AC 2008-99: A COLOR IMAGE MERGING ALGORITHM USING MATLAB

Eric Boyer, Pennsylvania State University-Harrisburg

Mr. Boyer is now a graduate student in the Master of Engineering Program, Electrical Engineering at Penn State University at Harrisburg.

Aldo Morales, Pennsylvania State University-Harrisburg

Dr. Morales received his electronic engineering degree with distinction from the University of Tarapaca, Arica, Chile, and M.S. and Ph.D. degrees in electrical and computer engineering from the State University of New York at Buffalo. His research interests are digital signal and image processing, and computer vision. He is now an Associate Professor of Electrical Engineering at Penn State University at Harrisburg.

A Color Image Merging Algorithm Using MATLAB

Eric Boyer and Aldo Morales
Electrical Engineering Program
Penn State Harrisburg
Middletown, PA 17057

Abstract:

Students in the Electrical Engineering program at Penn State Harrisburg have many opportunities to apply their acquired knowledge through hands-on course projects and laboratory experiences in electronics, digital and image processing, VLSI, power and other courses, in addition to their capstone senior project. This paper presents an example of an image processing course project in which an efficient algorithm to merge color images is developed using MATLAB. The algorithm's efficiency is based on the fact that only one color plane is used to detect similarities between images before the merging is accomplished. Results show that the proposed method performs well with sample images obtained from the Internet.

I. Introduction

The Electrical Engineering program at Penn State Harrisburg provides an opportunity for students to pursue interests in electrical and electronic circuits, including digital circuits and VLSI and its fabrication, microprocessors and their applications, electromagnetics, communications, control systems, digital signal/image processing and computer vision[1]. Typically, students demonstrate acquired knowledge through performance on exams, quizzes, homework and course projects. In this paper, we will present an efficient algorithm for color image merging based on an image processing course project.

Image merging of two images into a single image can be done using different methods. In A. German et al [2] entropy was used as the basis to merge images with different exposure and lighting conditions. Scheunders and De Backer[3] introduced multispectral image wavelet representation to merge Lanstat Thematic Mapper images. N. Soussi and J. L. Biuat[4] escribed simultaneous display, edges and structure superimposition, transparency, and hierarchical image merging for medical applications. However, note that some of these techniques are beyond an undergraduate level course in image processing. This paper will describe an efficient and simple procedure to produce a large merged image using MATLAB and will present the necessary code to implement it. The overall goal is to acquire several small and more detailed images of large objects and then compose a larger image file by combining these small image files. It is understood that large objects cannot be imaged with any great detail. The algorithm references two different images to be merged. The first image is a database image. The second image in the algorithm is the image to be added to the database image. The algorithm starts by extracting a single color plane from both color RGB images. The next step is to sample the pixel intensity values of a specified window in the second image and to store in a matrix the pixel intensities within this window. The contents of this matrix are then used to find a similar arrangement of pixel intensities in the database image. The algorithm starts at the left

uppermost pixel in the database image. Then it takes the absolute value of the pixel differences between a single color plane, in a database image matrix, and the window matrix acquired by sampling a piece of the second image. These differences are stored into another matrix for future processing. After these values are calculated, the window moves to the very next pixel in the image. This process continues until all pixels in the image are analyzed, excluding the pixels around the border due to window width constraints. After every pixel position has an effective difference associated with it, the minimum difference is located in the difference matrix. Once this pixel position is known, the two images are merged using this specific point as a reference. Results show that color images can be successfully and efficiently merged.

This paper is organized as follows: in Section II, the basics color imaging will be reviewed. The algorithm and results will be presented and discussed in Sections III and IV, respectively. Conclusions will be presented in Section V. This project was developed in the context of an image processing course in the Electrical Engineering Program at Penn State Harrisburg.

II. Color Image Basics

In this section, a review of color image processing is presented. Color images have three times as many pixels as a grayscale image[5,6]. Therefore, color imaging processing requires significant additional processing. There are several different standards for color image formats. The most well-known formats include the RGB, YCbCr, and HSL. The RGB format stores its images based upon three color bit-planes or matrices of data, the Red, Green, and Blue bit-planes. Each bit-plane contains the pixel information of its respective color intensity; RGB color planes are shown in Figure 1.

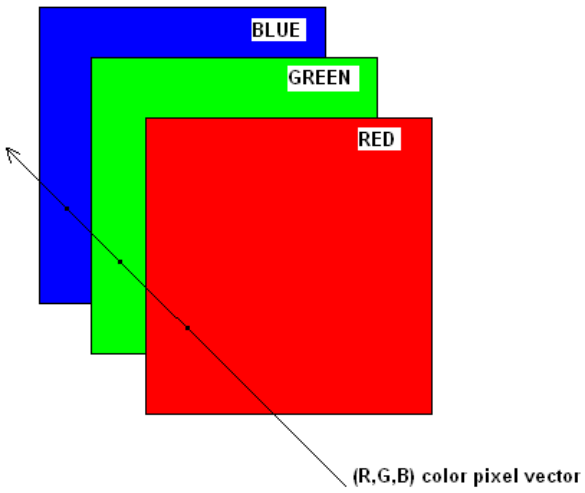

Figure 1. RGB color bit-planes [3]

When displayed together, the RGB planes form the final color image. The YCbCr format also stores three bit-planes of information. Luminance is represented by Y and two-color difference components are represented by Cb and Cr. Cb represents the difference

between blue and a reference value; Cr represents the difference between red and a reference value. Conversion between these two color formats is directly available in MATLAB[7] as ycbcr2rgb and rgb2ycbcr routines. The equations that relate these two formats are displayed below:

From RGB to YCbCr Standard

$$Y = \left(\frac{77}{256}\right)R + \left(\frac{150}{256}\right)G + \left(\frac{29}{256}\right)B$$

$$Cb = -\left(\frac{44}{256}\right)R - \left(\frac{87}{256}\right)G + \left(\frac{131}{256}\right)B + 128$$

$$Cr = \left(\frac{131}{256}\right)R - \left(\frac{110}{256}\right)G - \left(\frac{21}{256}\right)B + 128$$

From YCbCr to RGB

$$R = Y + 1.371(Cr - 128)$$

$$G = Y - 0.698(Cr - 128) - 0.336(Cb - 128)$$

$$B = Y + 1.732(Cr - 128)$$

Where *R*, *G*, and *B* are the red, green and blue pixel values of the RGB vector respectively and Y, Cb and Cr are luminance and the color differences described above.

III. Color Image Merging Algorithm

Image merging, or the processing of blending two images into a single image, can be done in many ways. In papers[2,3,4] several techniques were described for image merging, where entropy, wavelets or edge information were used. However, many of the concepts described in those papers include topics not typically available to an undergraduate student. In addition, there are some open source software that perform image merging, particularly the popular Picasa[8]. The approach taken in this paper uses a simple yet effective technique to prove this concept. The algorithm is being designed to satisfy a requirement for an application relating to switchboard imaging. The overall goal is to acquire smaller, more detailed images of a switchboard that is relatively large in scale. It is understood that an object that is of large scale cannot be imaged with any great detail. This is because a camera cannot develop the small details of an object if it is positioned at a distance further away to capture it at full-scale. To solve this problem, this paper will propose a method to capture images that represent smaller portions of the object and piece together those images in order to form a large scale image of the overall object.

The algorithm presented in this paper utilizes a simple difference technique. The procedure references two different images to be merged. The first image is a database image. The second one is an image to be added to the database image. The algorithm starts by extracting a single color plane from both color RGB images. This assists with the overall processing speed. The next step is to sample the pixel intensity values of a

specified window in the second image and to store in a matrix the pixel intensities within this window. The window dimensions can be adjusted at run-time by the user. Subsequently, the algorithm proceeds with the left uppermost pixel, assuming that the window dimensions do not exceed the image borders. In this position, a new matrix is defined which includes the pixels found in this particular portion of the database image. Before the window moves to the next pixel in the database image, a simple equation takes the absolute value of the differences between the database image matrix and the matrix acquired by sampling a piece of the image to be added. The sum of these differences is stored into another matrix for future processing. After this value is calculated, the window moves to the very next pixel in the image. This process continues until all pixels in the image are analyzed, excluding the pixels around the border due to window-width constraints.

After every pixel position has an effective difference associated with it, the minimum difference is located in the difference matrix. This minimum difference point is the point at which the two images have maximum similarities, or maximum correlation. Once this pixel position is known, the two images are merged using this specific point as a reference. Using this technique, it is possible to capture large objects by assembling different image shots and merging the images to compose a larger image file. This allows retention of the small details with respect to the overall image dimensions. In the next section, we will show some results.

IV. Results

Overall, the algorithm appears to work and only focuses on two images at this point. The results of the algorithm are illustrated below. Figures 2 and 3 illustrate the database image and the image to be added respectively. These images were found in Google Earth.

Figure 2. Database image

Figure 3. Image to be added

The sample image (50 × 50 pixels) taken from the red plane of image to be added is shown below

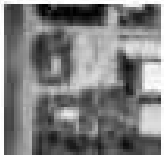

Figure 4. Sample image

The difference of the scan across the database image is illustrated in Figure 5. For simplicity, we have shown the difference in a one-dimensional graph rather than two dimensions, applying the reshape function in MATLAB. The plot clearly shows the minimum difference to be used for alignment purposes between the two images. This is the point of maximum correlation.

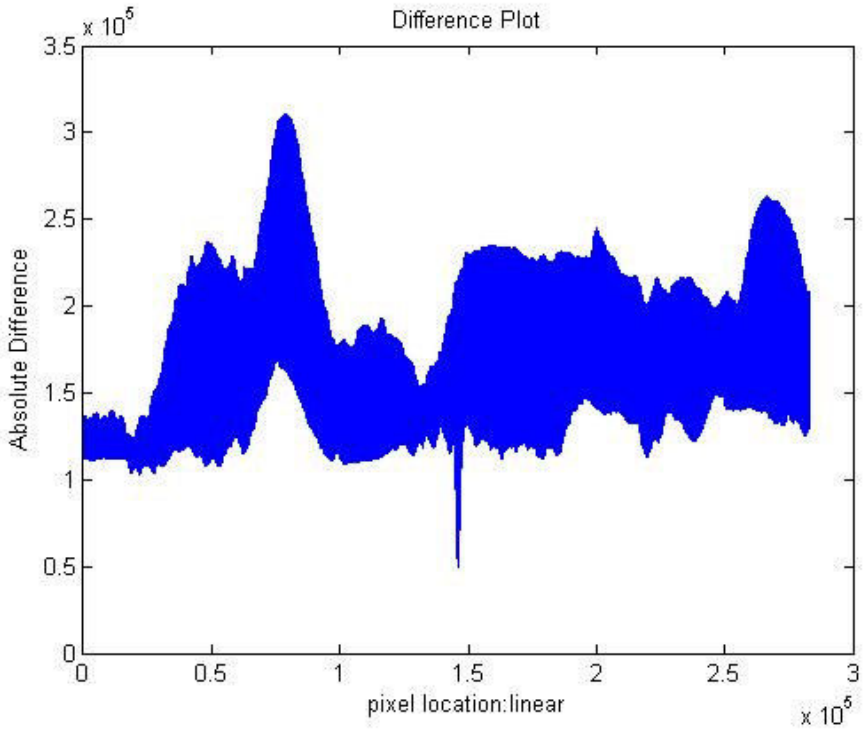

Figure 5. Difference plot

The next image illustrates the merged image. By comparing Figures 2, 3, and 6, it can be determined that the two images have been expanded to create an overall image, which reveals the whole field of view. This concept will be carried through in the future to apply to switchboards and other objects too large to capture adequately and maintain the detail required to extract information at a smaller scale.

Figure 6. Merged image

V. Conclusions

This paper focused on a simple merging algorithm for color images. Once images are pre-processed and merged into a single image, additional processing can be done on the image. For instance, an edge detection algorithm could be applied to the image to detect edges of the object being photographed. After the image has been run through an edge detection algorithm, an AutoCAD utility can be used to convert the edge-detected image into a drawing file. With the image in this format, it can be edited by a drafter. Having a system like this could potentially eliminate time spent by manually drafting objects into AutoCAD. To get a drawing of an existing object onto a product drawing, the object simply needs to be surveyed using photographs and then passed through the algorithms described. This algorithm is intended for a large switchboard. It is very difficult to capture an entire switchboard in an image and maintain readability of every device mounted in the switchboard itself. Using this technique, it will be possible to fully capture an entire switchboard and retain small details with respect to the overall image dimensions. While exhaustive comparison with other more complex algorithms[2,3,4] has not been performed, we suspect that, given the simplicity of the algorithm, it uses less CPU time than its counterparts.

Bibliography

[1] E. Wasatonic, S. Agili and A. Morales, "Range Determination Algorithm Performed on Mars Exploration Rover Stereo Images," *Proceedings of the 2005 American Society for Engineering Education Annual Conference & Exposition.* Portland, Oregon. June 2005.

[2] A. German, M.H. Jenkin, and Y. Lesperance, "Entropy-Base Image merging," *Proceedings of the Second Canadian Conference on Computer and Robot Vision,* Victoria, BC, Canada, May 2005.

[3] P. Scheunders and S. De Backer " Mutispectral Image Fusion and Merging Using Multiscale Fundamental Forms," *Proceedings of the IEEE International Conference on Image Processing*, Thessaloniki, Greece, 2001.

[4] N. Soussi and J.L. Barat, " Merging in Medical Multimodality Imaging," *Proceedings of the 18th Annual International Conference of IEEE Engineering and Biology Society*, Amsterdam, 1996.

[5] R**.** Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd edition, Prentice Hall, 2002.

[6] R. Mahoney, "Refinements in Adaptive Quantization for Images Using the New Universal Quality Index," Masters Paper, Penn State Harrisburg, March 2006.

[7] MATLAB Image Processing Toolbox User's Guide, version 5.

[8] http://picasa.google.com/

ERIC BOYER

Mr. Boyer was an undergraduate student in the Electrical Engineering Technology Program. He is now a part-time graduate student in the Master of Engineering Program, Electrical Engineering at Penn State University at Harrisburg.

ALDO MORALES

Dr. Morales received his Electronic Engineering degree with distinction from the University of Tarapaca, Arica, Chile, and M.S. and Ph.D. degrees in Electrical and Computer Engineering from the State University of New York at Buffalo. His research interests are digital signal and image processing, and computer vision. He is an Associate Professor of Electrical Engineering at Penn State University at Harrisburg.

Appendix : Basic Color Image merging Algorithm in MATLAB

```
% This first portion of code will read in the two images to be added,
% extract the red information and capture a small window of the image
% to
% be added so that a comparison can be made with database image to
% match
% relative position in database image.
% developed by Eric Boyer and Aldo Morales, Electrcica Engineering
% Program Penn State Harrisburg
close all;
clear all;
D = imread('tmi 1.jpg'); % Read database image into MATLAB
A = imread('tmi 2.jpg'); % Read image to add into MATLAB
%figure, imshow(D); % Display database image
%figure, imshow(A); % Display image to add

Dr = D(:,:,1); % Extract red from database image
Ar = A(:,:,1); % Extract red info from image to add


[DW,DH]=size(Dr); %captures size of database image to use for
% processing
[AW,AH]=size(Ar); %captures size of image to be added to use for
% processing
WS = 50;  %window size (WR x WR)


WA=Ar(200:249,70:119);
[WAx,WAy]=size(WA)  % Capture window from image to add
figure;imshow(WA);
figure;imshow(Ar);
%**********************************************************************

% This next portion of code will use the window information captured in
% the last section to compare against the database image looking for
% a match.

SI=1; % specifies the step index for scanning resolution.  A SI of one
      % will scan every pixel of the database image
mycount=0; % Counter checking
DXX=0;
for DX=(WS/2)+1:SI:DW-(WS/2)-1 % set up X scan of database image
    DXX=DXX+1;
     DYY=0;
    for DY=(WS/2)+1:SI:DH-(WS/2)-1; % set up Y scan of database image

        WD=Dr(DX-(WS/2):DX+(WS/2)-1,DY-(WS/2):DY+(WS/2)-1); % builds a
                  % window over red database image to compare
                  % with window captured from image to be added to look
        DYY=DYY+1;  % for similarities
                   % builds a window over red database image to compare
                   % with window captured from image to be added
                   % to lookfor similarities
```

```
        mydata(DXX,DYY)=sum(sum(abs(double(WD)-double(WA)))); % This
            % function take the difference of the two windows WD and
            % WA. These values will be used to find minimum difference
            % in the database image (indicating best match).

        if(mydata(DXX,DYY)==0)
            mycount=mycount+1;
        end
    end
end
data_size=size(mydata);
data_rsh=reshape(mydata,1,data_size(1)*data_size(2)); % Convert in one
                                % dimensional graph for display purposes
figure, plot(data_rsh);title('Difference Plot');xlabel('pixel
location:linear');ylabel('Absolute Difference');
disp('The minimum');
mymin=min(min(mydata));
[myrow,mycol]=find(mydata==mymin);
% disp('zero count')
% disp(mycount);
myNewD=D(:,1:mycol-50,:); % the offset is due to the starting point of
the mydate matrix
figure;imshow(myNewD);  % show the portion of the image to be added
myNewD1=[myNewD A];     % concatenate the two images using the row as
                        % marker
figure;imshow(myNewD1); % show the final image
```