Generation and Authoring of Augmented Reality Terrains Through Real-Time Analysis of Map Images

Theodore Panagiotopoulos, Gerasimos Arvanitis(✉), Konstantinos Moustakas, and Nikos Fakotakis

Electrical and Computer Engineering,
University of Patras, Patras, Greece
theopanag7@gmail.com,
{arvanitis,moustakas}@ece.upatras.gr, fakotaki@upatras.gr

Abstract. In this paper we present a novel method for real time 3D terrain creation and augmented reality rendering based on contour map images. Initially, terrain information is extracted from the contour images and a flat Delaunay triangulated terrain is generated. Then elevation information is added and remedying is performed so as to maintain smooth local surface representation. Then augmented reality rendering is performed in real time using the 2D contour map as a marker to manipulate the 3D terrain. The proposed framework also demonstrates potential editing applications like manual design of auxiliary information on the contour map, like roads, that can be automatically converted into 3D information and rendered on the 3D terrain, thus resulting in a more immersive and intuitive design experience.

Keywords: Augmented reality · Contour lines · Topographic maps

1 Introduction

The main augmented reality (AR) paradigm is related to the superimposition of digital-virtual information on top of the real environment, usually in real time. The virtual objects are traditionally rendered using devices like mobile phones or specialized augmented reality glasses. This powerful new technology brings out the components of the digital world (graphics) into a person's perceived real world by rendering them in a suitable device. Despite the significant advantages, there are still some limitations that are yet to be overcome. A challenging issue is the real time scene acquisition and scene generation.

Cartography is a traditional scientific field with numerous applications, like navigation systems and other contemporary applications. A topographic map is a map characterized by large-scale detail and quantitative representation of relief, usually using contour lines. The traditional way of reading contour lines in such a map is not an easy and intuitive task especially for people without experience.

The proposed framework introduces a computationally efficient image-based approach for real time topographic map sensing and reconstruction. A 3D terrain is thus created and rendered into an augmented reality environment. The 3D representation of topographic maps is more intuitive and engaging for the users. Moreover, it can help users, like civil and surveying engineers or students, to better understand the topography of contour lines, thus increasing perception and immersion.

1.1 Related Work

A method for extraction of contour lines from paper-based topographic maps is presented in [1]. In [2] authors propose an automatic approach to reconstruct gaps in contour lines. This parameterless reconstruction scheme is based on the extrapolation of the gradient orientation field from the available pieces of thinned contours. In [3] an application is presented using camera and HMD for presenting augmented maps. In [4] an application is presented that allows the generation of virtual terrains interactively, using augmented reality markers. This application also allows the user to navigate in the generated virtual environment. [5] describes an educational Augmented Reality application for mobile phones that supports landscape architecture students who learn how to read and analyze contour maps. AR Sandbox [6] is an application that allows users to create topography models by shaping real sand, which is then augmented in real time by an elevation color map, topographic contour lines, and simulated water. The system teaches geographic, geologic, and hydrologic concepts such as how to read a topography map and the meaning of contour lines. In [7] authors developed a browsing tool for visualizing information about geographic surfaces using map-based augmented reality.

In most cases, the terrain has already been created and then is superimposed in the real environment as dictated by the position and orientation of the marker. On contrary the proposed scheme makes dual use of a contour map. First of all it extracts the 3D terrain information dynamically and secondly uses the map itself as a marker to superimpose the 3D terrain.

1.2 Contribution

The contributions of our work are outlined below:

- Fast image processing and rendering for converting the information of a printed map into the corresponding 3D terrain.
- The created 3D objects are rendered into user's mobile device using as marker the topographic map itself.
- An example of dynamic interaction between users and the application is showcased, allowing the users to design roads on the 2D maps that are automatically converted and superimposed on the 3D terrain.

The paper is organized as follows: Sect. 2 presents a brief overview of the proposed system. In Sect. 3 we describe the algorithms and the basic steps that we follow for the image processing analysis. In Sect. 4 we analyze the way that the 3D objects are created. Moreover, we show how the 3D objects are displayed on user's device and we also present the way for user-application interaction by drawing roads in real time. In Sect. 5 we provide experimental evaluation of the proposed framework, while in Sect. 6 conclusions, limitations and future work is discussed.

2 Framework Overview

The proposed system accomplishes two main operations, the creation of 3D models and the 3D object rendering. For the 3D model creation we use image processing techniques that help us to classify the contour polygons of the topographic map and based on this classification we create the 3D object by performing iterative Delaunnay triangulation. The 3D model creation is the most computational consuming process, thus we use a server for the processing instead of a smartphone. In Fig. 1 we show the architecture scheme of our method. Details and specifics on the building blocks of the proposed framework are presented in the next sections.

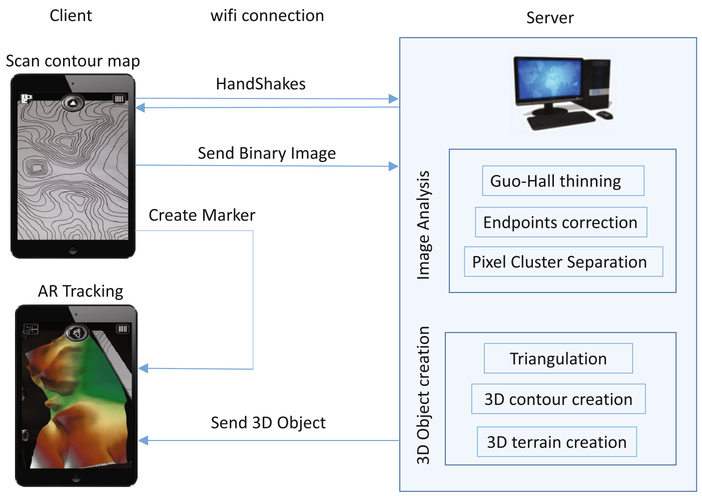

Fig. 1. Architecture scheme of our proposed method

3 Sensing and Image Processing

The sensing and image processing module initially converts the map image in binary form, then performs skeletonization that is followed by denoising so as to finally allow for smooth extraction of the contour lines.

Binary Image. For the creation of binary image we apply an adaptive threshold filter as described in Eq. (1).

$$I_b(i) = \begin{cases} 1 \; if \;\; A(i) < \frac{\sum_{n=1}^{|N_i|} A(N_i(n)) - \delta}{|N_i|} \\ 0 \qquad\qquad otherwise \end{cases} \quad \forall \; i = 1, L \tag{1}$$

where I_b represents the binary image, $A(i)$ is the intensity of i pixel, $|N_i|$ is the number of i's neighbors, $N_i(n)$ is the n neighbor of i pixel, L is image length and δ is a small positive value. The adaptive threshold filter has satisfying results even in cases with bad light conditions, as we can see in Fig. 2.

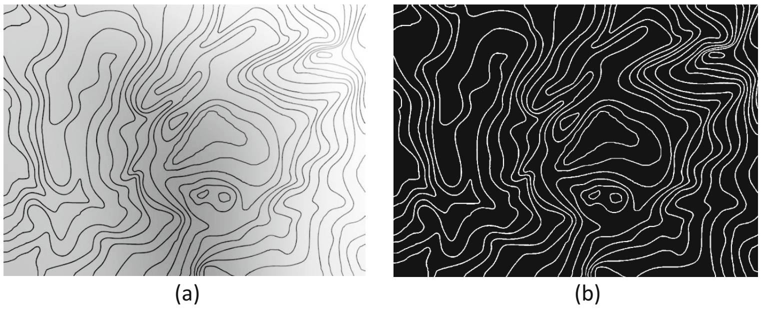

Fig. 2. (a) Bad light condition for a grayscale topographic map, (b) binary image created by adaptive threshold filter

Skeletonization and Edges Correction. The binary image is skeletonized using the Guo-Hall thinning algorithm [8] so all lines have the same thickness and we can continue with the classification. An effective contour map model requires connected lines (polygons). However, there are cases where polygons are not continuous. We handle this issue by using the following very simple three step algorithm: (a) neighbors pixels are classified into the same cluster, (b) clusters with low number of pixels are considered as noise and are removed, (c) lines with common points are classified into the same Pixel Cluster, even if they are different polygons (we deal with this limitation in the next step). There are two types of pixels, those that have two or more neighbors and those that have only one (edges) Fig. 3(a). Our goal is to connect all edges with others of the same Pixel Cluster until no edges exist. The parameters that we take into account for applying an efficient connection are: The distance d between two edges and the angle of edge's vectors $\theta_w = 180° - \hat{\theta}_{w_1 w_2}$. We assume that if the extended directions of two edges are connected then it is more likely that they are connected with each other. In the general case we connect edges that maximize the following equation:

$$S = \frac{1}{d \cdot \theta_w} \tag{2}$$

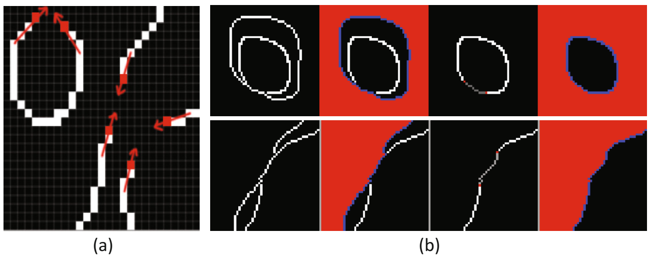

Fig. 3. (a) Example of pixels with only one neighbors (edges) (b) steps for Pixel Cluster classification

Pixel Cluster Separation. As we mentioned earlier, there are cases where two or more different polygons have common pixels as the result of being incorrectly classified into the same Pixel Cluster. Figure 3(b) shows the steps that we follow in order to handle these situations. We start the separation from the outwards polygons. We classify all the pixels into the same class and remove them. Inevitably, lines with missing pixels are created but we complete them and continue with the same procedure until all polygons are separated and classified. At the end we have managed to ensure that each one of the Pixel Clusters is a continuous line that can easily be recognized as a different poly-line.

4 Geometry Processing and Rendering

A 3D model can be represented as an indexed face set $M = (V, F)$, being composed of vertices (V) and the indexed faces (F). Each vertex can be represented as a point in space $v_i = (x_i, y_i, z_i)\ \forall\ i = 1, k$. In this case we create a vector of vertices $\mathbf{v} = [\ \mathbf{x},\ \mathbf{y},\ \mathbf{z}\ ]$ in a 3D coordinate space such as $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \Re^{k \times 1}$ and $\mathbf{v} \in \Re^{k \times 3}$. This means that we have a set of k points such that $V = \{v_1,\ v_2,\ \ldots\ v_k\}$. Additionally, each face is represented as a set of 3 vertices $f_i = [v_{i1},\ v_{i2},\ v_{i3}]\ \forall\ i = 1, m$ where $m > k$, so we have m faces $F = \{f_1,\ f_2,\ \ldots\ f_m\}$.

3D Terrain Model Creation. The Pixel Clusters classification step helps us to recognize the different areas with different heights as shown in Fig. 4(a). We colorized each Pixel Cluster with different color in order to be separated from each other. The sea level (outer level) area is represented always by black color Fig. 4(b). We define each area's relative height H in comparison with the sea level area. We start by setting the black area with value H = 0 and subsequently, for each other area with common border the H value increases by one. We continue the procedure using the same logic and at the end all areas will have been defined.

The previous procedure is a vital step, however, additional information is required for the 3D terrain creation. Indeed, we need to assign each pixel i of the image (polygons and internal pixels) with a height value $h_i\ \forall\ i = 1, L$. For

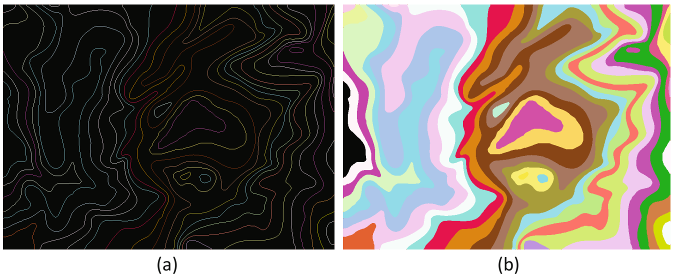

Fig. 4. (a) Separated pixel clusters, (b) separated cluster areas (Color figure online)

finding this value, we use linear interpolation and we estimate the weighted average of the distance d between pixel and its relative polygons. More specifically, assuming that pixel i lies between polygons $c1$ and $c2$, with relatives heights H_{c1} and H_{c2} correspondingly, then its height value is estimated according to:

$$h_i = \frac{d_{c1} \cdot H_{c1} + d_{c2} \cdot H_{c2}}{d_{c1} + d_{c2}} \tag{3}$$

where $d_c = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2}$. However, in cases where the area does not contain other polygons (peaks or valleys) we need to create an auxiliary edge-line that represents a higher or lower height. After this adjustment we can use the linear interpolation as previously. In order to create this auxiliary edge-line we skeletonize the area using the Guo-Hall thinning algorithm. Depending on the case, the edge-line takes value +0.5 if the area is a peak or −0.5 if the area is a valley. In Fig. 5 we can see an example of an auxiliary edge-line creation for an area without other enclosed polygons.

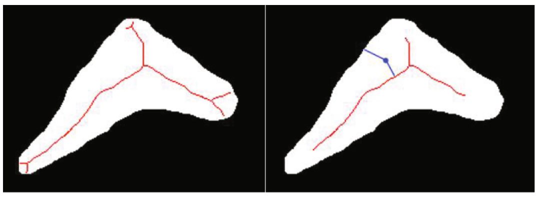

Fig. 5. Polygon's skeletonization

At the end we assign each pixel to a corresponding color based on its height value. The colormap that we use is presented in Fig. 6(c).

Triangulation. A 3D object is represented by a number of vertices that create triangle faces when they are connected with each other. We have knowledge about the position of the vertices based on the position of the pixels and the height information. However, there is lack of knowledge about how these vertices are connected with each other. Triangulation is the solution but we need to go further and investigate the ideal number of points that is needed for the representation. On the one hand, we need a lot of points for preserving detail but on the other we should use the fewest possible points so as not to increase the computational complexity of the framework.

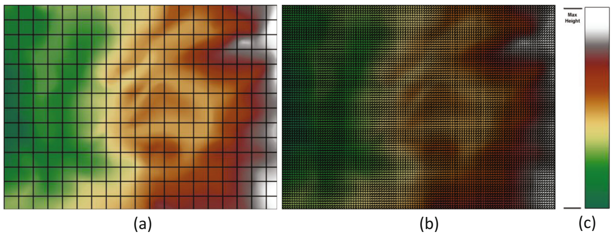

Fig. 6. (a) An example of a grid triangulation where each square is divided into two triangles, (b) a satisfying grid, (c) colormap (Color figure online)

In the context of the proposed framework we used grid triangulation. Figure 6(a) shows a simple example of a grid triangulation and Fig. 6(b) shows a satisfying grid triangulation that can be used in our case. The only thing that we need to consider is the need for more accuracy in situations where a steep slope exists, but using this type of triangulation this requirement is satisfied.

3D Contour Model Creation. Before the triangulation a process of sampling is necessary. In Fig. 7(a) we can see how the sampling algorithm is applied to a polygon, while in Fig. 7(b) we can see a sampling example.

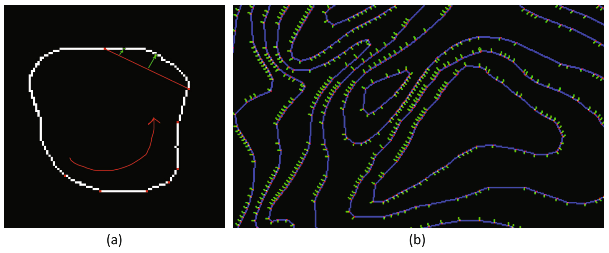

Fig. 7. Example of sampling operation for contour polygons

The sampling algorithm starts by randomly choosing a pixel p_r of a polygon and a straight line is created $l_{p_r p_{r+1}}$ by connecting it with its first counterclockwise neighbor pixel p_{r+1}. The Algorithm 1 describes how it is used in the general case.

```
$polygon = 1$;
while $polygon < polygon.length$ do
    $\lambda = 1$;
    while [Unsearching pixel exists] do
        for $i \leftarrow 0$ to $\lambda$ do
            estimate $l_{p_r p_{r+\lambda}}$
            if $max(p_{r+i} \perp l_{p_r p_{r+\lambda}}) > O$ then
                $r = r + \lambda - 1$ ;                // set a new initial pixel
                $\lambda = 1$;
            else
                $\lambda = \lambda + 1$ ;    // continue with the next counterclockwise
                 neighbor pixel
            end
        end
    end
    $polygon = polygon + 1$;
end
```

Algorithm 1. Sampling each polygon

We can change the sampling results by changing the threshold O. The experiments show that sampling by using a very low value of O we can decrease the number of vertices ∼88% without significantly decreasing detail. For the triangulation we use the Ear Clipping algorithm as it is described in [9].

4.1 Rendering and Augmented Reality Interaction

At every augmented reality application a special marker is needed that represents the specific area in which the virtual object will appear. This marker must be easily recognizable, meaning that it must has high frequency features like edges. There are two different solutions known as AR marker-based (with target) and AR markerless tracking (without target) [10]. In our case the contour map itself can be used as a marker. For the 3D object presentation [11] we use a transformation $\mathbf{T} = \mathbf{v}\ x\ \mathbf{Q}\ x\ \mathbf{P}$ in order to estimate the points $\mathbf{T} = \{u, m\}$ that are displayed in the screen. where $\mathbf{Q}$ is the Model View Matrix and $\mathbf{P}$ is the Projection Matrix.

4.2 Intuitive, Interactive Authoring

The developed application provides tools that allow users add information with two different ways: (a) by selecting points (b) by drawing lines. In the first case we just have to create a graph and subsequently to present it in the 3D model. In the second case we need to follow the same approach but an image processing step takes initially place.

Create Graph. For the creation of the graph we need to assign each of the selected pixel to the ideal representative faces. Firstly, each one of the selected pixel is converted to a 3D ray. Subsequently, we apply a ray casting algorithm and we choose the triangle faces with the shortest distance.

Image Processing for Drawn Line. Firstly, we separate the colored road from the rest lines. We transform the RGB image to HSV, making the color separation easier. This step gives us a binary image with very good appearance but there are some missing areas Fig. 8 (color recognition). To cover the imperfections we apply a dilation-erosion filter that achieves the creation of a continuous line (road). Next, we apply the Guo-Hall thinning algorithm in order to skeletonize the road. The last step is the creation of a graph from the skeletonized road. For this purpose some extra steps are required: (a) **Extra thinning.** Although removing only ∼1 − 2% of pixels, this procedure is vital for our purpose (b) **Crossroads Finding.** We assume that if any pixel has 3 neighbors then it is crossroad (c) **Remove small areas.** If the length of lines between two endpoints is shorter than a threshold then it is removed. (d) **Road sampling.** The same sampling procedure is used as in polygon sampling case, (e) **Graph creation.** After that we have the necessary points for the graph's creation.

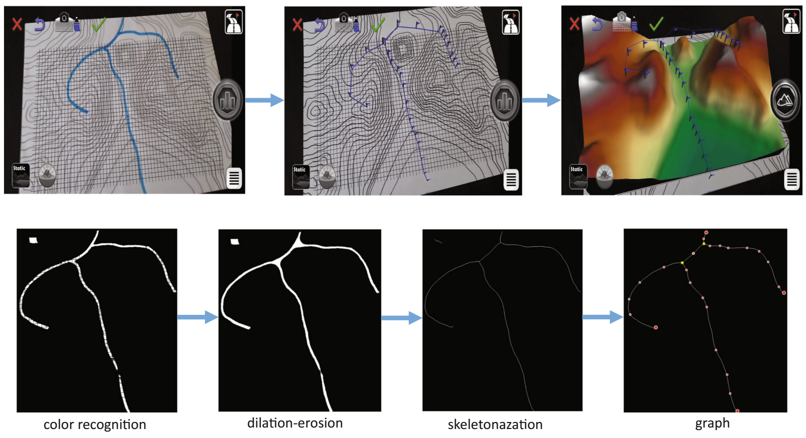

Fig. 8. Steps for real time 3D road creation and rendering in the 3D models by a drawn line. (Color figure online)

5 Experiments and Results

5.1 Timings and Complexity Analysis

Having a stable wifi connection, the time for sending (binary image) and receiving (two 3D models) data between smartphone and server is very fast ∼1 s. More time consuming operations, like image processing techniques and 3D model creation, take place in the server exclusively and take 2–5 s to be completed.

The most computational consuming step is the skeletonization algorithm (Thinning) that can correspond to the 20% of the total time. A typical smartphone is capable to handle the rendering of 3D models with approximately 20.000 faces, in augmented reality at interactive rates.

5.2 Examples

We have examined several scenarios using different conditions in each case. Below we present the results of three different experiments representing the most common cases appeared in real scenarios. In the first experiment a significant amount of noise is present, due to bad illumination conditions, Fig. 9. In the second experiment, we have different light conditions, we use a little different color of ink and the paper of the map is a little tearing, as shown in Fig. 10. The third experiment has better light conditions however the contour lines are very closed each other, as illustrated in Fig. 11. The experimental results show that the proposed approach is capable to efficiently manage different case studies overcoming problems caused by bad illumination, different color of ink or illegible lines.

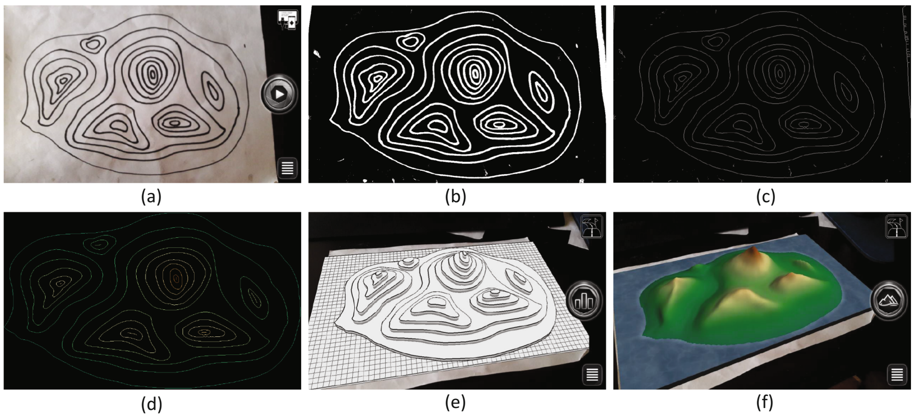

Fig. 9. Experiment 1. (a) Topographic Map, (b) Binary Image, (c) Skeletonized map, (d) Contour Clusters, (e) 3D contour Model, (f) 3D terrain Model

5.3 Hardware Setup and Libraries

We need to manage both client (android smartphone) and server (computer) operations. For the server, the code for the image processing and for 3D objects creation is written in C++. For the client, we use java scripts for the connection with the server and for 3D object presentation and display. The external libraries that are used, are: Vuforia (Augmented Reality), OpenCV (Computer Vision) for image processing, PolyPartition (ear-clipping) for triangulation.

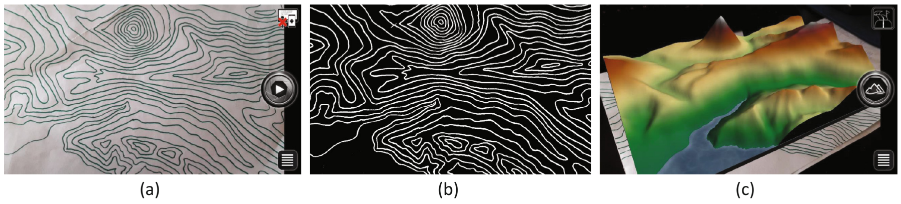

Fig. 10. Experiment 2. (a) Topographic Map, (b) Binary Image, (c) 3D terrain Model (Color figure online)

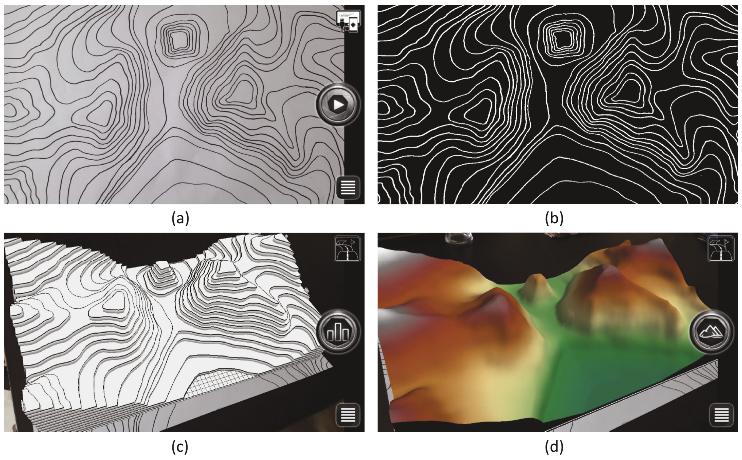

Fig. 11. Experiment 3. (a) Topographic Map, (b) Binary Image, (c) 3D contour Model, (d) 3D terrain Model (Color figure online)

6 Conclusions

We proposed a framework for the reconstruction of 3D terrains based on topographic maps that are subsequently rendered in an augmented reality environment. Preliminary application scenarios demonstrate the potential of the approach for the intuitive and interactive investigation/editing of contour maps.

6.1 Limitations

Despite the fact that the proposed method operates reliably in most of the cases, there are some limitations that may affect the final results, such as: (a) It needs a good resolution camera. (b) We use topographic maps without extra symbols or colors just only simple contour lines.

6.2 Future Work

In a future edition of our work we will try to overcome some of the limitations that we mentioned before and additional to add more new features. Some futures extensions of our work will be: (i) Digit recognition of the contour map, for an automated height estimation. (ii) Simplified contour maps by removing extra symbols. (iii) Adding more user interaction tools in order to support manual creation and handling 3D object structures (e.g. bridges).

References

1. Xin, D., Zhou, X., Zheng, H.: Contour line extraction from paper-based topographic maps. J. Inf. Comput. Sci. **1**(5), 275–283 (2006). ISSN 1746-7659, England, UK
2. Pouderoux, J., Spinello, S.: Global contour lines reconstruction in topographic maps, pp. 779–783, January 2007
3. Bobrich, J., Otto, S.: Augmented maps. In: Symposium on Geospatial Theory, Processing and Applications, Ottawa (2002)
4. Dembogurski, R., Sad, D.O., de Souza Filho, J.L., Silva, R., Vieira, M.B., Dembogurski, B.: Interactive virtual terrain generation using augmented reality markers. SBC J. 3D Interact. Syst. **3**(3), 29–36 (2012)
5. Schroth, O., Zhang, C.: Augmented landform an educational augmented reality tool for landscape architecture students. In: Peer Reviewed Proceedings of Digital Landscape Architecture, ETH Zurich (2014)
6. Augmented Reality Sandbox. http://idav.ucdavis.edu/~okreylos/ResDev/SARndbox/
7. Asai, K., Kondo, T., Kobayashi, H., Mizuki, A.: A geographic surface browsing tool using map-based augmented reality. In: 2008 International Conference Visualisation, London, pp. 93–98 (2008)
8. Guo, Z., Hall, R.W.: Fast fully parallel thinning algorithms. J. CVGIP Image Understand. **55**(3), 317–328 (1992)
9. Eberly, D.: Triangulation by ear clipping. In: Geometric Tools, LLC, November 2002
10. Wagner, D.: Handheld Augmented Reality, Graz, Austria (2007)
11. Ginsburg, D.: OpenGL ES 3.0 Programming Guide, 2nd edn. Addison-Wesley Professional, New York