Novel Web-based Autostereogram Creation using GPU Stereo Vision

Minh Nguyen, Alexander Woodward, Roy Sirui Yang, Georgy Gimel'farb, and Patrice Delmas
Dept. of Computer Science, The University of Auckland, Auckland 1142, New Zealand

***Abstract*—We present a fully featured, web-based, online autostereogram creation system that allows a user to upload their own stereo images, generate depth data via computational stereo vision, and then turn this depth data into an autostereogram. The system can also perform the reverse process and extract depth data from a given autosteregram or generate anaglyphs from them. By leveraging the parallelism of modern graphics processors (GPUs), the system can process video streams, creating depthmaps and converting them into autostereogram videos at real-time frame-rates transmitted over the internet. For usability the system provides automatic image rectification for the user provided stereo image pairs. These novel features place the system ahead of current alternatives and allows a wide variety of users to experience stereo reconstruction and autostereogram generation in a quick and easy manner. Additionally, the system could serve as a platform for online based visual perception studies.**

I. Introduction

Autostereograms are two-dimensional images that have the unique property that when viewed correctly, are designed to create the visual illusion of a three-dimensional (3D) scene from a two-dimensional image. They are a type of stereogram, also known as a single-image stereogram (SIS), that consists of a repeating pattern with small variations across itself that are determined by the 3D scene (given as a depth map) that the autostereogram is designed to show.

Since the images are two-dimensional, they can be easily shown on any flat display such as a computer screen, a painting canvas, or a flat glass panel, and they can be viewed without the aid of special 3D equipment such as a 3D display and 3D glasses.

Autostereograms have been found to be a useful tool for studying stereoscopic vision in humans [1], as they provide a straightforward way to manipulate particular depth cues and to insert conflicts between these cues. The fact that they require no extra hardware devices for viewing is an added benefit. As examples, Likova et al. used autostereograms to study the dynamic aspects of stereo vision in humans [2] and in order to study stereo-vision deficiency in humans, Skrandies [3] analysed how random-dot stereograms, related to autostereograms, evoked activity in cortical neurons that were sensitive to binocular disparity.

In addition to their scientific merit, autostereograms are popular as entertainment - cheap to produce and easy to use. There are currently many online applications that allow Internet users to generate autostereograms within a fraction of a second, such as [4], [5], [6], [7]. Most of these applications let the user create new autostereograms from a number of pre-given depth map and pattern images from which the user can choose from.

Subsequently, we present a complete and novel web-based system for autostereogram creation with a number of benefits over the alternative online applications. The main benefits are:

1) The 3D model to be hidden in the autostereogram can be generated from computational stereo matching on a user provided input image pair, rather than choosing from a selection of given models. Here, depth maps can be created on the fly by providing image pairs captured by any hand-held camera. From only two images taken slightly to the left and slightly to the right of a simple object, our application will do auto-rectification to align the two images into canonical stereo geometry, process the pair to generate a depth map, and then generate a new autostereogram for the user to view in 3D.
2) The system has the ability to create an autostereogram live stream from a stereoscopic camera source by using the power of GPUs to compute depth maps - currently implemented using a Minoru webcamera online.
3) Our system lets users upload a autostereogram and generate the depth map that was associated with the autostereogram, again using stereo vision. This also allows those who have trouble seeing autostereogram to find out what is hidden within it.
4) The system can also turn autostereograms into anaglyphs, viewable by red-cyan glasses as an additional viewing method.

Additionally, since this project is web-based, it is open to

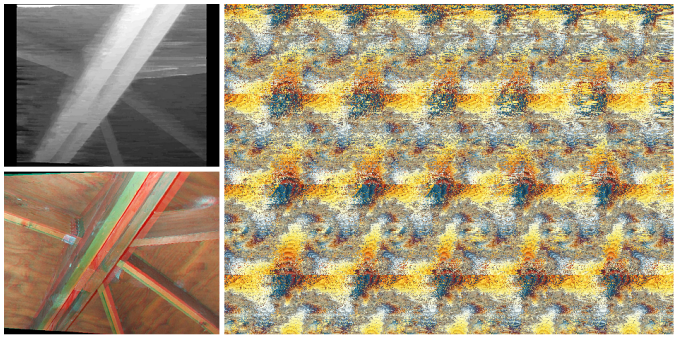

Fig. 1. An example autostereogram from our system (right), generated from a pair of unaligned photos of a wooden roof, the top-left image is the generated depthmap and the bottom-left image is of an anaglyph.

the public and available to anyone from around the world who has Internet access.

II. Background

A. Autostereogram description

Autostereograms were first introduced in [9], where Tyler et al. demonstrated the ability to create depth illusions by using patterns consisting of random dots. Such autostereograms can contain an unlimited range in depth and can create 3D in depth planes both closer and further away from the display depth plane.

The simplest type of autostereogram consists of horizontally repeating patterns known as *wallpaper autostereograms*, e.g. Fig. 2. When viewed with proper vergence of the eyes, the repeating patterns appear to float above or below the background, thus creating a 3D illusion. The second type of autostereogram involves scenes that do not necessarily repeat themselves and have at first glance a more random, repeating texture, hence these slightly more complex autostereograms can display 3D scenes of single objects and are used in this work.

With an autostereogram, the brain receives repeating 2D patterns from both eyes and attempts to match them. Depth can be created by making pairs of patterns have a different repetition rate from the background, leading the brain to focus certain pairs at different parallax angles, hence they appear to exist at a different depth.

There are two ways of viewing an autostereogram: wall-eyed and cross-eyed. Wall-eyed viewing requires the eyes adopt relatively parallel angle, often with the instruction to look into the distance, while cross-eyed viewing requires a relatively convergent angle, or as looking at an object very close to one's eyes. Most autostereograms are designed to be viewed using the wall-eyed technique. When viewed using the cross-eyed viewing, the viewer will see the same 3D scene only with the depth reversed.

B. Related work

There are currently many Internet based applications for autostereogram viewing and most allow the user to create autostereograms online. One such website is Easy Stereogram builder, where the user can select a shape mask and a pattern of choice to generate new autostereograms [4]. Stereocreator [5] is another website that allows the users to create autostereograms of words of their choice. Another advanced website [6] allows users to create their own depth image designs using Flash and then create an autostereogram from them. Furthermore, many commercial applications exist, such as Stereographic Suite, Easy Stereo, Bigle 3D, 3DMiracle, 3DMiracle, 3DMonster, 3DMonster [7].

As mentioned in the introduction, our proposed application presents a number of benefits over competing solutions, the most important being the ability to create personalised autostereograms from stereo images a user has photographed themselves.

III. Design and implementation

A. Input from single-sensor camera and image rectification

Users can upload and process images from a conventional single-sensor camera using computational stereo vision algorithms. Therefore, the process of image rectification is crucial for calculating a high quality depth map. Points and epipolar lines in a stereo pair are related by the 3×3 *fundamental matrix* F, such that $\mathbf{x}_r^\top F \mathbf{x}_l = 0$, where $\mathbf{x}_l$ and $\mathbf{x}_r$ are column 3-vectors of homogeneous coordinates, corresponding to the 2D conjugate point pair from the left and right images, respectively. With knowledge of F one can rectify a stereo image pair to conform to the canonical stereo geometry where feature points are aligned on the same horizontal scan line and the geometry's epipoles existing at horizontal infinity.

We assume the user has an auto-focus camera (the focal length may differ between left and right images), and the image sensor has an arbitrary resolution and fidelity. Therefore, rectification must be achieved without any prior knowledge

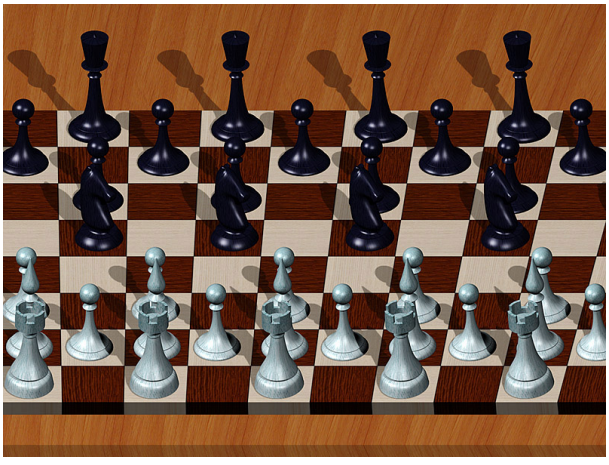

Fig. 2. An example of a wallpaper autostereogram showing 3D scene of a chess board [8].

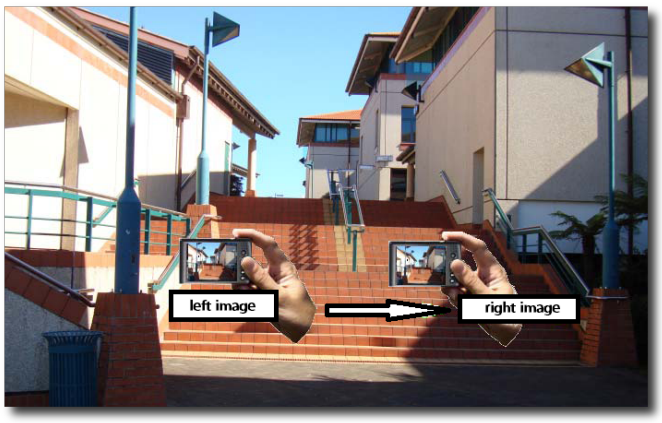

Fig. 3. Diagram of how to take a stereo photo pair from a single camera that is suitable for use with the system.

of camera parameters of which several methods have been proposed, e.g. in [10], [11], [12].

Figure 4 depicts the image rectification procedure which rectifies the left and right images by estimating the fundamental matrix F, in this the 8-point algorithm [13] is run on a set of user-defined or automatically selected point correspondences and then RANSAC [14] is further used to ensure that the feature points selected to estimate F are reliable.

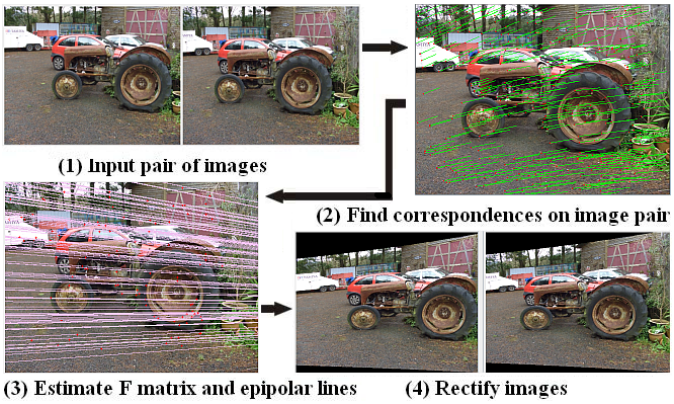

Fig. 4. The process of image rectification: (1) inputs from a single camera, (2) detected feature points on right image - red dots are the good features to track and green lines are feature tracking paths between left and right images, (3) epipolar line estimation; purple lines represent epipolar lines on Fig. 5. Finally, (4) image alignment.

1) Selecting reliable features: The KLT feature tracking algorithm by Shi and Tomasi [15] was used to find reliable feature points to track between the left and right images. KLT considers reliable points with large intensity variations in both the x and y directions. Let $g(x, y)$ denote an image intensity function and $g_x(x, y)$, and $g_y(x, y)$ its first derivatives in the x and y directions, respectively. The eigenvalues, λ_1 and λ_2, of the local 2×2 intensity variation matrix:

$$Z(x, y) = \begin{pmatrix} g_x^2(x, y) & g_x(x, y)g_y(x, y) \\ g_x(x, y)g_y(x, y) & g_y^2(x, y) \end{pmatrix}$$

determine the reliable feature points such that $\min(\lambda_1, \lambda_2) > \theta$, here θ is a chosen fixed threshold.

The n strongest reliable points in the stereo images - here $n = 500$, chosen so our application can run well on most user's computers - are selected as candidates for correspondence assignment. Because image noise can lead to false feature point detection, Gaussian filtering [16] was first applied to the images.

2) Assigning point correspondences between left and right images: After feature point detection in both images, a good set of feature points should now be collected. This task aims at collecting as many correspondences (conjugate pixel pairs) as possible. We used the Lucas-Kanade optical flow in a pyramid [17] tracking algorithm to track feature points between the left and right images.

In order to remove mismatches among the set of correspondences, a number of iterations of forward/backward tracking are carried out.

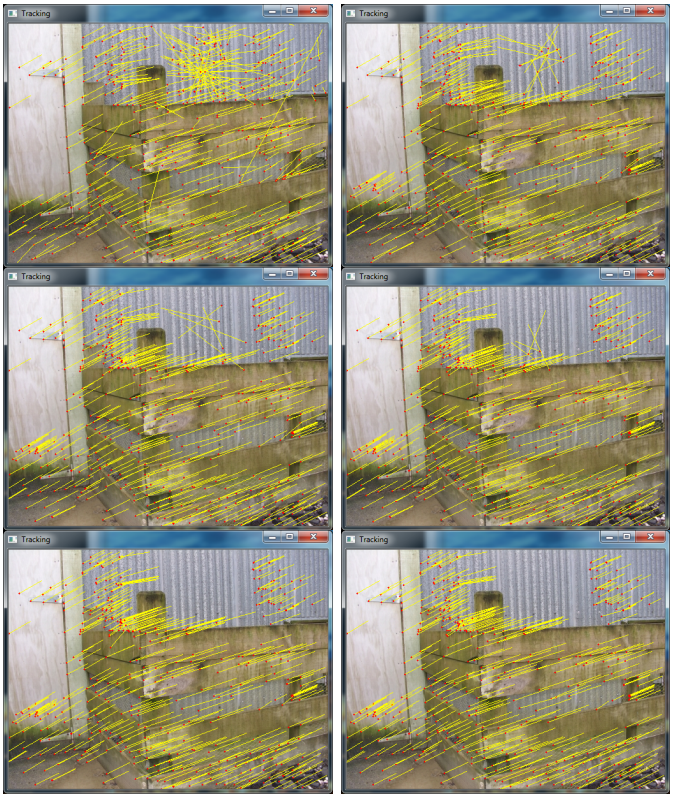

Fig. 5. Feature point tracking after 1, 3 (top row), 5, 7 (middle row) and 9, 11 (bottom row) iterations of mismatch removal - shown on left image of a fence. Red dots are features to track and yellow lines are feature tracking paths between left and right images

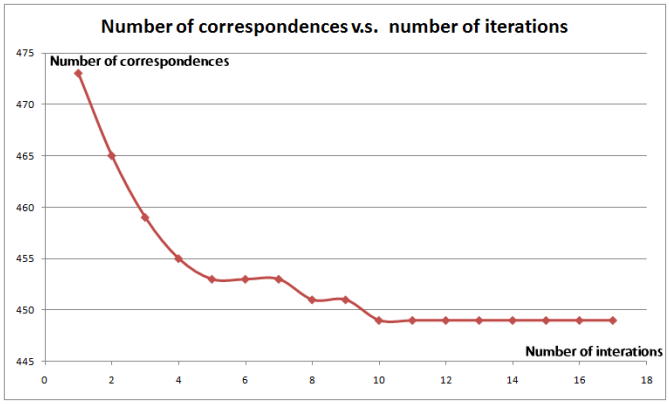

Fig. 6. Trend of number of correspondences after 17 iterations applied on the fence image pair shown in Fig. 5.

In Fig. 5, mismatches have tracking paths that do not agree with the global trend (in this case, a movement approximately downward and left at 30 degrees from the horizontal direction). After each iteration, mismatches are removed to give a correspondence set with less bad corresponding feature points. Figure 6 shows the relationship between the number of iterations with the number of correct correspondences and mismatches respectively. Both Fig. 5 and 6 agree that after 10 iterations, no mismatches could be found, leaving the

Fig. 7. Rectified images of a children's playground, with raw images above and rectified images below.

remaining points to be the best matches for estimating F.

3) Calculating F using the 8-point algorithm and RANSAC: We use the 8-point algorithm to find the fundamental matrix F [13]. To select the best eight-point set to compute an accurate matrix F we implemented the random sample consensus technique (RANSAC) [14], described in [18]. Here, eight correspondences are randomly selected at each step in order to calculate a tentative fundamental matrix using the 8-point algorithm [13]. The tentative matrix F is then used in the equation $\mathbf{x}_r^\top F \mathbf{x}_l = 0$ to estimate how many correspondences fit the generated model within a small margin of error. Correspondences which lead to large error values are rejected from the test set. All matrices F that have an error value smaller than a threshold were stored and this is repeated many times to obtain the best solution of F.

From F the rectified image pair is produced by aligning pixels located on the same epipolar lines. After rectification the re-sampled images will be horizontally aligned and we can draw parallel lines across the stereo pair to visually check this alignment. Figure 7 shows an example rectification result.

B. Stereo matching

A number of different stereo matching algorithms have been implemented in our system: simple SAD/SSD window-based matching [19], semi-global algorithms such as Symmetric Dynamic Programming Stereo (SDPS) and its variants [20], [18], [21], [22], as well as 1D Belief Propagation (BP) [23].

Each algorithm has different memory and processing requirements and our application can automatically choose to either compute the result directly on the user's computer, or transfer the images to a more powerful server for remote computation.

The most suitable approach for real-time stereo is an algorithm that can be easily scaled up in quality when faster hardware becomes available. Due to this, the Semi-global matching (SGM) algorithm, first proposed by Hirschmüller [24], was chosen as the default running algorithm. This algorithm is based around multiple 1D dynamic programming optimisations in different scans through the disparity cost volume. This algorithm is implemented on the GPU using Nvidia's CUDA API and runs remotely on our CUDA server. Using remote computational resources allows the web application to be used on platforms with restricted computational capabilities such as mobile phones or laptop computers, where transmitting the data to a more powerful remote server to handle complex computations is typically better.

The computational complexity of the algorithm is $O(WHd_{range})$ [24], where W, H are the dimensions of the input images and $d_{range} = d_{max} - d_{min}$ is the disparity range. Its regularisation parameters control how smooth the disparity volume should be and act to remove noise. When these parameters are set to zero the algorithm functions as a simple winner takes all (WTA) approach. With a single optimisation pass along scan-lines, SGM performs as a traditional dynamic programming stereo algorithm and our implementation supports up to 8 passes. This scalability allows a range of GPUs to be supported. On our test computer, an Intel Core i7-2600 Quad Core 3.4 Ghz, 16 Gb RAM and Nvidia GeForce GTX580 graphics card, the SGM algorithm was capable of $\approx$ 25 fps with 8 passes and up to $\approx$ 85 fps running as a traditional dynamic programming stereo algorithm for test images of 640×480 pixels.

C. Autostereogram creation

Once a depth map has been created the user has the option to choose from a number of texture patterns to generate the autostereogram. Currently the autostereogram algorithm has been implemented in both standard and GPU based code. Conceptually, for viewing an autostereogram, both eyes should focus on a point behind the screen so that the repeating texture patterns match on top of each other at the zeroth depth plane.

The 3D depth of any point is dependent on the disparity of correspondences, in this case, for a depth map normalised to the range $[0, 255]$, the maximum disparity is when $depth = 0 = black$ and the minimum disparity is when $depth = 255 = white$. We construct an autostereogram so that all points of the depthmap are represented. As a requirement, each pixel in the depthmap corresponds to two points on the autostereogram and these two points act as virtual conjugate pixel projections in 3D and must have the same colour.

The algorithm for autosteregram generation is as follows:

1) Start autostereogram construction horizontally from left to right. For each scan-line, keep two pointers $A = 0$ and $B = pattern\ width$, initial depth = 0 = black.
2) Concurrently read the depthmap horizontally from left to right and for every point:
   a) If $current\ depth == previous\ depth$ then make the colour at A and B the same by copying the colour of point A to point B and moving pointers A, B both to the right by a one pixel increment.
   b) If $current\ depth > previous\ depth$, move pointer A to the right by n pixel increments, where

n is the depth difference between the current and previous depths, the position of B stays the same - this creates monocular points visible only to the left eye. Eventually pointer A may reach the same position of B when the depth value reaches $255 = white$.

c) If $current\ depth < previous\ depth$, move B to the right by n pixel increments, where n is the depth difference between the current and previous depths, while A stays the same. In the same manner, this creates right monocularly visible points.

3) After this process a rudimentary autostereogram has been constructed, but side effects may occur because monocular points updated by pointer A have changed some values assigned by B. In this situation, some left monocular pixels are viewed by the right eye and depending on their colour, may be matched in the brain to create an incorrect 3D illusion. Therefore we process the autostereogram again, but from right to left to make sure that all visible points of the depthmap have correspondences with the same colour value. This is achieved by copying the value at point B to point A for a particular disparity correspondence.
4) At this point autostereogram creation complete.

IV. Application Results

A number of features are provided by our online application, these are detailed in the following subsections.

A. Autostereogram from stereo image pairs

Our online platform can generate depthmaps from any of five input types: pairs of left and right images, a combined left and right image (e.g. the .jps file format), an anaglyph image, an MPO image (Fujifilm 3D photo format) or a stereogram image. The depthmap can then be passed to our online autostereogram generator to create the final autostereogram image. An example of this novel feature is shown in Figure 1, generated from a pair of unaligned photos of a wooden roof.

B. Live autostereogram video from an online Minoru stereo webcam

We have also successfully built an on-the-fly autostereogram generation system, capable of real-time stereo matching on a video stream from a Minoru stereo web-cam, an example result is shown in Fig. 8. Through this novel online system we are able to give users the opportunity to interact with real-time stereo reconstruction and dynamic autostereograms.

The Minoru webcam is connected to the internet and it remotely sends its stereo image pair to our CUDA server. On the server a depth map is created and can be returned to the users at the client side for viewing in real-time. Alternatively, we can return a live autostereogram stream which the user can view using a standard autostereogram viewing technique. This method removes the need of for specialised 3D viewing devices on the user side. It also lets the user evaluate the depth of a scene in a meaningful way opposed to just a gray scaled depthmap, which can be unintuitive when one has had little experience viewing such maps. The current bottleneck of the system is the transmission of frames across the internet (at $\approx$ 2 fps).

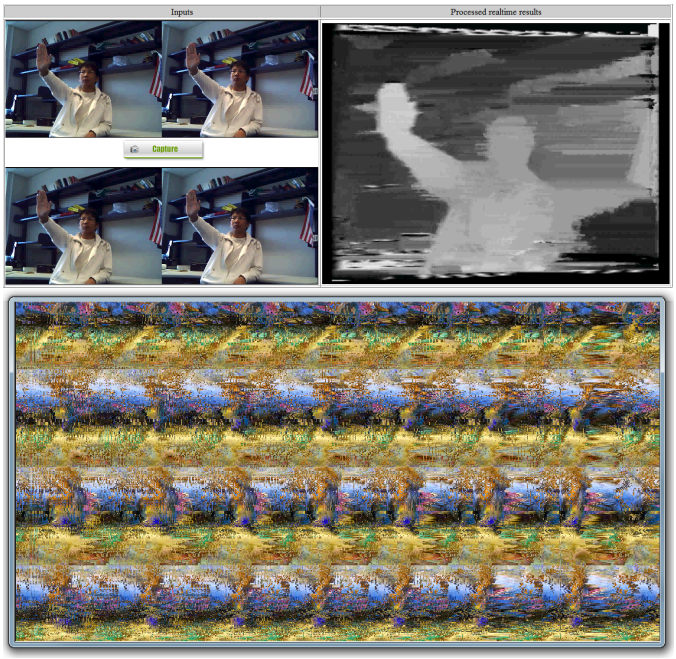

Fig. 8. Screenshot of our live stereo-matching web page includes: raw live video from a local site at 15 fps with selected frame sent to the server at $\approx$ 2 fps (top-left), generated depth-image (top-right) and generated stereogram (bottom) also at $\approx$ 2 fps.

C. Generating a depthmap from an autostereogram

We alternatively provide back reconstruction from an autostereogram to a depthmap. Here, the autostereogram is separated into its left and right image components and stereo matching is performed to detect disparities:

- Detect the width, p, of the texture pattern of the autostereogram by finding the repetition distance using a window-based SAD algorithm.
- Remove a vertical region with width = p on the right of the autostereogram to form the left stereo image and do the same process on the left side of the original autostereogram to form the right stereo image.
- Run stereo matching on the image pair with disparity range set to [0, p].
- Visualise the depthmap as the result.

D. Dynamic guidance using anaglyph images to help viewing autostereograms

In the anaglyph viewing technique, red/cyan glasses are used to filter the red and cyan channels of a colour image and guide the eye viewing directions, making the point of convergence be in front or behind the viewing plane. Generally, anaglyph images are constructed so that the convergence point is just slightly behind the screen; making it quite comfortable for most people to experience the 3D illusion. In contrast,

to view an autosteregram properly, the point of convergence needs to be set to a point far behind the screen, which is not easily achieved by inexperienced viewers.

Therefore, we have designed a process to use red/cyan glasses to slowly guide the point of convergence of a viewer to a correct location for viewing the autostereogram. Here the user starts with an easy to view anaglyph, we then employed an HTML5 script to slowly move the red and cyan pixels away from each other. In doing so the convergence point is slowly moved toward a point further behind the screen. When the desired point is reached the anaglyph image is slowly substituted with the autostereogram and viewing a 3D illusion is generally achieved.

V. Conclusion and future work

We have presented an online system designed to be a portable way to create on the fly, personalised autostereogram images using computational stereo vision techniques to reconstruct depth from a user provided image pair. The system is available online on the Internet and usable on the majority of personal computers around the world without much preparation. The computational load of stereo reconstruction can be moved to a remote server which leverages the power of GPU computation. In effect the system can operate in real-time and provides a number of benefits over other, currently available online autostereogram systems. The most important of which is that the user can generate their own stereograms based on real scenes by uploading a stereo image pair taken from any standard digital camera. All of the preprocessing is handled automatically, such as the important image rectification step, and this makes the system intuitive for new users, allowing a wide audience to play with both stereo reconstruction from images that they have taken, and also autostereogram generation. Not only does the system have entertainment value, but it could also be used as an easily accessible platform for human perceptual experiments conducted online.

Currently our application requires users to select a pattern image for autostereogram creation. For 3D reconstruction by stereo matching the depth data is often overlaid with a texture map derived from the input image data (often the left image is used for texturing). Subsequently, for a generated autostereogram it would be nice to have a pattern image that reflects the colour content of the original images its depthmap was derived from. As future work we will investigate such automatic pattern generation. Additionally, addressing the bottleneck of transmitting real-time autostereograms at frame-rates greater than 2 fps will be investigated.

References

[1] K. Minev and L. Likova, "Autostereograms as a research tool in stereoscopic vision: Interactions between some cues in perception of motion-in-depth," in *Procceedings of European Conference on Visual Perception 28 (ECVP99)*, Trieste, Italy, August 1999.

[2] L. T. Likova and C. W. Tyler, "Spatiotemporal relationships in a dynamic scene: stereomotion induction and suppression," *Journal of Vision*, vol. 3, no. 4, 2003. [Online]. Available: http://www.journalofvision.org/content/3/4/5.abstract

[3] W. Skrandies, "Assessment of depth perception using psychophysical thresholds and stereoscopically evoked brain activity," *Documenta Ophthalmologica*, vol. 119, pp. 209–216, 2009, 10.1007/s10633-009-9202-9. [Online]. Available: http://dx.doi.org/10.1007/s10633-009-9202-9

[4] "Easy stereogram builder." 2011. [Online]. Available: http://www.easystereogrambuilder.com/3d-stereogram-maker.aspx

[5] "Online stereogram generator." 2011. [Online]. Available: http://www.eyetricks.com/stereograms/onlinetools/stereocreator.htm

[6] "Flash gear stereo." 2011. [Online]. Available: http://www.flash-gear.com/stereo/

[7] "Stereogram creation packages." 2011. [Online]. Available: http://stereogram.qarchive.org/

[8] 3Dimka, "3Dimka art gallery." 2011. [Online]. Available: http://3dimka.deviantart.com/gallery/?offset=96

[9] C. Tyler and M. Clarke, "The autostereogram," in *SPIE Stereoscopic Displays and Applications*, vol. 1258, 1990, pp. 182–196.

[10] A. Fusiello and L. Irsara, "Quasi-euclidean uncalibrated epipolar rectification," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. IEEE, 2008, pp. 1–4.

[11] D. Papadimitriou and T. Dennis, "Epipolar line estimation and rectification for stereo image pairs," *Image Processing, IEEE Transactions on*, vol. 5, no. 4, pp. 672–676, 1996.

[12] A. Fusiello, E. Trucco, and A. Verri, "A compact algorithm for rectification of stereo pairs," *Machine Vision and Applications*, vol. 12, no. 1, pp. 16–22, 2000.

[13] R. Hartley, "In defence of the 8-point algorithm," in *Proc. Fifth International Conference on Computer Vision*. IEEE, 1995, pp. 1064–1070.

[14] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[15] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, jun. 1994, pp. 593 –600.

[16] F. Nielsen, "Visual computing: Geometry, graphics, and vision (graphics series)," 2005.

[17] J. Bouguet *et al.*, "Pyramidal implementation of the lucas kanade feature tracker description of the algorithm," *Intel Corporation, Microprocessor Research Labs, OpenCV Documents*, vol. 3, 1999.

[18] M. Nguyen, G. Gimel'farb, and P. Delmas, "Web-based on-line computational stereo vision," in *International Conference Image and Vision Computing New Zealand*, nov. 2008.

[19] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: theory and experiment," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 9, pp. 920 –932, sep 1994.

[20] G. Gimel'farb, "Stereo terrain reconstruction by dynamic programming," *Handbook of Computer Vision and Applications. Signal Processing and Pattern Recognition*, vol. 2, pp. 505–530, 1999.

[21] M. Nguyen, G. Gimel'farb, and P. Delmas, "Stereo vision: A Java-based online platform," in *IAPR Conference on Machine Vision Applications*, May 2009.

[22] M. Nguyen, "Web-Based On-Line Computational Stereo Vision," Master's thesis, The University of Auckland, New Zealand, Computer Science department, 2008.

[23] R. Gong, "Belief Propagation Based Stereo Matching with Due Account of Visibility Conditions," Master's thesis, The University of Auckland, New Zealand, Computer Science department, 20011.

[24] H. Hirschmller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2005, pp. 807–814.