ICT FOR HEALTH

Report n. **2**

Student: Gianluca Amprimo, s259455

Academic Year: 2019/2020

1. Introduction

1.1 Melanoma and moles ABCDE rule

Moles are skin lesions that are normally observed in an increasing number during a person's life. In most of the cases, moles are not dangerous but they should be kept under control because they could degenerate into melanomas. To identify when a mole is a potential melanoma, doctors have defined an "ABCDE" rule. The five letters represent properties of the lesion that should be regularly checked: Asymmetry, Border, Colour, Diameter and Evolution.
An interesting application of telemedicine in dermatology is to allow a patient to have a mole checked by a remote dermatologist, through pictures taken and sent to the doctor by the patient himself. Using a "live interactive" or "store and forward" approach, the image is analysed by the physician who establishes the severity of the lesion, without the need of visiting the patient in person.
This report discusses a possible implementation of a software to support doctor's decision process, focusing on extracting from mole images two numerical features to evaluate border indentation and asymmetry of the lesion. Moles being very asymmetric and having indented borders are very likely to be melanomas, even if also the other features are significant as well for a correct and complete diagnosis. These two extracted features could also be employed to train a classifier model able to automatically distinguish pictures of normal moles versus melanomas.

1.2 Mole dataset

The software has been tested on a dataset containing 54 RGB mole images of size 583x583, classified in three categories of increasing severity: low risk, medium risk and melanoma. The three sets contain respectively 11, 16 and 27 images. Figures 1,2,3 show examples of images contained in the dataset.
The images have been acquired under different light conditions and additional noise is provided by other captured elements, like body hair overlapping with the mole. Indeed, filtering was employed over the images to improve the results of the analysis.

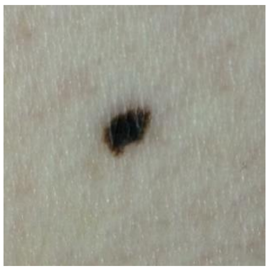


Figure 1: low risk mole (8)

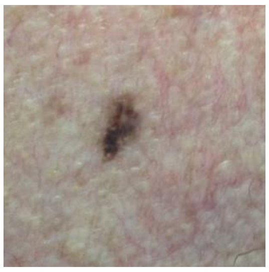


Figure 2: medium risk mole (6)

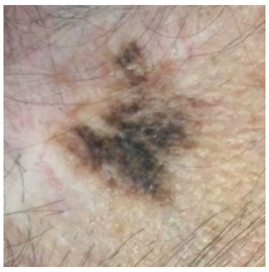


Figure 3: melanoma (23)

1.3 Software architecture

Figure 4 is a block diagram representation of the pipeline employed to extract the two coefficients of interest from each mole picture.
First of all, the input image is blurred by applying a rough median filter, in order to remove noise (ex: body hair) and have a better separation of the mole from the background. Then, the image is quantized applying K-Means algorithm to pixels, in order to identify n different coloured regions. The region with the darkest colour corresponds to the mole. To remove small dark regions around the real mole, DBSCAN is applied on the spatial coordinates of mole pixels: the cluster of pixels closest to the image centre is the real mole to analyse. The extracted mole (a binary image) is filtered again using a median filter to remove salt and pepper noise; the image is then cropped to focus on the lesion, whose centre is evaluated using the mean value of pixels coordinates. At this point, Sobel filter is applied to extract edges of the mole. However, some holes could be present inside the found border, so spatial DBSCAN is applied again to remove them: only the biggest contour is maintained. Finally, the lesion border is filled with colour using the flood fill algorithm; on this reconstructed mole, indentation and asymmetry coefficients are evaluated.

All the steps and their implementation are described in detail in section 2.

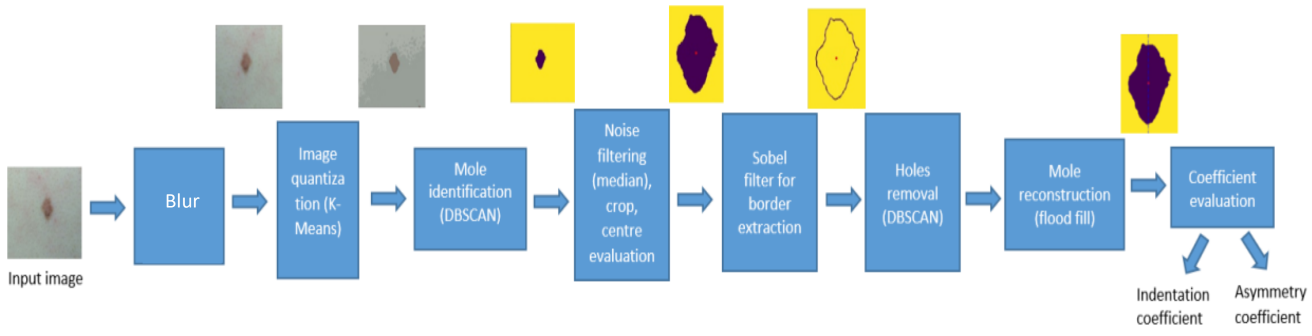

Input image
Blur
Image quantization (K-Means)
Mole identification (DBSCAN)
Noise filtering (median), crop, centre evaluation
Sobel filter for border extraction
Holes removal (DBSCAN)
Mole reconstruction (flood fill)
Coefficient evaluation
Indentation coefficient
Asymmetry coefficient

Figure 4: Block diagram of software workflow

2. Software pipeline implementation

2.1 Step 1- Blur

2.1.1 Blur rationale

Blur is a common operation in image processing. It consists in weighting each pixel by its neighbours, such that very strong colour variations are removed, smoothing the original image. A low level of blur can be used to remove noise, preserving the main image features: in figure 5, for example, blur reduces the effect of body hair (noise) in a mole image.

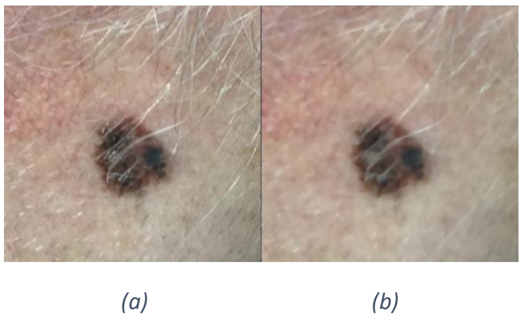


Figure 5: a) image of a melanoma before blur; b) same image of melanoma after blur

2.1.2 Median filter

Median filter is a digital filter generally applied to remove "salt and pepper" noise from images. With respect to other filters, like Gaussian filter, the median filter is a non-linear filter, because the outcome produced for each pixel cannot be expressed as a linear combination of all the pixels considered in input. However, it tends to better preserve the edges and has a quite straightforward implementation; the only significant drawback is related to computational time, which is longer than other possible filters for noise removal. Nevertheless, having no specific time constraints and observing that execution time was still competitive, this algorithm has been selected for all the filtering operations in the software.

Median filter works sliding over each pixel of the image a window of size SxS, where S is a parameter to be defined. Typical value for S to perform noise removal is S=3, meaning that 9 pixels are considered at a time. For large S, more pixels fall in the window, therefore a rougher result is produced.

It is important to define what will be the behaviour of the window when it slides over the borders of the image. In this case, supposing that the contour of the mole does not fall over the image border, the filter was implemented such that it works only on pixels where the window perfectly fits. Pixels that cannot be filtered are identically reproduced in the filtered image.

Supposing an image of size N_1xN_2 with only 1 channel and a window with radius $r = floor\left(\frac{S}{2}\right)$, median filter implementation is the following:

```
For each pixel $P$ with coordinates (i,j)
    1. If $(i - r < 0 \; or \; j - r < 0 \; or \; i + r >= N_1 \; or \; j + r >= N_2)$ is true (window does not fit)
       then set $filtered\_image[i, j] = P$ and move to next pixel, else go to point 2
    2. Extract from the image the window $W$ centred in $P$ and add elements in $W$ to list $L$
    3. $Sort(L)$
    4. Select median $M = L[floor\left(\frac{S^2}{2}\right)]$ (list $L$ starts from index 0)
    5. Set $filtered\_image[i, j] = M$
```

In code implementation, the sorting was performed using the default sort method provided by Numpy package, which is Quicksort, but any efficient algorithm is equally effective. Figure 6 contains a graphical representation of the algorithm for a 4x4 matrix with a 3x3 window. For RGB images having 3 channels, the filter must be applied to each channel separately.

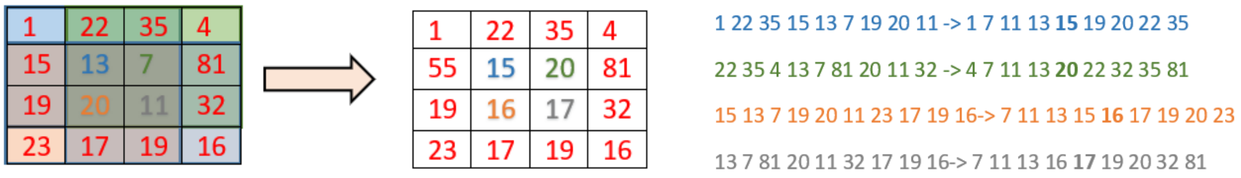


Figure 6: median filter in action: red pixels are not filtered, the other colours represent a movement of the filter window

2.1.3 Blur using median filter

As already mentioned, if the size of the window is large, each pixel is smoothed considering a larger region, which produces a rougher filtering. This can be exploited to blur the image. The default blur was performed using a window with size S=7, but some images required even more blur so windows with S=15,20,25 were employed. This is mainly the case of melanoma images that contain too many colour shades inside. The blur was necessary to mix them enough for clustering to identify the whole mole and not only part of it. Figure 7 shows the example of image *melanoma_4*, which required a window of size 20.

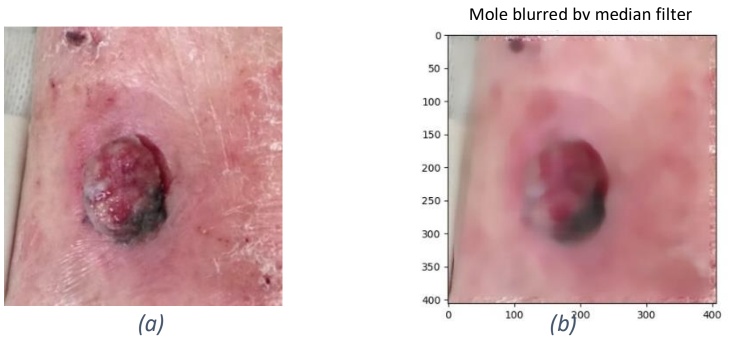


Figure 7: a) melanoma_4 before blur; b) melanoma_4 after median blur with size 20

2.2 Step 2- Colour quantization

To extract the mole, colour quantization is applied. Colour quantization means clustering pixels of an image according to their value, which represents a colour shade. For each found cluster, its centroid is evaluated.

At this point, the original image can be reconstructed replacing all pixels with the centroid of the cluster they were assigned to. The resulting picture has only K different colours, K being the number of identified clusters. The darkest of them represents the mole region.

To perform clustering, K-Means algorithm was selected, because it allows to choose a priori the number of clusters to find. The default setting was K=3. However, this choice was not enough to separate well the mole from the background for some images, so larger values, like 5 and 7, were selected. Moreover, for other moles with many shades inside but quite a large contrast with the rest of the image, a lower value K=2 proved better, as shown in figure 8.

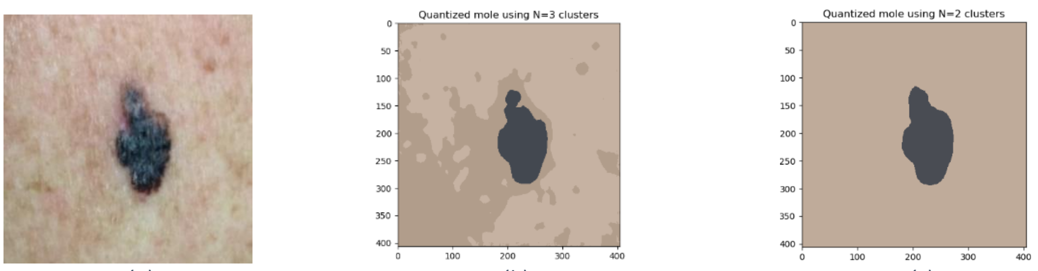

Quantized mole using N=3 clusters
Quantized mole using N=2 clusters
(a) (b) (c)

Figure 8: a) melanoma_3; b) melanoma_3 quantized in 3 clusters; c) melanoma_3 quantized in 2 clusters

2.3 Step 3 - Mole identification through DBSCAN

In general, other regions in addition to the mole can be associated to the darkest colour after quantization: they represent noise to remove before next operations.
In this step, the quantized image is converted to a binary image where all the dark pixels are mapped to 0 (mole pixels) and the remaining are all mapped to 1. Figure 9 shows, for example, the case of image *low_risk_4*, which has in binary form a large noisy region on the right. Hence, to remove noise, clustering is performed again, but using another algorithm, DBSCAN, on the spatial coordinates of mole pixels, not their colour. DBSCAN identifies cluster using a concept of density, therefore it separates well the real mole from external noise, assigning them to different clusters. A summary of the algorithm is presented in the following:

1. Extract from the binary mole image only the coordinates of pixels with value 0 (mole pixels)
1. Run DBSCAN over coordinates found in 1
2. Compute centroids of identified clusters
3. Compute Euclidean distance between image centre and each centroid
4. Select as mole the cluster associated to the centroid closest to image centre
5. In the original binary image maintain to 0 only pixels of cluster selected in point 4, put to 1 all the others.

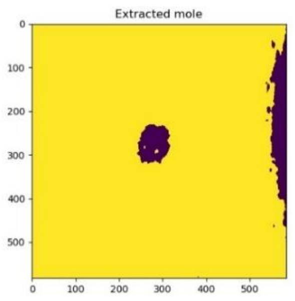


Figure 9: Binary low_risk_4 after quantization

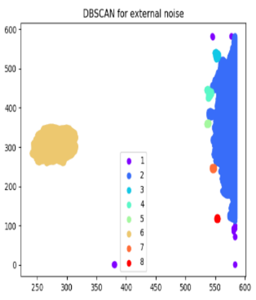


Figure 10: DBSCAN on low_risk_4

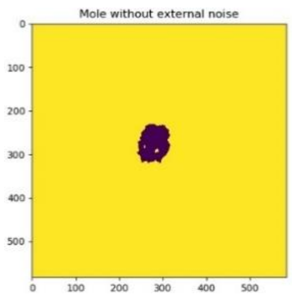


Figure 11: mole after DBSCAN filtering

DBSCAN requires two hyperparameters: the radius ε and M, the number of points that must be present in a point hypersphere to consider it a core. These were set to ε =5 and M=15 using a "trial and error" approach and they provide an effective clustering for all images in the dataset. Different images could require modifications to these values.
Results of DBSCAN filtering are shown in figure 10 and 11. As described in the algorithm, only the yellow cluster is maintained, being the closest to the image centre. This choice comes from the reasonable assumption that all images will be centred around the mole to analyse. It fails only in a very special case, *melanoma_17*: the mole, figure 12, has two components, a large one and a small one, separated by a lighter coloured region. Only one of the two parts (the closest to the centre) is considered, whereas the other is discarded. Due to the strong colour separation between the two parts it would be impossible, even applying a lot of blur, to consider both simultaneously. The only way to fix this issue would be to ask the doctor for help; nevertheless, being a very special case, it was not addressed in this report.

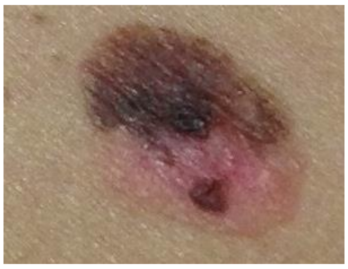


Figure 12: Melanoma_17

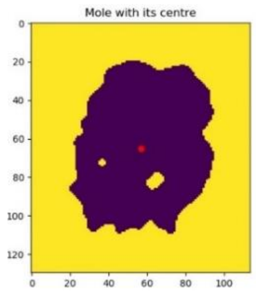


Figure 13: low_risk_4 after crop, in red the mole centre

2.4 Step 4: Binary filter, crop and centre evaluation

The binary image containing the extracted mole passes again three times under a median filter (implemented as discussed in 2.1.2) of size S=3, in order to remove "salt and pepper" noise after quantization. This improves results for the subsequent border extraction. Then, the image is cropped, maintaining a 20 px border around the mole, as shown in figure 13 for image *low_risk_4*. After this operation the coordinates of mole centre are evaluated using the concept of mean

$$x_{centre} = \frac{1}{N}\sum_{i=1}^{N} x_i \quad y_{centre} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

in which (x_i, y_i) are the coordinates of a mole pixel, N is the total number of pixels composing the mole.

2.5 Step 5: Sobel filter for border extraction

To extract the contour of the mole, which is used to evaluate indentation and asymmetry, Sobel filter (also known as Sobel operator) is applied to the binary image generated after step 4.
Sobel filter evaluates in each pixel of the image the gradient G of the image intensity function: points where the modulus of the gradient is large correspond to high colour intensity variations, hence edges of the figure. The gradient computation is done using two 3x3 kernels K_x and K_y and split along a horizontal component G_x and a vertical one G_y:

$$K_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad K_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$G_x = K_x * A \quad G_y = K_y * A$$

in which A is the binary image and $*$ is the convolution operator (G_x and G_y are two matrices storing the gradient along the 2 components for each pixel in A). The matrix G, which stores the modulus of the overall gradient for each pixel, can be then evaluated as

$$G = \sqrt[2]{{G_x}^2 + {G_y}^2}$$

Matrix G is the filtered image containing the extracted contour of the mole.
This operator was implemented as follows:

```
For each pixel $P$ with coordinates $(i, j)$ in the binary image
    1. Extract a 3x3 window $W$ centred in $(i, j)$. If $P$ is on image border, extract only values available
       and perform zero padding for empty positions.
    2. Compute $g_x = \sum_{h=1}^{3}\sum_{k=1}^{3} w_{h,k} k_{x\,h,k}$ , with $w_{h,k} = W[h,k]$ , $k_{x\,h,k} = K_x[h,k]$
    3. Compute $g_y = \sum_{h=1}^{3}\sum_{k=1}^{3} w_{h,k} k_{y\,h,k}$ , with $w_{h,k} = W[h,k]$ , $k_{y\,h,k} = K_y[h,k]$
    4. Compute $g = \sqrt[2]{{g_x}^2 + {g_y}^2}$
    5. If $g$ > threshold, then set $filtered\_image[i,j] = 0$, else set $filtered\_image[i,j] = 1$.
```

Figure 14 shows an example (without thresholding) of one iteration of the algorithm.

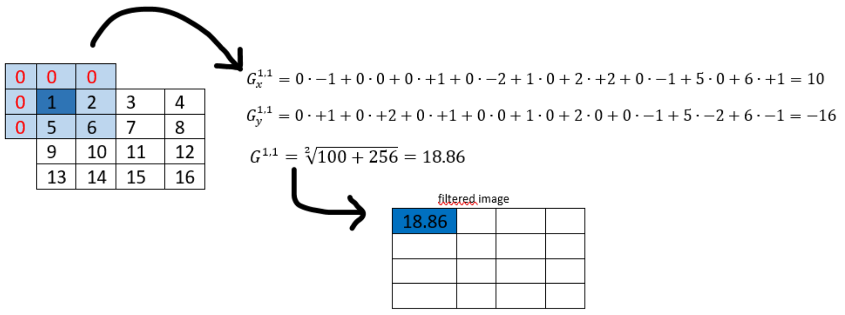

0 0 0
0 1 2 3 4
0 5 6 7 8
9 10 11 12
13 14 15 16
$G_x^{1,1} = 0\cdot-1+0\cdot0+0\cdot+1+0\cdot-2+1\cdot0+2\cdot+2+0\cdot-1+5\cdot0+6\cdot+1 = 10$
$G_y^{1,1} = 0\cdot+1+0\cdot+2+0\cdot+1+0\cdot0+1\cdot0+2\cdot0+0\cdot-1+5\cdot-2+6\cdot-1 = -16$
$G^{1,1} = \sqrt[2]{100+256} = 18.86$
filtered image
18.86

Figure 14: one iteration of Sobel filter; the filter does not fit so zero padding is performed on borders

As stated in point 1, when the window to extract does not fit, because we are considering as centre a pixel on image border, missing values are replaced with zeroes (red in the example). This is generally known as zero padding.
Point 5 is needed in order to obtain again a binary image, where contour pixels are set to 0 and all the other pixels to 1: this operation is called thresholding.
Thresholding applies a threshold to all pixels in the image: all values above threshold are mapped to 0, the others to 1. As an example, the result for image *low_risk_4* can be seen in figure 15. The value of the threshold has great importance, because a too high threshold can produce a thin border with holes, which fails all the subsequent steps to perform. For the provided dataset, in order to extract a continuous border for all images, the threshold was set to 2.8, using again a trial and error approach. With this value, however, the found border becomes "thick": as shown in figure 16, it is made on average by 2 lines of pixels placed side by side. This provides a better visualization, but it affects mole perimeter evaluation as explained in 2.8.1.

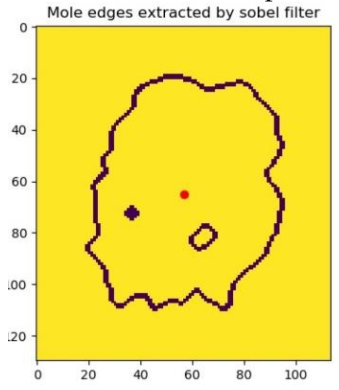


Figure 15: contour of *low_risk_4* extracted by Sobel filter

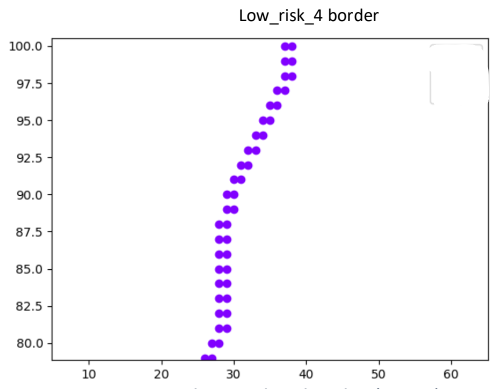


Figure 16: *low_risk_4* border (zoom)

2.6 Step 6: Holes removal using DBSCAN

As figure 15 shows, the found contour can contain inside some holes: due to noise, colour quantization does not always identify a full mole. Small holes are eliminated by the additional filtering in step 4, but the bigger ones require further processing to be removed.
To perform holes removal an algorithm similar to the one described in 2.3 was developed with few adjustments:

- DBSCAN is executed only on coordinates of pixels identified as border by Sobel filter, with ε =3 and M=2. These very small values for the parameters are necessary to avoid that holes close to edges are considered as part of the mole border as well.
- The cluster to choose is the one having more pixels inside, hence the largest perimeter. The perimeter P_{mole}, indeed, is defined simply as a count of the pixels composing the chosen border and it is precomputed here but used later to define indentation. All smaller clusters are removed from the binary image of the border, setting the corresponding pixels to 1.

Figure 17 and 18 show an example of the algorithm for *low_risk_4*. In Figure 19, how found border overlaps with the original mole image.

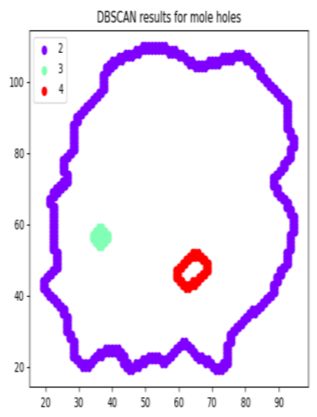


Figure 17: holes identified by DBSCAN

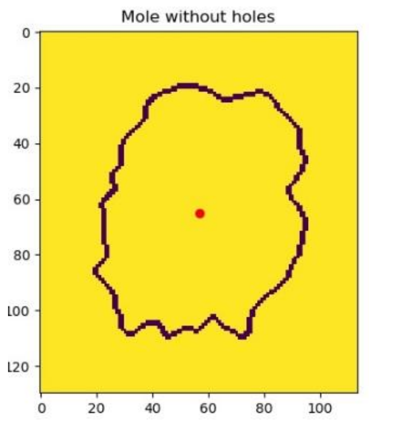


Figure 18: clean border after holes removal

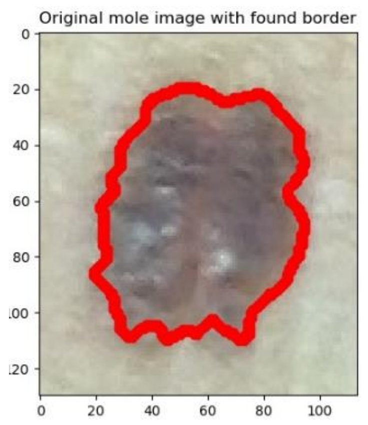


Figure 19: border of mole low_risk_4

2.7 Step 7: Mole reconstruction through flood fill algorithm

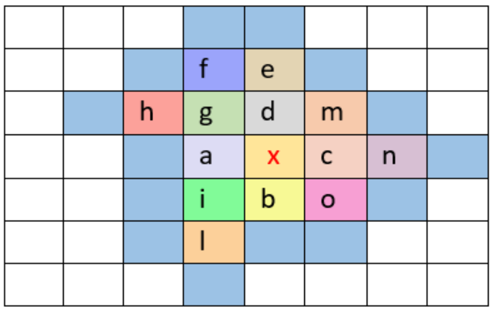
In order to evaluate indentation and asymmetry, points inside the border should be identified. For this purpose, the mole is reconstructed filling the extracted binary border with colour (0 pixels). This operation is performed using the flood fill algorithm.

Flood fill algorithm is a common approach for colouring a region enclosed by a contour, starting from an inside point. When border pixels are encountered, the algorithm stops (for this reason is very important to start from a contour without holes, otherwise flood fill fails).

This algorithm can be implemented using a recursive approach; however, this technique was avoided because it can produce stack overflow when running on very large images on a computer with limited RAM available. A non-recursive version, based on a queue of points to explore, exists as well and was the one implemented:

1. Insert mole centre $C\ (x_c, y_c)$ in queue Q
2. Extract point of coordinates (x, y) from Q head
3. If $image[x, y] == 1$ then set $image[x, y] = 0$ (colour it), else go directly to point 5
4. Consider a neighbourhood of radius 1 centred in (x, y): push neighbour points with value 1 in queue Q, points with values 0 (border) are not considered
5. Repeat from point 2 until Q is empty

Figure 20 shows an example.



Q={x}
It 1: Q={g,d,m,a,c,i,b,o}
It 2: Q={d,m,a,c,i,b,o,h,f,e}
It 3: Q={m,a,c,i,b,o,h,f,e}
It 4: Q={a,c,i,b,o,h,f,e}
It 5: Q={c,i,b,o,h,f,e}
It 6: Q={i,b,o,h,f,e,n}
It 7: Q={b,o,h,f,e,n}
It 8: Q={o,h,f,e,n,l}
It 9: Q={h,f,e,n,l}
It 10: Q={f,e,n,l}
It 11: Q={e,n,l}
It 12: Q={n,l}
It 13: Q={l}
It 14: Q={}
END

Figure 20: Flood fill example starting from centre x. Letters stand for coordinates, colours show order in which mole is filled

2.8 Step 8: Coefficients evaluation

2.8.1 Indentation coefficient

Once obtained the reconstructed mole, indentation coefficient is evaluated using this procedure:

1. Count number of mole pixels in reconstructed mole. This is the mole area A
2. Compute circumference of an equivalent circle: $C_{ideal} = 2\sqrt[2]{\frac{A}{\pi}}\,\pi$
3. Evaluate indentation coefficient using mole perimeter P_{mole}, computed in 2.6, as $indentation = \frac{\frac{P_{mole}}{2}}{C_{ideal}}$

An equivalent circle is used because it is the least indented figure in geometry. As already pointed out in section 2.5, the mole border is a thick line made on average by two parallel pixel lines, therefore the value found in 2.6 for perimeter is the double of the true mole perimeter. Since this property was observed for all moles, P_{mole} is empirically divided by 2 in indentation formula, to have a truer perimeter estimation at the numerator. Large values of the coefficient are associated to very indented moles, as proved by results presented in section 3.

2.8.2 Asymmetry coefficient

Asymmetry is computed evaluating first the complementary concept of symmetry.
Reflection symmetry was used: an object is symmetric if it has at least one axis of symmetry, hence cutting the object with this axis, the two obtained halves are identical.
The axis of symmetry always passes through the centre of the object, so its generic equation on a Cartesian plane can be defined as

$$y = m(x - x_c) + y_c$$

in which m is the angular coefficient, (x_c, y_c) are the coordinates of mole centre C. If the axis is vertical (m goes to infinity), the equation becomes

$$x = x_c$$

To compute the specular point P' $(x_{p'}, y_{p'})$ given a non-vertical axis and a point P (x_p, y_p):

1. Find the line perpendicular to the axis passing through P, whose equation is
$$y = -\frac{1}{m}(x - x_p) + y_p$$
2. Find intersection I (x_i, y_i) between line in point 1 and axis of symmetry
3. Compute P' $(x_{p'}, y_{p'})$ imposing that I is the middle point in segment $\overline{PP'}$
$$x_i = \frac{x_{p'} + x_p}{2}, \quad y_i = \frac{y_{p'} + y_p}{2}$$

From this method, P' coordinates can be computed as a function of m, P and C

$$x_{p'} = 2\frac{x_p + m^2 x_c + m(y_p - y_c)}{1 + m^2} - x_p$$

$$y_{p'} = 2\left(m\left(\frac{x_p + m^2 x_c + m(y_p - y_c)}{1 + m^2} - x_c\right) + y_c\right) - y_p$$

If the axis is vertical, these coordinates reduce to

$$x_{p'} = x_c + (x_c - x_p)$$
$$y_{p'} = y_p$$

Starting from this theoretical background, the symmetry of a mole with respect to one of its axes can be measured in this way:

```
1. Set $n_{left} = 0, n_{right} = 0,$ $match_{left} = 0,$ $match_{right} = 0$
2. For each mole pixel $P$ in reconstructed mole:
    i. Classify $P(x_p, y_p)$ as left, right or axis according to its position with respect to axis.
       Case left: increment $n_{left}$ by 1. Case right: increment $n_{right}$ by 1. Case axis: increment
       all counters by 1 (points on axis are their own specular) and move to next pixel
    ii. Compute specular $P'\left(x_{p'}, y_{p'}\right)$ as shown. If $P' == 0$ (mole point) then increment
       $match_{left}$ (if $P$ is left, otherwise increment $match_{right}$ ) by 1, else do nothing
3. Compute $symmetry_{left} = \frac{match_{left}}{n_{left}}$ , $symmetry_{right} = \frac{match_{right}}{n_{right}}$
4. Compute $symmetry_{axis} = \min(symmetry_{left}, symmetry_{right})$.
```

The relative position of a pixel required in point 1.i can be inferred replacing pixel coordinates in the axis equation: if a value smaller than 0 is obtained, point is on the left; if one bigger than 0, it is on the right and if one equal to 0, it is on the axis itself. Overall, symmetry is defined as the minimum between symmetry computed only considering points on the left side of the axis and symmetry considering only points on the right. This is done because one half of the mole could completely match (symmetry coefficient=1), but the other side could be much less symmetric. The selection of the minimum provides a more precise measurement.
Symmetry coefficient, because of its definition, is included between [0,1]. Therefore, asymmetry coefficient can be evaluated as

$$asymmetry = 1 - symmetry_{main\ axis}$$

where $symmetry_{main\ axis}$ is symmetry value measured considering the main axis of symmetry of the mole, which is the axis with respect to the mole is the most symmetric.
To identify the main axis and its corresponding symmetry value, the following procedure was used:

```
1. Start from a vertical axis ($m = \infty$), set $symmetry_{main\_axis} = 0$, $i = 0$.
2. Compute $symmetry_{current\_axis}$ as previously shown
5. If $symmetry_{current\_axis} > symmetry_{main\_axis}$, consider current axis as main axis of
   symmetry and set $symmetry_{main\_axis} = symmetry_{current\_axis}$ . Set $i = i + 1$.
6. Rotate axis by α: compute new $m = \tan(90° - \alpha i)$ . If after rotation axis is vertical again stop
   (a 180° total rotation has been performed), else iterate again from point 2 for the new axis
```

For the dataset analysis, rotation α performed at each step was set to 20°: smaller values can provide better results, but they make computations longer (more axes are considered). Large values of the asymmetry coefficient imply a more asymmetric shape. It should be taken into account that pixels have integer coordinates, hence some approximation is performed when computing the specular P'. However, this technique provides good results, as shown in figure 21, which highlights points that found a match on both sides.

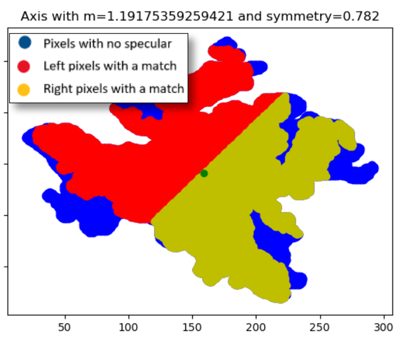


Figure 21: best axis for melanoma_23

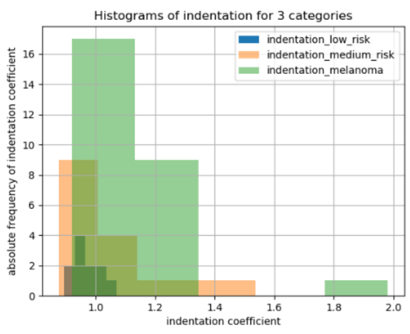


Figure 22: histograms of indentation

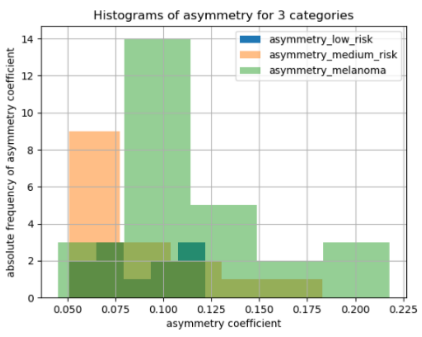


Figure 23: histograms of indentation

3. Results and conclusions

Table 1 contains indentation and asymmetry for all the moles in the dataset and mean and standard deviation of the two coefficients for each of the three categories. As expected, mean value of both indentation and asymmetry increases moving from low risk to melanoma. Moreover, also standard deviation grows: more dangerous categories show more variability for the two coefficients (indeed low risk moles are all very regular and similar with respect to these properties). In general, low risk moles have an indentation value smaller or very close to 1, as shown in indentation histograms (fig. 22); medium risk moles are concentrated in the same interval in 57% of the cases, whereas 43% of the lesions exceeds 1, so they show an intermediate behaviour; melanomas have an indentation bigger than 1 81% of the times and in the remaining 19% the value is never smaller than 0.9. The minimum indentation 0.876 is associated to *medium_risk_1*, which is very close to a circle; the maximum indentation 1.982 is reached for *melanoma_23* which is indeed the most indented mole in the dataset. Therefore, with respect to indentation, a value larger than 1 should trigger an alarm for a possible melanoma. Considering asymmetry (histograms fig. 23), values are more spread: low risk moles have values smaller than 0.125; medium risk lesions overlap with low risk (they even have the same mean) but for few values bigger than 0.125. Melanomas, instead, show more separation: 88% of melanomas have an asymmetry coefficient bigger than 0.075, 48% bigger than 0.100. The maximum of this coefficient, 0.218, is found again for *melanoma_23* (fig. 21). In general, a mole with asymmetry coefficient bigger than 0.100 is likely to be a melanoma. However, it should be considered that the number of moles provided in the dataset is small: more data could lead to better histograms and a clearer view on possible thresholds to distinguish non dangerous moles from melanomas. Finally, it should also be considered that the other 3 features of the ABCDE rule are relevant as well to identify a melanoma: a smooth edged, symmetric mole can still be a melanoma because, for example, it contains different colour shades or it is quite large or it has undergone a strange evolution in time. These cases can bias the results, nevertheless the two coefficients extracted by the proposed pipeline are a good starting point for further analysis.

Table 1: Indentation coefficient, asymmetry coefficient for all analysed images

	Low risk		**Medium risk**		**Melanoma**	
ID	**Indent**	**Asym**	**Indent**	**Asym**	**Indent**	**Asym**
1	1.010	0.095	0.876	0.070	1.014	0.086
2	0.945	0.051	0.942	0.059	1.068	0.128
3	1.071	0.108	0.926	0.051	1.063	0.108
4	1.000	0.073	0.961	0.108	1.066	0.099
5	0.994	0.105	1.293	0.142	1.204	0.096
6	0.949	0.067	1.152	0.183	1.331	0.118
7	0.964	0.122	1.079	0.076	1.085	0.084
8	1.024	0.113	1.028	0.068	0.965	0.045
9	0.895	0.092	1.064	0.103	1.137	0.097
10	0.899	0.057	0.950	0.092	1.092	0.094
11	0.931	0.069	1.536	0.107	1.125	0.104
12			0.940	0.055	1.001	0.072
13			0.974	0.071	0.989	0.084
14			0.955	0.084	1.041	0.089
15			1.016	0.068	1.312	0.175
16			0.982	0.053	1.141	0.094
17					0.986	0.214
18					0.968	0.080
19					1.126	0.079
20					1.145	0.155
21					1.267	0.136
22					1.021	0.115
23					1.982	0.218
24					1.245	0.186
25					0.921	0.091
26					1.136	0.128
27					1.011	0.110
mean	0.971	0.087	1.042	0.087	1.127	0.114
STD	0.052	0.023	0.161	0.035	0.198	0.041