Applied Computing and Geosciences 11 (2021) 100064

Contents lists available at ScienceDirect

Applied Computing and Geosciences

journal homepage: www.sciencedirect.com/journal/applied-computing-and-geosciences

Efficiency of template matching methods for Multiple-Point Statistics simulations

Mansoureh Sharifzadeh Lari [a], Julien Straubhaar [b], Philippe Renard [b,*]

[a] Department of Electrical and Computer Engineering, University of Hormozgan, Bandarabbas, 7916193145, Iran
[b] Center of Hydrogeology and Geothermics, University of Neuchâtel, Rue Emile Argand 11, 2000, Neuchâtel, Switzerland

ARTICLE INFO

Keywords:
Multiple-point statistics
Template matching

ABSTRACT

Almost all Multiple-Point Statistic (MPS) methods use internally a template matching method to select patterns that best match conditioning data. The purpose of this paper is to analyze the performances of ten of the most frequently used template matching techniques in the framework of MPS algorithms. Performance is measured in terms of computing efficiency, accuracy, and memory usage. The methods were tested with both categorical and continuous training images (TI). The analysis considers the ability of those methods to locate rapidly and with minimum error a data event with a specific proportion of known pixels and a certain amount of noise.

Experiments indicate that the Coarse to Fine using Entropy (CFE) method is the fastest in all configurations. Skipping methods are efficient as well. In terms of accuracy, and without noise all methods except CFE and cross-correlation (CC) perform well. CC is the least accurate in all configurations if the TI is not normalized. This method performs better when normalized training images are used. The Binary Sum of Absolute Difference is the most robust against noise. Finally, in terms of memory usage, CFE is the worst among the ten methods that were tested; the other methods are not significantly different.

1. Introduction

Multiple-point statistics (MPS) is a flexible method for the simulation of complex geological patterns (Guardiano and Srivastava 1993). The approach overcomes some limitations of classical geostatistical simulation techniques based on two-point statistics (covariance or variogram). The principle of MPS is to employ a conceptual model described via one or more training data sets, often called the training image (TI), displaying the typical patterns that one would like to simulate and mimic. The approach consists of borrowing patterns from the TI to simulate random fields that consequently share some spatial features with the training data set. Many algorithms have been introduced in the last 20 years to extend the initial method of Guardiano and Srivastava (1993) such as SNESIM (Strebelle 2002), FILTERSIM (Zhang et al., 2006), SIMPAT (Arpat and Caers 2007), Direct Sampling (Mariethoz et al., 2010), IMPALA (Straubhaar et al., 2011; Straubhaar et al., 2013), and CCSIM (Tahmasebi et al., 2012). A detailed review of MPS methods is available in Mariethoz and Caers (2014).

A striking feature of almost all MPS algorithms is that they all need to compare patterns repeatedly during the simulation or TI analysis steps. A pattern, also called a template, is a subset of an image (Fig. 1). In the field of image processing, Template Matching (TM) methods are algorithms designed to locate the presence of a predefined template within a reference image as shown in Fig. 1. It is one of the most common techniques used in signal and image processing (Omachi and Omachi 2007). It is widely employed in many applications including object detection (Dufour et al., 2002; Pham et al., 2015), pattern recognition (Zeng 2011), video coding (Peng and Chen 2013), target tracking (Ipsen et al., 2020) and frequency response estimation (Bai et al., 2016). It is also used in geosciences for TI corrections (Straubhaar et al., 2019) and spatial machine learning algorithms based on high-order spatial statistics (Talebi et al., 2020). In general, the best match is found when a minimum distance or maximum correlation between a template and the reference image is identified. This requires the definition of a similarity measure.

The aim of this paper is therefore to analyze the performances of ten of the most used TM algorithms and to evaluate if these approaches could be used to enhance the efficiency of MPS algorithms. The analysis considers the ability of those methods to locate a data event with a specific proportion of known pixels and amount of noise rapidly and with minimum error. This is different from the usual and standard application of TM which usually considers a fully informed template.

* Corresponding author.
E-mail addresses: sharifzadehmn@gmail.com (M. Sharifzadeh Lari), julien.straubhaar@unine.ch (J. Straubhaar), philippe.renard@unine.ch (P. Renard).

https://doi.org/10.1016/j.acags.2021.100064
Received 5 April 2021; Received in revised form 9 July 2021; Accepted 18 July 2021
Available online 5 August 2021
2590-1974/

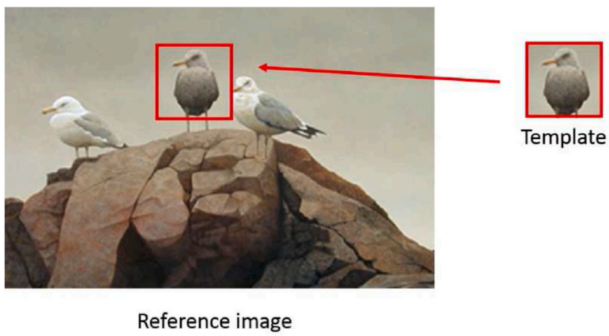

Fig. 1. Example of template matching procedure.

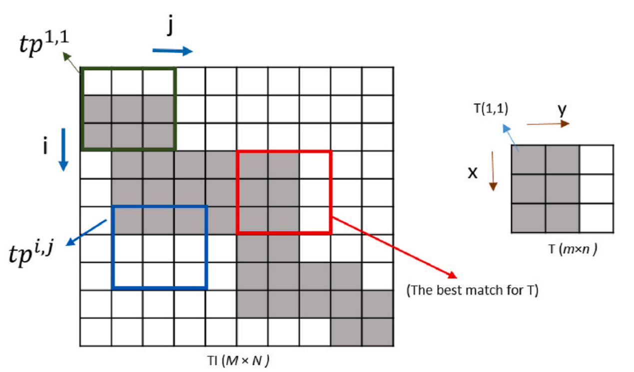

Fig. 2. $m \times n$ template (T), $M \times N$ Training Image and $tp^{i,j}$ (a pattern at location (i,j) of Training Image).

The reason for considering this situation is that often in MPS algorithms the simulation procedure is sequential, and templates are incomplete during the simulation. Noise can also occur because of possible discrepancies between conditioning data and the selected training data set. By studying the performances of different similarity measures in registering a template in an image, this paper provides indications about the applicability of TM techniques for locating the best match. Note also that the performances of the TM techniques are evaluated for categorical and continuous TIs. All techniques are tested for the same templates. Their efficiency and accuracy are studied as a function of template size, proportion of known pixels, and proportion of noise.

The paper is organized as follows. Section 2 describes the ten selected TM algorithms and the proposed method. Section 3 introduces the four TIs that are used, the procedures for constructing the templates, and for comparing the various TM algorithms. Section 4 presents all results in terms of computing efficiency and accuracy. Finally, the discussion and conclusions are in section 5. Note that notations for 2D images are used in the following for the sake of simplicity, however, each method can be straightforwardly extended to the 3D case.

2. Template matching methods

All TM methods are based on the definition of a (dis)similarity measure between a template and a part of an image. In this paper, we consider ten of the most frequently used TM methods.

The following measures between two patterns are considered. First, three dissimilarity measures are selected: the sum of absolute difference (SAD), the binary SAD (BSAD), and the sum of squared difference (SSD). Moreover, a skipping technique is also considered for these measures; it consists in stopping the computation early under some conditions. Second, the Cross-Correlation (CC), and its normalized version (NCC) are used as similarity measures. Third, we consider a more sophisticated approach named Coarse to Fine TM using Entropy (CFE) which consists of preselecting candidate patterns based on their entropy. Finally, the nearest neighbor (NN) technique is employed to find the best match. All these metrics and methods are described in detail in the following subsections.

Note that all the methods were implemented in Matlab with similar implementation details to allow a fair comparison of the methods and minimize the effect of the specific details of the implementations.

2.1. Definition and notation

In the presentation of the methods, we will use systematically the same notations. As shown in Fig. 2, T represents an $m \times n$ template. It is a matrix with m rows and n columns. T (x,y) represents the value of the pixel at location (x,y) within the template where x and y change from 1 to m and n respectively. The TI (or training data set) is a larger matrix of size $M \times N$. Within the TI, a training pattern $tp^{i,j}$ is a submatrix of size $m \times n$ such that its top-left corner is positioned at location (i,j)in the TI where i, j change from 1 to M-$m+1$ and N-$n+1$ respectively. As for the template, $tp^{i,j}(x,y)$represents the value of the pixel at location *(x,y)*

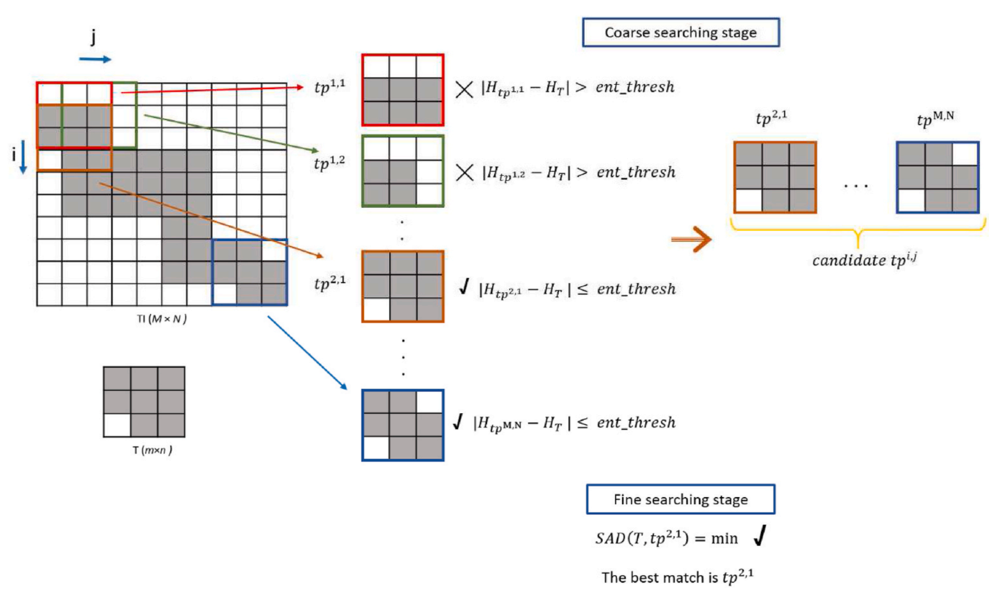

Fig. 3. The procedure of the CFE algorithm.

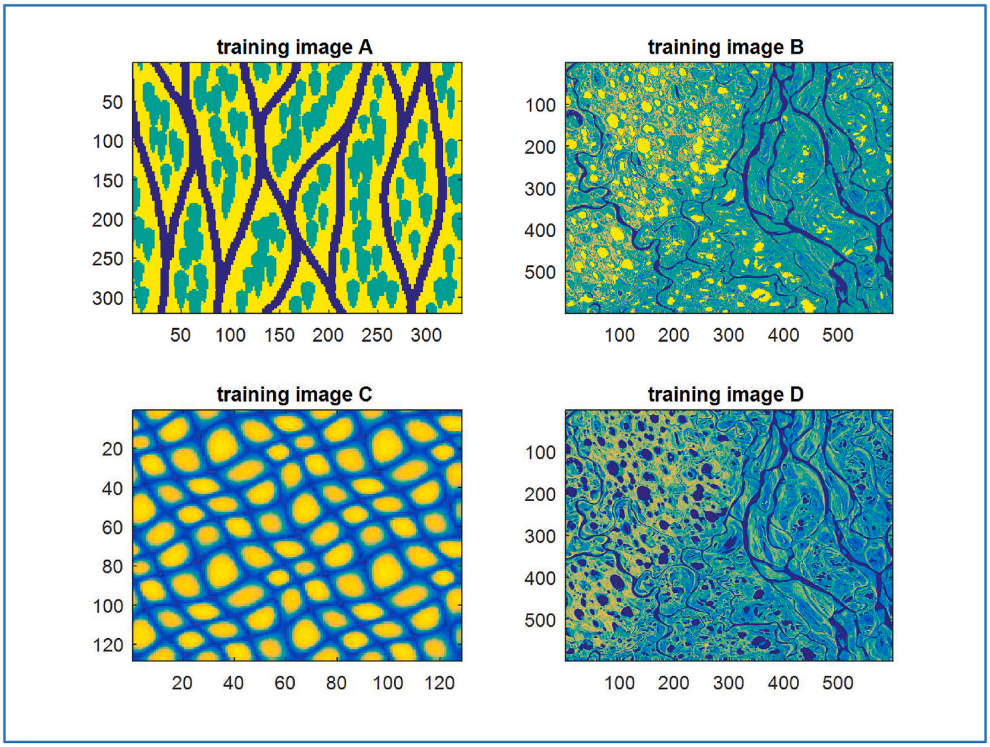

Fig. 4. The four training images.

Table 1
The characteristics of all training images used in this paper.

Training image name	Type	Size	Stationarity
A	Categorical with 3 categories	320 × 335	stationary
B	Categorical with 5 categories	599 × 598	Non stationary
C	Continuous	128 × 128	stationary
D	Continuous	599 × 598	Non stationary

within the $tp^{i,j}$.

2.2. SAD and BSAD

The sum of absolute distance *SAD* is defined as (Devijver and Kittler 1982):

$$SAD\left(T, tp^{i,j}\right) = \sum_{x=1}^{m} \sum_{y=1}^{n} \left| tp^{i,j}(x,y) - T(x,y) \right| \tag{1}$$

Its calculation requires $m \times n$ subtractions for each search location. MPS simulation methods such as SIMPAT and FILTERSIM use this similarity measure to find the best match.

A special kind of *SAD* named B*SAD* was also used in this study. *BSAD* is defined as

$$BSAD\left(T, tp^{i,j}\right) = \sum_{x=1}^{m} \sum_{y=1}^{n} \left(1 - \delta\left(tp^{i,j}(x,y) - T(x,y)\right)\right) \tag{2}$$

where

$$\delta(x) = \begin{cases} 1, & a = 0 \\ 0, & otherwise \end{cases} \tag{3}$$

BSAD consists in comparing two templates pixel by pixel and counting those that have a different value. This dissimilarity measure is typically used in MPS algorithms for simulating categorical variables (e.g., Direct Sampling).

2.3. SSD

SSD is defined as the sum of the squared difference between each pixel value of two templates (Burt 1982):

$$SSD\left(T, tp^{i,j}\right) = \sum_{x=1}^{m} \sum_{y=1}^{n} \left(tp^{i,j}(x,y) - T(x,y)\right)^2 \tag{4}$$

2.4. CC and NCC

Correlation is a measure of the degree to which two variables are linearly correlated. They do not necessarily have identical values but display similar behavior. In the signal processing scientific literature, CC is defined as a convolution product (Di Stefano, Mattoccia et al., 2005):

$$CC\left(T, tp^{i,j}\right) = \sum_{x=1}^{m} \sum_{y=1}^{n} tp^{i,j}(x,y) \cdot T(x,y) \tag{5}$$

In the MPS algorithm CCSIM, Tahmasebi et al. (Tahmasebi et al., 2012) use CC to find the best match. In statistics, the definition of the cross-correlation is slightly different and expressed in a normalized form

$$NCC\left(T, tp^{i,j}\right) = \frac{\sum_{x=1}^{m} \sum_{y=1}^{n} \left[tp^{i,j}(x,y) - \overline{tp} \right] \cdot \left[T(x,y) - \overline{T} \right]}{\sqrt{\sum_{x}^{m} \left[tp^{i,j}(x,y) - \overline{tp} \right]^2 \sum_{y}^{n} \left[T(x,y) - \overline{T} \right]^2}} \tag{6}$$

with $\overline{tp}$ the mean value of the pixels in the template $tp^{i,j}$, and $\overline{T}$ the mean value in the searched template.

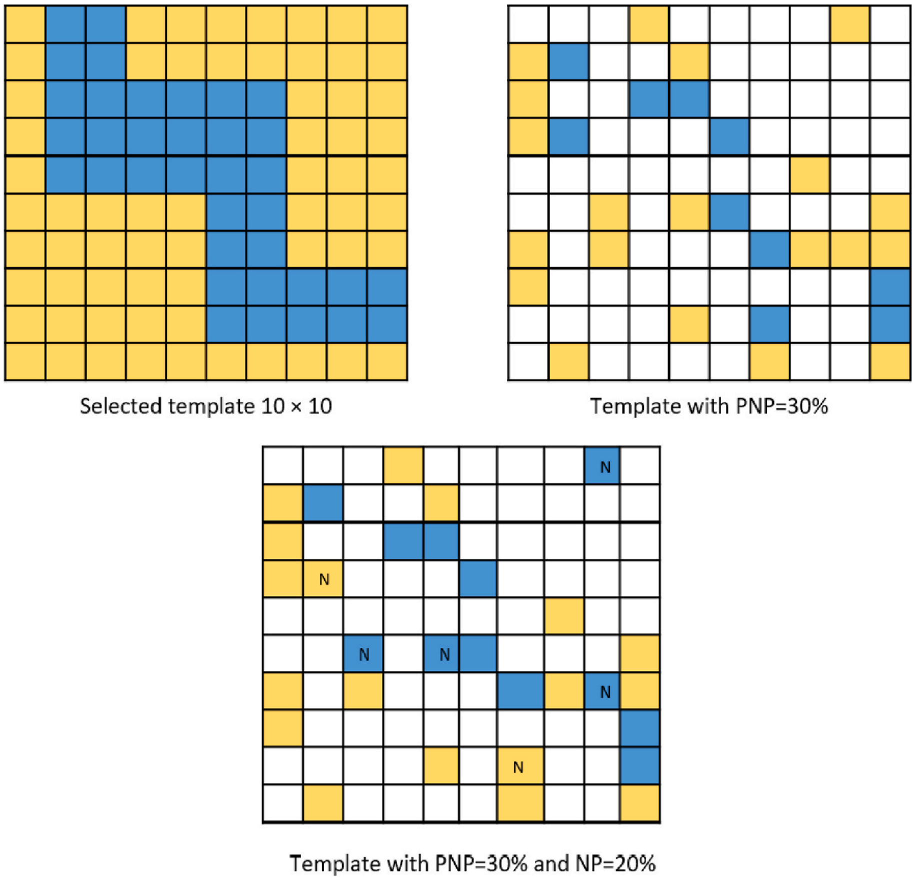

Fig. 5. Example of a selected template, template with PNP = 30 % and the same template with N = 20 % in addition.

2.5. Standard versus skipping method

The basic (or standard) TM algorithm consists in scanning the whole TI to extract the exhaustive set of training patterns $tp^{i,j}$ and comparing all of them with a reference pattern T using the dissimilarity or similarity measure. For SAD, BSAD, and SSD, the best match is obtained when the dissimilarity is minimum. For CC and NCC, it is obtained when the similarity is maximum.

As the comparison of two patterns is repeated very frequently in MPS algorithms, the efficiency of the TM technique is crucial. A classical approach to accelerate the TM is to skip the comparison of the current pattern with the reference one, i.e., prematurely terminate the computation of the dissimilarity as soon as it is found that this location cannot be the best match location (Kawanishi et al., 2003; Mahmood and Khan 2012; Zhang et al., 2012).

The method generally works as follows. First, a threshold *Skip_T* is initialized. The skipping algorithm stops the calculation of the dissimilarity measure, expressed as a sum, if the current value (a partial sum) is larger than the threshold. The candidate cannot be the best match and the algorithm jumps to the next $tp^{i,j}$. In this paper, we use an adaptive *Skip_T* which is discussed in section 3.4. The skipping method was implemented with SAD (skipping SAD), BSAD (skipping BSAD), and SSD (skipping SSD).

Note that another way to accelerate an MPS algorithm is to stop the scan of the training patterns as soon as the best match met so far is considered satisfactory (as can be expressed using an acceptance threshold).

2.6. CFE

Another more advanced set of *TM* methods are named *coarse to fine algorithms*. They are used in TM objects (Gharavi-Alkhansari 2001; Lai et al., 2009; Ma et al., 2009). In this paper, a method based on entropy differences is applied. The entropy of a system as defined by Shannon gives a measure of uncertainty about its possible states. Shannon's function is based on the concept that the information gained from an event is inversely related to its probability of occurrence. The entropy of an image is defined as (Gonzalez et al., 2004):

$$H = -\sum_{k=0}^{G-1} p_k log_2(p_k) \tag{7}$$

where G is the number of gray levels in the image and p_k is the probability associated with gray level k.

The CFE method works as follows. First, a threshold *ent_thresh* is initialized. In this paper, *ent_thresh* is selected by trial and error. Then the entropy of all training patterns $tp^{i,j}$ extracted from the *TI* is calculated and stored as H_{tp}. The search process starts at a coarse resolution. The entropy *HT* of the template T is calculated and the difference between *HT* and all the values H_{tp} is computed. If the difference satisfies equation (8), the corresponding $tp^{i,j}$ are the candidates for the best match and are listed ($Ctp^{i,j}$).

$$\left|H_{tp} - H_T\right| \leq ent\_thresh \tag{8}$$

At the fine searching stage, $SAD(Ctp^{i,j}, T)$ is used to find the exact match among all $Ctp^{i,j}$. If SAD is the same for some $Ctp^{i,j}$s, one of them is randomly selected as the best match. The procedure is illustrated in Fig. 3.

2.7. NN

The nearest neighbor algorithm assigns to a template (T) the $tp^{i,j}$ of its

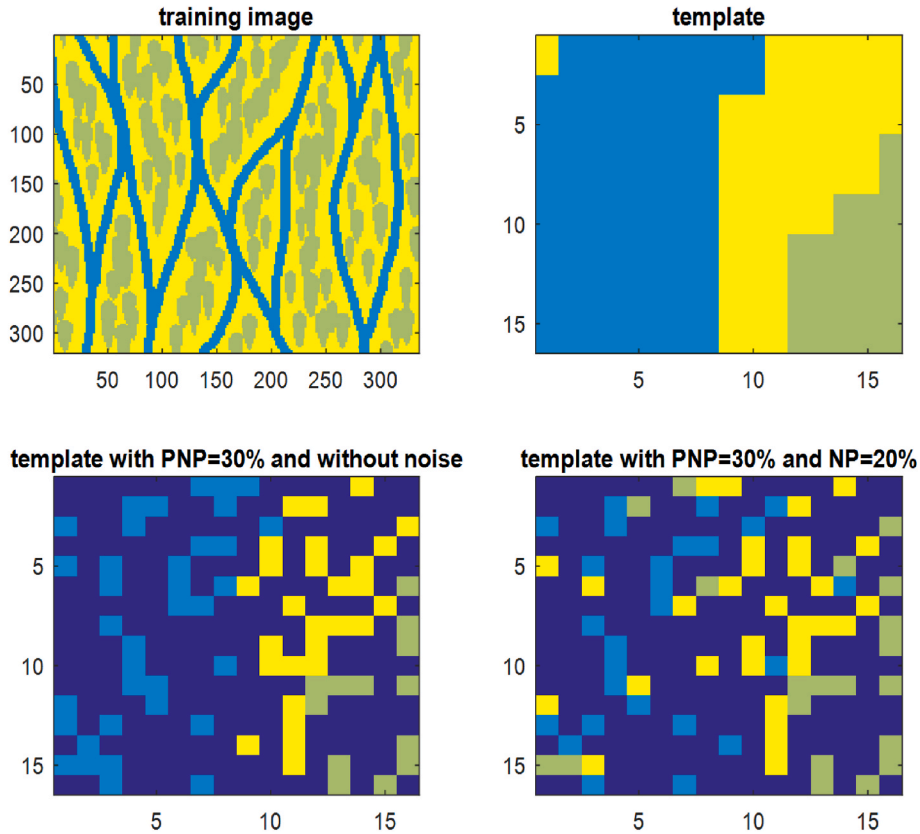

Fig. 6. A 320 × 335 categorical TI with 3 categories, a 16 × 16 template from the TI, the template with PNP = 30 % and without noise, and finally, a template with PNP = 30 % and NP = 20 %, (purple regions are unknown).

closest neighbor in TI. The aim is to find the similarity between the test pattern (T) and every pattern in the training set. In MPS applications, an N-dimensional nearest neighbor algorithm is needed while ordinary NN algorithms are not efficient for more than three dimensions. Zhou et al. present an idea called nearest neighbor convex hull (NNCH) for classification (Zhou et al., 2007). The convex hull (co) of a set $S\subset R^d$ is the smallest convex set containing set S: $co(S) = (\sum_{i=1}^{n}\alpha_i x_i | x_i \in S,\ \ \alpha_i \geq 0, \sum_{i=1}^{n}\alpha_i = 1, i = 1, 2, \ldots n)$. The convex hull of a set S is simply the set of all linear combinations of elements of S in which the coefficients of elements of S are non-negative and sum to 1. Such constrained linear combinations are known as convex combinations (Zhou et al., 2007). In this paper, we use *dsearchn* function for N-dimensional nearest point search in Matlab to perform the NN algorithm. *dsearchn* is based on Qhull that is a general dimension code for computing convex hulls (Barber et al., 1996).

3. Materials and methods

This research aims to compare the performances of TM methods for MPS simulation. For this purpose, we applied the ten TM methods described in the previous section on four different TIs (sec. 3.1) and using different templates. Sensitivity to the template size (TS), the proportion of known pixels (PNP), and the proportion of noise (NP) in the template are analyzed. Two performance indicators are calculated: computing efficiency (CE) and accuracy which is evaluated using an error proportion (EP). This section provides the details of the procedure outlined above.

3.1. The training images

Four TIs have been used. They are shown in Fig. 4 and Table 1 summarizes their properties. TI A contains 3 categories: channels, flood plains, and lenses in the middle of the flood plain. TI C and D are continuous images with integer values ranging between 0 and 255. TI D is a satellite image from the Lena River (from Landsat 7 image, USGS/EROS, and NASA Landsat Project), one of the largest rivers in the world. Finally, TI B is a discretized version of TI D having five categories. Note that TIs A and C are essentially stationary, meaning that the same patterns have identical probabilities to occur in any region of the TI while TIs B and D are not. Finally, note also that in some tests the original values within the TIs are used directly while in other tests they are normalized or centered (Normalized TI): the average value is subtracted to all pixel values.

3.2. The templates

After selecting a TI, square templates (T) with specific sizes are randomly picked from the TI. The template size dimensions (TS) are taken as predefined proportions - 5 %, 10 %, 15 %, and 20 % - of the mean length (TI_L) and width (TI_W) of the TI:

$$TS = round\left(\frac{a \times (TI_L + TI_W)}{2}\right) a = 0.05, 0.10, 0.15, 0.20 \tag{9}$$

In some TIs, like B and D in Fig. 4, the TI is very large, but it contains small objects, so we select $a = 0.025, 0.05, 0.10, 0.15$ instead.

Because many MPS algorithms are based on a sequential simulation procedure, the templates that the algorithm must search are often incomplete. At the beginning of the simulation, only a few conditioning points are known. During the simulation, the number of previously simulated nodes is increasing. Toward the end of the simulation, most of

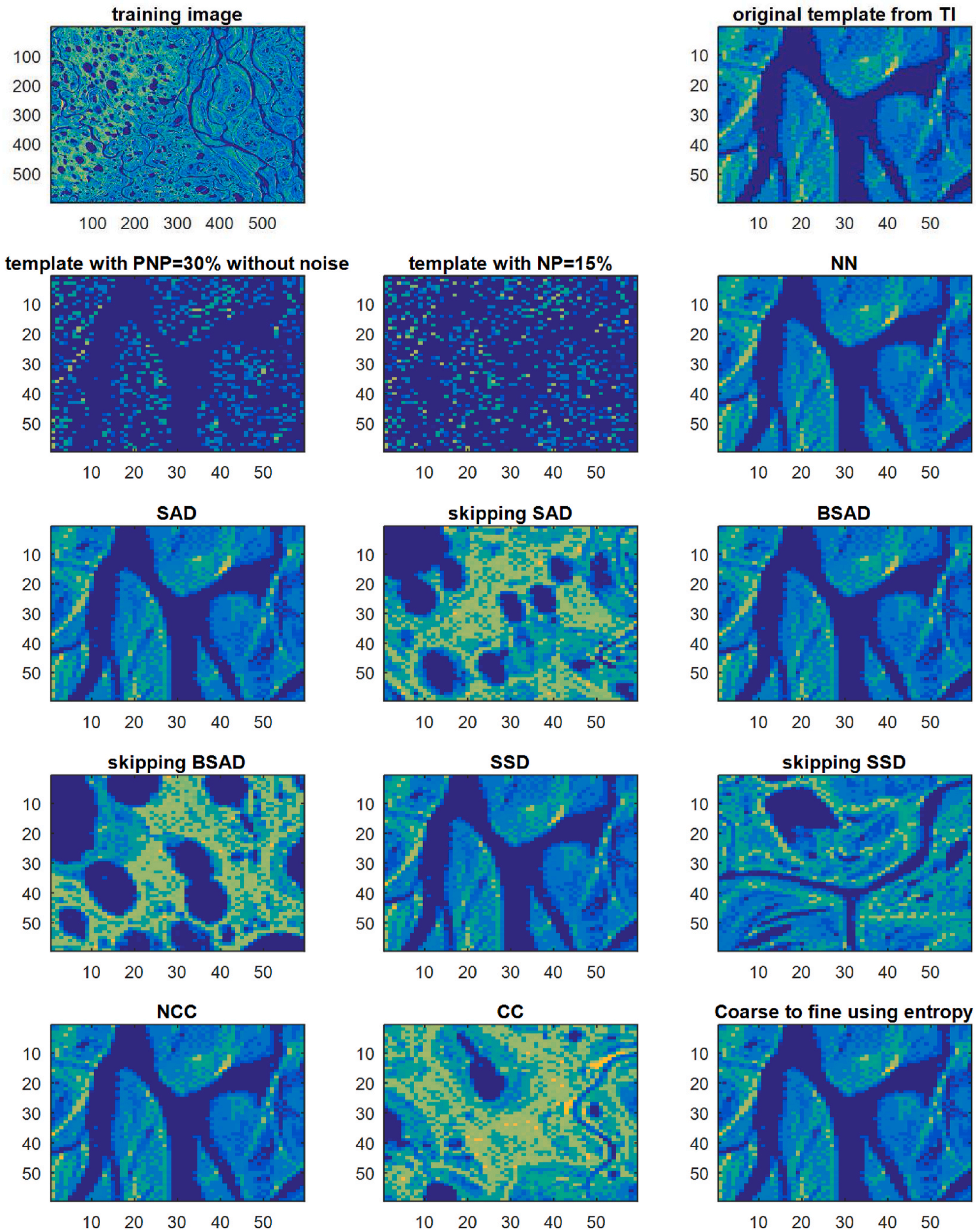

Fig. 7. An Example of simulation. TS = 59, PNP = 30, NP = 15, and the original shape of the TI is used.

the pixels are already known in the template. Therefore, we consider a proportion PNP of known pixels in *T*. It is taken systematically equal to 10 %, 20 %, 30 %, 40 %, and 50 %. The proportion of unknown pixels is $1 - PNP$. In practice, the location of the unknown pixel is randomly selected in T and their values are erased.

In addition, when we consider a sequential MPS simulation, the algorithms may accept some candidate values for a given pixel when the surrounding configurations are not exactly similar between the simulation grid and the TI. Therefore, some noise usually occurs in the simulation. As we are interested in the impact of this noise, we added noise to *T* by changing the value of some known pixels to test the robustness of the TM algorithms. The noise proportion *NP* is defined as the proportion of pixels whose value is perturbed. In our experiments, NP is taken equal to 0 %, 5 %, 10 %, 15 %, and 20 %. Given NP, we randomly select a proportion *NP* of pixels (rounded number) among the known ones and replace their value with a value randomly extracted from the TI.

This procedure is illustrated in Fig. 5. The TI A, a selected template with size 16×16, the template with $PNP = 30\%$, and the final template with $NP = 20\%$ is shown in Fig. 6.

3.3. Quality metrics

For each TI and candidate template, the TM procedures are applied. For each procedure, the best template is selected. Fig. 7 shows one example of this procedure for TI D. We see in this figure that the best template may be different depending on the TM technique employed. The results depend on the size of the template and the level of noise. Some visual examples are provided in the Appendix (Figs. 16–21). But to obtain a global understanding of the performances of the methods, we

Table 2
Summary of the main parameters used for the tests for calculating CE.

Test case	Training Image	PNP	NP	TS	*S_Thresh* for SAD and SSD	*S_Thresh* for BSAD	Changing parameter	Training image statues
1	A	30 %	0	16	3	2	TS	original
				32	12	6		
				48	28	14		
				64	49	25		
2	D	30 %	0	15	292	1	TS	original
				30	1161	5		
				60	4644	22		
				90	10,449	49		
3	C	10 %	0	30	7	2	PNP	original
		20 %			14	4		
		30 %			22	5		
		40 %			29	7		
		50 %			36	9		
4	D	30 %	0 %	30	1161	5	NP	original
			5 %					
			10 %					
			15 %					
			20 %					

Table 3
Summary of the main parameters used for the tests for calculating accuracy.

Test case	Training Image	PNP	NP	TS	*S_Thresh* for SAD and SSD	*S_Thresh* for BSAD	Changing parameter	Training image statues
5	D	30	0	15	292	1	TS	original
				30	1161	5		
				60	4644	22		
				90	10,449	49		
6	D	30	0	15	292	1	TS	normalized
				30	1161	5		
				60	4644	22		
				90	10,449	49		
7	A	30	0	16	3	2	TS	original
				32	12	6		
				48	28	14		
				64	49	25		
8	A	30	0	16	3	2	TS	normalized
				32	12	6		
				48	28	14		
				64	49	25		
9	D	10 %	0	30	361	2	PNP	original
		20 %			722	3		
		30 %			1084	5		
		40 %			1445	7		
		50 %			1810	8		
10	D	10 %	0	30	361	2	PNP	normalized
		20 %			722	3		
		30 %			1084	5		
		40 %			1445	7		
		50 %			1810	8		
11	D	20 %	0 %	30	722	3	NP	original
			5 %					
			10 %					
			15 %					
			20 %					
12	D	20 %	0 %	30	722	3	NP	normalized
			5 %					
			10 %					
			15 %					
			20 %					

compared their numerical efficiency and the quality of their results systematically using the following criteria.

EP is a measure of the quality or accuracy of the match. It quantifies the differences between the known reference template (REF) selected from the TI and the one selected by the TM method (SR). The difference is computed using the percentage of non-matching pixels (BSAD) as follow:

$$EP = \frac{BSAD(SR - REF)}{TS^2} \times 100 \tag{10}$$

EP is expressed in percent.

The computing time (t_c) is calculated for each algorithm separately. To have comparable values, all the algorithms were programmed in Matlab using similar structures for the codes. We then compare the methods using CE defined as:

$$CE_i = \frac{t_c^i}{\min\limits_{j=1,\dots,10}\left(t_c^j\right)}\ ,\ i = 1, \dots, 10 \tag{11}$$

where t_c^i is the computing time in second for the method i.

Table 4
CE for test case 1.

Methods	Template size				
	16	32	48	64	Mean
CFE	1	1	1	1	1
Skipping BSAD	2.7	3.3	3.5	3.4	3.4
Skipping SAD	2.7	3.4	3.5	3.4	3.4
Skipping SSD	2.7	3.4	3.5	3.4	3.4
NN	2.6	3.4	3.6	3.5	3.4
CC	3.0	3.9	4.1	4.0	3.9
BSAD	2.9	3.9	4.1	4.1	4.0
SAD	3.0	3.9	4.1	4.1	4.0
SSD	3.0	3.9	4.2	4.1	4.0
NCC	3.2	4.3	4.7	4.6	4.5

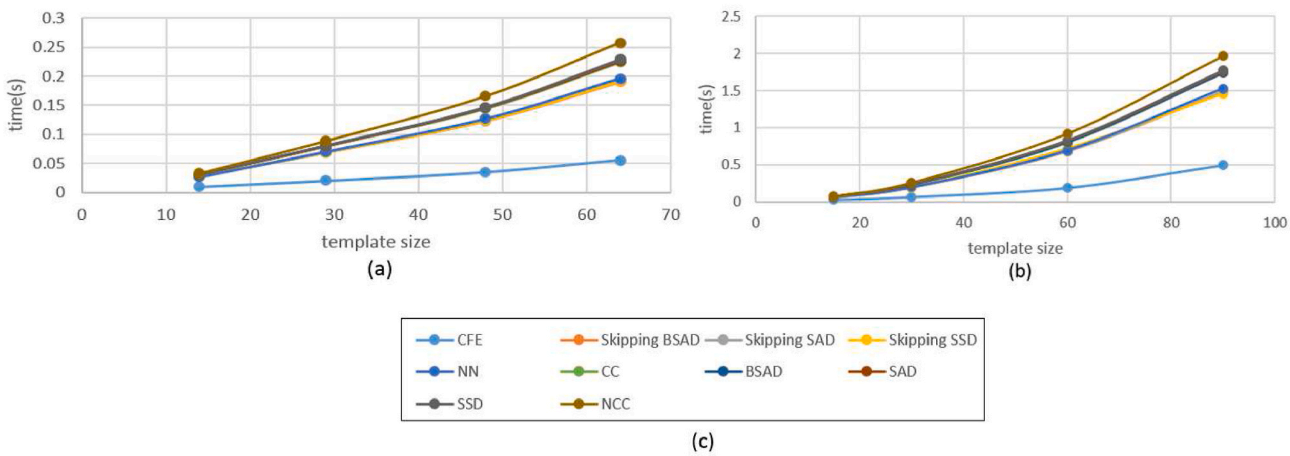

Fig. 8. Comparing the efficiency of TM methods. (a) Results of test case 1 with PNP = 30 and NP = 0, (b) Results for test case 2 with PNP = 30 and NP = 0, (c) Legend for figures a and b.

Table 5
CE for test case 3).

Methods	PNP					
	10	20	30	40	50	Mean
CFE	1.0	1.0	1.0	1.0	1.0	1.0
Skipping BSAD	3.5	3.3	3.0	2.6	2.6	2.9
NN	3.5	3.3	3.0	2.6	2.7	3.0
Skipping SSD	3.6	3.3	3.0	2.6	2.7	3.0
Skipping SAD	3.6	3.3	3.0	2.6	2.7	3.0
CC	3.7	3.6	3.5	3.1	3.3	3.4
SAD	3.7	3.6	3.5	3.1	3.3	3.4
SSD	3.7	3.6	3.5	3.2	3.3	3.4
BSAD	3.7	3.7	3.6	3.3	3.5	3.5
NCC	3.9	3.9	3.9	3.6	3.9	3.8

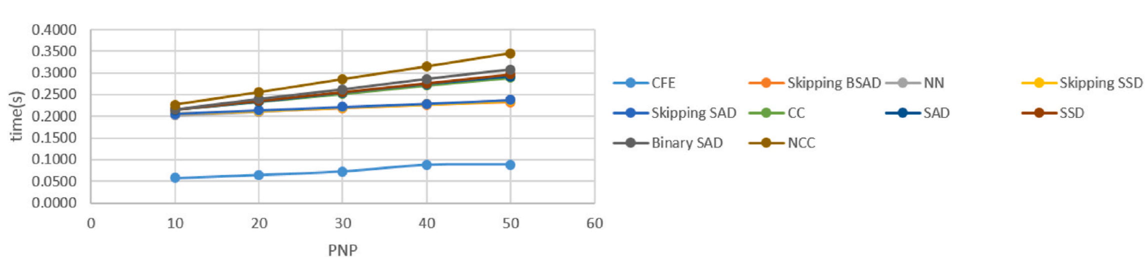

Fig. 9. The efficiency of the ten methods for test case 3.

Table 6
Efficiency results of test case 4.

Methods	NP				
	0 %	5 %	10 %	15 %	Mean
CFE	1.0	1.0	1.0	1.0	1.0
Skipping SSD	2.9	2.9	2.9	2.9	2.9
Skipping BSAD	2.9	2.9	2.9	2.9	2.9
NN	2.9	2.9	2.9	2.9	2.9
Skipping SAD	2.9	2.9	2.9	2.9	2.9
BSAD	3.4	3.3	3.3	3.3	3.3
CC	3.4	3.4	3.4	3.3	3.4
SAD	3.4	3.4	3.4	3.4	3.4
SSD	3.4	3.4	3.4	3.4	3.4
NCC	3.8	3.8	3.8	3.8	3.8

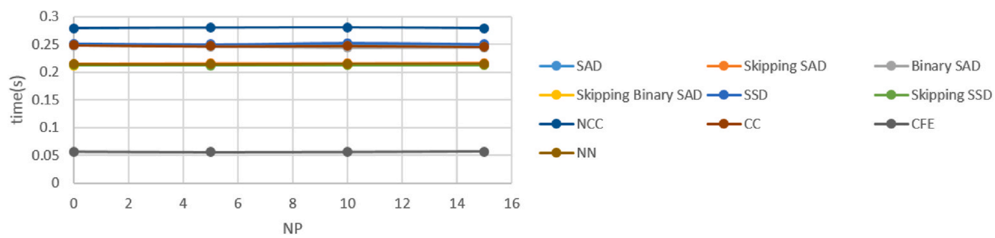

Fig. 10. Efficiency of the ten methods for test case 4.

Table 7
Average EP in percent for CC and CFE for test case 5. EP is zero for the other methods.

Methods	Template size			
	15	30	60	90
CC	99.9	95.9	62.0	55.9
CFE	43.9	0	0	0

For memory usage, we use the *memory* function in Matlab for comparing all methods.

3.4. General testing procedure

For every TI, and every template size, the TI is scanned, and all patterns are extracted. For the methods SAD, BSAD and SSD, the best match is easily identified by looking for the minimum distance between the searched template T and all patterns extracted from the TI.

For the skipping methods, the skipping threshold (*Skip_T*) must be set first. it is specified depending on the TIs, and the number of known pixels in the template. If *Skip_T* is small, the algorithms cannot find any match and if it is chosen too large, the benefit compared to the non-skipping version can be negligible or even null. In this paper, we use an adaptive $Skip_T$ defined as:

$$Skip\_T = \begin{cases} round(0.02 \times (max_{TI} - min_{TI}) \times nnp) & \textit{for skipping SAD and SSD} \\ round(0.02 \times nnp) & \textit{for skipping BSAD} \end{cases} \tag{12}$$

where max_{TI} and min_{TI} are the maximum and minimum values in TI respectively, and nnp is the number of known pixels in T. Like the

Table 8
Average EP in percent for CC and CFE for test case 6. EP is zero for the other methods.

Methods	Template size			
	15	30	60	90
CC	20.0	0	0	0
CFE	6.0	0	0	0

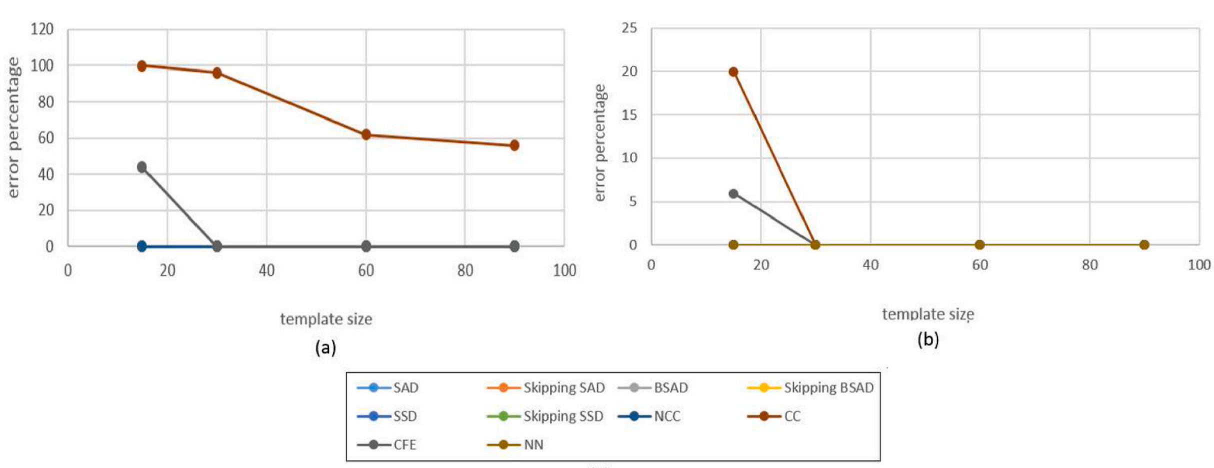

Fig. 11. (a) Results for test case 5 with the original TI, (b) Results for test case 6 with the normalized TI but normalized.

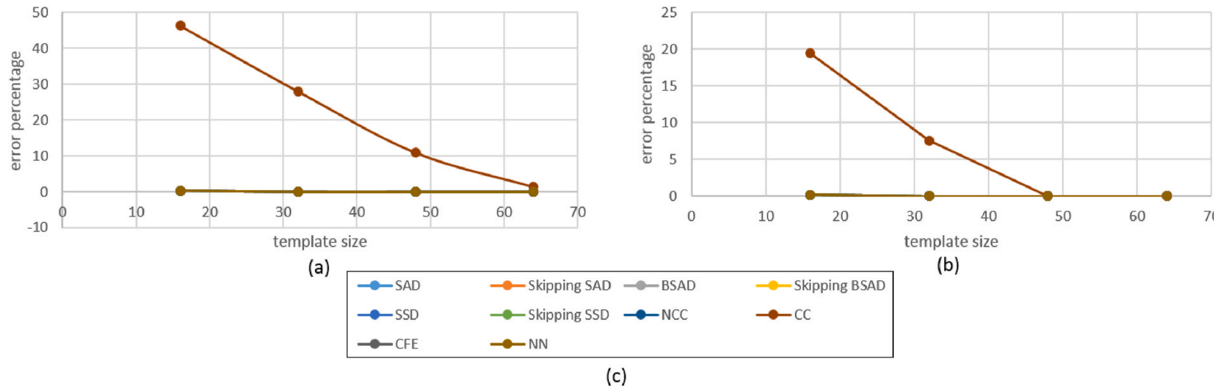

Fig. 12. (a) Results for test case 7 with the original TI, (b) Results for test case 8 with the normalized TI.

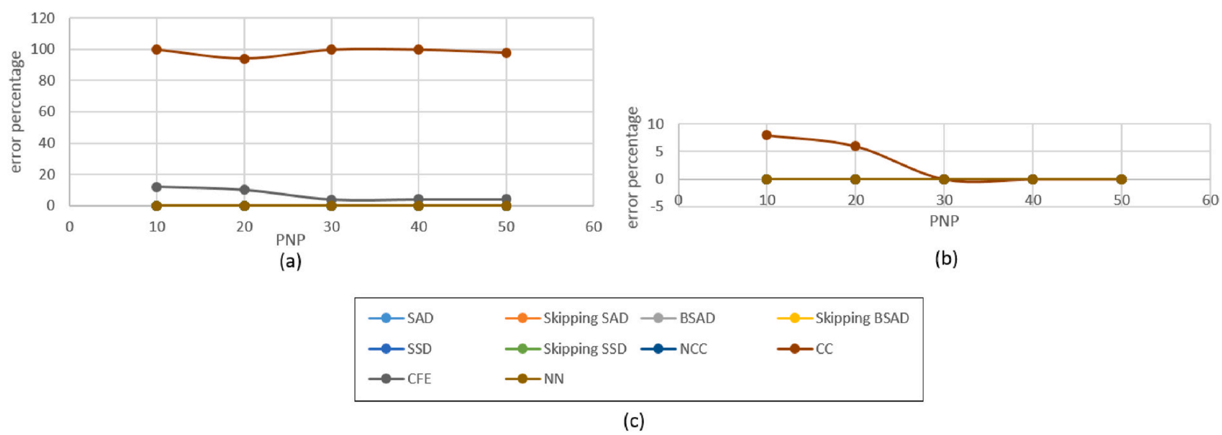

Fig. 13. (a) Results for test case 9, (b) Results for test case 10.

Table 9
Average EP in percent for the ten methods for test case 11.

Methods	NP				
	0 %	5 %	10 %	15 %	20 %
SAD	0.0	0.0	0.0	0.0	89.9
Skipping SAD	0.0	99.6	99.9	99.6	99.8
BSAD	0.0	0.0	0.0	0.0	0.0
Skipping BSAD	0.0	99.6	99.7	99.7	99.6
SSD	0.0	0.0	4.0	79.8	99.9
Skipping SSD	0.0	99.8	99.8	99.5	99.5
NCC	0.0	0.0	8.0	73.9	99.9
CC	97.8	99.9	97.9	99.9	99.9
CFE	0.0	0.0	0.0	0.0	89.9
NN	0.0	0.0	4.0	79.8	99.9

Table 10
Average EP in percent for the ten methods for test case 12.

Methods	NP				
	0 %	5 %	10 %	15 %	20 %
SAD	0.0	0.0	0.0	2.0	81.7
Skipping SAD	0.0	99.8	99.8	99.6	99.7
BSAD	0.0	0.0	0.0	0.0	0.0
Skipping BSAD	0.0	99.6	99.5	99.7	99.7
SSD	0.0	0.0	2.0	75.9	97.8
Skipping SSD	0.0	99.7	99.7	99.7	99.7
NCC	0.0	0.0	0.0	9.7	89.3
CC	0.0	4.0	4.0	29.9	91.4
CFE	0.0	0.0	0.0	2.0	81.7
NN	0.0	0.0	2.0	75.9	97.8

original algorithms, for the skipping methods, the best match is found for the minimum distance between T and all retained (i.e. not skipped) tp.

In CC and NCC, the algorithm was run to find the best match. Here the best match is indicated by the maximum correlation between T and all patterns from TIs.

For CFE, first, a threshold (*ent_thresh*) must be determined. In this paper, we use a fixed number 1.1 for *ent_thresh* which is obtained by trial and error.

For NN algorithm, we use *dsearchn* in Matlab to find the best match. Firstly, all *tps* are reshaped as a horizontal vector and stored in a matrix (M_{tp}). Vector form of the template (VT) is calculated and finally, $dsearchn(M_{tp}, VT)$ is used to find the best match.

The summary of the parameters used for all the test cases is shown in Table 2 and Table 3. All TM methods are applied 50 times on 50 different and randomly selected templates for each case and then average values of the performance criteria are calculated.

4. Results

In this section, we present the raw results of the comparison and focus first on the efficiency of the methods, then the quality of the matches, and finally memory usages.

4.1. Computing efficiency

The values of CE are compared for all the algorithms when changing TS, PNP and NP.

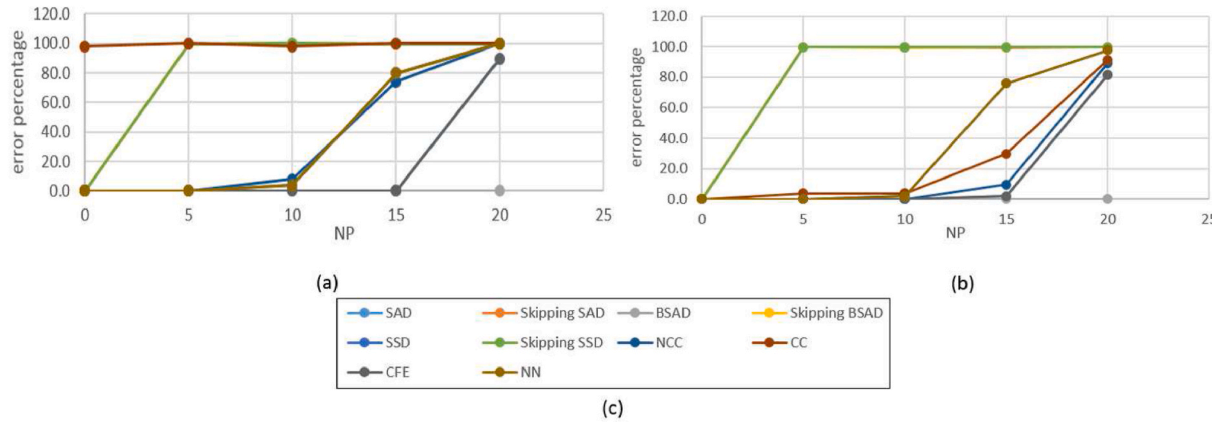

Fig. 14. (a) Results for test case 11, (b) Results for test case 12.

Table 11
Summary of the results obtained for all the ten methods and in all conditions.

	Criteria used to sort the TM methods from the best to the worst				
	In terms of CE with varying TS	In terms of CE with varying PNP	In terms of Accuracy with varying TS and PNP	In terms of Accuracy with varying NP for raw TI	In terms of Accuracy with varying NP for normalized TI
The best methods	CFE	CFE	All methods except CC and CFE	BSAD	BSAD
	Skipping BSAD	Skipping BSAD			
	Skipping SAD	NN			
	Skipping SSD	Skipping SSD			
	NN	Skipping SAD			
	CC	CC		SAD, CFE	CFE, SAD
	BSAD	SAD		NCC	NCC
	SAD	SSD		SSD, NN	CC
	SSD	BSAD	CFE	Skipping methods	NN, SSD
The worst methods	NCC	NCC	CC	CC	Sipping methods

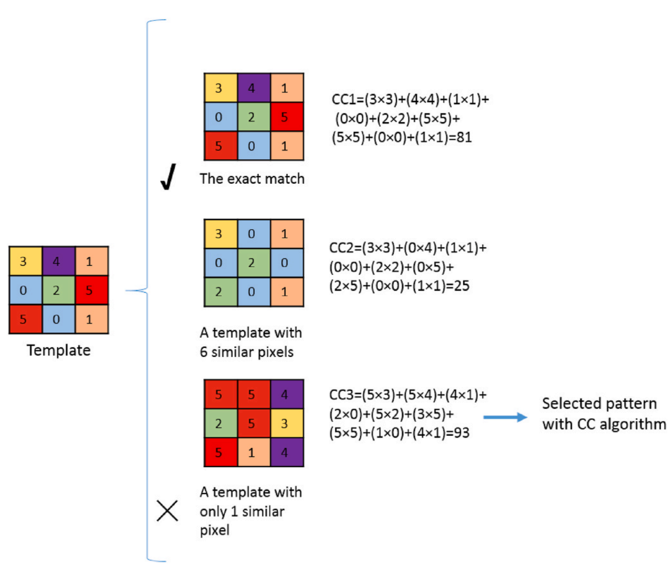

Fig. 15. Calculation of CC for finding the best match for the template. Although the first template is the best match, the CC algorithm selects the worst template as the best match.

4.1.1. *CE with varying TS*

Table 4 and Fig. 8a provide the results for test case 1, Fig. 16 in the Appendix shows one example of template from TI A and the simulation results for the ten methods.

Fig. 8 shows that the method CFE is the most efficient in all configurations and then skipping methods perform well. We note that computing time increases with TS, but not in a linear manner (Fig. 8). Computing time increases very slowly for CFE as compared to the other methods. These results showing the influence of TS on computing time are identical for all four TIs. For example, the results for test case 2 (non-stationary continuous TI) are shown in Fig. 8b. Normalizing the training images does not affect computing efficiency. In all the cases described in this part, *PN* is equal to zero because the purpose is to evaluate the efficiency of the algorithms as a function of *TS* only.

4.1.2. *CE with varying PNP*

Table 5 provides the detailed results for test case 3. Once again CFE is the most efficient for all configurations. Fig. 9 shows the computing time per iteration for the ten methods for test case 3. Results are identical for each TI. One can observe that the computing time increases with PNP. This is logical since the computation of the distance involves more pixels and takes more time.

4.1.3. *Computing efficiency with changing NP*

Table 6 provides the detailed results of test case 4 for the ten methods and Fig. 10 shows the computing time per iteration for all the methods for test case 4. Results show that changing NP and normalizing the TIs do not affect the computing efficiency. Results are the same for all the TIs.

4.2. *Accuracy*

The accuracy of the algorithms, evaluated with the percentage of error (EP) parameter, is compared when changing TS, PNP and NP for all TIs with or without normalization.

4.2.1. Accuracy with varying TS

Table 7 provides the average EP for CC and CFE for test case 5. All the other methods show no error at all. Fig. 11a shows that method CC is the least efficient in all configurations. All the other methods perform much better.

If a TI is normalized, the results change. Table 8 provides the detailed results of test case 6 in which a normalized TI is used. Fig. 11b shows that all methods perform better. For CC and CFE, the error is reduced with the increase in TS, and then it is zero for all methods. These results are the same for all four TIs. Results for the categorical TI A in the original (test case 7) and normalized (test case 8) shape are shown in Fig. 12a and Fig. 12b. Simulation results for a continuous TI when the values are shifted to be centered around 0 are shown in Fig. 19 in the Appendix, in this case, all methods can find a perfect match.

4.2.2. Accuracy with changing PNP

Fig. 13a shows the average EP for test case 9. When the TI values are not normalized, the cross-correlation technique is the least accurate in all configurations. NCC and CFE perform better when PNP increases. The other methods perform much better. When a normalized TI is used, results are much better. Fig. 13b shows the results for test case 10. These results are similar for all four TIs.

4.2.3. Accuracy with changing NP

Table 9 and Table 10 provide the detailed results for test cases 11 and 12. Fig. 14a shows that accuracy decreases when NP increases. But some methods like BSAD, SAD, and CFE (which uses the SAD criteria) are the most robust again noise. In this case, it is interesting to note that the skipping methods become less accurate than CFE even when they use the SAD or BSAD criteria. Fig. 14b shows that CC and NCC perform better when normalized TIs are used. These results are identical for all four TIs. An example of the results of a template search for all the ten methods for the original and normalized TI is shown in Fig. 7 and Fig. 21 in the appendix.

4.3. Memory usage

In terms of memory usage, CFE was the least efficient, and the other methods performed better. This is easily explained because, in all methods, the patterns with specific TSs must be extracted and saved. So, for all methods, the same memory was needed to store all patterns. In addition, CFE needs to store first the entropy values. It is one real number per pattern (*tp*) which is the entropy of *tp*. So, memory requirements are relatively small (less than 10 % differences).

5. Discussion and conclusion

This paper has reported the first template matching methods comparison for MPS simulation, aiming to help researchers to select the method that is the most appropriate for a given application. All the results are summarized in Table 11.

In terms of computing efficiency, the *coarse to fine with entropy* method is the fastest. This is because only one parameter is used during the coarse searching stage to select some templates as candidates for the best match. This initial selection allows saving a substantial amount of time, at a cost of only less than 10 % of additional memory. In the final stage SAD is used to search only among the candidate patterns instead of all patterns. A possible further research direction could be to analyze whether the use of the entropy is the best quantity to make this initial selection. One could argue that other statistical descriptors could be more efficient. Then, all the skipping methods perform better than the naive searches. By terminating soon, an obviously useless computation, a lot of time can be saved. These types of methods are implemented for example in the Direct Sampling algorithm (Mariethoz et al., 2010) and this paper shows that it is a reasonable choice. Note that in this paper, a fixed threshold depending on the TI and the template is involved in the skipping methods, allowing a fair comparison of the methods. A more in-depth analysis would be to include in the comparison a skipping method with an automated varying threshold set to the dissimilarity value of the best match met so far during the scan of the TI (and initialized to infinity).

We also observe that an increase in the proportion of known pixels causes a direct increase in computing time. When PNP increases, the number of known pixels increases and it directly impacts the number of calculations that must be done when comparing two patterns.

In the situation without noise, the cross-correlation criterion is the least accurate. As shown in Fig. 15, in some situations, CC cannot find the best match because the criterion is incomplete and it provides a larger value for matches that are worse than the exact best match. This shows that this method should be used only with normalized training images.

Concerning noise, BSAD is the most robust because when noise is present in a template, BSAD values are only reduced by one and the results are not strongly affected. The second-best techniques are SAD and CFE. Skipping methods are not robust against noise. The reason for this counterintuitive result is that sometimes the skipping threshold is reached wrongly because of the noise and we miss the best match.

Finally, CFE is the method that uses the largest amount of memory. It is the worst among the ten TM methods but the additional memory requirements are relatively small (less than 10 % differences). The other methods have almost identical memory requirements.

Before concluding, it is important to note that the results obtained in this work cannot be compared with more standard applications of TM in image analysis because here the templates are not completely known and there is often some noise in the template.

According to our research, for stationary TIs, if NP > 5, CC cannot reproduce a pattern properly, but the other methods do not have any problem in reproducing the TI patterns even for NP as large as 15. For non-stationary TIs, if NP $= 0$, all methods do the best but if NP > 0 CC cannot reproduce the patterns properly and if NP > 10, skipping methods (skipping SAD, skipping SSD, and skipping BSAD) do not work properly in reproducing proper objects from the TI but the other methods work properly.

The study presented in this paper deals only with univariate TIs, but in practice, it is frequent to employ multiple variables simultaneously either because one needs to simulate these variables jointly or because one or several variables are already known and used to constrain the simulation of the other variables. The study of TM efficiency should therefore be extended in future work to investigate the impact of the multivariate case. The main difference with the present work will be related to the way the distance between two patterns will be evaluated. If the distance is based on an average of the distances computed independently for each variable, the results will likely be rather close to the one obtained in this paper. If a more complex definition of the best multivariate template is used, then a detailed study will be required.

To conclude, this paper indicates that BSAD and CFE seem to be the best criteria for MPS simulations because these two methods are robust against noise, they are fast, and they allow retrieving accurately the proper template.

Author contributions

MS had the original idea, she designed and made the experiments, prepared the illustrations, and wrote the paper. PR and JS helped design the experiment, they participated in the analysis of the results and contributed to the writing of the manuscript.

Credit author statement

Mansoureh SharifzadehLari: Original idea, Designed and conducted the numerical experiments, Visualisation, Writing. Julien Straubhaar and Philippe Renard: Support for the desiring of the experiments, Participated in analysis of the results, Reviewing and Editing the manuscript

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Appendix

This appendix provides a set of figures illustrating some of the results. Fig. 16 shows for example that CC is not efficient for that specific situation, but the other methods perform better.

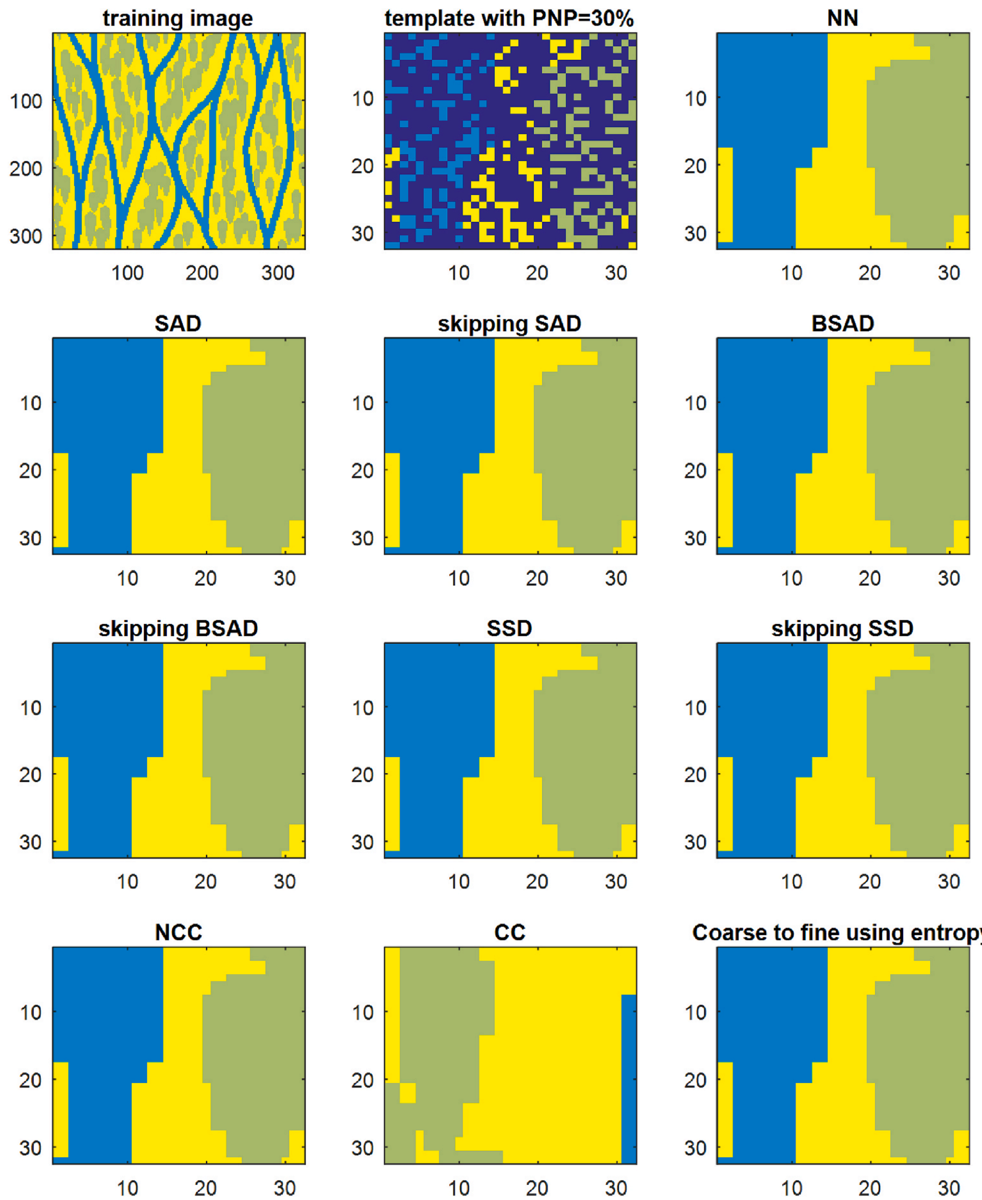

Fig. 16. An Example of results for training image A, template of size 32 × 32 with PNP = 30 %, and NP = 0 %.

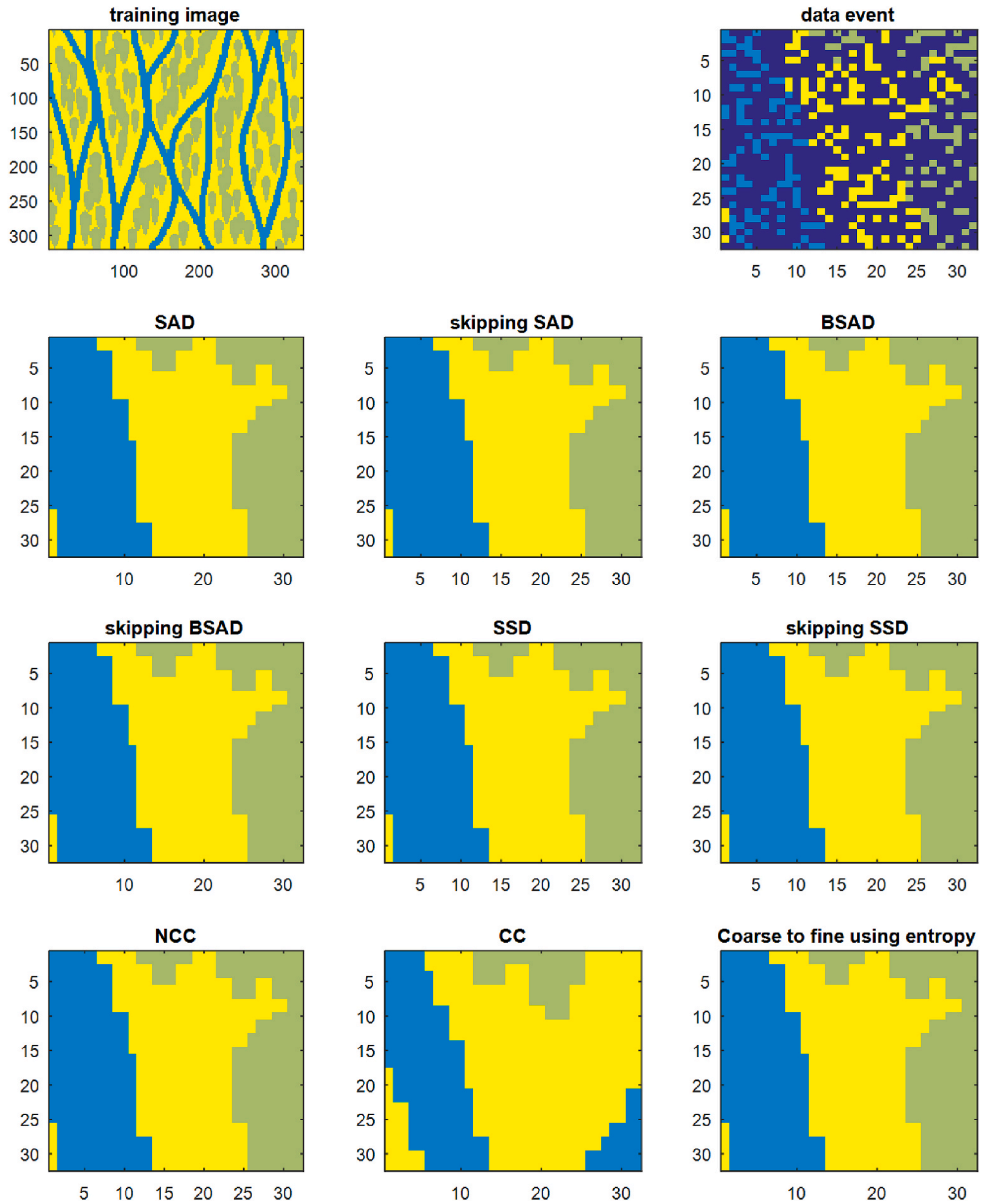

Fig. 17. An Example of results for training image A, template of size 32 × 32, with PNP = 30 %, and NP = 10 %.

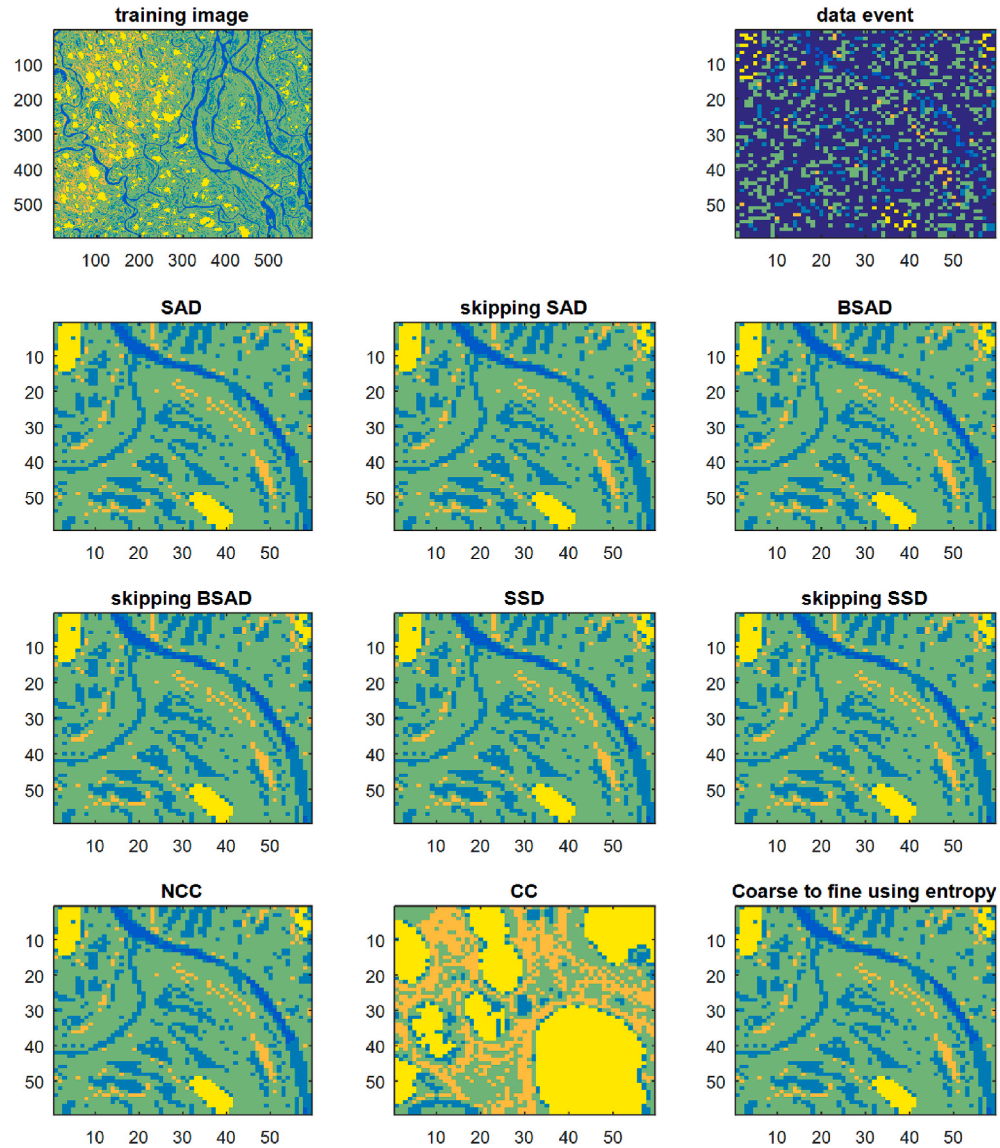

Fig. 18. An Example of results for training image B, template of size 59 × 59, with PNP = 30 %, and NP = 5 %.

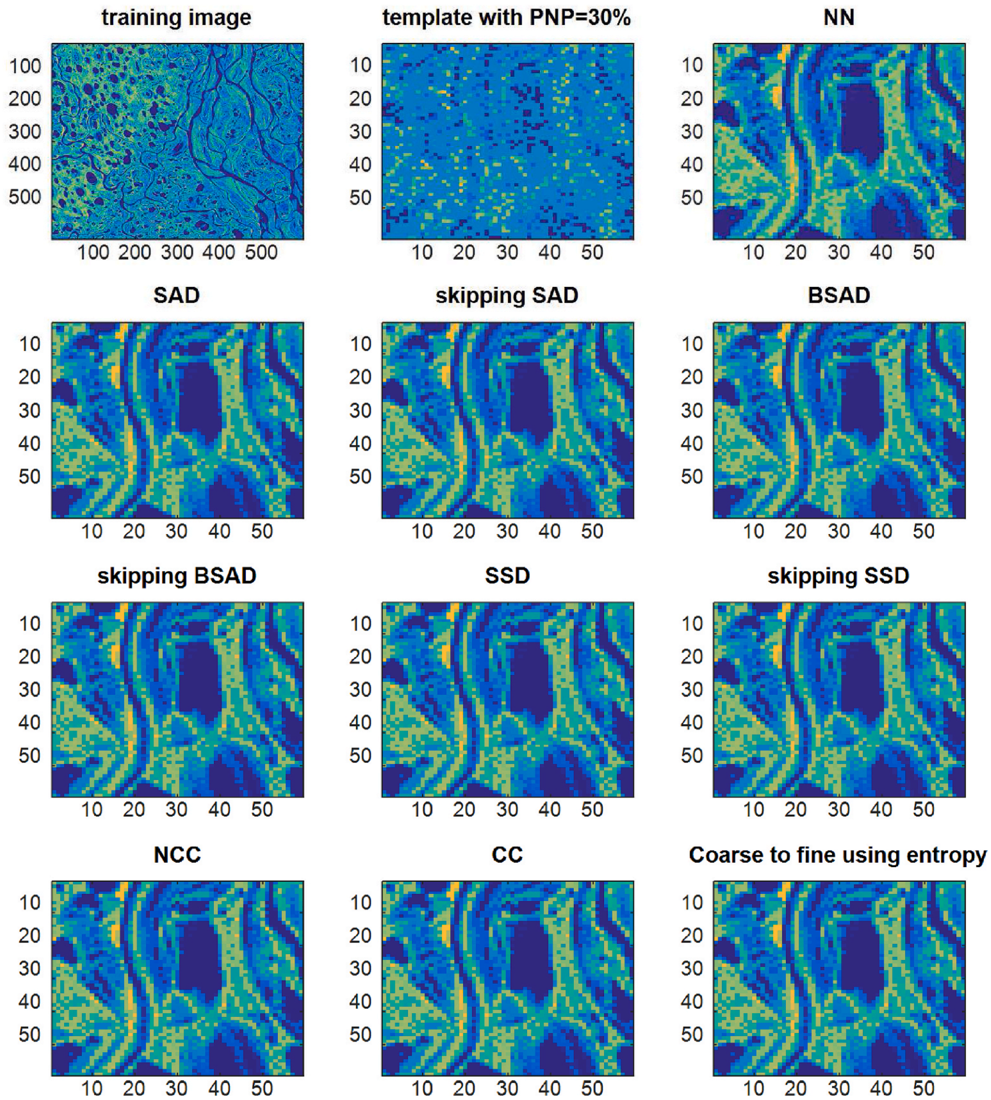

Fig. 19. An Example of results for training image C, template of size 59 × 59, with PNP = 30 %, and NP = 0 %.

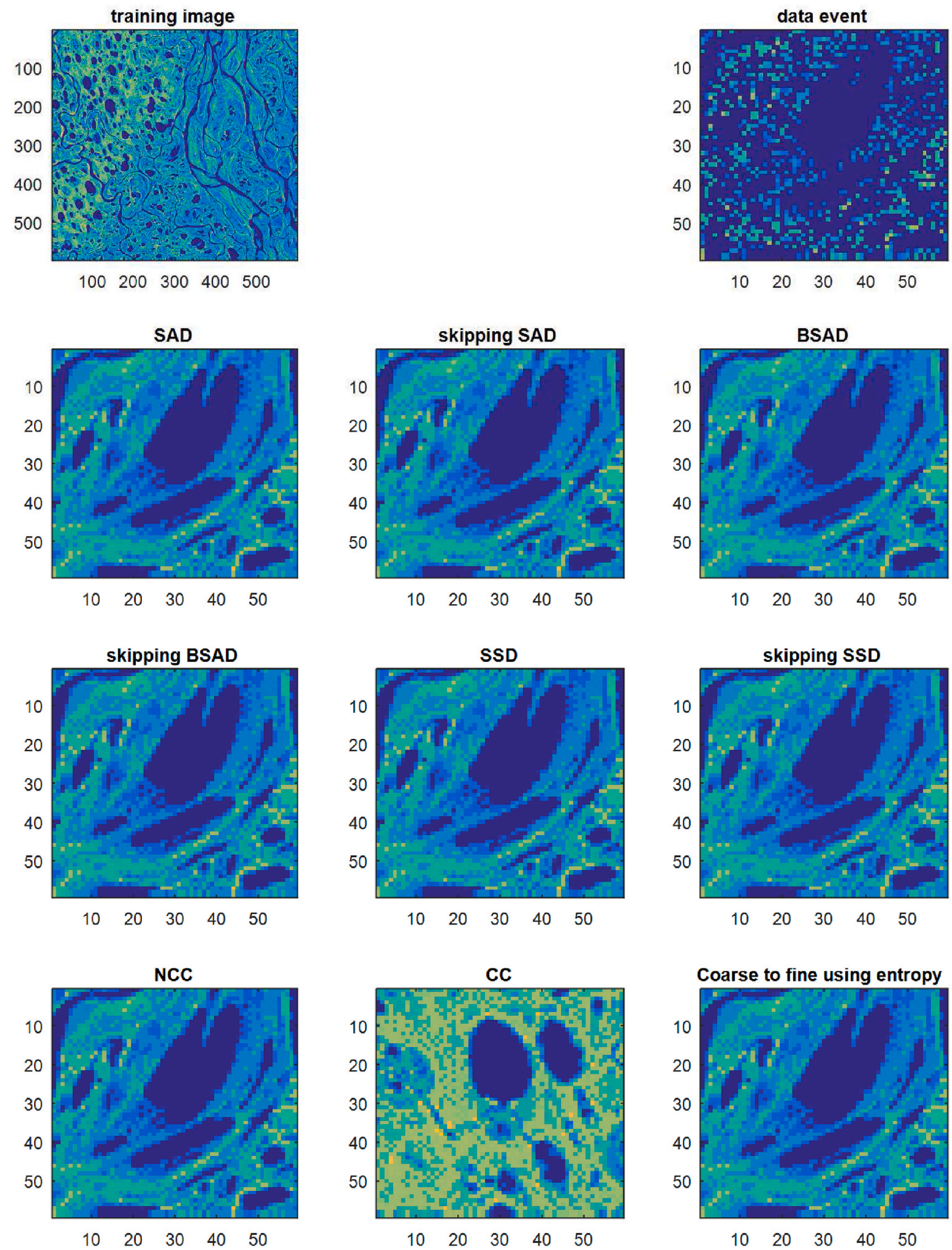

Fig. 20. An Example of results for training image C, template of size 59 × 59, with PNP = 30 %, and NP = 5 %.

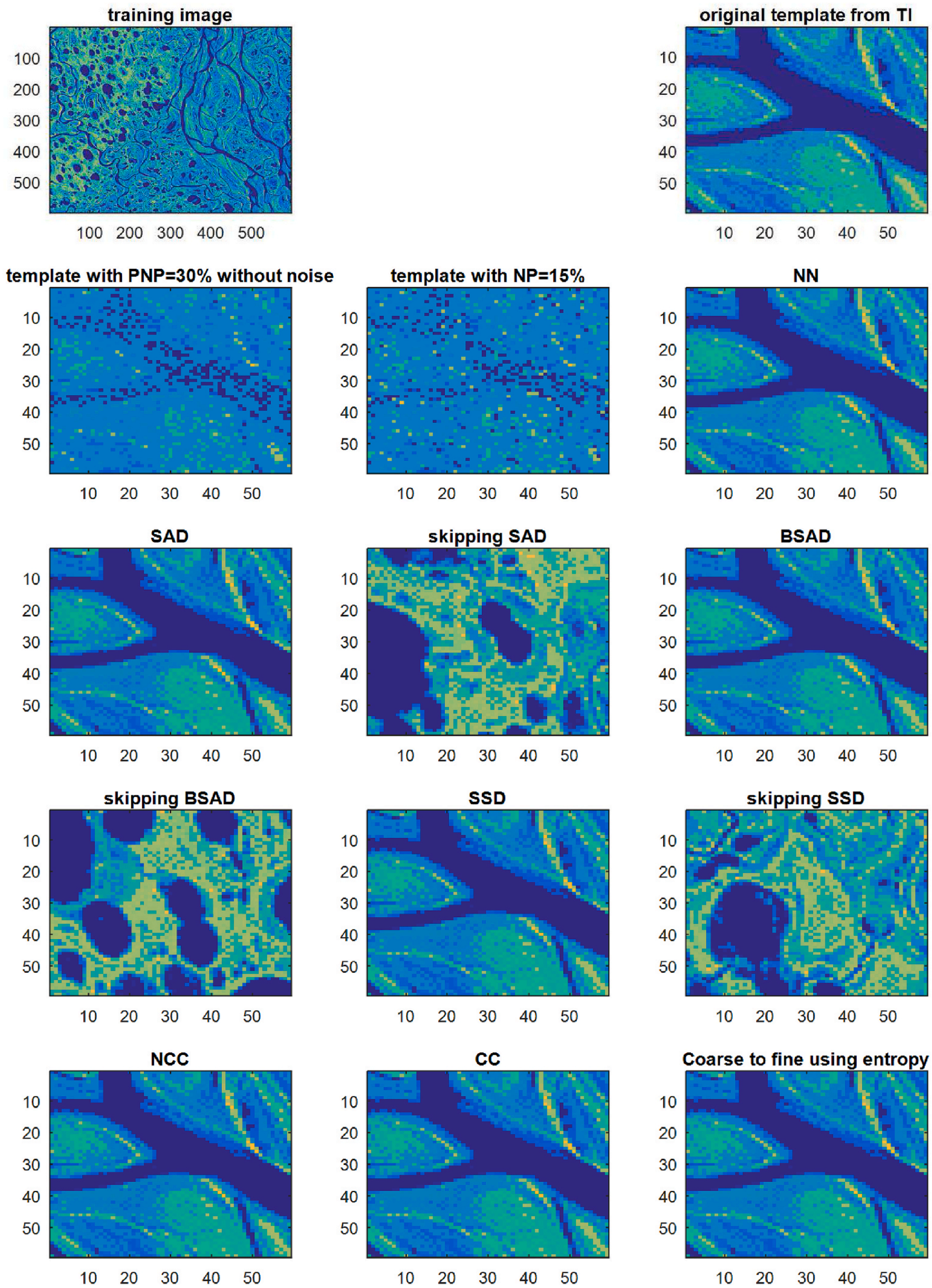

Fig. 21. An Example of results for training image C, template of size 59 × 59, with PNP = 30 %, and NP = 15 %.

References

Arpat, G.B., Caers, J., 2007. Conditional simulation with patterns. Math. Geol. 39 (2), 177–203.

Bai, F., Wang, X., Liu, Y., Liu, X., Xiang, Y., Liu, Y., 2016. Measurement-based frequency dynamic response estimation using geometric template matching and recurrent artificial neural network. CSEE J. Power Energy Syst. 2 (3), 10–18.

Barber, C.B., Dobkin, D.P., Huhdanpaa, H., 1996. The quickhull algorithm for convex hulls. ACM Trans. Math Software 22 (4), 469–483.

Burt, P.J., 1982. Local correlation measures for motion analysis : a comparative study. In: Proc. Pattern Recognition and Image Processing Conf. Las Vegas, 1982.

Devijver, P., Kittler, J., 1982. Pattern Recognition: A Statistical Approach. Prentice/Hall International." Hemel Hemstead, Hertfordshire, UK.

Di Stefano, L., Mattoccia, S., Tombari, F., 2005. ZNCC-based template matching using bounded partial correlation. Pattern Recogn. Lett. 26 (14), 2129–2134.

Dufour, R.M., Miller, E.L., Galatsanos, N.P., 2002. Template matching based object recognition with unknown geometric parameters. IEEE Trans. Image Process. 11 (12), 1385–1396.

Gharavi-Alkhansari, M., 2001. A fast globally optimal algorithm for template matching using low-resolution pruning. IEEE Trans. Image Process. 10 (4), 526–533.

Gonzalez, R.C., Eddins, S.L., Woods, R.E., 2004. Digital Image Publishing Using MATLAB. Prentice Hall.

Guardiano, F.B., Srivastava, R.M., Soares, A., 1993. Multivariate geostatistics: beyond bivariate moments. Geostatistics Tróia '92. In: Dordrecht, Springer Netherlands, ume 1, pp. 133–144.

Ipsen, S., Böttger, S., Schwegmann, H., Ernst, F., 2020. Target tracking accuracy and latency with different 4D ultrasound systems – a robotic phantom study. Curr. Direct. Biomed. Eng. 6 (1).

Kawanishi, T., Kurozumi, T., Takagi, S., Kashino, K., 2003. Skipping template matching guaranteeing same accuracy as exhaustive search. Proc. of ICAPR.

Lai, W., Wang, W., Huang, B., Ren, H., Shi, T., 2009. Fast template matching based on local entropy difference. 2009. In: 2nd International Congress on Image and Signal Processing.
Ma, L., Sun, Y., Feng, N., Liu, Z., 2009. Image fast template matching algorithm based on projection and sequential similarity detecting. 2009. In: Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing.
Mahmood, A., Khan, S., 2012. Correlation-coefficient-based fast template matching through partial elimination. IEEE Trans. Image Process. 21 (4), 2099–2108.
Mariethoz, G., Caers, J., 2014. Multiple-point Geostatistics: Stochastic Modeling with Training Images. John Wiley & Sons.
Mariethoz, G., Renard, P., Straubhaar, J., 2010. The Direct Sampling method to perform multiple-point geostatistical simulations. Water Resour. Res. 46 (11).
Omachi, S., Omachi, M., 2007. Fast template matching with polynomials. IEEE Trans. Image Process. 16 (8), 2139–2149.
Peng, W., Chen, C., 2013. An interframe prediction technique combining template matching prediction and block-motion compensation for high-efficiency video coding. IEEE Trans. Circ. Syst. Video Technol. 23 (8), 1432–1446.
Pham, I., Jalovecky, R., Polasek, M., 2015. Using template matching for object recognition in infrared video sequences. 2015. In: IEEE/AIAA 34th Digital Avionics Systems Conference (DASC).
Straubhaar, J., Renard, P., Mariethoz, G., Chugunova, T., Biver, P., 2019. Fast and interactive editing Tools for spatial models. Math. Geosci. 51 (1), 109–125.
Straubhaar, J., Renard, P., Mariethoz, G., Froidevaux, R., Besson, O., 2011. An improved parallel multiple-point Algorithm using a list approach. Math. Geosci. 43 (3), 305–328.
Straubhaar, J., Walgenwitz, A., Renard, P., 2013. Parallel multiple-point statistics algorithm based on list and tree structures. Math. Geosci. 45 (2), 131–147.
Strebelle, S., 2002. Conditional simulation of complex geological structures using multiple-point statistics. Math. Geol. 34 (1), 1–21.
Tahmasebi, P., Hezarkhani, A., Sahimi, M., 2012. Multiple-point geostatistical modeling based on the cross-correlation functions. Comput. Geosci. 16 (3), 779–797.
Talebi, H., Peeters, L.J.M., Mueller, U., Tolosana-Delgado, R., van den Boogaart, K.G., 2020. Towards geostatistical learning for the geosciences: a case study in improving the spatial awareness of spectral clustering. Math. Geosci. 52 (8), 1035–1048.
Zeng, Y., 2011. Pattern recognition using rotation-invariant filter-driven template matching. In: 2011 18th IEEE International Conference on Image Processing.
Zhang, Q., Gao, H., Kang, Z., 2012. Fast template matching with partial skipping using sub-template. 2012. In: International Conference on Audio, Language and Image Processing.
Zhang, T., Switzer, P., Journel, A., 2006. Filter-based classification of training image patterns for spatial simulation. Math. Geol. 38 (1), 63–80.
Zhou, X., Jiang, W., Yang, J., 2007. Kernel nearest neighbor convex hull classification algorithm. J. Image Graph. 7, 1209–1213.