Increasing Efficiency of Template-Matching Algorithm using Binary-Image instead of Color-Image

[1]Nikunj Bhensadadiya, [2]Sheetal Solanki
[1]M.E. Scholar, [2]Assistant Professor
Department of Information Technology,
Shantilal Shah Government Engineering College, Bhavnagar, Gujarat (India)

***Abstract* - Template matching algorithms uses color-images most of time, but it is not necessity. They sometimes find shapes in particular images. So use of binary-images has been proposed instead of color-images. As binary-image have all information related to shape same as in color-image, only color information is dropped. Binary-image is smaller in size than color-image, which make storage and transmission more efficient as well as makes processing faster [4]. Some application of template matching where time is first concern than use of color-images may lead us to delay. Here use of binary-image as an input to template matching algorithms has been proposed, where identifying particular shape is more important. Omitting color information from image makes image more efficient for processing, storage and transmission.**

***Keywords* - Template Matching, Color-Image v/s Binary-Image, Edge Detection**

I. Introduction

Template matching is a technique in digital image processing for finding small parts of an image that matches a template image. It can be used in manufacturing as a part of quality control [1], a way to navigate machine robot [2]. A basic method of template matching uses a convolution mask (template), tailored to a specific feature of the search image, which we want to detect. This technique can be easily performed on grey or edge images. The convolution output will be highest at places where the image structure matches the mask structure, where large image values get multiplied by large mask values. Improvements can be made to the matching method by using more than one template (Eigen spaces), these templates can have different scales and rotations. It is also possible to improve the accuracy of the matching method by hybridizing the feature-based and template-based approaches [3]. Naturally, It is required that search and template images should have features that are apparent enough to support feature matching.

A binary image is a digital image that has only two possible values for each pixel. Typically the two colors used for a binary image are black and white though any two colors can be used. The color used for the object(s) in the image is the foreground color while the rest of the image is the background color[4].

II. Background

A. Template Matching

Code Snippet 1 shows generic code for template matching algorithm, in this simple implementation, it is assumed that the above described method is applied on grey images: This is why Grey is used as pixel intensity. The final position in this implementation gives the top left location for where the template image best matches the search image. One way to perform template matching on color images is to decompose the pixels into their color components and measure the quality of match between the color template and search image using the sum of the SAD computed for each color separately.

```
// loop through the search image
for ( int x = 0; x <= S_rows - T_rows; x++ ) {
   for ( int y = 0; y <= S_cols - T_cols; y++ ) {
      SAD = 0.0;
           // loop through the template image
           for ( int j = 0; j < T_cols; j++ )
         for ( int i = 0; i < T_rows; i++ ) {
             pixel p_SearchIMG = S[x+i][y+j];
            pixel p_TemplateIMG = T[i][j];
             SAD     +=     abs(     p_SearchIMG.Grey       -
p_TemplateIMG.Grey );
         }
       // save the best found position
      if ( minSAD > SAD ) {
         minSAD = SAD;
```

```
                // give me min SAD
                position.bestRow = x;
                position.bestCol = y;
                position.bestSAD = SAD;
            }
        }
    }
```

Code Snippet 1: Generic Template Matching

B. Color Image

A color image is usually stored in memory as a raster map, a two-dimensional array of small integer triplets; or (rarely) as three separate raster maps, one for each channel.[5] Eight bits per sample (24 bits per pixel) seem adequate for most uses, but faint banding artifacts may still be visible in some smoothly varying images, especially those subject to processing.[6] Particularly demanding applications may use 10 bits per sample or more. On the other hand, some widely used image file formats and graphics cards may use only 8 bits per pixel, i.e., only 256 different colors, or 2–3 bits per channel. Converting continuous-tone images like photographs to such formats requires dithering and yields rather grainy and fuzzy results. Graphics cards that support 16 bits per pixel provide 65536 distinct colors, or 5–6 bits per channel.[5] This resolution seems satisfactory for non-professional uses, even without dithering.

C. Binary Image

Binary images are also called bi-level or two-level. This means that each pixel is stored as a single bit—i.e., a 0 or 1.[5] The names black-and-white, B&W, monochrome or monochromatic are often used for this concept, but may also designate any images that have only one sample per pixel, such as grayscale images. In Photoshop parlance, a binary image is the same as an image in "Bitmap" mode[7]. Binary images often arise in digital image processing as masks or as the result of certain operations such as segmentation, thresholding, and dithering. Some input/output devices, such as laser printers, fax machines, and bilevel computer displays, can only handle bilevel images. Binary images can be interpreted as subsets of the two-dimensional integer lattice Z^2; the field of morphological image processing was largely inspired by this view.

D. Conversion to Binary Image

Figure 1 show, Conversion of image to binary image. Apply any of Edge-Detector algorithms on input image, and mark Edges and Background with “1” and “0” respectively. Value of threshold of edge detector algorithms will be based on input image environment.[8] For dark and dull images threshold value will be low, on other hand for bright images it will be high.

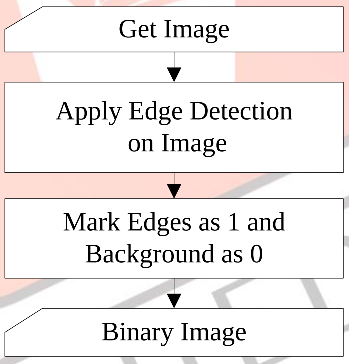
Get Image
Apply Edge Detection on Image
Mark Edges as 1 and Background as 0
Binary Image

Figure 1: Conversion to Binary-Image

III. PROPOSED SCHEME

Where finding particular shape is more important, using binary-image is proposed. Proposed system has two step to conduct on any image before applying template matching algorithm, 1) Convert input image in to binary-image, if it is not already. 2) Convert all templates in to binary. After these two steps only remaining is to apply template matching algorithm on both.

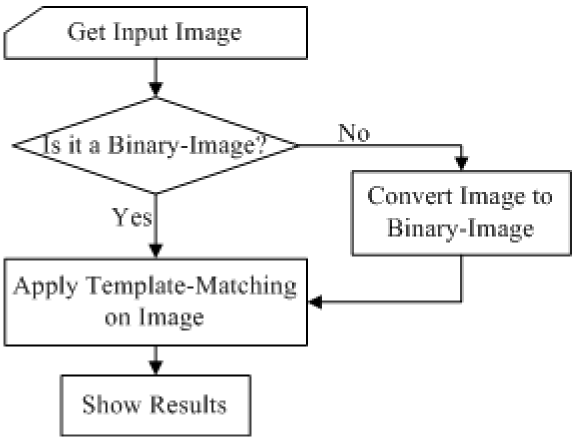
Get Input Image
Is it a Binary-Image?
No
Yes
Convert Image to Binary-Image
Apply Template-Matching on Image
Show Results

Figure 2: Proposed Scheme

As shown in figure 2, First check for input image whether it is in binary format or not, if not convert it into binary image, and later on apply template matching algorithm. Here input image can be anything ranging from a still image to live camera feed

IV. Experimental Results

The proposed algorithm was tested on video sequences captured at several crossroads in various lighting and weather conditions. They include day and night time scenes of the congested as well as free traffic flow. Figure 3 some comparative analysis of video sequences between current system and proposed system during different weather conditions. The size of experimental images was 2048 X 1536 pixels with 32-bit color images. A non-optimized implementation of the algorithm achieved processing speed of 30 fps on a Windows PC with Intel CORE i5 2.5 GHz. Thus, it fulfils the real-time performance requirement for the video streams operating at 25 fps frame rate. The accuracy of template matching was increased by 0.72% for bright and clear images.

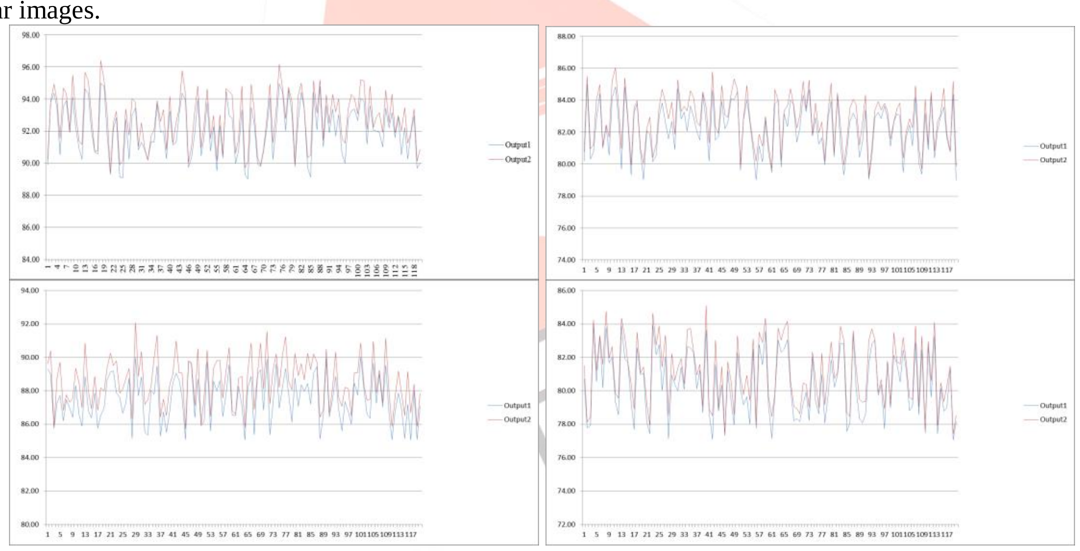

Figure 3: Comparative Result of frame vs. matching percentage
(a) Full light (b) Low light (c) Video with Average 5% Noise (d) Video with Average 12% Noise

V. Conclusion

The experimental results demonstrate that the proposed algorithm is feasible and promising that the efficiency of template matching algorithms can be increased if considered for only shape detection. Increment in efficiency is shown in Table 1.

Table 1: Comparative Results

Weather Condition	Existing System	Proposed System	Increment in Efficiency
Full Light	92.07%	92.79%	0.72%
Low Light	82.19%	82.82%	0.63%
With Avg. 5% Noise	87.49%	88.17%	0.68%
With Avg. 12% Noise	80.40%	80.97%	0.57%

VI. Limitation and Future Scope

Proposed system is only intended to find particular shape from given image, if it comes to relate with color proposed system will not work. For image with too many edges edge detector algorithm might takes more time to create binary image, this reflects on total time taken, and system might not give result in desired time.

In future, first step will be implementing proposed system on real world scenario by using graphics boards. Later if needed, noisy images can be dealt with, as results shows that, with increasing noise, efficiency decreases.

REFERENCES

[1]. Aksoy, M. S., O. Torkul, and I. H. Cedimoglu. "An industrial visual inspection system that uses inductive learning." Journal of Intelligent Manufacturing 15.4 (August 2004): 569(6). Expanded Academic ASAP. Thomson Gale.

[2]. Kyriacou, Theocharis, Guido Bugmann, and Stanislao Lauria. "Vision-based urban navigation procedures for verbally instructed robots." Robotics and Autonomous Systems 51.1 (April 30, 2005): 69-80. Expanded Academic ASAP. Thomson Gale.

[3]. C. T. Yuen, M. Rizon, W. S. San, and T. C. Seong. “Facial Features for Template Matching Based Face Recognition.” American J. of Engineering and Applied Sciences 3 (1): 899-903, 2010.

[4]. "Conversion of a Color Image to a Binary Image". CoderSource.net. 2005-04-18.

[5]. R. C. Gonzalez, Paul, and Wintz, “Digital Image Processing”, Addison-Wesley Publishing Company Limited.

[6]. Alvarez, Luis, Pierre-Louis Lions, and Jean-Michel Morel. "Image selective smoothing and edge detection by nonlinear diffusion. II." SIAM Journal on numerical analysis 29.3 (1992): 845-866.

[7]. http://www.graphics.com/article-old/photoshop-fundamentals-working-different-color-modes

[8]. O'Grady, Paul D., and Scott T. Rickard. "Automatic ASCII art conversion of binary images using non-negative constraints." (2008): 186-191.

IJEDR