Chroma-Key Algorithm Based on Combination of K-Means and Confident Coefficients

Nguyen Ngoc Tai, Le Quoc Bao Tri, and Truong Quang Vinh

***Abstract*—Chroma-key is an application which is used many in video industries, TV programs. In this paper, we present an efficient algorithm for Chroma-key and its FPGA implementation. In the proposed method, we firstly employ K-means to segment input frames in to a trimap including background, foreground, and unknown regions. In the next stage, we utilize comparison of confident neighborhood pixels to refine unknown region to separate background and foreground regions. In order to apply the proposed algorithm for real-time chroma-key effect, we implement its VLSI architecture on Bitec Cyclone III development board. Our design is able to perform Chroma-key process with pleasing quality HD video in real-time.**

***Index Terms*—Chroma-key, K-means algorithm, FPGA, confident coefficient, hardware architecture.**

I. Introduction

Nowadays, many applications of video industries such as, cinema film, magazines covers, video game industries, television programs, which have become widely to serve entertainment demand, are using Chroma-key effects [1], [2]. Chroma-key is a technique which is extracting a foreground object from a foreground frame and combining with a new background frame to create a new composite frame for a special effect. The foreground frame has two parts includes: foreground and background objects, in which the background object is a solid color, usually green or blue. Fig. 1 is an example of a foreground frame using green screen background and a background frame. The frame (c) is created by using Chroma-key effects.

Fig. 1. Chroma-key effects. (a) Foreground frame. (b) Background frame. (c) Result frame.

The chroma-key can be processed offline or online. The offline process, which is applied for cinema film, magazines covers, and video game, does not require real-time processing. In contrast, the online chroma-key process has to be performed in real-time for direct program such as talk shows, weather forecasting, and news programs.

Some algorithms for offline chroma-key have been presented in [3]-[7]. Jue Wang proposed a chroma-key software based on K-means algorithm and alpha matting [6]. By using alpha matting algorithm and extracting a trimap, the chroma-key effects can segment object of picture exactly. However, the computation of confidence values in alpha matting process is very complicated [8]. Moreover, this process requires many rounds to obtain accurate alpha values. So this algorithm can used as software which cannot process Chroma_key effects in real-time.

Some researchers have proposed real-time Chroma-key method for online applications. In the paper [9], Nguyen Thanh Sang presented Chroma-key algorithm based on improved K-Means method. He used two filters: Croase Filter and Fine Filter to extract the foreground and background region. He choosed green or blue background by hand. However, K-Means based method cannot extract exactly objects from original image. Simple calculations in this paper are not enough to process light, noise, bad quality of background, boundaries which are complicated in foreground image. Moreover, Nguyen Thanh Sang's algorithm processes only 640x480 resolution video which cannot satisfy the demand of currently video industry.

In this paper, we propose a Chroma-key algorithm and its FPGA implementation which can process HD video in real-time and avoid noise in background when extracting the object. The proposed Chroma-key algorithm is based on K-Means and confident coefficients. The K-Means method is used to create trimap including: foreground, background and unknown region. By using highest confident neighborhood pixels, the unknown pixels are processed to determine whether they belong to foreground or background region. Foreground and background coefficients are updated continuously to avoid the effect of noise in original image frames. Moreover, a FPGA implementation for Chroma-key in real-time is also presented in this paper. Our hardware architecture is designed with Avalon interface using full pipeline structure. The completed proposed Chroma-key system on FPGA has been introduced to demonstrate Chroma-key effect for HD video in real-time.

II. Proposed Chroma-Key Algorithm

The main objective of Chroma-key effects is to find a mask which distinguishes pixels of foreground and background. Then, pixels which belong to background region will be replaced by the same position pixels of new background. The equation describes relation of frame and mask is:

$$C = \mathrm{Fg} \times \mathrm{mask} + \mathrm{Bg} \times (\sim \mathrm{mask}) \quad (1)$$

Manuscript received October 23, 2013; revised February 25, 2014.

Nguyen Ngoc Tai, Le Quoc Bao Tri, and Truong Quang Vinh are with the Faculty of Electrical and Electronics Engineering, Ho Chi Minh City University of Technology, Vietnam National University - Ho Chi Minh City (VNU-HCM), Vietnam (e-mail: tqvinh@hcmut.edu.vn, {baotribk, ngoctai0308}@gmail.com).

where *C* is color of the frame after using Chroma-key effect. *Fg* is the original frame, and *Bg* is the new background. "*mask*" is the mask of Chroma-key effect. If pixel belongs to foreground area, "*mask*" is 1. Otherwise, "*mask*" is 0.

We proposed a chroma-key algorithm for FPGA system using K-Means based on confident values. We used *Y*×*Cb*×*Cr* color space to segement blue, green background, because *Cr* component of green background pixels is much less than ones that belong to the foreground object region and *Cb* component of blue background pixels is much greater than ones that belong to the foreground object region. We assume that background is only blue or green, which usually uses in video industries. The proposed algorithm includes three main parts as follows:

- At the beginning, the system calculated two centroids for background and foreground.
- Subsequently, the trimap is created by using proposed K-Means method and safe threshold. The trimap includes 3 regions: foreground, background and unknown.
- At the end of process, the unknown pixels are classified into background or foreground pixels by using estimated confident coefficients.

A. Automatically Calculate Two Centroids

Main purposed of this process is choosing background (green or blue) automatically and define two centroid values. The algorithm supposes pixel (1, 1) belong to background region, because main object of scene is usually put in middle frame. The problem of Nguyen Thanh Sang's algorithm [9] is selecting blue or green background by hand. So, we proposed one way to determine background blue or green.

The different values of background blue and green are *Cb* values. The background blue has low *Cr* value and high *Cb* value. In another way, the background green has low *Cr* and *Cb* value. Therefore, *Cb* value is used to determine green or blue background. In our system, we define background values is low, and foreground value is high. So, if the background is green, the system processes Chroma-Key effects using *Cr* value. In contrast, the system processes Chroma-Key effects using (255 - *Cb*) value.

TABLE I: THRESHOLD OF BACKGROUND

Cb (1,1) Value	Background Type
>128	Blue
<128	Green

Table I shows the way to detect background type. We choose value "128" because it is average of 0 and 255. By using *Cb* value, the system can detect background green or blue. After that, *Cr* value (green) or *Cr* value (blue) is used to process in our Chroma_key algorithm. To define first foreground centroid (F_f), the system uses B_f and adding deviation between background and foreground by using following equation:

$$F_f = B_f + dev \tag{2}$$

B. Using Proposed K-Means Method to Extract Trimap

Table II Shows how to established pixel region of trimap. In Table II, we used two centroids (D_{FC}, D_{CB}) of the pixel *C* to *F* and *B*.

TABLE II: COMPARE THE DISTANCE TO DETERMINE TRIMAP

Trimap region of pixel	Compare the distance
Background	$D_{FC} - D_{CB} > U$
Foreground	$D_{FC} - D_{CB} < U$
Unknown	Other

Two centroids *F* and *B* are calculated by following method:

$$\begin{aligned} D_{FC} &= \|F - C\|^2 \\ D_{CB} &= \|C - B\|^2 \end{aligned} \tag{3}$$

where D_{FC}, D_{CB} are the distances between foreground – pixel and background – pixel. *F* is foreground confident value, *B* is background confident value, *C* is processing pixel value. To extract the trimap, the system using coefficient for unknown region which is calculated from "*F*-*B*" value:

$$U = \alpha(F - B) \tag{4}$$

where "*U*" is confident distance of unknown region, α is estimate value of deviation between foreground and background. *U* is updated continuously, because *F* and *B* coefficients are updated continuously. To determine pixel belong to background or foreground, the difference between D_{FC} and D_{CB} is higher *U*.

To increase precision of calculating, updating F and B values is important stage. Following Nguyen Thanh Sang's method, he using only two centroids in coarse filter, and then updated by average values of one line (K_l) by using equation: $K = \frac{\sum K_l}{n_l}$. Coarse filter using only two centroids will have large errors in frames. Fine filter which is calculated from coarse filter has higher errors. So, we proposed average method to update *F* and *B* value in equation (3) by using average rely values F_i and B_i (F_i , B_i are foreground and background pixels which are calculated most recent). We presented following method to evaluate more accuracy.

$$\begin{aligned} F &= \frac{\sum \gamma_F * F_i}{n_F} \\ B &= \frac{\sum \gamma_B * B_i}{n_B} \end{aligned} \tag{5}$$

where γ_F and γ_B are confident coefficient of F_i , B_i . $n_F = \sum \gamma_F$, $n_B = \sum \gamma_B$. γ and *n* is calculated suitable for VLSI architectures which is target of this algorithm. So we used $n_F = 2^n$ and $n_B = 2^n$ (n=8), and γ_F , γ_B is divided 5 levels. γ_F and γ_B are established by following Table III. where *B*, *F* are average values of background and foreground which are updated from equation (5). *C* is value of processing value. T1, T2 is threshold of background and foreground which can define from Table II.

TABLE III: CALCULATING FIVE LEVELS OF GAMMA

Compare C with B value	γ_B	Compare C with F value	γ_F
$C \le B$	1	$C \ge F$	1
$B < C \le \frac{T1-B}{4}$	0.75	$F > C \ge \frac{3(T2-F)}{4}$	0.75
$\frac{T1-B}{4} < C \le \frac{T1-B}{2}$	0.5	$\frac{3(T2-F)}{4} > C \ge \frac{T2-F}{2}$	0.5
$\frac{T1-B}{2} < C \le \frac{3(T1-B)}{4}$	0.25	$\frac{T2-F}{2} > C \ge \frac{T2-F}{4}$	0.25
$\frac{3(T1-B)}{4} < C \le T1$	0	$\frac{(T2-F)}{4} > C \ge T2$	0

C. Processing Unknown Pixels by Using Neighborhood Pixels

Unknown region has foreground pixels mixed background pixels, so processing unknown region is a challenge. The values for calculating unknown pixels require highest accuracy.

TABLE IV: CALCULATING F AND B FROM 5×5 MATRIX

Distance of max B and min B	B_n	Distance of max F and min F	F_n
$(B_{\max} - B_{\min}) > X$	$B_n = \frac{\sum B_i}{n_i}$	$(F_{\max} - F_{\min}) > X$	$F_n = \frac{\sum F_i}{n_i}$
$(B_{\max} - B_{\min}) < X$	$B_{\max}$	$(F_{\max} - F_{\min}) < X$	$F_{\max}$

Although, background is one color (blue or green) but the intensity is not constant because of background quality, noise, light, angle of rotation, and etc. Foreground may include many texture, so it have big different between them.

To optimize usage of processing memory and buffer memory, we proposed 5×5 matrix nearby unknown pixels for assumed values. In 5×5 matrix, values of foreground and background have fluctuation. So, we proposed two methods to calculate and values following Table IV, where X is threshold of fluctuation.

TABLE V: COMPARE TWO CENTROIDS TO SELECT UNKNOWN PIXELS

Compare two centroids	Unknown pixel type
$\|F_n - C\|^2 < \|C - B_n\|^2$	Foreground
$\|F_n - C\|^2 \ge \|C - B_n\|^2$	Background

Comparing two centroids will decide unknown pixel belongs to foreground or background as following Table V. Where C is value of unknown pixel.

After processing unknown pixels, we have mask of frame (background: 0, foreground: 1). The result is determined by using following equation (6). Where $C(i, j)$ is color of result frame after using Chroma-key algorithm, $Fg(i, j)$ is value of original frame, $Bg(i, j)$ is value of new background.

$$C(i,j) = Fg(i,j) \times (\sim mask) + Bg(i,j) \times mask \quad (6)$$

III. HARDWARE ARCHITECTURE

Top block of hardware design of Chroma_key is shown in Fig. 2.

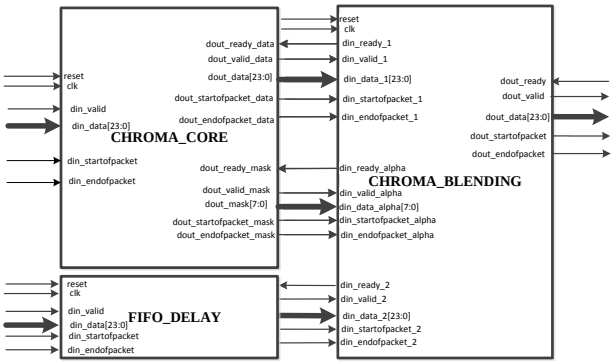

Fig. 2. Top blocks of hardware design of Chroma-key.

We applied full pipeline structure for Chroma_key system. All blocks of Chroma_Key_Core are synchronized by using reset signal and clock signal. The system processed two in data flows with 24bits for each, and 24bits output data flow. The system is designed to complied Avalon_ST interface. Top block hardware includes: Chroma_Core block, Chroma_Blending block and Fifo_Delay block. Chroma_Core block processes foreground input data (24bits) and Fifo_Delay block delays new background input data (24bits). After processing from Chroma_Core block, the mask of Chroma_Key algorithm is created. Combining foreground and background video based on the mask, Chroma_Blending block creates the result of Chroma_key algorithm.

A. Design of Chroma_Core

The design of Chroma_Core block is divided into 7 sub-blocks: Avalon_ST_Sink block, 3 blocks Fifo, Chroma_processing block, 2 blocks Avalon_ST_Source as shown in Fig. 3.

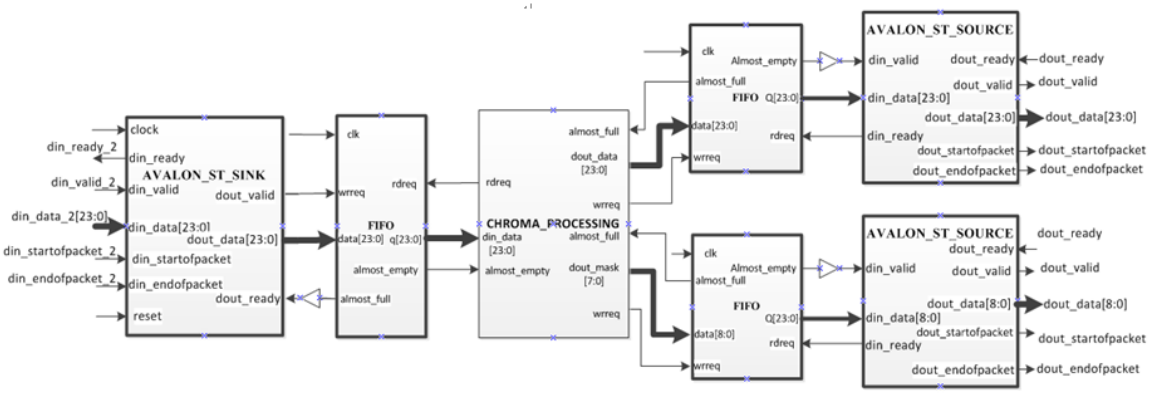

Fig. 3. Desgin of Chroma-Core block.

Our VLSI architecture is designed using Avalon_ST interface, so we designed Avalon_ST_Sink and

Avalon_ST_Source at begin and end of processing block. Fifo blocks which contain processing data are used in our design. Chroma_processing block which keep Chroma_Key algorithm, is main block of Chroma_Core block. Chroma_processing block is divided into 4 sub-blocks: Key_Selection, Segmentation, Fifo_Delay and Unknown_Processing as shown in Fig. 4.

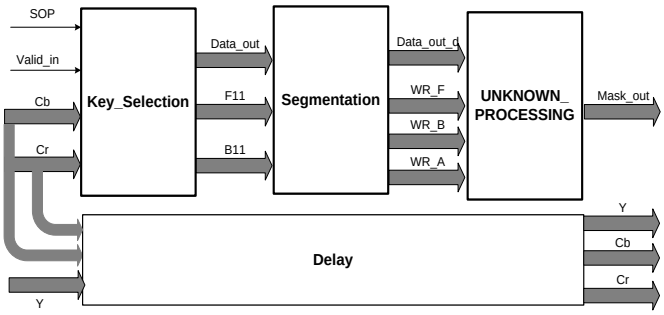

Fig. 4. Design of Chroma_Processing block.

Key_Selection block is used to detect background (blue or green) automatically, and calculate two first centroids (F_f, B_f). Using coefficients from Key_Selection block, the Segmentation extracts pixels of frame to trimap (foreground, background, unknown region).

After extracting trimap, Unknown_processing block calculates unknown pixels and decides that pixels are foreground or background pixels.

After processing by Chroma_Core block, we have foreground data and mask which contains information to detect foreground or background pixels.

B. Design of Chroma_Blending

At the end of the process, Chroma_Blending block which blend foreground and background frame using the mask, is introduced. Input data of Chroma_Blending blocks includes: foreground data (24bits) and the mask (8bits) which created from Chroma_Core block, background data (24bits) which delayed by using Fifo_Delay. The design of Chroma_Blending block is divided into 5 sub-blocks: 3 blocks Avalon_ST_Sink, Chroma_Blending_Processing block, Avalon_ST_Source block as shown in Fig. 5.

To response Avalon_ST interface, Chroma_Blending block uses 3 blocks Avalon_ST_Sink for 3 data flows at the begin and 1 block Avalon_ST_Source at the end of block. Module Chroma_Blending_Process is designed to blend two data flows by using equation (6). The structure of this module is shown in Fig. 6.

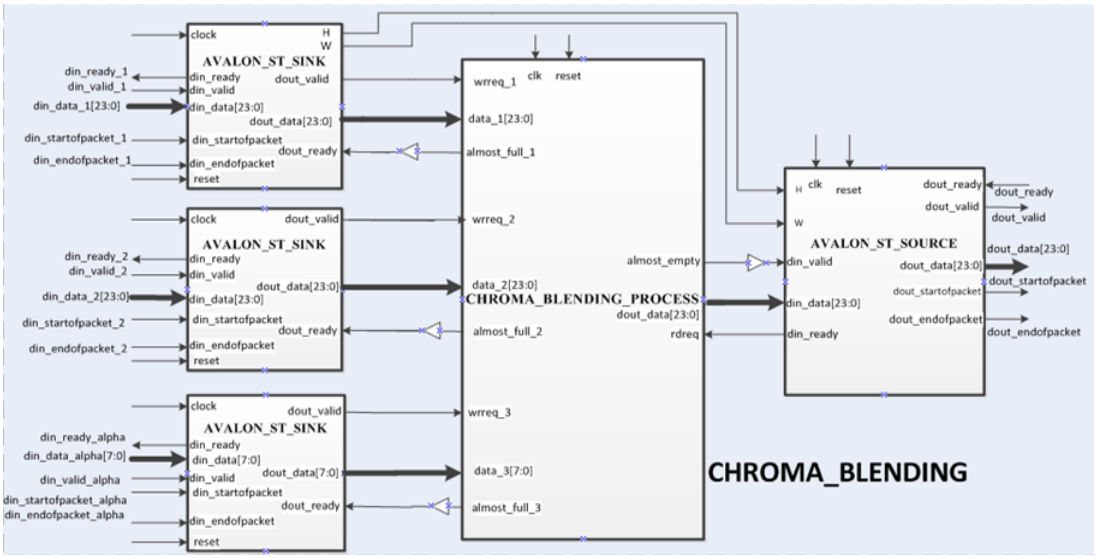

Fig. 5. Design of Chroma_Blending block.

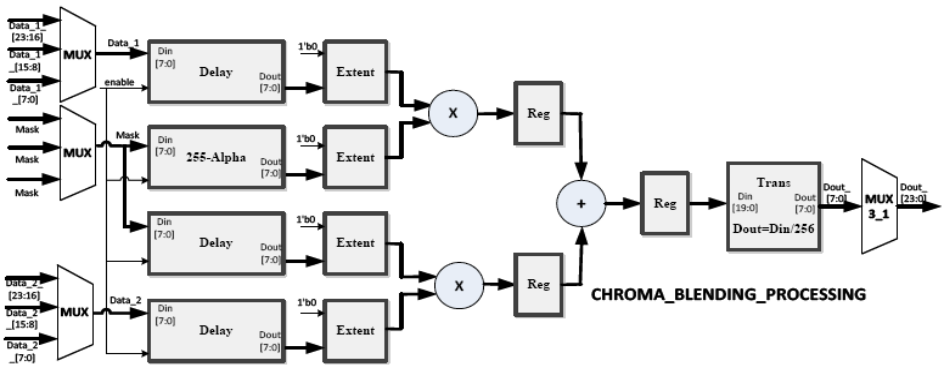

Fig. 6. Chroma_Blending_Processing block.

IV. Experimental Results

We implemented the proposed hardware architecture on Bitec Cyclone III development board. Fig. 7 shows our verification model for the Chroma_Key design. Verilog language is used to describe the hardware architecture.

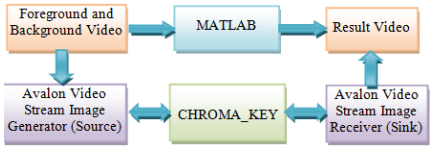

Fig. 7. Testing environment.

The Chroma_Key design was packaged as an IP core using Avalon_ST standard. Matlab software is used to verify Chroma-key algorithm and compare with hardware design. We used Cyclone III to verify the hardware design follow Avanlon interface. The foreground and background video is test using Matlab. Two result video is combined together.

A. Hardware Synthesis Result

A FPGA design for Chroma-key effect is general by using Quatus II software for the Bitec Cyclone III. We used Modelsim software to simulate, and Bitec Cyclone III development board to display experiment. A comparison among our system and other system which is also executed on FPGA is shown on Table VI. According to the table, our hardware design has some advantage characteristics with other system. The FPGA design can operate at 111.71 MHz, and process HD resolution video.

TABLE VI: SYNTHESIZE HARDWARE COMPARISON BETWEEN THE PROPOSED DESIGN AND OTHER DESIGN

Features	[9]	Proposed
Device	Cyclone II	Cyclone III
Chroma-key algorithm	Croase Filter and Fine Filter	K-Means combining confident coefficients
Maximum frequency	40.33 MHz	111.71 MHz
Logic cell usage	3086	25k
Memory bit usage	86282	1963k
Resolution	640x480	1920x1080

B. Chroma-Key Implementation Results

We used many different background types for testing. The test frames are 24-bit RGB images with 640x480 resolutions. We give a comparison between our algorithm and Nguyen Thanh Sang's algorithm. The experimental results of Nguyen Thanh Sang's algorithm come from the version with default parameters provided by the author. In the proposed algorithm, we set parameter dev to 40.

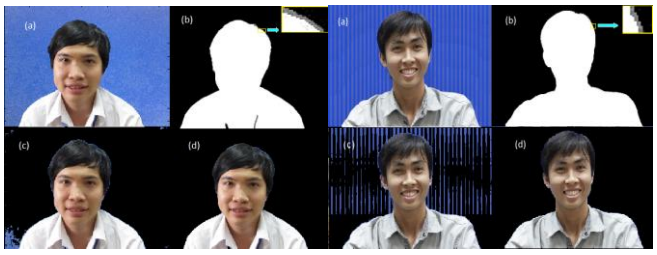

Fig. 8. Comparison of Nguyen Thanh Sang's algorithm and our algorithm. (a). Original image. (b). Our Trimap. (c). Nguyen Thanh Sang's algorithm result. (d). Our result.

In Fig. 8, Nguyen Thanh Sang's algorithm cannot solve problems in which the background contains noise or large difference of background color. The reason is that Nguyen Thanh Sang's algorithm used only one coefficient in Coarse Filter and updated value using wrong samples from Coarse Filter. Our design is able to solve that problem by using confident coefficients.

V. CONCLUSION

A Chorma-key algorithm has been proposed and analyzed. We also have proposed a Chroma_Key hardware architecture in real-time. The algorithm is based on combination between improved K-Means and confident coefficients which are useful for Chroma_Key algorithm. The Chroma_Key hardware architecture is designed by using pipeline structure. Therefore, the design can process real-time HD video.

REFERENCES

[1] A. R. Smith and J. F. Blinn, "Blue Screen Matting," *SIGGRAPH*, pp. 259–268, 1996.
[2] C. Kim, J. Park, J. Lee, and J. N. Hwang, "Fast extraction of objects of interest from images with low depth of field," *ETRI Journal*, vol. 29, no. 3, 2007.
[3] F. van den Bergh and V. Lalioti, "Software chroma keying in an immersive virtual environment," *South African Computer Journal*, pp. 155-162, 1999.
[4] K. Dimitropoulos, "Improve 3D video synthesis combining graph cuts and chroma key technology," in *Proc. 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, 2010, pp. 1-4.
[5] B. E. Moshe, "Segmentation with invisible keying signal," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2000, pp. 32–37.
[6] R. Gvili, "Depth keying," in *Proc. SPIE-IS&T Electronic Imaging*, 2003, pp. 564-574.
[7] B. Vidal, "Chroma key visual feedback based on non-retroreflective polarized reflection in retroreflective screens," *IEEE Transactions on Broadcasting*, vol. 58, no. 1, pp. 144-150, March 2012,.
[8] J. Wang and M. F. Cohen, "Optimized color sampling for robust matting," in *Proc. Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, 2007, pp. 1-8.
[9] N. T. Sang and T. Q. Vinh, "FPGA implementation for real-time chroma-key effect using coarse and fine filter," in *Proc. Computing, Management and Telecommunications (ComManTel), 2013 International Conference on*, Jan. 2013, pp. 157-162.

Nguyen Ngoc Tai received the bachelor of electronics – telecommunications engineering degree from Ho Chi Minh City University of Technology in April 2012. Currently, he is a researcher at Electronics Department, Faculty of Electrical and Electronics Engineering, HCMUT. His research interests include include VLSI design on FGPA, ASIC design, video and image processing, computer vision.

Le Quoc Bao Tri received the bachelor of electronics – telecommunications engineering degree from Ho Chi Minh City University of Technology in April 2012. Currently, he is a researcher at Electronics Department, Faculty of Electrical and Electronics Engineering, HCMUT. His research interests include include VLSI design on FGPA, ASIC design, video and image processing, computer vision.

Truong Quang Vinh received the B.E. in electronics from Ho Chi Minh University of Technology, Vietnam in 1999, the M.E. in computer science from the Asian Institute of Technologies, Bangkok, Thailand in 2003, and Ph.D. in computer engineering from Chonnam National University, Korea in 2011. He is currently a faculty member of Department of Electronic Engineering at Ho Chi Minh University of Technology. His research interests include VLSI design on FGPA, ASIC design, embedded system/system-on-chip design, and video processing.