Novel Web-based Online Autostereogram Creation using GPU Stereo Vision

Minh Nguyen, Roy Sirui Yang, Alexander Woodward, Patrice Delmas, and Georgy Gimel'farb
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 GPUs, the system has the capability to process video streams, creating depthmaps and converting them into autostereogram videos at real-time framerates transmitted over the internet. For ease of use, the system provides automatic image rectification for the user provided stereo image pairs. These novel features place the system ahead of current online alternatives and allows for a wide variety of users to experience stereo reconstruction and autostereogram generation in a quick and easy manner.**

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 single-image stereogram (SIS), that consist of a repeating pattern that has small variations across itself which are determined by the 3D scene (given as a depth map) that the autostereogram is designed to show.

Since the images themselves 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 [] as they provide a straightforward means 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 stereovision in humans []. In order to study stereo-vision deficiency in humans, Skrandies [] 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 [1], [2], [3], [4]. 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.

In this work we present a complete and novel web-based system for autostereogram creation with a number of benefits over the other available online applications. The benefits include:

1) The 3D model to be hidden in the autostereogram can be generated from a computational stereo 3D reconstruction of a user provided input image pair, rather than choosing from a selection of given models. Depth maps can be created on the fly by providing a stereo image pair 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; currently implemented using a Minoru webcam online.
3) Our system also lets users upload a autostereogram and using stereo-matching algorithm to generate the depth map that was associated with the autostereogram. This also allows those who have trouble seeing autostereogram to find out what they are supposed to see in the stereogram.
4) The system can also turn autostereograms into anaglyphs, viewable by red-cyan glasses, for users that have difficultly in seeing autostereograms.

Additionally, since this project is web-based, it is open to the public and available to anyone from around the world who has Internet access.

II. Background

A. Autostereogram description

The idea of autostereogram was first introduced by [6], where Tyler et al. demonstrated the ability to create depth by using patterns consisting of random dots. Such autostereograms can contain an unlimited range in 3D depth and can create 3D in depth plane both closer and further away from the display depth plane.

The simplest type of autostereogram consists of horizontally repeating patterns known as *wallpaper autostereograms*. When viewed with proper vergence of the eyes, the repeating patterns appear to float above or below the background, thus creating a 3D illusion. Another type of autostereogram involves scenes

that do not necessarily repeat themselves, hence these slightly more complex autostereograms can display 3D scenes of single objects.

Autostereograms are similar to normal stereograms, except that they can be viewed without a stereoscope. A stereoscope presents two separate 2D images of the same object, often taken from slightly different angles, separately to the two corresponding eyes, allowing the brain to reconstruct the original object via binocular disparity. 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 an autostereogram can be viewed: wall-eyed and cross-eyed. Wall-eyed viewing requires that the two eyes adopt a 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

On the Internet, there are many applications for autostereogram viewing and most allow the user to create stereograms 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 [1]. Stereocreator [2]] is another website that allows the users to create autostereograms of words of their choice. Another advanced website [3] 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 [4].

All the applications mentioned above allowed users to choose inputs as depth maps, either preset or created by the user, then create autostereograms as the output. As mentioned in the introduction, our proposed application presents a number of benefits over competing solutions, the most important of which is it allows users to reconstruct depth maps, from their uploaded stereoscopic images they have captured via camera to create their own personal autostereograms.

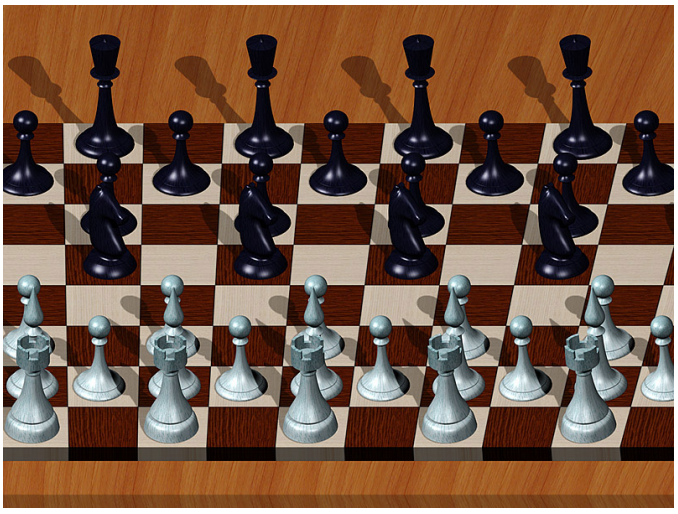

Fig. 1. An example of wall paper autostereogram showing 3D scene of a chess board [5].

III. Design and Implementation

A. Input from single-sensor camera and image rectification process

Our system allows users to upload and process images from a conventional single-sensor camera using computational stereo vision algorithms. Therefore, the process of image rectification is crucial for a high quality depth map to be calculated. Rectification transforms two images into a stereo pair with canonical stereo geometry, where feature points are aligned on the same horizontal scan line and the geometry's epipoles existing at horizontal infinity.

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 resample a stereo image pair to conform to the canonical stereo geometry. Importantly, image rectification reduces the correspondence search of stereo reconstruction algorithms from two dimensions to one dimension and is a key assumption in practically all stereo reconstruction algorithms.

For our system, 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 of the camera parameters. To achieve this, several methods have been proposed, e.g. [8], [9], [10].

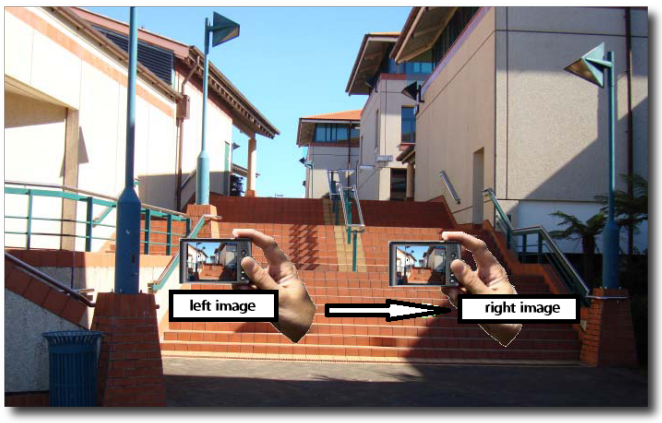

Fig. 2. Depiction of how to take a stereo photo pair from a single camera, suitable for use with the system.

Alternatively if the pose and internal parameters of the cameras were known, i.e. if the cameras were calibrated, then in principle it is possible to directly compute the transformation required to rectify the two images. However, with images acquired by arbitrary cameras, it is often difficult to determine the calibration parameters of the camera such as focal length; and some self-calibration methods require strict constraints to obtain good estimates. For example, the method proposed in [7] requires images with to be absent of skew.

Figure 3 depicts the image rectification procedure which rectifies the left and right images by estimating the fundamental matrix F. This is achieved through the 8-point algorithm [11] on a set of user-defined or automatically selected corresponding points. RANSAC [12] is further used to ensure that the feature points selected to estimate F are reliable.

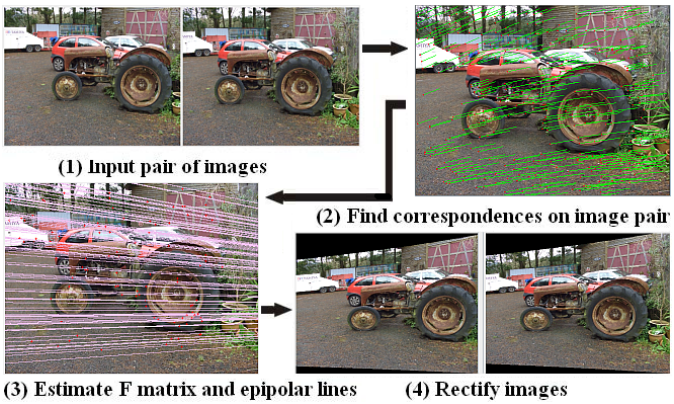

Fig. 3. The process of image rectification: (1) inputs from a single camera, (2) detected feature points on right image - red dots are the good features to track and green lines are feature tracking paths between left and right images, (3) epipolar line estimation; purple lines represent epipolar lines on Fig. 4. Feature point tracking after 1, 3, 5 (top row) and 7, 9, 11 (bottom row) iterations of mismatch removal - shown on the left image of a fence scene. Red dots are features to track and yellow lines are feature tracking paths between the left and right images. Finally, (4) image alignment.

1) Selecting reliable features: The KLT feature tracking algorithm by Shi and Tomasi [13] 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 - we chose $n = 500$, which works best for our target image resolution ($\times$ pixels) and processing complexity - in the stereo images are selected as candidates for correspondence assignment. Because image noise can lead to false feature point detection, Gaussian filtering [14] was first applied to the images.

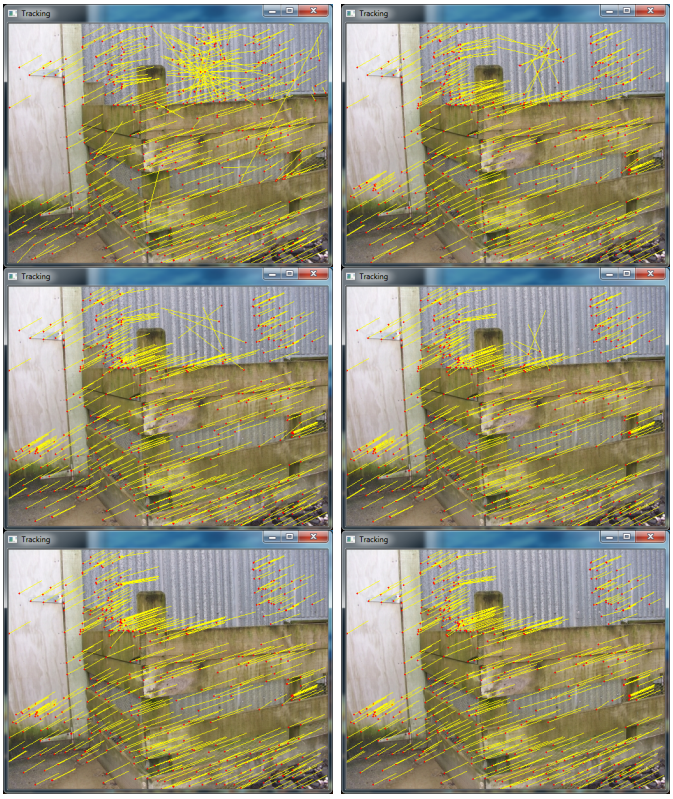

Fig. 4. Feature point tracking after 1, 3 (top row), 5, 7 (middle row) and 9, 11 (bottom row) iterations of mismatch removal - shown on left image of a fence. Red dots are features to track and yellow lines are feature tracking paths between left and right images

2) Assigning point correspondences between left and right images: After the feature point detection process in both images, a good set of feature points should now be collected. The following task aims to collect as many correspondences (conjugate pixel pairs) as possible. We used the Lucas-Kanade optical flow in a pyramid [15] tracking algorithm to track feature points between the left and right images. Feature points being tracked are searched for in the other image by using conventional window-based correlation matching with a small window size. Only feature points with small distances around the two candidate points are explored to reduce computation time and the possibility of false matching (in other words, the actual two corresponding points are assumed to be close to the selected candidate positions).

In order to remove mismatches among the set of correspondences, a number of iterations of forward/backward tracking are carried out. Potential matches are rechecked with all adjacent neighbours to make sure that the selected pairs correctly correspond to one another. In other words, features from the left image are tracked in the right image, and then the same feature points in the right image are used to track points in the left image. At the end of each iteration, well matched points should return back to their original positions and mismatches should not and will be removed. This process is repeated for a number of iterations until it reaches a stable

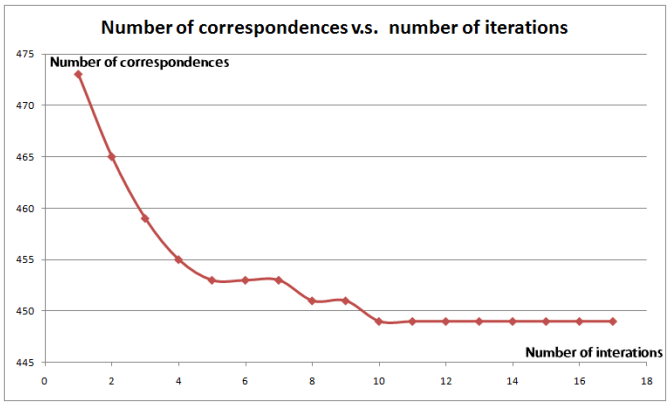

Fig. 5. Trend of number of correspondences after 17 iterations applied on the fence image pair shown in Fig. 4.

Fig. 6. Rectified images of a children's playground, with raw images above and rectified images below.

state where all points are backward tracked correctly. Fig 4 show six of the feature tracking results with 1, 3, 5, 7, 9, and 11 iterations on a pair of images acquired by a Casio Exilim EX-P505 camera.

Notice in Fig. 4, mismatches will 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. As can be seen in Fig. 5, which shows the relationship between the number of iterations with the number of correct correspondences and mismatches respectively, both diagrams agree that after 10 iterations, no mismatches could be found, leaving the remaining points to be the best matches for estimating the fundamental matrix F.

3) Calculating F using the 8-point algorithm and RANSAC: We use the 8-point algorithm to find the fundamental matrix F - the full algorithm being described in [11]. For each set of eight corresponding points, a fundamental matrix can be calculated; the algorithm requires that these eight correspondences are as accurate as possible and lie far apart from each other within the image. In fact, if these eight points are located within a small distance from each other, only a subregion of the image will be properly rectified. Additionally, any small error in feature correspondences can lead to a considerable change in the F matrix's values which can further decreases the quality of image rectification. Subsequently, it is necessary to find a solution that selects the best eight-point set to compute an accurate matrix F.

To achieve this, we have implemented the random sample consensus technique (RANSAC) [12], described in [16]. Here, eight correspondences are randomly selected at each step in order to calculate a tentative fundamental matrix using the 8-point algorithm [11]. 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 testing set. All matrices F that have an error value smaller than a threshold were stored and this process was repeated many times to return the best solution of F.

Despite working well for correspondence sets that contains a large number of mismatches, RANSAC is significantly dependent on the chosen threshold value. The rectified image pair is produced by aligning pixels located on the same epipolar lines defined by F. After rectification the resampled images should be horizontally aligned and we can draw parallel lines across the stereo pair to visually check this alignment. Figure 6 shows an example of the rectification result.

B. Stereo matching process

A number of different stereo matching algorithms have been implemented within our system: simple SAD/SSD window-based matching [19], semi-global algorithms such as Symmetric Dynamic Programing Stereo (SDPS) and its variants [20], [16], [21], [22], as well as 1D Belief Propagation (BP) [23].

Though there are many algorithms that can be used in our system, each algorithm has different memory and processing requirements. 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. By default, we chose 1D BP as the algorithm to reconstruct the 3D scene from the rectified stereo image pair. This algorithm is implemented for 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.

Binocular stereo involves the recovery of depth information from a pair of cameras viewing the same scene. Consequently, a stereo algorithm involves the identification of conjugate points in the stereo image pair. The most suitable approach for real-time stereo is an algorithm that can be easily scaled up in quality when faster hardware becomes available. In

light of this, the Semi-global matching (SGM) algorithm, first proposed by Hirschmüller [**?**], was chosen. This algorithm is based around multiple 1D dynamic programming optimisations in different scans through the disparity cost volume.

1) Pixelwise cost calculation: A disimmilarity measure (cost) $C(x, y, d)$ is taken between each pixel grey-value at $\mathbf{p} = (x, y)$ in the base (left) image, $I_b(\mathbf{p})$, and at $\mathbf{q} = (x-d, y), d \in [d_{min}, d_{max}]$ in the match (right) image, $I_m(\mathbf{q})$. This measure is taken as the sum of dissimilarities within local matching windows around $\mathbf{p}$ and $\mathbf{q}$ (of size $M \times N$), appropriate sizes were empirically found to be in the range $M, N \in [1, 15]$. Tested measures included the Birchfield and Tomasi sampling insensitive cost measure C_{BT} [**?**], and the sum of absolute pixelwise differences (SAD), $C_{SAD}(x, y, d)$. The computationally lightweight SAD measure was chosen for this work.

2) Optimisation step: For a particular scan direction $\mathbf{v}$, the optimised cost $L_{\mathbf{v}}(\mathbf{p}, d)$ for a pixel position $\mathbf{p}$ and disparity d is recursively given as:

$$\begin{aligned} L_{\mathbf{v}}(\mathbf{p}, d) = C(\mathbf{p}, d) + \min(&L_{\mathbf{v}}(\mathbf{p} - \mathbf{v}, d), \\ &L_{\mathbf{v}}(\mathbf{p} - \mathbf{v}, d-1) + P_1, \\ &L_{\mathbf{v}}(\mathbf{p} - \mathbf{v}, d+1) + P_1, M_{\mathbf{p},\mathbf{v}} + P_2) - M_{\mathbf{p},\mathbf{v}} \end{aligned} \tag{1}$$

where $M_{\mathbf{p},\mathbf{v}} = \min_i L_{\mathbf{v}}(\mathbf{p} - \mathbf{v}, i)$ is the minimum matching cost for the previous pixel position, $\mathbf{p} - \mathbf{v}$. The regularisation parameters, P_1 and P_2 ($P_1 \leq P_2$), are set with respect to local matching window size since pixel-wise costs are summed. Costs $L_{\mathbf{v}}$ are summed over directional scans through the cost volume:

$$S(\mathbf{p}, d) = \sum_{i=1}^{n} L_{\mathbf{v_i}}(\mathbf{p}, d) \tag{2}$$

where n is the number of scan directions and the upper limit for S is $S \leq n(C_{max} + P_2)$, here C_{max} can be set to an arbitrary 'large' value, dependent on an implementation's primitive data type. Finally, the disparity for pixel $\mathbf{p}$ can be chosen by taking the minimal aggregated cost of the column $S(\mathbf{p}, *)$, doing this for all pixels generates the scalar disparity map $D(x, y)$, $d \in [d_{min}, d_{max}]$.

The computational complexity of the algorithm is $O(WHd_{range})$ [**?**], where W, H are the dimensions of the input images and $d_{range} = d_{max} - d_{min}$ is the disparity range. Regularisation parameters P_1 and P_2 control how smooth the disparity volume should be and act to remove noise. When $P_1 = P_2 = 0$ the algorithm functions as a simple winner takes all (WTA) approach. With a single optimisation pass along scanlines, SGM performs as a traditional dynamic programming stereo algorithm. The number of optimisation passes and local matching window size are the parameters that most influence computation time and our implementation supports up to 8 passes. This scalability allows a range of GPUs to be supported.

C. Autostereogram creation

Once a depth map has been created the user has the option to choose from a number of texture patterns which together are used to generate the autostereogram. Currently the autostereogram algorithm has been implemented in both standard and GPU based code. The main idea is that upon 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 to two points on the stereogram and these two points act as virtual conjugate pixel projections in 3D and must have the same colour.

The algorithm for autosteregram generation proceeds as follows:

1) Start autostereogram construction from left to right, horizontally and for each scan-line, with two pointers $A = 0$ and $B = pattern\ width$, corresponds to initial depth = 0 = black.
2) Concurrently read the depmap from left to right and for every point: 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 one pixel increment. If $current\ depth > previous\ depth$, move pointer A to the right by n pixel increments, where n is the depth difference between current and previous depths, while 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 $255 = white$. 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 value assigned by B. In this situation, some left monocular pixels are viewed by the right eye and depending on their colour, may be matched to create an incorrect 3D illusion. In order to remedy this, 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. Applications and Results

A. Stereogram from stereo pair images

We have combined the ideas and implemented a web-based online mechanismn to generate depthmap from any of the five input types: left and right images, combined left-right image, anaglyph image, MPO image and stereogram image at location http://www.ivs.auckland.ac.nz/quick_stereo. The depthmap can then be passed to a online stereogram generator for a complete stereogram embedding the depthmap. Fig. 7 show an example of stereogram generated from a pair of unligned photos of a wooden roof. Obviously, our system stereogram can be generated from real life objects rather than choosing from a number of provided depth models; this demonstrates the power and flexibility of our application over other direct competitors.

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 webcam, which is shown on 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 realtime. Alternatively, we can instead return a live autostereogram stream which the user can view using one of the standard autostereogram viewing techniques. This method removes the need of having a specialised 3D viewing device on the user side. It also lets the user evaluate the depth of a scene in a meaningful way over using just a gray scaled depthmap, which can often be unintuitive when one has had little experience viewing such maps.

C. Generate depthmap from autostereogram

It's easy to notice the importance of depthmap in practical applications and research. Up to now, it's obvious that stereogram is generated from depthmap. Interestingly, searching

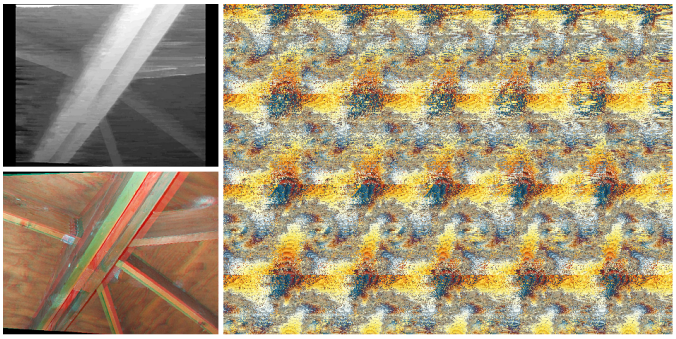

Fig. 7. An example of a stereogram (right) generated from a pair of unligned photos of a wooden roof, top-left image is the genrated depthmap and bottom-left one is anaglyph image.

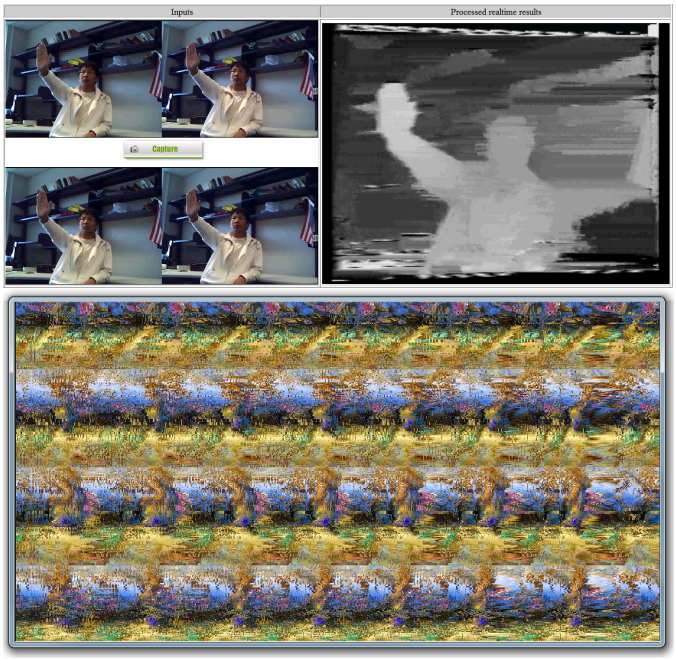

Fig. 8. Screenshot of our live stereo-matching webpage includes: raw live video from a local site at 15 fps with selected frame sent to the server at $\approx$ 2 fps (top-left), generated depth-image (top-right) and generated stereogram (bottom) also at $\approx$ 2 fps.

on the Internet, there are more results of "stereogram" than "depthmap". As a result, we alternatively introduce another related system to allow back reconstruction from stereogram to depthmap. The overall idea is to use stereo matching technique on a pair of left and right images to detect disparity of each corresponding pixel, the algorithm is shown below:

- Detect size of pattern p by finding distance of repeated pattern using window-based SAD algorithm.
- Remove a vertical region with width = p on the right of stereogram to form left stereo image and remove on the left to form right stereo image.
- Run stereo matching on the image pair with disparity range set to [0, p].
- Show depthmap as the result.

D. Dynamic guidence to view autostereogram from anaglyph images

The use of red/cyan glasses in anaglyph viewing technique is to filter the red and cyan channels of colour and from that the guide the two eye viewing directions, make the convergence point (of two optical viewing directions) to be in front or behind the flat screen/paper. Generally, anaglyph image is made so that the range of convergence point is just slightly above and behind the screen; making it quite comfortable for most people to feel the 3D illusion. While in order to view steregram properly, the point of convergence need to be set to a far point behind the screen, which is not achieved easily by inexperienced viewers.

Therefore, we have a process to use red/cyan glasses to slowly guide the point of convergence of viewer to a correct

location for viewing stereogram. In order to achieve this, we start the user with a general easy to view anaglyph, we then employed an HTML5 script to slowly move the red and cyan pixels away from each other. By this, the convergence point is slowly moved toward a further point behind the screen. When the good point is reached, anaglyph image is slowly subtituted by the stereogram and 3D illusion is generally achieved.

V. Conclusion and futher works

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.

At current stage, our application still requires users to select pattern image for stereogram to create with. The stereo matching process is a major part of this proposed system while beside depthmap, colour texture is also reconstructed to directly mapped on top of every depthmap pixel. In the future, we would like to let the pattern generated dynamically based on this texture map to fit nicely the 3D object that is hidden underneath.

References

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

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

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

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

[5] 3Dimka, "3Dimka art gallery." 2011. [Online]. Available: http://3dimka.deviantart.com/gallery/?offset=96

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

[7] M. Pollefeys, R. Koch, and L. Gool, "Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters," *International Journal of Computer Vision*, vol. 32, no. 1, pp. 7–25, 1999.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1993, pp. 593–600.

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

[15] 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.

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

[17] P. Rousseeuw, "Least median of squares regression," *Journal of the American statistical association*, pp. 871–880, 1984.

[18] P. Meer, D. Mintz, A. Rosenfeld, and D. Y. Kim, "Robust regression methods for computer vision: A review," *International Journal of Computer Vision*, vol. 6, pp. 59–70, 1991, 10.1007/BF00127126. [Online]. Available: http://dx.doi.org/10.1007/BF00127126

[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] G. M. Nguyen, Gimel'farb and P. Delmas, "Stereo vision: A java-based online platform," in *MVA2009 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.