3
Denoising by random walks

In this chapter the feature preserving filter for vector fields based on random walks, our first contribution, is presented (21). In Section 3.1 the concepts of random walks in graphs are discussed in the context of Markov processes. Then in Section 3.2 these concepts are applied to build the filter. In Section 3.3 implementation details are discussed together with a brief suggestion on how the parameters can be chosen and the results are discussed at the end. This chapter follows the notation presented by Sun *et al.* (30).

3.1
Random walk

Random walk (RW) was one of the first chance-processes studied in the theory of probability and has gained a lot of attention in several areas in visual computing. The name random walk is used because one may think of it as being a model for an individual walking on a straight line who at each point of time either takes one step to the right with probability p or one step to the left with probability $1 - p$, for example.

Given a graph and a starting node, one selects one of its neighbor at random and moves to this neighbor then selects a neighbor of this node at random and moves to it and so on. This sequence of nodes selected randomly this way is a random walk on the graph. In Section 3.2 we see that the denoising method to be proposed applies random walks on a graph whose nodes are the base points of the vector field, and whose links represents the neighborhood relation between them. To do such random walk, a probability has to be assigned to each edge on the graph and this represents the chance to move from a vertex to its adjacent neighbor through an edge. In fact a Random walk on graph is a very special case of a Markov process (16).

A *Markov process* is a sequence of possibly dependent random variables $(X_1, X_2, X_3, \ldots)$ identified by increasing values of their index, commonly time. Its main property is that any prediction of the next value of the sequence (X_n), knowing the preceding states $(X_1, X_2, X_3, \ldots, X_{n-1})$, may be based only on the last state X_{n-1}. That is, the future value of such a variable is independent of

its past history: $P(X_{n+1} = x_{n+1}|X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_1 = x_1) = P(X_{n+1} = x_{n+1}|X_n = x_n)$.

When a Markov process is a sequence of discrete-valued variables it is called a *Markov chain* (20). The possible values of X_n are called the *state space* $\mathcal{I}$, which is a countable set and can be either finite or infinite. In this dissertation, the state space $\mathcal{I}$ is finite and has L possible values. In the denoising method proposed here, L will represent the number of points in the input set.

A *transition probability* from state i to state j at the step n, where $i, j \in \mathcal{I}$, is equal to $P(X_{n+1} = j|X_n = i)$ and is denoted by $p_{i,j}(n)$. A Markov chain is *stationary* when the transition probability does not depend on n, that means: $P(X_{n+1} = j|X_n = i) = P(X_n = j|X_{n-1} = i)$.

The *transition probability matrix* $\Pi(n) \in \mathbb{R}^{L\times L}$ is the matrix whose the entry at the i^{th} row and j^{th} column is $p_{i,j}(n)$. Observe that each of its rows sums one. The probability that the Markov chain reaches the state i at the n^{th} time step is equal to $P(X_n = i)$ and is denoted by $p_i(n)$. The *probability distribution* of the Markov chain over all states at time n is represented by the vector $P(n) = [p_1(n), \ldots, p_L(n)]$. Note that $\sum_{i=1}^{L} p_i(n) = 1$.

Given an initial probability distribution, denoted by $P(0)$, the distribution of the Markov chain in the first step is $P(1) = P(0)\Pi(1)$, and in the second step is $P(2) = P(1)\Pi(2) = P(0)\Pi(1)\Pi(2)$. So, after n steps, the distribution of the Markov chain is $P(n) = P(0)\Pi^n$ where $\Pi^n = \Pi(1)\cdots\Pi(n)$ is the *n-step transition probability matrix*. The entry at the i^{th} row and j^{th} column of Π^n is the probability of moving from state i to the state j after n steps, and is denoted by $p^n_{i,j}$. Observe that if the Markov chain is stationary, $\Pi(1) = \Pi(2) = \cdots = \Pi(n)$, so $\Pi^n = (\Pi(1))^n$.

3.2 Feature-preserving filtering

3.2.1 Problem description

Following the notation for unstructured vector fields from Chapter 2 we supposed that the vectors $\mathbf{v}_i \in \mathcal{V}$ are sampled from an unknown map $\mathcal{F}$ and corrupted by an additive random noise. The problem is to develop a method that suppress the noise from the samples and maintains the relevant features of the vector field.

3.2.2
Random walk filter

The basic principle used in previous work (28, 30, 31), translated to a graph setting, is that the probability for moving from one node to its neighbor on the graph depends on how similar they are. Suppose that a single virtual particle i is located at every node $n_i \in G_R$, and that each particle i knows not only the position $\mathbf{p}_i$ but also the vector $\mathbf{v}_i$. At each step of the random walk the particle moves from n_j to one of its neighbors or stays at its current position. After the application of n steps of these random walks, the L particles are redistributed on the graph according to the transition matrix Π^n. Such matrix induces a weighted average filter to be applied to each vector $\mathbf{v}_i \in \mathcal{V}$. The *random walk filter* computes, for each node n_i of the graph G_R, a new vector $\mathbf{v}_i$, denoted by $\mathbf{v}'_i$, and is computed according to:

$$\mathbf{v}'_i = \sum_{j \in \mathcal{I}} p^n_{i,j} \mathbf{v}_j, \tag{3-1}$$

where $\mathcal{I} = \{1, 2, \ldots, L\}$ and $p^n_{i,j}$ is the probability of moving from state i to the state j after n steps, which is the entry at the i^{th} row and j^{th} column of Π^n. The main question now is how to define the similarity functions for the transition matrices.

3.2.3
Similarity functions for vector fields

The idea to define the transition matrix $\Pi(n)$ is based on the fact that the larger the "difference" between two vectors is, the less similar they are. Sun *et al.* (30) suggest a set of similarity functions whose independent variable is the norm d of the difference between the normals of adjacent faces, like for example $s(d) = \frac{1}{C} e^{-\alpha d^2}$, where $\alpha \in (0, \infty)$ is a scale parameter and C is a normalization constant. When α is small, only faces with very close normals are considered similar. Thus, using such kind of similarity function, one has the property that the larger the difference between the normal vectors is, the smaller is the probability one should use to move a particle between the nodes. This function s is adopted in all examples of this dissertation.

A specific measure of similarity to cope with vector fields, inspired by Eibl and Brundle (6), is suggested here. For their vector field segmentation, they proposed three different measures for two given pairs of point/vector $\mathbf{f}_i = (\mathbf{p}_i, \mathbf{v}_i)$, $\mathbf{f}_j = (\mathbf{p}_j, \mathbf{v}_j)$:

– *Squared Euclidean distance* $\rightarrow d_1^2 = ||\mathbf{p}_i - \mathbf{p}_j||^2 + ||\mathbf{v}_i - \mathbf{v}_j||^2$

– *Mahanalobis distance* $\rightarrow$ $d_2^2 = (\mathbf{f}_i - \mathbf{f}_j)\Sigma^{-1}(\mathbf{f}_i - \mathbf{f}_j)^T$, where Σ is the covariance matrix of the coordinates of $\mathbf{f}_k$'s.

– *Weighted additive distances* $\rightarrow$ Given weights w_p, w_θ, w_r and w_β, $d_3^2 = w_p\|\mathbf{p}_i - \mathbf{p}_j\|^2 + w_\theta(\angle(\mathbf{v}_i, \mathbf{v}_j))^2 + w_r(\|\mathbf{v}_i\| - \|\mathbf{v}_j\|)^2 + w_\beta(\angle(\mathbf{p}_j - \mathbf{p}_i), \frac{1}{2}(\mathbf{v}_j + \mathbf{v}_i))^2$, where $\angle(\cdot,\cdot)$ is the angle between two vectors. Those weights balance the effects of each distances: the Euclidean distance from the base points, the vectors angle and norm difference and the difference of the points segment with the vector average direction.

After several experiments, we decided to adopt the *weighted additive distances*. The transition probability to move the particle from the node n_i to the node n_j at the n^{th} step is given by:

$$p_{i,j}(n) = \begin{cases} \frac{1}{C}e^{-\alpha d_{i,j}^2} & \text{if } n_j \in N(n_i), \\ \\ 0 & otherwise, \end{cases} \tag{3-2}$$

where $d_{i,j}^2$ is the weighted additive distance between $(\mathbf{p}_i, \mathbf{v}_i)$ and $(\mathbf{p}_j, \mathbf{v}_j)$ and the value of the normalization constant is

$$C = \sum_{n_j \in \mathrm{N}(n_i)} e^{-\alpha d^2}.$$

3.3 Implementation and results

3.3.1 Implementation

There are two ways to implement Equation (3-1). One is what Sun *et al.* (30) called the *batch* scheme, and the alternative one is what they called the *progressive* scheme. In the batch scheme the entries $p_{i,j}^n$ are computed by growing the neighborhood of the nodes, and computing for each step all transition probabilities, and use them at the end to compute the weighted average. In the progressive scheme, the algorithm runs step by step. It traverses only the first neighbors of the spot vertex and computes the probabilities for its neighbors. In the dissertation we use the progressive scheme, since it shows to be faster than the batch one in the majority of the experiments and the denoising requires only a few iterations. Moreover, the steps forms a scale-space, a notion we will intensively use in the next chapter.

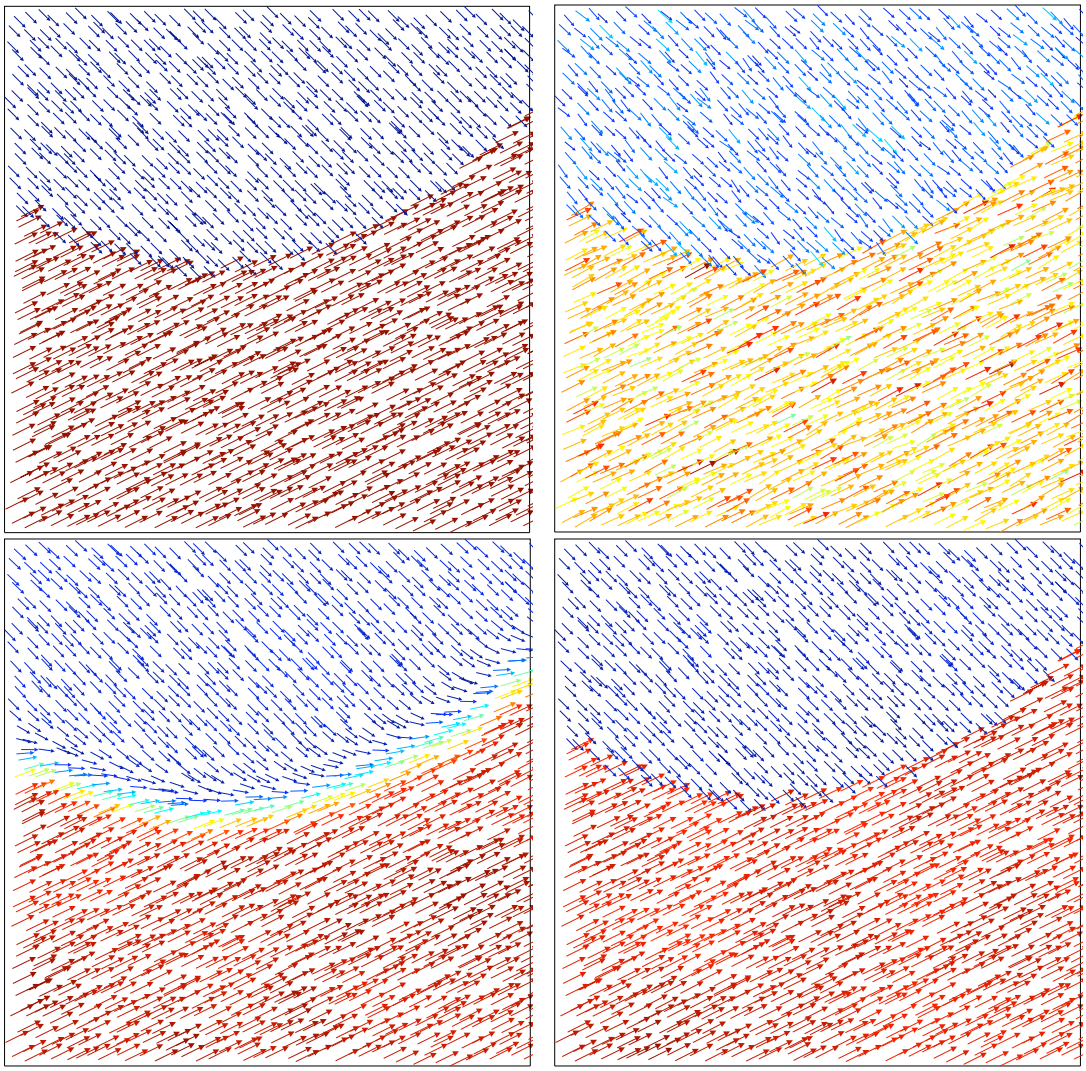

Figure 3.1: A simple discontinuous vector field (left) with gaussian noise added (center left). The gaussian filter (center right) blurs the interface, while the random walk (right) preserves it.

3.3.2 Parameters of the method

Besides the radius R used to construct the connections between the nodes of the graph, and the number n of steps for the random walk, there are more four parameters, the ones for the weighted additive distances: w_p, w_θ, w_r and w_β.

A suggestion for the weight w_p is $1/(2R^2)$, in order to give more weights to the points closer to each other in the ball of radius R. Notice that the term $w_p||\mathbf{p}_i - \mathbf{p}_j||_2^2$ naturally incorporates the distance between the base points, which is a nice advantage when the set of input points are unstructured. To fix parameter w_θ independently of the experiment, we optimize it for an average configuration: when the angles $(\angle(\mathbf{v}_i, \mathbf{v}_j))$ are uniformly distributed

in the interval $[0, \pi]$. Then one can set as default w_θ to be the variance of this distribution, i.e. $w_\theta = \pi/12$. Finally, if σ^2 is the variance of the lengths of the vector, then a suggestion for the value of w_r is $100/(2\sigma^2)$, since in this case it is considering the gaussian distribution with variance equal to the total variance over ten. Since the application is on denoising, as default the weight w_β is set to zero, because it usually destroys the interface of discontinuity of the vector field if it exists.

3.3.3 Results

We tested our denoising method on three kind of models: a simple noisy discontinuity test, where we expect the random walk to outperform the gaussian filter, measured vector field of physical systems and simulation models. For all examples of this section, we choose the parameters according to the suggestions presented in the previous section. We compare our method to a gaussian filter, which corresponds to a particular case by setting $w_p = (2R^2)^{-1}$, and $w_\theta = w_r = w_\beta = 0$. For all examples, we set $n = 2$ for both filtering methods.

Synthetic data This simple discontinuity test is constructed using a synthetic vector field:

$$\mathbf{v}(x, y) = \begin{cases} (2, \ \ 1) & \text{if } 10y < (x+1)^2, \\ (1, -1) & \text{otherwise.} \end{cases}$$

The samples are created by evaluating this map on 900 base points that are randomly generated using a stratified distribution in the grid $[-3, 3] \times [-3, 3]$ (5). To each component of the sampled vectors we add an independent and identically distributed random gaussian noise with mean 0 and standard deviation equals to 0.05. Figure 3.1 shows that the gaussian filter blurs the interface, while the random walk nicely preserves it.

Simulation data We also checked our method on a simulation of shear bands in granular flows (3). The vectors on this example are placed on a 50×50 grid. The top picture of Figure 3.2 shows the equilibrium state of the mobility of grains in a dense granular system under shear, which almost half of the rows are moving one way, half moving the other way, with the shear band being formed at the very center. At this center area, the velocity is randomly distributed and its module is almost zero, resulting in a shear band. In this figure, for visualization purpose, the size of the vectors are the same, the colors are used

to represent their norm. In this example, the samples are originally with an unknown noise. The middle and the right pictures shows the filtered vector field by the gaussian and by the random walks method. The gaussian filter almost removes the shear band, while the random walk stresses it.

We finally checked our method on simulation models of a two-dimensional landslide and its impact into a water body (10), such data is available from SPHERIC (11). First our method is tested on an landslide measured by PIV methods (Figure 3.3) and then it is again tested on a SPH simulation data (Figure 3.4). We can see on both cases that the random walk matches the global behavior, sketched on Figure 3.4.

Measured data We perform a second test of our approach on a real data acquired from a PIV device. The top picture of Figure 3.5 shows the original data on $\Omega = [-1, 1] \times [-1, 1]$ with 15607 points. This sampled velocity field corresponds to a flow of water that is continuously injected vertically on the bottom right corner. The resulted vector fields after applying a gaussian and a random walk filter are illustrated, respectively, by the middle and the bottom pictures on Figure 3.5. On one hand, we see that the gaussian is more successful than the Random walk in removing the noise, however it destroys the singularities on the right near the wall. On the other hand, the random walk preserves the features but is unable to fully remove the noise on the left. There's a delicate tradeoff between noise and information in this data set which will be addressed in next chapter.

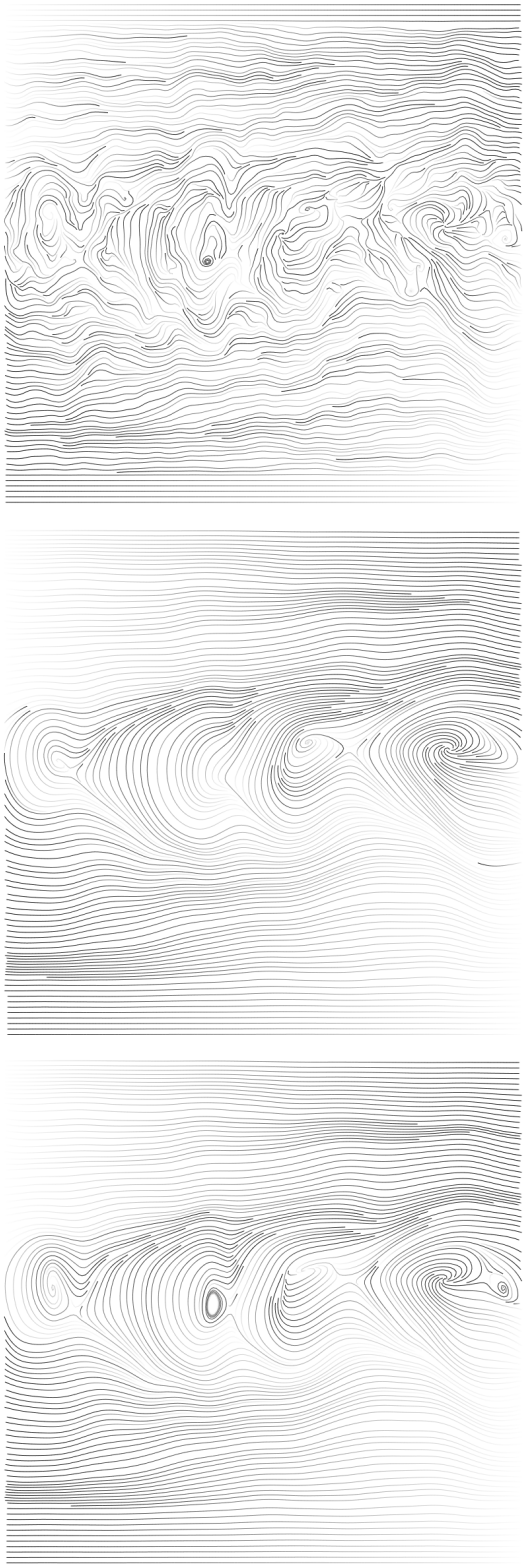

Figure 3.2: A shear band simulation of a granular flow (top): the gaussian filter (middle) removes the shear band, while the random walk (bottom) stresses it.

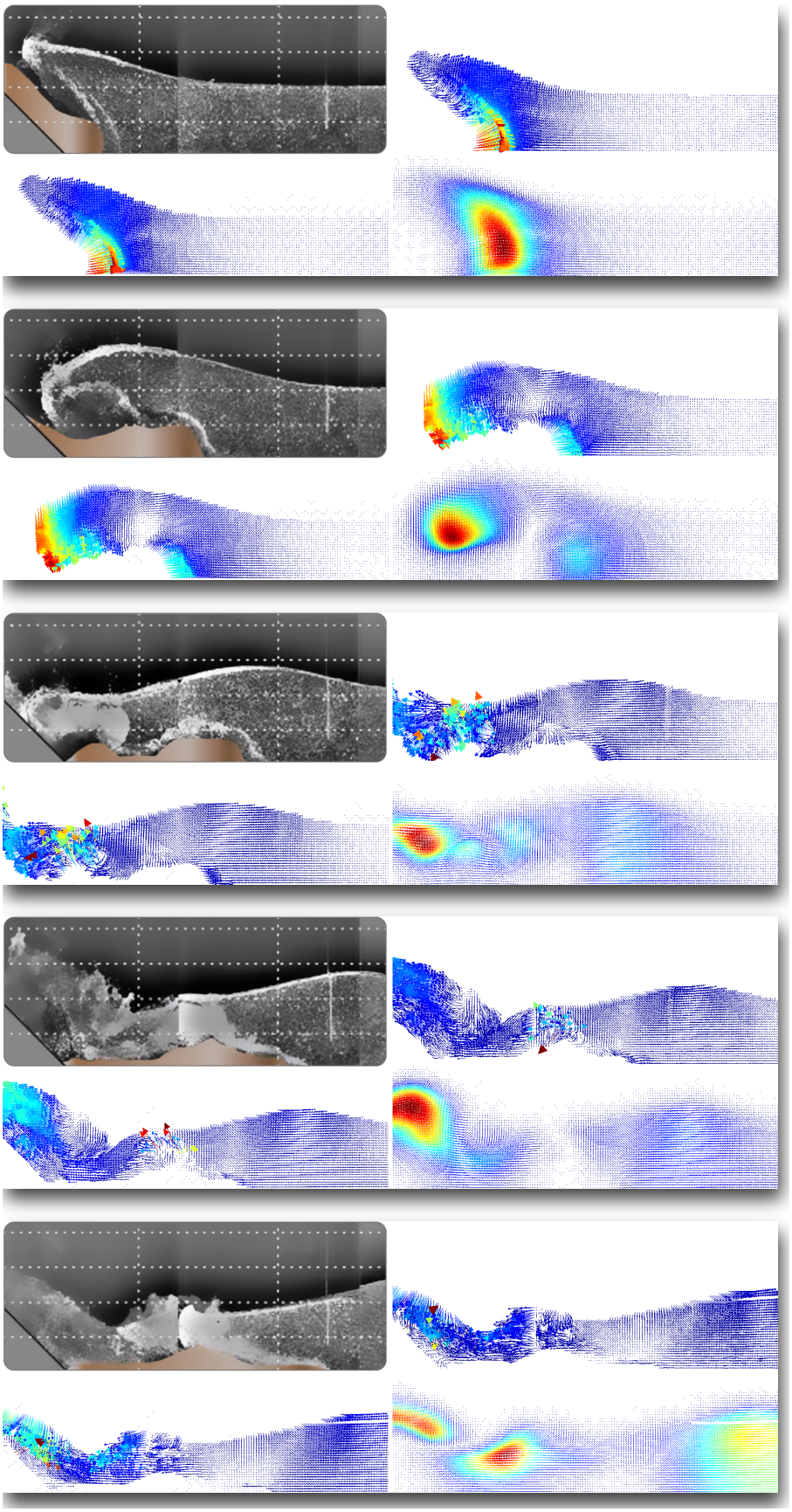

Figure 3.3: Landslide in 5 steps (each block): PIV measure (top left), simulated SPH vector field (top right). The bottom left and right pictures show the random walks and the gaussian filtered vector fields, respectively. The gaussian method oversimplifies the model.

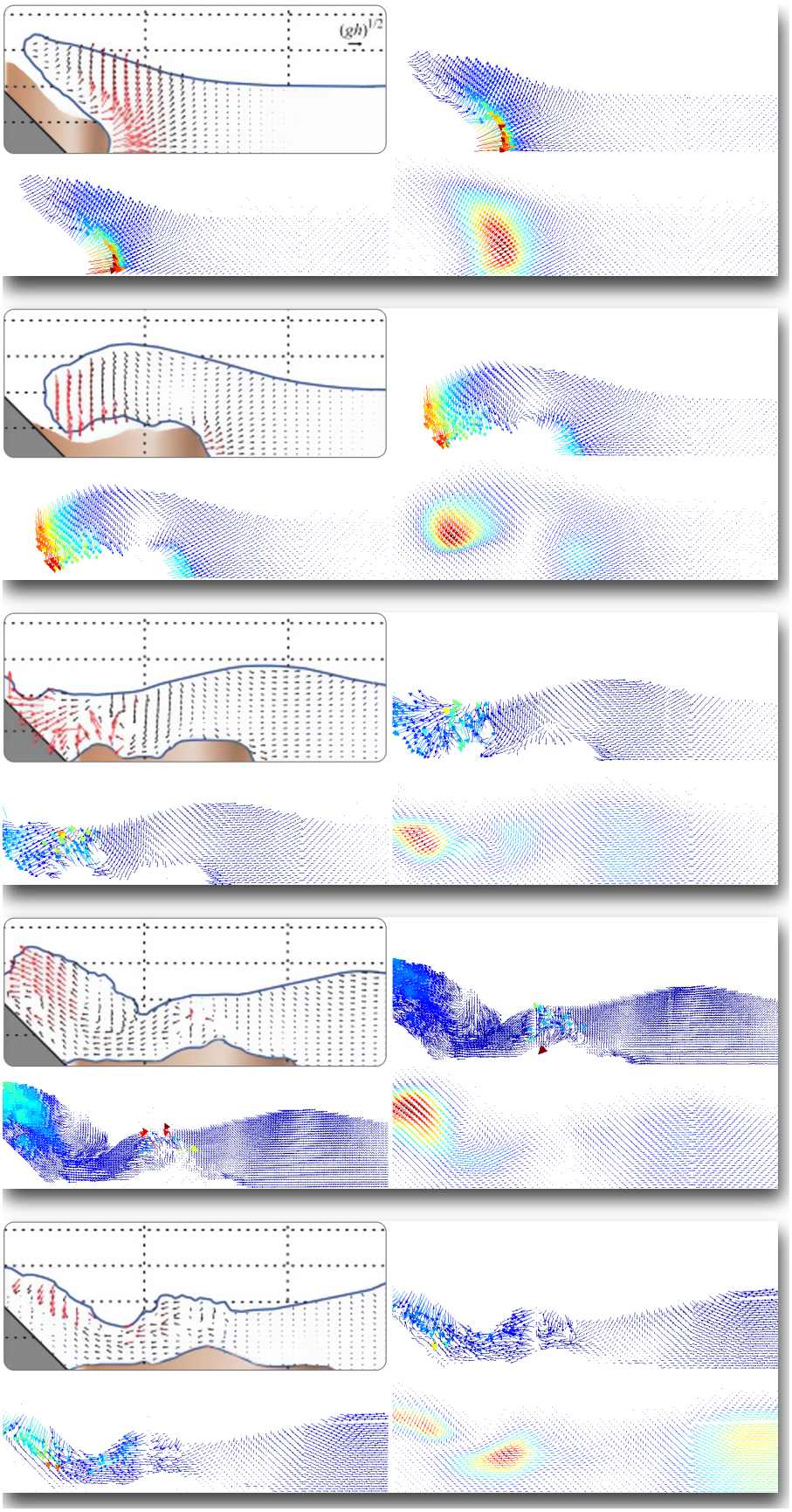

Figure 3.4: Sketched model of the landslide of Figure 3.3 (top left) that corresponds to the vector field on the top right rendered with a third of the samples. The bottom left and right pictures show the random walks and the gaussian filtered vector fields, respectively.

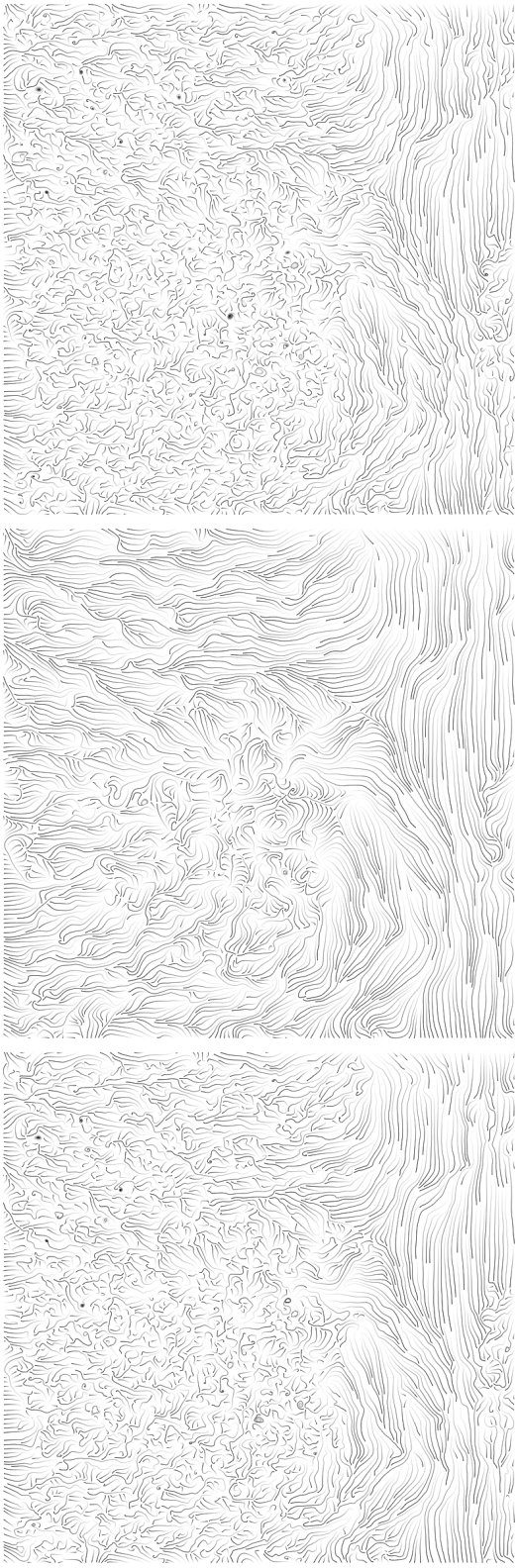

Figure 3.5: A PIV model of a fluid flow (left), filtered by a gaussian filter (middle) and by our random walk (right): while the gaussian removes the noise, it destroys the singularities on the right near the wall, the random walk preserves them but is unable to fully remove the noise on the left