Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.
Digital Object Identifier XX.XXXX/YYY.20XX.ZZZZZZZ

Sistema para Detecção de Defeitos na Fabricação de Tecido Jeans

I. INTRODUÇÃO

O presente documento descreve o desenvolvimento e a conclusão do projeto Sistema para Detecção de Defeitos na Fabricação de Tecido Jeans, que tem como objetivo detectar falhas no padrão de qualidade do tecido jeans utilizado na indústria têxtil. Esse projeto foi desenvolvido como atividade final da disciplina "TI0147 - Fundamentos de Processamento Digital de Imagens" , ofertada pelo Departamento de Engenharia de Computação da Universidade Federal do Ceará (UFC-DETi), turma 2021.2, no qual houve participação de toda a turma e gerenciamento do aluno Paulo Douglas Melo da Silva. No Apêndice A encontra-se uma listagem detalhada de cada equipe, com seus respectivos participantes e atividades desenvolvidas. O objetivo geral do projeto consiste em desenvolver uma aplicação computacional capaz de detectar defeitos visualmente distinguíveis no tecido jeans, que o caracterizem como fora do padrão, tais como: manchas de óleo, buracos na trama, fibras sobrepostas, etc. Fazendo uma simulação que imita uma linha de produção real . O sistema deve ser capaz de detectar automaticamente o defeito com o mínimo de margem de erro possível. O projeto se justifica pela alta aplicabilidade comercial, podendo ser implementado na indústria têxtil, otimizando a linha de produção em tempo e em qualidade, onde substituiria a inspeção feita pelo olho humano, que está eventualmente sujeita ao erro e a fadiga. O formato final do sistema utilizou linguagem Python, biblioteca OpenCV.

A. CONTEXTUALIZAÇÃO

Temos hoje como um dos principais desafios da fabricação de tecido Jeans a detecção de falhas no tecido, sejam essas ocasionadas por manchas, rasgões, furos ou irregularidades no tecido. Essa detecção em estados iniciais é feita por um funcionário, encarregado de visualizar a esteira em busca de alguma falha no tecido corrente. Porem após um grande período de visualização, pode ocorrer da visão do funcionar ficar saturada, e por isso diminuir a percepção de possíveis defeitos e aumentar a desatenção. Isso leva a necessidade da criação de um sistema automatizado para tal detecção.

B. PROPOSTA

Esse trabalho busca então a produção de um sistema que possa detectar defeitos na fabricação de tecido Jeans, com variação da luminosidade, tonalidade do tecido e tipo da falha. Texto da proposta inicial

II. BACKGROUND

Temos que a Aquisição de imagem se trata do processo de conversão de uma cena real tridimensional em uma imagem analógica. O primeiro passo na conversão de uma cena real tridimensional em uma imagem eletrônica é a redução de dimensionalidade. Assumiremos que uma câmera fotográfica, câmera de vídeo ou outro dispositivo converterá a cena 3-D em uma representação 2-D adequada. O dispositivo de aquisição de imagens mais utilizado atualmente é a câmera CCD (Charge Coupled Device). Ela consiste em uma matriz de células semicondutoras fotossensíveis, que atuam como capacitores, armazenando carga elétrica proporcional à energia luminosa incidente. O sinal elétrico produzido é condicionado por circuitos eletrônicos especializados, produzindo à saída um Sinal Composto de Vídeo (SCV) analógico e monocromático. Para a aquisição de imagens coloridas utilizando CCDs é necessário um conjunto de prismas e filtros de cor encarregados de decompor a imagem colorida em suas componentes R, G e B, cada qual capturada por um CCD independente. Os sinais elétricos correspondentes a cada componente são combinados posteriormente conforme o padrão de cor utilizado (NTSC (National Television Standards Committee) ou PAL (Phase Alternating Line), por exemplo).

Quanto ao pré processamento, temos que a limiarização é o processo de segmentação da imagem baseado em diferentes escalas de cinza que compõem os vários objetos da imagem. Esse limiar é determinado de acordo com os objetos da imagem, no caso manchas e rasgos na calça. Com isso, a imagem é dividida em dois grupos, sendo o primeiro o grupo de pixel que ficam abaixo desse limiar, anteriormente determinado, de cinza e os pixel que ficam acima desse limiar.

Com relação aos filtros, foi usado filtros gaussianos (reduzindo os componentes de alta frequência). Esse filtro desfoca a imagem e remove detalhes e ruídos, o kernel gaussiano pesa os Pixels em seu centro com mais força que os limites.

Em seguida é utilizado um filtro Laplaciano, realçando as descontinuidades de intensidade da imagem, isso atenua as regiões com intensidade de variação suave, destacando as descontinuidades de nível de cinza. Sua base é a segunda derivação espacial da imagem. Sendo determinado o operador Laplaciano através da equação.

$$Laplace(f) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

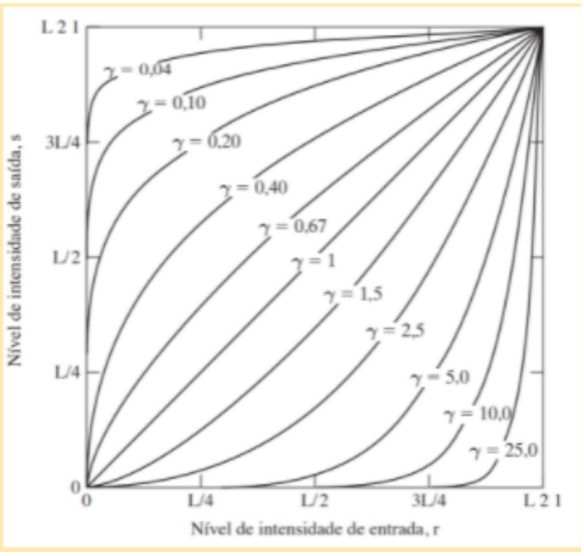
Para detectar as bordas da imagem é usado um Kernel que detecta derivados de segunda ordem dos níveis de intensidade da imagem usando uma passagem. É possível usar um segundo Kernel para detectar arestas com diagonais, dando uma melhor aproximação.

Ademais usasse um filtro de media onde o pixel central da mascara é substituído por uma média dos seus vizinhos, esse numero de vizinhos é determinado pelo tamanho da mascara.

Já no desenvolvimento o método implementado recorreu à técnica de limiarização, técnica em que divide-se os intervalos na escala de cinza em dois valores arbitrados, com o intuito de identificar mais facilmente elementos específicos na imagem. Além disso, foi feito uso da função de Canny para a identificação das bordas das falhas, e posteriormente, a demarcação do contorno dos defeitos. Finalmente, a identificação das falhas é limitada por um valor que leva em conta a dimensão da falha.

Foi utilizado alterações morfológicas, nos ruídos restantes. Erosão, os limites dos defeitos sofrem erosão, a mascara passa pela imagem binarizada e o pixel será considerado 1 ser os pixels na mascara forem 1, caso contrário é 0. Dilatação, que aumenta a região branca do objeto de tal forma que caso pelo menos um pixel na mascara for 1 então será 1 ou 0 caso o contrário.

Para lidar com os diferentes tipos de iluminação, foi utilizado transformação de potência, já que apresentava falhas quando a luminosidade era baixa. A transformação de potência é útil para manipulação de contraste e têm a forma básica s = cr, onde s é o resultado; c e γ são constantes positivas e r é a intensidade original do pixel. Basicamente, para valores de γ menores que 1, ocorre expansão dos níveis de intensidade e o contrário ocorre para $\gamma > 1$. A transformação de potência faz uma expansão dos níveis de intensidade, deixando a imagem mais clara.



III. METODOLOGIA

Descrição de como o conteudo do background foi mesclado e utilizado para produção do projeto PRÉ-PROCESSAMENTO A principal adversidade deu-se ao tentar destacar a mancha que aparecia nos frames utilizados para os testes. Como era mais discreta sendo, por vezes, difícil de enxergar a olho nu, acabava sendo indetectável nas operações que tinham como o objetivo separar de forma satisfatória os defeitos do tecido. Enquanto isso, o furo era facilmente percebido em quase todos os testes. Combinando diferentes tipos de limiarizações e filtragens, chega-se a uma solução que permite destacar os dois tipos de defeitos nos dois tecidos.

- deepcopy: Realiza a cópia da imagem
- blur: Realiza a homogeneização da imagem com um filtro de média 13 x 13.
- threshold: Aplica o método da limiarização na imagem. Obs: Foram utilizados o valor 20 como limite para a limiarização, além de o valor máximo ser 255, como estamos lidando com imagens 8 bits.
- A imagem de entrada é convertida para escala de cinza:

```
cv2.cvtColor(img,
cv2.COLOR_BGR2GRAY)
```

- Um filtro gaussiano é aplicado. cv2.GaussianBlur(img, (3,3), 0) A imagem é borrada utilizando um filtro gaussiano, com uma máscara de tamanho especificado (3,3). O filtro gaussiano é um filtro passa-baixa que reduz os componentes de alta frequência.
- Um filtro laplaciano é aplicado.

```
cv2.Laplacian(img, cv2.CV_16S, ksize=3)
```

  Esse filtro realça as descontinuidades de intensidade da imagem, atenuando as regiões com intensidade de variação suave. É útil para destacar linhas de bordas.
- Os valores da imagem são convertidos novamente para uint8: valores inteiros entre 0 e 255.

cv2.convertScaleAbs(img) Isso é necessário porque o resultado do filtro laplaciano contém números tanto positivos quanto negativos.

- O histograma do resultado anterior é equalizado. cv2.equalizeHist(img) Realizado com o objetivo de aumentar o contraste do resultado anterior. É feito um "espalhamento" do histograma da imagem de entrada dessa função.
- Um filtro de média é aplicado. cv2.blur(imgeq, (5,5)) O valor do pixel central da máscara é substituído por um média dos valores de seus vizinhos. O número de vizinhos depende do tamanho da máscara utilizada.
- É realizada uma limiarização binária invertida.

```
ret,thresh1 = cv2.threshold(imgblur,120,
255,cv2.THRESH_BINARY_INV)
```

Os pixels que são menores ou iguais ao valor do limiar são transformados em branco (255), enquanto que os que ultrapassam, tem sua intensidade alterada para 0 (preto). Esse valor de limiar (120) foi escolhido pois era possível destacar ambos os tipos de defeitos.

- findContours: Como o nome sugere, encontra os contornos na imagem.
- arcLength: Retorna o comprimento do contorno detectado.
- Obs: foi escolhido o limite superior de 260 unidades de comprimento por meio de testes para a dimensão do contorno e evitar a identificação errônea de falhas.
- boundingRect: Gera um retângulo de área mínima que abrange a falha detectada

```
def detect_threshold(img):
    output = deepcopy(img)
    jeans = cv2.blur(output, (13,13))
    _, jeans = cv2.threshold(jeans, 20, 255,
                             cv2.THRESH_BINARY)
    edges = cv2.Canny(jeans, 50, 100)
    contours, _ = cv2.findContours(edges,
                  cv2.RETR_LIST,
                  cv2.CHAIN_APPROX_NONE)
    coordinates = []
    for idx, contour in enumerate(contours):
        length = cv2.arcLength(contour,
                               closed=True)
        if length < 260: continue

        x,y,w,h = cv2.boundingRect(contour)
        coordinates.append((x+(w//2),
        y+(h//2)))

    return coordinates
```

Quanto ao desenvolvimento, temos que a função desenvolvida utiliza unicamente a biblioteca OpenCV, além de módulos auxiliares da própria linguagem Python para concretizar o método descrito. No código, foram utilizadas as seguintes funções:

```
def detect_threshold(img):
    output = deepcopy(img)
    jeans = cv2.blur(output, (13,13))
    _, jeans = cv2.threshold(jeans, 20, 255, cv2.THRESH_BINARY)
    edges = cv2.Canny(jeans, 50, 100)
    contours, _ = cv2.findContours(edges, cv2.RETR_LIST, cv2.CHAIN_APPROX_NONE)
    coordinates = []
    for idx, contour in enumerate(contours):
        length = cv2.arcLength(contour, closed=True)
        if length < 260: continue

        x,y,w,h = cv2.boundingRect(contour)
        coordinates.append((x+(w//2),y+(h//2)))

    return coordinates
```

- deepcopy: Realiza a cópia da imagem
- blur: Realiza a homogeneização da imagem com um filtro de média 13 x 13.
- threshold: Aplica o método da limiarização na imagem. Obs: Foram utilizados o valor 20 como limite para a limiarização, além de o valor máximo ser 255, como estamos lidando com imagens 8 bits.
- Canny: Utiliza a técnica da detecção de bordas de Canny com os limites de 50 e 100.
- findContours: Como o nome sugere, encontra os contornos na imagem.
- arcLength: Retorna o comprimento do contorno detectado.
- Obs: foi escolhido o limite superior de 260 unidades de comprimento por meio de testes para a dimensão do contorno e evitar a identificação errônea de falhas.
- boundingRect: Gera um retângulo de área mínima que abrange a falha detectada

IV. SIMULAÇÃO

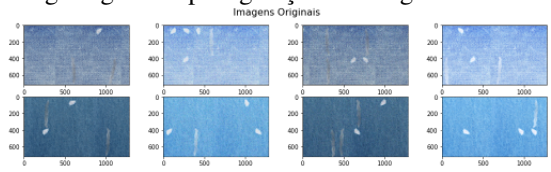
Os parâmetros essenciais para gerar um vídeo são: a resolução do vídeo, a altura, a largura, o campo de visão, a distância da câmera, e a quantidade de imagens por coluna e por linha, além de também ser necessário a escolha das imagens, das quais são redimensionadas de forma aleatória com base em um vetor que possua os possíveis tamanhos que ela possa ter. Além disso, também deve existir uma rotação da imagem que, da mesma forma que a escolha das imagens, é decidida de forma aleatória, utilizando um vetor de probabilidades que determina a probabilidade de cada aspecto aparecer em cada posição da tela. Para deixar mais próximo da realidade, será utilizado um ruído gaussiano no vídeo, porém, será possível mudar os tipos de ruídos de acordo com a necessidade, como "salt" e "pepper". O processo de leitura é realizado por um frame de cada vez, sendo adicionado a luminosidade que foi escolhida na interface e, em seguida, enviando a imagem para a fila de pré-processamento, para que então em sequência, seja direcionado para o processamento em si, onde são retornadas as coordenadas das falhas para ser possível que a aquisição marque na imagem original. Todas essas comunicações entre etapas são feitas por meio de threads, para que sejam realizadas de forma simultânea, já que há a necessidade de que as marcações das falhas sejam feitas em tempo real. Essas marcações são feitas com o uso das coordenadas, do frame e de uma função para circular a área desejada.

Os objetivos principais consistiram no desenvolvimento de uma interface na qual o usuário fosse capaz de interagir, gerar

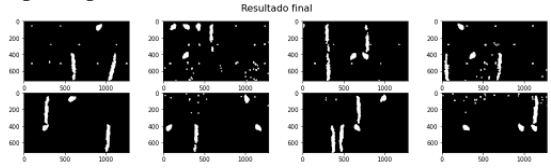
os vídeos para a realização da análise e marcar as falhas. Os desafios encontraram-se na parte de comunicação entre os processos (ler os vídeos, criar as filas etc.). Os resultados saíram conforme o esperado. O vídeo foi gerado e as falhas foram marcadas com uma boa precisão.

O processo apresentou resultados satisfatórios, em termos de precisão. Abaixo estão três exemplos com três imagens cada um, representando a imagem original, a etapa de pré-processamento e a etapa de desenvolvimento, respectivamente. Obtivemos as seguintes imagens:

Imagens geradas pela geração de imagens:

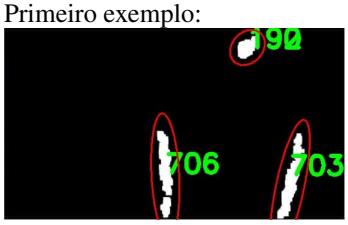


Após o processamento:



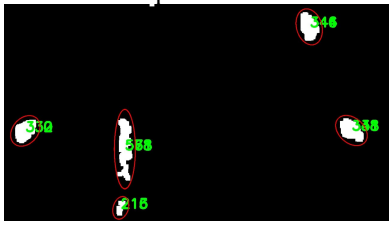
Em seguida, após a aplicação da detecção de falhas desenvolvidas, temos os seguintes resultados:

Primeiro exemplo:



Segundo exemplo:

Terceiro exemplo:



V. ANÁLISE ESTÁTICA

Sabendo que existe a probabilidade de que o código final tenha erros tanto nos requisitos, quanto de design ou funcionalidade, os testes de software foram realizados para identificar possíveis problemas de funcionamento, configuração ou usabilidade do sistema. No sistema foi utilizado o teste funcional, ou seja, um tipo de teste caixa preta no qual se baseia em testar a(s) funcionalidade(s) do sistema. Nesse primeiro momento, o código em si não foi testado, pois o teste funcional teve o objetivo de medir a acurácia e detectar as falhas no tecido que, inicialmente, o projeto não conseguiu identificar. Mas, posteriormente, foi realizada uma análise estática do código-fonte para testar a sua qualidade. Tal análise foi realizada mediante a ferramenta de análise de código chamada Codacity, cuja análise usa seis categorias de análise:

- Segurança(Security):problemas de seguranca, vulnerabilidades potenciais, dependencias inseguras.
- Estilo de Código(Code Style): relacionado ao estilo do codigo, comprimento da linha, tabulacoes e espaco.
- Compatibilidade(Compatibility): adaptação do código a sistemas mais antigos e suporte a plataformas cruzadas.
- Propensão a erros(Error Prone): praticas ou padroes inadequados que fazem com que o codigo falhe ou potencialize a ocorrência de bugs.
- Codigo nao utilizado(Unused code): codigo desnecessario nao em uso
- Desempenho(Performance): codigo escrito de forma ineficiente

A análise do Codacity classificou o código como level C encontrando 113 atividades, ou problemas, que precisam ser melhorados para que a qualidade atinja um nível A(que seria o ótimo). Desses, 97 erros eram de code style, 14 de segurança e 2 de propensão a erros(onde estes últimos são considerados como erros muito graves). Como vemos na imagem abaixo.

VI. CONCLUSÃO

Dessa forma, temos como resultado uma aplicação que efetivamente consegue detectar defeitos específicos na fabricação de tecido Jeans, passando nos critérios de qualidade estabelecidos e testados de forma procedural, gerando assim um resultado solido com pouca margem de erros, esses relacionados à quantidade de iluminação. Como possíveis próximos passos teríamos a utilização de tecidos mais diversos e aplicação em tempo real com aplicação pratica do sistema.

Temos então como entregar, a apresentação de um sistema para detecção de falhas em tecidos Jeans, um documento

descritivo do sistema e documentações parciais referentes a cada frente.