Improved Template Matching Using 3-Shear Template Rotation

Ethan Garnier
egarnie1@unb.ca
3658981

***Abstract*—Template matching is a digital image processing technique that attempts to locate sub-images, or templates, within a larger base image. This technique is commonly used in image processing pipelines where certain features or objects must be detected in captured images, such as automated assembly lines or computer vision. A common issue with traditional template matching techniques is that the rotation of the template image must perfectly match the rotation of any occurrences of the template in the base image for successful matching. This requirement is less than ideal, as it is common that the rotation of the template is not something that can be controlled. This paper explores a method of template matching that incrementally rotates the template image to improve matching accuracy when template orientation cannot be guaranteed.**

***Index Terms*—template-matching, digital image processing, Fourier transform, shear rotation**

I. Introduction

THE advent of digital image processing has enabled considerable innovation in the realm of engineering. From enhancing the detail in images taken of your friends, to blurring the background in family photos, digital image processing is a part of our every day lives. This vast field contains many different techniques for transforming and extracting information from these digital images. One such technique, template matching, is a method of digital image processing that allows for a sub-image, or a template, to be identified and located within a larger image.

Template matching represents an incredibly powerful tool that enables object recognition and detection in images. There exists two approaches to template matching: feature-based and template-based matching. Feature-based template matching relies on feature extraction from the image, such as shape or color, and will often use some machine learning model to identify these extracted features [1]. This method works well, yet it can over-complicate potentially simple template matching problems. The alternative is template-based template matching, where the similarity between a template and a base image is computed through the cross-correlation of the two. This approach speeds up the template matching process, making it very useful when searching large images or large numbers of images [1], however requires that the object of interest is known. Template-based template matching will remain the focus of this paper.

Despite its usefulness, there still exists drawbacks to traditional template-based template matching techniques that commonly make it only applicable in very specific situations. One such shortcoming is the fact that the rotation of the template of interest in the image must be known for successful matching. Typically this is not something that can be controlled, therefore modified methods of template matching must be employed that are rotation independent. This paper will explore one such method, 3-Shear Rotation Template Matching, developed by Marvin Chandra Wijaya in their paper titled *Template Matching Using Improved Rotations Fourier Transform Method* [1]. The method proposed by *Wijaya* will be implemented and extended to allow for N-matching of rotation independent templates within a given base image. The matching accuracy of this extended implementation will be demonstrated visually using a custom Matlab program to draw bounding boxes around all successful matches in the tested base images to demonstrate the effectiveness of the 3-shear rotation template matching technique.

II. 3-Shear Rotation Template Matching

When an object of interest, and its orientation, is known within a given base image, then simple cross-correlation between the base image and a template of the object of interest can quickly provide effective template matching results. However, as previously discussed, when the orientation of the object of interest cannot be guaranteed, as is often the case, different methods must be employed. There arises a trade-off between method simplicity and accuracy: the more simple the approach, the less accurate the results will be. The 3-shear rotation template matching method, as outlined by *Wijaya* [1], presents a method that balances simplicity and accuracy by making a slight modification to the traditional template matching technique which results in greatly increased matching accuracy when template rotation is not guaranteed. This technique, which uses frequency domain cross-correlation for matching, rotates the template image incrementally through several shear rotations that reduce noise and loss of information.

A. Frequency Domain Cross-Correlation

Traditional template matching of images may be performed in the spatial frequency domain, as opposed to the spatial domain. The Fourier transform is used to determine the frequency responses of both the base and template images, the complex conjugate of one response is computed, and then the results are multiplied as described in Equation 1.

$$C(u,v) = F(u,v)T^*(u,v) \tag{1}$$

The result, $C(u,v)$, of multiplying the base image $F(u,v)$ by the complex conjugate of the template image $T^*(u,v)$ is the frequency response of the matching. Transforming this response back into the spatial domain through the inverse Fourier transform yields a grayscale image with large pixel amplitudes where matching between the base image and template was strong. The benefit of cross-correlating images in the frequency domain is that the Fourier transform, more specifically the Fast Fourier Transform (FFT) discrete Fourier transform algorithm, is incredibly optimized for large datasets in digital systems. Images are represented discretely as large matrices of numbers; as such, doing the spatial domain cross-correlation could be incredibly inefficient. The FFT algorithm is used to instead bring the images into the frequency domain and reduce the computation to a simple element-wise multiplication between two matrices.

B. Shear Rotations

General counter-clockwise rotation by an angle θ is accomplished by multiplying each (x,y) point by the rotation matrix M found in Equation 2 [2].

$$M = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \tag{2}$$

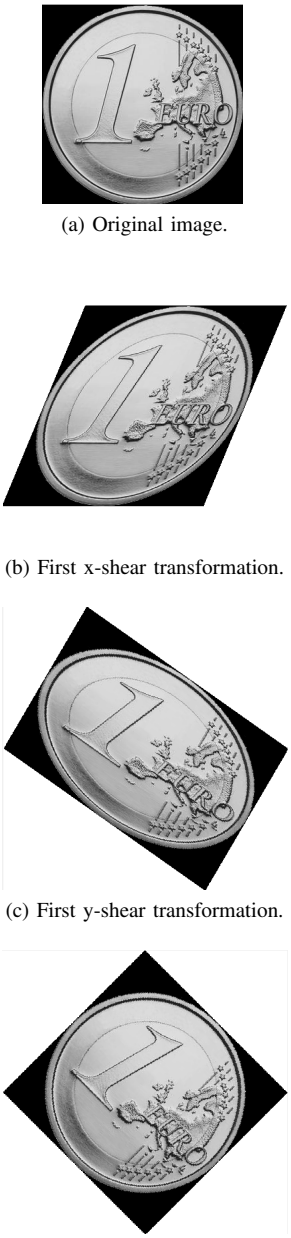
This can be extended to the rotation of discrete two-dimensional images, where each (x,y) point is a pixel at column index x_1 and row index y_1 offset by the center of rotation (x_0, y_0), usually the center of the image. The rotated coordinates (x_2, y_2) can then be described by Equation 3 and Equation 4 [1].

$$x_2 = (x_1 - x_0)cos\theta - (y_1 - y_0)sin\theta \tag{3}$$

$$y_2 = (x_1 - x_0)sin\theta + (y_1 - y_0)cos\theta \tag{4}$$

The issue with rotating discrete 2D images using the mappings described in Equation 3 - 4 is that the rotated coordinates of each pixel, (x_2, y_2), will rarely fall on integer values. This will result in a noisy rotated image as these non-integer coordinates cannot be represented in a discrete two-dimensional image, thus interpolation must be used to fill the gaps.

In the context of template matching, a noisy template is unacceptable as it will significantly reduce matching accuracy and effectively render the process of rotating the template pointless. Overcoming this problem can be accomplished by using shear rotations, where the rotation matrix seen in Equation 2 is decomposed into a sequence of one-dimensional transformations along the image's x and y axis [3]. This decomposition, as visualized in Fig. 1 rotating a Euro by $45°$, consists of three steps: shearing the image along the x-axis, shearing the image along the y-axis, and finally shearing the image along the x-axis again to yield the rotated image. Since each step only modifies a pixel's location relative to a single axis, the potential intersection of a rotated pixel on non-integer pixel boundaries is reduced, resulting in a less noisy image.



(a) Original image.

(b) First x-shear transformation.

(c) First y-shear transformation.

(d) Final x-shear transformation.

Fig. 1: Each step of rotating an image by $45°$ using the 3-shear rotation method.

The sequence of shear transformations on a discrete image visualized in Fig. 1 is represented by the product of shear matrices seen in Equation 5.

$$\begin{bmatrix} 1 & -tan(\theta/2) \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ sin\theta & 1 \end{bmatrix} * \begin{bmatrix} 1 & -tan(\theta/2) \\ 0 & 1 \end{bmatrix} \tag{5}$$

The first and final matrices of the product in Equation 5 represent the x-shear transformations, whereas the second matrix represents the y-shear transformation. Each matrix has a determinant of 1 [1].

Paeth [2] reduced the matrix product seen in Equation 5 to three simple arithmetic instructions to rotate a given discrete (x, y) point by angle θ to points $(\hat{x}, \hat{y})$. These three instructions, as seen in Equation 6, are what was used in our 3-shear rotation implementation.

$$\begin{aligned} x_1 &= x - tan(\theta/2) * y \\ \hat{y} &= y + sin\theta * x_1 \\ \hat{x} &= x_1 - tan(\theta/2) * \hat{y} \end{aligned} \tag{6}$$

C. Template Matching Algorithm

Now that frequency domain cross-correlation and 3-shear image rotation have been defined, their use in the complete template matching algorithm can be explained. Fig. 2 is a flowchart adapted from Fig.3 in *Wijaya*'s work [1] that outlines the general control flow of the template matching algorithm.

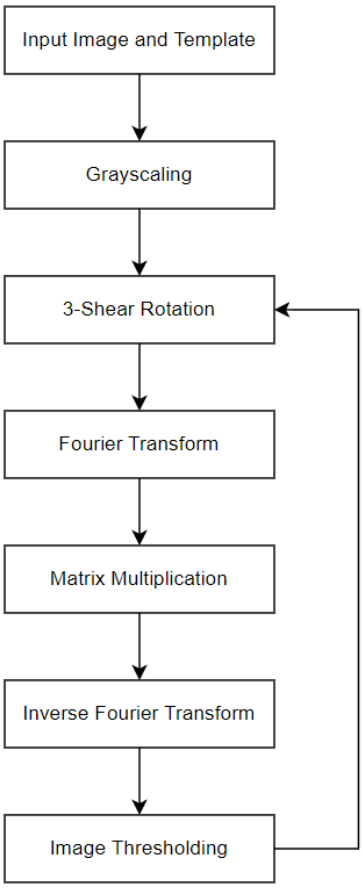


Fig. 2: Control flow of the 3-shear rotation template matching algorithm.

The control flow defined in Fig. 2 begins by reading in the base image and the template image to be searched for. What follows is a very common pre-processing step, where these images are converted to grayscale, i.e., the intensities of each pixel are normalized to an 8-bit integer value (0-255) and only a single channel of the image is preserved. Following is the main loop of the algorithm, which executes the following steps:

1) Rotate the template image by some preset $\Delta\theta$ angle.
2) Transform the base image and rotated template image into the Frequency domain using the Fourier transform.
3) Perform the frequency domain cross-correlation by multiplying the frequency response of the base image by the complex conjugate of the frequency response of the rotated template image.
4) Transform the cross-correlation result back into the spatial domain using the inverse Fourier transform.
5) Store the brightest pixel intensity of the cross-correlation result.
    a) If this intensity is larger than the previous iteration's brightest intensity then restart from 1. (maximum matching may not be found).
    b) If this intensity is not larger than the previous iteration's brightest intensity then break from the loop (maximum matching has been found).

Once the maximum matching has been located in the image with the above loop, the image is segmented through image thresholding and converted into a binary image where non-zero pixel values represent the pixels matched with the template.

III. Extension of the Algorithm

The 3-shear rotation template matching algorithm outlined by *Wijaya* [1] and described in Section II-C was implemented and extended in Matlab. The extension to the base algorithm added the ability to match any number of templates in a given base image, as opposed to only one, as what is done by the original algorithm. This was done as a learning exercise and to make the algorithm more practical for potential real-world adoption in computer vision systems.

Accomplishing this was done by slightly modifying the main loop of the algorithm described in Section II-C. The $\Delta\theta$ by which the template image is incrementally rotated by has been fixed as 1°, and the base-cases by which the loop continues or exits from, 5 in the list above, has been completely re-written. On each loop iteration, any pixels from the cross-correlation output with intensities above a set threshold have their column and row indices saved to identify this location as a successful match. After 360 loop iterations, or after the template image has been rotated through the circle in 1° increments, these match indices stored over each loop are returned to the main program as all successful template matches. After some brief post-processing to remove duplicate location matches, what is left is all matches of the provided template found within the base image. Since the template is being rotated through 360°, template matching becomes completely independent of the template's rotation in the base image, given the templates aren't rotated in non-integer increments. If that is the case, however, the threshold value by which matches are identified can be experimented with to increase matching accuracy.

IV. Template Matching Results

An image of circles, triangles, and squares was created as the base image for testing our extended 3-shear rotation

template matching algorithm. The shapes in this image were randomly rotated to emphasize and demonstrate the rotational independence of the algorithm. This base image can be seen in Fig. 3.

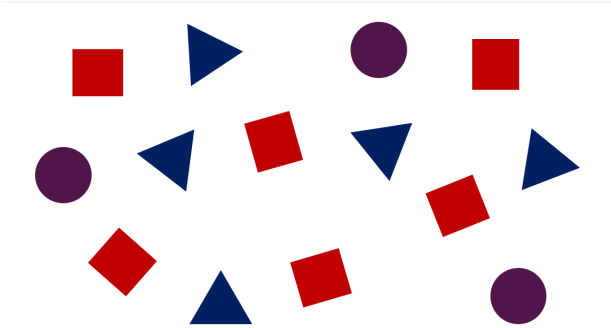


Fig. 3: Base image used to test template matching algorithm.

The different shapes present in Fig. 3 were then used as the templates to search for. The main Matlab program running the template matching algorithm used the builtin *drawrectangle* function to draw bounding boxes around all identified matches. Fig. 4 shows the matching results of matching against a square template, which succeeded with 100% accuracy.

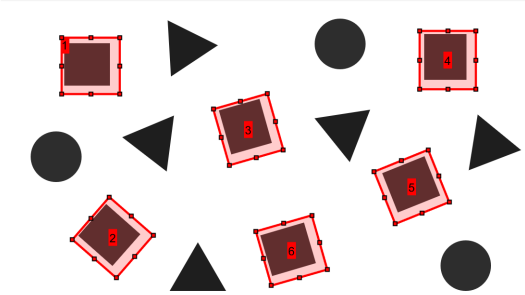


Fig. 4: Matching results of template matching Fig. 3 with a template image of a square.

Fig. 5 shows the matching results of matching against a triangle template, which succeeded with 100% accuracy.

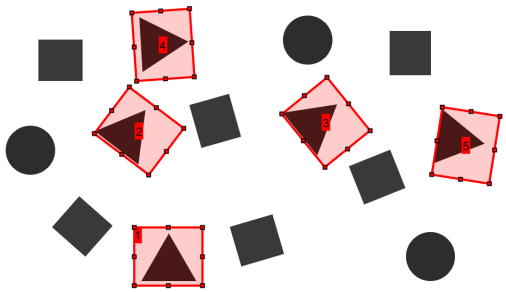


Fig. 5: Matching results of template matching Fig. 3 with a template image of a triangle.

Finally, Fig. 6 shows the matching results of matching against a circle template, which succeeded with 100% accuracy.

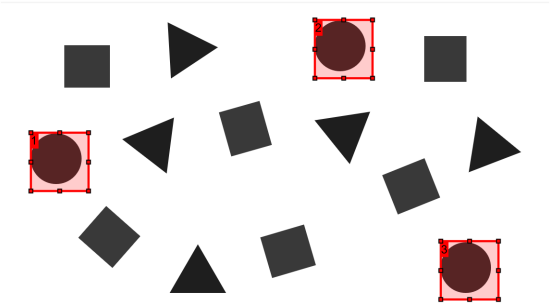


Fig. 6: Matching results of template matching Fig. 3 with a template image of a circle.

It must be noted, however, that a higher threshold for matching was required for the circle example. For Fig. 4 and Fig. 5, the threshold value was 0.0275 and 0.029, respectively. Yet at these values, the circle template was misidentifying non-circles in the base image. Increasing the threshold value to 0.04 resolved this issue. This demonstrates a further improvement to the algorithm that could be added in the future, where an adaptive threshold is automatically determined before matching is conducted.

The examples thus far have demonstrated the template matching algorithm's usefulness in identifying shapes, however it is important that it can also match with more complicated templates. Fig. 7 shows an image of various European coins that will be used as a more complex base image. These coins each have different rotations and some overlap with eachother, which will prove to be an excellent challenge for the algorithm.

Fig. 7: Base image of European coins.

Fig. 8 shows the matching results of matching Fig. 7 against a template of a 1 euro coin.

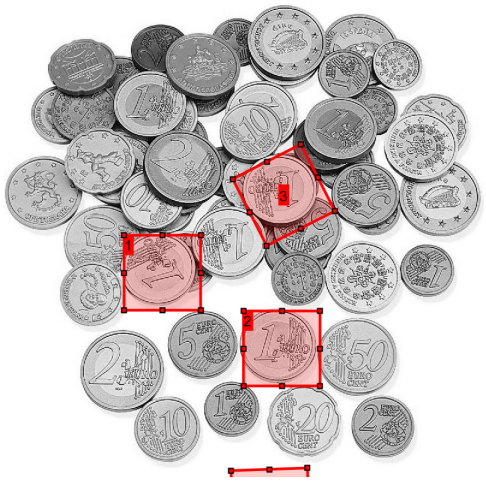


Fig. 8: Matching results of template matching Fig. 7 with a template image of a 1 euro coin.

As can be seen in Fig. 8, the extended 3-shear rotation template matching algorithm is capable of identifying 3 of the 6 visible 1 euro coins, with one misidentification along the image border. Given that many of these coins are obstructed by other objects, this is a very successful result and demonstrates the algorithms applicability to more complicated template matching problems.

V. Future Work

A major drawback of simple cross-correlation template matching, as demonstrated by the results in Fig. 8, is that template scale and orientation is crucial to matching success. The 3-shear rotation template matching algorithm implemented and discussed in this paper solves the problem of template orientation, yet template scale still plagues the success of this algorithm. Future work may involve a scale independent approach to template matching which, in combination with the algorithm implemented in this work, could significantly improve matching accuracy when template scale and orientation is not guaranteed.

VI. Conclusion

This paper explored the 3-shear rotation template matching algorithm proposed by *Wijaya* [1] to improve simple template matching when template rotation was not guaranteed. Their algorithm enabled rotationally independent template matching by rotating the template images in angle increments using consecutive shear transformations, allowing for improved matching accuracy and reduced burden on template choice. This work provides an extension to this algorithm to allow any number of templates to be matched in a given base image, further enhancing its applicability. The results presented in this paper demonstrate the usefulness of the 3-shear rotation template matching algorithm and its practicality as an image processing technique.

References

[1] M. C. Wijaya, "Template Matching Using Improved Rotations Fourier Transform Method," *International Journal of Electronics and Telecommunications; 2022; vol. 68; No 4; 881-888*, 2022. [Online]. Available: https://journals.pan.pl/dlibra/publication/143898/edition/125484

[2] A. W. Paeth, "A FAST ALGORITHM FOR GENERAL RASTER ROTATION," in *Graphics Gems*. Elsevier, 1990, pp. 179–195. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B9780080507538500462

[3] M. Unser, P. Thevenaz, and L. Yaroslavsky, "Convolution-based interpolation for fast, high-quality rotation of images," *IEEE Transactions on Image Processing*, vol. 4, no. 10, pp. 1371–1381, Oct. 1995. [Online]. Available: https://ieeexplore.ieee.org/document/465102