Faster Upper Body Pose Estimation Using CUDA

Dane L. Brown[1], Mehrdad Ghaziasgar[1], James Connan[2]
Department of Computer Science
University of the Western Cape[1], Private Bag X17 Bellville, 7535, South Africa
Tel: + (27) 21 959-3010, Fax: + (27) 21 959-3006
and Department of Computer Science
Rhodes University[2], P O Box 94 Grahamstown, 6140
Tel: + (27) 46 603-8291, Fax: + (27) 46 636-1915
email: {2713985, mghaziasgar}@uwc.ac.za[1]; j.connan@ru.ac.za[2]

Abstract – Determining upper body poses using computer vision can have long execution times when using traditional linear methods on the CPU. This paper shows how parallel processing methods, and in particular the usage of a GPU API called CUDA, can increase system performance.

Index Terms— Upper Body Pose, Linear Methods, Parallel Processing, GPU, API

I. Introduction

Upper Body Pose Estimation determines the gestures made by the arms, hands and other parts of the upper body of a human being. The estimation of the upper body pose has a wide range of applications such as those in security systems, medical systems, human computer interaction and the focus of this research, namely, sign language recognition.

The SASL research group at the University of the Western Cape is in the process of designing a machine translation system that can automatically translate between South African Sign Language (SASL) and English [1]. A prototype of this system called iSign was created by the group. iSign is a digital phrase book that can translate simple phrases from SASL to English and vice versa using whole gesture recognition [1]. In order to be able to translate complex phrases, gestures need to be broken into their constituent parts which can be achieved by Upper Body Pose estimation.

Upper Body Pose Estimation extracts semantic information about the sign language performed by a sign language speaker. This can be used to determine the meaning of the gestures in English.

Imran Achmed created the current Upper Body Pose estimation system used by the SASL group [2]. This system uses a web camera and computer vision techniques and is able to determine the motions of the arms of a signer with a high accuracy of 81%. All processing of the system is performed on the CPU without focusing on parallel processing techniques.

This Work-in-Progress paper discusses the use of parallel processing techniques using the Graphics Processing Unit (GPU) with OpenCV and CUDA to enhance the processing speed of the existing system.

Modern GPUs contain 100s of cores and are capable of running millions of threads simultaneously, far greater than typical modern CPUs, containing between 4 and 12 cores. An example is the NVIDIA 580GTX GPU which contains 512 cores. Parallel computing using the GPU can address the problem of speed in image processing.

The CUDA API provides the required access to the GPU [3]. The porting of computer vision algorithms to the GPU using CUDA has shown a significant improvement in performance [4] [5] as is explained in a subsequent section.

Additionally, the popular image processing library OpenCV has recently provided partial support for the CUDA API and image processing on the GPU. However, many functions in this library do not yet support this capability.

The rest of this paper is organised as follows: Section II discusses related work in the field; Section III discusses the research goals; Section IV discusses the research methodology; the paper is concluded in Section V.

II. Related Work

This section first discusses the Upper Body Pose estimation system developed by Achmed, which will be improved in this research. The next subsection discusses two studies that compare the performance of image processing techniques on the CPU and GPU.

A. Achmed's Upper Body Pose Estimation System

The existing system was designed using OpenCV. It uses an example-based and a learning-based method to recognise the Upper Body Pose. However, in this research only the learning-based method is considered.

His system uses Haar Classifiers to determine the position of the face. The centre of the facial frame is usually situated on the nose. Achmed proposed that the nose colour is representative of the person's skin colour and is used to perform skin detection to eliminate non-skin pixels in the image. This is combined with frame differencing to obtain only moving skin pixels, that is, the hand(s). This is then sent to a Support Vector Machine which determines the position of the hands. The position is used to animate a 3D humanoid avatar in Blender [2].

It is important to note that many of the processes carried out can be performed in parallel, but are currently performed in series. An example is the frame differencing and the skin detection.

The system was tested on 6 different signers, 20 different SASL signs and once per signer. It achieved an average accuracy of 81%. The system was designed solely for the

optimisation of the accuracy. It runs on a single core. It was not tested to determine the processing speed.

B. Comparison of Image Processing on the GPU and CPU

This subsection first discusses how the OpenCV CUDA module can increase run-time performance of Gaussian Blur and Colour Conversion algorithm.

It then discusses how the Canny Edge Detection can be sped up using the CUDA API.

1. Gaussian Blur and Colour Conversion

Lengyel *et al.* performed a comparison of the computation rate per image as performed on the CPU and GPU.

The OpenCV CUDA module was used to speed up Colour Conversion and Gaussian Blur algorithms respectively. This module automatically assigns the amount of threads used in a block by the GPU.

The CUDA implementation on the GPU was compared to the CPU. The GPU used was a NVIDIA 9800GTX and the CPU was an Intel(R) Core(TM) 2 duo 6400 @ 2.13GHz [6].

The test ran on a 1280x1024 image and was performed an unknown number of times. The computation was found to be 2.7 times faster on the GPU than the CPU [6].

2. Canny Edge Detection on CUDA

Luo *et al.* used the CUDA API to speed up the canny edge detection algorithm.

In this implementation of CUDA programming, OpenCV was not used. The memory allocation was manually set. The block size was set to 16 x16 threads per block. Each thread correlates to a pixel, which was executed in parallel with neighbouring pixels inside the block [7].

The target GPU used for testing was the NVIDIA 8800GTX. The CPU used was an Intel(R) Core(TM) 2 CPU 6600 @ 2.40 GHz [7]. Each test was performed 1000 times on varied image sizes and the average was recorded for each size. The difference in results of the GPU version and the CPU version of edge detection are shown in TABLE I.

TABLE I
CUDA VS. OPENCV WITHOUT THE GPU MODULE

Image Size	CUDA (ms)	Opencv (ms)	Speedup
256 x256	1.82	3.06	1.68
512 x 512	3.40	8.28	2.44
1024 x 1024	10.92	28.92	2.65
2048 x 2048	31.46	105.55	3.35
3936 x 3936	96.62	374.36	3.87

III. RESEARCH GOAL

The aim of this research is to port the existing system to run on the GPU in order to achieve a real-time performance of 15 to 25 frames per second.

The CUDA API partially supports the computer vision library OpenCV, therefore most of the current OpenCV functions on which Achmed's system is built need to be ported to run on the GPU.

IV. RESEARCH METHODOLOGY

1. The existing system will be analysed to determine which functions need to be ported.
2. The functions will be ported to the CUDA API to run on the GPU.
3. A time metric will be used to evaluate whether increased performance has been achieved. Ideally real-time performance will be achieved.

V. CONCLUSION

This paper proposed a parallel processing approach by using the CUDA API for the GPU. This proposed implementation strives to improve the performance of the current linear approach used on the CPU. When utilised efficiently, this approach can help to improve the run-time performance of the machine translation system of the SASL project such that phrases can be translated in real-time.

REFERENCES

[1] Mehrdad Ghaziasgar, "Investigating the Intelligibility of Synthetic Sign Language Visualization Methods on Mobile Phones," University of the Western Cape, South , 2010.

[2] Imran Achmed and James Connan, "Upper Body Pose Estimation towards the translation of South African Sign Language," University of the Western Cape, Cape Town, 2010.

[3] David B. Kirk and Wen-mei W. Hwu, *Programming Massively Parallel Processors*.: Morgan Kaufmann, 2010.

[4] J Fung and S Mann, "Using graphics devices in reverse," in *GPU-based Image Processing and Computer Vision*., 2008, p. 912.

[5] Daniel Hefenbrock, Jason Oberg, and Nhat Tan Nguyen Thanh, "Accelerating viola-jones face detection to fpga-level using gpus.," in *IEEE Annual International Symposium on FieldProgrammable Custom Computing Machines*., 2010, pp. 11-18.

[6] Tamas K. Lengyel, James Gedarovich, Antonio Cusano, and Thomas J. Peters, "GPU Vision:Accelerating Computer Vision algorithms with Graphics Processing Units," 2011.

[7] Yuancheng Luo and Ramani Duraiswami, "Canny Edge Detection on NVIDIA CUDA," University of Maryland, College Park, 2008.

Dane Brown is currently an M.Sc student at the University of the Western Cape. He is currently doing research on sign language synthesis and novel communication applications on mobile interfaces for the Deaf and hearing impaired.

Mehrdad Ghaziasgar is the project manager of the South African Sign Language (SASL) research group. His interests are internet programming, mobile computing, computer vision and machine learning.

James Connan heads up the South African Sign Language (SASL) research group. His interests are computer vision and machine learning.