3D Views of Traditional 2D Hatching Illustrations using Layering

Bachelor Thesis

Tijmen Klein (S1755250)

July 15, 2010

university of groningen

Abstract

Traditional hatching illustrations are 2D representations of a 3D scene. This thesis is concerned with the creation of 3D views based on such illustrations. A Java application is developed that allows the user to interactively select layers in an illustration, which can be enhanced by snakes. The selected layers are automatically extracted, placed into a 3D scene and modified, by scaling and positioning, to match the perspective of the original illustration. The user is able to adjust the depth of each individual layer. Finally, an anaglyphicly rendered output scene is created, that allows the viewers to perceive the image as a spatial scene.

Contents

Chapter 1

Introduction

Throughout human history, people have generated a lot of two-dimensional (2D) illustrations of historic events and places, these are 2D representations of a three-dimensional (3D) scene. An example of such an illustration can be seen in Figure 1.1, which shows a scene of the murder on Willem of Orange. Currently there are many techniques to generate 3D images, for example using a 3D camera system or a 3D modeling computer program. Unfortunately, we can not go back in time to regenerate the old images using these modern equipment to get a similar 3D impression. This makes it impossible to appreciate the original 3D scene, unless we use a technique to extract information about the third dimension (depth) from the old illustrations.

Figure 1.1: An example of an old hatching illustration, showing the murder on Willem of Orange by Balthasar Gérard on 10 july 1584, source: http://en.wikibooks.org/wiki/File:Moordwillemzwijger2.jpg.

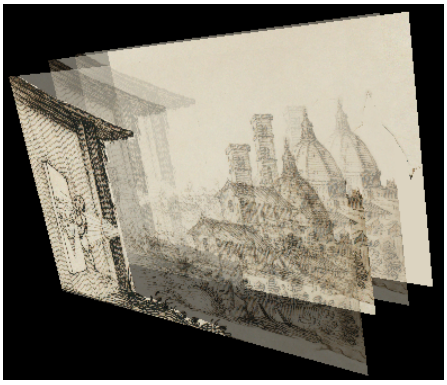
The scene depicted in a illustration consists of implicit layers; these layers are an abstraction of the true 3D scene and are perspectively projected in the illustration. Thus, in order create a 3D scene, these layers need to be extracted and they need to be modified so that the perspective is corrected, as can be seen in Figure 1.2a and 1.2b. This assumes that each scene consists of a limited amount of implicit layers. Furthermore, it is not possible to extract the actual depth of each layer, so these need to be guessed and adjusted by the user.

When these layers have been extracted, they can be placed into a 3D scene. This enables the possibility to move a virtual camera around the scene, or to create a stereoscopic rendered image based on the scene, which will allow the viewer to perceive the depth. This makes old illustrations more interesting for the viewer. This thesis examines how to extract layers from traditional illustrations with the support of precise selection using snakes. A method is provided to place these layers into a 3D view at different depths while maintaining a correct perspective. Furthermore, a possibility is added to create an anaglyphicly rendered image from this 3D scene.

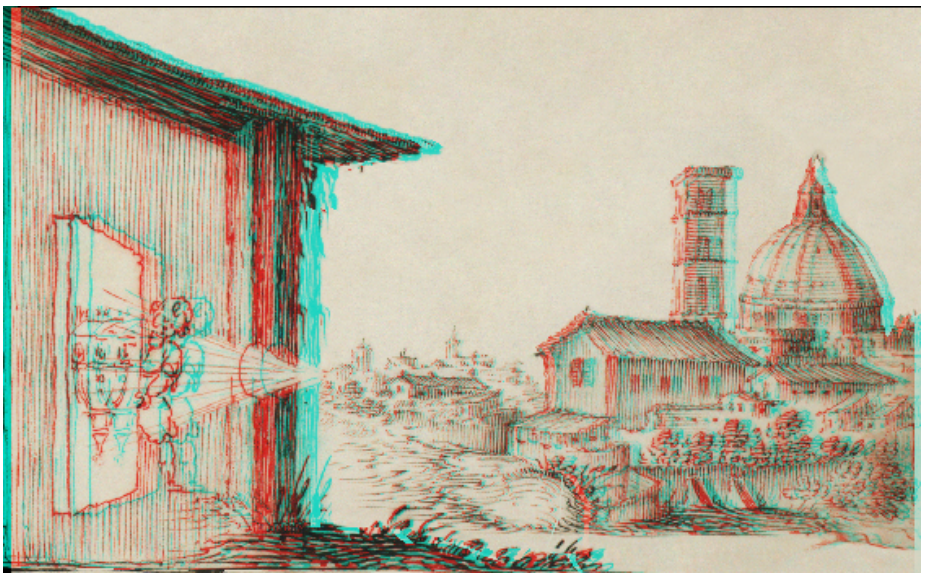
The result of this process is a Java program, that can be executed using Web Start, and is able to create a 3D scene with only a small effort of the user. For example, Figure 1.2c shows the anaglyphic rendering of an old illustration from a manuscript of military designs.

The remainder of this thesis first discusses some work that is related to the problems that need to be solved for this project. Next, the research issues and realization of the project are presented. This is followed by an evaluation of the results that are obtained. Finally, the conclusion and some optional future work is presented.

(a) Original.



(b) Extracted layers.



(c) Anaglyph.

Figure 1.2: Example input image, the extracted layers shown in third person perspective, and an anaglyphicly rendered image, source: http://en.wikipedia.org/wiki/File:Camera_obscura2.jpg.

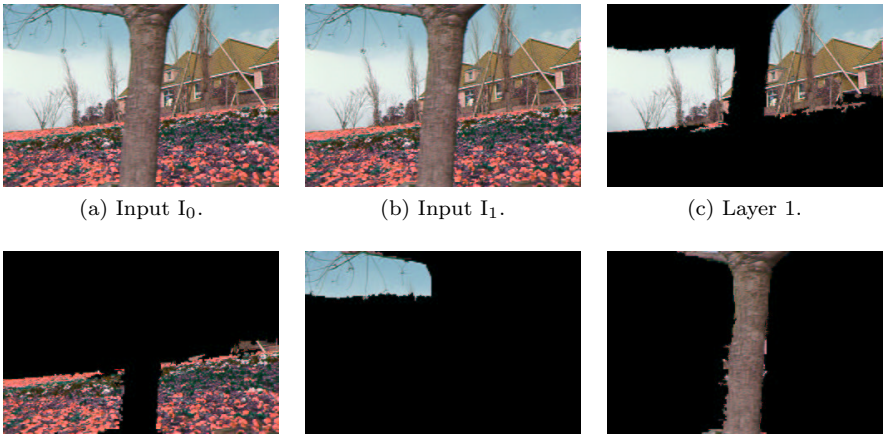
Chapter 2

Related Work

The conversion of 2D material to a 3D scene is possible through many different approaches. In this chapter a few of these approaches are mentioned by discussing some related work in the fields of layer extraction, lighting of objects, the extraction of 3D shapes from sketches, layering and active contours. All these fields propose several techniques to solve (parts of) the 2D-3D conversion problem.

2.1 Layer extraction

Most research on layer extraction focuses on multiple input images, for example 2 frames from a video. For example, Ke and Kanade [7] present a subspace approach to extract the layers from a sequence of images. In Figure 2.1 the results of this method are shown. Figure 2.1a and 2.1b show two frames from the input sequence, Figures 2.1c to 2.1f show the 4 automatic extracted layers. This kind of extraction requires accurate calibration; in the work of Uriol et al. [12]



(a) Input I_0. (b) Input I_1. (c) Layer 1.

(d) Layer 2. (e) Layer 3. (f) Layer 4.

Figure 2.1: Input images and results of automatic layer extraction (from [7]).

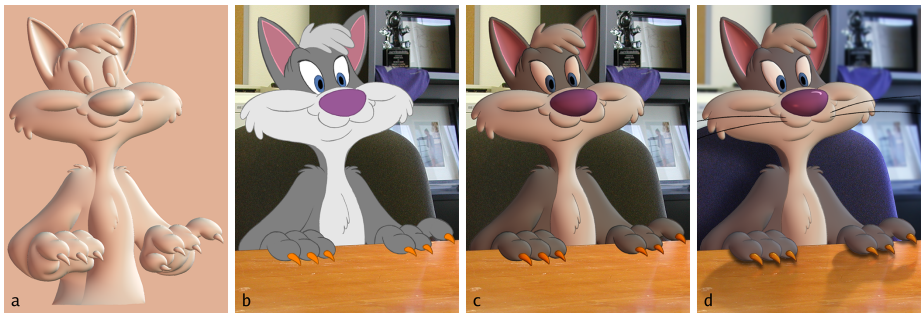


Figure 2.2: Results of Lumo: normal image illuminated by a diffuse spheremap, base palette cat, cat scaled by illumination, final cat (from [4]).

different calibration techniques and their applications are reviewed. This shows that the automatic extraction of layers from a sequence of images is possible.

These techniques are not usable in this project because the input illustrations that are used do most of the time not exist of multiple images from different camera positions, which is a requirement for the techniques described. Another issue that complicated the automatic extraction of layers if the fact that many old illustrations are drawn with the use of hatching (or cross-hatching) techniques. By making parallel lines, this technique can be used to create shading effects. All these individual lines make it harder to find the actual edge of an object.

The industry-standard way to extract layers from an illustration is the 'Pen tool' from Adobe Photoshop [2], which is a manual way to create straight or curved paths based on anchor points. While this is a good technique to extract layers, it is completely manual, and therefore slower than (semi) automatic layer extraction techniques. This project provides a simplified version of the 'Pen tool', that will be used to create the basic outlines of the layers.

2.2 Lighting of objects

3D information about objects is essential for lighting a scene. However, when real scenes are combined with cel animation, like in the movie "Space Jam", this is a problem for the flat cel animation part. Lumo is a method to approximate lighting on 2D cel animation, it derives approximate normal vectors based on the edges [4]. This method assumes that the surfaces are smoothly curved, and within this limitation the results achieved are visually appealing, as can be seen in Figure 2.2. However, the assumption that the surfaces are smoothly curved does not hold true for the illustration that are used in this project. While in some illustrations all surfaces might be smoothly curved, this is definitely not true for all illustrations.

Another downside of Lumo is that it creates depth in a single object rather than in the whole scene. This means that individual objects may appear to be 3D, while the remainder of the scene is flat. This might be a good addition for the application developed in this project, but the primary goal is to create depth in the whole scene. Therefore, this technique is not used in this project.

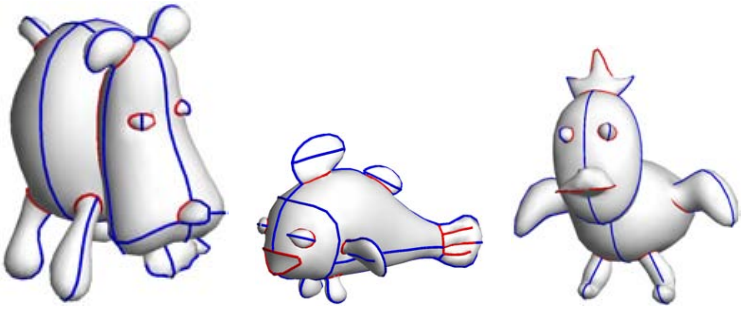


Figure 2.3: Some of the results obtained by FiberMesh (from [9]).

2.3 Extraction of 3D shapes from sketches

Both SmoothSketch [5] and FiberMesh [9] are systems for inferring 3D free-from shapes from contour-based sketches. System like this allow a user to create simple 3D shapes in a quick and simple manner. A set of output images from FiberMesh can be seen in Figure 2.3. These systems, however, all focus on creating a single 3D objects, not a whole 3D scene.

The input for systems like these only exists of simple contour-based sketches. For illustrations, however, we do not know what the contours of the objects are. This means that the mentioned systems are unable to deduct 3D shapes from the illustrations. Furthermore, when an illustration would be given as input to these systems, the whole illustration would be seen as a single object. This results in one big 3D output object, which is in contrast with the desired result: a scene with individual objects at different depths.

2.4 Layering

The most common way to achieve a 3D effect is by using Adobe After Effects [1]. This method is often combined with layer extraction using Adobe Photoshop (as described in Section 2.1). After the extraction of layers in Adobe Photoshop, the project is imported in Adobe After Effects, where each layer is placed into a 3D space. The positioning and scale of the individual layers can then be adjusted manually to adjust the perspective. A demonstration of this method is given in the tutorial written by Kramer [8], an example of the effects that can be achieved when the camera is moved can be seen in Figure 2.4. As can be observed, these 2 images still look similar; the Adobe After Effects approach works best for creating videos.

This approach has an obvious downside, all the steps that are required to create a 3D scene need to be performed by the user, it is completely manual. In consequence, this approach is time intensive and placing the layers in the correct perspective is not trivial. In fact, there is no guarantee that this manual perspective will be correct. The aim of this project is to provide a semi-automatic alternative to the Adobe After Effects approach, with an automatically correct perspective.

(a) Default view.

(b) Changed view, by moving the camera position.

Figure 2.4: An example from the tutorial of Kramer [8], showing the result of layering with Adobe After Effects.

2.5 Active contour model

The layer selection with a simplified pen tool, that is described in Section 2.1, works best for making rough outlines of the layers. These layers need to be enhanced with an with an automatic method. One method that is capable of this, is the active contour model. The principle of active contours, or snakes, is used to find edges of objects in an image. The technique dates back to 1988, when it was introduced by Kass et al. [6], and it used a lot in the field of biological imaging. The snake is a spline that attempts to minimize the energy that is associated with it by internal and external forces that that push and pull it towards the contours and edges of an image. If the position of the snake is $v(s) = (x(s), y(s))$, then the energy function of the snake can be written as

$$E^*_{snake} = \int_0^1 E_{snake}(\mathbf{v}(s))\, ds \tag{2.1}$$

$$= \int_0^1 E_{internal}(\mathbf{v}(s)) + E_{image}(\mathbf{v}(s)) + E_{con}(\mathbf{v}(s))\, ds \tag{2.2}$$

where $E_{internal}$ is the internal energy of the snake due to bending, E_{image} gives rise to the image forces and E_{con} gives rise to the external constraint forces.

Different methods have been proposed to improve the snake, for example, to make it more robust against noise. One of these proposals is to make the snake adaptive, so that the stiffness of the snake it adjusted during the process of finding the contour Andrey and Boudier [3]. This adaptive method is used in this project to help the user in making a layer selection.

Chapter 3

Concept & Realization

The development of this project brings several problems that need to be solved. In this chapter these problems and their solutions are discussed. Furthermore, important design decisions concerning the development of the applications are justified. First, the platform and language of choice are discussed, this is followed by an explanation of the techniques used for the extraction and layering. Next is an clarification of placing the layers into perspective, and finally the creation of an anaglyphicly rendered image will be discussed.

The program provides different views in its interface, that all serve a different purpose. The selection of the layers is done in a standard 2D view, where each layer can be highlighted. A third person based 3D view is used to adjust the depth of the layers, since it is easier to see the adjustments of depth in this camera position. Another two views are provided to watch the end results of the 3D scene: a normal first person based 3D view, and an anaglyphicly rendered first person based 3D view.

3.1 Platform & Language

The program resulting from the project should be able to run on as many platforms as possible with the least amount of effort. Therefore, the program is written in the Java programming language, which is architecture-independent. Java 3D, an application programming interface (API) for OpenGL and Direct3D, is used for the rendering of the 3D scenes. Java 3D tries to wrap the the graphics programming in an object orient way, it uses a scene graph to represent all objects. This should facilitate the programming of 3D graphics. Java 3D is currently a community source project, and not part of the default Java Development Kit or Java Runtime Environment, thus it needs to be downloaded and installed separately. This, however, is not necessary a downside, since Java 3D also provides a Java Web Start release, which means that programs depending on Java 3D can be executed from the web without any installation of Java 3D. The support of Web Start is one of the main reasons for choosing Java 3D as an graphics API. To create a Web Start application that needs 3D rendering, Java 3D needs to be added as an extension to the resource section of the web start launch file, an example of this is shown in Listing 3.1. The use of Web Start makes it possible to start the application directly on any computing using only

a web browser.

Listing 3.1: Sample Web Start file using Java 3D

```
<resources>
  <j2se version="1.6+"/>
  <jar href="Hatching_Layers.jar" main="true"/>
  <extension href="http://download.java.net/media/
      java3d/webstart/release/java3d-latest.jnlp"/>
</resources>
```

3.2 Extraction & Layering

The process of extraction and layers consists of multiple steps. The first step is the selection of the layers, after which they can be extracted. After this a separate background layer is added. Finally, a 3D scene can be build from these layers.

3.2.1 Guided selection of layers

The layers are selected by the user in a simple 2D environment using a selection method that is inspired by Adobe Photoshops 'Pen Tool'. A layer is created by defining a number of points that are connected by straight lines, once the first point is clicked on for the second time the layer is considered to be completed (see Figure 3.1). When a layer is completed it can be used in the 3D view.

Active contours (snakes) are used to automatically increase the precision of the selection of a layer that has been defined by a user, since the snakes are not able to find the edges completely automatically. Andrey and Boudier [3]

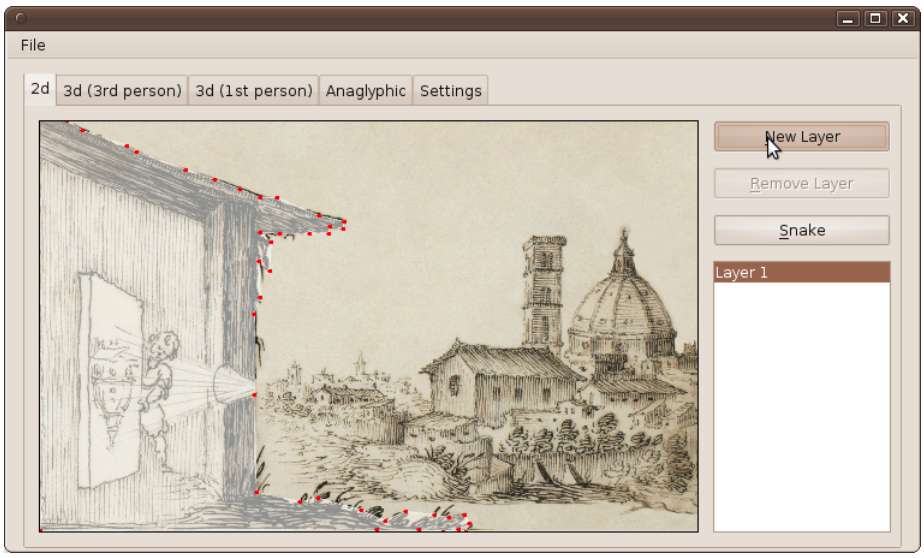


Figure 3.1: A layer is defined by a number of points.

Figure 3.2: A background layer with the holes of other layers.

proposes a adaptive snake method for the ImageJ library, which is used in this project. The method of Andrey and Boudier has originally been published as a standalone plugin for ImageJ, but it can also, without any modifications, be used as a snake library based on ImageJ.

However, using the snakes with hatching illustrations is not easy. Since these illustration consists of many parallel lines, the snakes have trouble to find the actual edges of the object in the scene. Unfortunately, this makes the use of the snakes quite limited in this application.

3.2.2 Background layer

There is always an extra layer that will be added next to the layers that are defined by the user, this is the background layer. The background layer consists of all sections that are not selected in any of the other layers. If the user would not select a single layer, the background layer covers the whole illustration. This background layer is added so that it is guaranteed that every part of the original illustration will always end up in the 3D view.

One of the problems that occurs after the extraction of layers are the "holes" that can appear in the background layer when an objects is moved forward, an example of such a background layer can be seen in Figure 3.2. These holes can be seen when the camera is not exactly in front of the scene. In order to get a background layer without any flaws, the background layer will have to be restored by inpainting the holes. While there are very complex and accurate inpainting techniques, this project uses a very simple inpainting method. Every pixel in the background layer that would be empty gets assigned the average color of all non-empty pixels from the background layer. The reason for this simple method is the scope of the project: going for a complex method would expand the project too much in comparison with the added results.

3.2.3 Layering in 3D

For the 3D views each layer is drawn on a single quad using a transparent texture. This makes the drawing of the layers very easy, since no complex shapes are needed. However, the downside of this approach is that the creation of the transparent textures is slow. These textures are drawn once when a 3D view is

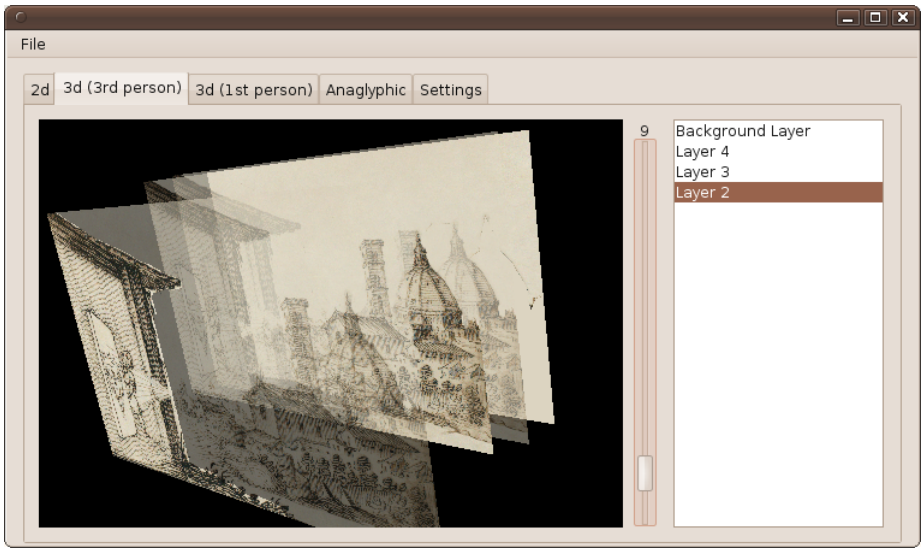


Figure 3.3: Adjusting the depth of a layer in third person view.

opened, and can not be adjusted while the user is in a 3D view.

In the third-person based 3D view it is possible to adjust the depth of the individual layers, an example of this can be seen in Figure 3.3. This is the only parameter that can be adjusted in the third-person view and this is a deliberate choice to keep the user interaction fairly simple. By limiting the freedom of the user in positioning the layers the program can guarantee a correct perspective.

3.3 Perspective

When the layers are selected and their depths are adjusted, they can be placed into a 3D space to be able to render a 3D view. The placement of the layers into a 3D scene brings a problem concerning the perspective, if the layers would only be moved on the z-axis, then the perspective would get distorted, since the original illustration is already in the correct perspective. This would make a layer appear to be on the wrong location, and the object in the layer would be to large, as can be seen in Figure 3.4. Solving this problem requires a number of steps:

1. Determine center point of the layer,
2. Move the center point of the layer towards the eye-position, based on the depth defined by the user, and
3. Scale layer down (or up) based on the depth.

When these steps are followed, the 2D view should look exactly the same as the starting position of the first-person 3D view. Only when the camera is adjusted, either by position or angle, the depth can be seen. If the first-person view is adjusted to use a true 3D view using, for example, an anaglyphic view, the

Figure 3.4: A 3D scene without any perspective adjustments.

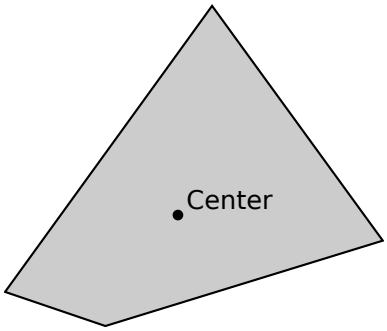


Figure 3.5: The average point of a polygon.

depth can always be seen.

The steps that are needed to get the correct perspective require some extra explanation, which are provided as follows.

3.3.1 Determining the center point

Since the selection of a layer only consists of straight lines between a number of points, it is actually an irregular polygon. In this project a very simple method is used to determine the center point of the layer. The center point is determined by calculating the average of all the points that define the layer (see Figure 3.5), which is a fast and plain method to determine a center point.

While this method does not give the true mathematical centroid of a polygon, there are two reasons for using this method instead of the normal method for calculating the centroid of a irregular polygon. First, the normal method assumes that the polygon is non-self-intersecting. While this might be true for most layers that are selected, there is no guarantee that this holds true for all layers. Second, the use of the average-method gives results that work really well with the positioning of layers.

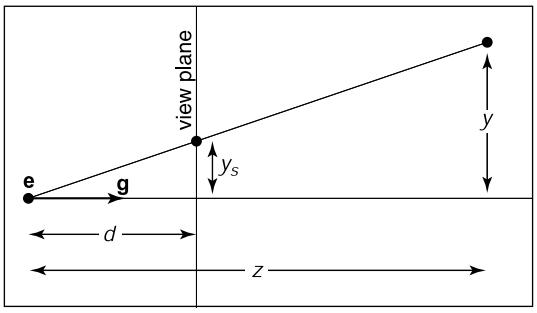


Figure 3.6: The viewer's eye is at **e** and the gaze direction **g**. The new depth of a layer is **d**, and original depth is **z** (from [11, p. 166]).

3.3.2 Move the center point toward the eye-positions

After the center point is determined, the layer can be placed at the right position in the 3D space. If a layer would only be moved on the z-axis, it would appear to the user that the layer has moved to a different position in comparison with the original illustration. Therefore, the layer needs to be moved towards the eye-position of the viewer. The background-layer is chosen as the default layer, all calculations are done with respect to the background layer. The center point of the layer is moved on the line between the eye-position of the viewer and the center point of the layer on the background layer (Figure 3.6).

3.3.3 Scale the layer down

When a layer has been moved towards the viewer it will appear to be larger than it should. This is because the original illustration is already in the right perspective, objects that are closer are drawn larger. If these objects are actually moved closer they get oversized, and therefore should be scaled down to perspective. When a layer is closer to the viewer than the background layer they should be scaled down, and when a layer is further away it should be scaled up, as can be seen in Equation 3.1:

$$scale = \frac{ld + cd}{bld + cd} \tag{3.1}$$

where ld is the depth of the layer, cd is the depth of the camera and bld is the depth of the background layer.

While the use of this scaling results in perfect perspective, this might not be what everyone wants. Slightly blowing up the layers can give a nice effect, it allows slight perspective changing without introducing any holes. Therefore, a parameter is introduced that allows the user to set a scaling factor. When this factor is exactly 1 it results in correct scaling, a factor of 0 would result in not scaling the layer at all. This gives Equation 3.2:

$$scale = sf \times \frac{ld + cd}{bld + cd} + 1.0 - sf \tag{3.2}$$

where sf is the scaling factor, that is defined by the user.

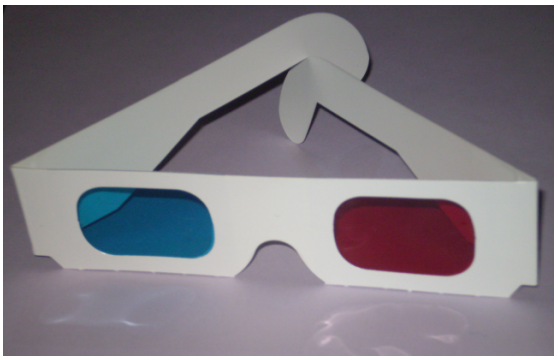


Figure 3.7: Anaglyph glasses, source: http://en.wikipedia.org/wiki/File:Anaglyph_glasses.png

3.4 Anaglyphic rendering

In this project, an anaglyphic render of the 3D scene is used to provide a stereoscopic view. The anaglyph consists of two superimposed renders of the scene, with a slightly offset eye position. These two renders are filtered through contrasting colors, one using a red filter and the other using a cyan filter. Anaglyphic renders can be viewed using anaglyph glasses (Figure 3.7).

For the implementation of the anaglyphicly rendered view, the "Anaglyph Canvas3D" library [10] is used. This library provides an anaglyph extension, `AnaglyphCanvas3D`, to the Java `Canvas3D` class. This class, by default, acts as a normal `Canvas3D`, but can also render an anaglyph based on the same data. To make this anaglyphic render two calls are needed: one to set the eye-distance and another set the colors of the filters that are used. This simplicity makes the "Anaglyph Canvas3D" library an ideal choice.

3.5 Summary

With the implementations that are mentioned in this chapter, the user is able to start the Java application using Web Start. An illustration can be opened in this application, and a number of layers can be selected. These layers can be enhanced with the use of snakes, however, this does not always give the desired results. The selected layers are extracted and placed into a 3D scene, where the user can change the depth of the individual layers. The scaling and positioning of the layers is automatically adjusted, so that the perspective matches the original illustration. Finally, a anaglyphicly rendered 3D scene is used to let the user actually perceive the depth that has been created.

Chapter 4

Evaluation & Results

In this chapter some input and output images are presented, showing the results of this project. The steps that are needed to achieve these results are explained and evaluated.

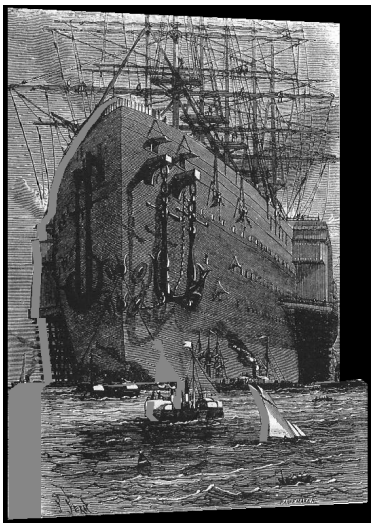
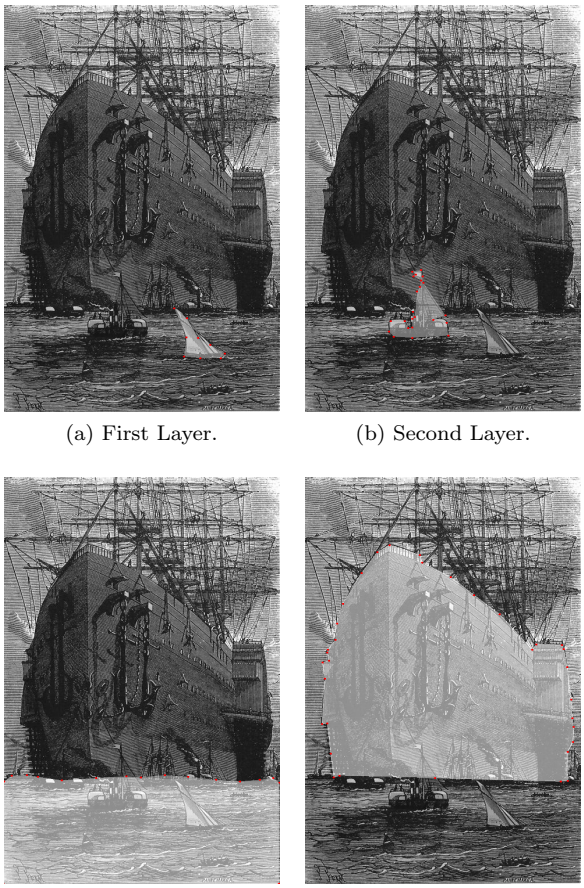


Figure 4.1: A rendered 3D output of the selected layers in Figure 4.2, the camera is moved slightly so that the depth can be seen.

Four different layers have been selected for the first example, which can be seen in Figures 4.2a to 4.2d. Creating rough selections can be done very quickly, especially when most edges are straight and long. The first and second layer (Figures 4.2a and 4.2b) contain the two ships in the front of the scene. Selecting both boats in a different layer has two reasons: they can be placed at different depths, and it is not possible to select one layer consisting of multiple polygons. The third layer (Figure 4.2c) consists of the sea, and the fourth layer (Figure 4.2d) contains the hull of the ship in the back. The masts, sails and ropes of the ship have not been added to any layer, and therefore will be added to the



(a) First Layer. (b) Second Layer.

(c) Third Layer. (d) Fourth Layer.

Figure 4.2: The four different layers that have been selected, source of original input image: http://commons.wikimedia.org/wiki/File:Verne-M%C4%9Bsto2.jpg, from "Une Ville flottante" by Jules Verne.

background layer. Selecting complex shapes like the masts is currently really difficult. The selected layers have not been enhanced with the snakes, using the snakes results in worse selections with each of these four layers.

After selecting these layers, the depths of the layers are adjusted. A 3D version of this scene can be seen in Figure 4.1a, where the camera is slightly moved from the first-person based view. A large anaglyphicly rendered version from the default first-person view can be seen in Figure 4.3, which clearly shows that even with only four simple layers a nice perception of depth can be given. The creation of this example only took 10 minutes.

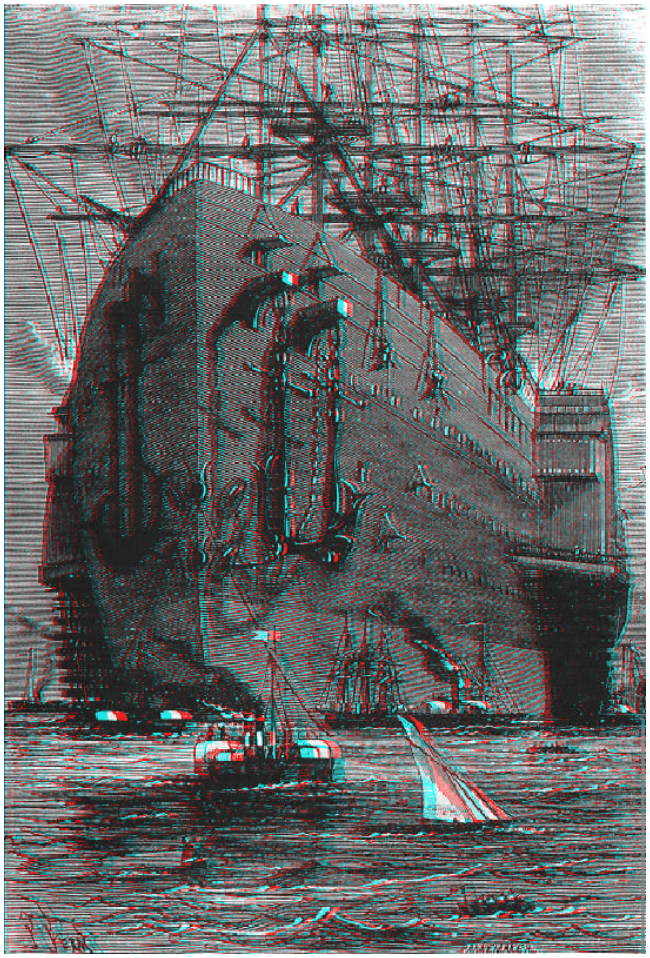


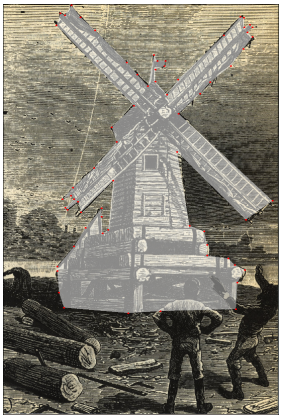
Figure 4.3: A front-view anaglyphicly rendered output image from the layers in Figure 4.2.

Another example can be seen in Figures 4.4 and 4.5. This example is created by a person who has no experience with image editing software, and was created in 15 minutes. Before creating this example, the person tried the program for about 10 minutes. Five different layers have been selected (Figure 4.4). The selection of the two men in front (Figures 4.4a and 4.4b) took the most time, selecting smooth shapes with only straight lines requires adding many points to the layer. The selection of the windmill and the logs (Figures 4.4c to 4.4e) took less time.

Some adjustments were made to the depths of the layers in the third person perspective, for example, the two men in the front were given the same depth. An output image from first-person perspective, with a rotated camera, can be seen in Figure 4.5a; the depths can clearly be seen in this image. Furthermore, an anaglyphicly rendered output image has been created, which can be seen in Figure 4.5b.

(a) First Layer.

(b) Second Layer.

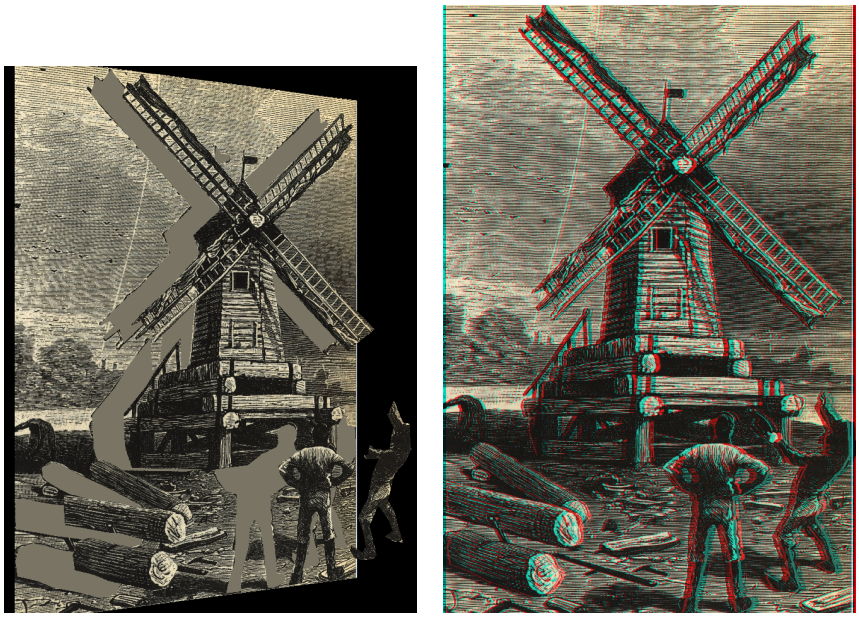


(c) Third Layer.

(d) Fourth Layer.

(e) Fifth Layer.

Figure 4.4: Layers selected by a person who has no experience using image editing software.



(a) Perspective. (b) Anaglyph.

Figure 4.5: Rotated first-person and anaglyphicly rendered results based on the selected layers from Figure 4.4.

The selection of the layers is a considerably harder with complex shapes, for example, a group of people. The snakes can sometimes assist in these situations, but with hatching illustrations they are most of the time unable to find the edges of objects. The current selection tool also is a limitation when selecting smooth surfaces or round objects, since the layers can only consist of straight lines. This could be solved by added the possibility to create curved lines, for example using Bézier curves.

The results of the simple inpainting method are demonstrated in Figure 4.6. For this example all objects in the original illustration have been selected in layers (Figure 4.6a). The background layer which shows the inpainting method is shown in Figure 4.6b, none of the selected layers are visible in this figure. The example image has a monochromatic background colour, which lends itself to the use of this simple inpainting method.

(a) The original file, with highlighted layers

(b) Background layer, with inpainting

Figure 4.6: The inpainting method demonstrated, all missing pixels in the background layer get the average color of all other pixels.

Chapter 5

Conclusion & Future Work

The goal of this project was the semi-automatic extraction of 3D views from traditional illustrations. When the resulting images are observed, it is possible to conclude that this extraction is able to produce good results. One of the major advantages of this project, when compared to the After Effects method, is the guarantee that the perspective will always be correct. This advantage is extra useful when someone wants to create a realistic 3D rendered scene from an old illustration, for example of an historical scene. This forced perspective, on the other hand, is also one of the limitations of this project. If one would like to move objects around, or create an other non-realistic effect, then this project would not be an ideal tool.

The basic inpainting method that is used in this project provides the best results in the default camera position in an anaglyphic render. In this case you only see a marginal part, with each eye on an other side, behind the layer. The fact that the inpainting is not very accurate gets compensated by the fact that only a very small inpainting part can be seen. When a normal 3D view is rendered from a camera position that allows a clear view behind the layers, then the inpainting method achieves poor results. This is even more emphasized if the background layer consists of a complex multi colored texture.

Unfortunately, the snakes are not usable to reliably enhance the selection of the layers. Therefore, the layer selection is mostly manual instead of guided, as the intention of this project was. With all the parallel lines in hatching based illustrations, snakes are not able to find accurate edges. At this moment, no clear solution for the problem exists. However, a possible solution may involve: preprocessing the illustration so that the snakes work better, improving the snakes function to work better with hatching, or using an other technique for edge detection.

In the current implementation all layers are flat, each and every layer is a simple quad with a transparent texture placed in a 3D scene. While the overall effect created is 3D, all objects are still 2D. This could be enhanced by adding automatic bump mapping to the layers or approximating the lightning of the objects with a technique as described by Johnston [4]. This could really improve illustrations that contain large objects with a lot of depth, for example, the ship in Figure 4.3.

Possible application areas for this project include, but are not limited to: historic documentaries, interactive illustration viewing, and educational programs.

A 3D view of a old illustration is more interesting to watch, and may even give new insights to historic events that are depicted in a scene. A rendered movie of a 3D scene, where the camera movies around, might be interesting for a documentary; it would be more attractive than a static illustration.

Overall, this project shows that it is possible to interactively extract layers from a traditional illustration to create a 3D view. In its current form, the program created is able to quickly generate a anaglyphic render from a 2D illustration. However, the selection of the layers is still somewhat coarse and the inpainting method is not optimal. For integrating 3D views of 2D illustrations into movies, one is still better of with a traditional Adobe After Effects based approach. For the creation of quick 3D (anaglyphic) renders of illustrations, this project is ideal.

Bibliography

[1] Adobe Systems. Adobe After Effects CS3. Software Tool, February 2008.

[2] Adobe Systems. Adobe Photoshop CS3. Software Tool, April 2007.

[3] P. Andrey and T. Boudier. Adaptive active contours (snakes) for the segmentation of complex structures in biological images. In *ImageJ Conference*, 2006.

[4] Scott F. Johnston. Lumo: illumination for cel animation. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 45–52, New York, USA, 2002. ACM Press.

[5] Olga A. Karpenko and John F. Hughes. SmoothSketch: 3D free-form shapes from complex sketches. *ACM Transactions on Graphics*, 25(3): 589–598, July 2006.

[6] M. Kass, A. Witkin, and D. Terzopolous. Snakes: Active contour models. *Internation Journal of Computer Vision*, 1(4):321–331, 1988.

[7] Qifa Ke and Takeo Kanade. A subspace approach to layer extraction. In *Conference on Computer Vision and Pattern Recognition (CVPR'01)*, volume 1, pages 255–262. IEEE Computer Society, December 2001.

[8] Andrew Kramer. Virtual 3d photos, September 2007. URL http://www.videocopilot.net/tutorials/virtual_3d_photos/. Web page, accessed 2-June-2010.

[9] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Fibermesh: designing freeform surfaces with 3D curves. *ACM Trans. Graph*, 26(3):41–48, 2007. URL http://doi.acm.org/10.1145/1276377.1276429.

[10] Hendrik Rohn. Anaglyph canvas3d. Software Tool, http://anaglyphcanvas3.sourceforge.net/.

[11] Peter Shirley, Michael Ashikhmin, Michael Gleicher, Stephen Marschner, Erik Reinhard, Kelvin Sung, William Thompson, and Peter Willemsen. *Fundamentals of Computer Graphics, Second Ed.* A. K. Peters, Ltd., Natick, MA, USA, 2005. ISBN 1568812698.

[12] Maria-Cruz Villa Uriol, Gautam Chaudhary, Falko Kuester, Tara Hutchinson, and Nader Bagherzadeh. Extracting 3d from 2d: selection basis for camera calibration. In *7th IASTED International Conference on Computer Graphics and Imaging (CGIM 2004)*, pages 315–321. ACTA Press, 2004.