Vergence Using GPU Cepstral Filtering

Luis Almeida[1,2], Paulo Menezes[1], Jorge Dias[1]

[1] Institute of Systems and Robotics,
Department of Electrical and Computer Engineering , University of Coimbra – Polo II,
3030-290 Coimbra, Portugal
{paulo, jorge}@isr.uc.pt
[2] Department of Informatics Engineering, Institute Polytechnic of Tomar
2300 Tomar, Portugal
laa@ipt.pt

Abstract. Vergence ability is an important visual behavior observed on living creatures when they use vision to interact with the environment. The notion of active observer is equally useful for robotic vision systems on tasks like object tracking, fixation and 3D environment structure recovery. Humanoid robotics are a potential playground for such behaviors. This paper describes the implementation of a real time binocular vergence behavior using cepstral filtering to estimate stereo disparities. By implementing the cepstral filter on a graphics processing unit (GPU) using Compute Unified Device Architecture (CUDA) we demonstrate that robust parallel algorithms that used to require dedicated hardware are now available on common computers. The overall system is implemented in the binocular vision system IMPEP (IMPEP Integrated Multimodal Perception Experimental Platform) to illustrate the system performance experimentally.

Keywords: Cepstrum, GPU, CUDA, vergence

1 Introduction

Vergence ability is an important visual behavior observed on living creatures when they use vision to interact with the environment. In binocular systems, vergence is the process of adjusting the angle between the eyes (or cameras) so that they are directed towards the same world point. Robotic vision systems that rely on such behavior have proven to simplify tasks like object tracking, fixation, and 3D structure recovery. Verging onto an object can be performed by servoing directly from measurements made on the image. The mechanism consists of a discrete control loop driven by an algorithm that estimates single disparity from the two cameras. There are several methods to measure stereo disparities (features or area based correspondence, phase correlation based method, etc) and although some of them present better performance they were not used due to their computation requirements. Cepstral filtering is more immune to noise than most other approaches [1,2], but computing the Fast Fourier Transform (FFT) of images and inverse FFT presents some real-time challenges for the processing devices. This work describes the implementation of a real-time binocular vergence behavior using GPU cepstral filtering to estimate stereo disparities. By implementing the real-time cepstral filter

on a current graphics processing unit (GPU) using Compute Unified Device Architecture (CUDA) [4] we demonstrate that robust parallel algorithms can be used on common computers. The overall system is implemented in the binocular vision system IMPEP [25] (figure 1) to experimentally demonstrate the system performance. The main body of the cepstral algorithm, processed in parallel, consists of a 2-D FFT, a point transform (the log of the power spectrum), and the inverse 2-D FFT. The goal of the control strategy is to compensate the disparity between the cameras. Gaze holding behaviors and vergence processes are very useful for the emergent humanoid robotics area that aims to mimic humans. The following text presents the background for disparity estimation using cepstral filtering, a description of CUDA IMPEP implementation, experimental results and conclusions.

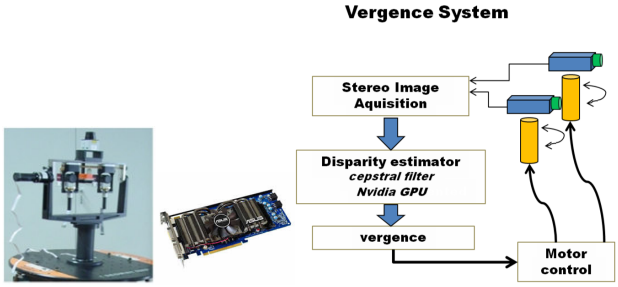

Figure 1: Integrated Multimodal Perception Experimental Platform (IMPEP). The active perception head mounting hardware and motors were designed by the Perception on Purpose (POP - EC project number FP6-IST-2004-027268) team of the ISR/FCT-UC, and the sensor systems mounted at the Mobile Robotics Laboratory of the same institute, within the scope of the Bayesian Approach to Cognitive Systems project (BACS - EC project number FP6-IST-027140). On the right it is presented an overview of the IMPEP vergence system architecture and the NVIDIA GPU used for data parallel processing.

2 Contribution to sustainability

Knowledge of the world allows the visual system to limit the amount of ambiguity and to greatly simplify visual computations. By demonstrating that computational power is available on computers at affordable costs we expect to contribute for the sustainability of computer vision complex tasks (intelligent surveillance systems, vision-guided autonomous vehicles, fingerprint/face/iris recognition, humanoid robotics, etc). The real-time cepstral filter implementation on a current graphics processing unit (GPU) using Compute Unified Device Architecture (CUDA) demonstrates that robust parallel algorithms can be used on common computers. By using the NVIDIA GPU multicore processors architecture and parallel programming we speed up the cepstral filtering algorithm more than sixteen times than on a CPU. The main body of the our GPU cepstral algorithm consists of a 2-D FFT, a point transform (the log of the power spectrum) and the inverse 2-D FFT. It takes only 0,43 ms to process an [256x256] image. The complete vergence control iterations cycle can be performed in 31ms (f=32,25Hz). The use GPU Cepstral Filtering to

perform vergence on binocular head systems is, to our knowledge, an new contribution for the state-of-art.

3 Background and related work

Animals, especially predators, that have their eyes placed frontally can use information derived from the different projection of objects onto each retina to judge depth. By using two images of the same scene obtained from slightly different angles, it is possible to triangulate the distance to an object with a high degree of accuracy. For primates like ourselves the need for a vergence mechanism is obvious. Human eyes have non-uniform resolution, so we need a way to direct both foveas at the same world point so as to extract the greatest possible amount of information about it. The human brain has an extraordinary ability to extract depth information from stereo pairs, but only if the disparities fall within a limited range. Verging on surfaces usually constrains points near the fixation point to fall inside this range [2].

Binocular systems heads have been developed in recent decades. For example, VARMA head [13], MDOF head [14], Rochester [15], the “Richard the First" head [16] and the KTH robot head [17] were capable of mimicking human head motion. More recent robot heads include the POP head [25,26] used on the Bayesian Approach to Cognitive Systems project (IMPEP)[8], the LIRA-head [18], where acoustic and visual stimuli are exploited to drive the head gaze; the Yorick head [19], and the Medusa head [20] where high-accuracy calibration, gaze control, control of vergence or real-time speed tracking with log-polar images were successfully demonstrated.

In binocular camera systems, the vergence process has to adjust the angle between the cameras, by controlling the camera’s pan angle, so that both sensors are directed at the same world point. The process must estimate the angle between the current direction of the non-dominant camera optical axis and the direction from the camera center to the desired direction (fixation point). The compensation angle is driven by continuously minimizing the binocular disparity. The IMPEP cameras do not have foveas. Even so, there are good reasons to have a low-level mechanism that maintains vergence. As Ballard and Olson argues [11,12], having a unique fixation point defines a coordinate system which is related as much to the object being observed as it is to the observer, and hence is a step in the direction of an object-centered coordinate system. Verging the eyes also provides an invariant that may be useful to higher level processes. It guarantees that the depth of at least one world point is known, even if we do not attempt stereo reconstruction in the usual sense. Additionally, by acquiring images that contain the focus of interest near the optical axis it is possible to avoid the effects due the camera lens radial distortion.

There are many different possible models for implementing vergence using disparity in the context of a robotic binocular system [2,3,7,12,13]. For example, by means of saliency detection or using stereo-matching techniques such as: phase correlation method like cepstral filtering, area based matching and feature-based matching. This work uses cepstral filtering to obtain a single disparity due their immunity to noise [1,2] and proves that the associated exponential calculus ovehead (FFT) can be overcome by common parallel GPU's. Scharstein and Szeliski [23], and Brown [24], present thorough reviews of these techniques.

4 Visual Vergence using Cepstral Disparity Filtering

A single disparity is estimated from the two cameras using the cepstral filtering. The cepstrum of a signal is the Fourier transform of the log of its power spectrum. Cepstral filter it is a known method of measuring auditory echo and it was introduced by Bogert [21]. The power spectrum of an audio signal with an echo present has a strongand easily identified component which is a direct measure of the echo period [1]. The binocular disparity measurement is obtained by applying of a non local filter (cepstral filter), followed by peak detection. Yeshurun and Schwartz [1,2] developed a method of using two-dimensional cepstrum as a disparity estimator. The first step of their method is to extract sample windows of size *h x w* from left and right images. The sample windows are then spliced together along one edge to produce an image *f(x,y)* of size *h x 2w*. Assuming that right and left images differ only by a shift, the spliced image may be thought as the original image at (0,0) plus an echo at $(w+d_h,d_v)$, where d_h and d_v are the horizontal and vertical disparities. The periodic term in the log power spectrum of such signal will have fundamental frequencies of $w+d_h$ horizontally and d_v vertically. These are high frequencies relative to the window size. The image dependent term, by contrast will be composed of much lower frequencies, barring pathological images. Thus, as some authors [1] show, the cepstrum of the signal will usually have clear, isolated peaks at $(^{+}_{-}(w+d_h),^{+}_{-} d_v)$.

The image *f(x,y)* composed by the left and right images pairs can be mathematically represented as follow:

$$f(x,y) = s(x,y) * [\delta(x,y) + \delta(x-(W+d_h), y-d_v)] \tag{1}$$

Where *s(x,y)* is the left image, $\delta(x,y)$ is the delta function, W the image width and * operator represents two dimensional convolution. The Fourier transform of such image pair is

$$F(u,v) = S(u,v).(1+e^{-j2\pi[(W+d_h)u+(d_v)v]}) \tag{2}$$

The power spectrum and the logarithm of equation (1), are:

$$|F(u,v)|^2 = |S(u,v).(1+e^{-j2\pi[(W+d_h)u+(d_v)v]})|^2 \tag{3}$$

$$logF(u,v) = logS(u,v) + log(1+e^{-j2\pi[(W+d_h)u+(d_v)v]}) \tag{4}$$

and the Cepstral filter is the inverse Fourier transform of equation (4)

$$F^{-1}[logF(u,v)] = F^{-1}[logS(u,v)] + \sum_{1}^{\infty}(-1)^{n+1}\frac{\delta(x-n(W+d_h), y-nd_v)}{n} \tag{5}$$

In the equation (5), the second term represents the prominent peak located in the output of Cepstral filter response. By determining these peak points positions it is possible to obtain disparity (figure 2).

4.1 Implementation on GPU using CUDA

Our system uses the GeForce 9800 GTX+ with 128 cores and 512MB of dedicated memory to process the cepstral filter. The main body of the cepstral algorithm

consists of a 2-D FFT, a point transform (the log of the power spectrum), and the inverse 2-D FFT.

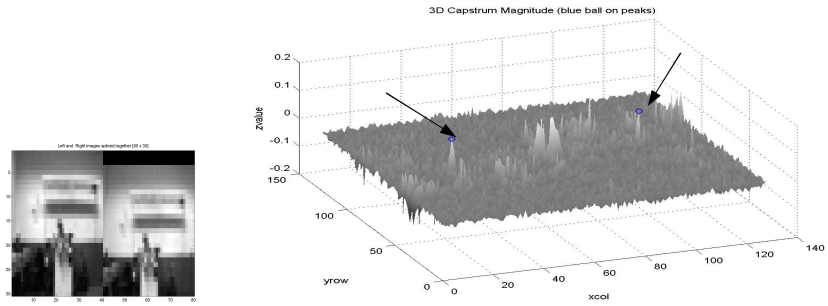

Fig. 2: Input subsampled spliced images [40x30], image pair with horizontal disparity=3 (left figure). Surface plot of the power spectrum of the cesptral filter (right figure). Peaks are visible at dominant global disparity location (marked with arrows)

For this 2D cepstrum algorithm we developed a GPU custom kernel to perform the point-wise absolute log in parallel using several threads, a GPU kernel to pad input data arrays with zeros (FFT requirement), GPU FFT transformations and all data allocation and data transfer procedures. A summarized global system algorithm loop is presented on figure 3. The 2D GPU FFT routines are implemented using CUFFT library [10], which are eight times faster than a CPU version using an optimized FFT and running on one core of a 2.4-GHz Core2 Quad Q6600 processor [9]. As the cepstral algorithm performs two FTT operations and the absolute log operation in parallel, the speedup is more than sixteen times faster than a CPU version. This multithreaded program is partitioned into blocks of threads that execute independently from each other, so that a GPU with more cores will automatically execute the program in less time than a GPU with fewer cores.

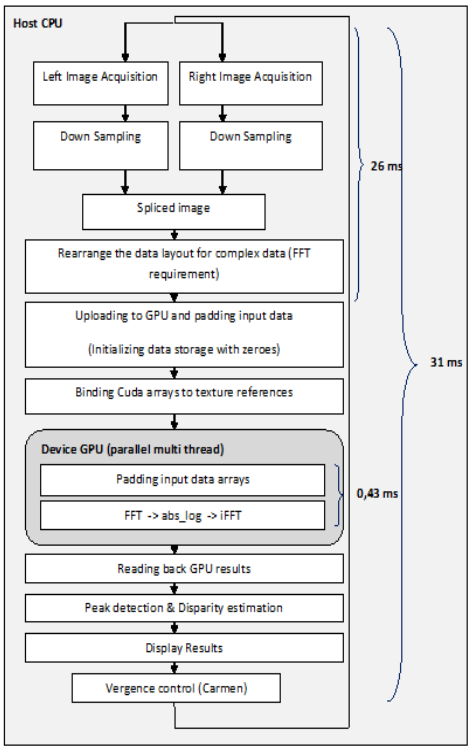

Fig. 3: Schematic block diagram of GPU cepstral filtering algorithm

5 Experiments

Experiment 1 – Image alignment

Figure 4 presents the real-time image alignment process frame sequence driven by the vergence control algorithm when an object is "instantly" positioned in front of the system. Both cameras changes alternate their angles to minimize the disparity. The performance measurements, according the schematic block diagram of figure 3, are shown on table 1.

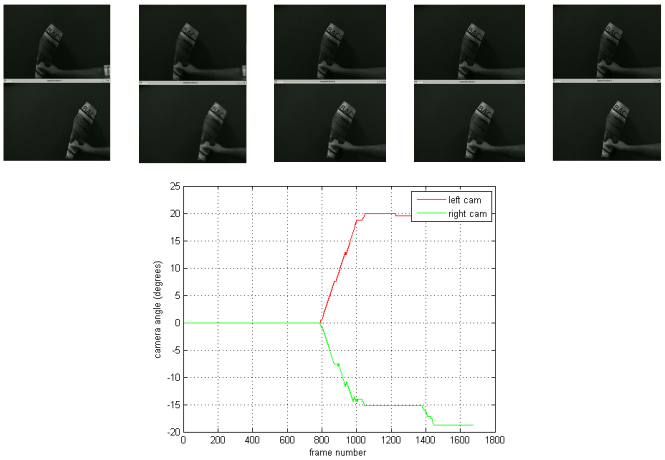

Fig. 4 – Real-time image alignment process frame sequence (each colum pair is an stereo pair). Below are the left camera angle values (red line) and right camera angle values (green) in degrees during the image alignment process.

Table 1. Processing time measurements

Task Set A	Processing Time
GPU (FFT abs log iFFT) [256x256]	0,43ms
OpenCV image acquisition 2x[640x480] and preprocessing	26 ms
Complete iteration cycle with vergence control	31 ms (f=32,25Hz)

Task Set B	Processing Time
GPU (FFT abs log iFFT) [256x256]	0,43ms
OpenCV image preloaded 2x[640x480] and preprocessing	3,2-4,5 ms
Complete iteration cycle without vergence control and image aquisition	6,9-9,1 ms (f=144,92Hz-109,89Hz)

Experiment 2 – Image alignment with a dominant camera

We have also implemented an experiment where the left camera follows a color object (a ball) using CPU OpenCV camshift algorithm [5] and the right camera equally follows the object while trying to minimize the disparity using the GPU Cepstral Filtering (figure 5). By demonstrating this behavior we show that binocular heavy tracking algorithms can be applied to one only camera allowing CPU extra computational power for other tasks. Work on vergence controller should be carrying out to enable smooth movements.

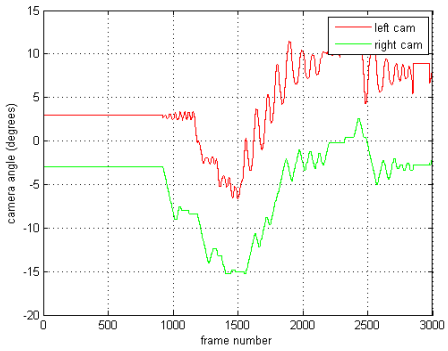

Fig. 5 – Right camera follows left camera during a tracking task

6 Conclusions

By implementing the cepstral filter on a graphics processing unit (GPU) using Compute Unified Device Architecture (CUDA) we demonstrate that robust parallel algorithms that use to require dedicated hardware are now available on common computers for real time tasks. Using the GPU for low level tasks allows CPU extra computational power for other high level tasks. The cepstral filtering algorithm speed up is more than sixteen times than on a CPU and the use of GPU Cepstral Filtering to perform vergence on binocular head systems is, to our knowledge, an contribution for the state-of-art.

6 References

1. Yehezkel Yeshurun, Eric L. Schwartz, Cepstral Filtering on a Columnar Image Architecture: A Fast Algorithm for Binocular Stereo Segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 11(7): 759-767 (1989).
2. David Coombs, Real-time Gaze Holding in Binocular Robot Vision, PhD. Thesis, Department of Computer Science, University of Rochester, June 1992.
3. Ki-Chul Kwon, Haeng-Su Lee, and Nam Kim, Hybrid Cepstral Filter for Rapid and Precise Vergence. Control of Parallel Stereoscopic Camera, Journal of the Research Institute for Computer and Information Communication. Vol. 12, No. 3, Dec 2004.
4. NVIDIA CUDA C ProgrammingGuide 3.1, NVIDIA, 2010
5. OpenCV (Open Source Computer Vision)
7. J. R. Taylor, T. Olson, and W. N. Martin, Accurate vergence control in complex scenes, in Proc. Computer Vision and Pattern Recognition (Seattle, USA, Jun. 1994), pp. 540-545
8. Ferreira, J. F., Lobo, J., Dias, J., Bayesian Real-Time Perception Algorithms on GPU - Real-Time Implementation of Bayesian Models for Multimodal Perception Using CUDA, Journal of Real-Time Image Processing, Special Issue, Springer Berlin/Heidelberg, published online (ISSN: 1861-8219), February 26, 2010.
9. M. Garland, S. Le Grand, J. Nickolls, J. Anderson, J. Hardwick, S. Morton, E. Phillips, Y. Zhang, and V. Volkov, Parallel computing experiences with CUDA, IEEE Micro, 28(4):13-27, July/August, 2008.
10. CUDA: CUFFT Library, NVIDIA Corp., 2010.
11. Olson, T.J., and Coombs, D., Real-Time Vergence Control for Binocular Robots, IJCV(7), No. 1, November 1991, pp. 67-89.
12. Dana H. Ballard, Reference Frames for Animate Vision, In International Joint Conference on Artificial Intelligence. AAAI, 1989.

13. J. Dias, C. Paredes, I. Fonseca, H. Araujo, J. Batista, and A. T. Almeida, Simulating Pursuit with Machine Experiments with Robots and Artificial Vision, IEEE Transactions on Robotics and Automation, Vol 3, N. 1, pp. 1-18, Feb. 1998.
14. J. Batista, J. Dias, H. Araujo, A.de Almeida, The ISR Multi-Degree of Freedom Active Vision Robot Head: Design and Calibration, SMART Program Workshop, Instituto Superior Técnico, Lisboa, Portugal, 27-28 April, 1995.
15. C. M. Brown, The Rochester robot, Tech. Rep. 257, Dept. Comp. Sci., Univ. Rochester, NY, 1988.
16. P. Mowforth, J. Siebert, Z. Jin, and C. Urquhart, A head called Richard, in Proceedings of the British Machine Vision Conference 1990, Oxford, UK, p.p. 361-366, 1990., pp. 361-366, N/A, 1990.
17. D. Betsis and J. Lavest, Kinematic calibration of the KTH head-eye system, in ISRN KTH, 1994.
18. L. Natale, G. Metta, and G. Sandini, Development of auditory-evoked reflexes: Visuo-acoustic cues integration in a binocular head, Robotics and Autonomous Systems (39), 2002, pp. 87-106.
19. J.-O. Eklundh, M. Bjrkman, Recognition of objects in the real world from a systems perspectiveâ, Kuenstliche Intelligenz, vol. 19, no. 2, pp. 12-17, 2005.
20. A. Bernardino, J. Santos Victor, Binocular tracking: integrating perception and control, In IEEE Transactions on Robotics & Automation, vol. 15, no. 6, pp. 1080-1094, 1999.
21. B. Bogert, M. Healy, J. W. Tukey, The Quefrency Alanysis of Time Series for Echoes: Cepstrum, Pseudo-autocovariance, Cross-Cepstrurn, and Saphe Cracking, In M. Rosenblatt, editor, Proc. Symp. Time Series Analysis, pages 209-243, John Wiley and Sons, 1963.
23. D. Scharstein, R. Szeliski, A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms, International Journal of Computer Vision, vol.47 no.1-3, pp.7-42, April-June 2002.
24. M.Z. Brown, D. Burschka, G.D. Hager, "Advances in Computational Stereo", IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), Vol. 25, No. 8, pp. 993-1008, 2003.
25. POP project (Perception on Purpose), number FP6-IST - 2004-027268, http://perception.inrialpes.fr/POP/
26. Niklas Wilming, Felix Wolfsteller, Peter König, Rui Caseiro, João Xavier, Helder Araújo: Attention Models for Vergence Movements based on the JAMF Framework and the POPEYE Robot. VISSAPP (2) 2009: 429-435.