AN ARM NEON OPTIMISED IMAGE ABSTRACTION METHOD UTILISING THE COSINE INTEGRAL IMAGE METHOD

Ryan Mark Gibson, David John James Round, Mark David Jenkins, Peter Barrie and Gordon Morison

School of Engineering and Built Environment, Glasgow Caledonian University
Cowcaddens Road, G4 0BA, Glasgow, Scotland, UK
phone: + (44)141-331-3352, fax: + (44)141-331-3370, email: gordon.morison@gcu.ac.uk
web: www.gcu.ac.uk

ABSTRACT

In this paper we present a fast implementation of an automatic non-photorealistic image processing technique which transforms an input image frame of a video stream into a non-photorealistic abstracted cartoon stylised render. The approach presented utilises a fast cosine integral image method to create a separable bilateral filtering stage which operates in constant time. This is subsequently put through a colour quantisation stage and combined with an edge overlay system to generate the abstracted image output. The algorithm is implemented with OpenCV on a Beagleboard-xM running Angstrom GNU/Linux to demonstrate the improved performance obtained utilising the cosine integral image bilateral filter over the OpenCV standard bilateral filter implementation, and to demonstrate further performance improvements can be obtained through utilising optimised routines on the ARM NEON floating point unit of the Beagleboard-xM.

1. INTRODUCTION

Non-photorealistic image abstraction simplifies images through efficient smoothing functions, while simultaneously enhancing object boundaries and details, resulting in a stylised and abstracted image similar to that of a cartoon. Developed image abstraction frameworks within literature utilise significant mathematically complex and multiple stage algorithms for image smoothing and edge boundary enhancement functions to obtain a non-photorealistic abstract stylised effect. Smoothing functions are utilised to reduce image detail, while simultaneously preserving the high-spatial frequency edge components. Complex edge preserving smoothing operations such as the Kuwahara [1], bilateral [2] and mean shift filters [3], have been previously utilised within non-photorealistic image abstraction framework techniques demonstrated from Kyprianidis et al. [4], Winnemöller et al. [5] and DeCarlo and Santella [6] respectively. Frequent approaches to visually enhancing object edge boundaries and details within non-photorealistic image abstraction algorithms involve overlaying the associated edge gradients as black components on top of the efficiently smoothed image. There are many high-spatial frequency edge detection algorithms available, while mathematically complex methods such as the Difference of Gaussian (DoG) has extensively been utilised in non-photorealistic algorithms due to Gooch et al. [7] demonstrating DoG edges can produce highly recognisable monochrome stylised facial abstractions, which led to Winnemöller et al. [5] utilising DoG edge components for obtaining abstracted images through DoG edge overlay. Zhao et al. [8] further demonstrated an improved DoG edge stylisation response for image abstraction through feature-flow techniques.

The previously described non-photorealistic abstraction and stylisation algorithms are mathematically complex and intensive, multiple stage techniques, where extensive implementation occurs on high level GPU and CPU devices with a significant abundance of available resources [4, 5, 8 9]. Embedded device non-photorealistic abstraction implementation in comparison requires significant optimisation and design consideration for the stylisation algorithm to efficiently operate within resource limited platforms.

In this paper we present a fast abstraction algorithm for implementation on a Texas Instruments Beagleboard-xM DM3730 ARM Cortex-A8 embedded platform [10]. The fast abstraction algorithm is designed and optimised for resource limited embedded devices, where the implementation utilises the Neon optimised instruction set [11] and cosine integral image decomposition [12] to obtain significant frame rate performance improvement for abstracted non-photorealistic image and video.

The significant work in image abstraction literature and associated complex algorithms have been described, in addition to presenting the problem of image abstraction suitability within embedded devices. The developed fast abstraction algorithm and the various stages designed and optimised for embedded implementation are presented in Section 2. Section 3 describes the abstraction algorithms implemented realisation on a Beagleboard-xM device running Angstrom GNU/Linux [13] with a camera input video and monitor output. Section 4 presents the fast abstraction implementation performance results for the various optimisations, while Section 5 discusses and concludes the work presented in this paper.

2. ABSTRACTION ALGORITHM FOR EMBEDDED PLATFORMS

The developed non-photorealistic abstraction algorithm for image and video rendering is demonstrated in Figure 1. The image and video frame input is converted from RGB colour space to YCrCb space, where the luminance Y component

978-1-4799-6843-5/14/$31.00 ©2014 IEEE

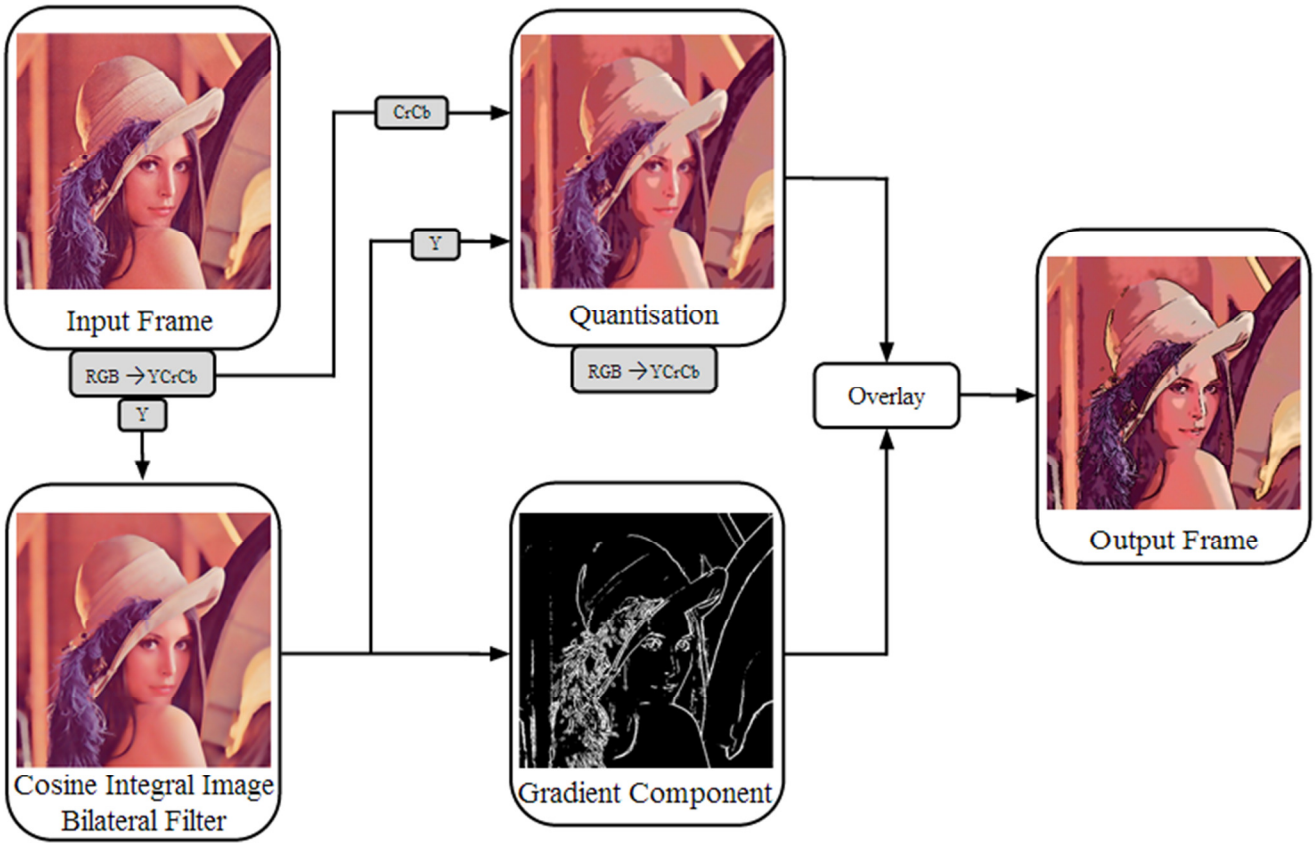


Figure 1 – Non-photorealistic abstraction algorithm framework

is extracted and modified to control the perceptual colour contrast levels when converted back to RGB for the system output [14]. The extracted luminance component is efficiently smoothed through a bilateral filter implemented with the Cosine Integral Image (CII) decomposition method [12], which then has its associated edge components extracted through an edge gradient operator. The smoothed luminance frame is then quantised to further enhance the non-photorealistic abstraction effect before being combined with the original frame colour components CrCb and converted back to RGB colour space. The edge components are then overlaid and presented as black on the output frame as shown in Figure 1.

2.1 CII Bilateral Filter Decomposition

The bilateral filter is a nonlinear region adaptive image processing algorithm, which efficiently smooth's an image while preserving the images important high-spatial frequency edge components. The bilateral filter H performs an image region weighted average of image f as demonstrated in (1). The pixel weight value applied within the image region mask E is determined through Gaussian spatial f_s and intensity f_i components shown in (2) and (3). The spatially variant bilateral function determines only pixels spatially close and of a similar intensity range value, which are used for obtaining the output. Abstraction algorithms frequently implement multiple iterative bilateral

$$H(x,y) = \frac{\sum_{(s,t)\in E} f(x,y) f_s(x,y) f_i(x,y)}{\sum_{(s,t)\in E} f_s(x,y) f_i(x,y)} \quad (1)$$

$$f_s(x,y) = e^{\frac{-(\|x-s,y-t\|)^2}{2{\sigma_s}^2}} \quad (2)$$

$$f_i(x,y) = e^{\frac{-(|x-s,y-t|)^2}{2{\sigma_i}^2}} \quad (3)$$

filters for image smoothing as demonstrated by Winnemöller et al. [5] and Kypriandis et al. [9].

Embedded devices are significantly resource limited and require efficient algorithm design to improve realised performance. The CII approach presented by Elboher and Werman [12] allows the bilateral filters spatial and range characteristics to be represented through frequency decomposed cosine functions to obtain a significant performance improvement in comparison to standard bilateral filtering. The bilateral filters Gaussian kernels are decomposed to less complex operations with the Discrete Cosine Transform (DCT) to obtain an integral image with a constant number of operations per pixel, regardless of filter size. The bilateral filters associated Gaussian kernels are approximated through inverse DCT functions of k-terms,

where highly accurate approximations of Gaussian kernels can be achieved with 3 cosine function terms as demonstrated by Elboher and Werman. These linear cosine functions are combined, with a constant number of computations, to obtain CII approximated functions of the bilateral filters Gaussian spatial and intensity kernels.

A cosine kernel function *cos(ux),* with various frequencies *u,* can be efficiently convolved with a 1D image *f(x)* for a spatial mask *E,* where $s \in E$, as demonstrated as follows:

$$c(x) = \sum_{s \in E} f(x) \cos\big(u(x-s)\big) \tag{4}$$

The convolution may be expanded through the trigonometric identity $\cos(\alpha - \beta) = \cos(\alpha)\cos(\beta) + \sin(\alpha)\sin(\beta)$, resulting in the function demonstrated in (5), where $\cos(us)$ and $\sin(us)$ values can be obtained from a look-up table, therefore removing their calculation cost during implementation.

$$c(x) = \cos(us) \sum_{s \in E} f(x) \cos(ux) + \sin(us) \sum_{s \in E} f(x) \sin(ux) \tag{5}$$

Similarly, a 2D image *f(x,y)* convolved with cosine spatial kernel function $\cos(u_s s)\cos(u_t t)$ can be derived to perform the convolution of two 1D filters; C_s and C_t, as shown in (6).

$$C(x,y) = (C_s(x) * C_t(y)) * f(x,y) \tag{6}$$

Therefore convolution can be determined through applying function C_s with image rows *f(x)*, then filtering the resultant column values of $C_s(y)$ with function C_t, as demonstrated in (7) and (8) for a spatial mask E, where $(s,t) \in E$.

$$C_s(x) = \cos(u_s s) \sum_{s \in E} f(x,y) \cos(u_s x) + \sin(u_s s) \sum_{s \in E} f(x,y) \sin(u_s x) \tag{7}$$

$$C(x,y) = \cos(u_t t) \sum_{t \in E} C_s(x,y) \cos(u_t y) + \sin(u_t t) \sum_{t \in E} C_s(x,y) \sin(u_t y) \tag{8}$$

Furthermore, the computational cost is further reduced through utilising look-up tables for $\cos(u_s s)$, $\sin(u_s s)$, $\cos(u_t t)$ and $\sin(u_t t)$. In addition, convolving an image with separable C_s and C_t kernels produce only double the computational cost of the 1D CII convolution function.

The decomposition process presented is for spatial pixel functions suitable for utilisation with the bilateral filters spatial f_s component. The bilateral filters CII intensity component f_i can be obtained in a similar methodology, where the pixel values of locations *s* and *t* within the spatial mask *E* would be utilised rather than the spatial distance. The look-up tables for $\cos(u_s s)$, $\sin(u_s s)$, $\cos(u_t t)$ and $\sin(u_t t)$ of the intensity CII function should be indexed with pixel value range 0-255.

The abstraction algorithm presented for embedded device implementation utilises the described CII decomposition function to ensure efficient bilateral filtering operations occur. Furthermore, the CII bilateral filter is applied for a single iteration to significantly reduce computation requirements, while still obtaining highly stylised images.

2.2 Edge Gradient

There are a wide range of edge detection algorithms with various computational complexity, ranging from the relatively simple Laplacian and Sobel methods [15] to the complex multiple stage techniques such as the Canny [16] and DoG methods. Abstraction algorithms frequently utilise visually stylised edges, such as the DoG edge gradient as demonstrated by Winnemöller et al. [5], Gooch et al. [7] and Kypriandis et al. [9].

The Sobel edge detection algorithm is a relatively simple method suitable for obtaining fast operations in resource constrained embedded devices. The Sobel algorithm determines edges by derivatively evaluating the intensity gradient change of pixels within Sobel operator weighted masks for x-axis G_x and y-axis G_y components demonstrated in (9) and (10).

$$G_x = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{9}$$

$$G_y = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{10}$$

The Sobel weighted mask components G_x and G_y are convolved with image *f* to obtain intensity difference functions Ψ_x and Ψ_y for the image x-axis and y-axis as demonstrated in (11) and (12).

$$\Psi_x(x,y) = \sum_{(s,t) \in G_x} G_x(s,t) f(x-s, y-t) \tag{11}$$

$$\Psi_y(x,y) = \sum_{(s,t) \in G_y} G_y(s,t) f(x-s, y-t) \tag{12}$$

The Sobel operators absolute difference is then determined from intensity functions Ψ_x and Ψ_y, which is then compared against a pre-assigned threshold value τ to obtain the output edge *G* as demonstrated in (13). Values exceeding the threshold parameter are determined to be an edge and assigned a value of 255 and those less than the threshold parameter are ignored and assigned a value of 0.

$$\mathrm{G}(x,y) = \sqrt{\Psi_x(x,y)^2 + \Psi_y(x,y)^2} > \tau \tag{13}$$

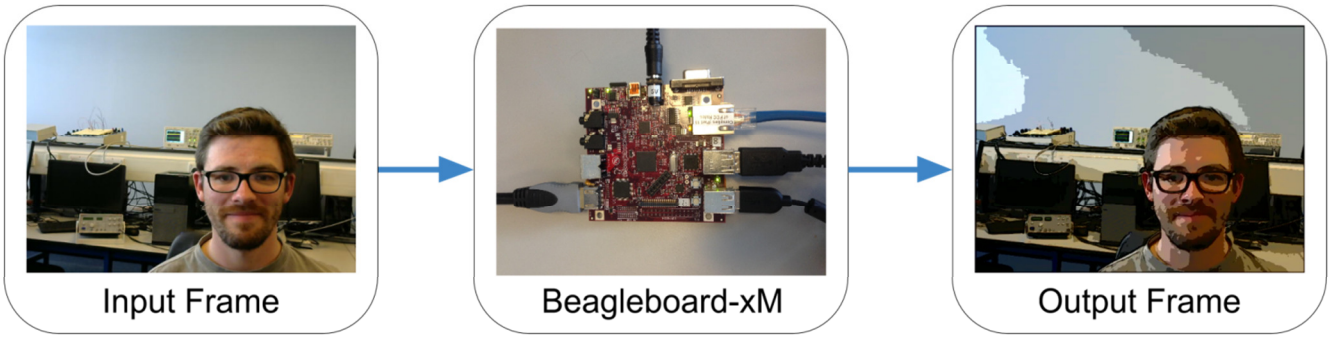


Figure 2 – Beagleboard-xM experimental set-up

2.3 Quantisation

Quantisation enhances the non-photorealistic stylisation as the frame is segregated into fixed colour regions, obtaining a more cartoon style effect. Quantisation functions can be of various complexities, such as a simple colour space range look-up table split into a specific number of even ranges or a significantly more complex hyperbolic tangent function as implemented by Winnemöller et al. [5]. A simple rounding function look-up table of 5 colour region with a level difference of 50 was applied for fast quantisation on low-level embedded devices.

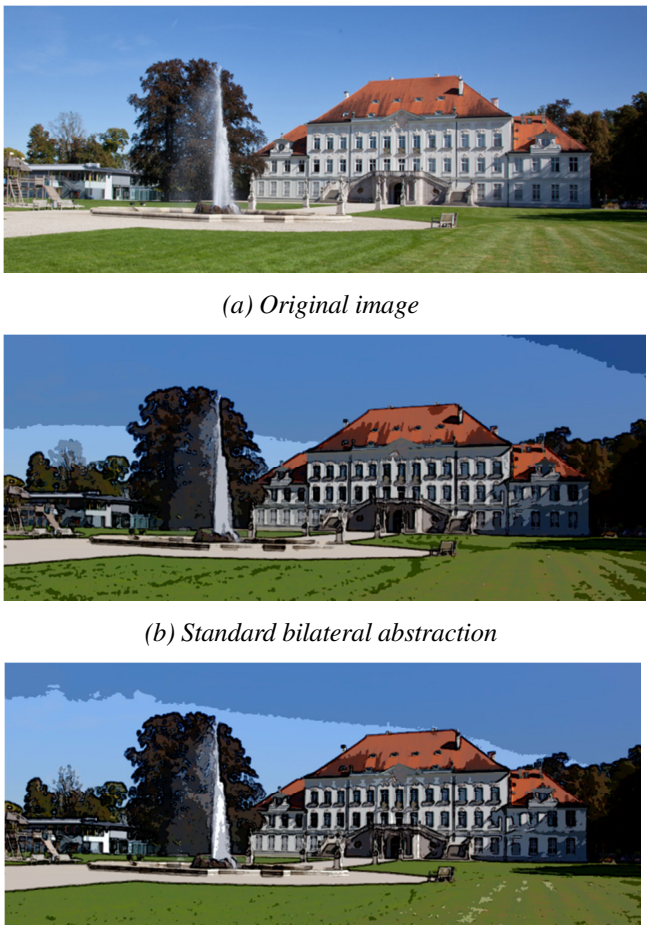
3. IMPLEMENTATION

The described non-photorealistic abstraction algorithm for embedded platforms was implemented on a Beagleboard-xM DM3730 ARM Cortex-A8 device with OpenCV running on Angstrom GNU/Linux at 1GHz as demonstrated in Figure 2. A camera input configured to 320 x 240 pixel resolution was processed by the Beagleboard-xM. The Beagleboard-xM applied the abstraction algorithm, and then displayed the obtained non-photorealistic abstraction rendered video within the Gnome desktop environment.

4. RESULTS

The presented abstraction algorithm with a single iteration of the CII bilateral decomposition function and abstraction with a standard OpenCV bilateral filter of 5 iterations, typical of other abstraction algorithms [5, 9], produced significantly stylised and abstracted images as demonstrated in Figure 3. The bilateral and CII abstraction methods obtained highly non-photorealistic stylised images, which are both visually similar to an extent, with the only significant visual difference being the sky quantisation levels. The standard OpenCV bilateral filter implementation utilised multiple iterations, in comparison to the CII which only utilised a single pass. In addition a CII abstraction example of a Cameron Diaz test image is presented in Figure 4, demonstrating a significantly non-photorealistic stylised and abstracted rendered image.

The presented embedded platform non-photorealistic abstraction algorithm was implemented on a Beagleboard-xM as demonstrated in Figure 2. Abstraction algorithms utilising the standard OpenCV bilateral filter, CII bilateral function and a Neon optimised CII bilateral decomposition function were implemented to determine the performance variations. Implementing the standard OpenCV bilateral filter in the non-photorealistic framework obtained 7.3 fps and 5.4 fps for 1 and 5 bilateral filter iterations, where 5 iterations are common with other abstraction algorithms utilising the bilateral filter [5]. In comparison the CII bilateral function



(a) Original image

(b) Standard bilateral abstraction

(c) CII bilateral abstraction

Figure 3 - Schoolhouse abstraction comparison

(a) Original Image *(b) CII Abstracted Image*

Figure 4 – Cameron Diaz image abstraction

obtained 10.2 fps and demonstrated a performance increase to 10.4 fps with the Neon optimised instruction set. The implemented Neon CII bilateral decomposition abstraction algorithm demonstrated 42.5% and 92.6% improvements in comparison to using the standard OpenCV bilateral filter for 1 and 5 iterations. Furthermore, the presented abstraction algorithm configured to a 640 x 480 video resolution obtained 5.2 fps, in contrast the abstraction algorithm presented by Winnemöller et al. [5] obtained 9-15 fps and 0.3-0.5 fps for implementation on a GeForce GT6800 GPU and an Athlon 64 3200+ CPU high level platform devices with significantly more resources than the Beagleboard-xM.

5. CONCLUSION

In this paper we present a fast non-photorealistic abstraction algorithm suitable for resource limited embedded devices. The presented abstraction algorithm efficiently smooth's and reduces detail, while preserving and enhancing object edge boundaries to produce a non-photorealistic stylised image similar to cartoons. The abstraction method performs operations suitable for embedded devices, such as utilising the Sobel edge detection algorithm over the computationally complex DoG edges, a quantisation look-up table and applying a single bilateral function in comparison to using multiple iterations. In addition the method presented utilises a CII bilateral decomposition function to significantly improve performance. The abstraction algorithm presented was implemented on a Texas Instruments Beagleboard-xM DM3730 ARM Cortex-A8 embedded device running OpenCV on Angstrom GNU/Linux at 1 GHz with a camera input of 320 x 240 resolution, which was abstracted and displayed in the Gnome desktop environment. The abstraction algorithm utilising CII bilateral decomposition with the optimised Neon instruction set obtained a frame rate of 10.4 fps, demonstrating a significant 42.5% performance improvement in comparison to 7.3 fps obtained utilising the standard OpenCV bilateral function.

REFERENCES

[1] M. Kuwahara, K. Hachimura, S. Eiho and M. Kinoshita,"Processing of ri-angiocardiographics images", *Digital Processing of Biomedical Images*, Springer, pp. 187-202, 1976.

[2] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images", *Computer Vision, 1998, International Conference on,* pp. 839-846, 1998.

[3] D. Comaninciu and P. Meer, "Mean shift: a robust approach toward feature space analysis", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 603-619, 2002.

[4] J. E. Kyprianidis, H. Kang and J. Döllner, "Image and video abstraction by anisotropic Kuwahara filter", *Computer Graphics Forum*, vol. 28, no. 7, pp. 1955-1963, 2009.

[5] H. Winnemöller, S. C. Olsen and B. Gooch, "Real-time video abstraction", *Graphics (TOG), ACM Transactions on* , vol. 25, no. 3, pp. 1221-1226, 2006.

[6] D. DeCarlo and A. Santella, "Stylization and abstraction of photographs", *Graphics (TOG), ACM Transactions on,* vol. 21, no. 3, pp. 769-776, 2002.

[7] B. Gooch, E. Reinhard and A. Gooch, "Human facial illustrations: Creation and psychophysical evaluation", *Graphics (TOG), ACM Transactions on,* vol. 23, no. 1, pp. 27-44, 2004.

[8] H. Zhao, X. Jin, J. Shen, X. Mao and J. Feng, "Real-time feature-aware abstraction", *The Visual Computer,* vol. 24, no. 7-9, pp. 727-734, 2008.

[9] J. E. Kypriandis and J. Döllner, "Image Abstraction by Structure Adaptive Filtering", in *Proc. EG UK Theory and Practice of Computer Graphics,* pp. 51-58, 2008.

[10] Texas Instruments DM3730 Digital Media Processor: http://www.ti.com/product/dm3730

[11] ARM Cortex-A8 Technical Reference Manual: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0344k/index.html

[12] E. Elboher and M. Werman, "Cosine integral images for fast spatial and range filtering", *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pp. 89-92, 2011.

[13] Angstrom GNU/Linux: http://www.angstrom-distribution.org/

[14] G. Wyszecki and W. S. Stiles, *Colour science*: Wiley New York, 1982

[15] R. Gonzalez, R. Woods, *Digital Image Processing, 3rd Edition.* Prentice Hall, 2007.

[16] J. Canny, "A computational approach to edge Detection" *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 8, no. 6, pp:679-698, 1986.