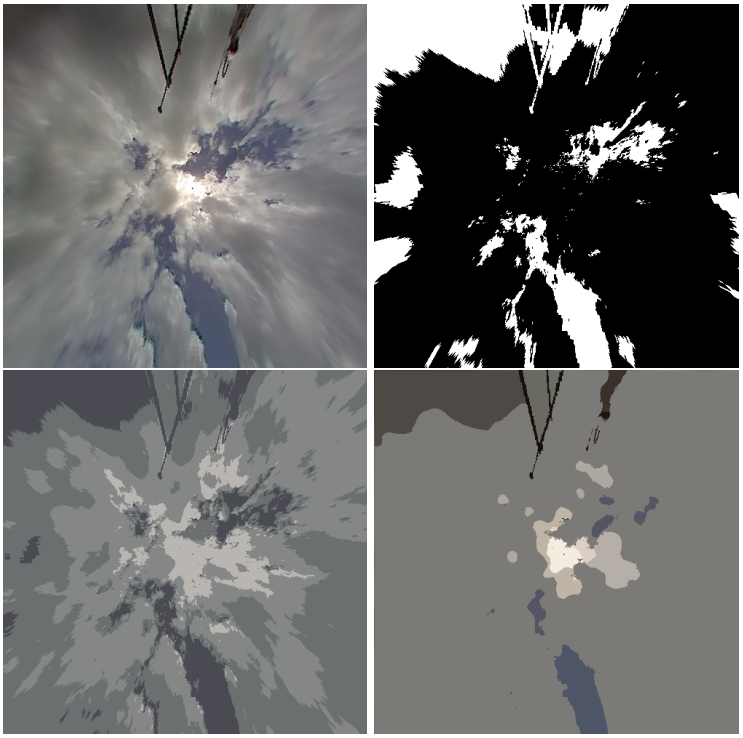

Figure 1: Examples of clear sky segmentation on an image acquired on 2013-01-29 at 07:51:35. Top: original image and thresholding. Bottom: k-means and mean-shift results.

K can be obtain in several ways from a statistical study of GHI time series. In this deliverable, we consider the simple persistence method, with $K = 1$, in order to demonstrate the potential of this image-based approach.

Let define CS as the Clear Sky index, giving the proportion of sky without cloud in the image. The Clear Sky index of an image I is computed from the image segmentation and defined by:

$$CS(I) = \frac{|S(I)|}{|\Omega|} \tag{19}$$

where $S(I)$ is the set of pixels identified as not cloudy in the image I, $|.|$ is the cardinal operator, and $|\Omega|$ is the number of pixels of image I.

It is therefore a key point to apply a reliable method of segmentation of clear sky for computing the set S. Several methods have been investigated (simple threshold, k-means [5], mean-shift [6]) to obtain the best possible cloud cover segmentation (see Figures 1 and 2). The following solution is chosen in this deliverable: clear sky is defined as pixels having a blue channel value greater than 10% of the red and green channels values; if I_R, I_G, I_B denotes respectively the RGB components of image I, then:

$$S(I) = \{\mathbf{x} \in \Omega | I_R(\mathbf{x}) > 1.1 \times \max(I_B(\mathbf{x}), I_G(\mathbf{x}))\}$$

This method is both fast and robust. It has been tested on each image of the three series (around 1800 images), and the quality of the segmentation has been verified on each one. Figure 3 displays two examples of clear sky segmentation in this data set.

Calculating a reliable GHI index requires locating accurately the Sun position and a segmentation method in order to identify the bright and dark clouds, and the solar and

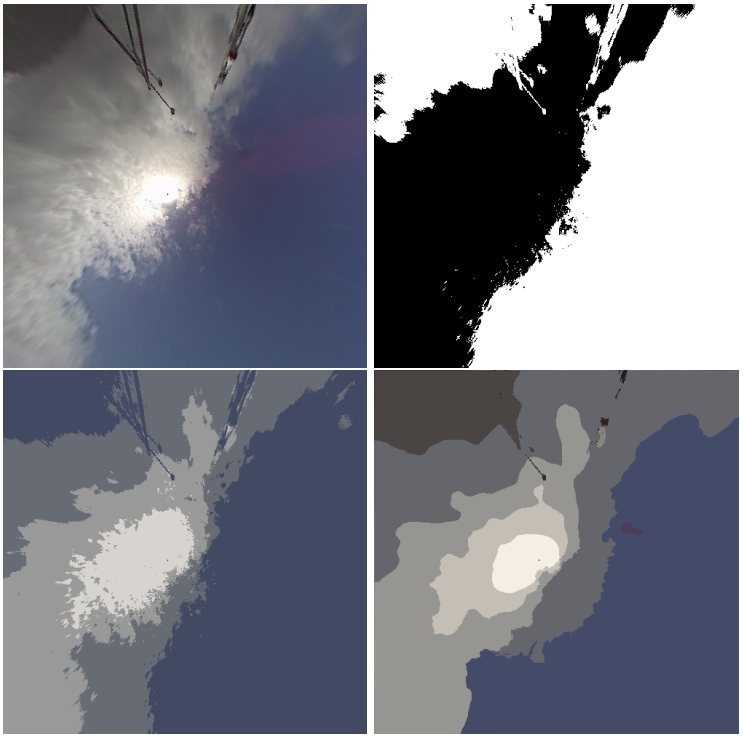

Figure 2: Examples of clear sky segmentation on an image acquired on 2013-10-22 at 09:28:01. Top: Original image and thresholding. Bottom: k-means and mean-shift results.

circumsolar disk (see [7]). However, a first index of GHI is computed as the sum of pixels luminescence in the clear sky:

$$GHI(I) = \sum_{\mathbf{x} \in S(I)} I(\mathbf{x}) \tag{20}$$

As our approach is based on the forecast of the cloud cover, its evaluation is linked to the one of clouds location in the image domain. Hence, it is more representative to evaluate the forecast error in terms of the Clear Sky index. This corresponds to a definition of GHI denoted by GHI_{CS} and given by:

$$GHI_{CS}(I) = \sum_{\mathbf{x} \in S(I)} 1 = CS(I). \tag{21}$$

Figure 4 shows the evolution according to time horizon of the error (computed as the discrepancy between the forecasted index and its ground truth) on GHI (computed using Equation (20)) and on GHI_{CS} (from Equation (21)).

Let remind that $I(t)$ denotes the acquisition from the sky imager at time t and $F(t)$ denotes the forecast of the sky image, computed with Solar Nowcast. The proposed protocol is:

1. Given four successive images $I(t_i)$, $i = 0, 1, 2, 3$, compute Motion Estimation and the forecast at time $t_3 + h$: $F(t_3 + h)$;

2. Compute $CS(F(t_3 + h))$ from Equation (21). This quantity is the Clear Sky index, obtained from the Solar Nowcast method;

3. Compute $CS(I(t_3))$. This quantity is the Clear Sky index with the simple persistence model (see Equation (17));

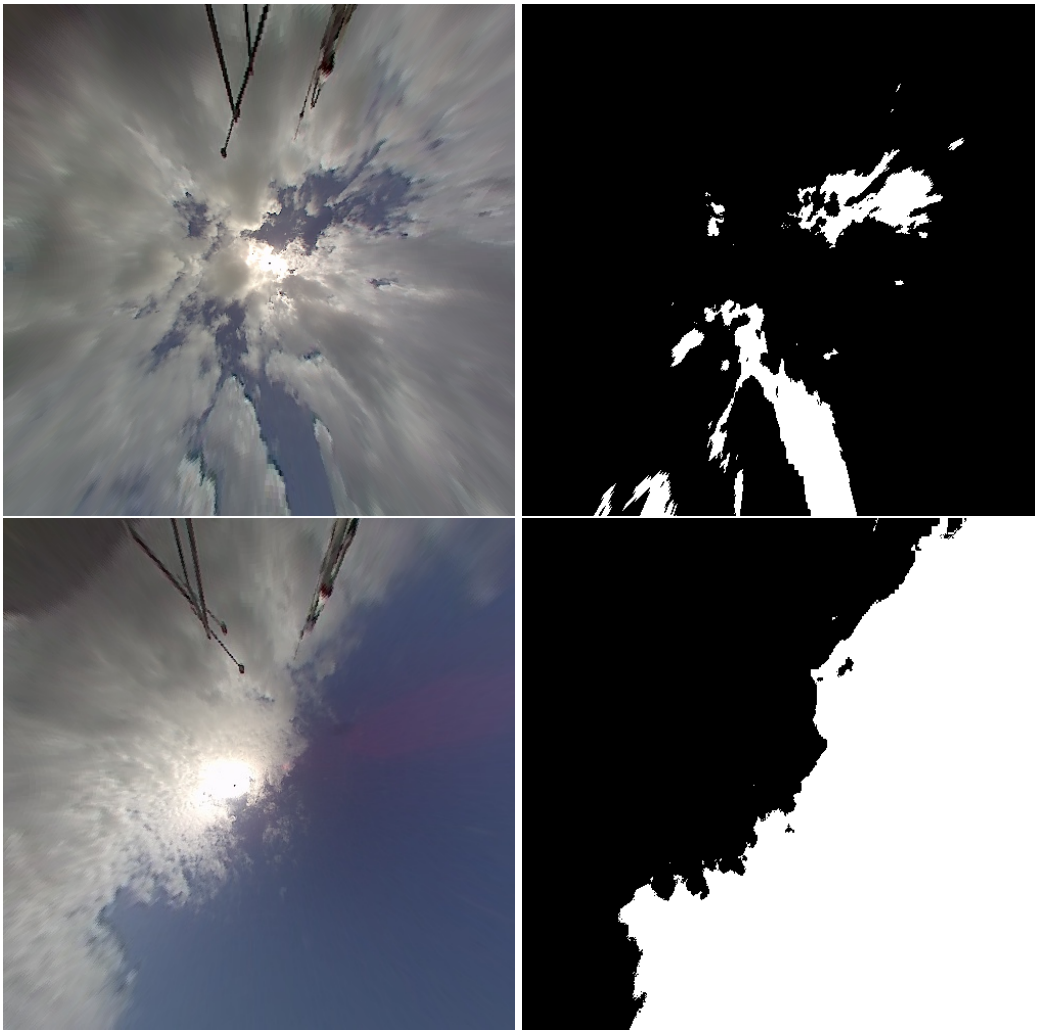

Figure 3: Segmenting clear sky. Left: original image. Right: result.

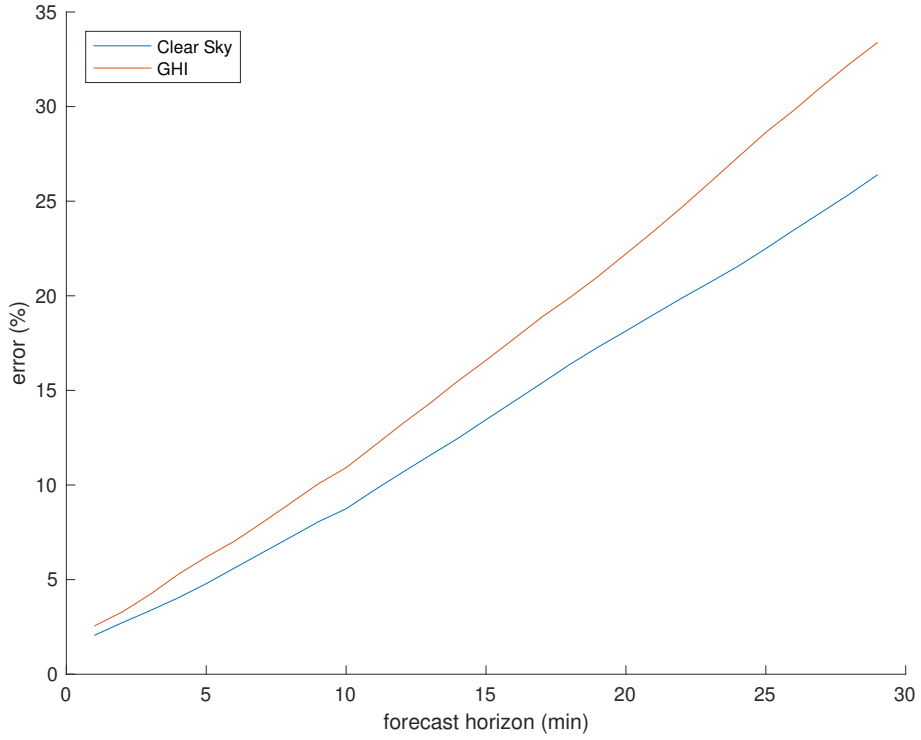

Figure 4: Error comparison using the two indices proposed in (20) (red line) and (21) (blue line). Episode of October 2013.

4. Compute $CS(I(t_3+h))$. This quantity is the Clear Sky index of the ground truth.

5. Compute the Solar Nowcast error as the discrepancy between $CS(F(t_3+h))$ and $CS(I(t_3+h))$, and the persistence error as the discrepancy between $CS(I(t_3))$ and $CS(I(t_3+h))$.

4.2 Results

Several error measures are applied when conducting experiments for different values of the temporal horizon h. This allows analyzing the robustness of Solar Nowcast both for short and long horizon terms. Moreover, this image-based method is compared to the simple persistence method on each data set. Figure 5 displays a few images acquired in October 2013 by the fish-eye sensor. Figure 6 gives the evolution of the forecasted Clear Sky index over one hour, on these data. Figures 7 and 8 illustrate the same information on the index evolution for data acquired respectively in January and June 2013. As seen on these images, for a nowcast at 1 minute horizon, the index of the forecast is strongly close to the ground truth. However, if the forecast horizon increases, the forecast error simultaneously increases.

These three episodes are corresponding to different clouds' evolution and are further and statistically analyzed in this section. These episodes correspond to one to two hours period of images acquisition. According to the time interval of ten seconds between two acquisitions in our setting, the statistics discussed in the following rely on about 1800 images of 501×501 pixels. Figure 9 displays the evolution of the forecast error on the Clear Sky index, according to the time horizon. As expected, the persistence model leads to an error increasing almost linearly with the time horizon. Consequently, Solar Nowcast

Figure 5: Four images acquired in October 2013.

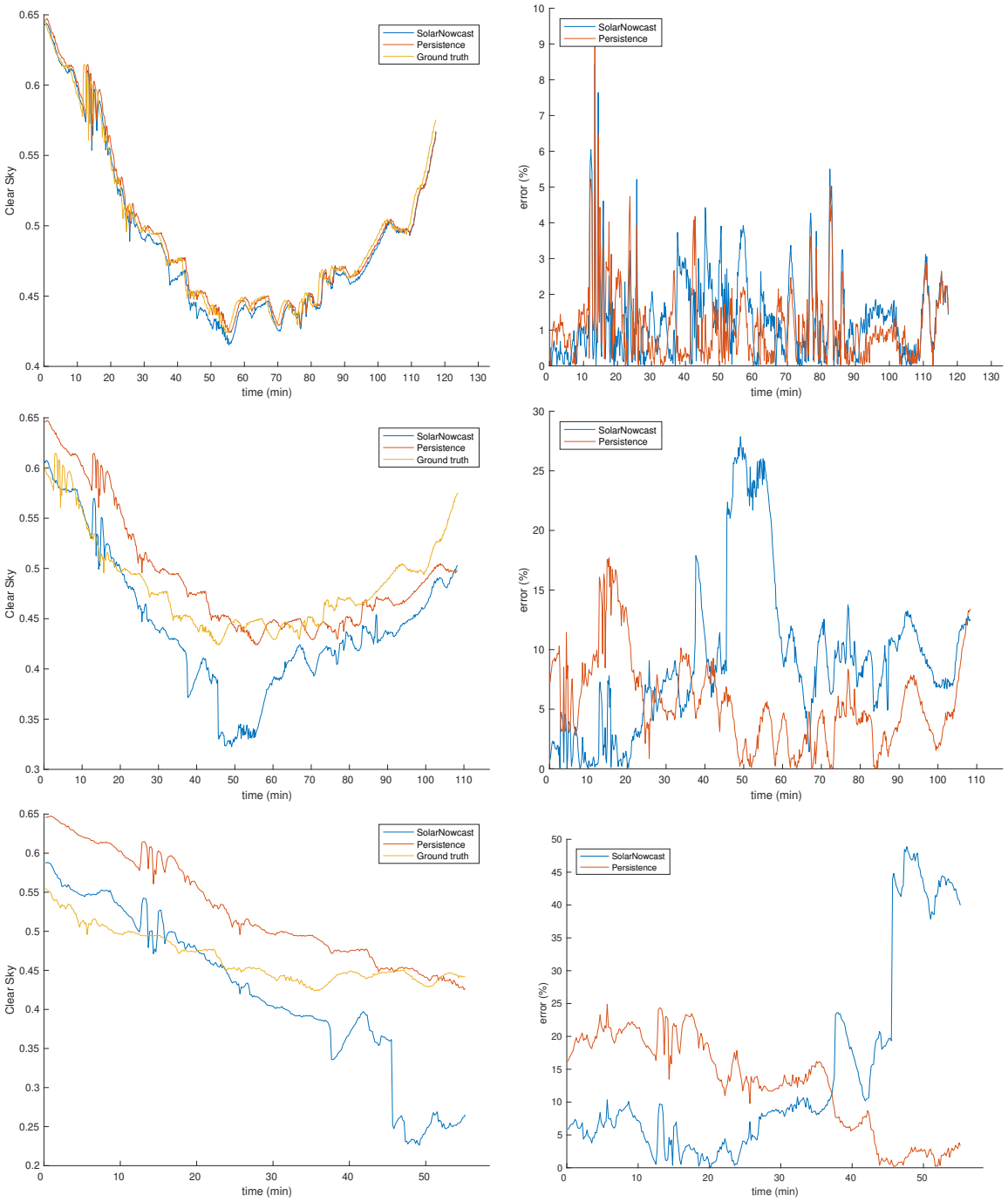

Figure 6: Left: evolution of the Clear Sky index at horizon $h = 1\,\text{min}$ (top), $h = 10\,\text{min}$ (middle), $h = 20\,\text{min}$ (bottom). Right: corresponding errors. Episode of October 2013.

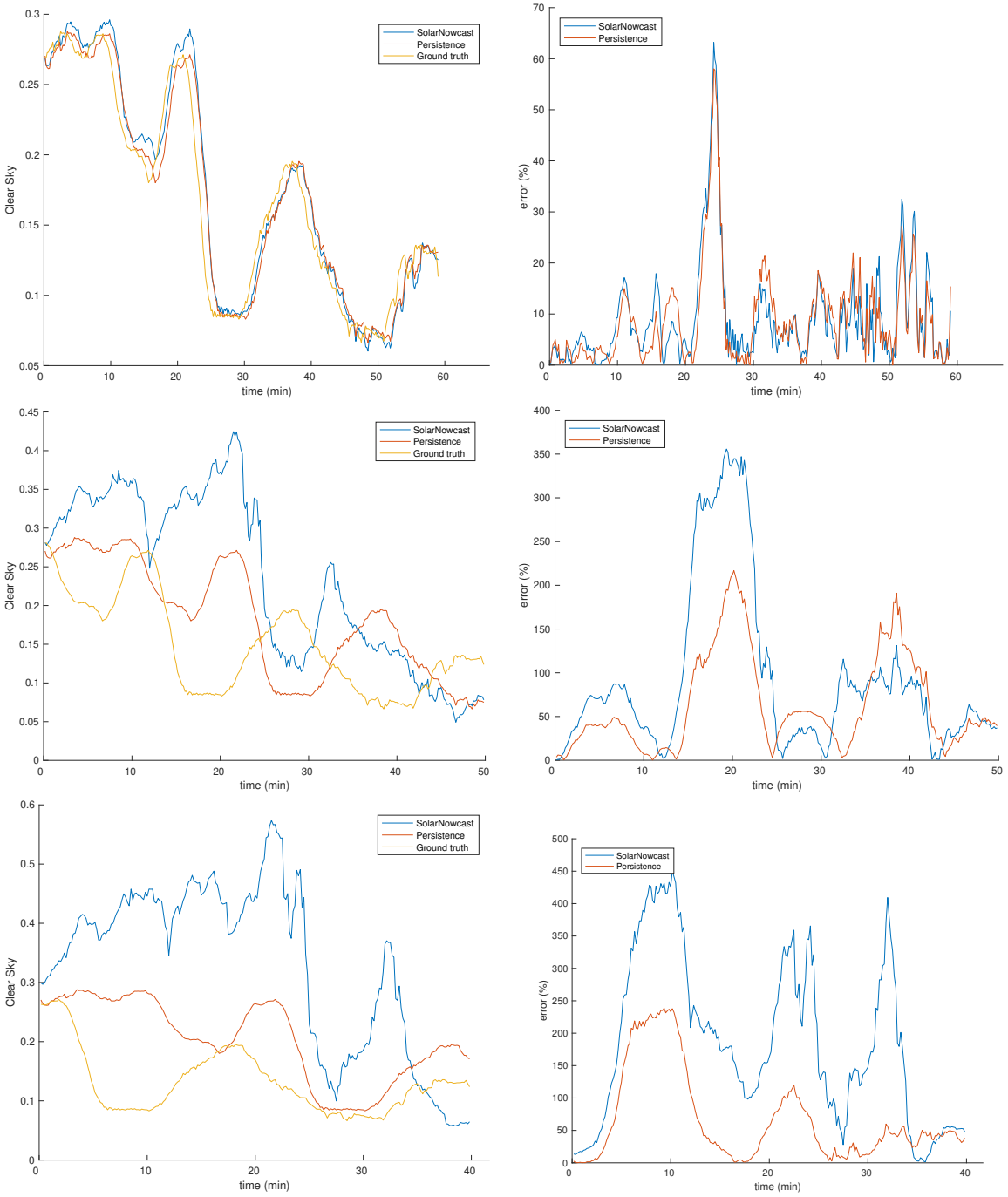

Figure 7: Left: evolution of the Clear Sky index at horizon $h = 1\,\text{min}$ (top), $h = 10\,\text{min}$ (middle), $h = 20\,\text{min}$ (bottom). Right: corresponding errors. Episode of January 2013.

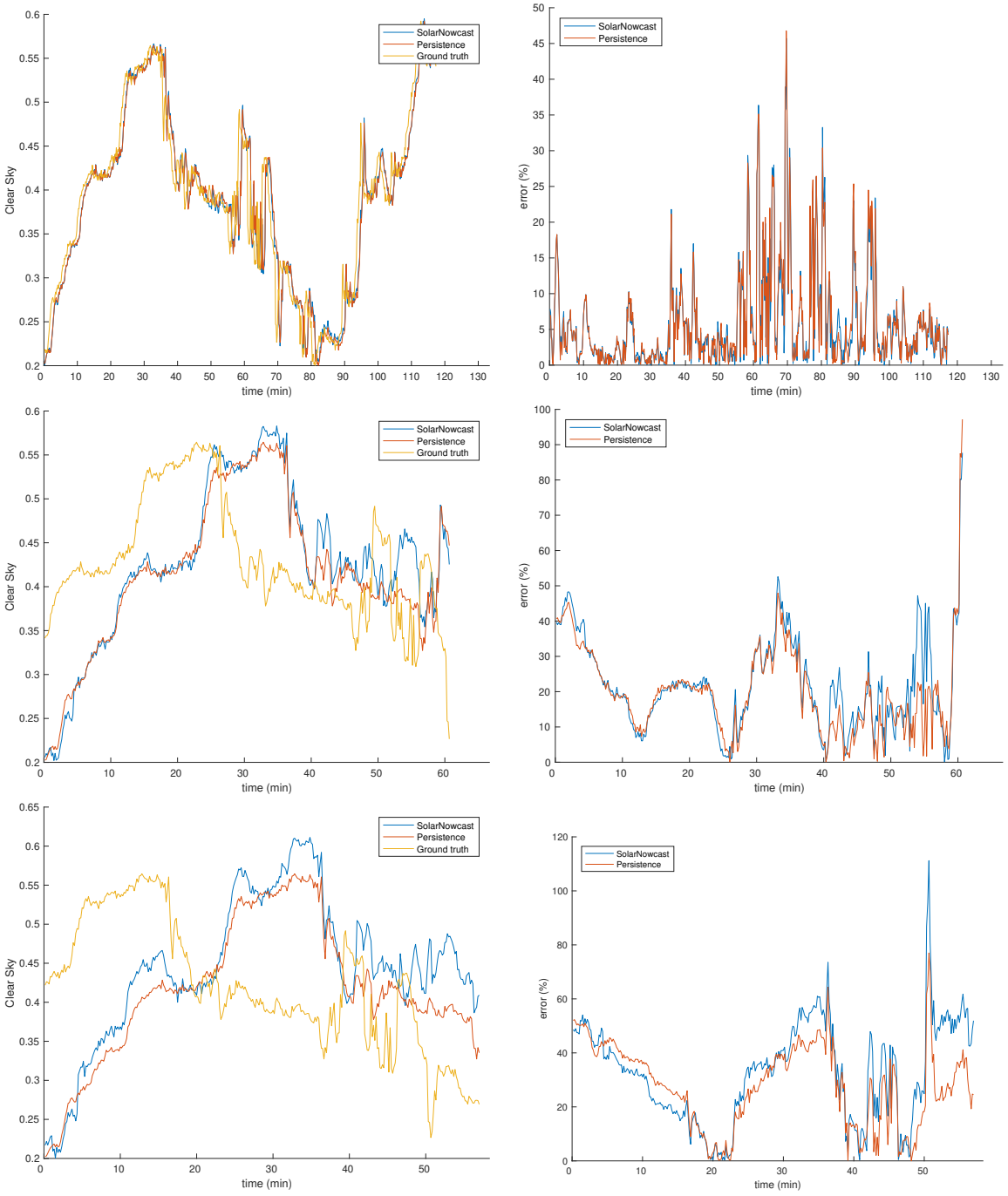

Figure 8: Left: evolution of the Clear Sky index at horizon $h = 1\,\text{min}$ (top), $h = 10\,\text{min}$ (middle), $h = 20\,\text{min}$ (bottom). Right: corresponding errors. Episode of June 2013.

becomes usually better than the simple persistence for a forecast of above 10 minutes. Below 10 minutes, the persistence error and Solar Nowcast error are similar as it can be seen on Figure 10, the error difference between the two methods being less than 1%.

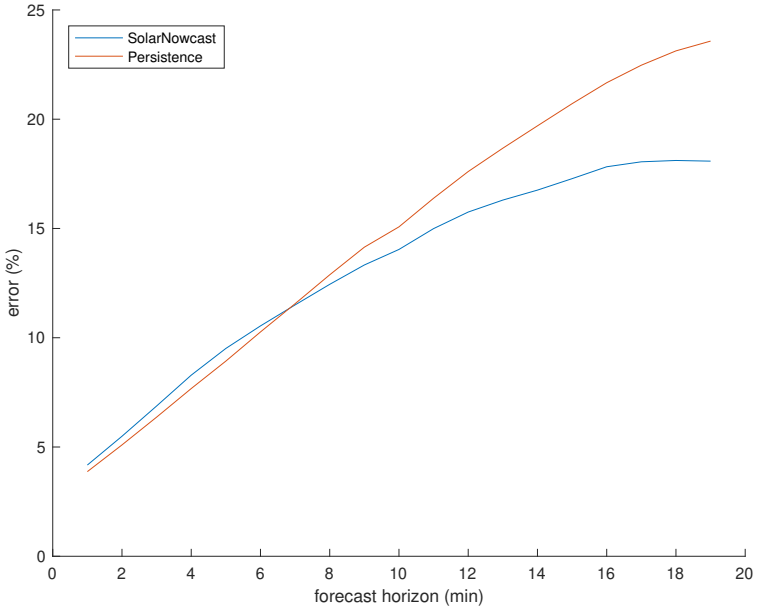

Figure 9: Evolution of the forecast error on the Clear Sky index according to time horizon using the whole set of images from the three episodes.

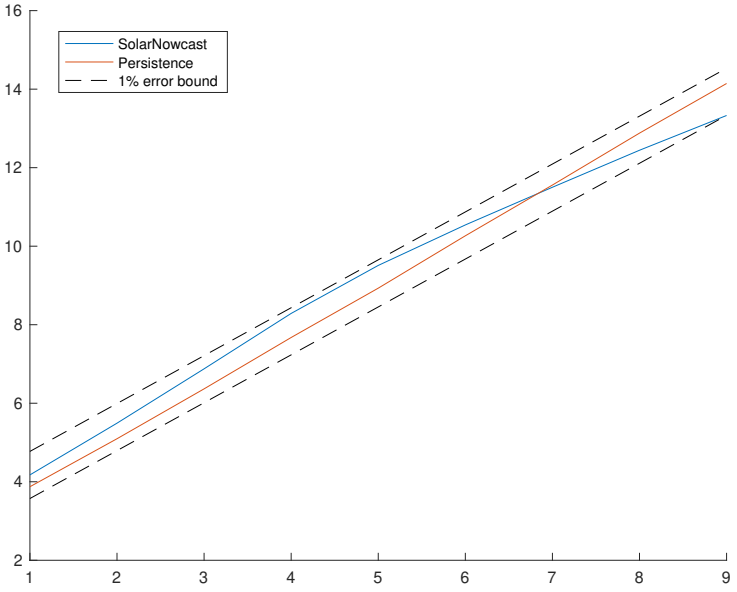

Figure 10: Close up on the evolution of forecast error on the Clear Sky index with 1% error bounds.

Solar Nowcast gets even better statistics than the simple persistence above 5 minutes, as shown on Figure 11, for the set of images acquired in October 2013.

Figure 12 displays image results for this episode. It permits to see that Solar Nowcast reaches its limitation at 20 min of forecast horizon. The numerical schemes used in the forecast step relie on a semi-Lagrangian approach. This approach is widely used for the

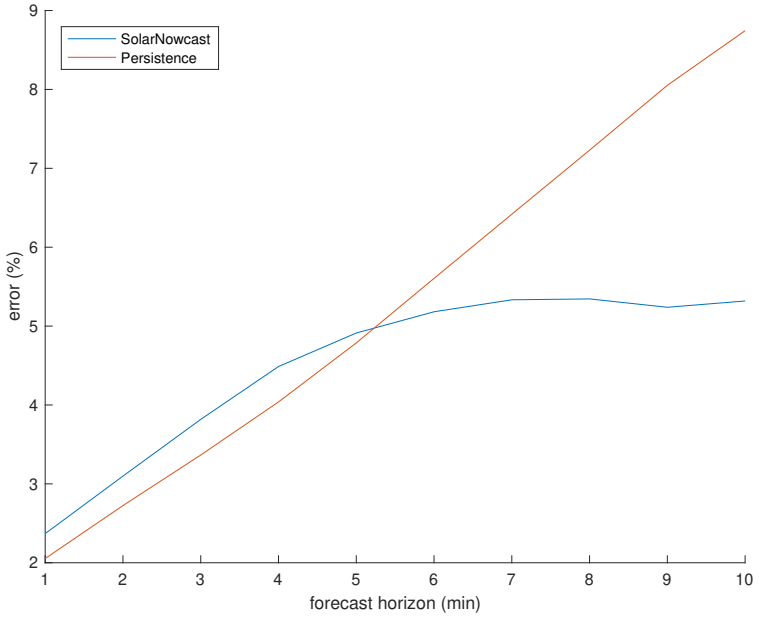

Figure 11: Evolution of the forecast error on the Clear Sky index according to time horizon on data of October 2013.

time integration of the equations governing atmospheric motion, especially for numerical weather prediction. However, the main issue of such semi-Lagrangian approach is that it induces a smoothing effect. This effect increases with the forecast horizon and becomes a serious drawback when the time horizon reaches 30 minutes. In some cases, the smoothing effect might even be problematic with a smaller forecasting horizon, especially if the sky is very cloudy and rapidly evolves. This is the case on the data acquired in January, 2013, as seen on Figure 13.

5. Conclusion

This deliverable describes and analyzes the forecast results obtained by Solar Nowcast, while the method was fully detailed in Deliverable D2.9 [1]. The efficiency of the method is evaluated according to two angles: the availability of a quasi real-time implementation and the quality of the forecasted results in comparison to a state-of-the-art method.

The first topic is a complete success thanks to the research work conducted in EoCoE. From the first mono-thread sequential code, the resulting parallelized multi-threads implementation raises a scaling efficiency of 54% with a forecast result obtained in less than 7 seconds. Moreover, few ideas are proposed further decreasing the computational time.

Concerning the quality of the results, Solar Nowcast gives more or less the same results than the persistence approach when the forecast horizon is less than 10 minutes. However, when the time horizon is between 10 and 20 minutes, Solar Nowcast provides usually better results. As expected, due to the chosen numerical schemes, the forecast quality decreases above 20 minutes. In case of a highly cloudy sky with fast evolution, Solar Nowcast reaches its limit even sooner, due to the underlying hypothesis, which are used for Motion Estimation and Forecasting and are not coherent with these physical conditions. In this particular situation, the use of the structure information included in

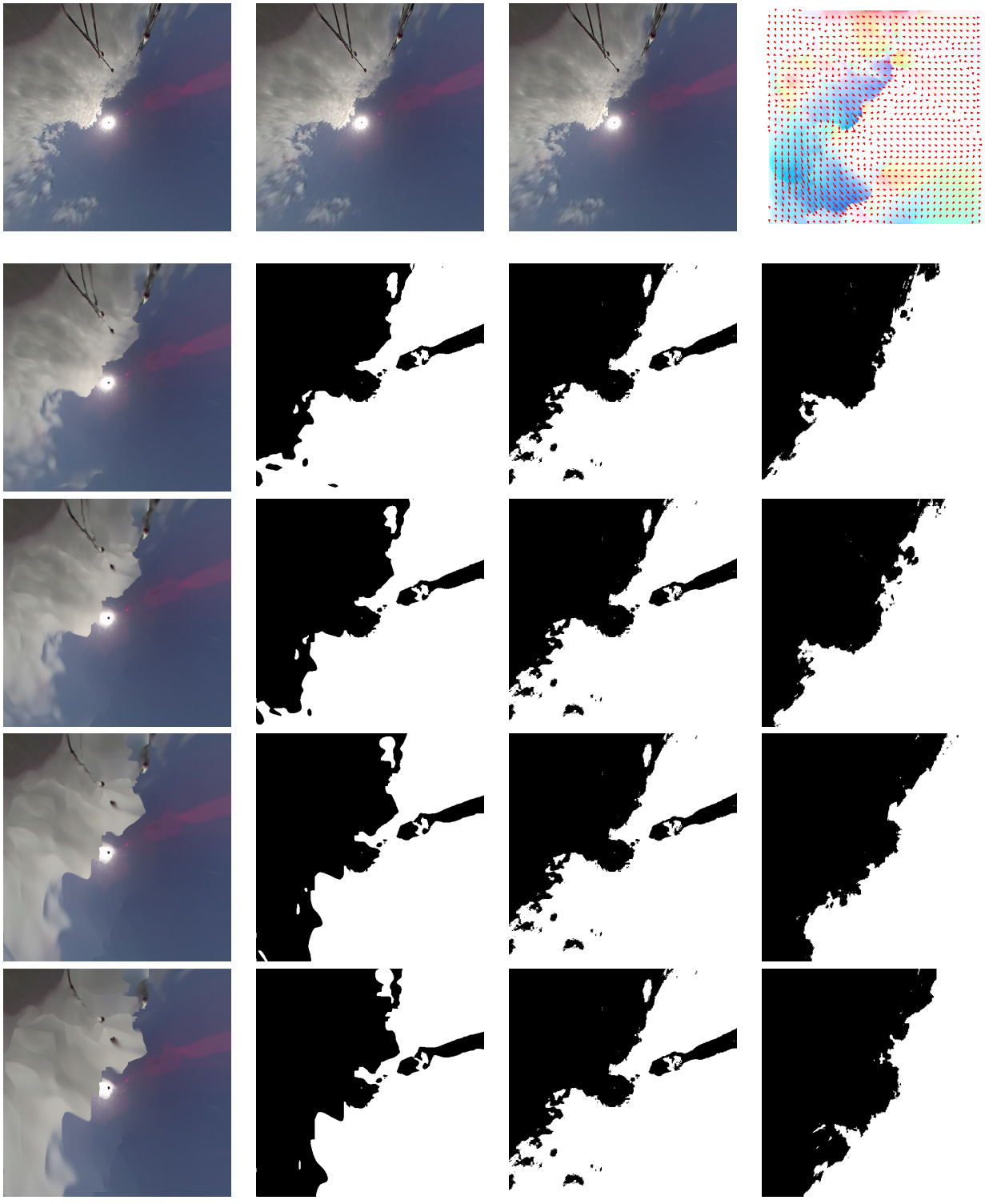

Figure 12: First line: three acquisitions of the October data set and the estimated motion. Remaining lines: forecasting results at horizons $h = 5$ min, 10 min, 20 min, 30 min; from left to right: forecasted image F obtained with Solar Nowcast, Clear Sky of F, Clear Sky image with the persistence method, Clear Sky image of ground truth.

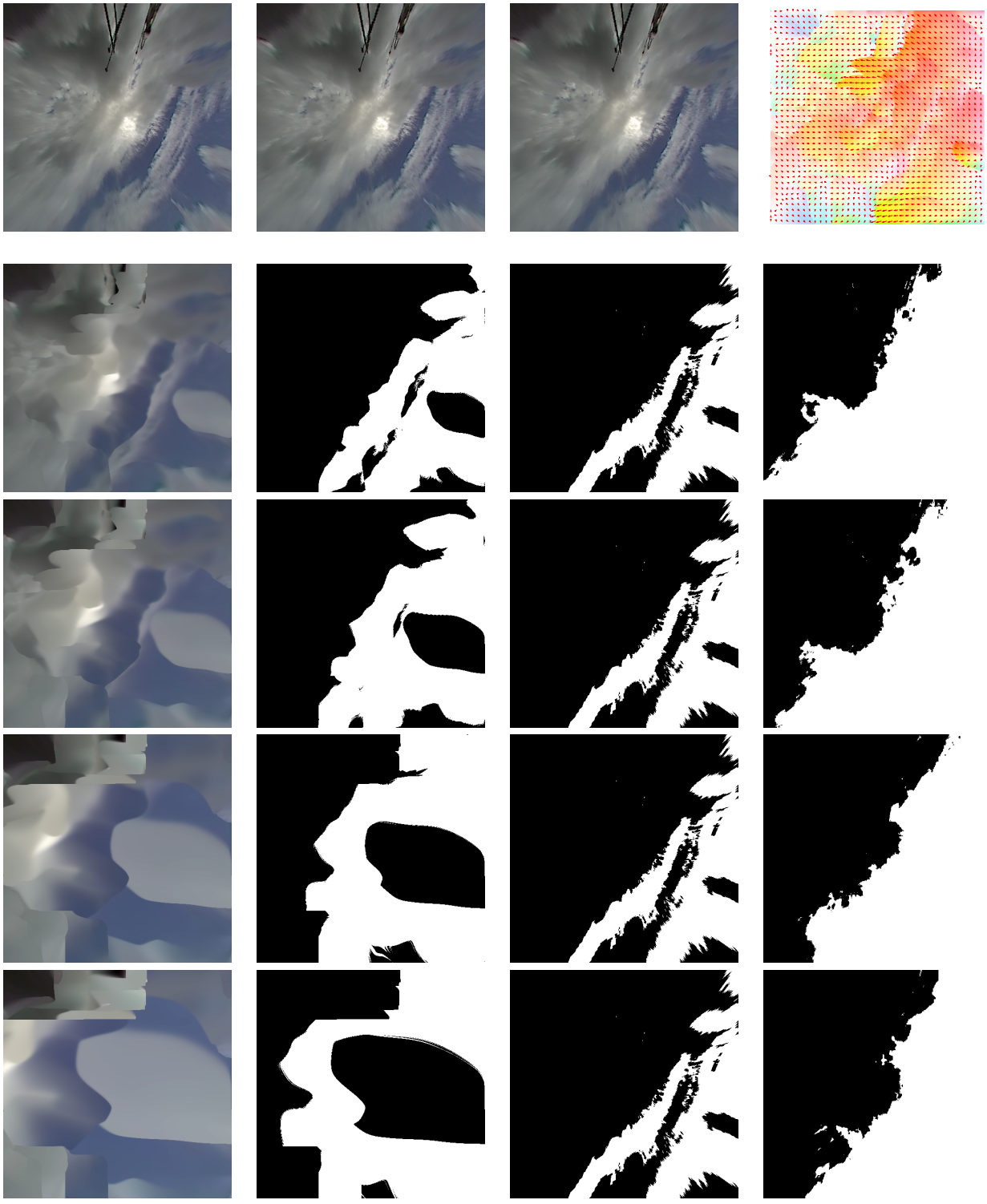

Figure 13: First line: three acquisitions of the January data set, and the estimated motion. Remaining lines: forecasting results at horizons $h = 5$ min, 10 min, 20 min, 30 min; from left to right: forecasted image F obtained with Solar Nowcast, Clear Sky of F, Clear Sky image using the persistence method, Clear Sky image of ground truth.

the Motion Estimation is no more pertinent and the choice of the semi-Lagrangian scheme in the forecast step degrades the numerous image structures.

Implicit schemes being known to have smoothing effects, literature should be investigated for alternatives. For instance, the Constrained Interpolation Profile methods [8] have the property to preserve discontinuities and consequently to better transport image structures, such as clouds, along time. But they are explicit schemes, submitted to CFL condition and requiring a small time step and an expensive computational time. Another possible improvement of the method is based on a multi-model approach: first a classification of events is done, for instance using machine learning techniques; second the best model according to the class of events (considering different hypothesis for Motion Estimation, and different implementations for Forecasting) is chosen, in order to obtain a better irradiance forecast.

References

[1] I. Herlin and D. Béréziat. GHI forecast at local scale. Technical Report Deliverable D2.9, EoCoE, 2018.

[2] L. Hascoët and V. Pascual. Tapenade 2.1 user's guide. Technical Report 0300, INRIA, 2004.

[3] M. Diagne, M. David, P. Lauret, J. Boland, and N. Schmutz. Review of solar irradiance forecasting methods and a proposition for small-scale insular grids. *Renewable and Sustainable Energy Reviews*, 27:65–76, 2013.

[4] C. Voyant and G. Notton. Solar irradiation nowcasting by stochastic persistence: a new parsimonious, simple and efficient forecasting tool. *Renewable and Sustainable Energy*, 2018. In press.

[5] S.P. Lloyd. Least squares quantization in pcm. *Transactions on information theory*, 28(2):129–137, 1982.

[6] Y. Cheng. Mean shift, mode seeking, and clustering. *Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, Aug 1995.

[7] C. Gauchet, P. Blanc, B. Espinar, B. Charbonnier, and D. Demengel. Surface solar irradiance estimation with low-cost fish-eye camera. In *Workshop on Remote Sensing Measurements for Renewable Energy*, May 2012.

[8] T. Yabe, F. Xiao, and T. Utsumi. The constrained interpolation profile method for multiphase analysis. *Journal of Computational Physics*, (169):556–593, 2001.