Cartoon Generation for Text-Free User Interfaces

Indrani Medhi[1], Chandima Rathnaweera Patabandhige[2], Kentaro Toyama[1]

[1]Microsoft Research Labs India
Scientia, 196/36, 2nd Main Road, Sadashiv Nagar, Bangalore-560080, India

[2]University of Moratuwa, Sri Lanka

{indranim@microsoft.com, chandimacj@yahoo.com, kentoy@microsoft.com}

Abstract. This paper presents a tool which allows easy generation of semi-abstracted cartoon-like images from photographs, for the purposes of generating user interfaces that can be used by non-literate users.

Non-literate users of computers have difficulty navigating text-based user interfaces. Thus, in previous research, we developed design principles for creating what we call "text-free user interfaces" that are usable by non-literate, first-time users of computers. One of these principles was that text-free interfaces should use static hand-drawn images to represent objects and actions, because they were more consistently understood than oversimplified icons, and at the same time, they were generic enough to be taken for abstract concepts rather than the specific instances portrayed in photographs.

Drawing cartoons, however, is time-, labor-, and skill-intensive. In order to partially automate this process, we have developed a tool which allows a static, semi-abstracted cartoon image to be generated easily from photographs taken by users. The tool works by chaining together a number of simple image-processing algorithms: First, it applies an image blur filter to remove the details or unnecessary edges of the image. Second, the tool identifies the major edges in the image using an edge-detection algorithm. Next, connected areas of the image are identified using a connected components analysis. Then, we repaint each connected component with its average color, so as to "flatten" the image. Finally, detected edges are superimposed back onto the flattened image. The tool allows user intervention at any stage so that particular steps can be customized for a better final appearance.

Key words: illiterate users, user interfaces, static hand-drawn representations

1. Introduction

Recently, there has been growing interest in computing applications for socio-economic development of underserved communities [21, 24, 25, 26, 27, 28, 29, 30, 33, 36]. These projects seek to provide underserved populations such as poor rural farmers, urban domestic workers, or the unbanked with relevant communication and information services, such as weather updates, employment opportunities, or microcredit. The projects often use Internet-connected personal computers and mobile phones as the technological substrate to deliver these services.

One characteristic that many of these communities share is low levels of literacy. Even conservative estimates of illiteracy suggest that there are over one billion non-literate people in the world [22]. In those cases where computing applications are meant to be used directly by such communities, illiteracy can come in the way of designing comprehensible applications.

Previous work in designing user interfaces for non-literate and semi-literate populations focuses on several characteristics, including graphics and icons [11, 14, 25, 26, 27, 29, 30, 31], voice feedback [14, 25, 26, 27, 31], and video [24].

[2] This author is an undergraduate student at the University of Moratuwa in Sri Lanka. This work was done while he was an intern at Microsoft Research Labs India.

Our research over the past couple of years confirms these findings. We have worked on what we call "text-free user interfaces", where we have looked at designing user interfaces for non-literate users, developed through an extensive ethnographic study involving over 300 hours and 250 people from urban slums in Bangalore, India. We established several design principles that we believe could be applicable to other user groups that are non-literate and new to computer use [24, 25, 26, 27]. Some of these are eliminating text, using static hand-drawn imagery of semi-abstracted cartoons, providing voice feedback for all functional units, providing a consistent help feature, and using full-context video to dramatize the purpose of the application.

In taking a closer look at what sort of imagery was best suited for text-free UIs, we conducted user tests to determine which of a variety of audio-visual representations of concepts were best understood by non-literate subjects. [25, 27]. Informal observations with non-literate user communities suggested that subjects recognized semi-abstract cartoons better than photographs or simplified icons. So, we conducted a formal study to verify this [26]. In the study, we presented each of 13 different health symptoms to 200 illiterate subjects in one representation randomly selected among the following ten: text, static drawings, static photographs, hand-drawn animations, and video, each with and without voice annotation. The goal was to see how comprehensible these representation types were for an illiterate audience. Using a methodology that generated different representations in a way that fairly stacks one representational type against the others, we generated representations for each of the aforementioned representational types for a number of healthcare-related concepts. The results verified our hypothesis. In the case of real photographs, the extraneous detail in the background confused our subjects. Photographs and video were also prone to be interpreted as the literal instance of the photograph, rather than a general concept. In some cases this was favorable to the understanding of a graphic while in others, there was confusion created because of direct association. In the case of dynamic representations like video and animation, placing a context-laying activity that was unrelated to the principal action cue caused confusion amongst the subjects. Our main result was that static-hand drawn representations with voice annotations were the best understood among all the media. So, for example, in trying to portray the concept of "fever," an overly simplified iconic representation might be mistaken for anger (or hot-headedness) or a type of illness; a photograph, on the other hand, could be mistaken for Mr. Rao's case of typhoid from last year. A semi-abstracted image of a person with a moist cloth on her forehead, was more likely than either of the other options to be taken to represent "fever" as an abstract concept.

In order to allow content authors to generate text-free user interfaces easily, we developed a text-free UI-generation tool which could be used to build user interfaces for non-literate users based on the design principles discussed above. Most of the process can be automated or done easily by non-specialists. However, creating the semi-abstracted cartoons requires reasonable skill at illustration. Even with skill, drawing such cartoons can be a labor-intensive and time-consuming process. To meet this challenge, we developed a cartoon-generation tool that makes it easy to generate semi-abstracted cartoons from photographs.

This paper presents the specifics of the cartoon-generation tool and a domain in which the tool was used to create a text-free user interface. The technical components of the tool itself are not novel, but we have chosen components which are easy to understand by a lay user, so that generation of a cartoon image from a photograph is as simple as possible. Although there is existing work to create cartoons from photographs with user assistance, to our knowledge, this is the first instance in which similar algorithms have been applied to the problem of generating semi-abstracted cartoons for the purposes of creating user interfaces for non-literate users.

2. Cartoon Generation Tool

The goal of our cartoon-generation tool is to allow users to easily create cartoon-like images from photographs. Cartoons are simplified drawings that generally have the following characteristics [15].

- Strong edges: Semantically coherent regions of the image are often delineated by strong black edges.
- Flat, simple, saturated coloring: Regions are represented by a uniform color, and relatively few colors are used in a single cartoon.
- Simplified, suppressed background: Background elements contain less detail and are often muted in color.

Our cartoon-generation tool attempts to capture these elements and to produce them automatically from a photograph. The tool can work in two modes: In automated mode, the tool runs through a number of algorithms

using parameters which were empirically chosen to produce cartoon-like images from photographs. In custom mode, users can set these parameters by hand.

Below, we describe the individual components which are involved in arriving at a cartoon-like output. These are all standard image-processing algorithms, and an effort has been made to choose those which are most easily understood by lay users, so that the authoring process itself is as simple as possible. The process can be run fully automated (though with mixed results depending on the input photograph), such that cartoon-like images are generated from photographs with no user intervention, or it can be run in custom mode with more manual tweaking.

2.1. Gaussian Blur

The first step in the cartoonization process is a Gaussian blur. Blurring eliminates the minor speckling in the image and reduces the number of connected components (to be described below) that the image is segmented into. A Gaussian blur provides a comparatively gentler smoothing and preserves edges better than a similarly sized mean filter.

Our default blur uses a 3x3 convolution matrix. Each pixel's value is set to a weighted average of that pixel's neighborhood, with the filter applied to each of the R, G, and B channels. The 3x3 matrix that we use for convolution is a standard approximation for a Gaussian filter, namely [1, 2, 1; 2, 4, 2; 1, 2, 1]. For greater blurs, there is the option of applying this filter multiple times, or of convolving with a larger filter (in which case, a fast Fourier transform operation is computationally more efficient).

This step is applied with blurring filters of various sizes in the automated mode, and it is altogether optional in custom mode.

2.2. Edge Detection

As the next step we try to identify the edges of the components containing in the image. This step assists us to draw outlines of the image at the latter stages. We use the standard Sobel edge-detection algorithm to detect edges in the image [13], and then take the summed absolute value of computed edges in the *x*- and *y*- directions. Edge detection is applied only to the green channel (which contributes the greatest component to luminance), for efficiency. Finally, the summed value is thresholded so that edges are a binary “on” or “off”. As an inexpensive way to detect wider edges, we also allow the Sobel filter to be expanded such that pixels *m*-pixels away from the center pixel are sampled when determining whether an edge exists. For example, if we are searching for an edge at coordinate (x, y), the Sobel filter would compute its non-zero coefficients on pixels (x-m, y-m), (x-m, y), (x-m, y+m), (x, y-m), (x, y), (x, y+m), (x+m, y-m), (x+m, y), (x+m, y+m).

In the automated mode of the tool, and for 8-bit grayscale images, we set the threshold value for an edge to be detected at 113, and an *m*-value of 3 pixel’s width. In custom mode, we allow these values to be set by the user.

2.3. Connected Components

In general, the input image is still at the photorealistic stage and requires image segmentation technique to identify connected regions of the given image for further processing. Here, we use connected-component analysis to identify what we hope to be semantically meaningful regions in the image [9]. This algorithm chooses a pixel at random, and then looks toward its eight neighbors to see whether any pixels are within a given threshold of the chosen pixel. If so, the algorithm continues in a breadth-first search manner, identifying pixel after pixel which is within a threshold of the seed pixel. (The breadth-first approach results in an efficient iterative, rather than a recursive, algorithm.)

In automated mode, a threshold is set *a priori*, at 10 grayscale values (assuming 8-bit values per color channel). However, this part of the algorithm is most sensitive to the quality of the input photograph, and we find that most images require some manual work to connect the right components together. As a result, there is a manually driven procedure that we describe next.

2.4. Merging Semantically Related Components

Since computer-vision techniques are not yet mature to the point of being able to identify semantically related regions in an image, we involve the user to help connect semantically related components in the image so that they

can be merged appropriately. This process involves a point-and-click UI where the user can select multiple pre-identified regions and indicate that they should belong to the same region. Merged regions are then treated as a single region afterwards.

2.5. Color Averaging

Having identified the connected components of the image, we calculate the average color of each component and repaint the image to make it more like a non-photorealistic painterly image. This is done separately for each of the RGB channels. For each connected region, the sum of the pixel values is computed and divided by the pixel count of the connected region. This step requires no manual intervention and is the same in both automated and custom modes.

2.6. Color Saturation Adjustment

Cartoons often use saturation and de-saturation of colors to indicate the relative importance of image elements. Prominent items are usually painted in saturated colors, whereas backgrounds are frequently de-saturated. To allow for this, we allow the user to set saturation levels for regions identified. For each region that needs to be adjusted, we compute hue, saturation and value/brightness (HSV/HSB) values corresponding to the region's RGB values [16]. We then allow users to set the saturation value. This step is entirely a manual process, as there are no reliable methods for automatically picking out what is semantically important in an image (and in any case, it may depend on the intent of the graphic).

2.7. Superimposition of Edges

Once regions are identified and adjusted, we superimpose the earlier detected edges on top. This is a straightforward process where identified edge pixels are set to black. For custom mode, we allow users to draw additional edges where they might feel they are necessary.

3. Implementation

The cartoon-generation tool is embedded in a larger tool that allows for users to design text-free UIs. The larger tool allows users to develop entire UIs for applications designed for non-literate users, and it incorporates simple interactions such as hypertext navigation through buttons, and so forth.

In the authoring tool, potential users design and create text-free UIs with images, voice annotations, animations and navigation between pages according to the type of content they want to convey to their users in the type of information structure they desire. Graphics and audio elements can be incorporated as necessary, with voice annotations being able to be input while the UI is being designed.

The cartoon-generation tool is set up as a preprocessing tool which allows users to select multiple photographs and turn them into cartoons for inclusion into a UI. If the user opts for the fully automated mode, the system offers three results per photograph, with the middle photograph using the parameters defined above, and the other two with some perturbation in the parameters. The user can then click on the desired output cartoon.

4. Example Domain and Results

In order to test our work in a concrete context, we chose healthcare information dissemination in rural clinics. To understand how these clinics operate, we visited a partner hospital, a specialty eye-care institution in Coimbatore, Tamil Nadu in India which is dedicated to eradicating preventable and curable blindness through a large rural outreach program. The hospital has two facilities in Tamil Nadu and one in Andhra Pradesh. Similar facilities are being currently set up in Karnataka and Gujarat.

The Coimbatore hospital has 500 free beds and 50 beds for paying patients. On an average 150-200 operations per day are performed at this hospital. On weekends, camps are set-up at villages in different districts in Tamil Nadu. At any peak time, there are 600-700 people at these camps, being screened by the medical staff to be brought to the base hospital for treatment. During such peaks, patients spend a lot of time in waiting areas, and the hospital uses these opportunities to educate patients, their family, and friends about basic preventive and curative health care.

Current methods of education include the distribution of pamphlets and audio recordings delivered over a public-announcement system. However, around 80% of the patients at this hospital, who come from villages of Tamil Nadu, are functionally non-literate, and unable to read books or newspapers with any fluency. So, while the pamphlets contain visual elements, the text is of little value; conversely the audio broadcasts are comprehensible to patients, but without visual information. Since the ratio of paramedics to the number of people is very low, it is prohibitive for paramedics to do a one-on-one interaction with patients for the purpose of education.

The hospital is seeking ways to disseminate information more effectively and needs tools to develop computer applications for its non-literate user base. Hence, the need for a text-free UI that can deliver information designed in a way to be used by non-literate users from the outset.

We have developed a pilot application that includes some basic information about diabetes (which can lead to blindness and other vision problems, if untreated). Some examples of the kinds of imagery used in this application, using the cartoon-generation tool are shown in Figure 1in the Appendix.

The tool is not perfect, by any means, and there is still work remaining to make the tool applicable to a wider range of photographs. In particular, depending on the graininess or texturing of the image, it can be difficult to generate images that are sufficiently cartoon-like to be seen as "semi-abstracted," which is the critical trait. The example in Figure 2 in the Appendix shows a case where the photograph is segmented into so many regions at first that the effort required to merge them by hand is probably much more than effort of tracing over relevant lines, etc.

5. Related Work

There are two areas of related work that are particular relevant to our research. The first is research work in tools for stylizing and abstracting images. The second is literature that looks at designing user interfaces for non-literate users. To our knowledge, our work occurs at the unexplored intersection of these two areas with some differences from existing research that differentiates our work.

5.1. Tools for Stylizing and Abstracting Images and Non-Photorealistic Rendering

The first stream of relevant work looks at developing tools for stylizing and abstracting images. All of this work focuses on non-photorealistic rendering of photographs. Among these, there are studies that look at tools converting photographs into cartoon sketches [1] , [32] , [38] , [35] free-hand style drawing [2] , [8] , [17] , [19] , [21] and artistic brush stroke images [5] , [12] , [10] , [20] , [32]. There is also some work that looks at converting video stills into abstracted cartoon animation into a range of styles [18].

The approach of almost all of the previous work is that they first try to identify semantic components and then simplify colors within a region through various means. Except for the work that seeks to make artistic paint-brush strokes, identification of outline edges is done through a manual process. All of the previous work also uses image segmentation [35] and color segmentation techniques (with the one exception which integrates eye-tracking technique with image segmentation [7] to identify the areas that are required to be abstracted). Once the output is available some of this work also looks at adding features to generated image [4]. Most previous work in this vein also appears requires a great amount of manual detailing, as well, and the effects are meant primarily to help achieve a certain "look" than to simplify the abstraction process. Some of this work goes as far as to try to imitate certain styles of artists.

In our work, the resulting images are not as artistically interesting, but the output image can be generated with fewer steps and less user intervention.

We also note that commercialized software, such as Adobe Photoshop is also available to many of the intermediate steps in our algorithm [39]. However, such software has no one-button feature to convert photographs to cartoon-like images. Users have to try several image-processing filters and approaches to generate a cartoon image using those tools. In contrast, we provide an automated mode that makes generation of cartoons from photographs trivially simple.

5.2. User Interfaces for Non-literate Users

The second stream of relevant work investigates user interfaces for non-literate users.

Most previous work with non-literate users focuses on the mechanics of the UI. In particular, researchers immediately intuited the value of imagery in place of text, and extensive use of graphics is advocated by this work [11, 14, 25, 26, 27, 29, 30]. Among these, some also investigated the value of voice annotations and instructions, which are of obvious value to non-literate users [25, 26, 27, 31]. Much of the interesting work in this area investigates the subtleties of graphics-intensive, audio-based UIs. Some authors note the plausible inclusion of number, as non-literate users are often numerate [25, 27, 29, 30, 31]. Others focus on the need for ultra-simplified navigability as a design element [11]. While this previous work suggests excellent UI design elements for the non-literate user, none so far looks at providing capacity to build such interfaces to organizations that are directly required to address the information needs of non-literate users. With the cartoon-generation tool, our goal was not only to explain what kinds of UI design was required for use by non-literate users, but also to provide a tool that allows lay computer users to design such UIs. The hope is that organizations who cater to end-user information needs of non-literate users would be able to design their own text-free UIs.

In summary, while generating cartoons from photographs in itself is not a new area, our work takes this research a step further to practical application within a particular domain. To our knowledge, the work presented in this paper is the first of its kind with this focus.

6. Conclusions and Future Work

As part of a comprehensive text-free UI authoring tool for generating user interfaces for non-literate users, we developed a cartoon-generation tool that allows users to easily transform photographs into semi-abstracted cartoons. We believe that organizations which cater to non-literate users will be able to use this tool to generate content based on design principles established in earlier work on text-free user interfaces.

In future work, we are looking at testing the usability of this tool with our target communities – organizations building computer applications for their non-literate customers and clients. We would like to do controlled experiments with target users to test the usability of functionalities and ease of use of this tool. We would also like to deploy the tool at these organizations so that we can ultimately measure how accurately automatically generated cartoon images are understood by non-literate and semi-literate users.

References

1. Agarwala, A. Snaketoonz: A semi-automatic approach to creating cel animation from video. In Proceedings of NPAR 2002. (2002).
2. Atsushi Kasao, Kazunori Miyata: "Enhanced SIC (Synergistic Image Creator) for Artistic Use," *IV*, pp. 903-911, Ninth International Conference on Information Visualisation (IV'05), (2005).
3. Chand, A: Designing for the Indian rural population: Interaction design challenges. Proc. Development by Design Conference, (2002).
4. Chen, H., Zheng, N. N., Liang, L., Li, Y., Xu, Y. Q. And Shum, H. Y.: "Pictoon: A Personalized Image-Based Cartoon System", in Proc. 10th ACM Int'l Conf. on Multimedia, (2002).
5. Collomosse, J. P. and Hall, P. M.: Cubist Style Rendering from Photographs. IEEE Transactions on Visualization and Computer Graphics, (2002).
6. Cooper, A. and Reimann, R.: About Face 2.0, the Essentials of Interaction Design. Wiley Publishing Inc. USA, (2003).
7. Decarlo, D. and Santella, A.: Stylization and abstraction of photograph. In Proceedings of ACM SIGGRAPH 2002, 769–776. (2002).
8. Diego Nehab and Luiz Velho.: Multiscale moment-based painterly rendering. In SIBGRAPI'02, pages 244–251. IEEE, (2002).
9. Duda R.O. and P.E. Hart.: Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM, 15(1):11-15. (1972).
10. Gooch, B., Coombe, G., and Shirley, P.: Artistic Vision: Painterly Rendering Using Computer Vision Techniques. Proceedings of NPAR, p. 83-90, (2002).

11. Grisedale, S., Graves, M and Grünsteidl, A.: Designing a graphical user interface for healthcare workers in rural India, Proceedings of the SIGCHI conference on Human factors in computing systems, p.471-478, March 22-27, 1997, Atlanta, Georgia, United States
12. Hertzmann, A.: Painterly Rendering with Curved Brush Strokes of Multiple Sizes. (1998).
13. Horn, B. K. P.: Robot Vision, MIT Press, McGraw-Hill Book Company. (1986).
14. Huenerfauth, M.: Developing design recommendations for computer interfaces accessible to illiterate users. Master's thesis, University College Dublin, (2002).
15. Ianeva, T., de Vries, A., and Rohrig, H.: Detecting cartoons: A case study in automatic video-genre classification, in IEEE International Conference on Multimedia and Expo, vol. 1, pp. 449 – 452, (2003).
16. James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, and Richard L. Phillips: Computer Graphics: Principles and Practice, Second Edition, Addison-Wesley, pp. 488-491. (1995)
17. Johan, H., Hashimota, R., and Nishita, T.: Creating watercolor style images taking into account painting techniques. Journal of the Society for Art and Science 3, 4, 207.215. (2005).
18. Jue Wang, Yingqing Xu, Heung-Yeung Shum and Michael F. Cohen.: Video Tooning. ACM Transaction on Graphics, 23(3) (Proc. SIGGRAPH2004), 574-583, (2004).
19. Kovács, L., and Szirányi, T.: 2D Multilayer Painterly Rendering with Automatic Focus Extraction.WSCG 2006, Full Papers Proceedings, ISBN 80-86943-03-8, p. 141-145, Plzen, Czech Republic, (2006).
20. Kovács, L., and Szirányi, T: Efficient Coding of Stroke-Rendered Paintings. ICPR (2): 835-838. (2004).
21. Kovács, L., and Szirányi, T: Painterly rendering controlled by multiscale image features, Proceedings of the 20th spring conference on Computer graphics, Budmerice, Slovakia, (2004).
22. Lourie S. World literacy: where we stand today - One Billion Illiterates – editorial, UNESCO Courier**.** (July 1990)
23. Medhi, I. and Kuriyan R.: Text-Free UI: Prospects for Social Inclusion. International Conference on Social Implications of Computers in Developing countries. Brazil, May (2007).
24. Medhi, I. and Toyama, K.: Full-Context Videos for First-Time, Illiterate Users. ALT. CHI forum at ACM CHI '07, San Jose, USA, April (2007).
25. Medhi, I., Pitti B. and Toyama K: Text-Free UI for Employment Search. Asian Applied Computing Conference. Nepal, December (2005).
26. Medhi, I., Prasad, A. and Toyama K: Optimal Audio-Visual Representations for Illiterate Users. International World Wide Web Conference Committee. Canada, May (2007).
27. Medhi, I., Sagar A., and Toyama K: Text-Free User Interfaces for Illiterate and Semi-Literate Users. International Conference on Information and Communication Technologies and Development. Berkeley, USA, May (2006).
28. Mitra. S. Self organizing systems for mass computer literacy: Findings from the hole in the wall experiments. *International Journal of Development Issues,* Vol. 4, No. 1 (2005) 71 – 81.
29. Parikh, T. Ghosh K. and Chavan, A.: Design Considerations for a Financial Management System for Rural, Semi-literate Users. ACM Conference on Computer-Human Interaction, (2003).
30. Parikh, T. Ghosh K. and Chavan, A.: Design Studies for a Financial Management System for Micro-credit Groups in Rural India. ACM Conference on Universal Usability, (2003).
31. Parikh, T.: HISAAB: An Experiment in Numerical Interfaces, Media Lab Asia Panel Discussion, Baramati Initiative on ICT and Development, (2002).
32. Philippe Decaudin: Cartoon-Looking Rendering of 3D-Scenes, Research Report INRIA. (1996).
33. Ratnam, B. V., Reddy P.K., and Reddy, G. S. eSagu: An IT based personalized agricultural extension system prototype – analysis of 51 Farmers' case studies *International Journal of Education and Development using Information and Communication Technology* (IJEDICT), 2005, Vol. 2, Issue 1, pp. 79-94.
34. Szirányi, T., and Tóth, Z.: Optimization of Paintbrush Rendering of Images by Dynamic MCMC Methods. In Lecture Notes on Computer Science, Springer Verlag, Vol.LNCS 2134, 201-215. (2001).
35. Tang, X. and Wang, X.: "Face sketch synthesis and recognition", Proc. Ninth IEEE Int. Conf. on Computer Vision (ICCV), pp.687-694, (2003).
36. The DIL team, Innovative ICT Tools for Information Provision in Agricultural Extension *International Conference on Information and Communication Technologies and Development (Berkeley, USA)*, May 2006.
37. Wang, J., Thiesson, B., Xu, Y., and Cohen, M.: Image and video segmentation by anisotropic kernel mean shift. In Proc. European Conference on Computer Vision (ECCV'04), Volume 2, pages 238.249, (2004).
38. Wen, F., Luan, Q., Liang, L. , Xu, Y., and Shum, H.: Color Sketch generation, The 4th International Symposium on Non-Photorealistic Animation and Rendering (NPAR2006), Annecy, France. (2006).
39. www.adobe.com/products/photoshop/index.html

Appendix

Fig. 1. Several examples of images created from photographs. In the first three examples, we show the photograph, a cartoon drawn by an artist based on the photo (for comparison), an intermediate result from our cartoon-generation tool, and the final result of our cartoon-generation tool.

Fig. 2. Some photographs are difficult to turn into cartoons, even with our tool. Usually, this occurs because the image has so much rich texture that the segmentation process results in many small connected-regions.