3D Map Generator

Vanshaj Agrawal[1], Vrushali Mehta[1],Yukti Doshi[1], Rushabh Chheda[2], Punit Lodha[2],Nemil Shah[2], Revathi AS[3]
[1] U. G. Student, Department of Electronics and Telecommunication, D.J.Sanghvi College of Engineering, Vile Parle (W), Mumbai- 400056
[2]U. G. Student, Department of Computer Engineering, D.J.Sanghvi College of Engineering, Vile Parle (W), Mumbai- 400056
[3] Assistant Professor Electronics and Telecommunication Department D.J.Sanghvi College of Engineering, Vile Parle (W), Mumbai- 400056
E-mail: [1]vanshajagrawal@ymail.com,[1]vrume2813@gmail.com, [1]yuktidoshi@gmail.com,
[2]punitlodha@pm.me,[2]nemilyshah@gmail.com, [2]rushabhchheda28@gmail.com, [3]revathi.AS@djsce.ac.in

***Abstract*— 3D modelling techniques for interacting with 3D objects that have up till now been constrained to only 2D representations, have become attractive with recent advances in computer vision. The scope of the presented work is the design and performance of a system to effectively visualize a 2D physical map by transforming it into an interactive 3D model. The 3D digital elevation model is generated such that the height of a particular point is proportional to its whiteness on a special gray scale 2D map called a height map. The paramount task is to derive a suitable 2D height map from the physical map based on its colour conventions which are different for terrains of different heights. This system extracts the individual terrains using the centroid based K-Means Algorithm to cluster map colours, adaptive thresholding, edge detection, terrain contour mask creation, contour inscribing, discrete time Fourier transform and several other techniques. It is implemented through the OpenCV Python library. Suitable textures, materials and custom shades are applied to the 3D model. An online engine and Android app for easy interaction is developed. Subsequent implementations are to have the ability to view the 3D model through AR and travel inside it using VR.**

Keywords—OpenCV, 3D Digital Elevation Models, Heightmap Generation, Machine Learning, K-Means Algorithm, Mask Extraction, Image Processing, Python, Computer Vision

I. INTRODUCTION

A. Background

Over the past few millennia, methods of navigation and educational aids to teach the same have largely remained the same. Maps, quite similar to the colour coded maps which were used by the merchants travelling on the Silk Road from India to the Mediterranean, are still being used today. Yet, greater accuracy and more distance measurements can never dispel the need for visualization methods which provide more information about the terrain and that too in a natural way.

Education too urgently needs better tools to capture the interest of students and facilitate the transfer of knowledge.

Recent advances in Computer Vision and Object Recognition have shown great promise in providing alternative methods to model geographic data, from predicting suitable paths for autonomous navigation toComputer Image Processing to produce interactive VR based renderings for architects.

B. Motivations for building a 3D Map Generator

Inspired by the quote ‘Once a new technology rolls over you, if you're not part of the steamroller, you're part of the road.’, we have sought to apply cutting edge innovations and emerging technologies to mapping and navigation. Easy and quick access to 3D maps has tremendous applications amongst which are:

A. From being used in data acquisition systems by vehicle manufacturers to determine vehicle performance on specific slopes, to military usage for simulating the performance of vehicles and tanks on real-life locations in the event of a war.

B. 3D maps are soon to become mainstream in flow simulations before building dams to determine the possible areas for flooding due to overflow and excessive rain. [1]

C. Usage by amateur hikers and mountain climbers for planning the best possible routes is expected to grow in the next few years. [2]

D. Topography and Map Reading taught in schools all over the world can be made much more accurate, interactive and interesting through 3D modelling techniques. [3]

E. Video games can soon include custom real-life surroundings, added by users, such as their very own neighbourhood.

F. The possibilities for growth in this sector are thus limitless.

II. APPROACH

The objective of the system designed was to be able to take a photograph of a physical map, and convert it into a 3D model with the height of each point of the 3D structure, being proportional to the whiteness of the grayscale Height Map[4]. This was to be produced from a Physical Map, sourced fromwidely available textbooks, and so, the most commonlyaccepted colour convention requirements were used tocategorize the terrains so that the algorithms would have thehighest probability of functioning on any given map. [5]

The ideal source map is primarily composed of the colours, brown (representing mountains), green (representing grassy fields and forests), yellow or ochre (representing plateaus or barren lands) and blue (representing water). The presence of any other colour such as white or black was to indicate the end of the map. [6]

The next objective was to be able to extract each separate terrain of the map, since they were to be given different heights in the 3D Model. Images are typically stored on a computer in the RGB Format. However, this model requires 3 separate channels or values for us to be able to classify a particular colour as reddish, bluish or greenish.

Moreover, colours perceived distinctly by humans may not be separated by as great a value in the RGB colourspace.

On the other hand, images in the HSV format require only one number to determine whether a particular colour is brownish, yellowish, bluish or greenish.

A colour is represented in the HSV Format by 3 values:

H (Hue): This represents the broad colour (brownish,yellowish, bluish or greenish)

S (Saturation): This represents how strong the shade is, suchas fluorescent yellow vs ochre

V (Value): This represents the value of brightness

To extract each terrain, each pixel of the image was compared and categorized into one of the colour categories. Thus, suitable bounding ranges for the colours present in the map were required. To obtain this range, the maximum and minimum values for the H, S and V of each of the 4 colours were noted for over 200 images of each shade using a colour picker. The H values of Brown and Yellow/Ochre one hand and Blue and Green on the other were quite distinct.

However, the H values for Brown and Yellow/Ochre greatly overlapped but the V value or brightness of the shades of Yellow were much higher and gave us a way to divide them on that basis. The H values of Blue and Green were also intersecting and so statistical methods were used to obtain the ideal range for separating them with the highest accuracy.

Similar statistical methods were used to determine the ideal upper bound and lower bounds for each of the colours, in order to be able to classify maximum colours possible while yielding minimum false positives.

The classification of each pixel was greatly simplified by the K-Means Clustering Algorithm Machine Learning Algorithm. [7]

K-Means Clustering Algorithm is one of the fastest unsupervised Machine Learning techniques. As K-Means is an unsupervised machine learning algorithm, the data doesn't require any labels. The objective of K-Means Clustering Algorithm is to group similar data points. In the K-Means Clustering Algorithm, data is divided into numerous Clusters. All data points are Clustered according to some similarity criteria. The users can specify the number of Clusters depending on their requirements. These Clusters can then be used to find some underlying pattern amongst the data points.

A. Algorithm 1: K-MEANS CLUSTERING ALGORITHM

i. The K-Means algorithm performs the Clustering Algorithm of various data points through the following process:

ii. The required number of Clusters (k) is selected.

iii. The algorithm randomly assumes some points as centroids.

iv. After the centroids are chosen, the similarity of each data point from all k centroids is calculated.

v. After calculating the similarity, each point is assigned to a Cluster.

vi. Then the centroid of all the points in a single Cluster is again calculated.

vii. The K-Means algorithm continues to repeat this process until the Clusters have become largely constant and there is no change in the coordinates of the centroid of any Cluster for a minimum of 2 iterations. The process may stop if the algorithm has run for the required number of iterations determined by the user.

The K-Means Clustering Algorithm can also be used for colour detection. As the images captured by cameras are affected by various factors like different lighting conditions, camera quality, and so on, colour identification can be a tedious task. However, since in terrain extraction the primary aim is to identify only the type of colour and the exact shades of various colours is not a binding criteria, the K-Means Clustering Algorithm can be used.

That is since the colour of each pixel is replaced by the colour of its centroid, the value of each shade of green needn't be compared to classify a pixel as red. Merely, the value of the pixel of the k centroids needs to be compared to classify every pixel into k regions or terrains. Thereby, RGB value of the pixel colours of these k centroids is thus very useful and K-Means is used to identify the prominent colours of the map.

Subsequently, the RGB values of these Clusters are compared with the RGB value of the colours used in physical maps.

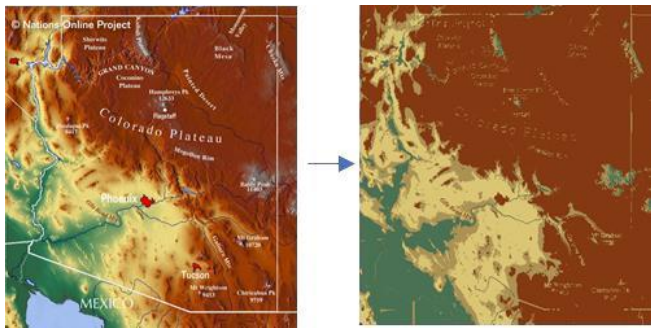


Fig 1: K-Means algorithm used by this system to remove city names and other undesirable noise

This determines the colour category for a pixel. Then these RGB values are converted into their corresponding HSV values to avail the aforementioned advantages, and each of the colours obtained using K-Means are compared to the preset range of colours that were obtained using the colour picker.

B. Classification Algorithm

An algorithm was designed to avoid the need to compare every pixel value with every value in the bounding range, in order to classify it. A standard deviation of (4,4,4) for (H,S,V) was introduced to accommodate for slight changes in colour of the pixels while interconverting them between different image formats. Each H value was iteratively compared to identify the region as belonging to one of 3 categories: Blue, Green and (Brown+Yellow/Ochre) since only these 3 regions had distinct H values.

Thus, the Blue and Green terrains were classified. Since Brown and Yellow/Ochre have overlapping H values they are distinguished on the basis of the statistical study performed, from which it was inferred that assigning the colour having the higher V value as Yellow/Ochre and the colour having the lower V value as Brown would give the maximum accuracy and minimum false positives.

C. Masking Algorithm

A blank 'black' coloured canvas of the same size as our image was created in the grayscale format. This was done to conserve memory and computing time since a predetermined set of operations are to be performed on each terrain according to its colour which has already been identified. Hence the extra information given by its HSV values are redundant. The colour 'white' was then assigned to the pixelsat the same coordinates in the canvas as the coordinates having that particular colour (say brown or blue) in the original image. This will also serve as the masking image for subsequent masking operations.

The white border to the region will serve as a contour, which refers to the bounding coordinates of the region of interest of the image. Image Processing Operations will only be performed within this region of interest. Thresholding was performed to increase the contrast between the bordering regions, introducing a certain amount of noise. The Morphological operations of erosion and dilation were performed using a convolution kernel that was experimentally optimized. This removed noise without losing image quality.

D. Algorithm 2: CLASSIFICATION ALGORITHM

i. Take the (H,S,V) of the first of the k cluster colours. Proceed to step 2.

ii. Compare its H value with the upper and lower bounds Hi and Hf value of Blue. If it lies between them, it's identified as Blue and the relevant operations for the water terrain are performed on it. If not, then proceed to step 3.

iii. Compare its H value with the upper and lower bounds Hi and Hf value of Blue. If it lies between them, it's identified as Green and the relevant operations for the grassy terrain are performed on it. If not, then proceed to step 4

iv. Compare its H value with the upper and lower bounds Hi and Hf value of Brown and Yellow/Ochre. If it lies between them, it's identified as Brown/Yellow/Ochre and the relevant operations for the grassy terrain are performed on it. If yes, then proceed to step 5. If not then jump to step 5

v. Compare the V value of the cluster centroid pixel with the upper and lower bounds Vi and Vf value of Yellow/Ochre which has a higher V value. If it lies between them, it's identified as Yellow/Ochre and the relevant operations for the plateau terrain are performed on it. If not then proceed to step 6

vi. Compare the V value of the cluster centroid pixel with the upper and lower bounds Vi and Vf value of Brown which has a higher V value. If it lies between them, it's identified asYellow/Ochre and the relevant operations for the plateau terrain are performed on it. If not then proceed to step 6.

vii. Since it doesn't match any of the 4 preset bounding ranges, it is classified as unsorted and the relevant operations for the unsorted terrain are performed on it.

E. Algorithm 3: MASKING ALGORITHM

i. Take all pixels with (H,S,V) ± value of the colour classified as C where is the Standard Deviation in (H,S,V) value of the pixels. Assign them colour white and the rest of the image as colour black through thresholding. A Grayscale Mask is obtained.

ii. Take the Mask and perform Morphological Operations to smooth the image.

iii. Proceed to perform the Contouring Algorithm.

F. Contouring Algorithm

To create the peaks for mountainous regions, the concept of contours was used, which are as explained, curves joining all the contiguous points (along the boundary), having the same colour or intensity. Image moments were used to compute the centroid of the contours. For the brown region, a mountainous terrain was needed. Consequently, using a loop, the distance of the coordinates of the contours from the centroid were reduced and simultaneously, their whiteness was increased, leading to a corresponding increase in their heights in the 3D Model. This was observed to generate a mountain with a peak at the centroid, in the 3D Model. [8] The range of whiteness for the Yellow region was lesser in magnitude and slower in frequency, yielding a plateau like structure in the 3D Model. The contouring algorithm was performed for brown to get mountains and yellow/ochre to get plateaus. Brown had whiter contours drawn, giving it a greater height and gradient in the 3D Model while Yellow/Ochre had less whiter contours drawn, giving it a lesser height and gradient.

The grassy terrain was assigned a constant grayish tone [pixel value of 20] since fields tend to be at a greater slope than water. Gaussian noise with an optimized standard deviation was used to introduce a specific pattern which was observed to give a 'grass like effect' in the 3D model. The mask for Blue was not extracted since its canvas was already assigned black colour which would yield the lowest depth, such as that of a river valley or lake, in the 3D model.

The terrain having an unclassified colour was given a constant grayish tone; lower than that of grass, to denote it as a distinct region, not part of the map. The option to add a 'grassy/gaussian', 'poisson-like' or 'salt-and-pepper' texture was also given.

G. Algorithm 4: CONTOURING ALGORITHM

i. Obtain image moments for each contour. Calculate Centroid.

ii. For Brown terrain, obtain concentric smaller contours, by performing for every point:

point = point - centroid

point = point * k

point = point + centroid

Here k is an increasing constant, k ∈ (0,1)

As the constant decreases assign lower values of colour to it:Colour_pt=colour-colour*k

iii. For Yellow/Ochre do the same except decrease k more slowly.

iv. For Green, assign a constant value c1 to give it a certain elevation and add Gaussian Noise for a Grassy effect

v. For Blue do nothing

vi. For Unsorted Colours, assign a constant value c2 and give it a desired texture/noise.

H. Error Analysis and Correction Algorithm

For use in surveying, commercial and simulation applications, an absolute accuracy analysis of the system is mandatory. Preliminary results from initial approaches, yielded a detailed understanding on which techniques worked as expected and which needed to be replaced.

It was noticed that while performing the contouring, if the contour is concave in shape and has a reflex angle in it, subsequent iterations of our contouring algorithm will result in the contours being drawn to extend beyond the periphery of the reflex angle. This is because the coordinates of the centroids of concave figures can be outside its boundary.

The pixels which will be shifted beyond the contour boundary will distort the final image and need to be removed.

This is where we used the mask to fix this. Bitwise AND operations were performed on the pixels outside the mask, which were black and those incorrectly drawn outside, which were white. The result was that these unwanted pixels were turned black and thereby, 'erased' from the image. The masking operation resulted in jagged edges which had to be removed using up scaling, Fourier Transform and downscaling. The scimage library was used for this.

Contouring resulted in striped noise due to distinct contours being visible. To reduce them, a combined wavelet andFourier Transform approach was used and was implemented through the pystripe library. To give a smooth slope and hence, reduce sharp changes in gradient, the Gaussian Filtering Algorithm and Median Filtering Algorithm was used to blur the image.

I. Algorithm 5: ERROR ANALYSIS AND CORRECTION ALGORITHM

i. Bitwise AND output of Masking Algorithm to the corresponding output of the Contouring Algorithm

ii. Apply Gaussian smoothing operations, Low Pass Filtering Operations and Fourier Transform to remove noise.

J. Combined 2D Heat Map Generation

A 2D height map was created by performing a bitwise OR operation on all the individual masks which produced a superposition of the contour regions of all the different terrains. This was further used to generate the 3D model.

K. Algorithm 6: COMBINED 2D HEIGHT MAPGENERATION ALGORITHM

Bitwise AND the output of the Error Analysis and Correction Algorithm

L. 3D model generation algorithm

The three.js library was used, which facilitates working with 3 D structures in Javascript. First the image was loaded onto an HTML canvas. The canvas function was then used to access the image's width and height. Geometry defines the geometry of the object to be drawn. A plane geometry was selected for the 3D model

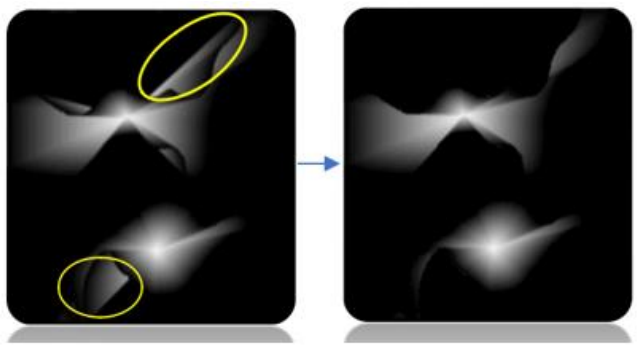


Fig 2: Contours drawn beyond original region, corrected and removed using bitwise ANDing operations and Gaussian Filtering.

pystripe library. To give a smooth slope and hence, reduce sharp changes in gradient

A texture for the same was selected by supplying a suitable material to it. Material defines the attributes of the material which the object is made of. Phong material was used to give some lustre to the model. Then each point in this geometry was assigned a height, directly proportional to its whiteness in the Height Map, using a Height Amplifier. This geometry and material were then converted to a mesh which was added to the scene. [9] A perspective camera was also used for visualization of the 3d model. Perlin and Diamond-Square noise were added to give a more natural texture to the terrain surface. [10]

M. Algorithm 7: 3D MODEL GENERATION ALGORITHM

i. Load output of Combined 2D Height Map Generation Algorithm into the canvas

ii. Define the geometry, texture and material attributes for the model

iii. Raise each point to a specific height using:plane_ geometry_height = heightmap * height_amplifier

iv. Add Perlin and Diamond-Square Noise.

III. OPTIMIZATION PERFORMED

The Big-O analysis was performed to determine the time complexity of computation for iterations and convolutions. Upon timing each module separately, operations consuming significant memory and computational resources were isolated and then replaced with less computationally intensive ones. Bilateral Filtering Operation was replaced with the Gaussian Filtering Algorithm due to the extensive resources it took. The number of comparisons needed for colour classification was also reduced using the ‘K-Means Algorithm’ and the ‘Classification Algorithm’ that was designed specifically keeping this in mind.

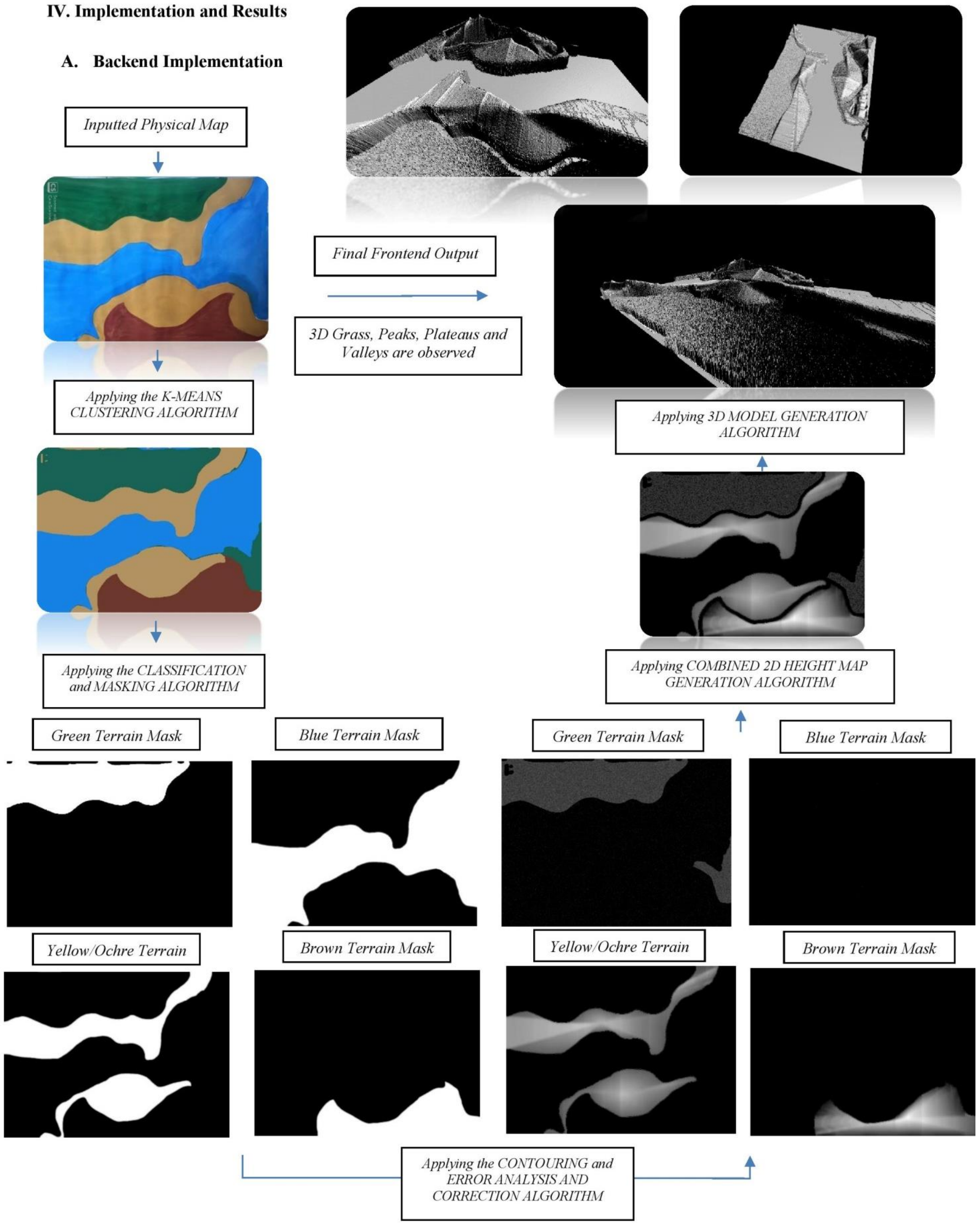
External dependencies were minimized by replacing functions called from external libraries with blocks of code calling inbuilt functions so that memory overheads caused by external libraries are freed up. This also enabled the algorithms to be more platform-independent. Uses of smoothing functions were closely monitored and the use of those having lesser impact in improving the 3D model was eliminated.

While testing the algorithms on a large dataset of maps, inaccurate renderings were analyzed and new algorithms were added to ensure that a significant number of maps were able to have their terrains separately identified and modelled.

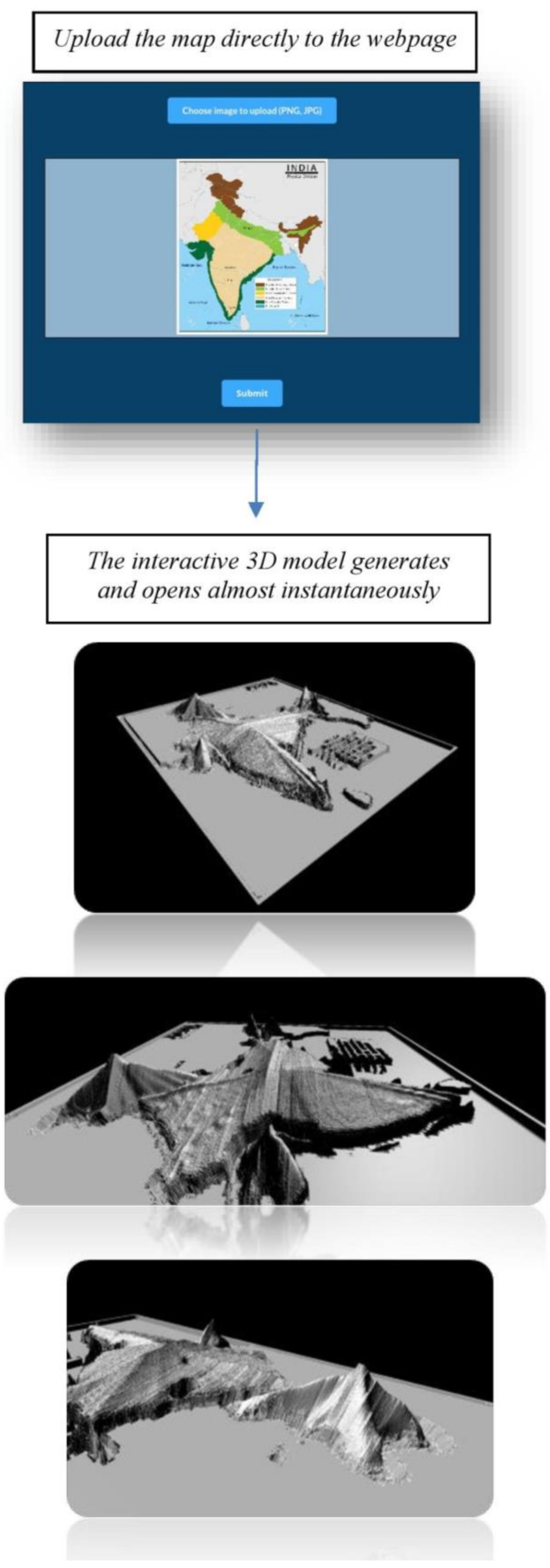
IV. IMPLEMENTATION AND RESULTS

IV. Implementation and Results

A. Backend Implementation


Inputted Physical Map
Applying the K-MEANS CLUSTERING ALGORITHM
Applying the CLASSIFICATION and MASKING ALGORITHM
Green Terrain Mask
Blue Terrain Mask
Yellow/Ochre Terrain
Brown Terrain Mask
Applying the CONTOURING and ERROR ANALYSIS AND CORRECTION ALGORITHM
Green Terrain Mask
Blue Terrain Mask
Yellow/Ochre Terrain
Brown Terrain Mask
Applying COMBINED 2D HEIGHT MAP GENERATION ALGORITHM
Applying 3D MODEL GENERATION ALGORITHM
Final Frontend Output
3D Grass, Peaks, Plateaus and Valleys are observed

B. Frontend Implementation


Upload the map directly to the webpage
Choose image to upload (PNG, JPG)
INDIA
Submit
The interactive 3D model generates and opens almost instantaneously

V. IMPROVEMENTS BEING WORKED UPON

Computer vision is a rapidly developing field and new techniques keep being invented. This makes contributing to it very exciting.

At times, images having colours higher than 4 or images having terrains split into small contours produce radial lines and distortion. This can be rectified by performing radial and circular blurring operations. [11]

An app is being worked on to enable rapid scanning and 3D modelling of bulk images. However, the libraries that the algorithm was augmented with were not compatible with it. Consequently, replacement libraries are being sought. Images need to be cropped before running the algorithm. This can be handled by the algorithm itself, by training it using Warp Affine Transform and Perspective Transforms. The 3D model can be given individual shades to represent each terrain. TensorFlow and Keras are to be used to train the algorithm using appropriate datasets to give greater accuracy in 3D Map Generation.

VI. CONCLUSION

A reliable approach to the design and performance of a system generating an interactive 3D model was presented. The paper explained the approach and logical reasoning employed in the algorithms used to extract the terrains from the 2D Physical Map.

The methodology on how a change of colour space from RGB to HSV facilitated the extraction of pixels belonging to one terrain was justified. The error analysis and the steps employed to resolve them were also stated.

The operations behind the generation of the 3D Digital Elevation Model from the Height Map were also explained.

Results of the tests conducted on the final system demonstrated clear improvements compared to previous versions. However, some technical limitations of the current design such as those mentioned above may need resolution and are being worked upon.

REFERENCES

[1] Y. Yagi and M. Yachida, ‘Real-time generation of environmental map and obstacle avoidance using omnidirectional image sensor with conic mirror’, presented at the 1991 IEEE Computer Society Conference of Computer Vision and Pattern Recognition, doi: 10.1109/cvpr.1991.139681.

[2] A. Jain, M. Mahajan, and R. Saraf, ‘Standardization of the Shape of Ground Control Point (GCP) and the Methodology for Its Detection in Images for UAV-Based Mapping Applications’, in Advances in Intelligent Systems and Computing, Springer International Publishing, 2019, pp. 459–476.

[3] J. Spencer, O. Mendez, R. Bowden, and S. Hadfield,‘Localisation via Deep Imagination: Learn the Features Not the Map’, in Lecture Notes in Computer Science, Springer International Publishing, 2019, pp. 710–726

[4] Y. R. Muratov and D. I. Ustukov, ‘Scaling of Depth Map for Stereo Vision Tasks’, presented at the IEEE 2019 8th MediterraneanConference onEmbeddedComputing(MECO), Jun. 2019, doi: 10.1109/meco.2019.8760202

[5] W. Gu, J. Yin, X. F. Yang, and P. Liu, ‘Disparity Map Acquisition Based on Matlab CalibrationToolbox and OpenCV Stereo Matching Algorithm’, AMR, vol. 926–930, pp. 3030–3033, May 2014, doi: 10.4028

[6] G. Verhoeven, M. Doneus, Ch. Briese, and F. Vermeulen, 'Mapping by matching: a computer vision-based approach to fast and accurate georeferencing of archaeological aerial photographs', Journal of Archaeological Science, vol. 39, no. 7, pp. 2060–2070, Jul. 2012, doi: 10.1016/j.jas.2012.02.022

[7] T. N. Tu, D. D. Van, H. N. Hoang, and T. V. Van, 'AMethod to Enhance the Remote Sensing Images Based on the Local Approach Using KMeans Algorithm', in Advances in Informationand Communication Technology,Springer International Publishing, 2016, pp. 41–52

[8] Image Credits: S. LAZEBNIK, C. S., AND PONCE., J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York, June 2006, vol. II, pp. 2169-2178.

[9] Gallup, David & Frahm, Jan-Michael & Pollefeys, Marc & Zuerich, Eth. (2011). A Heightmap Model for Efficient 3D Reconstruction from Street-Level Video.doi = 10.1.1.187.6455

[10] Gallup, David & Pollefeys, Marc & Frahm, Jan-Michael. (1970). 3D Reconstruction Using an n-Layer Heightmap. 6376. 1-10. 10.1007/978-3-642-15986-2_1.

[11] D. Li, D. Hu, Y. Sun, and Y. Hu, '3D scene reconstruction using a texture probabilistic grammar', Multimed Tools Appl, vol. 77, no. 21, pp. 28417–28440, May 2018, doi: 10.1007/s11042-018-6052-z.

.