Image colourisation for compression using GPU hardware

Vaughan H. Lee, Yuko Roodt and Willem A. Clarke
Department of Electrical and Electronic Engineering
University of Johannesburg, P. O. Box 524, Auckland Park
Tel: +27 11 4775904, Mobile: +27 74 1748554
email: {920401897.student,yukor, willemc}@uj.ac.za

Abstract-There is a growing demand for high definition (HD) graphics with multimedia content. This demand requires significantly more computational power than before. The increased demand in video content will continue to grow, resulting in vast volumes of data continuously shifted across networks and the internet. The volume of video data must be decreased in order to better align to trends while maintaining efficient transmission and real-time processing requirements. In this work, the volume of information in HD images are decreased by reducing the image to greyscale while maintaining real-time performance requirements. The real-time processing requirement is met by shifting the computation to the graphics processing unit (GPU). The GPU is a programmable massively parallel processing unit with numerous cores. The performance of the colourisation technique which adds chrominance to the greyscale image is drastically increased using the GPU, which would otherwise take several minutes to colourise a 1080p image. The results indicate that a video compression scheme for HD video is viable.

Index Terms—Colourisation, compression, general-purpose computing, graphics processing unit (GPU).

I. INTRODUCTION

The application and services of multimedia entertainment are vast: IPTV, SD/HD video, video gaming, SD/HD video conferencing, virtual classroom, and video/audio streaming to mention a few. In addition there are also numerous handheld devices available to consumers, with increasing expectation for integrated HD graphic content in multimedia. Within this context, the use of compression is mandatory as the storage and transmission requirements are too great [1], [4], [6].

A recent, though not greatly researched approach to video compression makes use of image colourisation, a technique where colour is added to a greyscale image. There is significantly less information in a greyscale image than a colour image. Here compression is achieved since two channels of intensity representing the colours are not present, though approximated and added later to finally produce a colour image. In work presented by Kumar et. al. [2], selected key frames are encoded in luminance only at the encoder side. The scheme used, was integrated into MPEG-2 with the chrominance added to the selected luminance frames at the decoder side.

This work in progress presents work done to improve computational time required to colour an image from a video sequence. The colourisation algorithm is offloaded to the programmable GPU, for real time computing.

II. THEORY

The background theory used to colourise the frame is based upon an approach used by Hertzmann et. al. [3], which involves image comparisons. In image analogies work done by Hertzmann et. al. [3] a training set is used to setup a database. This database contains features which are used to determine the best match as well as the desired output. In order to determine the desired colours of the output image, a dataset needs to be trained initially. This training set consists of the input image A and the desired output image A' which has the desired characteristics respectively. The input test image B which needs to be matched to the database is the image to be synthesised into B'. The nearest pixel match found in the database is used as the desired output.

The features used to determine the best match is the luminance channel only. The number of incorrect matches when searching the whole database for a match quickly presents a problem. A neighbourhood is constructed for each lookup, when exact match test fails. In setting up a neighbourhood, performance is improved since the possible candidates are narrowed down to the most likely.

The luminance as determined directly from the RGB colour space is linearly dependant on RGB. Thus selecting the best match will result in a slightly different output than required. These differences are image artifacts and intensity differences which result in an output that does not have the same texture as the test image B. It is for this reason that the $\Upsilon C_r C_b$ colour space is used as an alternative; which is acceptable as this colour space is commonly used in video codec's. Then the luminance remains the same between the test image B and the synthesised output image B', resulting in the same image intensity only with chrominance added.

A. Algorithm

The database is determined from A and A', the neighbourhood $\mathcal{N}$ for each pixel is constructed using pixels ρ in an image $\mathcal{S}$, $\mathcal{N} = \{\mathcal{N}_k \mid \forall\, k \in \mathcal{S}\}$ while $\mathcal{N}_k$ is the set of pixels neighbouring pixel ρ where $\rho \neq \mathcal{N}_k$. The luminance Υ from the input test image A is transferred to the synthesised output image. The feature outputs are C_r and C_b which is determined by the best match in the database. The algorithm used to implement the colourisation is given in pseudocode.

$start\ algorithm$:
  $output_{i,j} = \Upsilon^{B}_{i,j}$
  $given\ \Upsilon_B, for\ \forall\ \rho\ in\ A\ do$:
  $if\ \rho_{i,j} \in \Upsilon_B == \Upsilon^{A}_{i,j}$
    $output_{i,j} = \rightarrow feature\ output$
  $else\ construct\ neighbourhood,$
    $for\ \forall\ \rho_{i,j}\ in\ neighborhood\ \mathcal{N}_k\ do$:
      $if\ \rho_{i,j} \approx neighbourhood\ \mathcal{N}^{k}_{m,n}$,
        $output_{i,j} = neighbourhood\ \mathcal{N}^{k}_{m,n} \rightarrow feature\ output$
      $else\ if\ \rho_{i,j} \neq any\ \rho_{i,j}\ in\ neighbourhood\ \mathcal{N}^{k}$
        $average\ pixel\ not\ matched \rightarrow feature\ update$
      end
  end
$end\ algorithm$

III. Experimental Results

The experimental images are HD 1080p frames taken from the HD video sequence "touchdown_pass". In each experiment, the training set A and A' were taken from the previous frame with the test input image the next frame in the sequence, which is shown in Figure 1.

Figure 1: Left is luminance input image B, centre image is the synthesised output image B' and the right image is the exact image. The frame number is 551 from HD touchdown_pass.

The CPU implementation has a computational time of several minutes, while the GPU implementation is in the order of 0.0227s for the complete 1080p image. The errors present in the synthesised image is attributed to rounding errors in the colour space conversion from RGB to $\Upsilon C_r C_b$ and the temporal difference of the next sequential frame. There are no motion vectors present to direct the optical movement, thus regions with movement larger than the neighbourhood will have large regions of error. Furthermore, very few image artifacts are visible which is verified by the cross-correlation coefficient of each channel is $0.9512, 0.9537, 0.9336$ for RGB respectively. A scaled top view of a surface plot depicts that the cross-correlation matrix shows that B^{exact} and $B^{'synt\ hesised}$ are highly correlated.

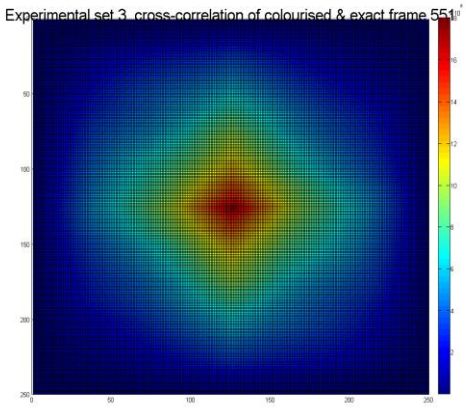

Figure 2: Experimental set 3, cross-correlation of colourised and exact frame 551

The experiment was applied to a variety of images, the results of sequential images are well matched using the described algorithm. In the case of a large shift in the images frames, the result is undesirable. The computational speed of the algorithm implemented on the GPU is most suitable to real-time application.

IV. Conclusion

This work in progress discusses a real-time colourisation implementation on GPU hardware. The colourisation technique is used to add the chrominance channels to the monochrome luminance channel based on the closest match in a neighbourhood. The synthesised output image is constructed from a database. This database is constructed from an initial training set, specifically using the $\Upsilon C_r C_b$ colour space as features. The implementation has outperformed the CPU implementation, with the GPU implementation averaging 0.02s while the CPU implementation takes several minutes.

This work suggests that the volume of full HD video can be further reduced by dropping chrominance channels of non-key frames while maintaining good performance. It is in this light that a full 1080p HD video scheme can now be implemented, continuing work based on these positive results. In this scheme, processes like frame colourisation, motion estimation, temporal interpolation and final rendering will be offloaded to the GPU with other processes continue computation on the CPU.

V. Acknowledgements

A special thanks for financial support from University of Johannesburg. Finally, thanks to my dad, mom and God.

VI. References

[1]. Guobin Shen, Guang-Ping Gao, Shipeng Li, Heung-Yeung Shum, and Ya-Qin Zhang, "Accelerate Video Decoding With Generic GPU", IEEE Transactions on circuits and systems for video technology, vol. 15, no. 5, May 2005.

[2]. Ritwik Kumar, Suman K. Mitra, "Motion Estimation based Color Transfer and its Application to Color Video Compression", 2008.

[3]. Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, David H. Salesin, "Image Analogies", 2001.

[4]. Mauricio Alvarez, Esther Salam´ı, Alex Ram´ırez and Mateo Valero, "HD-VideoBench. A Benchmark for Evaluating High Definition Digital Video Applications",2007.

[5]. John D. Owens, Mike Houston, David Luebke, Simon Green, John E. Stone, and James C. Phillips, "GPU Computing", Proceedings of the IEEE, Vol. 96, No. 5, May 2008.

[6]. Cebrail Taşkin and Serdar Kürşat Sarikoz, "An Overview of Image Compression Approaches", The Third International Conference on Digital Telecommunications, 2008.

Vaughan Lee received his B. Eng in electrical and electronic engineering in 2009 from the University of Johannesburg and is presently studying towards his Master of Engineering degree at the same institution. His research interests include video compression and signal processing.