JAA 2023

Image Processing on Images of Ancient Artifacts with the Help of Methods Based on Artificial Intelligence

Mahyar Radak
Department of Atomic and Molecular Physics, Mazandaran University

Anita Akhgar
Department of Archeology, Faculty of Art and Architecture, University of Mazandaran, Babolsar, Iran

Article Information
Received 06/05/2023
Revision Accepted 17/05/2023
Available Online 23/06/2023

Abstract: Artificial intelligence (AI) has the potential to revolutionize the field of archaeology by enabling researchers to analyze large amounts of data quickly and accurately. In this article, we have tried to implement some methods and algorithms in image processing on the image of ancient artifacts. We implemented the algorithms on two historical models as examples, one of which is the image of a coin decorated with the image of Farkhan the Great and the other is a coin with the image of Khursheed Daboui to obtain the details of these works from the images on the computer. We used Edge Detection, Hough Transform, contour, and Filter Images Using Predefined Filters algorithms in MATLAB software, each of these algorithms is used for specific purposes in image processing. By using digital image analysis techniques, researchers can gain a deeper understanding of the objects and sites they are studying and can make new and important discoveries about the history and culture of ancient civilizations.

Keywords: *Image Processing, Artificial Intelligence, Algorithms, Historical Image Information, Mat lab*.

* Corresponding Author
Email Address: m.radak03@umail.umz.ac.ir *(Mahyar Radak)*

Introduction

Archaeology is essentially a "visual" discipline because visual perception informs us of the essential properties of objects and allows us to discover how objects were produced and used in the past (Barceló, Juan A, 2010: 93-156). Soon, archaeologists can use the results of these new fields, computer science, and artificial intelligence to improve their research. The digital world, in addition to visual information management techniques and knowledge discovery, in databases, will be useful for understanding information sources (Puyol-Gruart, Josep, 1999: 19-28). The researchers also point to the potential for archaeologists to explore such artificial intelligence (AI) approaches in various ways, such as identifying archaeological features and classifying them (Argyrou, Argyro, and Athos Agapiou, 2022: 6000). Image processing plays an important role in archaeology by helping researchers to analyze, enhance, and interpret images of artifacts, archaeological sites, and other archaeological materials. Image processing techniques can be used to reveal details that are not visible to the naked eye, reconstruct damaged or incomplete artifacts, and create 3D models of archaeological sites.

For example, image processing can be used to enhance photographs of ancient texts to make them more legible, as well as to analyze and classify ancient pottery fragments based on their shape and texture. It can also be used to create 3D models of archaeological sites, enabling researchers to visualize and explore the site remotely (K. Saibaba, S. Suresh Kumar, and S. S. S. Pavan Kumar, 2018).

AI can be used to create predictive models of past human behavior, such as settlement patterns, trade networks, or migration routes. These models can be based on a variety of data sources, including archaeological data, satellite imagery, and climate data. One example of this is the use of machine learning algorithms to predict the locations of ancient Maya settlements in Central America (Mara, Hubert, et al. 2015). AI can be used to analyze and interpret large amounts of text data, such as historical texts, inscriptions, or excavation reports. This can help archaeologists to identify key themes and topics and to uncover new insights into past societies and cultures. One example of this is using natural language processing (NLP) techniques to analyze ancient Greek inscriptions and identify patterns in the use of certain words and phrases (Kitzinger, Ernst, 1954: 83-150). AI can be used to create virtual reconstructions of archaeological sites and artifacts, based on data from excavation reports, 3D scans, and other sources. These reconstructions can be used to visualize and explore past environments and societies and to test hypotheses about the use and function of archaeological objects. One example of this is the use of machine learning algorithms to generate 3D models of ancient Roman pottery from 2D images [Gyselen, Rika, 2011]. The Sasanian Empire minted coins in various regions, including Tabaristan (also known as Mazandaran), which is located in present-day northern Iran. The coins minted in Tabaristan typically feature the bust of the Sasanian king on the obverse, and a Zoroastrian fire altar or a Sassanian symbol on the reverse. Here is an example of a Sassanian Tabaristan coin:

Obverse: Bust of the Sassanian king facing right, wearing a crown with a star and crescent, and with two ribbons hanging from the back of the crown. The legend around the bust is in Pahlavi's script and reads "mzdysn bgy Arsacy" (meaning "the Mazda-worship-

ing divine Arsaces").

Reverse: A Sassanian symbol, possibly a star or a sun, surrounded by a circle of pellets. The legend around the symbol is in the Pahlavi script and reads "nyzyk" (meaning "of Tabaristan") (Gyselen, Rika, 2011). Image processing has several benefits in archaeology, such as enhanced visualization, non-destructive analysis, improved documentation, efficient data analysis, and remote analysis. Enhanced visualization helps researchers better understand the objects and sites they are studying, while non-destructive analysis is important for fragile or rare artifacts. Improved documentation can be used for research purposes, preservation, and conservation. Efficient data analysis helps identify patterns and relationships between artifacts and sites. The remote analysis is useful in difficult-to-reach locations (S. K. Jayaraman and R. K. Lohani, 2015). Archaeology is an ongoing field of research, with discoveries and insights being made regularly. Recent examples include the discovery of a major Mayan ceremonial site in Mexico's Yucatan peninsula, analysis of ancient DNA, reconstruction of ancient cities using advanced imaging techniques, and investigation of ancient trade routes. These developments are just a few examples of the many exciting new developments in the field of archaeology, which can help us learn more about the history and culture of ancient civilizations (Gyselen, Rika: 2011; K. Saibaba, S. Suresh Kumar, and S. S. S. Pavan Kumar, 2018; S. K. Jayaraman and R. K. Lohani, 2015; Capriles, José M., et al. 2021).

Research Method

Edge Detection algorithm on the coin

Edge detection is used to identify edges in an image, using edge function and derivative estimates. The Canny method is the most powerful edge detection method because it uses two different thresholds and only includes weak edges when connected to strong edges. We run this algorithm on the sample coins and show how the edges can be identified using Canny and Sobel Edge detectors.

Here is the MATLAB code for implementing the Canny edge detection algorithm on an image of a coin:

```
% Canny edge detection code
% Read the image of the coin
coin = read('coin.jpg');
% Convert the image to grayscale
coin_gray = rgb2gray(coin);
% Apply Gaussian smoothing to reduce noise
coin_smooth = imgaussfilt(coin_gray, 1.5);
% Apply the Canny edge detection algorithm
coin_edges = edge(coin_smooth, 'Canny');
% Display the original image and the edges
figure;
subplot(1,2,1); imshow(coin); title('Original Image');
subplot(1,2,2); imshow(coin_edges); title('Edges');
```

In this code, we first load an example image using the `imread` function and convert

Fig. 1. Implementation of Canny edge detection algorithm on coins, a) image of Farkhan the Great, b) image of Khursheed Daboui

it to grayscale using the `rgb2gray` function. We then apply the Canny edge detection operator to the grayscale image using the `edge` function with the `'Canny'` option. The `edge` function uses the Canny method to detect edges in the image. Finally, we display the original and processed images side by side using the `subplot` and `imshow` functions.

MATLAB code for implementing the Sobel edge detector on the coin image:

```
% Sobel edge detector on the coin image
% Read in the image
img = imread('image.jpg');
% Convert the image to grayscale
img_gray = rgb2gray(img);
% Apply Sobel edge detection
sobel_horiz = [-1 -2 -1; 0 0 0; 1 2 1];
sobel_vert = [-1 0 1; -2 0 2; -1 0 1];
img_edges_horiz = imfilter(double(img_gray), sobel_horiz);
img_edges_vert = imfilter(double(img_gray), sobel_vert);
img_edges = sqrt((img_edges_horiz.^2) + (img_edges_vert.^2));
% Display the original image and the edges
figure;
subplot(1,2,1); imshow(img); title('Original Image');
subplot(1,2,2); imshow(uint8(img_edges)); title('Edges');
```

Fig. 2: Implementation of Sobel edge detector algorithm on coins, a) image of Farkhan the Great, b) image of Khursheed Daboui

In this code, we first load an example image using the `imread` function and convert it to grayscale using the `rgb2gray` function.

We then apply the Sobel edge detection operator to the grayscale image using the `edge` function with the `'Sobel'` option. The `edge` function returns a binary image where edges are indicated by white pixels and non-edges are indicated by black pixels.

Finally, we display the original and processed images side by side using the `subplot` and `imshow` functions.

Note that the specific parameters used in this example may need to be adjusted for different images and applications.

Hough Transform algorithm on the coin

Image processing supports functions that enable users to use Hough Transform to detect lines in an image. The Hough function uses the standard Hough (SHT) conversion and produces a parameter space matrix whose rows and columns correspond to this RHO and the TTA values. After calculating the Hough conversion, the performance of Hough peaks can be used to find peak values in the parameter space, and the performance of Houghlines can be used to find the end points of the Hough conversion lines.

rho = x X cos(theta) + y X sin(theta) (1)

```
%% Hough Transform
I  = imread('coin.jpg');
I=rgb2gray(I);
rotI = imrotate(I,33,'crop');
figure(1)
```

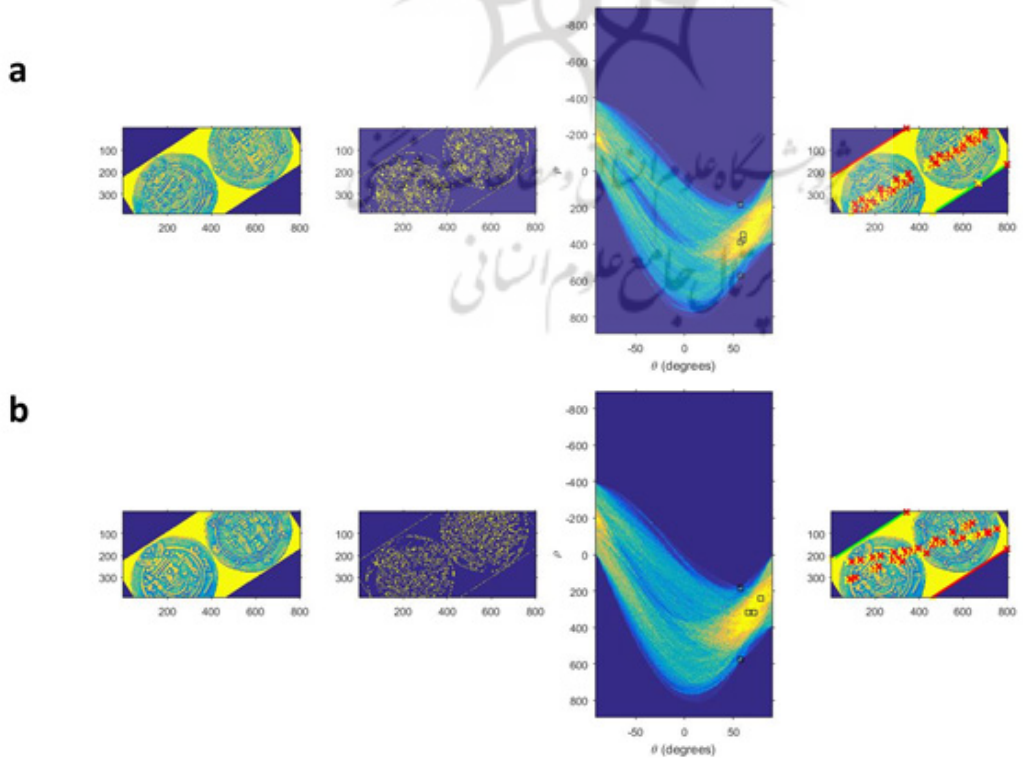

Fig. 3: Hough Transform algorithm on the coin, a) image of Farkhan the Great, b) image of Khursheed Daboui

```
subplot(1,4,1)
fig1 = imshow(rotI);
BW = edge(rotI,'canny');
figure(1)
subplot(1,4,2), imshow(BW);
[H,theta,rho] = hough(BW);
figure(1)
subplot(1,4,3), imshow(imadjust(mat2gray(H)),[], 'XData', theta, 'YData', rho,...
      'InitialMagnification', 'fit');
xlabel('\theta (degrees)'), ylabel('\rho');
axis on, axis normal, hold on;
colormap(hot)

P = hough peaks(H,5,'threshold',ceil(0.3*max(H(:))));
x = theta(P(:,2));
y = rho(P(:,1));
plot(x,y,'s', 'color','black');

 lines = hough lines(BW,theta,rho,P,'FillGap',5,'MinLength',7);
figure(1)
subplot(1,4,4), imshow(rotI), hold on
max_len = 0;
for k = 1:length(lines)
  xy = [lines(k).point1; lines(k).point2];
  plot(xy(:,1),xy(:,2), 'LineWidth',2, 'Color', 'green');

  % Plot beginnings and ends of lines
  plot(xy(1,1),xy(1,2), 'x', 'LineWidth',2, 'Color', 'yellow');
  plot(xy(2,1),xy(2,2), 'x', 'LineWidth',2, 'Color', 'red');

  % Determine the endpoints of the longest line segment
  Len = norm(lines(k).point1 - lines(k).point2);
  if ( len > max_len)
    max_len = len;
    xy_long = xy;
  end
end

% highlight the longest line segment
plot(xy_long(:,1),xy_long(:,2), 'LineWidth',2, 'Color', 'red');
```

Imcontour algorithm on the coin

Create a contour plot of image data

Here, `imread` is used to load the image "cameraman.tif" into the variable `I`. The `imshow` function is then used to display the image. The `imcontour` function is called three

times to display contour lines at different levels and with different line styles and colors. The resulting contour plots are displayed in separate figures.

```
%% imcontour
I = imread('coin1.jpg');
I=rgb2gray(I);
figure(1)
subplot(1,2,1)
imcontour(I,3)
I = imread('coin2.jpg');
I=rgb2gray(I);
figure(1)
subplot(1,2,2)
imcontour(I,3)
```

Filter Images

Here we apply some filter algorithms to increase the clarity and improve the quality of the images of the coins so that the details of the coin can be seen better.

A.4) sharpening filter

In MATLAB, you can apply a sharpening filter to an image using the "imsharpen" function. This function enhances an image's edges and other high-frequency components, making it appear sharper.

the "Amount" parameter controls the strength of the sharpening effect (lower values produce a more subtle effect), while the "Radius" parameter controls the size of the filter kernel (larger values enhance larger features in the image).

Here's an example of how to use the "imsharpen" function:

```
% Apply a sharpening filter with custom parameters
% Read an image
img = imread('coin.jpg');

% Apply a sharpening filter with default parameters
```


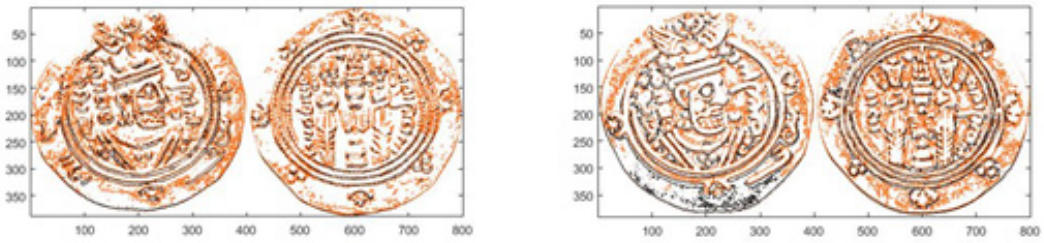

Fig. 4: imcontour algorithm on the coin, a) image of Farkhan the Great, b) image of Khursheed Daboui

Fig. 5: Sharpening filter on coin image and implementing "Amount" and "Radius" parameters on images, a) image of Farkhan the Great, b) image of Khursheed Daboui

```
sharpened_img = imsharpen(img);

% Display the original and sharpened images side by side
figure;
subplot(1,2,1);
imshow(img);
title('Original Image');
subplot(1,2,2);
imshow(sharpened_img);
title('Sharpened Image');
sharpened_img = imsharpen(img, 'Amount', 0.5, 'Radius', 2);
% Display the original and sharpened images side by side
figure;
subplot(1,2,1);
imshow(img);
title('Original Image');
subplot(1,2,2);
imshow(sharpened_img);
title('Sharpened Image (Amount=0.5, Radius=2)');
```

B.4) Filter Images Using Predefined Filters

```
%% Filter Images Using Predefined Filters

I = imread('coin.jpg');
I=rgb2gray(I);

% Create filter, using fspecial.
h = fspecial('unsharp')
```

```
% Apply filter to image using imfilter.
 I2 = imfilter(I,h);

% Display original image and filtered image for comparison.
imshow(I)
title('Original Image')

figure
imshow(I2)
title('Filtered Image')
```

C.4) unsharp masking filter on the coin image

```
% Load the image
img = imread('coin.jpg');

% Convert the image to double precision
img_double = im2double(img);

% Apply unsharp masking to the image
unsharp_img = img_double - imfilter(img_double, fspecial('gaussian', [5 5], 1));
```

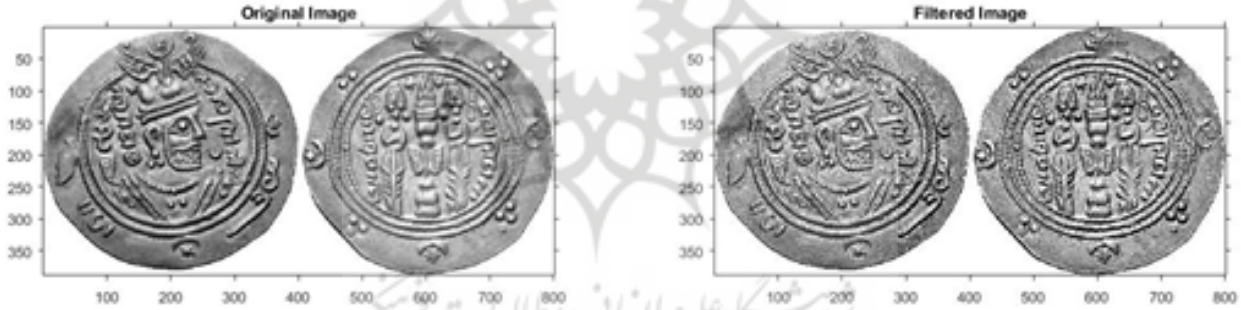

Fig. 6: Filter coin Images Using Predefined Filters


```
unsharp_img = img_double + 1.5 * unsharp_img;

% Increase the contrast of the image
contrast_img = imadjust(unsharp_img, [0.2 0.8], [0 1]);

% Display the side of the original and processed image by side
figure;
subplot(1,2,1);
imshow(img);
title('Original Image');
subplot(1,2,2);
imshow(contrast_img);
title('Processed Image');
```

This method applies several image processing techniques to an input image:

First, after loading the image in MATLAB, the “im2double” function is used to convert the image to double precision, which is necessary for some subsequent processing steps. The “fspecial” function is used to create a Gaussian filter of size 5x5 and standard deviation 1. This filter is applied to the image using the “imfilter” function to create a blurred version of the image. The blurred image is subtracted from the original image to create a high-pass filtered image. This is known as unsharp masking and is a technique used to enhance edges and fine details in an image. The unsharp masked image is then added to the original image, scaled by a factor of 1.5. This step increases the overall contrast of the image. Finally, the “imadjust” function is used to stretch the pixel values of the image, so

Fig. 7: Unsharp masking filter on the coin image, a) image of Farkhan the Great, b) image of Khursheed Daboui

that the darkest 20% of pixels are black, the brightest 80% are white, and the remaining pixels are scaled linearly in between. This step increases the contrast of the image.

The processed image is then displayed along with the original image in a figure window, with titles indicating which is which.

D.4) Contrast stretching filter

contrast stretching is a technique used to enhance the contrast of an image by linearly scaling its pixel values to span the full dynamic range (0 to 1 by default). Here, the “imadjust” function stretches the pixel values so that the darkest 20% of the pixels become black, the brightest 80% become white, and the remaining pixels are linearly scaled in between. The resulting stretched image is then displayed along with the original image in a figure window using the “subplot” and “imshow” functions.

```
% Read an image
img = imread('C:\Users\Mahyar\Desktop\archaeology\1.jpg');
```

```
% Perform contrast stretching
img_stretched = imadjust(img, [0.2 0.8], [0 1]);

% Display the side of the original and processed image by side
figure;
subplot(1,2,1);
imshow(img);
title('Original Image');
subplot(1,2,2);
imshow(img_stretched);
title('Stretched Image');
```

Fig. 8: Contrast stretching filter on the coin image, a) image of Farkhan the Great, b) image of Khursheed Daboui

5. *2D Fourier expansion of the algorithm*

Matlab code for performing 2D Fourier expansion on the coin image:

```
% Load an example image
img = imread('coin.jpg');

% Convert the image to a double-precision grayscale image
img = im2double(rgb2gray(img));

% Compute the 2D Fourier transform of the image
F = fft2(img);

% Shift the zero-frequency component to the center of the spectrum
F = fftshift(F);

% Compute the magnitude spectrum
magF = abs(F);
```

```
 % Display the magnitude spectrum
imshow(log(1+magF),[]);

% Compute the inverse 2D Fourier transform to reconstruct the image
img_recon = ifft2(F);

% Display the side of the original and reconstructed image by side
imshowpair(img, img_recon, 'montage');
```

In this code, we first load an example image and convert it to grayscale. We then compute the 2D Fourier transform of the image using the `fft2` function. We shift the zero-frequency component to the center of the spectrum using the `fftshift` function and compute the magnitude spectrum using the `abs` function. We display the magnitude spectrum using the `imshow` function. We then compute the inverse 2D Fourier transform of the Fourier spectrum using the `ifft2` function to reconstruct the original image. Finally, we display the original and reconstructed images side by side using the `imshowpair` function.

Conclusion

Artificial intelligence (AI) has the potential to revolutionize archaeology by enabling researchers to analyze large amounts of data quickly and accurately. This article has attempted to implement methods and algorithms in image processing on two historical models, using Edge Detection, Hough Transform, imcontour, and several Filter Images. By using digital image analysis techniques, researchers can gain a deeper understanding of the objects and sites they are studying and make new and important discoveries about the history and culture of ancient civilizations. Image processing plays an important role in archaeology by helping researchers to analyze, enhance, and interpret images of artifacts, archaeological sites, and other archaeological materials. AI can be used to create predictive models of past human behavior, analyze and interpret large amounts of text data, and create virtual reconstructions of archaeological sites and artifacts. AI can also be used to create virtual reconstructions of archaeological sites and artifacts, enabling researchers to visualize and explore the site remotely. Overall, image processing has become an increasingly important tool in archaeology, enabling researchers to analyze and interpret archaeological materials in new and innovative ways.

Bibliographical References

Barceló, Juan A.

2010 "Visual analysis in archaeology. An artificial intelligence approach", Morphometrics for Nonmorphometricians: 93-156.

Puyol-Gruart, Josep.

1999 "Computer science, artificial intelligence, and archaeology", BAR International Series 757: 19-28.

Argyrou, Argyro, and Athos Agapiou

2022 "A Review of Artificial Intelligence and Remote Sensing for Archaeological Research." Remote Sensing 14.23: 6000.

K. Saibaba, S. Suresh Kumar, and S. S. S. Pavan Kumar

2018 "Role of Image Processing Techniques in Archaeology." International Journal of Scientific Research in Computer Science, Engineering and Information Technology, vol. 3, no. 1, pp. 35-40.

Mara, Hubert, et al.

2015 "A bridge to digital humanities: geometric methods and machine learning for analyzing ancient script in 3D." Keep the Revolution Going: Proceedings of the 43rd Annual Conference on Computer Applications and Quantitative Methods in Archaeology. Open Access.

Kitzinger, Ernst

1954 "The cult of images in the age before iconoclasm." Dumbarton Oaks Papers 8: 83-150.

Gyselen, Rika

2011 "Mints and minting." In Sasanian Iran (3rd-7th centuries): Political history, religion, and society, edited by Rika Gyselen, 281-318. Leuven: Peeters.

S. K. Jayaraman and R. K. Lohani

2015 "Image processing techniques in archaeology: An overview," International Journal of Computer Applications, vol. 111, no. 11, pp. 12-16, Feb.

Capriles, José M., et al.

2021 "Pre-Columbian transregional captive rearing of Amazonian parrots in the Atacama Desert." Proceedings of the National Academy of Sciences 118.15 (2021): e2020020118.