Implementation of Live Video Cartooning Using Thresholding Technique in YUV Format

K. Martin Sagayam[1], Nitin John James[2], Hudson John[3], Dan Thomas Jarard[4]
Assistant Professor, Electronics and Communication Engineering, Karunya University, Coimbatore, India[1]
UG Students, Electronics and Communication Engineering, Karunya University, Coimbatore, India[2, 3, 4]

Abstract: The paper discussed here deals with inducing cartoon like features to an input video using DSP processor (Texas Instrument product, TMS3206416). The basic process involved is taking input, a color video which is to be converted into a cartoon like video in DSP board can be displayed on VM3224K2 Daughter kit (LCD). VM3224Daughter Kit is embedded with DSK6416 Kit to display the output video. The conversion process is carried out in the YUV colour format. Converting in the YUV format is advantageous because applying thresholding technique to the intensity component (Y') of the video can render a cartoon like appearance to the video as the YUV colour model is based on human perception. Only the intensity component of the video in YUV format is modified as the YUV colour model allows apparent changes in the colour perception without actually altering the chrominance component of the video. Suitable thresholding techniques; single level or multi level thresholding technique can be used to get the desired effect. YUV colour model has three components namely Y, U and V. 'Y' is the intensity component whereas 'U' and 'V' are chrominance components. As this colour model facilitates apparent change in colour by simply varying the intensity component it is a very simple and intuitive technique to convert an input video to a cartoon like video. In order to observe the quality and suppleness of the algorithm various simulations in image level using different techniques has been compared with the proposed algorithm in MATLAB software. The proposed algorithm namely thresholding technique in YUV format were well implemented practically on our hardware (TMS6416 kit). Implementation of algorithm has been carried out in C-programming language using CCstudio v3.1 and various simulations have been carried out in MATLAB.

Keywords: Video Cartooning, Cartoonized, Thresholding, YUV Colour Model, TMS3206416, VM3224K2 Kit, CC Studio

I. Introduction

The purpose of this project is to create a code for implementing cartoonish fe atures in a real time video such that the fine detailing of the video is lost while preserving the edges of various objects so that the output video is less visually disturbing than the original video. We have used MATLAB for simulation purposes and 6416 DSK CCStudio v3.1 for code developing and implementation.

The main challenges include creating a code that does video processing with acceptable speed of execution and buffering the live video feed instead of storage and retrieval to minimize time delay. Techniques such as masking and filtering have already been implemented for imparting cartoonish features to an image, however such techniques cannot be implemented for live video processing as they take considerable execution time.

This was overcome by applying the thresholding technique on a video that was in the YUV format. The YUV colour model is based on the human perception of colour. The main concept of YUV colour model is that the as the intensity of an image is varied, our perception of its colour changes.

Mostly images are represented in compressed format to save memory space and bandwidth. So it is better if enhancement of the image can be achieved in compressed domain rather than transforming to spatial domain and applying the enhancement technique and transforming back to compressed domain; thereby increasing the computational overhead.

In our work we have represented the color image using Y-Cb-Cr color space so that we can preserve both luminance and color component. Previous works have used the RGB colour model and have created a mask or used filters such as bilateral filter so that the input image is made to look like a cartoon. In our approach we have adopted a method of thresholding only the intensity component ‘Y’ of Y, Cb and Cr which substantially lowers computational burden and at the same time renders cartoon like attributes to the image. After performing thresholding on the intensity component, the Y-Cb-Cr is converted into RGB using mathematical equation. All computations have been performed in C-source code as it is to be implemented in Code Composer Studio. Code Composer Studio (CCS) provides an integrated development environment (IDE) for real - time digital signal processing applications based on the C programming language. It incorporates a C compiler, an assembler, and a linker. It has graphical capabilities and supports real - time debugging. Code Composer Studio acts as an interface between user and DSK6416 (TMS3206416). The final obtained YUV values, is of a cartooned output video. This YUV values are again converted into RGB565 form as our displaying LCD (VM3224 Daughter kit) stores the image pixels in RGB565 in its memory.

A. V Simulations in MATLAB and their comparisons

As an example, consider the MATLAB simulation shown below. It compares the two existing methods for cartooning with the proposed method. From the respective MATLAB command windows it can easily be seen that the existing methods consume considerable time for execution. The proposed method in addition to saving the execution time also gives a better output performance considering the standard assumptions of trade-off concept. Though the project is concerned about video processing, satisfactory assumptions on the performance of the code can be derived from the outputs of images as videos are basically a stream of images. The time consumed for execution for the first two methods are impractical for live video processing as the cumulative delay for each image or frame in the video would be unacceptable.

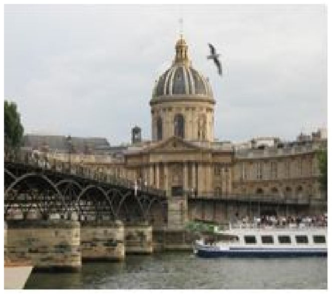
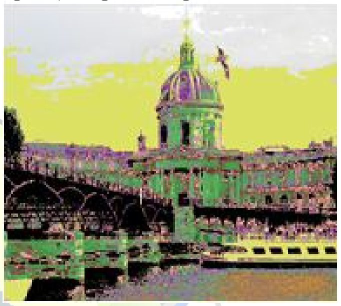
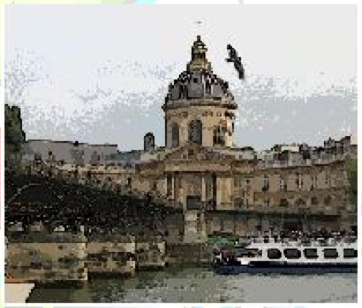
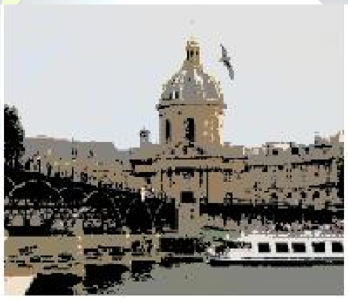


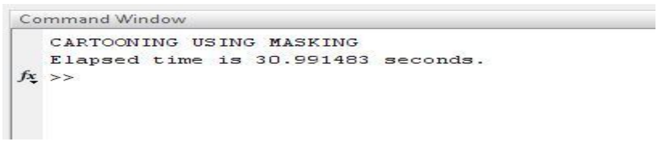
Fig. 1 a) Original Image


b) Masking Output


c) Bilateral Filtered Output

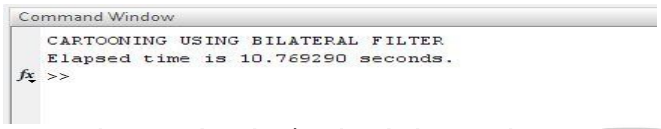

d) Thresholded Output

B. MATLAB Command Windows



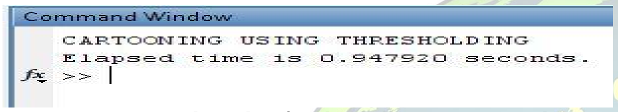
```
Command Window
CARTOONING USING MASKING
Elapsed time is 30.991483 seconds.
>>
```

Fig 2 a) Command window for Masking Technique



```
Command Window
CARTOONING USING BILATERAL FILTER
Elapsed time is 10.769290 seconds.
>>
```

b)Command Window for Bilateral Filtering Technique



```
Command Window
CARTOONING USING THRESHOLDING
Elapsed time is 0.947920 seconds.
>> |
```

c)Command Window for Thresholding on YUV model

Current techniques incorporated in cartooning a video are Masking and Bilateral Filtering. However they are performed on stored video ie; they are not used for live video processing. Live video cartooning finds its scope in news broadcasts as it enables a news channel to air sensitive contents such as a bomb blast or a gruesome attack without blurring the image or video fearing the impact on the viewers without a censor. Current techniques allow only blurring of the frame however with the proposed method it will be possible to air the scene as a video with cartoon like attributes is not visually disturbing. To achieve this we need a code that can process live video feed and hence we need a code that has very less execution time.

It is clearly inferred that though the thresholding on YUV method leads to an output that is slightly inferior to the Bilateral Filtering method's output, the execution time is phenominally reduced. This enables the code to be used for live video processing as the faster execution speed minimises the delay so that it is negligible.

The simulations have been carried out in MATLAB v10.1. The actual implementation of the code is done on the TMS320C6416 DSK subsequently the code for live video cartooning have been developed in CCStudio v3.1.

II. MATHEMATICAL PRELIMINARIES

The PAL CCD camera outputs the captured input video in the YUV colour model format. The computation is carried out on the YUV format. However the LCD display on the video daughter card outputs only RGB 565 format we convert the YUV colour model to RGB colour model for the sake of display. As we require only the YUV value for processing, the conversion from RGB to YUV is not required. The video after processing is converted into the RGB format using a look up table that is defined in the program using the conversion formulae. Formulae for conversions from YUV to RGB and vice versa have been given below.

The following equations are used to convert Y-Cb-Cr to RGB and vice-versa:

$$R = Y + 1.402\,(Cr-128) \qquad (1)$$

$$G = Y - 0.34414\,(Cb-128) - 0.71414\,(Cr-128) \qquad (2)$$

$$B = Y + 1.772\,(Cb-128) \qquad (3)$$

$$Y = 0.299R + 0.587G + 0.114B \qquad (4)$$

$$U = -0.147R - 0.289G + 0.436B \qquad (5)$$

$$V = 0.615R - 0.515G - 0.100B \qquad (6)$$

III. THE PROPOSED METHOD

The CCD camera directly outputs in YUV format. Rest of the operation and computation of our algorithm is carried out in simple C-source code. We have chosen C-program as we have to show practical real time video cartooning in DSP hardware that is TMS6416 toolkit and as programming of TMS6416 is done in code composer studio, & code composer studio is an IDE which accepts simple C-program. As the input video is in the desired format we first plan on the thresholding limits so as to convert the video. The input video can be split into three parts; they are the intensity component 'Y' and the two chrominance component 'U' and 'V'. The main advantage of this proposed method is that it is very easy to implement. The desired cartoonish effect can be obtained on the video just by varying the intensity in each frame. Many kinds of thresholding can be implemented. The simplest is single thresholding. Here we specify only one threshold limit. If the number of limits on the threshold is increased we obtain multi level thresholding. The higher the level of thresholding, the more detail each frame of the video will hold.

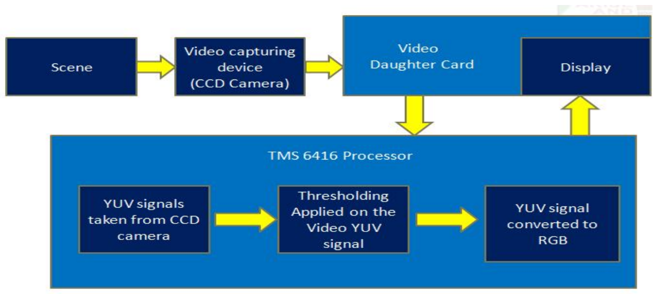

Scene
Video capturing device (CCD Camera)
Video Daughter Card
Display
TMS 6416 Processor
YUV signals taken from CCD camera
Thresholding Applied on the Video YUV signal
YUV signal converted to RGB

Fig. 3 Block diagram for Thresholding in YUV mode

The scene which is to be cartoonized is captured by the CCD camera. The output of the camera is given to the DSP processor through the video daughter card. The processor converts the video to a cartoon based on the code inputted. The cartoonized video in YUV format is converted into RGB 565 format by the use of a RGB look-up table defined in the program. The RGB output is transferred back to the video daughter card and is displayed on the LCD.

Now, the paper will discuss about the hardware & software that is used in our research, that are:

A. *TMS320C6416 DSK toolkit*

The DSP on the 6416T DSK interfaces to on-board peripherals through one of two busses, the 64-bit wide EMIFA and the 8-bit wide EMIFB. The SDRAM, Flash and CPLD are each connected to one of the busses. EMIFA is also connected to the daughter card expansion connectors which are used for third party add-in boards. An on-board AIC23 codec allows the DSP to transmit and receive analog signals. McBSP1 is used for the codec control interface and McBSP2 is used for data. Analog I/O is done through four 3.5mm audio jacks that correspond to microphone input, line input, line output and headphone output. The codec can select the microphone or the line input as the active input. The analog output is driven to both the line out (fixed gain) and headphone (adjustable gain) connectors. McBSP1 and McBSP2 can be re-routed to the expansion connectors in software.

A programmable logic device called a CPLD is used to implement glue logic that ties the board components together. The CPLD also has a register based user interface that lets the user configure the board by reading and writing to the CPLD registers. The DSK includes 4 LEDs and 4 position DIP switch as a simple way to provide the user with interactive feedback. Both are accessed by reading and writing to the CPLD registers.

An included 5V external power supply is used to power the board. On-board switching voltage regulators provide the 1.2V DSP core voltage and 3.3V I/O supplies. The board is held in reset until these supplies are within operating specifications. A separate regulator powers the 3.3V lines on the expansion interface. Code Composer communicates with the DSK through an embedded JTAG emulator with a USB host interface. The DSK can also be used with an external emulator through the external JTAG connector.

B. *Code Composer Studio (CCS)*

CCS provides an IDE to incorporate the software tools. CCS includes tools for code generation, such as a C compiler, an assembler, and a linker. It has graphical capabilities and supports real-time debugging. It provides an easy-to-use software tool to build and debug programs. The C compiler compiles a C source program with extension .c to produce an assembly source file with extension.*asm*. The assembler assembles an.*asm* source file to produce a machine language object file with extension.*obj*. The linker combines object files and object libraries as input to produce an executable file with extension .*out*. This executable file represents a linked common object file format (COFF), popular in Unix-based systems and adopted by several makers of digital signal processors. This executable file can be loaded and run directly on the C6416 processor. A linear optimizer optimizes this source file to create an assembly file with extension.asm (similar to the task of the C compiler).

Some useful files used in Code Composer Studio, given as:

- file.pjt: to create and build a project named file
- file.c: C source program
- file.asm: assembly source program created by the user, by the C compiler, or by the linear optimizer
- file.sa: linear assembly source program. The linear optimizer uses *file.sa* as input to produce an assembly program *file.asm*
- file.h: header support file
- file.lib: library file, such as the run-time support library file rts6700.lib
- file.cmd: linker command file that maps sections to memory
- file.obj: object file created by the assembler
- file.out: executable file created by the linker to be loaded and run on the C6713 processor
- file.cdb: configuration file when using DSP/BIOS

C. *VM3224K2 LCD display DAUGHTER CARD*

The DSP STAR TFT LCD Video Daughter card (VM3224K2) is a video in/out hardware module, which provides developers with an easy-to-use, cost-effective way to evaluate and develop video processing algorithm based on TMS320C6000TM DSP. The VM3224K2 acquires NTSC/PAL analog video signal and displays digital video data on TFT LCD display screen. This product is a plug-in for the Texas Instruments' C6000 Starter Kit. Since the VM3224K2 complies to TI DSK standard daughter-card interface, the product is also compatible with TI Starter Kit, so with TMS3206713 kit. The VM3224 is embedded over to see the results of processing done in TMS6713 hardware.

The daughter card is also compatible with Image processing algorithms, as we are able to display input and output enhanced images. It is only used to display input and output images. Rest of the processing is done in TMS6713.

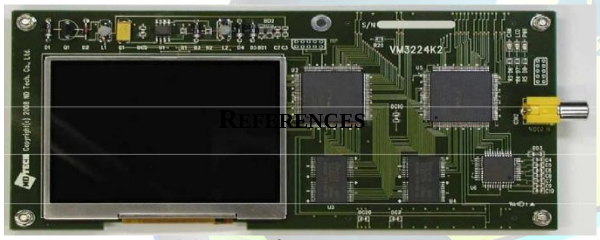


Fig. 4 Board diagram of VM3224K2 LCD Display

The TFT LCD panel uses an RGB565 pixel expression that is 320x240 in size. The LCD panel must provide pixel data periodically according to the pixel array pattern. Hence, the video module contains memory that can store 320x240 pixel data. The module also contains an LCD controller that conveys memory data to the LCD panel in synchronization with the horizontal and vertical sync signals. The LCD controller generates signals to drive the LCD, and the 18-bit address generator generates pixel data addresses directed to the LCD. Note that the 512 KB DRAM is divided into two pages consisting of 256 KB each. One page is used as ongoing picture display buffer while the other is being read by DSP. These two pages of display memory are automatically toggled whenever the high address register is accessed. Thus, the DSP stores image data in the RGB565 format, in order to display it on the TFT LCD. So, output obtained in the project is RGB which is first converted into RGB565 format which is a mandatory for this daughter kit in order to display the output image in the respective kit.

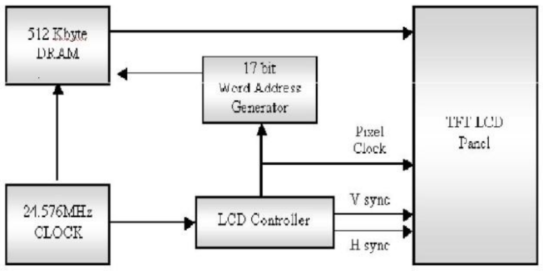


Fig. 5 Structure of the TFT LCD Display

IV. EXPERIMENT AND RESULT

The following section shows the output obtained after successful execution in CCStudio v3.1 and implementation in TMS6416DSK kit. The figures represent the output obtained.

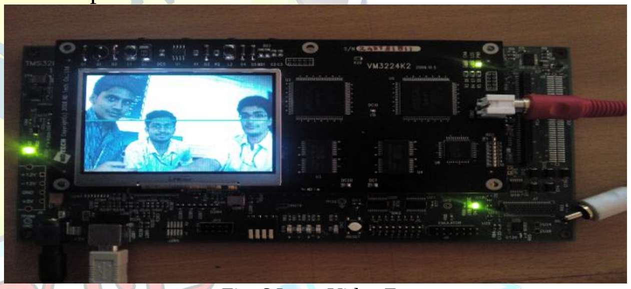


Fig. 6 Input Video Frame

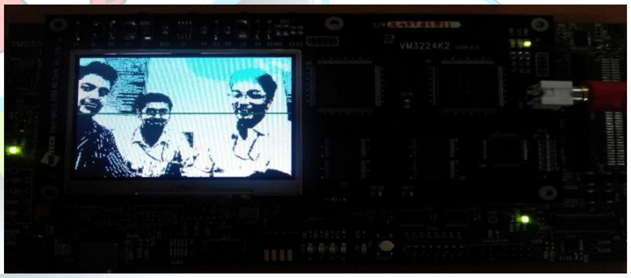


Fig. 7 Cartoonized Video Frame

V. CONCLUSION

In this paper, we have presented a simple method for converting an input live video feed into an output video having cartoon like appearance by thresholding the intensity component using less computational overhead. We have also shown a great implementation of our objective in DSP hardware using C-programming language.

REFERENCE

[1]. Gonzalez, Rafael C. and Woods, Richard E. Digital Image Processing, Pearson, Prentice Hall, Third edition, 2008.

[2]. Texas Instruments, TMS320C6000, Peripherals, Reference Guide, Dallas, TX, March 2000.

[3]. TMS 320C6416T DSK Technical Reference, DSP Development Systems, March 2004.

[4]. TMS 320C6000 EMIF-to-External SDRAM interface, September 2007.

[5]. Song-Hai Zhang, Tao Chen, Yi-Fei Zhang, Shi-Min Hu, Ralph Martin, Vectorizing Cartoon Animations, IEEE Transaction on Visualization and Graphics, 15: 518-629, 2009.

[6]. J.F.Aujol and G.Gilboa and T.Chan and S.Osher, Structure-texture image decomposition modeling, algorithms and parameter selection, International Journal of Computer Vision, Vol 67, No 1, 111-136, 2006.

[7]. A.Chambolle, An algorithm for total variation minimization and applications, Journal of Mathematical Imaging and Vision, Vol 20, No 1-2, 89-97, 2004.

[8]. J.Gilles, Noisy image decomposition: a new structure, texture and noise model based on local adaptivity, Journal of Mathematical Imaging and Vision, Vol 28, No 3,285-295, 2007.

[9]. T.Goldstein and S.Osher, The Split Bregman Method for L1 Regularized Problems, SIAM Journal on Imaging Sciences, Vol 2, No 2, 323-343, 2009.

[10]. U. Clarenz, M. Rumpf, and A. Telea. Robust feature detection and local classification for surfaces based on moment analysis. IEEE Transactions on Visualization and Computer Graphics, 10(5):516–524, 2004.

[11]. Selim Esedo¯glu and Stanley J. Osher. Decomposition of images by the anisotropic Rudin Osher-Fatemi model. Comm. Pure Appl.Math., 57(12):1609–1626, 2004.

[12]. J. B. Garnett, T. M. Le, and L. A. Vese. Image decompositions using bounded variation and homogeneous besov spaces. Technical Report 05-57, UCLA CAM Reports, 2005.

[13]. A. Haddad and Y. Meyer. Variational methods in image processing. Technical Report 04-52, UCLA CAM Reports, 2004.

[14]. A. Haddad and S. Osher. Texture separation BV − G and BV − L1. Technical Report 06-26, UCLA CAM reports, 2006.

[15]. S. J. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation-based image restoration. SIAM Multiscale, Modeling and Simulation, 4(2):460–489, 2005.

[16]. S. J. Osher, A. Sole, and L. A. Vese. Image decomposition and restoration using total variation minimization and the H−1 norm. Technical Report 02-57, UCLA CAM Reports, 2002.

[17]. L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise-removal. Physica D, 60:259–268, 1992.

[18]. R. Schaback and H. Werner. Numerische Mathematik. Springer-Verlag, Berlin, 4te Aufl. edition, 1992.

[19]. O. Scherzer and C.W. Groetsch. Inverse scale space theory for inverse problems. Lecture Notes in Computer Science, Springer, 2106:317–325, 2001.

[20]. J. Shen. Piecewise H−1 + H0 + H1 images and the Mumford-Shah-Sobolev model for segmented image decomposition. Applied Math. Research Exp., 4:143–167, 2005.

[21]. J. E. Taylor, J. W. Cahn, and W. C. Carter. Variational methods for microstructural evolution. JOM, 49(12):30–36, 1998.

[22]. J. Weickert. Anisotropic diffusion in image processing. Teubner, 1998.

[23]. G. Wulff. Zur Frage der Geschwindigkeit des Wachstums und der Aufl¨osung der Kristallfl¨achen. Zeitschrift der Kristallographie, 34:449–530, 1901.