MagicToon: A 2D-to-3D Creative Cartoon Modeling System with Mobile AR

Lele Feng* Xubo Yang† Shuangjiu Xiao‡

Digital ART Lab, School of Software
Shanghai Jiao Tong University, China

Abstract

We present **MagicToon**, an interactive modeling system with mobile augmented reality (AR) that allows children to build 3D cartoon scenes creatively from their own 2D cartoon drawings on paper. Our system consists of two major components: an automatic 2D-to-3D cartoon model creator and an interactive model editor to construct more complicated AR scenes. The model creator can generate textured 3D cartoon models according to 2D drawings automatically and overlay them on the real world, bringing life to flat cartoon drawings. With our interactive model editor, the user can perform several optional operations on 3D models such as copying and animating in AR context through a touchscreen of a handheld device. The user can also author more complicated AR scenes by placing multiple registered drawings simultaneously. The results of our user study have shown that our system is easier to use compared with traditional sketch-based modeling systems and can give more play to children's innovations compared with AR coloring books.

Keywords: 2D-to-3D, modeling, augmented reality, mobile devices, user interface, coloring.

Index Terms: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques

Figure 1: An AR scene created by our system.

*e-mail: lelefeng1992@gmail.com
†e-mail: yangxubo@sjtu.edu.cn (corresponding author)
‡e-mail: xsjiu99@cs.sjtu.edu.cn

2017 IEEE Virtual Reality (VR)
March 18-22, 2017, Los Angeles, CA, USA
978-1-5090-6647-6/17/$31.00 © 2017 IEEE

1 Introduction

Cartoon painting with pens and paper is a natural activity and an important experience for children to practice and express their creative skills. In recent years, augmented reality (AR) even enhances the experience by providing a bridge between real-world colored drawings and digital cartoon characters. There are several AR coloring book systems either from academic or industry, such as colAR [7], Crayola Color Alive [8], Disney's Color and Play [11] and Chromville [6]. These systems can recognize the colors users paint on paper and use similar colors on 3D models in AR context. However, the 3D elements used in these systems are all manually modeled in advance, and the 2D drawings are also ready-made specific line drawings. This kind of systems limits the creativity of children because children cannot create their own personalized cartoon drawings and 3D models within the system.

For novice users such as children, it is not easy to build 3D models using professional modeling systems. Traditional 3D modeling tools require users to learn a complicated interface, which are daunting challenges for children. In order to ease such a situation, sketch-based modeling systems such as Teddy [19] and its follow-up works [25, 21, 14, 5] proposed approaches to create 3D models from 2D strokes. These works indeed provide more opportunities for children to create personalized 3D content but have some common limitations. First, most of these systems require users to sketch from a large number of different views, making it difficult for children to accomplish their tasks. Second, all of these systems need children to sketch in front of digital monitors or at most on tablets, separating them from real-world activities and losing physical enjoyment they have experienced from traditional activities like drawing and coloring. In addition, it is not easy for children to get satisfactory sketches just through mouse dragging without the help of common tools like erasers and rulers.

In this paper, we present **MagicToon**, an AR system on mobile devices that allows children to create three-dimensional textured models directly from their own two-dimensional cartoon drawings and further easily author complicated AR scenes. There are two major design goals for MagicToon:

- Design a system on mobile devices that can generate 3D textured cartoon models by fully leveraging children's drawing skills in the real world and require minimum inputs.
- Provide an approach for children to author their own creative cartoon scenes in AR environments easily through some simple multi-finger gestures.

Oviatt [23] summarized that new interfaces for education should be designed to minimize users' cognitive load by modeling their pre-existing behavior, so we aim to maintain children's entertainment and creativity of drawing and coloring on real paper as much as possible, keeping children more attuned to the real world around them. In addition, Hürst et al. [18] concluded that touchscreen operations outperformed freehand movements in creating and editing 3D models in AR applications on mobile devices. Hence we avoid

complex operations in virtual 3D spaces and employ simple touchscreen gestures.

The main contributions of our work are:

- Present an automatic 2D-to-3D model creator on mobile devices that can directly convert personalized 2D drawings into 3D textured models with AR enabled, bringing life to flat cartoon drawings.
- Propose a creative modeling pipeline including an automatic model creator and an interactive model editor, enabling children to construct personalized AR scenes (Figure 1) from scratch.
- Conduct an user study showing the comparison results among three cartoon systems. The results have shown that our system is easier to use compared with traditional sketch-based systems and can give more play to children's innovations than AR coloring books.

2 Related Work

2.1 AR Books

AR has shown great potential to enhance entertainment and education for children. Zünd et al. [29] proposed the concept of Augmented Creativity as employing AR on modern mobile devices to enhance real-world education, opening new interaction possibilities. MagicBook [4] is one of the earliest systems that combined virtual digital content with a book in real world. In MagicBook, children can see three-dimensional virtual models appearing out of the book pages through a handheld augmented reality display. Clark and Dunser [7] presented a new type of interactive AR book experience. In their prototype, users can color the pages of a book and the system automatically maps the colored results to virtual pop-up scenes and 3D models. There is also a commercial product derived from their work called QuiverVision[1]. Since then, many other coloring book products, such as Crayola Color Alive [8], Chromville [6] and Disney's Color and Play [11], have shown immense potential in this field. Magnenat et al. [20] proposed a method that can detect and track the drawing process alive, enhancing the overall drawing experience further. There is one main limitation among these works that the 3D content cannot be created by users entirely. Although users can color 3D content, they always need to prepare printed line arts provided by the manufacturers. The systems do not enable them to create their own virtual character models.

2.2 Sketch-based Modeling System

Sketch-based 3D modeling has been a popular research field, which simplifies the traditional modeling pipeline making it much more easier for children to create 3D models. Systems such as Teddy [19] and its descendants FiberMesh [21] and ShapeShop [25] approached the problem by asking users to sketch from many different views. RigMesh [5] proposed an approach to modeling and rigging in contrast to the traditional sequence. The range of models that these systems can handle is limited, because their modeling methods are limited to spherical topology. Another problem of these systems is that they require users to sketch from a large number of different views. Schmidt et al. [24] found that novice users such as young children were unable to accomplish their tasks and became frustrated very easily with the change of views.

To address the problems in multi-view sketch-based modeling systems, several single-view sketch-based modeling systems are proposed. In Gingold et al. [14], structured annotations were introduced for 2D-to-3D modeling. In their implementation, the user can add deformable primitives along with some structured and semantic annotations. However, adding annotations from a single view demands decent 3D perspective understandings, which are difficult for children. A similar work using annotations is Natura sketch [22]. The system presented a new modeling method based on distance transforming which is free from the limitation of spherical topology in prior works. A common problem of these methods is that they require tedious inputs to produce personalized models, which is time-consuming for children.

2.3 Authoring Models in AR

Several applications have been developed to support authoring 3D models in AR. ARpm [12] was a prototype system which allowed the use of complex 3D polygonal modeling tools provided by 3D Studio Max to bring models into the user's world. Air-Modeling [1] provided a CAD interface to create virtual conceptual products by hand gestures in AR environments. The modeling methods in those systems were the same as the traditional methods per se, which were difficult for children to learn. Bergig et al. [3] presented a framework for authoring 3D scenes for AR based on hand sketching and implemented an application that constructed AR scenes of mechanical systems from 2D sketches. Their system was limited to simple solid models with a limited set of properties, which was more suitable for specific systems such as mechanical systems. Hagbi et al. [15] described a sketch-based AR racing game called Sketchaser with In-Place Augmented Reality Sketching experiences. Players can draw on real paper and the system can author virtual game elements according to the sketched elements. A similar work can be found in [9], where an interactive storytelling system was presented. The user can draw a set of predefined sketches on a sheet of paper. If the system recognizes a sketch, it will render the virtual world objects and characters superimposed over the real world. A common limitation of these systems is that all of the authored models require to be created in advance, which leaves no space for users to author their personalized content. Hürst et al. [18] evaluated the feasibility of free-hand interaction using mobile devices in order to create and edit 3D models in AR applications. Based on their experiments, they concluded that freehand drawing was too difficult for most users and it seemed obvious that touchscreen operations outperformed freehand movements. In contrast to existing works, our system provides a novel modeling workflow that simplifies the modeling operations in AR by automatically generating 3D models from 2D drawings and employs simple touchscreen gestures to edit models and compose scenes.

3 Workflow

To illustrate our personalized modeling pipeline in AR environments, an example workflow using MagicToon is described as follows. The user first sketches and colors a cartoon drawing on a blank paper card, and then uses MagicToon to capture an image of the drawing (Figure 2a). The automatic model creator of our system first splits the cartoon image into several regions based on dark edges (Section 4.1). For each region, a 3D inflated mesh is generated and the original cartoon texture is attached to it with stylized cartoon shading (Section 4.2). The system then registers the textured model to the cartoon drawing via a mobile AR interface, so that the user can see the virtual model overlaying on the real world (Figure 2b). The user can perform several intuitive operations on the 3D personalized model through our interactive model editor in AR environments (Figure 2c, Section 4.3). The system supports affine transformations like rotation, translation and scaling by detecting multi-finger gestures. The user can also copy the model by double tapping on the virtual floor. Animation can be added to a model just after the user clicks a certain button. The system will produce a simple animation by interpolating positions, rotations and scales between two end points by default. The user

[1] http://quivervision.com/

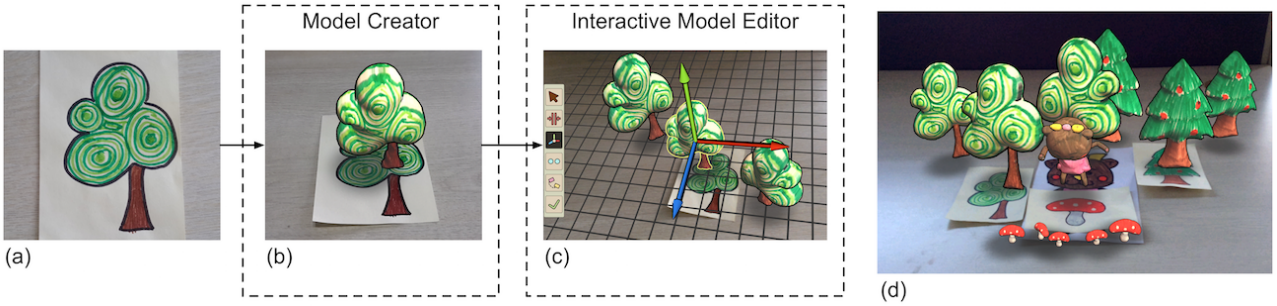

Figure 2: (a) The input cartoon drawing. (b) The model creator generates a 3D cartoon model based on the input. (c) The user performs affine transformations and copy operations on the model through our interactive model editor in AR environments. (d) The user constructs a more complicated AR scene by placing several registered cartoon drawings simultaneously.

is also allowed to rig the cartoon model with character animations with a few inputs (Section 4.4). The user can easily compose a more complicated AR scene by placing multiple registered drawings simultaneously (Figure 2d).

4 Implementation

In this section, we will describe more details about the implementation.

4.1 Segmentation

Given a picture of a cartoon drawing as input, our system first scales the picture with a fixed height of 600 pixels. The system then extracts several regions from the picture automatically. This is done by extracting outline pixels from the drawing and then using a filling algorithm to find region maps.

In cartoon drawings, outline pixels are those who are darker than their neighboring pixels. As the saturation channel contains the most sensitive visual content, we first convert the RGB image to HSV color space (Figure 3a), and apply an adaptive threshold algorithm only on the S channel (Figure 3b). We then remove outliers whose saturation values exceed a threshold value. Morphology close and open steps are followed to eliminate extra noises. However, there are still some dark pixels along the cross-sections, which may be very noticeable in the final models. To address this problem, we dilate the outline map by using a small kernel (the size is 3 by default), which makes the outline map a little thicker.

Once we have the outline map, a flood filling algorithm is applied to find enclosed region maps. The system ignores small regions whose areas are smaller than a threshold (50 in our implementation), such as some regions of the *Elephant*'s toes. Those small region maps are then added to the final outline map (Figure 3c). For those regions whose areas are larger than the threshold, we then apply a morphological close operation. The size of the kernel is obtained by evaluating the average outline width of this cartoon image. In our implementation, the average outline width is equal to the number of iterations of a thinning algorithm [28] applied to the outline map in most cases. However, we will interrupt the algorithm if the number is beyond some threshold (which is 20 in our implementation) when some parts of the outline are too thick and return this threshold value in consideration of performance. This step is essential to eliminate internal outlines within the region maps. Figure 4 shows such an example. The final region maps of the *Elephant* image is shown in Figure 3d. The outline removal steps introduce small gaps between adjacent region maps, so we enlarge each generated mesh a little by a factor of 1.08 to decrease the gaps between meshes.

The segmented region maps will be also used as textures for front and back faces in the further rendering. The resulted cartoon model

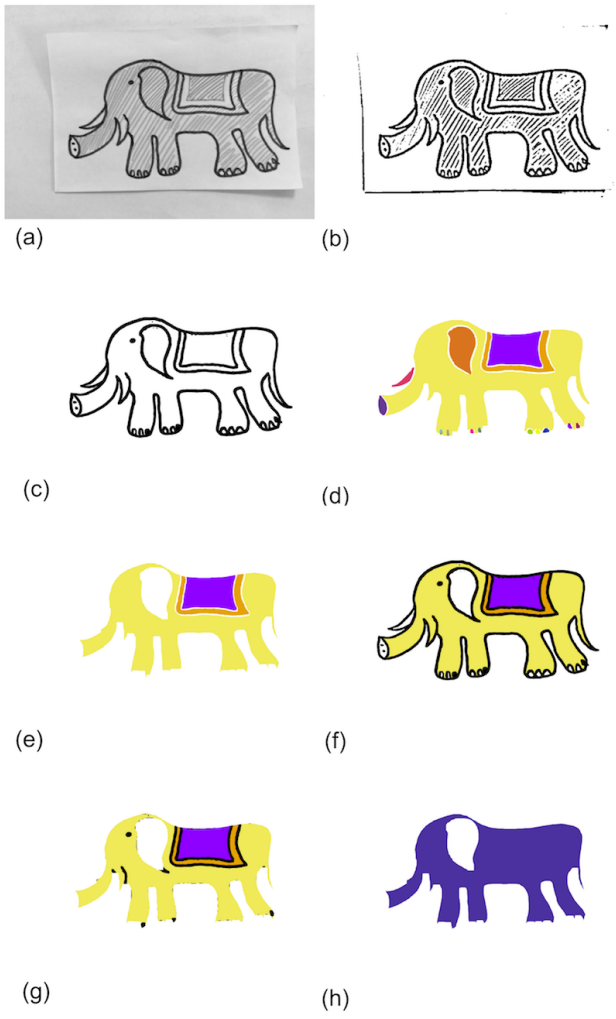

Figure 3: (a) The saturation channel of the input cartoon drawing. (b) The result after applying the adaptive threshold algorithm. (c) The final outline map. (d) The region maps. Different colors denote different regions. (e) The region maps of the regions being merged. (f) Combine (e) with the outline map to obtain a temporary region map. (g) The result after applying the morphological erosion operation. (h) The new region map.

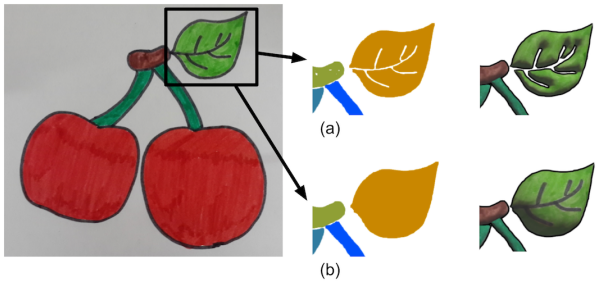

Figure 4: Apply a morphological close operation to eliminate internal outlines. The left shows there are some internal outlines within the leaf. (a) The region maps with no morphological close operation and the resulted model. (b) The region maps with a morphological close operation and the resulted model.

of the *Elephant* image is shown in the first image of Figure 6a. When the user performs a merging operation (Section 4.3) to merge over-segmented regions as shown in Figure 6a, we first combine their region maps (Figure 3e) with the outline map into a new region map (Figure 3f), and then apply a morphological erosion operation to remove outline pixels (Figure 3g). The size of the kernel is equal to the average width value we mentioned above. Finally a new mesh will be generated from this new region map (Figure 3h).

4.2 Mesh Generation

The system then finds boundaries for each region map. We only generate meshes for the extreme outer boundaries to avoid too many holes and tiny parts, so the ear of the *Elephant* in Figure 3 will not result in an internal hole of the body.

We first apply conforming constrained Delaunay triangulation [26] to each boundary, obtaining a region with many discrete vertices. In order to improve the performance on mobile devices, we use different maximum area constraints based on the region area when triangulating the region. The smaller a region is, the more triangles will be generated. We then use the same inflation method as described in NaturaSketch [22]. For each region, we first calculate a distance map to encode the distance of every pixel to the nearest black pixel, as shown in Figure 5a. The distance value indicates how far the pixel should be inflated from the plane. The shape will be too sharp if we use distance values to inflate the region directly due to the linearity of the distance map (Figure 5b). In order to get a smooth result, we use a circular mapping function as below to transform distance values to the real height values (Figure 5c):

$$H(x) = \sqrt{D_{max} * D_{max} - a * a} \quad (1)$$

$$a = D_{max} - D(x) \quad (2)$$

where $D(x)$ is the distance value of the current vertex and D_{max} is the maximum distance value within this region. This method may produce sharpness along the cross sections, so we employ the local Laplacian smoothing method [27] to smooth the whole mesh. Once the mesh is generated, the original cartoon drawing will be used as both front and back textures for each region. Our system then uses a tone-based shading to deliver a stylized look (Figure 5d).

4.3 Interaction

The system then registers the generated 3D models to AR environments. In our implementation, we use Vuforia SDK [10] to detect and track image targets. The goal of our system is not only to generate 3D models but also to provide an interface for children to create their own AR scenes. As a result, our system supports several operations in AR environments to edit more complicated scenes. After the user touches one model on the screen, our interactive model editor will be popped up on the left of the screen and the system enters the editing mode. A virtual grid plane will be placed at the same location of the virtual image target. The user can click one of the six buttons as shown in Figure 6. By default, the first button is selected, which means the system is in the idle mode. If the user clicks the last button, the system will exit the editing mode and render the scene normally. There are four main operations supported by our system:

- **Merge**. Areas enclosed by dark pixels will be treated as regions. Some decorative elements may be mistaken as individual small regions, which make the outer region divided into several over-segmented regions. In Figure 6a, we show such an example to merge over-segmented regions of the *Elephant* model. The user can perform a dragging gesture to move across those regions and the system will merge them automatically.

- **Affine transformation**. As shown in Figure 6b, the user can drag the virtual coordinate axis to translate a cartoon model. The system also detects pinch gestures to scale the model. Rotations will be applied to the model once the system detects a dragging gesture whose start point is touched on the model.

- **Copy**. The user may want to make several copies of a certain model. As shown in Figure 6c, a forest can be created by making a few copies of a tree model. The user can double tap somewhere on the virtual grid plane and the system will instantiate a copy in that location. The other three operations can also be performed on each individual model copy to make them different from each other.

- **Animation**. Our system supports simple animations by interpolating positions, rotations and scales between two end points by default, as well as advanced character animations. Once the user clicks the animation button, a dummy virtual model (the end point) will be placed in the scene with a certain offset to the selected model (the start point). The user can perform affine transformations on this dummy model as we describe above to change the position, rotation and scale of this end point. Figure 6d shows an example of animating a floating cloud. The user can also author character animations to the selected model by placing several skeleton joints on desired positions on the model and the system will automatically rig it. We will describe more details about character animations in Section 4.4.

More complicated scenes can be composed by placing multiple registered cartoon drawings within the camera's field of view. The system supports up to five cartoon drawings at the same time, in consideration of the accuracy and performance of the tracking algorithm. The AR environments provide a novel approach for children to edit virtual scenes conveniently and creatively. The relative positions and rotations of different cartoon models can be easily changed by just moving the drawing cards back and forth in the real world.

4.4 Character Animation

Our system also supports complicated character animations. In order to rig a cartoon model, the system needs to calculate two kinds of information: skeleton embedding and skinning. Skeleton embedding determines the positions of each bone and skinning needs to find bone weights for each vertex.

For skeleton embedding, we employ a method that partly depends on the user inputs. Our system currently only supports the human skeleton and motions. The human skeleton we choose consists

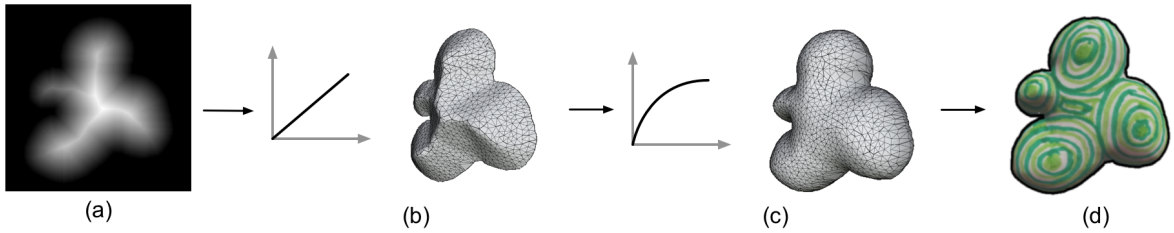

Figure 5: (a) The distance map. (b) The generated model after using linear distance values to inflate the region directly. (c) The smooth model after applying a circular mapping function. (d) The textured 3D model.

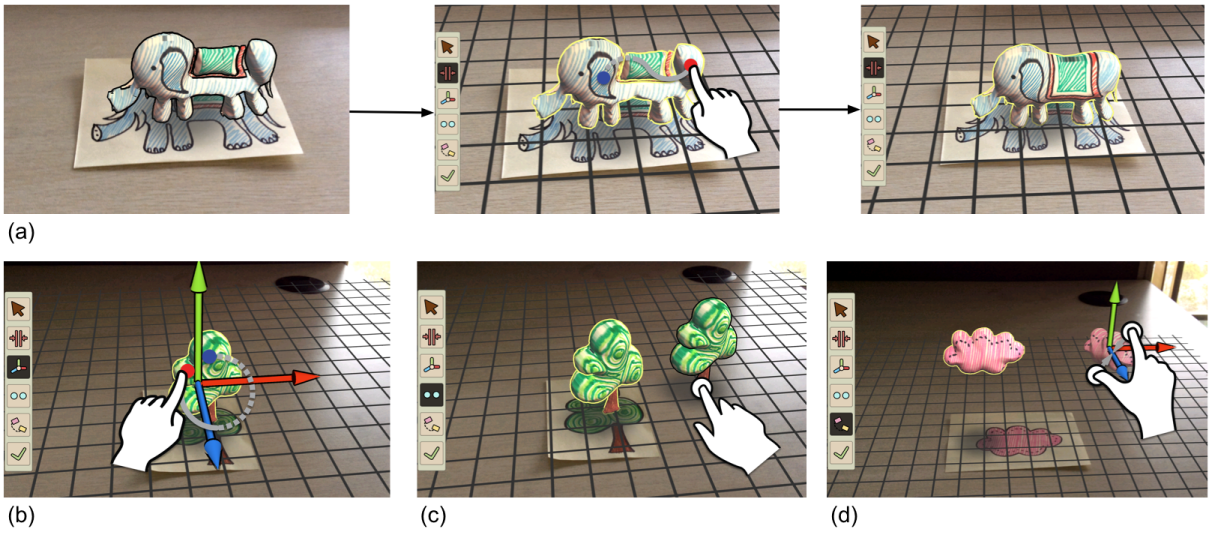

Figure 6: Main operations supported by our system. (a) The *Elephant* example has some over-segmented regions. The user drags a finger through those regions to merge them. (b) The user can translate, rotate and scale a model through certain gestures. (c) The user can double tap on the virtual plane to make a copy of the tree model. (d) The user can edit the end point of an animation through affine transformations.

of 18 joints (17 bones) and the user needs to place 9 particular joints such as shoulders and feet to the corresponding positions through the touch screen. The joints can be placed outside the model volume, such as the feet joints of the *Tree* in Figure 7a. The system then calculates the positions of the remaining joints automatically (Figure 7b and Figure 7c).

For those joints whose positions are not determined by the user, we define five parameters for each of them: $(\boldsymbol{J}_{start}, \boldsymbol{J}_{end}, r_1, r_2, \boldsymbol{D})$. The position of this joint $\boldsymbol{J}$ can be computed by using the following formula:

$$\boldsymbol{J} = \boldsymbol{J}_{start} + r_1 \cdot (\boldsymbol{J}_{end} - \boldsymbol{J}_{start}) + r_2 \cdot ||\boldsymbol{J}_{end} - \boldsymbol{J}_{start}|| \cdot \boldsymbol{D}$$

where $\boldsymbol{J}_{start}$ and $\boldsymbol{J}_{end}$ are the two dependent joints; $\boldsymbol{D}$ is a vector that describes the direction of the offset; r_1 and r_2 are the ratios of lengths. If the positions $\boldsymbol{J}_{start}$ and $\boldsymbol{J}_{end}$ are also not determined, the algorithm will be executed recursively.

To create a rigged cartoon model, the next step we need to do is to calculate the bone weights for each vertex (skinning). These weights define how much each bone transformations affect each vertex. Our system uses the skinning algorithm proposed in Pinocchio [2]. The algorithm uses the key idea of heat diffusion and the bone weights can be computed by solving a linear system of equations. In the original implementation, Pinocchio needs to construct distance field in 3D space to evaluate the exact distance to the surface from an arbitrary point, and then uses distance field to check whether a vertex can see a bone when calculating the nearest bone for this vertex. In our implementation, we decide to remove this step for two main reasons: first of all, this step is time-consuming, which takes more than one minute on mobile devices; second, this check also makes calculating weights of those bones outside the model volume impossible, because no vertex can see these bones at all. Although removing this step may introduce some occasional artifacts, the heat diffusion method gives acceptable results in all of our experiments and we consider the two benefits mentioned above much more important in such an interactive system for children. After removing this step, the skinning algorithm only takes about one second on our test devices. Figure 8 shows the skinning results of Figure 7 for the bone marked in green.

Given the skeleton and skinning information as inputs, we then use the linear blend skinning (LBS) method to animate the cartoon model with a set of predefined character motions, such as jump and running as shown in Figure 9.

5 Results

We have implemented MagicToon using the Unity3D game engine[2] and Vuforia SDK [10]. MagicToon can run on Android and iOS platforms.

[2]http://unity3d.com/

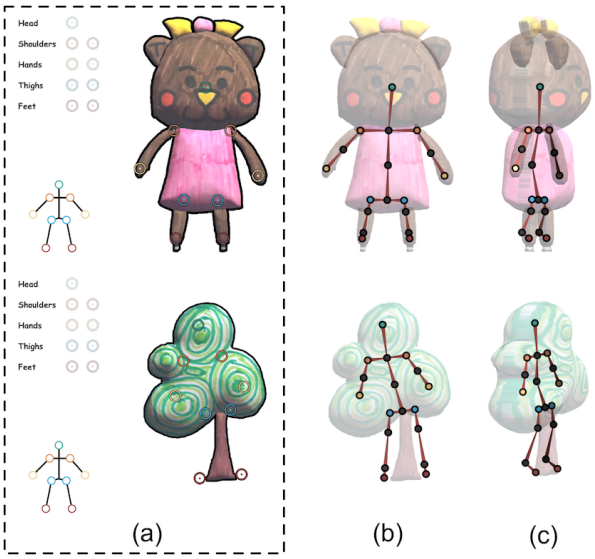

Figure 7: Skeleton embedding. (a) The user places nine particular joints on the model, such as joints for the head and shoulders. (b)(c) The system calculates all the positions of the 18 joints based on (a). Joints can be placed outside the model volume, such as the feet joints for the *Tree* model.

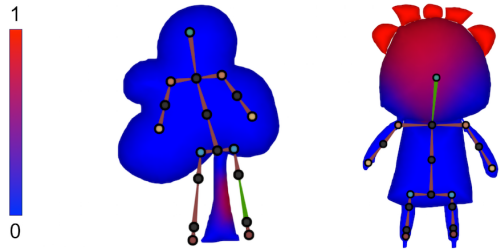

Figure 8: Bone weights for the bone marked in green.

In order to get performance statistics of our model creator, we tested MagicToon on a 1.3GHz iPad Mini 2 to model four example cartoon drawings (Figure 10). The drawings were created by different users: the *Cherry* (Figure 10a) and the *Tree* (Figure 10c) were drawn by one colleague who had no 2D artistic experience, the *Bunny* (Figure 10b) was drawn by another colleague with five years 3D modeling experience and the *Bear* (Figure 10d) was drawn by a twelve years old child.

Table 1 shows mesh statistics and Table 2 shows timing statistics of each step for each example. The total time for mesh generation was about 0.7 second and most of the time was spent on inflation. Drawings with more regions tended to require more time for inflation, since that more computation was required to calculate distance maps for these regions. In general, the performance of our model creator is sufficient for the requirements of interactions in MagicToon.

Table 1: Mesh statistics for examples

	Cherry	Bunny	Tree	Bear
Regions	7	8	11	11
Vertices	3600	3036	2700	4594
Triangles	21516	18096	16056	27432

Figure 9: Character animations of the *Bear* model. The left shows a jump animation and the right shows a running animation.

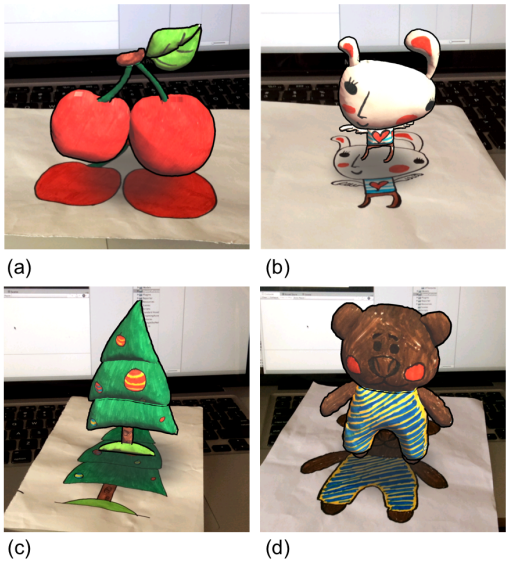

Figure 10: Example models.

6 User Study

In order to compare our novel MagicToon system with prior sketch-based modeling systems and AR coloring books, we set up a comparative user study. We chose RigMesh [5] as the representative of traditional sketch-based modeling systems and Chromville [6] as the representative of AR coloring systems.

6.1 Evaluated Systems

The three evaluated systems were as follows:

- **MagicToon**: our interactive 2D-to-3D modeling system on mobile platforms. The user can input a cartoon drawing and the system will generate a textured 3D model automatically.

Table 2: Timing statistics for examples (ms)

	Cherry	Bunny	Tree	Bear
Segmentation	89	95	71	54
Triangulation	51	38	55	61
Inflation	501	512	576	583
Total	641	645	702	698

Figure 11: Children were using our system. (a) A boy drew a sunflower and then used our system to construct a virtual garden. (b) A girl drew a yellow cat and a red cherry and then used our system to model them.

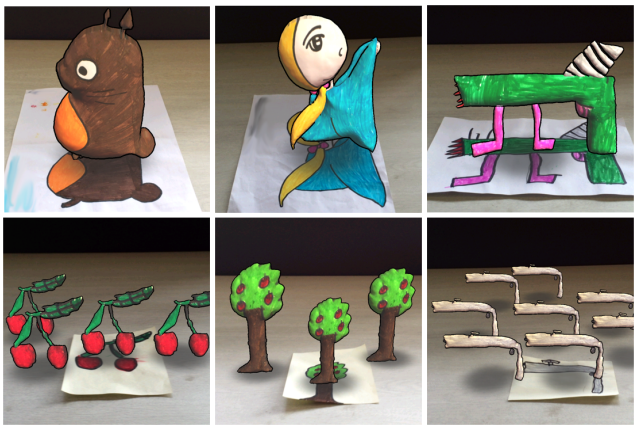

Figure 12: More results authored by children with our system.

The user can also edit and animate the model in AR context to construct more complicated cartoon scenes.

- **RigMesh**: a sketch-based 3D modeling system on Windows platforms with no AR interface, following the idea of modeling by parts as described in [5]. Users interactively create 3D models part-by-part on screen using personal drawings as reference.

- **Chromville**: an augmented reality coloring pages application on mobile platforms, as shown in [6]. Chromville consists of a series of coloring template pages that can be downloaded and printed off ready for coloring. Once children have finished their drawings, Chromville can be used to recognize the coloring pages and then show pre-made 3D models textured with the coloring results.

We note that the three systems are different, but we focus on comparing them based on their similarities on usages or workflows.

To investigate the influence of modeling process, we compare MagicToon and RigMesh since they both support 3D modeling from 2D inputs. They differ in their workflows and interactions to generate models. MagicToon generates 3D models from 2D cartoon inputs directly, which can produce stylized models with minimum user inputs. RigMesh requires users to sketch on screen part-by-part based on 2D inputs to create deformable primitives, which can produce models with more details but is more time-consuming and lack textures.

On the other hand, MagicToon and Chromville have similarities in their workflows combining paper drawing and AR. They both require users to draw on real paper and then use mobile devices to interact with stylized textured 3D models in AR environments. While MagicToon enables creatively sketching and coloring on paper and generating novel 3D models, Chromville only allows coloring on pre-made line drawings and texturing pre-made models.

6.2 Participants

43 participants were participated in our user study, 18 male and 25 female. These participants were aging from 10 to 13 years. None of the subjects had significant 3D modeling experience, 2D artistic experience or AR experience. All subjects stated that they regularly draw and color on paper, and a few had experience of drawing via applications on computers or tablets.

Figure 11 shows two subjects were using our system to model objects and author cartoon scenes. Figure 12 shows more results created by the subjects.

6.3 Experimental Design and Procedure

After a 20 minutes training of the three systems, each subject had to accomplish four tasks:

- **Task**$_{\text{rigO}}$: use RigMesh to model a creative object.

- **Task**$_{\text{magO}}$: use MagicToon to model the same object in Task$_{\text{rigO}}$ by using the model creator.

- **Task**$_{\text{magS}}$: use MagicToon to author a cartoon scene by using our interactive model editor.

- **Task**$_{\text{chrS}}$: use Chromville to color a template page of a cartoon scene in line drawing.

When modeling objects, the subjects were allowed to look at some example drawings we provided, but they must use the same reference drawings during $Task_{rigO}$ and $Task_{magO}$. If the subject intended to model an object from imagination, she/he was told to first draw it on a blank sheet of paper and then used the drawing as inputs of both RigMesh and MagicToon. The duration of each task was planned as 15 minutes. For $Task_{magS}$, the subjects were allowed to use their own drawings along with existed drawings created by other subjects or provided by us.

After accomplishing each task, the subjects needed to answer a questionnaire with six questions to measure the six dimensions of NASA-TLX [17] (physical demand, mental demand, temporal demand, effort, performance and frustration). After they accomplished all tasks, five additional questions were asked:

- **Q1**: which one do you prefer to use to model an object, RigMesh or MagicToon?
- **Q2**: which type of models do you prefer, the detailed but uncolored models with RigMesh or the textured but coarse models with MagicToon?
- **Q3**: which one do you prefer to use to get a coloring cartoon scene, MagicToon or Chromville?
- **Q4**: Rank the three systems according to their supports for your imagination and creativity from high to low.
- **Q5**: Rank the three systems according to your preference from high to low.

We chose NASA-TLX because it has been widely used in human factors research since its introduction in 1988 and most studies addressed questions about interface design or evaluation [16]. Our research is also a kind of interface design and therefore we chose NASA-TLX as a part of our questionnaire to rate perceived workload in order to assess the three systems. We did not stick to comparing interactions among the three evaluated systems because it is almost impossible and unfair in consideration of their differences in interactions. We intended to allow our subjects to experience those differences and similarities through these tasks and analyze which types of interactions and results were more attractive to them through our user study questions.

6.4 Study Results

Figure 13 shows mean values of the six dimensions of NASA-TLX for all the tasks. Figure 14 shows the vote results of the first three additional questions. For the last two questions of ranking, we count a score of 3 for the system in the highest ranking and 1 for the system in the lowest ranking. Figure 15 shows the average scores of the last two questions. Figure 16 shows the percentages of the first ranking frequencies of the three systems in the last two additional questions, in other words, how many female/male gave the highest rankings to the system. Note that Figure 14, Figure 15 and Figure 16 also show differences between the results from male and female.

According to Figure 13, we can get some findings. First, the physical demand values of $Task_{rigO}$ (M = 2.28) and $Task_{chrS}$ (M = 2.28) were almost the same and higher than $Task_{magO}$ (M = 1.46) and $Task_{magS}$ (M = 1.25). Similarly, the temporal demand values of $Task_{rigO}$ (M = 1.86) and $Task_{chrS}$ (M = 1.88) were almost the same and higher than $Task_{magO}$ (M = 1.30) and $Task_{magS}$ (M = 1.39). This indicated that MagicToon was less time-consuming and laborious than RigMesh and Chromville.

Compared with the tedious part-by-part modeling operations in RigMesh, MagicToon combined real-world drawing and a few

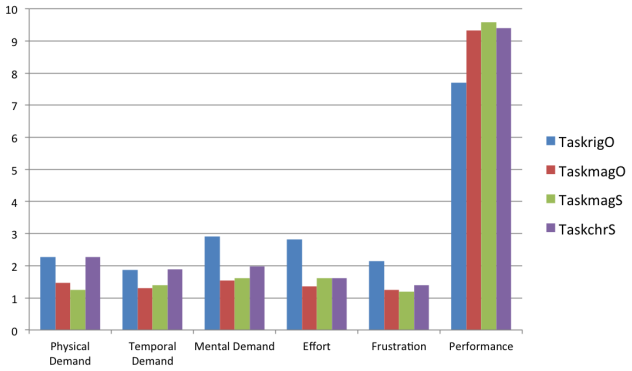

Figure 13: Mean values of the six dimensions of NASA-TLX.

touchscreen operations, which was much easier to use for children. In our experiments, we found that it was not much difficult for the subjects to draw their own cartoon images. When doing $Task_{magO}$ and $Task_{magS}$, many subjects searched on the Internet to find reference images of their favorite cartoon characters and then just drew them on the paper. Some other subjects decided to draw simple images of their familiar objects, such as trees or animals, which only took a few minutes. The subjects were free to draw on small pieces of paper and they had much control on the complexity of the shapes.

We also found that it was not as easy as we predicted for the subjects to color pages when using Chromville. The template pages often contained a large number of blank regions and the subjects struggled to decide how to color all of these regions, which increased the physical and mental demand as well as temporal values of Chromville. In fact, many subjects completed their own cartoon drawings within 5 minutes but needed more than 10 minutes to color the template pages of Chromville. The robustness of Chromville also affected the results. Chromville failed to recognize some pages (5 in 43) after the subjects colored the whole pages. Therefore, those subjects gave high effort rates, which increased the mean effort value of Chromville. Although MagicToon also failed to generate some regions of a drawing occasionally, it was easy to fix the problem by just making their outlines thicker.

RigMesh also showed much higher values in mental demand (M = 2.91), effort (M = 2.81) and frustration (M = 2.14) values and lower value in performance (M = 7.70) compared with MagicToon and Chromville. The subjects often required more than 15 minutes to complete their modeling tasks when using RigMesh. Although RigMesh allowed them to model objects with more details, many children struggled with creating models with sharp corners and complained the system could not create models with the ex-

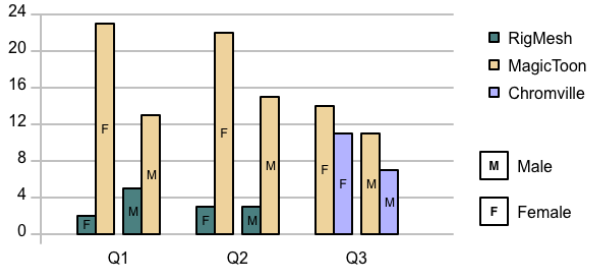

Figure 14: Vote results of the first three additional questions.

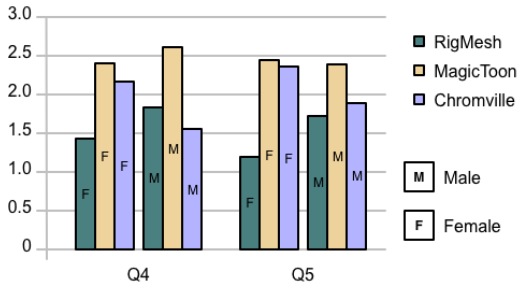

Figure 15: Mean ranking scores for the last two additional questions.

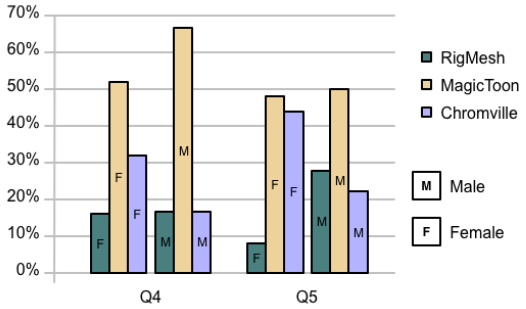

Figure 16: Percentages of the first ranking frequencies of the three systems in the last two additional questions.

actly same silhouettes they sketched, which made them easily mentally tired and discouraged. Another reason of the low performance value was that the resulted models of RigMesh lacked texture information from their feedbacks.

The results also showed that a majority subjects preferred the drawing input procedure and textured results of MagicToon than those of RigMesh, as shown in Figure 14. In addition, more than 50% subjects considered MagicToon stimulate their creativity the most and chose it as their favorite system among the three systems. From their feedbacks, we found that coloring and modeling in AR context gave them a novel way to interact with virtual models when using MagicToon.

We also found there were distinct differences between the preferences of male and female, as shown in Figure 16. In Q4, nearly twice as many female subjects gave the highest ranking to Chromville (P = 32.0%) as RigMesh (P = 16.0%), however the percentages of male were the same (P = 16.7%). In Q5, the percentages of the highest ranking of MagicToon (P = 48.0%) and Chromville (P = 44.0%) were almost the same from female subjects, which were much higher than the percentage of RigMesh (P = 8.0%). However, almost the same number of male gave the highest rankings to RigMesh (P = 27.8%) and Chromville (P = 22.2%), which were lower than the number for MagicToon (P = 50%). From their feedbacks, we found that female enjoyed in coloring activities much more than male and became frustrated more easily when using RigMesh compared with male. As a result, female preferences of RigMesh were much lower than those of the other two systems. For male subjects, they showed less enjoyment in coloring activities. Some male subjects stated they did not want to color any template pages of Chromville, so RigMesh was more attractive to them than Chromville. Although MagicToon also required them to draw on the paper, they were free to quickly create their own objects such as guns and cars with much control of the sizes and complexity of the shapes.

6.5 User Feedbacks

We also collected user feedbacks after the subjects had used our system. In summary, children gave many positive feedbacks when using our system to model and author cartoon scenes. They spoke highly of our automatic model creator that saved a lot of their time to get personalized models. In addition, the interactions with mobile AR in MagicToon were much attractive to them. Children who were not interested in drawing and coloring so much were also willing to use our system due to that they could quickly obtain personalized cartoon models just by sketching on the paper within a few minutes. The subjects deeply enjoyed showing and sharing their cartoon scenes with the others. For example, the gun drawings shown in the right column of Figure 12 were drawn by two male children. They intended to create different guns to compete with each other for fun.

During our user study, we found that the copy operation was the highest frequently used, followed by the merge and affine transformation operations. The subjects preferred to use simple operations than those with more steps, such as making animations. We also asked their suggestions for the future features they wanted to see in MagicToon. One suggested we could further simplify the animation operation and include more interesting types of animations. A few subjects expressed their wishes to add personalized dialogs and sounds to their virtual cartoon characters. At the end of our user study, many subjects said they would like to download MagicToon if it was uploaded to online app stores.

7 Limitations

Our system still has its limitations. First, the surrounding environments, such as the lighting conditions, shadowing and reflections, sometimes influence the robustness of our segmentation algorithm, especially when the contrast between the outline and region colors is not very distinct. For example, because the color of the trunk of the *Tree* in Figure 12 is dark brown, which is closer to the outline color, the system might fail to generate a model from the drawing when the lighting conditions are not so well. The system also fails to generate models when the outlines are too thin or not enclosed. This problem can be easily solved by just re-outlining the image, making the outlines thicker and enclosed. We plan to improve our segmentation algorithm to avoid these situations in the future.

Second, the 3D models generated by our system lack depth orders between parts. Due to this limitation, some parts of the resulted model cannot be seen such as the ornaments of the *Tree* in Figure 10, because they are hidden in the interior of the model. We intend to employ more intelligent methods to predict the relative depth orders between regions, such as [13]. In addition, our system is best suitable for front-view cartoon drawings. Models generated from non-front view drawings might seem counter-intuitive, especially when observed from side views. Figure 17(a) shows such an example. Another limitation of the mesh generation is that it cannot generate sharp models such as cubes. We will introduce more tools that allow users to modify the distance field in order to create more different shapes. Capturing drawings from inclined views may also result in odd cartoon models, as shown in Figure 17(b). This can also greatly decrease the tracking quality in AR environments. An automatic image calibration may ease the situation.

Finally, the original drawings are used as textures for both front and back faces in our implementation, which may produce unsatisfactory results. This problem can be fixed by using more advanced texture synthesis methods.

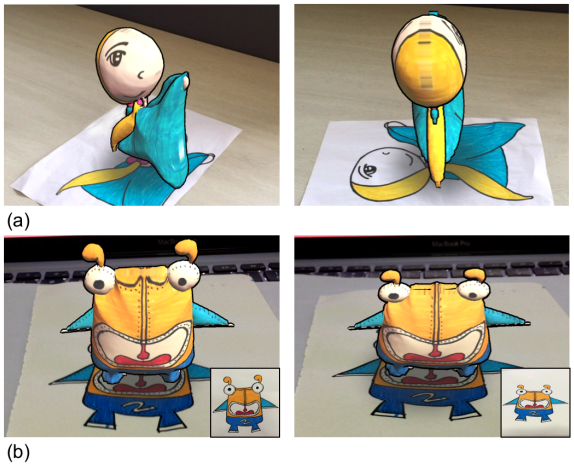

Figure 17: Failure cases. (a) A failure case when modeling from a non-front view cartoon image. (b) The left shows the normal model obtained by capturing a drawing from an overhead view and the right shows the odd one obtained from an inclined view.

8 Conclusions

We present MagicToon, a creative 2D-to-3D personalized modeling system on mobile devices. Given cartoon drawings as inputs, our model creator can automatically generate 3D textured models with minimum inputs and overlay them on the real world, bringing life to flat drawings. We have implemented an interactive model editor that supports to easily author cartoon scenes in AR environments, giving full play to children's innovations. Our results have shown that MagicToon has relative advantages compared with both traditional sketch-based modeling systems and AR coloring books. A possible extension of our system is to support storytelling functionalities in the future, such as adding dialogs and sounds to the virtual cartoon characters.

Acknowledgements

We would like to thank Fan Jiang, Cheng Yang, Chao Yin and Jieyu Chu for their assistance and advice during the research. We are grateful to all the subjects who participated in our user studies. This work is supported in part by the National Natural Science Foundation of China (No. 61173105 and 61373085) and the National High Technology Research and Development Program of China (No. 2015AA016404).

References

[1] S. Arroyave-Tobón, G. Osorio-Gómez, and J. F. Cardona-McCormick. Air-modelling: a tool for gesture-based solid modelling in context during early design stages in ar environments. *Computers in Industry*, 66, 2015.

[2] I. Baran and J. Popović. Automatic rigging and animation of 3d characters. In *ACM Transactions on Graphics (TOG)*, volume 26, 2007.

[3] O. Bergig, N. Hagbi, J. El-Sana, and M. Billinghurst. In-place 3d sketching for authoring and augmenting mechanical systems. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, 2009.

[4] M. Billinghurst, H. Kato, and I. Poupyrev. Magicbook: Transitioning between reality and virtuality. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '01, 2001.

[5] P. Borosán, M. Jin, D. DeCarlo, Y. Gingold, and A. Nealen. Rigmesh: Automatic rigging for part-based shape modeling and deformation. *ACM Trans. Graph.*, 31(6), 2012.

[6] Chromville. Chromville. https://chromville.com/.

[7] A. Clark and A. Dunser. An interactive augmented reality coloring book. In *3D User Interfaces (3DUI), 2012 IEEE Symposium on*, 2012.

[8] Crayola. Crayola color alive. http://www.crayola.com/.

[9] E. S. De Lima, B. Feijó, S. D. Barbosa, A. L. Furtado, A. E. Ciarlini, and C. T. Pozzer. Draw your own story: Paper and pencil interactive storytelling. *Entertainment Computing*, 5(1), 2014.

[10] V. Developer. Sdk, unity extension vuforia–2.8, 2014.

[11] Disney. Color and play. http://www.onlycoloringpages.com/.

[12] P. Fiala and N. Adamo-Villani. Arpm: an augmented reality interface for polygonal modeling. In *Mixed and Augmented Reality, 2005. Proceedings. Fourth IEEE and ACM International Symposium on*, 2005.

[13] D. Geiger, H. Pao, and N. Rubin. Salient and multiple illusory surfaces. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, 1998.

[14] Y. Gingold, T. Igarashi, and D. Zorin. Structured annotations for 2d-to-3d modeling. In *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09, 2009.

[15] N. Hagbi, R. Grasset, O. Bergig, M. Billinghurst, and J. El-Sana. In-place sketching for content authoring in augmented reality games. In *Virtual Reality Conference (VR), 2010 IEEE*, 2010.

[16] S. G. Hart. Nasa-task load index (nasa-tlx); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 50, 2006.

[17] S. G. Hart and L. E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Advances in psychology*, 52, 1988.

[18] W. Hürst and J. Dekker. Tracking-based interaction for object creation in mobile augmented reality. In *Proceedings of the 21st ACM international conference on Multimedia*, 2013.

[19] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3d freeform design. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, 1999.

[20] S. Magnenat, D. T. Ngo, F. Zund, M. Ryffel, G. Noris, G. Rothlin, A. Marra, M. Nitti, P. Fua, M. Gross, et al. Live texturing of augmented reality characters from colored drawings. *Visualization and Computer Graphics, IEEE Transactions on*, 21(11), 2015.

[21] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Fibermesh: Designing freeform surfaces with 3d curves. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, 2007.

[22] L. Olsen, F. F. Samavati, and J. A. Jorge. Naturasketch: Modeling from images and natural sketches. *Computer Graphics and Applications, IEEE*, 31(6), 2011.

[23] S. Oviatt. Human-centered design meets cognitive load theory: designing interfaces that help people think. In *Proceedings of the 14th annual ACM international conference on Multimedia*, 2006.

[24] R. Schmidt, T. Isenberg, P. Jepp, K. Singh, and B. Wyvill. Sketching, scaffolding, and inking: A visual history for interactive 3d modeling. In *Proceedings of the 5th International Symposium on Non-photorealistic Animation and Rendering*, NPAR '07, 2007.

[25] R. Schmidt, B. Wyvill, M. C. Sousa, and J. A. Jorge. Shapeshop: Sketch-based solid modeling with blobtrees. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, 2007.

[26] J. R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational geometry*, 22(1), 2002.

[27] J. Vollmer, R. Mencl, and H. Mueller. Improved laplacian smoothing of noisy surface meshes. In *Computer Graphics Forum*, volume 18, 1999.

[28] T. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3), 1984.

[29] F. Zünd, M. Ryffel, S. Magnenat, A. Marra, M. Nitti, M. Kapadia, G. Noris, K. Mitchell, M. Gross, and R. W. Sumner. Augmented creativity: Bridging the real and virtual worlds to enhance creative play. In *SIGGRAPH Asia 2015 Mobile Graphics and Interactive Applications*, SA '15, 2015.