Explicação do código

A escala depende de cada foto, o ideal que todas sigam um padrão de igual: distância focal e tamanho dos furos.

**Image.py**

Basicamente, o arquivo image.py é quem está responsável pelos tratamentos ou alguma manipulação na imagem que podem ser necessários, como o desenho dos círculos na imagem para se identificar a área do furo e da “coroa”.

**Importações:**

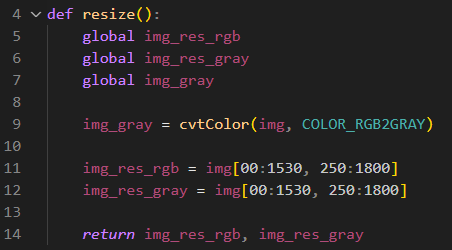


cv2: Biblioteca OpenCV

cvtColor: Método que permite alterar o formato de cores de imagens. Exemplo: Transformar uma imagem com pixeis de cores RGB em apenas pixeis de tons de cinza.

COLOR\_RGB2GRAY: Transforma uma imagem com pixeis de cor baseada em RGB para tons de cinza.

**Função “resize”:**

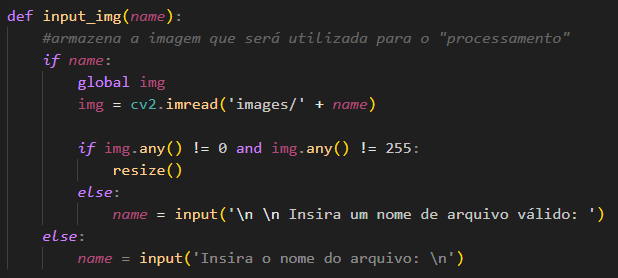


Na imagem acima, a função “resize()” está criando variáveis globais que serão utilizadas por outras funções. Seu papel principal é delimitar o tamanho da imagem a ser tratada.

img\_gray: Transforma a imagem inserida pelo usuário em cores que estão apenas na escala de cinza.

Img\_res\_rgb e Img\_res\_gray: Redimensionamento da imagem indicada pelo usuário a ser analisada. São retornadas para que sejam utilizadas em outras funções. “img\_res\_gray” será utilizado no cálculo do fator de delaminação a partir da contagem de pixeis.

**Função “input\_img(name)”:**



A função “input\_img(name)” receberá a imagem que o usuário quer verificar.

“if name”: verifica se algum nome foi inserido pelo usuário

“global img”: declaração da variável que armazenará a imagem do usuário

“img = cv2.imread(‘imagens/’ + name)” = armazenamento da imagem na variável.

“cv2.imread(‘caminho\_arquivo’)”: método da biblioteca OpenCV que armazena uma matriz de pixeis correspondentes as da imagem. Dentro dos parênteses deve ser colocado o caminho da imagem que se deseja guardar.

“if img.any() != 0 and img.any() != 255”: faz a verificação se há algum valor na variável “img”, a partir da análise de todos os valores de cada pixel da imagem.

Essa verificação foi feita dessa forma devido as matrizes armazenadas serem tridimensionais, então a função “any()” pega qualquer valor de um pixel da variável “img”, que se torna uma matriz por causa dos valores da imagem.

“resize()”: redimensiona a imagem caso a variável tenha conseguido encontrar uma imagem que possui um nome correspondente ao que o usuário informou.

“name=input(...)”: Exibi um texto e junto com ele, permiti a digitação do nome do arquivo, caso estivesse incorreto ou não tenha inserido.

**Função “measures()”:**



A função acima tira as medidas da imagem.

“center\_x, center\_y”: Juntos dão o ponto central da imagem, será usado como base para a medição do círculo que contornará os sinais de delaminação, pois o furo está posicionado no centro da imagem, facilitando.

“cols”: Quantidade de colunas da imagem. Será utilizada para encontrar o centro do eixo x.

“lines”: Quantidade de linhas da imagem. Será utilizada para encontrar o centro do eixo y.

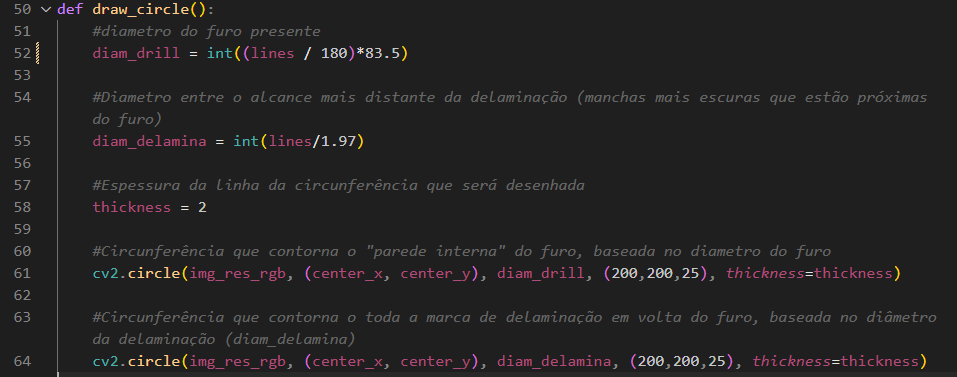
“wResi”: Comprimento da janela que exibirá a imagem com o desenho do círculo formado.

“hResi”: Altura da janela que exibirá a imagem com o desenho do círculo formado.

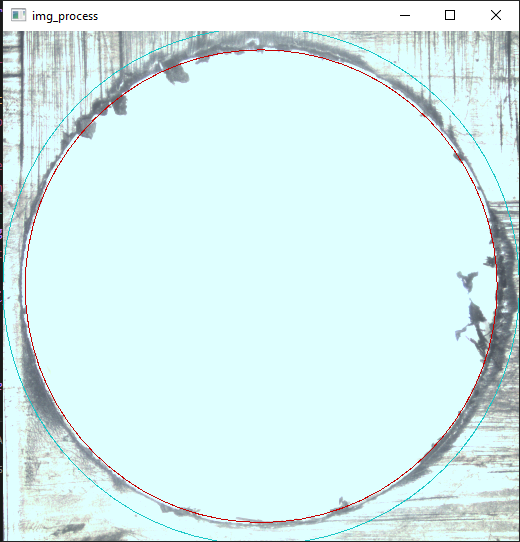
“cols = img\_res\_rgb.shape[1]”: Armazena a quantidade de pixeis do eixo x.

“lines = img\_res\_rgb.shape[0]”: Armazena a quantidade de pixeis do eixo y.

**Função “draw\_circle()”:**



A função “draw\_circle()” está responsável por desenhar dois círculos, um corresponde ao furo realizado e o outro ao diâmetro máximo alcançado pela delaminação.



“diam\_drill”: Corresponde ao círculo que contornará as beiradas do furo realizado. Sua medida não é algo padronizado e tão prático, é necessário fazer ajustes nele até que passe a contornar as beiradas do furo. Baseado na imagem acima, é o círculo vermelho.

“diam\_delamina”: Corresponde ao círculo que contornará o alcance máximo alcançado pela delaminação. Sua medida não é algo padronizado e tão prático, é necessário fazer ajustes nele até que passe a cercar toda a delaminação presente. Baseado na imagem acima, é o círculo azul.

“thickness”: Espessura da linha que constituirá o círculo.

1° “cv2.circle()”: Desenha o círculo que contornará o furo.

2° “cv2.circle()”: Desenha o círculo que contornará o alcance máximo da delaminação.

**Explicação dos parâmetros passados para “cv2.circle()”**

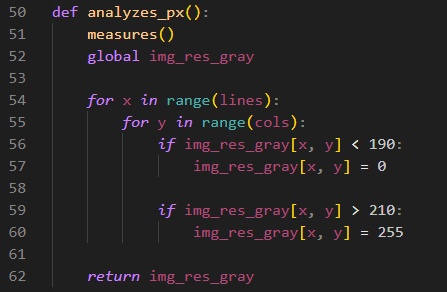
1° parâmetro (img\_res\_rgb): Imagem que a função deverá realizar o desenho. “Img\_res\_rgb” está sendo usado por receber a imagem que o usuário deseja que o algoritimo trabalhe em cima.

2°parâmetro [(center\_x, center\_y)]: Qual se localiza o centro do círculo que se deseja que seja desenhado. “center\_x” e “center\_y” estão sendo utilizado por passarem o valor mais próximo do centro do furo.

3°parâmetro (diam\_drill): Representa o diâmetro da circunferência que deve ser desenhada.

4°parâmetro [(200,200,25)]: Valor da cor da linha a ser traçado, sendo o contorno da circunferência. Está ordenado em “BGR” (blue, green, red).

5°parâmetro (thickness = **thickness**): Expressa a espessura da linha que vai ser usada. 1° (thickness) faz parte da declaração do parâmetro, já o 2°(**thickness**) é a igualdade. Legível como: thickness será igual a variável **thickness**. Caso não seja desejado utilizar uma variável para isso, é possível simplesmente declarar um número diretamente, ficando assim: thickness = 4. A função interpreta o número passado em pixeis.

Função “analyzes\_px()”:  


Faz uma verificação do valor de cada pixel, arredondando-o para preto ou branco, tornando os valores da imagem padronizados, apenas branco ou preto, deixando de existir a escala de cinza.

“measures()”: Chama a função measures para se obter as variáveis “cols” e “lines”.

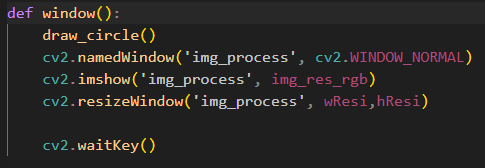
1°for: Passa por todas linhas (pixels) da imagem.

2°for: Passa por por todas as colunas da imagem realizando duas verificações.

1°if: Se o valor do pixel do par ordenado (x,y) for menor do que 190, tornará aquele pixel totalmente preto.

2°if: Se o valor do pixel do par ordenado (x,y) for maior do que 210, tornará aquele pixel totalmente branco.

**Função “window()”:**



A função “window()” está responsável por retornar uma janela que exibe o resultado da imagem com o círculo que demarca as bordas da delaminação(alcance máximo) e do furo.

“draw\_circle()”: A função é chamada para que a circunferência seja desenhada na imagem e então exibida para o usuário.

“cv2.namedWindow()”: Criação de uma janela onde será exibida a imagem.

“cv2.imshow()”: Exibe a imagem desejada na janela que foi criada anterior a essa função.

“cv2.resizedWindow()”: Ajuste no tamanho da janela de exibição.

“cv2.waitKey()”: Fecha a janela apenas quando o usuário clicar em alguma tecla ou então fechar com o mouse no “x” da própria janela.

**Explicação dos parâmetros de “cv2.namedWindow()”:**

1°parâmetro (‘img\_process’): Nome atribuído a janela.

2°parâmetro (cv2.WINDOW\_NORMAL): Representa se o tamanho da janela é ajustado automaticamente ou ajustável. Como o parâmetro passado é “WINDOW\_NORMAL”, o tamanho da janela se torna ajustável. “WINDOW\_NORMAL” deve ser importado na classe cv2(openCv).

**Explicação dos parâmetros de “cv2.imshow()”:**

1°parâmetro(‘img\_process’): Deve ser colocado entre aspas simples (‘’) o nome da janela (mesmo nome definido na função “cv2.namedWindow()”) que o programa exibirá a imagem.

2°parâmetro(img\_process\_rgb): Imagem que será mostrada na janela, deve ser inserida por meio da variável que armazena a imagem.

**Explicação dos parâmetros de “cv2.resizeWindow()”:**

1°parâmetro(‘img\_process’): Nome da janela que terá o tamanho ajustado, deve ser colocado entre aspas simples (‘’).

2°parâmetro(wResi): Comprimento (em pixeis) que a janela deverá ter, pode ser inserido por meio de uma variável (como foi feito no código) ou colocar um valor diretamente. Exemplo:cv2.resizeWindow(‘img\_process’, **1000**...).

3°parâmetro(hResi): Altura (em pixeis) que a janela deverá ter, pode ser inserido por meio de uma variável (como foi feito no código) ou colocar um valor diretamente. Exemplo:cv2.resizeWindow(‘img\_process’, wResi**, 1000**).

**calculo.py**

Este módulo está responsável por realizar cálculos que serão necessários para se extrair o fator de delaminação(*F*) das imagens.

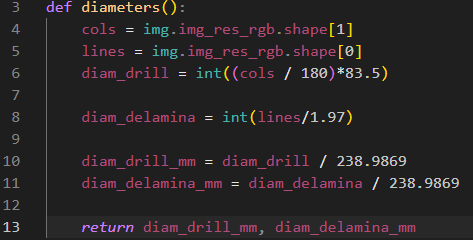
**Importações:**

****

image: Importação do módulo “image” (image.py).

as img: Forma como o módulo será chamado para acessar seus métodos.

**Função “diameters()”:**



Essa função está responsável por calcular os diâmetros dos círculos que foram desenhados para demarcar a circunferência do furo e do alcance máximo da delaminação.

“cols”: Armazena o comprimento da imagem inserida pelo usuário.

“lines”: Armazena a altura da imagem inserida pelo usuário.

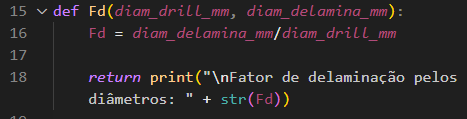
“diam\_drill”: Calcula o diâmetro do furo presente na imagem (é o mesmo valor de “diam\_drill” presente em “image.py”).

“diam\_delamina”: Calcula o diâmetro do alcance máximo presente na imagem (é o mesmo valor de “diam\_delamina” presente em “image.py”).

“diam\_drill\_mm” e “diam\_delamina\_mm”: Conversão dos diâmetros do furo e do alcance máximo da delaminação (respectivamente) de pixels para milímetros (pois é a escala determinada com base nas imagens que foram utilizadas durante o desenvolvimento).

“return diam\_drill\_mm, diam\_delamina\_mm”: Retorna os diâmetros convertidos para sejam utilizados em outros métodos.

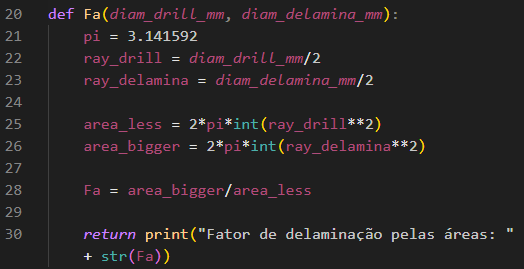
**Função “Fd(diam\_drill\_mm, diam\_delamina\_mm)”:**

****

A função acima calcula o fator de delaminação comum (*Fd*) que é extraído a partir da divisão do diâmetro de maior alcance da delaminação pelo diâmetro do furo.

“Fd”: Variável que calcula o fator de delaminação.

“return print(...)”: Retorna um texto informando o fator de delaminação comum (a partir dos diâmetros).

**Função “Fa(diam\_drill\_mm, diam\_delamina\_mm)”:**

Calcula o fator de delaminação por meio da razão entre a área de maior alcance da delaminação pela área do furo.

“pi”: Armazena o valor de pi para calcular a área das circunferências.

“ray\_drill”: Raio do círculo do furo.

“ray\_delamina”: Raio do círculo que engloba toda a delaminação.

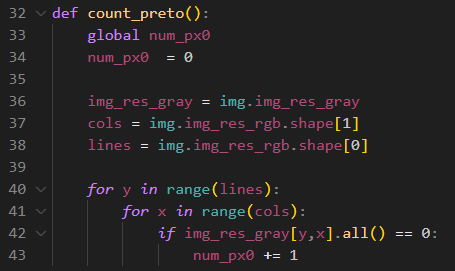
“area\_less”: Calcula a área do círculo do furo.

“area\_bigger”: Calcula a área que engloba toda a delaminação.

“Fa”: Calcula o fator de delaminação utilizando a área do círculo do furo e o que engloba toda a delaminação.

“return print(...)”: Retorna um texto que exibe o fator de delaminação pelas áreas.

**Função “count\_preto()”:**



Essa função realiza a contagem de pixels pretos para se calcular a delaminação a partir da quantidade de pixels. Porém ela só deve ser usada caso a delaminação esteja demarcada, representada por pixels pretos.

“num\_px0”: Variável que armazenará a quantidade de pixels pretos que constituem a delaminação.

“img\_res\_gray”: Importação da imagem com todos pixels em uma escala de tom de cinza.

“cols”: Obter o comprimento da imagem.

“lines”: Obter a altura da imagem.

“range()”: Retorna a quantidade de itens presentes em uma matriz, independentemente de ser unidimensional ou não, por isso foi utilizado

“for y in range(lines)”: A variável “y” percorrerá pela matriz “lines”. “lines” foi utilizada como base por possuir mais itens que “cols”.

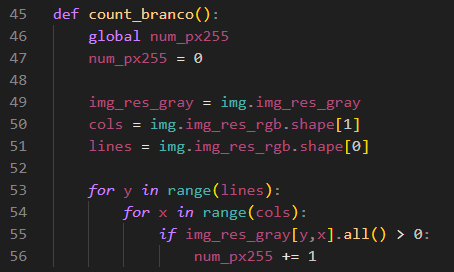
“for x in range(cols): A variável x corresponderá a todos pixels da imagem (um por vez), será por meio dela que será feito a verificação da tonalidade de cinza do pixel (valor = 0, pixel preto).

“if img\_res\_gray[y,x].all() == 0”: Verifica os valores do pixel da imagem. Leitura prática: “Se o pixel da posição[y,x] for = 0 (preto), acrescente mais um na variável que armazena a quantidade de pixels pretos (num\_px0)”.

“img\_res\_gray”: é utilizado por conseguir destacar melhor os pixels, devido a escala de cinza.

“img\_res\_gray[y,x]”: Corresponde a posição do pixel que será verificado.

“all()”: Leva em consideração todos os valores de cada item da matriz, embora a imagem esteja em uma escala de cinza, seus pixels ainda são constituídos por 3 valores, por isso, os 3 valores precisam ser analisados simultaneamente, precisam atender a condição.

**Função “count\_branco()”:**

Essa função realiza a contagem de pixels brancos para se calcular a delaminação a partir da quantidade de pixels. Porém ela só deve ser usada caso a delaminação esteja demarcada, representada por pixels brancos.

“num\_px255”: Variável que armazenará a quantidade de pixels brancos que constituem a delaminação.

“img\_res\_gray”: Importação da imagem com todos pixels em uma escala de tom de cinza.

“cols”: Obter o comprimento da imagem.

“lines”: Obter a altura da imagem.

“range()”: Retorna a quantidade de itens presentes em uma matriz, independentemente de ser unidimensional ou não, por isso foi utilizado

“for y in range(lines)”: A variável “y” percorrerá pela matriz “lines”. “lines” foi utilizada como base por possuir mais itens que “cols”.

“for x in range(cols): A variável x corresponderá a todos pixels da imagem (um por vez), será por meio dela que será feito a verificação da tonalidade de cinza do pixel (valor = 255, pixel != preto).

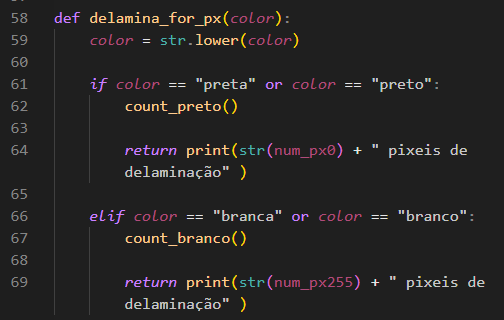
“if img\_res\_gray[y,x].all() == 0”: Verifica os valores do pixel da imagem. Leitura prática: “Se o pixel da posição[y,x] for = 255 (branco), acrescente mais um na variável que armazena a quantidade de pixels brancos (num\_px255)”.

“img\_res\_gray” é utilizado por conseguir destacar melhor os pixels, devido a escala de cinza.

“img\_res\_gray[y,x]”: Corresponde a posição do pixel que será verificado.

“all()”: Leva em consideração todos os valores de cada item da matriz, embora a imagem esteja em uma escala de cinza, seus pixels ainda são constituídos por 3 valores, por isso, os 3 valores precisam ser analisados simultaneamente, precisam atender a condição.

**Função “delamina\_for\_px(color)”:**

****

Realiza o cálculo da delaminação por meio da contagem dos pixels que constituem a delaminação.

“color(parâmetro)”: Parâmetro passado que possuirá seu valor inserido pelo usuário. Corresponde a cor que a delaminação está marcada.

“color(variável)”: Recebe o valor passado pelo parâmetro da função (um texto) e deixa-o todo em minúsculo.

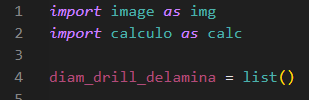
1° “if”: Se a delaminação estiver representada por pixels pretos, é chamado a função “count\_preto()” para se obter a quantidade de pixels pretos (num\_px0). “return print(...)” exibe por meio de um texto a quantidade de pixels da delaminação.

2° “if”: Se a delaminação estiver representada por pixels brancos, é chamado a função “count\_branco()” para se obter a quantidade de pixels brancos (num\_px255). “return print(...)” exibe por meio de um texto a quantidade de pixels da delaminação.

**run.py**

“run.py” é responsável pela “interação” com o usuário, ele fará as chamadas dos métodos necessários para realizar os cálculos e passar o resultado para o usuário.

**Importações**

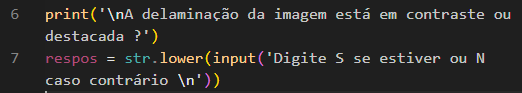
****

“*import* image *as* img”: Importa o módulo “image” (image.py) e o torna possível ser chamado por apenas “img”.

“*import* calculo *as* calc”: Importa o módulo “calculo” (calculo.py) e o torna possível ser chamado por apenas “calc”.

“diam\_drill\_delamina”: Variável do tipo lista que armazenará os valores de delaminação.

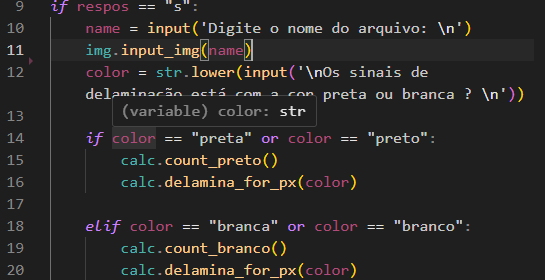
**1° “print(...)”**



Responsável por decidir se será necessário a utilização de algum método do módulo “image”, que é utilizado para manipulação da imagem.

“respos”: Variável que armazena (em minúsculo – str.lower()) a resposta do usuário com relação a delaminação da imagem.

**Verificação da resposta do usuário (parte 1):**



Identificará os métodos que deverão ser chamados caso a delaminação da imagem do usuário esteja contrastada ou destacada.

“name”: Variável que armazena o nome do arquivo que contém a imagem.

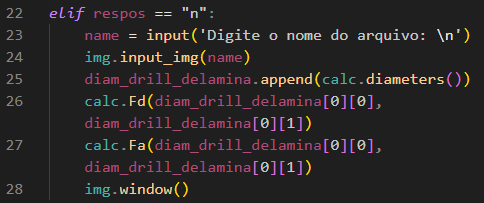
“img.input\_img(name)”: Passa para a função “input\_img()” como parâmetro a variável “name”.

“color”: Variável que receberá a cor pela qual a delaminação está em destaque.

*“if* color == "preta" or color == "preto"”: Verifica se a cor da delaminação está destacada em preto. Caso esteja, chamará a função responsável por contar a quantidade de pixels pretos (calc.count\_preto()) e a que determina a delaminação por pixels (calc.delamina\_for\_px(color) – Color é passado como parâmetro para função).

*“elif* color == "branca" or color == "branco"”: Verifica se a cor da delaminação está destacada em branco. Caso esteja, chamará a função responsável por contar a quantidade de pixels brancos (calc.count\_branco()) e a que determina a delaminação por pixels (calc.delamina\_for\_px(color) – Color é passado como parâmetro para função).

**Verificação da resposta do usuário (parte 2):**

****

Responsável por chamar as funções responsáveis pela manipulação da imagem para a análise da delaminação.

“name”: Variável que armazena o nome do arquivo que contém a imagem.

“img.input\_img(name)”: Passa para a função “input\_img()” como parâmetro a variável “name”.

“diam\_drill\_delamina.append(calc.diameters())”: Adiciona na variável “diam\_drill\_delamina” por meio do método “append()”, que recebe como parâmetro “calc.diameters” o valor do diâmetro do furo convertido para milímetros.

“calc.Fd(diam\_drill\_delamina[0][0], diam\_drill\_delamina[0][1])”: Chama a função responsável pelo cálculo da delaminação por meio dos diâmetros (calc.Fd()) que recebe como parâmetro os valores de obtidos na conversão dos diâmetros de pixels para milímetros.

“calc.Fa(diam\_drill\_delamina[0][0], diam\_drill\_delamina[0][1])”: Chama a função responsável pelo cálculo da delaminação por meio das áreas (calc.Fa()) que recebe como parâmetro os valores de obtidos na conversão dos diâmetros de pixels para milímetros.

“img.window()”: Exibi a imagem com os círculos desenhados. Sua função é mais visual, para que o usuário saiba se será necessário manipular o tamanho do diâmetro em pixels, para que dai sim os círculos contornem o furo e inclua todos sinais de delaminação.