Conjecturas

DOI: 10.53660/CONJ-835-F21

Desenvolvimento de um sistema para a identificação de furos do gabinete traseiro de televisores para inserção automática de parafusos

Development of a system for identifying screews holes in the rear cover of televisions for process automation

Jozias Parente de Oliveira[1]*, Matheus Libório Lima de Andrade[1], Israel Gondres Torné[1], Fábio de Souza Cardoso[1], Angilberto Muniz Ferreira Sobrinho[1], André Luiz Printes[1].

RESUMO

O presente trabalho teve como objetivo a utilização e validação de um método de comparação de imagens chamado casamento por correlação. Esse método utiliza o coeficiente de correlação de Pearson para medir a correlação entre os pixels presentes em um *template* (objeto alvo) e os pixels de outra imagem. Dependendo do valor da correlação, é possível verificar se um determinado objeto está presente em uma imagem ou não, bem como a localização desse objeto. Para a proposta desse trabalho, utilizou-se esse coeficiente para localizar as coordenadas de furos de parafusos em gabinetes traseiros de televisores a fim de se automatizar o processo de inserção e aperto dos parafusos. Todo o algoritmo foi desenvolvido no MATLAB e foram realizados testes que avaliaram seu comportamento através da análise da eficiência em imagens onde o posicionamento do televisor, brilho e contraste foram intencionalmente variados para se produzir mais de 2500 condições diferentes. Ao final do projeto, foi possível verificar que a eficiência do algoritmo se manteve em 100% para uma grande faixa de brilho e contraste, o que valida e comprova a robustez que a técnica confere ao processo produtivo.

Palavras-chave: Coeficiente de Correlação Pearson; Processamento Digital de Imagem; PDI; Casamento por Correlação; Identificação de furos.

ABSTRACT

The aim of the present work is to use and validate a method of comparing images known as image matching. This method uses the Pearson correlation coefficient to measure the correlation between pixels present in a template (target object) and pixels in another image. Depending on correlation value, it is possible to check whether a certain object is present in an image or not, as well as the location of that object. In this work, this coefficient was used to locate coordinates of screw holes in televisions rear cabinets in order to automate the insertion and tightening process of the screws. The entire algorithm was developed at MATLAB and tests were made to evaluate its behavior by analyzing the efficiency of images where the TV's positioning, brightness and contrast were intentionally varied to produce more than 2500 different conditions. At the end of the project, it was possible to verify that the efficiency of the algorithm remained 100% efficient for a wide range of brightness and contrast, which validates and proves the robustness that the technique gives to the process productive.

Keywords: Pearson Correlation Coefficient; Digital Image Processing; DIP; Image Matching; Template Matching; Holes Identification.

[1] Universidade do Estado do Amazonas.
*E-mail: jpoliveira@uea.edu.br

INTRODUÇÃO

A sociedade vem observando há décadas um crescente avanço tecnológico em diversas frentes da sociedade. Grande parte dessas tecnologias impactam de forma positiva a vida das pessoas em geral. Um grande campo em específico que vem crescendo de forma substancial é o da visão computacional. Essa área vem se desenvolvendo de tal forma a prover diversas automações que beneficiam tanto a indústria quanto aos indivíduos. Ao se falar de benefícios da visão computacional para os indivíduos, pode-se citar, por exemplo, os carros com piloto automático da TESLA. Veículos que possuem um robusto sistema de visão computacional que confere ao carro maior segurança na dirigibilidade. A visão computacional parte do princípio de dar visão a uma máquina/computador, nesse contexto, é uma área de fundamental interesse no que diz respeito à automação industrial. (RUDEQ, COELHO, JUNIOR 2005). Além disso, o mercado vem se mostrando cada vez mais desafiador, onde as empresas estão, constantemente, aperfeiçoando seus métodos produtivos e se tornando, consequentemente, mais competitivas. Nesse contexto se faz necessário, para garantir a sobrevivência das empresas no mercado atual, a automação de processos em que são observados grande impacto das limitações humanas, tais como a fadiga, a baixa eficiência, a questão ergonômica etc. (SCHEIFLER, FAIZ, LUDWIG, 2016)

Diante disso, o presente trabalho apresenta a implementação e validação de um sistema de visão computacional em MATrix LABoratory (MATLAB) a fim de promover a automação de um processo visando a identificação com exatidão dos pontos de atuação de uma parafusadeira robótica para aplicação no processo de montagem da tampa traseira de televisores em uma linha de produção de uma empresa do polo industrial de Manaus, onde certo processo exige a inserção e o aperto de parafusos de gabinetes de televisores.

Analisando-se o processo, atualmente realizado por mão de obra humana, nota-se pontos críticos, como por exemplo, a fadiga do colaborador, lesões por movimento repetitivo, eficiência, entre outros. Dessa maneira, o trabalho consistiu no desenvolvimento de um sistema computacional que realiza a comparação de imagens utilizando o coeficiente de correlação de Pearson com um banco de imagens, adquirido da linha de produção e localizada as coordenadas dos pontos que devem ser inseridos os parafusos para fixação da tampa traseira dos televisores.

2 REFERENCIAL TEÓRICO

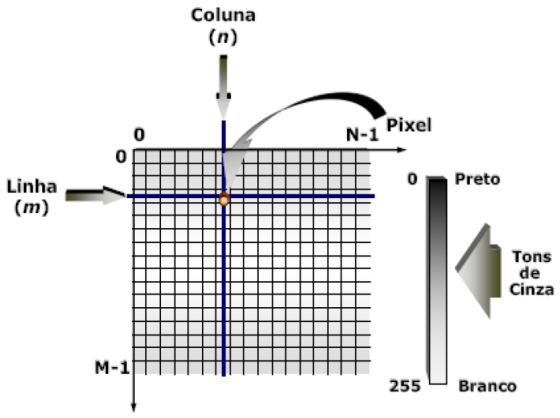
2.1 AQUISIÇÕES DE IMAGEM

2.1.1 Representação digital de uma imagem

De acordo com Jähne (2002), uma imagem monocromática é uma função bidimensional contínua f(x,y) onde as variáveis x e y são coordenadas espaciais e o valor de f é proporcional à intensidade luminosa no ponto considerado.

Cada ponto na grade bidimensional que representa a imagem digital é denominado pixel. Na figura 1, apresenta-se a notação matricial usual para a localização de um pixel no arranjo. Nesta figura a imagem foi quantizada com 8 bits, portanto, os pontos mais escuros da imagem (preto) são representados por 0 e os pontos mais claros (brancos) por 255. No intervalo entre estes valores estão as gradações de cinza.

Figura 1 - Representação de uma imagem digital bidimensional.



Fonte: Jähne, 2002.

2.1.2 Câmeras Digitais

De acordo com Tommaselli, Hasegawa e Galo (2018) um sistema básico de coleta (ou aquisição) é sempre composto por um conjunto de lentes, um material fotossensível e um sistema de gravação final da imagem. As câmeras digitais são dispositivos para a coleta e armazenamento de imagens digitais. Uma câmera possui um conjunto de lentes,

um sensor para captar luz, processadores e uma memória para o armazenamento das imagens. Um dos principais componentes das câmeras digitais é o sensor. O sensor usado pela maioria das câmeras digitais é o CDD, (*charge coupled device*). Entretanto, há também a tecnologia CMOS (*Complementary metal oxide semicondutor*). Ambos os sensores transformam a luz em elétrons. Uma maneira simplificada de pensar a respeito destes sensores é imaginar uma matriz bidimensional de milhares ou mesmo milhões de minúsculas células solares. Ao finalizar o processo de conversão de luz em elétrons, o sensor lê a carga acumulada de cada célula na imagem. É nesse ponto que estão as diferenças entre os dois principais tipos de sensores: O CCD transporta a carga através do chip e lê em um canto da matriz. Um conversor analógico digital transforma o valor de cada pixel em um valor digital por meio da medição da quantidade de carga de cada diodo fotossensível e converte essa medição para a forma binária; já os dispositivos CMOS usam diversos transistores em cada pixel para amplificar e mover a carga usando fios tradicionais. O sinal de CMOS é digital, assim ele não necessita do conversor A/D.

2.2 PROCESSAMENTO DIGITAL DE IMAGEM

O processamento de sinais consiste na análise e/ou modificação de sinais utilizando teoria fundamental, aplicações e algoritmos, a fim de extrair informações dos mesmos e/ou torná-los mais apropriados para alguma aplicação. Esses sinais podem ser grandezas físicas que variam no tempo, espaço ou em função de qualquer variável. Para o estudo em questão, o sinal trata-se de uma imagem. Portanto, processamento digital de imagem refere-se ao campo que analisa e/ou modifica imagens a fim de extrair informações ou torná-los mais apropriados para alguma função.

De acordo com Esquef, Albuquerque e Albuquerque (2003) as etapas que compõem um sistema de processamento digital de imagem são: aquisição da imagem, digitalização, pré-processamento, segmentação, pós-processamento, extração de atributos, classificação e reconhecimento.

A etapa de aquisição é responsável pela captura da imagem por meio de um dispositivo ou sensor e pela sua conversão em uma representação adequada para o processamento digital subsequente. Os principais dispositivos para aquisição de imagens são: câmeras de vídeo, tomógrafos médicos, satélites e scanners. A imagem digital resultante de processo de aquisição pode apresentar imperfeições ou degradações

decorrentes, por exemplo, das condições de iluminação ou características dos dispositivos. A etapa de pré-processamento visa melhorar a qualidade da imagem por meio de aplicações de técnicas para atenuação de ruídos. Uma das técnicas que podem ser citadas é a técnica chamada de “histograma de luminância”.

Já a etapa de segmentação, de acordo com Esquef, Albuquerque e Albuquerque (2003), é a etapa responsável por definir as regiões de interesse para processamento e análise posteriores. Essa etapa é muito crítica, pois se a segmentação das regiões de interesse não for feita de maneira adequada, pode-se ter ao final do processamento resultados não desejados. Finalizada a etapa de segmentação, começa a etapa de pós-processamento, cujo objetivo é corrigir pequenas imperfeições geradas na etapa de segmentação.

Na etapa de extração de atributos são realizadas medidas na imagem pós-processada. Através dessas medidas, os grupos de pixels são descritos por atributos característicos, gerando dados quantitativos para o objetivo final. Por fim, tem-se a etapa de Classificação e Reconhecimento, cujo objetivo é realizar de forma automática a identificação dos objetos segmentados na imagem. Para essa identificação, são necessárias duas etapas: o aprendizado e o reconhecimento propriamente dito.

2.3 COEFICIENTE DE CORRELAÇÃO PEARSON

O coeficiente de correlação de Pearson é um teste que mede a relação estatística entre duas variáveis contínuas. Garson (2009) afirma que a correlação é uma medida de associação bivariada (força) do grau de relacionamento entre duas variáveis. Para Moore e McCabe (2007) a correção mensura a direção e o grau da relação linear entre duas variáveis quantitativas. Dessa forma, o coeficiente de correlação Pearson mede a associação linear entre variáveis.

O coeficiente de correlação (r) entre duas variáveis (x) e (y) é dado por:

$$r = \frac{1}{n-1}\sum \frac{(X_i - \underline{X})}{S_x}\frac{(Y_i - \underline{Y})}{S_y} \quad (1)$$

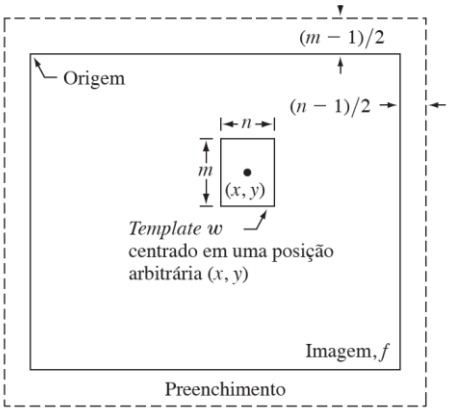
O coeficiente de correlação de Pearson pode ter um intervalo de valores de +1 a -1. Um valor de 0 indica que não há associação entre as duas variáveis. Um valor maior que 0 indica uma associação positiva. Isto é, à medida que o valor de uma variável aumenta, o mesmo acontece com o valor da outra variável. Um valor menor que 0 indica

uma associação negativa, ou seja, à medida que o valor de uma variável aumenta, o valor da outra diminui.

2.3.1 Correlação de Pearson em processamento de imagens

Utilizando o coeficiente de correlação de Pearson em imagens (Figura 2), será possível fazer o casamento por correlação. Esse casamento consiste em fornecer ao programa um template (que seria o alvo) e fazer com que a imagem de entrada seja varrida pelo algoritmo. Em outras palavras, durante a varredura será feito o cálculo de correlação entre os pixels do template (cuja dimensão é de m x n) e os pixels da imagem de entrada delimitados pela mesma área do template.

Figura 2 - Coeficiente de correlação de pearson em imagens.

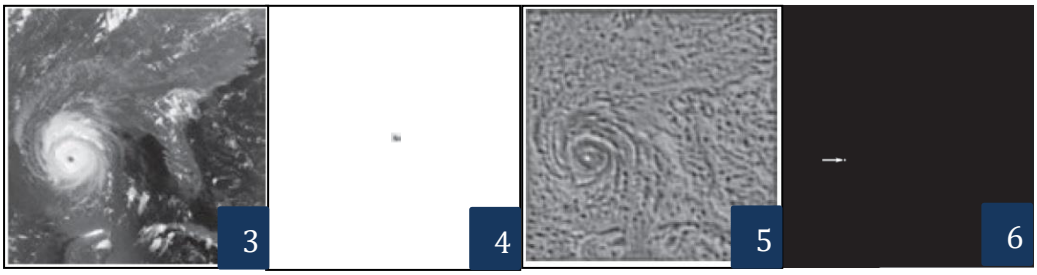


Fonte: Gonzalez; Woods, 1996.

Para exemplificar, observe a figura 3 que mostra a imagem captada por satélite do furacão Andrew, obtida em 24 de agosto de 1992. A partir dessa imagem, é possível fazer um recorte do olho do furacão. Esse recorte está sendo mostrado na figura 4 e será o nosso template para realizar a posterior varredura da imagem. A seguir, na figura 5, está sendo mostrado o resultado da correlação em formato de imagem.

Na figura 5, a intensidade dos pixels é proporcional ao valor da correlação, de modo que quanto maior a correlação num pixel, mais brilhante ele vai ser. Por fim, na figura 6 é mostrado o resultado da figura 5 após passar por um processo de binarização que zerou as intensidades dos pontos onde a correlação é diferente de 1 e intensificou o ponto onde a correlação é forte.

Figura 3 a 6 - Exemplificação da Correlação de Pearson em imagens. **Figura 3** - Imagem capturada por Satélite do furacão Andrew, 24 de agosto de 1992. **Figura 4** - Template (olho do furacão). **Figura 5**- Resultado da Correlação. **Figura 6** - Resultado da binarização da figura 5.



Fonte: Gonzalez; Woods (1996).

2.5 SOFTWARE (MATLAB)

Para o desenvolvimento do software que realiza todo o processamento digital de imagem foi utilizado o MATLAB, um software interativo de alta performance voltado para o cálculo número. Ele integra análise numérica, cálculo com matrizes, processamento de sinais e construção de gráficos. É um programa de computador especializado e otimizado para cálculos científicos e de engenharia (CHAPMAN, 2003).

Dentre as vantagens do MATLAB, pode-se citar a facilidade do uso: o programa pode ser usado como prancheta de rascunho para avaliar expressões digitadas em linha de comando, ou pode ser utilizado para executar programas grandes escritos previamente, além disso, esses programas podem ser facilmente escritos e modificados no ambiente integrado de desenvolvimento e depois depurados por meio do depurador MATLAB; independência da plataforma: o programa tem suporte em diferentes sistemas computacionais; funções predefinidas: MATLAB vem com uma grande biblioteca de funções predefinidas (CHAPMAN, 2003).

3 METODOLOGIA

O Trabalho apresentado foi uma Pesquisa Aplicada, e teve como objetivo a realização de Pesquisa Exploratória sobre o material bibliográfico adquirido sobre o assunto. Foi utilizado o procedimento técnico de pesquisa bibliográfica. O método de abordagem utilizado foi o hipotético-dedutivo e o método de procedimento de elaboração foi o monográfico. Para coleta de dados foi utilizada documentação indireta, com auxílio

de documentos primários e secundários, e a análise e interpretação de seus dados foi quantitativa.

3.1 MATERIAIS

Os materiais utilizados para a realização da pesquisa foram: um notebook modelo Dell G3, com processador intel core i5 8ª geração, com armazenamento de 1TB (HD) e 256 gb (SSD), tendo com sistema operacional o Windows 10 home; uma câmera de celular (Samsung A7 2017, 12 Mp) para aquisição das imagens e uma Smart TV LED 46'' Samsung FULL HD (UN46F8000). Dentre os poucos modelos de tv ao alcance para realização da pesquisa, esse foi o mais apropriado para avaliar a hipótese defendida nessa dissertação devido ao seu gabinete traseiro completamente plano, facilitando uma incidência mais uniforme da luz e, consequentemente, tornando menos complexo o processo da aquisição de imagem.

3.2 PREMISSAS DE PROJETO

Definiu-se que os furos a serem reconhecidos são os destacados na figura 7.

Figura 7 – Furos a serem reconhecidos na Smart TV LED 46''.

Fonte: Autoria própria

Conforme dito no início, o objetivo é que esse sistema seja instalado em uma linha de produção e neste caso é importante a padronização de processos e suas demais

variáveis. Dessa forma, para a simulação, foi considerado que o posicionamento do televisor e da câmera não iria variar de forma considerável.

3.3 AQUISIÇÃO DE IMAGENS

Para a aquisição das imagens, o televisor foi colocado em cima de uma mesa, em ambiente aberto a fim de fazer uso da luz natural. Os parafusos dos furos alvos foram retirados para uma melhor simulação. Com o televisor em cima da mesa, a câmera do celular foi apontada a uma distância da tv tal que permitisse capturar todo o seu gabinete traseiro. Nas condições descritas foram adquiridas 15 imagens diferentes do gabinete traseiro conforme ilustrado na figura 8. Algumas delas têm variações sutis de posicionamento a fim de simular as possíveis variações presentes em uma linha de produção.

Figura 8 - Imagens adquiridas do gabinete traseira da tv.

Fonte: Autoria própria.

Após a aquisição, foi necessário realizar uma análise das imagens com a finalidade de selecionar a que tivesse melhor qualidade quanto a iluminação e ângulo da câmera. A imagem escolhida para extração dos 4 *templates* de furo foi a que se refere ao televisor 9, que pode ser visualizada na figura 8. A escolha da imagem se deu devido a câmera do celular parece estar mais centralizada em relação ao gabinete da tv.

Para a aquisição dos *templates* a imagem foi aberta no programa PAINT e foram feitos recortes dos 4 furos conforme ilustrado na figura 9. Neste ponto, é importante destacar a diferença entre as imagens. Essas diferenças são causadas devido à diferença de angulação vertical entre a câmera do celular e o posicionamento dos furos. Tais variações no *template*, embora sutis ao olho humano, interferem no cálculo do coeficiente de correlação. Outro importante esclarecimento a se fazer é que as figuras referentes aos 4 *templates* foram aumentadas para que fossem observadas as diferenças entre eles, entretanto, no sistema desenvolvido eles são usados em seus tamanhos originais (cerca de 25x25 pixels).

Figura 9 – Templates usados no algoritmo.

Fonte: Autoria própria.

3.4 DESENVOLVIMENTO DO SCRIPT

Neste capítulo será apresentado em detalhes o código desenvolvido. Entretanto, antes de compreender seu funcionamento, é importante destacar que foi explorada uma premissa de projeto a fim de conferir mais eficiência ao código: a de que o televisor não irá ter grandes variações em seu posicionamento. Nesse caso, as coordenadas achadas provavelmente irão estar sempre em uma determinada região. Para fazer uso dessa premissa, foram localizadas as coordenadas exatas dos 4 furos das 15 imagens mostradas na figura 8. Após registrar todas as coordenadas, foi feito o cálculo das médias, conforme demonstrado na tabela a seguir, a tabela 1.

Como esperado, não há tanta variação dos pontos no eixo y, porém no eixo x houve uma maior variação. Com esses valores, é possível criar um “vetor base" contendo, respectivamente, os valores 280, 652, 1044 e 1396. Esses valores referem-se às

coordenadas x dos furos 1, 2, 3 e 4, respectivamente. Sabendo disso, pode-se iniciar o detalhamento do código com a figura 10.

Tabela 1 – Coordenadas dos 4 pontos das 15 imagens.

	x1	y1	x2	y2	x3	y3	x4	y4
tv1	250	774	634	772	1020	772	1357	774
tv2	205	757	550	771	942	785	1327	804
tv3	345	819	713	804	1088	791	1419	782
tv4	283	815	656	822	1059	831	1434	841
tv5	221	811	611	823	1027	836	1412	850
tv6	249	832	638	822	1035	812	1383	805
tv7	277	825	654	828	1045	831	1394	836
tv8	276	814	634	816	1024	819	1392	824
tv9	324	820	674	825	1053	832	1407	840
tv10	304	807	682	809	1077	812	1434	817
tv11	324	837	694	844	1079	853	1423	862
tv12	335	832	696	827	1071	825	1350	813
tv13	239	826	601	828	1013	830	1419	835
tv14	285	859	681	830	1054	805	1360	786
tv15	281	818	667	819	1071	821	1436	824
média	280	816	652	816	1044	817	1396	820

Fonte: Autoria própria.

Na parte inicial do código, foi utilizada a função *imageDatastore* para atribuir à varável "banco_imagens" a pasta banco_furos que contém as imagens dos *templates* (furos). Dando continuidade, a imagem de entrada é lida com a função *imread* e atribuída à variável I.

Figura 10 – Parte inicial do código.

```
clear all;
close all;
banco_imagens = imageDatastore('banco_furos_sem_parafusos');
A1 = imread('televisor13.png');
A = A1(:,:,1);
I1 = A;
I = I1
figure, imshow(I);
title('imagem de entrada');

X_base = [280, 652, 1044, 1396]
Y_base = [817];
```

Fonte: Autoria própria.

Além disso, foram criados dois vetores X_base e Y_base. Esses vetores contêm os valores achados na tabela 1. A função deles é a de fornecer um valor para que,

posteriormente, o programa consiga decidir se a coordenada encontrada está correta ou não. Essa etapa será explicada em outro momento. A seguir, na figura 11, é mostrada parte do código onde são feitos os comandos para localização das coordenadas dos furos.

Figura 11 – Comandos para localização das coordenadas dos furos.

```
13 -  while i<=4
14 -      B1 = readimage(banco_imagens,i);
15 -      B = B1(:,:,1);
16 -      template = B;
17 -      cross = normxcorr2(template, I);
18 -      [ elem, Ind ] = sort(cross(:),1,'descend');
19 -      [Y(i),X(i)] = ind2sub(size(cross), Ind(1));
20
21 -      r=1;
22 -      while r<100
23 -          if ((X(i)>X_base(i)-150) && X(i)<(X_base(i)+150) && Y(i)>720 && Y(i)<920)
24 -              break
25 -          else
26 -  %           X(i) = [];
27 -              Ind(1) = [];
28 -              [Y(i),X(i)] = ind2sub(size(cross), Ind(1));
29 -          end
30 -          r = r+1;
31 -      end
32 -  i = i+1;
33 -  end
```

Fonte: Autoria própria.

Os comandos entre as linhas 14 e 32 estão dentro de um *While*. Para facilitar o entendimento desse loop, basta observar a figura 12.

Figura 12 – Fluxograma do loop principal.

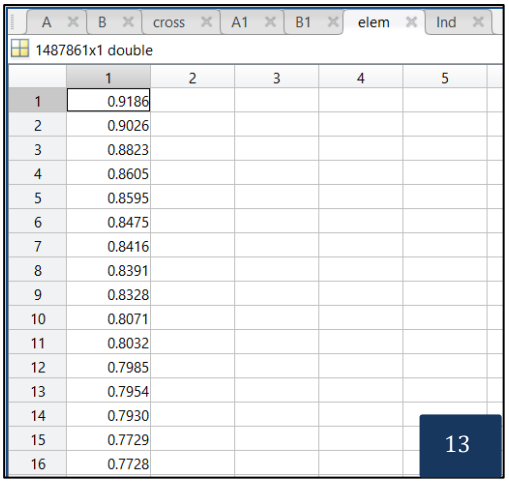
Fonte: Autoria própria.

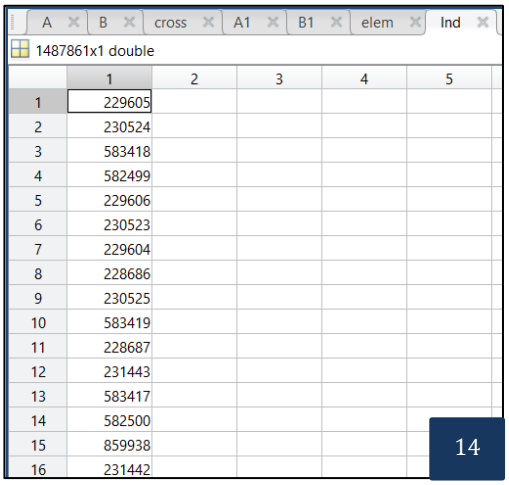
Como o objetivo é encontrar 4 furos diferentes, serão necessárias 4 iterações. Na linha 14, será atribuída à variável B1 a imagem que se encontra na posição "i" da variável banco_imagens. Na linha 17 é realizado o cálculo de coeficiente de correlação de Pearson, através do comando "normxcorr2", entre os pixels da imagem de entrada (I) e a imagem do furo (template). O resultado do cálculo é colocado na matriz cross.

Na linha 18, são criados dois vetores: elem (refere-se a elementos) e Ind (refere-se a índice). Ao vetor "elem" são atribuídos todos os valores de correlação da matriz cross

em ordem decrescente. Ao vetor Ind são atribuídos todos os índices da matriz cross, entretanto, esses índices são ordenados em ordem decrescente do valor de correlação. Para esclarecer o entendimento dessa parte, observe a figura 13 e a figura 14.

Figura 13 – Conteúdo do vetor "Elem". **Figura 14** – Conteúdo do vetor "Ind".





Fonte: Autoria própria.

Tais valores foram obtidos do primeiro loop do *while*. Portanto, referem-se à matriz cross, que correlacionou a imagem de entrada com o furo 1 (primeira iteração, i=1). Conforme explicado anteriormente, ao vetor "elem" foram atribuídos os valores da correlação ordenados em ordem decrescente e no vetor "Ind" foram colocados os índices correspondentes aos valores presentes no vetor "elem", de tal forma que o primeiro elemento do vetor "elem" corresponde ao primeiro elemento do vetor "Ind", assim como o segundo elemento do vetor "elem" corresponde ao segundo elemento do vetor "Ind" e assim sucessivamente. Portanto, caso seja desejado obter a coordenada do ponto onde a correlação foi máxima, basta acessar a posição 1 do vetor "Ind" e através da função ind2sub converter o valor do índice para coordenadas cartesianas.

É justamente esse o procedimento feito na linha 19, onde é atribuído à posição i do vetor Y e à posição i do vetor X valores que juntos compõem as coordenadas do índice Ind(1) na matriz cross. Para o exemplo das figuras 13 e 14 a linha 19 iria alocar na posição 1 do vetor Y e na posição 1 do vetor X os valores que juntos iriam compor a coordenada do índice 229605 (onde a correlação foi de 0,9186) da matriz cross.

Após alocar os valores nos vetores X e Y, é necessário verificar se a coordenada do ponto achado está de fato na região esperada através de um outro loop while (linha22). Para isso, delimitou-se uma região de decisão fazendo o seguinte: para o eixo X, é

esperado que o furo 1 esteja próximo do valor 280, que o furo 2 esteja próximo do valor 652, que o furo 3 esteja próximo do valor 1044 e que o valor 4 esteja próximo do valor 1396. Para delimitar o quão próximo desses pontos a coordenada achada pode estar para ser classificada corretamente, fez-se o seguinte cálculo:

$$valor = \frac{\frac{x2 - x1}{2} + \frac{x3 - x2}{2} + \frac{x4 - x3}{2}}{3} \quad (2)$$

$$valor = \frac{\frac{280 - 652}{2} + \frac{1044 - 652}{2} + \frac{1396 - 1044}{2}}{3}$$

$$valor = 186\ unidades\ de\ pixels$$

Já para o eixo y, conforme dito anteriormente, não é esperado variações consideráveis. Para fazer a delimitação da região nesse segmento, escolheu-se arbitrariamente o valor de 100 unidades de pixels.

Dessa forma, para o furo 1 o valor achado no eixo x deve estar entre 280 mais ou menos 186, ou seja, se o valor achado para a primeira iteração estiver entre 94 e 466, o programa considera que achou o ponto correto e o comando break é executado para que o programa saia imediatamente do *loop* e inicie a próxima iteração do *while* principal, mas caso o valor não esteja dentro desse intervalo, o programa exclui o elemento que está na posição 1 do vetor Ind (linha 28) e atribui novamente aos vetores X e Y valores que juntos vão compor a coordenada do índice referente ao elemento 1 do vetor Ind atualizado.

Em outras palavras, o programa extrai a coordenada do ponto onde a correlação deu máxima e verifica se ele está dentro da região esperada, caso ele não esteja, o programa extrai as coordenadas do segundo ponto onde a correlação deu máxima e verifica se ele está dentro da posição esperada e assim sucessivamente até que o programa ache uma coordenada que obedeça duas condições: seja o elemento presente na primeira posição do vetor Ind e que esteja dentro da região esperada para o furo em questão.

As demais linhas (30 e 32) têm a única função de garantir o incremento dos dois loops. A seguir, na figura 15, são mostrados os comandos para que o sistema desenhe retângulos ao redor dos furos.

Ao final das 4 iterações do *while* principal, os vetores X e Y estarão preenchidos com 4 valores diferentes cada um. Esses valores combinados irão compor as coordenadas dos 4 furos. Para que o programa desenhe retângulos ao redor da coordenada achada, é

necessário realizar um ajuste para que seu tamanho fique adequado, por essa razão os comandos da linha 36 à linha 46. O comando "*figure*, *imshow*(I)" na linha 48 mostra a imagem de entrada. Os 4 últimos comandos utilizam a função "*drawrectangle*" para que sejam desenhados os 4 retângulos mostrando os furos.

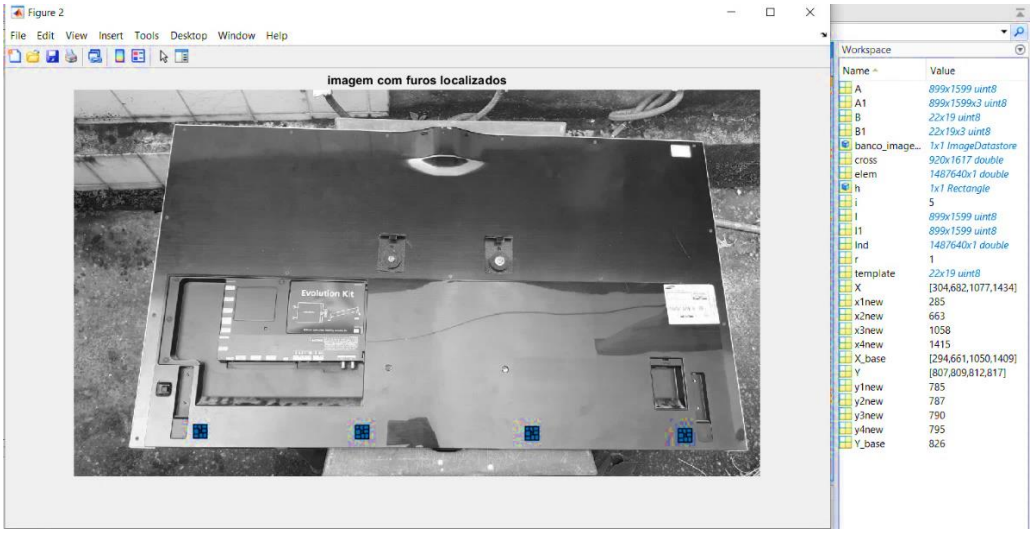
Figura 15 – Comando para desenho do retângulo ao redor dos furos.

```
36 -    y1new = Y(1) - size(template,1);
37 -    x1new = X(1) - size(template,2);
38
39 -    y2new = Y(2) - size(template,1);
40 -    x2new = X(2) - size(template,2);
41
42 -    y3new = Y(3) - size(template,1);
43 -    x3new = X(3) - size(template,2);
44
45 -    y4new = Y(4) - size(template,1);
46 -    x4new = X(4) - size(template,2);
47
48 -    figure, imshow(I);
49
50 -    h = drawrectangle('Position',[x1new,y1new,size(template,2), size(template,2)],'StripeColor','r');
51 -    h = drawrectangle('Position',[x2new,y2new,size(template,2), size(template,2)],'StripeColor','r');
52 -    h = drawrectangle('Position',[x3new,y3new,size(template,2), size(template,2)],'StripeColor','r');
53 -    h = drawrectangle('Position',[x4new,y4new,size(template,2), size(template,2)],'StripeColor','r');
```

Fonte: Autoria própria.

A seguir, na figura 16 é possível ver o resultado da execução do código para a imagem de entrada.

Figura 16 – Exemplo de execução do código.



Fonte: Autoria própria.

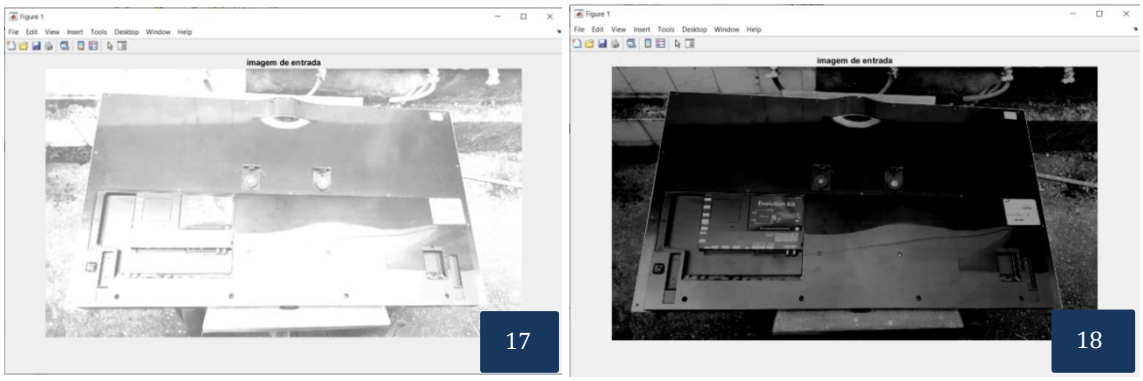
Nessa mesma figura é possível ver ambos os vetores X e Y preenchidos com 4 elementos. Esses elementos combinados formam as coordenadas dos 4 furos destacados.

3.5 TESTES

Para verificar a eficiência do sistema e simular as variações possíveis de acontecer dentro de uma linha de produção, foram realizados testes do programa variando o brilho juntamente com o contraste.

Levando em consideração que o brilho é um ganho na imagem, pode-se variá-lo através do comando da linha 7, onde para aumentá-lo, soma-se uma constante a cada pixel da imagem. Já para diminuir o brilho basta subtrair o valor desejado da imagem. Na figura 17, é mostrada a imagem I após o ganho de 100 e na figura 18, é possível observar a imagem I após o ganho de -100.

Figura 17 a 18 – Teste variando o brilho. **Figura 17** – Exemplo de imagem com brilho em +100. **Figura 18** – Exemplo de imagem de entrada com brilho de -100.



Fonte: Autoria própria.

A fim de conferir mais controle na variação do contraste, foi utilizada uma ferramenta online de edição de fotos chamada Peko-Step (disponível em https://www.peko-step.com/pt/tool/brightness_contrast.html). Utilizando o Peko-Step, variou-se o contraste das imagens de -90, -50, +50, +90. De tal forma que cada valor de contraste fosse aplicado em todas as 15 imagens da figura 8.

Para cada valor de contraste, o experimento foi feito variando-se o brilho de -150 até +200 em intervalos de 10 unidades. Para cada valor de brilho, o programa foi executado 15 vezes diferentes. Em cada execução, mudou-se a imagem de entrada para que até o final o teste fosse realizado em todas as 15 imagens diferentes mostradas na figura 8. Ao final de cada execução, foi registrado se o programa conseguiu localizar corretamente os 4 pontos desejados.

Ao final do experimento, foi calculada a eficiência do programa para cada condição de brilho e contraste, fazendo-se:

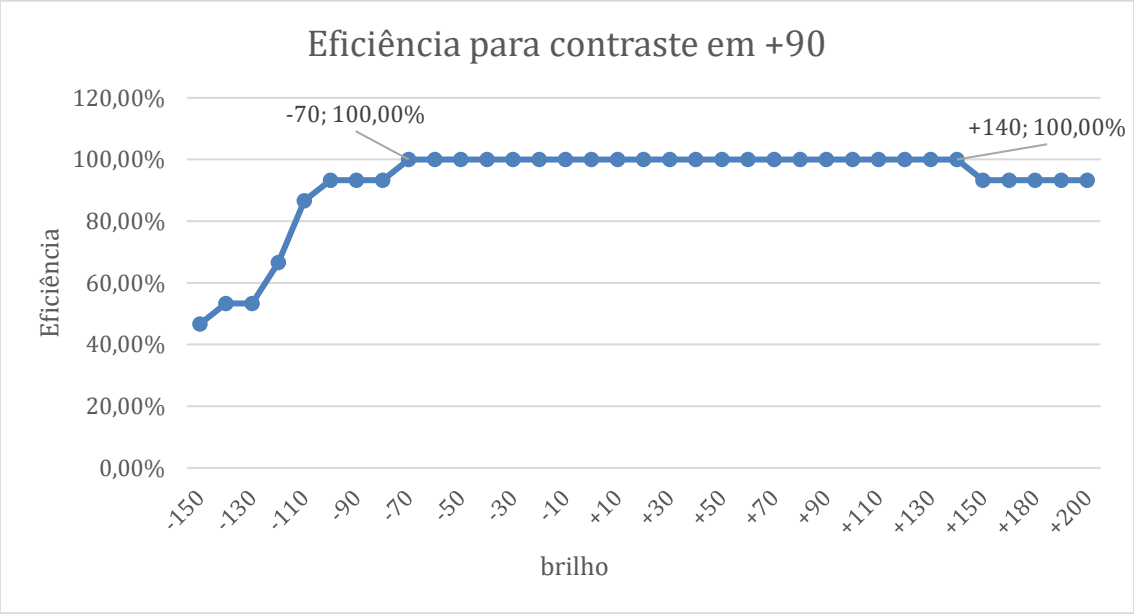
$$Eficiência = \frac{Qtde\_Acertos}{15} * 100\ (\%) \quad (3)$$

Importante destacar que só foi contabilizado como acerto se e somente se o programa conseguiu localizar corretamente todos os furos e que nos testes foram avaliadas mais de 2500 imagens.

4. RESULTADOS

Nesse tópico serão mostradas as curvas de eficiência e brilho para cada valor de contraste. Na figura 19 é ilustrado o gráfico com o desempenho do algoritmo com contraste ajustado em +90 e o brilho variando entre -150 e +200. Neste caso, a taxa de acerto manteve-se em 100% para valores de brilho entre -70 e + 140.

Figura 19 – Eficiência do algoritmo para contraste ajustado em +90.

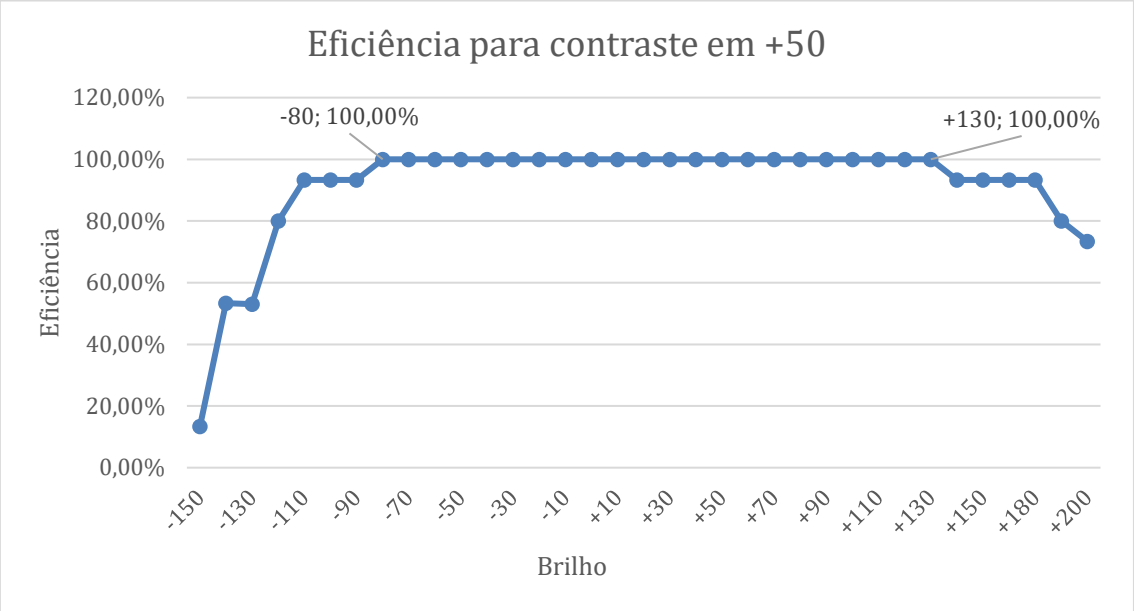


Fonte: Autoria própria.

O gráfico representado na figura 20 apresenta o desempenho do algoritmo com contraste ajustado em +50 e o brilho variando entre -150 e +200. Neste caso, a taxa de acerto manteve-se em 100% para valores de brilho entre -80 e + 130.

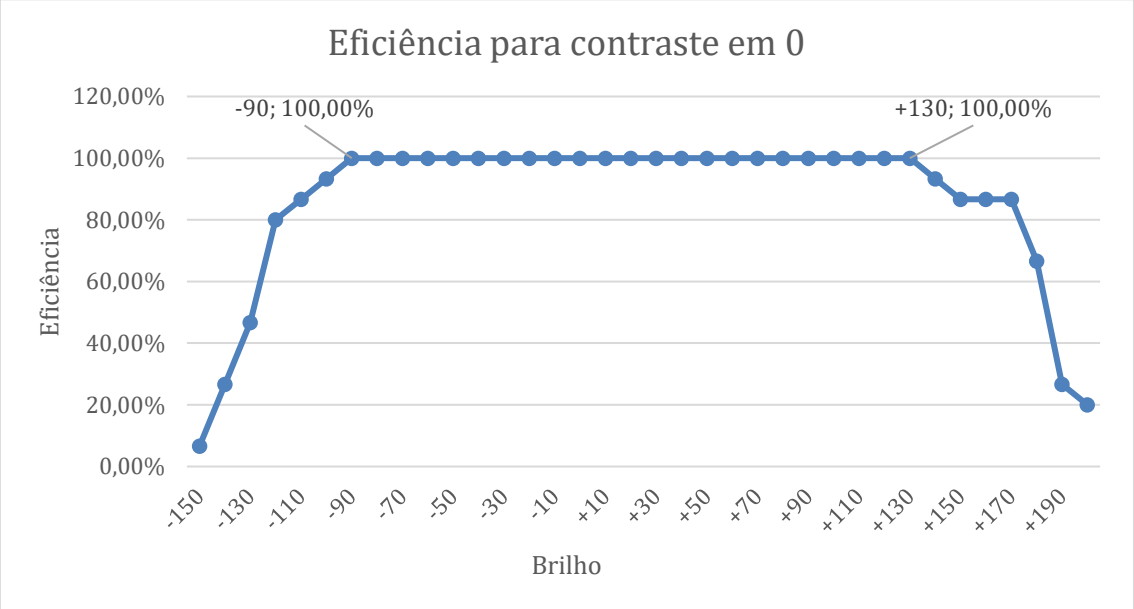
Na figura 21 é ilustrado o gráfico com o desempenho do algoritmo com contraste sem variação e o brilho variando entre -150 e +200. A taxa de acerto manteve-se em 100% para valores de brilho entre -90 e + 130.

Figura 20 – Eficiência do algoritmo para contraste ajustado em +50.



Fonte: Autoria própria.

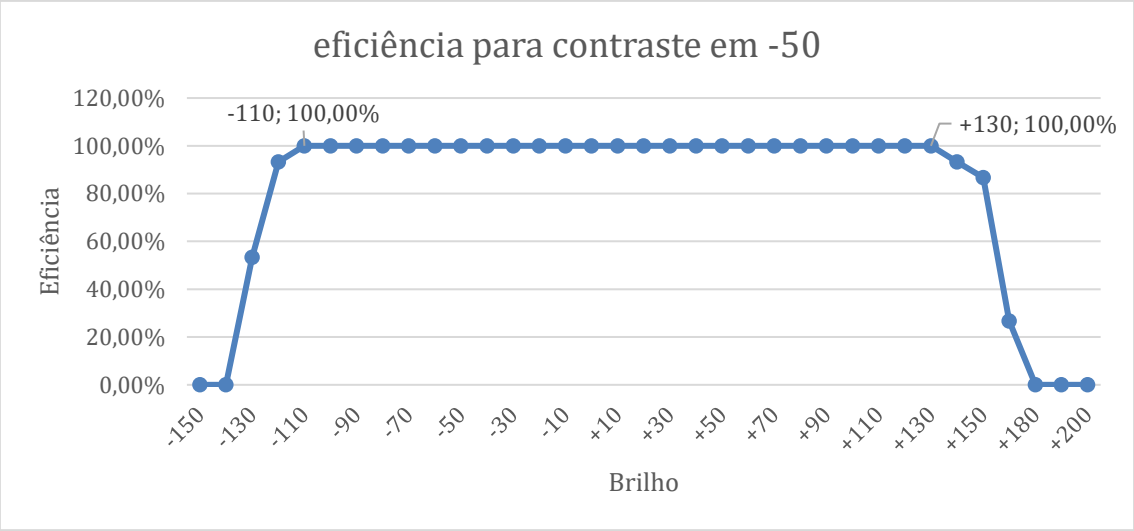
Figura 21 – Eficiência do algoritmo para o caso de contraste sem alteração (+0).



Fonte: Autoria própria.

O gráfico ilustrado na figura 22 mostra o desempenho do algoritmo com contraste ajustado em -50 e o brilho variando entre -150 e +200. Neste caso, a taxa de acerto manteve-se em 100% para valores de brilho entre -110 e + 130.

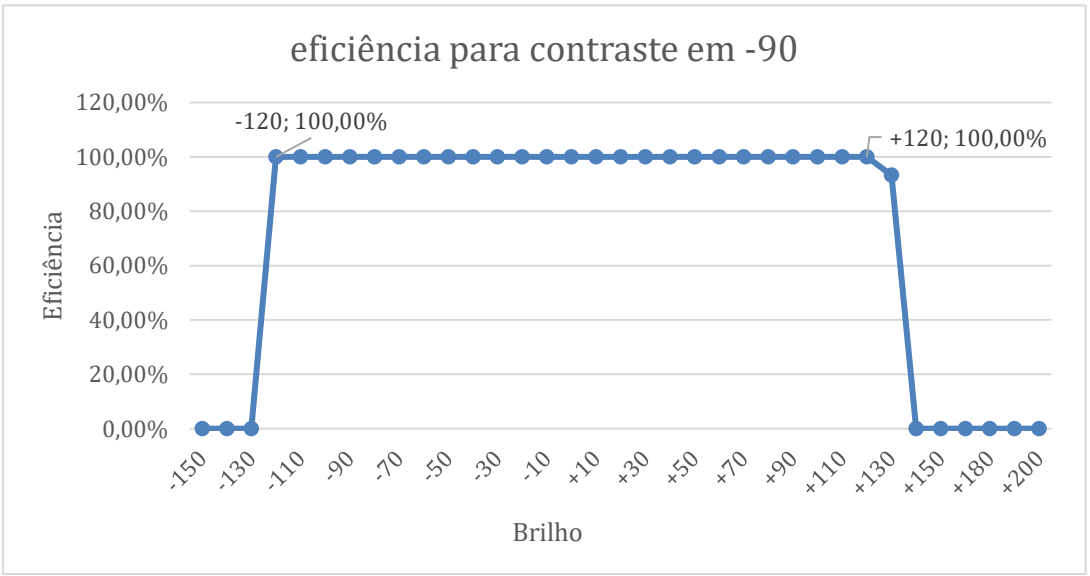
Figura 22 – Eficiência do algoritmo para contraste ajustado em -50.



Fonte: Autoria própria.

Na figura 23 é ilustrado o gráfico com o desempenho do algoritmo com contraste ajustado em -90 e o brilho variando entre -150 e +200. Neste caso, a taxa de acerto manteve-se em 100% para valores de brilho entre -120 e + 120.

Figura 23 – Eficiência do algoritmo para contraste ajustado em -90.



Fonte: Autoria própria.

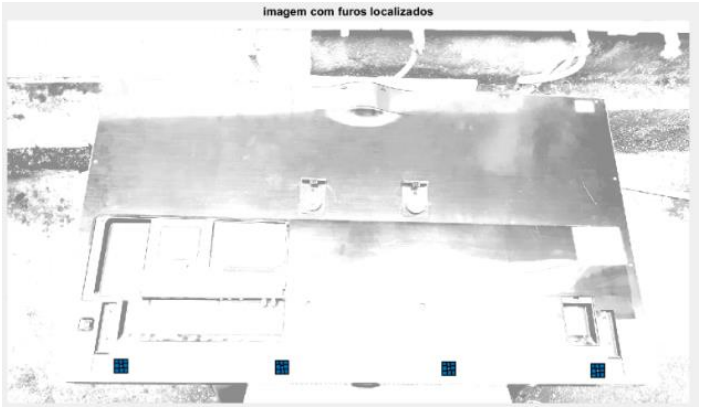
A seguir, serão mostrados alguns casos após o algoritmo fazer a localização dos furos. Na figura 24 é apresentado o desempenho do algoritmo na localização de furos para o brilho ajustado em -70 e contraste em +90. Conforme mostra o gráfico da figura 19, -70 foi o menor valor de brilho que o algoritmo conseguiu manter sua eficiência em 100%, O desafio dessa situação está na região inferior direita do gabinete do televisor, que conforme o brilho vai diminuindo, mais escura ela fica e, consequentemente, torna-se mais difícil diferir os objetos dessa região e achar corretamente o furo.

Figura 24 - Localização dos furos para brilho em -70, contraste +90 (televisor4).

Fonte: Autoria propria.

Já na figura 25 tem-se o desempenho do algoritmo na localização de furos para o brilho ajustado em +140 e contraste em +90. Essa situação refere-se ao último valor de brilho que o algoritmo se manteve eficiente em 100% dos casos para o gráfico da figura 19. A partir desse valor, ou seja, para valores maiores de brilho, o desafio foi identificar corretamente o primeiro furo da esquerda para direita. O motivo para isso foi que a luz refletida pelo gabinete estava mais intensa nessa área, tornando a região mais sensível às variações positivas de brilho.

Figura 25 - Localização dos furos para brilho em +140, contraste +90 (televisor4).



Fonte: Autoria própria.

Na figura 26 pode-se observar o desempenho do algoritmo na localização de furos para o brilho ajustado em +100 e contraste em -50. Na figura 27 é ilustrado o desempenho do algoritmo na localização de furos para o brilho ajustado em -100 e contraste em -90 (gráfico da figura 23).

Figura 26 – Localização dos furos para brilho em +100, contraste -50 (televisor5). **Figura 27** – Localização dos furos para brilho em -100, contraste em -90 (televisor5).

Fonte: Autoria própria.

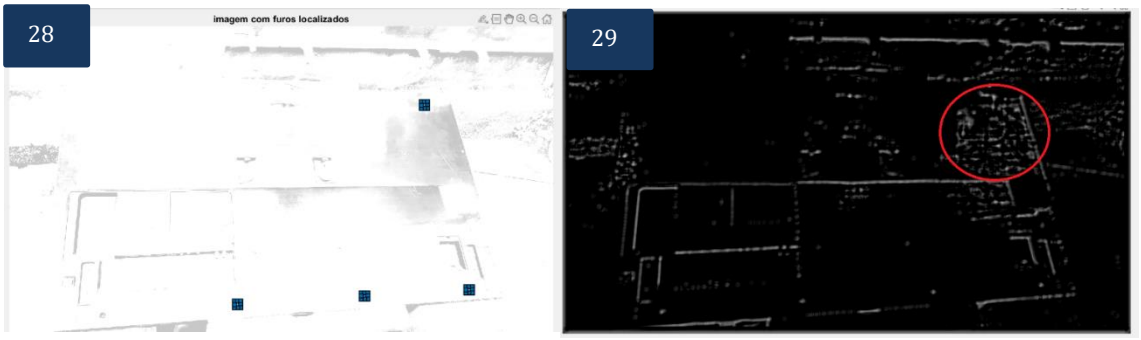
Ao observar todos os gráficos, é possível perceber que conforme o contraste diminui, a faixa de valores de brilho para os quais a eficiência do algoritmo se mantém em 100% vai sendo “deslocada” para a esquerda. A faixa para o valor de contraste em +90 vai do brilho -70 a +140; enquanto para o contraste em +50 a faixa vai do brilho de -80 a +130; para o caso em que não há alteração do contraste, a faixa vai de -90 a + 130; para o contraste em -50, -110 a +130; para contraste de -90, vai de -120 a +120.

Importante também destacar alguns casos em que o algoritmo não conseguiu localizar adequadamente todos os furos, como é ilustrado na figura 28, o desempenho do algoritmo na localização de furos para o brilho ajustado em +200 e contraste em +90.

Para investigar a causa, observe a figura 29, que mostra em formato de imagem o resultado do cálculo da correlação de Pearson entre a imagem de entrada e o template1.

Conforme já explicado anteriormente, os pontos mais claros são pontos onde a correlação foi maior. Nesse caso, muitos ruídos presentes em decorrência da superfície refletora do gabinete acabaram interferindo no resultado final, uma vez que tais ruídos foram acentuados pelo ajuste no contraste de tal maneira que o coeficiente foi maior nesses pontos que no próprio ponto onde se encontra o furo 1. Esse mesmo problema se repetiu em vários casos em que o ajuste do brilho foi positivo.

Figura 28 – Localização dos furos para brilho em +200, contraste em +90 (televisor14). **Figura 29** – Imagem resultante da correlação entre template1 e televisor 14 com brilho em 200, contraste em +90.

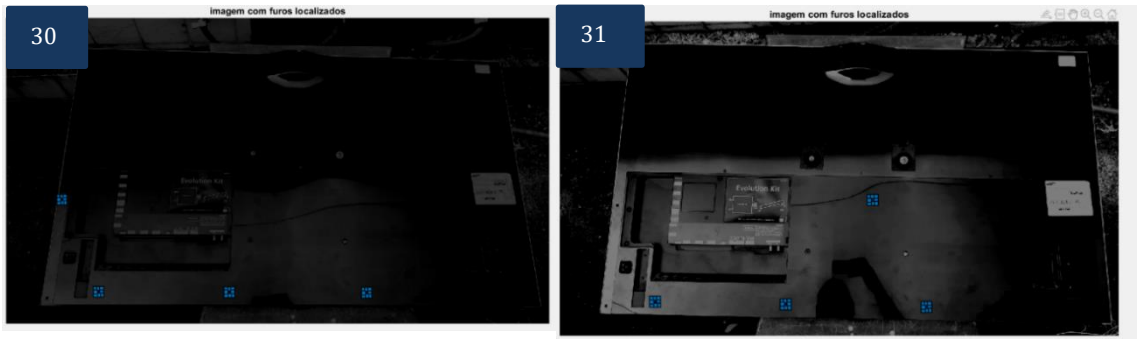


Fonte: Autoria própria.

Outros casos em que o algoritmo não conseguiu localizar corretamente os furos estão sendo mostrados nas figuras 30 e 31. Na figura 30 tem-se o desempenho do algoritmo na localização de furos para o brilho ajustado em -140 e contraste em -50, enquanto que na figura 31 é ilustrado o desempenho do algoritmo na localização de furos para o brilho ajustado em -130 e contraste em +50.

No caso dessas figuras (30 e 31), a combinação entre brilho e contraste fez com que não fosse mais possível diferir os objetos na região inferior direita da imagem.

Figura 30 e 31 – Erros no algoritmo. **Figura 30** - Localização dos furos para brilho em -140, contraste em -50 (televisor15). **Figura 31** – Localização dos furos para brilho em -130 e contraste em +50 (televisor4).



Fonte: Autoria própria.

No geral, conforme os valores de brilho e contraste eram combinados, os pixels das imagens de entrada eram alterados de tal forma que interferissem no resultado final do cálculo de correlação. O que é algo esperado. Ainda assim, notou-se alguns padrões no comportamento da eficiência do algoritmo. Conforme o brilho se intensificava juntamente do contraste, problemas parecidos com o da figura 29 contribuíram para que o coeficiente de correlação Pearson fosse maior em pontos que não eram os furos. Em contrapartida, conforme o brilho era reduzido juntamente com o contraste, a região mais afetada nesses casos foi sempre a região inferior direita das imagens, por essa razão, na maioria das vezes em que o algoritmo não conseguiu localizar corretamente todos os furos, o problema estava na localização do quarto furo, por estar na região mais afetada.

Ainda assim, tais erros eram esperados. Além disso, é importante destacar que em todos as combinações feitas, dentre os 35 valores diferentes de brilho, o algoritmo sempre se manteve 100% eficiente na maioria: para os gráficos das figuras 19, 20 e 21 o algoritmo se manteve 100% eficiente em 22 dos 35 valores de brilho e para os gráficos das figuras 22 e 23 em 25 dos 35;

CONCLUSÃO

Conforme mostrado nos resultados, o algoritmo desenvolvido nesse trabalho conseguiu se manter 100% eficiente em uma faixa extensa de valores de brilho e contraste. Em outras palavras, o algoritmo desenvolvido consegue identificar corretamente os 4 furos alvos mesmo que haja variações entre as imagens, desde que elas não sejam tão grandes.

Ainda assim, é importante ressaltar algumas limitações: conforme descrito na metodologia, ao desenvolver o algoritmo foi considerada a premissa de que não haveria grandes variações no posicionamento do gabinete de TV. Essa premissa foi explorada para melhorar o desempenho do algoritmo, uma vez que ao saber onde provavelmente os furos devem ocorrer, ele descarta as possibilidades de furos longe das regiões esperadas. Entretanto, uma sugestão para trabalhos futuros a fim de permitir maiores variações no posicionamento da tv e ainda melhorar a eficiência do algoritmo seria criar um método robusto de classificação através de um banco de imagens para os templates. Onde a imagem de entrada seria correlacionada com todos os templates do banco e a cada iteração, fossem armazenadas as coordenadas dos 4 maiores pontos de correlação (uma vez que são 4 furos). Ao final de todas as iterações, o algoritmo iria decidir pelas 4 coordenadas que foram mais frequentes.

Outro problema a ser considerado é sobre o aspecto refletor de gabinetes, para aquele usado neste trabalho, esse problema foi solucionado na etapa de aquisição, onde houve preocupação em fazer a captura das imagens em ambiente aberto a fim de se aproveitar a iluminação e preocupação no posicionamento da câmera para não permitir que o gabinete refletisse alguns objetos. Como sugestão, esse problema poderia ser solucionado utilizando mais de uma câmera na etapa de aquisição, dessa forma cada câmera iria apontar para um furo e ficaria mais fácil de diferir os objetos na imagem.

Ainda assim, nesse trabalho, mesmo não sendo exploradas técnicas mais robustas para realizar o pré-processamento da imagem ou para atuar em conjunto com a técnica de correlação de Pearson, o algoritmo aliado com alguns recursos relacionados à programação, conseguiu obter um resultado satisfatório para aplicação na linha de produção.

REFERÊNCIAS

CHAPMAN, Stephen J.: **Programação em MATLAB para engenheiros**. São Paulo, 2003.

DE MACEDO, Sanderson Oliveira. **Desenvolvimento de um sistema de auxílio no diagnóstico de pneumonia na infância utilizando visão computacional** – Universidade Federal de Goiás, 2012.

ESQUEF, Israel Andrade; ALBUQUERQUE, Márcio Portes de; ALBUQUERQUE, Marcelo Portes de; **Processamento de Imagens: Métodos e Análises**. Centro Brasileiro de pesquisas físicas, 2003.

FILHO, Dalson Britto Figueiredo; JÚNIOR, José Alexandre da Silva. **Desvendando os mistérios do Coeficiente de Correlação Pearson (r).** Revista Política hoje, volume 18, número 1, 2009.

FRICK, André Dell'Aglio, **Caracterização de Minério de ferro por Visão**, 2008.

GARSON, G. David. (2009), **Statnotes: Topics in Multivariate Analysis.** Disponível em: https://faculty.chass.ncsu.edu/garson/PA765/statnote.htm.

GONZALEZ, R. e WOODS, R. **Processamento de imagens digitais**. Wilmington, USA: s.n., 1996.

JÄHNE, B. **Digital Image Processing**. Springer-Verlag, 2002.

MOORE, David S. & McCABE, George. **Introduction to the practice of statistics.** New York, Freeman, 2004.

MOORE, David S. The Basic **Practice of Statistics**. New York, Freeman, 2007.

RUDEQ, Marcelo; COELHO, Leandro dos Santos; JUNIOR, Osiris Cancigliere. **Visão computacional aplicada a sistemas produtivos: fundamentos e estudo de caso.** Pontifica Universidade Católica do Paraná, 2005.

SCHEIFLER, Tiago; FAIZ, Ederson B.; LUDWIG, Jean Pierre; DREGER; Ademir Anildo. **Automação como meio para aumento de produtividade e competitividade – Estudo de caso.** Revista Espacios, 2016.

TOMMASELLI, Antonio M. G; HASEGAWA, Júlio K.; GALO, Maurício; **Modernas tecnologias de aquisição de imagens em fotogrametria.** Boletim de ciências Geodésicas, 2018.

Recebido em: 28/02/2022

Aprovado em: 25/03/2022

Publicado em: 30/03/2022