Multi-Scale Template Detection

Ashish Ledalla (B18EE008)
Nagamalla Rohith (B18CSE065)
Rohit Kumar Sahu (B18BB029)
Jevala Sripad (B18CSE019)
Barad Dheeraj Bharadwaj (B18CSE009)

Abstract

In this project, we aim to find regions in a given image which are having similar features of a reference image. It is easy for our complex human brain to identify similar regions, but for a computer to do so, first we need to reduce the complexity of the initial images using image processing techniques. In this project we take help of canny edge detection method to simplify the images to only edges and also with only two colors (Binary image). This technique is widely used in the field of computer vision for object detection, medical imaging, feature tracking.

1. Introduction

Template matching (TM) is a straightforward and simple object detection method in image analysis. This is a basic process used in today's revolutionary technology of machine learning and deep learning which uses object detection. The project uses the smoothing, gradient/derivative calculation, double thresholding and hysteresis.

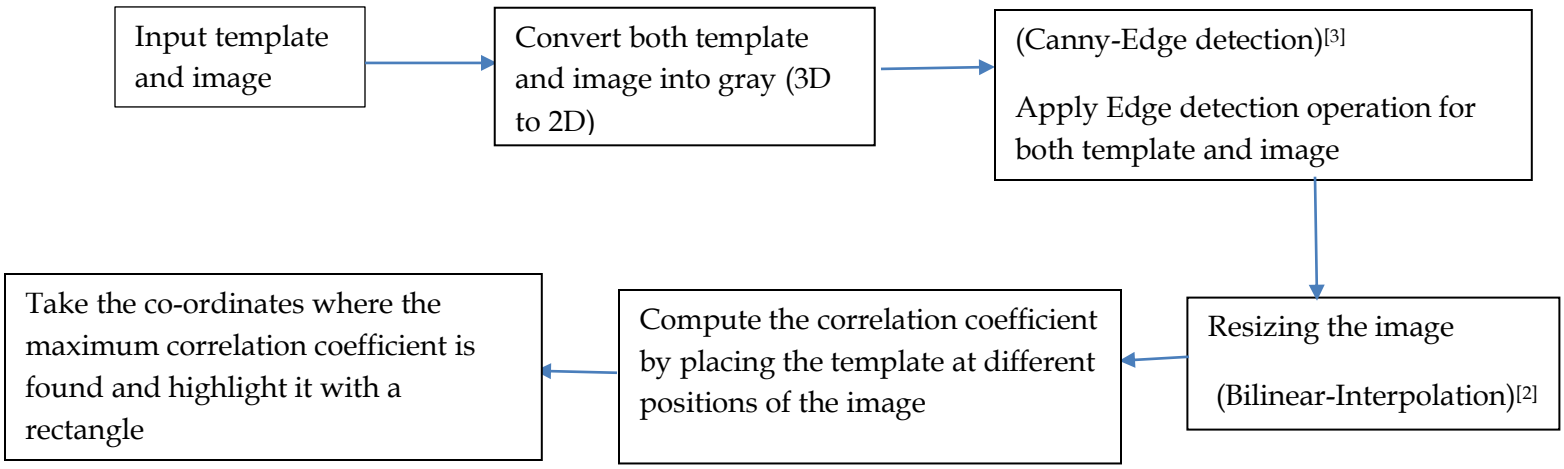
1.1 Tools of signals and systems: convolution, interpolation, normalized Correlation coefficient.

1.2 Background: Basically we apply edge detection for both template and image to increase the accuracy of template matching using correlation coefficient.

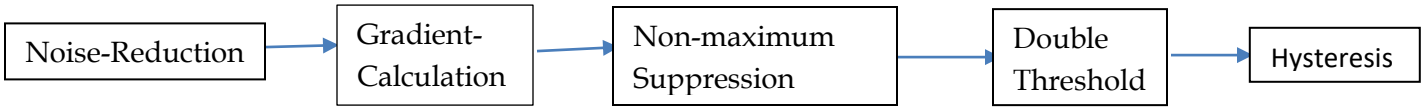
1.3 Problem statement: Given a template the objective is to find the best match for it in a given image.

2. Methodology:

2.1 Block Diagram:



2.1.1 Canny-edge detection:



2.2 Algorithm Explanation:

2.2.1 Edge Detection: [3]

This is a key part which increases the accuracy of the process which includes the following five subparts

2.2.1.1 Noise Reduction using Gaussian blur: Edge detection involves a calculation of derivative, so the noise can disturb the output to a great extent. To avoid that, we use the noise reduction process in which we use the Gaussian filter kernel.

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}\right); 1 \leq i, j \leq (2k+1)$$

This is the algorithm to calculate elements of the Gaussian kernel

2.2.1.2 Sobel filtering: This process gives us the gradient magnitude and the edge direction (in radians). Convolute the following filters over the image to get the gradient magnitudes and direction.

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}. \qquad |G| = \sqrt{I_x^2 + I_y^2}, \qquad \theta(x,y) = arctan\left(\frac{I_y}{I_x}\right)$$

2.2.1.3 Non-maximum Suppression: For each pixel, we compare it with other pixels that fall in the corresponding edge direction. If there is a pixel whose intensity is higher than the one being processed, then the pixel's intensity is made zero. If there are no pixels whose intensity is greater than the pixel being processed, no changes are made.

2.2.1.4 Double Thresholding: The double threshold step aims at identifying 3 kinds of pixels: strong, weak, and non-relevant. The high threshold is used to identify the strong pixels (intensity higher than the high threshold). The low threshold is used to identify the non-relevant pixels (intensity lower than the low threshold). All pixels having intensity between both thresholds are flagged as weak

2.2.1.5 Edge Tracking by Hysteresis: Based on the threshold results, the hysteresis consists of transforming weak pixels into strong ones, if and only if at least one of the pixels around the one being processed is a strong one, as described below.

2.2.2 Bipolar Interpolation: is a resampling method that uses the distance weighted average of the four nearest pixel values to estimate a new pixel value. The four-cell centers from the input raster are closest to the cell center for the output processing cell will be weighted and based on distance and then averaged.[2]

2.2.3 Correlation coefficient: The correlation coefficient is a statistical measure that calculates the strength of the similarity between two signals (images).

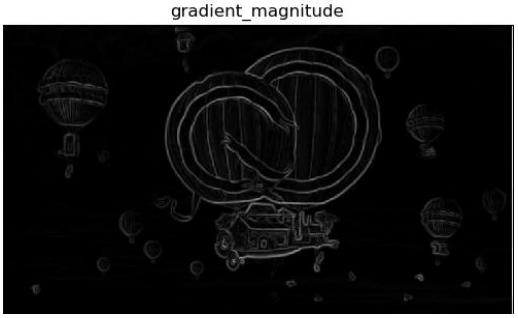
$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

3. Results and Observations

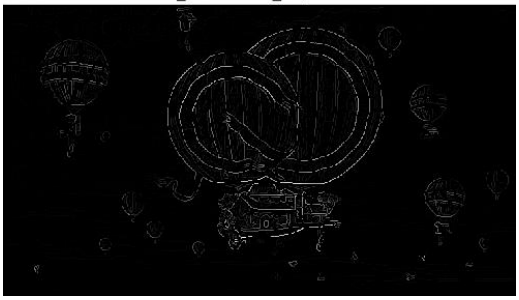
3.1 Intermediate results for Canny-Edge Detection:

noise_reduction

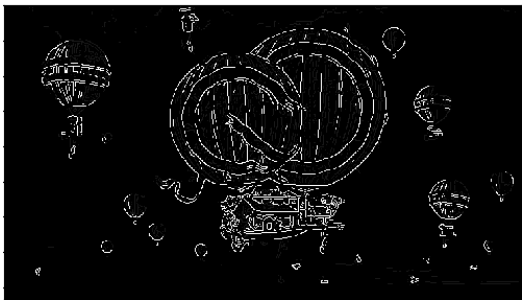
gradient_magnitude



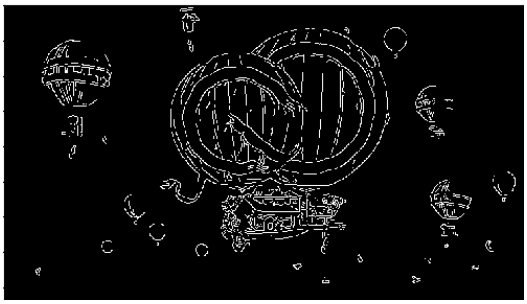
non_maximum_supression



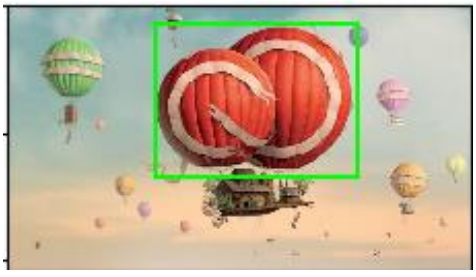
double threshold



hysteresis



template



(Image source : Google Images)

3.2 Comparison (of Sobel) with other methods:

We may also use Prewitt operator or Laplacian operator in the edge detection operation

Prewitt operator Laplacian operator

Prewitt operator is highly sensitive to noise. Laplacian operator is convoluted only once over the image to get the gradient image. So we won't get the gradient magnitudes to proceed to non-maximum suppression. It is also highly sensitive to noise. So we use the Sobel filter for our project.

4. Conclusions and Limitations

This technique is robust for images containing the template even with a lot of variations like color, shadows, and gradients.

4.1 Limitations: This method fails if the sub-image in the image is having variations like rotations and perspective change. We can only obtain the sub-image which is having the maximum correlation coefficient with the template. In an image containing more than one template, only one of them is detected.

4.2 Future Scope: We will try to extend the project by removing the limitations. We try to detect the patch in image although the patch is rotated, more than one patches exist or patch is seen through a perspective view.

References

[1] Adrian Rosebrock. *"Multi-scale Template Matching using Python and OpenCV"*. pyimagesearch.com https://www.pyimagesearch.com/2015/01/26/multi-scale-template-matching-using-python-opencv/(accessed Oct. 26, 2019).

[2] No Author. *"Bilinear Interpolation"*. rosettacode.com https://rosettacode.org/wiki/Bilinear_interpolation (accessed Oct. 26, 2019).

[3] Sofiane Sahir. "*Canny Edge Detection Step by Step in Python – Computer Vision*". towardsdatascience.com https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123 (accessed Nov. 17, 2019).