Lab 3, Question 3

# a)

## Watershed with compactness

The Watershed method is a technique used in image segmentation that is based on the concept of image topography. It treats the grayscale values of an image as heights and simulates the flooding of a landscape with water, starting at local minima, until distinct catchment basins are formed. Each catchment basin corresponds to a distinct object or region in the image.

Compact Watershed is a modification of the original Watershed algorithm that addresses the over-segmentation issue by combining small regions together. The method takes an additional parameter called compactness, which controls the degree of compactness of the resulting regions. A higher compactness value results in more compact regions, while a lower value leads to more irregularly shaped regions.

## Canny edge filter

Canny edge detection is a popular edge detection algorithm developed by John F. Canny in 1986. It is a multi-stage algorithm that can detect a wide range of edges in an image, while also suppressing noise and false edges. The algorithm works by first applying a Gaussian filter to the image to smooth it, then calculating the gradient of the image to find the edges. It then applies non-maximum suppression to thin the edges to a single pixel width, and finally uses hysteresis thresholding to determine which edges are "real" edges and which are noise or false edges. The result is a binary image where the edge pixels are set to white and non-edge pixels are set to black.

## Geodesic Active Contour

Morphological geodesic active contour (MAGC) is a type of active contour segmentation method that utilizes geodesic morphological operators to evolve the contour towards the object boundary. It is based on the geodesic active contour framework and incorporates the concept of morphological reconstruction to improve segmentation accuracy in the presence of noise, occlusions, and weak edges.

The MAGC method involves computing a geodesic distance map of the input image, which is used to initialize the contour. The contour is then evolved using morphological geodesic active contours, which combines the benefits of morphological operations and geodesic distance transforms to achieve more robust and accurate segmentation results.

The MAGC method has been shown to be effective in various applications, such as medical image segmentation, object tracking, and image restoration. It is particularly useful in scenarios where traditional active contour methods may fail due to noise or weak edges in the image.

The morphological geodesic active contours method is particularly useful in segmenting objects with irregular shapes, weak or blurred edges, or in images with noise or intensity inhomogeneity.

# b)

The functions ***adapted\_rand\_error*** and ***variation\_of\_information*** are used for the evaluation.

The function ***adapted\_rand\_error*** has this output:

* **are** : The adapted Rand error.
* **prec** : The adapted Rand precision: this is the number of pairs of pixels that have the same label in the test label image and in the true image, divided by the number in the test image.
* **rec** : The adapted Rand recall: this is the number of pairs of pixels that have the same label in the test label image and in the true image, divided by the number in the true image.

The function ***variation\_of\_information*** has this output:

* **vi** : ndarray of float, shape (2,). The conditional entropies of image1 given image0 and image0 given image1. The conditional entropies can be used to calculate the mutual information between the two images, which is a measure of how much information one image provides about the other.

# c)

When evaluating a segmentation algorithm using the adapted Rand error and variation of information, the adapted Rand error returns three values: the adapted Rand error score, the true segmentation label indices, and the predicted segmentation label indices. The true segmentation label indices refer to the ground truth labels of the image, while the predicted segmentation label indices refer to the labels predicted by the segmentation algorithm being evaluated.

The adapted Rand error score is a measure of the similarity between the true and predicted segmentation label indices. It ranges from 0 (indicating no similarity) to 1 (indicating perfect similarity). A higher adapted Rand error score indicates better segmentation performance.

On the other hand, the variation of information returns two values: the entropy of the true segmentation and the predicted segmentation. These values represent the amount of uncertainty or disorder in the true and predicted segmentations. The lower the entropy value, the less uncertain or disorder there is in the segmentation. In the context of evaluation, the goal is to minimize the variation of information, as this indicates more accurate segmentation.

In conclusion, when evaluating a segmentation algorithm using adapted Rand error and variation of information, a lower adapted Rand error score and lower variation of information values indicate better segmentation performance.

# d)

I tuned the iteration and ran the code provided in the documentation.

* **10 iterations**: the segmentation result is really bad.

The metrics are:

Adapted Rand error: 0.9119716589209266

Adapted Rand precision: 0.984573514153116

Adapted Rand recall: 0.046073846222349696

False Splits: 0.6080946208054223

False Merges: 2.3048215115311024



* **50 iterations**: the result is not good but there are some enhancements in the metrics and picture.

The metrics are:

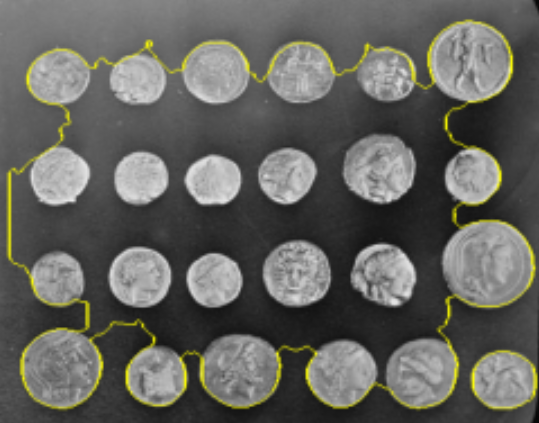
Adapted Rand error: 0.9056966493085734

Adapted Rand precision: 0.9443854771941276

Adapted Rand recall: 0.049629603042292865

False Splits: 0.7230027285899555

False Merges: 2.1329383565569766



* **100 iterations**: Some coins are segmented completely. But still, it needs more iterations.

The metrics are:

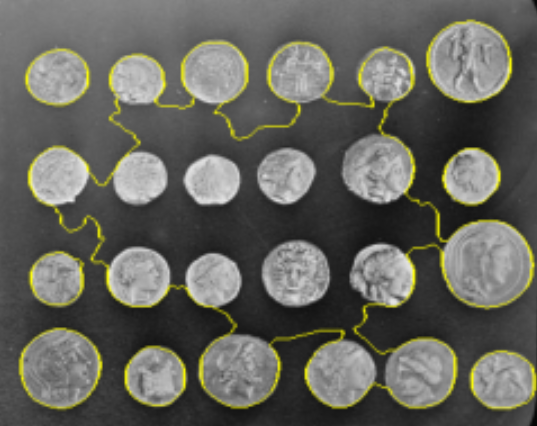
Adapted Rand error: 0.8345711457875802

Adapted Rand precision: 0.9180178567985586

Adapted Rand recall: 0.09090507520881418

False Splits: 0.6454300735439148

False Merges: 1.4650425836713417



* **150 iterations**: Almost all coins are segmented. Some coins in the center of the image aren’t segmented completely.

The metrics are:

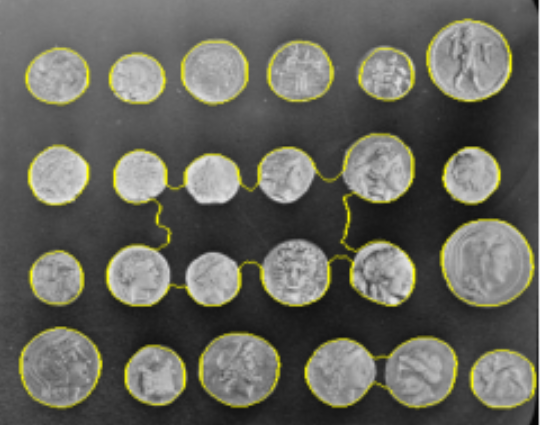
Adapted Rand error: 0.4838139758763862

Adapted Rand precision: 0.9016450734559539

Adapted Rand recall: 0.36159979398529624

False Splits: 0.38563210478910664

False Merges: 0.6573413920540792



* **200 iterations**: The segmentation is done completely for almost all coins in the image.

The metrics are:

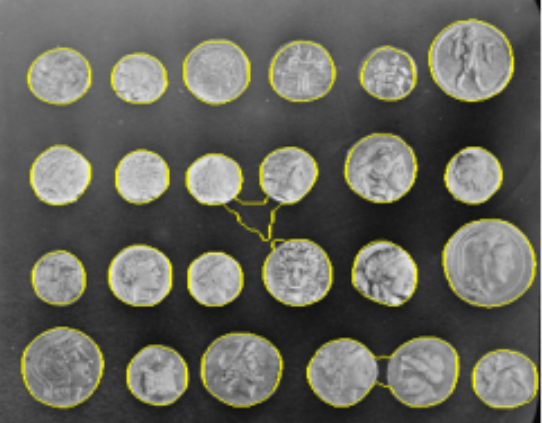
Adapted Rand error: 0.14142310863054275

Adapted Rand precision: 0.8979853366591377

Adapted Rand recall: 0.822481942691459

False Splits: 0.17767940682019046

False Merges: 0.2987818129745277



* **250 iterations**: Except for two coins, all coins are segmented completely. The result seems acceptable.

The metrics are:

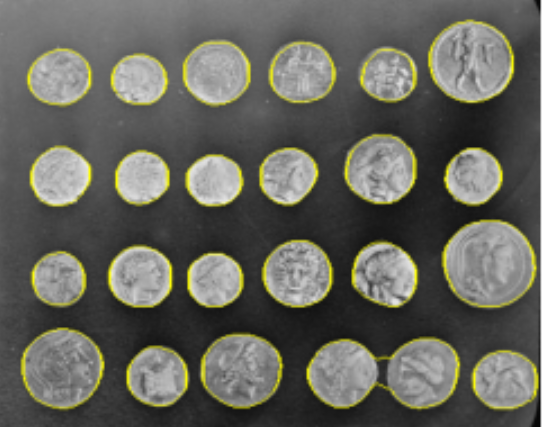
Adapted Rand error: 0.12817138318738164

Adapted Rand precision: 0.8973380473491516

Adapted Rand recall: 0.8477294540871629

False Splits: 0.1333507895392071

False Merges: 0.26256608273496207



The result is improved from 50 iterations to 250. The Geodesic Active Contour method is an iterative method, meaning that it refines the initial contour in a step-by-step process. During each iteration, the algorithm adjusts the contour to better align with the edges or other features in the image that it is trying to segment.

As the number of iterations increases, the algorithm has more chances to refine the contour and better fit it to the object boundaries. This can result in a more accurate and precise segmentation.

* **300 iterations**: The result hasn’t changed from the previous iterations in the image. Some small enhancements occurred in the metrics.

The metrics are:

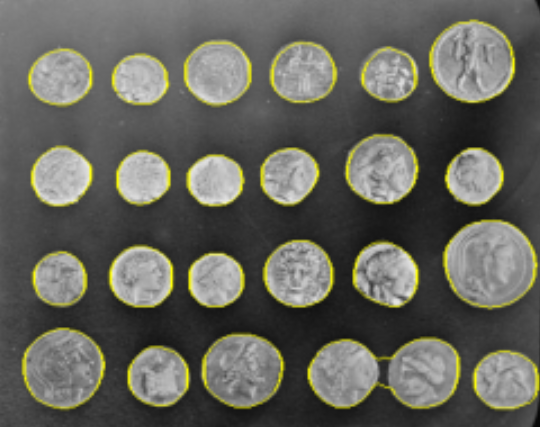
Adapted Rand error: 0.12781020674331223

Adapted Rand precision: 0.8975163154125038

Adapted Rand recall: 0.8482533945290367

False Splits: 0.1336520276781523

False Merges: 0.26243054688228085



* **400 iterations**: The result hasn’t changed from the previous iterations in the image and in the metrics.

The metrics are:

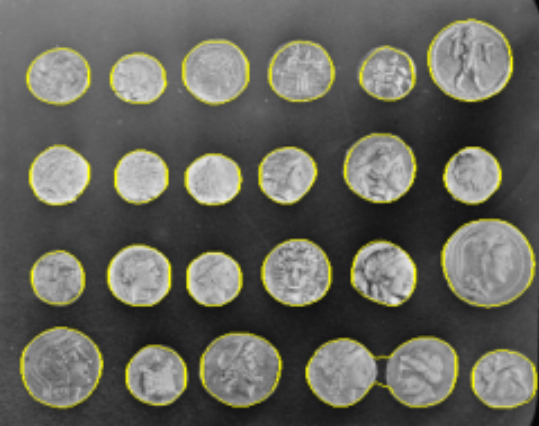
Adapted Rand error: 0.12781020674331223

Adapted Rand precision: 0.8975163154125038

Adapted Rand recall: 0.8482533945290367

False Splits: 0.1336520276781523

False Merges: 0.26243054688228085



* **500 iterations**: The result hasn’t changed from the previous iterations in the image and in the metrics.

The metrics are:

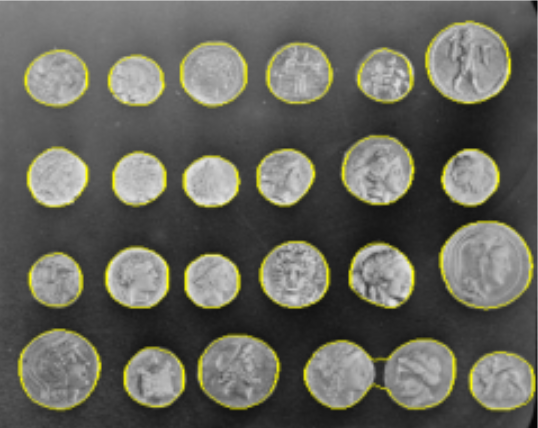
Adapted Rand error: 0.12781020674331223

Adapted Rand precision: 0.8975163154125038

Adapted Rand recall: 0.8482533945290367

False Splits: 0.1336520276781523

False Merges: 0.26243054688228085



The result hasn’t changed from 300 iterations to 500. Increasing the number of iterations of the GAC method leads to an optimization process that tries to minimize the energy functional to find the segmentation. However, after a certain number of iterations, the optimization process can converge to a local minimum, where further iterations do not lead to any significant improvement in the segmentation result or the evaluation metrics.

# e)

See python codes

# f)

* **ndi.label**: is a function from the *scipy.ndimage* module in Python that is used to label connected regions in an array. In the context of using the Canny edge detection method, after applying the Canny filter to an image, the resulting binary image (where each pixel is either 0 or 1, representing the absence or presence of an edge, respectively) can be passed through *ndi.label()* to identify the individual connected components (or regions) of the edge. This allows for further analysis and processing of the edges, such as computing their length, width, or curvature.
* **Remove**: Removes objects smaller than the specified size. Expects *ar* to be an array with labeled objects, and remove objects smaller than *min\_size*. If *ar* is bool, the image is first labeled. This leads to potentially different behavior for bool and 0-and-1 arrays.

In the example, 21 is the *min\_size*.

The labeled image is extracted from the output of the ndi.label() function using this indexing procedure. The output tuple given by ndi.label contains the labeled image as its first element (), and by indexing with [0], we only keep the labeled image while discarding any other data the function may have returned.