2D Cartoon Sketches to 3D Models – Project Evaluation Final, Virtual Reality Project

AKARSHA SEHWAG and SANIDHYA SINGAL, Indraprastha Institute of Information Technology Delhi

1 PROBLEM STATEMENT

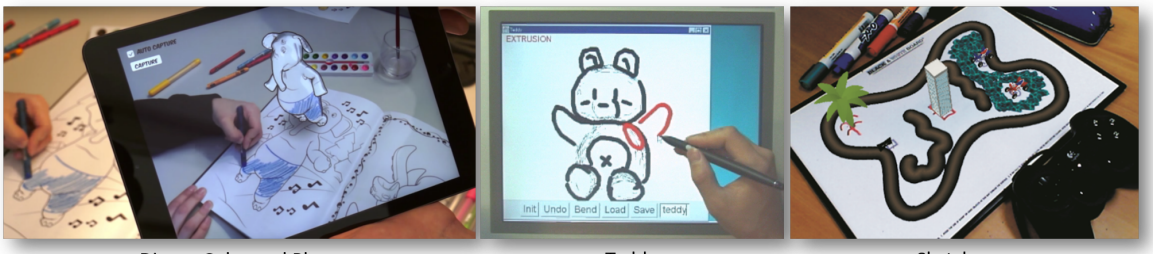
"Design a system to create 3D scene using objects synthesized from paper drawings and sketches"
The users, in general children (as children love drawing), draw a 2D sketch/cartoon on paper. Our system uses that sketch as a marker and generates a 3D model for the same 2D sketch in a Virtual Reality environment. We assume that the front and back texture of the 3D model is the same. Neither the marker nor the 3D model are available in advance. There are no ready-made templates or models.

We present a system that automatically converts personalized, lifeless 2D drawings into 3D textured models, thus bringing life to flat cartoon drawings.

2 RELATED WORKS

2.1 AR Books

AR books were created for the purpose of enhancing entertainment and education for children. These had predefined 2D sketches and also their corresponding 3D models. So, the children were just made to colour the 2D sketch drawn on a book page and could see the model appearing out of the same book page through a handheld AR display. However, this did not allow the children to show their creative drawing skills. They could not create their own virtual character models as well. Some popular examples: Crayola Color Alive, Disney Color and Play and QuiverVision [1].



Disney Color and Play | Teddy | Sketchaser

2.2 Sketch-based Modeling System

Sketch-based 3D modelling was a popular research field. It demanded users to create multiple sketches from many different views in order to obtain the desired 3D model. This is obviously a cumbersome and frustrating task for young children. In order to address this issue of multi-view sketch-based modelling, single-view sketch-based modelling was introduced. However, this required much more skills and was time consuming. Teddy is a great example of sketch-based 3D modelling [1].

2.3 Authoring Models in AR

Like AR books, these also had predefined sketches and the corresponding 3D models. However, their approach was slightly different. For example, Sketchaser was a sketch-based AR racing game. In this game, certain symbols were defined by the manufacturers. If the user drew one such symbol, the corresponding 3D model would show up in the AR display in its place. ARpm is another such example [1].

3 CHALLENGES

We faced a couple of challenges:

(1) Neither the marker nor the 3D model was available in advance.
   - There were no ready-made templates or models. So, we had to create everything from scratch.

(2) From 2D to 3D
   - This was a bigger challenge as we didn't have any knowledge about the depth of the 3D model. So, we needed to be careful with the output.
   - Also, how to texture the rear side of the model? We had the texture only for the front side of the model.

4 VR SYSTEM

(1) The user draws and colours a 2D sketch/cartoon on paper.
(2) We capture the drawing using handheld device (having a camera). We send this image to our MATLAB script.
(3) The script extracts the region of interest (the 2D sketch) from the image and the creates a texture with the help of outline and region maps.
(4) With the help of distance map, the system inflates the 2D image and converts it into a 3D model.
(5) We map the texture on top of the 3D model so obtained, both on the front and the back.
(6) This textured model is sent to Unity in order to view it in VR environment.

5 IMPLEMENTATION DETAILS AND ALGORITHMS

Basically, we split the cartoon into several regions based on the dark edges and generate an inflated mesh for each region. Then, we stylize the cartoon model with the original 2D cartoon texture. The details are as follows:

5.1 Image Boundary Extraction and Outline Maps

The user is asked to draw the image in such a way that it has very thick black coloured outline. This helps us in detecting boundaries. We need to ensure that the boundaries are discontinuous as well. Fig. 1 shows an example of the same.

The initial image is converted into an image with a fixed height of 600 pixels, preserving the aspect ratio. Then, it is converted into a binary image BW, where the boundary pixels have a value of 0 and others have a value of 1. We make use of MATLAB's in-built function – `imbinarize` for the same.

5.2 Region Maps

Region maps are meant to split the image into enclosed regions based on the dark edges. These maps are generated using flood filling algorithms. A flood filling algorithm visits every node in the neighbouring area to determine the area in the vicinity of the given cell in an N-Dimensional array. For dividing it into regions, we keep a minimum threshold, and ignore the outliers exceeding a threshold value. Every pixel now has a color code to determine the region it belongs to. Fig. 2 shows an example of region maps.

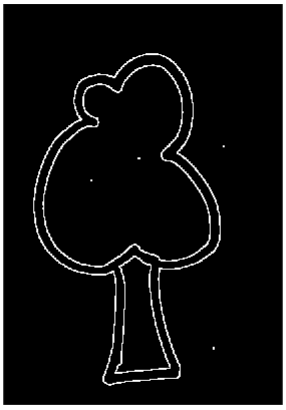


Fig. 1. Outline Map

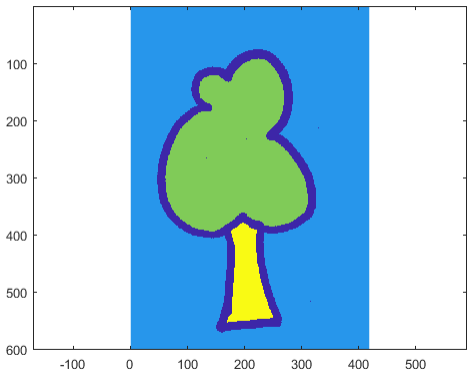


Fig. 2. Region Maps

5.3 Distance Maps

Distance maps denote the distance of each pixel to its nearest black pixel. One such is shown in fig. 3. These distance maps are used in inflating the 2D image to 3D model.

5.4 2D to 3D – Inflation

In order to convert the 2D image to 3D model, we make use of the distance maps. The distance value indicates how far the pixel should be inflated from the plane. We inflate both above and below the plane. However, the shape is too sharp if we use distance values to inflate the region directly due to the linearity of the distance map. In order to get a smooth result, we use a *circular mapping function* [1], to transform the distance values to the real height values. We obtain meshes as shown in the fig. 4 and 5.

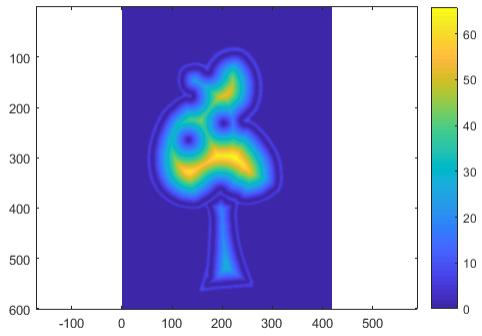


Fig. 3. Distance Map

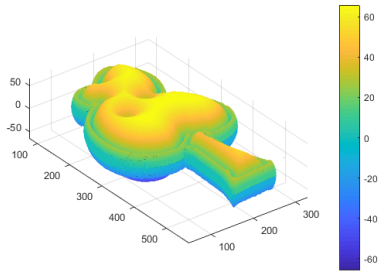


Fig. 4. Smooth Mesh

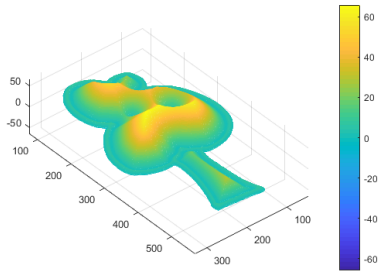


Fig. 5. Linear Distance Map

$$a = D_{max} - D(x)$$
$$H(x) = \sqrt{D_{max}^2 - a^2}$$

5.5 Texture mapping in Unity

The model created in MATLAB is saved in *.stl* format and converted to *.obj* format in order to import it in unity. Along with that, we import a normal map (Fig. 6), to estimate the surface normal for each polygon generated in MATLAB for accurate texture mapping.

Once the model is imported, a material is added to it, created using the normal map, occlusion map (determine the areas exposed or hidden from the ambient lighting) and a color map.

The model is then viewed in a virtual environment for an immersive experience.

6 RESULTS

We tried our system on a set of photographs. Some of them are shown in fig. 7 to 12.

7 CONCLUSION

We, through this project, were successfully able to generate a 3D model of any cartoon.

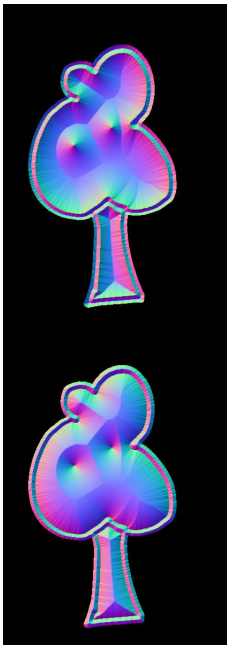


Fig. 6. Normal Map

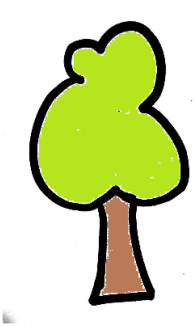


Fig. 7. Original 2D Image (1)

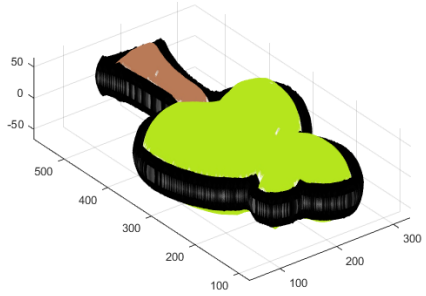


Fig. 8. Final 3D model (1)

8 POSSIBLE EXPANSIONS

(1) **Animation**
We plan to add basic animations such as rotation, etc. But the cartoon models can also be rigged, which requires skeleton embedding and skinning. Skeleton Embedding will determine the position of each bone and skinning shall find the bone weight for each vertex. Now, once we have the two inputs, using Linear

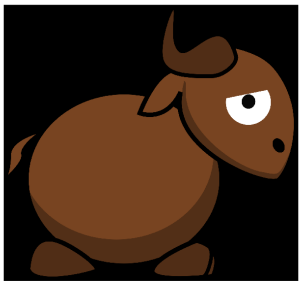


Fig. 9. Original 2D Image (2)

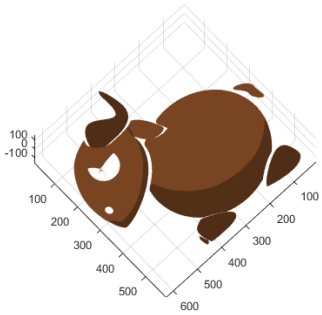


Fig. 10. Final 3D model (2)

Fig. 11. Original 2D Image (3)

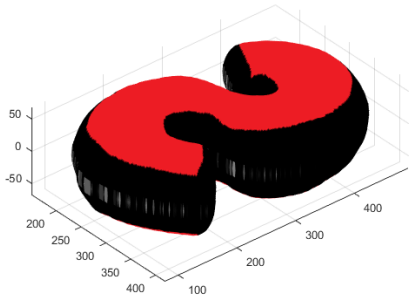


Fig. 12. Final 3D model (3)

Blend Skinning (LBS) [1], we can animate the cartoon with a set of pre-defined motions like jumping, running, et cetera.

(2) **Interaction**
A User Interface (UI) would improve the personalisation of the 3D model to a huge extent, wherein the user can copy, scale, create skeleton (of the object) and/or transform the model.

REFERENCES

[1] L. Feng, X. Yang and S. Xiao. "MagicToon: A 2D-to-3D creative cartoon modeling system with mobile AR." 2017 IEEE Virtual Reality (VR), Los Angeles, CA, 2017, pp. 195-204, doi: 10.1109/VR.2017.7892247

[2] L. Feng, X. Yang, S. Xiao and F. Jiang. "An Interactive 2D-to-3D Cartoon Modeling System." In: El Rhalibi A., Tian F., Pan Z., Liu B. (eds) E-Learning and Games. Edutainment 2016. Lecture Notes in Computer Science, Springer, Cham, vol 9654, 2016.

[3] http://candycat1992.github.io/2017/01/18/magictoon/