Digital Elevation Model (DEM) clustering for terrain modeling

E. R. Stefanescu [1], A.K. Patra[1], and M. Bursik[2]

[1]Department of Mechanical and Aerospace Engineering, University at Buffalo, Buffalo, NY 14260

[2]Department of Geology, University at Buffalo, Buffalo, NY 14260

February 24, 2014

Abstract

We consider the problem of Digital Elevation Models (DEMs) segmentation in homogeneous regions, aiming for identification of plateaux, ridges, small drainages, straight front slopes, valleys, and crests, in order to create ensembles of DEMs. These are then used in a systematic hazard analysis, resulting in a complete and complex hazard maps. In the paper we explore and implement a method for segmentation using clustering, that is required / needed when we want to construct a sparse representation of the DEM. The method – spectral clustering, is extensively and successfully used in image segmentation. It is a complex method that accounts for the spatial correlation of the elevation points and has the advantage that it can be used for almost any application where relationships between topographic features and other components of

landscapes are to be assessed. Here, the method is adapted for the case in which each data point has associated range of geomorphometric measures.

1 Introduction

Information about topography is necessary for landscape evaluation, erosion studies, hydrology and geophysical modeling, natural hazard prevention, etc. The classic way to incorporate relief units into a landscape assessment is to delineate them during field survey or using stereo aerial photographs. This approach is relatively time-consuming and the results depend on the subjective decision of the interpreter. Several methods for the creation of landform elements using elevation-derived attributes are described in the literature. Commonly, these techniques developed regions of homogeneity based on common attributes and then classified those regions (or groups of regions) as elements. The most widely used techniques are: self organizing map [**?**], watershed segmentation [**?**], support vector machine [**?**], segmentation using heuristic rules and fuzzy logic [**?**], fuzzy K-means classification [**?**] and object-based image analysis [**?**]. Many of these techniques have drawbacks, especially when the method relies heavily on hydrological information and requires data-specific knowledge; also the methods do not incorporate autocorrelation between the same attribute at two locations.

Digital Elevation Models (DEMs) are digital representations of terrain, and consist of an array of squared cells (data points/ pixels) with an elevation associated with each data point. They can have different resolutions (5m, 30m, 90m, 120m, etc.) and can be obtained from various methods (photogrammetry, radar interferometry, laser altimetry, etc.). Usually the size of a DEM to be used in a particular application varies from tens to hundreds of kilometers which can lead to thousands to millions of grid points.

The aim of this paper is to quantify the variation in the output of a computational

flow model for block and ash flows, when the model inputs, including the elevation values represented in the DEM, are uncertain or given as a range of possible values. Integrating these variations in the possible flows as a function of input uncertainties provides well-defined data on the probability of hazard at specific locations, i.e., a hazard map [**?**]. In particular, the focus here is on assessing the influence of DEM uncertainties and proposes an improved method of generating ensembles of DEMs for probabilistic pyroclastic flow hazards analysis as well as other applications in which DEM uncertainty could play a critical role in decision making.

The particular problem that is addressed in the present contribution is that of the variation in data quality and error structure in different regions on a given DEM due to features within the topography itself, or of sensors and their interaction with the topography. For example, radar interferometry has difficulty in properly capturing topography in steep regions, and especially in regions from which radar return is insignificant, such as slopes facing away from the sensor. In these regions, errors are relatively high compared, e.g., to flat regions of diffuse reflectance. In this context, we seek to implement a segmentation of a DEM of Mammoth Mountain to create non-overlapping groupings of homogeneous regions. These homogeneous regions consist not only of contiguous areas with similar slope or curvature, but more precisely any noncontiguous areas having features that should result in their having the same error structure.

Mammoth Mountain (Fig. 1) was chosen for this study due to the existence of appropriate data sets and geologic setting. Mammoth Mountain is a large, geologically young, composite dome volcano located on the southwestern rim of Long Valley Caldera, California [**?**]. There are many active geophysical flow hazards occurring on Mammoth Mountain, including snow avalanches, rock avalanches and debris flows. In addition, it is intersected by the Mono-Inyo Craters volcanic chain, which is the most active volcanic region in the southwestern U.S. If

Mono-Inyo type activity occurs on Mammoth Mountain, then domes may form. These new domes would be growing atop a steep edifice, and therefore could become gravitationally unstable. Given that block and ash flows occurred at Mammoth Mountain during its older dome growth stage, there is reason to believe that renewed dome formation would result in block and ash flow activity. If this is so, then parts of Mammoth Lakes, CA, are at risk from block and ash flows. Our previous work on Mammoth Mountain (Stefanescu et al., 2012xx) was the testing of the hypothesis that different DEMs result in different model outputs of block and ash flow inundation. The present contribution thus in part represents an exploration of a refinement of the methodology.

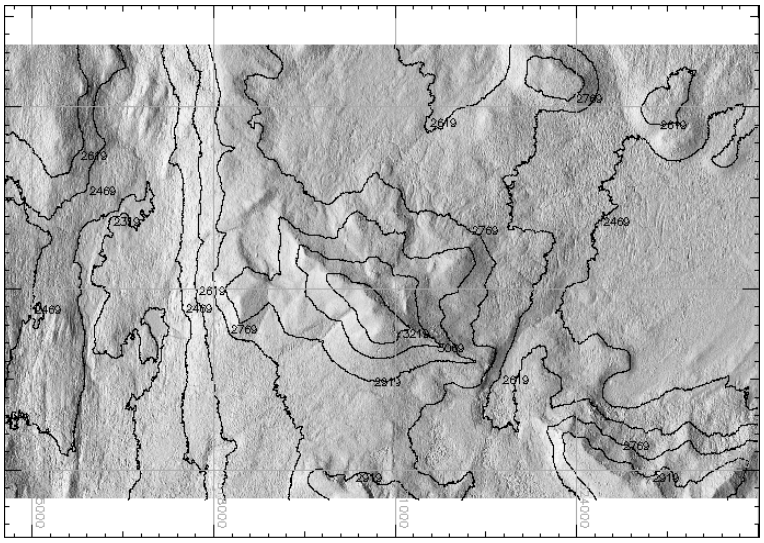


Figure 1: Hillshade plot of the Mammoth Mountain

2 Methodology

Segmentation is a broad term, covering a wide variety of problems and techniques. Segmentation methods are based on characterizing the similarity between some data point or region

in relation to its local neighborhood, measured either spatially or categorically. A variety of different methods have been proposed for image segmentation such as boundary-based segmentation, region-based segmentation and pixel-labeling. One view of segmentation is that we are trying to determine which components of the data set naturally "belong together". This is a problem known as *clustering*. Clustering is difficult to implement in practice for a number of reasons. Real-life data may contain clusters of varying size and shape, whose number is unknown in advance. Noise and outliers can further complicate the task by connecting separate clusters. Spectral clustering was first developed in the context of graph partitioning problems [**?**], where the problem is to partition a weighted graph into disjoint pieces, minimizing the sum of the weights of the edges linking the disjoint pieces. In the graph the nodes represent the grid points and arcs represent affinities ("couplings") between nearby grid points. The final cluster assignments of the dataset can be achieved by optimizing some clustering criteria defined on the graph. The criteria of some spectral clustering methods are to optimize some cut value on an undirected graph, such as a normalized cut [**?**], ratio cut [**?**], or min–max cut [**?**].

To be able to perform the segmentation of the DEM in homogeneous regions we need to specify a range of geomorphometric measures which can be extracted from the surface. We define a *feature matrix* of DEM attributes, consisting of elevation and first and second derivatives of elevation (slope, profile curvature and tangential curvature). Slope and curvature are easily extracted from a DEM within a Geographical Information System (GIS).

In the next sections a basic methodology for generating ensembles of DEMs using segmentation is presented, with emphasis on segmentation using spectral clustering. Subsequent sections summarize the TITAN2D flow simulation tool and its use in a systematic hazard analysis. The hazard analysis tool itself uses ensembles of TITAN2D simulations to construct statistical surrogate models of flow outcomes at different locations as a function of

model inputs, such as flow volume and initial location. Sampling of these surrogates leads to the construction of effective hazard maps that reflect the range of uncertainty in the model inputs.

2.1 Spectral Clustering

A digital representation of a terrain surface is an approximation of reality and is often subject to significant error. The error is usually not known in terms of both magnitude and spatial distribution. There are, in fact large uncertainties associated with the construction of DEMs. DEM vendors generally provide users with a measure of vertical accuracy in the form of the root mean squared error (RMSE) statistic.

However, one key feature of DEM grid points, which are spatial data, is the autocorrelation of observations in space. Generally, spatial autocorrelation refers to the correlation between the same attribute at two locations. Observations in close spatial proximity tend to be more related than observations at larger distances or separation. Based on this assumption spectral clustering is performed to identify homogenous regions within a DEM.

Compared with traditional clustering algorithms, spectral clustering has some advantages: it can stably detect non-convex patterns and linearly non-separable clusters [**?**], and can obtain the globally optimal solutions in a continuous domain by eigendecomposition [**?**]. As a discriminative approach, they do not make assumptions about the global structure of data. Instead, local evidence on how likely two data points belong to the same class is first collected and a global decision is then made to divide all data points into disjunct sets according to some criterion. Often, such a criterion can be interpreted in an embedding framework, where the grouping relationships among data points are preserved as much as possible in a lower-dimensional representation. What makes spectral methods appealing is that their global-optima in the relaxed continuous domain are obtained by eigendecompo-

sition. However, to get a discrete solution from eigenvectors often requires solving another clustering problem, albeit in a lower-dimensional space. That is, eigenvectors are treated as geometrical coordinates of a point set. Unfortunately, when the number of grid points (denoted as n) is large, spectral clustering encounters a quadratic resource bottleneck in computing pairwise similarity between n nodes and storing that large matrix. Moreover, the algorithm requires considerable computational time to find the smallest k eigenvalues of a Laplacian matrix. Eigenvalues and eigenvectors are at the heart of spectral clustering algorithms, and in spite of their importance, existing eigensolvers do not scale well. Fast computation schemes for spectral clustering have been proposed by different authors. They focus on the eigenvector computation of a graph Laplacian defined by a matrix of data similarities. The Krylov subspace methods [**?**], are iterative algorithms for finding leading eigencomponents of a sparse matrix, while **?** assume the availability of the similarity matrix and propose a method that does not use eigenvectors. **?** propose using the Nyström approximation to avoid calculating the whole similarity matrix. In this paper we are using a method proposed by **?**, which parallelize spectral clustering on distributed computers to address resource bottlenecks of both memory use and computation time.

2.1.1 Approach

For a given data set $P = \{p_1, \ldots, p_n \in R^d\}$, spectral clustering finds a set of data clusters, $\{C_1, \ldots, C_k \subset P\}$, on the basis of spectral analysis of a similarity graph. Spectral clustering builds a weighted graph $G(V, E)$, where V represents vertices and E, edges. We represent each elevation point as a node in the graph G and the links between the adjacent data points will form the edges of the graph. Spectral clustering partitions data points into groups such that the members of a group are similar to each other and dissimilar to data points outside of the group. Given data points, an affinity matrix can be represented by a weighted adjacency

matrix $W \in \mathcal{R}^{n \times n}$, where w_{ij} is a measure of the similarity between grid point i and grid point j. The affinity matrix is used to preserve the local structure of the patterns. It expresses the degree of similarity between points, and it must have the following properties: i) non-negative; ii) symmetric; iii) invertible. We have chosen the heat kernel for calculating the affinity matrix, as:

$$\mathbf{W}_{ij} = \begin{cases} \exp \frac{-\|F(i)-F(j)\|}{\sigma_F^2} * exp \frac{-\|x(i)-x(j)\|}{\sigma_x^2}, & if \parallel x(i) - x(j) \parallel \leq r \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where $F(i)$ represents the DEM feature vector for node i, and $x(i)$ represents the coordinate location of i^{th} node. σ_F and σ_x are positive tuning parameter that controls the decay of the affinity [**?**]. The graph partitioning can be interpreted as a minimization problem of an objective function. Common objective functions are the ratio cut (Rcut), normalized cut (Ncut) and min–max cut (Mcut) expressed as:

$$Rcut(C_1, \ldots, C_k) = \sum_{l=1}^{k} \frac{(C_l, P \setminus C_l)}{card\ C_l} \tag{2}$$

$$NcutC_1, \ldots, C_k) = \sum_{l=1}^{k} \frac{(C_l, P \setminus C_l)}{cut(C_l, P)} \tag{3}$$

and

$$Mcut(C_1, \ldots, C_k) = \sum_{l=1}^{k} \frac{(C_l, P \setminus C_l)}{cut(C_l, C_l)} \tag{4}$$

Here, $cut(X, Y)$ is the sum of the edge weights between $\forall p \in X$ and $\forall p \in Y$, $P \setminus C_l$ is the complement of $C_l \subset P$, and $card\ C_l$ denotes the number of points in C_l. The degree d_i of node i is the sum of all edge weights incident on x_i:

$$d_i = \sum_{j=1}^{n} w_{ij} \tag{5}$$

Let h_l be an $n-$dimensional vector indicating the members of the cluster C_l by its binary components. The minimization problem of any objective function in Eqs. **??**, **??** and **??** can be rewritten as a trace minimization problem under a constraint on a matrix $H = [h_1 \dots h_k]$:

$$\min_H tr(H^\top LH) \; subject \; to \; H^\top LH = I. \tag{6}$$

The matrix N is equal to I, D or W for Rcut, Ncut or Mcut, respectively [**??**]. The spectral clustering algorithms were derived from the minimization problem in Eq. 6 by relaxing the binary constraint on h_l. The relaxed trace minimization for $H \in \mathcal{R}^{n\times k}$ is the generalized eigenvalue problem [**?**]:

$$LH = NH\Lambda \tag{7}$$

The eigenvectors for Ncut and Mcut are identical due to this relaxation, while for Ncut, Eq.7 can be converted into a normal eigenvalue problem:

$$SZ = Z\Delta \tag{8}$$

where

$$S = S_{sym} = D^{-1/2}WD^{-1/2}, \; Z = D^{1/2}H \; and \; \Delta = I - \Lambda \tag{9}$$

or

$$S = S_{rw} = D^{-1}W, \; Z = H \; and \; \Delta = -\Lambda \tag{10}$$

The above method leads to the normalized graph Laplacians defined as:

$$L_{sym} = I - D^{-1/2}WD^{-1/2} \; and \; L_{rw} = I - D^{-1}W \tag{11}$$

L_{sum} is a symmetric matrix, while L_{rw} it is closely related to a random walk. A more detailed description for normalized graph Laplacian can be found in **?**. The data clustering by graph-cut boils down to the eigenvalue decomposition problem of S for finding the cluster indicators

$h_1, \ldots h_k$. These eigenvectors induce an embedding of the data points in a low-dimensional subspace wherein a partitioning based on the normalized cut (NCut) can be used. The solution of the minimization problem can be obtained from the Fiedler eigenvector [**?**]. The steps involved in DEM segmentation using spectral clustering are summarized in Figure 2.

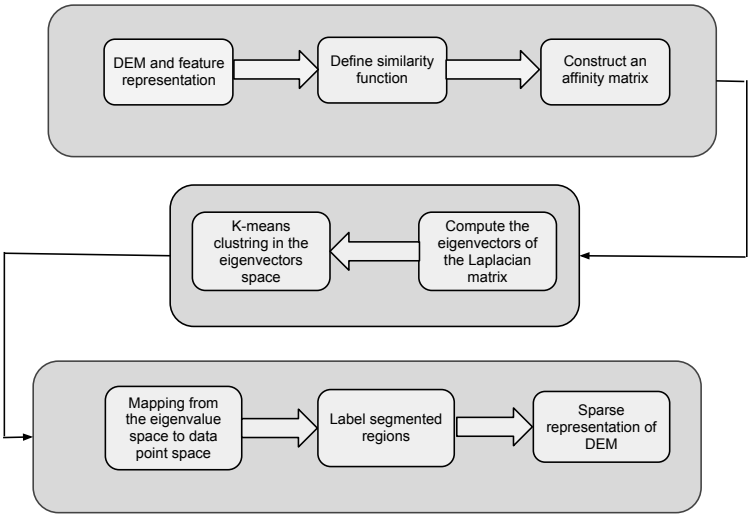


Figure 2: Spectral clustering for DEM segmentation workflow

2.1.2 Parallel implementation

When the number of data points (n) is large, the computational complexity of spectral decompositions can reach $O(n^3)$ (W dense). If the affinity matrix W is define as in Equation 1 then its construction takes $\mathcal{O}(n^2 d)$ flops, which can be also computationally intensive if the data cardinality n or dimensionality d is large. **?** investigate approaches for large-scale spectral clustering and propose a parallel implementation, which is also adopted in this paper. The strategy to address the computational and memory difficulties involves distributing n data instances onto p distributed machine node. On each node, parallel spectral clustering

(PSC) computes the similarities between local data and the whole set in a way that uses minimal disk I/O. Then PSC stores the eigenvector matrix on distributed nodes to reduce per-node memory use. Together with parallel eigensolver and K-means, PSC archives good speedup with large data sets.

3 DEM ensembles

When propagating uncertainty in DEMs through a geophysical system, stochastic methods are considered to be an effective way to estimate the probability density function of outputs by addressing uncertainties present in initial conditions and in model approximations. In previous work **?** presented a basic methodology for generating ensembles of DEMs representative of the true DEM. Here, we extended the methodology at the cluster level, by making the assumption that each homogenous regions has its own error model which leads to different random fields. The random fields are used in creating multiple equally likely representations of an actual terrain surface, following the approached suggested by **?**. A normal distribution (mean of 0.0 and variance of 1.0) of maps or realizations is computed to reproduce the spatial autocorrelation encountered in the original error surface, filtered using a Gaussian convolution filter, with kernel sizes derived from autocorrelation analysis of the original error surfaces. The random field function derives its spatial dependence from the use of a distance based decay filter function. The following equation is used to generate the random field:

$$Z(\mathcal{U}) = \frac{\sum_v w_{u,v}\epsilon_v}{\sqrt{\sum_v w_{u,v}^2}}, \quad u \in \mathcal{U},\ v \in \mathcal{V} \tag{12}$$

$$w_{u,v} = \begin{cases} 1 & : d_{u,v} \leq F \\ \left(1 - \frac{d_{u,v}-F}{D-F}\right)^E & F < d_{u,v} \leq D,\ u \in \mathcal{U},\ v \in \mathcal{V} \\ 0 & : d_{u,v} > D \end{cases} \tag{13}$$

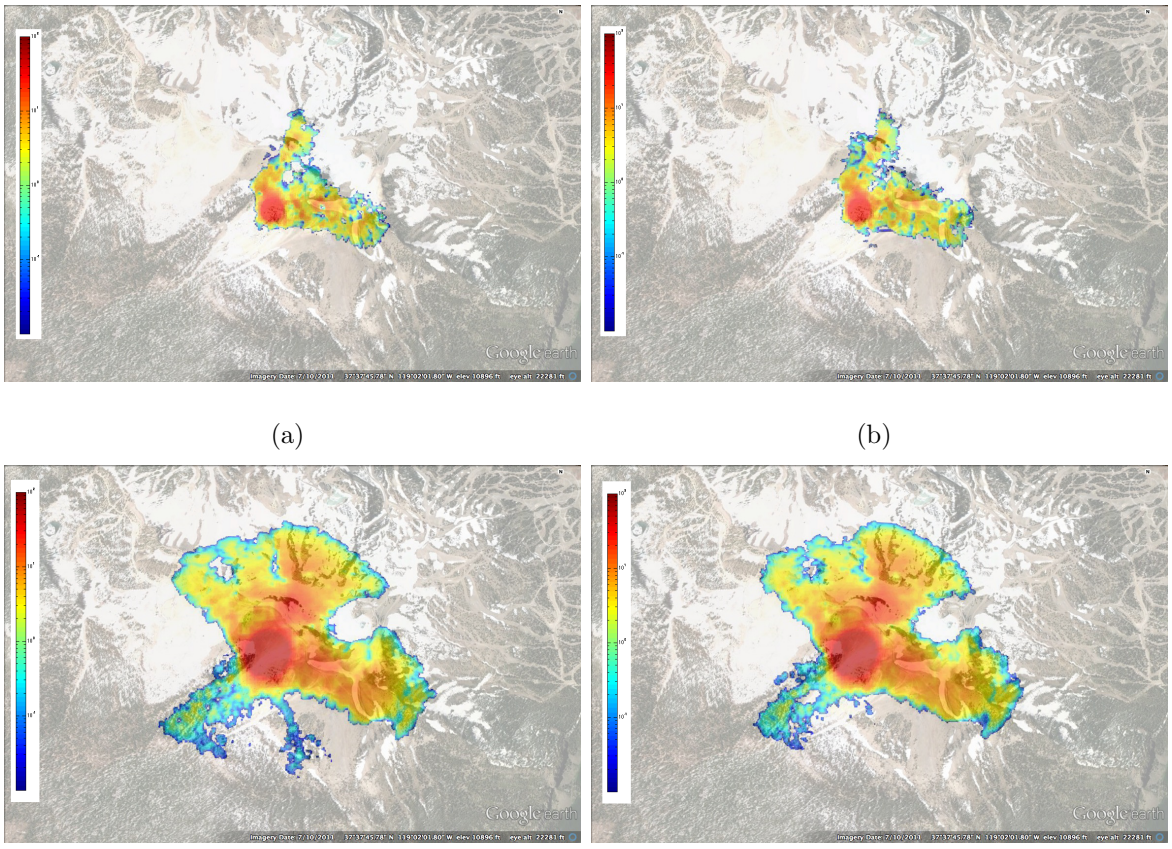
where $\mathcal{V}$ is the set of points potentially influencing points in a given area, $\mathcal{U}$, $w_{u,v}$ is the spatial autocorrelative effect between points $u \in \mathcal{U}$ and $v \in \mathcal{V}$, ϵ_v is a Gaussian random variable with a mean of 0 and variance of 1, $d_{u,v}$ is the distance between u and v, D is the minimum distance of spatial independence, E is the distance decay exponent, and F the distance at which errors are completely correlated.

A set of random fields are created for each homogenous region/ cluster and are calibrated to the spatial variation of the field being simulated using a correlogram function. This is done by fitting the correlogram and choosing the best descriptive parameters of the random field (the minimum distance of spatial independence, the correlated distance decay exponent and the filter parameter) in a weighted least-square estimator. After running hundreds of tests with multiple combinations of D, E and F, the best random field was found by fitting the error map characteristics such that the sum of least squares difference between an error field's correlogram and the target correlogram is minimized.

Each error realization was added to the "true" DEM indicated as $m(\mathcal{U})$, to generate equally probable realizations of the topography for the error structure of a DEM under consideration:

$$R(\mathcal{U}) = m(\mathcal{U}) + m(m(\mathcal{T})) + (m(s^2(\mathcal{T})) \cdot \epsilon) \cdot Z(\mathcal{U}) \tag{14}$$

where $R(\mathcal{U})$ is a realization of the elevation dataset $m(\mathcal{U})$, $\mathcal{T}$ is a group of sets of spatially uncorrelated sample points in $m(\mathcal{U})$, and ϵ is a Gaussian random variable with mean 0.0 and variance 1.0. $m(m(\mathcal{T}))$ and variance $m(s^2(\mathcal{T})$ is mean and variance, respectively, of all sets in $\mathcal{T}$. $Z(\mathcal{U})$ specifies the random field as defined in Equation 12 for each homogenous region.



(a) (b)

(c) (d)

Figure 3: Flow maps for two selected parameters values when DEM was created using a) no cluster b) cluster

4 Flow simulator

The threat of avalanches, block-and-ash flows, and mud-flows at volcanoes is a major global problem. These are complex phenomena and involve physics at multiple spatial and temporal scales. Developing satisfactory models and computational simulations of potential debris flows on natural terrains and integrating them with appropriate geographical information is difficult but extremely necessary. In recent years, many significant advances have been

made in the modeling and simulation of such flows by taking advantage of new models of the physics, new stable and accurate solution schemes for nonlinear hyperbolic systems, and the wide availability of high-performance computers.

Hazard maps are one of the most important instruments [**?**] for representing areas of potential inundation by dangerous phenomena at volcanoes. One major problem with using model to construct hazard maps is the strong dependence on the outcome on the choice of model parameters, such as bed friction, initial volumes and terrain.

4.1 TITAN2D

TITAN2D [**??**] was developed for modeling dry geophysical granular flows, such as debris avalanches and block and ash flows. The TITAN2D code combines numerical simulations of a natural granular flow with digital terrain data. It is based on a depth-averaged model for an incompressible granular materia governed by Coulomb-type friction interactions [**?**]. The governing equations are obtained by applying conservation laws to the incompressible continuum, providing appropriate constitutive modeling assumptions, and then taking advantage of the shallowness of the flows (flows are much longer and wider than they are deep) to obtain simpler depth-averaged representations [**?**]. The motion of the material is considered to be gravitationally driven and resisted by both internal and bed friction. The stress boundary conditions are: no stress at the upper free-surface and a Coulomb-like friction law imposed at the interface between the material and the basal surface.

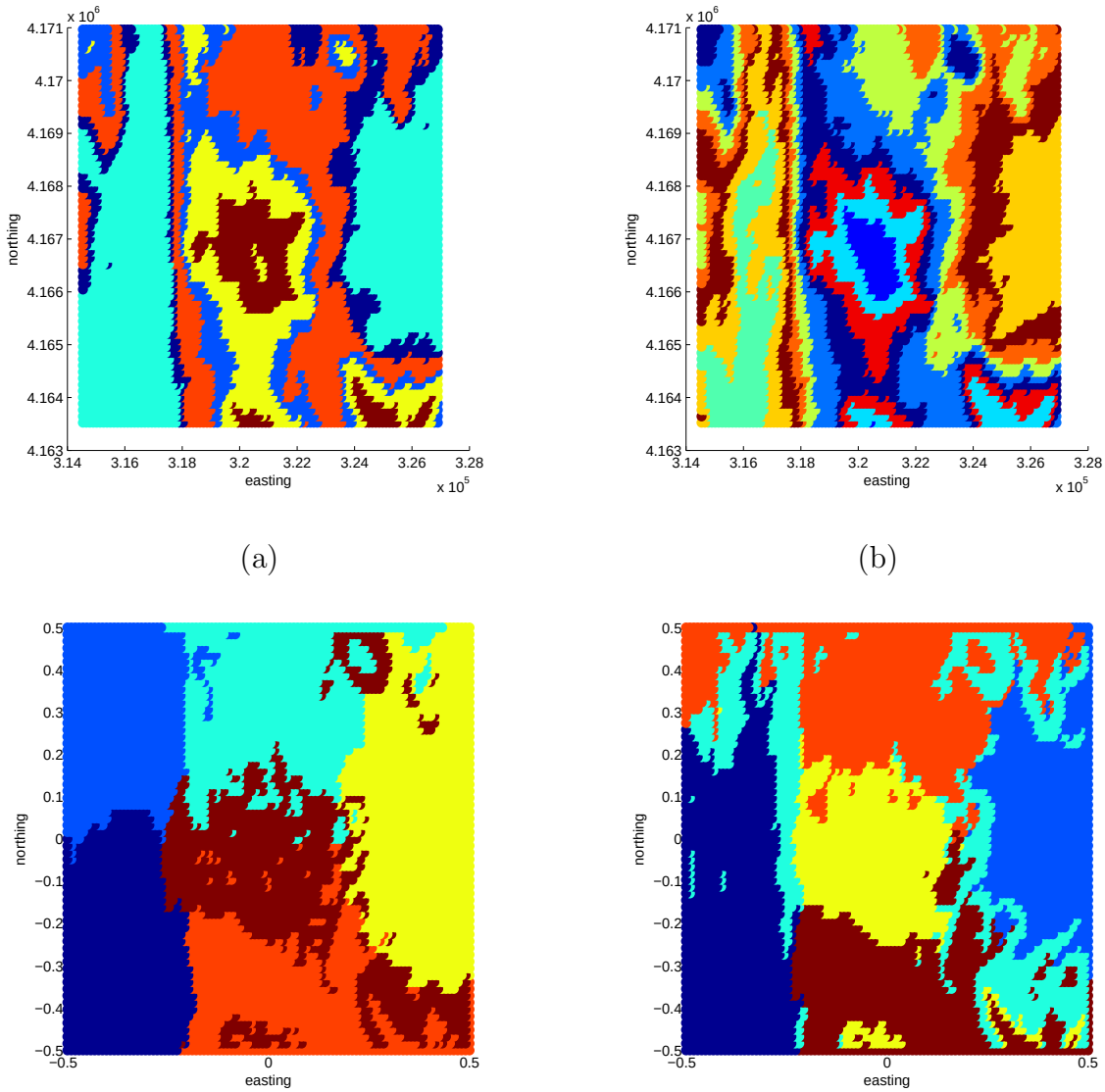
A principal feature of TITAN2D is the incorporation of topographical data from geographic information system (GIS) sources into the simulation and grid structure. Topographic data are included in the simulation through a preprocessing routine in which the digital elevation data are imported. TITAN2D performs flow simulations on a DEM of a desired region, the simulation accuracy being highly dependent on the level of the DEM

resolution and quality [**?**].

Inputs to the code are the size and location of the initial volume, the internal and bed friction and the DEM. **?** presented several methods for characterizing the effect of input data uncertainty on model output – except DEM, where uncertainty associated with spatial parameters like terrain elevation were not well understood. The output – the flow height at every grid point at every timestep – is a complete description of the mass flow over realistic terrain.

We define the stochastic input as $\Omega = (UTM_E, UTM_N, \mathbf{V}, \theta_r)^\top$. UTM_E and UTM_N are the East and North, respectively coordinates (UTM values) of the location of possible vents. These are considered to be uniform distributed around 321095 E and 4166433 N, within 400m radius. $\mathbf{V}$ is the initial volume of material, uniform distributed between 10^5 m^3 and $10^{7.5}$ m^3. θ_r includes the DEM uncertainty as described in Section 3. We ran 1024 flow simulations at design points in the region of the input space. These 1024 design points were chosen according to a Latin hypercube design, which is a space-filling design. This has been proven very successful for all-purpose designs of computer experiment runs since they require relative few design points per input to "fill" the design space [**?**].

5 Results



(a) (b)

(c) (d)

Figure 4: Spectral Clustering for a) K=6 – one feature b) K=20 – one feature c) K=6 where $\sigma_F = 0.8$ and $\sigma_x = 1$ – 4 feature d) K=6 $\sigma_F = 0.8$ and $\sigma_x = 0.8$ – 4 feature

6 Conclusions

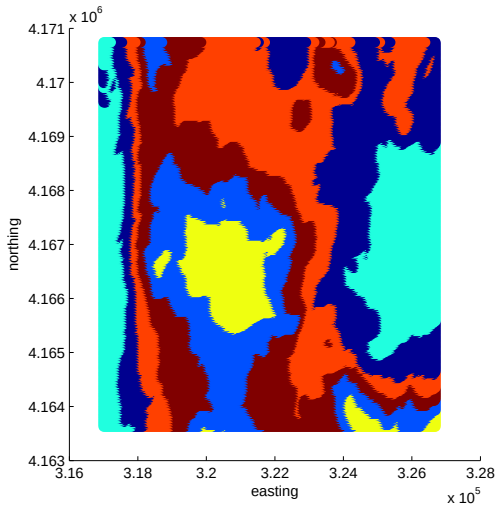
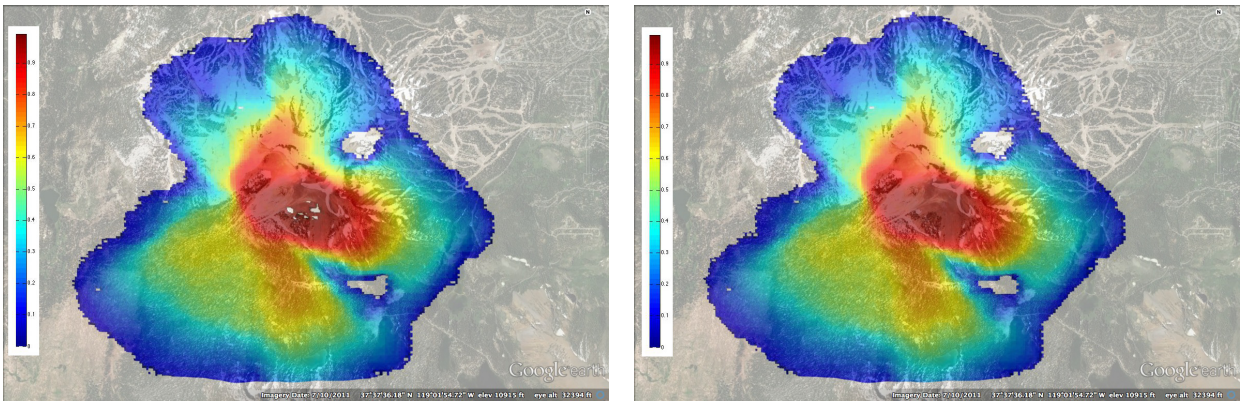


Figure 5: Parallel spectral clustering



(a) (b)

Figure 6: Hazard map a) no cluster b)clusters

Acknowledgment