

TDK dolgozat

Tövissy Judit

2014

Gábor Dénes Főiskola

Informatikai Intézet

Tövisy Judit

II. évfolyam

Témavezető: Dr. Kopácsi Sándor, főiskolai tanár

Automatikus háromdimenziós képkonverzió weblapokon

Automated 3D Image Conversion on the Web

by
Judit Tövisy

Table of Contents

Table of Contents	1
1. Introduction	3
1.1. Objective of the work	3
1.2. Related Project	3
2. Technical background	4
2.1. 3D Display techniques	4
2.1.1. Anaglyph	5
2.1.2. Side-By-Side	5
2.2. 3D Filtering Hardware	6
2.2.1. Polarised Lenses	6
2.2.2. Shutter Glass	7
2.2.3. Autostereoscopy	7
3. Theoretical background	8
3.1. Qualitative Depth Map	9
3.1.1. Image Segmentation	9
3.1.2. Region-based Qualitative Depth Map	12
3.1.3. Absolute Brightness	13
3.2. Geometric Depth Map	13
3.2.1. Perspective, Vanishing Points	13
3.2.2. Depth Gradient	15
3.3. Focus Map	17
3.4. Stereoscopic Image Pairs	18
4. Implementation	20
4.1. Objective	20
4.2. Limitations	21
4.3. Specification	21
4.4. Conversion Steps	22
4.4.1. Qualitative Depth Map	22

4.4.2.	Focus Map	24
4.4.3.	Depth-Focus Map	25
4.4.4.	Stereoscopic Conversion from Depth-Focus Map.....	29
4.5.	Further Results	37
4.5.1.	Brightly Lit Images	37
4.5.2.	Well-Focused Images	39
4.6.	Further Plans.....	41
5.	Summary	41
6.	Bibliography	42
7.	Table of Figures.....	43

1. Introduction

1.1. Objective of the work

As the development of hardware with the capability of three-dimensional display advances, it is a general observation that content utilizing this technology is not yet widely and openly available. Therefore, the need arises to make the new rendering opportunities more accessible by extending the worldwide web's range to include websites that can be viewed in 3D.

It is assumed that were three-dimensional content more easily and widely available, the demand for hardware capable of displaying 3D would be likely to produce significant increase, as well.

The goal of this project is to complement recently completed and ongoing studies in the field of stereoscopic rendering possibilities and find unique solutions for the automated 3D conversion of still, 2D images on the web, without any prior knowledge of the spatial position of the objects on the picture. It is another goal to further the field's advance by developing the prototype of a particular application that is capable of converting single-view images into stereoscopic 3D. The goal with this is to take images especially from the web and convert them into 3D formats with the use of image analysis methods.

The development process and this study encompasses previous works of stereoscopic image pair generation, including depth map generation and focus map extraction of image files.

1.2. Related Project

The opportunity for this work was created during a research and development project [1] at the Institute for Computer Science and Control of the Hungarian Academy of Sciences. That project envisioned a more immersive web experience and delivered software that could automatically convert a webpage into widely-used stereoscopic formats, which are going to be discussed in chapter [3D Display techniques](#), in [Chapter 2.1](#). One of the main results of the 3DWeb project can be seen on the web on Institute for Computer Science and Control's website (www.sztaki.hu/3d), as the whole website has been converted automatically and displayed in 3D. We tested different 3D display techniques, such as anaglyph, side-by-side and auto stereogram. These results can be seen on the 3DWeb project's

website, www.3dweb.hu. Test results showed that anaglyph and side-by-side technologies were more appropriate for displaying web content, therefore these were the technologies I based my work on.

While the 3DWeb project was a success, automatic conversion of the images was not yet developed and the images on the website require manual conversion, which leads to the issue that we have to keep up-to-date with the daily state of the website and manually create 3D versions of newly uploaded images, even though everything else is entirely automatic. It is clear that automatic conversion of images is greatly needed.

3D conversion of images is greatly different from that of other web content. While most elements can be described and analysed according to semantic characteristics, image-based content requires a different approach and this gave rise to the opportunity to develop a method that can take an image and produce a stereoscopic image pair that can be displayed in 3D with the other elements of the website. The resulting implementation is to be built into the conversion process of our already existing software for website conversion.

2. Technical background

2.1. 3D Display techniques

The theoretical basis for 3D display technology is stereovision, which is based on the mechanism of binocular vision. Two cameras simulate the way in which human vision and depth perception works. The brain receives two images, one with each eye, and analyses the differences between the images to determine depth information. Three-dimensional display technology relies on two images, taken from two slightly different points (6-7 cm away from each other) of view. The images are then both displayed and filtered with different techniques for each eye [2].

To recreate three-dimensional vision, we are in need of such images. In possession of a 3D scene, two such virtual cameras can be created without a problem, but cases where little information is available pose a nontrivial challenge.

Three-dimensional display technologies have two categories, which are entwined. Display Techniques are solutions as to how to display the two images and filtering techniques enable each eye to see only one image, usually by blocking the eye from seeing the other image.

2.1.1. Anaglyph

Anaglyph is a passive display and filtering technique that utilises a pair of coloured lenses to recreate three-dimensional content. Glasses use complementary colours, most commonly red-cyan or red-blue. Yellow-blue glasses also exist but are quite rare. Anaglyph display uses the same display space for both left and right images, and is there-



*Figure 1: Image encoded using Anaglyph 3D
(taken in Girona, Catalonia)*

fore capable of storing stereoscopic information in a single image. This characteristic makes anaglyph 3D the most flexible and wide-spread of current 3D techniques because it works on any screen and only requires a pair of tinted glasses, which are also quite cheap. Anaglyph technique displays the stereoscopic images by using colour layers that correspond to the respective colours of the glasses. The coloured lenses then filter their individual colours, so distributing information between the two eyes. For anaglyph 3D to work, the colour coding of the image needs to match the respective colours of the lenses.

While anaglyph encoding has its advantages, it also comes with a downside that the colour complexity of the original image can be affected and the initial colours of the image can get distorted. Certain combinations of colours, including shades of red and green, or red and blue, however, give authentic results. Figure 1 shows an example of an image displayed using anaglyph 3D and is able to be viewed in 3D on an appropriately calibrated screen, with suitable glasses.

2.1.2. Side-By-Side

While anaglyph 3D condenses data to a single image, Side-By-Side is a display technique that passes two separate images to be displayed according to the filtering method. Colours suffer no distortion, however, special or additional hardware is required for proper display and filtering. Figure 2 shows a stereoscopic image pair encoded with Side-By-Side technology.



Figure 2: Side-By-Side image (taken in Barcelona, Catalonia)

2.2. 3D Filtering Hardware

The objective of 3D hardware discussed below is to accomplish the accurate filtering of 3D information to each eye of the user. Their objective being the same, their methods performing it differ greatly. They can be characterized by way of these methods as either active or passive, either with or without glasses.

2.2.1. Polarised Lenses

Separation of the two images can be achieved using polarised glasses. This filtering technology requires a special kind of display, but the glasses themselves are passive, meaning they contain no electronic parts. By different polarisation of light it is possible to display two images at the same time, projected at the same space. The fundamental idea is to project two images with two different polarizations of light. Polarisation can be either linear, where the polarisation of the two images are perpendicular to each other; or circular, where the direction of polarisation is different, as is demonstrated in Figure 3. Filtering is done by the lenses of the glasses as they are equipped with different polarising film, therefore blocking out information from the other image. 3D movie theatres tend to use polarisation technology when showing 3D movies.

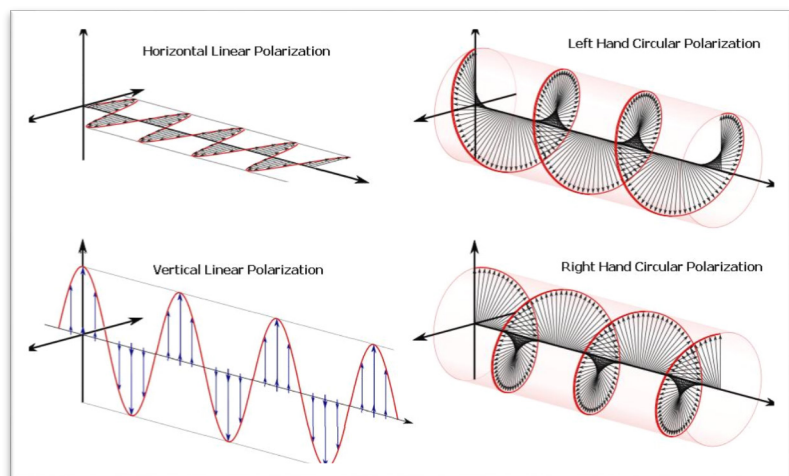


Figure 3: Different polarisations of light

2.2.2. Shutter Glass

Shutter Glass technology uses so-called 'active glasses' to filter 3D information. The technology can be called active because it involves the glasses in an active manner, not just as a passive medium. Left and right images appear on the screen in quick succession and the viewer's glasses, which are synchronised to the screen, block each eye while the other's image is being displayed, as is illustrated in Figure 4. In order to do this, the glasses contain liquid

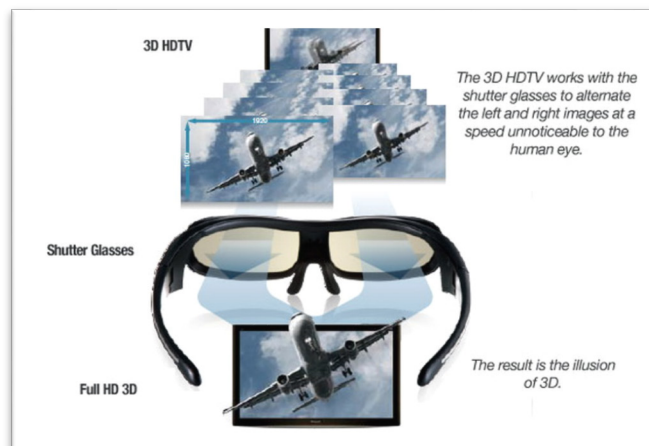


Figure 4: Shutter Glass Technology

crystal technology which can alternate between dark and see-through phases quickly enough (at least 60 times per second) to allow the process to remain unnoticed by the viewer, therefore keeping the 3D experience intact. Synchronisation between screen and glasses is of utmost importance considering this technology. A direct result of Shutter Glass Technology would be a significant drop in frame rate as each eye only sees half the number of images that are displayed, therefore the display needs to have a doubled refresh rate (at least 120 images per second) compared to standard displays.

2.2.3. Autostereoscopy

It is possible to achieve 3D view without the use of 3D glasses. Autostereoscopic display units attain 3D view using the same principles of stereoscopy as previously discussed methods. The innovation is present in the way these displays achieve filtering. Separation of the stereoscopic image pair is built into the screen itself, therefore requiring no additional equipment. Different manufacturers use different technologies to separate the two images, however, these technologies do

share certain fundamental principles. The most well-known of these is the Parallax Barrier technology, which uses a precision layer on top of an LCD screen which blocks certain areas of the screen when seen from certain angles. The precision layer is engineered in such a way that, while watching the same screen, both eyes see a different set of pixels, thus allowing the separation of stereoscopic image pairs.

Another technology based on the same principle uses lenses instead of barriers to achieve a similar result. Figure 5 demonstrates both technologies.

There is, however, an unfortunate disadvantage to this technology. By its inherent properties, the 3D effect can only be seen correctly from a limited range of viewing angles. Still, the possibilities of this technology do not end here. Autostereoscopic displays present the separation of two images on a single display, which, when constructed in certain ways, allow the screen to display two separate images or content streams to different viewers watching from different angles. To pose a practical example, when such a display unit is used inside a car, it could allow the driver to see a navigation software while the passenger could watch something entirely different.

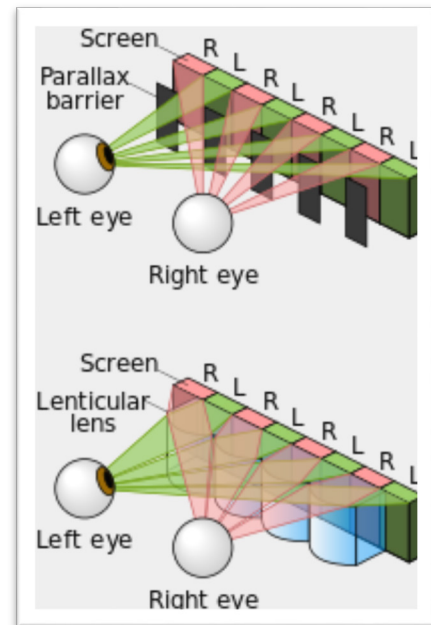


Figure 5: Illustration of Autostereoscopic technologies

3. Theoretical background

In order to display an image in 3D, depth information is needed for each of its pixels. However, certain optimisation steps allow us to reduce the complexity of the required computation for most pixels. The details of these steps will be discussed in [Chapter 4.4](#). We need to know how far the pixels should be from the viewer. For this purpose, relative depth can be represented by a floating point value between 0 and 1, which can be described as an exact shade of the colour grey. The Depth Map that is constructed from the depth value of each pixel and is a crucial ingredient to creating 3D images. As points of the image already have x and y coordinates that tell their place in two dimensions as indices of the pixels, the depth value is the crucial piece of information that supplies the coordinate z in a three-dimensional Euclidean space.

The represented depth values are defined as the z coordinates in virtual space V where $z=0$ is the further clipping plane that can be described, and $z=1$ is the nearer clipping plane for the virtual camera that describes the position of the viewer. Values $0 \leq z \leq 1$ are defined as the translation value of the pixel from the further clipping plane.

Being in possession of the Depth Map of an image makes it possible to define each of the pixels in an image as a vertex with coordinates $[x, y, z]$ and a colour value $[R, G, B]$. Once this space is populated with all vertices, virtual cameras can be simulated to render the image space from the point of view of each eye. In order to determine depth values, images are analysed from three different aspects: colour information, geometric information, and focus information, all of which will be discussed in this section. The main objective of the developed prototype is to produce a Depth Map that is a sufficiently accurate estimate of the space defined by the input image. The secondary objective is to use the created Depth Map to render the stereoscopic image pair that is based on the image space and the original position of the camera. For these purposes, research was conducted in exploration of the following theoretical fields, some of which came to be implemented during the development of the prototype.

3.1. Qualitative Depth Map

A Qualitative Depth Map builds fundamentally upon the colour information found in an image.

3.1.1. Image Segmentation

Image segmentation is the process of partitioning a digitalised image into multiple segments. These sets of pixels are also known as *superpixels*. The goal of segmentation is to simplify the representation of an image into something that is easier to analyse [3]. Image segmentation can be used to locate objects and their boundaries in an image. To put it more precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics. The basis of the segmentation is the colour value of the pixels in the image, therefore we can call such a method a colour-based segmentation. The result of the segmentation process is a set of segments that we can call regions and are chromatically homogenous. The goal of the process is identifying these regions that collectively cover the entire image. Each of the pixels in a region is similar with respect to some characteristic or computed

property, such as colour, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristics [3].

There are a number of methods and approaches to segmentation, such as *thresholding*, which turns a grayscale image into a binary image. To be in possession of a grayscale image, the original image has to, of course, be converted into grayscale. However elegant and generally useful this method is, its resulting image is oversimplified for the purposes of depth map retrieval, and therefore I took another approach.

3.1.1.1. Clustering by Mean Shift

A more accurate, but still simple enough result comes from clustering methods.

The common basis for these methods is called a Mean Shift Algorithm [4]. It seeks the point of highest density of a data distribution. Basically, the algorithm replaces each pixel's colour value with that of the average of the pixels in a range- r neighbourhood and whose value is within a distance d .

Mean Shift values are reached in the following general manner. Consider a set S of n data points x_i in Euclidean space X with dimension d . Define a kernel function K to indicate how much x contributes to the estimation of the mean [4]. This kernel function has to have the following properties [5]:

- $K \geq 0$
 - $K(x) \geq K(y)$ if $x < y$
 - K is piecewise continuous and
- $$(1) \quad \int_0^\infty K(x) dx < \infty$$

The usual choices for K are the Epanechnikov kernel function $\phi(x)$, where:

$$(2) \quad \phi(x) = \begin{cases} \frac{3}{4}(1 - x^2) & \text{if } |x| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

and the Gaussian kernel function, defined in d dimensions where σ is a bandwidth or dispersion parameter:

$$(3) \quad \phi(\vec{x}) = e^{-\frac{|\vec{x}|^2}{2\sigma^2}}$$

A kernel density estimation is given in the form

$$(4) \quad \hat{f}(x) = \frac{1}{n\sigma^d} \sum_{i=1}^n K\left(\frac{x-x_i}{\sigma}\right)$$

Utilising the Gradient Ascent nature of Mean Shift, the generic Gradient Ascent formula,

$$(5) \quad x_1 = x_0 + \eta f'(x_0)$$

When used with the above kernel density estimator, gives

$$(6) \quad \nabla \hat{f}(x) = \frac{1}{n\sigma^d} \sum_{i=1}^n K\left(\frac{x-x_i}{\sigma}\right)$$

and set to 0, gives the following equation:

$$(7) \quad \sum_{i=1}^n K'\left(\frac{x-x_i}{\sigma}\right) \vec{x} = \sum_{i=1}^n K'\left(\frac{x-x_i}{\sigma}\right) \vec{x}_i$$

Which leads to a sample mean $\mathbf{m}$:

$$(8) \quad m(\vec{x}) = \frac{\sum_{i=1}^n K'\left(\frac{x-x_i}{\sigma}\right) \vec{x}_i}{\sum_{i=1}^n K'\left(\frac{x-x_i}{\sigma}\right)}$$

Providing that $g(x) = -K(x)$, Mean Shift can be defined as the difference $\mathbf{m}(\mathbf{x}) - \mathbf{x}$ as follows:

$$(9) \quad M(x) = \frac{\sum_{i=1}^n g\left(\frac{x-x_i}{\sigma}\right) x_i}{\sum_{i=1}^n g\left(\frac{x-x_i}{\sigma}\right)} - x$$

In each iteration, $\mathbf{M}(\mathbf{x}) \rightarrow \mathbf{x}$. The algorithm stops when $\mathbf{M}(\mathbf{x}) = \mathbf{x}$.

For each pixel, Mean Shift associates its value with the nearby peak of the surrounding pixels' probability density function. Mean Shift defines a window around each pixel and computes the mean for the pixel according to a defined kernel. Then it shifts the centre of the window to that average pixel and repeats the algorithm till it converges. After each iteration, we can consider that the window shifts to a denser region of the image.

Algorithmically speaking, Mean Shift works through the following steps:

1. Place a window around each pixel. The size of this window is crucial to the result of Mean Shift and is referred to as the spatial term or spatial scale of Mean Shift. The actual value needs to remain in a certain Goldilocks zone that is unique to an application.
2. Compute the mean of data within this window.
3. Shift the window to the mean and repeat until convergence.

It is essential to mention that the Mean Shift algorithm is not limited to image processing, but is a general tool of data mining, as well.

3.1.2. Region-based Qualitative Depth Map

Studies [6] and [3] use the result of clustering to define regions in the image. A region is defined as an area of a data cluster in the image. A region, representing a group of pixels is likely to define an object in the image, and as such, can be treated as one. The objective of the Region-Based Qualitative Depth Map is to establish a certain kind of hierarchy between the regions based on their supposed distance from the camera. To determine such a hierarchy, studies [3] and [6] are using a heuristically defined set of complex guidelines to classify each region. This method is indirectly aimed at determining depth, but its foremost function is to classify the type of object the region represents. The guidelines analyse the colour and intensity values of the region and assign one of the previously defined labels to it. Such labels contain "Sky", "Furthest Mountain", "Far Mountain", "Near Mountain", "Land", "Other". The predefined system has depth relationships associated with all of the labels. By assigning such a label to a region, an estimated Depth Map of the image can be rendered.

To be assigned a label, the image is converted from RGB space into HSI (Hue/Saturation/Intensity) space so intensity values may also be tested. The colour and intensity values are then processed by a system of criteria. For example, according to [3], the intensity and colour values of a region have to fulfil the following criteria in order for the region to be classified as "Sky":

```
B(x,y) >= G(x,y)+30 AND G(x,y) >= R(x,y)
B(x,y) >= G(x,y) AND G(x,y) >= R(x,y)
B(x,y) >= R(x,y) AND R(x,y) >= G(x,y)
I(x,y) >= 200 AND |B(x,y)-G(x,y)| <= 30 AND |R(x,y)-
G(x,y)| <= 30
```

Since the sky is the furthest possible region, it can be assigned a depth value that is close or equal to zero. Other depth values are based on the study's model of region hierarchy, following the trivial thought that a region determined to be a "Far Mountain" is likely to be closer than a region that is "Sky" but further than a region labelled "Near Mountain". Classification makes it possible to not only determine depth, but to tell if an image depicts an indoor or an outdoor location and opens up even further possibilities.

3.1.3. Absolute Brightness

While the previously discussed method of creating a region-based qualitative depth map is very useful for further image analysis, it results in an abundance of information about the image, most of which is not required for depth map estimation. Furthermore, such a system of criteria is limited to the span of the heuristic guidelines that it is constructed of; therefore any accuracy issue that exists in the system will exist in the resulting data, as well. An optimum for maximising accuracy while minimising system complexity is needed, too.

I propose that a simpler method is preferable in those cases where only a depth map of the image is needed. As it is discussed in the Implementation section of this study, the prototype of this project builds upon the Absolute Brightness of regions, a value that is represented utilising the Euclidean Distance of the Mean Shifted value from the colour white, as the following formula shows:

$$(10) \quad A(x) = \sqrt{(255 - R(x))^2 + (255 - B(x))^2 + (255 - G(x))^2}$$

Values $R(x)$, $B(x)$, and $G(x)$ represent the region's colour channels. Results from this method are discussed in the Implementation section.

3.2. Geometric Depth Map

Another approach to gather spatial information from an image is to examine it based on its perspective attributes [6] and extrapolate the geometrical attributes of the environment depicted on the image. The depth information that is gathered from such sources, structured into a map of the original image makes up the geometric depth map of an image. The following paragraphs contain details as to how it is possible to achieve such a depth map.

3.2.1. Perspective, Vanishing Points

A vanishing point is a virtual tool that helps the human eye to detect depth from the perspective of an image, hence it plays an important role in perspective representation. It being a bijective function of 3D space, it is possible to use the concept in a reverse manner to calculate perspective from an image.

In graphical perspective, a vanishing point is a point in the picture plane p that is determined by a line in space. Given the oculus or point of view O and a line L not parallel to p , let M be the line through O and parallel to L . Then the vanishing

point of L is the intersection of M and p [7]. Another, perhaps more visual, rendition of the concept would be one to say that a vanishing point is a theoretical point in the perspective plane of the image where the straight lines converge. Perspective images contain one or two vanishing points. It can be assumed that depth at the vanishing point is infinite. It being the perspectively furthest point in the image, for the purpose of this study, we will set it for the value zero.

In order to calculate the position of the vanishing points, it is needed to obtain the so-called vanishing lines in the image (illustrated in Figure 6). The concept of a vanishing line is a tool of perspective representation, meaning the line parallel to any original plane in the picture. By definition, the vanishing point is the point resulting from the convergence of vanishing lines. To compute the vanishing lines, it is needed to detect the straight edges in the image. One way to detect lines is the Hough transform.

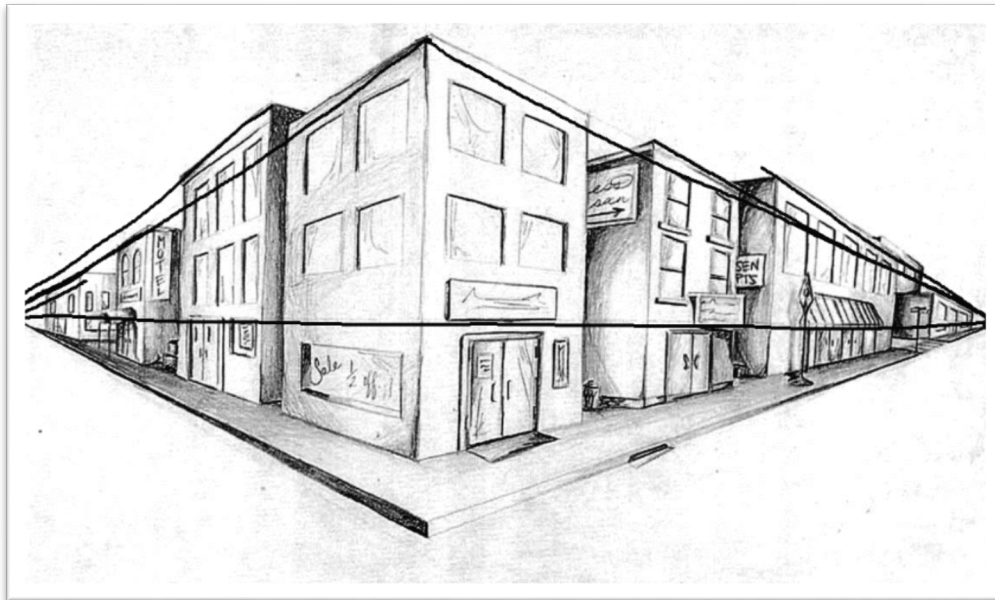


Figure 6: Vanishing lines and vanishing points

3.2.1.1. Hough Transform

The Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing [8]. The classical Hough transform was used to identify lines in an image, but later the Hough transform has been extended to identifying positions of arbitrary shapes, most commonly circles or ellipses.

Straight lines can be detected by a linear transform. In the image space, the straight line can be described by the model called slope-intercept model as $y = mx + b$ where the parameter m is the slope of the line, and b is the y-intercept. In the Hough transform, a main idea is to consider the characteristics of the straight line not as discrete image points (x_1, y_1) , (x_2, y_2) , etc., but instead, in terms of its parameters according to the slope-intercept model, i.e., the slope parameter m and the intercept parameter b . In general, the straight line $y = mx + b$ can be represented as a point (b, m) in the parameter space. Since vertical lines pose a problem as they are more naturally described as $x = a$ and would result in unbounded values of the slope parameter m . That is why the use of a different parameter set, denoted r and θ is a better representation for the lines in the Hough transform. These two values, taken in conjunction, define a polar coordinate, as is illustrated in Figure 7.

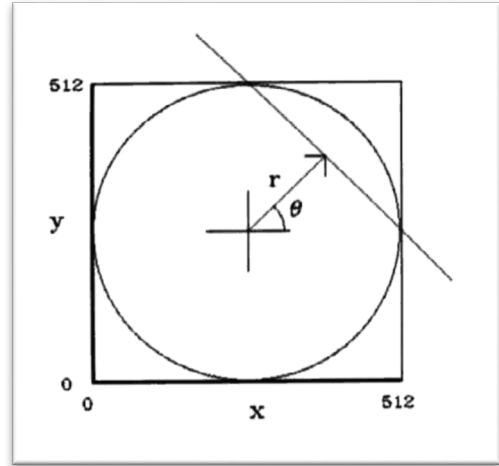


Figure 7: Polar coordinates of a line

The parameter r represents the algebraic distance between the line and the origin, while θ is the angle of the vector from the origin to this closest point. With these parameters, the equation of the line can be written as [8]:

$$(11) \quad y = \left(-\frac{\cos \theta}{\sin \theta}\right)x + \left(\frac{r}{\sin \theta}\right)$$

This can be rearranged to:

$$(12) \quad r = x \cos \theta + y \sin \theta$$

Therefore, it is possible to map a line in the image to a pair of polar coordinates (r, θ) .

3.2.2. Depth Gradient

The perspective setting of an image can be assumed from the position of the vanishing points and of the vanishing lines in the image plane. The perspective setting can consequently be used to deduce information about the three-dimensional space of the scene of the image. One study [3] suggests a series of tests based on heuristic rules to determine the approximate location of the vanishing point. According to the method, the vanishing point's relative location to the image itself on

the image plane can fall into one of these five categories, further classifying the image:

```
P.x<=0 AND (H-1/W-1)*P.x<P.y<-(H-1/W-1)*P.x+H-1 //Left case
P.x>=W-1 AND -(H-1/W-1)*P.x+H-1<P.y<(H-1/W-1)*P.x //Right
case
P.y<=0 AND (W-1/H-1)*P.y<=P.x<=(W-1/H-1)*(H-1-P.y) //Up
case
P.y>=H-1 AND (W-1/H-1)*(H-1-P.y)<=P.x<=(W-1/H-1)*P.y//Down
case
0<P.x<W-1 AND 0<P.y<H-1 //Inside
case
```

$P.x$ and $P.y$ coordinates stand for the x and y location of the vanishing point P , while W and H denote the horizontal and vertical dimension of the image, respectively.

Once the image has been classified, the method proposes the application of another set of heuristic rules to the vanishing lines, the result of which is the generated Depth Gradient for the image. The Gradient itself consists of a set of planes that meet at the vanishing point, thus recreating the two-dimensional representation of the three-dimensional scene. For that same reason and in order to stay consistent with the definition of the Depth Map, the planes are coloured with a grayscale, zero-to-one gradient.

Two assumptions can be made in order to keep the assigned values consistent with the Qualitative Depth Map:

1. Areas with greater depth will receive lower gray values
2. 2: The vanishing points are the furthest areas of the image

Based on these assumptions, gradient planes can be drawn between vanishing lines, their starting point being the vanishing point itself. This method will yield an image with the same dimensions as the original, where the gradient planes outline the three-dimensional setting of the image. For the fact that this method deduces depth from the geometric information stored in the image, it is called the Geometric Depth Map of the image.

3.3. Focus Map

The Focus Map is the result of an extraction process of the main objects in an image based on their local focus or blur [9]. It can be interpreted as a grayscale image where values show the relative amount of focus with respect to the camera. The assumption is that most images focus on an object or a set of objects in the foreground of the scene can be made, which leads to the opportunity to benefit from this new development.

The main reason for merging focus extraction into Depth Map creation is its potential in reducing errors introduced by colour-based segmentation. These errors occur in images of objects whose textures are difficult to homogenise. Being in possession of a new way of separating objects in an image improves the quality of the resulting depth map.

With the Focus Map extracted, we can tell the location of the main set of objects in the image. Utilising the previous assumption, these coordinates are likely to match the ones that are in the foreground of the image, and are therefore closer to the camera and have an especially low depth value.

Focus information can be obtained in a number of ways, many of them requiring multiple images. Provided the unique properties of the task at hand, it is required that the Focus Map extraction method work with no other information outside that can be extracted from a single image. For this reason, this study is in part based on a research [9] conducted by the Distributed Events Analysis Research Laboratory of the Institute for Computer Science and Control of the Hungarian Academy of Sciences.

The general approach for focus extraction is based on deconvolution methods, such as [9] and [10].

$$(13) \quad G(i) = F(i) * H(i)$$

where $G(i)$ stands for the observed image, $F(i)$ for the yet unknown original image and $H(i)$ for the Point Spread Function (PSF). The PSF contains the camera's unique distortion and the change in contrast. These two effects can be described as the phase and absolute value of the PSF's Fourier transformation. The convolution integral is described by the "*" sign.

The approach [8] followed by this study is based on a localised blind deconvolution method that is capable of obtaining the original image without being familiar with the unique PSF. For this end, the algorithm finds the original image that has

the maximum Bayesian probability of turning into the observed image, given the estimated image (f) of this form:

$$(14) \quad P(f_i g_l) = \left[\frac{P(g_l | f_i) P(f_i)}{\sum_j P(g_l | f_j) P(f_j)} \right]$$

With the definition of conditional probability, the following formula can be given:

$$(15) \quad \sum_l P(f_i g_l) = \sum_l P(f_i | g_l) P(g_l) = \sum_l \frac{P(g_l | f_i) P(f_i) P(g_l)}{\sum_j P(g_l | f_j) P(f_j)}$$

An example of applying the deconvolution algorithm can be seen on Figure 8.

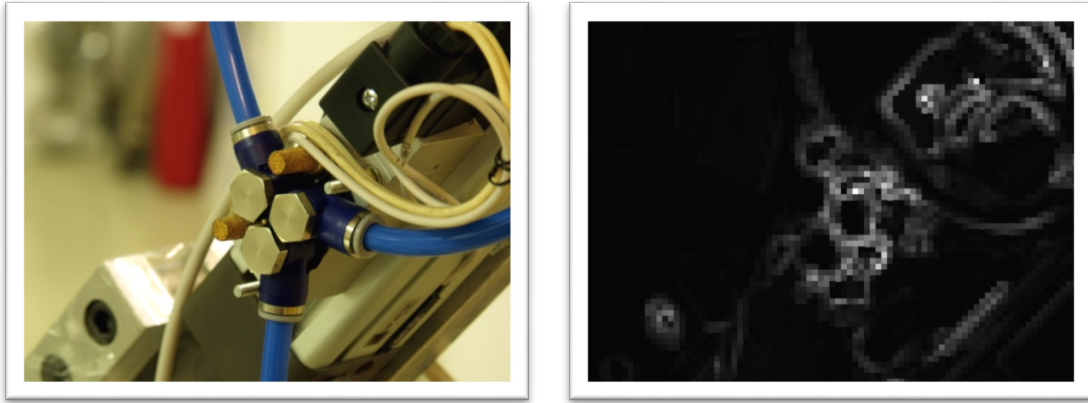


Figure 8: Test image (left) and the Focus Map created for it (right)

3.4. Stereoscopic Image Pairs

To generate the final three-dimensional image, both left-view and right-view images need to be created. In order to do this, parallax values need to be computed from the depth value of each pixel in the image. Then these values are used to shift pixels from the original image, generating the left-view and right-view image. Parallax values can be thought of as a function of inter-ocular distance [11], as can be seen in Figure 9. Knowing the average distance between human eyes (B), which is **usually 6-7 cm** and estimating the distance between the user and the display (D), setting the maximum depth of the 3D effect (P), the maximum of parallax values (M) can be described applying the following formula utilising similar triangles [11] and [6]:

$$(16) \quad \frac{B}{D+P} = \frac{M}{P}$$

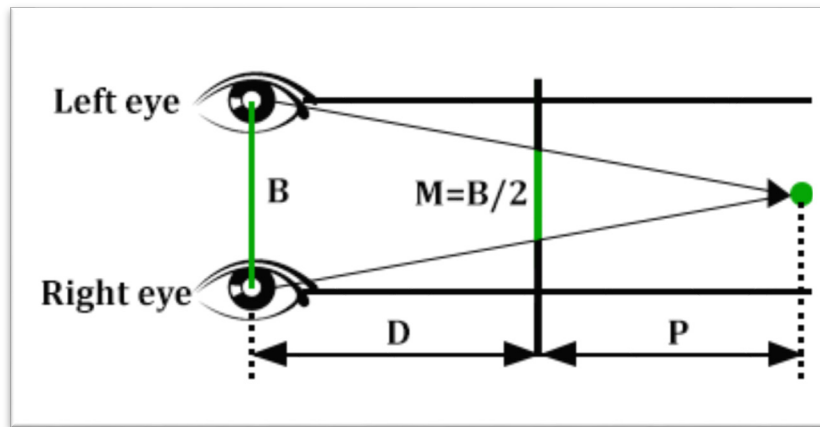


Figure 9: Maximum Parallax relations with respect to the screen

Once M is set, the parallax value of each pixel is computed according to a set resolution, the exact value of which is debated among previous studies [6] and [11]. While studies differ on the value of resolution (R), they agree on the best method for calculating parallax values:

$$(17) \quad \text{Parallax}(x, y) = M * \left(1 - \frac{\text{Depth}(x, y)}{R}\right)$$

Once the parallax values are calculated for the entire original image, the stereoscopic image pair can be created by a simple process. The original image remains the centre view of the stereoscopic pair but will not be directly projected onto the display. To split the parallax value into both views, each original pixel is shifted by the half of its local parallax value to the right when generating the right-view image and by the same value in the opposite direction when generating the left-view image as explained in Figure 10.

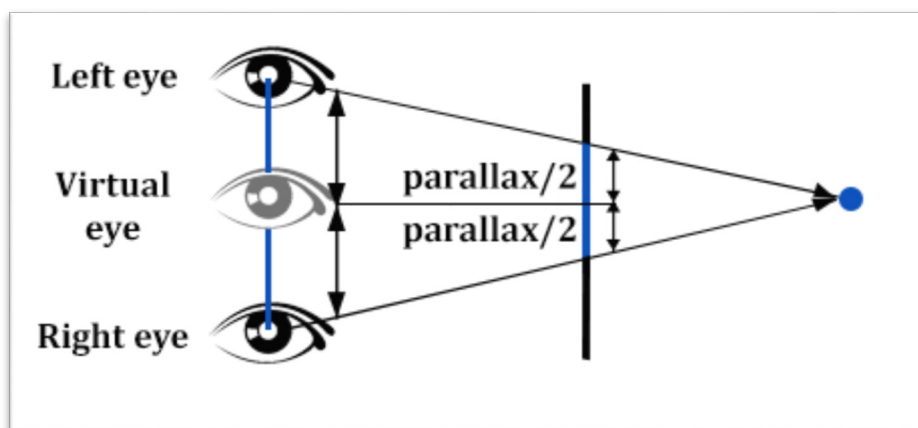


Figure 10: Stereoscopic pair generation

With both views of the stereoscopic pair created, the image is ready to be displayed in three dimensions by 3D ready displays using Side-By-Side technology. Further conversion into Anaglyph 3D, which is discussed in [Chapter 2.1.1.](#), enables the image to be viewed on any screen with suitable anaglyph glasses.

4. Implementation

4.1. Objective

The objective of the implementation was the development of a prototype for fluid stereoscopic conversion making use of the previously explained algorithms.

Since web images cannot guarantee any additional information, the conversion process has to be based on no additional information other than the image itself.

The conversion process has to add an extra dimension to that found in the image. Since this information can only be concluded and approximated, certain assumptions have to be made in order to create a basis on which the algorithms can work well.

During the development of the prototype, the following two assumptions were made:

1. Objects in focus are likely to be closer to the camera than others.

This assumption is presumed to be accurate in a large number of images, since when someone is taking a photo, the object of their interest is usually in the foreground.

To phrase it differently, one rarely focuses deliberately on something that is far away and in the background.

2. The other assumption is that objects that are brighter are likely to be in the foreground of an image. A trivial example of this are photos taken with flash.

It is believed that a great number of images fit at least one of these assumptions. By using multiple approaches, the probability for an image to fit into a category that is handled increases greatly; therefore a viable solution can be built upon these assumptions and the process will result in a good estimation for those cases where 3D reconstruction has to be based on such scarce information.

4.2. Limitations

Due to the unique goal of this study, the process retains certain limitations. Objects that are not visible but would be logically there, are not rendered. Since most of the work is done on a pixel level, objects are recognised as sets of pixels, not as shapes. This means that certain shapes that would logically be visible from a wider angle than the angle of the displayed 3D have their angle of visibility capped at a lower value. To put it through an example, a sphere will have the same limitations on the viewing angle that all the other objects share, even when it theoretically contains enough information to be viewed from any angle.

Objects that are transparent, highly reflective, or refracting are not taken into account while developing the prototype. Issues raised by their behaviour will be addressed in further development steps.

Images that are artistically blurred, or otherwise unrealistically manipulated before conversion do not follow the previously laid guidelines and may therefore give inaccurate results. Also, those images that do not line up with the previously mentioned assumptions may result in inaccurate conversion.

4.3. Specification

The prototype takes an image, preferably a photograph, and produces a stereoscopic image pair based on the explained process. The prototype has the ability to convert these images into a stereoscopic image pair, to be viewed on 3D displays by Side-By-Side technology or, after further encoding discussed in [Chapter 4.4.4.4](#), on any screen through red-cyan or red-blue anaglyph lenses.

For testing purposes, each step of the conversion generates a new map for the original image. These maps, as will be discussed on the following pages, are the following: region map, qualitative depth map, focus map, depth-focus map. All of these intermediate maps can be generated and saved. However, for better performance, most of the conversion steps can be, and are, condensed into a series of operations on each pixel, instead of running a ladder-like structure of operations, in which each map relies on data from the previous map. Minimising the number of accessing operations on image pixels leads to a significant increase in performance.

While the main objective of the prototype was to provide an automatic conversion, the application also hosts an interface for the user to customise certain parameters before conversion. Automatic parameters are set as default and can be reset to these default values, should the user wish to do so.

The prototype runs on Python 2.7 in order to make web integration easier. Image handling is done by Python Imaging Library's (PIL) Pillow version 2.2.2 and Matplotlib version 1.3.1. The slightly older Python 2.7 version was chosen over the more recent Python 3.3 version for stability issues of the imaging libraries.

The operation of the prototype is illustrated with its output at each step. The image in Figure 11 shows the image the program received as input.



Figure 11: Original (starting) image

4.4. Conversion Steps

This chapter explains all the steps that images went through until reaching stereoscopic conversion.

4.4.1. Qualitative Depth Map

The Qualitative Depth Map contains the information that is reached from analysing the colours of the images. The steps of creating it are detailed in the following paragraphs.

4.4.1.1. Mean Shift Clustering

In order to generate the Qualitative Depth Map (QDM – see [Chapter 3.1.](#)), the image is segmented by using a Mean Shift algorithm. The prototype imports the library module PyMeanShift (PMS), a Python module for Mean Shift Image Seg-

mentation under GNU General Public License until a specialised, faster implementation is developed in a stage of further development. PMS was tested with a score of different parameters as to its spatial range and the range of colour.

The prototype unites these parameters to a single precision index and supplies it with a universal value such as was deemed accurate for most test cases, thus enabling the user to view the conversion process as a Black Box while it also leaves them the option to customise the result, enabling the advanced user to fine-tune the result if they so wish.

The Mean Shift process is the longest step of the conversion and is extremely sensitive to the size of the image, given the quadratic computational complexity of the Mean Shift procedure, as read in [Chapter 3.1.1.1](#).

One way to overcome this performance issue is to downsample the image to a size that Mean Shift will run fast enough while the resulting segmentation has a high enough resolution without oversimplifying the image. Segmented regions from downsampled images get resized back to their original sizes after Mean Shift no longer needs them.

While downsampling is not a trivial task, Python's image processing methods can handle it nicely.

Figure 12 shows the prototype's Mean Shift clustering on the original image.



Figure 12: Mean Shift Clustering

4.4.1.2. Absolute Brightness

Based on Assumption #2 (see in [Chapter 4.1](#)), the prototype will assume that brighter colours receive more light, therefore are closer to the point of view. What needs to be realised is that a colour's brightness is independent of the actual hue of the colour. That is the reason for me basing this step of the conversion on the colour's Euclidean Distance from the colour white.

With this approach, the resulting map will contain values that are independent of the original colour being red, blue, yellow or purple. The method used on the

segmented image will yield a map containing a value between 0 and 255, thus creating the Qualitative Depth Map.

This straightforward algorithm maps each pixel to its brightness value calculated from the Euclidean Distance of its colour components, therefore needs no further options for customisation.

Since the Mean Shift greatly reduced the number of colours that appear on the image, the opportunity for further optimisation arose. Using dynamic programming, the brightness values were hashed into a table and values for subsequent pixels were able to be looked up.

This method was greatly advantageous because of Python's built-in support and optimisation for hash tables that allow for fast table lookup.

Figure 13 demonstrates the Absolute Brightness Formula and the Qualitative Depth Map itself.



Figure 13: Qualitative Depth Map

4.4.2. Focus Map

The prototype uses an algorithm [9] previously developed at the Distributed Events Analysis Research Laboratory of the Institute for Computer Science and Control of the Hungarian Academy of Sciences, as a module for this task. Details of this algorithm were previously discussed in [Chapter 3.3](#). Since this implementation has been written in C++, further steps of development may include a full integration into Python unless performance issues arise.

Figure 14 illustrates the generated Focus Map.

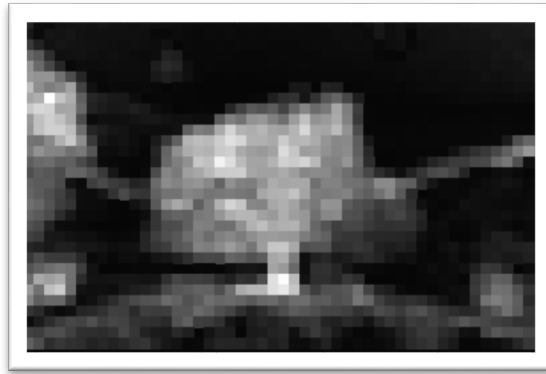


Figure 14: Focus Map

4.4.3. Depth-Focus Map

One of the goals of this study was to combine the advantages of introduced Depth Maps and those of the Focus Map into an innovative Depth-Focus Map (DFM). The concept of also merging a Geometric Depth Map (see [Chapter 3.2](#)) was not continued at this point as it affected performance too much, and the results from the Qualitative Depth Map and the Focus Map provided an estimation that was deemed accurate enough as to validate the continuation of the project with the Qualitative Depth Map and the Focus Map. Once both a Qualitative Depth Map and a Focus Map is available, the next step for the prototype is to merge these maps into a DFM. Several possible mapping patterns were tested until the optimal one was reached.

The following expressions serve as the pseudo code for easier representation of merging patterns and express each map's value at the pixel represented by the location (x,y). DFM stands for Depth-Focus Map, QDM for Qualitative Depth Map, FMO for Original Focus Map. If, at any case, resulting values are out of the spectrum, they are replaced by the highest representable value, 255, which corresponds to pure white.

4.4.3.1. Basic Mapping

```
(1)
DFM(x,y)=QDM(x,y)+FMO(x,y) //Simply add the 2 pixel
values
```

Two issues arose with this trivial kind of mapping. The first was that as the Focus Map's values were generally much higher than values in the Qualitative Depth Map, the balance of representation was shifted greatly towards values of the Focus Map while the Qualitative Depth Map's values went underrepresented, as can be seen in Figure 13. The second issue was that by definition, the implementation of

the Focus Map works with blocks of the image and its output contains blocks of pixels, and this caused the Depth-Focus Map to inherit the sharp edges of the blocks, leading to an unrealistic and aesthetically poor result. To counter these issues, different mapping patterns were tested, as seen on Figure 15.

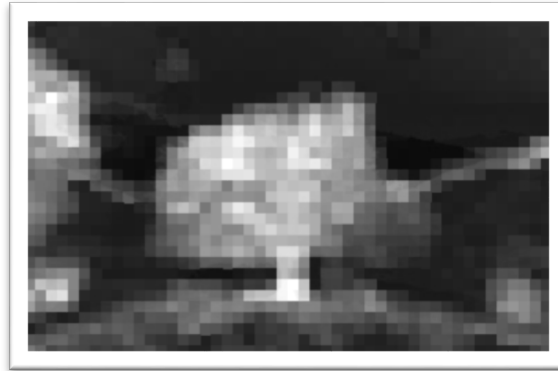


Figure 15: Basic Mapping

4.4.3.2. Weighted Mapping

To counter the issue of misrepresented balance, different compensating weights were introduced. (The results are presented in Figure 16.)

(2)
 $DFM(x, y) = 2 * QDM(x, y) + FMO(x, y)$ // Overrepresented QDM

By doubling the Qualitative Depth Map's weight, it does become more prominent but dark areas might not be dark enough, which might lead to a loss of spectrum.

(3)
 $DFM(x, y) = QDM(x, y) + FMO(x, y) / 2$ // Underrepresented FMO

By halving the weight of the Focus Map, no area received a value of 255, which leads to a possible loss of spectrum, as well.

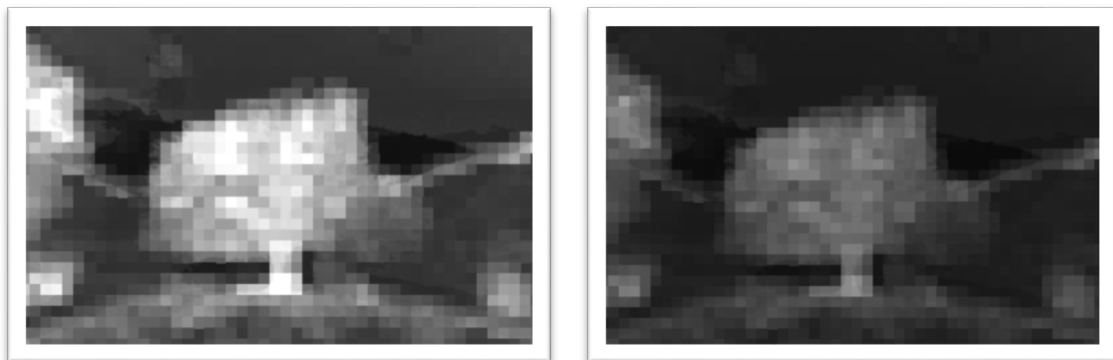


Figure 16: Overrepresented QDM (left); Underrepresented FMO (right)

4.4.3.3. Blurred Mapping

To compensate for sharp edges, Blurred Mapping adds additional Gaussian blur to the Focus Map. Blurring compensates for the edges and slightly homogenises the map while leaving the crucial information of the map untouched. The resulting map will be referred to as FMB, which stands for Blurred Focus Map. The results of the blurring process were favourable, therefore further tests incorporated the previously described process of constructing weights using the Blurred Focus Map.

```
(4)  
DFM(x,y)=QDM(x,y)+FMB(x,y) // equal weights
```

```
(5)  
DFM(x,y)=2*QDM(x,y)+FMB(x,y) //QDM dominant
```

```
(6)  
DFM(x,y)=QDM(x,y)+FMB(x,y)/2 //FMB less dominant
```

According to these results, the mapping pattern where the Qualitative Depth Map was dominant ended up with the most homogenous regions. Homogeneity is of the essence, since it prevents breaking an object into too many pieces. Figure 17 illustrates the different approaches to Blurred Mapping.

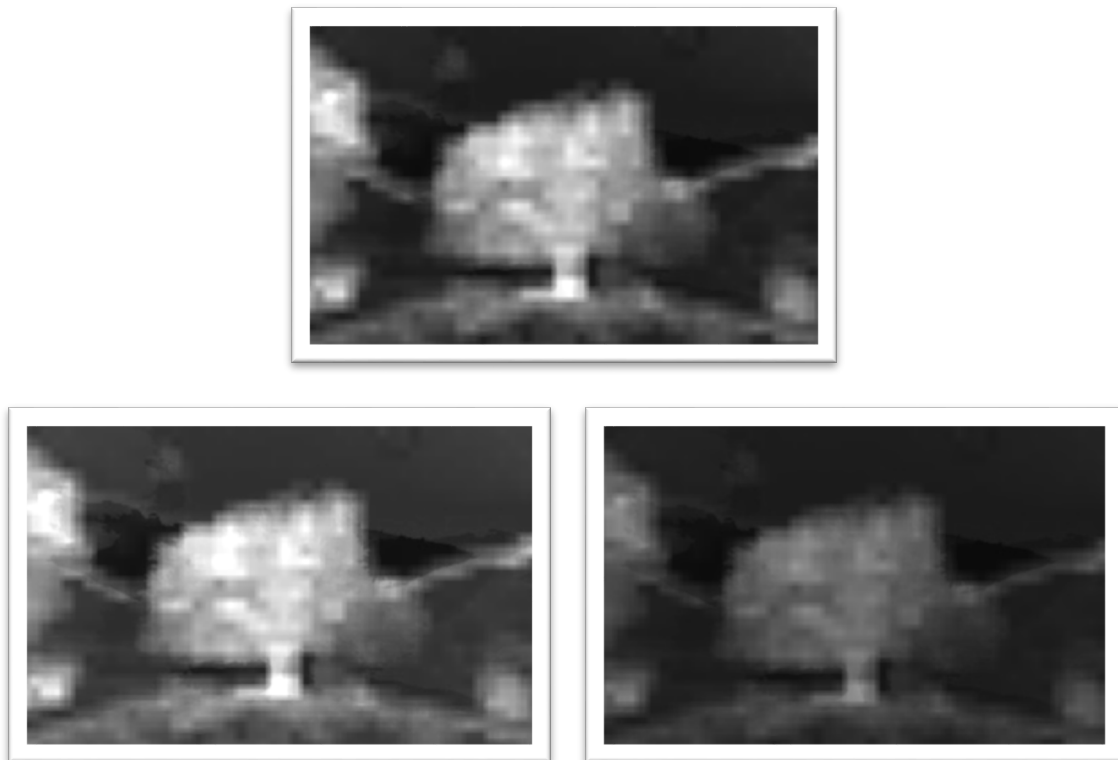


Figure 17: Blurred Mapping with equal weights (up); dominant QDM (left), less dominant FMB (right)

4.4.3.4. Region Filtering

Previously discussed mapping patterns were dealing with the representation of the Qualitative Depth Map and the Focus Map inside the Depth-Focus Map. Region Filtering takes this computed value adds a constant global multiplier to further homogenise and define regions. Since pixel values cannot exceed 255, the objective of this method is to bring the values that are close to that maximum value to be exactly 255, without damaging the depth proportions. This causes the object of interest in the image to be more defined during three-dimensional display. The instrument of this sort of filtering was decided to be a multiplier constant over a static threshold for aesthetic reasons. To set all pixels exceeding a certain depth value would cause sharp tears in the image and that is far less favourable than the smooth result of a constant global multiplier. Tests were run in order to define an optimal constant. Figure 18 shows the Depth-Focus Maps with different values of constant multiplier c . The prototype uses $c=1.25$ by default, however, users are still able to customize the value of c between 1.0 and 2.0. Constants greater than 2.0 condense the Depth-Focus Map to such an extent that it is no longer able to serve its purpose, as can be seen in Figure 19.

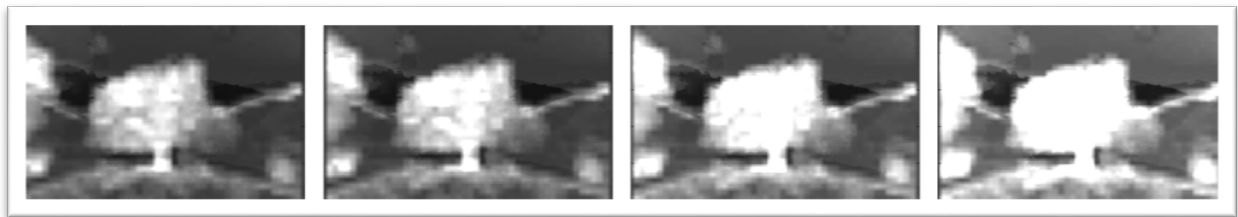


Figure 18: Depth-Focus Map with $c = 1.1, 1.2, 1.5, 2.0$, respectively

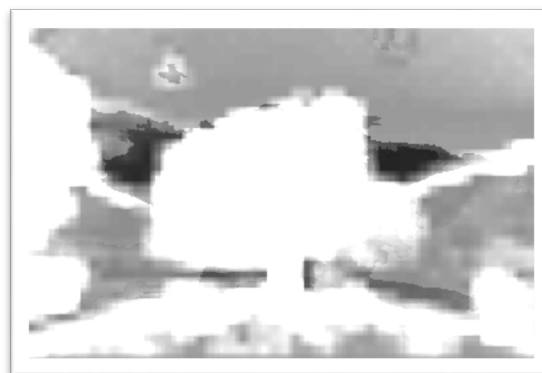


Figure 19: Depth-Focus Map overshot with $c = 3.0$

4.4.4. Stereoscopic Conversion from Depth-Focus Map

With the Depth-Focus Map created, all pixels of the starting image possessed a viable estimation of their distance from the camera, so the final stage of stereoscopic conversion could begin.

4.4.4.1. Parallax Values

Once we are in possession of a Depth-Focus Map, Parallax Values will be computed for each pixel. As seen in the formula below, certain parameters are needed.

Maximum parallax value is measured in pixels according to the theoretical background explained in a previous section and determines the upper limit with which a pixel can be shifted.

The parameter `depthScale` conveys the scale of constructed depth. In other words, how the relative values of the Depth-Focus Map, which are a representation of depth between 0 and 1, map onto absolute depth values of the display. It tells how far from the screen certain points of the image can get.

Both values are set to a default that was deemed accurate for test cases, but the user can modify them if they have the need for it.

$$(18) \quad Parallax(x, y) = maxParallax * \left(1 - \frac{depth(x, y)}{depthScale}\right)$$

When constructing the left and right image, we have to bear in mind that the parallax value thus computed is actually the total parallax for that pixel, the distance between the representation of the original pixel in the left and right image.

That is why the pixel needs to be shifted by half of the parallax distance to the left for the left image and with the same value to the right for the right image.

From an algorithmic point of view, the two objective image planes are initialized and filled by taking each pixel in the original image and creating the image of the previously mentioned pixel in both the left and the right image plane.

This approach is a trivial choice for efficiency reasons, however, it does leave certain pixels blank. This poses an issue of unfilled pixels.

The solution to the problem can be reached reasonably fast. Figure 20 and Figure 21 illustrate how the choice of the `maxParallax` parameter influences the future 3D effect by the size and position of the unfilled pixels as areas coloured in magenta. The colour magenta was a choice made so that these pixels could easily be identified for the next step of conversion with the assumption that this specific colour rarely appears on photographs.

In the case it does appear, the following error-handling solution can be used, but it does come with a certain additional cost in computation:

Another image can be created with a different, equally unlikely colour as a filler and identifier for blank areas and then the definition of a ‘blank pixel’ can be modified to mean ‘magenta in test image 1’ and ‘this specific colour in test image 2’.

If a pixel fails either test, it can be said that it is not a blank pixel and therefore can be left alone for the remaining tasks. Still, this is a somewhat costly method and can greatly affect performance.



Figure 20: Stereoscopic left and right images with blank pixels (`maxParallax` at 16 pixels)



Figure 21: Stereoscopic left and right images with blank pixels (`maxParallax` at 32 pixels)

4.4.4.2. Stencil filtering

Once the pixels were shifted, and the left and right images were created, I needed to handle the issue that arose with the shifting of pixels, as seen on magenta areas of Figure 20 and Figure 21 above.

The magenta-coloured areas show those pixels in the image pair that have no source pixel in the original image due to the amount of information a single image can hold.

However, it was possible to estimate the value of those points by taking the closest of those pixels that we do know and using them to work out a sufficient estimation of previously unknown pixels.

Stencil filtering is the result of two main components, the selection method and the filtering formula, both of which will be discussed during the course of the next few paragraphs.

4.4.4.2.1. Von Neumann Stencil

The von Neumann Stencil is an algorithm frequently used in image filtering tasks and one that is scalable to parallel computing tasks, as well.

For each pixel, it takes the von Neumann neighbours¹ of the pixel and computes its resulting value for the original pixel according to the values of the selected neighbours and a defined formula. One example of utilising such a stencil is the blurring of an image.

The von Neumann neighbours of a point in n -dimensional space can be defined as those points that have exactly one common coordinate with the original pixel and one or more of their other coordinates differ exactly by one.

In a two-dimensional image space that simplifies to those pixels that share an edge with the original pixel. In other words, the pixel directly above, below, to the left and right of the pixel in question.

An illustration of the von Neumann neighbours of a specific point in either 2D or 3D space can be seen on Figure 22.

¹ John von Neumann originally defined this neighbouring relationship to describe the workings of cellular automata.

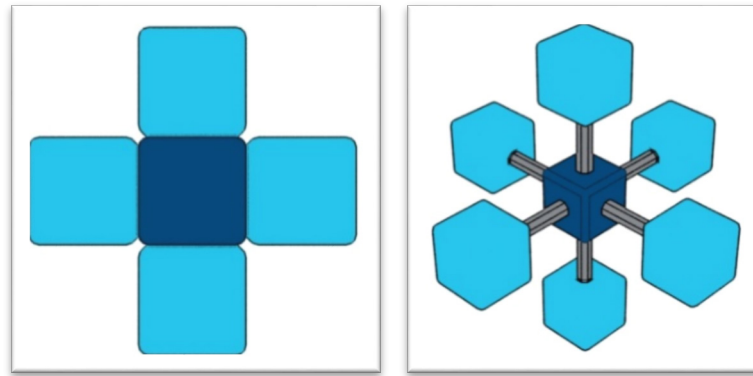


Figure 22: Von Neumann neighbours of a point in 2D (left) and 3D (right)

4.4.4.2.2. Selection by Recursive von Neumann Stencil

The previously discussed von Neumann stencil seemed a fine starting point. Still, it could not be applied in its original form, by the simple fact that due to region shifting, blank pixels appear in clusters, so many of their original von Neumann neighbours would be among the blank pixels as well. To avoid this issue, I modified the original concept of the von Neumann stencil and upgraded it to my own design. I took the concept of the von Neumann stencil and the idea of recursive ray tracing and merged these to form what I named recursive von Neumann stencil (RVNS). The idea behind selecting a suitable neighbour is selecting the closest non-blank pixel in the desired direction, and the method is sending a simple ray in that direction that returns with the first non-blank pixel it finds. If the ray reaches the end of the image, then no pixel data is returned and the computation will continue accommodating the lesser number of relevant pixels. If such rays are sent out in all four directions of the original two-dimensional von Neumann stencil, the result is the set of the closest pixels we can work with. This recursive method is, of course, scalable into three-dimensional space but that is not one of the goals of this project.

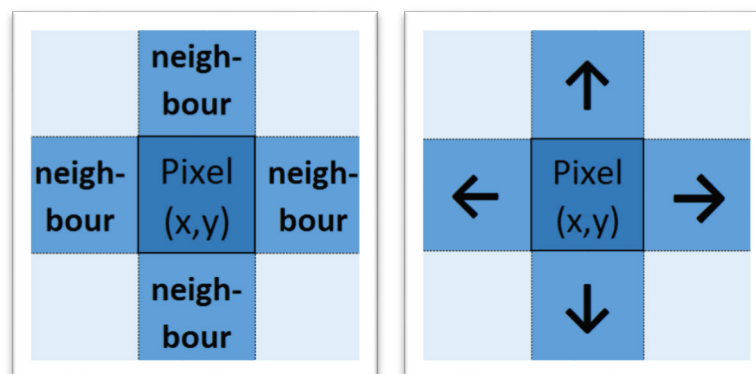


Figure 23: Illustration of the original Von Neumann Stencil (left) and the recursive solution used by the prototype (right)

The following Python code demonstrates the inner workings of the Recursive Stencil filter. The stencil is pixel-based and is called for each blank pixel.

```
def vonNeumannRStencil(objectiveImage, originalImage, pos):
    """Interprets the mean of pixels in a VonNeumann stencil
    from objectiveImage with center of originalImage[pos] where
    pos is an (x,y)-tuple"""
    assert pos<=objectiveImage.size
    original=originalImage.getpixel(pos)
    stencilBuffer=original #initialize to original pixel
    pixelCnt=1
    #Stencil
    tmp={-1,0,1}
    for dx in tmp:
        for dy in tmp:
            if dx!=dy: #Von Neumann neighbours
                update=updateStencil(stencilBuffer,
                                    pixelCnt,pos,(dx,dy),objectiveImage)
                stencilBuffer=update[0]
                pixelCnt=update[1]
    #Mean
    stencilBuffer=avgTupleValues(stencilBuffer,pixelCnt)
    return stencilBuffer

def updateStencil(stencilBuffer,pixelCnt,pos,(dx,dy),
                 objectiveImage):
    (x,y)=validateCoordinates(objectiveImage,(pos[0]+dx,
                                           pos[1]+dy)) #assure (x,y) in image
    try:
        stencilBuffer=addTupleValues(stencilBuffer,
                                     getNextPixel(objectiveImage,x,y,dx,dy))
        pixelCnt+=1
    except IndexError:
        coffeeBreak()
    return [stencilBuffer, pixelCnt]

def getNextPixel(Image, x,y, dx,dy):
    """Returns the next non-magenta pixel from x,y in dx,dy
    direction"""
    if Image.getpixel((x,y))==(255,0,255):
        return getNextPixel(Image,x+dx, y+dy, dx,dy)
    else:
        return Image.getpixel((x,y))
```

4.4.4.2.3. Filtering Formula

Once the relevant pixels are collected, the filtering formula can be used to calculate the resulting pixel value, in this case, its RGB colour value. The objective of this computation can be described as a sort of weighted averaging process. During the development of the prototype the following two filtering kernels were implemented and tested. Both methods work with five points, four of which are gathered by the RVNS described in the previous paragraph, and the fifth being the pixel at the original location of the starting image. The reason behind using the starting image is that parallax values guarantee that the resulting stereoscopic images

closely resemble the starting image, therefore it can be assumed that the points of the starting image hold valuable information to be considered in the averaging process. The filtering formula can be described in the following manner, provided that $P(x,y)$ is the resulting value of the pixel at coordinates x and y , $RVNS(x,y)$ is the set of relevant pixels gathered by the Recursive Von Neumann Stencil described above, $S(x,y)$ is the pixel value at location (x,y) in the starting image, and ϕ is the current filtering kernel. Descriptions of implemented filtering kernels are given in the following paragraphs.

$$(19) \quad P(x,y) = \phi(RVNS(x,y), S(x,y))$$

4.4.4.2.3.1 Basic Filtering Kernel

Basic kernel is the simplest and fastest way to achieve the above mentioned filtering effect. During this method, all weights are set to 1.0 and the result is a simple averaging process. When this method is chosen during configuration, even the RVNS's functionality is simplified according to the kernel's needs. The advantage to using this filtering method is its speed. When dealing with large images, or many images, as the case would be on websites, a small optimisation in performance can have a significant impact on the execution of the whole software. Figure 24 shows the result of applying Basic Filtering.



Figure 24: Stencil Filtering using Basic kernel

4.4.4.2.3.2 Cross Filtering Kernel

Cross filtering takes into account the distance of each of the relevant pixels from the original one. A pixel's weight is computed according to the distance the pixel has from the original one. In this case the RVNS is configured to give that distance, as well. Weights are distributed and handled by using a weight buffer built up from all relevant pixel data. I named this kernel cross filtering for the reason that it relies on the intuitive cross pattern of the used RVNS. The pixels are structured into pairs of horizontal and vertical neighbours and each pair contributes to the value of the

original pixel in a manner that is close to the one used in linear interpolation and filtering.

The theory behind this method is the following. The distance between the vertical or horizontal pair of pixels (A and B) is redefined as a unit and seen as a line with the pixel A at position 0 and B at position 1. As per the definition of RVNS, the original pixel P will be on this newly defined line, somewhere between 0 and 1. The position of P splits the section $[0,1]$ into two parts. The length PB will give the weight to pixel A, while the length AP will give the weight to pixel B. Using this method, it can be ensured that the closer a relevant pixel is to the original, the greater weight it will have in the final averaging process. The Cross kernel ends up with two pairs of pixels, and also the pixel in the starting image. This adds up the individual weights to three, which is then taken into account during the final step of the computation.

The Cross filtering kernel's method is similar to bilinear interpolation, but differs at one crucial point. While bilinear interpolation can be used to estimate a point's value where it lies in the rectangle defined by the four relevant points, Cross filtering estimates the value of the point at the intersection of the two lines defined by the pairs of points. Figure 25 illustrates this difference.

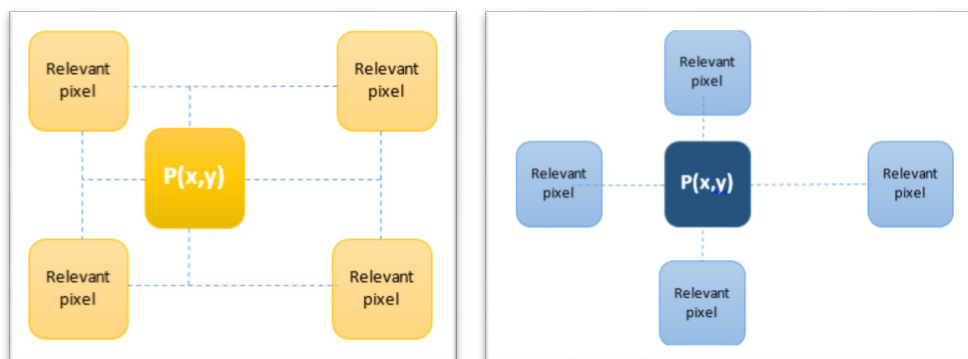


Figure 25: Bilinear Interpolation (left); Cross Filtering (right)

Cross filtering kernel lets the starting image have a larger impact on the resulting image, but it is slower than the basic filtering kernel. I recommend this option in situations when performance is less important. The result of the cross filtering kernel is shown in Figure 26.



Figure 26: Stencil Filtering using Cross Filtering Kernel

4.4.4.3. Stereoscopic Image Pair

Applying the Stencil filtering will result in the final stereoscopic image pair. These can easily be converted into either a Side-By-Side image, to be viewed on a 3D screen, or into a single anaglyph image that can be displayed on any screen. Figure 27 shows the image pair created from the original image.



Figure 27: Stereoscopic Image Pair, Left Image and Right Image, respectively.
(Using Basic Stencil Filtering for improved performance)

4.4.4.4. Anaglyph Image

Having both the left and the right image in possession, the anaglyph image can be created by taking the colour channels from either image. As anaglyph glasses are usually available with either red-blue or red-cyan lenses, the prototype is prepared to present the output in both formats. The colour channels of the anaglyph image are taken from the image pair according to the colour each lens is filtering. In practice, it is done in the following manner:

Red-Cyan Lenses:

Red:= Left Image's Red channel
Green:= Right Image's Green channel
Blue:= Right Image's Blue channel

Red-Blue Lenses:

Red:= Left Image's Red channel

Green:= Average of Left Image's Green channel and Right Image's Green channel

Blue:= Right Image's Blue channel

Figure 28 shows the resulting anaglyph image converted for Red-Cyan lenses.



Figure 28: Anaglyph conversion

4.5. Further Results

These paragraphs will demonstrate the prototype's performance on different images.

4.5.1. Brightly Lit Images

The image on Figure 29, Figure 30 and Figure 31 shows how the prototype performs on images that are well-lit in the foreground and darker in the background. It can be noted that the colour contrast of the skier and his surroundings became superimposed on the stereoscopic images as spatial effects in 3D space.

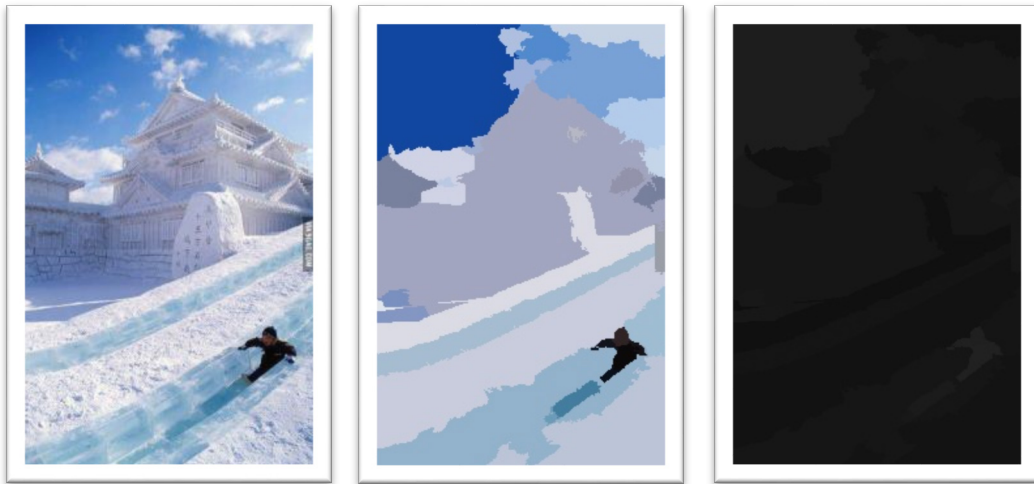


Figure 29: Starting image (left); Regions (middle); Qualitative Depth Map (right)

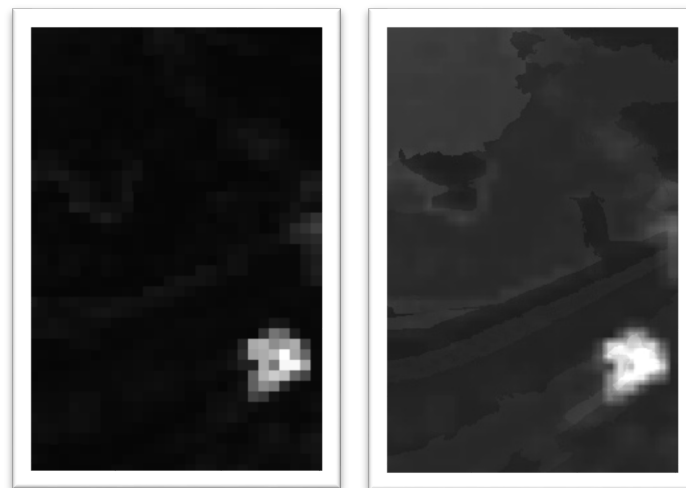


Figure 30: Focus Map (left); Depth-Focus Map (right)

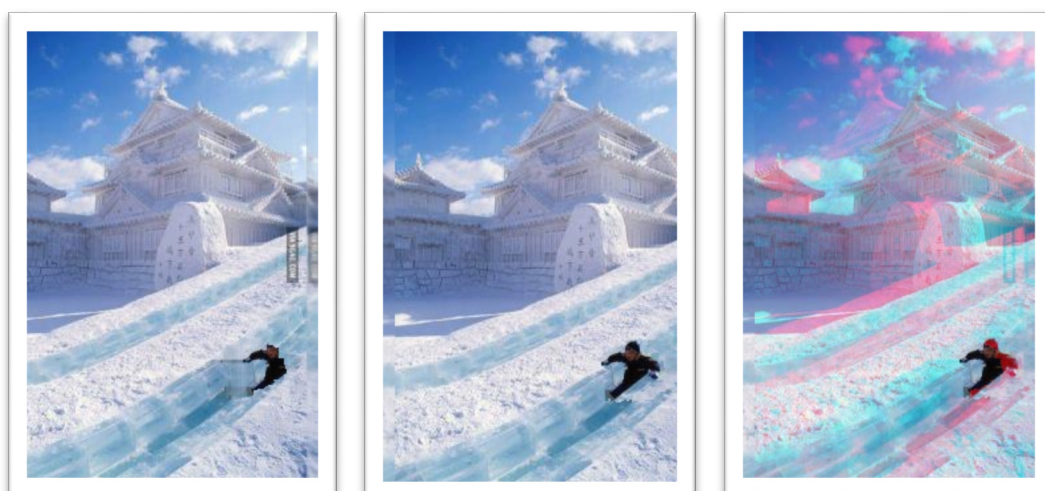


Figure 31: Stereoscopic Images (left and middle); Anaglyph Image (right)

4.5.2. Well-Focused Images

The image on Figure 32, Figure 33, Figure 34, Figure 35, and Figure 36 shows the prototype's performance on images that have an object in focus in the foreground, while objects in the background are not in focus.

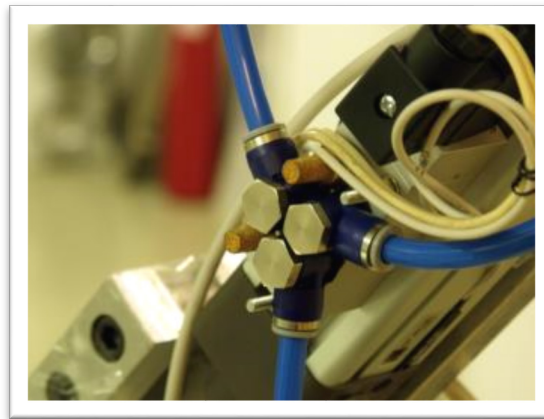


Figure 32: Starting image

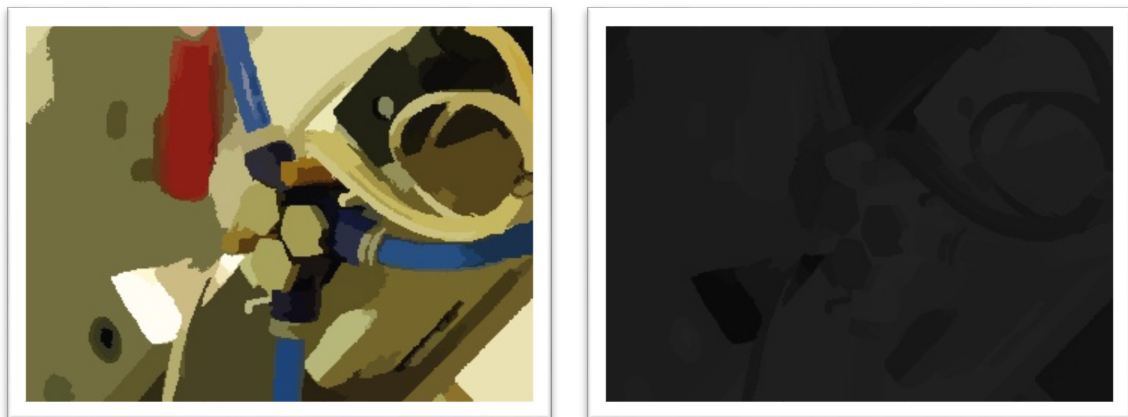


Figure 33: Regions (left); Qualitative Depth Map (right)

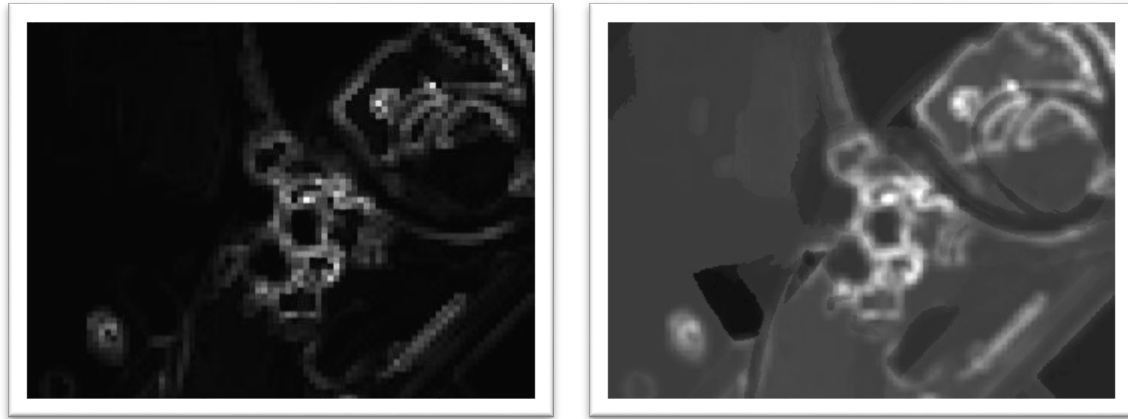


Figure 34: Focus Map (left); Depth-Focus Map (right)

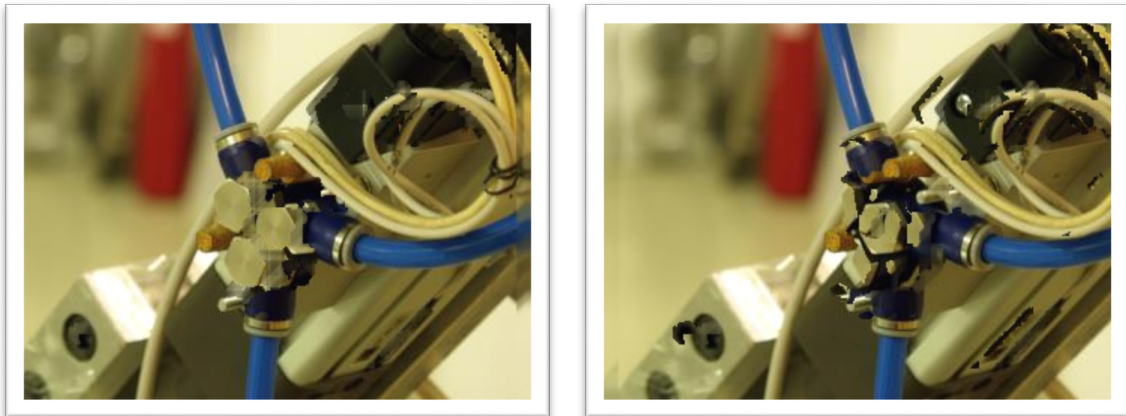


Figure 35: Stereoscopic Left and Right Images

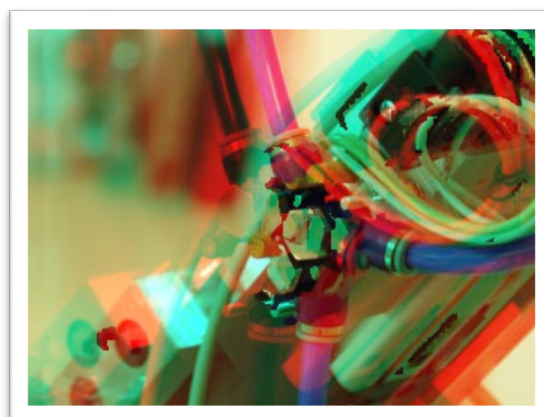


Figure 36: Anaglyph Image

4.6. Further Plans

The prototype has multiple branches of possible further development, as it is connected to more than one field. Concerning technical approaches, further optimisation of algorithms is always a possibility and should not be forgotten. As the main objective of this application is to be used on the web, continuous development is crucial for it to stay robust and up-to-date. New and improved algorithms can be introduced to ensure the same.

Another possibility is to take image analysis further and rely more on customisability than atomisation. By creating a three-dimensional space and utilising region-based segmentation, it is possible to redefine regions in the image as flat objects in 3D space. From that point on, objects can behave as semi-three-dimensional, meaning they can be translated, rotated, or scaled. Their estimated depth can also be modified, which not only leads to even more precision, but also to the possibility to create artistic effects and make the application into a multipurpose graphical tool.

5. Summary

I have examined methods of stereoscopic display and software technologies that utilise these methods to display three-dimensional content on capable hardware devices. I have also examined the theoretical background of and recent research results in the field of image analysis that enabled me to further develop concepts of computer graphics and customise them to fit the objectives of the project. After examining previous methods of similar objectives, I developed a method of depth map estimation from a single-view image, one that requires no additional input except for the image itself. That method proved to be successful and accurate in the majority of cases. Based on this method, I developed an application that is fit to be run online as part of the 3D conversion of webpages. The application was made to be a part of the implementation that transforms the Institute for Computer Science and Control of the Hungarian Academy of Sciences' website into stereoscopic formats and is running as such.

Another point was mentioned in the first chapter of this study (see [Chapter 1](#)); it was assumed that were three-dimensional content more easily and widely available, the demand for hardware capable of displaying 3D would be likely to produce a significant increase, too. With this application being a part of the 3DWeb project, it will be easier than ever before to create web content that is readily available in

3D. With this 3D suite it is possible to make web programmers able to explore the possibilities that 3D rendering has, without the pull-back factor of them having to address the purely technical issues that arise with the special requirements of stereoscopic rendering.

6. Bibliography

- [1] G. Neumann and S. Kopácsi, "Development of 3D Webpages," in *CSIT'2013. Proceedings of the 15th international workshop on computer science and information technologies*, Vienna-Budapest-Bratislava, 2013.
- [2] L.-M. Po, "Automatic 2D-to-3D Video Conversion Techniques for 3DTV," Department of Electronic Engineering City University of Hong Kong, 2010.
- [3] S. Battiato, S. Curti, M. La Cascia, M. Tortora and E. Scordato, "Depth map generation by image classification," *Three-Dimensional Image Capture and Applications VI*, pp. 95-104, April 16, 2004.
- [4] D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach toward Feature Space Analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on (Volume:24 , Issue: 5)*, pp. 603-619, May 2002.
- [5] S. Thirumuruganathan, „Introduction To Mean Shift Algorithm,” 29 03 2010.
Available:
<http://saravananthirumuruganathan.wordpress.com/2010/04/01/introduction-to-mean-shift-algorithm/>.
- [6] S. Battiato, A. Capra, S. Curti and M. La Cascia, Writers, *3d Stereoscopic Image Pairs by Depth-Map Generation*. [Performance]. 2004.
- [7] in *The Geometry of an Art: The History of the Mathematical Theory of Perspective from Alberti to Monge*, Springer, 2007.
- [8] R. O. Duda and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Commun. ACM 15(1)*, pp. 11-15, January 1972.

- [9] L. Kovács and T. Szirányi, "Focus Area Extraction by Blind Deconvolution for Defining Regions of Interest," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pp. 1080-1085, June 2007.
- [10] L. Kovács and T. Szirányi, "Relative Focus Map Estimation Using Blind Deconvolution," *Optics Letters, OSA*, vol 1. 30. 3021-3023, 2005.
- [11] J. Ko, M. Kim and C. Kim, "2D-To-3D Stereoscopic Conversion: Depth-Map Estimation in a 2D Single-View Image", in *Proc. SPIE 6696, Applications of Digital Image Processing XXX*, San Diego, September 24, 2007.

7. Table of Figures

Figure Number	Name	Sources
Figure 1	Image encoded using Anaglyph 3D	flickr.com
Figure 2	Side-By-Side image	flickr.com
Figure 3	Different polarisations of light	wikipedia.com
Figure 4	Shutter Glass Technology	Panasonic
Figure 5	Illustration of Autostereoscopic technologies	wikipedia.com
Figure 6	Vanishing lines and vanishing points	pinterest.com
Figure 7	Polar coordinates of a line	wikipedia.com
Figure 8	Test image and the Focus Map created for it	personal images
Figure 9	Maximum Parallax relations with respect to the screen	personal image
Figure 10	Stereoscopic pair generation	personal image
Figure 11	Original (starting) image	istockphoto.com
Figure 12	Mean Shift Clustering	personal rendering
Figure 13	Qualitative Depth Map	personal rendering
Figure 14	Focus Map	personal rendering

Figure 15	Basic Mapping	personal rendering
Figure 16	Overrepresented QDM and Underrepresented FMO	personal rendering
Figure 17	Blurred Mapping with equal weights; dominant QDM and less dominant FMB	personal rendering
Figure 18	Depth-Focus Map with $c = 1.1, 1.2, 1.5, 2.0$, respectively	personal rendering
Figure 19	Depth-Focus Map overshoot with $c = 3.0$	personal rendering
Figure 20	Stereoscopic left and right images with blank pixels (maxParallax at 16 pixels)	personal rendering
Figure 21	Stereoscopic left and right images with blank pixels (maxParallax at 32 pixels)	personal rendering
Figure 22	Von Neumann neighbours of a point in 2D and 3D	personal rendering
Figure 23	Illustration of the original Von Neumann Stencil and the recursive solution used by the prototype	personal rendering
Figure 24	Stencil Filtering using Basic kernel	personal rendering
Figure 25	Bilinear Interpolation and Cross Filtering	personal rendering
Figure 26	Stencil Filtering using Cross Filtering Kernel	personal rendering
Figure 27	Stereoscopic Image Pair, Left Image and Right Image, respectively. (Using Basic Stencil Filtering for improved performance)	personal rendering
Figure 28	Anaglyph conversion	personal rendering
Figure 29	Starting image; Regions; Qualitative Depth Map	istockphoto.com & personal rendering
Figure 30	Focus Map and Depth-Focus Map	personal rendering
Figure 31	Stereoscopic Images and Anaglyph Image	personal rendering
Figure 32	Starting image	personal image
Figure 33	Regions and Qualitative Depth Map	personal rendering
Figure 34	Focus Map and Depth-Focus Map	personal rendering
Figure 35	Stereoscopic Left and Right Images	personal rendering
Figure 36	Anaglyph Image	personal rendering