Design of Simulink Model for Real Time Video/Image Splitting

[1,] Naveen B.,[2,] Dr. K.R. Nataraj., [3,] Dr. K.R. Rekha
[1,]Department of ECE, Jnana Vikas Institute of Technology, Bangalore.
[2,] Department of ECE, SJB Institute of Technology, Bangalore.
[3,] Department of ECE, SJB Institute of Technology, Bangalore.

Abstract

For split the real time image/video of size P x Q into four image/video blocks. Each splitted image/video blocks are interpolated to the dimension of the original image without blurring and display video/image in four different screens. Video splitting is the process of dividing the video into non overlapping parts. In this paper , techniques for splitting an image are implemented using MATLAB simulink.

Keywords: Image, Video, Splitting, Simulink.

1. Introduction

Video is the technology of electronically capturing, recording, processing, storing, transmitting, and reconstructing a sequence of still images representing scenes in motion. Video Splitting is the process of dividing the video into non overlapping parts. Then row mean and column mean or each part is obtained. By using splitting higher precision and display in different Screen [1,2]. An image is defined as a two dimensional function, *f(x, y),* where *x* and *y* are spatial coordinates, and the amplitude of *f* at any pair of coordinates *(x, y)* is called the intensity or gray level of the image at that point. When *x, y* and the intensity values of *f* are all finite, discrete quantities, we call the image a digital image [3]. Digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are called picture elements, image elements, pels, and pixels. Pixel is a term used most widely to denote the elements of a digital image. Pixels are normally arranged in a two dimensional grid and are often represented using dots or squares. Number of pixels in an image can be called as resolution. Associated with each pixel is a number known as Digital Number or Brightness Value that depicts the average radiance of a relatively small area within a scene. The matrix structure of the digital image is shown in Fig 1.

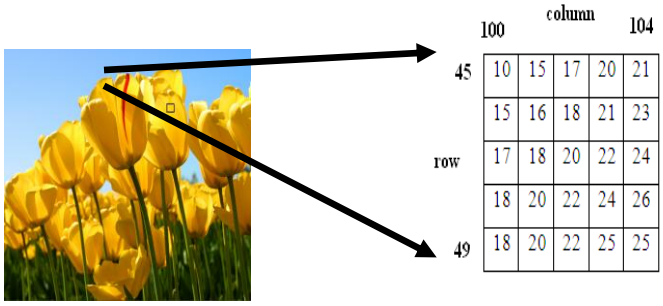


Fig 1: Structure of Digital Image

In Fig 2. These block diagram the input video image with resolution P X Q is applied to the image splitting system. After processing in image splitting system the video image is displayed on the four different LCD screens. Each splitted frame is interpolated to the original input video image resolution. a modified version of classical split and merge algorithm. Instead of performing a regular decomposition of image, it relies on a split at an optimal position that does a good inter region separation [4].The implementation of the algorithm uses an initial image processing to speed up computation. An interpolation algorithm for 3-D reconstruction of magnetic resonance images [5]. This investigation evaluates various interpolation algorithms for generating "missing data between multiple, 2D magnetic resonance image plane. Slices when stacked parallel and displayed, represents a 3D volume of data usually with poorer resolution in the third dimension. Interpolation algorithm have been developed to determine the missing data between the image slices and to compute a 3D volume data array representing equal spatial resolution in three dimension . An enhanced pixel-based video frame interpolation algorithms [6].

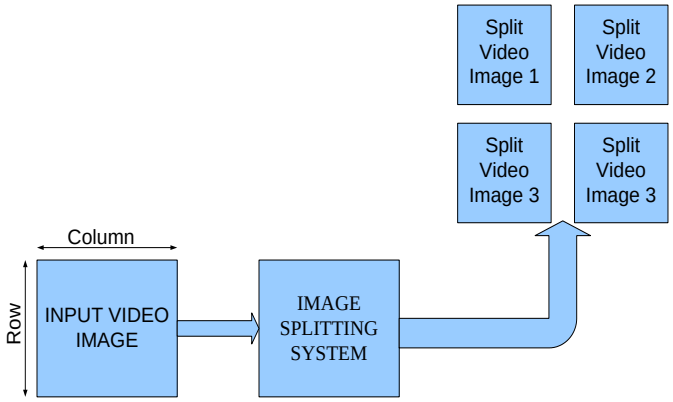


Fig 2: Block Diagram of the Video/Image Splitting

In paper [7] compares three motion compensated interpolation algorithms: adjacent frame motion compensated interpolation (AFI), and wide span motion compensated interpolation (WS-TH) and wide span motion compensated interpolation with spatial hinting. The latter represents an extension to WS-TS by adding spatial hinting to the generation of motion vectors. a design methodology for implementing DSP with Xilinx system generator for Matlab [7]. This methodology aims at improving the efficiency of learning the use of complex designs. This paper is organized as follows. Section 2 gives a Brief overview of the Image/video Splitting. Section 3 presents the Simulink parameters for the Image/video Splitting. Section.4 gives the design of simulink for proposed real time Image/video Splitting. For Simulink results are presented in Section 5. A conclusion and future work is given in Section 6.

2. Proposed Block Diagram Of Image/Video Splitting

The block diagram of the design is as shown in Fig. 3. Video from the web camera at 30 frames per second is applied to the video input block. Resize block enlarge or shrinks image size. Captured video will be in RGB format. It is converted into croma and luma components. Luma represents the brightness in an image and it represents the achromatic image without any color while the croma component represents the color information. Image split block splits the image into number of blocks. Each spitted block is resized using bilinear interpolation technique. The split image is displayed using the video display output block

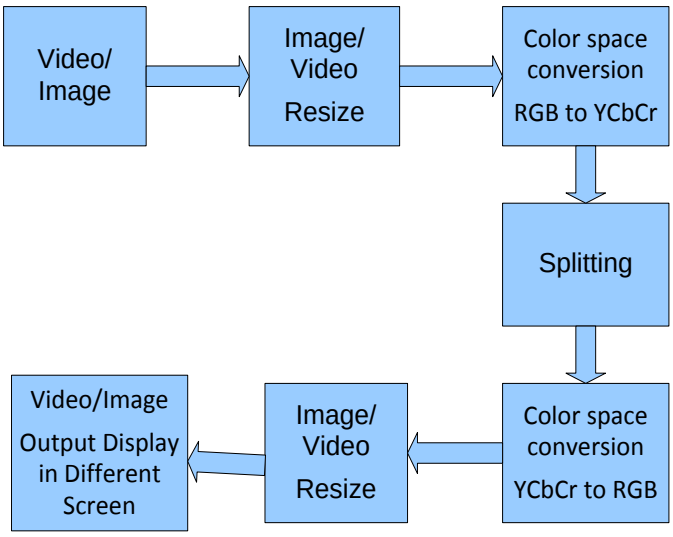


Fig. 3: Block Diagram of Image/video Splitting

3. Simulink Parameters

The Design is carried out using video and image processing blockset in MATLAB simulink version R2007b. here how images are spitted and various parameters required achieving the desired task.

A. Image From File:

Image From camera

Image From camera device is used to import an image from a supported image file. If the image is a M-by-N array, the block outputs a binary or intensity image, where M and N are the number of rows and columns in the image. If the image is a M-by-N-by-P array, the block outputs a color image, where M and N are the number of rows and columns in each color plane, P. For Video and Image Processing Blockset blocks to display video data properly, double- and single-precision floating-point pixel values must be between 0 and 1[2]. Use the Image signal parameter to specify how the block outputs a color video signal.

B. From Video Device:

From Video Device

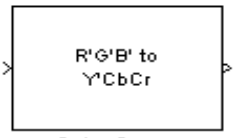
The From Video Device block allows acquiring image and video data streams from image acquisition devices, such as cameras and frame grabbers, in order to bring the image data into a Simulink model [2]. The block also allows configuring and previewing the acquisition directly from Simulink. The From Video Device block opens, initializes, configures, and controls an acquisition device.The opening, initializing, and configuring occur once, at the start of the model's execution. During the model's run time, the block buffers image data, delivering one image frame for each simulation time step. The block has no input ports. Block can be configured to have either one output port, or three output ports corresponding to the uncompressed color bands, such as red, green, and blue, or Y, C_b, C_r.

C. Resize:

Resize

The Resize block enlarges or shrinks an image by resizing the image along one dimension (row or column). Then, it resizes the image along the other dimension (column or row). If the data type of the input signal is floating point, the output has the same data type. Use the specify parameter to designate the parameters to use to resize the image. Important choices in this block are Output size as a percentage of input size, Number of output columns and preserve aspect ratio, Number of output rows and preserve aspect ratio, Number of output rows and columns. If, for the Specify parameter, we can select Output size as a percentage of input size, the Resize factor in % parameter appears in the dialog box.

D. Color Space Conversion:



Color Space
Conversion

The R'G'B' to Y'CbCr conversion and the Y'CbCr to R'G'B' conversion are defined by the following equation

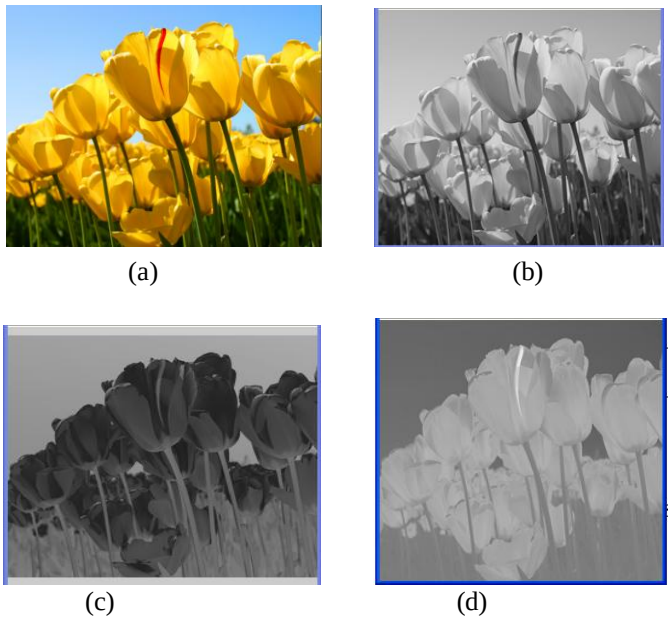
$$\begin{bmatrix} Y' \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + A \times \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = B \times \left(\begin{bmatrix} Y' \\ Cb \\ Cr \end{bmatrix} - \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \right)$$

The values in the A and B matrices are based on your choices for the Use conversion specified by and scanning standard parameters.

Why RGB To YC_bC_r Conversion:

A Bitmap image uses the RGB Planes directly to represent color images. Medical research proved that the human eye has different sensitivity to color and brightness. Thus there came about the transformation of RGB to YC_bC_r. Medical investigation shows that there are rods are 120 million in number and cons are 6-7 million. Rods are much more sensitive than cons. The rods are not sensitive to color, while the cons which provide much of the eyes sensitivity are found to be located close to a central region called the mocula. And another reason is conversion reduces the simulation time in other words increases the data transfer rate. Fig. 4. a, b, c, d shows RGB, Y, C_b, C_r image respectively.

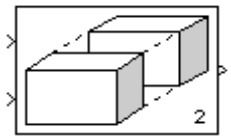


(a) (b)

(c) (d)

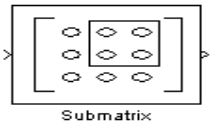
Fig 4: Color Representation (a) RGB image (b) Y Component (c) Cb Component (d) Cr Component

E. Matrix Concatenation*:*



The Concatenate block concatenates the signals at its inputs to create an output signal whose elements reside in contiguous locations in memory. This block operates in either vector or multidimensional array concatenation mode, depending on the setting of its Mode parameter. In either case, the inputs are concatenated from the top to bottom, or left to right, input ports. In vector mode, all input signals must be either vectors or row vectors [1xM matrices] or column vectors [Mx1 matrices] or a combination of vectors and either row or column vectors.

F. Submatrix Selection:



The Submatrix block extracts a contiguous submatrix from the M-by-N input matrix u as shown in Fig.5. The Row span parameter provides three options for specifying the range of rows in u to be retained in submatrix output y:

1. All rows: Specifies that y contains all M rows of 'u'.
2. One row: Specifies that y contains only one row from u. The Starting row parameter is enabled to allow selection of the desired row.
3. Range of rows: Specifies that y contains one or more rows from u. The Row and Ending row parameters are enabled to allow selection of the desired range of rows.

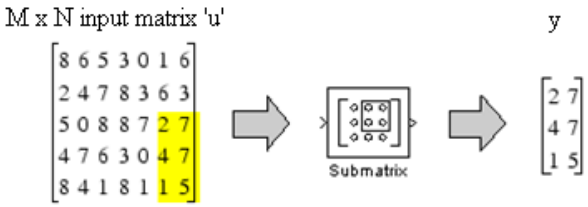


Fig. 5: Submatrix Selection

The Column span parameter contains a corresponding set of three options for specifying the range of columns in u to be retained in submatrix y: All columns, one column, or Range of columns. The One column option enables the Column parameter, and Range of columns options enables the Starting column and Ending column parameters. The output has the same frame status as the input.

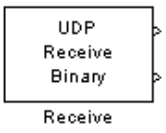
G. To Video Display:

To Video Display block sends video data to a DirectX supported video output device or video camera. Alternatively, we can send the video data to a separate monitor or view the data in a window on your own computer screen. For the block to display video data properly, double- and single-precision floating-point pixel values are from 0 to 1. For any other data type, the pixel values must be between the minimum and maximum values supported by their data type. Use the Video output device parameter to specify where we want the video stream to be sent. If On-screen video monitor is selected then video stream is displayed in the To Video Display window when we run the model. This window closes automatically when the simulation stops.

H. UDP Send:

The Send block has only one input port, which receives the uint8 vector that is sent as a UDP packet.IP address to send to specify the IP address to send the packet. IP port to send to specify the port to which to send the packet. Use the following local IP port Set this parameter to -1 (default) to allow the networking stack to automatically determine the local IP port that is used for sending. Otherwise, specify a particular port to send a packet from that port. Sample time You can set this parameter to -1 for an inheritable sample time, but it is recommended that this be set to some specific (large) value to eliminate chances of dropped packets. This is especially true when you are using a small base sample time.

I. UDP Receive:



The Receive block has two output ports. The first port is the output of the received data as a vector of uint8 while the second one is a flag indicating whether any new data has been received. This port outputs a value of 1 for the sample when there is new data and a 0 otherwise. The default behaviour of the Receive block is to keep the previous output when there is no new data. You can modify this behaviour by using the second port to flag when there is new data. IP address to receive from can be left with the default value of 0.0.0.0. This accepts all UDP packets from any other computer. If set to a specific IP address, only packets arriving from that IP address are received. IP port to receive from port that the block accepts data from. The other end of the communication sends data to the port specified here. Output port with width of the acceptable packets. You can obtain this when designing the other side (send side) of the communication. Sample time You can set this parameter to -1 for an inheritable sample time, but it is recommended that this be set to some specific (large) value to eliminate chances of dropped packets. This is especially true when you are using a small base sample time.

4. Simulink Design For Real Time Video Image Splitting

The simulink model for splitting the image into four parts is as show in Fig.6. The original real time image in RGB format. Resize block resizes the image to the specified value say 256 x 256.

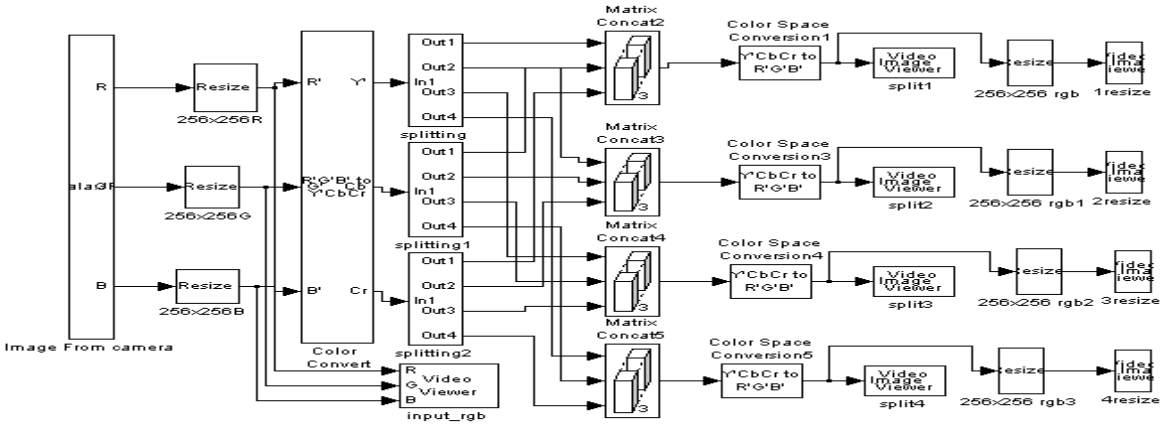


Fig 6 : Simulink Design for Video Image Splitting

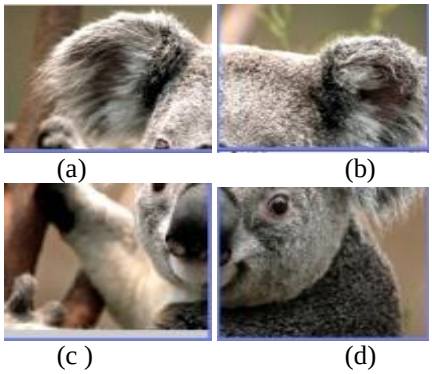
Color conversion block converts the input RGB format to YC_bC_r format. Y stands for intensity component (luma). C_b and C_r stands for blue difference and red difference (croma) components respectively. These croma and luma components is applied to the splitting block. Splitting block is a subsystem which is having one input and four outputs. This splitting block consists of submatrix block. Submatric is used to select the image matrix. Each Splitting block contains four outputs of dimension 128 x 128. One output from each splitting block is concatenated.ie first output from splitting block, splitting1 block and splitting2 block respectively. When we do this we will get one entire image. Similarly this procedure is carried out for all the outputs of splitting block. This image is in YC_bC_r format. The image is converted back to RGB format using color conversion block. Splitted bocks are resized to the original dimension of the input image. For resizing we use bicubic interpolation technique. Resized images are displayed using video display blocks.

5. Simulink Model Results

The results of the software reference model are shown below. The input image for the reference model is in RGB format. Its dimension is 256 x 256 shown in Fig. 7.

Fig 7: Input Image(256 x 256)

The input image/video of the software reference model is splitted as shown in Fig.8. Even the splitted image in RGB format and having dimension 128 x 128.



(a) (b)

(c) (d)

Fig 8: Splitted Image 128 x 128

6. Conclusion and Future Work

We proposed Video image splitting, in that first step in this work was to split the image using the Matlab simulink. Simulink design is carried out using the sub matrix block to split the image. Then this splitted image is resized using bilinear interpolation technique. Splitted image is sent to four different pc's by using simulink. In future the design can be extended for eight splitted blocks to different LCD screens to have a closer look of the spot also it can be implemented on system generator and FPGA.

References

[1] Cyril Prassana Raj P, Dr. S.L. Pinjare, and Swamy.T.N, **"FPGA implementation of efficient algorithm of image splitting for video streaming data"**, International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 2, Issue 5, page 1244-1247,October 2012.

[2] Naveen.B, Dr. K.R. Nataraj, and Dr. K .R. Rekha , **"Design and Analysis of Real Time Video/Image Splitting Using Matlab "**, International Conference IRD India at Bangalore, India page 46-48, Dec 2012.

[3] Rafael C. Gonzalez and Richard E. Woods, **"Digital Image Processing"**, Pearson Prentice Hall, 3 Edition, 2008.

[4] Alain Merigot, **"Revisiting image splitting"**, Proc of 12th international conference on image analysis and processing, page 314-319, 2003.

[5] Marco Aurelio, Nunu maganda, **"Real time FPGA based architecture for bi cubic interpolation: an application for digital image scaling"**, Proc of the International conference on reconfigurable computing and FPGAs.

[6] Belgacem Ben Yousef and Jim Bizzochi, **"Enhance Pixel-Based Video Frame Interpolation Algorithms"**, IEEE International symposium on Signal Processing and Information Technology, page 23-28, Dec 2007.

[7] Matthew Own by, Dr Wagdy.H. mhmoud, **"A design methodology for implementing DSP with xilinx system Generator for Matlab"**, processdings of 35th south eastern symposium, Vol 15, page 2226-2238, 2006.