Real-Time Stereo Vision on the VisionServer Framework for Robot Guidance

Alexander Woodward, David Berry, James Dunning

ControlVision Ltd
Unit 2, 527B Rosebank Rd, Avondale,
Auckland, New Zealand
Telephone: +64 9 828 0500
Fax:
Email: http://www.controlvision.co.nz/contacts/
Websites: http://www.controlvision.co.nz, http://www.visionserver.co.nz

Abstract

This paper introduces the VisionServer Framework, a computer vision application framework for development within a visual environment. Its functionality is demonstrated by documenting the design and theory behind a real-time stereo vision system for robot guidance. VisionServer implements the required run-time tasks of stereo image capture, image rectification, stereo correspondence and data visualisation as function blocks which can be visually manipulated within its sequence editor. In order to run in real-time, VisionServer implements stereo correspondence on a graphics processing unit (GPU). Additionally, an end-user GUI can be created within the framework, making it easy to control system settings. The framework is extendable through custom end-user written function blocks. These features promote rapid application development, code reuse, and application sharing. Results show that real-time stereo on the GPU is cost-effective and beneficial for interactive fine-tuning of a stereo algorithm. Moreover, real-time processing is important for time-critical computer vision applications.

Keywords: Binocular stereo, stereo vision, GPU, VisionServer, ControlVision

1 Introduction

Intuitive and powerful computer vision tools have become desirable as vision solutions in industry grow in tandem with research into advanced vision algorithms. The VisionServer Framework has been designed to address this need by providing a visual environment to organise computer vision and image processing algorithms into entire applications.

To demonstrate the framework's practicality within a research context, the implementation and design of a real-time computational stereo vision system for robot guidance within VisionServer is presented. The description is divided into VisionServer's implementation of fundamental 3D vision procedures as functional blocks i.e. camera calibration, image rectification, and dense stereo correspondence running on a graphics processing unit (GPU) - which now make up some of the key features of the framework - with the use of VisionServer as a design tool. This allows a user to create a vision solution through drag-and-drop system sequence design GUI creation.

VisionServers important features for the rapid design

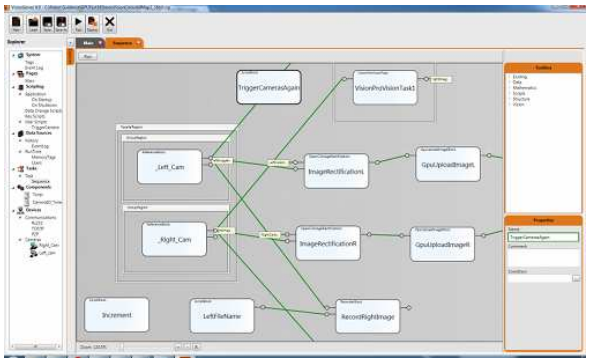


Figure 1: A screen capture of VisionServer's sequence editor, showing a subsection of the function blocks involved in the stereo vision system. The sequence is repeatedly run from left to right using an event based model coordinated with image capture.

of a computer vision application include:

- Intuitive function block sequence format allowing for the visual design of applications - blocks for data acquisition and communications, vision, math and logic, scripting, and

data logging - Fig. 1 shows an example of what the function blocks look like within VisionServer.

- Extendable through custom function blocks that can be written in the .NET programming framework.
- Supports multiple vision libraries including OpenCV [1] and Cognex VisionPro along with image processing devices such as Cognex's In-Sight technology [2].
- GPU based computer vision through function blocks that exploit the parallelism of the latest graphics cards.
- Supports a wide variety of industrial cameras and interface technologies: GigE, FireWire, Cameralink, USB, etc..
- GUI components allowing for the drag-and-drop visual design of user interfaces.
- Architectural scalability for multiple sensors, sequences and clients.
- Performance management for scheduling and running on multi-core systems.
- Optimised for real-time applications.
- Data logging and database connectivity with a built in SQL database.
- Access control features for security.
- Once an application has been developed a separate run-time can be deployed for end users.

VisionServer has already found success in many commercial vision solutions, examples of these and more information can be obtained by visiting ControlVision's product website [3].

We now detail the structure of this paper: firstly, related work into real-time computational stereo vision and commercial vision hardware and software is presented in Sec. 2. System hardware is then given in Sec. 3. Sections 4-6 describe the theoretical aspects of this work, focusing on camera calibration, the binocular stereo setup, image rectification and dense stereo correspondence for disparity map generation. It is noteworthy to remember that within VisionServer all of this theory is implemented in code as function blocks. Next, a brief description in Sec. 7 is given to how VisionServer can be used to organise these blocks and create a custom GUI, all without programming. A selection of visualisations of recovered depth data is presented in Sec. 8, along with commentary on the results.

2 Related Work

Binocular stereo vision is one of the most active research areas in computer vision. Subsequently, a large number of dense stereo correspondence algorithms have been created with various trade-offs between accuracy and execution time. Given a pair of stereo images, the stereo correspondence process creates a depth map of the scene; this depth information can then be used to guide a robot to a desired location when it is calibrated with the imaging system.

Stereo correspondence algorithms and related reconstruction approaches such as augmented binocular stereo with active illumination have been investigated in works such as Woodward et al. [4] and the well known Middlebury stereo website [5]. Findings have shown that many of the most advanced 2D optimisation approaches are unsuitable for real-time implementation - a general approach for real-time has thus been to investigate simpler optimisation schemes such as dynamic programming. By acquiring images at video rates the design of 3D video scanners based on stereo vision has emerged; this approach has been used to reconstruct highly realistic facial animations [6].

A quick glance at the literature shows that the majority of stereo vision systems reside in the non-commercial, academic domain. Despite this, a number of commercial stereo vision systems are available, but share a common trait in only using very simple stereo correspondence algorithms running in hardware: the Focus Robotics' PCI nDepth Vision System [7] and the Point Grey Research's Bumblebee Stereo Vision Camera System [8] both use the sum of absolute differences (SAD) approaches. Tyzx Inc.'s Deep Sea Product line, uses the Census matching algorithm [9], a hardware efficient yet relatively simple algorithm, and the Surveyor Corporation's SVS system [10] provides a hardware platform but no inbuilt stereo processing. These examples reflect the computational burden imposed by dense stereo correspondence algorithms - therefore we have looked at leveraging the parallelism of the GPU to provide real-time stereo vision, an area which is now quite active e.g. [11, 12] are good examples. Other hardware platforms such as stereo vision on FPGAs has also being investigated [13], but the benefits of GPUs lie in their cheaper cost, ease of programming and rapidity of compilation, and in turn scalability through code flexibility, and the rate at which newer and faster cards appear on the market (a new generation of card appears roughly every year). Lastly, the aforementioned commercial vision systems do not come with a complete visual framework like VisionServer.

3 System Hardware

System hardware design consists of two Basler Ace acA1300-30gm video cameras mounted and aligned on aluminum plates to conform as near as possible to standard stereo geometry (described in Sec.4.1). Any remaining misalignment is accounted for in software through image rectification, see Sec. 5.

Table 1: Hardware Platform

Cameras:	Two Basler Ace acA1300-30gm area scan cameras, capable of 1296 x 966 pixel images. Gig-E 1/3 sensor.
Camera lenses:	Fujion TF4DA-8 4 mm f/2.2 C-Mount.
Computer:	Intel Core i7 960 @ 3.2 GHz, 6.00 GB RAM.
Operating System:	Windows 7 64-bit.
Graphics card:	NVIDIA GTX 470 with 1280 MB RAM.

The Basler Ace cameras connect to a desktop computer running VisionServer through gigabit Ethernet and are powered using Power over Ethernet (PoE).

4 Camera Model and Binocular Stereo

VisionServer's geometric camera model is based on central projection, defining a camera's intrinsic and extrinsic parameters and lens distortion is modelled using low order polynomials. To estimate these camera parameters a camera calibration procedure must be performed, currently the Tsai and OpenCV calibration approaches are supported (see [14, 15] and [1, 16] for details).

Intrinsic parameters determine pixel coordinates of an image point, given in the camera reference frame. Extrinsic parameters describe the location and orientation of the camera in a world coordinate frame. The projection of a 3D point, $\mathbf{P} = (X, Y, Z, 1)^T$, represented in homogeneous coordinates in a world coordinate frame, into a point $\mathbf{p} = (u, v, 1)^T$, in pixel coordinates is given by:

$$s\mathbf{p} = \mathbf{K}[\mathbf{R}|\mathbf{T}]\mathbf{P}, \tag{1}$$

the scale factor s exists due to the homogeneous representation of a pixel position.

The rigid transformation of a camera in a scene is described by the extrinsic camera parameters contained in the joint rotation-translation matrix $[\mathbf{R}|\mathbf{T}]$, where $\mathbf{T} = -\mathbf{RO}$ and $\mathbf{O}$ is the camera optical centre defined in the world coordinate frame - this is often referred to as just the matrix of extrinsic parameters. The camera matrix, $\mathbf{K}$, or

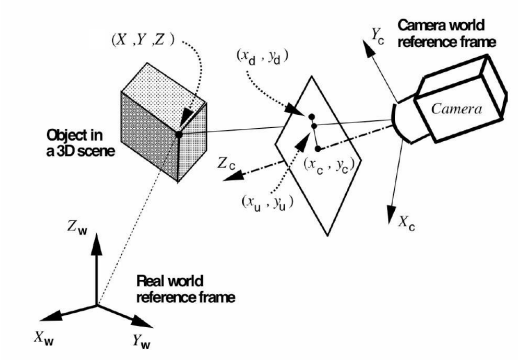


Figure 2: The geometric camera model with distortion correction [17].

matrix of intrinsic parameters contains focal length scaled by pixel scale factors, (f_x, f_y) and the principal point $\mathbf{c} = (x_c, y_c)$.

The camera model in Equ. (1) assumes linearity to be an accurate representation of the imaging process. However, in reality there exist deviations from this - many of these are systemic within the lens optics and are dealt with by a polynomial distortion model that accounts for deviations from the ideal pinhole camera model [18]. Adjustment is made to the distorted image plane coordinate $\mathbf{p}_d = (x_d, y_d)$, to generate an undistorted coordinate $\mathbf{p}_u = (x_u, y_u)$. The Tsai calibration algorithm models a single order radial distortion κ_1, whereas the OpenCV calibration algorithm models up to the second order in both radial κ_1, κ_2 and tangential distortions ρ_1, ρ_2.

The full geometric camera model is depicted in Fig. 2. Once a system is calibrated, metric information of the scene from the non-metrical projective geometry can be obtained through this section's equations.

4.1 Binocular Stereo Principles

Binocular stereo involves the recovery of depth information from a pair of cameras viewing the same scene. The stereo correspondence problem involves the identification of conjugate points in the stereo image pair. Through only the parallax of conjugate points one can computationally evaluate the depth over a scene.

The vectorial difference in pixel coordinates of the projection of a point $\mathbf{P}$ into the two image planes is known as the *disparity*, $\mathbf{d}$, or parallax.

A commonly used stereo camera geometry is the standard stereo geometry (SSG), also known as the canonical stereo geometry. This describes a particular camera setup where image planes are coplanar and epipoles[1] exist at infinity. It follows

[1]Points of intersection of the baseline with the image

that the SSG has scalar disparities and the symmetric epipolar constraint applies:

$$d = |x_1 - x_2| = x_1 - x_2. \quad (2)$$

For computational efficiency, stereo correspondence algorithms assume a SSG setup and they output data in the form of a scalar *disparity map* $D(x, y)$, defined in *disparity space*, (x, y, d), given by the image spatial dimensions and disparity in the range $d \in [d_{min}, d_{max}]$.

5 Stereo Image Rectification

Stereo image rectification is the process of transforming a stereo image pair to conform to standard stereo geometry (SSG). Rectification reduces a 2D search over the image for conjugate points into a 1D search along scanlines and a quality rectification is important for getting good results. VisionServer implements a calibration based rectification approach over a feature point, fundamental matrix approach such as in [19]. This gives greater rectification precision and is suitable for cameras arranged near standard stereo geometry. Rectified images can be conceptualised as being acquired by the system after the original cameras have been rotated to a common orientation. This common orientation is calculated using the calibration parameters.

6 Depth Map Computation using the Semi-Global Matching Algorithm on the GPU

A dense stereo correspondence algorithm was used to acquire a dense disparity map, D. From previous studies, a wide range of algorithmic approaches are available, ranging from the simplest winner takes all (WTA) strategies, to 1D optimisation such as dynamic programming, to full 2D optimisation techniques like belief propagation and graph-cuts [21, 4]. Generally, the more complex algorithms have longer computation times and the most suitable approach for real-time stereo is an algorithm that can be easily scaled up in quality when faster hardware becomes available. To this end the Semi-global matching (SGM) algorithm, first proposed by Hirschmüller [22], was chosen. This algorithm is based around multiple 1D dynamic programming optimisations in different scans through the disparity cost volume. Dynamic programming without back-tracing has a very small memory footprint and only requires the previous disparity column to be stored in local memory as the algorithm progresses. The entire stereo correspondence algorithm exists as a single function block within VisionServer; the block accepts a stereo image pair and the algorithm's run-time parameters.

planes.

6.1 Pixelwise Cost Calculation

A disimmilarity measure C is taken between each pixel grey-value at $\mathbf{p} = (x, y)$ in the base image, $I_b(\mathbf{p})$, and at $\mathbf{q} = (x - d, y), d \in [d_{min}, d_{max}]$ in the match 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 $M, N \in [1, 15]$. Any suitable measure can be chosen and for the current implementation the user has the choice of the Birchfield and Tomasi sampling insensitive cost measure C_{BT} [23], or the sum of absolute pixelwise differences (SAD), C_{SAD}. The similar sum of squared differences (SSD) measure was discarded due to its empirically found, slightly worse performance compared to SAD.

Cost calculation operates under the photo-consistency assumption; that object points appear with the same intensity in both images. This is one reason why traditional stereo algorithms struggle with highly specular, reflective surfaces that will appear different in each image. To enforce photo-consistency it is important to ensure that stereo cameras are photometrically calibrated. Although some stereo algorithms can cope with pixel intensity contrast and offsets (e.g. through the use of the normalised cross correlation similarity metric), prior removal of such imaging discrepancies is preferable.

6.2 SGM Optimisation Step

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

$$\begin{aligned} L_{\mathbf{v}}(\mathbf{p}, d) = C(\mathbf{p}, d) + \min(&L_{\mathbf{v}}(\mathbf{p} - \mathbf{v}, d), \\ &L_{\mathbf{v}}(\mathbf{p} - \mathbf{v}, d - 1) + P_1, \\ &L_{\mathbf{v}}(\mathbf{p} - \mathbf{v}, d + 1) + P_1, M_{\mathbf{p},\mathbf{v}} + P_2) - M_{\mathbf{p},\mathbf{v}} \end{aligned} \quad (3)$$

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:

$$S(\mathbf{p}, d) = \sum_{i=1}^{n} L_{\mathbf{v_i}}(\mathbf{p}, d) \quad (4)$$

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 $S(\mathbf{p}, *)$.

The computational complexity of the algorithm is $O(WHd_{range})$ [22], where W, H are the dimensions of the input images and $d_{range} = d_{max} - d_{min}$ is the disparity range. Here, the number of optimisation passes and local matching window size are the parameters that most influence computation time. 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 WTA approach. With a single optimisation pass along scanlines, SGM performs as a traditional dynamic programming stereo algorithm. This scalability allows a wide generation of GPUs to be supported. Our current implementation was written in CUDA using the NVIDIA GTX 470 graphics card.

6.3 Occlusion Detection

Occlusions can be found by comparing disparity maps generated using the costs from matching the base image to the match image, D_b, and costs from match to base, D_m. The final disparity map, D, can be marked with invalid disparities, $d_{invalid}$, if the two conjugate disparity values from both maps differ by a threshold ϕ:

$$D(\mathbf{p}) = \begin{cases} D_b(\mathbf{p}) & \text{if } |D_b(\mathbf{p}) - D_m(\mathbf{q})| < \phi \\ d_{invalid} & \text{otherwise} \end{cases} \tag{5}$$

where $|\cdot|$ denotes the absolute value. Occlusion identification is an enforcement of the *uniqueness constraint*, where only one to one mappings between conjugate pixels are allowed. Using a threshold value ϕ relaxes this constraint and for practical situations it can be set as high as 8 pixels when only the most prominent occlusions are sought after.

6.4 Data Visualisation

A traditional method to visualise a disparity map D is as a height map image, I, where the disparity range is mapped to $[0, 255]$ and each pixel position is assigned a grey-value, e.g. as in Fig. 3e. The downside of this approach is that it can be hard to see the change in depth over the disparity map. To address this a colour mapping of a grey scale image, I, to the *hue* range of the HSV colour space was performed. The HSV colour space is then mapped to the RBG colour space by the function $f : (H, S, V) \mapsto (R, G, B)$. The input to f is the triplet $(sI(\mathbf{p}), 1, 1)$, with s being a scale factor to exploit the convention $H \in [0, 360]; S, V \in [0, 1]; R, G, B \in [0, 255]$. This mapping procedure can be seen in the results shown in Fig. 4.

7 Designing the System in VisionServer

The theory presented in sections 4-6 has been implemented as function blocks within VisionServer. For the designer of a computer vision application this means that a large amount of code need not be written and application creation only involves the visual manipulation of functional blocks in a drag-and-drop environment along with some scripting.

Required function blocks are chosen from a menu and dragged into the sequence editor. Their inputs and outputs can then be connected in order to construct the logical flow of the system process. Blocks for each of the theoretical components described in this paper are available for the design of the system. Additionally, variables and overall system control are handled through a GUI that was also created within VisionServer.

As shown earlier, Figure 1 shows a screen-capture of VisionServer's user interface - here function blocks are arranged and connected, demonstrating the ease of creation and how the theory presented in this paper is subsumed into VisionServer blocks.

8 Results

Figure 4 shows depth map results of the stereo vision system running a seven pass stereo reconstruction and using the colour mapping technique of Sec. 6.4. Figure 3 shows the reference left camera image that acted as the base image for these depth map results - each pixel position of the base image maps one to one with the depth map. For reference, Fig. 3e shows an example grey-scale height map, where closer points are lighter. Figure 4 shows how the colour mapping brings out the detail of the test subject's collared vest (in the first three images) and the creasing in the clothing. This detail is something that is very hard to see when using a grey-scale height map visualisation.

The input image size was 648×482 pixels, with an adjustable disparity range of $0 - 256$ units. The system was capable of running a single-pass dynamic programming stereo reconstruction at over 60 fps when the disparity range was set to 196. With seven passes, results were dramatically improved but the frame-rate dropped to around 12 fps. To gain an appreciation of the speedup that

(a)

(b)

(c)

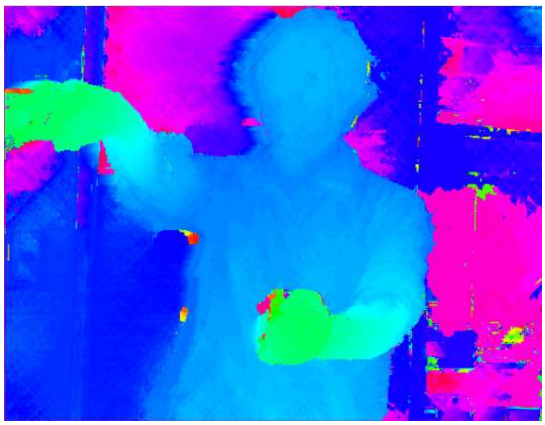
(d)

(e) Grey-valued height map of Fig. 3a

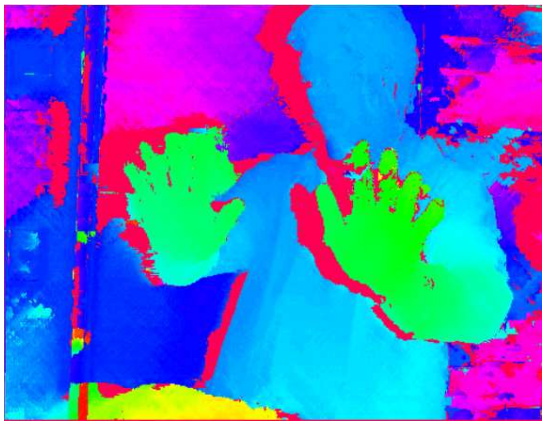
Figure 3: A selection of test images using our stereo system. These images are the base (left) images of the stereo pairs given as input into our stereo correspondence algorithm. The depth data for these test images, visualised in colour, is shown in Fig.s 4a-4d. As an example, a grey-valued height map visualisation of Fig. 3a is shown in Fig. 3e; the colour visualisation provides a lot more fidelity in change of depth to the human eye.

(a)

(b)



(c)



(d) With occlusion detection.

Figure 4: Visualisations using the depth-to-colour mapping described in Sec. 6.4. Sub-figure 4d shows an example of occlusion detection where poor disparity values are discarded and set to zero disparity (for colour reproductions of this paper, a zero disparity is shown in red); this is most easily seen around the right (physical) side of the hands, face and torso.

the GPU provides: for the seven pass algorithm a single reconstruction took about 0.08 seconds while

a comparable CPU implementation running on the same computer took approximately 7 seconds, this is roughly an 87 fold increase in speed for the GPU implementation. It is worthwhile noting that cost volume calculation took 5 seconds for the CPU implementation and it is also wise to assume that further CPU optimisations would reduce the noted speed increase, however not by an amount to disregard the benefit a GPU offers stereo vision.

Figure 4d shows a result with occlusion detection turned on - erroneous disparity values are discarded and set to zero disparity (for colour reproductions of this paper, a zero disparity is shown in red); this is most easily seen around the right (physical) side of the hands, face and torso. At the bottom of this same figure the top of a chair can be seen (in yellow).

Generally, noise appears in the background regions of these results, especially in the over-exposed window region in the right side of the images. Large homogeneous regions such as the walls were poorly reconstructed. The algorithm sometimes had difficulty delineating a person's fingers, primarily due to occlusions and also illumination variation between images. Using longer camera focal lengths, thus setting the view volume further away, could reduce large occlusions. It is also a general feature that reflective and specular objects, e.g. glass bottles, are poorly reconstructed due to their different appearance in each stereo image.

Being able to view a stereo algorithm in real-time provides a great deal of insight into how its runtime parameters affect results. It was found that the most important regularisation parameter of the SGM algorithm was P_2, which penalises large changes in depth value. To retain fine detail P_1 should be set much lower than P_2 but high enough to minimise surface noise. By adding further scans to the stereo algorithm the initial cost calculation window can also be reduced - this actually has the affect of reducing noise while also providing a slight speedup in computation time. Of course, these are general guidelines and parameters often have to be selected depending on scene content.

9 Conclusion

This paper has demonstrated how the VisionServer framework was used to develop a stereo vision system which runs in real-time on consumer hardware by leveraging the computational power of a GPU. The required theory and algorithms were implemented as function blocks within VisionServer, allowing the system to be rapidly designed in a drag-and-drop sequencing environment.

Modularising many low-level vision and image processing tasks greatly eases application design and promotes code reusability and application sharing. We see the intuitive GUI approach of VisionServer, along with the ability to write custom code and script blocks as key to creating larger and more complex computer vision applications.

The benefits of real-time stereo have also been elucidated: the interactive adjustment of a stereo algorithm's parameters provided greater insight into its operation. This made it easier to fine tune stereo parameters, validate system settings, and visually appraise the types of depth map artifacts that can occur due to occlusions and specularities. This a step up from when one had to wait seconds to hours for comparable CPU based stereo results.

Future work will involve further optimisations to the GPU stereo vision implementation. Framerates can be increased as the current top of the line NVIDIA card, the GTX 480, is roughly 25% faster than the card used in this paper (the GTX 470), and there is also the benefit of multi-GPU configurations using technologies such as SLI.

For many applications, further processing of depth data is often necessary and research into 3D feature detection and visualisation approaches will be undertaken.

Future plans for VisionServer are to continually expand the functionality of the framework and, of special interest for the vision community, the ability to share user written function blocks uploaded to the VisionServer website [3].

References

[1] Open source BSD license, "OpenCV computer vision library," Open source BSD license., 2010. [Online]. Available: http://sourceforge.net/projects/opencvlibrary/. [Accessed: Sep. 14, 2010].

[2] Cognex Corp., "Cognex Machine Vision Systems and Machine Vision Sensors," 2010. [Online]. Available: http://www.cognex.com/Main.aspx. [Accessed: Sep. 16, 2010].

[3] Control Vision Ltd., "Vision Server - Machine Vision Framework," 2010. [Online]. Available: http://www.visionserver.co.nz. [Accessed: Sep. 14, 2010].

[4] A. Woodward, D. An, Y. Lin, P. Delmas, G. Gimel'farb, and J. Morris, "An Evaluation of Three Popular Computer Vision Approaches for 3-D Face Synthesis," in Proceedings of Structural, Syntactic, and Statistical Pattern Recognition, (SSPR), China, 2006, pp. 270–278.

[5] D. Scharstein, "Middlebury Stereo Vision Page," 2007. [Online]. Available: http://vision.middlebury.edu/stereo/. [Accessed: Sep. 15, 2010].

[6] A. Woodward, P. Delmas, and G. Gimel'farb, "A 3D Video Scanner for Face Performance Capture," in Proceedings of Image and Vision Computing New Zealand Conference (IVCNZ), New Zealand, 2008, pp. 195–200.

[7] Focus Robotics, "PCI nDepth Vision System," 2008. [Online]. Available: http://www.focusrobotics.com/products/systems.html. [Accessed: Sep. 14, 2010].

[8] Point Grey Research, Inc., "Bumblebee Stereo Vision Camera System Datasheet," 2010. [Online]. Available: http://www.ptgrey.com/products/bumblebee2/bumblebee2_xb3_datasheet.pdf. [Accessed: Sep. 14, 2010].

[9] Tyzx Inc., "Tyzx Deep Sea Product Family," 2009. [Online]. Available: http://tyzx.com/products/index.html. [Accessed: Sep. 14, 2010].

[10] Surveyor Corporation, "Surveyor Stereo Vision System (SVS)," 2009. [Online]. Available: http://surveyor-corporation.stores.yahoo.net/srblstca.html. [Accessed: Sep. 14, 2010].

[11] A. Brunton, C. Shu, and G. Roth, "Belief propagation on the gpu for stereo vision," in Proceedings of the 3rd Canadian Conference on Computer and Robot Vision (CRV06), 2006, pp. 76–76.

[12] I. Ernst and H. Hirschmuller, "Mutual information based semi-global stereo matching on the gpu," in Proceedings of the International Symposium on Visual Computing (ISVC), 2008, pp. I: 228–239.

[13] K. J. J. Morris and G. L. Gimel'farb, "Intelligent vision: A first step - real time stereovision," in Proceedings of the Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS), 2009, pp. 355–366.

[14] B. K. P. Horn, "Tsais camera calibration method revisited," http://people.csail.mit.edu/bkph/articles/Tsai_Revisited.pdf, visited on February 16, 2009.

[15] R. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," IEEE Journal of Robotics and Automation, pp. 323–344, 1987.

[16] Z. Zhang, "A flexible new technique for camera calibration," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, pp. 1330–1334, 2000.

[17] G. Gimel'farb, "Camera calibration," in COMPSCI 773 ST Robotics and Real Time Control: Lecture Notes, 2002, p. 5.

[18] F. Devernay and O. Faugeras, "Straight lines have to be straight: automatic calibration and removal of distortion from scenes of structured enviroments," Machine Vision Applications, vol. 13, no. 1, pp. 14–24, 2001.

[19] A. J. Lacey, N. Pinitkarn, and N. A. Thacker, "An Evaluation of the Performance of RANSAC Algorithms for Stereo Camera Calibration," in British Machine Vision Conference, UK, 2000.

[20] R. Hartley, "Self calibration from stationary cameras," International Journal of Computer Vision, vol. 22, pp. 5–23, February 1997.

[21] P. Delmas, A. Woodward, P. Leclerq, and G. Gimel'farb, "Generation of an Accurate Facial Ground Truth for Stereo Algorithm Evaluation," in Proceedings of the International Conference on Computer Vision and Graphics (ICCVG), Poland, 2004, pp. 534–539.

[22] 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.

[23] S. Birchfield and C. Tomasi, "Depth discontinuities by Pixel-to-Pixel stereo," in Procedings of the Sixth International Conference on Computer Vision (ICCV-98), S. Chandran and U. Desai, Eds., Narosa Publishing House. New Delhi: Narosa Publishing House, 1998, pp. 1073–1080.