DEM EXTRACTION FROM A COLOR-CODED RELIEF SCANNED MAP

Carol Rus¹, and Corneliu Rusu¹,²

¹Dept. of Electronics and Telecommunications, Technical University of Cluj-Napoca, Str. Baritiu Nr. 26-28, RO-400027, Cluj-Napoca, ROMANIA
²Tampere International Center for Signal Processing, Tampere University of Technology, P.O.Box 553, FIN-33101, Tampere, FINLAND

e-mail: Carol.Rus@bel.utcluj.ro; Corneliu.Rusu@bel.utcluj.ro
e-mail: Corneliu.Rusu@tut.fi

ABSTRACT

The focus of this paper is in extracting a digital elevation model (DEM) from a color-coded relief scanned map. The map is preprocessed using a WHMM (Wavelet domain Hidden Markov Model) based algorithm for denoising. Then a linework removal algorithm – two methods are presented: one based on WHMMs and one that uses a vector gradient edge detector – is applied. Each distinct elevation region must be solid colored using distinct colors. In order to achieve this, for the printed scanned map, we used a median filtering, and a color clustering process based on a minimum variance quantization algorithm. The gap filling problem can be solved with a multi-pass majority filter algorithm. The elevation contours are extracted by detecting the borders between colors through a sliding-neighborhood operation. The color-coded contours are converted into a sparse elevation data set by means of the map's legend. Laplacian partial differential equations are used to interpolate the sparse DEM. The LSQR iterative solver is applied in order to solve the system of equations.

1 INTRODUCTION

Digital elevation models (DEMs) are bidimensional arrays of values representing the spatial distribution of the altitude data corresponding to a terrain area. Getting high quality DEM data for some locations on the earth is easy (e.g. US DEMs are available at USGS). However, this problem can be more challenging for many other earth surface areas. A low cost solution for producing a DEM consists in interpolating the elevation values extracted from a scanned map.

A widely used approach for producing digital elevation models from scanned map is the derivation of DEMs from topographical contour maps [1] [2]. Color-coded maps represent also, a way to express the altitude information. Most of the printed GIS maps combine different layers [4]. So before we can have the altitude information, we must remove all the data belonging to non-elevation layers. Thin and sharp enough edges within a map are referred to as linework. Within color-coded relief maps, in generally, the unwanted information is linework, while within topographical contour maps the altitude data is a specific colored linework.

In this paper we propose an approach for extracting a DEM from a color-coded relief map. The paper is organized as follows. A preprocessing method using a WHMM (wavelet domain hidden Markov model) algorithm for denoising [5] is presented in Section 2. In the next section, a linework removal algorithm is applied. We present a method based on wavelet domain hidden Markov models and a method that uses a vector gradient edge detector with a saturation-based combination of hue and intensity planes [6]. Section 4 is concerned with the color clustering problem. Each distinct elevation region must be solid colored using distinct colors. In order to achieve this, for the printed scanned map, we used a vector median filtering [3], and a color clustering process – based on a minimum variance quantization algorithm [7]. Eliminating the unwanted information yields in unfilled gaps within the map image. In Section 5 the gap filling problem is solved using the Wise’s algorithm [4]. The resulting image contains altitude data expressed by means of solid colored blocks. In Section 6 we present the extraction of the elevation contours by detecting the borders between colors through a 3 by 3 window sliding-neighborhood operation. The conversion of the color-coded contours into a sparse elevation is based on the map’s legend. In Section 7, Laplacian partial differential equations (PDEs) are used to interpolate between elevation contours. An iterative solver (the LSQR algorithm developed by Paige and Saunders [12]) is applied for solving the system of equations.
2 WHMM DOENISING

Dithering effect [13] is a technique used in dot-based printing. It uses a collection of dots to simulate different colors and it is widely used for color-coded relief maps. Dithering effect can be thought as being noise; therefore a noise removal filtering must be applied to the map image. An answer to this problem is to apply median filtering on the map image. An edge detection algorithm will be applied to the image in order to remove lineework and the median filter does not preserve well thin edges. A more suitable solution as a preprocessing would be the WHMM (wavelet domain hidden Markov model) noise removal method.

The noise removal algorithm based on WHMM was developed by Crouse et al. in [5]. We applied the algorithm to each of the RGB planes of a color map image [14]. The main drawback of the approach is the computation time.

The (wavelet domain) hidden Markov model consists in four parameters: $mu$ (means matrix), $si$ (variances matrix), $ps$ (probability mass functions matrix), and $es$ (state transition probability quadrants matrix). The model is obtained using a training step that applies the EM (expectation-maximization) algorithm, having the original image as the observation data. A WHMM models the wavelet coefficients of an image and uses two state variables: $S$ (small variance), and $L$ (large variance). We assumed that the wavelet coefficients have similar statistical features within the same scale. The likelihood computation step takes as inputs: the model, and the image to be denoised and gives the $pdf$s (probability density functions) for both the $S$ and the $L$ plane, for each wavelet coefficient of the original image. The model, the $pdf$ matrix and the original image’s wavelet coefficients are used for a 2D Wiener type filtering considering only the finest scale. Larger wavelet filters are more appropriate for the noise removal. We used Daubechies 8 wavelet.

By processing each RGB component separately with the WHMM noise removal algorithm, color distortions may appear. For the tested images the color distortions proved to be sufficiently small, so that the color clustering algorithm (that uses prior knowledge on the picture) was not affected.

3 LINEWORK REMOVAL

Lineework can be viewed as edges in a scanned map image. Most of the lineework are thin enough, and sharp enough edges, thus an edge detection algorithm can identify them.

3.1 WHMM based method

WHMMs are particularly well suited to images containing singularities (especially when using Haar wavelet) [8]. We considered the noise removal algorithm based on WHMMs presented in [5], and the gray scale image segmentation algorithm based on WHMMs described in [8], and we modified them in order to extract edges (linework) within a scanned map image [14]. We applied the algorithm on an intensity plane of the image obtained by taking a simple average of the RGB components.

A Haar wavelet transform, applied on the intensity plane, $Y$, outcomes $WY_{sb} = \{HL, LH, HH\}$, which is a collection of dyadic matrices, one for each scale, of size $2^k \times 2^k$, where $k$ is the scale index. At least two iterations must be used, due to the parent-child relationships within the wavelet quad-tree that is exploited by the model-training step. The inputs for the training are the wavelet subbands, $WY_{HL}$, $WY_{LH}$, and $WY_{HH}$, as the observation data. We assumed that the wavelet coefficients have similar statistical features within the same scale. So, each parameter matrix will have at least 2 lines, one for each scale, and 3 columns, one for each subband. The model, $M_f$, and the subbands make up the input for the likelihood computation step. This step outcomes the probability density functions matrices, $PP_{sb}(i, j)$, which contain the $pdf$s (for plane $S$, and for plane $L$) of the wavelet coefficient $WY_{sb}(i, j)$, assuming that $i$, and $j$ are indexes within the array, and $sb = \{HL, LH, HH\}$.

Equation (1) shows how the overall $pdf$ of a wavelet coefficient can be modeled using the joint $pdf$s of the hidden states, $PP_{sb}(i, j)|L$ and $PP_{sb}(i, j)|S$, weighted by their joint $pmfs$ (probability mass functions) $ps_{sb}|L$ and, respectively, $ps_{sb}|S$, where $sb$, $i$, $j$ have the same significance as in the previous paragraph.

\[
pdf(i, j) = [ps_{sb}|L] \cdot [PP_{sb}(i, j)|L] + [ps_{sb}|S] \cdot [PP_{sb}(i, j)|S]
\] (1)

Taking this into consideration, we can state that a wavelet coefficient is assumed to be large if it respects the following relation:

\[
[ps_{sb}|L] \cdot [PP_{sb}(i, j)|L] \geq [ps_{sb}|S] \cdot [PP_{sb}(i, j)|S]
\] (2)

A large wavelet coefficient at the finest scale within a subband denotes that the corresponding 2 by 2 pixels block, in the original image, contains an edge (i.e. lineework). The finest scale for $2^j \times 2^j$ image is $J - 1$. So, $WY_{sb,j-1}$ is a $2^{J-1} \times 2^{J-1}$ matrix. When using the WHMMs in order to detect lineework in a scanned map, we work at 2 by 2 pixels blocks level, and not at pixel level. This can be a drawback for identifying very fine edges. Based on equation (2) we can create a lineework mask that incorporates the edges within the scanned map. Overlapping the lineework mask and the scanned map image outcomes a map within the lineework was removed.

Subband $HL$ yields a lineework mask for the horizontal edges; subband $LH$ yields a lineework mask for vertical
edges; and subband $HH$ yields a linework for the diagonal edges. A practical overall mask can be obtained using the following formula:

$$MASK_{overall} = MASK_{horizontal} \cdot MASK_{vertical} \quad (3)$$

Diagonal edges are not considered in order to avoid the removal of too many non-linework pixels (that relies next to the linework pixels) from the scanned map. The overall mask computed with equation (3) can be combined, as it is, with the scanned image. However, better results can be obtained if the overall mask is median filtered with 3x3 window, before overlapping with the scanned map.

### 3.2 Vector gradient edge detector based method

A well-known edge detector in color image processing is the difference vector edge detector [6] which is a $3 \times 3$ operator calculating the maximum gradient across the central pixel. An other edge detector used in color image processing is the vector gradient edge detector – a local operator that computes the maximum distance in the desired metric between the center pixel and the 8-connected pixels adjacent to it. The methods were applied on the RGB map image using the Euclidean distance version and the vector angle version. The Euclidean distance is efficient in the intensity domain, while vector angle is efficient in hue domain. Thus the methods can be combined using a trade-off parameter calculated based on saturation or intensity.

The best results for the tested maps were obtained with the vector gradient edge detector with a saturation-based combination of hue and intensity planes, denoted by $C_{GV}$.

$$C_{GV} = \max_{i=1,...,8} \{ \sqrt{\alpha(S_i) \cdot \alpha(S_0)} \cdot VA(\vec{v}_i, \vec{v}_0) + (1 - \sqrt{\alpha(S_i) \cdot \alpha(S_0)}) \cdot ED(\vec{v}_i, \vec{v}_0) \} \quad (4)$$

where $i=0$ is the central pixel, and $i=1 \ldots 8$ are the neighboring pixels in $3 \times 3$ window.

The trade-off parameter $\alpha$ is defined by:

$$\alpha(S_i) = \frac{1}{1 + e^{-75(S_i-0.25)}} \quad (5)$$

where $S_i$ is the saturation of the pixel $i$. The saturation is that aspect of perception that varies most strongly as more and more white light is added to a monochromatic light [10]. As the value of saturation increases, the corresponding colors (hues) vary from unsaturated (shades of gray) to fully saturated (no white component).

The following relation gives the Euclidean distance:

$$ED(\vec{v}_1, \vec{v}_2) = ||\vec{v}_1 - \vec{v}_2|| = \sqrt{(v_{1,1} - v_{2,1})^2 + (v_{1,2} - v_{2,2})^2 + (v_{1,3} - v_{2,3})^2} \quad (6)$$

where $\vec{v}_i = [v_{1,1} \ v_{1,2} \ v_{1,3}]^T$ is a color triplet.

Vector angle is an alternate metrics and is defined in the next equation:

$$VA(\vec{v}_1, \vec{v}_2) = \sqrt{1 - \left( \frac{\vec{v}_1^T \cdot \vec{v}_2}{||\vec{v}_1|| \cdot ||\vec{v}_2||} \right)^2} \quad (7)$$

The image yielded by the edge detector algorithm is then thresholded using a relative strength of maximum 35%. The threshold value was selected in order to obtain thick enough edges, while losing less possible elevation data. The thresholding process gives a mask image for the linework within the map.

A morphological majority type operation is applied to the mask. The purpose of this step is to remove isolated 1 pixel (or small isolated groups of 1 pixel) for avoiding the presence of isolated linework pixels within the resulting map after overlapping the preprocessed map with the mask. We applied this morphological operation for maximum 3 times on the mask considering a trade-off between keeping as much as possible elevation data and removing as much as possible linework pixels.

Finally, the map image is overlapped with the mask, and the resulting image will contain white pixels to indicate the linework.

### 4 COLOR CLUSTERING

The white gaps left within the map after linework removal can be filled using the Wise’s algorithm [4]. But this method works well, only, with solid colored blocks. Thus we need to have a distinct color for each distinct elevation area that appears in the image. The WHMM preprocessing step has removed the dithered effect partially making easier the job for the edge detection algorithm. At this moment it is possible to apply a vector median filtering, because there is no more linework to detect. The $3 \times 3$ window median filtering is a preparation step for the color-clustering algorithm.

The minimum variance quantization algorithm is an efficient way for clustering colors [7]. Unfortunately, before we can apply the algorithm, we need to edit the image map manually. On tested maps the problem appeared with larger lakes. They cannot be fully eliminated by the linework removal algorithm, so the resulting images present some blue isles surrounded by white pixels. They can be easily eliminated using manually editing. The blue spots have high hue and intensity similarity with some green spots that denotes elevation between 0 and 100 m. So the automated process cannot be implemented so easy. On the other hand they need to be removed in order to reduce distortion in the clustering process.

The minimum variance quantization algorithm needs as an input the number of different elevation region within the image plus one (the white linework pixels). The specified number determines the number of optimally boxes into which the RGB color cube is divided.
The algorithm works by associating pixels into groups based on the variance between their pixel values. A set of pixels may be grouped together because their distance is smaller than a specific value from the center pixel of the group. The boxes that divide the color cube vary in size, and do not necessarily cover the entire color cube. The quantization allocates more of the colormap entries to colors that appear frequently in the input image and fewer entries to colors that appear infrequently [7].

The color clustering algorithm worked well with the maps we tested. However due to incompletely removal of the dithered effect, some isolated wrong classified pixels may appear and they need a manually removal.

5 GAP FILLING

In order to fill in the gaps left by linework removal a multi-pass majority filter – the Wise’s algorithm – [4] can be run on the color-clustered image to reclassify the white pixels on the basis of their surrounding pixels. The filter functions by taking a 3 pixel wide processing window, and then testing whether the central pixel of the window is a white pixel (linework pixel). The linework pixel located in the center of the analyzing window is then replaced by the most abundant, non-linework, colored pixel value from the surrounding 3×3 window. For large areas of white pixels, multiple passes of the algorithm are needed. For the tested images, we needed to run the algorithm, in the worst case, for 17 times in order to remove all the white pixels. The algorithm behaved well considering that the images contained a lot of unknown elevation information (linework pixels). However for a good enough result a manual editing is needed, too.

6 ELEVATION CONTOURS EXTRACTION

Having a solid color-coded relief map with all unwanted information removed we can easily extract the elevation contours by performing a sliding-neighborhood operation. Considering a 3 by 3 processing window, if at least one horizontal or vertical neighbor is different than central pixel, the color for the central element is preserved, otherwise the new color is white (not a border pixel). Next, the color-coded contours are converted into elevation (a sparse DEM). First, the colors are indexed in elevation ascending order as in the map’s legend. Then, a 3×3 sliding window is applied. If the central element is white it yields a zero in the output matrix. Otherwise, if the color index has neighbors with less index value, it produces the lowest elevation value denoted by the color, else it gives the highest elevation value denoted by the color.

7 INTERPOLATING BETWEEN ELEVATION CONTOURS

By converting the color-coded contours into elevation we obtained a sparse data set. Laplacian partial differential equations (PDEs) can be used to interpolate the sparse DEM. If this equation is solved by iteration on a grid, then the elevation of each point in the array, whose height is not already fixed, is the average of its four neighbors [9].

In [11] and [2] it is suggested the following formulation for the Laplacian system of equations:

- Considering that the sparse elevation data set is a N by N matrix, pretend that all the N² = M points have unknown elevations $z_{ij}$.
- Create an equation for each $z_{ij}$ setting it to the average of its neighbors as in the equation:

$$z_{i,j-1} + z_{i,j+1} + z_{i+1,j} + z_{i-1,j} - 4z_{ij} = 0 \quad (8)$$

- For each of the $K$ points whose elevation $e_i$ are known create an additional equation $z_i = e_i$. 

![Figure 1: The process of DEM extraction from a color-coded relief scanned map.](image)
This results in the system of equations:

\[ A \cdot z = b \]  \hspace{1cm} (9)

where \( A \) is an \((M+K)\) by \( M \) coefficient matrix - a sparse representation of the Laplacian PDE matrix, \( b \) is a \((M+K)\) by 1 vector of zeros or known elevation values - the right hand side (RHS) vector, and \( z \) is a \( M \) by 1 vector - the solution vector.

In order to solve the system, [2] proposes the LSQR implementation of conjugate gradients on the normal equations. The algorithm was developed by Paige and Saunders [12]. The iterative solver is efficient both from a computational and memory use standpoint, and it allows a certain degree of flexibility because solution accuracy may be traded off for decreased run time if necessary. It solves \( A z = b \) for consistent systems or minimizes \( ||Az - b||^2 \) or \( \sqrt{||Az - b||^2 + d||z||^2} \) for over determined systems, where \( d \) is a damping parameter. The solver uses a method based on Golub-Kahan bidiagonalization. It is algebraically equivalent to applying the symmetric conjugate gradient (iterative) method to the normal equations:

\[ (A' A + d^2 I) z = A' b \]  \hspace{1cm} (10)

but according to the authors has better numerical properties, especially if \( A \) is illconditioned.

For the tested images (128 by 128 pixels) we applied the LSQR algorithm considering a tolerance of \( 10^{-6} \) and a maximum number of iterations of 5000. For most of the tested images the algorithm ended, approximately, at iteration 3000.

8 CONCLUSIONS

In this paper, we tried to find the steps necessary for extracting a digital elevation model from a color-coded relief map and to offer a solution for each of them. The map was preprocessed using a WHMM based algorithm for denoising. Then a linework removal algorithm was applied. Two approaches of linework identification were presented: one based on wavelet domain hidden Markov models and one that uses a vector gradient edge detector. Each distinct elevation region was distinctly solid colored using a vector median filtering and a color clustering process based on the minimum variance quantization method. A multi-pass majority filter algorithm was used to fill in the linework gaps. The color-coded contours were extracted by detecting the borders between colors through a sliding-neighborhood operation and they were converted into a sparse elevation data set by means of the map’s legend. Laplacian partial differential equations were used to interpolate between contours and the LSQR iterative solver was applied in order to solve the system of equations.

Some steps of the method (e.g. WHMM, Wise’s, and LSQR algorithms) are computational expensive and the process cannot be fully automated. But the approach can be a good help for the map digitizing domain when used with manual editing. Our future work may be concerned with the improving of the automation problem and finding other solutions for some of the method’s steps (e.g. interpolation of the sparse DEM).
Figure 3: Experimental results. Linework removal using wavelet domain hidden Markov models - we used a Haar wavelet filter and a convergence error of $10^{-5}$.
(a) The resulting map image after WHMM preprocessing step; (b) The linework mask that combines only the horizontal and vertical edges - median filtered based on a $3 \times 3$ processing window; (c) Linework removal using the mask from (b); (d) The linework mask (it includes, also, only the horizontal and the vertical edges) - no median filtering was applied; (e) Linework removal using the mask from (d). By median filtering the mask, we can eliminate the isolated remaining linework pixels.

Figure 4: Experimental results. Linework removal using vector gradient edge detector with saturation based combination of hue and intensity.
(a) The resulting map image after WHMM preprocessing step; (b) The resulting image after edge detection; (c) The image from (b) is thresholded using a relative strength of 25%; (d) Linework removal using the mask from (c); (e) A morphological majority type operation is applied (2 times) to the mask from (c); (f) Linework removal using the mask from (e).

References


Figure 5: Experimental results. Median filtering before color clustering step; (a) The resulting image after linework removal using vector gradient edge detector with saturation based combination of hue and intensity (the image from Figure 4-f) (128 × 128 pixels); (b) Zoom of the map image from (a) (32 × 32 pixels); (c) A 3 × 3 window vector median filtering based on Euclidean distance was applied on the image from (a); (d) Zoom of the map image from (c).


Figure 7: Experimental results. Gap filling, Manual editing, Contours extraction, Interpolation; (a) The resulting image after color clustering process (the image from Figure 6-b); (b) The gap filling algorithm - Wises’s method, which is a multi-pass majority filtering approach - was applied (14 times) on the image from (a); (c) The image from (b) was manually edited; (d) The elevation contours were extracted based on the image from (c); (e) 3D representation of the DEM obtained using Laplacian partial differential equations to interpolate between elevation contours and the LSQR iterative algorithm for solving the system of equations (we considered a tolerance of 10^{-6} and a maximum number of iterations of 5000).


