EchoLocator: an Open Source Python Package for the Standardisation of Echographic Images in Multicentre Analysis

Giulio Del Corso[1,*], Laura De Rosa[2,3], Maria Antonietta Pascali[1], Francesco Faita[2], Sara Colantonio[1]

[1] "Alessandro Faedo" Institute of Information Science and Technologies (ISTI), National Research Council of Italy (CNR) - Pisa, Italy
[2] Institute of Clinical Physiology, National Research Council of Italy (CNR)- Pisa
[3] Department of Information Engineering and Computer Science, University of Trento, Trento, Italy
∗ Corresponding author: giulio.delcorso@isti.cnr.it

Contents

1 Context

Ultrasound (US) is a widely used imaging technique that provides real-time visualisation of the body's internal organs (e.g. liver, spleen, kidney, brain, spine, thyroid, breast). The main characteristics that make US imaging widely used in various clinical settings are non-invasiveness, safety and lack of radiation. As a result, it is becoming increasingly popular for diagnostic and management purposes [1]. The use of US images in combination with deep learning (DL) systems is also rapidly improving [2]; however, it raises several issues, especially

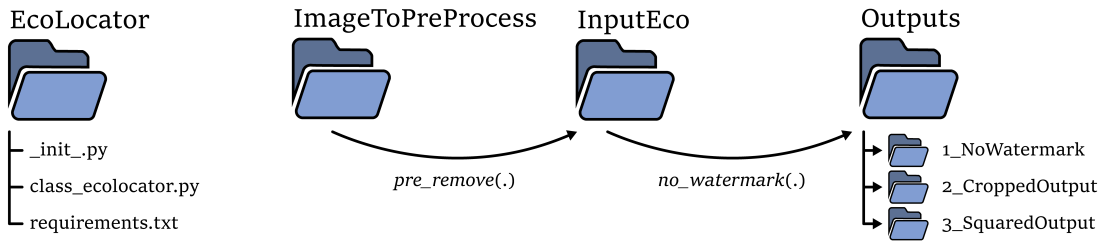

Figure 1: Structure of the EchoLocator package.

when the images are acquired with different machines and/or by using different acquisition settings. In fact, images acquired with different US devices from different manufacturers can vary significantly in terms of quality, resolution, penetration depth and imaging configurations. Differences in settings such as US frequency, gain, and imaging modes (B-mode, Doppler, etc.) can also affect the appearance of the images. In addition, scanner information and imaging parameters are often recorded as part of the image during US acquisition, e.g. textual information about image settings, measurement lines, arrows indicating the focus position in the field of view. Moreover, the surface area and width of the cone angle of the US scan can vary. Such heterogeneity has to be taken into account and managed when these data are fed into an AI-based system (e.g., to perform segmentation or to support diagnosis), especially when data are acquired in a multi-center study. For example, in a multi-center study with echographic images, an AI model, even if trained only to perform a clinical task, may easily learn, at the same time, to distinguish between the different clinical centers using watermarks or differences in the echographic cone, resulting in a AI model with very poor performances in the test phase.

Therefore, in this technical report, we provide a fully automated preprocessing package, developed entirely in Python 3.6, to reduce such heterogeneity in US images. This package allows the automatic removal of echographic watermarks, cropping and centering the echographic cone. Moreover, the echographic cone is converted in a rectangular region. The **EchoLocator** package is freely available on GitHub (`https://github.com/GDelCorso/EchoLocator`) under the MIT Public License. The folders organization is reported in Fig. 1. This software has been already applied in a real-world scenario as a fully automated preprocessing step to analyse fatty liver content using Bayesian Neural Networks.

2 EchoLocator Package

In this section we describe each part of the proposed algorithm, while in Fig. 2, the image preprocessing procedure is represented as a block diagram. All methods have been implemented in a Python 3.6 environment and the source code is available at `https://github.com/GDelCorso/EchoLocator` under the GNU General Public License. The package requires the following Python libraries: *NumPy*, *scikit-image*, *Pillow* (4.0.0), tqdm which can be automatically installed from the *requirements.txt* file. The preprocessing methods are wrapped in an *EcoLoc* class that deals with the management of US images, i.e. medical images produced by the US. The class aims to load images in different formats (i.e., .jpeg, .jpg, .png), allowing images to be loaded, verified and prepared for further/subsequent processing steps. This class takes the relative path to the folder containing the images (or frames).

2.1 Pre-cleaning

Several echographic images present annotations or watemarks inside the echographic cone.

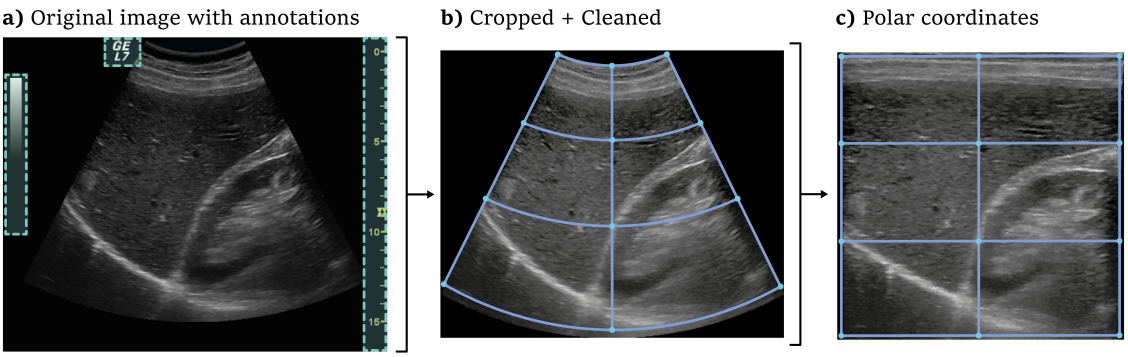

Figure 2: Normalisation steps available in the EchoLocator package including: cleaning of watermarks and annotations, cropping around the echographic cone, conversion to polar coordinates.

These confounding elements can be identified due to the fact that the pixels have different values between the colour channels, whereas the echographic image is usually stored in greyscale. Therefore, the *pre_remove*$(\cdot)$ method allows you to preprocess images as follows:

1. All pixels corresponding to values in $[0, \textit{black\_threshold}]$ are set to 0. This prevents spurious near-zero pixels outside the echographic cone, which can be generated by improper saving procedures (i.e., using lossy compression methods such as *.jpg*).

2. Colour pixels are identified using a utility function *test_color*$(\cdot)$ which identifies pixels whose difference between colour channels is greater than a given *color_threshold*$= 5$. These pixels are set to 0 and their positions are stored in an array *im_mask*.

3. If the reconstruction option *reconstruct* is set to true, the method tries to reconstruct the colour pixels. In particular, each pixel contained in the array *im_mask* is replaced by the average of the 4 top/bottom/right/left nearest pixels not contained in *im_mask*.

The preprocessed images are then stored in the input folder *InputEco*, ready for further processing.

2.2 Watermarks removal

Several images contain watermarks or annotations outside the echographic cone. These watermarks are also added using greyscale, so the Pre-cleaning strategy will not result in a clean image, as reported in Figure 3(a). The *no_watermark*$(\cdot)$ method applies a modified watershed algorithm to the echographic image.

Instead of using a single pixel as starting seed, this method uses a square at the image center, stored as a global list (*global_last_seeds_list*), with default size (square side length) *how_many_points*$=$ 10 (pixels). Starting with a square instead of a single seed avoids obtaining an incorrect echographic cone, which is usually made up of several disconnected components. The iterative utility function *fun_iterative_worm*$(\cdot)$ is called on each element of the global list. This function apply a standard watershed [3] to the given seed with two modifications:

1. Starting from the seed, the iterative function visits all the pixels around it, marking them as visited and adding them to the final image if the value is above a threshold (*threshold*$= 0$). The method will iterate a maximum number of times (500) before stopping and adding all endpoints to the *global_last_seeds_list* global list.

2. A global array of positions (*visited_mask*) is used to store the visited pixels to

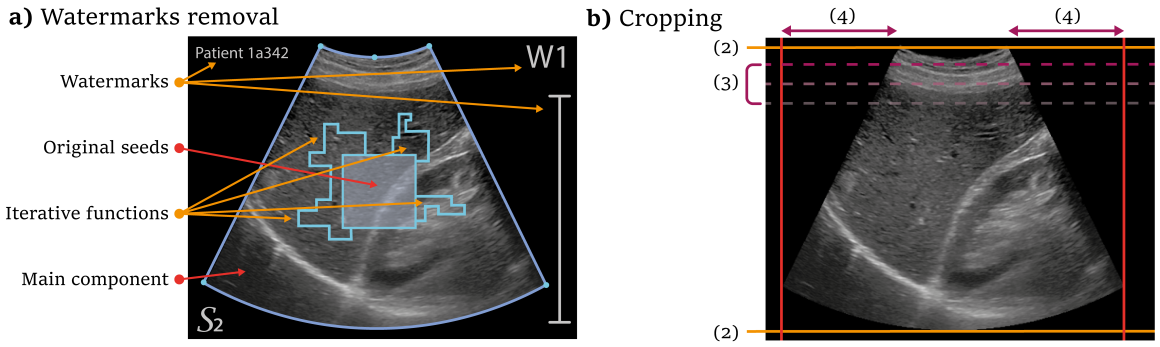

Figure 3: Panel a): Watermarks removal scheme. Panel b): Adaptive cropping strategy with the steps described in the section highlighted: Echographic Cone Cropping.

avoid unnecessary calculations or iterative loops.

The *fun_iterative_worm*$(\cdot)$ is applied to each element of the *global_last_seeds_list* for a number of times equal to the parameter *iteration*. In our tests, with a number of iterations equal to 3, we managed to visit every element of the echographic cone in every available patient.
In conclusion, this method allows to clean the echographic cone with a fast, computationally light and robust to multiple disconnected components modified watershed algorithm.

2.3 Echographic cone cropping

Echographic cones are usually not fully centred in the images. In addition, removing the watermark around the cone can result in a high percentage of the image being uninformative. Since the above techniques have already removed everything outside the echographic cone, the *standard_crops*$(\cdot)$ method is a modification of an approach based on identifying the minimum rectangle containing all non-null pixels in the image. Formally, as reported in Figure 3(b):

1. It defines a mask of non-null pixels as those with values above a given *threshold*.
2. First, the method removes all zero lines above and below the mask. In this way, the echographic cone is tangent to the upper and lower boundaries of the image (except for a few spurious pixels).
3. Then the top left/right elements of the cone are found by looking at the first *number_of_points_to_check*$=5$ rows. The top left is defined as the first column from the left, so that the first *number_of_points_to_check* are all non-null. The top right is defined in the same way, starting from the right. If the columns found are too far from the centre of the image (i.e., more than *correct_centre*$=10$ pixels), the step is repeated, doubling *number_of_points_to_check*.
4. The mask of non-null pixels is used to define the minimum left value and the maximum right value such that the mask is fully contained. The *left_distance* is defined as the distance between the top left column and the minimum left value. The *right_distance* is defined similarly. The best distance is the maximum between these two distances. This prevents the echographic cone from being cut in cases where half the cone is too dark to be retained.
5. The optimal left/right distance is defined as the minimum/maximum between the minimum left/maximum right value and the top left/right center minus/plus the best distance.

The resulting echographic cone is then linearly resampled to match the *resolution_output* reso-

lution parameter.

2.4 Polar coordinates

Convex probes produce images that are represented in a radial format. It may be useful converting these images to polar coordinates to better align the image representation with the probe geometry. Moreover, using Cartesian coordinates, the spatial resolution can vary depending on the distance from the probe, especially at the edges of the image. On the other hand, polar coordinates maintain a more uniform resolution with respect to the distance from the probe, improving image quality and reducing distortions. Accordingly, switching from Cartesian coordinates to polar coordinates is useful for improving the interpretation, analysis and processing of US images.

The *square_crops* method converts the images cropped in the previous step into polar coordinates and then resizes them into square images. This method takes as input the output of the *standard_crops*$(\cdot)$ method and is based on the definition of the *geometric cone*, as follows:

1. The top left/right points $(\text{Top}_L, 0)$, $(\text{Top}_R, 0)$ are computed as in Section 2.3.

2. The cone is defined by the aforementioned points and the two lower points $(\text{Top}_L - R, \text{Res}_y)$, $(\text{Top}_R + R, \text{Res}_y)$, where Res_y is the number of pixels on the y axis and R is the horizontal radius of the cone and has to be computed. R is computed using a bisection procedure, starting from the interval $R \in [\textit{delta\_pixel\_radius} = 5, \text{Res}_x]$, minimising the number of non-zero pixels outside the cone.

3. Once the optimal R is defined, the cone is fully determined and a polar transformation from polar coordinates to standard Cartesian coordinates is applied.

The new image is scaled to the given resolution *resolution_output* and saved.

3 Examples

Examples of the full pre-processing are shown in Figure 4. The original images include watermarks, some differences in the echographic cones, and annotations. After the *pre_remove(·)* method, the images are converted to greyscale and all colour elements are removed and replaced by a proper interpolation of the surrounding values. The *no_watermark(·)* and *standard_crops(·)* methods remove all but the echographic cone and crop to the echographic cone, providing an informative and standardised image without confounding factors. Then, the *square_crops(·)* method converts the images from Euclidean to polar coordinates, flattening the differences in cone radius and width.

4 Data availability

All images included in this technical report were extracted from video clips acquired from volunteers, who were fully informed about the use of their images in the study and provided explicit consent.

References

[1] Thomas L. Szabo. "Chapter 10 - Imaging Systems and Applications". In: *Diagnostic Ultrasound Imaging: Inside Out (Second Edition)*. Second Edition. Boston: Academic Press, 2014, pp. 365–430. ISBN: 978-0-12-396487-8. DOI: https://doi.org/10.1016/B978-0-12-396487-8.00010-0. URL: https://www.sciencedirect.com/science/article/pii/B9780123964878000100.

[2] Zhang Haoyan et al. "Application and prospects of AI-based radiomics in ultrasound diagnosis". In: *Visual Computing for Industry Biomedicine and Art* (2023).

[3] Luc Vincent and Pierre Soille. "Watersheds in digital spaces: An efficient algorithm based on immersion simulations". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.6 (1991), pp. 583–598.

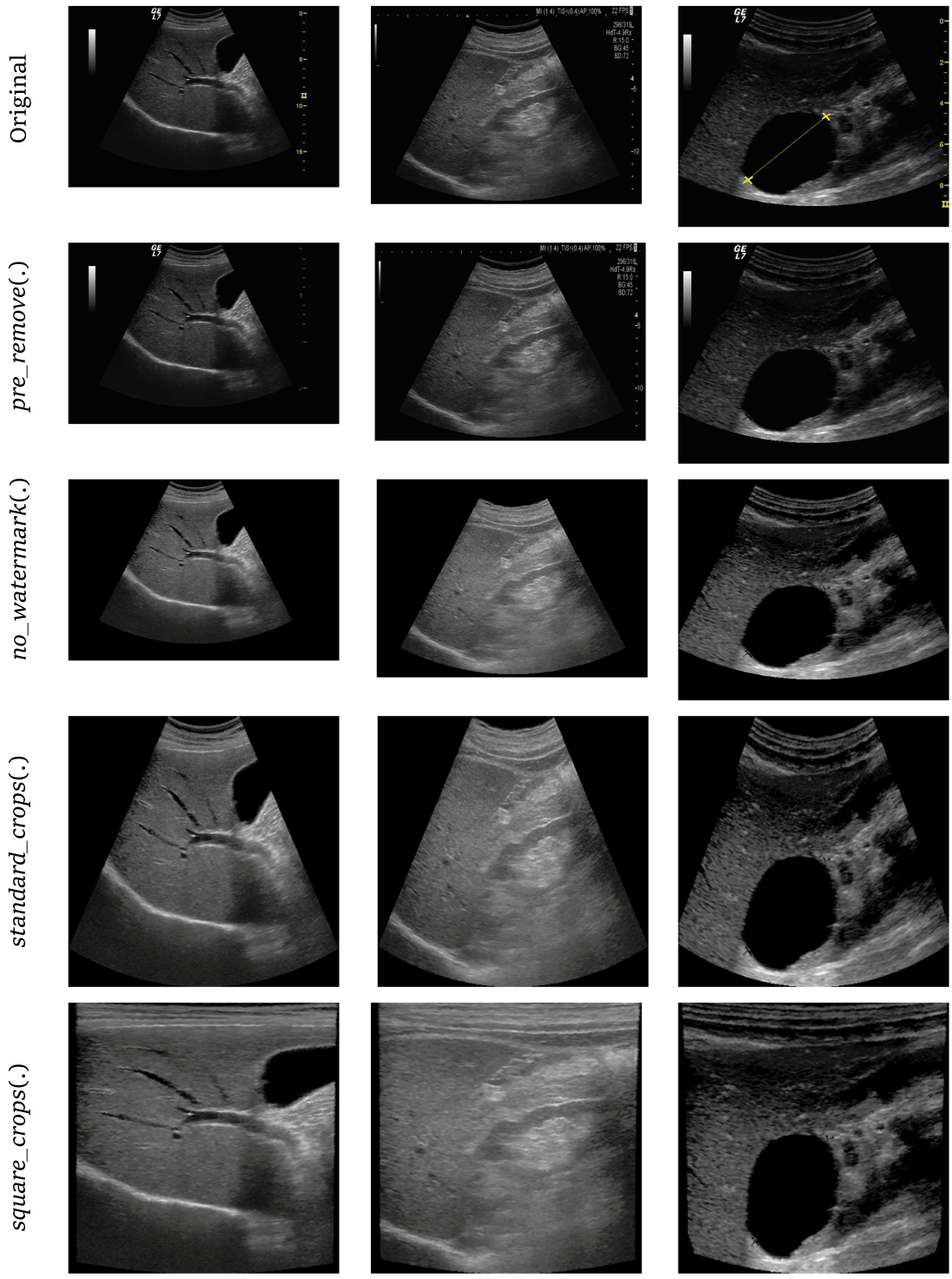

Figure 4: Examples of the entire preprocessing procedure, from the original images to those converted into polar coordinates.